准备: 微信小程序: WEB问题速查 小恐龙图标

1. 历届师兄师姐在今天课程中曾经问过的高频的经典的问题清单和解答: 小程序->首页->DOM->day01

自己可能没有发现问题，但是想知道别的同学都问过哪些问题

2. 今日课程对应的重点案例的视频以及鄙视面试所需的扩展知识视频：小程序->在线->DOM-> day01

如果因为某些原因错过直播讲解，不用着急！直接看小程序中视频的回看即可。

如果tmooc不能用了，立刻看小程序视频，小程序视频比tmooc要全，甚至有扩展的新知识。

如果学有余力，将来有能力拿更高薪资，小程序中所有视频必须都要看。但是不必现在都看完。只要在笔试面试前都看完即可！

问题: 个人版小程序有功能限制：禁止在线播放视频

解决: 长按视频连接地址，完整复制视频连接地址和验证码到浏览器打开。通过百度网盘下载后观看。

3. 今日课程中包含的英语单词的翻译：小程序->单词->DOM->day01

英文不太好的同学，可以提前看一下第二天需要用到哪些英文单词。

约定: 2件事:

1. 第三阶段禁止上课跟着敲笔记和敲代码！谁抄笔记抄的勤，谁抄代码抄的勤，谁学的越烂！

2. 每个完整功能的案例/视频，至少写三遍: (如果需要画图，必须亲手画图)

(1). 连代码+注释抄一遍 —— 明白这个例子是要做什么事儿

(2). 保留注释，自己试着看着注释，将代码填回来 —— 复习函数和对象

(3). 删除所有代码和注释，自己试着用自己的话，把注释写回来，再翻译为代码！—— 只有自己可以写出人话的注释，这个例子才算自己会！

如果没有问题的同学: （享受vip一对一答疑的时间还剩13天）

1. 上课时那句话没听懂，笔记中那句话看不懂，案例中那句话看不懂，立刻问！

2. 小程序问题清单中哪个问题或解答看不懂，立刻问！

3. 小程序视频中哪个视频看不懂，立刻问

4. 个人项目中哪个功能不会做，立刻问

5. 平时刷笔试题时，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. 什么是DOM

2. DOM树  
3. 查找元素

4. 事件

5. 综合示例: 纯客户端购物车

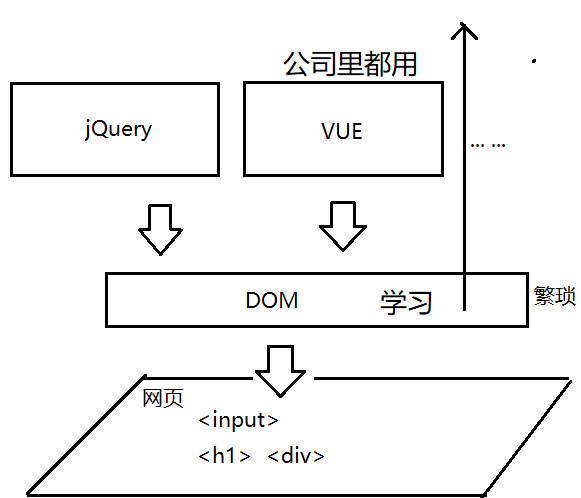
一. 什么是DOM: Document Object Model

文档 对象 模型

1. 什么是: 专门操作网页内容的一套对象和函数的整体，统称为DOM

2. 为什么: 因为ECMAScript，只规定了js语言的核心语法和内存原理，没有规定如何操作网页的内容。

3. 何时: 今后只要想操作网页的内容，实现页面交互效果，都要用DOM提供的对象和函数——唯一的办法。



4. DOM标准:

(1). 问题: 自打有浏览器那天开始，各个浏览器厂商就都有自己的一套DOM对象和函数。只不过，各个厂商之间的DOM对象和函数不兼容

(2). 解决: W3C组织出面制定了DOM对象和函数的国际标准

(3). 现在: 使用DOM标准的对象和方法来操作网页内容，几乎所有浏览器100%兼容。

5. DOM都能做哪些事儿: 5件事:

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 元素 节点 标签

1. 什么是: 浏览器在内存中使用一个树型结构来保存网页中的所有内容

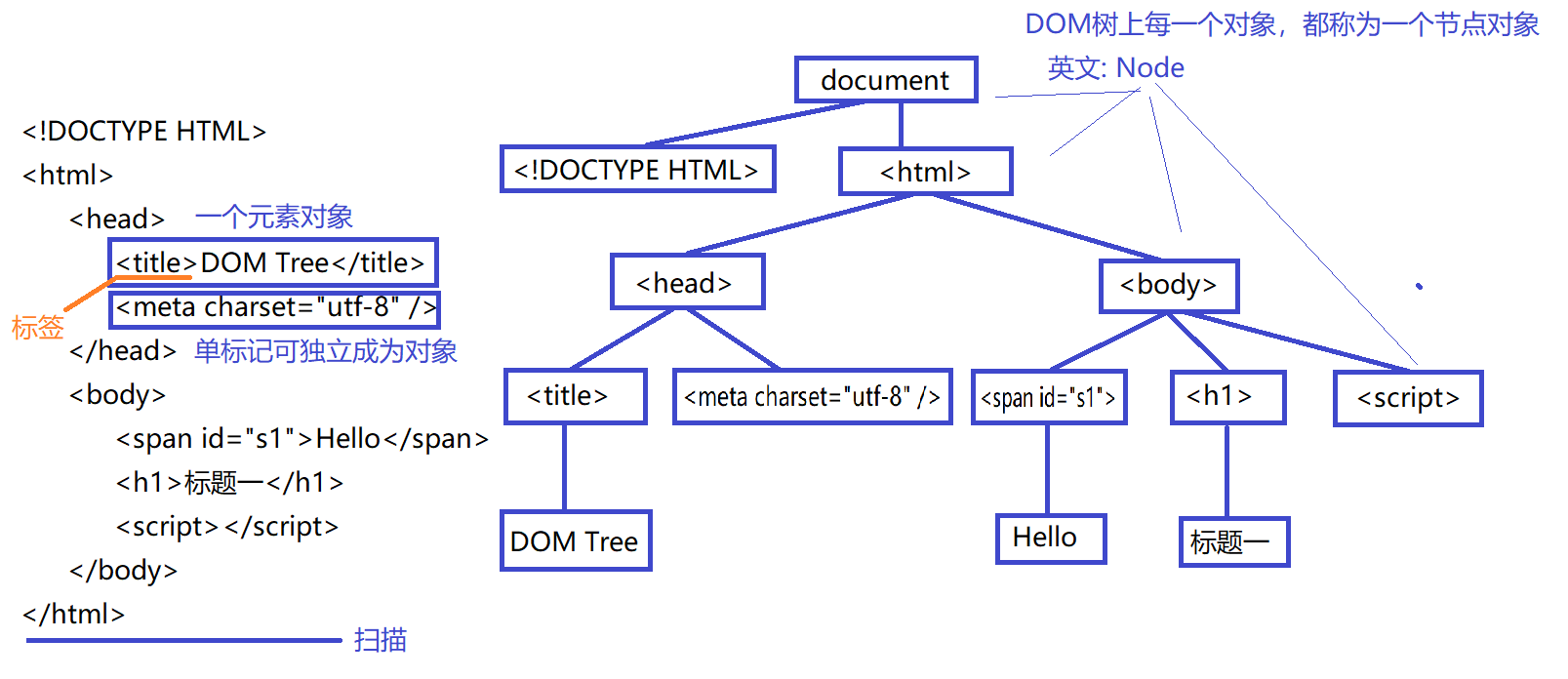
2. 为什么: 树形结构可以最直观的展现上下级包含关系。而网页中的元素及其内容，刚好也都是上下级包含关系的。

3. 何时: 只要浏览器读取到一个HTML文件，就会扫描HTML文件的内容，并将HTML文件的内容，保存到内存中一棵新创建的树结构上。

4. 如何:

(1). 只要浏览器读取到一个HTML文件，就会先在内存中创建一个唯一的树根对象document。

(2). 从上向下扫描网页的内容，每扫描到一项内容，就会在内存中创建一个对象来保存这项内容，并将对象挂在到document对象下对应位置。



三. 查找元素: 今后，只要想修改元素，都要先找到元素。4大类查找方式:

1. 不需要查找就可直接获得的元素对象: 4种:

(1). document 树根对象

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

(3). document.head <head>元素对象

(4). document.body <body>元素对象

2. 根据节点间关系查找:

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

(1). 节点树: 包含网页中一切内容的完整树结构

a. 父子关系: 4种:

1). 想获得当前元素对象的父对象: 元素对象.parentNode

父 节点

2). 想获得当前元素对象下的所有直接子节点对象们: 元素对象.childNodes

孩子 节点 们

3). 想获得当前元素对象下的第一个直接子节点对象: 元素对象.firstChild

第一个孩子

4). 想获得当前元素对象下的最后一个直接子节点对象: 元素对象.lastChild

最后一个孩子

b. 兄弟关系: 2个

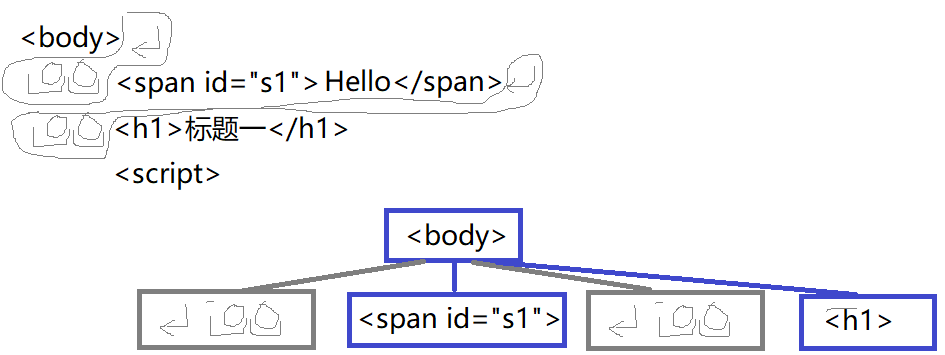
1). 获得当前元素对象的前一个兄弟节点对象: 元素对象.previousSibling

前一个 兄弟

2). 获得当前元素对象的后一个兄弟节点对象: 元素对象.nextSibling

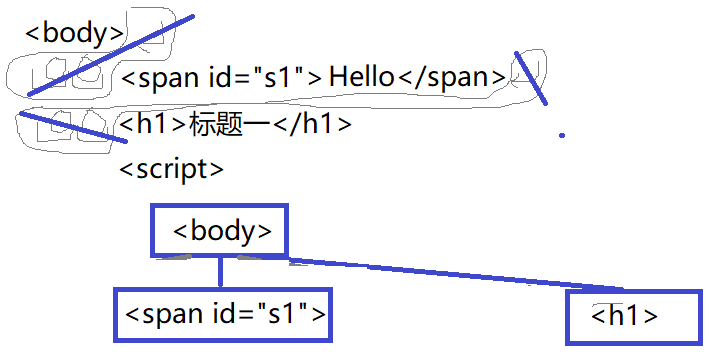
下一个兄弟

(2). 节点树的问题: 连看不见的回车和空字符都创建为DOM树上的节点，但是程序员从来不关心这些回车和空格。——严重干扰查找结果！——今后几乎不用



(3). 新树: 元素树: 仅包含元素节点对象，不包含其它文字节点 的新树结构

(4). 优点: 仅包含元素对象，所以不会受到看不见的回车或空字符干扰！



(5). 2大类关系: （根本不用背！用到时，能查笔记找到就行！）

a. 父子关系: 4个属性:

1). 获得当前元素的父元素对象: 元素对象.parentElement

父 元素

2). 获得当前元素的所有直接子元素对象: 元素对象.children

孩子们

3). 获得当前元素下第一个直接子元素对象: 元素对象.firstElementChild

第一个 元素 孩子

4). 获得当前元素下最后一个直接子元素对象: 元素对象.lastElementChild

最后一个元素 孩子

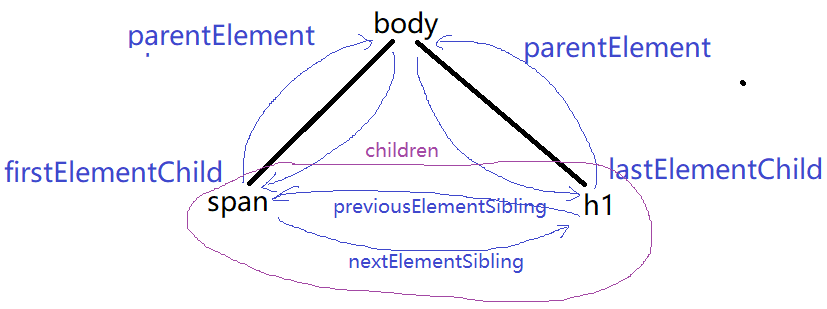
b. 兄弟关系: 2个属性

1). 获得当前元素的前一个兄弟元素: 元素对象.previousElementSibling

前一个 元素 兄弟

2). 获得当前元素的下一个兄弟元素: 元素对象.nextElementSibling

下一个元素 兄弟



(6). 何时使用: 如果已经获得一个元素，找周围附近的元素时，才用节点间关系查找

(7). 示例: 使用节点间关系查找指定元素，并验证节点间关系

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>  //前四个不需要查找，就可直接获得  console.log(document);  console.log(document.documentElement);//<html>  console.log(document.head);//<head>  console.log(document.body);//<body>    //想获得body下第一个直接子元素span  //错误: firstChild会受看不见的空字符干扰  //var span=document.body.firstChild;  //正确:  var span=document.body.firstElementChild;  console.log(span);  //想访问span的下一个兄弟元素h1  var h1=span.nextElementSibling  console.log(h1);  //h1的前一个兄弟是不是span  console.log(h1.previousElementSibling==span);//true  //访问body下所有直接子元素:  console.log(document.body.children);  //验证h1的爹是不是body  console.log(h1.parentElement==document.body);//true  </script>  </body>  </html>  运行结果: |

3. 按HTML特征查找: 4种:

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

(1). 按元素的id名查找一个元素:

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

b. 在整个网页中获得一个元素按id名

c. 强调:

1). .前的主语必须是document

2). 因为只找一个元素，所以函数名中的Element是单数，没有s结尾

3). 如果网页中碰巧两个元素id相同了！永远只返回先找到的一个元素对象。

d. 返回值:

1). 如果找到符合条件的元素对象，则只返回一个元素对象

2). 如果没找到，则返回null

(2). 按元素的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. 在指定父元素内获得多个元素按标签名

c. 强调:

1). .前的主语可以是任意父元素。查找时，只在.前的父元素下查找符合条件的子元素

2). 不仅查找直接子元素，而是在所有后代元素中查找所有符合要求的元素.

3). 因为很可能获得多个元素对象，所以函数名中Elements是复数，s结尾

4). 因为很可能获得多个元素，所以函数的返回值是一个类数组对象

5). 即使只找到一个符合条件的元素对象，也会放入一个类数组对象中返回！不会只返回一个DOM元素对象！

i. 问题: 如果想获得找到的唯一的DOM元素对象

ii. 解决: 类数组对象[0] 取出类数组对象中保存在0位置的一个元素对象

d. 返回值:

1). 如果找到符合要求的子元素，则返回一个类数组对象包含多个找到的子元素

2). 如果没找到，返回内容为空的类数组对象 { length:0 }

(3). 按元素的class名查找多个元素:

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

b. 在指定父元素下获取多个元素按class名

c. 强调:

1). .前的主语可以是任意父元素。查找时，只在.前的父元素下查找符合条件的子元素

2). 不仅查找直接子元素，而是在所有后代元素中查找所有符合要求的元素.

3). 因为很可能获得多个元素对象，所以函数名中Elements是复数，s结尾

4). 因为很可能获得多个元素，所以函数的返回值是一个类数组对象

5). 即使只找到一个符合条件的元素对象，也会放入一个类数组对象中返回！不会只返回一个DOM元素对象！如果想获得本次找到的唯一的DOM元素对象，必须在查找结果结尾加[0]

6). 如果一个元素上可能同时被多个class名修饰，那么只要使用其中任意一个class名就能找到该元素

d. 返回值:

1). 如果找到符合要求的子元素，则返回一个类数组对象包含多个找到的子元素

2). 如果没找到，返回内容为空的类数组对象 { length:0 }

(4). 按表单元素的name名查找多个表单元素:

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

b. 在整个网页内获得多个元素按name名

c. 强调:

1). .前的主语必须是document，表示在整个页面范围查找

2). 因为很可能获得多个元素对象，所以函数名中Elements是复数，s结尾

3). 因为很可能获得多个元素，所以函数的返回值是一个类数组对象

4). 即使只找到一个符合条件的元素对象，也会放入一个类数组对象中返回！不会只返回一个DOM元素对象！如果想获得本次找到的唯一的DOM元素对象，必须在查找结果结尾加[0]

d. 返回值:

1). 如果找到符合要求的子元素，则返回一个类数组对象包含多个找到的子元素

2). 如果没找到，返回内容为空的类数组对象 { length:0 }

(5). 示例: 按不同HTML特征查找指定的元素

2\_iterator.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>遍历节点树</title>  <meta charset="utf-8"/>  </head>  <body>  <span>Hello World !</span>  <form>  用户名: <input type="text" 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>  <ul id="nav">  <li class="menu-item parent">电影</li>  <li class="menu-item parent">综艺  <ul class="menu-sub">  <li class="menu-item child">跑男</li>  <li class="menu-item child">爸爸</li>  <li class="menu-item child">极限</li>  </ul>  </li>  <li class="menu-item parent">剧集</li>  </ul>  <script>  //先查找id为nav的元素对象  var ul=document.getElementById("nav");  console.log(ul);  //想在id为nav的ul下查找所有li元素  var lis=ul.getElementsByTagName("li");  console.log(lis)  //想获得id为nav的ul下第二个li下的唯一一个ul元素对象  var ul2=ul.getElementsByTagName("ul")[0];  console.log(ul2);  //只想查找ul2下的li元素  var lis=ul2.getElementsByTagName("li");  console.log(lis);  //在id为nav的ul下查找所有class为menu-item的元素  var items=ul.getElementsByClassName("menu-item");  console.log(items);  //在id为nav的ul下查找所有class为parent的元素  var parents=ul.getElementsByClassName("parent");  console.log(parents);  //在id为nav的ul下查找唯一一个class为menu-sub的ul元素对象  var ul2=ul.getElementsByClassName("menu-sub")[0];  console.log(ul2);  //想查找name名为sex的所有radio  var radios=document.getElementsByName("sex");  console.log(radios);  //只想查找唯一的name为uname的姓名文本框对象  var input=document.getElementsByName("uname")[0];  console.log(input);  </script>  </body>  </html>  运行结果: |

四. 事件:

1. 什么是事件: 浏览器自动触发的或用户手动触发的页面中内容或元素状态的改变

比如: 当单击一个按钮时会触发单击事件click

当鼠标进入一个元素范围内时会触发鼠标进入事件mouseover

当用户在文本框中输入内容时会触发内容改变事件change

... ...

2. 问题: 默认当单击一个按钮触发事件时，浏览器默认是什么也不会做的！但是，往往在事件发生时，我们都希望能自动执行一项任务。

3. 原因:

(1). 其实每个元素对象身上都有一批"on事件名"的特殊属性

(2). 当浏览器侦测到这个元素身上发生了某个事件时，就会找到元素身上对应的"on事件名"属性，尝试自动执行这个属性上提前保存的函数。

(3). 但是，所有on开头的属性的默认值都为null，所以默认情况下，就算触发了事件，浏览器也什么都不会做！

4. 解决: 今后，只要希望当一个元素上触发某个事件时，能自动执行一项任务，都要提前为这个元素身上对应的on开头的事件属性赋值一个函数——事件处理函数

5. 如何: 元素对象.on事件名=function(){ ... } ——绑定一个事件处理函数

6. 结果: 当这个元素身上发生了这个事件时，就会找到这个on开头的事件属性，自动调用其中提前保存的函数执行操作。

7. 问题: 事件处理函数中如何自动获得当前正在触发事件的这个元素对象？

(1). 错误: 使用事件处理函数外的全局变量。全局变量会被篡改或污染！当你用一个全局变量时，可能这个变量的值早就不是你想要的了！

(2). 正确: 今后只要在事件处理函数中想自动获得当前正在触发事件的元素对象，都要用this！因为浏览器自动执行事件处理函数时: 当前触发事件的元素对象.on事件名()

this->.前的当前正在触发事件的这一个元素对象

五. 综合示例: 纯客户端购物车（未完，明日待续...）

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>  //DOM 4步  //1. 查找触发事件的元素  //本例中: 用户点击table中每个button,页面会发生变化,所以，应该先查找table，然后再查找table下所有button元素  //先找id为data的table  var table=document.getElementById("data");  console.log(table);  //再在table下查找button元素  var btns=table.getElementsByTagName("button");  console.log(btns);  //2. 绑定事件处理函数  //本例中: 因为6个按钮都可单击，所以，应该给每个按钮都绑定一个单击事件处理函数  //遍历类数组对象btns中每个按钮元素对象  for(var btn of btns){  //每遍历一个按钮，就给当前按钮绑定单击事件的处理函数  btn.onclick=function(){  //alert("疼!");  //希望点哪个按钮，就让当前按钮的内容变成❀  //错误: 循环结束后，全局变量btn已经停留在最后一个按钮上了，今后执行这个函数，所有❀，都会给最后一个按钮，而不是当前按钮  //btn.innerHTML="❀";  //正确:  this.innerHTML="❀"; //this->当前正在触发事件的这个元素对象  }  }    //3. 查找要修改的元素  //4. 修改元素  </script>  </body>  </html>  运行结果: |

总结: DOM操作万能4步:

1. 查找触发事件的元素

2. 为找到的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 不用背！！！用的时候来找到即可！

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个

a. document 根节点对象

b. document.documentElement <html>

c. document.head <head>

d. document.body <body>

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

a. 父子关系: 4个属性:

1). 元素.parentElement  
 2). 元素.children

3). 元素.firstElementChild

4). 元素.lastElementChild

b. 兄弟关系: 2个属性

1). 元素.previousElementSibling

2). 元素.nextElementSibling

(3). 按HTML特征查找元素: 4个函数

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

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

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

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

|  |
| --- |
| 总结: 查找方法的返回值:  1. 如果原函数返回下标位置i，则如果找不到，返回-1  2. 如果原函数返回一个对象或数组，则如果找不到，返回null  3. 如果原函数返回一个类数组对象，则如果找不到，返回空类数组对象{ length:0 } |

2. 事件绑定:

元素对象.onclick=function(){

this -> 当前正在触发事件的元素对象

}

总结: this 5种:

1. obj.fun() this->obj

2. new Fun() this->new正在创建的子对象

3. 类型名.prototype.共有方法=function(){ ... } this->将来调用这个共有方法的.前的子对象

4. fun() 和 (function(){ })() 和回调函数中的this->window

5. 事件处理函数中的this->当前正在触发事件的这个元素对象

今日对应小程序视频列表:

小程序->在线->DOM->day01: 0 DOM Tree 树 元素...

1. 按节点间关系查找...

2. 按HTML特征查找...

3. 购物车 shoppingcart

作业:

1. 复习今日小程序问题清单(小程序->在线->DOM->day01)，预习明日小程序问题清单(小程序->在线->DOM->day02)

2. 看小程序视频复习和预习完整购物车实现方式:

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

3. 看小程序视频学习DOM预习优化查找元素

小程序->在线->DOM->day01 4. 优化 查找元素...

4. (学有余力!!!)看小程序视频学习递归遍历一个父元素下所有后代元素

小程序->在线->DOM->day01 作业: 高频笔试题：递归遍历一个父元素下所有后代元素...