**DOM**

* 事件
  + 事件类型
    - 鼠标事件
      * 点击事件
        + mydiv.onclick=function(){alert('点击事件')}
      * 双击事件
        + mydiv.ondblclick=function(){alert('双击事件')}
      * 鼠标按下
        + mydiv.onmousedown=function(){mydiv.innerHTML='按下事件';}
      * 鼠标抬起
        + mydiv.onmousedown=function(){mydiv.innerHTML='按下事件';}
      * 鼠标移动
        + document.onmousemove=function(){console.log('移动了')}
      * 鼠标移入
        + 会触发多次,取决于嵌套和鼠标位置(从内逐级向外依次触发)
        + mydiv.onmouseover=function(){console.log('移入了');}
        + mydiv.onmouseenter=function(){console.log('yichule')}
      * 鼠标移出
        + 会触发多次,取决于嵌套和鼠标位置(从内逐级向外依次触发)
        + mydiv.onmouseout=function(){console.log('移出了');}
        + mydiv.onmouseleave=function(){console.log('移出了');}
    - 键盘事件
      * 键盘的按钮按下时会触发onkeydown事件，在函数中的e参数代表时哪一个事件触发的
      * document.onkeydown=function(){alert('按下')}
      * document.onkeyup=function(){ alert('按下') }
  + Event事件对象
    - Event事件对象的概念
      * 当用户单击某个元素的时候,我们给这个元素注册的事件就会触发,该事件的本质就是一个函数,而该函数的形参接收一个event对象.事件通常与函数结合使用，函数不会在事件发生前被执行
    - Event事件对象接收方式
      * 通过事件触发时的函数，一参数的形式传递进该函数函数内，不用靠调用传参，也就是说event对象是事件触发是调用函数的第一个参数
      * document.onclick=function(e){alert(1)；}
    - event对象常用属性
      * 鼠标的坐标值：
        + e.clientX和e.clientY
      * 键盘的值：
        + e.keyCode
    - event的兼容性
      * event对象根据不同浏览器他的获取时不同的方式，一下是针对获取event对象的兼容性写法
      * document.onclick=function(e){//||ie低版本ievarev=e||window.event;console.log(ev.clientX)}
  + 对象事件
    - 页面（元素）加载完成后执行事件
      * indow.onload是等待页面完成之后执行函数的内容，当然函数没有重载也就是说window.onload只能写一次
      * //页面加载之后执行函数内的代码段。window.onload=function(){varmydiv=document.getElementById('mydiv');mydiv.onclick=function(){alert(mydiv.tagname)}}
* DOM概述
  + dom概念
    - DOM文档对象模型
  + DOM和JavaScript的关系
    - DOM能够使用JavaScript脚本语言进行修改
  + dom节点
    - 整个文档是一个文档节点(document)
    - 每个html元素就是元素节点
    - 每个html元素就是元素节点
    - 每个html属性都是属性节点
    - 注释也是节点,叫注释节点
  + DOM树
    - DOM文档树
      * 包括dom文档中的所有内容
    - DOM元素树
      * 包含的只有元素节点
* 获取元素
  + 用ID获取元素
    - vard=document.getElementById('d');
  + 用标签名获取元素
    - varlis=document.getElementsByTagName('li')
    - 伪数组
  + 利用类名获取元素
    - vara=document.getElementsByClassName('.a');
  + 用选择器获取元素
    - varx=document.querySelector('#div>diva');
      * 只能找到元素列表的第一个元素
    - varlis=ul.querySelectorAll('li');
      * 可以全部找到
    - 该方法性能没有直接利用标签寻找高.
  + 用name属性值获取元素
    - varsex=document.getElementsByName('sex');
    - 可以获取相同名称的name元素
    - 返回一个伪数组对象
* 获取和设置元素中的其他信息
  + 获取元素名
    - vardiv=document.getElementById('mydiv');console.log(div.tagName);//DIV
    - 属性：TagName
    - 该属性只能获取不能设置.
  + 获取和设置元素节点里的内容
    - 获取
      * 需要拿到元素中的内容(所有东西)
      * 元素里的内容可能包括:文本或元素
      * 属性：innerHTML
      * div.innerHTML
    - 设置
      * div.innerHTML='你好';
  + 获取和设置元素节点中的文本
    - 获取
      * 获取的只能是文本节点
      * div.innerText
    - 设置
      * div.innerText='hello'
  + 获取和设置元素的类名
    - 获取
      * div.className
      * 属性：className
      * 返回值是一个字符串
    - 设置
      * div.className='dac';
      * 如果更换其中一个类,则需要考虑该字符串中其他类是否应该一起设置
  + 获取元素样式
    - 属性：style
    - 可以获取元素内联样式的属性
    - 继续在style中找属性名如backgroundcolor，就可以找到该属性的值，以字符串的形式返回
    - 可以设置该属性，从而达到增加或更改样式
    - 以js形式增加的样式优先级大于css优先级
  + 获取元素的属性
    - div.getAttribute('box')
    - 返回值就是字符串形式的属性值
  + 设置元素属性
    - setAttribute('id','mydiv')
    - 第一个参数是属性名，第二个参数是属性值，都需要用引号括起来一逗号分隔
  + 删除元素属性
    - div.removeAttribute('id')
* 增加元素
  + 创建元素节点
    - 用一个变量承接在文档中‘创建’的一个元素
    - vardiv=document.createElement('div');
    - 将已经创建的元素放在相仿的位置
    - document.body.appendChild(div);
  + 创建文本节点
    - document.createTextNode('yiduanwenzi')
    - p.appendChild(text)
  + css样式赋予
    - div.style.cssText='width:100px;height:100px;';
  + 插入创建的新元素
    - 第一个是要插入的子元素，第二个是在谁之前插入，两个参数用逗号隔开
    - ul.nserBefore（li，pg）
* 删除元素和替换
  + 元素的替换
    - 元素的替换实在父元素中一个子元素需要被另一个新的子元素替代
    - document.body.replaceChild(myp,mydiv);
    - 第一个参数是将要替换的新元素，第二个是被替换的旧元素，中间用逗号分隔
  + 元素的删除
    - 在父元素的其中一个子元素需要删除
    - document.body.removeChild(mydiv);
* 查找节点
  + 层次节点
    - 父节点
    - 子节点
    - 兄弟节点
    - 知道其中之一的时候，可以用一些方法找到另一个
  + 获取所有子节点
    - myul.childNodes
    - 包含所有节点类型，不单单只是元素
    - 属性：childNodes
  + 获取第一个和最后一个子节点
    - myul.firstChild
    - myul.lastChild
    - 不一定是元素节点
  + 获取父节点
    - myul.parentNode
    - 父节点只有一个
  + 获取兄弟节点
    - 前一个兄弟节点
      * myul.previoussibling
    - myul.nextSibling
    - 只能获取一个
* 元素的位置属性
  + 包含边框和内边距
    - 使用offsetWidth和offsetHeight属性。改属性可以获取元素的占位宽高
      * 只能拿到内联样式
  + 不包含边框
    - clientwidth和clinetheight属性也可以获取元素的宽高
* 元素与父元素的距离
  + offsetLeft和offsetTop是距离body左边界和上边界的距离
  + 如果该子元素使用了定位属性，则offsetLeft和offsetTop参照的就不在是body而是距离它最近的父元素