正课:

1. 查找

2. 修改

3. 添加删除元素

1. 查找:

1. 不需要查找就可直接获得元素

2. 按节点间关系查找

何时: 必须已经获得一个元素，找周围附近的元素时

3. 按HTML特征查找:

何时: 如果一个元素都没获得，首次查找元素时

包括: 4个特征:

1. 按id查找一个元素

var 元素=document.getElementById("id值")

强调:

1. 只能用document调用，不能换成其他元素。

2. 只能找到一个元素对象。如果找不到返回null！

2. 按标签名查找多个元素

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

强调:

1. 可在任意父元素下查找。主语是哪个父元素，就只在哪个父元素内查找符合条件的元素

2. 返回多个符合条件的元素组成的集合。如果找不到，返回空集合: []

3. 不仅查找直接子元素，而是在所有后代中查找符合条件的元素。

3. 按class名查找多个元素

var 集合=任意父元素.getElementsByClassName('一个class名')

强调:

1. 可以在任意父元素上调用，控制查找的范围。

2. 返回多个符合条件的元素的集合。如果找不到，返回空集合: []

3. 不仅在直接子元素中查找，而是在所有后代中查找符合条件的。

4. 如果一个元素有多个class修饰，而查找时，只要用其中一个class，就能找到该元素。

4. 按name属性查找表单元素

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

强调:

1. 只能用document调用

2. 返回多个符合条件的表单元素的集合。如果找不到，返回空集合: []

问题: 一次只能按1个条件查找！如果查找条件复杂，元素藏得很深，还需要筛选时，用按HTML特征查找，就会需要很多次查找，才能找到想要的元素。代码繁琐！

解决: 选择器

4. 按选择器查找:

何时: 查找条件非常复杂时

如何: 2个函数：

1. 如果确定只找一个元素时：

var 元素=任意父元素.querySelector("任意css选择器")

2. 如果找到的是多个元素时:

var 元素=任意父元素.querySelectorAll("任意css选择器")

强调: ()中的选择器，是相对于.前的父元素下的相对选择器，而不是完整的选择器

提前讲: 事件:

什么是事件: 浏览器自动触发的，或用户手动触发的页面内容或状态的改变

比如: 单击一个按钮时

什么是事件属性: 每个元素上，都有一批以onxxx开头的特殊属性。表示"当发生什么事儿时"。每个onxxx属性，都对应着一个事件！

比如: btn.onclick: null 对应着"当单击这个按钮时"事件

什么是事件处理函数: 提前保存在元素的事件属性上，当事件发生时，自动调用的函数

何时使用事件处理函数: 只要希望一个元素上发生某件事时，可以自动执行一个提前设计好的任务

如何: 2步:

1. 查找要触发事件的元素

2. 为元素的对应事件属性onxxx赋值（绑定）一个函数。

btn.onclick=function(){ ... }

强调: 如果事件处理函数，只是赋值给onxxx事件属性，是暂时不执行的。只有当事件发生在这个元素上时，才自动调用处理函数。

原理: 当该元素，发生对应事件时，先找到元素的时间属性，然后自动加()，调用实现保存好的处理函数

比如: 当单击按钮时，会自动执行:

当前按钮.onclick()

如果要想在事件处理函数中快速获得当前触发事件的元素，只能用this->当前按钮

习惯: 因为this比较晦涩，于是多数程序员，在事件处理函数开头，都会给this起一个有意义的别名。

getElementsByxxx vs querySelectorAll(xxx)

1. 效率: getElementsByxxx() 高！

querySelectorAll(xxx) 低！

2. 易用: querySelectorAll(xxx) 简洁，好用

getElementsByxxx() 步骤繁琐！

总结: 如果只靠一个条件就能找到想要的元素时，首选getElementsByXxxx()

如果查找条件复杂时，就首选querySelectorAll()

2. 修改: 3种:

1. 获取或修改元素的内容:

1. 获取或修改元素开始标签到结束标签之间的原始的HTML片段：元素.innerHTML

2. 获取或修改元素开始标签到结束标签之间的纯文本内容: 元素.textContent

vs 元素.innerHTML

多做了两件事:

1. 去掉内嵌的标签

2. 将特殊符号翻译为正文

3. 获取或修改表单元素的值

不能用.innerHTML，因为表单元素多数是单标记，没有innerHTML

表单元素的内容都是用.value属性获取或修改的

2. 获取或修改元素的属性: 3种:

1. HTML标准属性: HTML标准中规定的属性

比如: id, title, class, href, src, ...

2种：

1. 最初的核心DOM函数: 4个  
 1. 获取属性的值:

元素.getAttribute("属性名")

比如: 获得a元素的href属性:

a.getAttribute("href")

2. 修改属性值:

元素.setAttribute("属性名","新值")

比如: 修改a元素的title属性:

a.setAttribute("title","go to tmooc")

3. 判断是否包含属性:

var bool=元素.hasAttribute("属性名")

比如: 判断a元素是否有id属性

a.hasAttribute("id")

4. 移除属性:

元素.removeAttribute("属性名")

比如: 移除a元素的target属性:

a.removeAttribute("target")

问题: 单词太长！

2. HTML DOM: 专门针对HTML进行的简化

HTML DOM提前将所有的HTML标准属性都定义在了元素对象上。只不过，属性值暂时是""

当手动在<开始标签>中加入了属性值或用程序在内存中添加了属性值，则该属性的值变为指定的新值。

福利: 其实所有HTML标准属性，都可用.简化访问。

1. 获取属性值: 元素.属性名

2. 修改属性值: 元素.属性名=值

3. 判断是否包含该属性:

元素.属性名!==""

4. 移除属性:

元素.属性名=""

特例: class属性，不能直接用.访问

class属性是ES中的关键词！

html不能再使用class属性

今后，如果操作页面上的class属性，只能改名为.className。使用"元素.className"属性，等效于使用<元素 class="">属性

2. 状态属性:

3. 自定义扩展属性:

3. 获取或修改元素的样式:

1. 只修改一个内联样式：

html: <div id="d1" style="display:none">

存储: div:{

nodeName: "DIV",

id:"d1",

style:{

... : ... ,

display:"block" ,

... : ...,

}

}

要控制div的显示和隐藏:

div.style.display="none"或"block"

相当于给div元素添加一个内联样式display:none或block

只要修改DOM树中的对象，都会被浏览器侦测到，浏览器就会自动重绘页面

问题: style如果用于获取样式属性值时，只能获得内联样式中的属性值！无法获得从样式表或浏览器默认样式中继承或重叠来的样式。所以，将来读取一个元素的完整css样式时，不能用"~~元素.style.css属性~~"。

解决: 获取样式时，要用专门的getComputedStyle(元素)——获得计算后的样式.

什么是计算后的样式: 最终应用到这个元素上的所有样式的集合！

何时: 只要想获取css属性值，又不用担心丢失样式时，都用getComputedStyle(元素)

如何: 2步:

1. 先获得完整的style对象

var style=getComputedStyle(元素)

强调: getComputedStyle()所有浏览器自带，不用自己定义！

2. 从style对象中获取想要的css属性值

style.css属性名

总结: 今后，只要修改一个样式:

元素.style.css属性="值"

今后，只有获取一个样式时:

var style=getComputedStyle(元素);

style.css属性 ——只读

单词:

1. selector 选择器

2. Content 内容

3. execute 执行

4. Attribute 属性: 特指html开始标签中出现的特性

比如: <h1 id="d1" title="welcome" ... >

5. property 属性: 特指内存中对象的属性

比如: lilei.sage, arr.length, ...