准备: 微信小程序: WEB问题速查 小恐龙图标

1. 历届师兄师姐在今天课程中曾经问过的高频的经典的问题清单和解答: 小程序->首页->DOM-> day02、03

自己可能没有发现问题，但是想知道别的同学都问过哪些问题

2. 今日课程对应的重点案例的视频以及鄙视面试所需的扩展知识视频：小程序->在线->DOM-> day02、03

如果因为某些原因错过直播讲解，不用着急！直接看小程序中视频的回看即可。

如果tmooc不能用了，立刻看小程序视频，小程序视频比tmooc要全，甚至有扩展的新知识。

如果学有余力，将来有能力拿更高薪资，小程序中所有视频必须都要看。但是不必现在都看完。只要在笔试面试前都看完即可！

问题: 个人版小程序有功能限制：禁止在线播放视频

解决: 长按视频连接地址，完整复制视频连接地址和验证码到浏览器打开。通过百度网盘下载后观看。

3. 今日课程中包含的英语单词的翻译：小程序->单词->DOM->day02、03

英文不太好的同学，可以提前看一下第二天需要用到哪些英文单词。

约定: 2件事:

1. 第三阶段禁止上课跟着敲笔记和敲代码！谁抄笔记抄的勤，谁抄代码抄的勤，谁学的越烂！

2. 每个完整功能的案例/视频，至少写三遍: (如果需要画图，必须亲手画图)

(1). 连代码+注释抄一遍 —— 明白这个例子是要做什么事儿

(2). 保留注释，自己试着看着注释，将代码填回来 —— 复习函数和对象

(3). 删除所有代码和注释，自己试着用自己的话，把注释写回来，再翻译为代码！—— 只有自己可以写出人话的注释，这个例子才算自己会！

如果没有问题的同学: （享受vip一对一答疑的时间还剩11天）

1. 上课时那句话没听懂，笔记中那句话看不懂，案例中那句话看不懂，立刻问！

2. 小程序问题清单中哪个问题或解答看不懂，立刻问！

3. 小程序视频中哪个视频看不懂，立刻问

4. 个人项目中哪个功能不会做，立刻问

5. 平时刷笔试题时，遇到不会的笔试题，立刻问

谁问的问题越多！谁的培训费交的越值！

正课:

1. 修改元素

2. 添加/删除元素

3. HTML DOM常用对象

4. BOM

一. 修改元素: 3种

1. 内容: 3种

2. 属性: 3种:

(1). 字符串类型的HTML标准属性: 2种:

(2). bool类型的HTML标准属性: 1种:

(3). 自定义扩展属性: 2种方式

a. 旧js: 元素.getAttribute("data-自定义属性名")

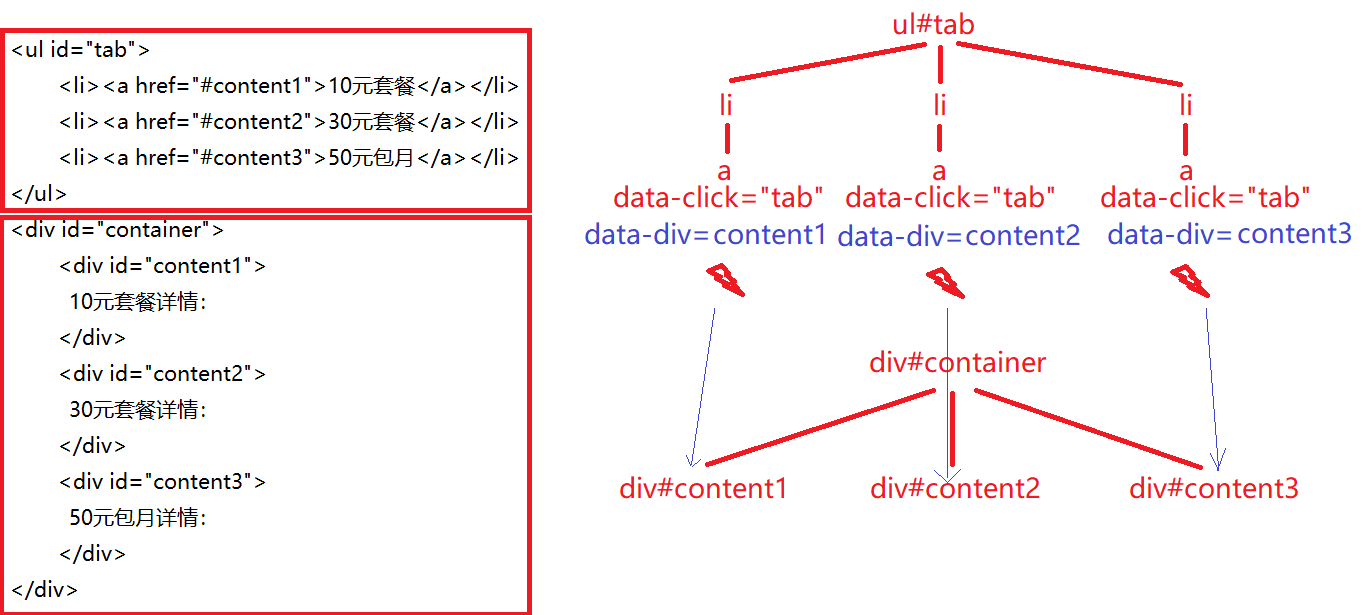
.setAttribute("data-自定义属性名","属性值")

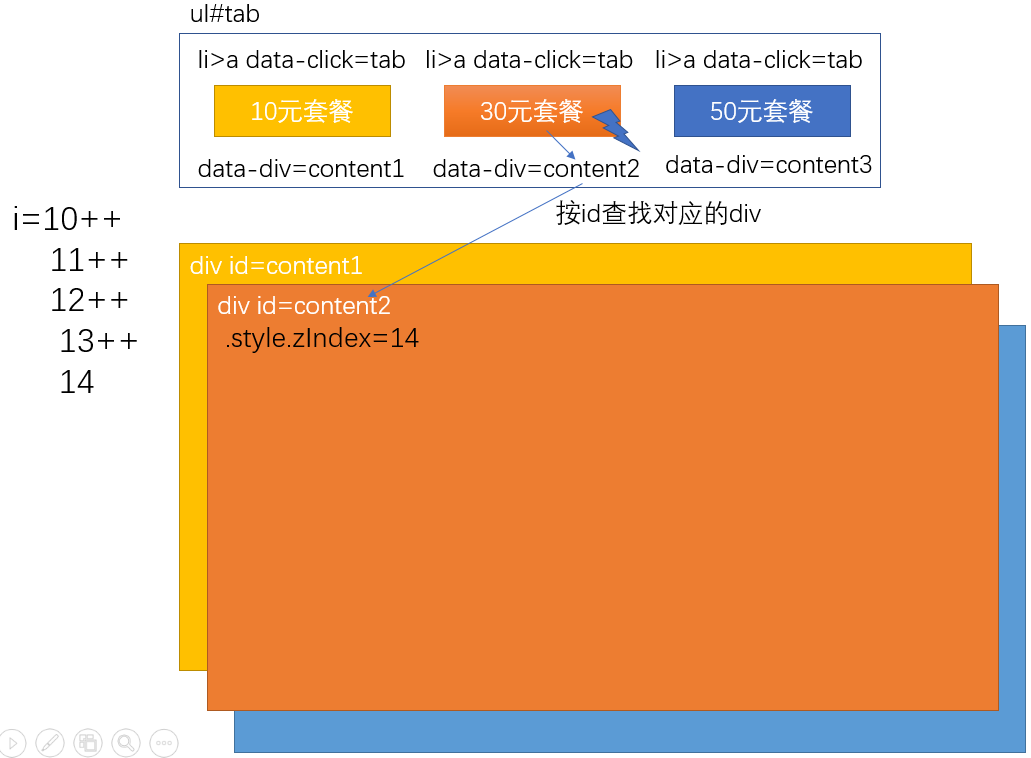
b. HTML5中: 元素.dataset.自定义属性名

c. 查找带有自定义扩展属性的元素: CSS 属性选择器: [data-自定义属性名=属性值]

(4). 示例: 标签页:

对应小程序视频: 小程序->在线->DOM->day02 4.2 标签页 tabs





day02 tabs/3\_tabs.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>读取并修改元素的属性</title>  <meta charset="utf-8" />  <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">  <li><a data-click="tab" data-div="content1" href="#content1">10元套餐</a></li>  <li><a data-click="tab" data-div="content2" href="#content2">30元套餐</a></li>  <li><a data-click="tab" data-div="content3" href="#content3">50元包月</a></li>  </ul>  <div id="container">  <div id="content1">  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: 用户点标签中每个按钮触发变化  //但是，实现按钮效果，不一定只能用a，还可能换成span、button等  //所以，应该为每个按钮添加自定义扩展属性，来摆脱元素名的影响。  //比如: 为每个按钮上都添加一个自定义属性: data-click="tab"  // 可以改名 可以改名  //所以，应该查找所有带有data-click属性，且属性值为tab的元素  var tabs=document.querySelectorAll("[data-click=tab]")  //2. 绑定事件处理函数  var i=10; //提前定义一个变量，累加zIndex数值  //本例中: 每个标签页按钮都可以单击  for(var tab of tabs){  tab.onclick=function(){  //3. 查找要修改的元素  //本例中: 每点一个标签页按钮，要修改当前按钮对应的div  //问题: 如何让每个按钮自己记住自己对应的哪个div呢？  //解决: 3.1为每个按钮添加自定义扩展属性: data-div=对应div的id  //3.2当单击一个按钮时，就可以获得这个按钮上保存的自己对应的div的id  var id=//this.getAttribute("data-div");//旧js  this.dataset.div;//HTML5  //3.3用id查找本次点击的按钮对应的div  var div=document.getElementById(id);//不要加"",因为id是变量  //4. 修改元素  //本例中: 设置当前div的z-index属性最大！  //问题1: js中不能写z-index，-会和减法的-号冲突  //解决: z-index改为zIndex  //问题2: 怎么保证本次这是的zIndex值最大？比之前的zIndex值都大！  //解决: 提前定义一个变量，保存一个数字，每使用一次，数字就+1，就可以保证最后一次使用的数字，一定比之前使用过的数字都大  i++;//先将i++  //再设置当前div的zIndex为新的i  div.style.zIndex=i;  }  }    </script>  </body>  </html>  运行结果: |

3. 样式:

对应小程序视频: 小程序->在线->DOM->day02 5 修改样式，计算属性

(1). 修改内联样式:

a. 元素.style.css属性="属性值"

b. 会被自动翻译为HTML中: <元素 style="css属性:属性值">

c. 问题: 有些css属性名中带-，但是js中不能随便写-，会和减法的-号冲突

d. 解决: 凡是带-的css属性名，在js中都要"去横线变驼峰"

比如: z-index -> zIndex

background-color -> backgroundColor

font-size -> fontSize

(2). 获取css属性值:

a. 坑: 因为.style属性，仅代表内联样式，所以获取时，也只能获取内联样式。但是实际开发中很少使用内联样式！尴尬: 明明知道这个元素上有很多css属性，但是用.style就是无法获得css属性值！

b. 解决: 今后如果想获得元素的所有css属性，应该使用计算后的样式

1). 什么是计算后的样式: 最终应用到这个元素上的所有css属性的总和

2). 如何: 2步

i. 先获得包含计算后的所有样式的对象:

var style=getComputedStyle(元素对象)

获得 计算后的 样式

可直接使用 不可改名

ii. 再用.方式从style中提取出想用的个别css属性值

style.css属性

c. 强调: 所有计算后的样式属性值都是只读的！不允许修改。因为计算后的样式属性来源不确定。很可能来自于外部样式表，同时被多个元素使用着。如果因为某一个元素擅自修改共用的css属性，会牵一发而动全身！——程序中最不希望看到的！

(3). 总结:

a. 今后只要获取一个元素的css属性值，都用getComputedStyle()

b. 今后只要修改一个元素的css属性值，都用.style

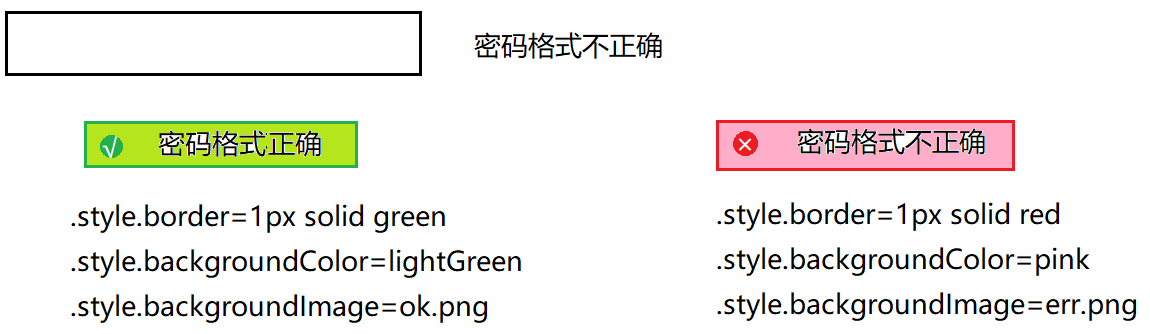
优点: 仅影响自己，不影响别人; 优先级最高，不会被其它属性覆盖！

(4). 示例: 使用getComputedStyle()获取元素的所有css属性值

0\_style\_getComputedStyle.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <style>  h1{  background-color:red;  }  </style>  </head>  <body>  <h1 id="h1" style="color:yellow">Welcome</h1>  <p>welcome to my web site</p>  <script>  var h1=document.getElementById("h1");  //想获得h1的字体颜色，背景颜色和字体大小  //错误：用h1.style方式  // console.log(h1.style.color);//内联  // console.log(h1.style.backgroundColor);//内部样式表  // console.log(h1.style.fontSize);//浏览器默认样式表  //正确: getComputedStyle(h1)  var style=getComputedStyle(h1);  console.log(style);//包含所有css属性,太大  console.log(style.color);//内联  console.log(style.backgroundColor);//内部样式表  console.log(style.fontSize);//浏览器默认样式表  //想通过计算后的样式对象style，修改某个css属性  style.fontSize="90px";//报错！  //Failed to set the 'font-size' property on 'CSSStyleDeclaration':  // 不能 设置 font-size 属性 在 xxx对象中  //These styles are computed, and therefore the 'font-size' property is read-only.  //这些 样式 是 计算后的 因此 font-size 属性  //是 只读的  </script>  </body>  </html>  运行结果:  CSSStyleDeclaration {0: "animation-delay", 1: "animation-direction", 2: "animation-duration", 3: "animation-fill-mode", 4: "animation-iteration-count", 5: "animation-name", …}  rgb(255, 255, 0)  rgb(255, 0, 0)  32px  Uncaught DOMException: Failed to set the 'font-size' property on 'CSSStyleDeclaration': These styles are computed, and therefore the 'font-size' property is read-only.  at http://127.0.0.1:5500/3\_style\_getComputedStyle.html:30:21 |

(5). 问题: .style.css属性，一句话只能修改一个css属性。而网页中很多效果，需要同时修改一个元素的多个css属性才能完整。用.stye.css属性做效果，代码会很繁琐！



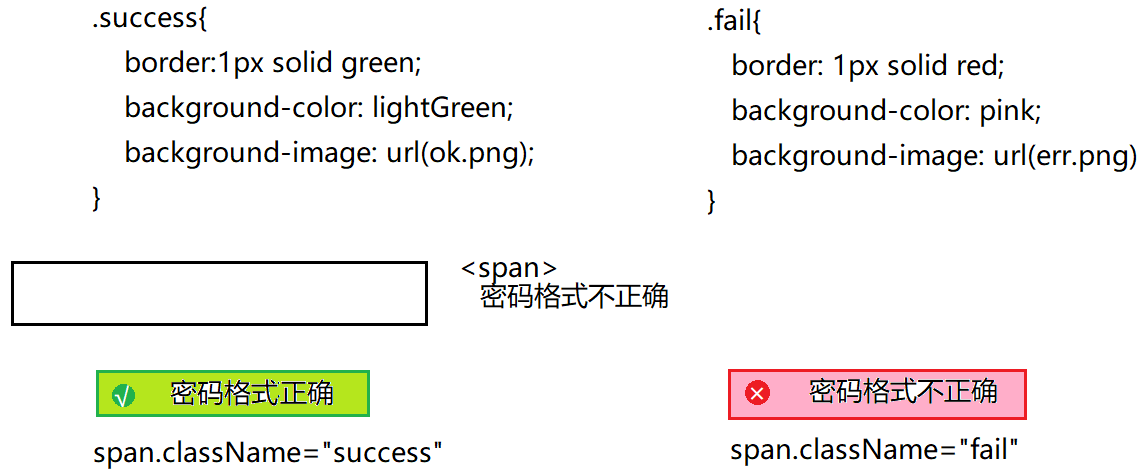
(6). 解决: 今后只要批量修改元素的多个css属性时，都用class代替.style方式

a. 先在css中定义好要使用的class，包含不同情况下的成套的css属性

b. 在js中，用程序根据不同的情况，判断该应用哪种class

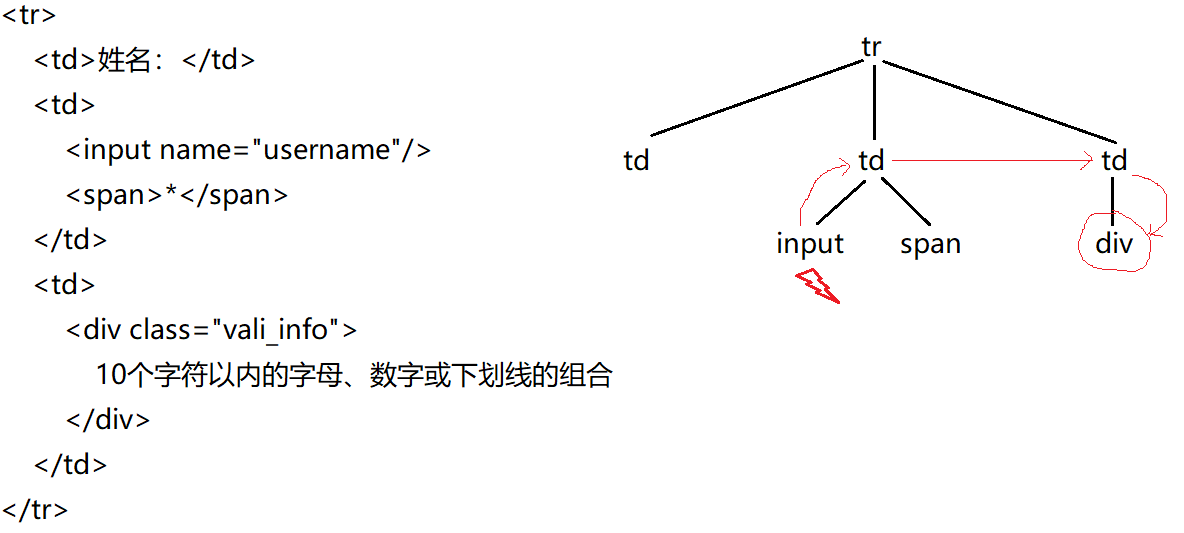
c. 优点: 无论要批量修改几个css属性，js语句永远只需要一句话:

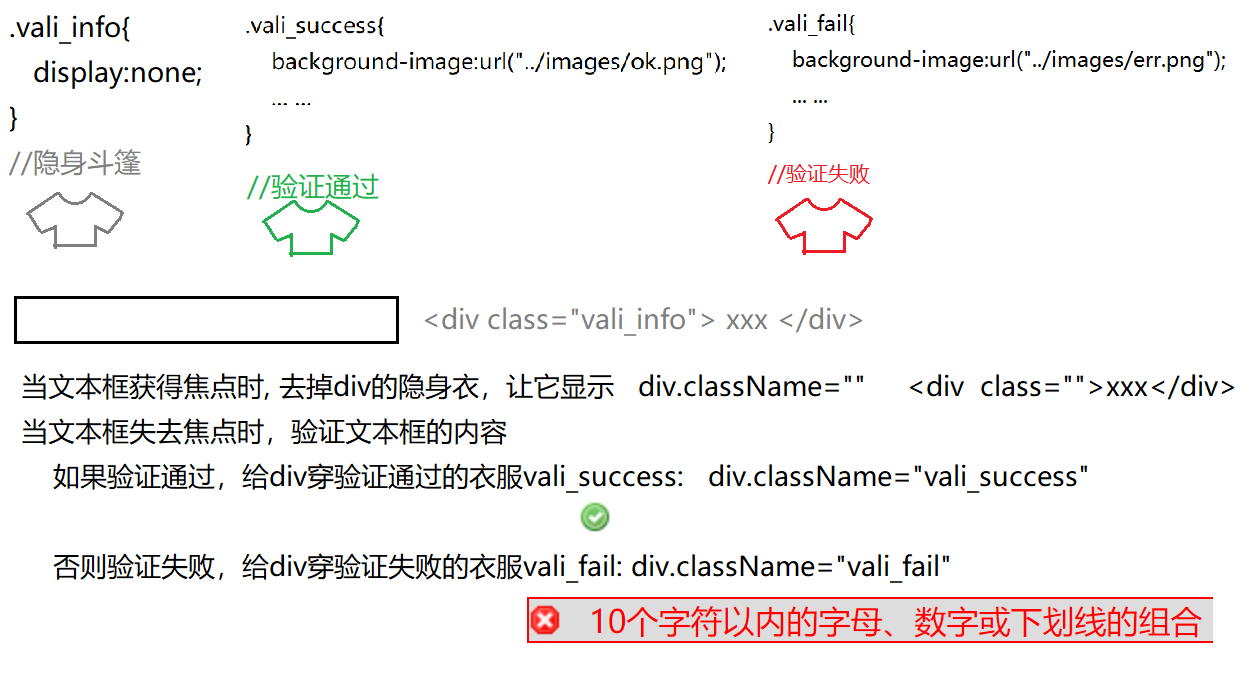
元素.className="class名"



(7). 示例: 带样式的表单验证:

对应小程序视频: 小程序->在线->DOM->day02 6. 带class样式的表单验证





0\_valiWithCss.html

|  |
| --- |
| <!doctype html>  <html>  <head>  <meta charset="UTF-8">  <title>实现带样式的表单验证</title>  <style>  table{width:700px}  td:first-child{width:60px}  td:nth-child(2){width:200px}  td:first-child+td{width:200px}  td span{color:red}  .vali\_info{  display:none;  }  .txt\_focus{  border-top:2px solid black;  border-left:2px solid black;  }  .vali\_success,.vali\_fail{  background-repeat:no-repeat;  background-position:left center;  display:block;  }  .vali\_success{  background-image:url("../images/ok.png");  padding-left:20px;  width:0px;height:20px;  overflow:hidden;  }  .vali\_fail{  background-image:url("../images/err.png");  border:1px solid red;  background-color:#ddd;  color:Red;  padding-left:30px;  }  </style>  </head>  <body>  <form id="form1">  <h2>增加管理员</h2>  <table>  <tr>  <td>姓名：</td>  <td>  <input name="username"/>  <span>\*</span>  </td>  <td>  <div class="vali\_info">  10个字符以内的字母、数字或下划线的组合  </div>  </td>  </tr>  <tr>  <td>密码：</td>  <td>  <input type="password" name="pwd"/>  <span>\*</span>  </td>  <td>  <div class="vali\_info">6位数字</div>  </td>  </tr>  <tr>  <td></td>  <td colspan="2">  <input type="submit" value="保存"/>  <input type="reset" value="重填"/>  </td>  </tr>  </table>  </form>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中: 用户操作姓名文本框时会触发变化  var txtName=document.getElementsByName("username")[0];  //[0]: 从查询结果的类数组对象中取出找到的唯一的文本框元素对象  console.log(txtName);  //2. 绑定事件处理函数  //本例中: 当文本框获得焦点时，旁边的提示框，显示出来  txtName.onfocus=function(){  //3. 查找要修改的元素  //本例中: 当文本款获得焦点时，应该查找文本框的爹的下一个兄弟td中的div  var div=this.parentElement.nextElementSibling.children[0];  //4. 修改元素  //本例中: 当文本款获得焦点时，让旁边的div显示,脱掉div的隐身衣class  div.className="";  }  //2. 绑定事件处理函数  //本例中: 当文本框失去焦点时，就要验证文本框中用户输入的内容，并根据验证结果修改旁边提示框的样式类  txtName.onblur=function(){  //3. 查找要修改的元素  //本例中：当文本框失去焦点时，还是要修改文本框旁边的div的class  var div=this.parentElement.nextElementSibling.children[0];  //4. 修改元素  //先定义正则表达式来规定姓名的格式  var reg=/^\w{1,10}$/;  //验证当前文本框的内容是否符合格式要求  var bool=reg.test(this.value);  //如果验证通过  if(bool==true){  //就修改div的className为vali\_success  div.className="vali\_success";  }else{//否则如果验证不通过  //就修改div的className为vali\_fail  div.className="vali\_fail";  }  }  </script>  </body>  </html>  运行结果: |

二. 添加删除元素:

1. 添加一个新元素到网页:3步

(1). 创建一个新的空元素对象:

a. var 一个新元素=document.createElement("标签名")

创建 元素

b. 比如: 想创建一个空的a元素:

var a=document.createElement("a");

结果: a : <a></a>

(2). 为新元素对象添加必要属性:

a. 比如: 如果想让这个a可以当做超链接使用，至少得设置href属性和内容

a.href="http://tmooc.cn";

a.innerHTML="go to tmooc";

结果: <a href="http://tmooc.cn"> go to tmooc </a>

b. 问题: 这个a这是在内存中游离的a元素对象。还没有挂载到dom树上，所以浏览器不知道应该把这个a显示在具体哪个位置上。所以这个a，用户暂时是看不见的！

(3). 将新元素添加到DOM树上指定父元素下: 3种:

a. 追加到一个指定父元素下所有直接子元素之后:

父元素.appendChild(新元素)

追加 子元素

b. 插入到一个指定父元素下的一个现有子元素之前:

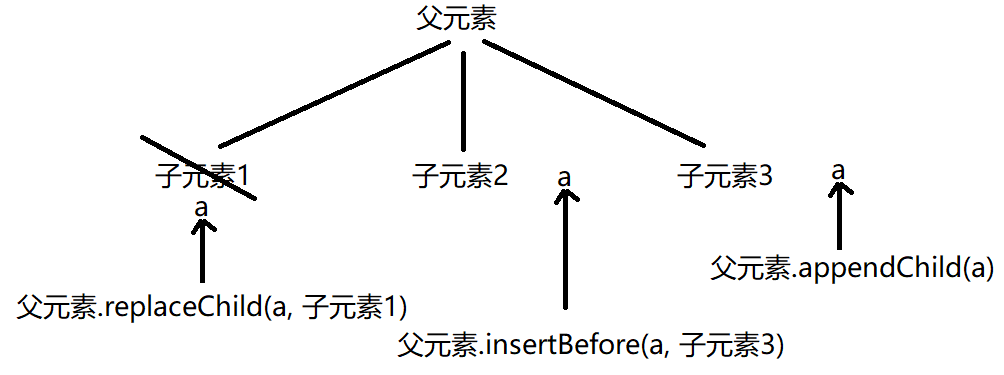
父元素.insertBefore(新元素, 现有子元素)

插入...之前

c. 替换一个指定父元素下的一个现有子元素:

父元素.replaceChild(新元素, 现有子元素)

替换 子元素



2. 示例: 向网页中添加一个新的a元素，再添加一个input元素，让input元素出现在a的指定位置上

1\_createElement.html

|  |
| --- |
| <script>  //想在body中添加一个a元素  var a=document.createElement("a");  a.href="http://tmooc.cn";  a.innerHTML="go to tmooc";  console.log(a);  document.body.appendChild(a);  //想创建一个input元素，放在a元素之前 / 之后 / 替换a元素  var input=document.createElement("input");  //input不需要设置任何属性，默认就是一个文本框  document.body.insertBefore(input,a);  //.appendChild(input);  //.replaceChild(input,a);  </script>  运行结果: |

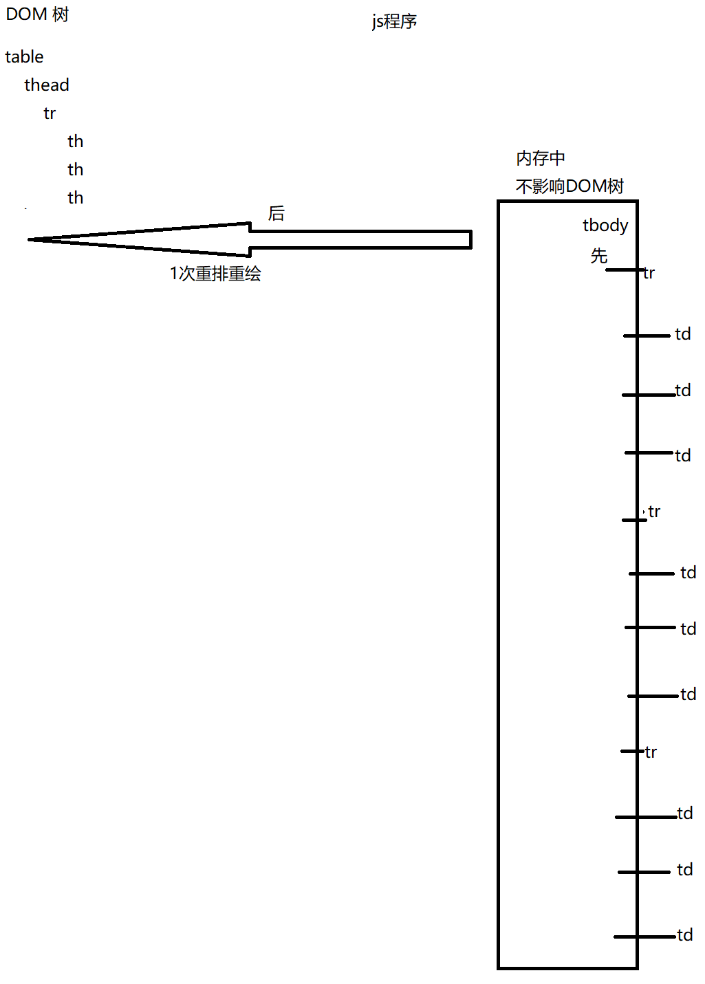
3. 优化:

(1). 问题: 浏览器显示网页的内容是经过先排版，再绘制的过程才显示出来的。只要js程序一修改DOM树，浏览器都被迫要重新对页面内容进行排版和绘制。过程非常的复杂和耗时。

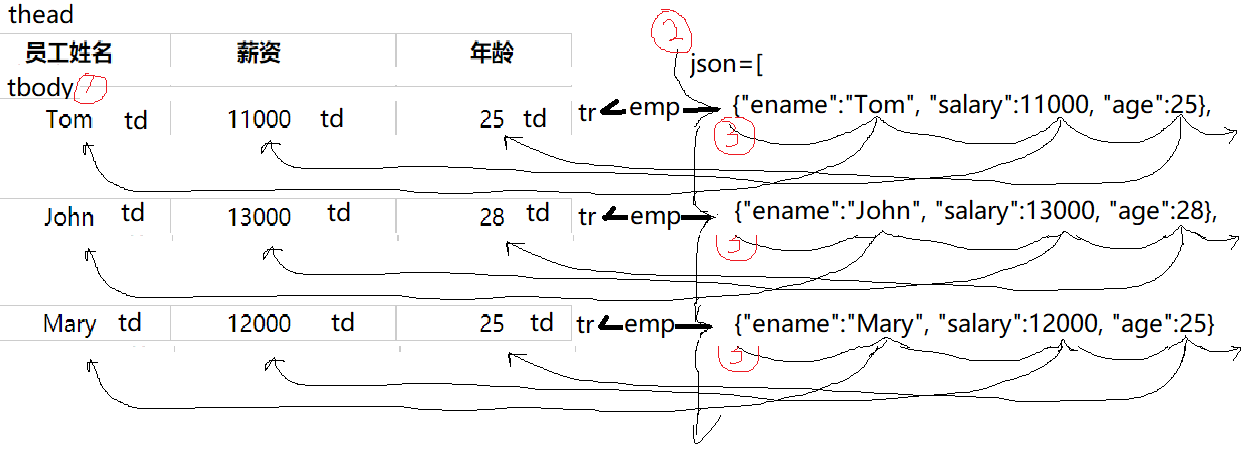
(2). 优化: 尽量减少操作DOM树的次数！

(3). 如何: 2种:

a. 如果同时添加父元素和子元素时，应该现在内存中将所有子元素添加到父元素中，最后再一次性将父元素添加到DOM树——一次重排重绘



1). 示例: 使用优化办法动态生成表格内容(tbody不在table上，也是临时创建的)



1\_createTable.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>动态创建表格</title>  <meta charset="utf-8" />  <style>  table{width:600px; border-collapse:collapse;  text-align:center;  }  td,th{border:1px solid #ccc}  </style>  </head>  <body>  <div id="data">  <table>  <thead>  <tr>  <th>员工姓名</th>  <th>薪资</th>  <th>员工年龄</th>  </tr>  </thead>  </table>  </div>  <script>  var json=[  {"ename":"Tom", "salary":11000, "age":25},  {"ename":"John", "salary":13000, "age":28},  {"ename":"Mary", "salary":12000, "age":25}  ];  //先创建一个tbody元素  var tbody=document.createElement("tbody");  //暂时不要将tbody元素追加到table中  //遍历json数组中每个员工对象  for(var emp of json){  //每遍历一个员工对象，就创建一个tr，将tr追加到tbody下  var tr=document.createElement("tr");  tbody.appendChild(tr);  //遍历当前emp对象中每个属性值  for(var key in emp){  //每遍历一个属性值，就创建一个td，并设置td的内容为当前属性值，最后将td追加到当前tr对象中  var td=document.createElement("td");  td.innerHTML=emp[key];//因为key是变量，所以不加""  tr.appendChild(td);  }  }  //等tbody中所有内容都加载完了，最后一次性将整个tbody追加到DOM树  //查找table对象  var table=document.querySelector("#data>table");  table.appendChild(tbody);  //浏览器只需要一次重排重绘  //对效果没有影响，却极大的优化了页面加载效率！  </script>  </body>  </html>  运行结果: |

b. 如果父元素已经在页面上了，要同时添加多个平级子元素，就要借助于一种新对象——文档片段对象

1). 什么是文档片段对象: 内存中临时保存多个平级子元素的虚拟父元素对象

2). 何时: 只要父元素已经在页面上了，要同时添加多个平级子元素，都要用文档片段对象

3). 如何: 3步:

i. 创建一个空的文档片段对象：

var frag=document.createDocumentFragment()

创建 文档 片段

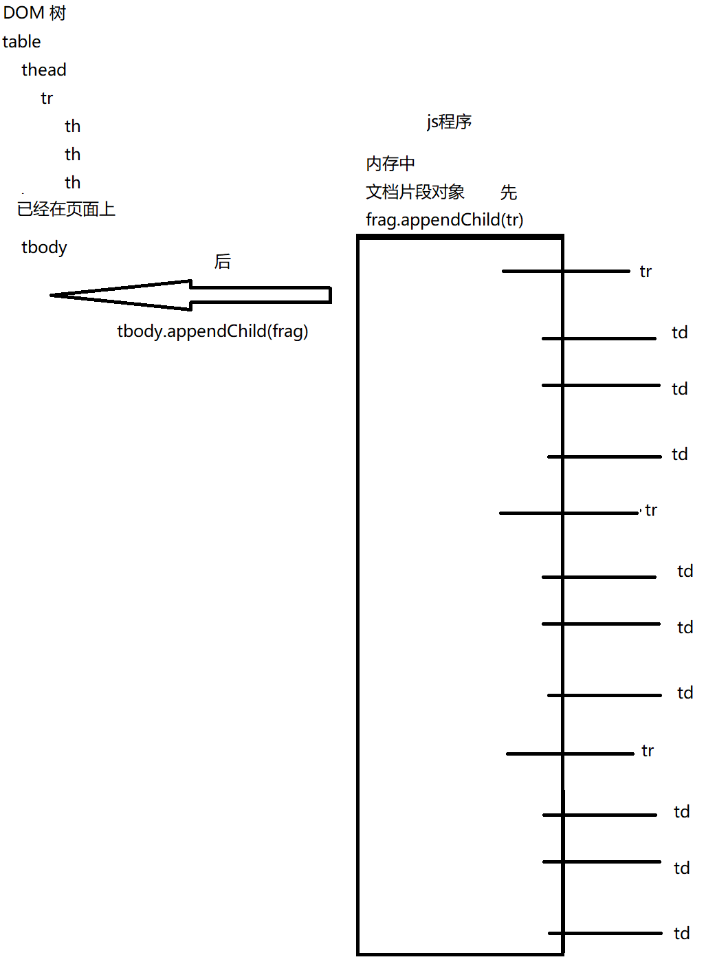
ii. 先将所有平级子元素临时加入到文档片段对象中暂存

frag.appendChild(子元素)

iii. 最后再将整个文档片段对象，一次性添加到DOM树上指定父元素下

父元素.appendChild(frag)

4). 结果: frag对象将平级子元素送到页面上指定父元素下之后，frag对象就释放了。不会成为页面上一级真正的元素！不影响页面元素的结构。



5). 示例: 使用文档片段动态生成表格内容(tbody已经在页面上了)

1\_createTable\_fragment.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>动态创建表格</title>  <meta charset="utf-8" />  <style>  table{width:600px; border-collapse:collapse;  text-align:center;  }  td,th{border:1px solid #ccc}  </style>  </head>  <body>  <div id="data">  <table>  <thead>  <tr>  <th>员工姓名</th>  <th>薪资</th>  <th>员工年龄</th>  </tr>  </thead>  <tbody>  </tbody>  </table>  </div>  <script>  var json=[  {"ename":"Tom", "salary":11000, "age":25},  {"ename":"John", "salary":13000, "age":28},  {"ename":"Mary", "salary":12000, "age":25}  ];  //因为tbody已经在页面上了  //所以，只能先创建一个临时的托盘——文档片段对象  var frag=document.createDocumentFragment();  //遍历json数组中每个员工对象  for(var emp of json){  //每遍历一个员工对象，就创建一个tr  var tr=document.createElement("tr");  //但是，不能将tr直接追加到tbody下  //而是，先将tr临时追加到文档片段对象托盘中暂存  frag.appendChild(tr);  //遍历当前emp对象中每个属性值  for(var key in emp){  //每遍历一个属性值，就创建一个td，并设置td的内容为当前属性值，最后将td追加到当前tr对象中  var td=document.createElement("td");  td.innerHTML=emp[key];//因为key是变量，所以不加""  tr.appendChild(td);  }  }  //最后,再一次性的将文档片段托盘添加到tbody中  var tbody=document.querySelector("#data>table>tbody");  tbody.appendChild(frag);  //查看以下tbody的孩子中有没有一个frag元素对象  console.log(tbody.children);  </script>  </body>  </html>  运行结果: |

4. 删除元素: 父元素.removeChild(要删除的元素)

三. HTML DOM常用对象（了解）: HTML DOM标准对个别常用的复杂的元素对象提供了简化的函数来操作这些对象。但是，不是所有元素都有这种简化。

1. <img>元素:

(1). 创建:

a. 旧DOM: var img=document.createElement("img")

b. HTML DOM: var img=new Image();

(2). 强调: 其实只有<img>和<option>两种元素可以new创建。其余元素都不能用new!

2. <table>元素: table是一个大家庭，从table到td，包含四级元素。所以，table采用逐级管理方式

(1). table管着行分组:

a. 添加行分组:

1). 旧DOM: 2步:

var 行分组=document.createElement("标签名")

table.appendChild(行分组);

2). HTML DOM: 1步，既创建行分组对象，又将行分组对象立刻添加到table上

var thead=table.createTHead()

var tbody=table.createTBody()

var tfoot=table.createTFoot();

3). 问题: 如果table已经在页面上了，就不应该用这种方式，过早的将空的行分组添加到table。结果，今后只要向行分组中添加任何东西都会导致重排重绘

b. 删除行分组:

table.deleteTHead()

table.deleteTFoot()

~~没有table.deleteTBody()~~

c. 获取行分组:

table.tHead 因为table中只能有一个thead，直接隶属于table

table.tFoot 因为table中只能有一个tfoot，直接隶属于table

table.tBodies[i] 因为HTML规定table下可以有多个tbody，所以多个tbody是放在一个集合中保存的。并不直接隶属于table

(2). 行分组管着行

a. 添加行:

1). 旧js中: 2步

var tr=document.createElement("tr");

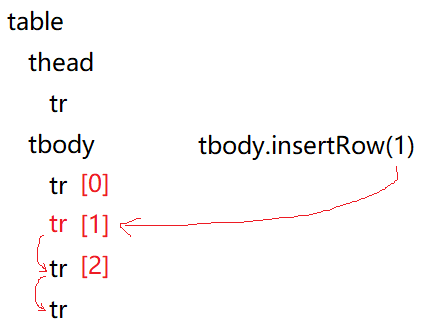
行分组.appendChild(tr)

2). HTML DOM中: 1步

i. var tr=行分组.insertRow(i)

ii. 意为: 先创建一个新的空行，然后再将新的空行插入到当前行分组中i位置。

iii. 强调: 如果i位置已经有行了，则新行不会替换i位置的行，而是将原i位置的行向后挤，新行占据i位置。



iv. 固定套路:

①在开头插入新行: var tr=行分组.insertRow(0)

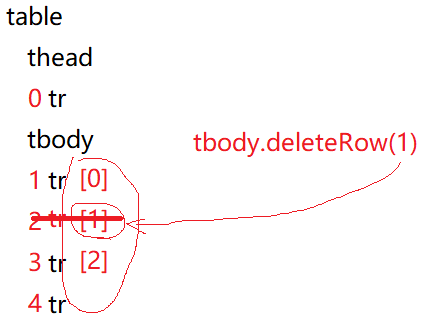
②在末尾追加新行: var tr=行分组.insertRow()

b. 删除行:

1). 行分组.deleteRow(i)

2). 意为: 删除行分组中i位置的行

3). 强调: 参数i，是在行分组内的相对下标位置，而不是行在整个表中的下标位置



4). 问题: 如果要删的行号，不确定，又不容易用肉眼看出行号来，deleteRow()行号参数应该写几?

5). 解决: 每个行对象都有一个tr.rowIndex属性，记录了当前行在整个表中的下标位置。所以，tbody.deleteRow(tr.rowIndex)，自动获得要删除行下标。

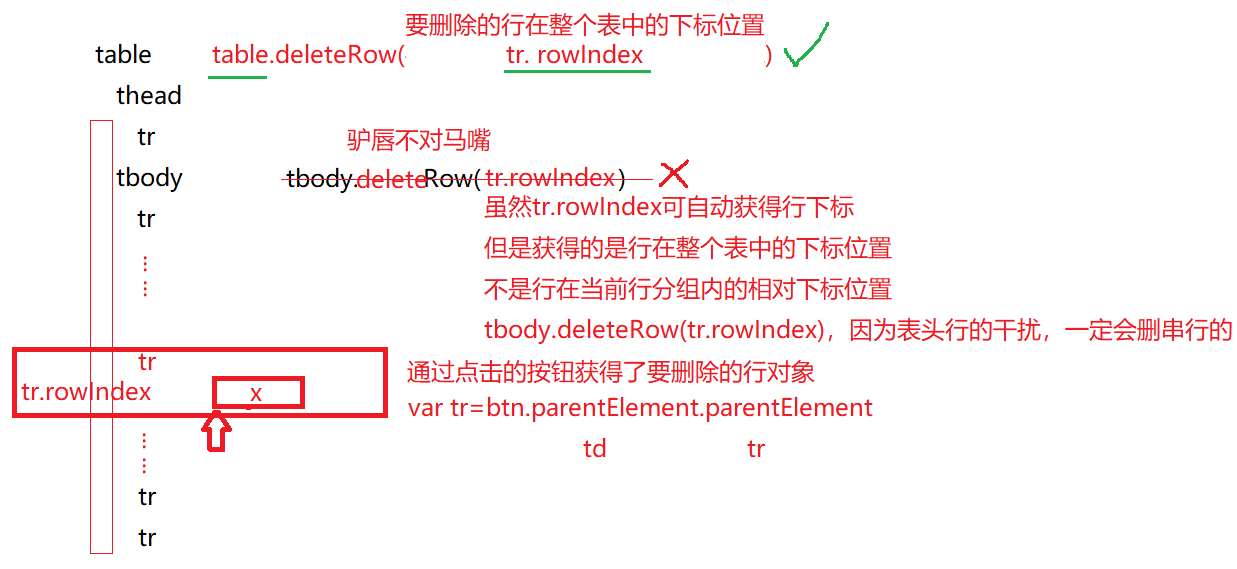
6). 问题: tbody要求行号是在tbody内的相对下标，但是tr.rowIndex自动获得只能是行在整个表中的下标位置。两者不一致！如果用tbody.deleteRow(tr.rowIndex)删除行，因为受到表头行的干扰，几乎一定会删串行/错位！

7). 终极删除行的最正确解决方案: table.deleteRow(tr.rowIndex)

因为table要的是行在整个表中的下标位置

tr.rowIndex给的刚好也是行在整个表中的下标位置

两者刚好配对！



c. 获取行: 行分组.rows[i] //获取当前行分组中i位置的行对象

(3). 行管着格:

a. 添加格: var td=tr.insertCell() //在行末尾追加一格

b. 删除格: tr.deleteCell(i) //删除i位置的一个格

c. 获取格: tr.cells[i] //获取行中i位置的格对象

(4). 示例: 动态创建表格，带删除行功能

对应小程序视频: 小程序->在线->DOM->day03 1 DOM添加新元素...

1\_createTable\_HTMLDOM.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>动态创建表格</title>  <meta charset="utf-8" />  <style>  table{width:600px; border-collapse:collapse;  text-align:center;  }  td,th{border:1px solid #ccc}  </style>  </head>  <body>  <div id="data">  <table>  <thead>  <tr>  <th>员工姓名</th>  <th>薪资</th>  <th>员工年龄</th>  <th>删除</th>  </tr>  </thead>  </table>  </div>  <script>  var json=[  {"ename":"Tom", "salary":11000, "age":25},  {"ename":"John", "salary":13000, "age":28},  {"ename":"Mary", "salary":12000, "age":25}  ];  //先创建一个tbody元素  var tbody=document.createElement("tbody");  //暂时不要将tbody元素追加到table中  //查找table对象  var table=document.querySelector("#data>table");  //遍历json数组中每个员工对象  for(var emp of json){  //每遍历一个员工对象，就创建一个tr，将tr追加到tbody下  // var tr=document.createElement("tr");  // tbody.appendChild(tr);  var tr=tbody.insertRow();//末尾追加新行  //tbody.insertRow(0); //始终在开头插入新行  //遍历当前emp对象中每个属性值  for(var key in emp){  //每遍历一个属性值，就创建一个td，并设置td的内容为当前属性值，最后将td追加到当前tr对象中  // var td=document.createElement("td");  // tr.appendChild(td);  var td=tr.insertCell();  td.innerHTML=emp[key];//因为key是变量，所以不加""  }  //当为当前行添加完数据格之后，再额外添加一个新格  var td=tr.insertCell();  //创建一个新按钮，并将新函数追加到td内容中  var btn=document.createElement("button");  btn.innerHTML="×";  td.appendChild(btn);  //为按钮绑定单击事件处理函数  btn.onclick=function(){  //测试:点哪个按钮，让当前按钮变成❀  //this.innerHTML="❀";  //查找要修改的元素  //本例中: 点按钮要删除当前按钮所在的行  var tr=this.parentElement.parentElement;  // btn td tr  //修改元素  //删除这类操作，都是危险操作，所以在正式删除前都要再跟用户确认一下是否真的继续删除！  //如何跟用户确认？弹出确认框，请用户选择是否继续  //为了让提示更有针对性，对用户更有帮助，应该加入标志性的员工姓名  //先获得当前行中第一个格里保存的人名  var ename=tr.cells[0].innerHTML;  // td td的内容  var result=confirm(`是否继续删除 ${ename} 吗？`);  //如果用户点的是确认按钮，确认框整体返回true，说明用户想继续删除行  if(result==true){  //本例中: 最终要执行的  table.deleteRow(tr.rowIndex);  //因为前边已经查找过table元素了，所以table可以直接使用！  }//否则如果用户点的是取消，确认框整体返回false，条件不满足，什么也不干！  }  }  //等tbody中所有内容都加载完了，最后一次性将整个tbody追加到DOM树  table.appendChild(tbody);  //浏览器只需要一次重排重绘  //对效果没有影响，却极大的优化了页面加载效率！  </script>  </body>  </html>  运行结果: |

补: 浏览器三大对话框:

1. prompt: 输入框

2. alert: 警告框 只有一个确定按钮，没得选择！

3. confirm: 确认框 有两个按钮: 确认和取消。有的选择。

确认框有返回值

如果用户点确认按钮，则确认框返回true，说明用户同意继续

如果用户点取消按钮，则确认框返回false，说明用户不同意继续

总结: DOM操作万能4步:

1. 查找触发事件的元素

2. 为找到的元素绑定事件处理函数

3. 查找要修改的元素

4. 修改元素

总结: DOM 不用背！！！用的时候来找到即可！

1. 查找元素: 4种:

(1). 不需要查找就可直接获得的元素对象: 4个

a. document 根节点对象

b. document.documentElement <html>

c. document.head <head>

d. document.body <body>

(2). 按节点间关系查找: 如果已经获得一个元素，找周围附近的元素时，2大类关系，6个属性

a. 父子关系: 4个属性:

1). 元素.parentElement  
 2). 元素.children

3). 元素.firstElementChild

4). 元素.lastElementChild

b. 兄弟关系: 2个属性

1). 元素.previousElementSibling

2). 元素.nextElementSibling

(3). 按HTML特征查找元素: 4个函数

a. var 一个元素对象=document.getElementById("id名")

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

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

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

(4). 按选择器查找元素: 2个函数

a. 只查找一个符合要求的元素：

var 一个元素对象=任意父元素.querySelector("选择器");

b. 查找多个符合要求的元素

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

|  |
| --- |
| 总结: 查找方法的返回值:  1. 如果原函数返回下标位置i，则如果找不到，返回-1  2. 如果原函数返回一个对象或数组，则如果找不到，返回null  3. 如果原函数返回一个类数组对象，则如果找不到，返回空类数组对象{ length:0 } |

2. 修改: 3种

(1). 内容: 3种

a. 绝大多数情况下，用.innerHTML就够了！

b. 除非在获取内容时，确实希望去掉内嵌标签，只要文本时，才用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). 旧核心DOM 4个函数

i. 元素.getAttribute("属性名")

ii. 元素.setAttribute("属性名","属性值")

iii. 元素.hasAttrbiute("属性名")

iv. 元素.removeAttribute("属性名")

2). 新HTML DOM 简写:

i. 元素.属性名

ii. 元素.属性名="属性值"

iii. 元素.属性名!==""

iv. 元素.属性名=""

b. bool类型的HTML标准属性: 只能用 "元素.属性名" 操作，~~不能用核心DOM4个函数~~

c. 自定义扩展属性: HTML5标准

1). 添加/修改自定义扩展属性:

i. HTML中: <元素 data-自定义属性名="属性值"

ii. 旧js中: ~~不能用.,~~ 只能用 元素.setAttribute("data-自定义属性名","属性值")

iii. HTML5标准: 元素.dataset.自定义属性名=属性值

2). 获取自定义扩展属性值:

i. 旧js中: ~~不能用.,~~ 只能用 元素.getAttribute("data-自定义属性名")

ii. HTML5标准: 元素.dataset.自定义属性名

3). 查找带有自定义扩展属性的元素: 属性选择器 [data-自定义属性=属性值]

(3). 样式:

a. 只修改一个css属性时: 元素.style.css属性="属性值"

b. 批量修改多个css属性时: 元素.className="class名"

c. 获取css属性值时: var style=getComputedStyle(元素对象)

style.css属性名

3. 添加元素:

(1). 只添加一个元素: 3步

a. 创建一个空元素

var 一个元素=document.createElement("元素名")

b. 为元素设置关键属性

新元素.属性名=属性值

c. 将新元素添加到指定的父元素下: 3种:

1). 末尾追加: 父元素.appendChild(新元素)

2). 插入到一个元素前: 父元素.insertBefore(新元素, 现有元素)

3). 替换现有元素: 父元素.replaceChild(新元素, 现有元素)

(2). 优化: 如果添加多个元素，尽量减少操作DOM树的次数

a. 如果同时添加父元素和子元素，应该先将子元素添加到父元素，最后再一次性将父元素添加到DOM树

b. 如果父元素已经在页面上，需要同时添加多个平级子元素，应该用文档片段: 3步

1). 先创建文档片段对象

var frag=document.createDocumentFragment();

2). 先将多个平级子元素加入到文档片段中

frag.appendChild(子元素)

3). 最后将文档片段添加到DOM树

父元素.appendChild(frag)

4. 删除元素: 父元素.removeChild(要删除的元素)

5. 事件绑定:

元素对象.onclick=function(){

this -> 当前正在触发事件的元素对象

}

总结: this 5种:

1. obj.fun() this->obj

2. new Fun() this->new正在创建的子对象

3. 类型名.prototype.共有方法=function(){ ... } this->将来调用这个共有方法的.前的子对象

4. fun() 和 (function(){ })() 和回调函数中的this->window

5. 事件处理函数中的this->当前正在触发事件的这个元素对象

今日对应小程序视频列表:

小程序->在线->DOM->day02 4.2 标签页

5 修改样式，计算属性

6 带class样式的表单验证

小程序->在线->DOM->day03 1. DOM添加新元素 3步...

作业:

1.复习今日小程序问题清单(小程序->在线->DOM->day02、03)，预习明日小程序问题清单(小程序->在线->DOM->day04)

2. 看小程序视频学习如何完成并优化带样式的表单验证

小程序->在线->DOM->day02 6 带class样式的表单验证

直接在day03/0\_valiWithCss.html里做

3. 看小程序视频学习如何实现倒计时关闭窗口

小程序->在线->DOM->day03 作业: 倒计时关闭窗口

4. 看小程序视频学习DOM版学子商城项目

小程序->在线->DOM->day04 扩展: DOM版学子商城...

家里环境xampp和nodejs能用，且第一阶段学的好的同学:

下载xzserver\_start.zip，看视频中day01，跟着做

家里环境xampp和nodejs不能用，或者第一阶段学的不好的同学：

只下载public.zip，看视频中day01，跟着做

然后，所有网页用live server运行

视频中所有ajax请求的服务器端接口地址，都改为我的新浪云服务器地址，就不需要你再运行服务器端了。也不需要你了解任何服务器端知识了:

/index 改为 http://xzserver.applinzi.com/index

/details 改为 http://xzserver.applinzi.com/details?lid=xxx

5. (学有余力!!!!!!)看小程序视频学习如何实现级联下拉列表

小程序->在线->DOM->day03 2. 选择省份和城市...

直接在day03/2\_select\_options.html上做