## 概述

NodeList对象是一个节点的集合，dom.childNodes，dom.getElementsBy(Tag)Name，dom.querySelectorAll等返回的都是 HTMLCollection 对象。

**属性**

**length**

* NodeList对象中包含的节点个数.

**方法**

* item ( idx )
* 返回NodeList对象中指定索引的节点,如果索引越界,则返回null.也可以简写为nodeList[idx].

**描述**

**一个"live"集合**

大多数情况下,NodeList对象都是个live集合.意思是说,如果文档中的节点树发生变化,则已经存在的NodeList对象也可能会变化.

[?](http://my.oschina.net/u/874424/blog/223318)

|  |  |
| --- | --- |
| 1 | var links = document.getElementsByTagName('a');// 假如现在links.length === 2.document.body.appendChild( links[0].cloneNode(true) ); // 文档中又添加了一个a.// 则links这个NodeList对象也会改变.// links.length === 3 |

但如果该NodeList对象是由document.querySelectorAll(或者Element.querySelectorAll)方法返回的, 则它是非live的(就算把页面内的所有节点清空,links.length还等于2).

**什么NodeList对象没有map和forEach方法?**

NodeList对象在某些方面和数组非常相似,看上去可以直接使用从Array.prototype上继承的方法.然而,这是不可以的.

JavaScript的继承机制是基于原型的.数组元素之所以有一些数组方法( 比如forEach和map),是因为它的原型链上有这些方法,如下:

myArray --> Array.prototype --> Object.prototype --> null (想要获取一个对象的原型链,可以连续的调用Object.getPrototypeOf,直到原型链尽头).

forEach, map这些方式其实是 Array.prototype这个对象的方法.

和数组不一样, NodeList的原型链是这样的:

myNodeList --> NodeList.prototype --> Object.prototype --> null

NodeList.prototype只有一个item方法, 没有Array.prototype上的那些方法, 所以NodeList对象用不了它们.

**解决办法**

一个解决办法就是把Array.prototype上的方法添加到NodeList.prototype上. 可是, 要注意扩展DOM对象的原型是非常危险的,尤其是在旧版本的Internet Explorer(6，7，8)中

for(prop in Array.prototype){

  if(Array.prototype.hasOwnProperty(prop) && typeof(Array.prototype[prop]) === 'function')

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

}

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

links.forEach(function(link){ //

  link.style.color = '#0F0';

});