小程序: WEB问题速查

1. 问题清单: 历届师兄师姐在今天课程中集中问过的高频的重点的问题！

那些自己发现不了问题，又迫切想知道其它同学都问了什么问题，的同学

小程序->首页->DOM->day02

当老师在回答其它同学的问题时，你不要闲着！你可以打开问题清单浏览别人曾经问过的经典问题！

2. 高清无码小视频: 小程序->在线->DOM->day02

2部分: 1. 课上重点知识和案例的回放！2. 重点高频笔试面试题讲解扩展视频

万一家里或中心网络断了！暂时看不了直播了！

因为某些事儿请假了，错过了当天的直播！

课下复习时，觉得TMOOC视频太大，浏览不方便或TMOOC不能看了！

——打开小程序中"在线"，就可下载观看当前所讲内容和当天所讲全部内容的回放。

基础比较好的，确实学有余力的同学，想拿高薪的同学，在我答疑时，不要闲着:

要看今天扩展视频: 小程序->在线->DOM->day02

1. 动画 开关门效果 door transition

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

6. 带class样式的表单验证(封装函数)

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. 凡是拼写错误，自己懒得找！我可以帮你找，但是找到后，错误的单词你要手抄20遍，并拍照！

出错: 先开f12，console，看错误提示！

3. 每个带注释的案例至少写三遍:

(1). 注释+代码抄一遍

(2). 保留注释，删除代码，自己试着把代码翻译回来！

(3). 删除注释和代码，自己试着用自己的话把注释写回来！

一个例子只有自己可以把人话的注释写回来！才算自己会！

问题: 一听就会一做就废！原因: 你自己无法用人话精确描述自己想干什么！

如何提问: （vip待遇的剩余时间10天）

1. 上课时，案例中，笔记中那句话哪行代码看不懂，立刻问！

2. 小程序中问题清单里哪个问题和回答看不懂！立刻问！

3. 个人项目中哪个功能不会做！立刻问！

4. 刷笔试题时，遇到不会的笔试题！立刻问！

谁问的问题越多！谁的培训费交的越值！

正课:

1. 修改:

一. 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 原始HTML内容...

(1). 获取或修改元素开始标签到结束标签之间的原始HTML内容: **元素.innerHTML**

(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**

|  |
| --- |
| textContent vs innerHTML  1. 获取内容时:  (1). innerHTML: 不做任何加工，直接取出原始的HTML代码内容  (2). textContent: 将原始内容加工后，只返回纯文本正文内容  a. 去掉所有内嵌的标签  b. 将特殊符号翻译为正文  2. 修改内容时:  (1). 交给innerHTML的内容，先交给浏览器解析，将解析后的内容显示给人看  (2). 交给textContent的内容，不交给浏览器解析，而是原样显示内容 |

(3). 获取或设置表单元素的值:

~~a. 因为大部分表单元素, 比如<input>是单标记，所以不能用innerHTML或textContent获得内容~~

b. 正确: **元素.value**

(4). 示例: 获取和修改各种元素的内容和值

0\_html\_text.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>    <p id="p1">来自<a href="javascript:;">&lt;&lt;新华社&gt;&gt;</a>的消息</p>    <p id="p2"></p>    <p id="p3"></p>    <input id="uname" type="text"/>    <button id="btn">提交</button>    <script>      var p1=document.getElementById("p1");      var p2=document.getElementById("p2");      var p3=document.getElementById("p3");      //获取内容:      //想获得p1中的原始的HTML内容      console.log(p1.innerHTML);      //想获得p1中的纯文本内容      console.log(p1.textContent);      //修改内容:      //想修改p2的原始的HTML内容:      p2.innerHTML=`来自<a href="javascript:;">&lt;&lt;新华社&gt;&gt;</a>的消息`;      //想修改p3的纯文本内容:      p3.textContent=`来自<a href="javascript:;">&lt;&lt;新华社&gt;&gt;</a>的消息`;      //想点提交按钮时，获取文本框中用户输入的内容      var btn=document.getElementById("btn");      btn.onclick=function(){        var uname=document.getElementById("uname");        //错误:        // console.log(`你输入的用户名是:${uname.innerHTML}`);        //正确:        console.log(`你输入的用户名是:${uname.value}`);      }    </script>  </body>  </html>  运行结果: |

(5). 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 door ...

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: 用户点击d2触发变化      var d2=document.getElementById("d2");      //2. 绑定事件处理函数      //本例中: 用户单击d2触发变化      d2.onclick=function(){        //3. 查找要修改的元素        //本例中: 要修改d1        var d1=document.getElementById("d1");        //4. 修改元素        //本例中:        //如果d1是关着的(d1的style中的display属性为none)        if(d1.style.display=="none"){          //就打开d1(清除d1的style中的display属性为"")          d1.style.display="";          //顺便将d2的内容改为<<,表示下次再点就是关闭          this.innerHTML="&lt;&lt;";        }else{//否则如果d1是开着的，          //就关上d1(修改d1的style中的display属性为none)          d1.style.display="none";          //顺便将d2的内容改为>>,表示下次再点就是打开          this.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: 小程序->在线->DOM->day02 0.2 修改元素的属性 字符串类型的HTML标准属性

a. 什么是: HTML标准中规定的大部分元素都有的值为字符串的HTML属性

b. 比如: title id name href src ...

c. 如何: 2种:

1). 早期核心DOM四个函数:

i. 获取属性值: var 属性值=元素.getAttribute("属性名")

获取 属性

ii. 修改属性值: 元素.setAttribute("属性名","属性值")

设置 属性

iii. 判断是否包含某个属性: var bool=元素.hasAttribute("属性名")

包含 属性

iv. 移除属性: 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">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);      //想判断a1是否包含title属性      var bool=a1.hasAttribute("title");      console.log(bool);//false      //想为a1设置title属性      a1.setAttribute("title","欢迎访问tmooc")      console.log(a1.outerHTML);//输出自己的HTML元素代码      //再判断a1是否包含title属性      var bool=a1.hasAttribute("title");      console.log(bool);//true      //想移除a1的title属性      a1.removeAttribute("title")      console.log(a1.outerHTML);//输出自己的HTML元素代码      //再判断a1是否包含title属性      var bool=a1.hasAttribute("title");      console.log(bool);//false    </script>  </body>  </html>  运行结果： |

2). 新版HTML DOM简化方式: 已经提前将元素所有可用的HTML标准属性都添加到内存中元素节点对象上！只不过页面上**暂时没有出现的属性，值暂时为""**而已。所以，我们可以用"**元素.属性名**"来访问HTML标准属性

i. 获取属性值: **元素.属性名**

ii. 修改属性值: **元素.属性名="属性值"**

iii. 判断是否包含某个属性: **元素.属性名!==""**

iv. 移除属性: **元素.属性名=""**

v. 特例: class属性:

html中: <元素 class="class名">

js中: ~~元素对象.class="class名"~~

ES6中，class是js的关键字，表示创建一种类型的意思！

DOM中，元素对象.className

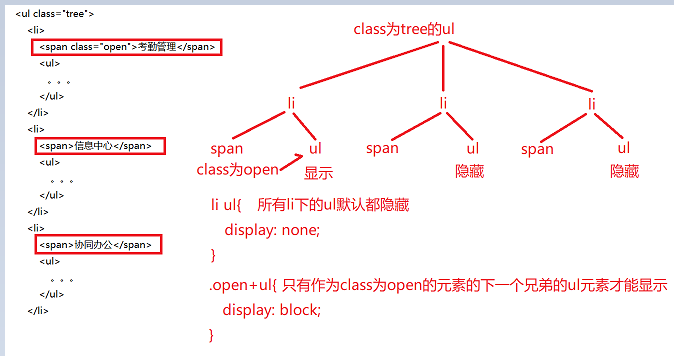
vi. 示例: 使用HTML DOM方式简化操作元素属性

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">go to tmooc</a>    <script>      var a1=document.getElementById("a1");      console.dir(a1);      //想获得a1的href属性值:      console.log(a1.href);      //想判断a1是否包含title属性      console.log(a1.title!=="");//false      //想为a1设置title属性      a1.title="欢迎访问tmooc";      console.log(a1.outerHTML);      //再判断a1是否包含title属性      console.log(a1.title!=="");//true      //想移除a1的title属性      a1.title="";      console.log(a1.outerHTML);      //再判断a1是否包含title属性      console.log(a1.title!=="");//false    </script>  </body>  </html>  运行结果: |

d. 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>      //需求:      //点击一级菜单:      //  如果自己(当前span)是开着的，只关闭自己（当前span）即可，不用管别人      //  否则如果自己(当前span)是关着的，就需要做两件事        //  先找到另一个开着的菜单(另一个class为open的span),把它先关上        //  再把自己(当前span)打开      //结果: 同一时刻最多只可能有一个菜单是开着的！      //      也有可能所有菜单都关上      //DOM 4步      //1. 查找触发事件的元素      //本例中: 用户点class为tree的ul下的每个li下的span触发变化      var spans=document.querySelectorAll("ul.tree>li>span");      //2. 绑定事件处理函数      //本例中: 每个span都可以单击      //遍历spans类数组对象中每个span元素      for(var span of spans){        //每遍历一个span元素，都要绑定单击事件        span.onclick=function(){          //测试: 点哪个span，让哪个span的内容变成❀          //this.innerHTML="❀";          //3. 查找要修改的元素          //本例中: 注定要修改的就是当前span自己——this          //所以不用找！          //4. 修改元素:          //如果当前span是开着的(如果当前span的class是open，则当前span旁边的ul一定是开着的)          if(this.className=="open"){            //只关闭自己即可            this.className=""; //清除当前span的class open，当前span旁边的ul，就默认隐藏了！          }else{//否则如果当前span是关着的            //先尝试查找另一个开着的span——注定最多只能找到一个            //查找class为tree下的class为open的span            var openSpan=document.querySelector(              "ul.tree>li>span.open"            );//如果没找到返回null            //如果找到另一个开着的span            if(openSpan!=null){              //就把另一个开着的span先关上              openSpan.className="";//清除另一个开着的span的class open，则另一个开着的span旁边的ul默认隐藏            }            //无论前边是否找到另一个开着的span。最后，只要自己是关着的，都要把自己打开！            this.className="open";//为当前span自己加class open，当前span旁边的ul就受牵连而显示！          }        }      }      //现在还不到你们原创的时候！      //现阶段！就是处在照猫画虎的阶段！      //你们要做的是反复练习人话！    </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类型的HTML标准属性

a. 什么是: HTML标准中规定的，但是不需要指定属性值，只要放在元素上就起作用的属性！

b. 比如: disabled 禁用 checked选中 ...

c. 如何: 只有一种访问方式:

~~1). 错误做法: 不能用核心DOM4个函数访问。因为核心DOM4个函数只支持字符串类型的属性值！不支持bool类型的属性值！~~

2). 正确做法: 只能用HTML DOM "**元素.属性名**"，且属性值都是bool类型的true或false！

d. 示例: 使用HTML DOM访问元素的bool类型的HTML标准属性:

2\_bool\_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>    <input id="chb" type="checkbox">同意    <button id="btn1">提交</button>    <script>      var btn1=document.getElementById("btn1");      btn1.onclick=function(){        var chb=document.getElementById("chb");        console.log(chb.checked==true?"选中":"未选中");      }    </script>  </body>  </html>  运行结果: |

e. 补充: CSS3中提供了一个选择器——：checked，专门匹配选中的checkbox或radio元素

f. 示例: 全选和取消全选  
 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>    <button>删除选定</button>    <script>      //需求:      //1. 点表头中的checkbox，可控制表体中每行第一个checkbox的选中状态      //2. 点表体中每行第一个checkbox，有可能反过来影响表头中的checkbox的选中状态      //需求一: 点表头中的checkbox，可控制表体中每行第一个checkbox的选中状态      //DOM 4步      //1. 查找触发事件的元素      //本例中: 用户点表头中的input，触发下边input的变化      var chbAll=document.querySelector("thead>tr>th:first-child>input");      console.log(chbAll);      //2. 绑定事件处理函数      //本例中: 用户单击全选按钮，触发下边的变化      chbAll.onclick=function(){        //this->全选按钮chbAll        //3. 查找要修改的元素        //本例中: 点表头中的全选，要修改表体中每行第一个td中的input        var chbs=document.querySelectorAll("tbody>tr>td:first-child>input");        //4. 修改元素        //如果全选按钮是选中的          //则下边每个按钮也都要选中        //否则如果全选按钮时未选中的          //则下边每个按钮也都要取消选中        //结论: 其实就是让下边每个chb的checked属性值和全选按钮的checked属性值保持一致！        for(var chb of chbs){          chb.checked=this.checked;        }      }        //需求二: 点表体中每行第一个checkbox，有可能反过来影响表头中的checkbox的选中状态      //DOM 4步:      //1. 查找触发事件的元素      //本例中: 用户点的是表体中每行第一个td中的checkbox      var chbs=document.querySelectorAll("tbody>tr>td:first-child>input");      //2. 绑定事件处理函数      //本例中: 表体中每个chb都可以点击      //遍历chbs中每个chb      for(var chb of chbs){        //每遍历一个chb元素，为其就绑定单击事件        chb.onclick=function(){          //3. 查找要修改的元素          //本例中: 当单击下方每个chb时，都有可能修改表头中第一个th中的input          var chbAll=document.querySelector("thead>tr>th:first-child>input");          //4. 修改元素          //尝试在tbody中查找一个未选中的checkbox          var unchecked=document.querySelector(            "tbody>tr>td:first-child>input:not(:checked)"          );          //如果找到了这样一个checkbox，则chbAll不选中。          if(unchecked!=null){            chbAll.checked=false;          }else{//否则如果没找到这样一个checkbox，则chbAll选中            chbAll.checked=true;          }        }      }    </script>  </body>  </html>  运行结果: |

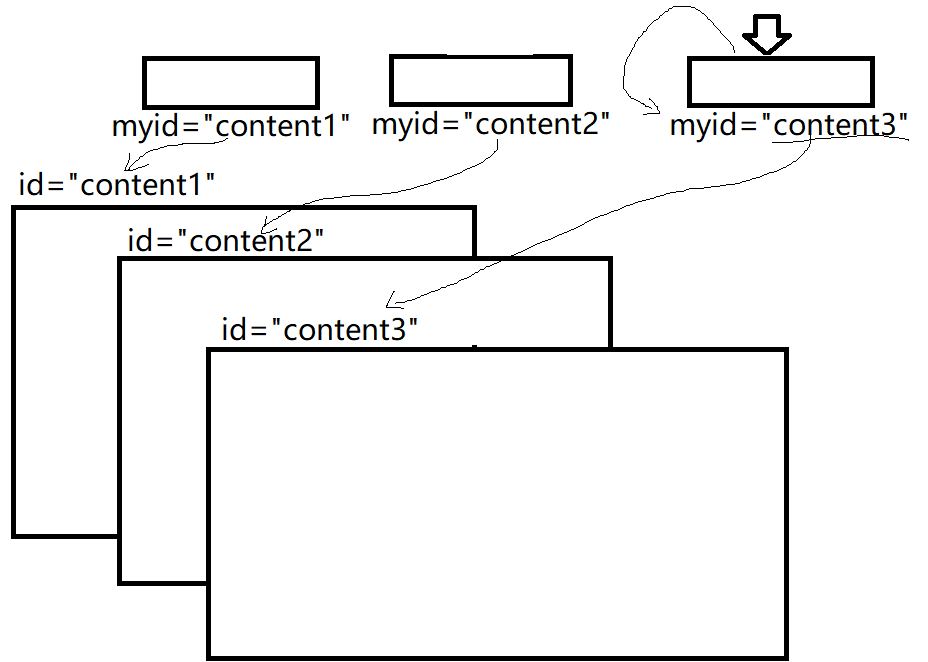
(3). **自定义**扩展属性:

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

a. 什么是: HTML标准中没有定义的，程序员根据自身需要，自发添加的自定义属性。

b. 何时: 2个场景

1). 在HTML元素上缓存一些业务相关的数据



2). 代替其他选择器，专门作为**查找触发事件**的元素的条件

i. 其他选择器的问题:

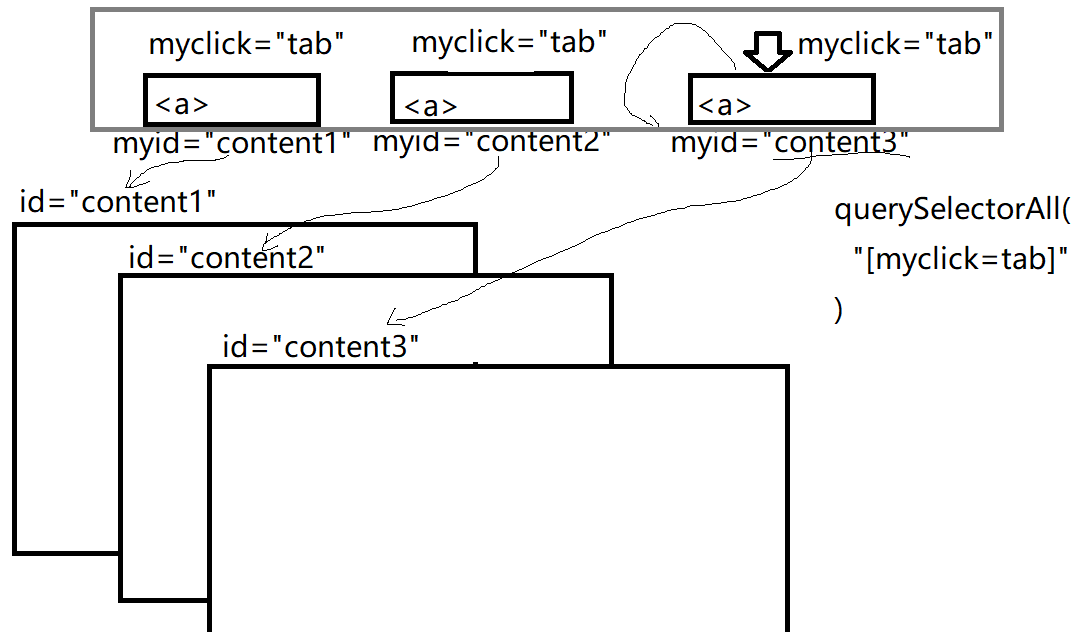
id选择器: 一次只能选一个元素，不能选择多个元素

元素选择器: 实现同一种效果，不同的人使用的元素可能各不相同！

类选择器: 本职工作就不是为js服务的！类选择器的本质工作是为了给元素添加样式！所以，在类选择器中如果出现与css无关的class名，很可能被轻易删掉。此时，如果你的js刚好用到这个被删除的类名，则js程序立刻出错！

ii. 以上三种选择器，尽量不要用于DOM中查找触发事件的元素！

iii. 今后在DOM中查找触发事件的元素最好都用自定义扩展属性



c. 如何使用自定义扩展属性: HTML5标准

1). 在HTML中元素上添加自定义扩展属性:

**<元素 data-自定义属性名="属性值">**

2). 在js中操作元素的自定义扩展属性:

~~i. 错误: 不能用.直接访问自定义扩展属性~~

因为自定义扩展属性是后天添加的，不在HTML标准范围内。所以HTML DOM无法提前预知自定义属性，也就无法自动添加到内存中的元素对象上。

ii. 正确: 2种:

①可以用旧式的**核心DOM4个函数**来操作——没有兼容性问题

获取: 元素对象.getAttribute("data-自定义属性名")

修改: 元素对象.setAttribute("data-自定义属性名","属性值")

强调: 如果用核心DOM4个函数，自定义属性名前必须加data-才行

②可以用HTML5提供的新的简写方式——兼容性问题

如果html中元素上定义自定义扩展属性时，开头加了data-前缀，则可用:

**元素对象.dataset.自定义属性名**

强调:

* 使用**dataset**时，**不用加**data-前缀！dataset会自动查找data-前缀的属性
* 只有加了data-前缀的自定义属性，才能被dataset访问到！

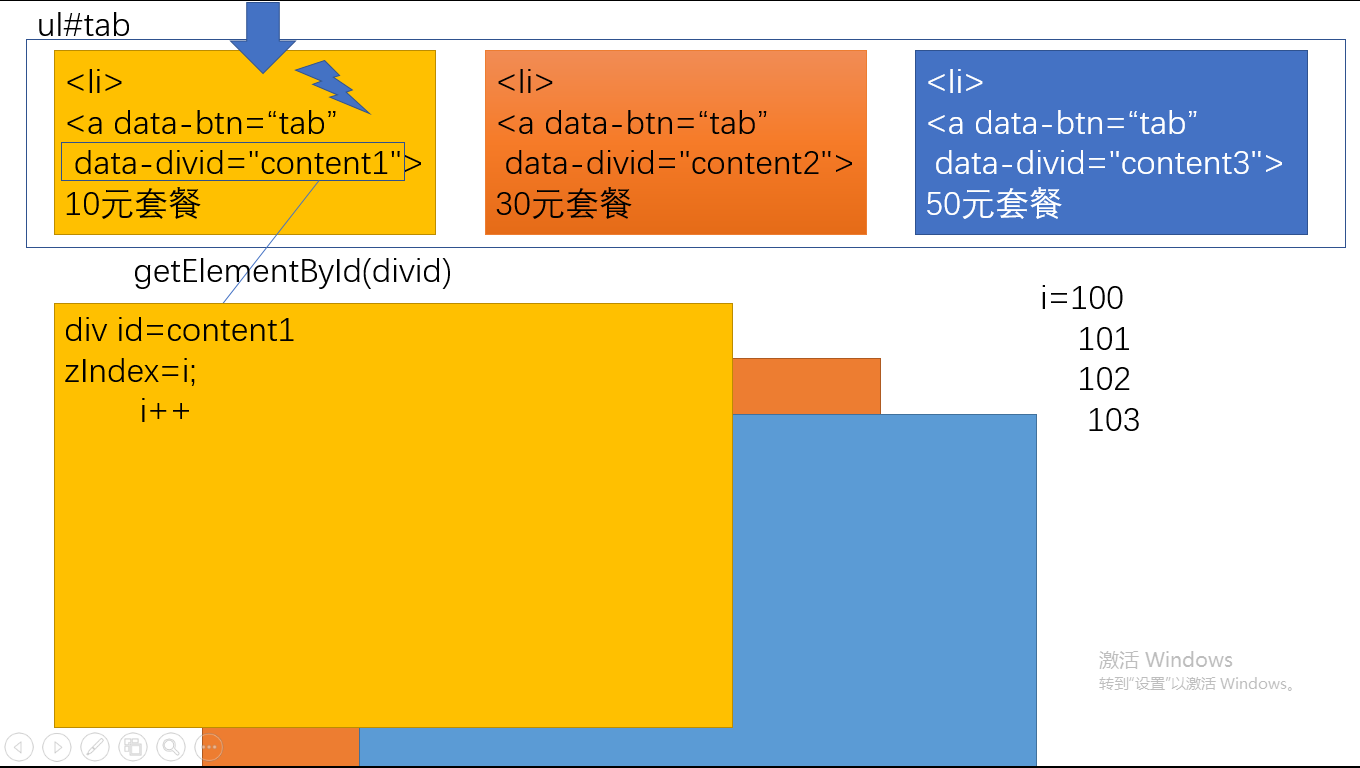
d. 示例: 使用自定义扩展属性查找元素，并操作元素的自定义扩展属性

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 id="a1" data-click="btn" data-content="欢迎访问tmooc" href="javascript:;">go to tmooc</a>    <script>      //想查找带有data-click属性的，且属性值为btn的a元素      var btn=document.querySelector("[data-click=btn]");      console.log(btn.outerHTML);      //想获得data-content属性的值      //console.log(btn.getAttribute("data-content"));      console.log(btn.dataset.content);      //想修改data-content属性的值为Welcome to tmooc      //btn.setAttribute("data-content","welcome to tmooc");      btn.dataset.content="welcome to tmooc";      console.log(btn);    </script>  </body>  </html>  运行结果:  <a id="a1" data-click="btn" data-content="欢迎访问tmooc" href="javascript:;">go to tmooc</a>  欢迎访问tmooc  <a id=​"a1" data-click=​"btn" data-content=​"welcome to tmooc" href=​"javascript:​;​">​go to tmooc​</a>​ |

e. 示例: 标签页效果:

对应小程序视频: 小程序->在线->DOM->day02 4.2 标签页 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 href="#content1" data-btn="tab" data-divid="content1">10元套餐</a></li>        <li><a href="#content2" data-btn="tab" data-divid="content2">30元套餐</a></li>        <li><a href="#content3" data-btn="tab" data-divid="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>      //准备工作:      //1. 为每个触发事件的按钮添加自定义属性，用于查找触发事件的元素      //2. 为每个按钮添加自定义扩展属性，提前保存每个按钮对应的div的id      //DOM 4步      //1. 查找触发事件的元素      //本例中: 用户点ul下的包含data-btn属性，且属性值为tab的按钮元素触发变化      var tabs=document.querySelectorAll(`[data-btn=tab]`);      console.log(tabs);      //先定义一个变量i=100，记录目前最大的zIndex值      var i=100;//每点一次标签页，都i++。      //结果，下次点标签页时的i，一定比上次点标签页时i大1      //2. 绑定事件处理函数      //本例中: 每个标签都可以点      //遍历tabs中每个标签按钮      for(var tab of tabs){        //每遍历一个标签按钮就要为其绑定单击事件        tab.onclick=function(){          //3. 查找要修改的元素          //本例中: 找到当前标签按钮对应的div          //2步:          //3.1 从当前按钮自己身上的data-divid属性中取出对一个div的id名          var divid=this.getAttribute("data-divid");                  //this.dataset.divid          //3.2 用拿到的divid，查找对应的div元素对象          var div=document.getElementById(divid);          //4. 修改元素          //本例中: 修改当前按钮对应的div的zIndex值最大！就可以让当前按钮对应div到最上方来，挡住其它div          div.style.zIndex=i;          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

(1). 修改元素的内联样式:

元素的style属性中的css属性为属性值

a. html中: <元素 style="css属性:属性值; ..."></元素>

b. js中: 元素对象.style.css属性="属性值";

的 的

c. 坑: 有些css属性名中带-，-会和js中的减法的减号冲突！

d. 解决: 所有css属性名中带-的属性，都要去横线变驼峰

比如: background-color backgroundColor

font-size fontSize

e. 强调: 大小，长短，位置有关的css属性值必须加px单位！不加单位不生效！

总结: DOM 4步:

1. 先查找可能触发事件的元素

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

4.1 取出元素的旧值或旧内容

4.2 计算/修改

4.3 将新值放回元素的内容或属性上

总结: DOM 5件事: 增删改查+事件绑定:

1. 查找元素:

(1). 不需要查找就可直接获得的节点对象: 4种:

document —— 唯一的树根节点

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

父子关系: 4种

元素的父元素: 元素.parentNode或元素.parentElement

元素下的所有直接子元素: 元素.children

元素下的第一个直接子元素: 元素.firstElementChild

元素下的最后一个直接子元素: 元素.lastElementChild

兄弟关系: 2种

元素的前一个兄弟: 元素.previousElementSibling

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:

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

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

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

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

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")

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: 查找方法的返回值规律: 3种  1. 如果原方法返回下标位置i，则如果找不到返回-1  2. 如果原方法返回一个数组或对象，则如果找不到返回null  3. 如果原方法返回类数组对象，则如果找不到返回空类数组对象: { length:0 } |

2. 修改元素: 3

(1). 修改内容: 3种

a. 获取或修改元素的原始HTML内容: 元素.innerHTML

b. 获取或修改元素的纯文本内容: 元素.textContent

c. 获取或修改表单元素的值: 元素.value

(2). 修改属性: 3种:

a. 字符串类型的HTML标准属性: 2种方式  
 1). 核心DOM 4个函数:

i. 获取属性值: var 属性值=元素.getAttribute("属性名")

ii. 修改属性值: 元素.setAttribute("属性名","属性值")

iii. 判断是否包含某个属性: var bool=元素.hasAttribute("属性名")

iv. 移除属性: 元素.removeAttribute("属性名")

2). HTML DOM简写:

i. 获取属性值:元素.属性名

ii. 修改属性值: 元素.属性名="属性值"

iii. 判断是否包含某个属性:元素.属性名!==""

iv. 移除属性: 元素.属性名=""

b. bool类型HTML标准属性:

只能用"元素.属性名"，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:

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: (待续...)

c. 使用class批量修改元素的样式: 元素.className="class名"

3. 添加/删除元素:(待续...)

4. 事件绑定: 3种:

(1). 在HTML中绑定: DOM和jquery中几乎不用

HTML中: <元素 on事件名="事件处理函数()">

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->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

4.2 标签页 tabs

作业:

1. 复习小程序中问题清单: 小程序->首页->DOM->day02

2. 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

3. 看小程序视频使用HTML5完成双态按钮效果

小程序->在线->DOM->day02 扩展: 如何单独操作某一个class HTML5 classList

4. 看小程序视频预习day03内容:

小程序->在线->DOM->day02 5 修改样式，计算属性

6 带class样式的表单验证 valiWithCss

//使用day03/3\_valiWithCss.html 直接做

小程序->在线->DOM->day03 1 DOM 添加新元素 3步 动态创建表格

//使用day03/1\_createTable.html 直接做

4. 打开新链接4种方式

//使用day03/4\_open.html 直接做