提前准备:

1. 今天的历届师兄师姐问过的经典的高频的问题清单都已经整理在小程序->首页->JQUERY->day03里

2. 今天的课程中所需的单词列表都已经整理在小程序->单词-> JQUERY ->day03里

3. 今天的重点视频、扩展高频鄙视题视频已经整理在小程序->在线-> JQUERY ->day03里

如果不知道问什么: 谁问的越多！谁培训费花的越值！

1. 笔记中或画图中那句话或者那个位置不明白，立刻问！

2. 案例中哪行代码或注释不明白，立刻问！

3. 课下自己做项目，功能不会做，立刻问！

4. 刷笔试题时遇到不会的笔试题，立刻问！——公众号: 前端大全 定期总结大厂笔试题

5. 看小程序问题清单时，遇到看不懂的问题，立刻问！

每个案例三遍:

1. 连注释+代码抄一遍

2. 保留注释，删掉代码，自己试着填回来

3. 删除注释和代码，自己使用用人话把注释写回来！——才算自己会！

正课:

1. 事件

2. 动画

3. 类数组对象操作

一. 事件:

1. DOM中最灵活的事件绑定方式:

(1). 绑定事件: 元素.addEventListener("事件名", 处理函数)  
 (2). 解绑: removeEventListener("事件名", 原处理函数对象)

2. jq中:

(1). 标准写法: 用on代替addEventListener，用off代替了removeEventListener

(2). 简写: 如果事件名属于常用的事件列表中，可简写为：

$元素.事件名(处理函数)

|  |
| --- |
| 可使用简写的常用事件列表:  blur 失去焦点  change 下拉列表选中项改变  click 单击  dblclick 双击  focus 获得焦点  keydown 键盘按键按下  keyup 键盘按键抬起  mousedown 鼠标按键按下  mouseenter 鼠标进入(jq)  mouseleave 鼠标移出(jq)  mousemove 鼠标移动  mouseout 鼠标移出(dom)  mouseover 鼠标进入(dom)  mouseup 鼠标按键抬起  resize 窗口大小改变  scroll 网页滚动 |

3. 差异: (只记现象，不用深究)

(1). DOM中: 如果绑定事件处理函数时，使用有名称的函数，则该函数只能绑定一次！

(2). Jq中: 即使绑定事件处理函数时，使用有名称的函数，也可以重复绑定多个！移除时，也可以都移除！

4. 示例: 绑定发射子弹，发射跟踪导弹的事件

|  |
| --- |
| <h1>事件绑定</h1>  <button id="btn1">发射子弹</button>  <button id="btn2">获得奖励</button>  <button id="btn3">失去奖励</button>  <script src="js/jquery-1.11.3.js"></script>  <script>  //为btn1绑定发射普通子弹  $("#btn1").click(function(){    console.log(`发射普通子弹...`)  })  function shoot2(){    alert("发射跟踪导弹...")  }  //为btn2绑定事件，其中为btn1多添加一种发射跟踪导弹的处理函数  $("#btn2").click(function(){    $("#btn1").on("click",shoot2)  })  //为btn3绑定事件，其中从btn1上移除发射跟踪导弹的处理函数  $("#btn3").click(function(){    $("#btn1").off("click",shoot2)  })  </script>  </body> |

2. 利用冒泡/事件委托:

(1). DOM中:

a. 何时: 一个父元素下多个子元素都要绑定相同的事件时

b. 如何: 3步

1). 事件绑定在父元素上

2). 用e.target代替this获得当前触发事件的目标元素

3). 判断目标元素是否符合条件

c. 固定套路：

$父元素.click(function(e){

var $tar=$(e.target);

if($tar符合xxx条件){

... ...

}

})

(2). Jq中提供了简化:

a. 相同点: 事件依然要绑定在父元素上

b. 不同点1: this又回来了！this又重新指向了实际触发事件的目标元素，可以不用e.target了！——仅jq可用，别的框架不能用！

c. 不同点2: 不用自己写if判断了，而是在on()中添加第二个选择器参数作为判断条件。On()会用选择器自动判断当前点击的目标元素是否符合选择器要求！只有符合选择器要求的元素，才能出发事件执行操作。

d. 固定套路:

$父元素.on("事件名", "选择器" ,function(){

var $tar=$(this)

//直接写后续操作

})

强调: "选择器"，必须是相对于当前父元素下的相对选择器，不能写的太完整。

比如: <div class="container"> //事件绑定在父元素上

<div class="block">

<span class="close">//这里是目标元素

想只允许class为close的元素触发事件:

~~$(".container").on("click",".container>.block>.close",function(){ ... }) //错误！~~

$(".container").on("click",".close",function(){ ... }) //正确

//.block>.close 也对

(3). 示例: 使用事件委托简化删除方块的例子

|  |
| --- |
| <head>  <title> new document </title>  <meta charset="utf-8">  <style>    .container {      border: 1px solid #aaa;  overflow:hidden;    }    .block {      float: left; margin: 10px;      border: 1px solid #aaa;      background: #faa;      width: 150px; height: 150px;    }    .block:hover{ box-shadow:0 5px 6px #000; }    .close {      float:right; padding:5px;      font-weight:bold; opacity:.2; cursor: pointer;    }    .close:hover { opacity: .5; }  </style>  </head>  <body>  <h1>添加/删除节点</h1>  <button id="add-block">添加区块</button>  <div class="container">    <!-- <div class="block">      <span class="close">×</span>    </div> -->  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中: 查找id为add-block的按钮  $("#add-block")  //2. 绑定事件处理函数  .click(function(){    //3. 查找要修改的元素    //本例中: 查找class为container的div    var $div=$(".container")    //4. 修改元素    //先创建一个新块div    var $block=$(`<div class="block">      <span class="close">×</span>    </div>`);    //将新div追加到$div中    //$block.appendTo($div) //return $block    $block.prependTo($div)          .css("background-color",`rgb(${            parseInt(Math.random()\*256)          },${            parseInt(Math.random()\*256)          },${            parseInt(Math.random()\*256)          })`);  })  //点x删除方块:  //DOM 4步  //1. 查找触发事件的元素  //本例中: 多个×都要点击，所以应该用事件委托优化！事件应该只绑定在父元素上一份即可  $(".container")  //2. 绑定事件处理函数  //.click(function(e){  //只允许class为close的元素触发事件执行操作  .on("click",".close",function(){    $(this) //this->重新指向了目标元素，且只要能进来执行操作的，说明已经是被on的.close选择器过滤过的符合条件的元素了。    //3. 查找要修改的元素    //本例中点×，要删除其父元素      .parent()    //4. 修改元素      .remove();  })  </script>  </body> |

3. 页面加载后自动执行:

(1). 问题: 页面中的很多数据都需要在页面一加载完成，就自动发送ajax请求，填充到页面。无需用户手动操作！有些元素的事件处理函数，也是在页面加载完成后，自动绑定上的！用户可直接使用元素执行操作！但是，这些初始化页面数据或事件绑定的代码应该下载哪儿？

(2). 解决: 由于页面是顺序加载执行，所以希望页面加载后自动执行的所有代码都放在body结尾的<script>中

(3). 问题: 将js代码写在HTML文件中，不符合内容与行为分离的原则，不便于维护。

(4). 解决: 今后js代码都要放在独立的js文件中，再引入到html文件里

(5). 问题: 今后别人在引入我们写的js文件时，有时会放在开头，有时会放在结尾，没准。如何保证无论别人把我们的js引入再开头还是结尾时，我们的js都能在页面加载后自动执行呢?

(6). 解决: 把页面初始化代码放在页面的加载完成事件中！等到页面加载完成后才执行触发执行！

a. 每个网页都有一个window.onload=function(){ ... }事件。

b. 无论在哪里绑定这个事件，这个事件始终只在整个页面内容(HTML+CSS+JS+图片)加载完才触发！

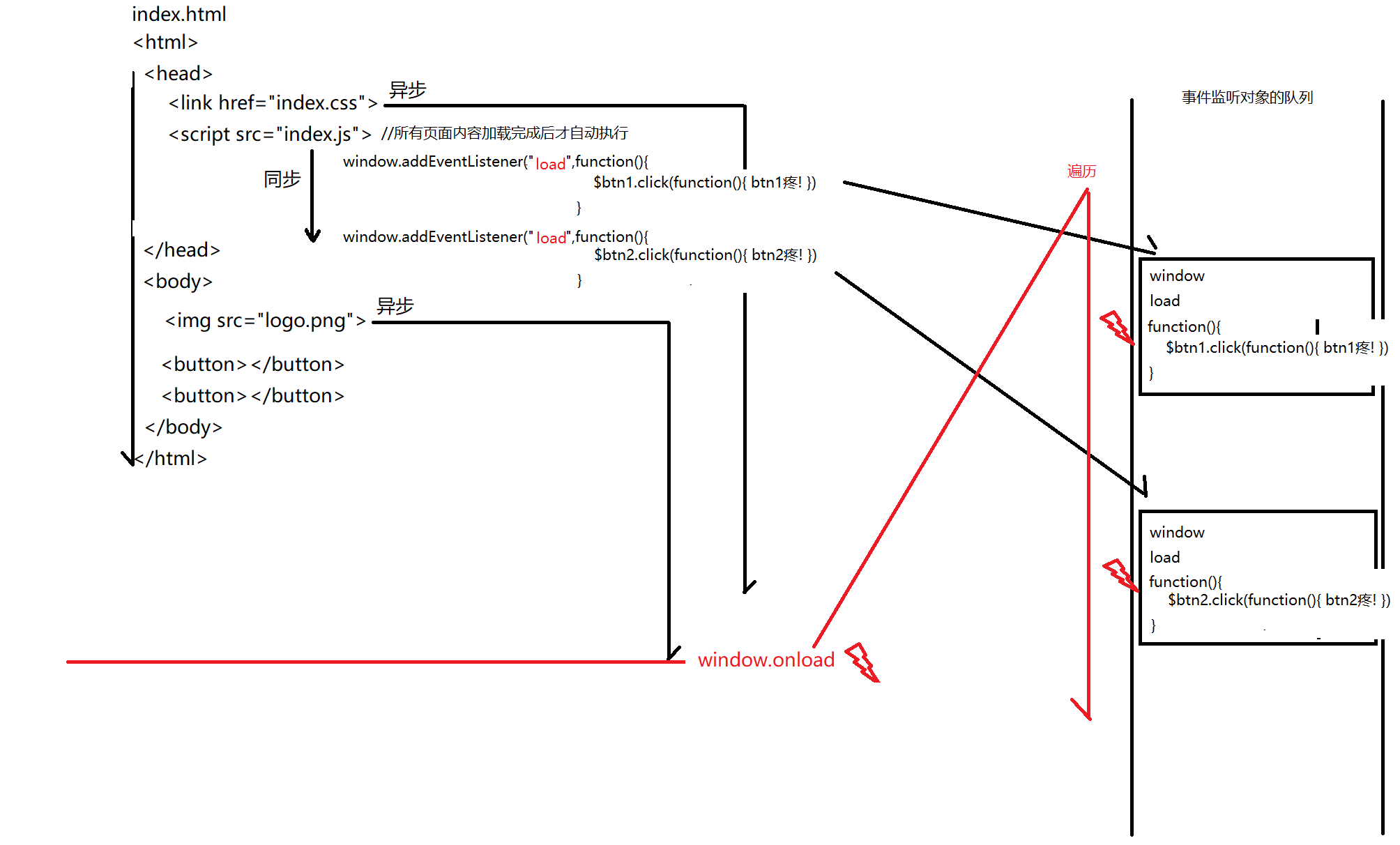
c. 但是，因为js代码会影响页面加载速度，所以最好还是放在body结尾！

(7). 问题: window.onload=function(){ ... }是以赋值方式绑定的事件，而将来一个网页加载的js文件可能有多个！那么，如果多个js文件中都有window.onload=function(){ ... }，结果

(8). 解决: 为了给一个onload同时添加多个事件处理函数，应该将赋值方式，替换为添加事件监听的方式

window.addEventListener("load",function(){ ... })

jq中可简写为: $(window).load(function(){ ... })



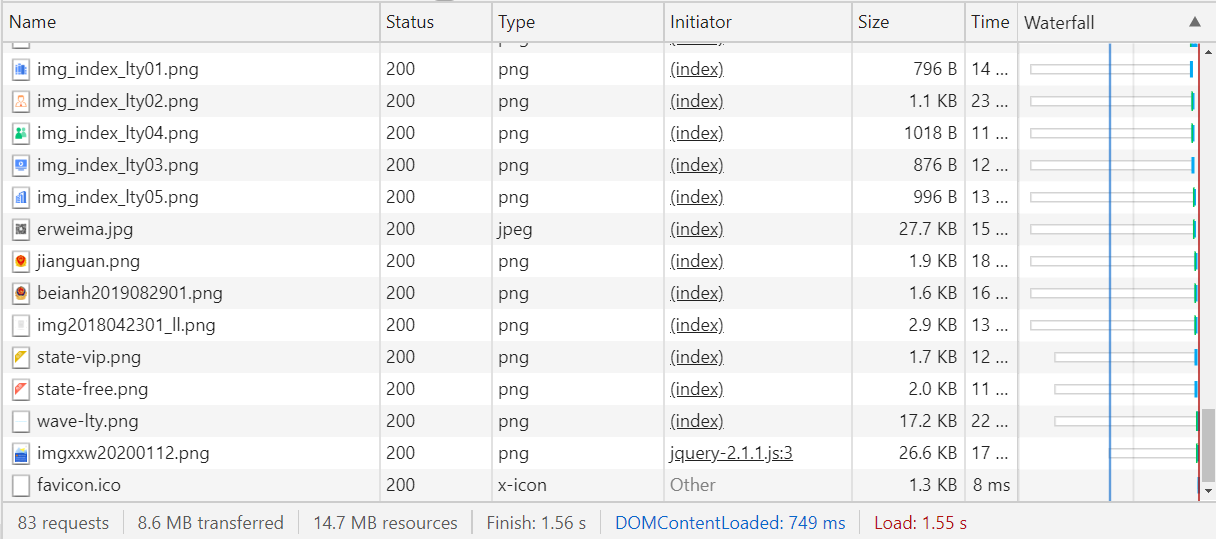
(9). 问题: 有时用户急于操作，不需要等待样式和图片加载完成——比如春运抢火车票！应该让用户在样式和图片加载完成之前就能提前用上操作，而不必等待css和图片。

(10). 解决:

a. 其实网页加载过程中，有两次页面加载完成事件:

1). 先:DOMContentLoaded : 仅HTML+JS下载完成就提前触发

2). 再:window.onload: 必须等HTML+CSS+JS+图片下载生成才触发



b. 只要将需要提前执行的操作，放在DOMContentLoaded中，就可以不必等待css和图片，就能让用户提前用上！

(11). 问题: DOMContentLoaded有兼容性问题

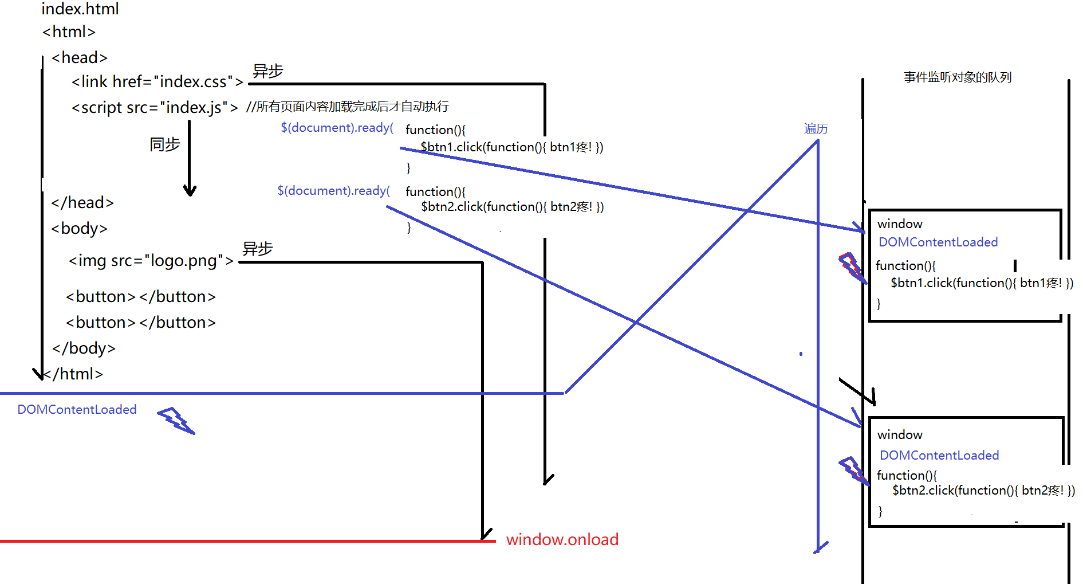
(12). 解决: jq已经封装了DOMContentLoaded事件

a. $(document).ready(function(){ ... }) //会在DOMContentLoaded阶段提前自动执行

b. 简写:

1). $().ready(function(){ ... })

2). $(function(){ ... })

s

(13). 总结:

a. 今后几乎所有jq代码，都要放在$(function(){ ... })中

es6还可简写为$(()=>{ ... })

b. 但是，个别必须css和图片才能正常执行的代码，只能放在window.addEventListener("load",function(){ ... })

jq中: $(window).load(function(){ ... })

(14). 示例: 使用2个独立的js文件为2个按钮绑定单击事件

14\_ready1.js

|  |
| --- |
| //所有页面内容加载完成后才自动执行  //window.onload=function(){  //window.addEventListener("load",function(){  //$(window).load(function(){  //$(document).ready(function(){  $(function(){    alert("仅HTML+JS加载完成就为btn1绑定事件")    $("#btn1").click(function(){ alert(`btn1疼！`) })  }) |

14\_ready2.js

|  |
| --- |
| //所有页面内容加载完成后才自动执行  //window.onload=function(){  //window.addEventListener("load",function(){  //$(window).load(function(){  //$(document).ready(function(){  $(function(){    alert("仅HTML+JS加载完成就为btn2绑定事件")    $("#btn2").click(function(){ alert(`btn2疼！`) })  }) |

14\_ready.html

|  |
| --- |
| <head>  <title> new document </title>  <meta charset="utf-8">  <script src="js/jquery-1.11.3.js"></script>  <script>    //后    window.onload=function(){      alert("所有页面内容加载完成！")    }  </script>  <script src="14\_ready2.js">    //先$(document).ready(function(){      //$btn2.click(btn2疼!)    //})  </script>  <script src="14\_ready1.js">    //先$(document).ready(function(){      //$btn1.click(btn1疼!)    //})  </script>  </head>  <body>  <h1>页面加载完成事件</h1>  <p>welcome to my first web page</p>  <button id="btn1">click me1</button>  <button id="btn2">click me2</button>  </body> |

4. 鼠标事件:

(1). DOM中: mouseover 鼠标进入元素; mouseout 鼠标移出元素时

(2). 问题: 频繁进出子元素，同样会频繁触发父元素的进出事件。与现实不符，产生歧义。

(3). Jq中: 用mouseenter代替了mouseover，用mouseleave代替了mouseout

进入 进入 离开 移出

(4). 优点: 即使反复进出子元素，也不会反复触发父元素上的鼠标进出事件，与现实相符！不会产生歧义。

(5). 简写: 如果同时绑定mouseenter和mouseleave事件，则可用hover简写:

$元素.hover( //=mouseenter+mouseleave

function(){ ... }, //给mouseenter

function(){ ... } //给mouseleave

)

(6). 示例: 使用mouseenter, mouseleave, hover，绑定鼠标事件

|  |
| --- |
| <html>  <head>  <title>事件处理</title>  <meta charset="utf-8"/>  <style>  #d1 #d2 #d3{cursor:pointer}  #d1 {    background-color: green;    position: relative;    width: 150px;    height: 150px;    text-align: center;    cursor: pointer;  }  #d2 {    background-color: blue;    position: absolute;    top: 25px;    left: 25px;    width: 100px;    height: 100px;  }  #d3 {    background-color: red;    position: absolute;    top: 25px;    left: 25px;    width: 50px;    height: 50px;    line-height: 50px;  }  </style>  </head>  <body>  <div id="d1">    <div id="d2">      <div id="d3">      </div>    </div>  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>  var $d1=$("#d1");  $d1.hover(//=mouseenter+mouseleave    function(){ console.log(`进入d1`) },//给mouseenter    function(){ console.log(`移出d1`) } //给mouseleave  )  //当鼠标进入d1时自动执行  // //.mouseover(function(){  // .mouseenter(function(){  //  console.log(`进入d1`)  // })//return $d1  // //当鼠标移出d1时自动执行  // //.mouseout(function(){  // .mouseleave(function(){  //  console.log(`移出d1`)  // })  </script>  </body> |

(7). 更简写: 如果你有本事将鼠标进入时的操作和鼠标移出时的操作改成一样的代码，完全相同的两个函数，其实只要写一个函数即可

$元素.hover(//=mouseenter+mouseleave

function(){ ... } //既给mouseenter，又给mouseleave

)

(8). 示例: 使用hover最简形式

|  |
| --- |
| <html>  <head>  <title> new document </title>  <meta charset="utf-8">  <style>  #target {    border: 1px solid #eee;    border-radius: 6px;    padding: 10px;    transition: all .5s linear;  }  #target.hover {    border: 1px solid #aaa;    box-shadow: 0 0 6px #aaa;    color:#fff;    background-color:red;  }  </style>  </head>  <body>  <h1>使用hover(fn,fn)</h1>  <h3>鼠标悬停在div上方，则突出显示；移出则取消突出显示</h3>  <div id="target">    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eligendi neque quae voluptatum ducimus culpa itaque maxime distinctio soluta cum cupiditate esse tenetur deserunt fuga perferendis sed veritatis asperiores. Numquam officia.</p>  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>  $("#target")  .hover(    function(){ $(this).toggleClass("hover") }    //,    //function(){ $(this).toggleClass("hover") }  )  // //想鼠标进入div时，为div加class hover  // .mouseenter(function(){ $(this).addClass("hover") })  // //想鼠标移出div时，从div上移除class hover  // .mouseleave(function(){ $(this).removeClass("hover") })  </script>  </body>  </html> |

5. 模拟触发:

(1). 什么是: 即使没有点击这个元素，也可以触发元素上之前绑定的事件处理函数

(2). 如何:

a. 标准: $元素.trigger("事件名")

触发

意为: 即使没有点在元素上，也能触发元素上某个事件处理函数

b. 简写: 如果要触发的事件属于常用事件列表中，则可简写为:

$元素.事件名()

其实: 事件处理函数也是一个函数两用:

如果给了function(){ ... }就绑定事件

如果没给function(){ ... }就自动切换为模拟触发该事件

(3). 示例: 按回车和点按钮，都能执行搜索

|  |
| --- |
| <body>  <input type="text"><button>百度一下</button>  <script src="js/jquery-1.11.3.js"></script>  <script>  //当单击按钮时  $("button").click(function(){    //获得文本框的内容并去空格    var kws=$(":text").val().trim();    if(kws!==""){//如果内容不为""      //就去服务器端搜索相应的结果...      console.log(`搜索 ${kws} 相关的内容...`);    }  })  //现实中很少点按钮，都是在文档框上按回车就执行查找  //    键盘按键 按下  // $(":text").keydown(function(e){  //    键盘按键 抬起  $(":text").keyup(function(e){    //如果按的是回车键(按键号==13)时    if(e.keyCode==13){      //找到按钮，触发按钮身上的单击事件处理函数，就可以执行和单击按钮完全相同的效果了      //$("button").trigger("click");      $("button").click();    }  })  </script>  </body> |

二. 动画:

1. 简单动画: 固定写死的三种动画效果: ——了解

(1). 显示隐藏:

a. $元素.show(动画持续时间) 让元素显示出来

b. $元素.hide(动画持续时间) 让元素隐藏

c. $元素.toggle(动画持续时间) 让元素在显示和隐藏之间来回切换

说明: 以上三个函数在不加参数时，默认使用display:none控制显示隐藏，没有动画效果。

如果需要加动画效果，就要提供一个"动画持续时间"实参值

(2). 上滑下滑:

a. $元素.slideUp() 让元素上滑

b. $元素.slideDown() 让元素下滑

c. $元素.slideToggle() 让元素在上滑和下滑之间来回切换

(3). 淡入淡出:

a. $元素.fadeIn() 让元素淡入

b. $元素.fadeOut() 让元素淡出

c. $元素.fadeToggle() 让元素在淡入和淡出之间来回切换

|  |
| --- |
| 以上简单动画函数的问题:  1. 在函数库的源代码中固定写死的，几乎不可维护的  2. 用js的定时器模拟的动画效果，效率远不如css的transition  Cpu Gpu  总结: 如果一个效果即可用css做，又可用js做，应该首选css——效率高，易维护 |

(4). 示例: 使用简单动画函数

|  |
| --- |
| <html>  <head>  <title> new document </title>  <meta charset="utf-8">  <style>    \*{margin:0; padding:0;}    #target{      border-radius:10px;      background:#eee;    }    .fade{/\*动画起始状态\*/      height:104px; width:970px; opacity:1;      padding: 10px; overflow:hidden;      border: 1px solid #aaa;    }    .out{/\*动画结束状态\*/    }  </style>  </head>  <body>  <h1>jQuery动画函数——显示隐藏动画</h1>  <button id="btn1">显示/隐藏div</button>  <div id="target">    <p><span>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Tempore debitis animi sint iste sequi sunt ad excepturi error labore molestiae est expedita eos nisi placeat provident dolorem quos facilis! Sapiente!</span><span>Accusamus neque id reprehenderit! Voluptatem in deleniti laboriosam commodi facere magnam impedit minima corrupti distinctio culpa amet optio natus esse. Inventore incidunt ab id perspiciatis atque minus magnam tempore harum.</span></p>  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>  $("#btn1").click(function(){    //让#target瞬间显示隐藏:    //$("#target").hide();    //让#target过度显示隐藏:    //$("#target").hide(2000);    //$("#target").toggle(2000);    //使用上滑下滑代替显示/隐藏效果    //$("#target").slideToggle()    //使用淡入淡出代替显示/隐藏效果    $("#target").fadeToggle();  });  </script>  </body>  </html> |

(6). 示例: 展开和隐藏品牌列表

|  |
| --- |
| <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>    <style>      body {        text-align: center;      }      ul {        list-style: none;      }      li {        margin: 0;        padding: 0;        display: inline-block;        width: 30%;      }    </style>  </head>  <body>  <ul id="list">    <li>尼康(234)</li>    <li>佳能(22)</li>    <li>索尼(932)</li>    <li>宾得(11)</li>    <li>爱国者(3234)</li>    <li>欧巴(32)</li>    <li>海鸥(2334)</li>    <li>卡西欧(334)</li>    <li>三星(834)</li>    <li>松下(234)</li>    <li>其它品牌(2343)</li>  </ul>  <button data-toggle="brandlist">精简显示品牌</button>  <script src="js/jquery-1.11.3.js"></script>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中查找data-toggle属性为brandlist的按钮  $("[data-toggle=brandlist]")  //2. 绑定事件处理函数  .click(function(){    //3. 查找要修改的元素    //本例中: 查找ul下的li中大于4位置的且不是最后一个元素的所有li    $("ul>li:gt(4):not(:last-child)")    //4. 修改元素    //本例中: 切换显示隐藏    .toggle();    //让当前元素的内容在"精简显示品牌"和"显示所有品牌"之间来回切换    var $btn=$(this);    var text=$btn.html();    $btn.html(text=="显示所有品牌"?"精简显示品牌":"显示所有品牌")  })  </script>  </body>  </html> |

2. "万能"动画:

(1). 什么是: 可对任意一个css属性应用动画效果

(2). 如何:

$元素.animate({

Css属性: 目标值,

... : ...

}, 动画持续时间 )

(3). 原理: 根据当前元素当前的状态和animate中指定的目标值之间的差，自动计算动画效果。如果元素已经在目标值位置，当前状态和animate中指定的目标状态之间无差别，自然就不会有动画效果了。

(4). 说明:

a. 进步: 可以自己指定要变化的css属性，不是写死的

b. 问题: 依然可以在elements中看到变化过程，说明还是用js定时器模拟的。效率依然不如css的transition。

(5). Jq animate vs css transition

a. 相同点: 都支持单个数值的css属性:

比如: top left bottom right padding margin width height ...

b. 不同点:

1). Css transition 还支持 颜色动画 css3变换

2). Jq animate 不支持 颜色动画，也不支持css3变换

(6). 示例: 测试animate支持哪些css属性，不支持哪些

|  |
| --- |
| <html>  <head>  <title> new document </title>  <meta charset="utf-8">  <style>    #d1{      border:1px solid #aaa;      border-radius:6px;      background:#eee;      width:50px; height:50px;      position:absolute; top:120px; left:0;    }  </style>  </head>  <body>  <h1>animate</h1>  <button id="btn1">启动动画</button>  <div id="d1">abcd</div>  <script src="js/jquery-1.11.3.js"></script>  <script>  $("#btn1").click(function(){    $("#d1").animate({      //left:300,//top,bottom,right      //padding:20, //margin      //fontSize:32      //backgroundColor:"red"      //transform:"rotate(360deg)"    },2000)  })  </script>  </body>  </html> |

3. 排队和并发:

(1). 并发: 一个元素的多个css属性同时变化

a. 如何: 放在一个animate中的多个css属性默认是并发变化的

(2). 排队: 一个元素的多个css属性先后依次变化

a. 如何: 对一个元素，反复调用多个animate()函数，多个链式调用的animate()函数中的css属性是排队变化的.

b. 原理:

1). Jq中，每个元素都有一个动画队列

2). 每当我们调用一个动画函数时，其实不是立刻执行动画的意思，只是将这个动画函数加入到元素的动画队列中等待

3). 如果动画队列中现在没有正在执行的动画函数，则新加入队列的动画函数，可以立刻执行

4). 如果动画队列中此时已经有正在执行的动画函数，则新加入队列的动画函数，只能等待队列中之前的动画效果执行完，才能轮到自己执行。

4. 停止动画:

(1). $元素.stop();

(2). 坑: stop()默认只能停止当前队列中正在播放的一个动画。但是队列中后续动画效果，依然执行！

(3). 解决: $元素.stop(true) 停止动画，并清空队列中后续动画效果

5. jq新增的选择器:

:animated 专门匹配正在播放animate动画的元素

6. 动画结束后自动执行:

每个动画函数都有最后一个实参值：一个回调函数

放在动画函数最后一个实参值回调函数中的代码，必须等到动画播放结束后才自动执行.

7. 示例: 点四颗小星星，按要求播放动画

|  |
| --- |
| <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>    <style>      img { position: relative; }    </style>  </head>  <body>  <img id="s1" src="img/star.png"><br/>  <img id="s2" src="img/star.png"><br/>  <img id="s3" src="img/star.png"><br/>  <img id="s4" src="img/star.png"><br/>  <script src="js/jquery-1.11.3.js"></script>  <script>  /\*  s1在屏幕左上角的小星星， 点击后从左移动到屏幕右边  s2在屏幕左上角的小星星，点击后从左移动到屏幕右边，再移动到下边——走直角  s3在屏幕左上角的小星星，点击后从左上角移动到屏幕右下边，走斜线  s4点击小星星，变大、变淡....  直至消失  \*/  $("#s1").click(function(){    var $s=$(this);    //如果正在播放动画，就停止    if($s.is(":animated")){      $s.stop();    }else{//否则如果没有播放动画，就启动      $s.animate({        left:300      },2000)    }  })  $("#s2").click(function(){    var $s=$(this);    //如果正在播放动画，就停止    if($s.is(":animated")){      $s.stop(true);    }else{//否则如果没有播放动画，就启动      $s      .animate({        left:300      },2000)// return $this      .animate({        top:100      },1000)    }  })  $("#s3").click(function(){    var $s=$(this);    //如果正在播放动画，就停止    if($s.is(":animated")){      $s.stop();    }else{//否则如果没有播放动画，就启动      $s.animate({        left:300,        top:100      },3000)    }  })  $("#s4").click(function(){    alert("疼!")    $(this).animate(//异步      {        width:172, height:172, opacity:0      },      2000,      function(){//动画播放后自动执行的回调函数        $(this).hide();      }    )  })  </script>  </body>  </html> |

|  |
| --- |
| var bool=$元素.is("选择器") 判断当前元素是否符合选择器的条件要求  是  $元素.is("span") 判断当前元素是否是一个span元素  $元素.is(".close") 判断当前元素是不是class为close的元素  $元素.is(":disabled") 判断当前元素是否禁用  $元素.is(":checked") 判断当前元素是否被选中  $元素.is(":checkbox") 判断当前元素是否是checkbox复选框  \*忘了的去复习css和jquery选择器 |

三. 类数组对象操作:

|  |
| --- |
| 类数组对象在js中非常受歧视，像数组但是用不了数组家的函数。但是jQuery家所有孩子都是类数组对象，如果不能使用数组家的函数，则功能会受很大影响。于是jQuery就自力更生，仿着jquery家两个常用函数forEach和indexOf，又定义了两个功能一模一样的函数。 |

1. each(): 今后只要希望对查找结果中的每个DOM元素对象执行相同的操作时

(1). Js中: arr.forEach(function(elem,i,arr){ ... }) 遍历一个数组中每个元素，对数组中每个元素自动执行回调函数规定的相同操作.

(2). Jq中: $查找结果.each(function(i, elem){ ... }) 遍历jquery查找结果中的每个DOM元素对象，每遍历一个DOM元素，就对整个DOM元素执行回调函数规定的相同的操作

(3). 示例: 遍历jq查找结果中每个元素

|  |
| --- |
| <body>  <ul id="list">    <li>98</li>    <li>85</li>    <li>33</li>    <li>99</li>    <li>52</li>  </ul>  <script src="js/jquery-1.11.3.js"></script>  <script>  //请给每个不足60分的成绩+10分，并将超过90分的成绩用绿色背景标识出来  $("ul>li").each(function(i, elem){ //elem=当前DOM元素    var $li=$(elem);    if($li.html()<60){      var score=parseInt($li.html())      score+=10;      $li.html(score);    }else if($li.html()>=90){      $li.css("background-color","green")    }  })  </script>  </body> |

2. index(): 今后只要想获得一个DOM元素在查找结果中的位置时

(1). Js中: var i=arr.indexOf(要找的值) 查找指定元素值在数组中出现的位置

(2). Jq中: var i=$查找结果.index(DOM元素) 查找一个DOM元素在jQuery查找结果中是第几个位置。

(3). 示例: 实现5星评价功能:

|  |
| --- |
| <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>    <style>      .score {        list-style: none;        margin: 0;        padding: 0;      }      .score li {        display: inline-block;        width: 50px;        height: 50px;        border: 1px solid #f00;        border-radius: 50%;        cursor: pointer;      }    </style>  </head>  <body>  <h3>请打分</h3>  <ul class="score">    <li></li>    <li></li>    <li></li>    <li></li>    <li></li>  </ul>  <script src="js/jquery-1.11.3.js"></script>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中: 因为多个li都要触发单击事件，所以应该用事件委托优化  //事件应该只绑定在父元素上一份即可  $("ul.score")  //2. 绑定事件处理函数  //事件委托必须用on，且只允许li元素触发事件  .on("click","li",function(){    //3. 查找要修改的元素    //4. 修改元素    //获得当前li    var $li=$(this);    //查找当前点击的li是ul下所有li中的第几个    var i=$("ul>li").index($li);    //alert(i);    //<i+1位置的元素都变为黄色    $(`ul>li:lt(${i+1})`).css("background-color","yellow")    //>i位置的元素都变为白色    $(`ul>li:gt(${i})`).css("background-color","#fff")  })  </script>  </body>  </html> |

总结:

1. 查找元素:

(1). 不需要查找就可直接获得的元素: 5个

a. document jq中: $(document)

b. <html> document.documentElement jq中: $("html")

c. <head> document.head jq中: $("head")

d. <body> document.body jq中: $("body")

e. <form> document.forms[i] jq中: $("form")

(2). 按节点间关系查找: 2大类关系 6个属性

何时: 如果已经获得一个元素，想找周围附件的元素时

a. 父子关系: 4个属性

1). 获得一个元素的父元素: 元素.parentNode

Jq中: $元素.parent()

2). 获得一个元素下的所有直接子元素: 元素.children

Jq中: $元素.children("选择器")

在所有后代中查找符合条件的元素: $元素.find("选择器")

3). 获得一个元素下的第一个直接子元素: 元素.firstElementChild

Jq中: $元素.children(":first-child")

4). 获得一个元素下的最后一个直接子元素: 元素.lastElementChild

Jq中: $元素.children(":last-child")

b. 兄弟关系: 2个属性

1). 获得一个元素的前一个兄弟元素: 元素.previousElementSibling

Jq中: $元素.prev()

获得一个元素之前所有兄弟: $元素.prevAll("选择器")

2). 获得一个元素的后一个兄弟元素: 元素.nextElementSibling

jq中: $元素.next()

获得一个元素之后所有兄弟: $元素.nextAll("选择器")

3). 获得除当前元素之外所有兄弟元素: $元素.siblings("选择器")

(3). 按HTML特征查找: 4个方法

何时: 只按一个条件就可找到想要的元素时

a. 按id查找一个元素: var 一个元素=document.getElementById("id名")

var $jq对象=$("#id名")

b. 按标签名查找多个元素:

var 类数组对象=任意父元素.getElementsByTagName("标签名")

var $jq对象=$("元素选择器")

c. 按class名查找多个元素:

var 类数组对象=任意父元素.getElementsByClassName("class名")

var $jq对象=$(".class名")

d. 按name名查找多个表单元素

var 类数组对象=document.getElementsByName("name名")

var $jq对象=$("[name=name名]")

(4). 按选择器查找: 2个方法

何时: 如果查找条件复杂时

a. 只查找一个符合条件的元素:

var 一个元素=任意父元素.querySelector("选择器")

b. 查找多个符合条件的元素:

var 类数组对象=任意父元素.querySelectorAll("选择器")

c. jq中统一为: var $jq对象=$("任意css选择器")

2. 修改元素: 3样:

(1). 内容: 3种:

a. 获取或修改元素的原始HTML内容: 元素.innerHTML

jq中: $元素.html("新html内容")

b. 获取或修改元素的纯文本内容: 元素.textContent

jq中:$元素.text("新纯文本内容")

c. 获取或修改表单元素的值: 元素.value

jq中:$元素.val("新表单元素值")

(2). 属性: 3种:

a. 字符串类型的HTML标准属性: 2种

1). DOM中:

i. 核心DOM 4个方法:

元素.getAttribute()/setAttribute()/hasAttribute()/removeAttribute()

ii. HTML DOM: 元素.属性名

2). Jq中:

i. $元素.attr()统一了getAttribute()和setAttribute()

ii. $元素.prop()代替了 元素.属性名

b. bool类型的HTML标准属性:

1). DOM中:

i. 只能用HTML DOM: 元素.属性名 值必须是bool类型

~~ii. 不能用核心DOM4个方法~~

2). Jq中: 只能用$元素.prop()

c. 自定义扩展属性: 2种:

1). DOM中:

i. 核心DOM 4个方法:

元素.getAttribute()/setAttribute()/hasAttribute()/removeAttribute()

ii. 如果自定义属性名以"data-"为前缀，还可用: 元素.dataset.属性名

iii. ~~不能用HTML DOM~~

2). Jq中: 只能用 $元素.attr();

(3). 样式:

a. 只修改一个元素的一个css属性: 元素.style.css属性

jq中: $元素.css("css属性名","属性值")

b. 获取一个元素的css属性: var style=getComputedStyle(元素); style.css属性

jq中: $元素.css("css属性名")

c. 批量修改一个元素的多个css属性:

1). DOM中: 元素.className

2). Jq中: $元素.addClass()/removeClass()/hasClass()/toggleClass()

3. 添加元素:

(1). 只添加一个元素: 3步

a. 创建新的空元素对象: var新元素=document.createElement("元素名")

b. 为新元素添加关键属性: 新元素.属性名=值

jq中: var $新元素=$(`HTML片段`)

c. 将新元素添加到DOM树: 3种:

1). 在父元素下所有子元素末尾追加一个新元素:

父元素.appendChild(新元素)

Jq中: $父元素.append($新元素)

$新元素. appendTo(父元素)

开头插入新元素: $父元素.prepend($新元素)

$新元素.prependTo(父元素)

2). 在父元素下的一个现有子元素前插入:

父元素.insertBefore(新元素, 现有子元素)

Jq中: $现有元素.before($新元素)

$新元素.insertBefore($现有元素)

在现有元素之后插入: $现有元素.after($新元素)

$新元素.insertAfter($现有元素)

3). 替换父元素下的一个现有子元素:

父元素.replaceChild(新元素, 现有子元素)

Jq中: $现有元素.replaceWith($新元素)

$新元素.replaceAll($现有元素)

(2). 添加多个元素时: 2种:

a. 同时添加父元素和多个子元素: 先将所有子元素加入到父元素中。最后再将父元素一次性添加到DOM树

b. 父元素已经在页面上，需要添加多个平级子元: 先将所有子元素加入到文档片段对象中，最后再一次性将文档片段对象加入到DOM树: 3步

1). 创建一个文档片段对象:

var frag=document.createDocumentFragment();

2). 将子元素加入到文档片段对象中:

frag.appendChild(新子元素)

3). 将文档片段对象整体加入DOM树

父元素.appendChild(frag)

c. jq中，都封装在了简化版函数内，很少再考虑这个问题了

4. 删除元素: 父元素.removeChild(子元素)

Jq中: $任意元素.remove()

5. 事件:

(1). 如何绑定事件处理函数: 2种:

a. 如果一个事件属性上只绑定一个处理函数: 赋值方式:

元素.on事件名=function(){ ... } 不推荐使用

|  |
| --- |
| 常用事件名:  blur 失去焦点  change 下拉列表选中项改变  click 单击  dblclick 双击  focus 获得焦点  keydown 键盘按键按下  keyup 键盘按键抬起  mousedown 鼠标按键按下  mouseenter 鼠标进入(jq)  mouseleave 鼠标移出(jq)  mousemove 鼠标移动  mouseout 鼠标移出(dom)  mouseover 鼠标进入(dom)  mouseup 鼠标按键抬起  resize 窗口大小改变  scroll 网页滚动 |

b. 如果一个事件属性上想同时绑定多个处理函数: 添加事件监听对象:

元素.addEventListener("事件名", 事件处理函数)

Jq中: $元素.事件名(事件处理函数)

比如: btn1.addEventListener("click",function(){ ... })

$btn1.click(function(){ ... })

移除事件监听: 元素.removeEventListener("事件名", 原事件处理函数对象)

(2). 事件模型: 3个阶段: 捕获、目标触发、冒泡执行

(3). 事件对象:

a. 如何获得事件对象: 元素.on事件名=function(e){ ... }

b. 取消冒泡: e.stopPropagation();

c. 利用冒泡/事件委托:

1). 优化: 如果多个平级子元素都要绑定相同的事件处理函数时，应该只在父元素上绑定一次处理函数即可！——结果: 点击任何一个子元素，都会冒泡到父元素上执行

2). 问题: 如果事件绑定在父元素上，this->父元素，不再指实际出发事件的子元素。

解决: 只要使用事件委托，都要用e.target代替this，获得目标元素

3). 问题: 如果只希望子元素中部分想要的元素执行事件处理函数，而不希望所有子元素都执行事件处理函数.

解决: 只要使用事件委托，都要先判断e.target是否想要的。

判断依据可以是任何条件: nodeName, className, innerHTML...

d. 取消默认行为: e.preventDefault()

e. 获得鼠标位置坐标: 3组坐标:

1). 鼠标所在位置相对于显示屏左上角的距离: e.screenX e.screenY

2). 鼠标所在位置相对于浏览器内文档显示区左上角的距离:

e.clientX e.clientY

3). 鼠标所在位置相对于事件绑定的元素左上角的距离:

e.offsetX e.offsetY

f. 页面滚动事件:

1). 在页面滚动时，自动执行:

// 当滚动时自动执行

window.onscroll=function(){

//希望获得页面滚动过的距离: body顶部超出文档显示区顶部的距离

var scrollTop=document.documentElement.scrollTop//能用第一就用第一个

||document.body.scrollTop//如果不能用第一个就退一步用第二个

}

2). 用程序主动控制让网页滚动到哪个位置:

window.scrollTo(水平方向滚动到的位置, 垂直方向滚动到的位置)

// 滚动 0

总结: $的用法: 4种

1. $("选择器") 查找DOM元素并包装进jQuery对象中

2. $(DOM元素) 无需查找，直接将一个DOM元素包装进jQuery中，$(this)

3. $(`HTML片段`) 创建新元素

4. $(function(){ ... }) 绑定DOM内容加载完成(DOMContentLoaded)就提前触发的事件处理函数

总结: jQuery简化版函数三大特点:

1. 自带for循环/遍历: 对整个jQuery对象调用一次简化版函数，等效于对jQuery对象中的每个DOM元素都调用一次等效的原生DOM函数或属性.

2. 一个函数两用: 凡是和获取/修改有关的函数，如果调用函数时没给新值，默认就读取旧值。如果给了新值，则自动切换为修改元素的属性值或内容。

3. 绝大多数简化版函数都会返回正在操作的.前的jQuery对象: 如果后续操作，依然是作用于同一个jquery对象上，可以继续连着写！——链式操作！

总结: this

1. obj.fun() this->obj

2. fun() 或 (function(){ ... })() 或 回调函数 this->window

3. new Fun() this->new正在创建的这个子对象

4. 构造函数.prototype.fun=function(){ ... } this->将来调用这个方法的某个子元素

5. 访问器属性中的this->当前访问器属性所在对象

Object.defineProperties(eric,{

eage:{

get(){return this.\_eage},

set(value){ this.\_eage=value }

}

})

6. 元素.onclick=function(){ ... } this->当前触发事件的DOM元素

7. $元素.click(function(){ ... }) this->当前触发事件的DOM元素

.addEventListener()

作业:

1. (扩展): jQuery发展过程中，共有几种事件绑定方式，差别是什么

思维导图->事件->鄙视：jQuery中共有几种事件绑定方式

2. (必须): 看小程序视频学习jQuery事件委托on off，并优化标签页、删除方块、切换图片三个案例

小程序->在线->jquery->day02 12. Jquery事件 事件委托 on off ...

3. (必须): 看小程序视频学习防抖和节流

小程序->在线->jquery->day03

2. 鼠标事件 模拟触发 ...

作业: 高频笔试题！防抖和节流

4. (必须): 看小程序视频学习做手风琴效果

小程序->在线->jquery->day03 作业: jQuery 动画函数 手风琴效果

5. (扩展): 看小程序问题清单实现一颗小星星点击后闪烁三次最后消失(6\_star.html)

小程序->首页 搜 小星星