Variable substitutions in the skin template are all managed by the FmtPageName() function from pmwiki.php. Pmwiki variable substitutions available on pages are managed by the substitutions from stdmarkup.php or superseded in local/config files.

:$ActionSkin:This array is used to override the current skin when performing a given action. The most common use is to set @@$Action Skin?['print']='foo' `to use the 'foo' skin when printing, regardless of what the` $Skin@@ variable is set to.

`$WikiTitle` A variable which contains the Wiki title as displayed by the browser
$EnablePageTitlePriorityA variable defining how to treat multiple `(:title ...:)` page directives (added in Pm Wiki 2.2.9). `$EnablePageTitlePriority = 0;` # Pm Wiki default, last encountered title wins (the title may be changed from included pages or Group Footer). `$EnablePageTitlePriority = 1;` # First title wins; if a title is defined in the page, directives from included pages cannot change it.
$EnableDiffInlineIf set to 0, this variable switches off the word-level highlighting on the markup in the page history. `$EnableDiffInline = 0;` # Disable colors, show plain text differences
`$HTMLStylesFmt` An array of CSS statements to be included in the page's output along with other HTML headers. This array provides an easy place for scripts to add custom CSS statements.
`$HTMLHeaderFmt` An array of HTML text to be included in the page's <head> section, at the point where the skin template specifies a `<!--HTMLHeader-->` directive. This array provides an easy place for scripts to add custom HTML headers.
For example, if you want to specify a logo for all the pages of your wiki (a png image for Firefox (and others...), an ico for Internet Explorer):

```
 $HTMLHeaderFmt['logo'] = '<link href="http://path/to/logo.png" type="image/png"
rel="icon" /> <link href="http://path/to/logo.ico" type="image/x-icon"
rel="shortcut icon" />';
```

Another example, if you want to get the rss notification on some browsers (the rss icon in firefox for instance):

```
 $HTMLHeaderFmt['rss'] = '<link rel="alternate" type="application/rss+xml"
title="Rss All recent Changes"
href="$ScriptUrl/Site/AllRecentChanges?action=rss" />';
```

:$HTMLFooterFmt:Like $HTML Header Fmt? above, this contains an array of HTML text to be included near the end of an HTML document, at the point where the skin template specifies a `<!--HTMLFooter-->` directive (usually just before a closing tag). Primarily used by scripts to add custom HTML output after the body of the page output.

:$MetaRobots:Sets the value of the `<meta name='robots' ... />` tag generated by PmWiki to control search engine robots accessing the site. PmWiki's default setting tells robots to not index anything but the normal page view, and to not index pages in the PmWiki [[wiki group]]. Explicitly setting $Meta Robots? overrides this default.

`# never index this site` `$MetaRobots` = 'noindex,nofollow'; # disable the robots tag entirely
`$MetaRobots` = '';

$MessagesFmtAn array of HTML text to be displayed at the point of any `(:messages:)` markup. Commonly used for displaying messages with respect to editing pages.
`$RecentChangesFmt` An array specifying the format of the Recent Changes ? listing.
The key of the array specifies the page where changes will be logged, as in

```
 $RecentChangesFmt['$SiteGroup.AllRecentChanges']
```

The value of the array specifies the format in which the changes will be logged, as in

```
 '* [[{$Group}.{$Name}]] . . . $CurrentTime $[by] $AuthorLink:
```

```
[=$ChangeSummary=]'
```

Note the two consecutive spaces before the three dots (`. . .`). The two spaces separate two parts of the format: the first part doesn't change (e.g. a link to the changed page) and the second part does change (e.g. the date and author of the change). Upon saving a page, Pm Wiki removes a line that matches the first part and adds a line with the current format before the first line with 2 spaces. This way, any line without two consecutive spaces stays at the top of the recent changes page.
You can use and adapt the following to change the format (put it in config.php):

```
 $RecentChangesFmt['$SiteGroup.AllRecentChanges'] = '* [[{$Group}.{$Name}]] . .
. $CurrentTime $[by] $AuthorLink: [=$ChangeSummary=]';
$RecentChangesFmt['$Group.RecentChanges'] = '* [[{$Group}/{$Name}]] . . .
$CurrentTime $[by] $AuthorLink: [=$ChangeSummary=]';
```

Note that changes made to the format will only affect new edits. In other words, you will need to edit a page for your new format to be visible. Note also that you need to have two spaces between the page name and the other information about the edit.
Also note that this variable has other uses, such as not reporting at all to Recent Changes? and All Recent Changes? as found here PmWiki Questions.

:$RecentUploadsFmt:An array specifying the format for uploaded files at the RecentChanges listing. It is similar to $Recent Changes Fmt?. If enabled, newly uploaded files will be logged to the Recent Changes? pages. Default is disabled. See Cookbook:RecentUploadsLog for more information.

$DraftRecentChangesFmtAn array specifying the format of the Recent Changes ? listing when saving Draft pages.
$RecentChangesFmt is set to $Draft Recent Changes Fmt? when a Draft page is saved. For example, you could save drafts in a separate Recent Draft Changes page and not list in the normal group's Recent Changes page:

```
 $DraftRecentChangesFmt['$Group.RecentDraftChanges'] = '* [[{$Group}/{$Name}]] .
. . $CurrentTime $[by] $AuthorLink: [=$ChangeSummary=]';
$DraftRecentChangesFmt['$Group.RecentChanges'] = '';
```

$RCLinesMaxThe maximum number of lines to be stored in Recent Changes ? pages. The default is zero, meaning "no limit".

```
$RCLinesMax = 1000; # maintain at most 1000 recent changes
```

`$PageRedirectFmt` The text to be used when a page is redirected via the `(:redirect:)` markup.

```
 $PageRedirectFmt = '<p><i>redirected from $FullName</p>'; $PageRedirectFmt =
'';
```

For display options, see also the FAQ on PageDirectives.
`$WikiStyle` An array which contains the predefined Wiki Styles which can be used on a textpage.
See: PmWiki.CustomWikiStyles
$WikiStyleApplyAn array which defines the scope of wiki styling per HTML element. Default settings are:

```
 'item' => 'li|dt', 'list' => 'ul|ol|dl', 'div' => 'div', 'pre' => 'pre', 'img'
=> 'img', 'block' => 'p(?!\\sclass=)|div|ul|ol|dl|li|dt|pre|h[1-6]', 'p' =>
'p(?!\\sclass=)'
```

This defines that we can apply wiki styling on:

- LI elements using the *item* keyword
- UL, OL, DL elements using the *list* keyword

- etc.

An example of applying scope to an LI element is below. For more information refer to WikiStyle scopes.

`* %apply=item red%Here is a red styled list item * This item would not be styled.`	Here is a red styled list item This item would not be styled.

You can add additional HTML elements to $WikiStyleApply to apply wiki styles to other HTML elements. For example to allow styling on table rows, or anchor tags.

:$MaxIncludes:Controls the number of times that pages can be included via the `(:include:)` and other directives, used to control recursion and otherwise pose a sanity check on page contents. $Max Includes? defaults to 50, but can be set to any value by the wiki administrator.

`$MaxIncludes` = 50; # default `$MaxIncludes` = 1000; # allow lots of includes `$MaxIncludes` = 0; # turn off includes

:$Skin:Lists the name(s) of skins to load, unless overridden by $Action Skin?. Normally $Skin contains a single string which is a the name of a skin directory, but it may also be an array of names, in which case the first skin found from the list is used.

`$SkinDirUrl` Set by *scripts/skins.php* to be the base url of the current skin's directory (i.e., within a 'pub/skins/' directory). This variable is typically used inside of a skin .tmpl file to provide access to .css files and graphic images associated with the skin.
$SkinLibDirsAn array which, given the filesystem path (array key) to a skin (or a directory containing several skins), provides the corresponding URL (array value).
The array key is the directory containing the skin.tmpl and skin.php files, as seen by the Pm Wiki program. It does not have to be publicly accessible.
The value is the URL (web address) of the directory containing the .css, .gif, and other files which appear in the HTML code sent by PM Wiki? to the browser. This directory must be publicly accessible.
By default $SkinLibDirs is set to:

```
 $SkinLibDirs = array( "./pub/skins/\$Skin" => "$PubDirUrl/skins/\$Skin",
"$FarmD/pub/skins/\$Skin" => "$FarmPubDirUrl/skins/\$Skin");
```

Extra details: When PM Wiki? is searching for a skin it looks for a directory named for the skin in the array index/keys, and if it finds it then it will use the files in that directory and also the files in the matching array value url. The two sides normally point to the same publicly accessible directory, but they do not have to.
$PageLogoUrl is the url that refers to a logo image which most skins display somewhere in the page's header (top left usually).
`$EnablePathInfo` Changes the handling of the page URL. When set to `1` page URL will be `...wiki.php/Main/Main`, when set to `0` (default) it will be `...wiki.php?n=Main.Main`.

:$EnableFixedUrlRedirect:When PmWiki is given a partial page name (e.g., just the name of a WikiGroup), it uses $Page Path Fmt? in order to make a complete page name from the partial one, then issues a "redirect" to the browser to tell it to reload the page with the correct full page name. Setting $EnableFixedUrlRedirect=0; blocks the redirect, so that Pm Wiki continues processing with the adjusted page name rather than issuing the redirect.

$GroupHeaderFmtDefines the markup placed at the top of every page. Default value is:

```
$GroupHeaderFmt = '(:include $Group.GroupHeader:)(:nl:)';
```

$GroupPrintHeaderFmtDefines the markup placed at the top of every page when `action=print`. Default value is:

```
SDV($GroupPrintHeaderFmt,'(:include $Group.GroupPrintHeader:)(:nl:)');
```

$GroupFooterFmtDefines the markup placed at the bottom of every page. Default value is:

```
$GroupFooterFmt = '(:include $Group.GroupFooter:)(:nl:)';
```

$GroupPrintFooterFmtDefines the markup placed at the bottom of every page when `action=print`. Default value is:

```
SDV($GroupPrintFooterFmt,'(:nl:)(:include $Group.GroupPrintFooter:)');
```

$PageNotFoundHeaderFmtSpecifies the HTTP header to send when attempting to browse a page that doesn't exist. Some webserver packages (notably Microsoft's "Personal Web Server") require that this variable be changed in order to work.

```
 # default $PageNotFoundHeaderFmt = 'HTTP/1.1 404 Not Found'; # return all pages as found $PageNotFoundHeaderFmt = 'HTTP/1.1 200 Ok';
```

Beware when expecting to return the content of a Group(header|footer) for an non existent page! By default Pm Wiki returns 404 (because the page does not exist), despite there is some content to show. Firefox shows the content, while Internet Explorer displays its default 404 page. $PageNotFoundHeaderFmt MUST be set to return 200 as described above in order to get the expected behaviour with all browsers.
$HTMLVSpaceSetting `$HTMLVSpace = '';` in a local customizationfile (e.g., `local/config.php`) prevents insertion of spacer paragraphs (`<p class='vspace'></p>`) in generated HTML code. To limit this change to a single skin, place the `$HTMLVSpace = '';` statement in a skin.php file, preceded by the statement `global $HTMLVSpace;`.

:$HTMLPNewline:This variable allows to enable linebreaks by default, i.e. without having the directive (:linebreaks:) in a page or in a GroupHeader. To enable line breaks, add to config.php such a line:
 @@$HTMLP Newline? = ''; @@

:$TableCellAttrFmt:For [[Tables]], defines the HTML attributes given to each @@@@ or @@@@ cell in the output. Can contain references to $Table Cell Count? which holds the horizontal column number of the current cell.

:$TableRowAttrFmt:For [[Tables]], defines the HTML attributes given to each @@@@ element in the output. Can contain references to $Table Row Count? to give the absolute row number within the table, or $TableRowIndex to provide a repeating row index from 1 to $Table Row Index Max?.

```
# Give each row a unique CSS class based on row number (tr1, tr2, tr3, ... )
$TableRowAttrFmt = "class='tr\$Table Row Count?'"; # Give each row alternating CSS classes (ti1, ti2, ti1, ti2, ti1, ... ) $TableRowIndexMax = 2; $TableRowAttrFmt = "class='ti\$Table Row Index?'";
```

:$TableRowIndexMax:The maximum value for $Table Row Index? in Tables.

```
# Set rows indexes as 1, 2, 3, 1, 2, 3, 1, 2, ... $TableRowIndexMax = 3;
```

$FmtV['$Table Cell Count ?']PM Wiki ? internal variable - Horizontal column number of the current cell. For use in $TableCellAttrFmt and $Table Row Attr Fmt ? . Administrators can use in $TableCellAttrFmt and/or $

Table Row Attr Fmt ?.

```
Example: $TableCellAttrFmt = 'class=col\$TableCellCount';
```

$FmtV['$Table Row Count ?']PM Wiki ? internal variable - Current row number. Administrators can use in $TableCellAttrFmt and/or $Table Row Attr Fmt ?.

```
Example: TableRowAttrFmt = "class='row\$TableRowCount'";
```

$FmtV['$Table Row Index ?']PM Wiki ? internal variable - Row index number derived from $TableRowIndexMax. (1,2,3,1,2,3,...). Administrators can use in $Table Cell Attr Fmt ? and/or `$TableRowAttrFmt`.

```
Example: $TableRowAttrFmt = "class='ind\$TableRowIndex'";
```

See also: Edit Variables