[DOM中的NodeList与HTMLCollection](http://www.cnblogs.com/summerTea/p/4943533.html)

　　最近在看《Javascript高级程序设计》的时候，看到了这样一句话：“理解NodeList和HTMLCollection，是从整体上透彻理解DOM的关键所在。”，所以觉得应该写一篇关于NodeList和HTMLCollection的博客来好好了解和总结下这方面的知识点。

**NodeList**

　　NodeList是一个**节点的集合**(即可以包含元素和其他节点)，在DOM中，节点的类型总共有12种，通过判断节点的nodeType来判断节点的类型。

　　我们可以通过Node.childNodes和document.querySelectAll() (返回NodeList的接口有很多，这里不一一列举，下同)来获取到一个NodeList对象。

　　NodeList对象有个length属性和item()方法，length表示所获得的NodeList对象的节点个数，这里还是要强调的是节点，而item()可以传入一个索引来访问Nodelist中相应索引的元素。

1 <body>

2 <div id="node">

3 文本节点

4 <!-- 注释节点 -->

5 <span>node1</span>

6 <span>node2</span>

7 <span>node3</span>

8 </div>

9 </body>

10 <script>

11 var node = document.getElementById('node'),

12 nodeLists = node.childNodes;

13 console.log(nodeLists.length) // 输出为9

14 </script>

　　上面的HTML代码中，“文本节点”和父节点子节点的空格（连着的文本）算做一个文本节点，然后是一个注释节点和注释节点和元素节点之间的空格（换行会产生空格，空格算做文本节点）的文本节点，紧接着的是一个元素节点和元素节点之间的换行的文本节点，三个元素节点和元素节点间的两个文本节点，最后是最后的元素节点和父元素之间的空格产生的文本节点，总共是9个节点。

　　NodeList对象的一大特点是它返回的内容是动态的（live）,也就是说我们上面代码获取nodeLists是类似于“指针”的东西，所以在下面代码中我们在获取了nodeLists之后再向node中插入一个创建的span标签后，发现获取到了nodeLists.length变为10了，但是querySelectorAll这个接口返回的nodeList对象比较特殊，它是个静态（static）的对象。而且是**元素的集合。**

1 <body>

2 <div id="node">

3 文本节点

4 <!-- 注释节点 -->

5 <span>node1</span>

6 <span>node2</span>

7 <span>node3</span>

8 </div>

9 </body>

10 <script>

11 var node = document.getElementById('node')

12 var nodeLists = node.childNodes

13 var queryNodes = node.querySelectorAll('span')

14 node.appendChild(document.createElement('span'))

15 console.log(nodeLists.length) // 输出为10

16 console.log(queryNodes.length) //输出为3

17 </script>

**HTMLCollection**

　　HTMLCollection是元素集合，它和NodeList很像，有length属性来表示HTMLCollection对象的长度，也可以通过elements.item()传入元素索引来访问。当然它还有一个nameItem()方法，可以返回集合中name属性和id属性值的元素。HTMLDocument 接口的许多属性都是 HTMLCollection 对象，它提供了访问诸如表单、图像和链接等文档元素的便捷方式，比如document.images和document.forms的属性都是HTMLCollection对象。

1 <body>

2 <img src="test.png" id="image1">

3 <img src="test.png" id="image2">

4 <img src="test.png" id="image3">

5 <img src="test.png" id="image4">

6 <img src="test.png" id="image5">

7 <img src="test.png" id="image6">

8 </body>

9 <script>

10 console.log(document.images.namedItem('image1')) //<img src="test.png" id="image1">

11 </script>

　　HTMLCollection的集合和NodeList对象一样也是动态的，他们获取的都是节点或元素集合的一个引用。

**HTMLCollection和NodeList 实时性**

　　前面都说到了它们连个对象都不是历史文档状态的一个静态快照，而是实时性的，这个是一个非常令人惊讶的特性，它们能随着文档的改变而改变，这个是很值得我们注意的，我们在平常使用一些DOM 接口来返回一些DOM集合的时候，常常会忽视掉这些。

　　HTMLCollection和NodeList的实时性非常有用，但是，我们有时要迭代一个NodeList或HTMLCollection对象的时候，我们通常会选择生成当前对象的一个快照或静态副本：

1 var staticLists = Array.prototype.slice.call(nodeListorHtmlCollection, 0)

 　这样的话，我们就可以放心的对当前的DOM集合做一些删减和插入操作，这个在DOM密集操作的时候很有用。

　　还有MDN上面提到了一个将NodeList转化为Array的DOM扩展原型的方法（在IE6/7中存在危险：<http://perfectionkills.com/whats-wrong-with-extending-the-dom/>）：

var arrayMethods = Object.getOwnPropertyNames( Array.prototype );

arrayMethods.forEach( attachArrayMethodsToNodeList );

function attachArrayMethodsToNodeList(methodName)

{

if(methodName !== "length") {

NodeList.prototype[methodName] = Array.prototype[methodName];

}

};

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

var firstDiv = divs[ 0 ];

firstDiv.childNodes.forEach(function( divChild ){

divChild.parentNode.style.color = '#0F0';

});

**结语**

　　DOM最初是为了解析XML而设计的，之后沿用到HTML上。我们可以把DOM分为两部分 core 和 html,Core 部分提供最基础的 XML 解析API说明，HTML 部分专为 HTML 中的 DOM 解析添加其特有的 API。NodeList接口是在core中体现的，HTMLCollection则是在html部分，不同浏览器也会实现它们的不同接口，厂商联盟性质的规范组织出现会让这些更加规范，也不出现之前返回的是NodeList对象，但是却是静态的。

　　这篇文章很多思想都是自己在平时和网上一些博客中了解到的，其中加了很多自己的组织和理解，目的在于梳理下一些比较深入的知识点，如果写的有疏漏和错误之处，还请指出。