小程序: WEB问题速查

1. 问题清单: 小程序->首页->DOM->day03、day04

2. 高清无码小视频: 小程序->在线->DOM-> day03、day04

今日扩展视频:

小程序->在线->DOM->day03、day04

3. 利用冒泡 事件委托 ... 优化购物车, 标签页, 表格删除行, 。。。

5. 鼠标坐标... 原生DOM拖拽效果

所有以"作业"和"扩展"开头的视频

赠送: 原生DOM轮播图

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->day03、day04

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. 现在正处在照猫画虎的阶段！每个DOM效果至少写3遍:

3.1 注释+代码抄一遍！

3.2 保留注释，删除代码，把代码再填回来！

3.3 删除注释和代码，试着用自己的话把注释写回来，根据自己的注释，把代码填回来

一个DOM效果，只有自己能写出人话注释，才算自己会！——标准！

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. HTML DOM常用对象

2. 什么是BOM

3. window对象

4. history

5. location

6. navigator

7. \*\*\*\*\*事件\*\*\*\*\*

一. HTML DOM常用对象: (了解)

对应小程序视频: 小程序->在线->JSCORE->day03: 3 HTML DOM table

1. table元素: 逐级管理:

(1). table管着行分组:

a. 创建:

1). var thead=table.createTHead();

i. 旧核心DOM: 2步:

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

table.appendChild(thead);

ii. 新HTML DOM中: 一句话办两件事: 既创建又追加

2). var tbody=table.createTBody();

3). var tfoot=table.createTFoot();

b. 删除:

1). table.deleteTHead()

2). table.deleteTFoot()

c. 获取:

1). table.tHead

2). table.tFoot

3). table.tBodies[i]

(2). 行分组管着行:

a. 创建行:

1). var tr=行分组.insertRow(i)

插入 行

2). 在行分组内下标为i的位置，插入一个新行

3). 一句话干两件事:

i. 相当于: var tr=document.createElement("tr");

ii. 相当于: 行分组.appendChild(tr);

4). 强调: 如果原来i位置已经有旧行，则新插入的行将原i位置的行及其之后所有行，都向后挤一位。

5). 固定套路:

i. 在行分组末尾追加一个新行:

var tr=行分组.insertRow() //不写参数，默认末尾追加

ii. 在行分组开头插入一个新行:

var tr=行分组.insertRow(0)

b. 删除行:

1). 行分组.deleteRow(i)

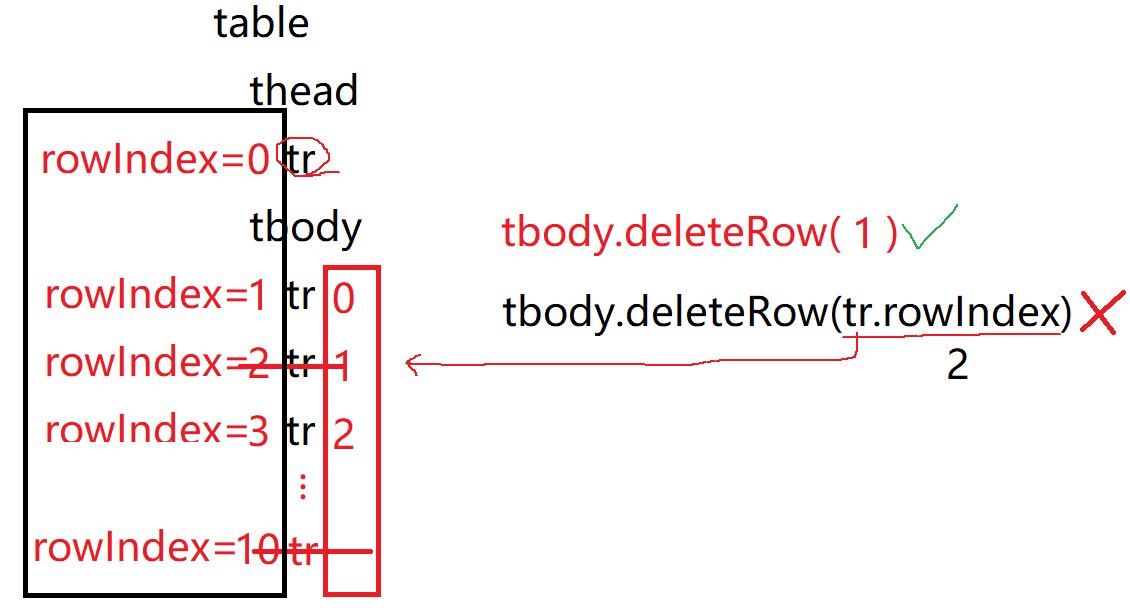
2). 意为: 删除行分组内，下标为i位置的行

3). 强调: i, 要求是行在行分组内的相对下标位置。

4). 问题: 无法提前预知并写死要删除行的下标！通常都是客户点哪一行，就删除哪一行！如何自动获得客户所点的行的下标位置？

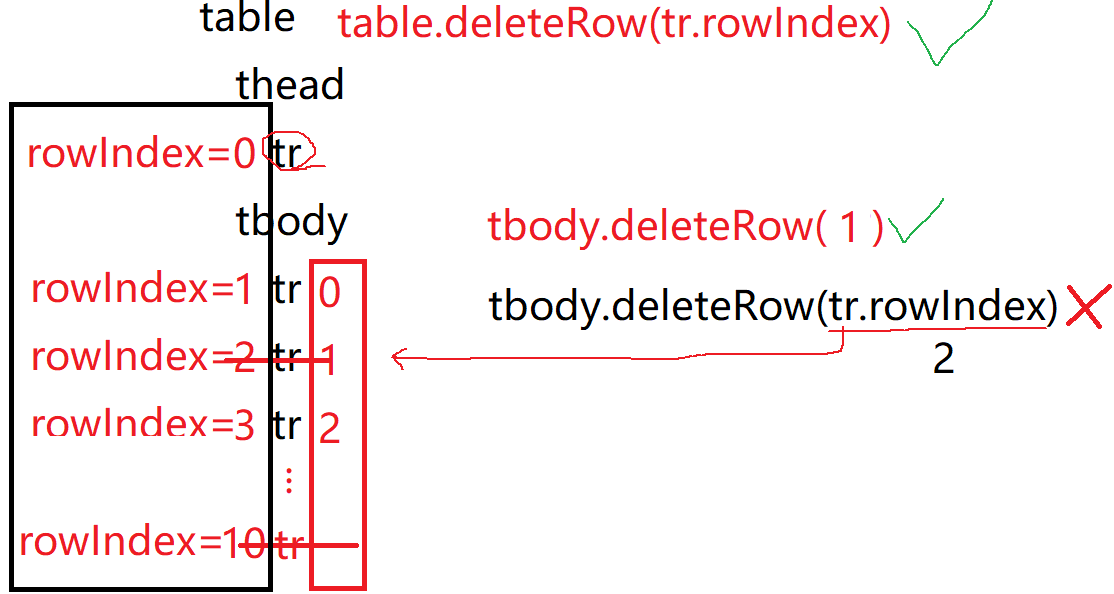
5). 解决: 只要我们可以获得当前用户点的行对象，那么每个行对象上，都有一个rowIndex属性，记录了行在整个表范围内的下标位置！

6). 问题: 如果.前的主语是行分组，则需要提供要删除的行在行分组内相对下标位置。而rowIndex只能获得行在整个表中的绝对下标位置。而受到表头行的影响，很可能行在行分组内的相对位置与行在整个表中的绝对下标位置是错误的！



7). 最终: 今后只要删除行，不要用行分组作为主语。应该用table作为主语！table.deleteRow(i)恰好需要的是行在整个表中的下标位置。与rowIndex表示的下标就一致了:

table.deleteRow(tr.rowIndex);



c. 获取行: 行分组.rows[i]

(3). 行管着格:

a. 添加格: var td=tr.insertCell() //末尾添加一格

插入 格

b. 删除格: tr.deleteCell(i)

c. 获取格: tr.cells[i]

(4). 示例: 动态生成表格，并删除行

day03剩余/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>姓名</th>            <th>薪资</th>            <th>年龄</th>            <th></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 }      ];      //0. 先找到id为data的div下的table元素      var table=document.querySelector(        "#data>table"      )      //1. 再创建tbody，暂时不要添加到DOM树上      var tbody=document.createElement("tbody");        //2. 然后遍历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，追加到tbody下        // var tr=document.createElement("tr");        // tbody.appendChild(tr);        var tr=tbody.insertRow();        //3. 遍历当前员工对象中每个属性        //每遍历一个属性，就创建一个td，追加到tr中，并设置td的内容为当前对象的当前属性值        for(var key in emp){          // var td=document.createElement("td");          // tr.appendChild(td);          var td=tr.insertCell();          td.innerHTML=emp[key];        }        //在为当前行添加完所有的数据格之后，再额外添加一个单元格td        // var td=document.createElement("td");        // tr.appendChild(td);        var td=tr.insertCell();        //创建一个按钮，放入格中:        var btn=document.createElement("button");        btn.innerHTML="×";        td.appendChild(btn);        //DOM 4步:        //1. 查找触发事件的元素: btn        //2. 绑定事件处理函数        btn.onclick=function(){          //this->当前点击的这个删除按钮          //3. 查找要修改的元素          //本例中: 点按钮删除当前按钮所在的当前行          var tr=            this.parentElement.parentElement;          // btn     td             tr          var ename=//tr.children[0].innerHTML;                    tr.cells[0].innerHTML;          //4. 修改元素          var bool=confirm(`是否继续删除${ename}吗？`);          if(bool==true){            table.deleteRow(tr.rowIndex);          }        }      }      //当所有子元素tr和td都加入到tbody之后，最后将tbody一次性添加到DOM树。      table.appendChild(tbody);    </script>  </body>  </html>  运行结果: |

|  |
| --- |
| 补: 浏览器中三大对话框:  1. 输入框: var input=prompt("提示信息");    2. 警告框: alert("警告信息")    3. 确认框: var bool值=confirm("提示信息");    如果用户点确定按钮，就返回true；否则如果用户点取消按钮，就返回false。 |

二. 什么是BOM: Browser Object Model

浏览器 对象 模型

1. 什么是: 专门操作浏览器软件或窗口的一组对象和函数的集合.

2. 何时: 今后只要希望获得浏览器软件的信息或希望操作浏览器窗口时，都用BOM

3. 问题: 没有标准，极大兼容性问题，所以，将来用的越来越少了！——学习时，不要扩展！

4. 包括:

window, history, location, navigator, document, screen, event

三. window对象:

1. 3个角色

(1). 代替ES标准中的global，充当全局作用域对象。

(2). 还包含了所有原生的对象和函数: ES+DOM+BOM

比如: parseInt() document alert()

(3). 代表了当前正在打开的这个浏览器窗口！

a. 获得浏览器窗口的大小: 2组:

1). 完整窗口大小:

window.outerWidth, window.outerHeight

2). 文档显示区大小:

i. 文档显示区: 浏览器窗口中专门用于显示网页的区域.

ii. window.innerWidth, window.innerHeight

b. 还可以打开和关闭窗口:

1). 打开窗口: window.open()

2). 关闭窗口: window.close()

2. 所有打开新链接的方式: 4种:

对应小程序视频: 小程序->在线->DOM->day03 4. 打开新链接4种方式

(1). 在当前窗口打开新链接，可后退

a. html: <a href="url" target="\_self">

b. js: window.open("url", "\_self");

(2). 在当前窗口打开新链接，禁止后退

a. html:

b. js:

(3). 在新窗口打开新链接，可打开多个

a. html: <a href="url" target="\_blank">

b. js: window.open("url", "\_blank")

(4). 在新窗口打开新链接，只能打开一个

a. html: <a href="url" target="blank">

b. js: window.open("url", "\_blank")

(5). 原理:

a. 浏览器中每打开一个新窗口，浏览器都会自动给窗口分配一个随机的名称。这个随机的名称就是窗口在内存中的唯一表示。人看不见这个随机名称

b. 其实，当我们用<a>或window.open()打开新窗口时，也可以给新窗口起自定义的名称:

1). <a href="url" target="自定义新窗口名">

2). window.open("url", "自定义新窗口名")

3). 我们可以通过新窗口的window.name查看新窗口自定义的窗口名。

4). 所以，今后，我们自己定义变量或属性名时，禁止使用"name"作为变量名或属性名。

c. 但是，浏览器有个规定，相同窗口名的窗口只能打开一个。后打开的同名窗口，会覆盖先打开的同名窗口，来保持只打开一个。

d. 为什么写\_self和写\_blank可以完成不同的功能？

1). \_self 表示用旧窗口的名字作为新窗口的名称，结果，新窗口会覆盖旧窗口——效果好像是在当前窗口打开

2). \_blank 表示不给新窗口起名。浏览器会随机给窗口命名。因为随机命名的窗口名不重复，所以，可以反复打开多个！

对应小程序视频: 小程序->在线->DOM->day04 0. history, location, navigator

四. history:

五. location对象:

六. navigator

七. \*\*\*\*\*事件\*\*\*\*\*

对应小程序视频：小程序->在线->DOM->day04 1. 添加事件监听，移除事件监听 ...

对应小程序视频: 小程序->在线->DOM->day04 2. 事件模型

对应小程序视频: 小程序->在线->DOM->day04 3. 利用冒泡 事件委托

对应小程序视频: 小程序->在线->DOM->day04 4. 阻止默认行为

对应小程序视频: 小程序->在线->DOM->day04 5. 鼠标坐标x y 原生DOM

总结: 不要背英文名字！反而应该记中文能做哪些事儿！

(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). 何时: 2种

i. 代替id、class、元素等选择器作为查找触发事件的元素的条件

ii. 在客户端元素上临时缓存业务数据

2) HTML中: <元素 data-自定义属性名="属性值">

3). 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"

总结: 不要背英文名字！反而应该记中文能做哪些事儿！

3. 添加/删除元素:

(1). 只添加一个新元素: 3步

a. 创建一个新元素:

var 新元素=document.createElement("标签名")

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DOM树: 3种:

1). 末尾追加:

父元素.appendChild(新元素)

2). 在某个元素前插入:

父元素.insertBefore(新元素, 现有元素)

3). 替换某个元素:

父元素.replaceChild(新元素, 现有元素)

(2). 优化: 尽量减少操作DOM树的次数，2种:

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，最后再将父元素一次性添加到DOM树

b. 如果父元素已经在页面上，要添加多个平级子元素。应该利用文档片段对象

1). 创建文档片段对象:

var frag=document.createDocumentFragment()

2). 将子元素添加到文档片段对象中:

frag.appendChild(子元素)

3). 最后将文档片段对象一次性添加到DOM树上父元素下  
 父元素.appendChild(frag);

(3). 删除元素: 父元素.removeChild(子元素)

4. HTML DOM常用对象:（了解即可）

(1). var img=new Image()

(2). table  
a. table管着行分组:

1). 添加行分组:

var thead=table.createTHead()

var tbody=table.createTBody()

var tfoot=table.createTFoot()

2) 删除行分组:

table.deleteTHead(); table.deleteTFoot()

3). 获取行分组:

table.tHead table. tFoot table.tBodies[i]

b. 行分组管着行:

1). 添加行:

i. 任意行插入新行: var tr=行分组.insertRow(i);

ii. 开头插入新行: var tr=行分组.insertRow(0)

iii. 末尾追加新行: var tr=行分组.insertRow()

2). 删除行: table.deleteRow(tr.rowIndex)

3). 获取行: 行分组.rows[i]

c. 行管着格:

1). 添加格: var td=tr.insertCell()

2). 删除格: tr.deleteCell(i)

3). 获取格: tr.cells[i]

(3). form:

a. 获取form元素: document.forms[i]

b. 获取form中的表单元素:

1). 标准: form.elements[i或id或name名]

2). 简写: 如果有name属性: form.name名

c. 让表单元素自动获得焦点: 表单元素.focus()

总结:

BOM:

1. window:

(1). 获得窗口大小:

a. 获得完整窗口大小:

window.outerWidth和window.outerHeight

b. 获得文档显示区大小:

window.innerWidth和window.innerHeight

(2). 打开和关闭窗口:

window.open()和window.close()

2. 打开新链接4种方式:

(1). 在当前窗口打开，可后退

a. html: <a href="url" target="\_self">

b. js: window.open("url", "\_self");

(2). 在当前窗口打开，禁止后退

a. js: location.replace("新url")

(3). 在新窗口打开，可同时打开多个

a. html: <a href="url" target="\_blank">

b. js: window.open("url", "\_blank");

(4). 在新窗口打开，只能打开一个

a. html: <a href="url" target="自定义窗口名">

b. js: window.open("url", "自定义窗口名")

3. history:

(1). 前进: history.go(n)

(2). 后退: history.go(-n)

(3). 刷新: history.go(0)

4. location:

(1). 属性: 分段获得url中各个部分:

a. location.href 完整url

b. location.protocol 协议

c. location.host 主机名+端口号

d. location.hostname 主机名

e. location.port 端口号

f. location.pathname 相对路径

g. location.search ?及其之后的查询字符串参数列表

h. location.hash #锚点地址

(2). 方法:

a. 在当前窗口打开，可后退：

location.assign("新url") 或 location.href="新url"

b. 在当前窗口打开，禁止后退:

location.replace("新url")

c. 刷新: location.reload();

5. navigator

(1). 查看浏览器的名称和版本号: navigator.userAgent

(2). 查看浏览器中安装的插件列表: navigator.plugins

总结: 事件:

1. 绑定事件: js中:

(1). 一个事件只绑定一个处理函数

元素.on事件名=function(){ ... }

(2). 一个事件绑定多个处理函数

元素.addEventListener("事件名", 事件处理函数)

(3). 移除一个事件监听:

元素.removeEventListener("事件名", 原事件处理函数对象)

2. 事件模型: 捕获，目标触发，冒泡

3. 事件对象:

(1). 获得事件对象:

元素.on事件名=function(e){ ... }

(2). 阻止冒泡: e.stopPropagation()

(3). 当多个子元素都要绑定相同事件时，利用冒泡/事件委托3步:

a. 事件只在父元素上绑定一次

b. e.target代替this

c. 判断e.target的任意特征是否是我们想要的元素

(4). 阻止元素默认行为:

e.preventDefault()

(5). 获取鼠标位置:

a. 相对于屏幕左上角的x，y坐标:

e.screenX, e.screenY

b. 相对于文档显示区左上角的x，y用坐标:

e.clientX, e.clientY

c. 相对于事件所在元素左上角的x，y坐标:

e.offsetX e.offsetY

今日对应小程序视频列表:

小程序->在线->DOM->day03 4. 打开新链接4种方式

小程序->在线->DOM->day04 0. history, location, navigator

1. 添加事件监听，移除事件监听 ...

2. 事件模型

3. 利用冒泡 事件委托

4. 阻止默认行为

5. 鼠标坐标x y 原生DOM

作业:

1. 复习今日问题清单: 小程序->首页->DOM->day03、day04

2. 看小程序视频，实现完整计算器效果:

小程序->在线->DOM->day04 3. 利用冒泡 事件委托 delegate 计算器...

3. 看小程序串讲视频，总结DOM所有知识点:

小程序->在线->DOM->day04 总结. 1小时DOM知识点串讲

4. （凡是觉得知识点串不起来）看小程序视频学习DOM版学子商城项目(持续到下周一之前)

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:

从小程序中网盘地址中下载xzserver\_start.zip，看视频中day01和day02，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：

从小程序中网盘地址中只下载public.zip，看视频中day01和day02，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