DOM

一、DOM概述

1、JavaScript= ECMAScript(核心语法)

+DOM(专门操作网页内容的API)

+BOM(专门操作浏览器窗口的API)

2、DHTML：所有实现网页动态效果的技术的统称

包含：HTML+CSS+JavaScript

3、HTML、XHTML、DHTML、XML作比较

HTML：超文本标记语言，专门定义网页内容的语言

XHTML：严格的HTML标准(严格的超文本标记语言)

DHTML：所有实现网页动态效果的技术的统称

XML：可扩展的标记语言，标签都是自定义的

XML和HTML的【语法】完全相同

例如 在XML中这样实现：

<Student id="1001">

<sname>...</sname>

<age>...</age>

</Student>

→这样的写法也叫结构化数据

在JS里面这样实现：

{

"id":"1001",

"sname":"...",

"age":...

}

→这样的写法也叫JSON

两者比较下来，还是js实现比较简单

4、什么是DOM：操作一切结构化文档的通用API

用DOM既可操作HTML，又可操作XML

(1)核心DOM：可以操作所有文档的通用API

特点：API繁琐，但是可以实现所有功能，比如增加(创建)、删除、修改、查找

(2)HTML DOM：基于核心DOM的API，扩展的专门操作HTML内容的API

特点：简单易用，对个别复杂的HTML元素扩展了简化方法，比如重点简化了【修改】操作

二、DOM Tree：DOM树

1、 网页加载进内存时，会先创建一个document对象，指代当前网页；

网页中的所有内容(包括元素、文本、属性、注释等)，都叫节点(Node)对象；

document作为DOM树的根节点，所有节点对象都是document对象的子节点

2、节点对象的类型

(1)document对象：文档类型(Document)

(2)元素节点对象：元素类型(Element)

(3)文本节点对象：文本类型(Text)

(4)属性节点对象：属性类型(Attr)

3、Node类型是所有节点的父类型，它的三个通用属性：

(1)nodeType：获得当前节点的类型，返回的是一个数字

何时使用：专门判断不同节点类型

比如 document.nodeType 返回9

document.body.nodeType 返回1

document.body.firstChild.nodeType 返回3

查看/验证方法：任意浏览器控制台打Node.prototype

ELEMENT\_NODE：1

ATTRIBUTE\_NODE：2

TEXT\_NODE：3

DOCUMENT\_NODE：9

(2)nodeName：获得当前节点的名称

何时使用：专门用于区分不同的元素节点

如果是元素节点，返回标签名(全大写字母)

例如document.body.nodeName，返回BODY

如果是文本节点，返回#text

例如document.body.firstChild.nodeName，返回#text

如果是文档节点，返回#document

例如document.nodeName，返回#document

例如：判断一个元素是不是按钮

if(elem.nodeName=="BUTTON")

(3)nodeValue：获得当前节点的值

\*如果是元素、文档节点，返回null，也就是说对元素、文档节点无效

例如document.nodeValue，返回null

例如document.body.nodeValue，返回null

如果是文本节点，返回文本内容

例如document.body.firstChild.nodeValue，返回"... ...(body里面的内容)"

三、节点树的六种关系

1、父子关系

parentNode childNodes firstChild lastChild

childNodes：NodeList集合，是一个类数组对象

它是一个【动态集合】，即不包含结果对象的完整属性，仅保存对象的引用

每次使用都重复查询

\*遍历动态集合：

for(var i=0,len=elem.childNodes.length;i<len;i++){ }

这样写之后，就避免了遍历动态集合时每次使用都要重复查询的弊端，提高执行效率

2、兄弟关系

previousSibling nextSibling

\*这六种关系中，除parentNode外，都会受到看不见的空字符(比如空格、换行等)的干扰

四、\*\*递归遍历节点树(见实例)

节点树：包含所有子节点对象的树结构

算法：深度优先遍历

→优先遍历下级节点，直到碰到叶子节点(即最末尾的子节点)，才返回并更换另一个分支继续遍历下级节点

递归：函数自己内部又调用了自己

arguments.callee()：指代当前函数对象

何时使用：多用于在递归调用内部取代当前函数名

今后只要递归调用，内部调用当前函数都要用callee

五、遍历元素树

元素树：仅包含元素节点的树结构

它仅是节点树的子集，并不是一个新的树

元素树的六种关系：

1、父子关系

parentElementNode children firstElementChild lastElementChild

2、兄弟关系

previousElementSibling nextElementSibling

有兼容性问题

六、遍历API： NodeIterator 和 TreeWalker

内部同样使用深度优先算法

2步：

(1)创建迭代器对象

var iterator=document.createNodeIterator(开始节点对象,显示何种节点,null,false);

显示何种节点分为： NodeFilter.SHOW\_ALL

NodeFilter.SHOW\_ELEMENT

迭代器开始时，站在第一个节点的前一个位置！

创建的两个方法：

(1)var nextNode=iterator.nextNode();

让迭代器向下一个对象跳一步，同时返回跳到的对象

如果跳到头(没有下一个了)，则返回null

(2)var previousNode=iterator.previousNode();

让迭代器向上一个对象跳一步

(2)利用循环推动迭代器反复向下一个节点移动

NodeIterator vs TreeWalker

创建和深度遍历的方法，两者完全一样

\*NodeIterator一开始站在开始节点之前的空位置；TreeWalker一开始就站在开始节点上

TreeWalker扩展了更灵活的跳转方法

实例见js/6.js

七、查找元素节点：2类

1、按HTML属性查找

(1)按id查找

var elem=document.getElementById("id值");

(2)按标签名查找

var elems=elem.getElementsByTagName("标签名");

它不仅查找直接子元素，且同时查找所有子代元素

(3)var elems=elem.getElementsByName("name名");

\*以上三种没有浏览器兼容性问题

(4)var elems=elem.getElementsByClassName("class名");

此种IE8不兼容

\*getElements... ，带s的，返回的是HTMLCollection动态集合

2、Selector API

专门【利用css选择器查找】任意元素的方法，它是Jquery的核心

(1)var elem=elem.querySelector("选择器");

只返回第一个匹配的元素对象

何时使用：只找一个元素时使用

(2)var elems=elem.querySelectorAll("选择器");

返回所有符合选择器的元素对象

\*是包含完整对象属性的集合→非动态集合

正常遍历for(var i=0;i<elems.length;i++){}

3、Selector API vs getElement...

效率：getElement... 最高

易用性：Selector API 更容易使用，便于反复查找

→ 如果只找一次就可获得元素时，用getElement...

如果需要反复查找才能获得元素时，用Selector API

\*4、在html中绑定的事件处理函数，如果要获得当前触发事件的元素对象，都要在html中显式用this关键字传入

\*不能在函数中直接使用this！因为此时函数内的this相当于window

何时需要使用this：事件处理函数中用到了当前触发事件的元素

例如<span class="open" onclick="toggle(this)">...</span>

八、获取/修改元素的内容：3种

1、innerHTML：获得/设置元素开始标签和结束标签之间的html原文

何时使用：只要获取或修改元素的HTML原文时

固定套路：

(1)批量删除父元素下【所有】子节点

elem.innerHTML="";

(2)批量替换父元素下【所有】子节点

elem.innerHTML="所有子元素标签组成的html字符串";

2、textContent：获得/设置元素开始标签和结束标签之间的转化后的正文内容

IE8不兼容

3、innerText

仅IE8支持

第2/3种方法不建议使用，innerHTML比较靠谱

\*\*4、如果遍历中的操作会减少数组的元素个数，就会影响后续下标，但不影响前边的元素的下标

解决方法：从后往前遍历

例如for(var i=arr.length-1;i>=0;i--){}

5、option对象：selected，可当做boolean类型使用

九、修改元素的属性/特性

1、 核心DOM：不带HTML开头的类型中定义的属性和方法

HTML DOM：在带HTML开头的类型中定义的属性和方法

比如 body元素 继承自(1级) HTMLBodyElement的原型

(2级) HTMLElement的原型(所有HTML元素的类型)

(3级) Element的原型(所有元素的类型)

(4级) Node的原型(所有节点的父类型)

继承：document.body.\_\_proto\_\_.\_\_proto\_\_. ...

→1、2级是HTML DOM，3级是核心DOM

2、获取元素的属性值：3种

一切从属性获取的值和设置到属性上的值必须都是字符串

所以都要先类型转换，再计算

(1)var strValue=elem.getAttribute("html属性名");

何时使用：只找一个属性值时

(2) elem.attributes[i/"html属性名"] 返回一个属性节点对象

elem.attributes[i/"html属性名"].value 返回属性值

elem.attributes集合：包含了当前元素所有属性的节点对象

何时使用：遍历一个元素所有属性时

(3) elem.getAttributeNode("html属性名") 返回一个属性节点

elem.getAttributeNode("html属性名").value 返回属性值

3、修改元素的属性值：2种

(1)elem.setAttribute("html属性名","新属性值");

何时使用：只修改一个属性值时

(2)elem.setAttributeNode(属性节点对象)

4、移除元素的属性：2种

(1)elem.removeAttribute("属性名")

(2)elem.removeAttributeNode(属性节点)

5、移除css属性：

elem.style.removeProperty("属性名");

6、判断是否包含指定属性

elem.hasAttribute("属性名")

返回值为bool类型

7、Property(属性) vs Attribute(html特性)

Property：内存中对象的属性，通常打.调用

Attribute：指出现在页面html代码中的属性

(1)对于标准属性：Property等效于Attribute

例如 a.setAttribute("href","url"); →核心DOM

a.href="url"; →HTML DOM

所以，今后操作标准属性，都用HTML DOM，因为所有标准属性都被HTMLxxxElement及其父类型预定义好了

\*特例(class)： txt.setAttribute("class","类名");

txt.className="类名";

因为class与js冲突，必须改为className

(2)HTML5中，自定义属性：

只能用setAttribute，不能直接用 . 调用

例如 a.age=18; →只定义在内存中，无法出现在html中

a.setAttribute("age","18"); →age会出现在html中

所以用 . 设置的，不能出现在页面上，就不能被getAttribute；setAttribute设置的，会出现在页面上，但是不能用 . 调用

→\*h5规定：凡是自定义属性都要加 data- 前缀，那么就可以用 xxx.dataset.属性名 来获取自定义属性值

设置自定义属性值：xxx.dataset.属性名=值;

8、\*\*将类数组对象变为原始的数组类型对象(固定套路)

数组=Array.prototype.slice.call(类数组对象);

9、\*\*能否对类数组对象排序：2步

(1)将类数组对象变为数组对象

(2)数组.sort();

十、修改元素的样式

1、仅获得或修改内联样式

(1)elem.style.属性名="值";

每个元素的style属性其实都是一个style对象

style对象中包含了所有css样式属性，默认值都是" "

\*无法获得样式表中层叠或继承来的属性，只能访问elem的内联样式(style属性中定义的样式)

(2)获得样式表中层叠或继承来的属性

①DOM标准： 第一步 var style对象=getComputedStyle(elem);

第二步 style对象.属性名

②IE8：第一步 var style对象=elem.currentStyle;

第二步 style对象.属性名

\*仅能获得样式，无法修改样式

如果要改：

改内联：elem.style.属性名="值";

改外部：见下4

2、获取或设置内联样式字符串

elem.style.cssText="";

3、设置一个内联属性值

elem.style.setProperty("属性名","值");

4、修改外部样式表：3步

(1)获得要修改的样式表对象

var sheet=document.styleSheets[i];

(2)获得要修改的cssRule

cssRule：样式表中每个大括号包裹的内容

var cssRule=sheet.cssRules[i];

(3)修改cssRule中的属性

cssRule.style.属性名=值;

\*牵一发而动全身，谨慎使用

解决：直接修改内联样式的属性，来覆盖样式表中的属性

十一、创建和删除节点

1、创建单个节点：3步

(1)创建空节点对象：

var elem=document.createElement("标签名");

例如var a=document.createElement("a"); →<a></a>

(2)设置关键属性：

例如 a.href="http://www.baidu.com";

a.innerHTML="baidu";

→<a href="http://www.baidu.com">baidu</a>

(3)将新节点挂到指定父元素下：3种

①追加 parent.appendChild(elem);

②插入 parent.insertBefore(elem,oldElem);

③替换 parent.replaceChild(elem,oldElem);

减少修改DOM树(第3步)的次数，可减少排版引擎渲染页面的次数，可提高执行效率

→将所有子元素都添加到父元素下后，再将父元素一次性加入页面

2、添加多个平级子元素：3步

(1)先创建一个文档片段

文档片段：内存中临时保存多个节点对象的空间，内存中一个临时的父元素

var frag=document.createDocumentFragment();

(2)将创建的子元素，先加入到片段中

frag.appendChild(newElem);

frag.insertBefore(xxx,xxx);

frag.replaceChild(xxx,xxx);

frag用法同普通父元素

(3)将文档片段作为整体，添加到页面指定父元素下

片段不出现在DOM树中，也不会出现在页面上

3、删除节点

var 子节点对象=parent.removeChild(子节点对象);

自己删除自己：

子节点.parentNode.removeChild(子节点);

十二、HTML DOM常用对象

1、Select对象

属性：

(1)sel.options，返回当前select下所有option对象的集合

(2)sel.selectedIndex，获得当前select的选中项的下标位置

方法：

(1)sel.add(opt)，添加一个option

等效于sel.appendChild(opt);

(2)sel.remove(i下标位置)，移除一个option

js如何绑定事件处理函数

元素的事件处理函数(比如onchange)，其实就是元素对象的成员

元素.onxxx=function(){

执行事件的操作

}

当事件发生时，自动执行 元素.onxxx();

2、Option对象

创建option对象：

var opt=new Option(innerHTML,value);

等效于： var opt=document.createElement("option");

opt.innerHTML="...";

opt.value=...;

属性：

(1)opt.selected，返回当前option是否被选中，可直接当bool用

(2)opt.index，获得opt在select中的下标位置

3、Table对象

方法：每种行分组都有一个create方法，都有一个delete方法(delete除tbody没有)

(1)插入空行：var tr=table.insertRow(i);

如果省略i，表示在末尾追加一行

(2)删除行：table.deleteRow(i);

如果要删除行，必须要知道下标

属性：

(1)tHead tFoot

(2)tBodies，多个tbody元素对象的集合

必须用[i]的方式获取每个tbody

(3)rows，获得表格中所有tr对象

HTMLTableSectionElement：表格中每个行分组

方法： var tr=tbody.insertRow(i);

tbody.deleteRow(i);

\*i是相对于tbody的序号，是不包含thead中tr的序号

HTMLTableRowElement：表示表格中的每一行

方法： var td=tr.insertCell(i);

tr.deleteCell(i);

如果省略i，表示在末尾追加一行

属性： tr.cells：保存了一行中所有td对象

tr.rowIndex：保存了当前行的下标，专用于删除行

HTMLTableCellElement：表示行中的每一个td

4、Form对象

如何获得Form对象：

var form=document.forms[i或name];

方法：form.submit()，用程序代替自动提交

事件：onsubmit，当表单提交之前自动触发

主要用于提交前的验证

如何获得Form中的数据采集元素：

form.elements[i或name];

方法：elem.focus();