小程序: WEB问题速查

1. 问题清单: 小程序->首页->DOM->day02

2. 高清无码小视频: 小程序->在线->DOM->day02

今日扩展视频:

小程序->在线->DOM->day02

1. 动画 开关门效果

2. 下拉菜单 手风琴效果 动画 过渡

5. 修改样式，计算属性，。。。 重排重绘

扩展: 如何单独操作某一个class

不用今天都看完！！！！！！\*\*\*两个月后\*\*\*鄙视面试前\*\*\*看完就行\*\*\*！

3. 单词列表: 小程序->单词->DOM->day02

第三阶段上课约定:

1. 上课千万不要跟着记笔记和写代码！

2. 拼写错误！自己找！让我帮你找，你需要手抄20遍，给我拍照！

3. 每个功能写三遍:

1). 注释+代码抄一遍，明白案例要实现什么功能

2). 保留注释，删除代码，自己试着看着注释把代码填写回来，复习对象和函数

3). 删除注释和代码，自己试着用自己的语言把注释写出来，再把代码添回来。

\*\*\*\*\*只有自己能写出人话的注释，这个案例才算自己会。\*\*\*\*\*

\*\*\*\*\*你们现在刚学DOM，正处在照猫画虎的阶段！不要一上来就想着原创\*\*\*\*\*

如何提问:

1. 上课时，案例中，笔记中那句话哪行代码看不懂，立刻微信私信问我！

2. 小程序中问题清单里哪个问题和回答看不懂！立刻微信私信问我！

3. 个人项目中哪个功能不会做！立刻微信私信问我！

4. 刷笔试题时，遇到不会的笔试题！立刻微信私信问我！

下载源代码: 右键->链接另存为！

正课:

1. 查找

2. 修改

一. 查找: 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. 按选择器查找元素:

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

(1). 何时: 今后如果查找一个元素的条件比较复杂时，首选按选择器查找！

(2). 如何: 2个函数

a. 只查找一个符合要求的元素:

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

查找(按)选择器

2). 意为: 在指定父元素范围内，查找第一个符合选择器要求的元素

3). 返回值:

i. 如果找到，返回一个元素对象；

ii. 如果没找到，返回null

4). 强调: 永远只能返回第一个符合选择器要求的元素！

b. 查找多个符合要求的元素：

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

2). 意为: 在指定父元素下，查找所有符合选择器要求的元素，放在一个类数组对象中返回！

3). 返回值:

i. 如果找到，返回类数组对象;

ii. 如果每找到，返回空类数组对象: { length:0 }

5. 示例:购物车:

day01 剩余/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tfoot中最后一个td背景变为黄色\*/    tfoot td:last-child{      background-color:yellow;    }    /\*想让tbody中每行最后一个td背景变为粉色\*/    tbody>tr>td:last-child{      background-color:pink    }  </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>      //DOM4步：      //1. 查找触发事件的元素      //本例中: 要找table中所有button元素      //1.1 先找id为data的table      var table=document.getElementById("data")      //1.2 在table内查找所有button      var btns=table.getElementsByTagName("button");      console.log(btns);      //2. 绑定事件处理函数      //本例中: 遍历查找结果中每个按钮元素对象      for(var btn of btns){        //每遍历一个按钮元素对象，就为当前按钮元素对象的onclick属性赋值一个事件处理函数        btn.onclick=function(){          /\*点击按钮修改数量\*/            //3. 查找要修改的元素          //本例中: 查找当前按钮旁边的span的元素          //好的办法: 找当前按钮的爹下的第二个孩子          var span=this.parentElement.children[1];          //4. 修改元素          //本例中:          //4.1 先取出span现在的内容，转为数字，保存在变量n中          var n=parseInt(span.innerHTML);          //4.2 做计算          //如果当前按钮的内容是+          if(this.innerHTML=="+"){            //就将n+1            n+=1;          }else if(n>1){//否则如果n>1            //才能将n-1;            n-=1;          }          //4.3 将新值放回span的内容中          span.innerHTML=n;          /\*数量改变，修改小计\*/          //3. 查找要修改的元素          //本例中: 找当前按钮的爹(td)的下一个兄弟td          var subTd=            this.parentElement.nextElementSibling          //4. 修改元素          //本例中:          //4.1 获得单价price，转为数字          var price=parseInt(            this.parentElement                .previousElementSibling                .innerHTML                .slice(1)          );          //4.2 用单价price\*数量n，算出小计          var sub=price\*n;          //4.3 放入小计格的内容中          subTd.innerHTML=`¥${sub.toFixed(2)}`;          /\*计算总计\*/            //3. 查找要修改的元素          //本例中: table下tfoot下最后一个td          var totalTd=table.querySelector(            "tfoot td:last-child"//复习第二阶段选择器          );          //4. 修改元素          //4.1 查找tbody中每行最后一个td          var subTds=table.querySelectorAll(            "tbody>tr>td:last-child"          );          //4.2 计算总计          //本例中: 遍历并累加每个小计格中的内容          var total=0;          for(var td of subTds){            total+=parseInt(              td.innerHTML.slice(1)            );          }          //4.3 将总计放入总计格的内容中          totalTd.innerHTML=`¥${total.toFixed(2)}`;        }      }    </script>  </body>  </html>  运行结果： |

二. 修改: 3种东西

1. 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内容:

a. 元素.innerHTML

内部的html

b. 指的是<开始标签>原始HTML代码内容</结束标签>

(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:

a. 元素.textContent

文本 内容

b. 与 innerHTML 相比多做了两件事:

1). 去掉了内嵌的标签

2). 将特殊符号翻译为正文

(3). 获取或修改表单元素的值:

a. 问题: 大部分input表单元素都是单标记，没有结束标签，所以无法使用innerHTML和textContent

b. 解决: 今后，只要想获得表单元素的值，都用表单元素.value

(4). 示例: 比较三种内容属性的不同:

0\_content.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>    <!--            什么也不做    -->    <p id="p1">来自<a href="javascript:;">&lt;&lt;新华社&gt;&gt;</a>的消息</p>    <input/><button id="btn">百度一下</button>    <script>      var p1=document.getElementById("p1");      //想获取p元素的内容      console.log(p1.innerHTML);      //想获得p元素的内容，但是想去掉内嵌的标签和特殊符号，只获取文字      console.log(p1.textContent);        //想点击按钮，获得旁边文本框的内容      var btn=document.getElementById("btn");      btn.onclick=function(){        var input=this.previousElementSibling;        console.log(`查找 ${input.value} 相关的内容...`)      }    </script>  </body>  </html>  运行结果: |

(5). 示例: 动态开关门效果:

0\_door.html

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

|  |
| --- |
| <!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" style="">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触发事件      var d2=document.getElementById("d2");      //2. 绑定事件处理函数      d2.onclick=function(){        //3. 查找要修改的元素        //本例中: 用户点d2，改的是d1        var d1=document.getElementById("d1");        //4. 修改元素        //如果d2的内容是>>，说明d1是关着的！        if(d2.innerHTML=="&gt;&gt;"){          //就打开d1: 其实就是让d1显示出来，其实就是去掉d1的display属性          d1.style.display="";          //等效于<div id="d1" style="">          //d2的内容变为<<          d2.innerHTML="&lt;&lt;";        }else{//否则如果d1是开着的          //就关闭d1: 其实就是让d1隐藏, 其实就是修改d1的display属性为none          d1.style.display="none";          //等效于: <div id="d1" style="display:none">          //d2的内容变为>>          d2.innerHTML="&gt;&gt;";        }      }    </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: 0.2 修改元素的属性 字符串类型

a. 什么是HTML标准属性: HTML标准中规定的属性

b. 如何: 2种:

1). 旧核心DOM 4个函数:

i. 获取一个元素的一个属性值:

元素.getAttribute("属性名")

获取 属性

ii. 修改一个元素的属性值:

元素.setAttribute("属性名","属性值")

修改 属性

iii. 判断元素上是否包含某个属性  
 var bool值=元素.hasAttribute("属性名")

有 属性(吗)?

iv. 移除元素上一个属性

元素.removeAttribute("属性名")

移除 属性

v. 示例: 使用核心DOM4个函数操作元素的属性

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" title="欢迎访问tmooc">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);      //修改a1的title属性值      a1.setAttribute("title","Welcome tmooc");      //判断a1是否包含target属性      var bool=a1.hasAttribute("target");      console.log(bool);//false      //移除a1的id属性      a1.removeAttribute("id");      console.log(a1);    </script>  </body>  </html>  运行结果: |

vi. 问题: 旧核心DOM函数名太长，不好用！

2). 新HTML DOM:

i. 什么是HTML DOM: 是专门对常用的HTML对象和属性提供的简化版DOM

ii. 如何简化: HTML DOM已经提前将所有HTML标准属性都保存在了内存中的元素对象身上，只不过暂时不用的属性值，默认都为""。可以直接用.方式，访问对象中的标准属性。

iii. 示例: 使用新HTML DOM简化版属性实现和旧版核心DOM 相同的功能

1\_attribute2.html

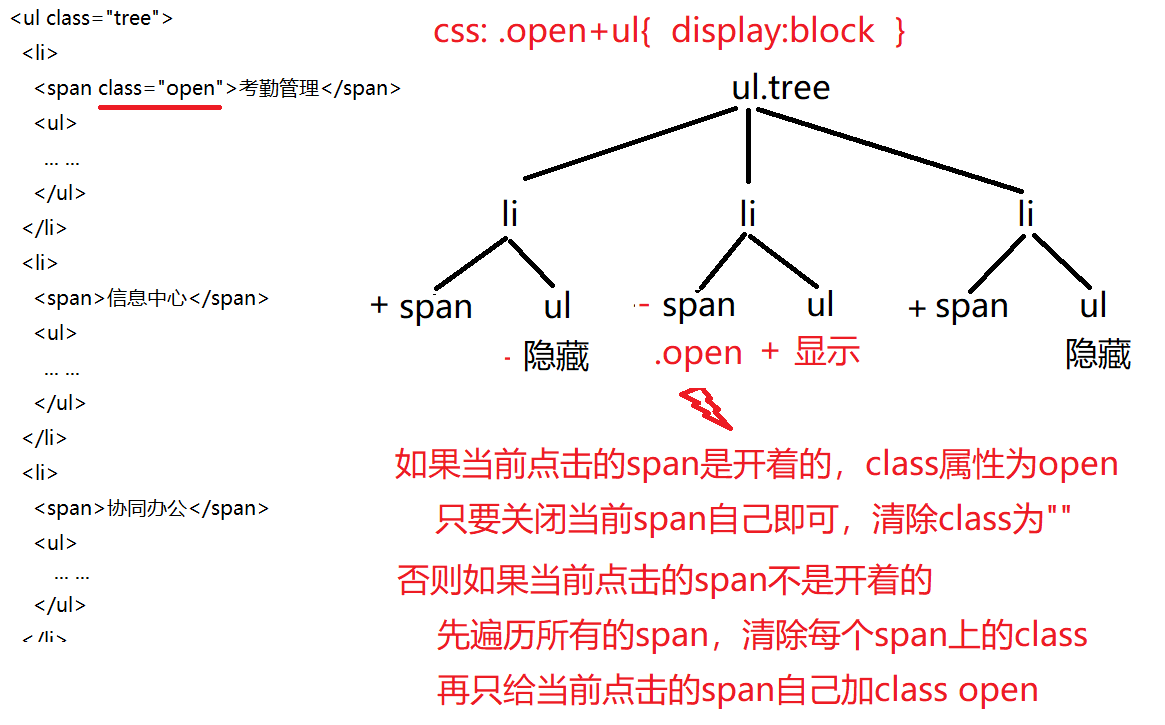
|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>  </head>  <body>    <a id="a1" href="http://tmooc.cn" title="欢迎访问tmooc">go to tmooc</a>    <script>      var a1=document.getElementById("a1");      //想获得a1的href属性值      console.log(a1.href);      //修改a1的title属性值      a1.title="Welcome tmooc";      //判断a1是否包含target属性      console.log(a1.target!=="");      //移除a1的id属性      a1.id="";        console.log(a1);      console.dir(a1);    </script>  </body>  </html>  运行结果: |

iv. 问题: HTML中有class属性来定义元素的样式类，而巧了，js中也有class关键字定义一种类型。两个class冲突了！

解决: DOM妥协了！从此修改元素的class属性，都必须更名为className。反之，修改className，就等效于修改元素的class属性。

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>      li {        list-style: none;      }      li span {        padding-left: 20px;        cursor: pointer;        background: url("images/add.png") no-repeat center left;      }      li ul {        display: none;      }      .open {        background: url("images/minus.png") no-repeat center left;      }      .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: 查找所有li下的span      var spans=document.querySelectorAll(        "li>span"      );      //2. 绑定事件处理函数      //本例中: 遍历找到的每个span，为每个span绑定单击事件      for(var span of spans){        span.onclick=function(){          //3. 查找要修改的元素          //4. 修改元素          //本例中:          //如果当前span自己是开着的          //其实就是看当前span的class属性值是不是open          // if(this.getAttribute("class")=="open"){          if(this.className==="open"){            //只要把自己关上即可！不用管别的span            //其实就是清除当前span的class属性值            // this.setAttribute("class","")            this.className="";          }else{//否则如果当前span不是开着的            //先遍历所有span，清除每个span的class            for(var span of spans){              // span.setAttribute("class","");              span.className="";            }            //再只给当前单击的span自己加class open            // this.setAttribute("class","open");            this.className="open";          }        }      }    </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bool类型的HTML标准属性: 在HTML标准属性中，有一类属性只要放在元素上，即使不提供=属性值，也能发挥作用！  
 b. 比如: <input type="button" disabled>

<input type="radio" checked>

c. 问题: 因为这种属性没有=属性值，所以不能用旧的核心DOM4个函数来操作。

d. 解决: 今后只要访问bool类型的HTML标准属性，都只能用HTML DOM打.访问。比如: 元素.disabled 元素.checked。且属性值必须是bool类型的true或false。

e. 补: css中其实已经提供了一套伪类选择器，专门选择处于某种状态的元素！——状态伪类，包括:

1) :checked —— 专门选择被选中的元素

2) :disabled —— 专门选择被禁用的元素

... ...

f. 示例: 全选 取消全选

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>      /\*点上方的全选，控制下方的所有checkbox\*/      //DOM 4步      //1. 查找触发事件的元素      //本例中: 查找表头中的input      var inputAll=document.querySelector(        "table>thead input"//复习第二阶段选择器      );      //2. 绑定事件处理函数      inputAll.onclick=function(){        //3. 查找要修改的元素        //本例中: 查找tbody下的所有input        var inputs=document.querySelectorAll(          "table>tbody input"        )        //4. 修改元素        //本例中: 让下边每个input的checked状态和当前点击的全选checkbox的checked状态一致        for(var input of inputs){          input.checked=this.checked;        }      }      /\*点下方的每个，都有可能影响上方的全选\*/      //DOM 4步      //1. 查找触发事件的元素      //本例中: 查找tbody下所有input      var inputs=document.querySelectorAll(        "table>tbody input"      );      //2. 绑定事件处理函数      //本例中: 遍历找到的每个input元素      for(var input of inputs){        //每遍历一个input，就绑定单击事件        input.onclick=function(){          //3. 查找要修改的元素          //本例中: 无论点下方哪个input，都只可能影响thead中的一个input          var inputAll=document.querySelector(            "table>thead input"          );          //4. 修改元素          //尝试查找tbody下未选中的一个input          var unchecked=document.querySelector(            "tbody input:not(:checked)"          )//复习第二阶段选择器          //如果找到未选中的          if(unchecked!=null){            //上方的全选就不选中！            inputAll.checked=false;          }else{//否则如果没找到未选中的            //上方的全选就选中！            inputAll.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

a. 什么是自定义扩展属性: HTML标准中没有规定的，程序员根据自己的需要，自发的添加到元素上的自定义属性。

b. 何时: 2种:

1). 自定义扩展属性经常代替id, class, 元素等其他选择器，作为查找触发事件的元素的条件。

i. id选择器的问题: 一次只能找一个元素

ii. 元素选择器的问题: 实现同一种页面效果，完全可能使用不同的元素。

iii. class选择器的问题: class属性本职工作是为元素添加样式！因为样式经常发生变化，所以class属性也经常变化！

2). 在客户端元素上临时缓存业务所需的数据

c. 如何: HTML5标准

1). 手工在HTML元素上添加自定义属性:

<元素 data-属性名="属性值">

2). js程序中访问自定义扩展属性:

i. 问题: 因为自定义扩展属性不是HTML标准规定的标准属性，所以HTML DOM并没有把自定义扩展属性提前保存到内存中的元素对象上！就无法用.方式快速访问！

ii. 解决: 2种:

① 用回旧核心DOM函数:

元素.getAttribute("data-自定义属性名")

元素.setAttribute("data-自定义属性名","新属性值")

②HTML5标准中提供了简写: HTML5标准用dataset属性将所有data-开头的自定义属性集中保存起来！访问时，可以用"元素.dataset.自定义属性名"访问自定义扩展属性

d. 示例: 使用自定义属性记录按钮的点击次数：

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>    <!--用自定义属性悄悄记录按钮的点击次数-->    <button data-n="0">click me</button>    <script>      //DOM 4步      //1. 查找触发事件的元素      //本例中: 查找带有data-n属性的元素      //复习第二阶段属性选择器      var btn=document.querySelector(        "[data-n]"      )      //2. 绑定事件处理函数      btn.onclick=function(){        //3. 查找要修改的元素        //本例中: 不用找，因为就是要修改自己        //4. 修改元素        //4.1 取出当前按钮身上自定义属性data-n的值        var n=parseInt(          //this.getAttribute("data-n")          this.dataset.n          //页面上的一切都是字符串        );        //4.2 数量+1        n++;        //4.3 将新值放回自定义属性中        //this.setAttribute("data-n",n);        this.dataset.n=n;      }    </script>  </body>  </html>  运行结果： |

小常识: 前端中:

Attribute: 特指写在HTML代码中元素的开始标签中的HTML属性，比如: <a href="http://tmooc.cn" target="\_blank">

Property: 特指js程序中对象中保存的可用.访问的属性，比如:

lilei.sname lilei.sage

总结: DOM 4步:

1. 先查找触发事件的元素

2. 再为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 5件事: 增删改查+事件绑定:

//不要背！用到时，来查找，复制粘贴就行了！

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种:

document 根节点 document

document.documentElement —— <html>

document.head —— <head>

document.body —— <body>

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

a. 父子关系: 4种

1). 元素的父元素: 元素.parentElement

2). 元素下的所有直接子元素: 元素.children

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

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

b. 兄弟关系: 2种

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

2). 元素的后一个兄弟: 元素.nextElementSibling

(3). 如果用一个特征就能找到想要的元素，就用按HTML特征查找：4个方法:

a. 按id查找:

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

b. 按标签名查找:

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

c. 按class名查找:

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

d. 按name名查找表单元素:

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

(4). 如果通过复杂的查找条件，才能找到想要的元素时，就用按选择器查找: 2个方法

a. 只查找一个符合条件的元素:

var 一个元素=任意父元素.querySelector("任意选择器")

b. 查找多个符合条件的元素:

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

|  |
| --- |
| 总结: 查找函数的返回值规律: 3种  1. 如果原函数返回的是下标位置i， 则如果找不到-1  2. 如果原函数返回的是一个数组或一个对象，则如果找不到返回null  3. 如果原函数返回的是类数组对象，则如果找不到，返回空类数组对象: { length:0 } |

2. 修改元素: 3种东西可修改

(1). 修改内容: 3种内容可修改:

a. 获取或修改元素的HTML内容:

1). 元素.innerHTML

2). 获取时: 返回原始HTML内容

3). 修改时: 先将新内容交给浏览器编译，再显示给人看

b. 获取或修改元素的纯文本内容:

1). 元素.textContent

2). 获取时: 返回去掉内嵌标签，将特殊符号翻译为正文后的纯文本

3). 修改时: 不会将新内容交给浏览器编译，而是原样显示给人看

c. 获取或修改表单元素的值:

表单元素.value

(2). 修改属性: 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. var bool=元素.hasAttribute("属性名")

iv. 元素.removeAttribute("属性名")

优点: 万能, 缺点: 繁琐

2). 新HTML DOM:

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=""

优点: 简单, 缺点: 不万能

b. bool类型的HTML标准属性：

1). 不能用旧核心DOM4个函数修改

2). 只能用HTML DOM的"元素.属性名"方式获取或修改，且值为bool类型

c. 自定义扩展属性:

1) HTML中: <元素 data-自定义属性名="属性值">

2). js中: 2种: （不能用.访问）

i. 核心DOM:

var 属性值=元素.getAttribute("data-自定义属性名")

元素.setAttribute("data-自定义属性名","属性值")

ii. HTML5标准: 元素.dataset.自定义属性名

(3). 修改样式:

a. 修改元素的内联样式:

元素.style.css属性="属性值"

b. 获取元素的完整样式:

var style=getComputedStyle(元素对象);

style.css属性

计算后的样式都是只读的

c. 批量修改元素的样式时，都用class:

元素.className="class名"

4. 事件绑定: 3种:

(1). 在HTML中绑定: DOM和jquery中几乎不用

a. HTML中: <元素 on事件名="事件处理函数()">

b. JS中: function 事件处理函数(){ ... }

(2). 使用赋值方式绑定:

元素对象.on事件名=function(){ ... }

(3). (待续...)

|  |
| --- |
| 总结: this 5种: 判断this，~~一定不要看定义在哪儿！~~只看调用时！  1. obj.fun() this->obj  2. fun() 或 (function(){ ... })() 或 多数回调函数 this->window  3. new Fun() this->new正在创建的新对象  4. 类型名.prototype.共有方法=function(){ ... } this->将来谁调用指谁，同第一种情况  5. 事件处理函数中的this->当前正在触发事件的DOM元素对象 |

今日对应小程序视频列表:

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 door

0.2 修改元素的属性 字符串类型

2. 下拉菜单 手风琴效果 动画

0.3 修改元素的属性 bool类型

3. 全选 取消全选 selectAll

4.1 自定义扩展属性

作业:

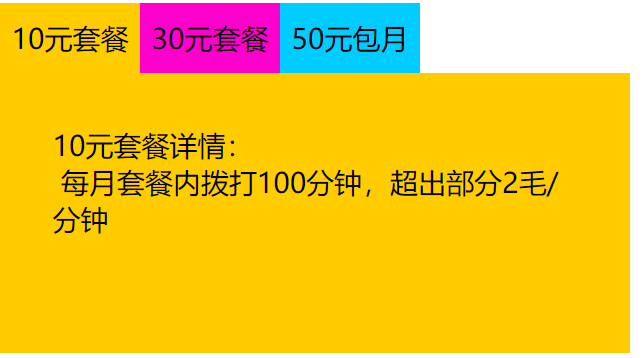
1. 复习今日小程序问题清单: 小程序->首页->DOM->day02

2. 复习第二阶段所有选择器！

3. 看小程序视频，完成标签页效果:

小程序->在线->DOM->day02 4.2 标签页 tabs

直接在DOM day02压缩包中3\_tabs.html中做



4. 预习: 小程序->在线->DOM->day03