# jQuery

可不必关心兼容性问题

内容：掌握jQuery编程思想，使用 jQuery进行常见网页效果开发。

目标：能够使用 jQuery开发常见网页效果。

DOM的顶级对象是document，BOMd的顶级对象是window，jQuery的顶级对象

1. jQuery文件的使用
2. jQuery中顶级对象
3. jQuery中事件加载
4. jQuery中的选择器

一、常见的JavaScript框架库

javaScript框架库：就是一个普通的js文件，封装了很多的函数，封装了很多兼容的代码。可以在不同浏览器中使用。

jQuery是javaScript框架库中的一种。

jQuery的好处：写的少，做的多，链式编程，隐式迭代......

可以解决js兼容的问题。

1、什么是JavaScript框架库？

普通JavaScript的缺点：每种控件的操作方式不统一，不同浏览器下有区别封装库，要编写跨浏览器的程序非常麻烦。因此出现了很多对JavaScript的。

2、常见的JavaScript框架库

Prototype、 YUI(网络反响一般)、 Dojo、 ExtJS、 jQuery等，这些库对JavaScript进行了封装，简化了开发。这些库是对JavaScript的封装，内部都是用JavaScript实现的。

3、jQuery:

jQuery就是JavaScript语法写的一些函数类，内部仍然是调用 JavaScript实现的，所以并不是代替JavaScript的。使用 jQuery的代码、编写jQuery的扩展插件等仍然需要JavaScript的技术， jQuery本身就是一堆JavaScript函数。

4、回调函数

回调就是一个函数的调用过程。那么就从理解这个调用过程开始吧。函数a有一个参数，这个参数是个函数b，当函数a执行完以后执行函数b。那么这个过程就叫回调。

其实中文也很好理解：回调，回调，就是回头调用的意思。函数a的事先干完，回头再调用函数b。

二、jQuery简介

1、什么是jQuery？

jQuery就是一个JavaScript函数库，没什么特别的。

2、jQuery能做什么？ jQuery是做什么的？

jQuery本身就是一堆JavaScript函数，**JavaScript是做什么的， jQuery也一样**。毕竟jQuery只是用JavaScript编写的函数库而已，有些功能jQuery没有封装，则还需要通过自己写原生JavaScript来实现。

3、为什么要学习 JQuery

Js中一个简单的功能需要大量的代码。

Js中兼容的问题很多。

Js中代码的容错性很差。

Js中window.onload也只能有一个。

4、好处

jQuery的特点？ Write Less,Do More

很好的解决了不同浏览器的兼容问题（ IE 6.0+, FF 2+,Safari 3.0+, Opera 9.0+, Chrome） css还是有问题的对于不同控件具有统一的操作方式。

体积小（几十KB）、使用简单方便（ Write Less Do More）

链式编程$("#div1"). draggble().show().hide().fly() 、隐式迭代、插件（别人写好的代码我们直接拿来用）丰富、开源、免费。插件多缺什么找什么。

学习js经常需要自己封装兼容的代码

学习jQuery不必考虑过多的兼容问题，但是css还是有需考虑，因为不同的浏览器对于css的支持还是不太一样的。

（1）初步体验写法

|  |
| --- |
| <!-- <script type="text/javascript">  var btn = document.getElementById('btn');  btn.onclick = function () {  var box = document.getElementById('box');  box.style.width = '200px';  box.style.height = '200px';  box.style.backgroundColor = 'pink';  };  </script> --> |
| <script type="text/javascript">  // 页面加载后的一个事件，第一个function的意思是等页面都加载完毕  $(function () {  $('#btn').click(function () {  $('#box').css({ 'width': '200px', 'height': '200px', 'backgroundColor': 'deeppink'});  });  });  </script> |

三、jQuery中的顶级对象$

jQuery中最常用的对象即$对象，要想使用jQuery的方法必须通过$对象。只有将普通的Dom对象封装成jQuery对象，然后才能调用jQuery中的各种方法。

$是jQuery简写，在代码中可以使用jQuery代替$，但一般为了方便大家都直接使用$。

写注释，后续jQuery代码会越写越多，所以必要的注释一定要写。

DOM中的顶级对象：document是页面中的顶级对象

document点出来的是DOM中的属性和方法。

BOM中的顶级对象：window是浏览器中的顶级对象

window点出来的是浏览器中的属性和方法。window.document页面中的一切内容都是属于window的。

jQuery的顶级对象：jQuery与$虽然它们的写法不同，但是表示的意义是一样的。$符号点出来的是jQuery中的方法。

5、编写简单的jQuery代码-页面的加载事件

（1）页面加载事件window.onload

只能写一次,重复会被后面重新进行赋值，要等到图片，外部文件引入加载后才触发。

$(document)用美元符号包括起来之后，他就不再是dom顶级对象，变成了jQuery对象。

|  |
| --- |
| // 原生js  /\*window.onload = function () {  console.log('原生js页面加载完毕');  }\*/  // jQuery页面中所有的内容加载完毕之后才触发，涉及标签、图片、文字内容等等,能存在多个  /\*$(window).load(function () {  console.log('jQuery window方式-页面加载完毕No1');  });  $(window).load(function () {  console.log('jQuery window方式-页面加载完毕No2');  });\*/ |

1. 页面中的基本的标签加载完毕之后就会触发，DOM元素加载完就会触发

|  |
| --- |
| // 第一种方式  $(document).ready(function () {  console.log('jQuery document方式页面加载完毕1');  });  $(document).ready(function () {  console.log('jQuery document方式页面加载完毕2');  }); |
| // 第二种方式  jQuery(function () {  console.log('jQuery 方式页面加载完毕1');  });  jQuery(function () {  console.log('jQuery 方式页面加载完毕2');  }); |
| // 第三种方式  $(function () {  console.log('$方式页面加载完毕1');  });  $(function () {  console.log('$方式页面加载完毕2');  }); |

注：（1）会比（2）慢执行，因为它不仅需要等DOM标签加载，同时还要等待外部图片等资源的加载。

（3）jQuery中引入文件注意问题

<script src="jquery-1.12.2.js"></script>

先引入文件,然后再使用开发的时候引入正常的jquery文件和压缩版的文件都没有问题。在引入文件的script标签中，不要再写其他的代码。需要重新写script标签。

建议:开发用普通版。

上线用压缩版。为了下载速度更快，给用户更好体验。

三、Jquery对象和DOM对象要

1、为什么需要互转。

DOM对象转Jquery对象操作方便互转，毕竟Jquery中方法都是封装好了的,而且兼容问题解决的很好，代码少，方便.

Jquery对象转DOM对象，因为Jquery中文件一直在更新，很多东西都是随着使用而进行封装和升级，不太可能把所有dom中用到的进行封装，还有很多未知的兼容问题没有封装进去，所以，有的时候jquery中不能解决的问题，还需要原生的js代码来解决。

所以，jquery对象有的时候需要转成dom对象来进行操作。

1. 如何互转。
2. DOM对象转jQuery对象，只需要把DOM对象放在**$(dom对象)**，此时就会转为jQuery对象。

|  |
| --- |
| // dom方式  **var btn** = document.getElementById('btn');  /\*btn.onclick = function () {  this.style.backgroundColor = 'red';  console.log('hello');  }\*/  // DOM对象转为jQuery  **$(btn)**.click(function () {  // $(this).css({ backgroundColor: 'red'});  $(this).css('backgroundColor','red');  }); |

（2）jQuery对象转DOM对象有两种方式：

|  |
| --- |
| 1、$(jQuery对象).get(0)  2、$(jQuery对象)[0]  这样就能转为DOM对象  注：jquery对象只能调用jQuery的方法，不能调用dom的方法，调用的话，**不起作用，但是不报错。**  **$(“#btn”)获取的是当前按钮的jQuery对象。**  目前一般还没有互换的操作。 |

|  |
| --- |
| <input type="button" value="点击改变自身颜色" id="btn">  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  // dom方式  var **btn** = document.getElementById('btn');  // DOM对象转为jQuery对象，再转为DOM对象  /\*$(btn)[0].onclick = function () {  this.style.backgroundColor = 'green';  };\*/  $(btn).get(0).onclick = function () {  this.style.backgroundColor = 'green';  };  </script> |

|  |
| --- |
| // jQuery方式：jQuery对象转为DOM对象  **$('#btn')[0]**.onclick = function () {  this.style.backgroundColor = 'pink';  console.log(typeof $('#btn')); // 输出是对象  }  // **$('#btn').get(0)**.onclick = function () |

|  |
| --- |
| // jQuery对象调用dom对象的方法  $('#btn').onclick = function () {  this.style.backgroundColor = 'pink';  console.log(typeof $('#btn')); // 输出是对象  }  // dom对象调用jQuery对象的方法  var btn = document.getElementById('btn');  **btn.click**(function () {  $(this).css('backgroundColor','yellow');  console.log('hello ');  });  这样子写不报错，也没有效果。  有的会报错，如dom对象调用html()方法就报错。  1 |

.val()方法，获取input的value属性的值。

.val(‘内容’)，是设置input的value属性的值。

3、网页开关小案例

|  |
| --- |
| // dom方式  /\*var btn = document.getElementById('btn');  btn.onclick = function () {  var body = btn.parentNode;  if (this.value === '关灯') {  body.style.backgroundColor = 'black';  this.value = '开灯';  } else {  body.style.backgroundColor = '';  this.value = '关灯';  }  };\*/ |
| // jQuery方式  // 等页面标签加载完毕  $(function () {  $('#btn').click(function () {  if ($(this).val() === '关灯') {  $('body').css({'backgroundColor': 'black'});  $(this).val('开灯');  } else {  $(this).val('关灯');  $('body').css({'backgroundColor': ''});  }  });  }); |

四、jQuery选择器(重点)

选择器：就是为了获取元素的，CSS中设置元素的样式可以通过选择器(c1-c3都用 )，jQuery中通过选择器来获取元素进行操作

DOM中获取元素的方式:

document.getElementById();根据id获取元素，一个

document.getElementByTagName();根据标签的名字获取元素，多个

document.getElementByClassName();根据类名来获取元素，多个，有兼容性问题，IE9才支持，跟选择器（.father是类选择器，类名是father）要进行区分。

jQuery获取元素的方式：通过各种选择器来获取元素。

1. 根据id来获取，id选择器------$(‘**#**id的值’)，获取一个
2. 根据标签的名字来获取，标签选择器------$(‘标签的名字’)，多个
3. 根据类样式的名字获取，类选择器------$(‘**.**类选择器名字’)，多个，不要写成类名了

1、小案例

（1）id选择器：点击按钮，设置div中的显示内容为:xxx，同时设置这个层的背景颜色，

.text()相当于是DOM中的innerText或者是textContent。

|  |
| --- |
| <input type="button" value="点击更改div" id="btn">  <div id="box"></div>    <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  // 等待页面中所有DOM标签完毕  $(function () {  $('#btn').click(function () {  $('#box').text('hello i am Div jQuery');  $('#box').css({'backgroundColor': 'lightgreen'});  });  });  </script>  #box {  width: 200px;  height: 200px;  background-color: pink;  } |

（2）标签选择器：点击按钮，设置多个p标签中显示内容为:我们都是

|  |
| --- |
| <input type="button" value="点击更改div" id="btn">  <p></p>  <p></p>  <p></p>    <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  // 等待页面中所有DOM标签完毕  $(function () {  $('#btn').click(function () {  $('p').text('hello i am P 标签');  **$('p').css**({'backgroundColor': 'lightblue'});  });  });  // 使用DOM方式的话。需要写循环  </script> |

（3）类选择器：点击按钮,设置页面上应用 cls类样式的元素的背景颜色

|  |
| --- |
| <style type="text/css">  .cls {  background-color: deeppink;  }  </style>  <input type="button" value="点击更改所有类cls的颜色" id="btn">  <p class="cls"> I am p </p>  <span class="cls">I am span </span>    <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  // 等待页面中所有DOM标签完毕  $(function () {  $('#btn').click(function () {  **$('.cls').css**({'backgroundColor': 'yellow'});  });  });  </script> |

2、标签+类选择器（交集选择器）

标签加类选择器：$(li.cls)

点击按钮设置页面上应用 cls类样式li标签的背景颜色。

|  |
| --- |
| <input type="button" value="点击更改所有类cls的颜色" id="btn">  <ul>  <li class="cls">西瓜</li>  <li class="cls">葡萄</li>  <li>芒果</li>  </ul>    <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  // 等待页面中所有DOM标签完毕  $(function () {  $('#btn').click(function () {  $('li.cls').css({'backgroundColor': 'blue'});  });  });  </script>  .cls {  background-color: deeppink;  } |

3、多条件选择器（并集选择器）

$(span,li,div)

点击按钮,设置页面中的span标签，li标签，div标签的背景颜色

|  |
| --- |
| <input type="button" value="点击多条件选择器的颜色" id="btn">  <ul>  <li class="cls">西瓜</li>  <li class="cls">葡萄</li>  <li>芒果</li>  </ul>  <div>我是div</div>  <footer>我是footer</footer>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  // 等待页面中所有DOM标签完毕  $('#btn').click(function () {  **$('li.cls,div,footer').css(**{'backgroundColor': 'purple'});  });  </script> |

4、层级选择器

（1）后代选择器(子元素中元素,子,仔仔,孙)。

$(“#dv li”)或者$(“ul li”)或者$(“.cls li”)

（2）子代选择器(直接的所有子元素,儿子)。

$(“#ul>li”)或者$(“div >span”)或者$(“.cls >li”)

|  |
| --- |
| <input type="button" value="点击多条件选择器的颜色" id="btn">  <div>我是第一个div  <p>我是div里的第一个p标签</p>  <ul>  <li>  <p>我是div ul li 中第一个p标签</p>  </li>  </ul>  <p>我是div里的第二个p标签</p>  </div>  <p>我是div下面的p标签</p>  <p>我也是div下面的p标签</p>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  // 等待页面中所有DOM标签完毕  $(function () {  $('#btn').click(function () {  **$('div>p')**.css({'backgroundColor': 'skyblue'});  });  }); |

（3）相邻选择器，当前元素的相邻元素：这个元素后紧跟着的**第一个**元素。

$(“div + span”)

（4）兄弟选择器，获取当前元素后面所有元素：这个元素后跟着的**所有元素**。

$(“div ~ span”)

（5）小案例--鼠标经过水果高亮显示，离开取消高亮mouseover

|  |
| --- |
| <ul id="fruit">  <li>西瓜</li>  <li>橙子</li>  <li>葡萄</li>  <li>芒果</li>  <li>杨梅</li>  </ul>  $(function () {  $('#fruit>li').**mouseover**(function () {  $(this).css('backgroundColor','gray');  });  $('#fruit>li').**mouseout**(function () {  $(this).css('backgroundColor','');  });  });  如果使用DOM的话，还得写循环。 |
| 点击按钮修改层中所有的标签的背景颜色。  <input type="button" id="btn" value="修改所有的颜色">  <ul id="fruit">  <li>西瓜</li>  <li>橙子</li>  <li>葡萄</li>  <li>芒果</li>  <li>杨梅</li>  </ul>  $(function () {  $('#btn').click(function () {  $('#fruit>**\***').css('backgroundColor','yellow');  });  }); |

（6）\*难点\*小案例--下拉菜单--show、hide，先多看视频，再思考

|  |
| --- |
| 对jQuery中的show、stop、hide它们的使用还不清楚，浪费了比较多的时间，看后续是否有讲解，在理解，暂时不耗费时间，**待解决？？？已解决**   1. 移动到下边选项时候出现消失、出现的问题原因，因为显示出来的子元素的li触发事件冒泡，会触发多次事件处理函数，产生了动画的积累。 2. mouseover与mouseout。建议使用mouseenter、mouseleave能够有效的减少。会正常很多。但是还是会存在积累的问题。没有加stop情况。   当你向下经过li时，比动画快一些放在芒果跟西瓜的中间区域时，会不断的有动画的积累。  具体分析：进入“水果3”时，show进入动画队列，离开“水果3”时，鼠标的位置在芒果和西瓜之间（此时的show已经在执行，但是没有到达鼠标位置），离开同时hide进行动画队列。  当show显示区域经过鼠标的时候，又有一个show进入动画队列（此时它的前面还有hide）。当show执行完之后，马上执行hide。执行完hide后，又执行刚才新加的show。等等一直这样循环，一上一下。    当菜单下拉动画未完成时，鼠标又移出了菜单，就会产生“动画积累"，当鼠标停止移动后，积累的动画还会持续执行，直到动画序列执行完毕。  （3）stop为什么加在前边  鼠标移动出去的时候，它马上把show的动画给停止了，就开始hide动画了。这样就不会产生叠加的条件了。  stop如果加在show的后边，show刚要执行就被停止了，看不到现象，所以只能加在它是前边。  鼠标进来的时候，先停止之前的动画，再去显示或者隐藏。 |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>jQuery学习</title>  <style type="text/css">  \* {  padding: 0;  margin: 0;  }  ul {  list-style: none;  }  a {  text-decoration: none;  color: black;  }  .wrap a {  /\*width: 100px;\*/  display: block;  }  .wrap a:hover {  color: #fff;  background-color: blue;  }  .wrap {  width: 300px;  margin: 100px auto;  }  .wrap>ul>li {  width: 100px;  float: left;  background-color: red;  text-align: center;  }  .wrap>ul>li>ul {  display: none;  }  </style>  </head>  <body>  <div class="wrap">  <ul>  <li><a href="javascript:void(0);">水果1</a>  <ul>  <li><a href="javascript:void(0);">西瓜</a></li>  <li><a href="javascript:void(0);">西瓜</a></li>  <li><a href="javascript:void(0);">西瓜</a></li>  </ul>  </li>  <li><a href="javascript:void(0);">水果2</a>  <ul>  <li><a href="javascript:void(0);">葡萄</a></li>  <li><a href="javascript:void(0);">葡萄</a></li>  <li><a href="javascript:void(0);">葡萄</a></li>  </ul>  </li>  <li><a href="javascript:void(0);">水果3</a>  <ul>  <li><a href="javascript:void(0);">芒果</a></li>  <li><a href="javascript:void(0);">芒果</a></li>  <li><a href="javascript:void(0);">西瓜</a></li>  </ul>  </li>  </ul>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('.wrap>ul>li').mouseover(function () {  **$(this).children('ul').stop().show(500);**  });  $('.wrap>ul>li').mouseout(function　() {  **$(this).children('ul').stop().hide(500);**  });  });  </script>  </body>  </html> |

（7）:even（偶数选择器）与:odd（奇数选择器）

这里的偶数，奇数可能跟我们现象看到奇、偶不一样，因为代码数组中，它的下标是从0 开始的。

|  |
| --- |
| <style type="text/css">  \* {  padding: 0;  margin: 0;  }  ul {  list-style: none;  }  .wrap {  width: 200px;  margin-left: 100px;  }  .wrap ul li {  width: 100px;  margin-top: 10px;  border: 1px solid #000;  text-align: center;  }  </style>  <input type="button" id="btn" value="添加颜色,代码even对应red，代码odd对应pink">  <div class="wrap">  <ul id="num">  <li>1</li>  <li>2</li>  <li>3</li>  <li>4</li>  <li>5</li>  <li>6</li>  <li>7</li>  <li>8</li>  </ul>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  **$('#num>li:even')**.css('backgroundColor', 'red');  **$('#num>li:odd')**.css('backgroundColor', 'pink');  });  });  </script>  1 |

5、索引选择器

（1）:eq(3)获取索引为3的元素。

（2）:gt(3)索引大于3的所有元素。

（3）:lt(3)索引小于3的所有的元素。（小写的L）

|  |
| --- |
| 如果冲突的话，后边的会覆盖前边的，层叠性。  <input type="button" id="btn" value="添加颜色,代码even对应red，代码odd对应pink">  <div class="wrap">  <ul id="num">  <li>索引为0-----我们说的1</li>  <li>索引为1-----我们说的2</li>  <li>索引为2-----我们说的3</li>  <li>索引为3-----我们说的4</li>  <li>索引为4-----我们说的5</li>  <li>索引为5-----我们说的6</li>  <li>索引为6-----我们说的7</li>  <li>索引为7-----我们说的8</li>  </ul>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  **$('#num>li:eq(5)').css('backgroundColor', 'red');**  **$('#num>li:lt(3)').css('backgroundColor', 'pink');**  **$('#num>li:gt(6)').css('backgroundColor', 'yellow');**  });  });  </script>  1 |
| $('#num>li:eq(5)').css('backgroundColor', 'red');  $('#num>li:gt(3)').css('backgroundColor', 'pink');  $('#num>li:lt(6)').css('backgroundColor', 'yellow');  1 |

（4）.index()方法--图片展示案例

|  |
| --- |
| 1、如果没有指定参数，则返回当前元素在其所有同辈元素中的索引位置。  2、如果为String类型，则将其视作选择器。  返回当前元素在**选择器所匹配的元素(ul li)**中的索引位置。  如果该选择器不匹配任何元素或者当前元素不在匹配到的元素内，则返回-1。  3、如果参数是一组DOM元素或者jQuery对象，那么返回值就是传递的元素相对于原先集合的位置。  第三点有点反过来如：  console.log($('.wrapper li').index($('li.temp')));  在.wrapper li匹配的集合中，查找类名是temp的li元素  注：在事件处理函数中，用方式1、2合适，3就不适合 |

获取当前元素的索引。

|  |
| --- |
| <script type="text/javascript">  $(function () {  $('#left>li').click(function () {  var index = $(this).index();  $('#center>li'+':eq('**+index+**')').**siblings('li')**.hide();  $('#center>li'+':eq('+index+')').show();  });  $('#right>li').click(function () {  var index = $(this).index() + 9;  $('#center>li'+':eq('+index+')')**.siblings('li')**.hide();  $('#center>li'+':eq('+index+')').show();  });  })  </script>  注：index的值是数字，要把它进行拼接才行，变成字符串，因为jQuery参数都是字符串。  在自己布局的时候，把height没有写，只是写了line-heigth导致出了差距，小差距乘以9就会导致，差距变得很大。 |

**注意点，height与line-height的效果是不一样的，line-height的高度的小于height值。导致出现的空隙比较大，这是一个问题。height确定的情况下，一定要出来。**

（5）.siblings(‘元素’)获取当前元素的所有兄弟元素。

（6）.find(‘元素’)，针对当前的元素查找里面的一些其他的元素，参数就是要查找的元素。

（6）高亮小案例

|  |
| --- |
| $(function () {  $('.wrapper>ul>li').mouseover(function () {  $(this).siblings('li').css('opacity','0.5');  $(this).css('opacity','1');  });  //这里会发生冒泡事件，看后续能怎么处理？？？  $('.wrapper').mouseout(function (e) {  $(this).find('li').css('opacity','1');  });  }); |

（7）.children()

可以通过可选的表达式来过滤所匹配的子元素。注意：parents()将查找所有祖辈元素，而children()只考虑子元素而不考虑所有后代元素。

它的参数是选择器，可以是并集选择器（div,span）。

（7）好友面板案例

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>jQuery学习</title>  <style type="text/css">  \* {  padding: 0;  margin: 0;  }  ul {  list-style: none;  }  .wrap {  width: 200px;  height: 500px;  border: 1px solid red;  }  .wrap>ul>li {  font-size: 20px;  text-align: center;  background-color: orange;  margin-bottom: 10px;  }  .wrap li ul {  background-color: purple;  display: none;  }  </style>  </head>  <body>  <div class="wrap">  <ul>  <li>小学  <ul>  <li>小学</li>  <li>小学</li>  <li>小学</li>  </ul>  </li>  <li>初中  <ul>  <li>初中</li>  <li>初中</li>  <li>初中</li>  </ul>  <span></span>  <div></div>  <div></div>  </li>  <li>高中  <ul>  <li>高中</li>  <li>高中</li>  <li>高中</li>  </ul>  </li>  </ul>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('.wrap>ul>li').click(function () {  $(this).siblings('li').children('ul').hide(500);  $(this).children('ul').show(500);  console.log($(this).children('ul,span,div'));  });  });  </script>  </body>  </html> |

（8）.parent()与.parents()是不一样的

1. 手风琴小案例

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>手风琴</title>  <style type="text/css">  \* {  padding: 0;  margin: 0;  }  ul {  list-style: none;  }  .wrapper {  width: 1150px;  height: 400px;  border: 1px solid red;  overflow: hidden;  }  .wrapper ul {  **width: 1300px;**  }  .wrapper ul li {  width: 230px;  float: left;  overflow: hidden;  }  </style>  </head>  <body>  <div class="wrapper">  <ul>  <li><img src="images/1.jpg"></li>  <li><img src="images/2.jpg"></li>  <li><img src="images/3.jpg"></li>  <li><img src="images/4.jpg"></li>  <li><img src="images/5.jpg"></li>  </ul>  </div>  <script type="text/javascript" src="jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('.wrapper>ul>li').mouseover(function () {  $(this).siblings('li').css('width','100px');  $(this).css('width', '800px');  });  $('.wrapper>ul>li').mouseout(function () {  $(this).parent().children('li').css('width', '230px');  });  });  </script>  </body>  </html> |

五、jQuery几个常见的方法

1. .html()方法，设置标签中间显示其他标签及内容,类似于innerHTML。

|  |
| --- |
| .html();小括号中可以直接写标签的字符串内容，就是设置div中的元素内容。  .html();小括号中什么也没有，表示的时候获取div中的元素内容。  .html()方法相当于DOM中的innerHTML。  点击按钮的时候向div中添加一个p标签。  <input type="button" value="innerHTML添加元素" id="btn">  <div id="box"></div>  $(function () {  $('#btn').click(function () {  **$('#box').html('<p><strong>hello w</strong></p>');**  });  }); |

1. .text()方法，设置标签中间显示的文本内容,类似于innerText。

|  |
| --- |
| .text();如果小括号中写内容，就是设置文字内容。  .text();如果小括号什么也不写，表示的是获取这个元素中的文字内容。  .text()方法相当于DOM中的innerText。 |

1. .val()方法，设置input标签中value的值,类似于value。

|  |
| --- |
| 点击设置文本框中的值  .val()方法，小括号中写内容就是设置元素value属性。  .val()方法，小括号中什么也不写，获取元素的value属性的值  这个就是元素的value属性  <input type="button" value="设置并获取文本框中的值" id="btn">  <input type="text" id="txt">  <div id="box"></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  // 等待页面中所有DOM标签完毕  $(function () {  $('#btn').click(function () {  **$('#txt').val**('文本值hello world');  $('#box').text($('#txt').val());  });  });  </script> |

1. .css()方法，设置元素的样式,类似于style。

|  |
| --- |
| .css();方法，该方法如果只是设置一个样式的属性和值，那么这个方法写两个参数，第一个参数是属性，第二个参数是值。  .css();方法，里面也可以直接写键值对的方式  <input type="button" value="设置宽高、颜色" id="btn">  <div id="box"></div>  $(function () {  $('#btn').click(function () {  /\*$('#box').css('backgroundColor','red');  $('#box').css('width','200px');  $('#box').css('height','200px');\*/  $('#box').css({'backgroundColor':'blue','width': '200px','height':'200px'});  });  }); |

注意:

**()中写字符串** ,一个参数,就是要设置的值,什么也不写,返回的是这个属性的值。

1. jQuery操作样式

1、.css()方法

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>jQuery学习</title>  <style type="text/css">  \* {  padding: 0;  margin: 0;  }  .wrap {  margin: 100px;  }  </style>  </head>  <body>  <input type="button" id="btn" value="显示效果">  <div class="wrap">  <ul>  <li>西瓜</li>  <li>葡萄</li>  <li>芒果</li>  <li>杨梅</li>  </ul>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $('#btn').click(function () {  // 第一种写法  /\*$('.wrap>ul>li').css('backgroundColor', 'yellow');  $('.wrap>ul>li').css('color', 'blue');\*/  // 第二种写法  // 对象的属性是字符串形式，所有加引号或者不加都ok  /\*var style = {  backgroundColor: 'red',  color: 'blue',  width: '200px',  fontFamily: '宋体'  }  $('.wrap li').css(style);\*/  // 是等价于 $('.wrap li').css({'backgroundColor': 'red','color':'blue','width':'200px' });  // 第三种写法  $('.wrap>ul>li').css('backgroundColor', 'yellow').css('width','200px');  });  </script>  </body>  </html>  1 |

2、获取css属性值

获取属性的值，写一个参数即可

|  |
| --- |
| console.log($('.wrap>ul>li').css(**'backgroundColor'**));  1 |

七、jQuery操作类样式

1、addClass()方法，添加类样式

在addClass方法中类样式的名字前面是没有一点（.）

addClass添加多个样式的时候，多个类样式的名字中间用空格隔开。

addClass();参数:类样式名字，添加样式的同时不会覆盖原有的样式。

2、removeClass()方法，移除类样式

removeClass();不写参数移除所有的类样式。

removeClass(“cls“);移除指定的一个类样式。

|  |
| --- |
| div {  margin: 100px;  width: 100px;  height: 100px;  border: 1px solid #000;  }  .big {  width: 150px;  height: 150px;  background-color: pink;  }  .addBorder {  border: 10px solid red;  box-shadow: 1px 2px 3px rgba(0,0,0,.5);  } |
| <input type="button" id="btn" value="添加类样式">  <input type="button" id="btn1" value="去除类样式">  <div></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  // $('div').addClass('big addBorder');  $('div').addClass('big').addClass('addBorder');  });  $('#btn1').click(function () {  $('div').removeClass();  });  });  </script> |

3、hasClass(‘类名’)

判断该元素是否有这个类样式

4、toggleClass()切换样式，有就移除，没有就添加

|  |
| --- |
| .light {  background-color: black;  } |
| <input type="button" id="btn" value="开关灯">  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  /\* if($('body').hasClass('light')) {  $('body').removeClass('light');  $(this).val('关灯');  } else {  $('body').addClass('light');  $(this).val('开灯');  }\*/  $('body').toggleClass('light');  });  });  </script> |

5、css()方法和操作类样式的方法比较

设置的样式如果很多那么直接使用类样式的方式进行操作，为了方便维护也最好使用类样式。

进行操作简单的操作直接使用css的方式。

|  |
| --- |
| css(‘属性’,’属性值’).css(‘属性’,’属性值’)  css({‘属性’：‘属性值’，‘属性’：‘属性值’}) |
| addClass(类样式名字)，添加一个类样式  addClass(类样式名字1 空格 类样式名字2) |
| removeClass()，移除的是当前元素中所有的类样式  removeClass(类样式名字)，移除类样式 |
| hasClass()，判断当前元素是否运用了某个类样式 |
| toggleClass()，切换元素的类样式 |

八、链式编程

多行代码合并成一行代码，前提要认清此行代码返回的**是不是当前同个对象。是同个对象**才能进行链式编程。（最后的一个方法可以不用返回对象也行，如html()）。

1、链式编程：对象.方法().方法().方法()....

|  |
| --- |
| .wrapper {  margin: 50px;  width: 100px;  height: 100px;  border: 1px solid #000;  } |
| <input type="button" id="btn" value="链式编程">  <div class="wrapper">  <p>p标签</p>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  /\*var obj = $('.wrapper').html('<p>动态添加p标签</p>');  var obj1 = $('.wrapper').css('backgroundColor', 'pink');\*/  $('.wrapper').html('<p>动态添加p标签</p>').css('backgroundColor', 'pink');  调换html跟css的顺序是没有问题  });  });  </script>  1 |

|  |
| --- |
| $('.wrapper')**.html()**.css('backgroundColor', 'pink');  不是返回一个对象就无法进行链式编程，报错；因为获取值时返回的是获取的字符串而不是对象本身所以不能链式编程。  这个称之为“断链”。  2  2 |
| var temp = $('.wrapper').css('backgroundColor', 'pink').html();  console.log(temp);  html返回的是字符串，它是在最后才调用的，所有没有问题。打印的是获取到了字符串 |

2、链式编程案例

|  |
| --- |
| 实现的效果：  获取列表中每个li，当时鼠标经过时候，当前的li有高亮显示  点击的时候可以让当前点击的li字体变大，字体的颜色改变，改变字体 |
| <div class="container">  <ul id="fruit">  <li>西瓜</li>  <li>葡萄</li>  <li>橙子</li>  <li>杨梅</li>  <li>芒果</li>  </ul>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#fruit>li').mouseover(function () {  $(this)**.css**('backgroundColor', 'orange')**.siblings**('li')**.css**('backgroundColor', '');  })**.click**(function () {  $(this).css({ 'fontSize':'20px','color':'green', 'fontFamily':'宋体'});  });  });  </script> |

|  |
| --- |
| 实现的效果：  点击按钮，改变按钮的value值。  鼠标进入按钮，按钮的宽、高改变。  鼠标离开的时候，按钮的背景颜色为绿色。 |
| <div class="container">  <input type="button" id="btn" value="按钮" />  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn')**.click**(function () {  $(this).val('val改变');  })**.mouseover**(function () {  $(this).css({'width':'200px','height':'200px'});  })**.mouseout**(function () {  $(this).css('backgroundColor','pink');  });  });  </script> |

3、获得兄弟元素的几个方法

（1）next(); //当前元素之后的紧邻着的第一个兄弟元素（下一个）。

（2）nextAll();//当前元素之后的所有兄弟元素。

（3）prev();//当前元素之前的紧邻着的兄弟元素（上一个）。

（4）prevAll();//当前元素之前的所有兄弟元素。

（5）siblings();//当前元素的所有兄弟元素。

|  |
| --- |
| <style type="text/css">  \* {  padding: 0;  margin: 0;  }  .container {  margin: 50px;  }  #fruit li {  margin: 5px;  border: 1px solid #ccc;  }  </style> |
| <div class="container">  <ul id="fruit">  <li>西瓜</li>  <li>橙子</li>  <li>葡萄</li>  <li>芒果</li>  <li>杨梅</li>  </ul>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#fruit>li').click(function () {  // $(this).next().css('backgroundColor', 'pink');  // $(this).nextAll().css('backgroundColor', 'pink');  // $(this).prev().css('backgroundColor', 'pink');  // $(this).prevAll().css('backgroundColor', 'pink');  $(this).siblings().css('backgroundColor', 'pink');  });  });  </script> |

4、end()解决断链

回到最近的一个"破坏性"操作之前。即，将匹配的元素列表变为前一次的状态。

如果之前没有破坏性操作，则返回一个空集。所谓的"破坏性"就是指任何改变所匹配的jQuery元素的操作。这包括在 Traversing 中任何返回一个jQuery对象的函数：'add'， 'andSelf',，'children'， 'filter'， 'find'， 'map'，'next'，'nextAll'，'not'，'parent'，'parents'，'prev'，'prevAll'，'siblings' and 'slice'--再加上 Manipulation 中的 'clone'。

解决断链：恢复到断链之前的一个效果，修复断链。

|  |
| --- |
| .container {  margin: 50px;  }  #beer li {  margin: 5px;  border: 1px solid #ccc;  text-align: center;  cursor: pointer;  }  <div class="container">  <ul id="beer">  <li>青岛啤酒（TsingTao）</li>  <li>瓦伦丁（Wurenbacher）</li>  <li>雪花（SNOW）</li>  <li>奥丁格教士（Franziskaner）</li>  <li>科罗娜喜力柏龙（Paulaner）</li>  <li>嘉士伯Kaiserdom</li>  <li>罗斯福（Rochefort）</li>  <li>粉象（Delirium）</li>  <li>爱士堡（Eichbaum）</li>  <li>哈尔滨牌蓝带</li>  </ul>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#beer>li').click(function () {  $(this).prevAll().css('backgroundColor', 'yellow')**.end()**.nextAll().css('backgroundColor', 'pink');  }).mouseenter(function () {  $(this).css('backgroundColor','red').siblings('li').css('backgroundColor','');  }).mouseleave(function () {  $(this).css('backgroundColor', '').siblings().css('backgroundColor', '');  });  });  </script> |
| **$(this).prevAll().css('backgroundColor', 'yellow')**.nextAll().css('backgroundColor', 'pink');  红色部分返回是科罗前边的集合，这个集合去调用nextAll，此时就会把集合第一个元素设置黄色，其他粉色 |

九、jQuery动画

1、show()、hide()

参数类型：可以是数值类型，字符串类型

数值类型：其单位是毫秒。

hide([speed,[easing],[fn]])

speed:三种预定速度之一的字符串("slow","normal", or "fast")或表示动画时长的毫秒数值(如：1000)

easing:(Optional) 用来指定切换效果，默认是"swing"，可用参数"linear"

fn:在动画完成时执行的函数，每个元素执行一次。

字符串类型：slow(对应600ms)、normal(对应400ms)、fast(对应200ms)

|  |
| --- |
| div {  width: 100px;  height: 100px;  background-color: pink;  }  <input type="button" value="显示" id="show" />  <input type="button" value="隐藏" id="hide" />  <div></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#hide').click(function () {  $('div').hide('fast');  });  $('#show').click(function () {  $('div').show('slow');  });  });  </script> |

2、last()、first()动画案例

|  |
| --- |
| $('#hide').click(function () {  $('.wrapper>ul>li').last().hide(500, function () {  $(this).prev().hide(500, arguments.callee);  });  });  $('#show').click(function () {  $('.wrapper>ul>li').first().show(500, function () {  $(this).next().show(500, arguments.callee);  });  });  **$(this).prev()**.hide(500, arguments.callee);这个是hide的调用者，所以下次调用的话，this指向了**$(this).prev()，初步理解** |

hide(800, arguments.callee)相当于是递归，但是这递归是不需要条件的。系统帮我们处理

3、arguments.callee的作用及替换方案

（1）arguments.callee的作用

　　 arguments 的主要用途是保存函数参数， 但这个对象还有一个名叫 callee 的属性，返回正被执行的 Function 对象，也就是所指定的 Function 对象的正文，这有利于匿名函数的递归或者保证函数的封装性。

　　 请看下面这个非常经典的阶乘函数

|  |
| --- |
| function factorial(num){  if (num <=1) {  return 1;  } else {  return num \* factorial(num-1)  }  } |

定义阶乘函数一般都要用到递归算法；如上面的代码所示，在函数有名字，而且名字以后也不会变的情况下，这样定义没有问题。

　　但问题是这个函数的执行与函数名 factorial 紧紧耦合在了一起。为了消除这种紧密耦合的现象，可以像下面这样使用 arguments.callee

|  |
| --- |
| function factorial(num){  if (num <=1) {  return 1;  } else {  return num \* arguments.callee(num-1);  }  } |

　　在这个重写后的 factorial()函数的函数体内，没有再引用函数名 factorial。这样，无论引用函数时使用的是什么名字，都可以保证正常完成递归调用。

（2）arguments.callee的替换方案

现在已经不推荐使用arguments.callee()；

原因：访问 arguments 是个很昂贵的操作，因为它是个很大的对象，每次递归调用时都需要重新创建。影响现代浏览器的性能，还会影响闭包。

（3）一道面试题。接受参数n=5,不用for循环输出数组【1,2,3,4,5】

这是用递归的思路，配合arguments.callee，代码如下

|  |
| --- |
| function show(n) {  var arr = [];  return (function () {  arr.unshift(n);  n--;  if (n != 0) {  arguments.callee();  }  return arr;  })()  }  show(5)//[1,2,3,4,5] |

　　现在arguments.callee 被弃用了。怎么办，其实很简单，给内部函数一个名字即可（当函数被调用时，它的arguments.callee对象就会指向自身，也就是一个对自己的引用。）

|  |
| --- |
| function show(n) {  var arr = [];  return (function fn() {  arr.unshift(n);  n--;  if (n != 0) {  fn();  }  return arr;  })()  }  show(5)//[1,2,3,4,5] |

4、slideUp()，slideDown，sliderToggle方法

这三个方法都可以写参数

参数：可以是数值类型 1000毫秒

参数：可以是字符串，slow、normal、fast

向上滑（滑入---消失）、向下滑（滑出---出现）、切换滑。

|  |
| --- |
| <input type="button" value="滑出" id="show" />  <input type="button" value="滑入" id="hide" />  <input type="button" value="切换滑" id="switch">  <div></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#hide').click(function () {  $('div').slideUp();  });  $('#show').click(function () {  $('div').slideDown();  });  $('#switch').click(function () {  $('div').slideToggle();  });  });  </script> |

5、fadeIn()，淡入，fadeOut(1000)，淡出，fadeToggle(1000)，切换淡入淡出

参数与上边一致。

6、fadeTo(1000, 0.3);

参数1:没有时间, 参数2:到达透明度，和css("opacity",0.1);效果一样。

在规定时间内达到指定透明度。

7、animate()方法

第一个参数:键值对，(数值的属性可以改,颜色不能改)

第二个参数:动画的时间

第三个参数:回调函数

|  |
| --- |
| div {  width: 200px;  height: 200px;  background-color: pink;  position: absolute;  }  <input type="button" value="效果" id="btn" />  <div></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  $('div').animate({'width':'100px','backgroundColor':'red'}, 1000, function () {  $('div').animate({'left':'200px','top':'200px'}, 500);  });  });    });  </script> |

8、停止动画效果:stop()方法，解决下拉框案例中的bug

十、jQuery中创建元素及追加元素

DOM中创建元素：

document.write();缺陷就是页面加载后创建元素，把页面中原有的内容全部的清除了

innerHTML

document.createElement()

1、jQuery中创建元素的方式:

（1）$(“标签的代码”)

（2）对象.html(“标签的代码”);

如果直接htm设置内容后，原内容丢失，返回的是当前对象。

如果不想丢失，得先获取原来的html中的内容，然后进行字符串的拼接，最后在设置html中的内容。

|  |
| --- |
| <style type="text/css">  #box {  width: 200px;  height: 100px;  background-color: pink;  margin: 10px;  }  </style> |
| <input type="button" id="btn" value="添加效果" />  <div id="box">  <p>我是原来就有的p标签</p>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  **var origin = $('#box').html();**  **var newP = '<p>我是新创建的p标签</p>';**  **$('#box').html(origin + newP);**  });  });  </script> |

2、添加元素方法

（1）append()，当前元素追加子元素

用来在元素的末尾追加元素（最后一个子节点）。增加元素末尾(**儿子**)。

|  |
| --- |
| <input type="button" id="btn" value="添加超链接" />  <div id="box"></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  // 先创建后追加  var a = $('<a href="javascript:void(0);">百度</a>');  console.log(a);  $('#box').append(a);  });  });  </script>  1 |

（1）append方法可以把页面中的元素添加到其他标签中,添加的方式属于移动，类似于剪切

|  |
| --- |
| <style type="text/css">  #box {  width: 100px;  height: 100px;  background-color: pink;  margin: 10px;  }  #box1 {  width: 200px;  height: 200px;  background-color: purple;  }  </style>  <input type="button" id="btn" value="添加效果" />  <div id="box"><p>111111</p></div>  <div id="box1"></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  **$('#box1').append($('#box>p'));**  });  });  </script>  12 |

（2）prepend()

在元素的开始添加元素（第一个子节点）。增加元素开始(**儿子**)

（3）after()、before()

在当前元素之后添加元素（添加兄弟），增加元素后面(**兄弟**)。

在当前元素之前添加元素（添加兄弟），增加元素前面(**兄弟**)

|  |
| --- |
| <input type="button" id="btn" value="添加效果" />  <div id="box"></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  var i = 0;  $('#btn').click(function () {  i++;  var a = $('<a href="javascript:void(0);">百度'+ i +'</a>');  console.log(a);  **$('#box').before(a);**  });  });  </script>  1  2 |

（4）appendTo()方法

所创建的元素主动加入到哪里去

|  |
| --- |
| <style type="text/css">  #box {  width: 100px;  height: 100px;  background-color: pink;  margin: 10px;  }  </style> |
| <input type="button" id="btn" value="添加效果" />  <div id="box"></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  var p = $('<p></p>')  p.text('hello world');  // $('#box').append(p);  // 主动加到box中  **p.appendTo($('#box'));**  });  });  </script> |

3、动态创建表格

通过 数组 拼接 字符串，然后给html

|  |
| --- |
| <style type="text/css">  #container {  margin: 50px;  }  table {  border-collapse: collapse;  }  table th {  width: 100px;  height: 40px;  background-color: skyblue;  }  table td {  height: 40px;  background-color: lightyellow;  }  </style> |
| <input type="button" id="btn" value="动态创建表格" />  <div id="container">  <table border="1">  <thead></thead>  <tbody></tbody>  </table>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  // 模拟从后台拿到的数据  var thead = ['分类', '课程'];  var data = [  {  name: "传智播客",  url: "javascript:void(0);",  type: "IT最强培训机构"  },  {  name: "黑马程序员",  url: "javascript:void(0);",  type: "大学生IT培训机构"  },  {  name: "传智前端学院",  url: "javascript:void(0);",  type: "前端的黄埔军校"  }];  $('#btn').click(function () {  var theadHTML = [];  theadHTML.push('<tr>');  for(var i = 0, len = thead.length; i < len; i++) {  theadHTML.push('<th>');  theadHTML.push(thead[i]);  theadHTML.push('</th>');  }  theadHTML.push('</tr>');  $('thead').html(theadHTML.join(''));  var tbodyHTML = [];  for (var i = 0, len = data.length; i < len; i++) {  tbodyHTML.push('<tr>');  tbodyHTML.push('<td><a href=\"' + data[i].url + '\">' + data[i].name + '</td>' );  tbodyHTML.push('<td>' + data[i].type + '</td>' );  tbodyHTML.push('</tr>');  }  $('tbody').html(tbodyHTML.join(''));  });  });  </script> |
| 1 |

4、清空元素方法

（1）html(‘’)置为空。清空元素中的内容

（2）empty();清除元素中的内容

|  |
| --- |
| #box {  width: 100px;  height: 100px;  background-color: red;  } |
| <input type="button" id="btn" value="清除元素内容">  <div id="box">  <p>hello world</p>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  $('#box').empty();  });  });  </script> |

5、移除元素方法

（1）remove()，移除元素自身。

这个方法不会把匹配的元素从jQuery对象中删除，因而可以在将来再使用这些匹配的元素。但除了这个元素本身得以保留之外，其他的比如**绑定的事件，附加的数据等都会被移除**。

某个节点使用该方法后，该节点所包含的所有后代节点也都会被同时删除，方法返回值是一个指向已被删除的节点的引用，所以以后还可以使用这些元素

|  |
| --- |
| <style type="text/css">  #box {  width: 100px;  height: 100px;  background-color: red;  }  </style> |
| <input type="button" id="btn" value="移除元素">  <input type="button" id="btn1" value="重新添加刚才删除的元素">  <div id="box">  <p>hello world</p>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  var temp;  $('#btn').click(function () {  temp = $('#box').remove();  });  $('#box').click(function () {  console.log('i am a box ');  });  $('#btn1').click(function () {  $(this).after(temp);  });  });  </script> |

6、克隆元素方法

参数意义：

V1.0：一个布尔值（true 或者 false）指示事件处理函数是否会被复制。V1.5以上版本默认值是：false。

V1.5：可以有俩参数

1：一个布尔值（true 或者 false）指示事件处理函数是否会被复制。

2：一个布尔值，指示是否对事件处理程序和克隆的元素的所有子元素的数据应该被复制。

|  |
| --- |
| 目前用版本的是1.12  <style type="text/css">  #box {  width: 100px;  height: 100px;  background-color: red;  }  </style> |
| <input type="button" id="btn" value="克隆元素">  <div id="box">  <p>hello world</p>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  var temp;  $('#btn').click(function () {  temp = $('#box').clone(**true**); // 默认是false  $(this).after(temp);  });  $('#box').click(function () {  console.log('i am a box ');  });  });  </script> |

十一、设置和获取表单的value

1、常用的表单控件

|  |
| --- |
| <body>  <input type="button" id="btn" value="按钮" />  <input type="text" id="txt" value="文本框" />  <input type="radio" id="radio" value="单选框" />男  <input type="checkbox" id="checkbox" name="复选框" />水果  <textarea>文本域</textarea>    <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  for (var i = 0; i < 4; i++) {  console.log($('body>input:eq('+i+')').val());  }  console.log($('body>textarea').val());  });  </script>  </body> |
| 1 |

表单控件中的**单选按钮、复选按钮**中的value值，不是给前端使用的。是给后台用来提交用的，一般情况下，不使用。

获取文本域中的内容，有两种方式，一是text方法，二是val方法，推荐使用第二种，因为文本域是属于表单控件。

2、select下拉菜单中的value

在select中.val()可以使得某一个选项被选中

|  |
| --- |
| <select id="fruit">  <option value="1">西瓜</option>  <option value="2">葡萄</option>  <option value="3">橙子</option>  <option value="4" selected="selected">芒果</option>  </select>  <input type="button" id="btn" value="随机选择其他">    <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  function getRandom(min, max) {  return Math.floor(Math.random()\*(max - min + 1) + min);  }  $(function () {  $('#btn').click(function () {  **$('#fruit').val(getRandom(1,4)+'');**  });  });  </script> |

3、设置、获取系统属性值或自定义属性

（1）attr()方法

参数一：属性

参数二：属性值

只写了一个参数，获取该元素的这个属性的值。

|  |
| --- |
| #box {  width: 100px;  height: 100px;  background-color: red;  }  <input type="button" id="btn" value="添加超链接">  <div id="box"></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  var a = $('<a></a');  // a.title = '电子科技大学中山学院';这样是不行的，此时的a不是DOM对象，是jQuery对象  a.attr('title', '电子科技大学中山学院');  a.text('ZSC');  a.attr('href', 'javascript:void(0);');  a.appendTo($('#box'));  console.log(a.attr('href'));  });  });  </script> |
| 1 |

（2）prop方法

当该方法用于返回属性值时，则返回第一个匹配元素的值。

当该方法用于设置属性值时，则为匹配元素集合设置一个或多个属性/值对。

prop() 方法应该用于检索属性值，例如 DOM 属性（如 selectedIndex、tagName、nodeName、nodeType、ownerDocument、defaultChecked 和 defaultSelected）

|  |
| --- |
| prop() 语法  1、返回属性的值：  $(selector).prop(property)  2、设置属性和值：  $(selector).prop(property,value)  3、使用函数设置属性和值：  $(selector).prop(property,function(index,currentvalue))  4、设置多个属性和值：  $(selector).prop({property:value, property:value,...}) |

|  |
| --- |
| prop() 和 attr() 之间的不同  1、操作对象不同  很明显，attr和prop分别是单词attribute和property的缩写，并且它们均表示"属性"的意思。  不过，在jQuery中，attribute和property却是两个不同的概念。attribute表示**HTML文档节点的属性**，property表示**JS对象的属性**。  <!-- 这里的id、class、data\_id均是该元素文档节点的attribute -->  <div id="message" class="test" data\_id="123"></div>    <script type="text/javascript">  // 这里的name、age是Person的property  var Person = { name: "小明", age: 18};  </script>  在jQuery中，prop()函数的设计目标是用于设置或获取指定DOM元素(指的是JS对象，Element类型)上的属性(property)；  attr()函数的设计目标是用于设置或获取指定DOM元素所对应的文档节点上的属性(attribute)。  2、用于设置的属性值类型不同  由于attr()函数操作的是文档节点的属性，因此设置的属性值只能是字符串类型，如果不是字符串类型，也会调用其toString()方法，将其转为**字符串类型**。  prop()函数操作的是JS对象的属性，因此设置的属性值可以为包括数组和对象在内的**任意类型。**  3、表单元素的checked、selected、disabled等属性，在jQuery 1.6之前，attr()获取这些属性的返回值为Boolean类型：如果被选中(或禁用)就返回true，否则返回false。  但是从1.6开始，使用attr()获取这些属性的返回值为String类型，如果被选中(或禁用)就返回checked、selected或disabled，否则(即元素节点没有该属性)返回undefined。并且，在某些版本中，这些属性值表示文档加载时的初始状态值，即使之后更改了这些元素的选中(或禁用)状态，对应的属性值也不会发生改变。  因为jQuery认为：**attribute**的checked、selected、disabled就是表示该属性初始状态的值，**property**的checked、selected、disabled才表示该属性实时状态的值(值为true或false)。 |

总结

|  |
| --- |
| 使用jQuery操作的元素属性时：  布尔值属性：使用prop() 。  非布尔值属性：使用attr() 。  对于HTML元素本身就带有的固有属性，在处理时，使用prop方法。  对于HTML元素我们自己自定义的DOM属性，在处理时，使用attr方法。  举例子：  <a href="http://www.baidu.com" target="\_self" class="btn">百度</a>  这个例子里<a>元素的DOM属性有“href、target和class"，这些属性就是<a>元素本身就带有的属性，也是W3C标准里就包含有这几个属性，或者说在IDE里能够智能提示出的属性，这些就叫做固有属性。处理这些属性时，建议使用prop方法。  <a href="#" id="link1" action="delete">删除</a>  这个例子里<a>元素的DOM属性有“href、id和action”，很明显，前两个是固有属性，而后面一个“action”属性是我们自己自定义上去的，<a>元素本身是没有这个属性的。这种就是自定义的DOM属性。处理这些属性时，建议使用attr方法。使用prop方法取值和设置属性值时，都会返回undefined值。 |

|  |
| --- |
| <input type="button" id="btn" value="获取状态" />  <input type="checkbox" id="ck" value="1" />复选框  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  if ($('#ck').**prop**('checked')) {  $('#ck').**prop**('checked', false);  } else {  $('#ck').**prop**('checked', true);  }  });  });  </script> |

使用attr无法获取实时状态。

（3）:checkbox选择器

|  |
| --- |
| <input type="button" id="btn" value="全选" />  <input type="button" id="btn1" value="全不选">  <input type="checkbox">1  <input type="checkbox">2  <input type="checkbox">3  <input type="checkbox">4  <input type="checkbox">5  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  $('body :checkbox').prop('checked', true);// 要加上空格  });  $('#btn1').click(function () {  $('body :checkbox').prop('checked', false);// 要加上空格  });  });  </script> |

4、全选和反选案例

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <style type="text/css">  .container {  margin: 50px;  }  #table {  border-collapse: collapse;  }  th {  width: 100px;  background-color: skyblue;  }  td {  text-align: center;  background-color: lightyellow;  }  </style>  </head>  <body>  <div class="container">  <table id="table" border="1">  <thead>  <tr>  <th><input type="checkbox" id="checkAll" /></th>  <th>课程</th>  <th>分类</th>  </tr>  </thead>  <tbody>  <tr>  <td><input type="checkbox" /></td>  <td>c++</td>  <td>计算机</td>  </tr>  <tr>  <td><input type="checkbox" /></td>  <td>c++</td>  <td>计算机</td>  </tr>  <tr>  <td><input type="checkbox" /></td>  <td>c++</td>  <td>计算机</td>  </tr>  </tbody>  </table>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  /\*$('th>input').click(function () {  if ($(this).prop('checked')) {  $('tbody :checkbox').prop('checked', true);  } else {  $('tbody :checkbox').prop('checked', false);  }  });\*/  $('th>input').click(function () {  var check = $(this).prop('checked');  $('tbody :checkbox').prop('checked', check);  });  $('tbody :checkbox').click(function () {  var checkboxNum = $('tbody').find(':checkbox').**length**;  var checkedNum = $('tbody').find(':checked').**length**;  // console.log(checkboxNum,checkedNum);  if (checkboxNum === checkedNum) {  $('thead :checkbox').prop('checked', true);  } else {  $('thead :checkbox').prop('checked', false);  }  });  // 代码要比原生的js少很多，直接能选出所有的数量  });  </script>  </body>  </html> |

十二、设置和获取元素的宽和高

1、设置宽、高

（1）css方式获取的是字符串类型的数据，需进行转换。

通过.css()方法写一个属性可以获取宽或者高，是字符串类型。如果获取后重新设置需要转换，麻烦

|  |
| --- |
| div {  width: 100px;  height: 100px;  background-color: pink;  }  <input type="button" id="btn" value="显示效果">  <div></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  var width = $('div').**css**('width');  var height = $('div').**css**('height');  console.log(width, height);  var width = **parseInt**($('div').css('width'));  var height =**parseInt**($('div').css('height'));  console.log(width, height);  });  });  </script>  1 |

（2）width()、height()直接获取数值型数据

设定CSS中 'width' 的值，可以是**字符串或者数字**，还可以是一个**函数(暂时没有进行测试，好像不能指定参数)**，返回要设置的数值。函数接受两个参数，第一个参数是元素在原先集合中的索引位置，第二个参数为原先的宽度。

|  |
| --- |
| <input type="button" id="btn" value="显示效果">  <div></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  var width = $('div').**width**();  var height = $('div').**height**();  console.log(width, height);  $('div').width(**300**);  $('div').height(**'300px'**);  });  });  </script>  1 |

2、位置操作

（1）offset()方法返回的是对象,并且,对象中有一个left和一个top，并且值是数字类型。

设置的时候也可以没有px。

设置的时候元素在设置前如果没有脱离文档流，设置的时候会自动进行脱离文档流，默认为relative。

如果设置前有脱离文档流，那么设置的时候直接改变位置。

**获取匹配元素在当前视口的相对偏移。跟DOM中的是有区别的，这个要注意**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <style type="text/css">  \* {  margin: 0;  padding: 0;  }  .father {  width: 200px;  height: 200px;  position: relative;  background-color: pink;  margin-top: **100px**;  margin-left: **100px**;  }  .son {  width: 100px;  height: 100px;  position: absolute;  margin-left: **100px**;  left: **100px**;  background-color: blue;  }  </style>  </head>  <body>  <input type="button" id="btn" value="显示效果">  <div class="father">  <div class="son"></div>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  console.log('jQuery');  console.log($('.son').offset());  console.log($('.son').offset().left,$('.son').offset().top);  console.log('DOM');  console.log(document.getElementsByClassName('son')[0].offsetLeft);  });  });  </script>  </body>  </html>  1 |

（2）position()

获取匹配元素相对父元素的偏移。

返回值：Object{top,left}

返回的对象包含两个整型属性：top 和 left。为精确计算结果，请在补白、边框和填充属性上使用像素单位。此方法只对可见元素有效。

（3）e.offsetX，不限于jQuery

clientX：当鼠标事件发生时（不管是onclick，还是omousemove，onmouseover等），鼠标相对于浏览器（这里说的是浏览器的有效区域）x轴的位置；

clientY：当鼠标事件发生时，鼠标相对于浏览器（这里说的是浏览器的有效区域）y轴的位置；

screenX：当鼠标事件发生时，鼠标相对于显示器屏幕x轴的位置；

screenY：当鼠标事件发生时，鼠标相对于显示器屏幕y轴的位置；

offsetX：当鼠标事件发生时，鼠标相对于事件源x轴的位置

offsetY：当鼠标事件发生时，鼠标相对于事件源y轴的位置

（4）固定导航栏案例

|  |
| --- |
| $(function)文档加载后执行所以有可能获取不到图片高度。  解决：$(window).load(); |

十三、事件绑定

1、bind()

bind()方法：第一个参数是事件名字，第二个参数是事件处理函数-匿名函数。

它的内部是调用了 on 函数。可以在js中看下源码的参数，它是没有选择器的，不能设置后代。

它可以为元素同时绑定多个事件。（感觉跟链式编程没什么区别，暂时没体会到其好处）

|  |
| --- |
| <input type="button" id="btn" value="显示效果">  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  // 链式编程  /\* $('#btn').click(function () {  console.log('点击事件');  }).mouseover(function () {  $(this).css('backgroundColor', 'green');  }).mouseout(function () {  $(this).css('backgroundColor', '');  });\*/  // 使用bind的方式，绑定一个事件  /\*$('#btn').bind('click', function () {  console.log('bind方式的点击事件');  });\*/  // 绑定多个事件，类似于传入一个对象，使用其中的方法  $('#btn').bind({'click': function () {  console.log('绑定多个事件click');  }, 'mouseover': function () {  $(this).css({'backgroundColor': 'orange'});  }, 'mouseout': function () {  $(this).css({'backgroundColor': ''});  }});  });  </script> |

|  |
| --- |
| 这个用法比较少用。  var i = 0; //多个事件绑定用一个事件  $("#btn"). bind("mouseover mouseout", function () {  i++;  console. log(i);  }); |

2、delegate()

此方法是父级元素为其后代的子元素绑定事件，不是它的后代就不行。

参数：

（1）要绑定的事件的元素，一个**jQuery选择器**，用于指定哪些**后代元素**可以触发绑定的事件。如果该参数为null或被省略，则表示当前元素自身绑定事件(实际触发者也可能是后代元素，只要事件流能到达当前元素即可)。

（2）要绑定的事件的名字---click

（3）绑定事件的处理函数---匿名函数

|  |
| --- |
| <style type="text/css">  \* {  margin: 0;  padding: 0;  }  #box {  width: 200px;  height: 200px;  background-color: orange;  }  </style>  <input type="button" id="btn" value="显示效果">  <div id="box">  <p>我是原来的p标签</p>  </div>  <p>我是box兄弟的p标签</p>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  var p = $('<p></p>');  p.text('我是新建的子代的 p 标签');  p.appendTo($('#box'));  /\*p.click(function () {  console.log('我是顺便被绑定的点击事件');// 这种跟delegate有大区别，这种方式只能是单个有效 });\*/  // delegate()方法  $('#box').delegate('p', 'click', function () {  console.log($(this).text());  });  });  });  </script> |

即使是执行delegate()函数之后新添加的元素，只要它符合条件，绑定的事件处理函数仍然对其有效。

此外，该函数可以为同一元素、同一事件类型绑定多个事件处理函数。触发事件时，jQuery会按照绑定的先后顺序依次执行绑定的事件处理函数。

**重要说明：**delegate()函数并不是为当前jQuery对象匹配的元素绑定事件处理函数，而是为它们的后代元素中符合选择器selector参数的元素绑定事件处理函数。

delegate()函数**并不是直接为这些后代元素挨个绑定事件**，而是"委托"给当前jQuery对象的匹配元素来处理。

由于DOM 2级的事件流机制，当后代元素selector触发事件时，该事件会在**事件冒泡**中传递给其所有的祖辈元素，当事件流传递到当前匹配元素时，jQuery会判断是哪个后代元素触发了事件，如果该元素符合选择器selector，jQuery就会捕获该事件，从而执行绑定的事件处理函数。on也使用。

3、绑定事件on（与delegate、bind相比，推荐使用这个）

如果你的html在注册事件之后，没有变化的，用click和用on注册事件都是可以的。

但如果你的html在注册事件后，还会通过JS脚本添加一些按钮，并者希望这些按钮也会有之前注册事件的按钮同样的事件话，就需要用on去为按钮的父节点去注册事件，而不应该用click去为按钮直接注册事件。

如果元素是动态添加的，那么非常推荐用on的方式为动态添加的元素绑定事件，使用click的话无法达到效果

on()方法，可以理解成事件委托。

（1）参数：

两个：事件的名字，事件处理函数。

三个：事件的名字，绑定子元素的选择器，事件处理函数。

on是父级元素调用的，目的为了子级元素去绑定事件。

|  |
| --- |
| #box {  width: 200px;  height: 200px;  background-color: orange;  }  p {  background-color: gray;  }  <input type="button" id="btn" value="显示效果">  <div id="box">  <p>我是原来的p标签</p>  </div>  <p>我是box兄弟的p标签</p>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  var p = $('<p></p>');  p.text('我是新建的子代的 p 标签');  p.appendTo($('#box'));  // on()方法，利用事件冒泡的机制,加上一些条件的判断，例如是某个选择器才发生  $('#box').on('click', 'p', function (e) {  console.log(e.target);  });  });  });  </script> |

4、表格动态添加数据

之前的案例是在设置数据的时候，就把事件顺便注册进去了。采用的是createElement的方式进行添加的，在jQuery中直接是采用字符串拼接方式。而且没有做动态添加的部分。如果做的话，也是在创建的时候就注册事件。

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <style type="text/css">  #container {  width: 300px;  margin: 0 auto;  }  table {  border-collapse: collapse;  }  table th {  width: 100px;  height: 40px;  background-color: skyblue;  }  table td {  height: 40px;  background-color: lightyellow;  text-align: center;  }  a {  text-decoration: none;  }  /\*输入数据样式\*/  .bg {  width: 100%;  height: 100%;  background-color: rgba(0,0,0,0.2);  position: absolute;  top: 0;  display: none;  }  .addDataArea {  width: 300px;  height: 200px;  background-color: #fff;  position: relative;  top: 50%;  left: 50%;  transform: translate(-50%, -50%);  font-size: 18px;  }  .top {  background-color: #999;  font-weight: 900;  height: 30px;  line-height: 30px;  font-size: 14px;  margin-bottom: 20px;  }  .main div {  height: 40px;  }  .main div input {  float: right;  width: 160px;  height: 25px;  border: 1px solid #000;  margin-right: 10px;  }  .confirm {  width: 100px;  height: 30px;  margin: 0 auto;  display: block;  cursor: pointer;  }  .close {  position: absolute;  top: 0;  right: 10px;  cursor: pointer;  }  </style>  </head>  <body>  <div id="container">  <input type="button" id="addSubject" value="增加课程" />  <table border="1">  <thead>  <tr>  <th>课程</th>  <th>学院</th>  <th>是否删除</th>  </tr>  </thead>  <tbody>  <tr>  <td>c</td>  <td>计算机</td>  <td><a href="javascript:void(0);">删除</a></td>  </tr>  <tr>  <td>c</td>  <td>计算机</td>  <td><a href="javascript:void(0);">删除</a></td>  </tr>  <tr>  <td>c</td>  <td>计算机</td>  <td><a href="javascript:void(0);">删除</a></td>  </tr>  </tbody>  </table>  </div>  <div class="bg">  <div class="addDataArea">  <div class="top">添加数据</div>  <div class="main">  <div><span>课程名称：</span><input type="text" placeholder="请输入课程名称" /></div>  <div><span>学院：<span><input type="text" placeholder="请输入学院" /></span></div>  <input type="button" value="确认" class="confirm">  <span class="close">x</span>  </div>  </div>  </div>      <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('.close').click(function () {  $('.bg').css('display','none');  });  $('.confirm').click(function () {  var subject = $('.main input:eq(0)').val();  var xueyuan = $('.main input:eq(1)').val();  if (subject !== '' && xueyuan !== '') {  var str = '<tr><td>'+subject+'</td><td>'+xueyuan+'</td><td><a href="javascript:void(0);">删除</a></td></tr>';  $('tbody').append($(str));  $('.close').click();  console.log(subject, xueyuan);  }  // 可以把字符串清空一下，下次进行就不会存在之前的内容  $('.main input:eq(0)').val('');  $('.main input:eq(1)').val('');  });  $('#addSubject').click(function () {  $('.bg').css('display', 'block');  });  //  $('tbody').on('click', 'td a', function () {  console.log($(this));  $(this).parent().parent().remove();  });  //采用click方式,动态添加的元素为其无法注册事件  /\*$('tbody tr td a').click(function () {  console.log('hello');  });\*/  })  </script>  </body>  </html> |

十四、解绑事件

1、用什么方式绑定事件就要用对应的解绑事件解除绑定。

|  |
| --- |
| $("#btn"). bind("click", function(){});  $("#btn"). unbind("click");  $("#dv"). delegate("p", "click", function(){});  $("#dv"). undelegate("p", "click"); |
| <label>绑定事件<input type="button" id="btn1" value="绑定事件"></label><br />  <label for="btn2">解绑事件</label>  <input type="button" id="btn2" value="解绑事件">  <div id="box"></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn1').click(function () {  **$('#box').on('click', function () {**  **console.log('hello i am box');**  **});**  });  $('#btn2').click(function () {  **$('#box').off('click');**  });  });  </script> |

2、解绑多个事件

$("#btn1"). off("click mouseover"); 多个事件解绑加空格。

3、$('.father').off('click'); 父级click、子级click

如果说父级元素和子级元素都是通过click的方式绑定事件，通过off('click')解绑的时候，**父级**元素的事件**解绑**了，**子级**元素的事件**没有解绑**。（它们是分别去注册的，各管各的）

|  |
| --- |
| <style type="text/css">  .father {  width: 200px;  height: 200px;  background-color: red;  }  .son {  width: 100px;  height: 100px;  background-color: pink;  }  </style>  <label>绑定事件<input type="button" id="btn1" value="绑定box事件"></label><br />  <label for="btn2">解绑事件</label>  <input type="button" id="btn2" value="解绑box事件">  <div class="father">  <div class="son"></div>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn1').click(function () {  $('.father').click(function () {  console.log('i am father');  });  $('.son').click(function () {  console.log('i am son');  });  });  $('#btn2').click(function () {  **$('.father').off('click');**  });  });  </script> |

4、$('.father').off('click'); 父级click、子级 delegate

如果子级元素是通过父级元素调用delegate的方式绑定的事件，父级元素使用off('click')方式解绑事件，这个时候**父级**元素和**子级**元素相同的事件都会被**解绑**（此时冒泡上来，没有事件处理函数去处理）

|  |
| --- |
| <script type="text/javascript">  $(function () {  $('#btn1').click(function () {  $('.father').click(function () {  console.log('i am father');  });  $('.father').**delegate**('div', 'click', function () {  console.log('i am son');  });  });  $('#btn2').click(function () {  $('.father').off('click');  });  });  </script> |

5、$("div"). off("click", "\*\*"); 父级click、子级仅限delegate，**click无效**

这是特殊固定写法，把**子级**元素的点击事件**解绑**了，父级元素的点击事件还是存在的

|  |
| --- |
| <script type="text/javascript">  $(function () {  $('#btn1').click(function () {  $('.father').click(function () {  console.log('i am father');  });  $('.father').delegate('div', 'click', function () {  console.log('i am son');  });  /\*$('.son').click(function () {  console.log('i am son');  });\*/  });  $('#btn2').click(function () {  **$('.father').off('click', '\*\*');**  });  });  </script> |

6、$('.father').off(); 父级click、子级delegate，click无效

全部解绑。

|  |
| --- |
| $('#btn2').click(function () {  $('.father').off();  }); |

十五、触发事件

1、方式：

（1）click直接调用元素的事件方法:

$("div"). click();

（2）trigger()方法，扳机，需要写事件类型

$("div"). trigger("click");

（3）triggerHandle()方法，需要写事件类型

|  |
| --- |
| <label>绑定事件<input type="button" id="btn1" value="触发box的事件"></label><br />  <div class="box"></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn1').click(function () {  // 有三种方式  // $('.box').click();  // $('.box').trigger('click');  $('.box').triggerHandler('click');  });  $('.box').click(function () {  $('.box').css('backgroundColor', 'yellow');  });  });  </script> |

2、trigger与triggerHandle区别，是否触发浏览器默认行为

第（1）种触发事件的方式和第（2）种触发事件的方式是相同的，都会触发浏览器的默认的事件（获得焦点，光标在文本框中闪烁）

第（3）种触发事件的方式不会触发浏览器的默认行为。triggerHandler方法不会引起浏览器默认的行为，而仅仅是执行绑定到focus事件的处理程序.

|  |
| --- |
| <label>获取焦点事件<input type="button" id="btn1" value="触发焦点事件"></label><br />  <input type="text" id="txt">  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn1').click(function () {  $('#txt').triggerHandler('focus');  });  $('#txt').on('focus', function () {  $(this).val('focus事件处理函数');  });  });  </script> |

十六、事件对象

1、event事件对象

（1）event.delegateTarget：代码绑定事件的对象，谁负责去注册的。

（2）event.currentTarget：绑定事件的对象，注册的对象是谁。

（3）event.target：真正触发事件的对象。

|  |
| --- |
| <div class="father">  <div class="son"></div>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('.father').on('click', 'div', function (event) {  console.log(arguments.length);  console.log(arguments);  console.log(event);  console.log(event.delegateTarget);  console.log(event.currentTarget);  console.log(event.target);  });  });  </script>  1 |

2、键盘事件

（1）e.keyCode：键盘按下后对应的键值

|  |
| --- |
| <div class="father"></div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $(document).on('keydown', function (event) {  var keyCode = event.keyCode;  switch (keyCode) {  case 65:  $('.father').css('backgroundColor', 'red');  break;  case 66:  $('.father').css('backgroundColor', 'purple');  break;  case 67:  $('.father').css('backgroundColor', 'orange');  break;  }  });  });  </script> |

3、取消事件冒泡

（1）return false

（2）e.stopPropagation()

|  |
| --- |
| .father {  width: 200px;  height: 200px;  background-color: red;  }  .son {  width: 100px;  height: 100px;  background-color: yellow;  }  <div class="father">  <div class="son"></div>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('.father').click(function () {  console.log('i am father');  });  $('.son').click(function (e) {  console.log('i am son');  // return false;  e.stopPropagation();  });  });  </script> |

4、取消默认事件

（1）return false

（2）e.preventDefault();

|  |
| --- |
| <a href="http:\\www.baidu.com" id="link">百度</a>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#link').click(function (e) {  console.log('click');  // return false;  e.preventDefault();  });  });  </script> |

十七、链式编程原理

1、链式编程原理：构造内部返回了 return this当前对象。

html()，带参数与不带参数的处理，是通过对参数是否进行传值，来决定返回值。

|  |
| --- |
| <script type="text/javascript">  $(function () {  function Person(name) {  this.name = name;  this.sayName = function () {  console.log('hell i am '+this.name);  **return this;**  };  this.eat = function (foodType) {  if (**foodType**) {  console.log('i am eating ' + foodType);  **return this;**  } else {  console.log('i am eating fruit');  }    }  }  var person = new Person('Mike');  // sayName不返回当前对象会出错，它会返回undefined，这个返回值并没有eat方法  person.sayName().eat();  // person.eat('hotDog').sayName();  person.eat().sayName();  });  </script> |

2、评分小案例

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <style type="text/css">  \* {  margin: 0;  padding: 0;  }  ul {  list-style: none;  }  .command {  height: 60px;  }  .command li {  float: left;  font-size: 50px;  font-family: '宋体';  color: orange;  }  .clearfix:after {  content: '.';  display: block;  height: 0;  visibility: hidden;  clear: both;  }  .clearfix {  \*zoom: 1;  }  </style>  </head>  <body>  <div class="wrapper">  <ul class="command clearfix">  <li>☆</li>  <li>☆</li>  <li>☆</li>  <li>☆</li>  <li>☆</li>  </ul>  <!-- text("★"). -->  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('.command').on('mouseenter', 'li', function () {  $(this).text('★').prevAll().text('★');  $(this).nextAll().text('☆');  });  var index;  $('.command').on('mouseleave', 'li', function () {  if (index) {  $('.command>li:eq('+index+')').text('★').prevAll().text('★');  $('.command>li:eq('+index+')').nextAll().text('☆');  } else {  $('.command>li').text('☆');  }  });    $('.command>li').click(function () {  index = $(this).index();  console.log(index);  });  });  </script>  </body>  </html>  li的mouseleave每次都会执行，但是会被新的mouseenter的事件覆盖了 |

设置浮动之后，忘记它会造成内部高度为0 的问题，导致没有外部的盒子，那么mouset肯定不会起作用。按照的自己的想法做，太麻烦了，因为冒泡事件的叠加。要把事情集中到同一个物品上。

3、each()方法

jQuery中有隐式迭代，不需要我们再次进行遍历对某些元素进行操作。但是，如果涉及到对获取到的元素**有不同的操作**，那么需要进行each遍历。

在each()方法中可以写一个函数，系统会给这个函数传入多少参数，我们不知道，此时就可以先通过打印arguments.length来确定。再去打印arguments[0]来看具体参数是什么。

|  |
| --- |
| $(function () {  $('.wrapper li').each(function () {  console.log(arguments[0]);  console.dir(arguments[1]);  });  });  1  打印的是索引和获取到的DOM对象。  系统会帮我们去循环出来获取到的所有的li的each方法。  1 |

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <style type="text/css">  \* {  margin: 0;  padding: 0;  }  ul {  list-style: none;  }  .wrapper li {  float: left;  width: 100px;  height: 100px;  margin-left: 10px;  background-color: red;  color: black;    }  .clearfix:after {  content: '.';  display: block;  height: 0;  visibility: hidden;  clear: both;  }  .clearfix {  \*zoom: 1;  }  </style>  </head>  <body>  <div class="wrapper">  <ul class="clearfix">  <li>1</li>  <li>2</li>  <li>3</li>  <li>4</li>  <li>5</li>  <li>6</li>  <li>7</li>  <li>8</li>  <li>9</li>  <li>10</li>  </ul>  </div>  <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('.wrapper li').each(function (index, element) {  // console.log(arguments[0]);  // console.dir(arguments[1]);  $(element).css('opacity', (index+1)/10);  });  });  </script>  </body>  </html> |

十八、多库共存

一般情况下，遇到的比较少。

1、$.noConflict()方法

释放$的使用权

释放操作**必须在编写其它jQuery代码之前编写**。

释放之后就不能再使用$，改为使用jQuery。

假如同一个页面不仅引入了 jQuery的外部文件，也引入了其他的库文件如果此时其他的库文件中也使用了 $符号。

使用 :$.noConflict()解决。其他语言中:这个方式叫解决命名空间的冲突。

|  |
| --- |
| $(function () {  // 让jQuery释放 $ 的控制权  var replace$ = $.noConflict();  var $ = 100;  replace$('div').click(function () {  replace$(this).css('backgroundColor', 'skyblue');  });  });  这样子写是无法进行释放的。$在释放之前已经被占用了，或者是其他的原因，**暂时未知？？？**  **1**  1 |
| <script type="text/javascript" src="jQuery/jquery-1.12.2.js"></script>  <script type="text/javascript">  // 让jQuery释放 $ 的控制权  var replace$ = $.noConflict();  replace$(function () {  var $ = 100;  replace$('div').click(function () {  replace$(this).css('backgroundColor', 'skyblue');  });  });  </script> |

2、使用jQuery代替$

|  |
| --- |
| var $ = 100;  **jQuery**('div').click(function () {  **jQuery**(this).css('backgroundColor', 'skyblue');  }); |

十九、jQuery插件玩法

一个插件就是一个功能。百度搜jQuery之家或者其他。

（1）引入jquery文件。

（2）引入插件的js文件(还需要引入css文件)。

（3）添加加载的事件。

（4）复制代码即可---按照插件的介绍，一般有流程，但是可能会有一些小问题，如文件引用可能出错了。

1、jQueryUI使用步骤（视频接着第六）

（1）引入jQueryUI的样式文件

（2）引入jQuery

（3）引入jQueryUI的js文件

（4）使用 jQueryUI功能

看到的是一个集合所有ui的页面，先F12找到那个组件的id，然后再查看页面源码中搜索相关的代码。可以自己再给它增加新的类样式，来改变效果。

2、自己模仿做插件

$.fn.是固定的语法，后面加方法，这样这个方法就可以被所有的jQuery元素对象去进行调用

二十、其他方法

1、jQuery还提供了一个通用的jQuery.each方法，用来处理对象和数组的遍历语法。

第一个方法的参数传递的就是一个对象或者数组；

第二个方法它是个回调函数。index是索引,也就是数组的索引。value就是数组中的值了。

|  |
| --- |
| $.each(data.s, function (index, value) {  strArr.push('<li>'+ value +'</li>');  }); |

jquery的时候，类选择器总是少写一点

2、width相关

（1）width：它只能获取当前元素的width

（2）innerWidth()：它能获取当前元素的width + padding

（3）outerWidth()：获取当前的元素的width + padding + border

（4）outerWidth(true)：获取元素的width + padding + border + margin

在css布局的时候，能撑开的盒子尽量不要写死，从内容来进行撑开，如li不要写死宽度，高度。