**DOM**

**Document**

**获取元素**

**用id获取元素**

**var d = document.getElementById('d');**

**用标签名获取元素**

**获取该元素名称下的所有元素,返回一个伪数组,或者是节点列表.**

**var lis =document.getElementsByTagName('li')**

**在某一个元素中,并不是在全局文档中利用标签获取元素**

**var ul = document.getElementById('ul')**

**var list = ul.document.getElementsByTagName('li')**

**当元素只有一个的时候,返回的也依然是一个列表,而不是一个元素.可以通过数组的下标找到该元素.**

**var div = document.getElementsByTagName('div');**

**console.log(div[0]);**

**用类名获取元素**

**var a = document.getElementsByClassName('a')**

**用name属性值获取元素**

**getElementsByName()方法可以获取相同名称的name元素,返回一个伪数组对象.**

**<input type="radio" name="sex" value="0">男**

**<input type="radio" name="sex" value="1">女**

**var sex = document.getElementsByName('sex');**

**console.log(sex);**

**用css选择器获取元素**

**querySelector()**

**只能找到元素列表的第一个元素**

**var a = document.querySelector('.a')**

**querySelectorAll()**

**可以全部找到**

**var c = document.querySelectorAll('#div>div a')**

**注意:该方法性能没有直接利用标签寻找高.**

**设置元素**

**设置元素属性**

**setAttribute()**

**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以逗号分隔**

**div[0].setAttribute('id','mydiv')**

**a[0].setAttribute('href','http：//www.baidu.com')**

**删除元素属性**

**removeAttribute()**

**括号中放入需要删除的属性名,用引号包裹.**

**div.removeAttribute('id')**

**获取元素中的其他信息**

**获取元素名**

**tagName**

**该属性只能获取,不能设置**

**获取元素节点里的内容**

**innerHTML**

**var div = document.getElementById('mydiv')**

**innerHTML属性除了可以获取标签内的内容外,还可以设置标签内的内容**

**获取元素节点中的文本**

**innerText**

**var div = document.getElementById('mydiv')**

**innerText属性获取的只能是文本节点,除了获取,也可以设置元素中的文本**

**获取元素的类名**

**className**

**以字符串的形式返回.同时可以设置新的class类名**

**div.className = 'd a c';**

**console.log(div.className);**

**var div = document.getElementById('mydiv')**

**console.log(div.className)**

**获取元素样式**

**style**

**可以获取内联样式的所有属性**

**值以字符串形式返回,同时也可以直接设置该属性,从而达到增加或更改样式,**

**注意: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**

**getAttrbute('')**

**要将属性名放在括号中记得要用引号括起来,返回值 就是字符串形式的属性**

**增加元素**

**创建元素节点**

**createTextNode('添加一段文本')**

**在括号中将要添加的字符串放入,最后插入到需要的元素中.**

**var text = document.createTextNode('添加一段文本')**

**var p = document.getElementById('p')**

**p.appendChild(text)**

**css样式赋予**

**style.cssText**

**它可以把css层叠样式表中的css样式直接放在该属性后,就不需要一条一条去赋予样式,可以一次性赋予样式.**

**div.style.cssText = 'width:100px;height:100px;background-Color:red'**

**在某元素前插入创建的新元素**

**insertBefore()**

**可以选择插入的位置**

**li.appendChild(t)**

**appendChild()**

**第一个参数是需要插入的元素,第二个参数是在谁之前插入,两者用逗号隔开**

**ul.insertBefore(li,jz)**

**删除和替换**

**替换**

**replaceChild(将要替换的新元素,被替换的旧元素)**

**document.body.replaceChild(myp,mydiv)**

**删除**

**removeChild(需要删除的元素)**

**document.body.removeChild(mydiv)**

**Element**

**查找节点**

**节点属性**

**mydiv.nodeValue**

**mydiv.nodeType**

**层次节点**

**父节点**

**子节点**

**兄弟节点**

**获取所有子节点**

**childNodes**

**拿到的是该元素下的所有节点,包含所有节点类型,不单单只是元素**

**console.log(myul.childNodes)**

**获取第一个和最后一个子节点**

**获取第一个**

**firstChild**

**console.log(myul.firstChild)**

**获取最后一个**

**lastChild**

**console.log(myul.lastChild)**

**获取父节点**

**parentNode**

**注意父节点只有一个**

**<body>**

**<ul id="myul"><li>111</li><li>222</li><li>333</li></ul>**

**<script type="text/javascript">**

**var myul = document.getElementById('myul')**

**console.log(myul.parentNode)**

**</script>**

**</body>**

**获取兄弟节点**

**前一个**

**previousSibling**

**console.log(mydiv.previousSibling)**

**后一个**

**nextElementSibling**

**console.log(mydiv.nextElementSibling)**

**元素的宽高节点**

**offsetHeigt和offsetwidth**

**该属性可以获取元素的占位宽高,他已包含了内边距和边框**

**console.log(mydiv.offsetWidth)console.log(mydiv.offsetHeigt)**

**clientHeigt和clientwidth**

**不包含边框**

**console.log(mydiv.clientWidth)console.log(mydiv.clientHeiht)**

**子元素与父元素的距离**

**offsetLeft和offsetTop**

**距离body左边界和上边界的距离**

**注意:如果该子元素使用了定位属性,则参照的就不在是body而是距离它最近的父级元素**

**<body>**

**<div id="baba">**

**<div id="erzi">**

**</div>**

**</div>**

**<script type="text/javascript">**

**var erzi = document.getElementById('erzi')**

**console.log(erzi.offsetLeft)**

**console.log(erzi.offsetTop)**

**</script>**

**</body>**

**DOM事件**

**鼠标事件**

**点击**

**点击事件**

**mydiv.onclick = function(){触发事件}**

**双击事件**

**mydiv.onmousedown = function(){触发事件}**

**鼠标抬起**

**mydiv.onmouseup = function(){触发事件}**

**多次触发**

**移动**

**document.onmousemove = function(){触发事件}**

**移入**

**mydiv.onmouseover = function(){触发事件}**

**移出**

**mydiv.onmouseout = function(){触发事件}**

**单次触发**

**移出**

**mydiv.onmouseleave = function(){触发事件}**

**移入**

**mydiv.onmouseenter = function(){触发事件}**

**键盘事件**

**按下**

**document.onkeydown = function(e){触发事件}**

**抬起**

**document.onkeyup = function(e){触发事件}**

**Event事件对象**

**概念:Evevt对象代表事件的状态**

**执行时间**

**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！**

**接收方式**

**通过事件触发时的函数,以参数的形式传递该进行函数内,不用靠调用传参,也就是说event对象是事件触发时调用函数的第一个参数**

**document.onclick = function(){触发事件}**

**常用属性**

**鼠标坐标值**

**e.clientX**

**e.clientY**

**键盘的值**

**e.keyCode**

**兼容性**

**document.onclick = function(e){**

**// 非ie 低版本ie**

**var ev = e || window.event;**

**console.log(e.clientX)**

**}**

**对象事件**

**页面(元素)加载完成后执行事件**

**window.onload**

**等待页面加载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**

**window.onload = function(){执行事件}**

**窗口变化大小时触发事件**

**window.onresize**

**可以监听到窗口大小的变化,只要发生变化就会触发该事件**

**window.onresize = function(){触发事件}**

**表单事件**

**获取焦点**

**in1.onfocus = function(){触发事件}**

**失去焦点**

**in1.onblur = function(){触发事件}**

**用户输入触发**

**in1.oninput = function(){触发事件}**

**内容改变触发**

**in1.onchange = function(){事件触发}**

**事件绑定**

**概念**

**希望运用js让用户操作DOM元素,先要让DOM元素处理一个事件函数,将这个函数绑定在该DOM元素上.**

**绑定方法**

**<button id="btn1" onclick="a()">按钮</button>**

**<button id="btn2">按钮</button>**

**<script type="text/javascript">**

**var btn1 = document.getElementById('btn1')**

**var btn2 = document.getElementById('btn2')**

**function a(){**

**alert('按钮1')**

**}**

**btn2.onclick = function(){**

**alert('按钮2')**

**}**

**</script>**

**解除绑定的方法**

**<button id="btn2">按钮</button>**

**<script type="text/javascript">**

**var btn2 = document.getElementById('btn2')**

**btn2.onclick = function(){**

**alert('按钮2')**

**}**

**btn2.onclick = null ;**

**</script>**

**事件监听**

**方法**

**对某一些对象进行事件触发的监听,需要增加监听器addEventListener(),该方法有三个参数(监听事件,方法,冒泡),冒泡默认false,监听事件不加on. 但是监听有兼容性问题,IE的监听需要attachEvent('onclick')第一个参数为监听的事件,需要加on,第二个参数是触发的事件函数**

**img1.addEventListener('dblclick',a,false)**

**function a(){**

**alert('鼠标按下')**

**}**

**ie监听的方法**

**img1.attachEvent('onclick',c)**

**封装监听**

**function addE(target,type,fn){**

**if(target,addEventListener){**

**target.addEventListener(type,fn);**

**}else {**

**target.attachEvent('on'+type,'fn');**

**}**

**}**

**addE(mydiv,'click',a);**

**封装解绑**

**function removeE(target,type,fn){**

**if(target,removeEventListener){**

**target.removeEventListener(type,fn);**

**}else {**

**target.datachEvent('on'+type,'fn');**

**}**

**}**