IE8及更早版本中的范围

IE8及之前版本不支持DOM范围，但是它们支持一个类似的概念，即文本范围。（这个是IE专有的特性，其他浏览器都不支持）。

顾名思义，文本范围处理的主要是文本（不一定是DOM节点）。通过<body>、<button>、<input>、和<textarea>等这几个元素，可以调用createTextRange()方法来创建文本范围。

文本范围处理的主要是文本（不一定是DOM节点），通过<body>、<button>、<input>和<textarea>等元素，调用createTextRange()方法来创建文本范围

var range=document.body.createTextRange();　　//通过其他元素创建的范围只能在相应元素中使用

用IE范围实现简单的选择

选择页面中某一区域的最简单的方式，就是使用范围的findText()方法，该方法会找到第一次出现的给定文本，并将范围移过来环绕改文本，找到返回true，未找到返回false

<p id="p1"><b>Hello</b> world!</p>

var range=document.body.createTextRange();

var found=range.findText("Hello");　　//选择“Hello”

执行第二行代码以后，文本“Hello”就被包含在范围之内了，这时可以检查范围的text属性或findText()的返回值

alert(found);　　//true

alert(range.text)　　//"Hello"

还可以为findText()传入另一个参数，一个表示继续搜索方向的数值，负值表示从当前位置向后搜索，正值表示从当前位置向前搜索

Html

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

js

var range1 = document.body.createTextRange();  
range1.findText("hello",1);  
alert(range1.htmlText);//<div>hello</div>  
range1.findText("hello",-1)  
alert(range1.htmlText);//hello

HTML

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

JS

var range1 = document.body.createTextRange();  
range1.findText("hello");//<p>hello</p>  
alert(range1.htmlText);  
range1.findText("hello",-1);//<p>hello</p>  
alert(range1.htmlText);  
range1.findText("hello",1);//<p>hello</p>  
alert(range1.htmlText);

这个方法真的很奇怪。。。测试了很久，一开始是这样，两个hello，改了第二个参数，三个alert还是返回一样的结果

然后加了个DIV，更是第二个alert()直接返回hello了。。。

能不用就不用吧。。

IE中与DOM中的selectNode()方法最接近的方法是moveToElementText()，该方法接收一个DOM元素，并选择该元素的所有文本，包括HTML标签

var range=document.body.createTextRange();

var p1=document.getElementById("p1");

range.moveToElementText(p1);

在文本范围包含HTML的情况下，可以使用htmlText属性取得范围的全部内容，包括HTML和文本：

alert(range.htmlText);

有个parentElement()方法与DOM的commonAncestorContainer属性类似

var ancestor=range.parentElement();　　//这样得到的父元素始终都可以反映文本选区的父节点

使用IE范围实现复杂的选择

IE提供了4个方法：move()、moveStart()、moveEnd()和expand()，这些方法都接收2个参数：移动单位和移动单位的数量。移动单元是下列一种字符串值：

"character"：逐个字符移动  
"word"：逐个单词（一系列非空字符）移动

"sentence"：逐个句子（一系列以句号、问号或叹号结尾的字符）移动

"textedit"：移动到当前范围选区的开始或结束位置

moveStart()、moveEnd()分别移动范围的起点和终点：

range.moveStart("word",2);　　//起点移动2个单词

expand()方法可以将范围扩展，例如当前选择的是一个单词中间的两个字符，调用expand("word")将选中整个单词，将其包含在范围中

move()方法则会首先折叠当前范围（起点和终点相等），然后再将范围移动指定的单位数量

range.move("character",5);　　//移动5个字符

调用move()之后，范围的起点和终点相同，必须再使用moveStart()、moveEnd()创建新的选区

var range1 = document.body.createTextRange();  
var p1 = document.getElementById("p1");  
range1.moveToElementText(p1);  
alert(range1.text);//hello world!  
range1.moveStart("character",1);  
alert(range1.text);//ello world!  
range1.expand("word");  
alert(range1.text);//hello world!  
range1.moveStart("word",1);  
alert(range1.text);//world!  
range1.move("character",1);  
alert(range1.text);//空

操作IE范围中的内容

在IE操作范围的内容可以使用text属性或pasteHTML()方法（替换原来范围的内容）

range.text="world";　　//设置属性范围中的内容文本，HTML标签不变

range.pasteHTML("<em>world</em>");　　//这个方法可以插入HTML标签

折叠IE范围

IE为范围提供collapse()方法与相应的DOM方法用法一样；传入true把范围折叠到起点，传入false把范围折叠到终点，却没有相应的collapsed属性让我们知道范围是否已经折叠完毕，必须使用boundingWidth属性，返回该范围的宽度（以像素为单位），boundingWidth属性等于0，表示已经折叠

var isCollapsed=(range.boundingWidth==0);

此外还有boundingHeight、boundingLeft、boundingTop等属性，它们并不像boundingWidth属性那么有用

比较IE范围

IE中compareEndPoints方法与DOM范围中compareBoundaryPoints()方法类似，它接收两个参数： 要比较的类型和要比较的范围。比较类型的取值范围是下列几个字符串值：“StartToStart”、“StartToEnd”“EndToEnd”和“EndToStart”，IE中compareEndPoints方法与DOM范围中compareBoundaryPoints()方法返回的值相同，即如果第一个范围的边界位于第二个范围的边界前面，返回-1；相同，返回0；后面，返回1。

alert(range1.compareEndPoints("StartToStart",range2));　　//返回一个数值

isEqual()用于确定两个范围是否相等：range1.isEqual(range2);

inRange()用于确定一个范围是否包含另一个范围range1.inRange(range2);　　//range1是否包含range2

复制IE范围

duplicate()方法创建原来范围的副本

var newRange=range.duplicate();