DOM

DOM的概念

DOM文档对象模型,DOM对象不仅仅是一个普通的内置对象,他还是一个巨大的API的核心对象,他将文档的内容呈现在js面前,并赋予了js操作文档的能力.

DOM和JavaScript的关系

一个web页面是一个文档,这文档可以在浏览器窗口或者作为html源代码显示出来.但上述两种情况都是同一个文档,DOM提供了对同一份文档的另一种表现,存储和操作的方式.DOM能够使用JavaScript脚本语言进行修改.

DOM节点

加载HTML页面是,web浏览器生成一个树形结构,用来表示页面的内部结构.DOM将这种结构理解为节点组成.根据w3c的html DOM标准,html文档找那个所有的内容都是节点.

> 整个文档是一个文档节点

> 每个HTML元素是元素节点

> HTML元素内的文本是文本节点

> 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树体现着html页面的层级结构,而DOM树有DOM文档树和DOM元素树两种,DOM元素树包含的只有元素节点,而DOM文档树则包括DOM文档中的所有内容.

获取元素

用Id获取元素

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.

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

6.2.2 用标签名获取元素

getElementsByTagName()可以获取该元素下所有的元素,返回一个伪数组,或者是节点列表.

在某一个元素中,并不是在全局文档中利用标签获取元素,这样更加精准.

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

var list = ul.docoment.getElementsByTagName('li');

console.log(list);

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,可以通过数组的下表找到该元素.

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

console.log(div[0]);

用类名获取元素

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

console.log(a);

用选择器获取元素

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.

var ul = document.getElementById('ul')

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

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

console.log(lis);

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);

获取和设置元素中的其他信息

获取元素名

当我们使用id获取元素的时候,元素的名称会和整个元素一起显示出来.如果想要拿到该元素的元素名,也就是标签名则需要用tagName属性.

tagName属性,该属性只能获取,不能设置.

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

console.log(div.tagName);//DIV

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 div = document.getElementById('mydiv');

console.log(div.innerHTML);//获取元素内容

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

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('mydiv');

console.log(div.innerText);

div.innerText = 'hello';

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,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

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

console.log(div.className);

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中找到属性名如: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优先级.

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

<script>

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

//js拿到和获取的是内联

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

获取元素的属性

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');

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

console.log(a[0].getAttribute('href'))

有get 有set

设置元素属性

setAttribute()可以设置元素的某个属性,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');

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,用引号包括.

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

div.removeAttribute('id');

增加元素

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

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创放入,最后插入到需要的元素中.

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

var text = document.createTextNode('一段文字');

p.appendChild(text);

css样式赋予

style.cssText是一个css的样式集合,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了,可以一次性赋予.

div.style.cssText = 'width:100px;height:100px;background-color:blue;';

在某元素前插入创建新元素

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,他需要插入到某一个元素之前的位置,因此需要那个子元素,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.

删除和替换元素

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代替,使用replaceChlid()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,第二个是旧的元素

document.body.replaceChild(myp,mydiv);

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.nodevalue);

元素查找

节点是dom树上的每一个节点,其中包括:元素、文本、属性、注释等等.常用的有三个,元素,文本和属性

节点的属性

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

console.log(mydiv.nodeName);

console.log(mydiv.nodeValue);

console.log(mydiv.type)

层次节点

节点可以分为:父节点于子节点,兄弟节点,当我们知道其中之一时,可以用一些方法找到另一个

获取所有子节点

使用ChildNodes属性拿到的是该元素下的所有节点,包括所有节点类型,不单单是属性.

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

<script>

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

//childNodes是所有节点

console.log(myul.ChildNodes);

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.

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

<script>

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

console.log(myul.firstChild);

console.log(myul.lastChild);

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.

<ul id="myul"></ul>

<script>

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

//childNodes是所有节点

console.log(myul.parentNode);

</script>

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.

<h2></h2><div id="mydiv">div</div><p>p</p>

<script>

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

console.log(mydiv.previousSibling);

console.log(mydiv.nextSibling);

</script>

元素的宽高属性

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站位宽高,它也包括了内边距和边框.

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

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

console.log(mydiv.offsetWidth);

clientWidth和clientHeight

clientWidth和clientHeight属性也是可以获取元素的宽高,但与offsetWidth不同的是不包含边框.

console.log(mydiv.clientWidth)

子元素与父元素的距离

offsetLeftheoffsetTop是距离baby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二是距离他最近的父级元素.