折叠DOM范围

折叠范围，就是指范围中未选择文档的任何部分。

collapse()方法

使用collapse()方法来折叠范围，这个方法接受一个参数，一个布尔值，表示要折叠到范围的哪一端。参数true表示折叠到范围的起点，false表示折叠到范围的终点。要确定范围已经折叠完毕，可以检查collaosed属性。

var *range1* = document.createRange();  
var *p1* = document.getElementById("p1");  
var *b* = *p1*.children[0];  
var *span* = *p1*.children[1];  
*range1*.setStart(*b*.firstChild,1);  
*range1*.setEnd(*b*.firstChild,4);  
var *sp* = document.createElement("span");  
*sp*.style.color = "red";  
*range1*.surroundContents(*sp*);  
*range1*.collapse(true);  
alert(*range1*.collapsed);  
alert(*range1*);//弹出空 但是页面显示还是ell是红色的

检测某个范围是否处于折叠状态，可以帮我们确定范围中的两个节点是否紧紧相邻

<**p** id=**"p1"**>  
 <**b**>**hello**</**b**><**span**> **world!**</**span**>  
</**p**>

比如说我现在HTML是这样的，b元素和span元素是相邻的，但是我们现在假装不知道，假装这是动态生成的，我来用折叠的方式检测他们是否相邻

var *range1* = document.createRange();  
var *b* = p1.children[0];  
var *span* = p1.children[1];  
*range1*.setStartAfter(*b*);  
*range1*.setEndBefore(*span*);  
alert(*range1*.collapsed);//true

因为b之后，span之前什么都没有，所以这个范围是折叠的。

比较DOM范围

compareBoundaryPoints()方法，可以在由多个范围的情况下确定这些范围是否有公共的边界（起点或终点）。这个方法接受两个参数：表示比较方式的常量值和要比较的范围。

表示比较方法的常量值

Range.START\_TO\_START(0):比较第一个范围和第二个范围的起点；

Range.START\_TO\_END(1):比较第一个范围的起点和第二个范围的终点；

Range.END\_TO\_END(2):比较第一个范围和第二个范围的终点；

Range.END\_TO\_START(3):比较第一个范围的终点和第二个范围的起点；

compareBoundaryPoints()方法可能的返回值如下：如果第一个范围中的点位于第二个范围中的点之前，返回-1；如果两个点相等，返回0；如果第一个范围中的点位于第二个范围中的点之后，范围1.

var *range1* = document.createRange();  
var *range2* = document.createRange();  
var *b* = p1.children[0];  
var *span* = p1.children[1];  
*range1*.setStartBefore(*b*);  
*range1*.setEndAfter(*span*);  
*range2*.setStartAfter(*b*);  
*range2*.setEndAfter(*span*);  
alert(*range1*.compareBoundaryPoints(Range.START\_TO\_START,*range2*));//-1  
alert(*range1*.compareBoundaryPoints(Range.END\_TO\_END,*range2*));//0

复制DOM范围

cloneRange()方法复制范围。这个方法会创建调用它的范围的一个副本。

var newRange = range.cloneRange();

新创建的范围与原来的范围包含相同的属性，而修改它的断电不会影响原来的范围。

cloneRange()和cloneContents()的区别

var *range1* = document.createRange();  
var *newRange* = *range1*.cloneRange();  
var *fragment* = *range1*.cloneContents();  
alert(*newRange*);//空  
alert(*fragment*);//[object DocumentFragment]

然后自己看一下他们的介绍

cloneRange()方法复制范围。这个方法会创建调用它的范围的一个副本。

cloneContents()创建范围对象的一个副本，然后再文档的其他地方插入该副本。 cloneContents()返回的文档片段包含的是范围中节点的副本，而不是实际的节点.

清理DOM范围

在使用完范围之后，最好调用detach()方法，以便从创建范围的文档中分理出该范围。调用detach()之后，就可以放心地解除对范围的引用，从而让垃圾回收机制回收其内存的。

range.detach(); //从文档中分离

range = null; //解除引用

在使用范围的最后再执行这两个步骤使我们推荐的方式

一旦分离范围，就不能再恢复使用啦！