**02【jQuery】 常用API**

学习内容: jQuery 选择器, jQuery 样式操作, jQuery 效果, jQuery 属性操作, jQuery 文本属性值, jQuery 元素操作, jQuery 尺寸、位置操作

**1. jQuery 选择器**

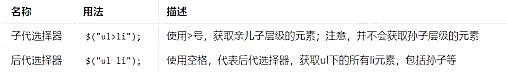
**1.1 jQuery 基础选择器**

原生 JS 获取元素方式很多，很杂，而且兼容性情况不一致，因此 jQuery 给我们做了封装，使获取元素统一标准。

$(“选择器”) // 里面选择器直接写 CSS 选择器即可，但是要加引号



**1.2 jQuery 层级选择器**



知识铺垫

**jQuery 设置样式: $('div').css('属性', '值'), 不需要循环=隐式迭代（内部进行遍历，给每个元素都添加）**

**1.3 隐式迭代（重要!!!）**

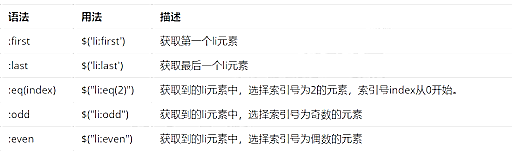
遍历内部 DOM 元素（伪数组形式存储）的过程就叫做隐式迭代。

简单理解：**给匹配到的【所有元素】进行循环遍历**，执行相应的方法，而不用我们再进行循环，简化我们的操作， 方便我们调用。

|  |
| --- |
| <body>      <div>惊喜不，意外不</div>      <div>惊喜不，意外不</div>      <div>惊喜不，意外不</div>      <div>惊喜不，意外不</div>      <ul>          <li>相同的操作</li>          <li>相同的操作</li>          <li>相同的操作</li>      </ul>      <script>          // 1. 获取四个div元素          console.log($("div"));          // 2. 给四个div设置背景颜色为粉色 jquery对象不能使用style          $("div").css("background", "pink");          // 3. 隐式迭代就是把匹配的所有元素内部进行遍历循环，给每一个元素添加css这个方法          $("ul li").css("color", "red");      </script>  </body> |

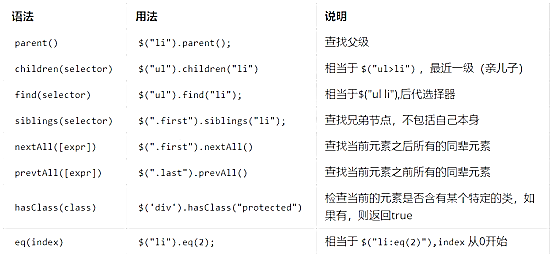
**1.4 jQuery 筛选选择器**

**想选中间某一个，1，3，5，or 2, 4, 6应该怎么选呢？--筛选选择器**



|  |
| --- |
| $(function() {              $("ul li:first").css("color", "red");              $("ul li:eq(2)").css("color", "blue");              $("ol li:odd").css("color", "skyblue");              $("ol li:even").css("color", "pink");          }) |

**1.5 jQuery 筛选方法（重点）**



**重点记住： parent() children() find() siblings() eq()**

|  |
| --- |
| <div class="yeye">          <div class="father">              <div class="son">儿子</div>          </div>      </div>      <div class="nav">          <p>我是屁</p>          <div>              <p>我是p</p>          </div>      </div>      <script>          // 注意一下都是方法 带括号          $(function() {              // 1. 父  parent()  返回的是 最近一级的父级元素 亲爸爸              console.log($(".son").parent());              // 2. 子              // (1) 亲儿子 children()  类似子代选择器  ul>li              // $(".nav").children("p").css("color", "red");              // (2) 可以选里面所有的孩子 包括儿子和孙子  find() 类似于后代选择器              $(".nav").find("p").css("color", "red");              // 3. 兄          });      </script> |
| <ol>          <li>我是ol 的li</li>          <li>我是ol 的li</li>          <li class="item">我是ol 的li</li>          <li>我是ol 的li</li>          <li>我是ol 的li</li>          <li>我是ol 的li</li>      </ol>      <ul>          <li>我是ol 的li</li>          <li>我是ol 的li</li>          <li>我是ol 的li</li>          <li>我是ol 的li</li>          <li>我是ol 的li</li>          <li>我是ol 的li</li>      </ul>      <div class="current">俺有current</div>      <div>俺木有current</div>      <script>          // 注意一下都是方法 带括号          $(function() {              // 1. 兄弟元素siblings 除了自身元素之外的所有亲兄弟              $("ol .item").siblings("li").css("color", "red");              // 2. 第n个元素              var index = 2;              // (1) 我们可以利用选择器的方式选择              // $("ul li:eq(2)").css("color", "blue");              // $("ul li:eq("+index+")").css("color", "blue");              // (2) 我们可以利用选择方法的方式选择 更推荐这种写法              // $("ul li").eq(2).css("color", "blue");              // $("ul li").eq(index).css("color", "blue");              // 3. 判断是否有某个类名              console.log($("div:first").hasClass("current"));              console.log($("div:last").hasClass("current"));          });      </script> |

**【案例】下拉菜单**

一个 ul 里分好多个 小 li， 每个小 li中再分（上下）两块，上边一个 a 下边一个ul ,

原先里边的 ul 都是 隐藏的，当鼠标放到小 li上的时候，让他显示出来

|  |
| --- |
| <ul class="nav">          <li>              <a href="#">微博</a>              <ul>                  <li>                      <a href="">私信</a>                  </li>                  <li>                      <a href="">评论</a>                  </li>                  <li>                      <a href="">@我</a>                  </li>              </ul>          </li>          <li>              <a href="#">微博</a>              <ul>                  <li>                      <a href="">私信</a>                  </li>                  <li>                      <a href="">评论</a>                  </li>                  <li>                      <a href="">@我</a>                  </li>              </ul>          </li> |
| <script>          $(function() {              // 鼠标经过              $(".nav>li").mouseover(function() {                  // $(this) jQuery 当前元素  this不要加引号                  // show() 显示元素  hide() 隐藏元素                  $(this).children("ul").show();              });              // 鼠标离开              $(".nav>li").mouseout(function() {                  $(this).children("ul").hide();              })          })      </script> |

**1.6 jQuery 里面的排他思想**

想要**多选一**的效果，排他思想：**当前元素设置样式，其余的兄弟元素清除样式。**

**$(this).css(“color”,”red”);**

**$(this).siblings(). css(“color”,””);**

|  |
| --- |
| <button>快速</button>      <button>快速</button>      <button>快速</button>      <button>快速</button>      <button>快速</button>      <button>快速</button>      <button>快速</button>      <script>          $(function() {              // 1. 隐式迭代 给所有的按钮都绑定了点击事件              $("button").click(function() {                  // 2. 当前的元素变化背景颜色                  $(this).css("background", "pink");                  // 3. 其余的兄弟去掉背景颜色 隐式迭代                  $(this).siblings("button").css("background", "");              });          })      </script> |

**【案例】淘宝服饰精品案例**

【案例分析】

① 核心原理：**鼠标经过左侧盒子某个小li**，就让**内容区盒子相对应图片显示，其余的图片隐**藏。

② 怎么知道放在了那个小li? 需要得到当前小li 的索引号，就可以显示对应索引号的图片

③ jQuery 得到当前元素索引号 **$(this).index()**

④ 中间对应的图片，可以通过 **eq(index)** 方法去选择

⑤ 显示元素 show() 隐藏元素 hide()

**1.7 链式编程**

链式编程是为了节省代码量，看起来更优雅。

**$(this).css('color', 'red').sibling().css('color', '');**

**2. jQuery 样式操作**

**2.1 操作 css 方法**

jQuery 可以使用 css 方法来修改简单元素样式； 也可以操作类，修改多个样式。

（1）. 参数只写属性名，则是返回**属性值**：$(this).css(''color'');

（2）. 参数是属性名，属性值，逗号分隔，是设置一组样式，属性**必须加引号**，**值如果是数字可以不用跟单位和引号（内部做了处理 ）**：$(this).css(''color'', ''red'');

（3）. 参数可以是 **对象形式**，方便设置多组样式**。属性名和属性值用冒号隔开**， 属性可以不用加引号，$(this).css({ "color":"white","font-size":"20px"});

|  |
| --- |
| <script>          // 操作样式之css方法          $(function() {              console.log($("div").css("width"));              // $("div").css("width", "300px");              // $("div").css("width", 300);              // $("div").css(height, "300px"); 属性名一定要加引号              $("div").css({                  width: 400,                  height: 400,                  backgroundColor: "red"               // **如果是复合属性则必须采取驼峰命名法，**如果值不是数字，则需要加引号              })          })      </script> |

**2.2 设置类样式方法**

作用等同于以前的 classList，可以操作类样式， 注意操作类里面的参数不要加点。

（1）. 添加类：$(“div”).addClass(''current'');

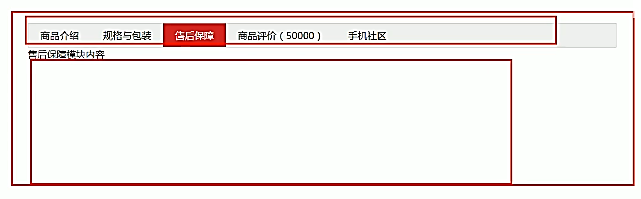
（2）. 移除类：$(“div”).removeClass(''current'');

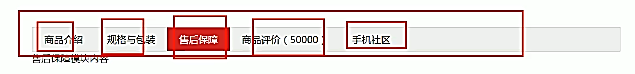
（3）. 切换类：$(“div”).toggleClass(''current'');

|  |
| --- |
| $(function() {              // 1. 添加类 addClass()              // $("div").click(function() {              //     // $(this).addClass("current");              // });              // 2. 删除类 removeClass()              // $("div").click(function() {              //     $(this).removeClass("current");              // });              // 3. 切换类 toggleClass()              $("div").click(function() {                  $(this).toggleClass("current");              });          }) |

【案例】tab栏切换

【案例分析】





① 点击上部的li，**当前li 添加current类，其余兄弟移除类。**

② 点击的同时，得到当前li 的索引号

③ **让下部里面相应索引号的item显示，其余的item隐藏**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <meta http-equiv="X-UA-Compatible" content="ie=edge">      <title>Document</title>      <style>          \* {margin: 0; padding: 0;}          li {list-style-type: none;}          .tab {width: 978px; margin: 100px auto;}          .tab\_list {              height: 39px;              border: 1px solid #ccc;              background-color: #f1f1f1;          }          .tab\_list li {              float: left;              height: 39px;              line-height: 39px;              padding: 0 20px;              text-align: center;              cursor: pointer;          }          .tab\_list .current {background-color: #c81623; color: #fff;}          .item\_info {padding: 20px 0 0 20px;}          .item {display: none;}      </style>      <script src="jquery.min.js"></script>  </head>  <body>      <div class="tab">          <div class="tab\_list">              <ul>                  <li class="current">상품 소개</li>                  <li>상세 페이지</li>                  <li>애프터(서비스)</li>                  <li>상품 평가（50000）</li>                  <li>휴대폰 </li>              </ul>          </div>          <div class="tab\_con">              <div class="item" style="display: block;">상품 소개</div>              <div class="item">상세 페이지</div>              <div class="item">애프터(서비스)  </div>              <div class="item">상품 평가（50000） </div>              <div class="item"> 휴대폰  </div>          </div>      </div>      <script>          $(function() {              // 1.Click on the top li, current li to add a curient class, and the rest of the brother removal class.              $(".tab\_list li").click(function() {                  // chain programming operation                  $(this).addClass("current").siblings().removeClass("current");                  //2.Get the index number of the current li while clicking                  var index = $(this).index();                  console.log(index);                  //3.Allow the iTem to display the corresponding index number in the lower part, and the rest of the iT(让下部里面相应索引号的item显示，其余的item隐藏 )                  $(".tab\_con .item").eq(index).show().siblings().hide();              });          })      </script>  </body>  </html> |

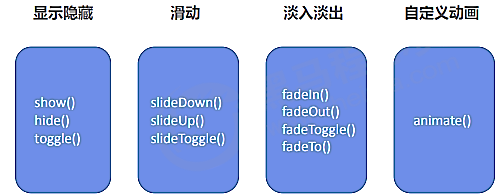
**2.3 类操作与className区别**

**原生 JS 中 className 会覆盖元素原先里面的类名。**

**jQuery 里面类操作只是对指定类进行操作**，不影响原先的类名。相当于追加类名

**3. jQuery 效果**

jQuery 给我们封装了很多动画效果，最为常见的如下：



**3.1 显示隐藏效果**

**1. 显示语法规范**

**show([speed,[easing],[fn]])**

**2. 显示参数**

（1）参数都可以省略， 无动画直接显示。

（2）**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)。

（3）**easing**：(Optional) 用来指定切换效果，默认是“**swing**”，可用参数“**linear**”。

（4）**fn**: **回调函数**，在动画完成时执行的函数，每个元素执行一次。

**3. 隐藏语法规范**

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

**4. 隐藏参数**

（1）参数都可以省略， 无动画直接显示。

（2）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)。

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

（4）fn: 回调函数，在动画完成时执行的函数，每个元素执行一次

**4. 切换语法规范**

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

**5. 切换参数**

（1）参数都可以省略， 无动画直接显示。

（2）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)。

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

（4）fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

建议：平时一般不带参数，直接显示隐藏即可。

|  |
| --- |
| $(function() {              $("button").eq(0).click(function() {                  $("div").show(1000, function() {                      alert(1);                  });              })              $("button").eq(1).click(function() {                  $("div").hide(1000, function() {                      alert(1);                  });              })              $("button").eq(2).click(function() {                      $("div").toggle(1000);                  })                  // 一般情况下，我们都不加参数直接显示隐藏就可以了 |

        });

**3.2 滑动效果**

**1. 下滑效果语法规范**

**slideDown([speed,[easing],[fn]])**

**2. 下滑效果参数**

（1）参数都可以省略。

（2）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)。

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

（4）fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

**3. 上滑效果语法规范**

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

**4. 上滑效果参数**

（1）参数都可以省略。

（2）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)。

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

（4）fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

**5. 滑动切换效果语法规范**

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

**6. 滑动切换效果参数**

（1）参数都可以省略。

（2）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)。

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

（4）fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

|  |
| --- |
| <button>下拉滑动</button>      <button>上拉滑动</button>      <button>切换滑动</button>      <div></div>      <script>          $(function() {              $("button").eq(0).click(function() {                  // 下滑动 slideDown()                  $("div").slideDown();              })              $("button").eq(1).click(function() {                  // 上滑动 slideUp()                  $("div").slideUp(500);              })              $("button").eq(2).click(function() {                  // 滑动切换 slideToggle()                  $("div").slideToggle(500);              });          });      </script> |

**3.3 事件切换**

**hover([over,]out)**

（1）over:鼠标移到元素上要触发的函数（相当于mouseenter）

（2）out:鼠标移出元素要触发的函数（相当于mouseleave）

（3）如果只写一个函数，则鼠标经过和离开都会触发它 **slideToggle();**

|  |
| --- |
| $(function() {              // 鼠标经过              // $(".nav>li").mouseover(function() {              //     // $(this) jQuery 当前元素  this不要加引号              //     // show() 显示元素  hide() 隐藏元素              //     $(this).children("ul").slideDown(200);              // });              // // 鼠标离开              // $(".nav>li").mouseout(function() {              //     $(this).children("ul").slideUp(200);              // });              // 1. 事件切换 hover 就是鼠标经过和离开的复合写法              // $(".nav>li").hover(function() {              //     $(this).children("ul").slideDown(200);              // }, function() {              //     $(this).children("ul").slideUp(200);              // });              // 2. 事件切换 hover  如果只写一个函数，那么鼠标经过和鼠标离开都会触发这个函数              $(".nav>li").hover(function() {                  $(this).children("ul").slideToggle();              });          })      </script> |

**3.4 动画队列及其停止排队方法**

**1. 动画或效果队列**

动画或者效果一旦触发就会执行，**如果多次触发，就造成多个动画或者效果排队执行。**

**2. 停止排队 stop()**

(1）stop() 方法用于停止动画或效果。

(2) 注意： **stop() 写到动画或者效果的前面**， 相当于停止结束上一次的动画

|  |
| --- |
| $(".nav>li").hover(function() {                  // stop 方法必须写到动画的前面                  $(this).children("ul").stop().slideToggle();              }); |

**3.5 淡入淡出效果**

**1. 淡入效果语法规范**

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

**2. 淡入效果参数**

（1）参数都可以省略。

（2）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)。

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

（4）fn: 回调函数，在动画完成时执行的函数，每个元素执行一次

**3. 淡出效果语法规范**

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

**4. 淡出效果参数**

（1）参数都可以省略。

（2）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)。

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

（4）fn: 回调函数，在动画完成时执行的函数，每个元素执行一次

**5. 淡入淡出切换效果语法规范**

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

**6. 淡入淡出切换效果参数**

（1）参数都可以省略。

（2）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)。

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

（4）fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

**7. 渐进方式调整到指定的不透明度**

fadeTo([[speed],opacity,[easing],[fn]])

8. 效果参数

（1）opacity 透明度必须写，取值 0~1 之间。

（2）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)。必须写

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

（4）fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

|  |
| --- |
| <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <meta http-equiv="X-UA-Compatible" content="ie=edge">      <title>Document</title>      <style>          div {              width: 150px;              height: 300px;              background-color: pink;              display: none;          }      </style>      <script src="jquery.min.js"></script>  </head>  <body>      <button>淡入效果</button>      <button>淡出效果</button>      <button>淡入淡出切换</button>      <button>修改透明度</button>      <div></div>      <script>          $(function() {              $("button").eq(0).click(function() {                  // 淡入 fadeIn()                  $("div").fadeIn(1000);              })              $("button").eq(1).click(function() {                  // 淡出 fadeOut()                  $("div").fadeOut(1000);              })              $("button").eq(2).click(function() {                  // 淡入淡出切换 fadeToggle()                  $("div").fadeToggle(1000);              });              $("button").eq(3).click(function() {                  //  修改透明度 fadeTo() 这个速度和透明度要必须写                  $("div").fadeTo(1000, 0.5);              });          });      </script>  </body> |

【案例】图片突出显示



**3.6 自定义动画 animate**

**1. 语法**

animate(params,[speed],[easing],[fn])

**2. 参数**

（1）params: 想要更改的样式属性，以对象形式传递，必须写。 属性名可以不用带引号， 如果是复合属性则需要采 取驼峰命名法 borderLeft。其余参数都可以省略。

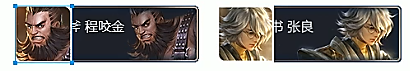
（2）speed：三种预定速度之一的字符串(“slow”,“normal”, or “fast”)或表示动画时长的毫秒数值(如：1000)。

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

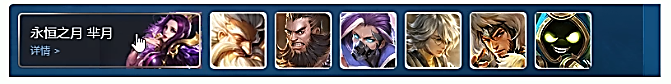
（4）fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

|  |
| --- |
| <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <meta http-equiv="X-UA-Compatible" content="ie=edge">      <title>Document</title>      <script src="jquery.min.js"></script>      <style>          div {              position: absolute; // 有了定位才能动起来              width: 200px;              height: 200px;              background-color: pink;          }      </style>  </head>  <body>      <button>动起来</button>      <div></div>      <script>          $(function() {              $("button").click(function() {                  $("div").animate({                      left: 500,                      top: 300,                      opacity: .4,                      width: 500                  }, 500);              })          })      </script>  </body> |

【案例】王者荣耀手风琴效果



小图片定位 到大图片上边，大图片隐藏起来先，第一个当前默认大图片



【案例分析】

① 鼠标经过某个小li 有两步操作：

② **当前小li 宽度变为 224px**， **同时里面的小图片淡出，大图片淡入**

③ **其余兄弟小li宽度变为69px，** 小图片淡入， 大图片淡出

|  |
| --- |
| <!doctype html>  <html>  <head>      <meta charset="utf-8">      <title>手风琴案例</title>      <style type="text/css">          \* {              margin: 0;              padding: 0;          }            img {              display: block;          }            ul {              list-style: none;          }            .king {              width: 852px;              margin: 100px auto;              background: url(images/bg.png) no-repeat;              overflow: hidden;              padding: 10px;          }            .king ul {              overflow: hidden;          }            .king li {              position: relative;              float: left;              width: 69px;              height: 69px;              margin-right: 10px;          }            .king li.current {              width: 224px;          }            .king li.current .big {              display: block;          }            .king li.current .small {              display: none;          }            .big {              width: 224px;              display: none;          }            .small {              position: absolute;              top: 0;              left: 0;              width: 69px;              height: 69px;              border-radius: 5px;          }      </style>  </head>  <body>      <script src="js/jquery.min.js"></script>      <script type="text/javascript">          $(function() {              // 鼠标经过某个小li 有两步操作：              $(".king li").mouseenter(function() {                  // 1.当前小li 宽度变为 224px， 同时里面的小图片淡出，大图片淡入                  $(this).stop().animate({                      width: 224                  }).find(".small").stop().fadeOut().siblings(".big").stop().fadeIn();                  // 2.其余兄弟小li宽度变为69px， 小图片淡入， 大图片淡出                  $(this).siblings("li").stop().animate({                      width: 69                  }).find(".small").stop().fadeIn().siblings(".big").stop().fadeOut();              })          });      </script>      <div class="king">          <ul>              <li class="current">                  <a href="#">                      <img src="images/m1.jpg" alt="" class="small">                      <img src="images/m.png" alt="" class="big">                  </a>              </li>              <li>                  <a href="#">                      <img src="images/l1.jpg" alt="" class="small">                      <img src="images/l.png" alt="" class="big">                  </a>              </li>              <li>                  <a href="#">                      <img src="images/c1.jpg" alt="" class="small">                      <img src="images/c.png" alt="" class="big">                  </a>              </li>              <li>                  <a href="#">                      <img src="images/w1.jpg" alt="" class="small">                      <img src="images/w.png" alt="" class="big">                  </a>              </li>              <li>                  <a href="#">                      <img src="images/z1.jpg" alt="" class="small">                      <img src="images/z.png" alt="" class="big">                  </a>              </li>              <li>                  <a href="#">                      <img src="images/h1.jpg" alt="" class="small">                      <img src="images/h.png" alt="" class="big">                  </a>              </li>              <li>                  <a href="#">                      <img src="images/t1.jpg" alt="" class="small">                      <img src="images/t.png" alt="" class="big">                  </a>              </li>          </ul>      </div>  </body>  </html> |

**（下）**

**4. jQuery 属性操作**

**4.1 设置或获取元素固有属性值 prop()**

所谓元素固有属性就是元素本身自带的属性，比如 元素里面的 href ，比如 元素里面的 type。

1. **获取** 属性语法：prop(''属性'')

2. **设置** 属性语法：prop(''属性'', ''属性值'')

|  |
| --- |
| <a href="http://www.itcast.cn" title="都挺好">都挺好</a>      <input type="checkbox" name="" id="" checked>      <div index="1" data-index="2">我是div</div>      <span>123</span>      <script>          $(function() {              //1. element.prop("属性名") 获取元素固有的属性值              console.log($("a").prop("href"));              $("a").prop("title", "我们都挺好");              $("input").change(function() {                  console.log($(this).prop("checked"));              });              // console.log($("div").prop("index"));              // 2. 元素的自定义属性 我们通过 attr()              console.log($("div").attr("index"));              $("div").attr("index", 4);              console.log($("div").attr("data-index"));              // 3. 数据缓存 data() 这个里面的数据是存放在元素的内存里面              $("span").data("uname", "andy");              console.log($("span").data("uname"));              // 这个方法获取data-index h5自定义属性 第一个 不用写data-  而且返回的是数字型              console.log($("div").data("index"));          })      </script> |

**4.2 设置或获取元素自定义属性值 attr()**

用户自己给元素添加的属性，我们称为自定义属性。 比如给 div 添加 index =“1”。

1. 获取属性语法：attr(''属性'') // 类似原生 getAttribute()

2. 设置属性语法：attr(''属性'', ''属性值'') // 类似原生 setAttribute()

改方法也可以获取 H5 自定义属性

**4.3 数据缓存 data()**

data() 方法可以在指定的元素上存取数据，并不会修改 DOM 元素结构。一旦页面刷新，之前存放的数据都将被移除。

1. 附加数据语法：data(''name'',''value'') // 向被选元素附加数据

2. 获取数据语法：date(''name'') // 向被选元素获取数据

同时，还可以读取 HTML5 自定义属性 data-index ，得到的是数字型

【案例】：购物车案例模块-全选

【案例分析】：

① **全选思路**：里面3个小的复选框按钮（j-checkbox）选中状态（checked）跟着全选按钮（checkall）走。 (当我的全选按钮 发生改变change的时候，当前处于啥状态，选中还是没选中)

② 因为**checked** 是复选框的 **固有属性**，此时我们需要利用**prop**()方法获取和设置该属性。

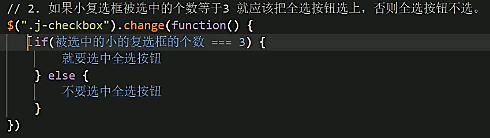
 $(".j-checkbox, .checkall").prop("checked", $(this).prop("checked"));

③ 把 **全选按钮** **状态赋值给 3小** 复选框就可以了。

④ 当我们每次点击小的复选框按钮，就来判断：

⑤ **如果小复选框被选中的个数等于3 就应该把全选按钮选上**，否则全选按钮不选。

**⑥ :checked 选择器 :checked 查找被选中的表单元素**



**5. jQuery 内容文本值**

主要针对元素的**内容**还有**表单的**值操作

**1. 普通元素内容 html()（ 相当于原生inner HTML)**

html() // 获取元素的内容

html(''内容'') // 设置元素的内容

**2. 普通元素文本内容 text() (相当与原生 innerText)**

text() // 获取元素的文本内容

text(''文本内容'') // 设置元素的文本内容

**3. 表单的值 val()（ 相当于原生value）**

val() // 获取表单的值

val(''内容'') // 设置表单的值

|  |
| --- |
| <div><span>我是内容</span></div>      <input type="text" value="请输入内容">      <script>          // 1. 获取设置元素内容 html()          console.log($("div").html());          // $("div").html("123");            // 2. 获取设置元素文本内容 text()          console.log($("div").text());          $("div").text("123");          // 3. 获取设置表单值 val()          console.log($("input").val());          $("input").val("123"); // 修改      </script> |

【案例】购物车案例模块 -**增减商品数量**

【案例分析】

① **核心思路**：首先声明一个 **变量**，当我们点击 **+号（increment**），**就让这个值+**+，然后赋值给文本框。

② 注意1： 只能增加 **本商品的数量**， 就是 当前+号的兄弟文本框（itxt）的值发生变化。

③ 修改表单的值是val() 方法

④ 注意2： 这个变量 **初始值应该是这个文本框的值**，**在这个值的基础上++。**要获取表单的值

⑤ 减号（decrement）思路同理，但是如果文本框的值是1，就不能再减了。

【案例】：购物车案例模块-**修改商品小计**

【案例分析】

① 核心思路：**每次点击+号或者-号，**根据**文本框的值** 乘以 **当前商品的价格** 就是 **商品的小计**

② 注意1： **只能增加本商品的小计**， 就是当前商品的小计模块（p-sum）

③ 修改普通元素的内容是 **text() 方法**

④ 注意2： 当前商品的价格，要把￥符号去掉再相乘 截取字符串 substr(1)

⑤ parents(‘选择器’) 可以返回指定祖先元素

⑥ 最后计算的结果如果**想要保留2位小数 通过 toFixed(2) 方法**

**⑦ 用户也可以直接修改表单里面的值**，同样要计算小计。 用表单change事件

**⑧ 用** **最新的表单内的值** 乘以 **单价** 即可 但是还是当前商品小计

|  |
| --- |
| $(function() {      // 1. All-selected. No-selected module. （全选 全不选功能模块）      // 就是把全选按钮（checkall）的状态赋值给 三个小的按钮（j-checkbox）就可以了      // 事件可以使用change      $(".checkall").change(function() {          // console.log($(this).prop("checked"));          $(".j-checkbox, .checkall").prop("checked", $(this).prop("checked"));          if ($(this).prop("checked")) {              // 让所有的商品添加 check-cart-item 类名              $(".cart-item").addClass("check-cart-item");          } else {              // check-cart-item 移除              $(".cart-item").removeClass("check-cart-item");          }      });      // 2. 如果小复选框被选中的个数等于3 就应该把全选按钮选上，否则全选按钮不选。      $(".j-checkbox").change(function() {          // if(被选中的小的复选框的个数 === 3) {          //     就要选中全选按钮          // } else {          //     不要选中全选按钮          // }          // console.log($(".j-checkbox:checked").length);          // $(".j-checkbox").length 这个是所有的小复选框的个数          if ($(".j-checkbox:checked").length === $(".j-checkbox").length) {              $(".checkall").prop("checked", true);          } else {              $(".checkall").prop("checked", false);          }          if ($(this).prop("checked")) {              // 让当前的商品添加 check-cart-item 类名              $(this).parents(".cart-item").addClass("check-cart-item");          } else {              // check-cart-item 移除              $(this).parents(".cart-item").removeClass("check-cart-item");          }      });      // 3. 增减商品数量模块 首先声明一个变量，当我们点击+号（increment），就让这个值++，然后赋值给文本框。      $(".increment").click(function() {          // 得到当前兄弟文本框的值          var n = $(this).siblings(".itxt").val();          // console.log(n);          n++;          $(this).siblings(".itxt").val(n);          // 3. 计算小计模块 根据文本框的值 乘以 当前商品的价格  就是 商品的小计          // 当前商品的价格 p          var p = $(this).parents(".p-num").siblings(".p-price").html();          // console.log(p);          p = p.substr(1);          console.log(p);          var price = (p \* n).toFixed(2);          // 小计模块          // toFixed(2) 可以让我们保留2位小数          $(this).parents(".p-num").siblings(".p-sum").html("￥" + price);          getSum();      });      $(".decrement").click(function() {          // 得到当前兄弟文本框的值          var n = $(this).siblings(".itxt").val();          if (n == 1) {              return false;          }          // console.log(n);          n--;          $(this).siblings(".itxt").val(n);          // var p = $(this).parent().parent().siblings(".p-price").html();          // parents(".p-num") 返回指定的祖先元素          var p = $(this).parents(".p-num").siblings(".p-price").html();          // console.log(p);          p = p.substr(1);          console.log(p);          // 小计模块          $(this).parents(".p-num").siblings(".p-sum").html("￥" + (p \* n).toFixed(2));          getSum();  });      //  4. 用户修改文本框的值 计算 小计模块  【bug 解决】      $(".itxt").change(function() {          // 先得到文本框的里面的值 乘以 当前商品的单价          var n = $(this).val();          // 当前商品的单价          var p = $(this).parents(".p-num").siblings(".p-price").html();          // console.log(p);          p = p.substr(1);          $(this).parents(".p-num").siblings(".p-sum").html("￥" + (p \* n).toFixed(2));          getSum();  });      // 5. 计算总计和总额模块      getSum();      function getSum() {          var count = 0; // 计算总件数          var money = 0; // 计算总价钱          $(".itxt").each(function(i, ele) {              count += parseInt($(ele).val());          });          $(".amount-sum em").text(count);          $(".p-sum").each(function(i, ele) {              money += parseFloat($(ele).text().substr(1));          });          $(".price-sum em").text("￥" + money.toFixed(2));      }      // 6. 删除商品模块      // (1) 商品后面的删除按钮      $(".p-action a").click(function() {          // 删除的是当前的商品          $(this).parents(".cart-item").remove();          getSum();      });      // (2) 删除选中的商品      $(".remove-batch").click(function() {          // 删除的是小的复选框选中的商品          $(".j-checkbox:checked").parents(".cart-item").remove();          getSum();      });      // (3) 清空购物车 删除全部商品      $(".clear-all").click(function() {          $(".cart-item").remove();          getSum();      })  }) |

**6. jQuery 元素操作**

主要是遍历、创建、添加、删除元素操作。

**6.1 遍历元素**

**jQuery** **隐式迭代** **是对同一类元素做了同样的操作**。

如果想要给 **同一类元素做不同操作，就需要用到遍历**

**【语法**1】：$("div").**each**(function (index, domEle) { xxx; }）

1. each() 方法遍历匹配的每一个元素。主要用DOM处理。 each 每一个

2. 里面的回调函数有2个参数： index 是每个元素的索引号 = i;

demEle 是每个DOM元素对象，不是jquery对象

3**. 所以要想使用jquery方法**，需要给这个**dom元素转换为jquery对象** $(domEle)

**【语法2】：**$.each(object，function (index, element) { xxx; }）

1. $.each()方法可用于遍历 **任何对象**。主要用于 **数据处理**，比如数组，对象

2. 里面的函数有2个参数： index 是每个元素的索引号; element 遍历内容

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <meta http-equiv="X-UA-Compatible" content="ie=edge">      <title>Document</title>      <style>      </style>      <script src="jquery.min.js"></script>  </head>  <body>      <div>1</div>      <div>2</div>      <div>3</div>      <script>          $(function() {              // $("div").css("color", "red");              // 如果针对于同一类元素做不同操作，需要用到遍历元素（类似for，但是比for强大）              var sum = 0;              // 1. each() 方法遍历元素              var arr = ["red", "green", "blue"];              $("div").each(function(i, domEle) {                  // 回调函数第一个参数一定是索引号  可以自己指定索引号号名称                  // console.log(index);                  // console.log(i);                  // 回调函数第二个参数一定是 dom元素对象 也是自己命名                  // console.log(domEle);                  // domEle.css("color"); dom对象没有css方法                  $(domEle).css("color", arr[i]);                  sum += parseInt($(domEle).text());              })              console.log(sum);              // 2. $.each() 方法遍历元素 主要用于遍历数据，处理数据              // $.each($("div"), function(i, ele) {              //     console.log(i);              //     console.log(ele);              // });              // $.each(arr, function(i, ele) {              //     console.log(i);              //     console.log(ele);              // })              $.each({                  name: "andy",                  age: 18              }, function(i, ele) {                  console.log(i); // 输出的是 name age 属性名                  console.log(ele); // 输出的是 andy  18 属性值              })          })      </script>  </body>  </html> |

【案例】：购物车案例模块-**计算总计和总额**

【案例分析】

① 核心思路：把 **所有文本框里面的值相加就是**总计数量。总额同理

② **文本框里面的值不相同**，如果想要 相加需要用到 **each**遍历。声明一个变量，相加即可

③ **点击+号-号，会改变总计和总额**，如果用户修改了文本框里面的值同样会改变总计和总额

④ **因此可以 封装一个函数求总计和总额的function getSsum(){}**， 以上2个操作调用这个函数即可。

⑤ 注意1： 总计是文本框里面的值相加用 val() 总额是普通元素的内容用text()

⑥ 要注意普通元素里面的 **内容要去掉￥**并且**转换为数字型**才能相加

**6.2 创建元素**

语法：$(''<li></li>''); 动态的创建了一个 <li>

**6.3 添加元素**

1. 内部添加 **element.append(''内容'')** 把内容放入匹配元素内部最后面，类似原生 appendChild

**element.prepend(''内容'')** 把内容放入匹配元素内部最前面。

2. 外部添加

**element.after(''内容'') // 把内容放入目标元素后面**

**element.before(''内容'') // 把内容放入目标元素前面**

① 内部添加元素，生成之后，它们是父子关系。

② 外部添加元素，生成之后，他们是兄弟关系。

**6.4 删除元素**

**element.remove()** // 删除匹配的元素（本身）

**element.empty() //** 删除匹配的元素集合中所有的子节点

**element.html('''')** // 清空匹配的元素内容

① remove 删除元素本身。

② empt() 和 html('''') 作用等价，都可以删除元素里面的内容，只不过 html 还可以设置内容。

|  |
| --- |
| <ul> <li>原先的li</li> </ul>      <div class="test">我是原先的div</div>      <script>          $(function() {              // 1. 创建元素              var li = $("<li>我是后来创建的li</li>");              // 2. 添加元素              // (1) 内部添加              // $("ul").append(li);  内部添加并且放到内容的最后面              $("ul").prepend(li); // 内部添加并且放到内容的最前面              // (2) 外部添加              var div = $("<div>我是后妈生的</div>");              // $(".test").after(div);              $(".test").before(div);              // 3. 删除元素              // $("ul").remove(); 可以删除匹配的元素 自杀              // $("ul").empty(); // 可以删除匹配的元素里面的子节点 孩子              $("ul").html(""); // 可以删除匹配的元素里面的子节点 孩子          })      </script> |

【案例】：购物车案例模块-**删除商品模块**

① 核心思路：把商品**remove**() 删除元素即可

② **有三个地方需要删除**： 1. 商品后面的删除按钮 2. 删除选中的商品 3. 清理购物车

③ 商品后面的删除按钮： 一定是删除当前的商品，所以从 **$(this)** 出发

④ 删除选中的商品： 先判断小的复选框按钮是否选中状态，**如果是选中，则删除对应的商品**

⑤ 清理购物车： 则是把所有的商品全部删掉

【案例】案例：购物车案例模块-**选中商品添加背景**

① 核心思路：选中的商品添加背景，不选中移除背景即可

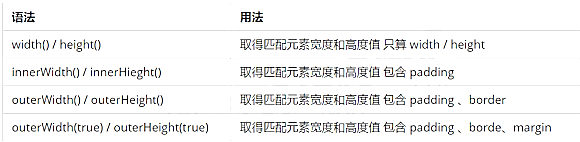
② **全选按钮点击**：如果全选是选中的，则所有的商品添加背景，否则移除背景

③ **小的复选框点击**： 如果是选中状态，则当前商品添加背景，否则移除背景

④ 这个背景，可以通过类名修改，添加类和删除类

**7. jQuery 尺寸、位置操作**

**7.1 jQuery 尺寸**



以上参数为空，则是获取相应值，返回的是数字型。

如果参数为数字，则是修改相应值。

参数可以不必写单位。

**7.2 jQuery 位置**

位置主要有三个： offset()、position()、scrollTop()/scrollLeft()

**1. offset() 设置或获取元素偏移**

① offset() 方法设置或返回被选元素相对于文档的偏移坐标，跟父级没有关系。

② 该方法有2个属性 left、top 。offset().top 用于获取距离文档顶部的距离，offset().left 用于获取距离文档左侧的距离。

③ 可以设置元素的偏移：offset({ top: 10, left: 30 })

2. position() 获取元素偏移

① position() 方法用于返回被选元素相对于带有定位的父级偏移坐标，如果父级都没有定位，则以文档为准。

② 该方法有2个属性 left、top。position().top 用于获取距离定位父级顶部的距离，position().left 用于获取距离定 位父级左侧的距离。

③ 该方法只能获取

**3. scrollTop()/scrollLeft() 设置或获取元素被卷去的头部和左侧**

① scrollTop() 方法设置或返回被选元素被卷去的头部。

② 不跟参数是获取，参数为不带单位的数字则是设置被卷去的头部。

【案例】：带有动画的返回顶部

① 核心原理： 使用animate动画返回顶部。

② animate动画函数里面有个scrollTop 属性，可以设置位置

③ 但是是元素做动画，因此 $(“body,html”).animate({scrollTop: 0})

【案例】： 品优购电梯导航

① 当我们滚动到 今日推荐 模块，就让电梯导航显示出来

② 点击电梯导航页面可以滚动到相应内容区域

③ 核心算法：因为电梯导航模块和内容区模块一一对应的

④ 当我们点击电梯导航某个小模块，就可以拿到当前小模块的索引号

⑤ 就可以把animate要移动的距离求出来：当前索引号内容区模块它的offset().top

⑥ 然后执行动画即可

① 当我们点击电梯导航某个小li， 当前小li 添加current类，兄弟移除类名

② 当我们页面滚动到内容区域某个模块， 左侧电梯导航，相对应的小li模块，也会添加current类， 兄弟移除current类。

③ 触发的事件是页面滚动，因此这个功能要写到页面滚动事件里面。

④ 需要用到each，遍历内容区域大模块。 each里面能拿到内容区域每一个模块元素和索引号

⑤ 判断的条件： 被卷去的头部 大于等于 内容区域里面每个模块的offset().top

⑥ 就利用这个索引号找到相应的电梯导航小li添加类。