Interspire Website Publisher Developer Documentation

Template Customization Guide

Table of Contents

Interspire Website Publisher Template Customization Guide

Introduction

Interspire Website Publisher contains a powerful template system that can be used to completely customize the look and feel of your website to your needs to integrate it with an existing website design or create something new.

The Interspire Website Publisher template system uses one XHTML file called styleguide.html. The style guide file is processed by Interspire Website Publisher when selecting the template or when applying changes via the Site Layout page of the Control Panel. If you make modifications to the style guide and want them reflected on your website, simply reselect your template from the Control Panel's Website Design page.

Interspire Website Publisher depends on the style guide file being valid XHTML. If there are validation errors, they will be displayed when the style guide is being processed.

The style guide file is mostly normal XHTML, but with a few additions including two new attributes for defining blocks and special template class names used to direct the style guide processing. These are discussed in the sections in this article labeled *Blocks* and Template *Classes*.

Template Directory Structure

Interspire Website Publisher templates exist within the “templates” directory that exists within the Interspire Website Publisher folder. Each folder in this directory refers to a different template selectable from the Website Design page of the Control Panel.

Templates have the ability to have multiple color schemes based off the one set of HTML and stylesheet files.

Each template contains several files and folders. The structure of the contents of this folder is outlined below:

- **images/**
  This contains any images used by the styleguide.html file.
- **logo/**
  This contains the logo generator PHP class.
- **modules/**
  This contains template files and styles to override modules default.
- **previews/**
  This contains the preview images. Each one should correspond to a color stylesheet file.
- **styles/**
  This contains the stylesheet files used by this template.
  - **styles.css**
    This is the primary stylesheet. It should include all styles not defined by color.css
  - **color.css**
    This defines color specific styles. More information about this stylesheet can be found below.
  - **datepicker.css**
    This defines the styles for the date picker used on the advanced search page.
  - **ie.css**
    This defines the styles that are used by Internet Explorer only.
- **changelog.txt**
  This contains the list of changes in each version of the template

- **config.php**
  This contains configuration information about the template used the application. Such as number of columns, available colors, and version.

- **styleguide.html**
  This is the style guide XHTML file. This contains all HTML used by the application, other than modules.

Each template splits the CSS up into multiple files. Most styles are to be placed into styles/styles.css (such as font, positions, padding) while anything related to color should be placed in the {color}.css where {color} is the color name for the theme. Color names should be kept simple and standard, i.e. blue.css or pink.css. This allows a single template to have more than one color variation by just including an alternate {color}.css file.

The Style Guide File

The purpose of the style guide file (styleguide.html) is to provide one file and point of reference for styling an entire website. One benefit is the ability to visually see the result of CSS changes. Unlike other template systems, the style guide file can be viewed directly in a web browser with content and no PHP or template code interfering with the design.

All text used within the file can be considered sample text and will be replaced by user generated content or language variables. Editing text in the file will not result in a change on the website, but rather aid in designing the template by populating it with the appropriate lengths of text.

The style guide has three main uses. Firstly it acts as an aid to styling the overall template and all the generic styles (e.g. for h1, h2, h3, tables, forms and all other types of elements). Secondly, it styles the individual blocks that are used n the website by different features and lastly, it controls overall XHTML structure of the page.

The style guide is processed by the application and stored into cached template files. If modifications are made to the style guide and should be reflected on the website, simply reselect the template from the Website Design page in the Control Panel for the style guide will be reprocessed.

Blocks

What are blocks?

Blocks are sections of code that are pulled out from the style guide and used separately. Most blocks will undergo some processing before being finally output. These blocks are then template snippets that are used by Interspire Website Publisher for different sections.

Blocks can define the layout for a menu, a list of content, for displaying content, for the footer, etc. Essentially, anything that contains content and doesn't control the overall layout of the page (can be moved and may not be displayed on every page) can be considered a block.

Block Attributes

There are a few things you may notice in the style guide file that are not standard XHTML attributes. These signify the different blocks within the file. For example, you might see something like this:

```
<div id="Menu" block="list" blockStyle="Top Standard">
      <!-- other html in here -->
</div>
```

In the above example, there are two non-standard XHTML attributes. These are `block="list"` and `blockStyle="Top Standard"`.

The `block` attribute is the name for the block and used by the system to determine how to process the XHTML within the tag. Possible values are pre-defined by the system. (A list of blocks can be found in Appendix 1: Block Reference Table.)

The `blockStyle` attribute gives the block a style name and allows an alternate block layout/style. These are not pre-defined and are just labels. The `blockStyle` attribute should be unique for the block it's used for (i.e. you can't have two lists with a block style of 'Standard', but you can have a list block and a `customContent` block with the same `blockStyle` name). Any block with 'Standard' or 'Standard {Position}' (where {Position} can be top, left, middle, right or bottom) is used as the default style for that block type.

Ignore Comments

When processing the style guide, the application removes the blocks from the layout HTML whilst retaining everything else. If you wish to include additional HTML in your style guide (perhaps to make something easier to style) but do not want it output, you can use ignore comments to tell the system to remove certain portions of HTML from the final output.

To tell the application to ignore something, use the ignore comments. An sample ignore comment can be found below:

```
<!--{ignore}-->
<blockquote>This will be stripped out for the final template, but I
would like it in the styleguide.html file so I can style blockquote
elements</ blockquote>
<!--{/ignore}-->
```

When styling a list or menu, you may find that you wish to have 5 or so items in the menu to have an accurate representation of how the list will look when it contains actual website content. To do this, you can use the template classes on the first item, and surround the next 4 with ignore comments so they're removed – but they are still there when you wish to adjust the document styles.

You can also use an ignore comment to strip out just itself. This allows for style guide specific HTML comments that are removed so the final website will not contain them. For example:

```
<!--{ignore/} I'd like to mention something about the code here –
but I don't want website visitors to see this comment in the final
XHTML -->
```

Template Classes

As mentioned earlier, the style guide is made up of different blocks that are styled for different situations. Within these blocks is the XHTML that will be used to display the content or content items. The application needs instructions on what to do with the XHTML, so the style guide has a preset list of CSS classes that tell it what to do and where.

A list of all the available template classes can be found in Appendix 2: Template Classes Reference Table.

A brief example of the use of template classes in the HTM can be seen below.

```
<div id="Menu" block="list" blockStyle="Top Standard">
      <div class="tplrepeat-parent">
            <a href="#" class="tpllang-ReadMore tplvar-listitem-url-
href">My Sample Text</a>
            <!--{ignore}-->
            <b>My Text</b>
            <!--{/ignore}-->
      </div>
</div>
```

Resulting HTML:

```
<div id="Menu">
      <div class="tplrepeat-parent">
            <a href="http://www.example.com/articles/my-
article.html" class="tpllang-ReadMore tplvar-listitem-url-href">Read
More</a>
            <a href="http://www.example.com/articles/another-
article.html" class="tpllang-ReadMore tplvar-listitem-url-href">Read
More</a>
            <a href="http://www.example.com/articles/third-
article.html" class="tpllang-ReadMore tplvar-listitem-url-href">Read
More</a>
      </div>
</div>
```

As can be seen, the classes that were applied to the element performed different actions:

- The `tpllang-ReadMore` variable replaced the inner text of the element with the value of the language variable `ReadMore`
- The `tplvar-listitem-url-href` class changed the value of the `href` attribute to the URL of the current list item.
- The `tplrepeat-parent` specified that the content inside it was to be repeated, and the content between the ignore comments was stripped out.

There are 6 different types of classes that are used. Each type serves a different purpose. Some classes define where content should be placed whilst others are flags for loops or conditions.

Template Class Type: Language Variable

Purpose: Outputs a language variable from the frontend.ini language file.

Format: tpllang-[variablename](-[attribute])

Samples: tpllang-Example, tpllang-Example-alt

Full Example:

```
<a href="#" class="tpllang-SubscribeRSS">Example Text</a>
```

Resulting HTML:

```
<a href="#" class="tpllang-SubscribeRSS">Subscribe via RSS</a>
```

The value can also be inserted into an attribute of the current element by adding the attribute name on the end:

```
<a href="#" class="tpllang-SubscribeRSS-title">Example Text</a>
```

Resulting HTML:

```
<a href="#" class="tpllang-SubscribeRSS-title" title="Subscribe via
RSS">Example Text</div>
```

You can use more than one class, too:

```
<a href="#" class="tpllang-ReadMore tpllang-SubscribeRSS-
title">Example Text</a>
```

Resulting HTML:

```
<a href="#" class="tpllang-ReadMore tpllang-SubscribeRSS-title"
title="Subscribe via RSS">Read More</a>
```

Attributes can also be chained if you'd like the value of one class to be inserted into multiple places. When doing this you can use -inside to place the value inside the current element.

```
<a href="#" class="tpllang-ReadMore-inside-title">Example Text</a>
```

Resulting HTML:

```
<a href="#" class="tpllang-ReadMore" title="Read More">Read More</a>
```

Template Class Type: Content Variable

Purpose: Output dynamic content such as site-wide variables or variables specific to blocks. See Appendix 2: Template Classes Reference Table for a full list of available variables.

Format: tplvar-[scope]-[variablename](-[attribute])

Samples: tplvar-site-title, tplvar-site-url-href, tplvar-listitem-title, tplvar-listitem-url-href

Full Example:

```
<a href="#" class="tplvar-site-name-inside-title tplvar-listitem-
url-href tplvar-listitem-target-target">Example Text</a>
```

Resulting HTML:

```
<a href="http://www.example.com/articles/my-post.html"
class="tplvar-site-name-inside-title tplvar-listitem-url-href
tplvar-listitem-target-target" title="My Website Name"
target="_blank">My Website Name</a>
```

In the above example the site-wide variable `tplvar-site-name` is placed inside the element and in the title attribute. The list block specific variable `tplvar-listitem-url` is placed into the `href` attribute. Lastly you would have noticed `tplvar-listitem-target-target`, which is correct. Both the variable name and the attribute to insert it into are called 'target'.

There are four special attribute modifiers that can be used. The are -styleprepend, -styleappend, -classprepend and -classappend. These modifiers append/prepend to the `class` or `style` attribute (rather than replacing the value) and includes a space before or after the value.

Full Example:

```
<a href="#" class="tplvar-search-showadvanced-styleprepend"
style="color: red;">Example Text</a>
```

Resulting HTML:

```
<a href="#" class="tplvar-search-showadvanced-styleprepend"
style="color: red; display: none;">My Website Name</a>
```

Template Class Type: Condition

Purpose: Predefined conditionals to o many different things. Most commonly conditionals will allow you to show or hide certain elements based on a value.

The conditionals are all predefined and have specific actions. See

Appendix 2: Template Classes Reference Table for a full list of all available condition classes.

Format: tplcond-[condition-[name]]

Samples: tplcond-showif-haswebsitelogo, tplcond-hideif-1column-2columns

Full Example:

```
<h1 class="tplcond-hideif-haswebsitelogo"><a href="#" class="tplvar-site-name tplvar-site-siteurl-href">Website Title</a></h1>
```

In this example, the `tplcond-hideif-haswebsitelogo` class adds a condition to the entire `h1` tag and its contents. If the website has logo generated, the `h1` tag will not be output. If it doesn't have a logo generated, the `h1` tag will be output.

Template Class Type: Section

Purpose: Sections are used in the style guide to specify where and how blocks are placed. Blocks can be positioned on the 'Site Layout' page into one of 5 sections. These are: top, left, middle, right and bottom.

Format: tplsection-[location](-[position])(-[blocknumber])

Samples: tplsection-top, tplsection-bottom-append-block1, tplsection-right-block2, tplsection-left-after

Detailed Usage Information:

Basic usage is `tplsection-top`. This will output all blocks that have been position in the top section at the beginning of the inside of the element. A position modifier can be added to the class to specify a different location for the blocks to be output in relation to the element that has the class. For example: `tplsection-top-after` will add the blocks after the element, instead of inside it. Possible positions are: inside (default), replace, after, before, prepend and append.

- **inside**
  This places the section's blocks inside the current element, removing any existing HTML. This is used by default when no position is defined.
- **after**
  This places the section's blocks after the current element.
- **before**
  This places the section's blocks before the current element.

- **prepend**
  This places the section's blocks inside the current element, at the beginning but not removing any html inside it.

- **append**
  This places the section's blocks inside the current element, at the end but not removing any html inside it.

- **replace**
  This replaces the current element with the section's blocks.

The blocks that are output can also be controlled by adding -block[number]. These can be chained as well. For example `tplsection-top-block1` will output the first block only, `tplsection-top-block2` will output the section block only and `tplsection-top-block1-block3` will output only the first and third blocks.

Full Example: (Default or -inside)

```
<div class="tplsection-top"><b>Existing text and HTML</b></div>
```

Resulting HTML:

```
<div class="tplsection-top">
      All the blocks placed in the 'top' section on the 'Site
Layout' page will appear here.
</div>
```

Full Example: (Changing the position using -before)

```
<div class="tplsection-top-before"><b>Existing text and
HTML</b></div>
```

Resulting HTML:

```
All the blocks placed in the 'top' section on the 'Site Layout' page
will appear here.
<div class="tplsection-top-before"><b>Existing text and
HTML</b></div>
```

Full Example: (Changing the position using -after-block2)

```
<div class="tplsection-top-after-block2"><b>Existing text and
HTML</b></div>
```

Resulting HTML:

```
<div class="tplsection-top-after-block2"><b>Existing text and
HTML</b></div>
The second block placed in the 'top' section on the 'Site Layout'
page will appear here.
```

Full Example: (Changing the position using -replace)

```
<div class="tplsection-top-replace"><b>Existing text and
HTML</b></div>
```

Resulting HTML:

```
All the blocks placed in the 'top' section on the 'Site Layout' page will appear here.
```

Full Example: (Changing the position using -prepend-block1-block2)

```
<div class="tplsection-top-prepend-block1-block2"><b>Existing text and HTML</b></div>
```

Resulting HTML:

```
<div class="tplsection-top-prepend-block1-block2">
The first and second blocks placed in the 'top' section on the 'Site Layout' page will appear here.
<b>Existing text and HTML</b>
</div>
```

Full Example: (Changing the position using -append)

```
<div class="tplsection-top-append"><b>Existing text and HTML</b></div>
```

Resulting HTML:

```
<div class="tplsection-top-append">
<b>Existing text and HTML</b>
All the blocks placed in the 'top' section on the 'Site Layout' page will appear here.
</div>
```

Template Class Type: Repeat

Purpose: Repeatables are used by list blocks to define the flow of the loop over each item in the list.

Format: tplrepeat-[name]

Samples: tplrepeat-parent, tplrepeat-listitem, tplrepeat-listitem-first

Template Class Type: Flags

Purpose: Flags are used as markers for the application to perform specific actions. This might be to define and input field to insert submitted data, or to insert hook variables that module can use.

Format: tplflag-[flagname]

Samples: tplflag-category-paging, tplflag-searchoption

Validating the styleguide.html File

With the `block=""` and `blockStyle=""` attributes, the style guide file itself will not validate as XHTML. However, these attributes are removed out when the style guide is processed, so the end result will/can validate. If you would like the style guide file to validate, you can use our custom Doctype with our extended DTD:

```
<!DOCTYPE html PUBLIC "-//Template//ITE XHTML Extension//EN"
          "http://www.buildertemplates.com/DTDs/ITEv1.2.dtd">
```

The config.php File

The config.php file contains configuration information about the template and defines 'layout sets' used by application. Template information defined includes template name, available colors, and version. This file is required.

A 'layout set' is a specified combination of columns, e.g. 2 columns, using the left and middle columns. These are defined with the 'CreateSet' function, explained below.

A default config file will need the following code.:

```
// Sample template config file
$config = new iwp_template_config();

// Set the name of the template
$config->SetAttribute('name', iwp_CurrentDirectory(__FILE__));

// Set the version of the template
$config->SetAttribute('version', '1.00');
$config->SetAttribute('colors', array('blue'));

// Using the default layouts
// This auto creates commonly used sets using the CreateSet
function.
$config->UseDefaultSets(1,2,3);

// Set the default layout set to use
$config->SetAttribute('defaultLayoutSet', '3columns');

// Recommended Logo Size
$config->SetAttribute('logoWidth', '285');
$config->SetAttribute('logoHeight', '40');

// Optional: Define which column holds the content.
$config->SetContentColumn(3);
```

These are the available function calls:

SetAttribute

This function sets the information about the current template. The template attributes that can be set are: name, version, colors, defaultLayoutSet, logoWidth, logoHeight.

name - This is the name of the template, it should be the same as the name of the directory. iwp_CurrentDirectory(__FILE__) will return the name of the directory.
version - This is the version number of the template. It should follow: [majorNumber].[minorNumber] where minor number is always 2 digits, if it is lower than 10 use a leading zero. e.g. 2.03
colors - This is an array of the colors that are used by the template. The color names passed in should reflect the names of .css files located in the /styles/ directory of the template.
defaultLayoutSet - This defines what layout this template should default to when not other valid layout set has been selected. The value should be the valid name of a layout set.
l**ogoWidth, logoHeight** - These are the recommend dimensions of the logo image. This is displayed to users on the 'Website Logo' page in the control panel.

Example Usage:

```
$config->SetAttribute('name', iwp_CurrentDirectory(__FILE__));
$config->SetAttribute('version', '1.00');
$config->SetAttribute('colors', array('blue', 'red'));
$config->SetAttribute('defaultLayoutSet', '3columns');
```

CreateSet

This function defines a 'layout set'. A 'layout set' is a specified combination of columns, e.g. 2 columns, using the left and middle columns. A template should only define layout sets that it supports within its styleguide.html file.

Usage:

```
$this->CreateSet([Set Id], [Set Name], [Columns Used]);
```

[Set Id] - This should be a unique name for the layout set. This is used in the code to handle and identify this layout set. This value should be a string of lowercase alphanumeric characters, with underscores also permitted.
[Set Name] - This is the name of the layout set that is displayed to users on the 'Site Layout' page when selecting the number of columns to use. This can be a string of any characters.
[Columns Used] - This is an array of the columns to use on the template. Available columns are: top, left, middle, right, bottom.

Examples:

```
$this->CreateSet('1column', GetLang('Layout_1Column'), array('top', 'middle', 'bottom'));
```

$this->CreateSet('2column_right', '2 Column, Middle and Right', array('top', 'middle', 'right', 'bottom'));

UseDefaultSets

This function automates the creation of 3 layout sets. This defines a 1 column layout, 2 column layout and a 3 column layout. These are defined by passing in the column numbers to the function.

Usage:
$config->UseDefaultSets([Column Numbers]);
[Column Numbers] - The columns numbers for which layout sets should be created. These should be passed in as separate arguements. Supported numbers are: 1, 2 and 3.

Examples:
$config->UseDefaultSets(1,2,3); (Creates 1, 2 and 3 column sets)
$config->UseDefaultSets(3); (Only creates a 3 column set)

SetContentColumn

This function defines which center column to place the content in. i.e. to display content in the left, right or middle columns. These are represented by numbers, 1, 2 and 3. By default the middle column is used to hold the content, so the default value for this function would be '2'.

Usage:
$config->SetContentColumn([Column Number]);
[Column Number] - The column number for which the content should be placed into. 1 = left, 2 = middle, 3 = right

Examples:
$config->SetContentColumn(1); (content is placed in the left column)

$config->SetContentColumn(3); (content is placed in the right column)

Appendix 1: Block Reference Table

Below is the table of blocks that are used by the application. The Block Name is the value used in the `block=""` attribute in the HTML. For example, for the list block, it is "list" and in the HTML it would be used like:

```
<div block="list" blockStyle="Standard"></div>
```

The 'Multiple' column defines whether there can be more than one style of a certain block.

Block Name	Description	Multiple
advancedSearch	Displays the advanced search and search results page.	No
breadcrumbTrail	Used whenever a breadcrumb trail needs to be used for any reason. It will usually only appear in the middle content area.	No
customContent	Used for any "Custom Content Blocks" created by the user on the Site Layout page. These only consist of a title and body HTML.	Yes
customImage	Block created on the Site Layout page of the Control Panel. This block decides how an image uploaded should be displayed.	Yes
htmlHead	By default this block is just the HTML between the `<head>` and `</head>` tags. This block does not require the `block=""` attribute.	No
list	Wide-ranging block used to output menus/lists created in the Control Panel. These can output a menu with links to content pieces, categories, users, etc or can be used to list content in categories, search results, etc.	Yes
pagingLinks	Used when a set of paging links are needed when there is more than one page to be displayed.	No
rssLink	Displays the 'Subscribe via RSS' icon and link. It links to the RSS page that is controlled by the block 'rssPage'.	No

rssPage	Displays the page that lists all of the RSS feeds available in the application.	No
smallSearch	Displays the quick search box. It is often placed in the header or in a side menu. Some templates may hard code the search box into a different position such as the header and if that is the case then this block is not necessary.	No
viewAuthor	Used when viewing an author's profile. Will display an author biography with a list of content items they've created.	No
viewCategory	Displays a category with its sub-categories listed, and content items placed into the category listed as well.	No
viewContent	Displays and formats a piece of content added in the control panel.	No

Appendix 2: Template Classes Reference Table

This is a list of all the template class variables that are available in the system, with the exception of language variables.

Note: The list of language variables can be found in the /language/en-US/frontend.ini file.

Global Classes

These can be used anywhere within the style guide and are not limited to a specific block.

Variables

- **tplvar-site-title**
  Outputs the website title. The value is set on the Website Settings page of the Control Panel.

- **tplvar-site-metadescription**
  Outputs the meta description for the current page. The default is set on the Website Settings page of the Control Panel whilst content and categories can define their own. It is recommend to use this on a `<meta>` tag using: `tplvar-site-metadescription-content`.

- **tplvar-site-metakeywords**
  Outputs the meta keywords for the current page. The default is set on the Website Settings page of the Control Panel whilst content and categories can define their own. It is recommend to use this on a `<meta>` tag using: `tplvar-site-metakeywords-content`.

- **tplvar-site-color**
  Outputs the path to the color.css file in use by the current template. It is recommend to use this on a `<link>` tag using: `tplvar-site-color-src`.

- **tplvar-site-logoimage**
  Outputs the path to the current logo image (if there is one). It is recommend to use this on an `<img>` tag using: `tplvar-site-logoimage-src`.

- **tplvar-site-siteurl**
  Outputs the URL of the website.

- **tplvar-site-slogan**
  Outputs the slogan of the website. This value is set on the Website Settings page of the Control Panel.

- **tplvar-site-slogan**
  Outputs the slogan of the website.

Conditionals

- **tplcond-showif-haswebsitelogo**
  Displays the current element only if the website has a logo image.

- **tplcond-hideif-haswebsitelogo**
  Displays the current element only if the website does not have a logo image.

- **tplcond-addcolumncounttoclass-x**
  Takes `x` then adds the current number of columns to it and in turn adds that class to the current element. For example, having `tplcond-addcolumncounttoclass-myClass` on a page that has 2 columns displayed would result in a class called `myClass2` added to the current element.

- **tplcond-hideif-1column-2columns-3columns**
  Hides the current element depending on the attached modifiers. For example, having `tplcond-hideif-1columm` would result in the current element being hidden if the page only has 1 column. Having `tplcond-hideif-1column-2columns` would result in the current element being hidden if the current page has 1 or 2 columns (i.e. only displaying on a 3 column page).

Block Specific Classes for “advancedSearch”

These styles can only be used inside the “advancedSearch” block.

Variables

- **tplvar-advancedsearch-pageurl**
  Outputs the URL to the advanced search page and is also the location of where the search form submission should go to.

- **tplvar-search-message**
  Outputs any message returned by a search, such as missing fields.

- **tplvar-search-query**
  Outputs the search query submitted by the website visitor.

- **tplvar-search-showadvanced**
  Outputs code that determines whether to show the current element or hide

it depending on whether the visitor came from the advanced search page or the integrated search box. If they came from the advanced search page, it will show the element. The style code that is output is only useful in the `style=""` attribute. It is recommended to use `tplvar-search-showadvanced-styleprepend`.

- **tplvar-search-hideadvanced**
  The opposite of `tplvar-search-showadvanced`. It will hide the element if the visitor came from the advanced search page. It is recommended to use `tplvar-search-hideadvanced-styleprepend`.

- **tplvar-search-startdate**
  Outputs the start date that the website visitor has submitted in their search.

- **tplvar-search-enddate**
  Outputs the end date that the website visitor has submitted in their search.

- **tplvar-search-submitbuttonaction**
  Outputs the Javascript function used to submit the advanced search form.

Conditionals

- **tplcond-search-titlecheckbox**
  Adds `checked="checked"` to the current element if the website visitor had the “Title” checkbox ticked when they submitted the advanced search form.

- **tplcond-search-contentcheckbox**
  Adds `checked="checked"` to the current element if the website visitor had the “Content” checkbox ticked when they submitted the advanced search form.

- **tplcond-search-authorcheckbox**
  Adds a `checked="checked"` to the current element if the website visitor had the “Author” checkbox ticked when they submitted the advanced search form.

- **tplcond-search-showifresults**
  Hides the current element unless there are search results to show.

Flags

- **tplflag-search-fromadvanced**
  Returns 1 or 0 if the visitor came from the advanced search page or not. Common use for this would be to set a hidden input field's `value` attribute using: `tplflag-search-fromadvanced-value`

- **tplflag-searchAdvancedOption**
  Used by JavaScript to hide or show options considered to be advanced search options when the user clicks the hide/show “Advanced Options” link.

- **tplflag-search-categorylist**
  Used to populate a select box with `option` element for the categories on the website.

- **tplflag-search-contenttypelist**
  Used to populate a select box with `option` element for the content types on the website.

- **tplflag-search-paging**
  Inserts the paging (next, previous, etc) links for the search results into the current element.

- **tplflag-search-results**
  Inserts the HTML list of search results into the current element.

Block Specific Classes for “breadcrumbTrail”

These styles can only be used inside the “breadcrumbTrail” block.

Variables

- **tplvar-breadcrumb-url**
  Used inside the breadcrumb loop and outputs the URL to a page. It is recommended to be used on an anchor tag (`<a>`) using the `href` attribute, like: `tplvar-breadcrumb-url-href.`

- **tplvar-breadcrumb-label**
  Outputs the name of the page in the trail. It is recommended to be used on an anchor tag (`<a>`) or an element inside the anchor tag. For example: `tplvar-breadcrumb-label`

Conditionals

- **tplcond-breadcrumb-iflinked**
  Must be used inside the element with `tplrepeat-breadcrumb`. The HTML element and its children that have this class will be displayed if the output page has a link, i.e. all pages in the trail except the last/current page. Otherwise, it will not be output.

- **tplcond-breadcrumb-ifnotlinked**
  Must be used inside the element with `tplrepeat-breadcrumb`. The HTML element and its children that have this class will be displayed if the output

page doesn't not have a link, i.e. the last/current page. Otherwise, it will not be output.

Repeatables

- **tplrepeat-breadcrumb**
  Specifies the parent element for the breadcrumb trail loop. All HTML inside is repeated.

Block Specific Classes for "customContent"

These styles can only be used inside the "customContent" block. Custom content blocks are created from the Site Layout page in the Control Panel and can consist of a title and any user created content.

Variables

- **tplvar-customcontent-title**
  Outputs the title of the content block.

- **tplvar-customcontent-name**
  Outputs the name of the content block. The name is not usually displayed to end users. This field might be useful to apply additional CSS classes to elements if named appropriately. For example, using `tplvar-customcontent-name-classprepend` and naming the block "QuoteBlock GreenBlock" would apply the CSS classes to the object.

- **tplvar-customcontent-content**
  Outputs the content of the block.

Block Specific Classes for "customImage"

These styles can only be used inside the "customImage" block. Custom image blocks are created from the Site Layout page in the Control Panel and consist of an image and optionally a linked URL.

Variables

- **tplvar-image-path**
  Outputs the path to the image that was uploaded. It is recommend to use this on an `<img>` tag using the `src` attribute, like: `tplvar-image-path-src`.

- **tplvar-image-label**
  Outputs the label/alternative text for the image that was uploaded. It is recommend to use this on an `<img>` tag using the `alt` attribute, like: `tplvar-image-label-alt`.

- **tplvar-image-width**
  Outputs the width of the image that was uploaded. It is recommend to use this on an `<img>` tag using the `width` attribute, like: `tplvar-image-width-width`.

- **tplvar-image-height**
  Outputs the height of the image that was uploaded. It is recommend to use this on an `<img>` tag using the `height` attribute, like: `tplvar-image-height-height`.

- **tplvar-imagelink-url**
  If the image has a link, outputs the URL of that link. It is recommend to use this on an anchor tag (`<a>`) using the `href` attribute, like: `tplvar-imagelink-url-href`.

- **tplvar-imagelink-target**
  If the image has a link this will output the target window for that link. It is recommend to use this on an anchor tag (`<a>`) using the `target` attribute like: `tplvar-imagelink-target-target`.

Conditionals

- **tplcond-ifimagenotlinked**
  The element with this class is displayed if there is no link for the image.

- **tplcond-ifimagelinked**
  The element with this class is displayed if there is a link for the image.

Block Specific Classes for “htmlHead”

The “htmlHead” block (i.e. the `<head>` and `</head>` tags) makes use of the global variables only. It does not currently have a set of its own CSS classes.

Block Specific Classes for “list”

These styles can only be used inside the “list” block. List blocks output menu/lists created from the control panel.

Variables

- **tplvar-list-title**
  Outputs the main title of the list.

- **tplvar-listitem-title**
  Outputs the title of the list item. This is available for all lists.

- **tplvar-listitem-url**
  Outputs the url of the list item. This is available for all lists. It is

recommended to use this on an anchor tag (<a>) using the `href` attribute, like: `tplvar-listitem-url-href`.

- **tplvar-listitem-target**
  Outputs the target window of the list item's link. This is available for all lists. It is recommended to use this on an anchor tag (<a>) using the `target` attribute, like: `tplvar-listitem-target-target`.

- **tplvar-listitem-authors**
  Outputs the list of authors of the current content item. It includes their names linked to their user profile. This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-categories**
  Outputs the list of categories the current content item is categorized in. It includes their names linked to view the category. This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-contenttypename**
  Outputs the name of the content type the current content item belongs to. This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-publishdate**
  Outputs the publish/start date of the content item. Outputs only the date and is formatted according to the "Long Date Format" setting on the Website Settings page of the Control. This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-publishdatewithlabel**
  Outputs the publish/start date of the content item with the label "Published " prefixed (uses the language variable “ListPublishedOnDate”). This is formatted according to the "Long Date Format" setting on the Website Settings page of the Control. This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-publishdate-monthshort**
  Outputs the short name for the month of the publish/start date of the content item. e.g. "Jan" for January. This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-publishdate-monthlong**
  Outputs the full name for the month of the publish/start date of the content

item. e.g. "January". This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-publishdate-yearlong or tplvar-listitem-publishdate-year**
  Outputs the full 4-digit year of the publish/start date of the content item. e.g. "2009". This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-publishdate-yearshort**
  Outputs a 2-digit year of the publish/start date of the content item. e.g. "09". This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-publishdate-daynum**
  Outputs the day of the month number of the publish/start date of the content item. e.g. "5" or "15". This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-publishdate-monthnumber**
  Outputs the month of the year in number form of the publish/start date of the content item. e.g. "5" or "12". This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-publishdate-monthnumberwithzero**
  Outputs a 2-digit the month of the year in number form of the publish/start date of the content item. e.g. "05" or "12". This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-summary**
  Outputs the summary of the content item. This is only available when outputting a generated list of content and not for a custom link list or RSS list.

- **tplvar-listitem-content**
  Outputs the full body content of the content item. This is only available when outputting a generated list of content and not for a custom link list or RSS list.

Conditionals

- **tplcond-listitem-iflast**
  If the current list item is the last in the list, the element containing this class will be output instead of other `tplrepeat` occurrences in this list.

- **tplcond-listitem-ifnotlast**
  If the current list item is not the last in the list, the element containing this class will be output instead of other `tplrepeat` occurrences in this list.

- **tplcond-listitem-iffirst**
  If the current list item is the first in the list, the element containing this class will be output instead of other `tplrepeat` occurrences in this list.

- **tplcond-listitem-ifnotfirst**
  If the current list item is not the first in the list, the element containing this class will be output instead of other `tplrepeat` occurrences in this list.

Repeatables

- **tplrepeat-parent**
  Specifies the parent element for the loop, all HTML inside will be looped over.

- **tplrepeat-list**
  Specifies the default list item to be output, this only needs to be used if any `tplrepeat`s like `tplrepeat-listitem-ifnochildren` are used. It is used as the “else” or “fallback” option if none of the conditions are met.

- **tplrepeat-listitem-ifnochildren**
  If the current list item does not have any children, the element containing this class will be output instead of other `tplrepeat` occurances.

- **tplrepeat-listitem-ifchildren**
  If the current list item has children, the element containing this class will be output instead of other `tplrepeat` occurances.

- **tplrepeat-parent2**
  Specifies the parent element for the child loop, all HTML inside will be looped over for the child loops.

- **tplrepeat-list2**
  Specifies the default list item to be output. At the moment this is the only `tplrepeat-` that can be used on a child list. The following classes are available and work just the same as the ones listed below without the '2': `tplvar-listitem2-url-href, tplvar-listitem2-title, tplvar-listitem2-target`.

Flags

- **tplflag-listitem-contentdetails**
  Specifies the location of the current content item's content details. i.e. The publish date, authors, categories, etc. Hook variables that can be used by modules are placed inside the element with this flag class.

Block Specific Classes for “pagingLinks”

Variables

- **tplvar-paging-prevlink**
  Outputs the URL of the previous page if there is one.

- **tplvar-paging-nextlink**
  Outputs the URL of the next page if there is one.

- **tplvar-paging-url**
  Outputs the URL to the page number. It is recommended to use this on an anchor tag (`<a>`) using the `href` attribute, like: `tplvar-paging-url-href`.

- **tplvar-paging-number**
  Outputs the page number.

Repeatables

- **tplrepeat-paging-ifnotlinked**
  The element with this class is used if the current page number to be output is the current page being viewed and should not be linked.

- **tplrepeat-paging-iflinked**
  The element with this class is used if the current page number to be output should be linked.

Block Specific Classes for “rssLink”

These styles can only be used inside the “rssLink” block.

Variables

- **tplvar-rsslink-pageurl**
  Outputs the URL of the page containing all the RSS feeds on the website.

Block Specific Classes for “rssPage”

These styles can only be used inside the “rssPage” block.

Variables

- **tplvar-feedpage-url**
  Outputs the URL to the current RSS feed in the loop. It is recommended to use this on an anchor tag (`<a>`) using the `href` attribute, like: `tplvar-feedpage-url-href`.

- **tplvar-feedpage-label**
  Outputs the title (list name or category name) of the current RSS feed in the loop.

- **tplvar-feedpage-indentstyle**
  Outputs some CSS that increases the `padding-left` CSS property of the current item if it is a child item. It is recommended to use the `styleprepend` attribute, like: `tplvar-feedpage-indentstyle-styleprepend`.

Conditionals

- **tplcond-ifhascategoryfeeds**
  Element with this class is only displayed if there are one or more category RSS feeds to show.

- **tplcond-ifhasotherfeeds**
  Element with this class is only displayed if there are one or more "other" RSS feeds to show. "Other" RSS feeds currently only consist of RSS feeds created by lists/menus.

Repeatables

- **tplrepeat-categoryfeed**
  Element with this class is repeated for each category feed there is to display.

- **tplrepeat-otherfeed**
  Element with this class is repeated for each other RSS feed there is to display.

Block Specific Classes for “smallSearch”

These styles can only be used inside the “smallSearch” block.

Variables

- **tplvar-smallsearch-pageurl**
  Outputs the URL of advanced search page which is where the search query is submitted to.

Block Specific Classes for “viewAuthor”

These styles can only be used inside the “viewAuthor” block.

Variables

- **tplvar-author-name**
  Outputs the full name of the author/user.

- **tplvar-author-imageurl**
  Outputs the URL to the profile picture.

- **tplvar-author-imagewidth**
  Outputs the width of the profile picture. It is recommend to use this on an

`<img>` tag using the `width` attribute, like: `tplvar-author-imagewidth-width`.

- **tplvar-author-imageheight**
  Outputs the height of the profile picture. It is recommend to use this on an `<img>` tag using the `height` attribute, like: `tplvar-author-imageheight-height`.

- **tplvar-author-biography**
  Outputs the biography text of the author/user.

Flags

- **tplflag-author-contentlist**
  Specifies the element where the list of the author's content should be displayed.

- **tplflag-author-image**
  Specifies the profile picture element so that if the current user/author does not have a profile picture, the element is not output.

Block Specific Classes for “viewCategory”

These styles can only be used inside the “viewCategory” block.

Variables

- **tplvar-category-title**
  Outputs the category name.

- **tplvar-category-description**
  Outputs the description of the category.

Flags

- **tplflag-category-breadcrumb**
  Specifies the element where the breadcrumb trail should be displayed.

- **tplflag-category-subcategories**
  Specifies the element where the list of sub-categories should be displayed.

- **tplflag-category-paging**
  Specifies the element where the paging links for the category (if there are any) should be displayed.

- **tplflag-category-contentlist**
  Specifies the element where the list of content associated with the category should display.

Block Specific Classes for “viewContent”

These styles can only be used inside the “viewContent” block.

Variables

- **tplvar-content-title**
  Outputs the title of the content item.

- **tplvar-content-authors**
  Outputs the list of authors of the current content item. It includes their names linked to their user profile.

- **tplvar-content-categories**
  Outputs the list of categories the current content item is categorized in. It includes their names linked to view the category.

- **tplvar-content-contenttypename**
  Outputs the name of the content type the current content item belongs to.

- **tplvar-content-publishdate-monthshort**
  Outputs the short name for the month of the publish/start date of the content item. e.g. "Jan" for January.

- **tplvar-content-publishdate-monthlong**
  Outputs the full name for the month of the publish/start date of the content item. e.g. "January".

- **tplvar-content-publishdate-yearlong or tplvar-listitem-publishdate-year**
  Outputs the full 4-digit year of the publish/start date of the content item. e.g. "2009".

- **tplvar-content-publishdate-yearshort**
  Outputs a 2-digit year of the publish/start date of the content item. e.g. "09".

- **tplvar-content-publishdate-daynum**
  Outputs the day of the month number of the publish/start date of the content item. e.g. "5" or "15".

- **tplvar-content-publishdate-monthnumber**
  Outputs the month of the year in number form of the publish/start date of the content item. e.g. "5" or "12".

- **tplvar-content-publishdate-monthnumberwithzero**
  Outputs a 2-digit the month of the year in number form of the publish/start date of the content item. e.g. "05" or "12".

- **tplvar-content-summary**
  Outputs the summary of the content item.

- **tplvar-content-content**
  Outputs the full body content of the content item.

Flags

- **tplflag-content-details**
  Specifies the location of the current content item's content details. i.e. The publish date, authors, categories, etc. Hook variables that can be used by modules are placed inside the element with this flag class.

- **tplflag-content-paging**
  Outputs the paging links if the current content item has more than one page.

- **tplflag-content-modules**
  Denotes the default content module output area. Modules can hook into any variable on the page if specified. If they do not specify a location, they will be output inside the element containing this class.