# Working with Flags

Flags allow you simplify switching logic and improve performance within your themes...

Note: For best results we recommend using Theme Builder 3.3.6 or later as many more macros were made flag-aware.

## What are Flags?

A flag is a simple named register that stores a state:

- `true` - The flag is set
- `false` - The flag is *not* set

If you use a flag that's not been defined, it's treated as having state `false`.

Several macros are flag-aware, allowing you to toggle them on or off based on the state of one or more flags. You can also use flags to toggle the effect of parts of your CSS style sheets.

Flags are cleared after every page view so they are unique to the current page that the current user is viewing.

## How do I set a flag?

To set, and subsequently un-set a flag, you use the set-flag macro, for example the following markup will set a flag called foo, and then un-set it:

```
{set-flag:foo}
{set-flag:foo|state=false}
```

You can set flags in a wiki page or in a theme panel.

In most cases, you'll want to conditionally set a flag based on some other criteria, for example:

```
{builder-show:title=Forum|recurse=true}
 {set-flag:in-the-forum}
{builder-show}
```

The markup above would set the 'in-the-forum' flag only when the user is looking a page called 'Forum' or any of it's child pages (due to the 'recurse=true').

The macros most commonly used to set and un-set flags in this manner are:

- builder-show macro
- builder-hide macro
- bubbles-show macro
- bubbles-hide macro

These show/hide macros will show or hide their contents based on the parameters you set, allowing you to determine when the set-flag macro is triggered to set or un-set a flag.

## How do I unset a flag?

To un-set (clear) a flag, set it's state to false:

```
{set-flag:in-the-forum|state=false}
```

You can also completely remove a flag by setting remove:

```
{set-flag:in-the-forum|state=remove}
```

## What can I do with flags?

You can use flags to toggle the rendering of other macros and wiki markup. For example, using our 'in-the-forum' flag above we could show some extra content in a theme sidebar when that flag is set:

```
{builder-show:flag=in-the-forum,foo}
 this stuff will only be shown when either the 'in-the-forum' and/or 'foo' flag has been set
{builder-show}
```

Or we could show some content when the flag is *not* set:

```
{builder-show:notflag=in-the-forum,foo}
 this stuff will only be shown if neither the 'in-the-forum' and 'foo' flags have been set
 (in other words, if the 'in-the-forum' or 'foo' flags are set, ths won't be shown)
{builder-show}

{builder-hide:flag=in-the-forum,foo}
 this is does the same as the builder-show markup shown above, we've changed 'notflag' to 'flag'
 so it gets hidden when either of the flags are set
{builder-hide}
```

## Which macros support flags?

The following macros have `flag` and `notflag` parameters (requires Builder 3.3.6 or above):

- builder-show macro
- builder-hide macro
- compound-menuitem macro
- fav-menu macro
- import macro
- menu macro
- menuitem macro
- menulink macro
- set-flag macro
- submenu macro
- sub-submenu macro
- watch-menu macro

The flag parameters are as follows:

- `flag` – defines which flags must be set in order for the macro to be processed
- `notflag` – defines which flags must *not* be set in order for the macro to be processed.

## When should I use flags?

Flags become useful in more complex theme customisations because they allow you to separate out a lot of the logic that determines what should be displayed to the end-user. We'll cover some of the key use cases below...

### Setting defaults

There are many cases where you'll want to set a default state and then override it with a known state based on various criteria. If none of the known-states are found, you'll be left with the default state.

A simple example is determining whether a user is logged in or not, and whether they are a member of staff:

```
{set-flag:anonymous-user}

{builder-show:group=confluence-users}

  {set-flag:anonymous-user|clear=true}
  {set-flag:registered-user}

  {builder-show2:group=staff-group}
    {set-flag:staff-user}
  {builder-show2}

{builder-show}
```

The result will be:

- If the user is not logged in, only the anonymous-user flag will be set. (Default state)
- If the user is logged in, the 'anonymous-user' flag will be un-set (cleared) and the 'registered-user' flag will be set

- If the user is logged in and is also a member of your 'staff-group', the 'staff-user' flag will be set in addition to the 'registered-user' flag

Important: Flags should never be treated as a security mechanism. In the example above, a user could set the staff-user flag from a wiki page. You could override any flags set by users by defining some more defaults at the start of your flag logic, for example:

```
{set-flag:registered-user|state=false}
{set-flag:staff-user|state=false}
```

### Improved performance

There are lots of times when a state will be used several times in a theme layout. Although lots of caching is used in Builder and Confluence, if you're repeatedly checking for a specific state (particularly complex states based on several criteria) it can degrade performance and make your theme config really messy.

One of the most common scenarios we've found so far in themes we've developed for customers is whether or not to show an edit button and other links related to page editing. The general logic runs along these lines:

- Don't show the edit features on top-level pages or the home page
- Only show edit features if the user is logged in (regardless of permissions for anonymous users)
- Only show edit features in the 'Documentation' area of a space if the user is a member of staff

There's obviously quite a few things that need to be checked to confirm whether the edit features should be shown, and in many cases the outcome will be performed in several locations and across multiple theme panels. An ideal use case for flags!

Based on the flags set by earlier examples, the logic to do this would be:

```
{builder-hide:page=<at:var at:name="parent," />home}
  {set-flag:top-level-page}
{builder-hide}

{builder-show:flag=registered-user|notflag=top-level-page}
  {set-flag:show-edit}
{builder-show}

{builder-show:page=Documentation|recurse=true|not-flag=staff-user}
  {set-flag:show-edit|state=false}
{builder-show}
```

You can now toggle the edit links anywhere in your theme based on the 'show-edit' flag. This way you don't need to keep checking where you are in a space (top level pages, home page, documentation section) and the type of user (registered or staff).

Any developer looking at the flag logic above will see several optimisations that can be made - this is key reason for separating out the flag logic in to a wiki page so it's easier to modify...

## Where should flags be defined?

We generally recommend creating a wiki page to store the flag logic for the following reasons:

- Version control - if you make a mistake while editing, you can roll back to an earlier version
- De-cluttering - reduce amount of markup in your theme panels
- Maintainability - separate the flag logic from the theme customisation

Obviously, the page will need to be available to users (otherwise the theme won't be able to import it) so we normally put it in a space that anonymous users can see but not edit.

Once you've made the page, you can use the import macro to import it in to a theme panel and render it in the context of the page the end-user is looking at. Don't use the 'include' macro as that would render it in the context of the flag logic page and not the page being viewed.

We recommend importing the flags in to the Header panel - it's the first panel to be rendered and thus the flags will be available to all panels in the theme layout.

If you can't store flag logic on a wiki page, you could put it in to a user macro (in Confluence administration console) - this will make it available to all users regardless of whether they are logged in.

## CSS customisation with flags

When a flag is set, a class will be added to the HTML <body> tag. Let's take the 'in-the-forum' flag we described earlier as an example:

```
<body class="flag-in-the-forum" ...
```

As you can see, the class is prefixed with 'flag-' so remember that when you're using it in your style sheet.

So, let's say you want to change the colour of the heading 1 style when you're in a forum, you could use this CSS in the CSS Tab:

```
.flag-in-the-forum h1 {
 color: #0f0;
}
```