**正课:**

**1. 查找:**

**2. 修改:**

**1. 查找:**

**1. 按节点间关系查找:**

**何时: 如果已经获得一个元素，要找周围元素时**

**如何:**

**节点树: 包含所有网页内容(元素,文本等)的树结构**

**2大类:**

**1. 父子关系:**

**node.parentNode 获得node的父节点**

**node.childNodes 获得node的所有直接子节点**

**node.firstChild 获得node的第一个直接子节点**

**node.lastChild 获得node的最后一个直接子节点**

**2. 兄弟关系:**

**node.previousSibling 获得node的前一个兄弟**

**node.nextSibling 获得node的后一个兄弟**

**问题: 会受到看不见的空字符的干扰！**

**元素树: 仅包含元素节点的树结构**

**优点: 不包含空字符节点，不受空字符干扰！**

**强调: 元素树不是一棵新树，只是节点树的子集**

**包括:**

**2大类:**

**1. 父子关系:**

**node.parentElement 获得node的父元素**

**node.children 获得node的所有直接子元素**

**node.firstElementChild 获得node的第一个直接子元素**

**node.lastElementChild获得node的最后一个直接子元素**

**2. 兄弟关系:**

**node.previousElementSibling 获得node的前一个兄弟元素**

**node.nextElementSibling 获得node的后一个兄弟元素**

**鄙视: 遍历指定父元素下所有子节点: 2步:**

**1. 先仅遍历直接子节点**

**问题: childNodes和children返回的是动态集合**

**动态集合: 不实际存储属性值，每次访问集合，都重新查找DOM树**

**优: 首次查找速度快，因为不必返回所有数据**

**缺: 反复访问集合，会导致反复查找DOM树**

**所以: for(var i=0;i<children.length;i++) 不好**

**解决: for(var i=0,len=children.length;i<len;i++)**

**2. 对每个直接子节点，调用和父节点完全相同的函数**

**2. 按HTML查找: 4个:**

**1. 按id查找:**

**var elem=document.getElementById("id")**

**强调: 1. 只能用document调用**

**2. 每次只能找到一个元素对象**

**2. 按标签名查找:**

**var elems=parent.getElementsByTagName("标签名")**

**强调: 1. 可在任意父元素上调用**

**2. 返回动态集合**

**3. 不仅查找直接子元素，且查找所有后代元素**

**3. 按name属性查找:**

**var elems=document.getElementsByName("name")**

**强调: 1. 只能在document上调用**

**2. 专门用于查找表单元素**

**3. 返回动态集合**

**4. 按class属性查找:**

**var elems=parent.getElementsByClassName("class")**

**强调: 1. 可在任意父元素上调用**

**2. 返回动态集合**

**3. 不仅查找直接子元素，且查找所有后代元素**

**4. 只要用修饰当前元素的多个class中任意一个class，都可找到当前元素。**

**3. 按选择器查找:**

**1. 仅查找一个元素:**

**var elem=parent.querySelector("选择器")**

**2. 查找多个元素  
 var elems=parent.querySelectorAll("选择器")**

**强调：1. 可在任意父元素上调用**

**2. 不仅查找直接子元素，且查找所有后代**

**3. 受制于浏览器的兼容性限制**

**4. 返回非动态集合**

**非动态集合: 返回完整数据的集合，即使反复访问集合，也不会反复查找DOM树**

**优: 即使反复访问集合，也不会反复查找DOM树  
 缺: 首次查找效率低**

**鄙视: 按HTML vs 按选择器, 如何选择:**

**1. 返回值:**

**按HTML查找，返回动态集合**

**按选择器查找，返回非动态集合**

**2. 首次查找效率:**

**按HTML查找，效率高**

**按选择器查找，效率低**

**3. 易用性:**

**按选择器查找，简单好用**

**按HTML查找，繁琐**

**总结: 如果只靠一个条件就可找到元素时，首选按HTML找**

**如果查找条件复杂，首选按选择器查找**

**2. 修改: 3种:**

**内容: 3种:**

**1. 元素的HTML代码片段内容: .innerHTML**

**2. 元素的纯文本内容: .textContent**

**vs .innerHTML: 删除内嵌标签 并 转移字符翻译为正文  
 3. 表单元素的值: .value**

**属性: 3种:**

**1. HTML标准属性:2种:**

**1. 核心DOM API:**

**.getAttribute("属性名")**

**.setAttribute("属性名","值")**

**.hasAttribute("属性名")**

**.removeAttribute("属性名")**

**2. HTML DOM API: 对核心DOM中常用的API进行的简化**

**HTML DOM已经提前将所有标准属性做成了内存中元素对象的属性，默认值都是"": elem.属性名**

**特例: class**

**ES标准中已经有了class属性**

**所以DOM中将网页的class属性，更名为className**

**2. 状态属性: enabled disabled selected checked**

**值是bool类型！不能用核心DOM操作**

**只能用HTML DOM打.操作**

**3. 自定义扩展属性**

**什么是: 程序员自发添加的自定义属性**

**何时: 2个场景:**

**1. 在客户端临时缓存业务数据**

**2. 代替元素/class等选择器，用于查找元素绑定事件**

**元素和class做选择器添加行为的弊端:**

**元素和class极有可能频繁修改，导致js程序失效**

**解决: 今后，所有事件绑定都要查找自定义属性**

**如何:**

**添加自定义扩展属性:**

**在开始标签中: data-属性名="值"**

**访问自定义扩展属性:**

**肯定不能用HTML DOM，因为内存中的对象上，没有！**

**2种:**

**1. 用核心DOM:**

**2. HTML5标准: elem.dataset.自定义属性名**

**dataset可自动收集开始标签中的data-开头的自定义属性**

**样式:**

**只有一种样式可修改: 内联样式**

**elem.style.css属性="值"**

**相当于:<elem style="css属性:值"**

**css属性名都要去横线变驼峰**

**z-index => zIndex**

**获取样式: 不能用.style，仅能获得内联样式**

**解决: 获得计算后的完整样式:**

**var style=getComputedStyle(elem)**

**var value=style.css属性**

**限制: 计算后的属性禁止修改！因为来源不确定！可能牵一发而动全身！**

**总结: 如果修改样式: .style.css属性**

**如果获取样式: getComputedStyle(elem)**