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->day01、02

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-> day01、02

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->day01、02

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12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. 查找元素

2. 修改元素

3. 添加/删除元素

一. 查找元素: 4种:

1. 不需要查找就可直接获得的元素: 4个

2. 按节点间关系查找: 2大类关系，6个属性

3. 按HTML特征查找: 4个函数

4. 按选择器查找:

(1). 问题: 如果一个元素藏的很深，查找条件比较复杂，则使用按节点间关系查找或按HTML特征查找，步骤都会很繁琐！

(2). 解决: 因为CSS选择器在查找复杂元素时的语法非常简洁，所以DOM也支持按选择器查找元素

(3). 如何: 2个函数

a. 如果只查找一个符合条件的元素：

var 一个元素对象=任意父元素.querySelector("任意复杂的选择器")

b. 如果查找多个符合条件的元素:

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

5. 示例：纯客户端版购物车

对应小程序视频: 小程序->在线->DOM->day01 3. 购物车 shoppingcart

day01 购物车/3\_shoppingCart.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>使用Selector API实现购物车客户端计算</title>  <meta charset="utf-8" />  <style>  table{width:600px; text-align:center;  border-collapse:collapse;  }  td,th{border:1px solid black}  td[colspan="3"]{text-align:right;}  /\*想让table下tfoot下最后一个td背景变为粉色\*/  table>tfoot td:last-child{  background-color:pink;  }  /\*想让table下tbody下每行最后一个td背景变为黄色\*/  table>tbody td:last-child{  background-color:yellow;  }  </style>  </head>  <body>  <table id="data">  <thead>  <tr>  <th>商品名称</th>  <th>单价</th>  <th>数量</th>  <th>小计</th>  </tr>  </thead>  <tbody>  <tr>  <td>iPhone6</td>  <td>¥4488.00</td>  <td>  <button>-</button>  <span>1</span>  <button>+</button>  </td>  <td>¥4488.00</td>  </tr>  <tr>  <td>iPhone6 plus</td>  <td>¥5288.00</td>  <td>  <button>-</button>  <span>1</span>  <button>+</button>  </td>  <td>¥5288.00</td>  </tr>  <tr>  <td>iPad Air 2</td>  <td>¥4288.00</td>  <td>  <button>-</button>  <span>1</span>  <button>+</button>  </td>  <td>¥4288.00</td>  </tr>  </tbody>  <tfoot>  <tr>  <td colspan="3">Total: </td>  <td>¥14064.00</td>  </tr>  </tfoot>  </table>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中: 用户点击table中每个button,页面会发生变化,所以，应该先查找table，然后再查找table下所有button元素  //先找id为data的table  var table=document.getElementById("data");  console.log(table);  //再在table下查找button元素  var btns=table.getElementsByTagName("button");  console.log(btns);  //2. 绑定事件处理函数  //本例中: 因为6个按钮都可单击，所以，应该给每个按钮都绑定一个单击事件处理函数  //遍历类数组对象btns中每个按钮元素对象  for(var btn of btns){  //每遍历一个按钮，就给当前按钮绑定单击事件的处理函数  btn.onclick=function(){  //alert("疼!");  //希望点哪个按钮，就让当前按钮的内容变成❀  //正确:  //this.innerHTML="❀"; //this->当前正在触发事件的这个元素对象  /\*功能一. 实现点击按钮，修改数量\*/  //3. 查找要修改的元素  //本例中: 点击按钮，要修改当前按钮旁边的span的内容，所以，这里应该查找当前按钮旁边的span元素对象    //可以先查找当前按钮的爹td，然后查找td下所有直接子元素中第二个元素  var span=this.parentElement.children[1];  //console.log(span);  //4. 修改元素  //本例中: 如果当前按钮的内容是+,就对span的内容+1，否则如果当前按钮的内容是-，就对span的内容-1，但是购物车中数量不能小于1，到1就不能再减了。  //要想修改一个元素的内容，必须先获得元素的内容，修改后，再放回去  //4.1 先获得span的内容，因为之后要做算术计算，保险起见转为数字  //\*\*\*凡是从页面获得的一切，都是字符串类型。算术计算之前都要转为数字  var n=parseInt(span.innerHTML);//发生了按值传递,n只是span内容的副本  //4.2 根据点的按钮不同，修改n的值  //如果当前按钮的内容是+  if(this.innerHTML=="+"){  n++;//就n+1  }else if(n>1){//否则如果当前按钮的内容是-，且n还>1时  n--;//才能n-1  }//在程序中修改n，是不影响页面显示的  //4.3 要想将修改后的新值显示在页面上，必须将修改后的n再放回元素内容中  span.innerHTML=n;  /\*功能二. 数量修改同时，修改小计\*/  //3. 查找要修改的元素  //本例中: 数量修改，同时要修改当前行最后一个td中的小计内容  //查找当前按钮的爹的下一个兄弟  var lastTd=this.parentElement.nextElementSibling;  //4. 修改元素  //本例中: 先用单价\*数量n，计算出小计，再将小计放入lastTd的内容中    //4.1. 获得当前按钮的爹的前一个兄弟元素的内容中的单价  //特殊加工: 还要截取1位置到结尾的子字符串，可以去掉开头的¥  // 因为钱可能有小数，所以用parseFloat将截取后的字符串，转为数字  var price=parseFloat(  this.parentElement //td  .previousElementSibling //前一个td  .innerHTML //"¥4488.00"  .slice(1) //"4488.00"  );//4488  //console.log(price);  //4.2 计算小计=单价\*数量n  var subtotal=price\*n;  //4.3 将计算后的小计放入本行最后一个td的内容中，前加¥，后保留两位小数  lastTd.innerHTML=`¥${subtotal.toFixed(2)}`;  /\*功能三. 计算总价\*/  //3. 查找要修改的元素  //本例中: 总价应该修改到table下tfoot下最后一个td的内容中，所以应该查找table下tfoot下最后一个td    //var lastTd=table.querySelector("table>tfoot td:last-child");  // 已经在table下查找了 选择器只要从table下开始写即可  var lastTd=table.querySelector("tfoot td:last-child");  //console.log(lastTd);  //4. 修改元素  //本例中: 应该先获得tbody中每行最后一个td中的小计内容，去¥转为小数后，累加起来计算出总计。最后将总计放入tfoot中最后一个td中  //4.1 先查找table下tbody中每行最后一个td  var tds=table.querySelectorAll("tbody td:last-child");  //console.log(tds);  var total=0;//声明变量准备累加所有小计  //4.2 遍历tds中每个td  for(var td of tds){  //每遍历一个td，就获取当前td的内容，去¥转为小数后，累加到变量total中  total+=parseFloat(td.innerHTML.slice(1));  }  //4.3 将计算好的total总价，放入tfoot最后一个td中  lastTd.innerHTML=`¥${total.toFixed(2)}`;  }  }  </script>  </body>  </html>  运行结果: |

二. 修改: 当我们找到一个元素后，可以改元素的三样东西:

1. 获取或修改元素的内容: 3种

对应小程序视频: 小程序->在线->DOM->day02 0.1 修改元素的内容...

(1). 获取或修改元素开始标签到结束标签之间的原始的HTML代码内容:

元素.innerHTML=`新HTML片段`

内部 html

(2). 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=`纯文本内容`

文本 内容

|  |
| --- |
| .innerHTML vs .textConent  1. 获取元素内容时:  .innerHTML: 获取的是原始的HTML代码内容  .textContent: 多做了两件事: 1. 去掉内嵌标签，2. 特殊符号翻译为正文  2. 修改元素内容时:  .innerHTML:会将新的内容先交给浏览器解析，将解析后的结果给人看  .textContent: 不会将新内容交给浏览器解析，努力保持内容的原样输入给人看 |

(3). 获取或修改表单元素的内容:

a. 问题: 大多数input表单元素是单标记！比如: <input/>，没有innerHTML或textContent

b. 解决: 今后只要想获取或修改表单元素的内容都要用：表单元素.value=新值

(4). 示例: 开关门效果

对应小程序视频: 小程序->在线->DOM->day02 1 动画 开关门效果

0\_door.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <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: 用户点d2，触发开关门  var d2=document.getElementById("d2")  //2. 绑定事件处理函数  //本例中: 用户点击d2，触发开关门  d2.onclick=function(){  //3. 查找要修改的元素  //本例中: 点d2，要来回切换d1的显示和隐藏  var d1=document.getElementById("d1");  //4. 修改元素  //本例中: 第一次点d2，隐藏d1，第二次再点d2，则显示d1  //如果d1的style的display不是none  if(d1.style.display!=="none"){  d1.style.display="none";//就变成none  //当d1关上时，就修改当前点击的d2的内容为>>  this.innerHTML="&gt;&gt;";  }else{//否则如果d1的style的display是none  d1.style.display="";//就清除none，元素默认都是显示的  //当d1打开时，就修改当前点击的d2的内容为<<  this.innerHTML="&lt;&lt;";  }  }  </script>  </body>  </html>  运行结果: |

2. 属性: 3种:

(1). 字符串类型的HTML标准属性

对应小程序视频: 小程序->在线->DOM->day02 0.2 修改元素属性 字符串类型...

a. 什么是: HTML标准中规定的，值为字符串类型的HTML元素属性

b. 比如: id="xxx", name="xxx", title="xxx", href="xxx", src="xxx" ...

c. 如何: 2种:

1). 旧DOM标准提供的核心DOM4个函数:

i. 获得一个元素上一个属性的属性值: var 属性值=元素.getAttribute("属性名")

ii. 修改一个元素上一个属性的属性值: 元素.setAttribute("属性名","属性值")

iii. 判断一个元素上是否包含一个属性: var bool=元素.hasAttribute("属性名")

iv. 移除一个元素上的一个属性: 元素.removeAttribute("属性名")

v. 示例: 使用核心DOM 4个函数操作元素的字符串类型HTML标准属性

1\_attribute.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>  </head>  <body>  <a id="a1" href="http://tmooc.cn" target="\_blank">go to tmooc</a>  <script>  var a1=document.getElementById("a1");  //想获得a1的href属性值  var href=a1.getAttribute("href");  console.log(href); //"http://tmooc.cn"  //想修改a1的title属性值  a1.setAttribute("title","欢迎访问tmooc");  //想判断a1是否包含target属性  var bool=a1.hasAttribute("target");  console.log(bool);//true  //移除a1的target属性  a1.removeAttribute("target");  //想判断a1是否包含target属性  var bool=a1.hasAttribute("target");  console.log(bool);//false  </script>  </body>  </html>  运行结果: |

2). 新HTML DOM标准提供的简写: 元素.属性名

i. 因为浏览器在扫描元素时，就已经将元素所有HTML规定的属性都提前定义在了元素对象身上。所以，所有的HTML标准属性，都可以用"元素.属性名"方式来访问

ii. 比如:

①获取一个元素上一个属性的属性值: 元素.属性名

②修改一个元素上一个属性的属性值: 元素.属性名="属性值"

③判断一个元素上是否包含一个属性: 元素.属性名!==""

④移除一个元素上的一个属性: 元素.属性名=""

iii. 示例: 使用.属性名简写上一个例子，操作元素的属性  
1\_property.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>  </head>  <body>  <a id="a1" href="http://tmooc.cn" target="\_blank">go to tmooc</a>  <script>  var a1=document.getElementById("a1");  console.dir(a1);  //想获得a1的href属性  console.log(a1.href);  //设置a1的title属性  a1.title="欢迎访问tmooc"  //判断a1是否设置target属性  console.log(a1.target!=="");//true  //移除a1的target属性  a1.target="";//清空target属性值，失效了！  //判断a1是否设置target属性  console.log(a1.target!=="");//false  </script>  </body>  </html>  运行结果: |

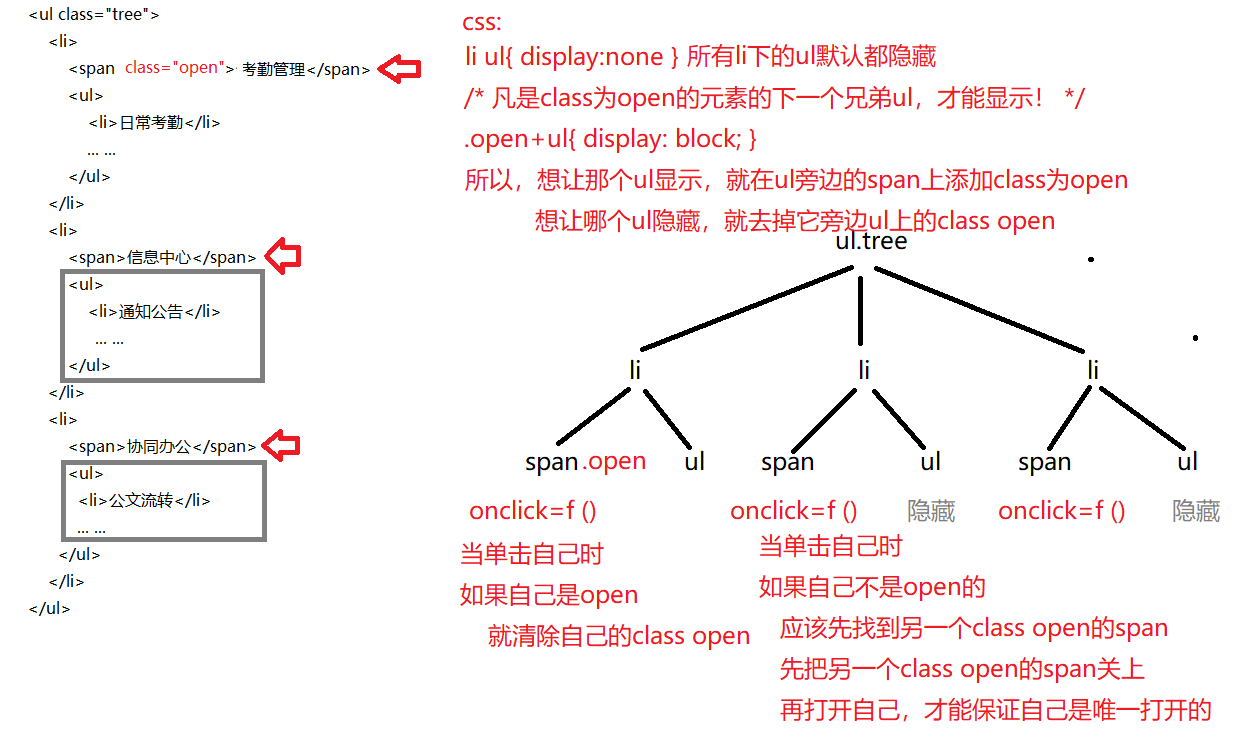
iv. 特例: class属性

①问题: 因为ECMAScript中已经将class定义为js语言的关键字，专门用于创建一种新类型。所以DOM标准中就不能再使用class作为属性名

②解决: 今后只要用.操作元素的class属性，都要被迫更名为className。操作.className，就等效于操作元素的class属性值。

v. 示例: 下拉菜单 手风琴效果

对应小程序视频: 小程序->在线->DOM->day02 2. 下拉菜单 手风琴效果...



1\_menu.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>1. 实现伸缩二级菜单</title>  <meta charset="utf-8" />  <style>  /\* 清空li前的小圆点 \*/  li{  list-style:none;  }  /\* 定义一级菜单span的默认样式 \*/  li span{  padding-left: 20px;  cursor: pointer;  background: url("../images/add.png") no-repeat center left;  }  /\* 所有li下的ul，默认都是隐藏的！ \*/  li ul{  display: none;  }  /\* 仅将span前的+号背景变成-号背景 \*/  .open{  background: url("../images/minus.png") no-repeat center left;  }  /\* 凡是class为open的元素的下一个兄弟ul，才能显示！ \*/  .open+ul{  display: block;  }  </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: 点击每个li中的span触发切换不同的菜单显示  //查找class为tree的ul下的所有span元素  var spans=document.querySelectorAll("ul.tree span");  //2. 绑定事件处理函数  //本例中: 因为每个span都可单击  //遍历spans中每个span，绑定单击事件  for(var span of spans){  span.onclick=function(){  //this->当前正在点击的这一个span元素  //测试: 点哪个span，哪个span就变成❀  //this.innerHTML="❀";  //3. 查找要修改的元素  //本例中: 不用查找，因为点一个span时，只要修改自己的class就行了！  //4. 修改元素  //本例中: 当单击一个span时，当前span应该在开关之间来回切换  //如果当前span的class是open，说明当前span旁边的ul是开着的，那么我们只要清除span的class open，就把当前span旁边的ul关上了  if(this.className=="open"){  this.className="";  }else{//否则如果当前span的class不是open，说明当前span旁边的ul是隐藏的。那么只要修改当前span的class为open，当前span旁边的ul就打开了！  //打开自己之前，为了防止别的span也开着，应该尝试去查找另一个开着的span。（根据需求，一次只可能有一个是开着的，所以只要找一个就可以了！不用找全部！）  var openSpan=document.querySelector("ul.tree span.open");  //如果找到另一个开着的span，应该先把另一个开着的span关上  if(openSpan!=null){  openSpan.className="";  }  //然后再打开自己  this.className="open";  //结果: 肯定这一次只有当前span自己是打开的！别的span都是关上的  }  }  }    </script>  </body>  </html>  运行结果: |

(2). bool类型的HTML标准属性

对应小程序视频: 小程序->在线->DOM->day02 0.3 修改元素的属性 bool类型...

a. 什么是: HTML标准中规定的，不写属性值也能直接发挥作用的特殊属性

b. 比如: disabled 禁用 checked 选中 selected 选中

c. 如何: 1种

~~1). 不能用旧的核心DOM4个函数操作。因为那四个函数只支持字符串类型的HTML标准属性。不支持bool值！~~

2). 只能用新HTML DOM 打.访问。且属性值必须是bool类型

d. CSS: 状态伪类:

1). 什么是: 专门根据元素的状态来匹配元素

2). 包括:

i. :disabled 专门匹配禁用的元素

ii. :checked 专门匹配选中的元素checkbox或radio

iii. :selected 专门匹配选中的元素select下的option

e. 示例:

对应小程序视频: 小程序->在线->DOM->day02 3. 全选 取消全选 selectAll

2\_selectAll.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <title>全选和取消全选</title>  </head>  <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>  /\* 功能一. 点全选，控制下方所有checkbox的选中状态 \*/  //DOM 4步  //1. 查找触发事件的元素  //本例中: 用户点thead下的input，触发了变化  var chbAll=document.querySelector("table>thead input");  //2. 绑定事件处理函数  //本例中: 用户单击input时，触发变化  chbAll.onclick=function(){  //3. 查找要修改的元素  //本例中: 点全选，要改tbody中所有input  var chbs=document.querySelectorAll("table>tbody input");  //4. 修改元素  //本例中: 修改tbody中所有input的checked属性值和当前点击的全选input的checked属性保持一致  for(var chb of chbs){  chb.checked=this.checked;  }  }  /\* 功能二. 点击tbody中每个input，也有可能影响thead中input的状态 \*/  //DOM 4步  //1. 查找触发事件的元素  //本例中: 用户点tbody中每个input都可能触发变化  var chbs=document.querySelectorAll("tbody input");  //2. 绑定事件处理函数  //本例中: tbody下每个input都可以点击  for(var chb of chbs){  chb.onclick=function(){  //3. 查找要修改的元素  //本例中: 点tbody中每个input，都可能影响thead中input  var chbAll=document.querySelector("thead input");  //4. 修改元素  //本例中: 如果tbody中所有input都选中，则上边的全选就选中，否则如果tbody中有未选中的input，则上边的全选就不选中  //4.1 尝试查找tbody下所有input中那个未选中的input  var unchecked=document.querySelector("tbody input:not(:checked)")  //4.2 如果找到一个未选中的input，则上方的全选就不选中  if(unchecked!=null){  chbAll.checked=false;  }else{//4.3 否则如果未找到未选中的input，则上方的全选就选中  chbAll.checked=true;  }  }  }  </script>  </body>  </html>  运行结果: |

(3). 自定义扩展属性

对应小程序视频: 小程序->在线->DOM->day02 4.1 自定义扩展属性 何时使用...

a. 什么是: HTML标准中没有规定的，程序员根据自己的需要自发添加的一些自定义属性

b. 何时: 2种:

1). 在HTML元素上临时缓存一些业务相关的数据：

比如: <ul>

<li score="89">亮亮</li>

<li score="61">然然</li>

<li score="91">东东</li>

2). 可以代替id、元素、class选择器作为查找触发事件的元素的条件

i. id、元素、class选择器为什么不适合作为查找触发事件的元素的条件

①id选择器: 一次只能选择一个元素

②元素选择器: 人家在界面上实现一种效果，不一定都用一种元素

比如: 想实现一个按钮，可能使用: a button input span ...

③class选择器: class的本职工作为元素添加css样式的。不是为js服务的。维护时，一旦别人发现一个和样式无关的class，很可能就会被清除掉！

ii. 解决: 可以添加自定义属性，来专门为js服务

比如: <button data-click="btn">

var btn=document.querySelector("[data-click=btn]")

像bootstrap学习，所有触发事件的元素上，都应该用自定义属性作为条件！

c. 如何: //HTML5标准

1). 为元素添加/修改自定义属性:

i. HTML中: <元素 data-自定义属性名="属性值">

ii. js中:

~~①不能用".自定义属性名"访问，因为所有自定义属性都是后天添加的，HTML标准中没有，自然也就不包含在内存中元素对象身上！~~

②旧HTML标准中，只能用核心DOM 4个函数中的:

元素.setAttribute("data-自定义属性名","属性值")

③HTML5中: 只要加了data-前缀的自定义属性，还可以用dataset属性来操作

元素.dataset.自定义属性名=新值

2). 获取自定义属性值:

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

ii. HTML5中: 元素.dataset.自定义属性名

3). 查找带有自定义扩展属性的元素: CSS属性选择器

var 元素/类数组对象=document.querySelectorAll("[data-自定义属性名=属性值]")

d. 示例: 使用HTML5方式操作自定义扩展属性

3\_data-.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>  </head>  <body>  <a href="javascript:;" data-click="btn" data-count="5">click me</a>  <script>  //用属性选择器查找到带有data-click属性且属性值为btn上按钮元素  var btn=document.querySelector("[data-click=btn]");  console.log(btn);  //想将btn上count属性值+1  var count=parseInt(  btn.dataset.count  //btn.getAttribute("data-count")  );  count++;  //btn.setAttribute("data-count",count);  btn.dataset.count=count;  </script>  </body>  </html>  运行结果: |

3. 样式:

(1). 修改内联样式:

a. HTML中: <元素 style="css属性: 属性值">

b. DOM中: 元素对象.style.css属性="属性值" 下边这句话会被自动翻译为上边那句话

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. 修改内联样式: 元素.style.css属性="属性值"

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:

小程序->在线->DOM->day01 3. 购物车 shoppingcart

小程序->在线->DOM->day02 0.1 修改元素的内容...

1 动画 开关门效果

0.2 修改元素的属性 字符串类型HTML标准属性

2 下拉菜单 手风琴效果

0.3 修改元素的属性 bool类型的HTML标准属性

3. 全选 取消全选 selectAll

4.1 自定义扩展属性 何时使用...

作业:

1.复习今日小程序问题清单(小程序->在线->DOM->day01、02)，预习明日小程序问题清单(小程序->在线->DOM->day03)

2.看小程序视频学习DOM学习优化查找元素

小程序->在线->DOM->day01 4. 优化 查找元素...

3. 看小程序视频预习标签页效果:

下程序->在线->DOM->day02 4.2 标签页 tabs

在day02/3\_tabs.html中直接做

4. 看小程序视频预习修改样式、计算属性

小程序->在线->DOM->day02 5 修改样式、计算属性...

5. （学有余力!!!!!!）看小程序视频学习如何为开关门效果添加过渡：

小程序->在线->DOM->day02 1. 动画 开关门效果 door transition

6. （学有余力!!!!!!）看小程序视频学习如何为下拉菜单效果添加过渡:

小程序->在线->DOM->day02 2 下拉菜单 手风琴效果