<body> attributes

ThtmlLite supports color and background information attributes entered with the HTML **<body>** tag. Thus each document may have a customized appearance. Color and background information entered in this manner overrides the default information specified by the DefFontColor, DefHotSpotColor, DefBackground DefVisitedLinkColor, and DefOverLinkColor properties.

The following **<body>** attributes are supported in this version:

Text="RRGGBB"

The hex number entered determines the normal text color.

Link="RRGGBB"

The hex number entered determines the link (HotSpot) color.

VLink="RRGGBB"

The hex number entered determines the color of a link which has been previously visited.

OLink="RRGGBB"

The hex number entered determines the link color when the mouse passes over it. A defined OLink value will make the links active even if htOverLinksActive is not set..

BgColor="RRGGBB"

The hex number entered determines the background color.

Background="imagefilename"

The Background attribute specifies an image file. When ViewImages is set, this image will be tiled to form the background of the document.

HTML hex entries are 2 digits of red, 2 digits of green, and 2 digits of blue. Note that this order is the reverse of normal color entries in Windows. Also, most HTML browsers require all six digits including leading zeros.

Example

```
<body text="00FF00" link="FF0000"
   bgcolor="000000" background="marble.gif">
```

produces green text, red hotspots on a black background. If ViewImages is set, `marble.gif` will be tiled to form the background.

Base Property

Description

```
property Base: string;
```

Run time only. The Base property returns the document's base directory as given in the **<base>** tag.

The Base property may also be assigned. In order for this to be effective, the assignment must be made before the document is loaded. A **<base>** tag in the document will override an assignment.

See Also

HTMLExpandFilename method

BorderStyle Property

Declaration

```
property BorderStyle: THTMLBorderStyle;
```

The BorderStyle property works slightly different from that for most Delphi components. BorderStyle may be assigned the following values:

htFocused	The border will appear when the viewer has the focus.
htNone	Never any border.
htSingle	A single line border will always be present.

CaretPos Property

Declaration

```
property CaretPos: LongInt;
```

The CaretPos property indicates the character position of the caret marker.

Setting CaretPos equal to zero will insure that text searches start from the beginning of the document.

The caret position may be made visible by setting htShowDummyCaret in the htOptions property.

See Also:

SelStart Property

SelLength Property

DisplayPosToXY Method

Charset Property

Applies to

Delphi 3, 4, and 5

Declaration

```
property Charset: TFontCharset;
```

The **Charset** property specifies which character set is to be used. Before changing the character set to anything but DEFAULT_CHARSET, be sure the proper fonts are installed.

Character sets which read right to left are currently not supported.

Clear Method

Declaration

```
procedure Clear;
```

Description

The Clear method clears the current document from the view window and deallocates the memory used.

ClearHistory Method

Declaration

```
procedure ClearHistory;
```

Description

The ClearHistory method clears the History List.

For additional information on setting up a history list, see Setting up a History List.

ThtmlLite Component, Version 7.25a

The ThtmlLite component provides an easy way to display HTML documents in an Inprise (Borland) Delphi program.

ThtmlLite is similar to to the Professional **ThtmlViewer**, **TFrameViewer, and TFrameBrowser** components but has reduced capabilities. It is designed for hobbyists, students, and other casual users. There is no charge for its use.

Version 7 of this component supports most of the HTML 3.2 specifications with many additional popular HTML enhancements:

- Animated GIFs
- Images (Bitmap and GIF. Also JPEG in the Delphi 3, 4, and 5 versions)
- Transparent images
- Image caching
- Left and Right floating images
- IsMap support for images used as anchors
- Client side image maps (UseMap)
- Image sizing through Height and Width attributes
- Large HTML files
- HTML Tables
- HTML Forms
- Font sizes, styles, and colors with HTML tags or default settings
- Background colors and images
- Text search
- Copy to clipboard
- Subscripts and Superscripts

The following additional features can be found in the Professional version of the **HTML Components**:

- Cascading Stylesheet support
- Support for Frames
- Formated Printing of HTML documents
- Source code supplied
- JPEG images in all Delphi and C++Builder versions
- PNG image support

Updates

For the most recent versions of the HTML components and for information on what's coming, check in occasionally at:

http://www.pbear.com/

Email

dbaldwin@pbear.com

CopyToClipboard Method

Declaration

```
procedure CopyToClipboard;
```

Description

The CopyToClipboard method copies the selected text to the clipboard.

See also

SelectAll Method

SelLength Property

SelText Property

GetSelTextBuf Method

CurrentFile Property

Declaration

```
property CurrentFile: String;
```

Read only and run time only. The CurrentFile property returns the full path and filename of the file currently loaded. If no file is loaded or the load was accomplished using the LoadFromBuffer method, CurrentFile will be an empty string.

DefBackground Property

Declaration

```
property DefBackground: TColor;
```

Description

DefBackground sets the viewer's default background color. This is the background color when no background information is present in the HTML document file. Either clBtnFace or clWindow are appropriate values.

See Also

<body> attributes

DefFontColor Property

Declaration

```
property DefFontColor: TColor;
```

Description

DefFontColor sets the viewer's default text color. This is the color used when no text attribute information is present in the HTML document file.

See Also

<body> attributes

DefFontName Property

Declaration

```
property DefFontName String;
```

Description

DefFontName sets the viewer's font name.

DefFontSize Property

Declaration

```
property DefFontSize: Integer;
```

Description

DefFontSize sets the default font size for the viewer's normal text. The value entered is the point size. Headings, spacing, and other various sizes are scaled appropriately.

DefHotSpotColor Property

Declaration

```
property DefHotSpotColor: TColor;
```

Description

DefHotSpotColor sets the viewer's default color for links. This is the color used if the link color is not otherwise specified in the HTML document file.

See Also

<body> attributes

DefPreFontName Property

Declaration

```
property DefPreFontName String;
```

Description

DefPreFontName sets the viewer's font name for fonts used with the <pre>, <code>, and similar tags. Normally a non-proportional font should be selected.

DocumentTitle Property

Declaration

```
property DocumentTitle: String;
```

Read only and run time only. The DocumentTitle property returns the document's title as found between the **<title>** tags.

Find Method

Declaration

```
function Find(const S: String; MatchCase: boolean): boolean;
```

Description

The Find method searches through the HTML document for the string, S, returning True if the string is found. If MatchCase is True, a case sensitive search is made.

Text is searched starting at the current cursor position. As each match is found, the cursor is placed at the end of the matched string.

Note that in the present version of ThtmlLite, the cursor is not visible. However, it's possible to place the cursor at any position by left clicking with the mouse.

FormControlList Property

Declaration

```
property FormControlList: TList;
```

Read only and run time only. The FormControlList contains a list of all the Form controls contained in the currently displayed document. This list enables the programmer to access the controls to read or initialize them while displayed on screen. Some hints on using this list:

- Declare **htmlsubs.pas** in your uses clause.
- Study the definition of TFormControlObj and its decendents (in htmlsubs.pas) as you will probably need to access the fields and methods to make use of the FormControlList.
- Give any control you want to access a distinctive Name (the **Name=** item) when defining it in the HTML document. This name can be used to identify the control when searching through the list.
- The FormControlList is used by ThtmlLite. Do not **Free** this list.

Examples:

In the code below, clicking Button4 causes the first form in the document to be submitted. (**Viewer** is the ThtmlLite displaying the document.)

```
procedure TForm1.Button4Click(Sender: TObject);
begin
if Viewer.FormControlList.Count > 0 then
  TFormControlObj(Viewer.FormControlList.Items[0]).MyForm.SubmitTheForm('');
end;
```

In the following, clicking Button5 initializes the TEdit control with the name, "Address"

```
procedure TForm1.Button5Click(Sender: TObject);
var
  I: integer;
begin
with Viewer.FormControlList do
  for I := 0 to Count -1 do
    if CompareText(TFormControlObj(Items[I]).Name^, 'Address') = 0 then
      begin
      if TFormControlObj(Items[I]).TheControl is TEdit then
        TEdit(TFormControlObj(Items[I]).TheControl).Text := '144 Sands
Point';
      Exit;
      end;
end;
```

GetSelTextBuf Method

Declaration

```
function GetSelTextBuf(Buffer: PChar, BuffSize: LongInt): LongInt;
```

Description

The GetSelTextBuf method copies the selected text into the buffer pointed to by Buffer, up to a maximum of BufSize characters, and returns the number of characters copied.

See also

SelText Property

SelLength Property

Important:

The use of GIF Images in a commercial product may necessitate royalty payments to Unisys Corporation. For further information, contact:

Mark T. Starr, Esq
Internet: StarrMT@po4.bb.unisys.com
Unisys Corporation - MS/C2SW1
PO Box 500
Blue Bell, PA 19424

HTMLExpandFilename Method

Declaration

```
function HTMLExpandFilename(const Filename: string): string;
```

Description

HTMLExpandFilename converts an HTML style file URL to a Dos filename and adds an appropriate path. The path added depends on:

1. If the URL starts with a **'/'** or **'\'**, the path defined by the ServerRoot property is used.
2. If the <base> tag specifies "DosPath", then no path is added. Files are then searched for in the usual Dos/Windows manner.
3. If the <base> tag specifies another path then that path will be added. File locations are then relative to the base path.
4. If there is no <base> tag, then the path of the current HTML file is used. File locations are then relative to that directory.

See also

Base Property

HTMLToDos Function

ServerRoot property

HTMLToDos Function

Syntax

```
function HTMLToDos(FileName: String): string;
```

Description

The HTMLToDos function converts an HTML style filename to one for Dos. If the filename is in Dos format to begin with, no change takes place.

Example

After the call,

```
S := HTMLToDos('file:///c|/Big/Images/Glass.bmp');
```

S will contain 'c:\Big\Images\Glass.bmp'.

History Property

Declaration

```
property History: TStrings;
```

Read only and run time only. The History property contains a list of the most recently loaded document filenames with the current filename at index 0.

For additional information on setting up a history list, see Setting up a History List.

HistoryIndex Property

Declaration

```
property HistoryIndex: Integer;
```

Run Time only. The HistoryIndex property shows which history item represents the currently loaded file and file position. Setting HistoryIndex to a new value automatically causes the loading and positioning of the applicable file.

For further information on setting up a History list, see Setting up a History List.

HistoryMaxCount Property

Declaration

```
property HistoryMaxCount: Integer;
```

The HistoryMaxCount property determines the number of History items which will be maintained. If HistoryMaxCount is 0, then no history list will be maintained.

For further information on setting up a History list, see Setting up a History List.

ImageCacheCount Property

Declaration

```
property ImageCacheCount: Integer;
```

To save time in reloading HTML files with images, image caching may be used to save images already loaded. The ImageCacheCount property determines the extent to which this is done.

With an ImageCacheCount of 5, for instance, images will be saved through 5 document reloads. If the image is not used during that period, it will be dropped from the cache. However, anytime an image is used, the count is reset to zero and it will remain for 5 more reloads.

The default value for ImageCacheCount is 5. A value of 0 indicates no caching. The cache may be flushed by setting the count to 0 temporarily and then resetting it to the desired value.

Since image storage can use a lot of memory, care should be taken not to overdo the amount of caching.

InsertImage Method

Declaration

```
function InsertImage(const Src: string; var Stream: TMemoryStream): boolean;
```

Description

The **InsertImage** method allows images previously requested by the OnImageRequest event to be loaded at a later time. This allows images to be downloaded and inserted when available.

Src	The identifier for the image originally obtained from the OnImageRequest event
Stream	The TMemoryStream being returned containing the image. *Stream* may contain a bitmap, GIF, JPEG, or PNG image.
Return	A **True** result indicates the image was accepted. A return of **False** indicates the image cannot be accepted at this time because something else in being processed. Another attempt should be made later.

A typical scenario for using the InsertImage method is as follows:

- The HTML file is loaded. As this is processed, a series of OnImageRequest events occur.
- The special **WaitStream** value is returned to the OnImageRequest event. This indicates that the image will be provided later. The *Src* and *Sender* parameters are saved.
- The images are then downloaded. Meanwhile the HTML file is displayed with default images.
- As the image information is obtained, the **InsertImage** method is called and the image is displayed.

See also:

OnImageRequest Event

LoadFromBuffer Method

Declaration

```
procedure LoadFromBuffer(Buffer: PChar; BufSize: LongInt);
```

Description

The LoadFromBuffer method loads an HTML document from a memory buffer. BufSize is the number of characters to be loaded. A terminating null character is not required.

For the 16 bit version, LoadFromBuffer is limited to 64k characters.

Example

Assume an HTML document exists in a TMemo object. The following code could be used to then copy the document to an ThtmlLite object:

```
var
  Buffer: PChar;
  Size: integer;
begin
Size := Memo1.GetTextLen;
Inc(Size);
GetMem(Buffer, Size);
Memo1.GetTextBuf(Buffer,Size);
Viewer.LoadFromBuffer(Buffer, Size-1);
FreeMem(Buffer, Size);
end;
```

See also:

- LoadFromFile Method
- LoadStrings Method
- LoadFromStream Method
- LoadFromString Method

LoadFromFile Method

Declaration

```
procedure LoadFromFile(const Filename: String);
```

Description

The LoadFromFile method loads an HTML document for viewing, where Filename is the name of the file to be loaded. If Filename has no extension, .HTML is assumed.

Filename may optionally contain a document position identifier separated from the actual filename by a **#** sign.

Example

```
Viewer.LoadFromFile('MyHTML#Contents');
```

would load MyHTML.htm and position the view to the location where the "Contents" target was defined.

See also:

LoadStrings Method

LoadFromBuffer Method

LoadFromStream Method

LoadFromString Method

LoadFromStream Method

Declaration

```
procedure LoadFromStream(AStream: TStream);
```

Description

The LoadFromStream method loads an HTML document from a stream.

LoadFromStream is **not** limited to 64k characters.

See also:

- LoadFromFile Method
- LoadStrings Method
- LoadFromBuffer Method
- LoadFromString Method

LoadImageFile Method

Declaration

```
procedure LoadImageFile(const Filename: String);
```

Description

The LoadImageFile method loads a Bitmap, GIF, JPEG, or PNG image for viewing. Filename is the name of the image file to be loaded.

See also:

LoadStrings Method

LoadFromBuffer Method

LoadFromStream Method

LoadTextFile Method

LoadStrings Method

Declaration

```
procedure LoadStrings(Strings: TStrings);
```

Description

The LoadStrings method loads an HTML document from a TStrings object.

Example

Assume an HTML document exists in a TMemo object. The following line of code could be used to then copy the document to an ThtmlLite object:

```
Viewer.LoadStrings(Memo1.Lines);
```

See also:

LoadFromFile Method

LoadFromStream Method

LoadStrings Method

LoadFromString Method

LoadTextFile Method

Declaration

```
procedure LoadTextFile(const Filename: String);
```

Description

The LoadTextFile method loads a file for viewing treating it as an ASCII text file. If the file is an HTML file, it will be displayed as text showing the HTML tags, etc.

See also:

LoadTextStrings Method

LoadImageFile Method

LoadFromFile Method

LoadTextStrings Method

Declaration

```
procedure LoadTextStrings(Strings: TStrings);
```

Description

The LoadTextStrings method loads a document from a TStrings object treating it as ASCII text.

See also:

LoadTextFile Method

LoadStrings Method

LoadImageFile Method

MarginHeight and MarginWidth Properties

Declaration

```
property MarginHeight: Integer;
property MarginWidth: Integer;
```

These two properties determine the margins around the ThtmlLite display. Values are in pixels.

The default vertical margin is 5 and the default horizontal margin is 10.

NameList Property

Declaration

```
property NameList: TStringList;
```

Read only and run time only. NameList contains a list of all the anchor names (the **Name=** attribute of the **<A>** tag) contained in the currently displayed document.

NameList is used by ThtmlLite. Do not **Free** this list.

NoSelect Property

Declaration

```
property NoSelect: Boolean;
```

Description

Set the NoSelect property to True if you wish to prevent the user from selecting text with the mouse. The default value is False.

OnFormSubmit Event

Declaration

```
property OnFormSubmit: TFormSubmitEvent;
```

Description

The OnFormSubmit event occurs when an HTML Form's submit button is depressed. See the Type TFormSubmitEvent description for details on the passed parameters.

OnHistoryChange Event

Declaration

```
property OnHistoryChange: TNotifyEvent;
```

Description

The OnHistoryChange event occurs whenever the history list changes. For further information on history lists, see Setting up a History List.

OnHotSpotClick Event

Declaration

```
property OnHotSpotClick: THotSpotClickEvent;
```

Description

The OnHotSpotClick event occurs when the mouse is over an hypertext link and the left mouse button is clicked. See the THotSpotClickEvent type for information on the parameters.

Example

The following code may be used to load the hypertext link URL into an Edit box when a mouse click occurs:

```
Viewer.OnHotSpotClick := HotSpotClick;
....
procedure TForm1.HotSpotClick(Sender: TObject; const URL: String;
           var Handled: Boolean);
begin
Handled := False;
Edit1.Text := HTMLToDos(URL);
end;
```

OnHotSpotCovered Event

Declaration

```
property OnHotSpotCovered: THotSpotEvent;
```

Description

The OnSpotCovered event occurs when the mouse is moved over or away from an hypertext link. This event may be used to display the destination URL of a hypertext link to allow the user to decide if it should be selected.

Example

The following code may be used to provide a status line display of the hypertext link at the mouse position:

```
Viewer.OnHotSpotCovered := HotSpotChange;
...
procedure TForm1.HotSpotChange(Sender: TObject; const URL: string);
begin
Panel1.Caption := URL;
end;
```

OnImageClick Event

Declaration

```
property OnImageClick: TImageClickEvent;
```

Description

The OnImageClick event occurs when the mouse is over an image and either mouse button is clicked. See the TImageClickEvent type for information on the parameters.

Example

The following code may be used to popup a menu when right clicking on an image:

```
procedure TForm1.ImageClick(Sender, Obj: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  Pt: TPoint;
begin
if Button = mbRight then
  if (Obj is TImageObj) then
    begin
    FoundObject := TImageObj(Obj);
    GetCursorPos(Pt);
    PopupMenu.Popup(Pt.X, Pt.Y);
    end;
end;
```

Here *FoundObject* is a *TForm1* field used to save the calling *TImageObj* for use after the menu selection has been made. See the TImageObj type definition in HTMLSUBS.PAS for further imformation on this type. In particular, the field, *TImageObj.Source*, holds the image's filename, and the property, *TImageObj.Bitmap*, can be used to obtain a copy of the bitmap itself.

See also:

OnImageOver Event

OnObjectClick Event

OnImageOver Event

Declaration

```
property OnImageOver: TImageOverEvent;
```

Description

The OnImageOver event occurs when the mouse is moved over an image. See the TImageOverEvent type for information on the parameters.

See also:

OnImageClick Event

OnImageRequest Event

Declaration

```
property OnImageRequest: TGetImageEvent;
```

Description

The OnImageRequest event occurs when ThtmlLite encounters an image request and the ViewImages property is set. You can use the OnImageRequest event to supply an image stored in stream form. The stream may contain a Bitmap, GIF (including animated), JPEG, or PNG image.

The image may also be supplied at a later time if it is necessary to download it. To indicate the image will be supplied later, return the special TMemoryStream value, **WaitStream**, rather that an actual stream. The InsertImage method may be used when the image is available.

Usage Note

The TMemoryStream returned is not freed by ThtmlLite.

Example

This event handler converts an image file to a TMemoryStream object for test purposes.

```
var
  MStream: TMemoryStream;
......
procedure TForm1.ImageRequest(Sender: TObject; const SRC: string;
          var Stream: TMemoryStream);
var
  Filename: string[80];
begin
Stream := Nil;        {in case of error}
Filename := FrameViewer.HTMLExpandFilename(SRC);
if FileExists(Filename) then
  try
    if not Assigned(MStream) then
      MStream := TMemoryStream.Create;
    MStream.LoadFromFile(FileName);
    Stream := MStream;
  except
    end;
end;
```

See Also

Type TGetImageEvent

InsertImage Method

OnInclude Event

Declaration

```
property OnInclude: TIncludeType;
```

Description

The OnInclude event allows text to be inserted in an HTML document at load time. Within the HTML document, the OnInclude event is generated when server side include syntax is encountered. See the TIncludeType for more information on the parameters and on the server side include syntax.

Example

The OnInclude handler below could be used to fill in the current date in an HTML document. Note that the string, S, cannpt be a local variable as it must persist after the procedure has returned:

```
var
  S: string[50];
.....
procedure TForm1.DoInclude(Sender: TObject; const Command: string; Params:
TStrings;
                                      var Buffer: PChar; var BuffSize:
LongInt);
begin
BuffSize := 0;
if CompareText(Command, 'Date') = 0 then
  begin
  S := 'Today''s date is '+DateToStr(Date);
  Buffer := @S[1];
  BuffSize := Length(S);
  end
Params.Free;
end;
```

In the HTML document, the following would be replaced in its entirety by the generated string.

```
   <!--#date -->
```

OnMeta Event

Declaration

```
property OnMeta: TMetaType;
```

Description

The OnMeta event occurs when a **<METAt>** tag is found while the HTML file is being parsed. See the Type TMetaType description for details on the passed parameters.

OnMetaRefresh Event

Declaration

```
property OnMetaRefresh: TMetaRefreshType;
```

Description

The OnMetaRefresh event occurs when a **<META>** tag is found with the Http_Eq="refresh" attribute. This event may be used to load a file to *refresh* the display. See the Type TMetaRefreshType description for details on the passed parameters.

OnObjectClick Event

Declaration

```
property OnObjectClick: TObjectClickEvent;
```

Description

The OnObjectClick event occurs when the mouse is left clicked on an Object. Currently, all form controls support this event. See the TObjectClickEvent type for information on the parameters.

Example

The following handler will display the status of radio buttons and checkboxes whenever they are changed. Display only occurs for those controls having the OnClick="display" attribute:

```
procedure TForm1.ObjectClick(Sender, Obj: TObject; const OnClick: String);
var
  S: string;
begin
if OnClick = 'display' then
  begin
  if Obj is TFormControlObj then
    with TFormControlObj(Obj) do
      begin
      if TheControl is TCheckBox then
        with TCheckBox(TheControl) do
          begin
          S := Value + ' is ';
          if Checked then S := S + 'checked'
            else S := S + 'unchecked';
          MessageDlg(S, mtCustom, [mbOK], 0);
          end
      else if TheControl is TRadioButton then
        with TRadioButton(TheControl) do
          begin
          S := Value + ' is checked';
          MessageDlg(S, mtCustom, [mbOK], 0);
          end;
      end;
  end;
end;
```

See also:

- OnImageClick Event
- OnImageOver Event

OnProcessing Event

Declaration

```
property OnProcessing: TProcessingEvent;
```

Description

The OnProcessing event occurs whenever the viewer either begins or finishes an operation. This event may be used to disable menu items and/or buttons during time consuming operations.

See the TProcessingEvent type for information on the parameters.

OnRightClick Event

Declaration

```
property OnRightClick: TRightClickEvent;
```

Description

The OnRightClick event occurs when the right button is clicked anywhere within ThtmlLite. It's purpose is to supply sufficient information for a popup menu. See the Type TRightClickEvent description for the details on the parameters.

See also:

- OnImageClick Event
- OnObjectClick Event

OnScript Event

Declaration

```
property OnScript: TScriptEvent;
```

Description

The OnScript event occurs when a **<script>** tag is found while the HTML file is being parsed. See the Type TScriptEvent description for details on the passed parameters.

OnSoundRequest Event

Declaration

```
property OnSoundRequest: TSoundType;
```

Description

The OnSoundRequest event occurs when the **<BGSOUND>** tag is encountered at load time. See the TSoundType type for information on the parameters.

Palette Property

Declaration

```
property Palette: HPalette;
```

Description

In order to best display many images each of which may have its own palette, the ThtmlLite component normally color matches each image to a compromise rainbow palette. The Palette property allows the user to access this palette or to change it.

A common reason for wishing to change the palette is when the user has complete control of all the images to be displayed and the images all use the same palette.

Usage Notes

The Palette property has no effect in modes other than 256 color mode.

Palettes are copied when read or written.

Example

Assume that all images will use a common palette. The following shows code in the form's OnCreate event handler which uses the palette from one of those images to change ThtmlLite's palette.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  Bitmap: TBitmap;
begin
Bitmap := TBitmap.Create;
{Load a bitmap that has the common palette}
Bitmap.LoadFromFile(ExtractFilePath(ParamStr(0))+'myimage.bmp');

{Replace the palette with the common palette}
Viewer.Palette := Bitmap.ReleasePalette;
Bitmap.Free;
end;
```

Position Property

Declaration

```
property Position: LongInt;
```

Description

The Position property is a coded number (LongInt) which specifies the part of the document appearing at the top of the window. Since this number is relatively independent of document reformatting, it may be used to save a position to be returned to at a later time.

Specifying a Position of 0 will take you to the top of the document. Other values are not easily calculated.

For an alternate method of positioning, see:

VScrollBarPosition Property

VScrollBarRange Property

Example

```
var
  SavePos: LongInt;
....
SavePos := Viewer.Position; {save the position}
....
{intervening operations which may reformat the document}
....
Viewer.Position := SavePos; {return to original spot in document}
```

PositionTo Method

Declaration

```
procedure PositionTo(Dest: String);
```

Description

Dest is a target name defined within the HTML document. PositionTo positions the document display to that location.

Example

```
Viewer.PositionTo('#Contents');
```

The **#** sign may be omitted if desired.

Processing Property

Declaration

```
property Processing: Boolean;
```

Description

Read only. The Processing property is set true whenever the viewer is busy processing data from a previous operation.

Some operations such as loading a file with many images may require a lengthy processing time. During this time, most other calls to the viewer will be ignored. The Processing property serves as a flag to indicate when calls can be made.

See Also:

OnProcessing Event

Reformatting occurs mainly when the window width is resized causing line wrap to change. Reformatting can also occur when **ViewImages** is toggled.

Reload Method

Declaration

```
procedure ReLoad;
```

Description

The ReLoad method reloads the previously loaded file or frameset. It take into account the mode in which the file was originally loaded (HTML, text, or image mode).

SelLength Property

Declaration

```
property SelLength: LongInt;
```

Description

The SelLength property gives the number of characters currently selected. Its value may be positive or negative indicating the direction of selection. The starting value of the selection is given by the SelStart property.

The SelLength property may be set to 0 to clear text selection.

See Also:

SelStart Property

SelectAll Method

CopyToClipboard Method

SelText Property

Declaration

```
property SelText: String;
```

Description

Read only and run time only. The SelText property returns the selected text in string form.

See Also

GetSelTextBuf Method

CopyToClipBoard Method

SelectAll Method

Declaration

```
procedure SelectAll;
```

Description

The SelectAll method selects all the document text. This might be used to copy all the text to the clipboard.

See Also:

SelLength Property

CopyToClipboard Method

ServerRoot Property

Description

```
property ServerRoot: string;
```

The ServerRoot property establishes an effective root directory on disk. URLs which begin with a slash (or backslash) are considered to be relative to this directory. If no ServerRoot is specified, the root is taken to be the root directory of the current drive.

Example

```
ServerRoot := 'C:\MyDirectory';
```

The URL:

```
/images/new.gif
```

will be now be translated to:

```
C\MyDirectory\images\new.gif
```

Setting up a History List

A history list is useful for backtracking to earlier loaded documents or places previously visited in the same document. From the users point of view, histories are normally accessed as menu items and/or Back/Forward buttons.

The key properties and events relating to history lists are the History, TitleHistory, HistoryIndex, and HistoryMaxCount properties, the ClearHistory method, and the OnHistoryChange event.

1. Set HistoryMaxCount to the number of list items wanted. HistoryMaxCount should be set to 0 if no history list is desired.
2. History (TStrings) will be automatically updated with filenames as files are loaded and positions changed. The History strings may be used as menu items or as listbox contents. Ordering is such that the current file is at index 0.
3. TitleHistory (TStrings) will be automatically updated with document titles as files are loaded and positions changed. The TitleHistory strings may be used as menu items or as listbox contents. Ordering is such that the current file is at index 0.
4. An OnHistoryChange event occurs each time the history list changes. A method responding to this event may be used to determine which buttons and/or menu items are visible, grayed, or checked.
5. The HistoryIndex property may be read to determine which history list item is currently active. Setting of the HistoryIndex property to a new value loads and positions the appropriate file. HistoryIndex can not be set outside the range of the current history list.
6. The ClearHistory method may be used to clear the History List.

ThtmlLite Component

Properties Methods Events Tasks

Unit

HTMLLite.Pas

Description

The ThtmlLite component displays single HTML documents. Documents may be loaded from disk files, memory buffers, StringLists, or streams.

ThtmlLite Events

Key Events

OnFormSubmit
OnHistoryChange
OnHotSpotCovered
OnHotSpotClick
OnImageClick
OnImageOver
OnImageRequest
OnInclude
OnMeta
OnMetaRefresh
OnObjectClick
OnProcessing
OnScript
OnSoundRequest

OnRightClick

ThtmlLite Methods

Clear
ClearHistory
CopyToClipboard
DisplayPosToXY
Find
FindDisplayPos
FindSourcePos
GetSelTextBuf
HTMLExpandFilename
InsertImage
LoadFromFile
LoadFromBuffer
LoadFromStream
LoadFromString
LoadImageFile
LoadStrings
LoadTextFile
LoadTextStrings
PositionTo
PtInObject
Reload
SelectAll

ThtmlLite Properties

Key Properties

Run Time Only Properties

Base

BorderStyle

CaretPos

CharSet

CurrentFile

DefBackground

DefFontColor

DefFontName

DefFontSize

DefHotSpotColor

DefOverLinkColor

DefPreFontName

DefVisitedLink

DocumentTitle

FormControlList

History

HistoryIndex

HistoryMaxCount

htOptions

ImageCacheCount

MarginHeight

MarginWidth

NameList

NoSelect

Palette

Position

Processing

SelLength

SelStart

SelText

ServerRoot

TitleHistory

URL

ViewImages

VisitedMaxCount

VScrollBarPosition

VScrollBarRange

TitleHistory Property

De**claration**

```
property TitleHistory: TStrings;
```

Read only and run time only. The TitleHistory property contains a list of the most recently loaded document titles with the current title at index 0.

For additional information on setting up a history list, see Setting up a History List.

Type TFormSubmitEvent

Declaration

```
TFormSubmitEvent = procedure(Sender: TObject; const Action,
           Target, EncType, Method: string; Results: TStringList) of Object;
```

Description

The TFormSubmitEvent type is the type for the OnFormSubmit event. This event occurs when the submit button of an HTML Form is depressed.

Action represents the URL to which the form content is to be sent, *Target*, the destination frame, *EncType*, the Mime type encoding for the Post Method, *Method*, the HTTP method (GET or POST) for sending information. *Action, Method ,Target*, and *EncType* are specified as attributes to the **<form>** tag in the HTML document.

Results is a TStringList representing the entries of the form's control elements. Entries in the TStringList are in the form:

```
NAME=VALUE
```

In general, NAME is the name attribute found in the control's tag and VALUE is either the value attribute or the field entry the user enters in the form's control.

Note: The TStringList, *Results*, should be freed when the user is finished with the data.

See also:

OnFormSubmit Event

Type TGetImageEvent

Declaration

```
TGetImageEvent = procedure(Sender: TObject;
       const SRC: string; var Stream: TMemoryStream) of Object;
```

Description

The TGetBitmapEvent type points to a method that handles image request events. *SRC* is the **SRC=** string recovered from the **<img>** tag. *Stream* is the TMemoryStream returned containing the image. The TMemoryStream may contain a bitmap, GIF, JPEG, or PNG image.

If a stream can't be supplied because of an error condition, etc., a **Nil** value must be returned.

See also:

OnImageRequest Event

Type THotSpotClickEvent

Declaration

```
THotSpotClickEvent = procedure(Sender: TObject;
       const URL: string; var Handled: Boolean) of Object;
```

Description

The THotSpotClickEvent type is the type for hot spot click events. These events occur when the mouse is clicked on a hypertext link. URL is the string referenced by the hot spot.

The method which processes the THotSpotClickEvent should returned Handled = True if it wants no further handling of the URL. If Handled is set to false, ThtmlLite will perform default handling.

IsMap

If the URL comes from an image used as an anchor and the **IsMap** attribute is used, the X, Y pixel position will be passed appended to the URL in the form "?x,y".

See also:

OnHotSpotClick Event

OnHotSpotCovered Event

Type THotSpotEvent

Declaration

```
THotSpotEvent = procedure(Sender: TObject;
       const URL: string) of Object;
```

Description

The THotSpotEvent type is the type for the hot spot covered event. This event occurs when the mouse is positioned over a hypertext link. URL is the string referenced by the hot spot.

See also:

OnHotSpotCovered Event

Type TImageClickEvent

Declaration

```
TImageClickEvent = procedure(Sender, Obj: TObject; Button: TMouseButton;
                        Shift: TShiftState; X, Y: Integer) of Object;
```

Description

The TImageClickEvent type is the type for OnImageClick events. These events occur when the mouse is clicked when over an image.

In the above, *Sender* is the ThtmlLite and *Obj* is the TImageObj where the click occured. *X* and *Y* are the position of the cursor within the image.

See also:

- OnImageClick Event
- OnImageOver Event

Type TImageOverEvent

Declaration

```
TImageOverEvent = procedure(Sender, Obj: TObject; Shift: TShiftState;
                      X, Y: Integer) of Object;
```

Description

The TImageOverEvent type is the type for OnImageOver events. These events occur when the mouse is moved over an image.

In the above, *Sender* is the ThtmlLite and *Obj* is the TImageObj where the click occured. *X* and *Y* are the position of the cursor within the image.

See also:

OnImageOver Event

OnImageClick Event

Type TIncludeType

Declaration

```
  TIncludeType = procedure (Sender: TObject; const Command: string;
                 Params: TStrings; var Buffer: PChar; var BuffSize: LongInt)
of Object;
```

Description

The TIncludeType type is the type for OnInclude events. These events occur when server side include syntax is encountered in the HTML document at load time.

The server side include syntax takes the form:

```
<!--#command arg1="value1" arg2="value2"  ...  -->
```

The '#' distinguishes this syntax from an ordinary comment. The *command* word appears as the *Command* parameter in the event call. The *command* word may be followed by zero or more *arg,value* pairs. These are formed into a TStringList object to make up the *Params* parameter. For use with ThtmlLite, the parameters can be anything that's convenient.

The HTML text to be inserted is passed back in *Buffer*. This text replaces the entire original include statement. *BuffSize* is the size of *Buffer* not includind any terminating null character.

Note that:

ThtmlLite only reads Buffer, it does not change or Free it.

The event handler should conatin a `Params.Free;` statement.

See also:

OnInclude Event

Type TMetaRefreshType

Declaration

```
TMetaRefreshType = procedure(Sender: TObject; Delay: integer;
                    const URL: string) of Object;
```

Description

The TMetaRefreshType type points to a method that might process the contents of a **<META .Http_Eq="refresh">** tag. *Delay* is the time delay in seconds and *URL*, the html document used to refresh the display. These parameters are parsed from the Contents= attribute.

See also:

OnMetaRefresh Event

OnMeta Event

Type TMetaType

Declaration

```
TMetaType = procedure(Sender: TObject; const HttpEq, Name,
                Content: string) of Object;
```

Description

The TMetaType type points to a method that might process the contents of a **<META>** HTML tag. *Sender* is the ThtmlLite which encounters the **<META>** tag. The other three parameters are the values of **<META>** tag attributes with the same name.

See also:

OnMeta Event

OnMetaRefresh Event

Type TObjectClickEvent

Declaration

```
  TObjectClickEvent = procedure(Sender, Obj: TObject;
              const OnClick: string) of Object;
```

Description

The TObjectClickEvent type is the type for OnObjectClick events. These events occur when the mouse is clicked on an Object.

In the above, *Sender* is the ThtmlLite where the Object is located and *Obj* is a pointer to the clicked object. *OnClick* is the string assigned to the OnClick attribute of the object's HTML tag.

Form controls currently support the OnObjectClick event. These objects are descendents of TFormControlObj. In particular, the **TheControl** property can give access to the underlying Delphi controls, such as **TButton**, **TRadiobutton**, and **TCheckbox** with code similar to:

```
if (Obj is TFormControlObj) then
  with TFormControlObj(Obj) do
    if (TheControl is TButton) then .....
```

See also:

OnObjectClick Event

OnImageClick Event

Type TProcessingEvent

Declaration

```
TProcessingEvent = procedure(Sender: TObject;
       ProcessingOn: boolean) of Object;
```

Description

The TProcessingEvent type is the type for the OnProcessing event. This event occurs whenever the viewer begins or finishes an operation.

ProcessingOn is set True at the start of an operation and False when the operation is completed.

See also:

OnProcessing Event

Type TRightClickEvent

Declaration

```
TRightClickEvent = procedure(Sender: TObject;
             Parameters: TRightClickParameters) of Object;
```

Description

The TRightClickEvent type points to a handler for the OnRightClick event. *Sender* is the ThtmlLite where the click occured. *Parameters* is a record whose fields contain information for constructing a popup menu. This form of parameter passing is used to allow additional fields to be added in the future.

Currently TRightClickParameters is defined as follows:

```
TRightClickParameters = Class(TObject)
  URL, Target: string;
  Image: TImageObj;
  ImageX, ImageY: integer;
  end;
```

URL and *Target* are information about an anchor. These fields are blank if the mouse is not over an anchor.

Image is a pointer to the TImageObject managing an underlying image. *ImageX* and *ImageY* give the X and Y position within the image.

See also:

OnRightClick Event

OnImageClick Event

OnObjectClick Event

Type TScriptEvent

Declaration

```
TScriptEvent = procedure(Sender: TObject; Const Name, Language: String;
     Script: PChar) of Object;
```

Description

The TScriptEvent type points to a method that might process the contents of a **<script> ... </script>** HTML tag sequence. *Name* and *Language* are the Name= and Language= attributes recovered from the **<script>** tag and *Script* is the entire text between the tags.

The event handler which processes the OnScript event is responsible for freeing the memory allocated for *Script.* In Delphi 1, use:

```
FreeMem(Script, StrLen(Script)+1);
```

For 32 bit Delphis, simply use:

```
FreeMem(Script);
```

See also:

OnScript Event

TSoundType Type

Declaration

```
TSoundType = procedure  (Sender: TObject; const SRC: string; Loop: integer;
                    Terminate: boolean) of Object;
```

Description

The TSoundType type is the type for OnSoundRequest events. These events occur when the **<BGSOUND>** tag is encountered at load time and at any other time when it is appropriate to terminate the sound.

In the above, *Sender* is the ThtmlLite loading the document, *SRC* and *Loop* are the values of these atttributes, and *Terminate* is a boolean indicating if the sound should be terminated.

URL - Uniform Resource Locator

A sequence of characters for specifying Internet resources. In HTML documents these specify hypertext link targets, image files, etc. As loosely interpreted for DOS HTML viewers, an URL can be a DOS path and filename.

URL Property

Declaration

```
property URL: String;
```

Description

Read only and run time only. The URL property returns the most recently selected (left mouse click) URL.

The URL property contains the URL exactly as it is read from the document. In some cases, it may be necessary to do some conversion to the string before using it. Depending on your application and the documents you are using, you might want to:

Convert from Internet format to Dos format using the HTMLToDos function.

Add a path if it doesn't already have one. The Base property provides the base path. If the Base property is empty, the path expected is generally that of the HTML document.

See Also

Target Property

Using The ThtmlLite Component

ThtmlLite Reference

Tasks

- To load an HTML document from a file, use the LoadFromFile method. To load a document from a TMemo or other memory buffer, use the LoadFromBuffer method.
- Use the Position property to save a location in a document and return to it later. For positioning to a **Name** field in the document, use the PositionTo method.
- To customize response to clicking on a hotspot, respond to the OnHotSpotClick event.
- To implement a status line which shows what URL a hot spot corresponds to, respond to the OnHotSpotCovered event.
- Use the DefFontName, DefFontSize, DefFontColor, DefHotSpotColor, and DefBackground properties to change the default font and color characteristics.for general display. Use the HTML **<body>** tag and it's attributes to specify the appearance of a particular HTML document.
- To implement a history list and/or Back|Forward buttons, see Setting up a History List.
- GIF images designed to be transparent are displayed as such automatically. Other images can be made transparent by adding the **transp** attribute to the **<img>** tag. When this is done, the lower left pixel of the image determines the transparent color.

VScrollBarPosition Property

Declaration

```
property VScrollBarPosition: Integer;
```

Description

The VScrollBarPosition Property gives access to the viewer's vertical scrollbar. This can be used for document positioning in situations where reformatting will not occur.

See Also:

VScrollBarRange Property

Position Property

VScrollBarRange Property

Declaration

```
property VScrollBarRange: Integer;
```

The VScrollBarRange Property gives access to the viewer's vertical scrollbar range value.

See Also:

VScrollBarPosition Property

Position Property

ViewImages Property

Declaration

```
property ViewImages: Boolean
```

Description

The **ViewImages** property determines whether inline images are drawn or simulated with a standard marker image. 

If **ViewImages** is True and an image error occurs, an error image is displayed.

WaitStream is defined in LiteUn2.pas. Include LiteUn2 in your Uses clause.

htOptions Property

Description

```
property htOptions: set of (htOverLinksActive,htNoLinkUnderline,
                            htShowDummyCaret, htShowVScroll);
```

The htOptions property currently has the following items:

htOverLinksActive

When set, link color will change when the mouse is over the link. Color is determined by the special **olink** attribute of the **<body>** tag, if present. If this attribute is absent, the DefOverLinkColor property determines the color.

htNoLinkUnderline

Turns off underlines on links.

htShowDummyCaret

Shows the current character position when text is not otherwise selected.

htShowVScroll

htShowVScroll alters the way the vertical scrollbar is displayed. If htShowVScroll is not set, no vertical scrollbar is shown if none is needed. If htShowVScroll is set, the vertical scrollbar will always be shown but may be shown as disabled.

DefOverLinkColor Property

Declaration

```
property DefOverLinkColor: TColor;
```

Description

DefOverLinkColor sets the default color for links when the mouse is over the link and the htOverLinksActive option is set. This is the color used if the active link color is not otherwise specified in the HTML document file.

See Also

<body> attributes

DefVisitedLinkColor Property

Declaration

```
property DefVisitedLinkColor: TColor;
```

Description

DefVisitedLinkColor sets the default color for links which have been previously visited. This is the color used if the **VLink** attribute is absent from the **<body>** tag in the HTML document file.

See Also

VisitedMaxCount Property

<body> attributes

VisitedMaxCount Property

Declaration

```
property VisitedMaxCount: Integer;
```

The VisitedMaxCount property determines the number of visited links which will be stored. If VisitedMaxCount is 0, then special coloring of visited links will be disabled. The default value is 50.

See also:

DefVisitedLinkColor Property

SelStart Property

Declaration

```
property SelStart: LongInt;
```

Description

SelStart specifies the first character position of selected text or the current position in the display if there is no selection. Character count starts with 0.

To make the current position visible even when no text is selected, see the htShowDummyCaret value in the htOptions property.

See Also:

SelLength Property

DisplayPosToXY Method

DisplayPosToXY Method

Declaration

```
function DisplayPosToXY(DisplayPos: integer; var X, Y: integer): boolean;
```

Description

Given a character position in the display, DisplayPosToXY finds the *X* and *Y* location of the character on the display. Dimensions are in pixels relative to the upper left corner of the display.

DisplayPosToXY returns False if the requested position does not exist.

Example:

The following code will check to see if the selected position is within the currently displayed area If not it will position the display to insure that it is

```
if Viewer.DisplayPosToXY(Viewer.SelStart, X, Y) then
  begin
  VPos := Viewer.VScrollBarPosition;
  if (Y < VPos) or (Y > VPos +Viewer.ClientHeight-20) then
    Viewer.VScrollBarPosition := (Y - Viewer.ClientHeight div 2);
  end;
```

See Also:

SelStart Property

FindSourcePos Method

FindDisplayPos Method

FindSourcePos Method

Declaration

```
function FindSourcePos(DisplayPos: integer): integer;
```

Description

Given a character position in the display, FindSourcePos finds the equivalent position in the HTML source document.

Note that the display position is that value given by the SelStart property.

See Also:

SelStart Property

FindDisplayPos Method

DisplayPosToXY Method

FindDisplayPos Method

Declaration

```
function FindDisplayPos(SourcePos: integer; Prev: boolean): integer;
```

Description

Given a character position in the HTML source, FindDisplayPos finds the **nearest** equivalent character position in the displayed document.

If there is no exact equivalent character position, setting *Prev* True will find the nearest previous character position. Otherwise the next position will be returned.

See Also:

SelStart Property

FindSourcePos Method

DisplayPosToXY Method

LoadFromString Method

Declaration

```
procedure LoadFromString(S: String);
```

Description

The LoadFromString method loads an HTML document from a string.

See also:

- LoadFromFile Method
- LoadStrings Method
- LoadFromBuffer Method
- LoadFromStream Method

PtInObject Method

Declaration

```
function PtInObject(X, Y: integer; var Obj: TObject): boolean;
```

Description

`PtInObject` determines if the point, (X,Y), is within a particular displayed object. *X* and *Y* are ThtmlLite client coordinates, and *Obj* is the TObject displayed at that coordinate.

Currently, the only objects tested are images (TImageObj). **`PtInObject`** may be used to ascertain information about images.

MouseOver Images

A ***Mouseover Image*** is one that changes when the mouse passes over it. ThtmlLite uses a special syntax to implement mouseover images. It's done with the **<img>** tag as follows:

- the SRC="..." attribute should specify an animated GIF image
- Add an **Active** attribute

With the **Active** attribute, the GIF will normally display the first frame. When the mouse passes over, it will display the second frame if there are 2 frames or will animate if there are more than 2 frames.