DOM节点

Dom是表示和处理一个HTML 或 XML 文档的常用办法。

一个html结构会被浏览器解析成一个dom树，js是可以操作这一棵dom树的语言，通过js来获取dom树的一个节点，进行增加或删除等操作，从而达到修改html 页面的效果。

根元素

<html>

元素

<head>

元素

<body>

元素

<title>

元素

<div>

元素

<a>

属性

href

元素

<h1>

一、获取节点的方法

1、使用document.getElementById获取dom元素

虽然id叫做唯一标识符，但是由于html不是java等编译型语言检查代码。因此，当页面中出现两个相同的id时，document.getElementById只能去一个节点，此时只会选择第一个查到的节点。

2、使用document.getElementsByTagName获取dom元素

3、使用docyment.getElementsByName获取dom元素

4、使用docyment.getElementsByClassName获取dom元素【需要做兼容性】

function getClassName(node,classname){

if(node.getElementsByClassName){

return node.getElementsByClassName(classname);

}else{

var elems = node.getElementsByTagName("\*");

var elements = [];

for(var i=0;i<elems.length;i++){

if(elems.className.indexOf(classname) != -1){

elements.push(elems[i]);

}

}

return elements;

}

}

二、节点的类型和属性

DOM1级定义了一个Node接口，该接口将右DOM中的所有节点类型实现。该Node接口在JavaScript中是作为Node类型实现的；除了IE之外，其他所有浏览器中都可以访问到这个类型。JavaScript中的所有节点类型都继承自Node类型，共享相同的基本属性和方法。

**1、nodeType属性，表明节点的类型。**

Node.ELEMENT\_NODE(1) //元素节点

Node.ATTRIBUTE\_NODE(2) //属性节点

Node.TEXT\_NODE(3) //文本节点

…

Node.DOCUMENT\_NODE(9) //document节点

**2、节点关系**

每个节点都具有childNodes属性，其中保存NodeList对象，是一种类数组对象，保存有序的节点。NodeList对象是基于DOM结构动态执行查询的结果，并不是Array的实例。

类数组转换为数组【兼容性】

function convertToArray(nodes){

var array = null;

// try-catch 块捕获错误，然后手动创建数组

try{

array = Array.prototype.slice.call(nodes,0);

}catch(ex){

array = new Array();

for(var i=0;i<nodes.length;i++){

array.push(nodes[i]);

}

return array;

}

}

**3、操作节点**

appendChild()、 insertBefore()、removeChild()、replaceChild()、cloneNode(true)//深复制

前四个方法，需取得父节点parentNode。父节点具有firstNode、lastNode属性；

其次，子节点之间具有兄弟关系previousSibling、nextSibling。

4、自定义数据属性

添加前缀data-，目的是为元素提供与渲染无关的信息或者提供语义信息。这些属性可任意添加、随意命名，只要以data-开头即可。

添加了自定义属性后，可以通过元素的dataset属性来访问自定义属性的值。dataset属性是DOMStringMap的一个实例，即键值对。该映射中，每个data-name 形式的属性都会有一个对应的属性，只是属性名没有data- 前缀。例如：

var div = document.getElementById(“mydiv”);

// 取得自定义属性的值<div id=”mydiv” data-appId=”123”data-myname=”lrn”>

var appId = div.dataset.appId;

var myname = div.dataset.myname;

**4、插入标记**

innerHTML返回调用元素的(所有子节点，包括元素、注释、文本节点)

outerHTML调用元素的所有html标签

innerText、outerText

**5、元素大小**

偏移量：元素在屏幕上占用的可见空间，包括内边距、滚动条、边框，不包括外边距。

|  |  |
| --- | --- |
| offsetHeight | 元素在垂直方向上占用的空间大小。包括元素的高度、可见的水平滚动条的高度、上边框高度和下边框高度。 |
| offsetWidth | 元素在水平方向上占用的空间大小。包括元素的宽度、可见的垂直滚动条的宽度、左边框的宽度和右边框宽度。 |
| offsetLeft | 元素的左外边框至包含元素左内边框的像素距离。 |
| offsetTop | 元素的上外边框至包含元素上内边框的像素距离。 |

cont

content

offsetTop

offsetLeft offsetHeight

(含水平滚动条高度)

offsetWidth(含垂直滚动条宽度)

客户区大小：元素内容及其内边距所占距的大小。

clientHeight: 元素内部的空间大小（content+padding），滚动条占用的空间不计算在内。

clientWidth: 元素内部的空间大小（content+padding），滚动条占用的空间不计算在内。

content

clientHeight

clientWidth

【兼容性】

function getViewPort(){

if(document.compatMode == "BackCompat"){ //标准兼容模式关闭

return {

width: document.body.clientWidth,

height:document.body.clientHeight

};

}else{

return {

width: document.documentElement.clientWidth,

height:document.documentElement.clientHeight

};

}

}

滚动大小：包含滚动内容的元素的大小。

|  |  |
| --- | --- |
| scrollHeight | 没有滚动条的条件下，元素内容的高度 |
| scrollWidth | 没有滚动条的条件下，元素内容的宽度 |
| scrollLeft | 被隐藏在内容区域左侧的像素数，该属性可改变元素的滚动位置 |
| scrollTop | 被隐藏在内容区域上方的像素数，该属性可改变元素的滚动位置 |

scrollWidth

隐藏的内容 scrollTop

scrollHeight

内容区+内边框

滚动条

滚动条

scrollLeft

区别：

对于不包含滚动条的页面，scrollWidth、scrollHeight与clientWidth、clientHeight关系：

Firefox认为相等，Opera、Safari、Chrome等认为不同，前者是视窗大小，后者是文档大小。IE浏览器在标准模式下，认为不相等，前者是文档区域大小，后者是视窗大小。

【兼容性】当确定文档总高度（包含基于视窗的最小高度）

var docHeight = Math.max(document.documentElement.scrollHeight, document.documentElement.clientHeight);

var docWidth = Math.max(document.documentElement.scrollWidth, document.documentElement.clientWidth)