```
1  /*
2  **  File: widgEdit.js
3  **  Created by: Cameron Adams (http://www.themaninblue.com/)
4  **  Created on: 2005-01-16
5  **  Last modified: 2008-03-01
6  **
7  **
8  **
9  **
10 **  License Information:
11 **  ----------------------------------------------------------------------
12 **  Copyright (C) 2008 Cameron Adams
13 **
14 **  This program is free software; you can redistribute it and/or modify it
15 **  under the terms of the GNU General Public License as published by the
16 **  Free Software Foundation; either version 2 of the License, or (at your
17 **  option) any later version.
18 **
19 **  This program is distributed in the hope that it will be useful, but
20 **  WITHOUT ANY WARRANTY; without even the implied warranty of
21 **  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 **  General Public License for more details.
23 **
24 **  You should have received a copy of the GNU General Public License along
25 **  with this program; if not, write to the Free Software Foundation, Inc.,
26 **  59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27 **
28 **
29 **
30 **
31 **  Purpose:
32 **  ----------------------------------------------------------------------
33 **
34 **  Replaces all textareas (class="widgEditor") in a HTML document with
35 **  enhanced editing windows to allow basic HTML formatting in a WYSIWYG
36 **  manner.
37 **
38 **
39 **
40 **
41 **  Function list:
42 **  ----------------------------------------------------------------------
43 **
44 **  run()
45 **
46 **  widgInit()
47 **
48 **  widgEditor(replacedTextareaID)
49 **  widtEditor.cleanPaste()
50 **  widgEditor.cleanSource()
51 **  widgEditor.convertSPANs(theSwitch)
52 **  widgEditor.detectPaste(e)
53 **  widgEditor.initEdit()
54 **  widgEditor.insertNewParagraph()
55 **  widgEditor.modifyFormSubmit()
56 **  widgEditor.paragraphise()
57 **  widgEditor.refreshDisplay()
58 **  widgEditor.switchMode()
59 **  widgEditor.updateWidgInput()
60 **  widgEditor.writeDocument()
61 **
62 **  widgToolbar()
63 **  widgToolbar.addButton(theID, theClass, theLabel, theAction)
64 **  widgToolbar.addSelect(theID, theClass, theContentArray, theAction)
65 **  widgToolbar.disable()
66 **  widgToolbar.enable()
67 **  widgToolbar.setState(theState, theStatus)
68 **
69 **  widgToolbarAction()
```

```
70  **
71  **    widgToolbarCheckState(theWidgEditor, resubmit)
72  **
73  **    widgToolbarMouseover()
74  **
75  **    acceptableChildren(theNode)
76  **
77  **    changeNodeType(theNode, nodeType)
78  **
79  **    replaceNodeWithChildren()
80  **
81  **    String.addClass(theClass)
82  **    String.classExists(theClass)
83  **    String.isAcceptedElementName()
84  **    String.isInlineName()
85  **    String.removeClass(theClass)
86  **    String.reverse()
87  **    String.validTags()
88  */
89
90  function isNumeric(n) {
91        return !isNaN(parseFloat(n)) && isFinite(n);
92        }
93
94
95  /*****************************************************************************
96  **    CONFIGURATION VARIABLES
97  *****************************************************************************/
98
99  /* Location of stylesheet file for editor content */
100 var widgStylesheet = "widgContent.css";
101
102 /* Items to appear in toolbar. */
103 var widgToolbarItems = new Array();
104
105 widgToolbarItems.push("bold");
106 widgToolbarItems.push("italic");
107 widgToolbarItems.push("underline");
108 widgToolbarItems.push("hyperlink");
109 widgToolbarItems.push("unorderedlist");
110 widgToolbarItems.push("orderedlist");
111 widgToolbarItems.push("image");
112 widgToolbarItems.push("htmlsource");
113 widgToolbarItems.push("blockformat");
114
115 /* Options on block format select element. Consists of string pairs (option value, option label) */
116 var widgSelectBlockOptions = new Array();
117
118 widgSelectBlockOptions.push("", "Change block type");
119 widgSelectBlockOptions.push("<h1>", "Heading 1");
120 widgSelectBlockOptions.push("<h2>", "Heading 2");
121 widgSelectBlockOptions.push("<h3>", "Heading 3");
122 widgSelectBlockOptions.push("<h4>", "Heading 4");
123 widgSelectBlockOptions.push("<h5>", "Heading 5");
124 widgSelectBlockOptions.push("<h6>", "Heading 6");
125 widgSelectBlockOptions.push("<p>", "Paragraph");
126
127 /* If widgInsertParagraphs = true, when content is submitted paragraphs will be
128 ** inserted around text without a parent element. Mozilla does not
129 ** automatically do this, so if this is set to false you will end up with some
130 ** plain text blocks. Uses a double <br /> as a pargraph marker.
131 */
132
133 var widgInsertParagraphs = true;
134
135 /* If widgAutoClean = true, when content is pasted into the WYSIWYG view, it
136 ** will automatically be cleaned. If widgAutoClean = false, the user will be
137 ** prompted as to whether they wish to clean the content.
138 */
```

```
139
140 var widgAutoClean = false;
141
142 /*****************************************************************************
143 **   END CONFIGURATION
144 *****************************************************************************/
145 /*
146       O comando run() inicializava todos os Textareas
147       Omitindo-o, quem passa a ditar o comportamento de widgEditor.js é widgInit
148       Data: 09/05/2019
149 */
150
151 //run();
152
153
154
155
156 function run()
157 {
158       var oldOnload = window.onload;
159
160       if (typeof(window.onload) != "function")
161       {
162             window.onload = widgInit;
163       }
164       else
165       {
166             window.onload = function()
167             {
168                   oldOnload();
169                   widgInit();
170             }
171       }
172 }
173
174 /*
175       O comando widgInit() inicializava todos os Textareas
176       Colocamos um parâmetro, na forma de Array, com os Ids dos Textareas que vão ser inicializados
177       com o controle de edição.
178       Data: 09/05/2019
179 */
180
181
182 function widgInit(elegiveis)
183 {
184       /* Detects if designMode is available, and also if browser is IE or Mozilla (excludes Safari) */
185       if (typeof(document.designMode) == "string" && (document.all || document.designMode == "off"))
186       {
187             //var theTextareas = document.getElementsByTagName("textarea");
188             var theTextareas = elegiveis;
189             for (var i = 0; i < theTextareas.length; i++)
190             {
191                   var theTextarea = theTextareas[i];
192
193                   if (theTextarea.className.classExists("widgEditor"))
194                   {
195                         if (theTextarea.id == "")
196                         {
197                               theTextarea.id = theTextarea.name;
198                         }
199
200
201                         setTimeout("new widgEditor('" + theTextarea.id + "')", 500 * (i));
202                   }
203             }
204       }
205       else
206       {
207             return false;
```

```
208         }
209
210         return true;
211 }
212
213
214
215
216 function widgEditor(replacedTextareaID)
217 {
218         var self = this;
219
220         this.theTextarea = document.getElementById(replacedTextareaID);
221         this.theContainer = document.createElement("div");
222         this.theIframe = document.createElement("iframe");
223         this.theInput = document.createElement("input");
224         this.theExtraInput = document.createElement("input");
225         this.IE = false;
226         this.locked = true;
227         this.pasteCache = "";
228         this.wysiwyg = true;
229
230         if (document.all)
231         {
232                 this.IE = true;
233         }
234
235         if (this.theTextarea.id == null)
236         {
237                 this.theTextarea.id = this.theTextarea.name;
238         }
239
240         this.theTextarea.style.visibility = "hidden";
241
242         /* Modify DOM objects for editor */
243         this.theContainer.id = this.theTextarea.id + "WidgContainer";
244         this.theContainer.className = "widgContainer";
245
246         this.theIframe.id = this.theTextarea.id + "WidgIframe";
247         this.theIframe.className = "widgIframe";
248
249         this.theInput.type = "hidden";
250         this.theInput.id = this.theTextarea.id;
251         this.theInput.name = this.theTextarea.name;
252         this.theInput.value = this.theTextarea.value;
253
254         this.theToolbar = new widgToolbar(this);
255
256         /* An extra input to determine if the submitted data is from the normal textarea or from the widgEditor */
257         this.theExtraInput.type = "hidden";
258         this.theExtraInput.id = this.theTextarea.id + "WidgEditor";
259         this.theExtraInput.name = this.theTextarea.name + "WidgEditor";
260         this.theExtraInput.value = "true";
261
262         this.theTextarea.id += "WidgTextarea";
263         this.theTextarea.name += "WidgTextarea";
264
265         this.theContainer.appendChild(this.theToolbar.theList);
266         this.theContainer.appendChild(this.theIframe);
267         this.theContainer.appendChild(this.theInput);
268         this.theContainer.appendChild(this.theExtraInput);
269         this.theContainer.style.visibility = "hidden";
270
271         this.theInput.widgEditorObject = this;
272
273         this.theTextarea.parentNode.replaceChild(this.theContainer, this.theTextarea);
274
275         /* Fill editor with old textarea content */
276         this.writeDocument(this.theInput.value);
```

```
277
278         /* Make editor editable */
279         this.initEdit();
280
281         /* Attach onsubmit to parent form */
282         this.modifyFormSubmit();
283
284         return true;
285 }
286
287
288
289
290 /* Clean pasted content */
291 widgEditor.prototype.cleanPaste = function()
292 {
293         if (widgAutoClean || confirm("Do you wish to clean the HTML source of the content you just pasted?"))
294         {
295                 var matchedHead = "";
296                 var matchedTail = "";
297                 var newContent =
    this.theIframe.contentWindow.document.getElementsByTagName("body")[0].innerHTML;
298                 var newContentStart = 0;
299                 var newContentFinish = 0;
300                 var newSnippet = "";
301                 var tempNode = document.createElement("div");
302
303                 /* Find start of both strings that matches */
304                 for (newContentStart = 0; newContent.charAt(newContentStart) ==
    this.pasteCache.charAt(newContentStart); newContentStart++)
305                 {
306                         matchedHead += this.pasteCache.charAt(newContentStart);
307                 }
308
309                 /* If newContentStart is inside a HTML tag, move to opening brace of tag */
310                 for (var i = newContentStart; i >= 0; i--)
311                 {
312                         if (this.pasteCache.charAt(i) == "<")
313                         {
314                                 newContentStart = i;
315                                 matchedHead = this.pasteCache.substring(0, newContentStart);
316
317                                 break;
318                         }
319                         else if(this.pasteCache.charAt(i) == ">")
320                         {
321                                 break;
322                         }
323                 }
324
325                 newContent = newContent.reverse();
326                 this.pasteCache = this.pasteCache.reverse();
327
328                 /* Find end of both strings that matches */
329                 for (newContentFinish = 0; newContent.charAt(newContentFinish) ==
    this.pasteCache.charAt(newContentFinish); newContentFinish++)
330                 {
331                         matchedTail += this.pasteCache.charAt(newContentFinish);
332                 }
333
334                 /* If newContentFinish is inside a HTML tag, move to closing brace of tag */
335                 for (var i = newContentFinish; i >= 0; i--)
336                 {
337                         if (this.pasteCache.charAt(i) == ">")
338                         {
339                                 newContentFinish = i;
340                                 matchedTail = this.pasteCache.substring(0, newContentFinish);
341
342                                 break;
```

```
343                     }
344                     else if(this.pasteCache.charAt(i) == "<")
345                     {
346                         break;
347                     }
348                 }
349
350                 matchedTail = matchedTail.reverse();
351
352                 /* If there's no difference in pasted content */
353                 if (newContentStart == newContent.length - newContentFinish)
354                 {
355                     return false;
356                 }
357
358                 newContent = newContent.reverse();
359                 newSnippet = newContent.substring(newContentStart, newContent.length - newContentFinish);
360                 newSnippet = newSnippet.validTags();
361
362                 /* Replace opening bold tags with strong */
363                 newSnippet = newSnippet.replace(/<b(\s+|>)/g, "<strong$1");
364                 /* Replace closing bold tags with closing strong */
365                 newSnippet = newSnippet.replace(/<\/b(\s+|>)/g, "</strong$1");
366
367                 /* Replace italic tags with em */
368                 newSnippet = newSnippet.replace(/<i(\s+|>)/g, "<em$1");
369                 /* Replace closing italic tags with closing em */
370                 newSnippet = newSnippet.replace(/<\/i(\s+|>)/g, "</em$1");
371
372                 /* Strip out unaccepted attributes */
373                 newSnippet = newSnippet.replace(/<[^>]*>/g, function(match)
374                     {
375                         match = match.replace(/ ([^=]+)="[^"]*"/g, function(match2, attributeName)
376                             {
377                                 if (attributeName == "alt" || attributeName == "href" || attributeName ==
    "src" || attributeName == "title")
378                                 {
379                                     return match2;
380                                 }
381
382                                 return "";
383                             });
384
385                         return match;
386                     }
387                     );
388
389                 tempNode.innerHTML = newSnippet;
390
391                 acceptableChildren(tempNode);
392
393                 this.theInput.value = matchedHead + tempNode.innerHTML + matchedTail;
394
395                 /* Final cleanout for MS Word cruft */
396                 this.theInput.value = this.theInput.value.replace(/<\?xml[^>]*>/g, "");
397                 this.theInput.value = this.theInput.value.replace(/<[^ >]+:[^>]*>/g, "");
398                 this.theInput.value = this.theInput.value.replace(/<\/[^ >]+:[^>]*>/g, "");
399
400                 this.refreshDisplay();
401
402                 /* Convert semantics to spans in Mozilla */
403                 if (!this.IE)
404                 {
405                     this.convertSPANs();
406                 }
407             }
408
409             return true;
410         }
```

```
411
412
413
414
415  /* Clean the HTML code of the content area */
416  widgEditor.prototype.cleanSource = function()
417  {
418       var theHTML = "";
419
420       if (this.wysiwyg)
421       {
422            theHTML = this.theIframe.contentWindow.document.getElementsByTagName("body")[0].innerHTML;
423       }
424       else
425       {
426            theHTML = this.theTextarea.value;
427       }
428
429       theHTML = theHTML.validTags();
430
431       /* Remove leading and trailing whitespace */
432       theHTML = theHTML.replace(/^\s+/, "");
433       theHTML = theHTML.replace(/\s+$/, "");
434
435       /* Remove style attribute inside any tag */
436       theHTML = theHTML.replace(/ style="[^"]*"/g, "");
437
438       /* Replace improper BRs */
439       theHTML = theHTML.replace(/<br>/g, "<br />");
440
441       /* Remove BRs right before the end of blocks */
442       theHTML = theHTML.replace(/<br \/>\s*<\/(h1|h2|h3|h4|h5|h6|li|p)/g, "</$1");
443
444       /* Replace improper IMGs */
445       theHTML = theHTML.replace(/(<img [^>]+[^\/])>/g, "$1 />");
446
447       /* Remove empty tags */
448       theHTML = theHTML.replace(/(<[^\/]>|<[^\/][^>]*[^\/]>)\s*<\/[^>]*>/g, "");
449
450       if (this.wysiwyg)
451       {
452            this.theIframe.contentWindow.document.getElementsByTagName("body")[0].innerHTML = theHTML;
453       }
454       else
455       {
456            this.theTextarea.value = theHTML;
457       }
458
459       this.theInput.value = theHTML;
460
461       return true;
462  }
463
464
465
466
467  widgEditor.prototype.convertSPANs = function(theSwitch)
468  {
469       if (theSwitch)
470       {
471            /* Replace styled spans with their semantic equivalent */
472            var theSPANs = this.theIframe.contentWindow.document.getElementsByTagName("span");
473
474            while(theSPANs.length > 0)
475            {
476                 var theChildren = new Array();
477                 var theReplacementElement = null;
478                 var theParentElement = null;
479
```

```
480                     for (var j = 0; j < theSPANs[0].childNodes.length; j++)
481                     {
482                           theChildren.push(theSPANs[0].childNodes[j].cloneNode(true));
483                     }
484
485                     /* Detect type of span style */
486                     switch (theSPANs[0].getAttribute("style"))
487                     {
488                           case "font-weight: bold;":
489                                 theReplacementElement =
    this.theIframe.contentWindow.document.createElement("strong");
490                                 theParentElement = theReplacementElement;
491
492                                 break;
493
494                           case "font-style: italic;":
495                                 theReplacementElement =
    this.theIframe.contentWindow.document.createElement("em");
496                                 theParentElement = theReplacementElement;
497
498                                 break;
499
500                           case "font-weight: bold; font-style: italic;":
501                                 theParentElement = this.theIframe.contentWindow.document.createElement("em");
502                                 theReplacementElement =
    this.theIframe.contentWindow.document.createElement("strong");
503                                 theReplacementElement.appendChild(theParentElement);
504
505                                 break;
506
507                           case "font-style: italic; font-weight: bold;":
508                                 theParentElement =
    this.theIframe.contentWindow.document.createElement("strong");
509                                 theReplacementElement =
    this.theIframe.contentWindow.document.createElement("em");
510                                 theReplacementElement.appendChild(theParentElement);
511
512                                 break;
513
514                           default:
515                                 replaceNodeWithChildren(theSPANs[0]);
516
517                                 break;
518                     }
519
520                     if (theReplacementElement != null)
521                     {
522                           for (var j = 0; j < theChildren.length; j++)
523                           {
524                                 theParentElement.appendChild(theChildren[j]);
525                           }
526
527                           theSPANs[0].parentNode.replaceChild(theReplacementElement, theSPANs[0]);
528                     }
529
530                     theSPANs = this.theIframe.contentWindow.document.getElementsByTagName("span");
531               }
532         }
533         else
534         {
535               /* Replace em and strong tags with styled spans */
536               var theEMs = this.theIframe.contentWindow.document.getElementsByTagName("em");
537
538               while(theEMs.length > 0)
539               {
540                     var theChildren = new Array();
541                     var theSpan = this.theIframe.contentWindow.document.createElement("span");
542
543                     theSpan.setAttribute("style", "font-style: italic;");
```

```
544
545                     for (var j = 0; j < theEMs[0].childNodes.length; j++)
546                     {
547                         theChildren.push(theEMs[0].childNodes[j].cloneNode(true));
548                     }
549
550                     for (var j = 0; j < theChildren.length; j++)
551                     {
552                         theSpan.appendChild(theChildren[j]);
553                     }
554
555                     theEMs[0].parentNode.replaceChild(theSpan, theEMs[0]);
556                     theEMs = this.theIframe.contentWindow.document.getElementsByTagName("em");
557                 }
558
559                 var theSTRONGs = this.theIframe.contentWindow.document.getElementsByTagName("strong");
560
561                 while(theSTRONGs.length > 0)
562                 {
563                     var theChildren = new Array();
564                     var theSpan = this.theIframe.contentWindow.document.createElement("span");
565
566                     theSpan.setAttribute("style", "font-weight: bold;");
567
568                     for (var j = 0; j < theSTRONGs[0].childNodes.length; j++)
569                     {
570                         theChildren.push(theSTRONGs[0].childNodes[j].cloneNode(true));
571                     }
572
573                     for (var j = 0; j < theChildren.length; j++)
574                     {
575                         theSpan.appendChild(theChildren[j]);
576                     }
577
578                     theSTRONGs[0].parentNode.replaceChild(theSpan, theSTRONGs[0]);
579                     theSTRONGs = this.theIframe.contentWindow.document.getElementsByTagName("strong");
580                 }
581             }
582
583             return true;
584         }
585
586
587
588
589         /* Check for pasted content */
590         widgEditor.prototype.detectPaste = function(e)
591         {
592             var keyPressed = null;
593             var theEvent = null;
594
595             if (e)
596             {
597                 theEvent = e;
598             }
599             else
600             {
601                 theEvent = event;
602             }
603
604             if (theEvent.ctrlKey && theEvent.keyCode == 86 && this.wysiwyg)
605             {
606                 var self = this;
607
608                 this.pasteCache =
    this.theIframe.contentWindow.document.getElementsByTagName("body")[0].innerHTML;
609
610                 /* Because Mozilla can't access the clipboard directly, must rely on timeout to check pasted differences in
    main content */
```

```
611                 setTimeout(function(){self.cleanPaste(); return true;}, 100);
612         }
613
614         return true;
615 }
616
617
618
619
620 /* Turn on document editing */
621 widgEditor.prototype.initEdit = function()
622 {
623         var self = this;
624
625         try
626         {
627                 this.theIframe.contentWindow.document.designMode = "on";
628         }
629         catch (e)
630         {
631                 /* setTimeout needed to counteract Mozilla bug whereby you can't immediately change designMode on
    newly created iframes */
632                 setTimeout(function(){self.initEdit()}, 250);
633
634                 return false;
635         }
636
637         if (!this.IE)
638         {
639                 this.convertSPANs(false);
640         }
641
642         this.theContainer.style.visibility = "visible";
643         this.theTextarea.style.visibility = "visible";
644
645         /* Mozilla event capturing */
646         if (typeof document.addEventListener == "function")
647         {
648                 this.theIframe.contentWindow.document.addEventListener("mouseup",
    function(){widgToolbarCheckState(self); return true;}, false);
649                 this.theIframe.contentWindow.document.addEventListener("keyup",
    function(){widgToolbarCheckState(self); return true;}, false);
650                 this.theIframe.contentWindow.document.addEventListener("keydown",
    function(e){self.detectPaste(e); return true;}, false);
651         }
652         /* IE event capturing */
653         else
654         {
655                 this.theIframe.contentWindow.document.attachEvent("onmouseup",
    function(){widgToolbarCheckState(self); return true;});
656                 this.theIframe.contentWindow.document.attachEvent("onkeyup",
    function(){widgToolbarCheckState(self); return true;});
657                 this.theIframe.contentWindow.document.attachEvent("onkeydown",
    function(e){self.detectPaste(e); return true;}, false);
658         }
659
660         this.locked = false;
661
662         return true;
663 }
664
665
666
667
668 /* Add elements to a paragraph and inserts the paragraph before a given element in the body */
669 widgEditor.prototype.insertNewParagraph = function(elementArray, succeedingElement)
670 {
671         var theBody = this.theIframe.contentWindow.document.getElementsByTagName("body")[0];
672         var theParagraph = this.theIframe.contentWindow.document.createElement("p");
```

```
673
674         for (var i = 0; i < elementArray.length; i++)
675         {
676             theParagraph.appendChild(elementArray[i]);
677         }
678
679         if (typeof(succeedingElement) != "undefined")
680         {
681             theBody.insertBefore(theParagraph, succeedingElement);
682         }
683         else
684         {
685             theBody.appendChild(theParagraph);
686         }
687
688         return true;
689 }
690
691
692
693
694 /* Add submit listener to parent form */
695 widgEditor.prototype.modifyFormSubmit = function()
696 {
697         return true;
698         }
699
700
701
702
703 /* Format the HTML with paragraphs. Any parentless text is enclosed in a paragraph, double breaks are paragraph
    markers */
704 widgEditor.prototype.paragraphise = function()
705 {
706         if (widgInsertParagraphs && this.wysiwyg)
707         {
708             var theBody = this.theIframe.contentWindow.document.getElementsByTagName("body")[0];
709
710             /* Remove all text nodes containing just whitespace */
711             for (var i = 0; i < theBody.childNodes.length; i++)
712             {
713                 if (theBody.childNodes[i].nodeName.toLowerCase() == "#text" &&
714                     theBody.childNodes[i].data.search(/^\s*$/) != -1)
715                 {
716                     theBody.removeChild(theBody.childNodes[i]);
717
718                     i--;
719                 }
720             }
721
722             var removedElements = new Array();
723
724             for (var i = 0; i < theBody.childNodes.length; i++)
725             {
726                 if (theBody.childNodes[i].nodeName.isInlineName())
727                 {
728                     removedElements.push(theBody.childNodes[i].cloneNode(true));
729
730                     theBody.removeChild(theBody.childNodes[i]);
731
732                     i--;
733                 }
734                 else if (theBody.childNodes[i].nodeName.toLowerCase() == "br")
735                 {
736                     if (i + 1 < theBody.childNodes.length)
737                     {
738                         /* If the current break tag is followed by another break tag */
739                         if (theBody.childNodes[i + 1].nodeName.toLowerCase() == "br")
740                         {
```

```
741                                     /* Remove consecutive break tags */
742                                     while (i < theBody.childNodes.length &&
    theBody.childNodes[i].nodeName.toLowerCase() == "br")
743                                     {
744                                         theBody.removeChild(theBody.childNodes[i]);
745                                     }
746
747                                     if (removedElements.length > 0)
748                                     {
749                                         this.insertNewParagraph(removedElements, theBody.childNodes[i]);
750
751                                         removedElements = new Array();
752                                     }
753                                 }
754                                 /* If the break tag appears before a block element */
755                                 else if (!theBody.childNodes[i + 1].nodeName.isInlineName())
756                                 {
757                                     theBody.removeChild(theBody.childNodes[i]);
758                                 }
759                                 else if (removedElements.length > 0)
760                                 {
761                                     removedElements.push(theBody.childNodes[i].cloneNode(true));
762
763                                     theBody.removeChild(theBody.childNodes[i]);
764                                 }
765                                 else
766                                 {
767                                     theBody.removeChild(theBody.childNodes[i]);
768                                 }
769
770                                 i--;
771                             }
772                             else
773                             {
774                                 theBody.removeChild(theBody.childNodes[i]);
775                             }
776                         }
777                         else if (removedElements.length > 0)
778                         {
779                             this.insertNewParagraph(removedElements, theBody.childNodes[i]);
780
781                             removedElements = new Array();
782                         }
783                     }
784
785                     if (removedElements.length > 0)
786                     {
787                         this.insertNewParagraph(removedElements);
788                     }
789                 }
790
791                 return true;
792             }
793
794
795
796
797             /* Update hidden input to reflect editor contents, for submission */
798             widgEditor.prototype.refreshDisplay = function()
799             {
800                 if (this.wysiwyg)
801                 {
802                     this.theIframe.contentWindow.document.getElementsByTagName("body")[0].innerHTML =
    this.theInput.value;
803                 }
804                 else
805                 {
806                     this.theTextarea.value = this.theInput.value;
807                 }
```

```
808
809         return true;
810 }
811
812
813
814
815 /* Switch between WYSIWYG and HTML source */
816 widgEditor.prototype.switchMode = function()
817 {
818         if (!this.locked)
819         {
820                 this.locked = true;
821
822                 /* Switch to HTML source */
823                 if (this.wysiwyg)
824                 {
825                         this.updateWidgInput();
826                         this.theTextarea.value = this.theInput.value;
827                         this.theContainer.replaceChild(this.theTextarea, this.theIframe);
828                         this.theToolbar.disable();
829                         this.wysiwyg = false;
830                         this.locked = false;
831                 }
832                 /* Switch to WYSIWYG */
833                 else
834                 {
835                         this.updateWidgInput();
836                         this.theContainer.replaceChild(this.theIframe, this.theTextarea);
837                         this.writeDocument(this.theInput.value);
838                         this.theToolbar.enable();
839                         this.initEdit();
840                         this.wysiwyg = true;
841                 }
842         }
843
844         return true;
845 }
846
847
848
849
850 /* Update hidden input to reflect editor contents, for submission */
851 widgEditor.prototype.updateWidgInput = function()
852 {
853         if (this.wysiwyg)
854         {
855                 /* Convert spans to semantics in Mozilla */
856                 if (!this.IE)
857                 {
858                         this.convertSPANs(true);
859                 }
860
861                 this.paragraphise();
862                 this.cleanSource();
863         }
864         else
865         {
866                 this.theInput.value = this.theTextarea.value;
867         }
868
869         return true;
870 }
871
872
873
874
875 /* Write initial content to editor */
876 widgEditor.prototype.writeDocument = function(documentContent)
```

```
877  {
878  	/* HTML template into which the HTML Editor content is inserted */
879  	var documentTemplate = '\
880  		<html>\
881  			<head>\
882  				INSERT:STYLESHEET:END\
883  			</head>\
884  			<body id="iframeBody">\
885  				INSERT:CONTENT:END\
886  			</body>\
887  		</html>\
888  	';
889
890  	/* Insert dynamic variables/content into document */
891  	/* IE needs stylesheet to be written inline */
892  	if (typeof document.all != "undefined")
893  	{
894  		documentTemplate = documentTemplate.replace(/INSERT:STYLESHEET:END/, '<link rel="stylesheet"
     type="text/css" href="' + widgStylesheet + '"></link>');
895  	}
896  	/* Firefox can't have stylesheet written inline */
897  	else
898  	{
899  		documentTemplate = documentTemplate.replace(/INSERT:STYLESHEET:END/, "");
900  	}
901
902  	documentTemplate = documentTemplate.replace(/INSERT:CONTENT:END/, documentContent);
903
904  	this.theIframe.contentWindow.document.open();
905  	this.theIframe.contentWindow.document.write(documentTemplate);
906  	this.theIframe.contentWindow.document.close();
907
908  	/* In Firefox stylesheet needs to be loaded separate to other HTML, because if it's loaded inline it causes Firefox
     to have problems with an empty document */
909  	if (typeof document.all == "undefined")
910  	{
911  		var stylesheet = this.theIframe.contentWindow.document.createElement("link");
912  		stylesheet.setAttribute("rel", "stylesheet");
913  		stylesheet.setAttribute("type", "text/css");
914  		stylesheet.setAttribute("href", widgStylesheet);
915
     this.theIframe.contentWindow.document.getElementsByTagName("head")[0].appendChild(stylesheet);
916  	}
917
918  	return true;
919  }
920
921
922
923
924  /* Toolbar items */
925  function widgToolbar(theEditor)
926  {
927  	var self = this;
928
929  	this.widgEditorObject = theEditor;
930
931  	/* Create toolbar ul element */
932  	this.theList = document.createElement("ul");
933  	this.theList.id = this.widgEditorObject.theInput.id + "WidgToolbar";
934  	this.theList.className = "widgToolbar";
935  	this.theList.widgToolbarObject = this;
936
937  	/* Create toolbar items */
938  	for (var i = 0; i < widgToolbarItems.length; i++)
939  	{
940  		switch (widgToolbarItems[i])
941  		{
942  			case "bold":
```

```
943                         this.addButton(this.theList.id + "ButtonBold", "widgButtonBold", "Bold", "bold");
944
945                         break;
946
947                 case "italic":
948                         this.addButton(this.theList.id + "ButtonItalic", "widgButtonItalic", "Italic", "italic");
949
950                         break;
951
952                 case "underline":
953                         this.addButton(this.theList.id + "ButtonUnderline", "widgButtonUnderline", "Underline",
    "underline");
954
955                         break;
956
957                 case "hyperlink":
958                         this.addButton(this.theList.id + "ButtonLink", "widgButtonLink", "Hyperlink", "link");
959
960                         break;
961
962                 case "unorderedlist":
963                         this.addButton(this.theList.id + "ButtonUnordered", "widgButtonUnordered", "Unordered
    List", "insertunorderedlist");
964
965                         break;
966
967                 case "orderedlist":
968                         this.addButton(this.theList.id + "ButtonOrdered", "widgButtonOrdered", "Ordered List",
    "insertorderedlist");
969
970                         break;
971
972                 case "image":
973                         this.addButton(this.theList.id + "ButtonImage", "widgButtonImage", "Insert Image",
    "image");
974
975                         break;
976
977                 case "htmlsource":
978                         this.addButton(this.theList.id + "ButtonHTML", "widgButtonHTML", "HTML Source",
    "html");
979
980                         break;
981
982                 case "blockformat":
983                         this.addSelect(this.theList.id + "SelectBlock", "widgSelectBlock",
    widgSelectBlockOptions, "formatblock");
984
985                         break;
986             }
987         }
988
989         return true;
990 }
991
992
993
994
995
996 /* Add button to toolbar */
997 widgToolbar.prototype.addButton = function(theID, theClass, theLabel, theAction)
998 {
999         var menuItem = document.createElement("li");
1000        var theLink = document.createElement("a");
1001        var theText = document.createTextNode(theLabel);
1002
1003        menuItem.id = theID;
1004        menuItem.className = "widgEditButton";
1005
```

```
1006        theLink.href = "#";
1007        theLink.title = theLabel;
1008        theLink.className = theClass;
1009        theLink.action = theAction;
1010        theLink.onclick = widgToolbarAction;
1011        theLink.onmouseover = widgToolbarMouseover;
1012
1013        theLink.appendChild(theText);
1014        menuItem.appendChild(theLink);
1015        this.theList.appendChild(menuItem);
1016
1017        return true;
1018  }
1019
1020
1021
1022
1023  /* Add select box to toolbar. theContentArray is an array of string pairs (option value, option label) */
1024  widgToolbar.prototype.addSelect = function(theID, theClass, theContentArray, theAction)
1025  {
1026        var menuItem = document.createElement("li");
1027        var theSelect = document.createElement("select");
1028
1029        menuItem.className = "widgEditSelect";
1030
1031        theSelect.id = theID;
1032        theSelect.name = theID;
1033        theSelect.className = theClass;
1034        theSelect.action = theAction;
1035        theSelect.onchange = widgToolbarAction;
1036        /*
1037              Modificado em: 10/05/2019
1038              Finalidade.....: Alterar a exibição dos Headers de HTML na cortina de SELECT
1039        */
1040        for (var i = 0; i < theContentArray.length; i += 2){
1041              var theOption = document.createElement("option");
1042              var theText = document.createTextNode(theContentArray[i + 1]);
1043              var tamFonte;
1044
1045              theOption.value = theContentArray[i];
1046              var lastChar =
      theText.textContent.substring(theText.textContent.length-1,theText.textContent.length);
1047              if( isNumeric(lastChar) ){
1048                    tamFonte = (7 - parseInt(lastChar))*(7 - parseInt(lastChar))/5+(7 -
      parseInt(lastChar))*3+10;
1049                    tamFonte = parseInt(tamFonte);
1050                    tamFonte = tamFonte.toString()+"px";
1051                    theOption.style.fontSize = tamFonte;
1052                    } else {
1053                    tamFonte = "";
1054                    }
1055
1056              theOption.appendChild(theText);
1057              theSelect.appendChild(theOption);
1058        }
1059
1060        menuItem.appendChild(theSelect);
1061        this.theList.appendChild(menuItem);
1062
1063        return true;
1064  }
1065
1066
1067
1068
1069  /* Turn off toolbar items */
1070  widgToolbar.prototype.disable = function()
1071  {
1072        /* Change class to disable buttons using CSS */
```

```
1073        this.theList.className += " widgSource";
1074
1075        /* Loop through lis */
1076        for (var i = 0; i < this.theList.childNodes.length; i++)
1077        {
1078            var theChild = this.theList.childNodes[i];
1079
1080            if (theChild.nodeName.toLowerCase() == "li" && theChild.className == "widgEditSelect")
1081            {
1082                /* Loop through li children to find select */
1083                for (j = 0; j < theChild.childNodes.length; j++)
1084                {
1085                    if (theChild.childNodes[j].nodeName.toLowerCase() == "select")
1086                    {
1087                        theChild.childNodes[j].disabled = "disabled";
1088
1089                        break;
1090                    }
1091                }
1092            }
1093        }
1094
1095        return true;
1096    }
1097
1098
1099
1100
1101    /* Turn on toolbar items */
1102    widgToolbar.prototype.enable = function()
1103    {
1104        /* Change class to enable buttons using CSS */
1105        this.theList.className = this.theList.className.replace(/ widgSource/, "");
1106
1107        /* Loop through lis */
1108        for (var i = 0; i < this.theList.childNodes.length; i++)
1109        {
1110            var theChild = this.theList.childNodes[i];
1111
1112            if (theChild.nodeName.toLowerCase() == "li" && theChild.className == "widgEditSelect")
1113            {
1114                /* Loop through li children to find select */
1115                for (j = 0; j < theChild.childNodes.length; j++)
1116                {
1117                    if (theChild.childNodes[j].nodeName.toLowerCase() == "select")
1118                    {
1119                        theChild.childNodes[j].disabled = "";
1120
1121                        break;
1122                    }
1123                }
1124            }
1125        }
1126
1127        return true;
1128    }
1129
1130
1131
1132
1133    /* Change the status of the selected toolbar item */
1134    widgToolbar.prototype.setState = function(theState, theStatus)
1135    {
1136        if (theState != "SelectBlock")
1137        {
1138            var theButton = document.getElementById(this.theList.id + "Button" + theState);
1139
1140            if (theButton != null)
1141            {
```

```
1142                    if (theStatus == "on")
1143                    {
1144                          theButton.className = theButton.className.addClass("on");
1145                    }
1146                    else
1147                    {
1148                          theButton.className = theButton.className.removeClass("on");
1149                    }
1150              }
1151        }
1152        else
1153        {
1154              var theSelect = document.getElementById(this.theList.id + "SelectBlock");
1155
1156              if (theSelect != null)
1157              {
1158                    theSelect.value = "";
1159                    theSelect.value = theStatus;
1160              }
1161        }
1162
1163        return true;
1164  }
1165
1166
1167
1168
1169
1170  /* Action taken when toolbar item activated */
1171  function widgToolbarAction()
1172  {
1173        var theToolbar = this.parentNode.parentNode.widgToolbarObject;
1174        var theWidgEditor = theToolbar.widgEditorObject;
1175        var theIframe = theWidgEditor.theIframe;
1176        var theSelection = "";
1177
1178        /* If somehow a button other than "HTML source" is clicked while viewing HTML source, ignore click */
1179        if (!theWidgEditor.wysiwyg && this.action != "html")
1180        {
1181              return false;
1182        }
1183
1184        switch (this.action)
1185        {
1186              case "formatblock":
1187                    theIframe.contentWindow.document.execCommand(this.action, false, this.value);
1188
1189                    theWidgEditor.theToolbar.setState("SelectBlock", this.value);
1190
1191                    break;
1192
1193              case "html":
1194                    theWidgEditor.switchMode();
1195
1196                    break;
1197
1198              case "link":
1199                    if (this.parentNode.className.classExists("on"))
1200                    {
1201                          theIframe.contentWindow.document.execCommand("Unlink", false, null);
1202                          theWidgEditor.theToolbar.setState("Link", "off");
1203                    }
1204                    else
1205                    {
1206                          if (theIframe.contentWindow.document.selection)
1207                          {
1208                                theSelection = theIframe.contentWindow.document.selection.createRange().text;
1209
1210                                if (theSelection == "")
```

```
1211                            {
1212                                  alert("Please select the text you wish to hyperlink.");
1213
1214                                  break;
1215                            }
1216                      }
1217                      else
1218                      {
1219                            theSelection = theIframe.contentWindow.getSelection();
1220
1221                            if (theSelection == "")
1222                            {
1223                                  alert("Please select the text you wish to hyperlink.");
1224
1225                                  break;
1226                            }
1227                      }
1228
1229                      var theURL = prompt("Enter the URL for this link:", "http://");
1230
1231                      if (theURL != null)
1232                      {
1233                            theIframe.contentWindow.document.execCommand("CreateLink", false, theURL);
1234                            theWidgEditor.theToolbar.setState("Link", "on");
1235                      }
1236                }
1237
1238                break;
1239
1240          case "image":
1241                var theImage = prompt("Enter the location for this image:", "");
1242
1243                if (theImage != null && theImage != "")
1244                {
1245                      var theAlt = prompt("Enter the alternate text for this image:", "");
1246                      var theSelection = null;
1247                      var theRange = null;
1248
1249                      /* IE selections */
1250                      if (theIframe.contentWindow.document.selection)
1251                      {
1252                            /* Escape quotes in alt text */
1253                            theAlt = theAlt.replace(/"/g, "'");
1254
1255                            theSelection = theIframe.contentWindow.document.selection;
1256                            theRange = theSelection.createRange();
1257                            theRange.collapse(false);
1258                            theRange.pasteHTML("<img alt=\"" + theAlt + "\" src=\"" + theImage + "\" />");
1259
1260                            break;
1261                      }
1262                      /* Mozilla selections */
1263                      else
1264                      {
1265                            try
1266                            {
1267                                  theSelection = theIframe.contentWindow.getSelection();
1268                            }
1269                            catch (e)
1270                            {
1271                                  return false;
1272                            }
1273
1274                            theRange = theSelection.getRangeAt(0);
1275                            theRange.collapse(false);
1276
1277                            var theImageNode = theIframe.contentWindow.document.createElement("img");
1278
1279                            theImageNode.src = theImage;
```

```
1280                                theImageNode.alt = theAlt;
1281
1282                                theRange.insertNode(theImageNode);
1283
1284                                break;
1285                          }
1286                    }
1287                    else
1288                    {
1289                          return false;
1290                    }
1291
1292              default:
1293                    theIframe.contentWindow.document.execCommand(this.action, false, null);
1294
1295                    var theAction = this.action.replace(/^./, function(match){return match.toUpperCase();});
1296
1297                    /* Turn off unordered toolbar item if ordered toolbar item was activated */
1298                    if (this.action == "insertorderedlist")
1299                    {
1300                          theAction = "Ordered";
1301                          theWidgEditor.theToolbar.setState("Unordered", "off");
1302                    }
1303
1304                    /* Turn off ordered toolbar item if unordered toolbar item was activated */
1305                    if (this.action == "insertunorderedlist")
1306                    {
1307                          theAction = "Unordered";
1308                          theWidgEditor.theToolbar.setState("Ordered", "off");
1309                    }
1310
1311                    /* If toolbar item was turned on */
1312                    if (theIframe.contentWindow.document.queryCommandState(this.action, false, null))
1313                    {
1314                          theWidgEditor.theToolbar.setState(theAction, "on");
1315                    }
1316                    else
1317                    {
1318                          theWidgEditor.theToolbar.setState(theAction, "off");
1319                    }
1320        }
1321
1322        if (theWidgEditor.wysiwyg == true)
1323        {
1324              theIframe.contentWindow.focus();
1325        }
1326        else
1327        {
1328              theWidgEditor.theTextarea.focus();
1329        }
1330
1331        return false;
1332  }
1333
1334
1335
1336
1337  /* Check the nesting of the current cursor position/selection */
1338  function widgToolbarCheckState(theWidgEditor, resubmit)
1339  {
1340        if (!resubmit)
1341        {
1342              /* Allow browser to update selection before using the selection */
1343              setTimeout(function(){widgToolbarCheckState(theWidgEditor, true); return true;}, 500);
1344        }
1345
1346        var theSelection = null;
1347        var theRange = null;
1348        var theParentNode = null;
```

```
1349        var theLevel = 0;
1350
1351        /* Turn off all the buttons */
1352        var menuListItems = theWidgEditor.theToolbar.theList.childNodes;
1353        for (var i = 0; i < menuListItems.length; i++)
1354        {
1355              menuListItems[i].className = menuListItems[i].className.removeClass("on");
1356        }
1357
1358        /* IE selections */
1359        if (theWidgEditor.theIframe.contentWindow.document.selection)
1360        {
1361              theSelection = theWidgEditor.theIframe.contentWindow.document.selection;
1362              theRange = theSelection.createRange();
1363              try
1364              {
1365                    theParentNode = theRange.parentElement();
1366              }
1367              catch (e)
1368              {
1369                    return false;
1370              }
1371        }
1372        /* Mozilla selections */
1373        else
1374        {
1375              try
1376              {
1377                    theSelection = theWidgEditor.theIframe.contentWindow.getSelection();
1378              }
1379              catch (e)
1380              {
1381                    return false;
1382              }
1383
1384              theRange = theSelection.getRangeAt(0);
1385              theParentNode = theRange.commonAncestorContainer;
1386        }
1387
1388        while (theParentNode.nodeType == 3)
1389        {
1390              theParentNode = theParentNode.parentNode;
1391        }
1392
1393        while (theParentNode.nodeName.toLowerCase() != "body")
1394        {
1395              switch (theParentNode.nodeName.toLowerCase())
1396              {
1397                    case "a":
1398                          theWidgEditor.theToolbar.setState("Link", "on");
1399
1400                          break;
1401
1402                    case "em":
1403                          theWidgEditor.theToolbar.setState("Italic", "on");
1404
1405                          break;
1406
1407                    case "li":
1408
1409                          break;
1410
1411                    case "ol":
1412                          theWidgEditor.theToolbar.setState("Ordered", "on");
1413                          theWidgEditor.theToolbar.setState("Unordered", "off");
1414
1415                          break;
1416
1417                    case "span":
```

```
1418                        if (theParentNode.getAttribute("style") == "font-weight: bold;")
1419                        {
1420                              theWidgEditor.theToolbar.setState("Bold", "on");
1421                        }
1422                        else if (theParentNode.getAttribute("style") == "font-style: italic;")
1423                        {
1424                              theWidgEditor.theToolbar.setState("Italic", "on");
1425                        }
1426                        else if (theParentNode.getAttribute("style") == "font-weight: bold; font-style: italic;")
1427                        {
1428                              theWidgEditor.theToolbar.setState("Bold", "on");
1429                              theWidgEditor.theToolbar.setState("Italic", "on");
1430                        }
1431                        else if (theParentNode.getAttribute("style") == "font-style: italic; font-weight: bold;")
1432                        {
1433                              theWidgEditor.theToolbar.setState("Bold", "on");
1434                              theWidgEditor.theToolbar.setState("Italic", "on");
1435                        }
1436
1437                        break;
1438
1439                  case "strong":
1440                        theWidgEditor.theToolbar.setState("Bold", "on");
1441
1442                        break;
1443
1444                  case "ul":
1445                        theWidgEditor.theToolbar.setState("Unordered", "on");
1446                        theWidgEditor.theToolbar.setState("Ordered", "off");
1447
1448                        break;
1449
1450                  default:
1451                        theWidgEditor.theToolbar.setState("SelectBlock", "<" +
      theParentNode.nodeName.toLowerCase() + ">");
1452
1453                        break;
1454            }
1455
1456            theParentNode = theParentNode.parentNode;
1457            theLevel++;
1458      }
1459
1460      return true;
1461  }
1462
1463
1464
1465
1466  /* Turn off browser status display for toolbar items */
1467  function widgToolbarMouseover()
1468  {
1469      window.status = "";
1470
1471      return true;
1472  }
1473
1474
1475
1476
1477  function acceptableChildren(theNode)
1478  {
1479      var theChildren = theNode.childNodes;
1480
1481      for (var i = 0; i < theChildren.length; i++)
1482      {
1483            if (!theChildren[i].nodeName.isAcceptedElementName())
1484            {
1485                  if (!theChildren[i].nodeName.isInlineName())
```

```
1486                    {
1487                        if (theNode.nodeName.toLowerCase() == "p")
1488                        {
1489                            acceptableChildren(replaceNodeWithChildren(theNode));
1490
1491                            return true;
1492                        }
1493
1494                        changeNodeType(theChildren[i], "p");
1495                    }
1496                    else
1497                    {
1498                        replaceNodeWithChildren(theChildren[i]);
1499                    }
1500
1501                    i = -1;
1502                }
1503            }
1504
1505            for (var i = 0; i < theChildren.length; i++)
1506            {
1507                acceptableChildren(theChildren[i]);
1508            }
1509
1510            return true;
1511        }
1512
1513
1514
1515
1516        /* Change the type of a node, e.g. h3 to p */
1517        function changeNodeType(theNode, nodeType)
1518        {
1519            var theChildren = new Array();
1520            var theNewNode = document.createElement(nodeType);
1521            var theParent = theNode.parentNode;
1522
1523            if (theParent != null)
1524            {
1525                for (var i = 0; i < theNode.childNodes.length; i++)
1526                {
1527                    theChildren.push(theNode.childNodes[i].cloneNode(true));
1528                }
1529
1530                for (var i = 0; i < theChildren.length; i++)
1531                {
1532                    theNewNode.appendChild(theChildren[i]);
1533                }
1534
1535                theParent.replaceChild(theNewNode, theNode);
1536            }
1537
1538            return true;
1539        }
1540
1541
1542
1543
1544        /* Replace a node with its children -- delete the item and move its children up one level in the hierarchy */
1545        function replaceNodeWithChildren(theNode)
1546        {
1547            var theChildren = new Array();
1548            var theParent = theNode.parentNode;
1549
1550            if (theParent != null)
1551            {
1552                for (var i = 0; i < theNode.childNodes.length; i++)
1553                {
1554                    theChildren.push(theNode.childNodes[i].cloneNode(true));
```

```
1555            }
1556
1557            for (var i = 0; i < theChildren.length; i++)
1558            {
1559                  theParent.insertBefore(theChildren[i], theNode);
1560            }
1561
1562            theParent.removeChild(theNode);
1563
1564            return theParent;
1565      }
1566
1567      return true;
1568 }
1569
1570
1571
1572
1573 /* Add a class to a string */
1574 String.prototype.addClass = function(theClass)
1575 {
1576      if (this != "")
1577      {
1578            if (!this.classExists(theClass))
1579            {
1580                  return this + " " + theClass;
1581            }
1582      }
1583      else
1584      {
1585            return theClass;
1586      }
1587
1588      return this;
1589 }
1590
1591
1592
1593
1594 /* Check if a class exists in a string */
1595 String.prototype.classExists = function(theClass)
1596 {
1597      var regString = "(^| )" + theClass + "\W*";
1598      var regExpression = new RegExp(regString);
1599
1600      if (regExpression.test(this))
1601      {
1602            return true;
1603      }
1604
1605      return false;
1606 }
1607
1608
1609
1610
1611 /* Check if a string is the nodeName of an accepted element */
1612 String.prototype.isAcceptedElementName = function()
1613 {
1614      var elementList = new Array("#text", "a", "em", "h1", "h2", "h3", "h4", "h5", "h6", "img", "li", "ol", "p",
     "strong", "ul");
1615      var theName = this.toLowerCase();
1616
1617      for (var i = 0; i < elementList.length; i++)
1618      {
1619            if (theName == elementList[i])
1620            {
1621                  return true;
1622            }
```

```
1623        }
1624
1625        return false;
1626  }
1627
1628
1629
1630
1631  /* Check if a string is the nodeName of an inline element */
1632  String.prototype.isInlineName = function()
1633  {
1634        var inlineList = new Array("#text", "a", "em", "font", "span", "strong", "u");
1635        var theName = this.toLowerCase();
1636
1637        for (var i = 0; i < inlineList.length; i++)
1638        {
1639              if (theName == inlineList[i])
1640              {
1641                    return true;
1642              }
1643        }
1644
1645        return false;
1646  }
1647
1648
1649
1650
1651  /* Remove a class from a string */
1652  String.prototype.removeClass = function(theClass)
1653  {
1654        var regString = "(^| )" + theClass + "\W*";
1655        var regExpression = new RegExp(regString);
1656
1657        return this.replace(regExpression, "");
1658  }
1659
1660
1661
1662
1663  /* Reverse a string */
1664  String.prototype.reverse = function()
1665  {
1666        var theString = "";
1667
1668        for (var i = this.length - 1; i >= 0; i--)
1669        {
1670              theString += this.charAt(i);
1671        }
1672
1673        return theString;
1674  }
1675
1676
1677
1678
1679  /* Make tags valid by converting uppercase element and attribute names to lowercase and quoting attributes */
1680  String.prototype.validTags = function()
1681  {
1682        var theString = this;
1683
1684        /* Replace uppercase element names with lowercase */
1685        theString = theString.replace(/<[^> ]*/g, function(match){return match.toLowerCase();});
1686
1687        /* Replace uppercase attribute names with lowercase */
1688        theString = theString.replace(/<[^>]*>/g, function(match)
1689              {
1690                    match = match.replace(/ [^=]+=/g, function(match2){return match2.toLowerCase();});
1691
```

```
1692                    return match;
1693               });
1694
1695          /* Put quotes around unquoted attributes */
1696          theString = theString.replace(/<[^>]*>/g, function(match)
1697               {
1698                    match = match.replace(/( [^=]+=)([^"][^ >]*)/g, "$1\"$2\"");
1699
1700                    return match;
1701               });
1702
1703          return theString;
1704     }
```