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->DOM->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->DOM->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->DOM->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！

今天开始！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

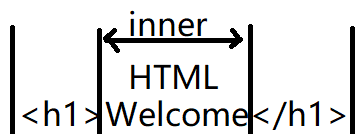
3. 删除注释和代码，自己用人话把注释写回来！——才算自己会！

正课:

1. 修改元素
2. 添加删除
3. 修改元素: 一个元素身上3种东西可修改

(1). 内容: 3种:

a. 获取或修改元素开始标签到结束标签之间的原始的HTML片段: 元素.innerHTML



获取元素的HTML内容时: 返回原始的HTML代码内容

修改元素的HTML内容时: 先将新的HTML代码内容交给浏览器解析后，再显示在元素内部

b. 获取或修改元素开始标签到结束标签之间的纯文本内容: 元素.textContent

文本 内容

获取元素的纯文本内容时: textContent会去掉内嵌标签，还会将特殊符号翻译为正文

修改元素的纯文本内容时: 不会将新内容交给浏览器解析，而是直接原样显示在元素内部

c. 获取或修改表单元素的值:

1). 因为大多数表单元素都是单标记，没有结束标签，所以不能用~~innerHTML或textContent~~

2). 只能用: 表单元素.value

|  |
| --- |
| 强调: 凡是从页面上获得的一律是字符串类型！尤其是进行后续算术计算之前，都要先转为数字，再计算！ |

d. 示例: 获取或修改元素的内容

|  |
| --- |
| <body>  <p id="p1">来自<a href="javascript:;">&lt;&lt;新华社&gt;&gt;</a>的消息</p>  <p id="p2">来自<a href="javascript:;">&lt;&lt;新华社&gt;&gt;</a>的消息</p>  <input type="text" id="txt1"><button id="btn1">百度一下</button>  <script>  //想点击按钮，可获得文本框中的值  //DOM 4步  //1. 查找触发事件的元素  //本例中: 先查找id未btn1的按钮  document.getElementById("btn1")  //2. 绑定事件处理函数  .onclick=function(){    //3. 查找要修改的元素    //本例中: 先获得txt1    var txt1=document.getElementById("txt1");    //4. 修改元素    console.log(`搜索 "${txt1.value.trim()}" 相关的内容`)                                //复习jscore day02  }  var p1=document.getElementById("p1");  //想获取p1内原始的HTML内容:  console.log(p1.innerHTML);  //想修改p1的内容为新的HTML内容:  p1.innerHTML=`来自<a href="javascript:;">&lt;&lt;路透社&gt;&gt;</a>的消息`;  var p2=document.getElementById("p2");  //想获得p2内的纯文本内容，不要内嵌标签，也不想看到特殊符号！  console.log(p2.textContent);  //想修改p2的内容为新的内容  p2.textContent=`来自<a href="javascript:;">&lt;&lt;路透社&gt;&gt;</a>的消息`;  </script> |

d. 示例: 开关门效果：

|  |
| --- |
| <head>  <title>读取并修改元素的内容</title>  <meta charset="utf-8" />  <style>    div{float:left; height: 100px; line-height: 100px; }    #d1,#d3{ background-color: #ccff00; }    #d2{ cursor: pointer; background-color: #ffcc00; }  </style>  </head>  <body>  <div id="d1">树形列表</div>  <div id="d2">&lt;&lt;</div>  <div id="d3">内容的主体</div>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中:点击d2，让d1切换显示隐藏，所以先查找id为d2的元素  document.getElementById("d2")  //2. 绑定事件处理函数  .onclick=function(){    //3. 查找要修改的元素    //本例中: d1切换显示隐藏，所以应该查找d1    var d1=document.getElementById("d1");    //4. 修改元素    //小技巧: 当判断元素是显示还是隐藏时，应该先判断元素display是否为null。    //因为第一次元素上暂时没有display:block，如果先判断display:block，也会被误认为正处于隐藏状态    //如果d1是隐藏的    if(d1.style.display==="none"){      //就让d1显示      d1.style.display="block";      //同时让当前d2的内容变为<<      this.innerHTML="&lt;&lt;";    }else{//否则如果d1是显示的      //就让d1隐藏      d1.style.display="none";      //同时让当前d2的内容变为>>      this.innerHTML="&gt;&gt;";    }  }  </script>  </body> |

(2). 属性: 3种属性

a. 字符串类型的HTML标准属性:

1). 什么是字符串类型的标准属性: HTML标准中规定的，几乎所有元素都有的，值为字符串类型的属性。

2). 比如: <元素 title=”xxx” style=”xxx” id=”xxx” class=”xxx” name=”xxx” href=”xxx” …>

3). 2种方式:

i. 使用早期的核心DOM函数: 4个

①获取一个属性值: 元素.getAttribute(“属性名”)

②修改一个属性值: 元素.setAttribute(“属性名”,”新属性值”) //有属性就改，没有就自动添加

③移除一个属性值: 元素.removeAttribute(“属性名”)

④判断是否包含某个属性: var bool=元素.hasAttribute(“属性名”)

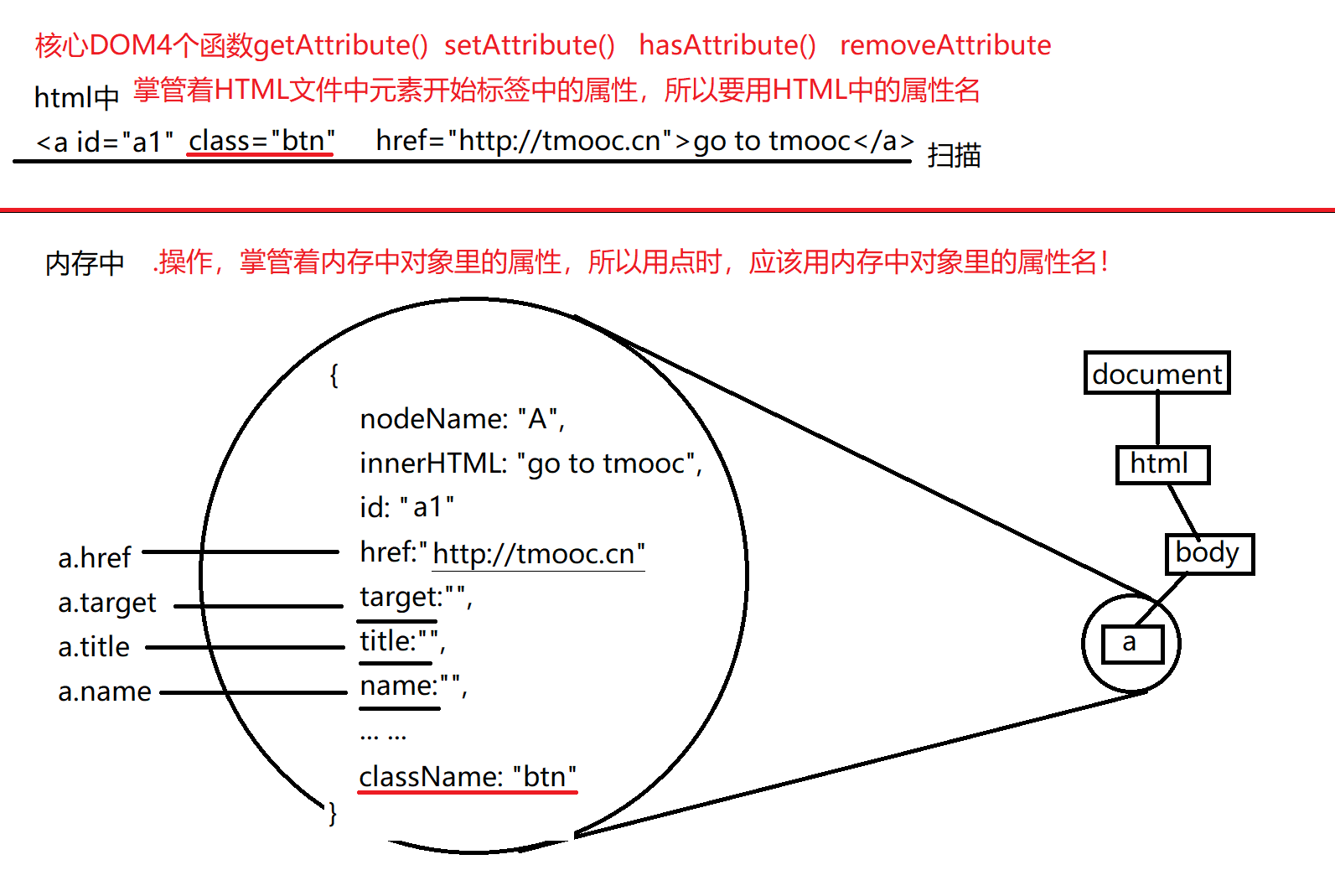
⑤示例: 使用核心DOM函数操作元素属性

|  |
| --- |
| <body>  <a id="a1" class="btn" href="http://tmooc.cn">go to tmooc</a>  <script>  //查找id为a1的元素  var a1=document.getElementById("a1");  //获得a1的href属性值  console.log(a1.getAttribute("href"));  //想修改a1的target属性为\_blank  a1.setAttribute("target","\_blank");  //判断如果a1没有title属性，就给a1添加title属性  console.log(a1.hasAttribute("title"));//false  if(a1.hasAttribute("title")==false){    a1.setAttribute("title","欢迎访问tmooc");  }  console.log(a1.hasAttribute("title"));//true  //想移除a1的id属性  a1.removeAttribute("id");  //想获得a1的class属性值  console.log(a1.getAttribute("class"))  //想修改a1的class为alert  a1.setAttribute("class","alert");  </script>  </body> |

ii. 较新版的DOM中规定了新的简化版属性: HTML DOM

①什么是HTML DOM: HTML DOM是DOM标准针对旧的函数较长的问题推出的一套简化版属性和方法。

②原理: 所有HTML标准属性，其实在浏览器扫描HTML文档创建元素对象时，就已经全部提前定义在了内存中的元素对象上。只不过，大部分没有值的HTML标准属性值暂时为””。



③如何:

获取属性值: 元素.属性名

修改属性值: 元素.属性名=”新值”

移除属性值: 元素.属性名=”” //因为””就表示无效属性

判断是否包含一个属性: 元素.属性名!==”” //说明包含该属性

④特例: html中的class属性

因为ECMAScript核心语法标准中已经提前规定了class是js语言的关键词，用于创建一种类型。所以，DOM标准就不能重复使用class作为属性名。

所以，今后只要用HTML DOM打.访问元素的class属性，必须更名为.className

同理，修改元素.className属性等效于修改元素的class属性。

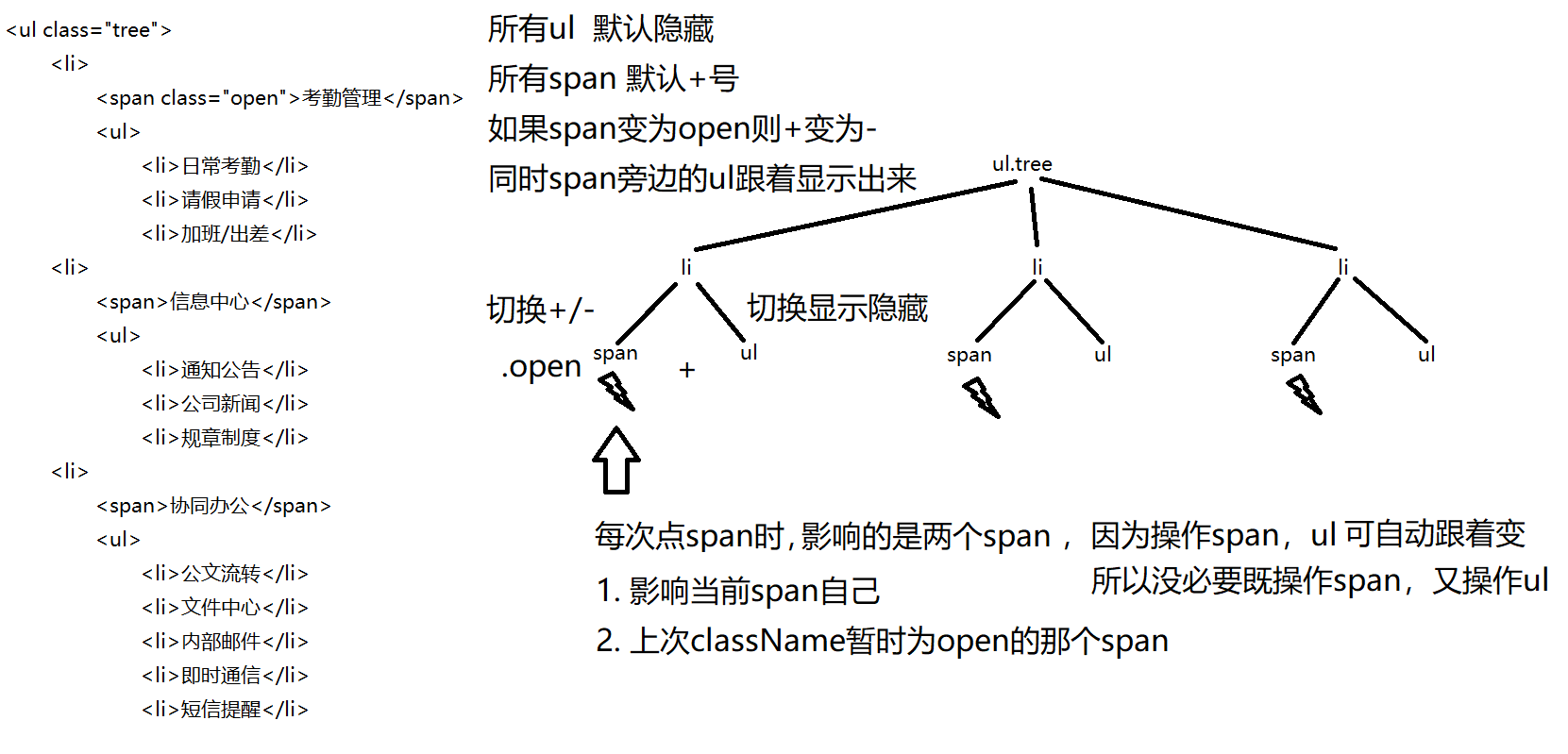
比如: ~~a1.class~~ 应该改为 a1.className

但是，如果用核心DOM函数getAttribute()/hasAttribute()/setAttribute()/removeAttribute()时，不能用className，必须换回使用class。

⑤示例: 使用HTML DOM简写操作元素属性

|  |
| --- |
| <body>  <a id="a1" class="btn" href="http://tmooc.cn">go to tmooc</a>  <script>  //查找id为a1的元素  var a1=document.getElementById("a1");  //获得a1的href属性值  console.log(a1.href);  //想修改a1的target属性为\_blank  a1.target="\_blank";  //判断如果a1没有title属性，就给a1添加title属性  console.log(a1.title!=="");//false                    //复习javascript思维导图->运算符和表达式->关系运算->隐式转换->特殊->null和undefined  if(a1.title===""){    a1.title="欢迎访问tmooc";  }  console.log(a1.title!=="");//true  //想移除a1的id属性  a1.id="";  //获得元素的class属性  console.log(a1.className);  //修改元素的class属性为alert  a1.className="alert";  //暂时不考虑多个class的情况  </script>  </body> |

⑥示例: 不带过渡的手风琴效果，伸缩二级菜单



|  |
| --- |
| <head>  <title>1. 实现伸缩二级菜单</title>  <meta charset="utf-8" />  <style>  li{    list-style:none;  }  /\*所有span默认+号\*/  li span{    padding-left: 20px;    cursor: pointer;    background: url("../images/add.png") no-repeat center left;  }  /\*所有ul默认隐藏\*/  li ul{display: none; }  /\*为将来的变化定预案\*/  /\*如果有span的class变为open，则span自己+号变为-号\*/  .open{ background: url("../images/minus.png") no-repeat center left; }  /\*span旁边的ul跟着span一起由隐藏变为显示\*/  .open+ul{ display: block; }  /\*结果: 将来只要修改span一个元素的className为open，就能同时让span变为-，同时让span旁边的ul显示出来\*/  </style>  </head>  <body>  <ul class="tree">    <li>      <span class="open">考勤管理</span>      <ul>        <li>日常考勤</li>        <li>请假申请</li>        <li>加班/出差</li>      </ul>    </li>    <li>      <span>信息中心</span>      <ul>        <li>通知公告</li>        <li>公司新闻</li>        <li>规章制度</li>      </ul>    </li>    <li>      <span>协同办公</span>      <ul>        <li>公文流转</li>        <li>文件中心</li>        <li>内部邮件</li>        <li>即时通信</li>        <li>短信提醒</li>      </ul>    </li>  </ul>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中: 点每个一级li中的span，开关span旁边的ul  //所以先找ul.tree下的li下的span  var spans=document.querySelectorAll("ul.tree>li>span")  //2. 绑定事件处理函数  //遍历: spans中每个span  for(var sp of spans){    //每遍历一个span，就为其绑定单击事件    sp.onclick=function(){      //this->当前点中的这一个span      //this.innerHTML="❀";      //3. 查找要修改的元素      //本例中: 点一个span，会影响两个span      //首先影响当前span自己(this)      //其次影响之前className处于open的span      var openSpan=document.querySelector("ul.tree>li>span.open")//根据需求同一时刻只能有一个span开着      //4. 修改元素      //本例中: 自己开，别人关      this.className="open";      //如果可以找到开着的span，      if(openSpan!=null){        openSpan.className="";//才有必要清除span的class。      }//否则如果所有span都关上了，就没必要清除span的class    }  }  </script>  </body> |

b. bool类型的标准属性:

1). 什么是bool类型的HTML标准属性: HTML标准中规定的，只要放在元素上就起作用，不再元素上就没作用。且不需要指定属性值！

2). 比如: <元素 checked disabled selected> ——也称为状态属性

3). 如何获取和修改:

i. 错误: 不能用核心DOM的四个函数(~~getAttribute()/setAttribute()/has Attribute()/removeAttribute()~~)修改。因为核心DOM四个函数，只能修改字符串类型的属性。

ii. 正确: 只能用HTML DOM的 元素.属性名 方式修改。且值为bool类型

4). 示例: 分别使用核心DOM和HTML DOM简写获取一个checkbox的选中状态

|  |
| --- |
| <body>  <input type="checkbox" id="chb1" >同意  <script>  document.getElementById("chb1")  .onclick=function(){    console.log(this.checked);    alert(this.checked==true?"我同意":"我不同意")    // 错误做法:  //console.log(this.getAttribute("checked"));    // alert(this.getAttribute("checked")==true?"我同意":"我不同意")  }  </script>  </body> |

5). 其实css中有一类选择器，专门匹配处于某种状态的元素:

a, :disabled 专门匹配禁用的元素

b. :checked 专门匹配选中的checkbox或radio

c. :selected 专门匹配选中的option

6). 示例: 全选和取消全选

|  |
| --- |
| <body>  <h2>管理员列表</h2>  <table border="1px" width="500px">    <thead>    <tr>      <th><input type="checkbox"/>全选</th>      <th>管理员ID</th>      <th>姓名</th>      <th>操作</th>    </tr>    </thead>    <tbody>      <tr>        <td><input type="checkbox"/></td>        <td>1</td>        <td>Tester</td>        <td>修改 删除</td>      </tr>      <tr>        <td><input type="checkbox"/></td>        <td>2</td>        <td>Manager</td>        <td>修改 删除</td>      </tr>      <tr>        <td><input type="checkbox"/></td>        <td>3</td>        <td>Analyst</td>        <td>修改 删除</td>      </tr>      <tr>        <td><input type="checkbox"/></td>        <td>4</td>        <td>Admin</td>        <td>修改 删除</td>      </tr>    </tbody>  </table>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中:查找table>thead下的input  var chbAll=document.querySelector("table>thead input")  //2. 绑定事件处理函数  chbAll.onclick=function(){    //3. 查找要修改的元素    //本例中: 要修改tbody下每行第一个td中的input    var inputs=document.querySelectorAll("table>tbody>tr>td:first-child>input");    //4. 修改元素    //遍历inputs中每个checkbox    for(var chb of inputs){      //让每个checkbox的checked属性和chbAll的checked保持一致！      chb.checked=chbAll.checked;    }  }  //DOM4步  //1. 查找触发事件的元素  //本例中:查找tbody下每行第一个td中的input  var inputs=document.querySelectorAll("table>tbody>tr>td:first-child>input");  //2. 绑定事件处理函数  //本例中: 遍历inputs中每个checkbox，为每个checkbox都绑定单击事件  for(var chb of inputs){    chb.onclick=function(){      //3. 查找要修改的元素（chbAll已经找到了）      //4. 修改元素      //先查找tbody下有没有未选中的checkbox      var checked=document.querySelector("table>tbody>tr>td:first-child>input:not(:checked)");      //如果tbody中没找到未选中的checkbox，则chbAll就选中      if(checked==null){        chbAll.checked=true      }else{//否则如果tbody中找到了未选中的checkbox，则chbAll就不选中        chbAll.checked=false      }    }  }  </script>  </body> |

c. 自定义扩展属性:

1). 什么是自定义扩展属性: HTML标准中没有规定的，程序员自发添加到元素上的自定义属性。

2). 何时: 2种典型场景:

i. 代替其他选择器(id, class, 元素,…)来专门用于选择触发事件的元素

①其他选择器的问题:

id选择器: 只能选择一个元素，无法选择多个元素

元素选择器: 实现同一种效果，不一定人们都用相同的元素，

比如: 实现一个按钮： a button span input div …

将来维护html内容时，很可能就会换成别的元素实现相同的效果。

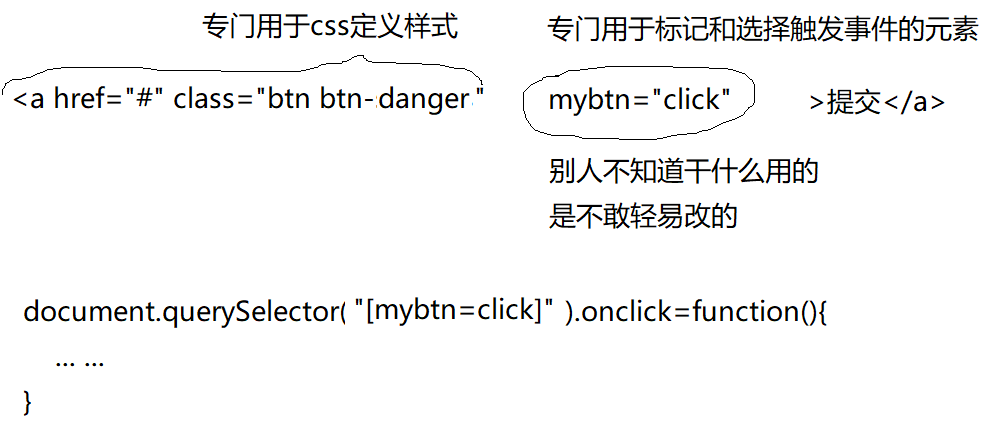
所以元素选择器也不能保证万无一失

class选择器: class的本职工作是定义元素样式！不是选择触发事件的元素写js。

将来维护css时，很可能就会修改元素的class。

所以类选择器也不能保证万无一失！

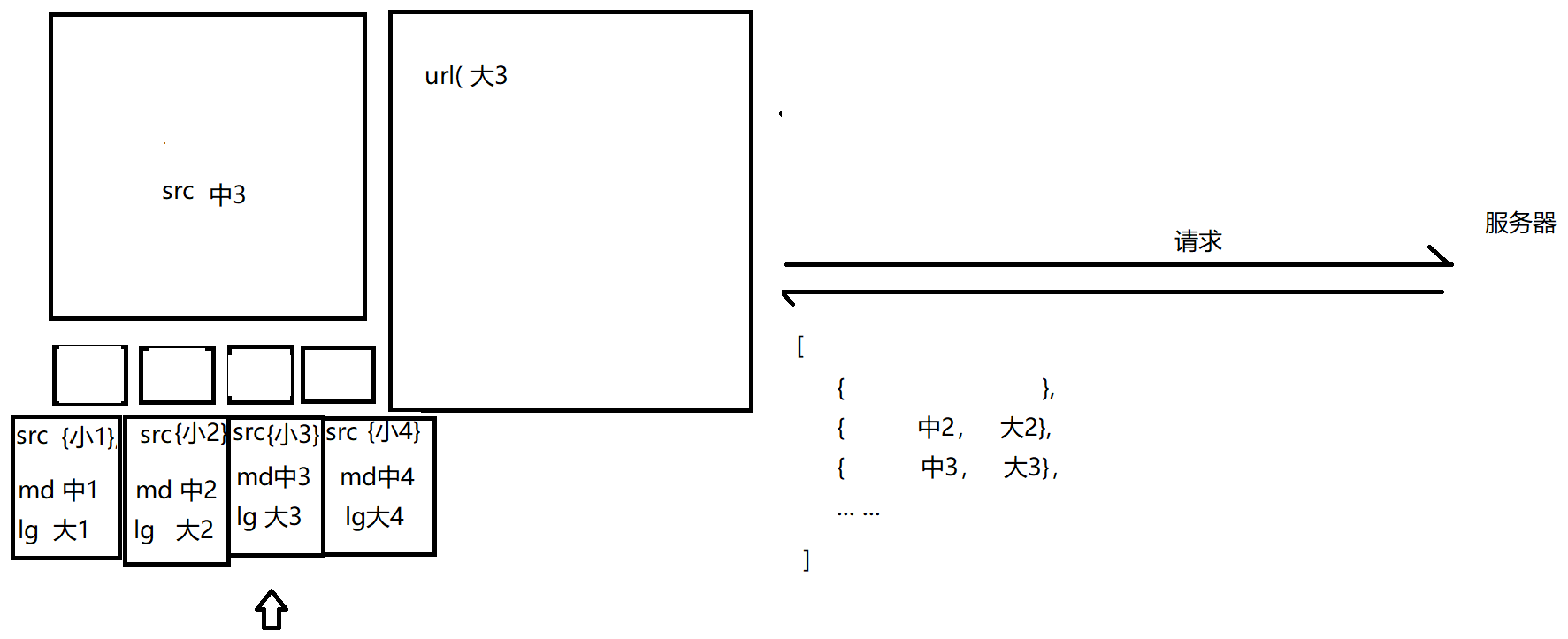
②解决: 今后只要选择触发事件的元素都应该用自定义属性+属性选择器！



ii. 在客户端临时缓存数据，减少向服务器端发送请求的次数

①问题: 有时为了减少后续操作向服务器端发送请求的次数，会在首次请求时，就拿到全部数据。之后再执行操作时，无需重复向服务器发送请求，只需要从浏览器本地取数据即可。但是，一个元素身上可用的标准属性有限，暂时用不到的属性？放哪儿？！

②解决: 为元素添加自定义扩展属性来保存暂时用不到的，但是后续操作可能会用到的数据。——最简单的缓存。



3). 如何使用自定义扩展属性:

i. 在HTML中元素的开始标签里添加自定义扩展属性:

①HTML中: <元素 自定义属性名=”属性值”

比如: <img md=”img/md1.jpg”>

②可通过js添加: 元素.setAttribute(“自定义属性名”,”属性值”)

比如: img.setAttribute(“md”, “img/md1.jpg”)

~~不能用HTML DOM的.操作自定义扩展属性！因为HTML DOM只能操作标准属性！不包含自定义属性！~~

ii. 获取元素上自定义扩展属性的值:

只能核心DOM的 元素.getAttribute(“自定义属性名”)

~~不能用HTML DOM的.方式获取自定义属性的值~~

iii. HTML5标准中新规定:

①所有的自定义扩展属性，在定义时都要以”data-“作为前缀

<img data-md=”img/md1.jpg”>

②简写: 如果定义属性时采用data-前缀，则获取或修改属性时，都有简写！

获取自定义属性值时: 元素.dataset.属性名

比如: img.dataset.md

强调: 只有data-前缀的属性才能用dataset属性访问到！

使用dataset.属性名方式访问自定义扩展属性时，不用加data-前缀！直接写属性名即可！

修改自定义属性值时: 元素.dataset.属性名=新值

比如: img.dataset.md=”img/md1.jpg”;

结果: md会被自动加上data-前缀: <img data-md=”img/md1.jpg”>

iv. 如何查找带有自定义扩展属性的元素: 用属性选择器查找！

document.querySelector(“[data-md=’img/md1.jpg’]”)

选择一个带有data-md属性且属性值为img/md1.jpg的元素

4). 示例: 标签页效果

|  |
| --- |
| <style>    \*{    margin:0;    padding: 0;  }  #tab li{    float: left; list-style: none;  }  #tab li a{    display:inline-block;    text-decoration:none;    width: 80px; height: 40px;    line-height: 40px;    text-align: center;    color:#000;  }  #container{    position: relative;  }  #content1,#content2,#content3{    width: 300px;    height: 100px;    padding:30px;    position: absolute;    top: 40px;    left: 0;  }  #tab li:first-child,#content1{    background-color: #ffcc00;  }  #tab li:first-child+li,#content2{    background-color: #ff00cc;  }  #tab li:first-child+li+li,#content3{    background-color: #00ccff;  }  </style>  </head>  <body>  <h2>实现多标签页效果</h2>  <div class="tabs">    <ul id="tab">      <!--         让a什么也不干-->      <li><a href="javascript:;" data-click="tab" data-div="content1">10元套餐</a></li>      <li><a href="javascript:;" data-click="tab" data-div="content2">30元套餐</a></li>      <li><a href="javascript:;" data-click="tab" data-div="content3">50元包月</a></li>    </ul>    <div id="container">      <div id="content1" style="z-index:10;">        10元套餐详情：<br />&nbsp;每月套餐内拨打100分钟，超出部分2毛/分钟      </div>      <div id="content2">        30元套餐详情：<br />&nbsp;每月套餐内拨打300分钟，超出部分1.5毛/分钟      </div>      <div id="content3">        50元包月详情：<br />&nbsp;每月无限量随心打      </div>    </div>  </div>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中: 查找id为tab的ul下的li下的带有data-click属性，且属性值为tab的三个按钮元素  var tabs=document.querySelectorAll("#tab>li>[data-click=tab]");  //附加一个计数的变量  var zIndex=10;//从10开始，每点一次标签，zIndex就+1，并将本次最大的zIndex赋值给当前div，就可保证当前div的zIndex暂时最大  //2. 绑定事件处理函数  //遍历tabs中每个tab，每遍历一个tab按钮就为其绑定单击事件  for(var tab of tabs){    tab.onclick=function(){      //3. 查找要修改的元素      //本例中: 从当前按钮自己身上取出对应div的id，用id去找自己对应的div      var id=this.getAttribute("data-div");            //this.dataset.div      var div=document.getElementById(id);//因为id是变量所以不加""      //4. 修改元素      //本例中: 让当前div在最上边，z-index最大！      //先将zIndex变量值+1      zIndex++;      //再将现在临时的最大zIndex赋值给div      div.style.zIndex=zIndex;    }  }  </script>  </body>  </html> |

(3). 样式:

a. 获取或修改内联样式:

1). 回顾: HTML中: <元素 style=”css属性:值; css属性:值; … ”

2). DOM中: 元素对象.style.css属性

执行时，DOM会将这句代码翻译为HTML中的<元素 style=”css属性:值;”>

3). 问题: 有些css属性名中带-，而程序中不能随意出现-，会个减号冲突

比如: background-color font-size background-image

错误: ~~元素.style.background-color~~

解决: 所有带-的css属性名必须去横线变驼峰(首字母小写，之后每个单词首字母大写)

比如: 元素.style.backgroundColor

总结:

1. 查找元素: 4种查找方式

(1). 不需要查找就可直接获得的元素: 4种

a. document

b. <html> document.documentElement

c. <head> document.head

d. <body> document.body

(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，要找周围附近的元素时

a. 父子关系:

1). 获得元素的父元素: 元素.parentNode

2). 获得元素的所有直接子元素: 元素.children

3). 获得元素的第一个直接子元素: 元素.firstElementChild

4). 获得元素的最后一个直接子元素: 元素.lastElementChild

b. 兄弟关系:

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

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

(3). 按HTML特征查找: 4种 ——首次查找效率高！首选

何时: 如果将来仅凭一个特征(id、标签名、class名、name名)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”)

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”)

c. 按class名查找多个元素: var 类数组对象=任意父元素.getElementsByClassName(“class名”)

d. 按name名查找多个表单元素: var 类数组对象=document.getElementsByName(“name名”)

(4). 按选择器查找: 2种——首次查找效率略低，如果条件复杂时，查找效率反而高！

何时: 如果查找条件复杂，就用按选择器查找

1. 如果只查找一个符合条件的元素时: var 一个元素=任意父元素.querySelector(“选择器”);
2. 如果查找多个符合条件的元素时: var 类数组对象=任意父元素.querySelectorAll(“选择器”);
3. 修改元素: 3种东西可修改

(1). 内容:3种:

a. 获取或修改原始HTML内容: 元素.innerHTML

b. 获取或修改纯文本内容: 元素.textContent

c. 获取或修改表单元素的值: 表单元素.value

(2). 属性: 3种:

a. 字符串类型的HTML标准属性: 2种:

1). 核心DOM4个函数:

i. 元素.getAttribute(“属性名”)

ii. 元素.setAttribute(“属性名”,”属性值”),

iii. 元素.hasAttribute(“属性名”)

iv. 元素.removeAttribute(“属性名”)

2). HTML DOM: 元素.属性名

特例: 元素.className

b. bool类型的HTML标准属性: 只能用HTML DOM的. : 元素.属性名 且属性值为bool类型

c. 自定义扩展属性

(3). 样式:

a. 获取或修改内联样式: 元素.style.css属性

总结: 函数返回值的规律:

1. 如果函数返回下标位置i，则找不到返回-1
2. 如果函数返回一个对象或数组，则找不到返回null
3. 如果函数返回类数组独享，则找不到返回空类数组对象{ length:0 }

总结: this

1. obj.fun() this->obj
2. fun() 或 (function(){ … })() 或 回调函数 this->window
3. new Fun() this->将来new正在创建的那个新子对象
4. 构造函数.prototype.方法=function(){ … } this->将来调用这个方法的某个子对象
5. 访问器属性中的this: this->访问器所在的对象本身

Object.defineProperties(eric,{

eage:{

get(){return this.\_eage},

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

}

})

1. 元素.onclick=function(){ … } this->当前正在触发事件的元素

作业:

1. (必须)看小程序视频学习带样式的表单验证

小程序->在线->DOM->day03: 作业: 带样式的表单验证 valiWithCss

1. (扩展)看小程序视频，修改开关门效果为过渡效果:

小程序->在线->DOM->day02 : 1. 动画 开关门效果…

3. (扩展)看小程序视频，修改伸缩二级菜单效果为过渡效果:

小程序->在线->DOM->day02: 2. 下拉菜单 手风琴效果 动画 过渡 高度不同