DOM document object model

文档对象模型

节点层次

<html>

<head>

<title> JS DOM</title>

</head>

<body>

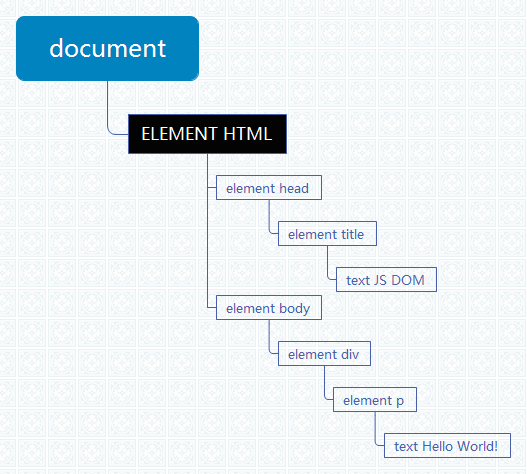
<div>

<p>Hello World!</p>

</div>

</body>

</html>



文档节点是每个文档的根节点，在上例中，文档节点只有一个子节点，即<html>元素，称之为文档元素，文档元素是文档的最外层元素，文档中的其他所有元素都包含在文档元素中，每个该当只能有一个该当元素，在HTML页面中，文档元素始终都是<html>，在XML中没有预定义的元素，因此任何元素都可能成为文档元素

Node类型：定义了一个Node接口，该接口将由DOM的所有节点类型实现，JS中的所有节点类型都继承自Node类型，因此所有节点类型都共享着相同的基本属性和方法。每个节点都有一个nodeType属性，用于表明节点的类型，节点类型由Node类型中定义的12个数值常量来表示，任何节点类型必是其一

nodeName/nodeValue属性

nodeName

nodeValue

节点关系

文档中的所有节点之间都存在着关系，文档树比喻成家谱，在HTML中，<html>是<body>的父元素，<head>是<body>的兄弟元素

每个节点都有一个childNodes属性，其中保存着一个NodeList对象，NodeList是一种类数组对象，用于保存一组有序的节点，可以通过位置来访问这些节点，它也有length属性，但它不是Array的实例，可以通过方括号，也可以使用item()方法

var 1child = someNode.childNodes[0];

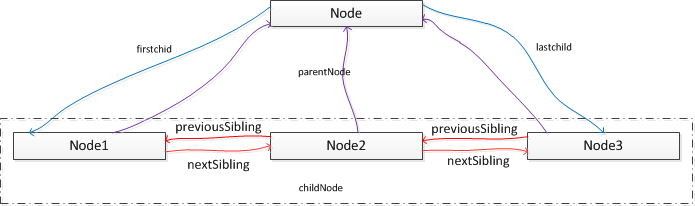
var 2child = someNode.childNodes.item(1);

var len = someNode.childNode.length;

跟arguments对象使用Array.prototype.slice()方法可以将其转换成数组，也可以采用同样的方法将NodeList对象转换成数组

var arrNodes = Array.prototype.slice.call(someNode.childNodes,0);

每个节点都有一个parentNode属性，指向文档树中的父节点，包含在childNodes列表中的所有节点都具有相同的父节点，每个节点相互之间都是兄弟节点，通过使用列表中每个节点的previousSibling、属性，可以访问同一个列表中的其他节点，列表中第一个节点的previousSibling属性值为null，列表中最后一个节点的nextSibling属性值也为null



操作节点

Document类型

JS通过Document类型表示文档，在浏览器中，Document对象HTMLDocument的一个实例，表示整个HTML页面，而且Document对象是window对象的一个属性，Document节点具有下列特征

nodeType = 9

nodeName = #document

nodeValue = null

parentNode = null

ownerDocument = null

文档的子节点

DOM标准规定Document节点的子节点可以是

documenttype,element.processingin-structior,comment,还有两个内置的访问其他子节点的快捷方式，

Documentelement属性，始终指向HTML页面中的<html>元素

childNodes列表访问文档元素

var html = document.documentElement; //取得对<html>的引用

var html = document.childNodes[0];

var html = document.firstChild;

body属性直接指向<body>元素

var body = document.body;

文档信息：作为HTMLDocument的一个实例，document对象还有一引动标准的Document对象所没有的属性，这些属性提供了document对象所表现的网页的一些信息

http://www.pschen.cn/Admin/

var title1 = document.title;

var url = document.URL; // http://www.pschen.cn/Admin/

var domain = document.domain; // www.pschen.cn

var ref = document.referrer;

查找元素：最常见的DOM应用，取得特定的某个或某组元素的引用，再执行相关的操作

Document类型提供了两个方法：

getElementById()：接收一个参数，要取得的元素的ID，找到返回该元素，找不到返回null。

这里的ID必须与页面中元素的id严格匹配，包括大小写。

<div id=”div1”>Hello World</div>

var div = document.getElementById(“div1”); //取得<div>元素的引用

getElementsByTagName()：接收一个参数，即要取得元素的标签名，而返回的是包含0个或多个元素的NodeList，在HTML文档中，这个方法返回一个HMTLCollection对象，作为一个动态集合，该对象与NodeList类似

var img = document.getElementByTagName(“img”); //取得页面中所有的<img>元素，并返回一个HMTLCollection，把HMTLCollection对象保存在img变量中，可以使用方括号、item()方法来访问HMTLCollection对象中的项，也有length属性

console.log(img.length); //输出图像的数量

console.log(img[0].src); //输出第一个图像元素的src特性

console.log(img.item(0).src); //输出第一个图像元素的src特性

HMTLCollection对象还有一个方法namedItem()，通过元素的name特性取得集合中的项

<img src=”ssss.jpg” name=”myimg” />

var img1 = img.nameditem(“myimg”); //直接访问到name为myimg的图像

var img2 = img[“myimg”]; 使用方括号也可以访问，其实就是在后台调用namedItem()、item(0)

HMTLDocument类型才有的方法：getElementsByName()：返回带有给定name特性的所有元素。最常使用它取得单选按钮，单选按钮必须具有相同的name

Element类型：用于表现XML/HTML元素，提供了对元素标签名、子节点及特性的访问，具有以下特征：

nodeType = 1

nodeName =元素的标签名

nodeValue = null

parentNode = Document/Element

ownerDocument = null

HTML元素：HTMLElement类型直接继承Element并添加一些属性

id：元素在文档中的唯一标识符

title：有关元素的附加说明

lang：元素内容的语言代码

dir：语言的方向，ltr left-to-right rtl right-to-left

className：与元素的class特性对应，即为元素指定的CSS类。

取得元素的特性：特性的名称是不区分大小写的

操作特性DOM方法有三个：

getAttribute()：获得元素的特性

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

console.log(div.getAttribte(“id”);

console.log(div.getAttribte(“class”);

console.log(div.getAttribte(“title”);

setAttribute()：设置元素的特性，接收两个参数：要设置特性名和值

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

div. setAttribute(“id”,”div1”);

div. setAttribute(“title”,”test”);

div. setAttribute(“class”,”fr”);

removeAttribute()：删除元素的特性

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

div. setAttribute(“class”);

attributes属性：包含一个NamedNodeMap，是一个动态的集合，元素的每一个特性都由一个Attr节点表示，每个节点都保存在NameNodeMap对象中。

创建元素：document.createElement() 只接收一个参数，即要创建元素的标签名，这个标签名在HTML文档中不区分大小写，在XML文档中，则是区分大小写

var div = document.createElement(“div”);

div.id = “div2”;

div.className = “box”;

元素的子节点：元素可以有任意数目的子节点和后代节点，元素的childNodes属性中包含了它的所有子节点，这些子节点可能是元素、文本节点，注释，处理指令

<ul id=”mylist”>

<li>item 1</li>

<li>item 2</li>

<li>item 3</li>

</ul>

IE解析那么<ul>元素有3个子节点，分别是<li>元素。其他浏览器<ul>元素都会有7个元素，包括3个<li>和4个文本节点（表示<li>元素之间的空白符）如下所写所有的浏览器都会一样

<ul id=”mylist”><li>item 1</li><li>item 2</li><li>item 3</li></ul>

遍历子节点：

for(var i =0,len = element.childNodes.length;I < len;i++){

if(element.childNodes[i].nodeType == 1){ //检查nodeType属性

}

}

通过某个特定的标签名取得子节点或后代节点，元素也支持getElementsByName()

var ul = document.getElementById(“mylist”);

var items = ul. getElementsByName(“li”);

注意：这里的ul的后代中只包含直接子元素，如果它包含更多层次的后代元素，那么各个层次中包含的li元素都会返回

Text类型

文本节点由Text类型表示，包含的是可以照字面解释的纯文本内容，可以包含转义后的HTML字符，但不能包含HMTL代码

nodeType = 3

nodeName =”#text”

nodeValue = 节点所包含的文本

parentNode = Element

不支持（没有）子节点

<div></div> //没有文本节点

<div> </div> //有空格，有一个文本节点

<div>Hello World!</div> //有内容，有一个文本节点

访问文本节点

var texetnode = div.firstChild

div.firstChild.nodeValue = “pschen”; //修改文本节点，输出的结果会被转义

Comment类型

注释

nodeType = 8

nodeName =”#comment”

nodeValue = 注释的内容

parentNode = Document/Element

不支持（没有）子节点

Comment类型和Text类型继承自相同的基类，它拥有队splitText()之处的所有字符串操作方法

CDASection类型

只针对基于XML的文档，表示的是CDATA区域，继承自Text类型

nodeType = 4

nodeName =”#cdata-section”

nodeValue = CDATA区域中的内容

parentNode = Document/Element

不支持（没有）子节点

Attr类型

元素的特性在DOM中以Attr类型来表示，所有的浏览器中都可以访问Attr类型的构造函数和原型，从技术角度讲，特性就是存在于元素的attributes属性中的节点

nodeType = 11

nodeName =特性的名称

nodeValue = 特性的值

parentNode = null

HTML不支持（没有）子节点

XML中子节点可以是Text/EntityReference

DOM操作技术