小程序: WEB问题速查

1. 问题清单: 小程序->首页->DOM->day01

2. 高清无码小视频: 小程序->在线->DOM->day01

今日扩展视频:

小程序->在线->DOM->day01 所有以"作业"开头的视频

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->day01

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. 什么是DOM:

2. DOM树:

3. 查找元素:

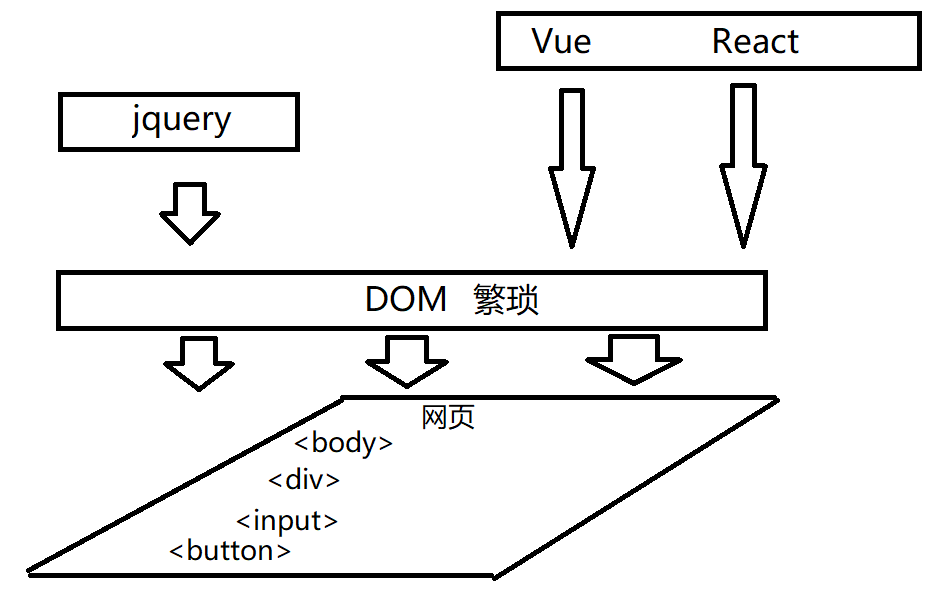
4. 购物车案例:

一. 什么是DOM: Document Object Model

文档 对象 模型

1. 什么是DOM: 一套专门操作网页内容的对象和函数的集合

2. 何时: 今后只要想用js程序操作网页内容，都必须使用DOM提供的对象和方法。



3. DOM标准: W3C制定的专门操作网页的内容的对象和函数的标准

4. 好处: 几乎所有浏览器100%兼容DOM标准。

5. 能做什么: 增删改查+事件绑定

二. DOM树:

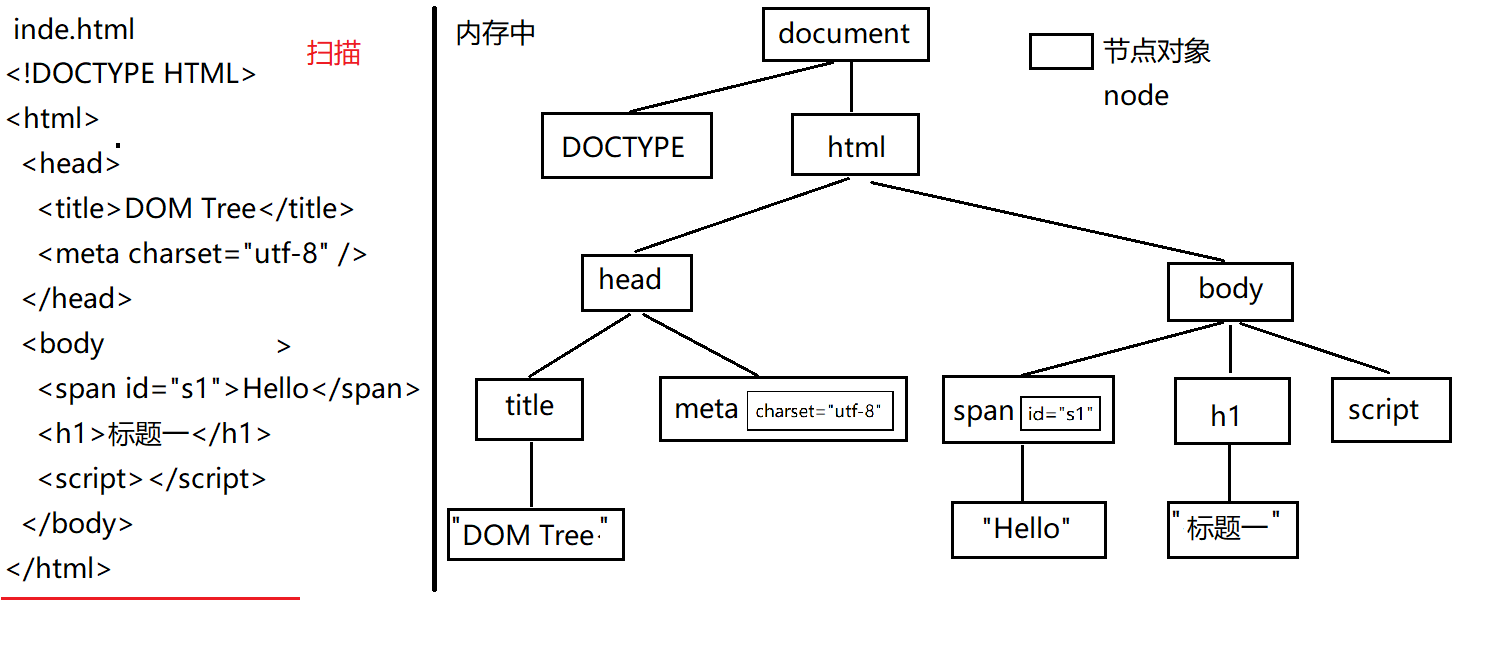
对应小程序视频: 小程序->在线->DOM->day01 0 DOM Tree 树

1. 什么是DOM树: 在内存中，专门保存一个网页中所有内容的树形结构

2. 为什么: 树形结构可以直观的展现上下级包含关系！又因为网页的内容刚好也是上下级包含关系的。

3. 如何:   
 (1). 每当浏览器读取到一个网页文件时，先会创建一个树根节点对象——document

(2). 浏览器开始扫描网页中的每一项内容(元素, 文本, 属性...)。每扫描到一项内容，就会在DOM树上相应位置创建一个节点对象，来保存扫描到的内容。



三. 查找元素: 4种:

1. 不需要查找就可直接获得的元素: 4种:

(1). 根节点对象 document

(2). <html> document.documentElement

(3). <head> document.head

(4). <body> document.body

(5). 示例:输出不需要查找就可获得的四个节点的DOM树关系和节点对象本身的属性

1\_domTree.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>    <head>      <title>DOM Tree</title>      <meta charset="utf-8" />    </head>    <body>      <span id="s1">Hello</span>      <h1>标题一</h1>      <script>        //用两种方式输出document节点对象        console.log(document);//输出DOM树        console.dir(document);//输出一个对象的存储结构        //用两种方式输出<html>元素节点对象        console.log(document.documentElement);        console.dir(document.documentElement);        //用两种方式输出<head>元素节点对象        console.log(document.head);        console.dir(document.head);        //用两种方式输出<body>元素节点对象        console.log(document.body);        console.dir(document.body);      </script>    </body>  </html>  运行结果: |

2. 按节点间关系查找:

对应小程序视频: 小程序->在线->DOM->day01 1. 按节点间关系查找

(1). 何时: 如果我们已经获得一个元素，想找它周围附近的元素时

(2). 如何: 其实在内存中有两种DOM树:

a. 节点树: 包含所有网页内容的完整树结构: 2大类关系, 6个属性

1). 父子关系: 4个属性

i. 获得一个节点对象的父节点: 节点对象.parentNode

父 节点

ii. 获得一个节点下所有直接子节点的集合: 节点对象.childNodes

子 节点们

iii. 获得一个节点下第一个直接子节点: 节点对象.firstChild

第一个孩子 iv. 获得一个节点下最后一个直接子节点: 节点对象.lastChild

最后一个孩子

2). 兄弟关系: 2个属性

i. 前一个相邻的兄弟节点: 当前节点.previousSibling

前一个兄弟

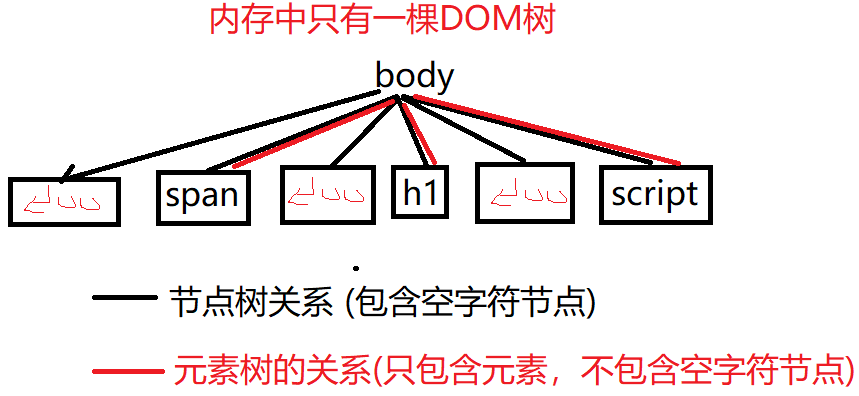
ii. 后一个相邻的兄弟节点: 当前节点.nextSibling

下一个兄弟

b. 问题: 节点树中连看不见的空字符，也会被认为是文本节点对象，严重干扰我们的查找

c. 解决: 新版DOM标准中，新增了一种新的树/关系，叫元素树。好处是，只包含元素节点，不包含文本节点，不受看不见的空字符节点的干扰！所以，今后，只要按节点间关系查找元素，都用元素树关系。不用节点树关系了。

d. 什么是元素树: 只包含元素节点，不包含文本节点的树结构



e. 元素树: 2大类关系, 6个属性

1). 父子关系:

i. 获得当前元素的父元素: 元素.parentElement

父 元素

ii. 获得当前元素的所有直接子元素: 元素.children

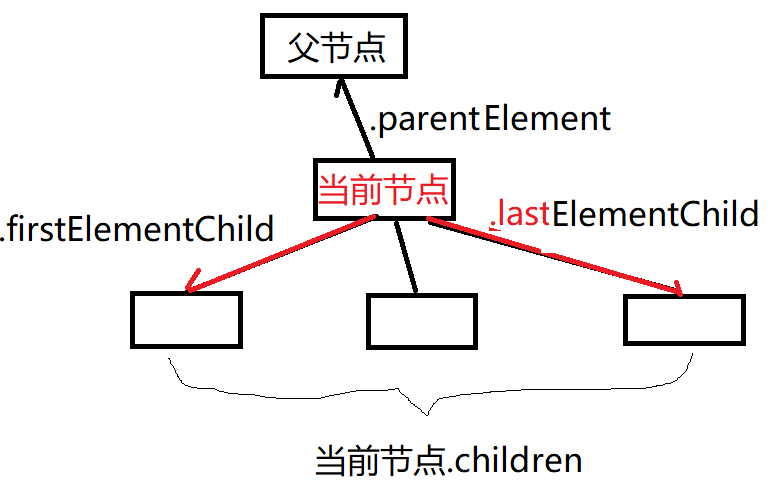
孩子们

iii. 获得当前元素的第一个直接子元素: 元素.firstElementChild

第一个 元素 孩子

iv. 获得当前元素的最后一个直接子元素: 元素.lastElementChild

最后一个 元素 孩子



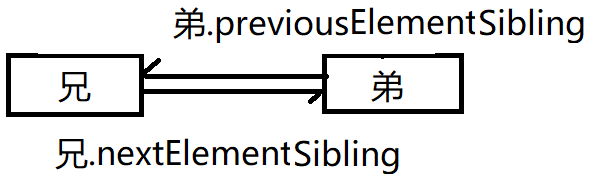
2). 兄弟关系: 2个

i. 前一个兄弟元素: 元素.previousElementSibling

前一个 元素 兄弟

ii. 后一个兄弟元素: 元素.nextElementSibling

后一个 元素 兄弟



(3). 示例: 使用元素树关系查找想要的元素

1\_domTree2.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>    <head>      <title>DOM Tree</title>      <meta charset="utf-8" />    </head>    <body>      <span id="s1">Hello</span>      <h1>标题一</h1>      <script>        //强调: 任何元素都必须先查找再使用！        //获得body元素        var body=document.body;        console.log(body);        //获得body下第一个孩子span        //错误:        // var span=body.firstChild;        //正确:        var span=body.firstElementChild;        console.log(span);        //获得span的下一个兄弟h1        //错误:        // var h1=span.nextSibling;        var h1=span.nextElementSibling;        console.log(h1);        //判断h1的父元素是不是body元素        console.log(h1.parentElement==body);//true      </script>    </body>  </html>  运行结果:  <span id=​"s1">​Hello​</span>​  <h1>​标题一​</h1>​  true |

3. 按HTML特征查找: 4个函数:

对应小程序视频: 小程序->在线->DOM->day01 2. 按HTML特征查找

(1). 按id查找一个元素对象:

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

在网页中 查找 元素 按id

b. 意为: 在整个页面中查找id为指定id名的一个元素对象

c. 返回值:

1). 如果找到，只能返回一个元素对象

2). 如果没找到，返回null

d. 强调:

1). .前的主语必须是document

2). 因为只找到一个，所以函数名中的element是单数，没有s结尾

(2). 按标签名查找多个元素对象:

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

在任意父元素内查找多个元素按标签名

b. 意为: 在一个指定的父元素范围内，查找所有指定标签名的元素，放入一个类数组对象中返回！

c. 返回值:

1). 如果找到符合要求的元素，则放在一个类数组对象中返回

2). 如果没找到符合要求的元素，则返回空类数组对象: { length:0 }

d. 强调:

1). .前的主语可以是任意父元素，也可以是document。但是，查找范围越小，查找速度越快！所以，应该尽量缩小查找的范围！

2). 因为有可能找到多个元素，所以函数名中Elements是复数，s结尾.

3). 不仅查找直接子元素，而且会在所有后代中查找符合要求的元素！

4). 即使只找到一个符合条件的元素，也会将这唯一的一个元素，放在类数组对象中返回！但是，如果我们想操作找到的这一个元素，必须用[0]从类数组对象中取出这唯一的元素才能继续对这个元素对象执行操作！

(3). 按class名查找多个元素

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

b. 意为: 在一个指定的父元素范围内，查找所有class属性包含指定class名的多个元素，放入一个类数组对象中返回！

c. 返回值:

1). 如果找到符合要求的元素，则放在一个类数组对象中返回

2). 如果没找到符合要求的元素，则返回空类数组对象: { length:0 }

d. 强调:

1). .前的主语可以是任意父元素，也可以是document。但是，查找范围越小，查找速度越快！所以，应该尽量缩小查找的范围！

2). 因为有可能找到多个元素，所以函数名中Elements是复数，s结尾.

3). 不仅查找直接子元素，而且会在所有后代中查找符合要求的元素！

4). 即使只找到一个符合条件的元素，也会将这唯一的一个元素，放在类数组对象中返回！但是，如果我们想操作找到的这一个元素，必须用[0]从类数组对象中取出这唯一的元素才能继续对这个元素对象执行操作！

5). 如果一个元素上被多个class同时修饰，那么，只需要使用其中一个class名，就可找到这个元素。

(4). 按name名查找多个表单元素:

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

b. 意为: 在整个页面范围内，查找所有name属性值为指定name名的多个表单元素，放入一个类数组对象中返回！

c. 返回值:

1). 如果找到符合要求的元素，则放在一个类数组对象中返回

2). 如果没找到符合要求的元素，则返回空类数组对象: { length:0 }

d. 强调:

1). .前的主语必须是document。

2). 因为有可能找到多个元素，所以函数名中Elements是复数，s结尾.

3). 即使只找到一个符合条件的元素，也会将这唯一的一个元素，放在类数组对象中返回！但是，如果我们想操作找到的这一个元素，必须用[0]从类数组对象中取出这唯一的元素才能继续对这个元素对象执行操作！

(5). 示例: 分别使用四个函数按不同的HTML特征查找元素

2\_iterator.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>    <head>      <title>遍历节点树</title>      <meta charset="utf-8"/>      </head>    <body>      用户名: <input name="uname"><br/>      <span>Hello World !</span>      <ul id="nav">        <li class="item parent">电影</li>        <li class="item parent">综艺          <ul>            <li class="item child">跑男</li>            <li class="active item child">爸爸</li>            <li class="item child">极限</li>          </ul>        </li>        <li class="item parent">剧集</li>      </ul>      <script>        //查找id为nav的一个ul元素        var ul=document.getElementById("nav");        console.log(ul);        //查找ul下的所有li        var lis=ul.getElementsByTagName("li");        console.log(lis);//六个        //查找ul下所有直接子元素li        var children=ul.children;        console.log(children);//3个        //想修改span的内容为❀        var span=document.getElementsByTagName("span")[0];        console.log(span);        span.innerHTML="❀";        //想找ul下所有class为item的元素        var items=ul.getElementsByClassName("item");        console.log(items);//6个        //想找到class为active的元素，将其内容改为❀        var active=ul.getElementsByClassName("active")[0];        console.log(active);        active.innerHTML="❀";        //想找到name属性为uname的文本框，并设置其内容为❀        var input=document.getElementsByName("uname")[0];        console.log(input);        input.value="❀";      </script>    </body>  </html>  运行结果: |

四. 购物车案例:

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

1. 需求:

(1). 点+/-号按钮，修改商品数量

(2). 数量修改，影响小计

(3). 小计修改，影响总计

2. 实现: 点+/-号按钮，修改商品数量:

(1). 今后只要做DOM效果，都可遵循4步完成:

a. 查找触发事件的元素

b. 绑定事件:

c. 查找要修改的元素

d. 修改元素

五. 事件概述:

1. 什么是事件: 用户手动触发的或浏览器自动触发的页面内容或状态的改变。

2. 如何在事件发生时能自动执行一项任务：

(1). 每个元素对象身上都有一批以on开头的特殊属性——事件属性，只不过所有事件属性，在开局时，都默认为null。

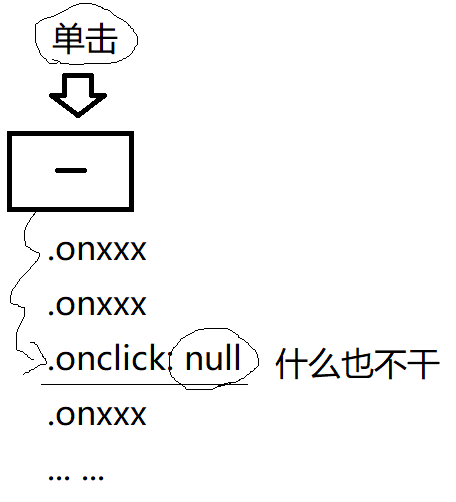
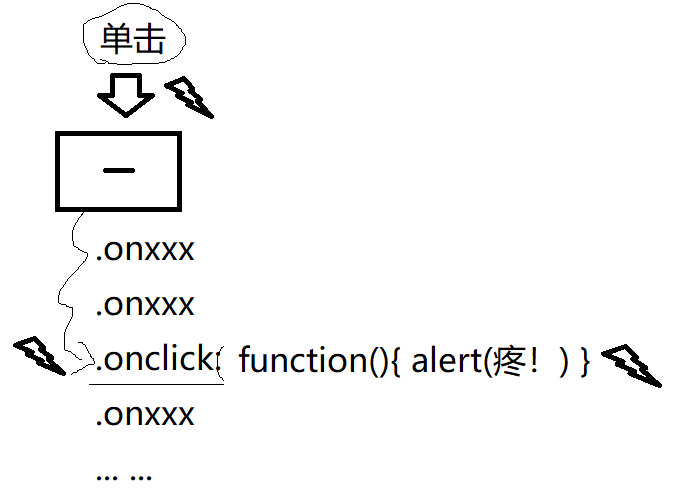
(2). 每当我们点击一个元素时，浏览器都会自动去找到这个元素的对应的事件属性。比如: 当单击一个元素时，浏览器会自动找这个元素身上的onclick属性

(3). 如果我们提前在一个元素的onxxx属性上提前保存一个函数。则，只要浏览器找到这个元素的这个事件属性时，就会自动触发我们提前保存的事件处理函数。

元素对象.on事件名=function(){ ... }

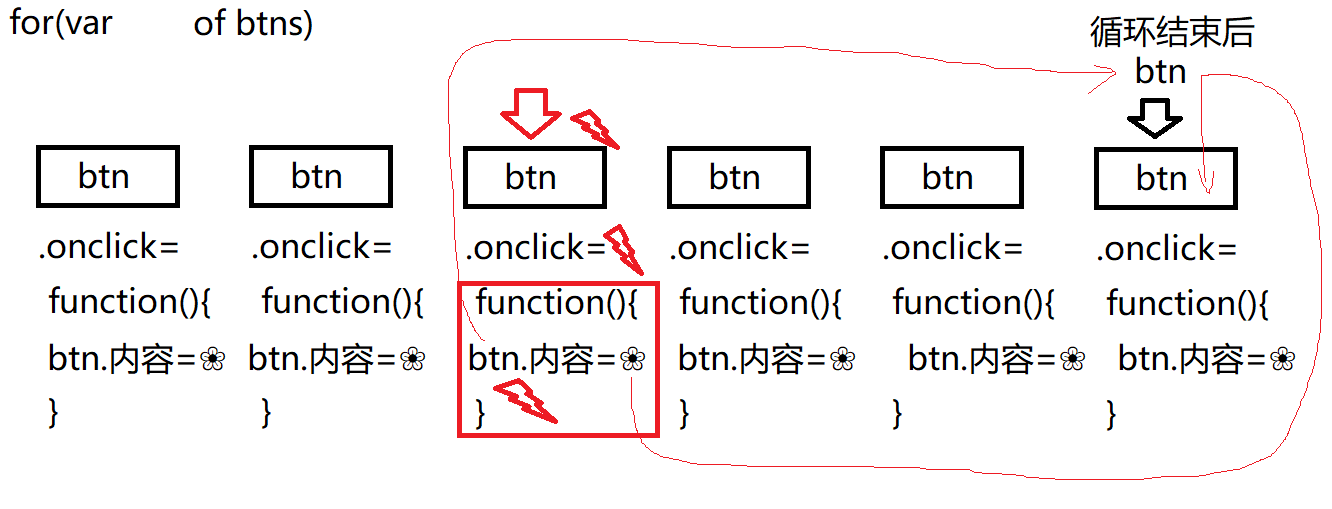
其中, 提前保存在事件属性上的这个函数，称为事件处理函数

强调: 事件处理函数，如果不触发事件，是不会执行的！只有触发了事件，才会执行处理函数的内容。如果反复触发事件，就会反复执行处理函数的内容

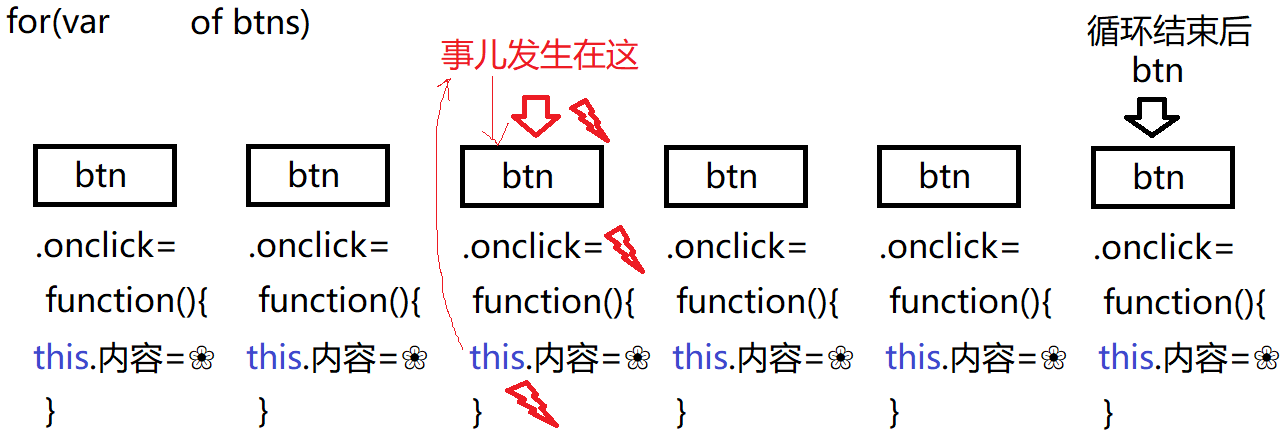
 

3. 问题: 在事件处理函数中，想获得当前触发事件的按钮，如果轻易使用事件处理函数以外的变量，极有可能出错！

4. 原因: 事件处理函数实在主程序执行后，不确定什么时候，才触发执行！而从主程序执行完，到实际触发事件处理函数这段时间间隔内，外部的变量值很有可能被篡改！



5. 解决: 今后，在事件处理函数中，想获得正在触发事件的当前元素对象本身，都要用this！



6. 示例: 实现点哪个按钮，就让哪个按钮变成小花:

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;}  </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      //想点第一个按钮，让第一个按钮喊疼      // var btn0=btns[0];      // btn0.onclick=function(){      //  alert("疼!")      // }      // console.dir(btn0);      //希望点哪个按钮，就能让哪个按钮内容变成❀      //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(){          //alert("疼!");          //错误:          //btn.innerHTML="❀"          //正确:          this.innerHTML="❀"        }        //3. 查找要修改的元素        //4. 修改元素      }    </script>  </body>  </html>  运行结果: |

|  |
| --- |
| 总结: 查找函数的返回值规律:  1. 如果原函数返回下标位置, 如果找不到，返回-1  2. 如果原函数返回一个数组或一个对象，如果找不到，返回null  3. 如果原函数返回类数组对象，如果找不到返回空类数组对象:  { length:0 } |

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:

(1). 不需要查找就可直接获得的元素对象: 4种:

document.documentElement —— <html>

document.head —— <head>

document.body —— <body>

document.forms[i] —— <form>

(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或元素.parentNode

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")

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:

元素.innerHTML

b. 获取或修改元素的纯文本内容: (待续...)

c. 获取或修改表单元素的值: (待续...)

(2). 修改属性: (待续...)

(3). 修改样式:

a. 修改元素的内联样式:

元素.style.css属性="属性值"

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

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 |

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. 当事件发生时，查找要修改的元素

4. 修改元素

今日对应小程序视频列表:

小程序->在线->DOM->day01 0 DOM Tree 树

1. 按节点间关系查找

2. 按HTML特征查找

3. 购物车 shoppingcart

作业:

1. 复习今日问题清单: 小程序->首页->DOM->day01

2. 看小程序视频完成购物车效果:

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

3. 看小程序视频，预习修改元素:

小程序->在线->DOM->day02 0.1 修改元素的内容

0.2 修改元素的属性 字符串类型

0.3 修改元素的属性 bool类型

4.1 自定义扩展属性

5. 修改样式