正课:

1. 什么是DOM

2. DOM树

3. 查找

4. 购物车

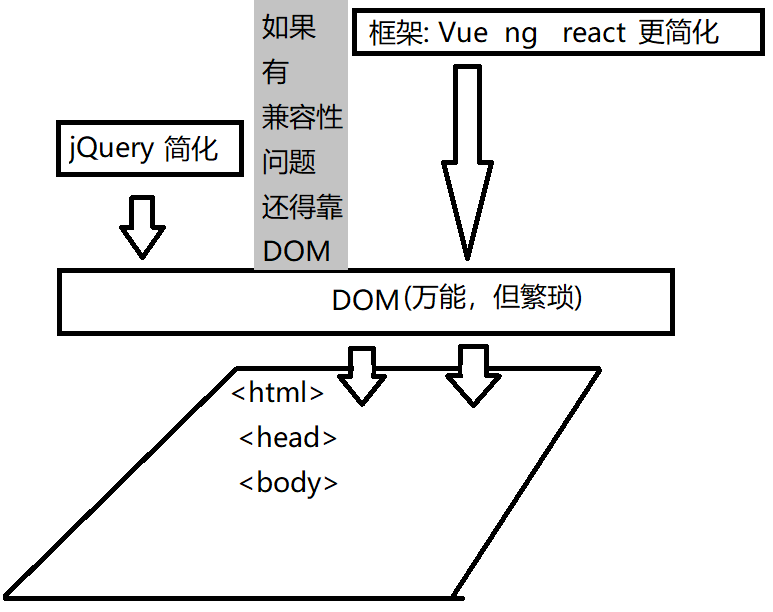
一. 什么是DOM: Document Object Model

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方法的集合.

2. 为什么DOM: ECMAScript只是规定了js语言的基本语法，并没有规定如何操作网页的内容。

3. 何时: 今后，只要操作网页中的内容，都要使用DOM



4. DOM标准: W3C负责制定和维护。

几乎所有浏览器都100%兼容DOM——万能

5. 能做5件事: 增删改查+事件绑定

二. DOM Tree

1. 什么是: 一个网页的所有内容，在浏览器读到内存中之后，都是以树形结构保存的！

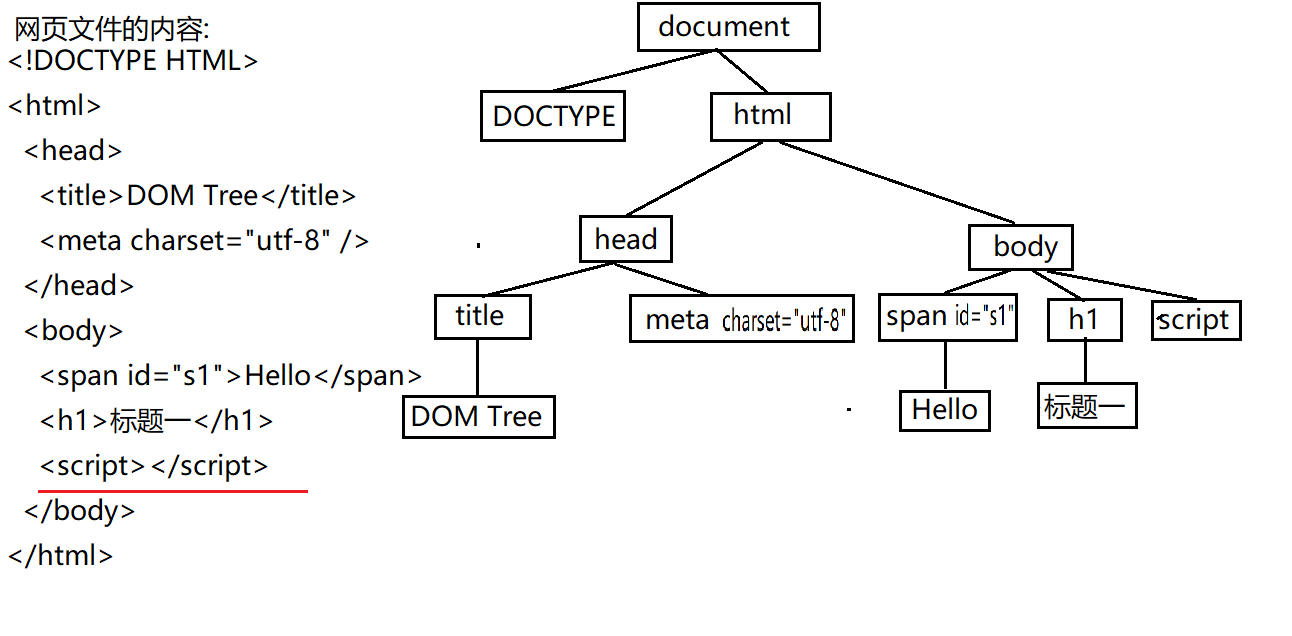
2. 为什么: 树形结构是最直观的保存上下级包含关系的结构。而网页中的标签结构，刚好就是上下级包含关系。

3. 何时: 只要浏览器请求到一个HTML网页文件，都会读取网页中的标签内容，然后在内存中创建出一棵DOM树结构来保存网页中所有内容.

4. 如何:

(1). 浏览器每读取到一个网页文件，就会先在内存中创建一个树根对象——document对象。

(2). 浏览器依次扫描网页文件中所有的HTML元素、文本等内容。每扫描到一项内容，就创建一个"节点对象"，保存元素或文本的内容和属性。然后，将节点对象挂到DOM树中指定的位置。



三. 查找: 4种:

所有元素必须经过查找，才能使用！

1. 不需要查找就可直接获得的4种节点对象:

(1). document对象 ——根节点

(2). document.documentElement对象 ——<html>元素对象

(3). document.head对象 —— <head>元素对象

(4). document.body对象 —— <body>元素对象

|  |
| --- |
| console.log()和console.dir()  console.log()输出的是DOM树结构关系  console.dir()输出的是一个节点对象在内存中的属性结果 |

2. 按节点间关系查找

(1). 什么是：按照元素在DOM树中的上下级或平级关系查找其它元素

(2). 何时: 如果已经获得一个DOM元素对象，想找这个DOM元素对象周围附近的元素时，才用节点间关系。

(3). 如何:

a. 节点树:

1). 什么是: 包含所有网页内容的完整树结构

2). 包含: 2大类关系，6个属性

i. . 父子关系: 4个属性

获得一个节点对象的父级节点对象

节点对象.parentNode —— 返回一个节点对象

父 节点

获得一个节点对象下的所有直接子节点对象的集合

节点对象.childNodes —— 返回类数组对象

子 节点们

获得一个节点对象下的第一个直接子节点对象

节点对象.firstChild —— 返回一个节点对象

第一个 孩子

获得一个节点对象下的最后一个直接子节点对象

节点对象.lastChild —— 返回一个节点对象

最后一个 孩子

ii. 兄弟关系: 2个属性

获得一个节点对象相邻的前一个兄弟节点对象

节点对象.previousSibling

前一个 兄弟

获得一个节点对象相邻的下一个兄弟节点对象

节点对象.nextSibling

下一个 兄弟

3). 问题: 节点树会将看不见的回车、空格等空字符也创建为节点对象。严重干扰我们的查找结果！所以，今后几乎不用节点树。

b. 元素树:

1). 什么是: 仅包含元素节点的树形结构

2). 优点: 不会受到看不见的空字符的干扰！所以，将来只要按节点间关系查找元素时，都用元素树

3). 包含: 2大类关系，6个属性

i. . 父子关系: 4个属性

获得一个元素对象的父级元素对象

元素对象.parentElement —— 返回一个元素对象

父 元素

获得一个元素对象下的所有直接子元素对象的集合

元素对象.children —— 返回类数组对象

孩子们

获得一个元素对象下的第一个直接子元素对象

元素对象.firstElementChild —— 返回一个元素对象

第一个 元素 孩子

获得一个元素对象下的最后一个直接子元素对象

元素对象.lastElementChild —— 返回一个元素对象

最后一个 元素 孩子

ii. 兄弟关系: 2个属性

获得一个元素对象相邻的前一个兄弟元素对象

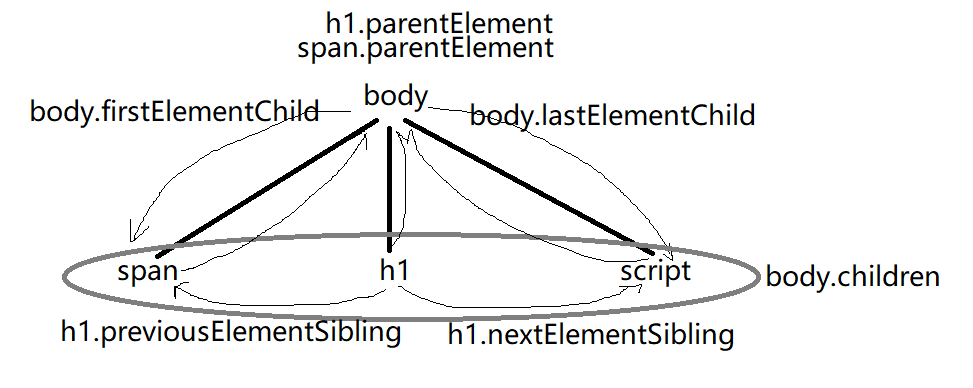
元素对象.previousElementSibling

前一个 元素 兄弟

获得一个元素对象相邻的下一个兄弟元素对象

元素对象.nextElementSibling

下一个 元素 兄弟



(4). 示例: 输出指定元素，并用节点间关系查找想要的元素

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>      //先获得四种可直接获得的节点对象      //DOM中: console.log输出的不是对象存储结构，而是DOM树结构(节点间关系)      console.log(document);      console.log(document.documentElement);      console.log(document.head);      console.log(document.body);      //console.dir(xxx)才是输出一个节点对象在内存中的属性结构      console.dir(document);      console.dir(document.documentElement);      console.dir(document.head);      console.dir(document.body);      //获得body的第一个直接子元素span      // var span=document.body.firstChild      var span=document.body.firstElementChild;      console.log(span);      //获得span的下一个兄弟h1      // var h1=span.nextSibling;      var h1=span.nextElementSibling;      console.log(h1);      //获得body下所有直接子元素的集合：3个元素      // console.log(document.body.childNodes);      console.log(document.body.children);    </script>  </body>  </html>  运行结果: |

3. 按HTML特征查找: 4个函数

(1). 按id名查找一个元素:

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

b. 意为 在整个页面中 通过id名  
 查找元素

c. 返回值:

1). 如果找到符合要求的元素，则只返回一个元素对象

2). 如果没找到，则返回null

d. 强调:

1). .前的主语必须是document

2). 因为只找一个元素，所以函数名中Element没有s结尾，是单数形式。

(2). 按标签名查找多个元素:

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

b. 意为: 在指定父元素范围内 通过标签名

获取多个元素

c. 返回值:

1). 如果找到符合要求的元素，就返回类数组对象

2). 如果没找到，返回空类数组对象: { length:0 }

d. 强调:

1). 因为找到多个元素，所以返回值为类数组对象

2). 最好指定只在一个父元素范围内查找符合要求的元素。——优化，查找范围越小，查找越快！

3). 因为找到多个元素，所以函数名中的Elements以s结尾，表示复数

4). 不仅查找直接子元素，而且会在所有后代元素中查找符合要求的元素。

5). 如果这个函数规定要返回类数组对象，那么在任何时候，都会返回类数组对象！即使只找到一个元素，也会把这唯一的一个元素放在类数组对象中第一个位置返回。所以，如果只找到一个符合要求的元素时，必须用[0]，取出类数组对象中第一个位置保存的元素对象，才能对这个元素对象继续执行后续操作！

(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: 在指定父元素范围内 通过class名

获取多个元素

c. 返回值:

1). 如果找到符合要求的元素，就返回类数组对象

2). 如果没找到，返回空类数组对象: { length:0 }

d. 强调:

1). 因为找到多个元素，所以返回值为类数组对象

2). 最好指定只在一个父元素范围内查找符合要求的元素。——优化，查找范围越小，查找越快！

3). 因为找到多个元素，所以函数名中的Elements以s结尾，表示复数

4). 不仅查找直接子元素，而且会在所有后代元素中查找符合要求的元素。

5). 如果这个函数规定要返回类数组对象，那么在任何时候，都会返回类数组对象！即使只找到一个元素，也会把这唯一的一个元素放在类数组对象中第一个位置返回。所以，如果只找到一个符合要求的元素时，必须用[0]，取出类数组对象中第一个位置保存的元素对象，才能对这个元素对象继续执行后续操作！

(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 通过name名

获取多个元素

c. 返回值:

1). 如果找到符合要求的元素，就返回类数组对象

2). 如果没找到，返回空类数组对象: { length:0 }

d. 强调:

1). .前必须是document，不能改为其它任意元素

2). 因为可能找到多个，所以也返回类数组对象

3). 因为可能找到多个，函数名中的Elements也是s结尾，表示复数

4). 如果这个函数规定要返回类数组对象，那么在任何时候，都会返回类数组对象！即使只找到一个元素，也会把这唯一的一个元素放在类数组对象中第一个位置返回。所以，如果只找到一个符合要求的元素时，必须用[0]，取出类数组对象中第一个位置保存的元素对象，才能对这个元素对象继续执行后续操作！

(5). 示例: 查找指定的元素，并修改元素的样式：

2\_iterator.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>    <head>      <title>遍历节点树</title>      <meta charset="utf-8"/>      </head>    <body>      <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 active">跑男</li>            <li class="item child">爸爸</li>            <li class="item child">极限</li>          </ul>        </li>        <li class="item parent">剧集</li>      </ul>      <form>        用户名:<input name="uname"><br/>        性别:<label><input type="radio" name="sex" value="1">男</label>        <label><input type="radio" name="sex" value="0">女</label>      </form>      <script>        //查找id为nav的一个ul元素        var ulNav=document.getElementById("nav");        console.log(ulNav);        //在id为nav的ul下查找所有li        var lis=ulNav.getElementsByTagName("li");        console.log(lis);        //在id为nav的ul下查找tr        var trs=ulNav.getElementsByTagName("tr");        console.log(trs);        //在body下查找span元素        var span=document.body.getElementsByTagName("span")[0];//从查找结果的类数组对象中取出保存在第一个位置的唯一一个DOM元素对象。        console.log(span);        span.style.color="red";        //查找id为nav的ul下的所有class为item的元素        var items=ulNav.getElementsByClassName("item");        console.log(items);        //查找id为nav的ul下的class为active的一个li，修改其字体颜色为红色        var li=ulNav.getElementsByClassName("active")[0];//取出查找结果的类数组对象中保存在第一个位置的唯一一个DOM元素对象        console.log(li);        li.style.color="red";        //查找name为sex的两个radio元素        var radios=document.getElementsByName("sex");        console.log(radios);        //查找一个name为uname的元素，为其添加阴影效果        var 姓名文本框=document.getElementsByName("uname")[0];        console.log(姓名文本框);        姓名文本框.style.boxShadow="0 0 5px red";      </script>    </body>  </html>  运行结果: |

4. 按?查找:

四. 购物车:

1. 简述: 事件绑定:

(1). 什么是事件: 浏览器自己触发的或用户手动触发的页面中元素内容或状态的改变。

(2). 什么是事件处理函数: 我们希望在事件发生时，能自动执行的一项函数。

(3). 什么是事件绑定: 提前将事件处理函数保存在元素对象的对应的事件属性中。暂存，暂不执行。

(4). 事件的原理:

a. 浏览器中某个元素身上触发事件

b. 浏览器会自动找到这个元素身上自带的对应的事件属性。

c. 浏览器会自动执行元素对象身上事件属性中提前保存的事件处理函数。

(5). 如何绑定事件处理函数: 暂时2种:

a. 在HTML中手工绑定:

1). <元素 onclick="函数名()">

<script>

function 函数名(){ ... }

2). 问题: 手工绑定，费时费力，极其不便于维护！

b. 在js中用程序自动绑定:

1). 2步:

i. 先查找要触发事件的元素对象

ii. 元素对象.onclick=function(){ ... }

2). 优点: 自动绑定，批量绑定，省时省力，极其便于维护！

(6). 问题: 在事件处理函数中，想获得当前触发事件的元素本身。如果用事件处理函数以外的其它变量，都不靠谱。

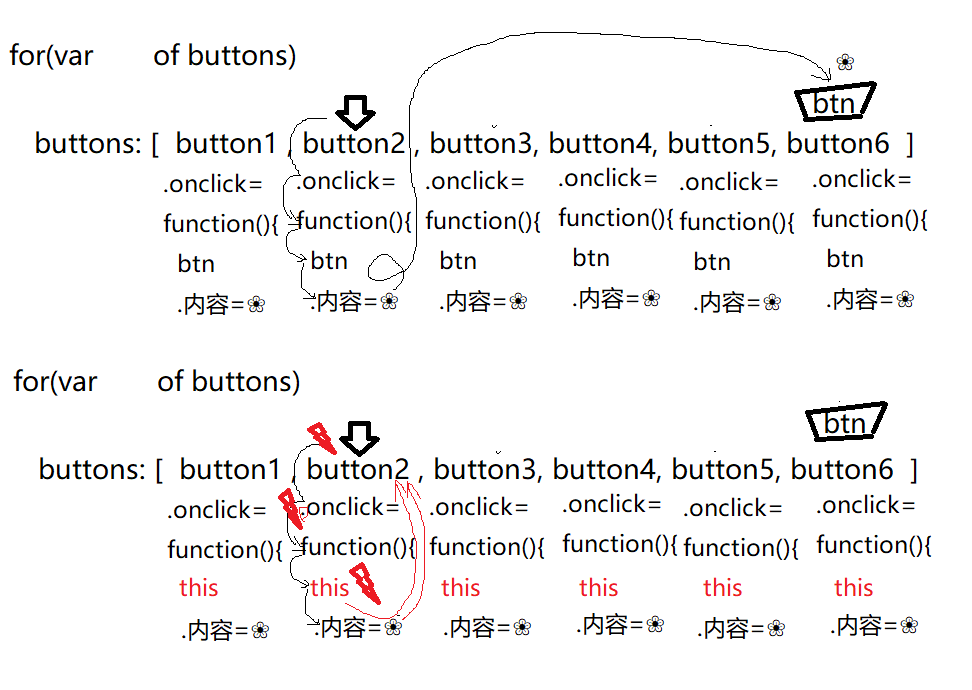
(7). 解决: 今后，只要在DOM的事件处理函数中，想获得当前触发事件的元素本身，都要用this

this->正在触发事件的当前元素本身。

(8). 示例: 为table下所有按钮提前绑定单击事件处理函数

3\_shoppingCart.html

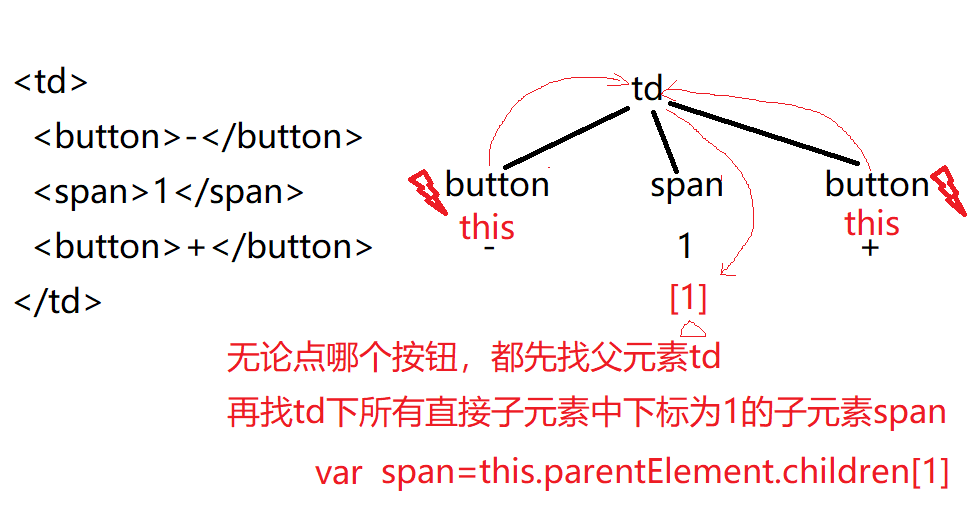
|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>    <title>使用Selector API实现购物车客户端计算</title>    <meta charset="utf-8" />    <style>      table {        width: 600px;        text-align: center;        border-collapse: collapse;      }      td,      th {        border: 1px solid black      }      td[colspan="3"] {        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>      //1. 查找触发事件的元素      //本例中: 在table下查找所有button元素      //1.1 先查找id为data的table元素      var table=document.getElementById("data");      //1.2 在table中查找所有button元素      var 按钮们=table.getElementsByTagName("button");      console.log(按钮们);      //2. 绑定事件处理函数      //本例中: 为找到的每个按钮都绑定单击事件处理函数      //所以遍历查找结果中每个DOM元素按钮      for(var 按钮 of 按钮们){        console.dir(按钮);        按钮.onclick=function(){          //点那个按钮，就让当前按钮的内容变成一朵小花          //错误: 始终改最后一个按钮          // 按钮.innerHTML="❀";          //正确: 修改当前触发事件的按钮本身          this.innerHTML="❀";        }      }    </script>  </body>  </html>  运行结果： |



2. 示例: 点按钮，修改数量:

3\_shoppingCart1.html

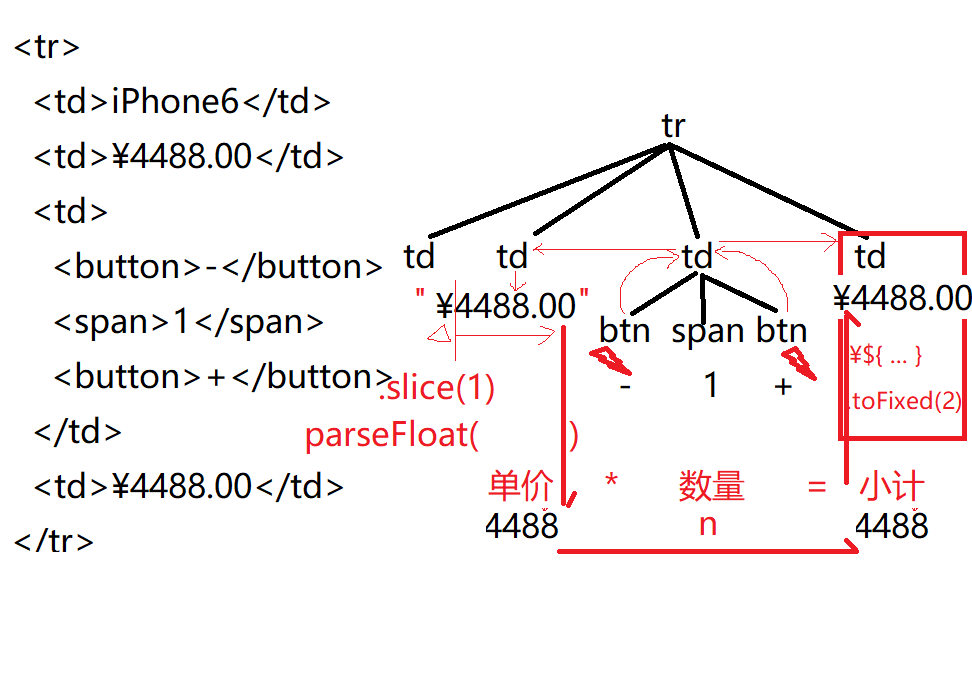
|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>    <title>使用Selector API实现购物车客户端计算</title>    <meta charset="utf-8" />    <style>      table {        width: 600px;        text-align: center;        border-collapse: collapse;      }      td,      th {        border: 1px solid black      }      td[colspan="3"] {        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>      //1. 查找触发事件的元素      //本例中: 在table下查找所有button元素      //1.1 先查找id为data的table元素      var table=document.getElementById("data");      //1.2 在table中查找所有button元素      var 按钮们=table.getElementsByTagName("button");      console.log(按钮们);      //2. 绑定事件处理函数      //本例中: 为找到的每个按钮都绑定单击事件处理函数      //所以遍历查找结果中每个DOM元素按钮      for(var 按钮 of 按钮们){        console.dir(按钮);        按钮.onclick=function(){          //3. 查找要修改的元素          //本例中: 要修改按钮旁边的span          var span=            this.parentElement.children[1];          //按钮      td      所有孩子中第2个          //4. 修改元素:3步          //4.1 取出现在的旧内容          //但是，因为从页面上获得的一切都是字符串！如果要进行算术计算，必须先转换，再计算          var n=parseInt(span.innerHTML);          //4.2 修改后          //本例中:如果当前点击的按钮的内容为+，则n++，否则如果当前点击的按钮的内容为-，且n>1，才能n--。因为购物车中商品数量不能<1          if(this.innerHTML=="+"){            n++;          }else if(n>1){            n--;          }          //4.3 再放回去          span.innerHTML=n;        }      }    </script>  </body>  </html>  运行结果: |



3. 示例: 数量修改，小计也修改

3\_shoppingCart2.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>      //1. 查找触发事件的元素      //本例中: 在table下查找所有button元素      //1.1 先查找id为data的table元素      var table=document.getElementById("data");      //1.2 在table中查找所有button元素      var 按钮们=table.getElementsByTagName("button");      console.log(按钮们);      //2. 绑定事件处理函数      //本例中: 为找到的每个按钮都绑定单击事件处理函数      //所以遍历查找结果中每个DOM元素按钮      for(var 按钮 of 按钮们){        console.dir(按钮);        按钮.onclick=function(){          //3. 查找要修改的元素          //本例中: 要修改按钮旁边的span          var span=            this.parentElement.children[1];          //按钮      td      所有孩子中第2个          //4. 修改元素:3步          //4.1 取出现在的旧内容          //但是，因为从页面上获得的一切都是字符串！如果要进行算术计算，必须先转换，再计算          var n=parseInt(span.innerHTML);          //4.2 修改后          //本例中:如果当前点击的按钮的内容为+，则n++，否则如果当前点击的按钮的内容为-，且n>1，才能n--。因为购物车中商品数量不能<1          if(this.innerHTML=="+"){            n++;          }else if(n>1){            n--;          }          //4.3 再放回去          span.innerHTML=n;          /\*修改小计\*/          //3. 查找要修改的元素          //本例中:查找当前按钮的爹td的下一个兄弟td          var subTd=this.parentElement.nextElementSibling;          //4. 修改元素          //4.1 先获得单价: 当前按钮的爹td的前一个兄弟td的内容，去掉开头¥，转为数字          var price=parseFloat(            this //当前按钮            .parentElement //当前td            .previousElementSibling //前一个td            .innerHTML//内容"¥4488.00"            .slice(1) //"4488.00"          );          //4.2计算小计:单价\*数量          var sub=price\*n;          //4.3将小计放入小计单元格:          subTd.innerHTML=`¥${sub.toFixed(2)}`        }      }    </script>  </body>  </html>  运行结果: |



4. 小计修改，总计也修改

总结: this 6种:

1. obj.fun() this->点前的obj对象

2. new Fun() this->new正在创建的新对象

3. 类型名.prototype.共有方法=function(){ ... }

this->将来谁调用这个函数，就指谁

将来调用这个函数的.前的某个子对象

4. fun() 和回调函数 和匿名函数自调 this->默认指window

5. 访问器属性中的this指访问器属性所在的当前对象

6. 在DOM的事件处理函数内的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，

/\*所有元素必须先查找才能使用，不能直接使用\*/

(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")

今日对应小程序视频列表:

小程序->在线->DOM->day01

0. DOM Tree 树  
 https://pan.baidu.com/s/1cCNhWz8tMLPOvIBGtRcNsg 提取码：xa87

1. 按节点间关系查找  
 https://pan.baidu.com/s/1T1Fzo2U1gbsf1ir269NRcg 提取码：wut1

2. 按HTML特征查找  
 https://pan.baidu.com/s/1SHfsFjxWF1C30ZeXOoWcCg 提取码：a1nf

3. 购物车 shoppingcart  
 https://pan.baidu.com/s/1FXlLwTg40DsMl48PU\_jZBg 提取码：bxb8

作业:

1. 复习今日小程序问题清单: 小程序->首页->DOM->day01

2. 复习第二阶段所有选择器！

3. 预习: 小程序->在线->DOM->day02

4. （选做）作业: 扩展鄙视题: 多个按钮，点哪个按钮，就提示，点的第几个.

链接：https://pan.baidu.com/s/1KOHhqXSbONhQzDLxQEvUHw 提取码：0xt0

5. （选做）作业: 扩展鄙视题: 递归遍历一个父元素下所有后代元素(鄙视时可能让找出符合条件的某些元素或者修改个别元素的内容).

链接：https://pan.baidu.com/s/1LCr\_b6B8u3aXgPc2F0j92A 提取码：phq5