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->day01

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->day01

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->day01

作业: 高频笔试题 动态集合与非动态几何

作业: 高频鄙视题 多个按钮 ... ...

作业: 高频笔试题 递归遍历一个父元素下所有后代元素

是否高频: 看我的朋友圈！历届拿到高薪的师兄师姐，反馈回来的鄙视面试重点内容！

有些有心的同学，已经把朋友圈里反复出现的高频词总结出来！

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. 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11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. 什么是DOM

2. DOM树  
3. 查找元素

一. 什么是DOM: Document Object Model

文档 对象 模型

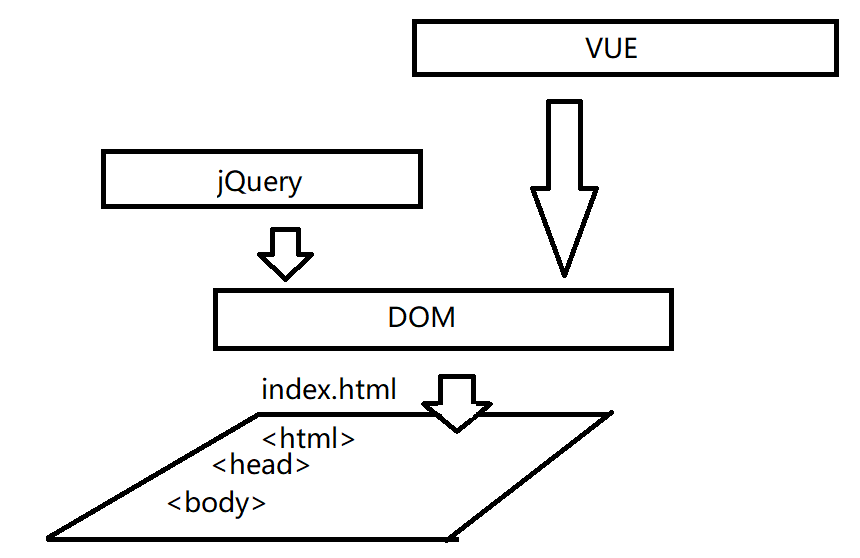
1. 什么是: 一套专门操作网页内容的对象、函数和属性的整体！

2. 为什么: ECMAScript仅规定了js语言的核心语法（怎么写对，怎么写不对）。但是，ES标准没有规定如何使用js操作网页内容！

3. 何时: 将来只要操作网页内容，都必须使用DOM提供的对象、函数和属性。

4. 问题: DOM提供的对象、函数和属性比较繁琐，不太好用！

5. 解决: 后人在DOM提供的对象、函数和属性基础上，又封装了简化的jQuery和Vue。



6. 问题: 因为ECMAScript标准不包括DOM的对象和函数。所以早期DOM没有标准，每种浏览器操作网页的方法各不相同！——严重的兼容性问题

7. 解决: W3C组织作为第三方机构又制定了一个DOM标准，规定了所有浏览器操作网页内容的标准方式——今后使用DOM标准操作网页，几乎所有浏览器100%兼容。

8. 学了DOM可以做哪些事儿？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

1. 什么是: 一个html网页中所有元素、内容和属性，在内存中都是被保存在一棵树形结构上！

2. 为什么: 因为HTML网页中的元素和内容都有明显的复杂的上下级包含关系，而树形结构是最直观的保存和反应上下级包含关系的结构！

3. 何时: 当浏览器窗口加载了一个HTML网页文件后，浏览器都会扫描HTML文件的内容，并在内存中创建DOM树以及DOM树上的节点对象来保存HTML中每项内容。

4. 节点: 树形结构或网状结构中多条线交汇的点，就称为节点！

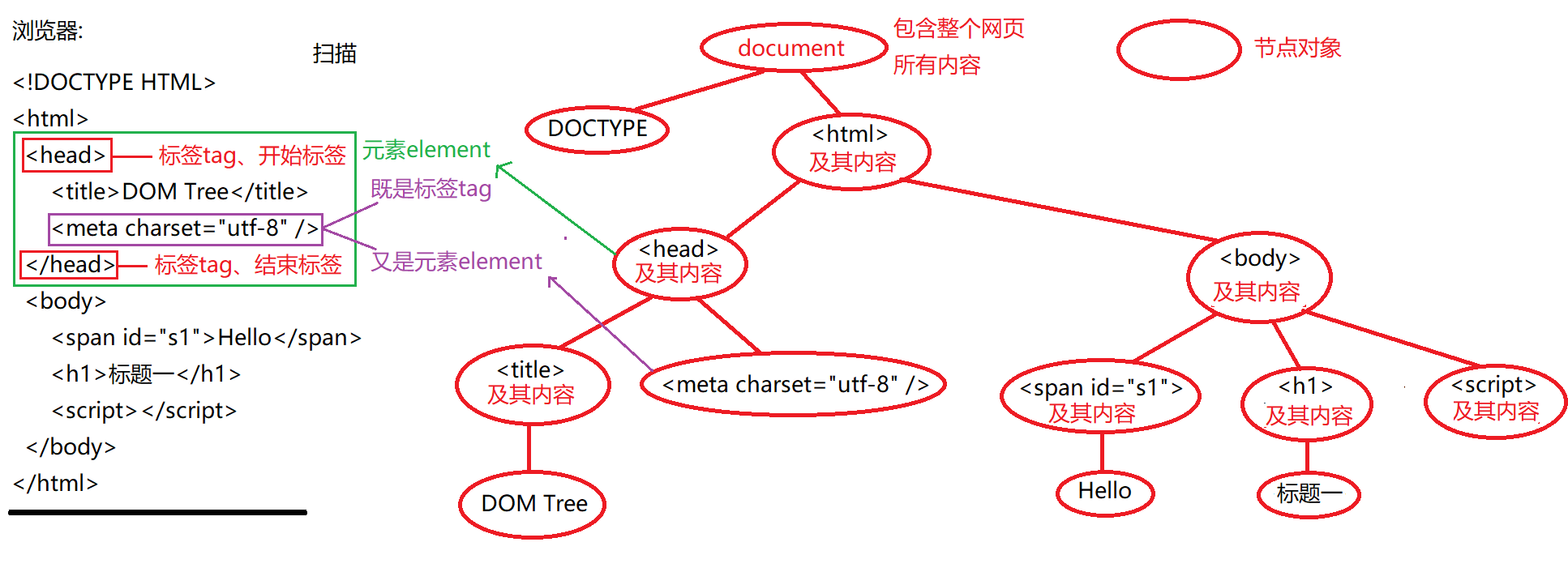
5. 如何:

(1). 先在内存中创建一个唯一的树根节点对象: document

(2). 浏览器扫描HTML文件内容

(3). 每扫描到一项内容(元素、文本、属性)，浏览器就会在树根节点document下对应位置，创建一个节点对象，保存当前扫描到的一项内容

(4). 当扫描整个HTML文件结束，内存中就形成了一棵属性结构



三. 查找元素: 4种:

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

1. 不需要查找就可直接获得的元素: 4种

(1). document对象: document —— 代表整个网页所有内容

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

(3). <head>元素对象: document.head

(4). <body>元素对象: document.body

2. 按节点间关系查找: 2大类关系，6个属性:

(1). 其实有两种DOM树: 节点树和元素树

a. 节点树: 包含所有节点对象(元素、文本等)的完整树结构

b. 元素树: 仅包含元素节点，不包含文本等其他类型节点的简化版树结构

(2). 节点树上的关系:

a. 父子关系: 4个属性:

1). 获得一个节点对象的父节点: 节点对象.parentNode

父 节点

2).获得一个节点对象下的所有直接子节点对象: 节点对象.childNodes

孩子节点们

3).获得一个节点对象下的第一个直接子节点: 节点对象.firstChild

第一个孩子

4).获得一个节点对象下的最后一个直接子节点: 节点对象.lastChild

最后一个孩子

b. 兄弟关系: 2个属性：

1). 获得当前节点对象相邻的前一个兄弟节点: 节点对象.previousSibling

前一个兄弟

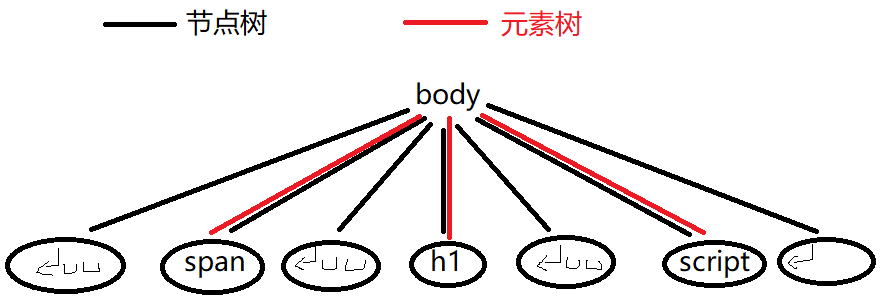
2). 获得当前节点对象相邻的下一个兄弟节点: 节点对象.nextSibling

下一个兄弟

(3). 节点树的问题: 节点树认为连看不见的换行和空格，也是文本类型的节点对象。也会成为子节点和兄弟节点。——严重干扰我们的查找结果！

(4). 解决: 新DOM标准中在原有完整DOM树基础上规定了一棵新的DOM树——元素树。元素树上的关系仅指向元素类型的节点。不再指向其他类型的节点。——好处，查找结果不会受到看不见的空字符的干扰！

(5). 强调: 元素树，不是一棵新树。仅仅是原完整节点树中的部分元素节点的一个子集。



(6). 总结: 今后按节点间关系查找时，都用元素树，而不用节点树

a. 父子关系: 4个属性:

1). 获得一个元素对象的父元素: 元素对象.parentElement

父 元素

2).获得一个元素对象下的所有直接子元素: 元素对象.children

孩子们

因为一个元素可能包含多个子元素，所以children属性返回一个类数组对象，其中包含找到的所有直接子元素对象。下标从0开始！

3).获得一个元素对象下的第一个直接子元素: 元素对象.firstElementChild

第一个 元素 孩子

4).获得一个元素对象下的最后一个直接子元素: 元素对象.lastElementChild

最后一个 元素 孩子

b. 兄弟关系: 2个属性：

1). 获得当前元素对象相邻的前一个兄弟元素:

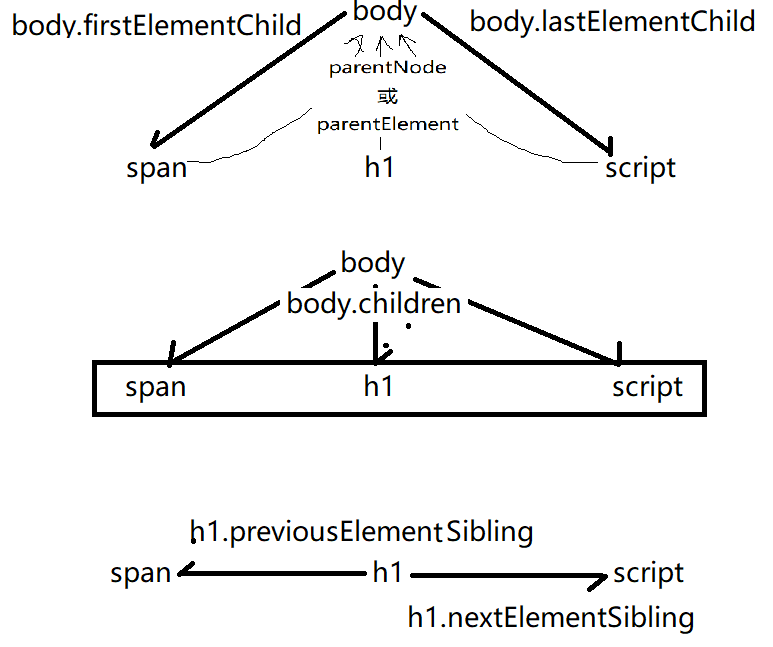
元素对象.previousElementSibling

前一个 元素 兄弟

2). 获得当前元素对象相邻的下一个兄弟元素:

元素对象.nextElementSibling

下一个 元素 兄弟



(7). 示例: 使用节点间关系，查找body下的元素

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>      //想输出document对象      console.log(document);      //想输出html元素对象      console.log(document.documentElement);      //想输出head元素对象      console.log(document.head);      //想输出body元素对象      console.log(document.body);      //本例暂时不用live server运行      //因为live server会自动插入一个多余的script，干扰我们的查找。      //应该直接在硬盘上的文件夹中找到这个.html文件，右键，打开方式，用chrome打开      //想获得body的父节点: 应该是<html>      //既可以用parentNode，又可以用parentElement      var html=document.body.parentNode;//首选      //var html=document.body.parentElement;      console.log(html);//正确      //因为在网页中能当爹，包含其它子内容的，只可能是元素。<开始标签>子内容</结束标签>      //想获得body下的所有直接子元素: 应该是3个      //错误做法:      //var childNodes=document.body.childNodes;      //console.log(childNodes);//7个      //正确做法:      var children=document.body.children;      console.log(children);//3个      //想获得body下的第一个直接子元素: 应该是span      //错误做法:      //var span=document.body.firstChild;      //正确做法:      var span=document.body.firstElementChild;      console.log(span);      //想实现body中最后一个直接子元素: 应该是script      var script=document.body.lastElementChild;      console.log(script);      //想获得body中第二个孩子：应该是h1      var h1=document.body.children[1];      console.log(h1);      //想获得h1，还可以通过span的下一个兄弟元素获得      var h1=span.nextElementSibling;      console.log(h1);      //想获得h1，还可以通过script的前一个兄弟元素获得      var h1=script.previousElementSibling;      console.log(h1);    </script>  </body>  </html>  运行结果: |

3. 按HTML特征查找: 4个方法:

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