正课:

1. 什么是DOM  
2. \*\*\*查询

3. 修改

1. 什么是DOM

DOM: Document Object Model

专门操作网页内容的API标准——W3C

何时: 只要操作网页内容，都必须用DOM提供的API

分为:

核心DOM: 最初制定的操作一切结构化文档的统一API

优: 万能

缺: 繁琐

HTML DOM: 专门操作HTML文档的API

对核心DOM中部分常用API进行了简化

优: 简洁

缺: 不是万能

总结: 实际开发中不必区分核心DOM和HTML DOM

都是优先使用简单的API

如果简单地实现不了，用复杂的补充

\*\*\*DOM树:

什么是:

网页中一切内容在内存中都是以树型结构存储在一起

每项内容(元素，文本，属性)都是树上的一个节点对象

为什么: 树形结构是保存不确定层级深度的上下级包含关系最好的结构

根节点: 树结构都有唯一的一个树根节点: document节点

所有网页内容，都是document的子节点

节点对象:

什么是: 网页中每一项内容都是一个节点对象

节点对象封装了节点可用的属性和功能

如何: 三大属性:

nodeType: 节点类型:

什么是: 定义了节点的类型

何时: 区分节点的类型时

为什么: 不同类型的节点可用的属性和可执行的操作不一样

包括: 4种:

document 9

element 1

attribute 2

text 3

问题: 无法进一步区分具体的元素名

解决: nodeName

nodeName:

什么是: 保存节点名称的属性

何时: 进一步区分具体的元素名

为什么: 不同的元素拥有的属性和可执行的操作都不一样

包括:

document #document

element 标签名(全大写)

attribute 属性名

text #text

nodeValue: 了解

什么是: 保存节点的值

何时: 几乎不用

包括:

document null

element null

attribute 属性值

text 文本内容

DOM操作的流程: 增删改查+事件处理

查找触发事件的元素->绑定事件处理函数

->查找要修改的元素->修改属性/样式,添加,删除

2. \*\*\*查找: 4种:

不需要查找可直接获得的元素:

document.documentElement -> html

document.head -> head

document.body -> body

1. 按节点间关系查找:

何时: 如果已经获得了一个元素，想找周围元素时

节点树: 包含所有节点的完整树结构

包括:

1. 父子关系:

child.parentNode 获得一个节点的父节点

parent.childNodes 获得父节点下的所有直接子节点

parent.firstChild 获得父节点下的第一个直接子节点

parent.lastChild 获得父节点下的最后一个直接子节点

2. 兄弟关系:

elem.nextSibling 获得一个节点相邻的下一个兄弟节点

elem.previousSibling获得一个节点相邻的前一个兄弟节点

问题: 受看不见的空字符的干扰

解决: 元素树

元素数: 仅包含元素节点的树结构

何时: 只要仅关心元素，不关心其它类型的节点时

如何:

1. 父子关系:

child.parentElement 获得一个节点的父元素

parent.children 获得父元素下的所有直接子元素

parent.firstElementChild 获得父元素下的第一个直接子元素

parent.lastElementChild 获得父元素下的最后一个直接子元素

2. 兄弟关系:

elem.nextElementSibling 获得一个元素相邻的下一个兄弟元素

elem.previousElementSibling获得一个元素相邻的前一个兄弟元素

说明: 元素树不是一棵新树,只是节点树的部分子集

2. 遍历节点:

什么是: 查找一个父元素下所有后代节点

如何: 2步:

1. 定义函数仅遍历直接子节点

2. 对每个碰到的子节点调用和父节点完全相同的操作

算法: 深度优先: 每当同时有子元素和兄弟元素时，总是先遍历子元素。子元素遍历完，才返回遍历兄弟元素

问题: 递归的执行效率极低

解决: 用循环代替

如何: 用节点迭代器对象

什么是节点迭代器对象: NodeIterator

专门按照深度优先遍历的顺序依次访问每个子元素的对象。

如何使用: 2步:

1. 创建NodeIterator:

var iterator=document.createNodeIterator(

parent, NodeFilter.SHOW\_ALL, null, false

SHOW\_ELEMENT

);

2. 用循环反复调用NodeIterator的nextNode()函数

做2件事:

1. 返回当前节点

2. 蹦到下一个节点

到返回null退出