正课:

1. 什么是DOM:

2. DOM树

3. 查找元素

4. 修改元素

1. 什么是DOM: document object model

DOM: 专门操作网页内容的一套函数和对象

DOM还是一个标准，由W3C制定

为什么:

广义JS=ECMAScript + DOM + BOM

核心语法 操作网页内容 访问浏览器软件

要想操作网页内容，为页面添加交互效果，其实只能用DOM函数和对象。

问题: 早起的DOM没有标准

解决: W3C制定了统一的DOM函数和对象的标准。

几乎所有浏览器100%兼容。

特例: IE8

何时: 只要操作网页内容，为网页添加交互行为，只能用DOM

包括: 5件事: 增删改查 事件绑定

2. DOM树:

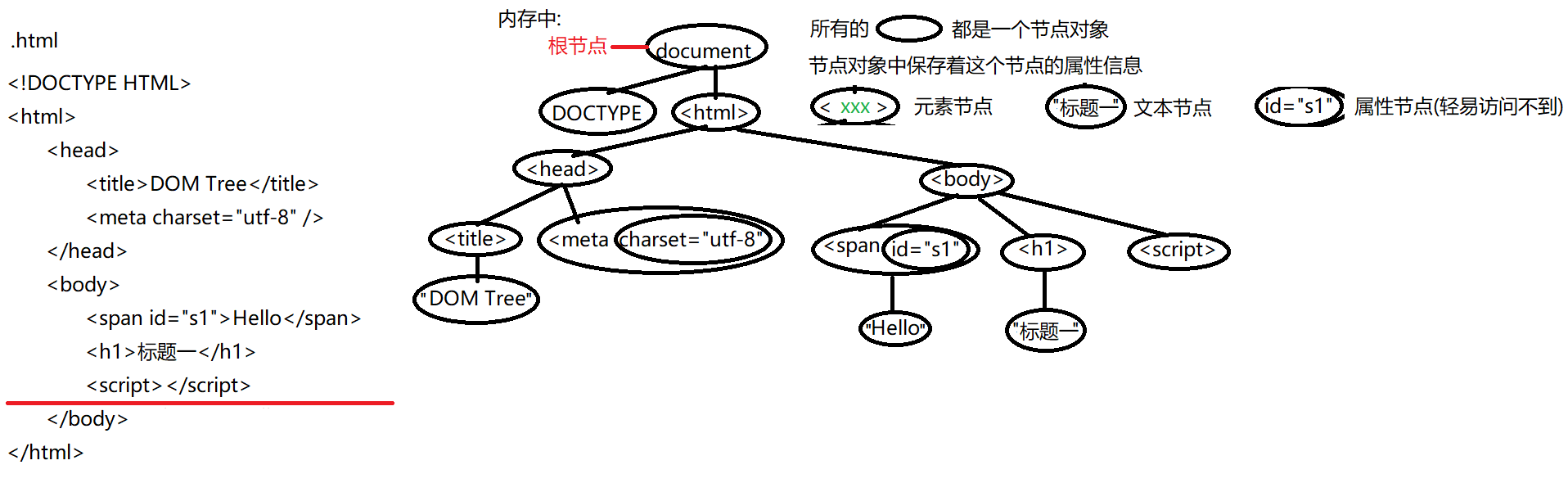
什么是: 在内存中，集中保存一个网页中所有内容的树形结构

为什么: 树形结构是最直观的保存上下级包含关系的数据结构。而网页中的HTML标签，刚好也是父子嵌套的上下级包含关系。所以，网页中每一项内容，在内存中，都是存在一棵树形结构上的。

如何:

1. 当浏览器读取到一个.html文件时，会先在内存中创建一个document对象，作为整棵树的树根对象

2. 开始扫描.html中每个元素,文本等内容。每扫描到一项内容，就在document下对应位置创建一个节点(node)对象。



3. 查找元素:

1. 不需要查找，就可直接获得:

document.documentElement <html>

document.head <head>

document.body <body>

2. 按节点间关系查找:

树上的每个节点都不是孤立存在的。都和上下左右的节点之间有各种各样的关系，可以互相访问到。

包括:

节点树: 包含所有节点内容的完整树结构

2大类关系：

1. 父子关系:

节点.parentNode 获得当前节点的父节点

父节点.childNodes 获得当前父节点下的所有直接子节点的集合。

强调: childNodes返回的是一个类数组对象，今后我都简称集合。

父节点.firstChild 获得当前父节点下的第一个直接子节点。

父节点.lastChild 获得当前父节点下的最后一个直接子节点。

2. 兄弟关系:

节点.previousSibling 获得当前节点平级的前一个相邻的兄弟节点

节点.nextSibling 获得当前节点平级的下一个相邻的兄弟节点

问题: 同时包含看不见的换行和空字符，也是节点对象，严重干扰查找。

解决: 元素树

元素树: 仅包含元素节点的树结构。

说明: 元素树不是一棵新树，而是原来节点树中添加了仅指向元素节点的新属性而形成一棵子树结构。

2大类关系：

1. 父子关系:

元素.parentElement 获得当前元素的父元素。

说明: 其实也可以用parentNode。

父元素.children 获得当前父元素下的所有直接子元素的集合。

强调: children返回的是一个类数组对象，今后我都简称集合。

父元素.firstElementChild 获得当前父素下的第一个直接子元素。

父元素.lastElementChild 获得当前父元素下的最后一个直接子元素。

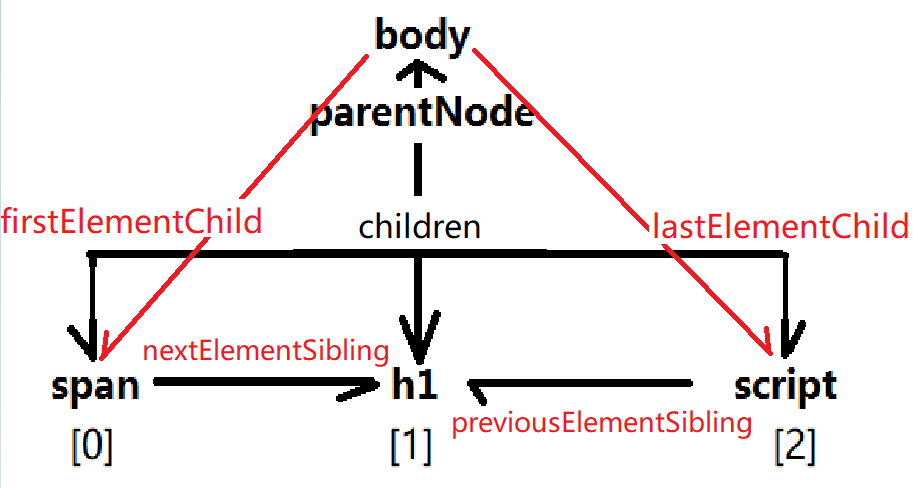
2. 兄弟关系:

元素.previousElementSibling 获得当前元素平级的前一个相邻的兄弟元素

元素.nextElementSibling 获得当前元素平级的下一个相邻的兄弟元素

总结: 今后只要按节点间关系查找，都用元素树的属性代替节点树的旧属性。

何时: 今后只要已经获得一个节点对象，找周围附近的节点时。就用节点间关系查找。



鄙视: 定义一个函数，遍历一个指定的父元素下的所有后代元素:

1. 定义一个函数，仅遍历直接子元素

2. 如果当前子元素有更下级直接子元素，则对当前子元素继续调用当前函数，查找子元素的直接子元素

function getChildren(parent){

//arguments.callee->getChildren

//1. 先获得当前父元素下的所有直接子元素

var children=parent.children;

//遍历所有直接子元素

for(var child of children){

//输出当前子元素的标签名

console.log( child.nodeName )

//元素的标签名

//2. 如果child有下级直接子元素

if(child.children.length>0){

//就对child继续调用getChildren

arguments.callee(child)

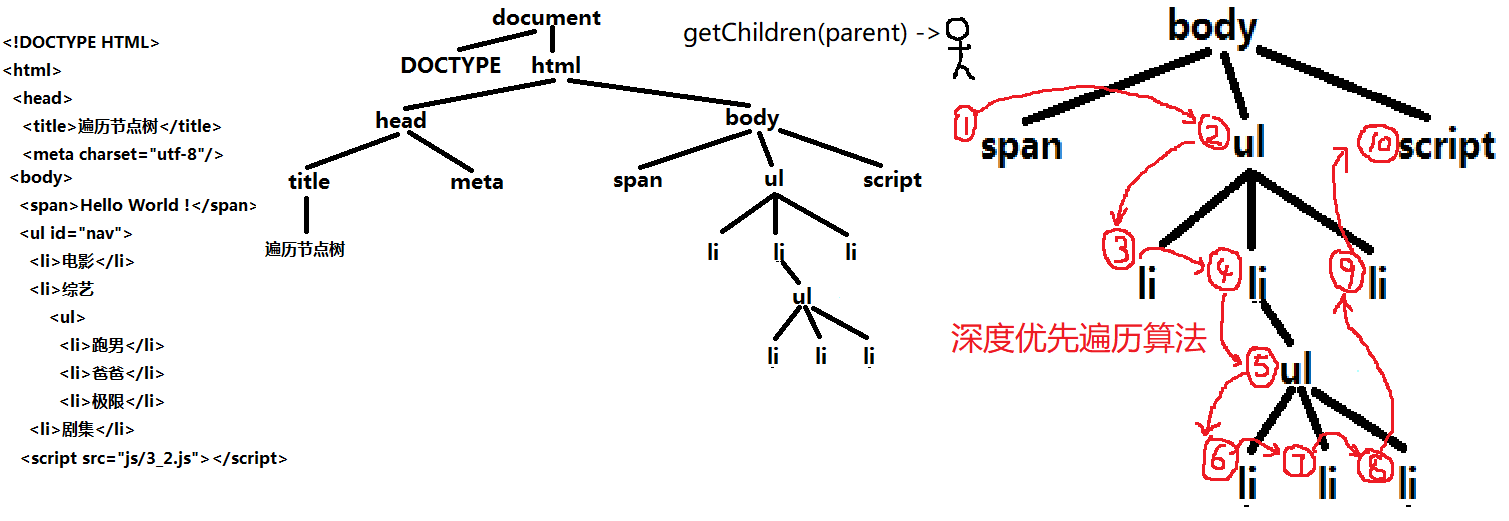
}

}

}

//查找body下所有后代元素

getChildren(document.body);



3. 按HTML特征查找:

4种:

1. 按id属性查找

var 元素=document.getElementById("id")

在整个页面中查找id为指定名称的一个元素

返回值: 如果找到，返回一个元素对象

如果没找到，返回null

强调: 只能用document调用。不能再随意父元素上调用。

2. 按标签名查找

var 集合=父元素.getElementsByTagName("标签名")

在指定父元素下查找所有标签名为指定标签名的后代元素。

返回值: 如果找到，返回多个元素组成的集合

如果没找到，返回空集合: [].length=0

强调:

1. 可在任意父元素下查找。通常指定在某个父元素下查找后代，是为了减少查找范围，提高查找效率。

2. 不止查找直接子元素，而是在所有后代中查找。

比如: nav.getElementsByTagName("li")

<ul id="nav">  
 <li>

<ul>

<li></li>

<li></li>

<li></li>

</ul>

</li>

<li>

<ul>

<li></li>

<li></li>

<li></li>

</ul>

</li>

</ul>

3. 按class属性查找

var 集合=父元素.getElementsByClassName("类名")

在指定父元素内，查找class属性中包含指定类名的所有元素

返回值: 如果找到，返回多个元素组成的集合

如果没找到，返回空集合: [].length=0

强调:

1. 可限制在任意父元素内查找，减少查找范围。

2. 不仅查找直接子元素，且在所有后代中查找。

3. 如果一个元素同时被多个class修饰，则使用其中一个class，就可找到该元素。无需所有class都满足

比如: nav.getElementsByClassName("active")

nav.getElementsByClassName("child")

<ul id="nav">  
 <li class="item parent">

<ul>

<li class="item child active"></li>

<li class="item child"></li>

<li class="item child"></li>

</ul>

</li>

<li>

<ul>

<li class="item child active"></li>

<li class="item child"></li>

<li class="item child"></li>

</ul>

</li>

</ul>

4. 按name属性查找

var 集合=document.getElementsByName("name名")

在整个网页中查找name属性值为指定name名的元素。

返回值: 如果找到，返回多个元素组成的集合

如果没找到，返回空集合: [].length=0

强调: 只能用document调用，不能用随意一个父元素调用。

何时: 在表单中查找表单元素时

问题: 所有返回集合的查找函数，在任何情况下都会返回一个集合。即使只找到一个元素，也放在一个集合中返回。

比如: <form>

<input name="uname">

var 集合=document.getElementsByName("uname")

集合: [ <input> ]

如果想使用集合中返回的这唯一的一个对象，应该?

解决: var 文本框对象=

document.getElementsByName("uname")[0]

其中: [0]表示取出集合中仅有的一个元素对象，单独使用。

以上四种查找方式，最大的问题是:

一次只能用一个特征作为查找条件。

如果查找条件非常复杂时，代码会非常繁琐

解决:

4. 用选择器当条件查找元素: 2个函数:

1. 只查找一个符合条件的元素:

var 元素=父元素.querySelector("任意复杂的选择器")

在指定父元素下查找符合选择器要求的一个元素

返回值: 如果找到，返回一个元素对象

如果没找到，返回-1

2. 查找多个符合条件的元素:

var 集合=父元素.querySelectorAll("任意复杂的选择器")

在指定父元素下查找符合选择器要求的多个元素

返回值: 如果找到，返回多个元素的集合

如果没找到，返回空集合: [].length=0

强调: 1. 可以以任意父元素调用来限制查找的范围

2. ()中的选择器参数，不用每次都写完整。只要以.前的父元素为起点开始写就行！

<ul> ul.querySelectorAll("li li")

<li>

<ul>

<li>

<li>

<li>

3. 不仅查找直接子元素，且在所有后代中查找符合条件的。

总结:

1. 不需要查找就可直接获得:

document.documentElement <html>

document.head <head>

document.body <body>

2. 按节点间关系查找:

1. 父子关系;

元素.parentNode

父元素.children

元素.firstElementChild

元素.lastElementChild

2. 兄弟关系:

元素.previousElementSibling

元素.nextElementSibling

3. 按HTML特征查找：

var 元素=document.getElementById("id")

var 集合=父元素.getElementsByTagName("标签名")

var 集合=父元素.getElementsByClassName("class名")

var 集合=document.getElementsByName("name名")

4. 按选择器查找:

var 元素=父元素.querySelector("任意选择器")

var 集合=父元素.querySelectorAll("任意选择器")

DOM优化之一:

如果只用一个条件就可以找到想要的元素时

首选使用getElementsByXXX()函数查找——效率高

只有查找条件复杂时，才选择按选择器查找——效率低

作业:

1. 页面上多个完全相同的按钮，不确定数量：

点那个按钮，报出自己是第几个！

单词列表:

1. sibling 兄弟

2. previous 前一个

3. relation 关系

4. children child的复数形式，表示所有孩子们