DOM

1. Dom节点
   1. 整个文档是一个文档节点
   2. 每个html元素是元素节点
   3. html元素内的文本是文本节点(空格也是)
   4. 每个html属性都是属性节点
   5. 注释也是节点,叫注释节点
2. 获取元素
3. 用id获取元素：
4. getElementById()的方法,接受一个参数:获取元素的id,如果找到相应元素,则返回该元素,否则返回null.

语法：var d = document.getElementById('d');

1. 用标签名获取元素：

getElementsByTagName()可以获取该元素名称下的所有元素,返回一个伪数组,或者说一个节点列表.

语法：var lis =document.getElementsByTagName('li');

1. 用类名获取元素：

语法：var a = document.getElementsByClassName('a');

1. 用标签名获取元素：

querySelector()和querySelectorAll()可以依靠选择器找到元素,但是前者只能找到元素列表的第一个元素,而后者可以全部找到.注意,该方法性能没有直接利用标签寻找高.

语法：var ul = document.getElementById('ul')

var x = document.querySelector('#div>div a');

var lis = ul.querySelectorAll('li');

1. 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.

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

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

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

1. 获取元素中的其他信息
   1. 获取元素名

想要拿到该元素的元素名,也就是标签名则需要使用taName属性.该属性只能获取,不能设置

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

console.log(div.tagName);

* 1. 获取元素节点里的内容

当获取了元素之后,如果我们需要拿到元素中的内容(所有东西),则需要用另一种方法得到,元素里的内容可能包括:文本或元素.需要用innerHTML属性.

var mydiv = document.getElementById('mydiv');

console.log(div.innerHTML);

* 1. 获取元素节点中的文本

利用innerText属性获取的只能是文本节点,不能是其他,当然innerText属性除了获取,也可以设置元素中的文本.

var div =document.getElementById('div');

console.log(div.innerText);

div.innerText = 'hello';

* 1. 获取元素的类名

利用className属性获取元素的类名,以字符串的形式返回.同时可以设置新的class类名,需要注意的是,返回值是一个字符串,

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

console.log(div.className);

如果更换其中一个类.则需要注意该字符串中其他类是否一起设置

div.className= 'd a c';

console.log(div.className);

* 1. 获取元素样式.

style属性可以获取元素内联样式的所有属性,当然如果继续在style 中

找属性名如:backgroundColor,就可以得到该属性的值,以字符串的形式返回.同时也可以设置该属性,从而达到增加或更改样式,需要注意的是,以js形式增加的样式的优先级大于css的优先级.

<div style="background-color: blue"></div>

<script>

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

//js拿到的和获取的是内联

console.log(div[0].style.backgroundColor);

</script>

* 1. 获取元素属性

getAttribute('')可以获取元素的某个属性，要将属性名称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值。

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

var a = document.getElementsByTagName('a');

//返回的是一个数组，数组下标值；

//凡是有get的一定有set(设置)；

console.log(div.getAttribute('data-lj'));

onsole.log(a[0].getAttribute('href'));

* 1. 设置元素属性

setAttribute()可以设置元素某个属性,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.

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

var a = document.getElementsByTagName('a');

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

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

* 1. 删除元素属性

使用removeAttribute(),可以删除某个元素的某个属性,括号中放入需要删除的属性名,用引号包裹.

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

div.removeAttribute('id');

1. 增加元素
2. 创建元素节点

利用js创建一个元素的方法是,先在文档中创建一个标签document.createElement()括号中写标签名,当然要用字符串形式.创建之后不是就存在了,而是需要将这个已经创建的元素放到你想放到的元素(父级元素)中去.

//用一个变量承接在文档中''创建'的一个元素

var div = document.createElement('div');

//将已经创建的元素放在想放的位置

document.body.appendChild(div);

1. 创建文本节点

利用js创建元素的文本节点,先在文档中创建文本document.createTextNode('一段文字')在括号中将要创建的字符串放入最后插入到需要的元素中.

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

var text=document.createTextNode('本无争锋欲.');

p.appendChild(text);

1. css样式赋予

insertBefore()这个函数和appendChild()用法基本一致,都是向父级中插入一个子元素,但区别是insertBefore()是可以选择插入的位置,它需要插入到某一个子元素之前的位置,因此需要哪个子元素,insertBefore()有两个参数,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,两个参数用逗号分隔.

<ul id="ul">

<li id="pg">苹果</li>

<li id="jz">橘子</li>

<li>香蕉</li>

</ul>

<script>

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

//获取苹果

var pg = document.getElementById('pg');

//获取桔子

var li =document.createElement('li');

//在父级中插入一个元素(插入元素,在谁前面)

ul.insertBefore(li,jz);

var t = document.createTextNode('火龙果');

li.appendChild(t);

</script>

1. 删除和替换元素
   * + 1. 元素的替换

元素的替换是在父元素中一个子元素需要被另一个新的子元素所替代，使用replaceChild()方法,该方法有两个参数,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.

var mydiv=document.getElementById('mydiv');

var myp=document.createElement('p');

//在父元素中替换子元素,第一个是新的元素,第二个是旧的.

document.body.replaceChild(myp,mydiv);

* + - 1. 元素的删除

元素的删除是在父元素的其中一个子元素需要删除,使用removeChild()方法,将需要删除的元素放入括号中.

var mydiv=document.getElementById('mydiv');

document.body.removeChild(mydiv);

1. 查找节点（Nodes）
2. 节点的属性

var mydiv=document.getElementById('mydiv');

console.log(document.body.nodeName);

console.log(document.body.nodeValue);

console.log(document.body.nodeType);.

1. 层次节点

节点可以分为：父节点与子节点、兄弟节点、当我们知道其中之一的时候，可以用一些方法找个另一个

1. 获取所有子节点

使用childNodes属性拿到的是该元素的所有节点，包含所有节点类型，不单单只是元素。

1. 获取第一个和最后一个子节点

用firstChild和lastChild可以拿到该父元素下的第一个和最后一个子节点，注意不一定是元素节点。

1. 获取父节点

使用parentNode属性可以拿到该元素的父节点，注意父节点只有一个。

1. 获取兄弟节点

使用previousSibling和nextSibling可以获得该元素的前一个和后一个兄弟节点，只能获取一个。

1. 元素的位置属性
   1. offsetWidth和offsetHeight

需要获取一个元素的宽度和高度，使用style是无法获取的，所以我们选择使用offsetWidth和offsetHeight属性，该属性可以获取元素的占位宽高，它也包含了内边距和外边距。

var mydiv=document.getElementById('mydiv')

console.log(mydiv.style);//这种方法只能拿到内敛样式

console.log(mydiv.offsetWidth);

* 1. clientWidth和clientHeight

clientWidth和clientHeight属性也可以获取元素的宽高但与offsetWidth不同的是不包含边框。

1. 子元素与父元素的距离

offsetLeft和offsetTop是距离body的左边界和上边界的距离，但如果该子元素使用了定位属性，则offsetLeft和offsetTop参照的就不再是body而是距离它最近的父级元素。

<div id="dd">

<div id="ez"></div>

</div>

<script>

var ez=document.getElementById('ez');

console.log(ez.offsetLeft);

console.log(ez.offsetTop);

</script>

DOM事件

1. 鼠标和键盘事件

鼠标事件

单一作用于某个元素的事件

1. 点击事件

mydiv.onclick = function(){

alert('点击事件');

}

1. 双击事件

mydiv.ondblclick = function(){

alert('双击事件');

}

1. 鼠标按下事件

mydiv.onmousedown = function(){

mydiv.innerHTML = '按下事件';

}

1. 鼠标抬起事件

// mydiv.onmouseup = function(){

// mydiv.innerHTML = '';

// }

1. 鼠标移动

//需要注意只要鼠标移动就触发

//document.onmousemove =function(){

// console.log('移动了');

// }

// mydiv.onmousemove = function(){

// console.log('div');

// }

可能鼠标移入移出情况

第一套移入移出会多次触发

//鼠标移入

// mydiv.onmouseover =function(){

// console.log('移入了');

// }

// mydiv.onmouseover =function(){

// console.log('红')

// }

//会触发冒泡(从内逐级向外依次发);

//会触发多次,取决于嵌套和鼠标位置

// box.onmouseover =function(){

// console.log('蓝')

// }

//鼠标移出

// mydiv.onmouseout =function(){

// console.log('离开红')

// }

// //会触发冒泡(从内逐级向外依次触发);

// //会触发多次,取决于嵌套和鼠标位置

// box.onmouseout =function(){

// console.log('离开蓝')

// }

第二套移入移出不会会多次触发

/鼠标移入

mydiv.onmouseenter =function(){

console.log('红')

}

//不会进行多次触发

box.onmouseenter =function(){

console.log('蓝')

}

//鼠标移出

mydiv.onmouseleave =function(){

console.log('离开红')

}

//不会进行多次触发

box.onmouseleave =function(){

console.log('离开蓝')

}

键盘事件

键盘的按钮按下时会触发onkeydown事件,在函数中的e参数代表是哪一个事件触发的.

document.onkeydown =function(e){

alert('按下');

}

键盘的抬起会触发onkeyup事件,它与onkeydown用法一样.

document.onkeyup =function(e){

alert('抬起了');

}

1. Event事件对象
2. Event事件对象的概念

Event对象代表事件的状态,比如事件在其中发生的元素,键盘按键的状态,鼠标的位置,鼠标按钮的状态等等.

1. Event事件对象产生的时间

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！

1. Event事件对象接受方式

通过事件触发时的函数,以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对象是事件触发时调用函数的第一个参数.

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

1. event兼容性

event对象根据不同浏览器他的获取是不同的方法,以下是针对获取event对象的兼容性写法.

document.onclick =function(e){

// 非ie 低版本ie

var ev =e || window.event;

console.log(e.clientX);

console.log(e.clientY);

}

1. 对象事件
2. 页面（元素）加载完成后执行事件

window.onload是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。

//页面加载完成之后执行函数内的代码段

window.onload =function(){

var mydiv=document.getElementById('mydiv');

mydiv.onclick = function(){

alert(mydiv.tagName);

}

}

1. 窗口大小化时触发事件

window.onresize是可以监听到窗口大小的改变，只要发生变化就会触发该事件。

window.onresize =function(){

console.log(1);

}

1. 表单事件
2. 获取焦点

in1.onfocus = function(){

in1.style.backgroundColor = 'blue';

}

1. 失去焦点

in1.onblur = function(){

in1.style.backgroundColor = 'red';

}

1. 用户输入触发

in1.onchage =function(){

in1.style.backgroundColor = 'blue';

}

1. 内容改变时触发

in1.input =function(){

in1.style.backgroundColor = 'red';

}

1. 事件绑定
2. 事件绑定概念

事件绑定的方法可以有标签内和js脚本内的方法。

<button id="btn1" onclick="a()">按钮1</button>

<button id="btn2">按钮2</button>

<script>

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

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

function a(){

alert('按钮1');

}

btn2.onclick =function(){

alert('按钮2');

}

1. 事件解除绑定的方法

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

btn2.onclick =function(){

alert('按钮2');

}

//解除绑定

btn2.onclick = null;

1. 事件监听
2. 事件监听的方法

事件监听是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，监听事件不用加on，但是监听有兼容性问题，IE的监听需要attachEvent('onclick',c)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

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

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

// 增加监听 监听事件，方法，冒泡

//btn2.addEventListener('click',a,false);

btn1.addEventListener('mousedown',a,false);

btn1.addEventListener('mouseup',b,false);

btn2.addEventListener('mousedown',a,false);

btn2.addEventListener('mouseup',c,false);

function a(){

alert('鼠标按下');

}

function b(){

alert('鼠标抬起1');

}

function c(){

alert('鼠标抬起2');

}

//ie监听得方法

btn1.attachEvent('onclick',c);

1. 解绑和封装

var mydiv =document.getElementById('mydiv');

//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');

}

}