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->day02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 ->day02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 ->day02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. 查找元素

2. 修改元素

3. 按节点间关系查找

4. 添加删除替换克隆

5. 事件绑定

一. 查找元素:

1. 内容过滤:

(1). 什么是: 根据元素内容中的关键词来选择符合条件的元素

(2). 何时: 如果通过元素身上其它特征无法有效区分元素时，可以尝试使用内容来区分不同的元素

(3). 包括:

a. :contains(关键词): 查找内容中包含指定"关键词"的元素

b. :has(选择器): 查找包含符合选择器要求的子元素的父元素，也就是根据元素内部子元素的特征来查找元素。

c. :parent : 查找所有包含内容(内容不为空)的元素

d. :empty : 查找所有内容为空的元素

(4). 示例: 使用内容过滤选择元素

|  |
| --- |
| <body>  <div class="container">  <h1>jQuery中的选择器——内容过滤选择器</h1>    <button>提交订单</button>    <button>Submit注册信息</button>    <button>马上提交</button>    <button>清空重填</button>    <hr/>    <div class="alert" id="alert1">      第一个警告框    </div>    <div class="alert" id="alert2">      <span class="close">×</span>      第一个警告框    </div>    <h1></h1>    <h1>Welcome</h1>  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>    //选择包含"提交"二字的按钮，变为绿色按钮    $("button:contains(提交)")    .css("background-color","lightGreen")    //选中包含.close按钮的.alert元素，让它们变为红色的警告框    $(".alert:has(.close)")    .css("background-color","pink")    //选中不包含.close按钮的.alert元素，让它们变为绿色的警告框    $(".alert:not(:has(.close))")    .css("background-color","lightGreen")    //选择没有内容的h1，高变为100px, 背景变为橙色    // $("h1:empty").css("height",100)    // $("h1:empty").css("background-color","orange")    $("h1:empty").css({      height:100,//只有jq中才有的福利，如果单位为px，可以不写      backgroundColor:"orange"    })    //选择有内容的h1，背景变为浅蓝色    $("h1:parent").css("background-color","lightBlue")  </script>  </body> |

2. 可见性过滤:

(1). 什么是: 根据元素是显示还是隐藏来选择元素

(2). 包括:

a. :visible 选择所有可见的元素

b. :hidden 选择所有不可见的元素

只能找到用display:none 和 type="hidden"隐藏的元素

无法找到: visibility: hidden 和 opacity:0 隐藏的元素

(3). 示例: 使用:hidden查找隐藏的元素

|  |
| --- |
| <body>  <div class="container">    <h1>可见性选择器</h1>    <div id="d1" style="display: none">lorem</div>    <div id="d2" style="visibility: hidden">lorem</div>    <div id="d3" style="opacity: 0">lorem</div>    <input id="d4" type="hidden" name="aa" value="bb"/>  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>    console.log( $('div.container>:hidden') )  </script>  </body> |

3. 表单元素过滤选择器:

(1). 什么是: 专门选择不同类型的表单元素的选择器

(2). 包括:

a. :input 可选择所有表单元素(input select textarea button)

vs input 只是一个元素选择器，只能选择input元素而已

b. 其实每种input的type属性都对应着一种专门的选择器，比如:

1). :text 专门选择 <input type="text"的元素

2). :password 专门选择 <input type="password"的元素

3). :button 专门选择 <input type="button"的元素

4). :submit 专门选择 <input type="submit"的元素

5). :radio 专门选择<input type="radio"的元素

6). :checkbox 专门选择<input type="checkbox"的元素

7). :image 专门选择<input type="image"的元素

8). :reset 专门选择<input type="reset"的元素

9). :file 专门选择<input type="file"的元素

(3). 示例: 使用表单元素选择器实现功能

|  |
| --- |
| <body>  <form>    用户名:<input disabled></br>    密码:<input type="password" disabled></br>    <input type="checkbox">我同意本站的使用条款<br>    <input type="submit" value="提交注册信息" disabled/>  </form>  <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type="checkbox"的元素  $(":checkbox")  //2. 绑定事件处理函数  .click(function(){    //3. 查找要修改的元素    //本例中: 查找除checkbox之外的所有其余表单元素    var $others=$(":input:not(:checkbox)")    //4. 修改元素    //如果当前chb选中，则其他元素启用    //         获取checked属性值    // if($(this).prop("checked")==true){    //  $others.prop("disabled",false)    //      // .disabled=false    // }else{//否则如果当前chb选中，则其他元素禁用    //  $others.prop("disabled",true)    // }    //基础好的同学: 上边这个if else，写成一句话！    $others.prop("disabled",!$(this).prop("checked"))  })  </script>  </body> |

二. 修改元素: 3种:

1. 内容: 3种:

(1). 原始的HTML内容:

a. DOM中: 元素.innerHTML

b. jq中: $元素.html()

(2). 纯文本内容:

a. DOM中: 元素.textContent

b. jq中: $元素.text()

(3). 表单元素的值:

a. DOM中: 元素.value

b. jq中: $元素.val()

(4). 示例: 使用元素内容实现表单验证:

|  |
| --- |
| <body>  <h1>操作元素的内容和值</h1>  <form action="">  用户名:<input type="text" name="uname">        <span></span><br>  密码:<input type="password" name="upwd">      <span></span><br>  <input type="submit" value="提交注册信息">  </form>  <script src="js/jquery-1.11.3.js"></script>  <script>  //正确时，使用图片:"<img src='img/ok.png'>"  //姓名错误时: "<img src='img/err.png'>用户名必须介于3~9位之间!"  //密码错误时: "<img src='img/err.png'>密码必须介于6~8位之间!"  //DOM 4步  //1. 查找触发事件的元素  //本例中: 查找type=text的文本框  $(":text")  //2. 绑定事件处理函数  .blur(function(){    var $input=$(this);    //3. 查找要修改的元素    //本例中: 查找当前input旁边的span    var $span=$input.next();    //获得当前文本框的内容    var value=$input.val().trim();    //4. 修改元素    if(value.length>=3&&value.length<=9){      $span.html("<img src='img/ok.png'>")    }else{      $span.html("<img src='img/err.png'>用户名必须介于3~9位之间!")    }  })  </script>  </body> |

2. 属性: 3种:

(1). 字符串类型的HTML标准属性:

a. DOM中: 2种:

1). 核心DOM 4个函数:

i. 元素.getAttribute()

ii. 元素.setAttribute()

iii. 元素.hasAttribute()

iv. 元素.removeAttribute()

2). HTML DOM: 元素.属性名

b. jq中: 2种:

1). 用$元素.attr()代替了元素.getAttribute()和元素.setAttribute()

i. 获取一个元素指定属性的属性值: $元素.attr("属性名")

ii. 修改一个元素指定属性的属性值: $元素.attr("属性名","新属性值")

2). 因为jQuery是函数库，所以不能用属性，所以用$元素.prop()代替元素.属性

i. 获取指定属性值: $元素.prop("属性名")

ii. 修改指定属性值: $元素.prop("属性名","新属性值")

(2). Bool类型的HTML标准属性:

a. DOM中: 只能用HTML DOM的.来修改，~~不能用核心DOM的4个函数~~

b. jq中: 只能用$元素.prop()，~~不能用$元素.attr()~~

(3). 自定义扩展属性: (不考虑HTML5)

a. DOM中: 只能用核心DOM的4个函数修改，~~不能用HTML DOM的.修改~~

b. jq中: 只能用$元素.attr()，~~不能用$元素.prop()~~

|  |
| --- |
| 简写:同时修改一个元素的多个属性或css属性时，都可用对象语法简写:  $元素.css/attr/prop({  属性名:属性值,  属性名:属性值,  }) |

|  |
| --- |
| 总结: attr() vs prop()  attr()代替核心DOM操作，凡是DOM中能用核心DOM 4个函数操作的属性，都能用attr()操作  prop()代替HTML DOM的.操作，凡是DOM中能用.访问的属性，都能用prop()操作！ |

(4). 示例: 点击图片切换下一张

|  |
| --- |
| <body>  <h1>操作元素的属性</h1>  <img src="img/1.jpg" alt="1">  <script src="js/jquery-1.11.3.js"></script>  <script>  //单击图片，切换下一张  //DOM 4步  //1. 查找触发事件的元素  //本例中: 查找img元素  $("img")  //2. 绑定事件处理函数  .click(function(){    //3. 查找要修改的元素    var $img=$(this);    //4. 修改元素    //取出当前img的alt属性值，转为整数    var n=parseInt($img.attr("alt"))                    // .prop("alt")    //如果alt值<4，就+1    //否则如果alt值==4了，就将alt值恢复成1    n<4?(n+=1):(n=1);    //再将alt值放回src属性和alt属性中    // $img.attr("src",`img/${n}.jpg`);    // $img.attr("alt",n);    $img.attr({//.prop({      src:`img/${n}.jpg`,      alt:n    })  }) |

(5). 示例: 点小图片，切换到大图片

|  |
| --- |
| <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>    <style>      body {        text-align: center;      }    </style>  </head>  <body>  <img src="img/1.jpg" data-target="img/1-l.jpg" class="my-small">  <img src="img/2.jpg" data-target="img/2-l.jpg" class="my-small">  <img src="img/3.jpg" data-target="img/3-l.jpg" class="my-small">  <img src="img/4.jpg" data-target="img/4-l.jpg" class="my-small">  <hr/>  <img src="img/1-l.jpg" class="my-big">  <script src="js/jquery-1.11.3.js"></script>  <script>  //点击小图片，下方my-big中显示大图片  //DOM 4步  //1. 查找触发事件的元素  //本例中: 查找class为my-small的img元素  $(".my-small")  //2. 绑定事件处理函数  .click(function(){    //3. 查找要修改的元素    //本例中要改class为my-big的img    var $img=$(".my-big");    //4. 修改元素    //先获取当前图片的data-target属性值    var src=$(this).attr("data-target");    //赋值给$img的src属性    //$img.attr("src",src);    $img.attr({src});//ES6简写  })  </script>  </body>  </html> |

3. 样式:

(1). 回顾: DOM中:

a. 修改一个元素的一个css属性: 元素.style.css属性

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

c. 批量修改一个元素的多个css属性: 元素.className

DOM中className的问题: 多个class名放在一个字符串中，非常不便于操作其中某一个class。  
 (2). Jq中:

a. 用$元素.css()统一了获取和修改一个元素的css属性

1). $元素.css("css属性") 默认执行获取属性值的操作，相当于getComputedStyle

2). $元素.css("css属性","新属性值") 变成执行修改属性值的操作，相当于style

b. 还提供了一组灵活的操作className的函数:

1). $元素.addClass("class名")

2). $元素.removeClass("class名")

3). $元素.hasClass("class名")

4). 如果确实是在有和没有一个class之间来回切换:

i. $元素.toggleClass("class名")

切换

ii. 等效于:

//如果当前按钮上有down class

if($btn.hasClass("down")){

//就移除down class

$btn.removeClass("down")

}else{//否则如果当前按钮上没有down class

//就添加down class

$btn.addClass("down")

}

iii. 强调: addClass()无论何时都是添加class，removeClass()无论何时都是移除class，非常可靠的。但是toggleClass()是一次添加，一次删除。没准！所以，绝不能用toggleClass()代替addClass()和removeClass()。只有确信实在有和没有一个class之间来回切换时，才能用toggleClass()。

5). 示例: 使用jQuery操作class

|  |
| --- |
| <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <style>  .btn {    padding: 5px 10px;    border-radius: 3px;    border: 1px solid #aaa;    outline: none;  }  .up {    background: #fff;    color: #333;  }  .down {    background: #ddd;    color: #fff;  }  </style>  </head>  <body>  <button class="btn up">双态按钮</button>  <script src="js/jquery-1.11.3.js"></script>  <script>  //双态按钮: 让按钮的class在up和down之间切换  //DOM 4步  //1. 查找触发事件的元素  //本例中: 查找一个按钮  $("button")  //2. 绑定事件处理函数  .click(function(){    //3. 查找要修改的元素    var $btn=$(this);    //4. 修改元素    // //如果当前按钮上有down class    // if($btn.hasClass("down")){    //   //就移除down class    //   $btn.removeClass("down")    // }else{//否则如果当前按钮上没有down class    //   //就添加down class    //   $btn.addClass("down")    // }    //在有和没有down这个class之间来回切换    $btn.toggleClass("down");  })  </script>  </body>  </html> |

三. 按节点间关系查找元素: 2大类关系 6个属性

1. 父子关系: 4个属性

(1). 获得一个元素的父元素:

a. DOM中: 元素.parentNode

b. jq中: $元素.parent()

(2). 获得所有直接子元素:

a. DOM中: 元素.children

b. jq中: $元素.children("选择器");

c. jq中新增: 获得一个元素下所有后代中符合条件的子元素

$元素.find("选择器")

(3). 获得第一个直接子元素:

a. DOM中: 元素.firstElementChild

b. jq中: $元素.children(":first-child")

(4). 获得最后一个直接子元素

a. DOM中: 元素.lastElementChild

b. jq中: $元素.children(":last-child")

2. 兄弟关系: 2个属性

(1). 前一个兄弟:

a. DOM中: 元素.previousElementSibling

b. jq中: $元素.prev()

c. 问题: 只能选前一个，无法选择之前所有

d. 解决: jq中: $元素.prevAll("选择器")

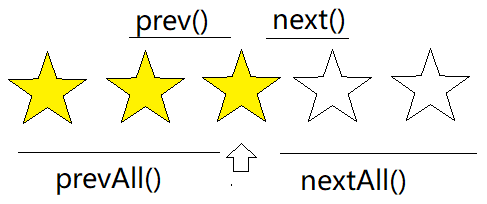
(2). 下一个兄弟:

a. DOM中:元素.nextElementSibling

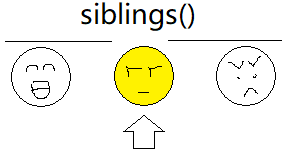
b. jq中: $元素.next()

c. 问题: 只能选下一个，无法选择之后所有

d. 解决: jq中: $元素.nextAll("选择器")



(3). Jq中: 除自己之外所有兄弟: $元素.siblings()



3. 示例: 使用按节点间关系查找的函数

|  |
| --- |
| <body>  <!--ul.top>(li.parent>ul>li.child\*3)\*2 -->  <ul class="top">    <li class="parent1">parent1      <ul>        <li class="child">child1</li>        <li class="child">child2</li>        <li class="child">child3</li>      </ul>    </li>    <li class="parent2">parent2      <ul>        <li class="child">child1</li>        <li class="child">child2</li>        <li class="child">child3</li>      </ul>    </li>  </ul>  <script src="js/jquery-1.11.3.js"></script>  <script>  //修改class为top的ul的所有直接子元素  $("ul.top").children()  .css("border","1px solid red")  //修改class为top的ul的所有后代li  $("ul.top").find("li")  .css("box-shadow","0 0 5px green")  //为class为child的li绑定单击事件  $("li.child").click(function(){    //选择当前元素的下一个元素/前一个元素/之前所有/之后所有/除自己之外所有    $(this).siblings()//.nextAll()//.prevAll()//.prev()//.next()    .css("background-color","yellow")  })  </script>  </body> |

4. 示例: 标签效果

|  |
| --- |
| <head>  <meta charset="UTF-8">  <style>  .tabs{ list-style:none; padding:0 }  .tabs a{    text-decoration:none;    color:#000;    padding:6px 12px;    display:inline-block;  }  .tabs>li{    float:left;    border-bottom:1px solid #000;  }  .tabs>.active{    border:1px solid #000;    border-bottom:0;  }  </style>  </head>  <body>  <h1>使用属性选择器实现标签页头的切换</h1>  <ul class="tabs">    <li class="active">      <a data-toggle="tab" href="#">十元套餐</a>    </li>    <li>      <a data-toggle="tab" href="#">二十元套餐</a>    </li>    <li>      <a data-toggle="tab" href="#">三十元套餐</a>    </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: 查找所有有data-toggle属性，且属性值为tab的元素  $("[data-toggle=tab]")  //2. 绑定事件处理函数  .click(function(){    //3. 查找要修改的元素    //4. 修改元素    //先给自己的爹加active class    $(this)  //当前a    .parent() //当前a的爹li    .addClass("active") //return 当前a的爹li    //再给爹的其余兄弟都去掉active class    .siblings() //当前a的爹li的其余兄弟    .removeClass("active")  })  </script>  </body> |

四. 添加/删除/替换/克隆

1. 添加:

(1). DOM中:

a. 添加一个元素:3步

1). 创建新的空元素对象: var 新元素=document.createElement("标签名")

2). 设置新元素的必要属性: 新元素.属性名=属性值

3). 将新元素添加到DOM树: 3种:

i. 在指定父元素下所有子元素末尾追加新元素: 父元素.appendChild(新元素)

ii. 在指定父元素下某个子元素前插入: 父元素.insertBefore(新元素,现有子元素)

iii. 替换指定父元素下某个子元素: 父元素.replaceChild(新元素, 现有子元素)

b. 添加多个元素: 优化:

1). 如果同时添加父元素和子元素: 应该先将所有子元素添加父元素中，再将父元素一次性添加到DOM树

2). 如果同时添加多个平级子元素: 应该先将所有子元素添加到文档片段对象中，再将文档片段对象添加到DOM树

(2). Jq中: 2步:

a. 创建新元素: 不管创建一个元素，还是创建多个元素，一律用:

var $新元素=$(`HTML片段`)

b. 将新元素添加到DOM树:

1). 在指定父元素下所有子元素末尾追加新元素:

$父元素.append($新元素)

$新元素.appendTo(父元素)

2). 在指定父元素下所有子元素开头插入新元素:

$父元素.prepend($新元素)

$新元素.prependTo(父元素)

3). 插入到一个现有元素之前:

$现有元素.before($新元素)

$新元素.insertBefore(现有元素)

4). 插入到一个现有元素之后:

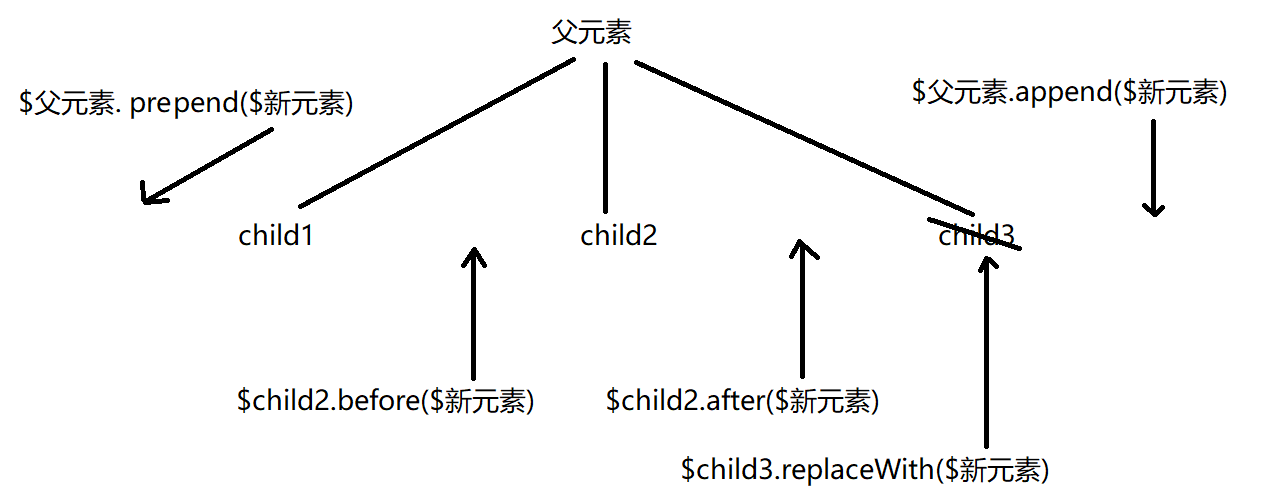
$现有元素.after($新元素)

$新元素.insertBefore(现有元素)

5). 替换一个现有元素:

$现有元素.replaceWith($新元素)

$新元素.replaceAll(现有元素)



c. 说明: 因为新元素加入到DOM树后，有时需要继续对新元素做修改，有时又需要对现有元素做修改，为了适应各种链式操作的需要，jq作者将五个添加到DOM树的函数，分别扩展出了一个兄弟，仅仅是主语不同而已。

d. 如何选择: 如果添加新元素后

1). 想继续修改新元素: 应该选择那些以新元素作为开头的函数

2). 想继续修改现有元素: 应该选择那些以现有元素作为开头的函数

3). 想继续修改父元素: 应该选择那些以父元素开头的函数.

2. 删除: $元素.remove()

3. 示例: 添加删除方块:

|  |
| --- |
| <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){    //事件委托2件事:    //1). 用e.target代替this //复习DOM第四天事件委托    var $x=$(e.target);    //2). 判断只允许符合条件的元素执行操作    //本例中: 只允许内容为×的元素触发事件    if($x.html().trim()=="×"){    //3. 查找要修改的元素    //本例中点×，要删除其父元素    $x.parent()    //4. 修改元素      .remove();    }  })  </script>  </body> |

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

2). 获得一个元素下的所有直接子元素: 元素.children

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

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

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

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

(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

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)

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)

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)

(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)

4. 删除元素: 父元素.removeChild(子元素)

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. ?

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

作业:

1. (必须)看小程序视频完成使用元素内容验证表单元素:

小程序->在线->jquery->day02 1. Jquery修改元素的内容 html() text() val()

2. (必须)看小程序视频学习添加删除替换克隆，完成选飞机案例

小程序->在线->jquery->day02 7. Jquery添加删除元素

3. (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 ...

4. (扩展)看小程序视频学习使用jQuery实现下拉菜单

小程序->在线->jquery->day02 作业: jQuery单击下拉菜单

5. (扩展)看小程序视频学习使用jQuery实现带动画效果的弹出对话框

小程序->在线->jquery->day02 作业:jQuery带多种动画效果的对话框alert...