HTML Editor Library

Table of Contents

Foreword 0

Part I Introduction 5

1 Getting Started ... 5

Part II Using HTML Editor 6

1 Loading HTML into editor ... 6
Importing from other formats ... 6
2 Saving modified HTML ... 6
Export to PDF ... 7
Export to image ... 7
Export to plain text ... 8
3 Editor events ... 8
4 Editor commands ... 9
5 Spellchecking ... 10
6 Word prediction ... 10
7 Context toolbar ... 11
Selection toolbar ... 11
8 Enabling unicode for non-unicode Delphi ... 11
9 Adding elements and HTML blocks ... 11
10 Adding images ... 12
11 Working with lists ... 13
12 Working with images ... 13
13 Working with selection ... 14
14 Caret moving commands ... 14
15 Modifing text style ... 15
16 Keyboard shortcuts ... 15
17 Autoreplace ... 17
18 Text distortion and caret misplacement ... 17
19 Search and replace ... 17
20 Printing and print preview ... 18
21 Undo and redo ... 19
22 Pasting from clipboard ... 19
23 Working with tables ... 20
24 Change tracking ... 21
25 Creating table of contents ... 21
26 Header, Footer, Footnotes ... 22
27 VCL/FMX controls ... 23

Index 0

1 Introduction

1.1 Getting Started

To start using the editor simply drop THtmlEditor or TDBHtmlEditor component on a form. This will add htmledit unit for VCL or fmx.fhtmledit into uses list.
Editor document (of THtDocument type) can be accessed using Doc or Document properties.

Some editor methods/events use DOM element classes which are located in the following units:
THtDocument - htmldraw / fmx.fhtmldraw
TElement -- htmldraw / fmx.fhtmldraw
THtmlStyledNode - htmlcss
THtNode - htmlpars

THtmlEditor is THtPanel descendant and can be used for displaying HTML too. Set Readonly to true and torn off Editor.Options.eoCaretVisible option.

2 Using HTML Editor

2.1 Loading HTML into editor

To load new document into the editor assign Editor.HTML.Text property or use one of HTML: TStringList methods: LoadfromFile, LoadfromStream, etc.
Also Editor has the following methods:

```
procedure LoadFromFile(const FileName: string)
```
- Load document from file or http url depending on FileName prefix.

```
procedure LoadFromURL(const AURL: string)
```
- load from web

```
procedure LoadFromString(const s: hstring; AEncoding: THtmlEncoding = heDefault; ACodePage: integer = 0)
```
- load from string using specified encoding and code page.

```
procedure LoadfromResource(const AResourceName: string; AEncoding: THtmlEncoding)
```
- load from application resource.

2.1.1 Importing from other formats

Using **htrtf** unit **RTF** and **DOCX** files can be converted to HTML and loaded into Editor.

Both TRTFParser and TDOCXConverter has two class methods:

```
 class function ConvertFile(const FileName: string; PrepareImage: TRTFPrepareImageEvent = nil): string;
 class function ConvertStream(const Stream: TStream): string;
```

2.2 Saving modified HTML

To get modified HTML use Editor.Document.OuterHTML property.
Also Editor has the following methods for saving modified HTML:

```
 procedure SavetoFile(const FileName: string) - save to file using default encoding (Editor.Encoding)
```

THtDocument (Editor.Document) has the following methods:

```
procedure SaveToFileUTF8(const FileName: string)
- save to file using UTF8 encoding.
procedure SavetoFileUnicode(const FileName: string)
- save to file using UTF16 encoding
procedure SavetoFile(const FileName: string; AEncoding: THtmlEncoding = heUTF8)
```

```
  - save to file using specified encoding
 procedure SavetoLocalFile(const FileName: string; AEncoding: THtmlEncoding  = heUTF8)
  - save document and its external content (stylesheets, images) into local files. External links changed
 procedure SavetoStream(const AStream: TStream; AEncoding: THtmlEncoding = heUTF8)
  - save to stream using specified encoding.
```

2.2.1 Export to PDF

PDF export is available on all platforms, but on Windows it requires **Skia** or free **SynPDF** library. Please download this library, add its path into IDE library path list, uncomment {$DEFINE SYNPDF} in **htmlinc.inc** and add **htsynpdf** unit into uses list.

To export into PDF without creating **THtDocument** use the following class methods of **THtDocument**:

```
class function HTMLtoPDF(const AHTML: hstring; const AStyles: hstring =
''): TBytes;
class procedure HTMLtoPDFFile(const AHTML, FileName: string);
```

Normal THtDocument created for displaying cannot be used for PDF export. Create separate THtDocument using CanvasClass.PrintCanvasClass, prepare paged layout and then use Surface.G.SavetoPDFFile/SavetoPDFStream methods.
Example:

```
var D: THtDocument;
begin
  {$IFDEF FMX}
  D := THtDocument.Create(HtFMXDefaultCanvasClass.PrintCanvasClass);
  {$ELSE}
  D := THtDocument.Create(HtDefaultCanvasClass.PrintCanvasClass);
  {$ENDIF}
  try
    D.Parse(AHTML);
    D.GeneratePagesForPrint;
    D.Surface.G.SavetoPDF(FileName);
  finally
    D.Free
  end;
```

2.2.2 Export to image

HTML document can be rendered to bitmap or any other canvas. Example:

```
var D: THtDocument;
begin
  D := THtDocument.Create;
  try
    D.Parse(AHTML);
```

```
    D.Draw(Bitmap.Canvas, Rect(0, 0, Bitmap.Width, Bitmap.Height));
  finally
    D.Free
  end;
end;
```

2.2.3 Export to plain text

To get plain text of any DOM element or whole document use **TElement.InnerText** function. F.e.

```
s := Editor.Document.InnerText
```

2.3 Editor events

property `OnCaretMoved: TNotifyEvent` - Occurs when editor caret is moved into new position. Use Current property to get a current element and Caret.CurrentChar to get index of char at caret.
property `OnSpellCheck: TSpellCheckEvent` - Use this event for live spellchecking. **Editor.SpellChecking** should be set to true.
property `OnWordCorrection: TWordCorrectionEvent` - Use this event for live word correction (while typing)
property `OnChange: TNotifyEvent` - Occurs when the text for the editor may have changed.
property `AfterChange: TNotifyEvent` - Occurs after changes
property `AfterLoad: TNotifyEvent` - Occurs when new document is loaded into the editor.
property `AfterSave: TNotifyEvent` - Occurs when document is saved into a file.
property `OnAfterAction: TEditorActionEvent` – Fired after any editor action is executed
property `OnPrepareClipboardText: THtPrepareClipboardEvent` - Use this action for additional preprocessing of pasted HTML
property `OnPrepareClipboardImage: THtPrepareClipboardImageEvent` - Use this action for additional preprocessing of pasted image data
property `OnPrepareClipboardImageEx: THtPrepareClipboardImageExEvent` - Use this action for replacing default processing of pasted image data
property `OnURLDetected: TElementNotifyEvent` – Use this event to set additional attributes of created **<a>** element
property `OnCreateSelectionToolbar: TNotifyEvent` - Use this event to customize selection toolbar. When event is set, toolbar is not created by Editor.

Any control can subscribe to Editor.OnCaretMoving event by implementing IHtEditNotify interface

```
IHtEditNotify = interface
['{A48D7182-F878-4F89-963F-C1CB5E64D18A}']
```

```
    procedure OnCaretMoved(Sender: THtmlEditor);
  end;
```

and calling the following methods

```
  procedure Subscribe(Component: TComponent);
  procedure Unsubscribe(Component: TComponent);
```

2.4 Editor commands

All basic commands are accessible via actions. Place an ActionList on the form and use the New standard action command to add actions from the HtmlEdit group.
The available pre-defined actions are:

THtActionNew - new document
THtFileOpen,- open file.
THtFileSaveAs,- save file as.
THtActionCopy - copy to clipboard
THtActionPaste -paste from clipboard.
THtActionPasteImage - paste image from clipboard.
THtActionUndo - Undo.
THtActionFontBold - set font bold.
THtActionFontItalic - set font italic.
THtActionFontUnderline - set font underline.
THtActionFontStrikeout - set font strikeout.
THtActionSubscript - subscript.
THtActionSuperscript - superscript.
THtActionAlignLeft - set paragraph alignment to left.
THtActionAlignRight - set paragraph alignment to right.
THtActionAlignCenter - set paragraph alignment to center.
THtActionUnorderedList - convert selection to unordered list.
THtActionOrderedList - - convert selection to ordered list.
THtActionIncreaseIndent - increase block or list indent.
THtActionDecreaseIndent - decrease block or list indent.
THtActionAddUrl - convert selection to URL (link).
THtActionSetHeader- convert current block to header (header level are defined by ActionComponent tag).
THtActionMarkdownHighlight - perform Markdown conversion on selection
THtActionPascalHighlight - highlight selection as Pascal code
THtActionHTMLHighlight - highlight selection as HTML code

To control font name and size use THtFontCombo and THtFontSizeCombo components. Just place them on toolbar and set Editor property if there is more than one THtmlEditor component on form.

2.5 Spellchecking

You can enable/disable live spellchecking and autocorrection using the **Spellchecking** and **WordCorrection** properties.

Windows spellchecker

Add htspellwin to uses list.

Addict

To use Addict library enable $DEFINE ADDICT in **htmlinc.inc** before installing the package. Add **TAddictSpell** component on the form and set **THtmlEditor.AddictSpell** property. To start spellchecking add htAddict unit and call `HtAddictCheckEditor(E, TCheckType.ctAll, AddictSpell1) method.`

Hunspell

Add **hthubspell** unit and use **THtNHunSpellChecker** class.

DevExpress

Add **htdxspell** unit and use **TdxSpellCheckerHtmlEditorAdapter** class.

Other spellcheckers

To use another spellchecking library write handlers for the **OnSpellCheck** and **OnWordCorrection** events.

2.6 Word prediction

To enable word predition add hteditcore and htpredict to uses list and create global predictor instance using the following code:

```
GlobalWordPredictor := THtWordPredictor.Create;
```

Predictor can be trained by parsing plain text:

```
GlobalWordPredictor.Parse(Text).;
```

After parsing, prepared data can be saved to file

```
GlobalWordPredictor.SavetoFile('predict.dat');
```

To load prepared data use

```
GlobalWordPredictor.LoadfromFile('predict.dat');
```

When typing, use Tab key to move to next word (Wanttabs should be set to true).

2.7 Context toolbar

Context toolbar is located at bottom of editor window. Context toolbar actions are similar to Selection toolbar actions and are located in hteditcontext unit.
To define custom action set use **property** `OnCreateContextToolbar: TNotifyEvent.` F.e.

```
THtEditContextPanel(ContextToolbar).Add(THtEditContexBold).Add(THtEditConte
xtItalic).Add(THtEditContextUnderline));
```

Context toolbar can be enabled/disabled by setting Editor.Options.eoContextToolbar property

2.7.1 Selection toolbar

Selection toolbar is located on a popup panel activated when uses select text block. Selection toolbar actions are located in hteditcontext unit.
To define custom action set use **property** `OnCreateSelectionToolbar: TNotifyEvent.` F.e.

```
THtEditContextPanel(SelectionToolbar).Add(THtEditContexBold).Add(THtEditCon
textItalic).Add(THtEditContextUnderline));
```

To hide selection toolbar call **Editor.FinalizeEditing.**

Selection toolbar can be enabled/disabled by setting Editor.Options.eoSelectionToolbar property

2.8 Enabling unicode for non-unicode Delphi

To use Unicode in old Delphi you should have TntUnicode library installed.

Open htmlinc.inc file, uncomment $WIDESTRINGS define and recompile library package.

2.9 Adding elements and HTML blocks

There are functions and procedures to add various types of objects into the current document.

procedure AddChar(Key: char);

Add one char at caret position.

procedure AddString(const Str: hstring);

Add string at caret position.

procedure AddHTMLAtCursor(const HTML: hstring);

Add HTML at caret position. If HTML contains block elements current block element will be split.

function AddPara: TElement;

Add paragraph at caret position.

function AddHR: TElement;

Add horizontal divider at caret position.

function AddLineBreak(AddAfter: TElement=nil): TElement;

Add line break at cursor or after AddAfter element.

function AddImageAtCursor(const ImageData: TBytes; Url: string=''): TImageElement;

Add embedded image at caret position.

function AddImageAtCursor(const Url: string; AWidth: integer=0; AHeight: integer=0; const Align: string=''): TImageElement;

Add image and set its alignment (left/right)

2.10 Adding images

To add image to document use
function AddImageAtCursor(const Url: string; AWidth: integer=0; AHeight: integer=0; const Align: string=''): TImageElement;

To set image alignment use Align parameter - set it to 'right' or 'left'.

To embed image to document use

function AddImageAtCursor(const ImageData: TBytes; Url: string=''): TImageElement;

2.11 Working with lists

procedure SetListStyle(const ListStyle: string);

Convert the current block to a list. Set ListStyle to ul for unordered list or ol for ordered list.

procedure UnListSelection(const NewTag: string='p');

Convert the current list to paragraph.

procedure IncreaseIdent;

Increase the indent of the current list item (create sublist)

procedure DecreaseIdent;

Decrease the indent of the current list item or remove the list style.

2.12 Working with images

To enable image moving and resizing add the following code into Editor.Styles

```
img, svg {draggable: true; resize: both }
```

Following Editor,Options are related to images:

eoEmbedDroppedImages - add dropped images in inline format,
eoEmbedPastedImages - add pasted images in inline format,
eoProportionalImageResize - enable/disable image distortion

Following properties affect image loading mode:

BackgroundImageLoading - load images in a separate thread
LazyImageLoading - load only visible images
WebLoading - enable web (http) images

For custom image loading code use

```
property OnGetImage: TGetImageEvent;
```

Other events related to images:
property OnAfterImageLoaded: TAfterImageLoaded - called after image is loaded
property OnImageLoadFailed: TAfterImageLoaded - called when image loading was failed

2.13 Working with selection

```
procedure SelectWordAtCursor;
```

Select word at caret position.

```
procedure DeleteSelection(DeleteEmptyElements: boolean=true);
```

Delete selection. DeleteEmptyElements=true means that empty block or inline element in selection will be deleted.

```
procedure TagSelection(const Tag: string; attributes: string='');
```

Wrap selected elements by tag. Additional attributes could be added to tag.

```
procedure UntagSelection(const Tag: string; constattributes: string='');
```

Remove parent tag from selected elements.

```
procedure Current.SelectPara;
```

Select paragraph at caret position.

2.14 Caret moving commands

There is a comprehensive suite of methods for managing caret movement and positioning:

```
procedure CaretStart;
procedure CaretEnd;
procedure CaretLineStart;
procedure CaretLineEnd;
function  CaretNext: boolean;
function  CaretPrevious: boolean;
procedure CaretNextWord;
procedure CaretPreviousWord;
procedure CaretNextCell;
procedure CaretPreviousCell;
```

```
procedure CaretFirstCell;
procedure CaretLastCell;
function  CaretDown: boolean;
function  CaretUp: boolean;
procedure CaretPageUp;
procedure CaretPageDown;
procedure CarettoStartof(E: TElement);
procedure CarettoEndof(E: TElement);
procedure CaretParaStart;
procedure CaretParaEnd;
```

2.15 Modifing text style

Many text style functions are accessible via the THtmlEditor.TextStyle class.

It has the following properties

```
property Bold: boolean;
property Italic: boolean;
property Underline: boolean;
property StrikeOut: boolean;
property Subscript: boolean;
property SuperScript: boolean;
property FontName: string;
property FontSize: integer;
property Color: cardinal;
property BGColor: cardinal;
property TextTransform: TCSSTextTransform;
property Alignment: THAlignment;
```

Changing these properties will change style of current selection, or current word at cursor (if nothing is selected) or style of subsequent text entered by user.

2.16 Keyboard shortcuts

Ctrl+Left/Right
  Next/Previous word
Ctrl+Up/Down
  Paragraph start/End or Next/Previous
Ctrl+Home/End
  Start/End of document
Ctrl+Alt+1..5

Header 1..5
Ctrl+1, 2, 5
Line spacing 1, 2, 1.5
Ctrl+Shift+A
Upper case
Ctrl+Shift+K
Lower case
Ctrl+B
Bold
Ctrl+I
Italic
Ctrl+U
Underline
Ctrl+E
Center alignment
Ctrl+L
Left alignment
Ctrl+R
Right alignment
Ctrl+C/Ctrl+Ins
Copy to clipboard
Ctrl+V/Shift+Ins
Paste from clipboard
Ctrl+Z
Undo
Ctrl+M
Increase indent
Ctrl+Shift+M
Decrease indent
Ctrl+Alt+C
©
Ctrl+Alt+R
®
Ctrl+Alt+T

Ctrl+Alt+-

Ctrl+-

Ctrl+Shift++
Superscript
Ctrl++

Subscript
Shift+Enter
Soft line-break

2.17 Autoreplace

Following sequences will be replaced

(c) - ©
(tm) -
(r) - ®
-
* at line start - unordered list
1. at line start- orderd list
-- -
---+Enter - horizontal divider

2.18 Text distortion and caret misplacement

Using HTML Editor 6

GDI+ (default VCL canvas) has some issues with text displaying and measurement. In most cases it is better to use **Direct2D** canvas. To enable it simply add **htcanvasdx** to uses list (VCL).

FMX default canvas also has problems with precise text measurement. You can use native canvases by assigning **HtFMXDefaultCanvasClass** global variable (**htcanvas** unit).
OSX: THtCanvasOSX (fmx.htcanvasosx unit)
Android: THtCanvasAndroid (fmx.htcanvasandroid)
iOS: THtCavasiOS (fmx.htCanvasiOS)

2.19 Search and replace

VCL version has the following build-in dialogs:

```
procedure FindDialog(ADialog: TFindDialog = nil);
procedure ReplaceDialog(ADialog: TReplaceDialog = nil);
```

Also Editor has build-in method for creating "Search page". Use THtmlEditor.CreateSearchResult(const s: hstring): hstring function to get search result and HtPanel to display it.

Recommended CSS is:

```
body {
 background: white ;
 Color: #202020;
 font-size: 10pt;
 font-face: Calibri;
 padding: 5 5;
}
a {color: #202020; text-decoration: none;}
a:hover {text-decoration: underline}
h3 {text-align: center; color: #aaa; margin: 10 0 5 0; border-bottom: solid
#aaa 1px}
span {color: #A09000; font-weight: bold}
```

Navigation code:

```
procedure TForm1.SearchPanelUrlClick(Sender: TElement);
var EX: TElement;
    n, k: integer;
begin
  if not TrystrtoInt(Sender['pos'], n) then
    exit;
  if not TrystrtoInt(Sender['charpos'], k) then
    k := 1;
  EX := Editor.GetElementbyAbsolutePosition(n, true);
  if Assigned(EX) then
  begin
    Editor.Doc.Selection.StartElement := EX;
    Editor.Doc.Selection.EndElement := EX;
    Editor.Doc.Selection.StartPos := k;
    Editor.Doc.Selection.EndPos := k + length(SearchEdit.Text);
    Editor.ScrollSelectionIntoView;
    Editor.Repaint;
  end;
end;
```

2.20 Printing and print preview

To display print preview window add HtPreviewFrame for VCL or fmx.HtPreviewFrame for FMX unit into uses list and call

```
ShowHtPrintPreview(Editor.Doc, Editor.Styles.Text)
```

(Requires HTML Report Library).

2.21 Undo and redo

function CanUndo: boolean - Returns true if undo history is not empty
procedure Undo - Undo last operation
function CanRedo: boolean - Returns true if redo history is not empty
procedure Redo - Redo last operation
procedure DocChanged(**const** TopElement: TElement = **nil**; AOperation: THtEditorOperation = eopOther; ClearRedo: boolean = true) - Use this method to notify the editor of any changes in document.

2.22 Pasting from clipboard

Clipboard content can be preprocessed before passing to editor. For HTML content use the following event:

property OnPrepareClipboardText: THtPrepareClipboardEvent

THtPrepareClipboardEvent = procedure(var ClipboardHTML: hstring) of object;

To prepare image content use the following events:

property OnPrepareClipboardImage: THtPrepareClipboardImageEvent
property OnPrepareClipboardImageEx: THtPrepareClipboardImageExEvent

In OnPrepareClipboardImageEx event Handled parameter can be used for skipping default processing.

Following Editor.Options are related to clipboard:

eoEmbedPastedImages - embed pasted images in <img src="data.." format.
eoEmbedPastedStyles - add styles from pasted HTML to document styles
eoClearPastedFormatting - remove inline styles and classes from pasted elements, also remove span elements without attributes
eoDisableBlockJoinOnPaste - do not join pasted blocks with current
eoPasteTextBlockAsPara - convert text blocks divided by blank lines to para.

Editor methods for pasting:

PasteFromClipboard - paste any content from clipboard (image, RTF, HTML, text, etc).
PasteTextFromClipboard - paste plain text only.

2.23 Working with tables

To add table at caret position use

```
function AddTable(Cols, Rows: integer; const Attributes, CellContent:
hstring; HeaderRows: integer = 0): TElement;
```

Parameters:
Cols: Column count
Rows: Row count
Attributes: Additional table attributes
CellContent: Default content for the cells (for example)

Table can be adjusted from code using the following methods of TTableElement:

`procedure InsertRow(NewRowIndex: integer; const CellContent: string)` - Insert new row
`function InsertCol(NewColIndex: integer; const CellContent: string; const AWidth: string = '100'): TElement;` - Insert column
`procedure DeleteRow(RowIndex: integer)` - Delete row
`procedure DeleteCol(ColIndex: integer)` - Delete column
`procedure SetColumnStyle(ColumnIndex: integer; const AStyle: string)` - Set style for column
`procedure MoveColumn(Index, NewIndex: integer)` - Exchange columns

Tables can have add col/row marks when mouse is between columns.rows. To enable it set **Editor.Options.eoAddColMarks/eoAddRowMarks.**

To enable column resizing add the following code into **Editor.Styles**:

```
th, td {resize: horizontal}
```

To enable row resizing add the following code into **Editor.Styles**:

```
tr {resize: vertical}
```

To merge table cells use the following Editor methods:

```
procedure MergeSelectedCells;
procedure UnMergeSelectedCell;
```

To delete row from table call

```
procedure DeleteTableRow(const Table: TTableElement; Row: integer);
```

Following Editor action are related to tables:

- THtTableAddRow
- THtTableDeleteRow
- THtTableInsertCol
- THtTableDeleteCol
- THtTableIncColspan
- THtTableDecColspan

2.24 Change tracking

To enable changer tracking set Editor.TrackChanges property to true.
Editor ha the following methods related to change tracking:

function ChangesReport: hstring – Generate tracked changes report
procedure ApplyAllChanges - Apply all changes made in TrackChanges mode

2.25 Creating table of contents

To create table of contents use **Editor.CreateTableofContents** method.
Following styles are recommended for displaying TOC:

```
body {background: white;
  font-family: Verdana; font-size: 9pt; color: #305090; padding: 3px 3px;}
h1,h2,h3,h4,h5 {white-space: nowrap; cursor: pointer;  }
h1:hover, h2:hover, h3:hover, h4:hover, h5:hover {text-decoration:
underline}
h1 {font-size: 14pt}
h2 {margin-left: 10px; font-size: 12pt}
h3 {margin-left: 20px; font-size: 10pt}
h4 {margin-left: 30px; font-size: 9pt}
h5 {margin-left: 40px; font-size: 8pt}
```

To navigate editor to selected item use the following code:

```
procedure TForm1.TOCPanelElementClick(Sender: TElement);
var s: string;
    EX: TElement;
begin
  s := Sender['content'];
  if s <> '' then
  begin
    EX := Editor.Doc;
    while Assigned(EX) do
    begin
      if (EX.Tag = Sender.Tag) and (EX.InnerText = s) then
      begin
        Editor.ScrollIntoTop(EX);
```

```
        exit
      end;
      EX := EX.NextElement;
    end;
  end;
end;
```

2.26 Header, Footer, Footnotes

Page header and footer can be defined using any block element and CSS style position: running(heading/footer). Example:

```
<style>
footer {position: running(footer); text-align: center; border-top: solid
#aaa 1px; text-align: center; font-size: 9pt; left: 5%; width: 90%; height:
36px }
.pagecount:before {content: counter(pagecount)}
</style>

<footer>Page <span class="page"></span>/<span
class="pagecount"></span></footer>
```

This sample also contains page number and page count which are defined using content property.
Footer size is calculated automatically from page footer block height.

Document can contain several headers/footers. CSS property **page** is used for selecting on which page header/footer will be displayed.
page can have following values:

last: only last lage
last-: all pages before last
odd: odd pages
even: even pages
<**number**> (1,2,..): only <number> page
<**number+**> all pages after <number>

To set page margins use CSS style:

```
@page {margin: 0px 0px 0px 0px}
```

Library supports document section element **doc-section** which allows to set footer and headers for document part (similarly to OOXML) .
Attrributes:
size: page size
margins: page margins

section can have several child **header** or **footer** elements with following attributes
ref ID of header or footer
style optional style to define page, f.e. **page:1** or **page:odd**

2.27 VCL/FMX controls

Any control class used in HTML should be registered first using RegisterClass or RegisterClasses, f.e.

```
RegisterClass(TCalendar);
```

Sample of control inside HTML:

```
<control type="TCalendar" name="Calendar1" >
</control>
```

There are two ways of defining control properties. First, using attributes:

```
<control type="TCalendar" rotationangle="10">
</control>
```

Second - using DFM object notation:

```
<control type="TCalendar" name="Calendar1">
object Calendar1: TCalendar
   Date = 44113.000000000000000000
end
</control>
```