**DOM 部分**

**DOM**（Document Object Model ）

文档对象模型

DOM是关于创建，插入，修改，删除页面元素的标准；

DOM赋予我们操作页面的能力。

页面中的内容本质上都是字符串，但是js会把这些字符串转成DOM树，DOM树中的内容会被解析成DOM节点，那就是说，我们在操作DOM本质上就是操作每一个节点，页面中的任何内容都是节点

**attributes**

元素的所有属性集合(不包含属性值)

元素.attributes

每一个属性都会对应一个下标，它还有一个length属性，代表属性的个数

**childNodes**

返回元素下的第一层所有子节点的集合，包含回车、文字、注释

元素.childNodes

每一个子节点都会对应一个下标，它还有一个length属性，代表子节点的个数

**nodeType**

返回节点的类型，这个节点类型是一个数字

节点.nodeType

**nodeName**

返回节点的名字

节点.nodeName

**tagName**

返回元素的标签名 (大写)

元素.tagName

只有标签元素才有这个属性

nodeName

返回节点的名字

节点.nodeName

任何节点都有这个属性（文本、注释）

**parentNode**

节点.parentNode

返回这个节点的父节点

属性节点是没有爹的

**childNodes**

返回的是元素第一层的所有子节点的集合，包含文本和注释节点

元素.childNodes

**children**

返回的是元素第一层的所有子节点的集合，但是它只包含标签元素.

它是一个类数组，每一个子节点都会对应一个下标，可以通过下标值取到某个子节点

元素.children

这个属性返回的结果都是标签，但是它是非标准的

**上一个兄弟节点** previousElementSibling

节点.previousElementSibling

如果没有找到，那返回一个null

**下一个兄弟节点** nextElementSibling

节点.nextElementSibling

如果没有找到，那返回一个null

**元素下的第一个子节点** firstElementChild

元素.firstElementChild

元素为父节点

返回这个元素下的第一个子节点

如果没有，返回null

**元素下的最后一个子节点** lastElementChild

元素.lastElementChild

元素为父节点

返回这个元素下的最后一个子节点

如果没有，返回null

**offsetParent**

作用

找到最近的有定位(不能为static)的父级，它会一层一层往外找，如果没找到的话，默认为body

父级必需为相对定位或者绝对定位，自己有没有定位无所谓

语法

元素.offsetParent

**offsetLeft**

作用

找到最近的有定位的父级，获取到它与父级的left距离

从元素的最左边到定位父级的最左边的距离

语法

元素.offsetLeft

**offsetTop**

作用

找到最近的有定位的父级，获取到它与父级的top距离

从元素的最上边到定位父级的最上边的距离

语法

元素.offsetTop

注意：如果没找到有定位的父级，那这是到html的距离

**getBoundingClientRect**

作用

获取元素的盒模型的相关信息

width

height

left

top

right

bottom

语法

元素.getBoundingClientRect()

返回值

返回一个对象，这个对象当中存了元素的盒模型的相关信息

注意

得到的left，是元素左边到左边可视区的距离

得到的right，是元素右边到左边可视区的距离

得到的top，是元素上边到上边可视区的距离

得到的bottom，是元素下边到上边可视区的距离

得到的width，是包含边框的

**属性操作**

1、.属性名

2、['属性名']

**getAttribute**

作用

获取属性，它不光能获取标签自带的属性，也可以获取到自定义属性

语法

元素.getAttribute(属性名)

参数

要获取的属性名字

返回值

返回值就是获取到的结果

它获取到的src、href这些属性的值

src里写的是什么内容，获取到的就是什么内容

**setAttribute**

作用

设置属性，可以设置自定义属性，也可以设置标签自带的属性

语法

元素.setAttribute(attr,value) 例： box.setAttribute('miav','kaivon');

参数

attr 要设置的属性名

value 要设置的属性值

返回值

无

**removeAttribute**

作用

删除属性

语法

元素.removeAttribute(attr)

参数

要删除的那个属性名字

返回值

它没有返回值

**元素的宽**

**offsetWidth**

得到元素的宽度，包含border的宽度

**clientWidth**

得到元素的宽度，但是不包含border的宽度

**元素的高**

**offsetHeight**

得到元素的高度，包含border的高度

**clientHeight**

得到元素的高度，但是不包含border的高度

**可视区的宽**

document.documentElement.clientWidth

**可视区的高**

document.documentElement.clientHeight

如果窗口改变，这两个值也会改变

**createElement**

作用

创建元素：根据参数的名字创建相对应的标签

语法

document.createElement(tagName)

只有document才有这个方法

参数

要创建的标签名字

返回值

返回新创建的元素

创建后的标签可以给它添加属性样式等

它不能创建文本、注释节点

**appendChild**

作用

添加元素：往父级元素里面添加相应的子节点

语法

父节点.appendChild(childNode)

参数

要添加的元素(只能是一个标签节点)

返回值

返回要添加的那个元素（参数），只能添加标签元素

**insertBefore**

作用

插入元素：把第一个节点插入到第二个节点前面

语法

父节点.insertBefore(childNode1,childNode2)

参数

childNode1 要插入的节点

childNode2 决定插入节点位置的节点

第二个参数的结果为null的话，相当于appendChild,把参数1的节点添加到父节点的最后

注意：两个参数必需同时存在，不然会报错

返回值

返回要插入的节点（第一个参数）

**removeChild**

作用

删除节点：

语法

父节点.removeChild(childNodes)

参数

要删除的子节点（只能为单个节点）

返回值

被删除的元素（参数）

注意：这个方法可以把文本节点与注释节点删掉

**replaceChild**

作用

替换节点：用节点1替换节点2

语法

父节点.replaceChild(node,childNode)

参数

node 替换成的节点

childNode 被替换的节点

注意：

1、两个参数必需同时出现，不然报错

2、如果两个参数都有同一个父节点，那么会把第二个删掉。并且位置也会换了，换到第二个节点的位置

返回值

返回被替换掉的那个元素（参数2）

**cloneNode**

作用

克隆节点

语法

要被克隆的节点.cloneNode(Boolean)

参数

true 克隆元素和元素包含的子孙节点

false 克隆元素但不包含元素的子孙节点

注意

1、如果没有参数，默认为false

2、克隆的只克隆css、html不克隆js

3、克隆后的节点的id与原来的是一样，所以要修改一下

**appendChild/insertBefore/replaceChild**在操作一个已有的元素时,是将已有的元素移动，而不是复制一份进行操作(剪切)

**innerHTML与DOM的区别**

用innerHTML方法添加后，其它是先把原来的内容清空，再把新的内容添加进来，所以原来的事件会被删除

DOM方法只是在原来的基础上增加了一些元素，原来的事件还会有

**getElementsByTagName**

动态获取元素

一旦获取的元素有变化，那它就会重新获取一下，原来的索引就会变成新获取的

**querySelectorAll**

不会动态获取元素

只获取一次，每个元素对应的下标是不会变的，即使元素有变化，它也不会重新再去获取，原来的下标不会变

**DOM的表格表单操作**

**获取表格头部**

表格.tHead

获取到表格头里面的所有内容，它是个字符串

表格头只有一个，如果不有的话返回null

**获取表格主体**

表格.tBodies

获取到表格主体内容，因为tbody在表格中是可以放多个的，所以它获取到的是类数组

**获取表格底部**

表格.tFoot

获取到表格底部里面的所有内容，它是个字符串

表格底部只有一个，如果不有的话返回null

**获取表格行**

tBodies[n].rows 获取表格主体里的某行

tHead.rows 获取表格头部里的行

tFoot.rows 获取表格底部里的行

表格.rows 获取表格里的行

获取到的是一行,是一个集合，用的话需要用下标去取，可以直接给这一行进行操作

**获取单元格**

rows[n].cells

获取到某一行的所有单元格，它是一个类数组，不能直接操作整行。只能操作单个

**表单操作**

获取表单的另一种方法用name

表单.name

获取值

text/radio/checkbox/select/textarea

node.value

事件

onchange 当有变化的时候会触发这个事件，一般用在select

oninput 当输入框里的内容有变化的时候触发

onsubmit 当表单被提交的时候会触发,当点击提交按钮的时候会触发，只能添加给form对象

onreset 当表单被重置的时候触发，当点击重置按钮的时候会触发，只能添加给form对象

方法

reset() 作用为重置表单，会触发onreset事件，只能给form对象添加

submit() 作用为提交表单，不会触发onsubmit事件，只能给form对象添加

**BOM操作**

**BOM(Browser Object Model)**

浏览器对象模型

BOM是关于浏览器的方法，属性，事件

**打开一个新窗口**

window.open()

参数：

1.指定要打开的页面地址。

2.打开方式 ： \_blank \_self ... iframName

3.浏览器的窗口特征 （宽，高，窗口位置等）

**关闭当前窗口**

window.close()

不同的浏览器表现形式不一样

在ff/O下，只允许关闭由window.open方法所打开的窗口

**用户代理信息**

window.navigator.userAgen

**地址栏信息**

window.location

window.location.href

字符串版的地址栏信息

window.location.search

地址栏查询信息 （问号到#号之间的所有内容,包含问号，不包含#）

window.location.hash

锚点信息（#号后面的所有内容,包含#）

**可视区尺寸**

（BOM方法）高版本的浏览器兼容

window.innerWidth / window.innerHeight

Window.onresize:当浏览器窗口大小改变是触发。

**事件**

系统中给我们提供的函数，这个函数会在某个动作发生的时候被触发。

函数要做的具体内容是由我们来定的

**鼠标事件**

onclick 鼠标点击事件

ondblclick 鼠标双击事件

onmousedown 鼠标按下事件

onmouseup 鼠标抬起事件

onmousemove 鼠标移动事件

onmouseover 鼠标移入事件

onmouseout 鼠标移出事件

onmouseenter 鼠标进入事件 (不会传递给子级，不存在是子级事件冒泡)

onmouseleave 鼠标离开事件 (不会传递给子级，不存在是子级事件冒泡)

**焦点事件**

onfocus

当有焦点的元素获取到焦点的时候触发

用tab键也会触发这个事件

onblur

当有焦点的元素失去焦点的时候触发

tabindex

tab键获取焦点的顺序

支持onfocus的对象有

button, checkbox, fileUpload, layer, frame, password, radio, reset, select, submit, text, textarea, window

设置焦点的方法

focus()

元素有焦点才可以用这个方法

作用：让这个元素获取到焦点

blur()

元素有焦点才可以用这个方法

作用：让这个元素失去焦点

**键盘事件**

onkeydown

当键盘的按键按下去的时候触发，如果按键没有抬起来，那这个事件会一直触发

onkeyup

当键盘的按钮抬起来的时候触发

onkeypress

当键盘按下数字与字母键可以触发，除了功能键（上下左右、ctrl、shift、alt）

**鼠标滚轮事件**

onmousewheel IE/Chrom

滚动方向 event.wheelDelta

上：120

下：-120

DOMMouseScroll FF(只能用addEventListener)

滚动方向 event.detail

上：-3

下：3

**滚动条事件**

**window.onscroll**

在滚动条滚动的时候触发的函数

**window.onresize**

在窗口有改变的情况下触发的函数（改变的频率由浏览器决定）

滚动条距离(DOM)

document.body.scrollTop //Chrome

document.documentElement.scrollTop //其它

滚动条距离(BOM)高版本的浏览器兼容

window.pageXOffset //横向滚动条

window.pageYOffset //纵向滚动条

设置滚动条的距离

window.scrollTo(x,y)

x 横向距离

y 纵向距离

这两个参数必需同时出现

内容高度:被撑开的内容

scrollHeight

元素.scrollHeight 取元素被撑开的高度，如果没有被撑开，那就取父级的高度

body.scrollHeight 有兼容性问题，Chrome下取到的是可视区的高度,在FF下取到的是内容的高度

要取到Body的实际宽高用

document.body.offsetWidth

document.body.offsetHeight

**事件对象**

记录了在事件触发时的一些详细的信息

一个事件发生的时候，这个事件可能是鼠标事件也可能是键盘事件，所以说，事件对象是不同的

例如鼠标位置，键盘按键，触发事件的元素等

事件对象是事件处理函数中的第一个参数

event有兼容性问题

event chrom/ie

参数（如ev） FF

select()

选中表单里的内容

如果同时出现多个select()那只会选中最后一个

event.clientX/event.clientY

event.clientX x轴的位置，相对于可视区左边的距离

event.clientY y轴的位置，相对于可视区上边的距离

该坐标是以可视区左上角开始计算

**target（事件源对象）**

在鼠标事件触发时，指向触发该事件的目标元素;

它找到的是触发这个事件的最里层的元素，它会一层一层往里面找，直到找到最里层的元素，但是鼠标点击的时候，鼠标的位置必需在这个最里

层元素身上;

事件代理/事件委托;

**键盘事件对象**

当键盘事件发生的时候，会产生一个键盘事件相关信息的对象

event.keyCode

返回键盘按下的那个键对应的Unicode码

不光键盘有这几个属性，鼠标也有

event.ctrlKey

事件触发时，ctrl键是否被按下，true按下，false未按下

event.shiftKey

事件触发时，shift键是否被按下，true按下，false未按下

event.altKey

事件触发时，alt键是否被按下，true按下，false未按下

**事件绑定**（给一个对象添加事件）

1、 对象.on事件名称=函数

弊端：一个元素用on添加同一个事件，只能添加一个。如果添加多的话，后面的会把前面的覆盖

2、 addEventListener

对象.addEventListener(事件名称,事件处理函数,Boolean)

参数：

参数1：事件的名称，不带on（click、mouseover、mouseout）它是个字符串

参数2：事件处理的函数，它可以是一个匿名函数，还可以是命名函数

参数3：布尔值，这个事件是在哪个阶段执行

true 捕获阶段执行

false 冒泡阶段执行（默认为false）

**移除事件绑定**

1、移除用on添加的事件

对象.on事件名称=null;

2、移除用addEventListener添加的事件

对象.removeEventListener(事件名称,事件处理函数,Boolean);

问题

1、移除的事件必需与添加的事件的参数一一对应。必需一致才能移除掉

2、无法移除掉匿名函数添加的事件

**事件流**

事件流指事件在触发时的一个流程，就像for循环一样有一个执行的过程。

在事件触发时，会先执行一个阶段叫做“捕获阶段”，从最外层向触发该事件元素寻找的一个过程。最终找到触发该事件的元素，如果该元素有绑定事件处理函数则执行该函数。

在找到触发该事件的元素之后，会进行下一个阶段“冒泡阶段”，冒泡阶段是从触发事件的元素开始往外层寻找，直到最外层时结束整个事件流程。

不论是捕获阶段还是冒泡阶段，在寻找目标元素的过程和向外返回的过程中，所遇到每一个元素身上如果有相同事件的事件处理函数都会被调用。

事件源对象身上的事件，不会按先捕获后冒泡的顺序执行，而是会按书写的顺序去执行

**阻止事件冒泡**

1、用on添加的事件

**ev.cancelBubble=true;**

cancelBubble是事件对象下的一个属性，默认的值为false,代表允许事件冒泡

设为true代表不允许事件冒泡

给谁添加的阻止冒泡，它以上的所有就会全阻止

2、用addEventListener添加的事件

**ev.stopPropagation();**

它也是事件对象下的，不过它是个方法

**浏览器默认行为**

非js添加的功能，浏览器默认会有

例如：右键菜单，图片/文字/链接的拖拽

阻止浏览器默认行为

1、阻止on添加的事件

在事件函数里**return false**

2、阻止addEventListener添加的事件

在事件函数里用**ev.preventDefault()**

它是一个方法，在事件对象里