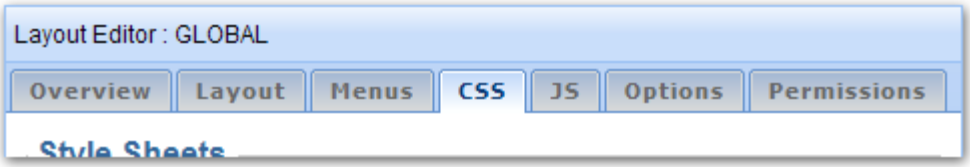
CSS Tab

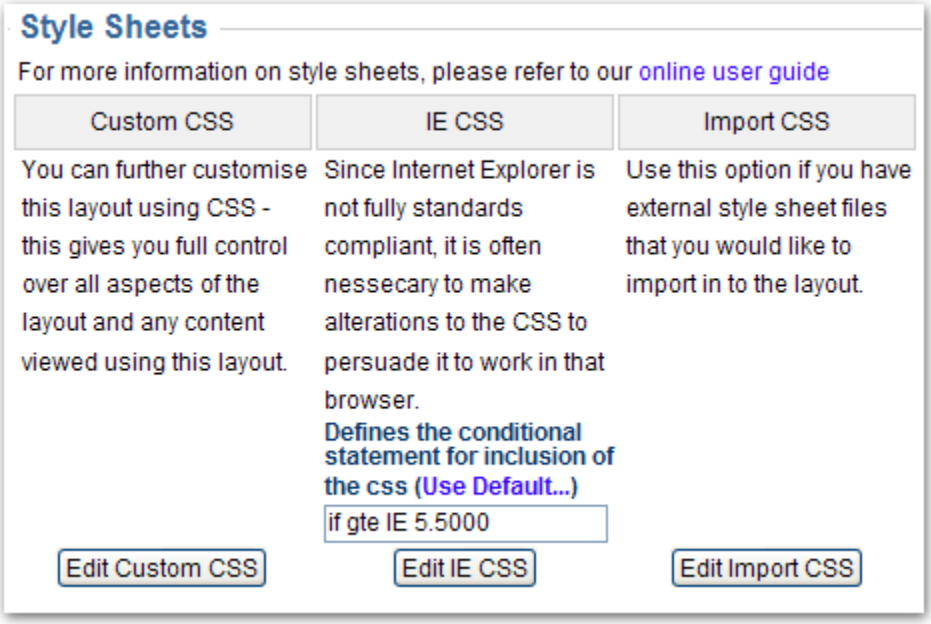
This tab is accessed from the Layout Manager in Theme Builder 3.0 and above.

The CSS tab allows you to define custom CSS for a layout and optionally disable the inclusion of various CSS resources to gain more control over the overall CSS used in the layout.



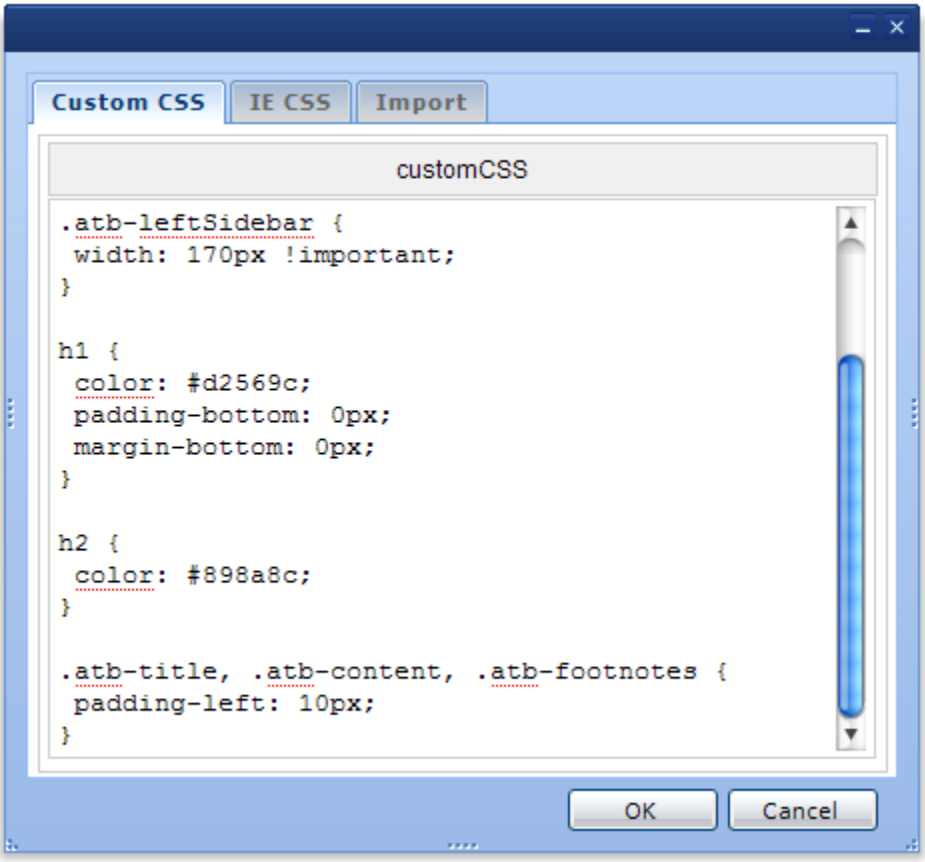
Style Sheets

There are three fully-editable style sheets within a layout:



- **Custom CSS** – This CSS will be used for all browsers.
- **IE CSS** – This CSS will only be included for Internet Explorer version 5.5 or above by default
- **Import CSS** – This option allows you to import one or more external style sheets.

To edit one of the style sheets, click the associated button to display the CSS editor window:



You can quickly switch between the CSS resources by clicking the tabs shown at the top of the editor window.

Once you have finished making changes, click the "OK" button. If you want to discard changes, click the "Cancel" button.

> *These CSS resources use a different form of inheritance. Please see Style Sheet Inheritance for more details.*

Changing the Internet Explorer version for the IE CSS

You can change which version of Internet Explorer the IE CSS will be used on by editing the "conditional if" setting above the "Edit IE CSS" button.

For more information on this feature, please see About Conditional Comments in the Microsoft Developer Network.

Importing External CSS Resources

If you have externally hosted CSS files you can use the Imports tab to import them in to a layout.

Normally this approach won't be required with Theme Builder 3 because you can use Layout Hierarchy to centrally define your CSS and then have child layouts automatically inherit that CSS and any subsequent changes.

> *When you import a CSS file the syntax is generally in the form:*
>
> ```
> @import url(http://www.adaptavist.com/download/attachments/7286/bar.css);
> ```
>
> ⚠️ *You should never put quotes around the URL to the CSS file as this will cause problems with some browsers!*
>
> *Some browsers will block the imported CSS file if it comes from a different domain to your wiki. For example, if your wiki was on www.foo.com and your CSS was on www.bar.com then depending on the browser settings it might not get imported.*
>
> *To work round that issue, it's always best to store the CSS file on the same domain as your wiki, for example you can attach the .css file to a page in your wiki. For more information see Creating CSS Files.*

If your wiki can be accessed from multiple domain names and the CSS file is stored in the wiki, you should use a relative URL, for example:

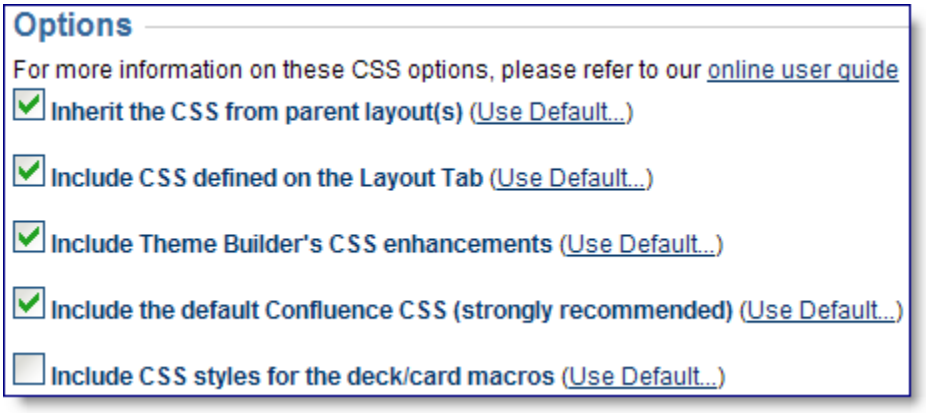
```
@import url(/download/attachments/7286/bar.css);
```

Imported CSS Resources are loaded before the Custom CSS and IE CSS. For more details, please see Style Sheets 3.x.

If you are using a Content Delivery Network (CDN) to host CSS files externally, please refer to the CDN documentation for information regarding best practice regarding importing of CSS files.

Options

There are several options which allow you to further customise which CSS resources are used by the layout. These options are intended for "power users" who need extreme levels of control in their quest for perfection.



Inherit the CSS from the parent layout(s)

By default, a layout will inherit the CSS defined in it's parent layout. For more information on this process, see Layout Hierarchy and Style Sheet Inheritance

There may be cases where you want to start from scratch, and not import the Custom, IE and Import CSS from the parent layout in which case you can disable this option.

Include CSS defined in the Layout Tab

The panel CSS is defined using the Panel Editor in the Layout Tab. It's a quick and easy way to set the majority of styles such as backgrounds, borders, text settings, etc.

If you want more control, we'd recommend simply enhancing the Panel CSS using the Custom CSS and IE CSS options because that allows you to still make the more macroscopic changes using the Panel Editor.

However, if you want ultimate control you can disable the Panel CSS resource completely with this option.

💡 **Tip:** You can turn off this option and then restrict CSS editing privileges to prevent people from customising the CSS in your layout.

Include Theme Builder's CSS enhancements

Theme Builder has a small amount of "in-built" CSS which ensures that certain elements (such as access key hits, page trees, etc) display correctly.

If you want complete control over these items, you can turn off the inbuilt CSS with this option.

The exact contents of the inbuilt CSS change slightly from version to version, but they will be similar to this:

```
/* pagetree2 */
.pagetree2 .cell1, .pagetree2 .cell2, .pagetree2 .cell3, .pagetree2 .cell4, .pagetree2 .cell5 {
    white-space:nowrap;
}

/* access keys */
#accessKeys { margin: 0; padding: 0; }
```

```
#accessKeys dt { margin: 0; padding: 0; position: absolute; top: -9999px; left: 0; }
#accessKeys dd { margin: 0; padding: 0; }
#accessKeys dd a { position: absolute; top: -9999px; left: 0; }
#accessKeys dd a:focus, #accessKeys dd a:active { background-color: #3556a2 !important; color:
#fff !important; top: 0; font-size: 1.2em; padding: .5em; }

/* make headings less ugly */
.atb-page h1, .atb-page h2, .atb-page h3, .atb-page h4, .atb-page h5, .atb-page h6 {
 background-color: transparent;
 border-style: none;
 padding: 0px;
}
.atb-title h1 {
 font-size: 18px;
 line-height: 1.0em;
 margin-top: 0px;
}

/* make breadcrumbs less rancid */
.breadcrumbs {
 background-color: transparent;
 border: 1px none transparent;
}

/* labels in menus */
div.dynarch-horiz-menu .label, div.dynarch-popup-menu .label {
 font-weight: normal;
}

/* allow panel css to set design of menus in .atb-menu */
.atb-menu div.dynarch-horiz-menu {
 background-color: transparent;
 background-image: none;
 border-style: none;
}

/* remove margins from forms */
.marginlessForm {
 margin: 0px;
}

/* don't force search box to be 100% width */
.confluence-searchbox {
 width: auto;
 white-space: nowrap;
}

/* remove padding from PageContent */
.PageContent, .pagecontent {
 padding: 0px;
}

/* ui table (e.g. attachments list) */
.tableview th, .tabletitle, .pageSectionHeader {
 border-bottom-width: 1px;
}
.tableview th, div.tabletitle {
 font-weight: bold;
}
.mode-view-attachments .tableview td {
 border-top: 1px solid #fff;
 vertical-align: middle;
}
.tableview td {
 border-bottom: 1px dotted #ccc;
}
.tableview tr:hover td {
 border-bottom: 1px solid #000;
}
.rowAlternate {
 background-color: #f7f7f7;
```

```
}

/* activity screens */
.context-space-activity .pagebody {
 padding: 0px;
}

/* no image borders */
a img {
 border-style: none;
}

/* hide accessibility features by default */
.accessibility {
 display: none;
}

/* left slider */
.atb-leftslider {
 cursor: w-resize !important;
}

/* right slider */
.atb-rightslider {
 cursor: e-resize !important;
}

/* allow special effects on tables, etc. */
.layout {
 direction:ltr;
}

/* inhibit selection where possible */
.noselect {
 -moz-user-select: none;
 cursor: default;
}

/* line spacing */
body, p, li, ul {
 line-height:1.6em;
}

/* citation */
cite:before { content: "\""; }
cite:after { content: "\""; }
cite, cite:before, cite:after {
 font-family: Georgia, "Times New Roman", serif;
 font-style: italic;
}

/* insertions and deletions */
ins {
 background-color: #DBFFDB;
}
del {
 background-color: #FFE5E5;
}

/* make italics more readable */
i, em {
 letter-spacing: 1px;
}

/* tables */
.confluenceTable {
 border: 1px solid #B2B2B2;
 border-collapse: collapse;
}
.confluenceTable th.confluenceTh {
 background-color: #F5F5F5;
```

```
 border: 1px dotted #B2B2B2;
 font-weight: bold;
 font-size: small;
 padding: 3px 4px 3px 4px;
}
.confluenceTable .confluenceTd {
 border: 1px dotted #B2B2B2;
 padding: 3px 4px 3px 4px;
}

/* ExtJS unselectable panels */
.x-unselectable { cursor: default; }

/* skip navigation */
@media screen, print, handheld, projection, tv {
 .non-visual {display: none; visibility: hidden;}
}

/* sliders */
.atb-slider .atb-slider-img {
 width: 9px;
 height: 20px;
}
.atb-leftslider .atb-leftslider-img {
 background-position: top right;
}
.atb-rightslider .atb-rightslider-img {
 background-position: top left;
}

.quicklinks {
 overflow: auto;
}
.quicklinks div {
 clear: none;
 float: left;
 margin-bottom: 4px;
 width: 49%;
}

/* Stop menu items wrapping */
div.dynarch-horiz-menu table tr td {
 white-space: nowrap;
}

/* Provide support for menu spacers */
div.dynarch-horiz-menu table tr td.spacer {
 width: 100%;
}
div.dynarch-horiz-menu table tr td.spacer div {
 display:none;
}

.poweredBy, .poweredBy a {
 color:#909090;
 font-size:small;
}
```

Include Atlassian's default CSS

As of Theme Builder 3.0, we have started to include Atlassian's master Confluence style sheet, known as "main-action.css". This contains all the core styles used by Confluence, including those for the page editor, the browse space screens, most bundled macros and more.

Because Atlassian make changes to this style sheet from one version of Confluence to the next, including it ensures you get the correct basic styles for your wiki regardless of which version of Confluence you are using.

However, because it contains so many styles it can sometimes get in the way of your own style sheets. Should the need arise, you can disable Atlassian's Confluence style sheet with this option (not recommended as you'll need to add quite a lot of Custom CSS to make up for the missing styles).

Include CSS styles for the deck & card macros

In Theme Builder you can now include the aqua, tan, red & green styles in the combined css resource for use with the composition plugin's deck & card macros. Enabling this option allows you to quickly and easily apply some basic styles to these macros.

💡 **Tip** You will need to have the free Composition Plugin installed in order to use the deck and card macros.

Basic Use

Add the deck & card macros to a page to create a tabbed panel which is ideal for separating out blocks of related content

```
{deck:id=my_deck}
  {card:label=Card 1}
    This is the first tab's content
  {card}
  {card:label=Card 2}
    This is the second tab's content
  {card}
{deck}
```

Which gives:

Unknown macro: {composition-setup}

Unknown macro: {card}

This is the first tab's content

Unknown macro: {card}

This is the second tab's content

Unknown macro: {deck}

Add CSS styles to your deck & card macros

To apply a CSS style simply add the appropriate class to the deck macro

```
{deck:id=my_deck|class=green}
  {card:label=Card 1}
    This is the first tab's content
  {card}
  {card:label=Card 2}
    This is the second tab's content
  {card}
{deck}
```

Which gives:

Unknown macro: {card}

This is the first tab's content

Unknown macro: {card}

This is the second tab's content

Unknown macro: {deck}

*The other classes you can you use are **aqua**, **tan** and **red***

FAQ's

The various CSS resources used by the theme depend on various settings. You can view more information on the Style Sheets 3.x page, including a list of all the menu style sheets, etc.

You can find the URL's to the style sheets by viewing the HTML source of a page and looking for the `<link>` tags at the top. If you browse to the URL's (remember to add in the path to your wiki at the start) you'll be able to view the CSS in the browser.

Currently the only browser-specific CSS you can add is for Internet Explorer 5.5 and above using the IE CSS resource.

In cases where you absolutely must apply a style to a specific version of a specific browser, there's a useful list of known CSS hacks that you can refer to. It should be noted that we don't recommend using CSS hacks - it's best to find an alternate way of applying your styles wherever possible.

Define your main CSS in a parent layout and then create child layouts based on that parent layout - by default the CSS defined in the parent layout will be included in all the child layouts. Changing the CSS in the parent layout will instantly update all the child layouts.

You can add additional CSS in the child layouts using the Custom CSS and IE CSS resources. If you add styles to the child layout that are already defined in the parent layout, they will override the styles defined in the parent layout because they are output after the parent layout styles. (hope that makes sense!)

The "Merge this layout's CSS with it's parent's" setting allows you to define whether a child layout also includes the CSS from the parent layout.

You can find more information on our Style Sheet Inheritance page.

Simply include the appropriate panel class in your style definition. For example, if you wanted to make the Heading 1 text in the left sidebar red, you would define:

```
.atb-leftSidebar h1 {
 color: #ff0000;
}
```

For a list of the available panel classes, please see Panel Classes and IDs.

The easiest way to solve this problem is to make your style definition more "specific" than that of the inbuilt CSS.

To make styles more specific, simply include more of the "path" to the element you want to style. For example, you could create a specific style for the text used in list items in the content panel as follows:

```
.atb-page .atb-content li {
 color: #008800;
}
```

Internet Explorer has a really nasty bug. It only looks at the last class!

For example, let's say you have CSS like this:

```
.foo {
 color: green;
}
.bar.foo {
 color: red;
}
```

And then somewhere on a wiki page the following notation:

```
{div:class=foo}This should be green{div}
{div:class=foo bar}This should be red{div}
```

In all other browsers, the text colours will be correct. However in Internet Explorer all of the text would be red! This is because Internet Explorer sees `.bar.foo` as just `.foo` (it only looks at the last part) and because the style for `.bar.foo` comes after the style for `.foo` it overrides the style for `.foo` making that text red instead of green.

To work around this problem, you need to make slight tweaks, for example:

```
.foo {
 color: green;
}
.foo.bar {
 color: red;
}
```

So Internet Explorer sees `.foo.bar` as `.bar`. Obviously, you're still going to get problems, but that's just something we have to live with until Microsoft manage to make a standards compliant browser (which is extremely unlikely).

Some browsers (Internet Explorer) and some web pages (generally XHTML pages) result in case-sensitive CSS.

For example, if you had this on a wiki page:

```
{span:class=FooBar}Wibble!{span}
```

Your style definition would need to be:

```
.FooBar {
 color: blue;
}
```

Read Layout Hierarchy to find out 🙂

See Also

The following pages contain additional useful information relating to style sheets:

- Style Sheet Inheritance - style sheet resources accessed from this tab use a different type of inheritance to the rest of the theme
- Style Sheets 3.x - technical overview of the style sheets used in Theme Builder 3.x
- Panel Classes and IDs - if you need to specify CSS that affects a specific panel, this lists the classes you can use to make your CSS more specific.
- Style Sheets Overview - this shows how various CSS resources are included in the layout.
- 70 CSS tips and tricks - this external guide has loads of useful CSS tips and tricks but be warned that not all of them will work in a dynamic application like Confluence.