Add localizable strings to themes and theme components

For those looking to add custom languages and translations to a Discourse theme or theme component, they can now include localised strings, which are made available for use in UI components. Translations are stored in the same format as core/plugin translations, and can be used in almost the same way.

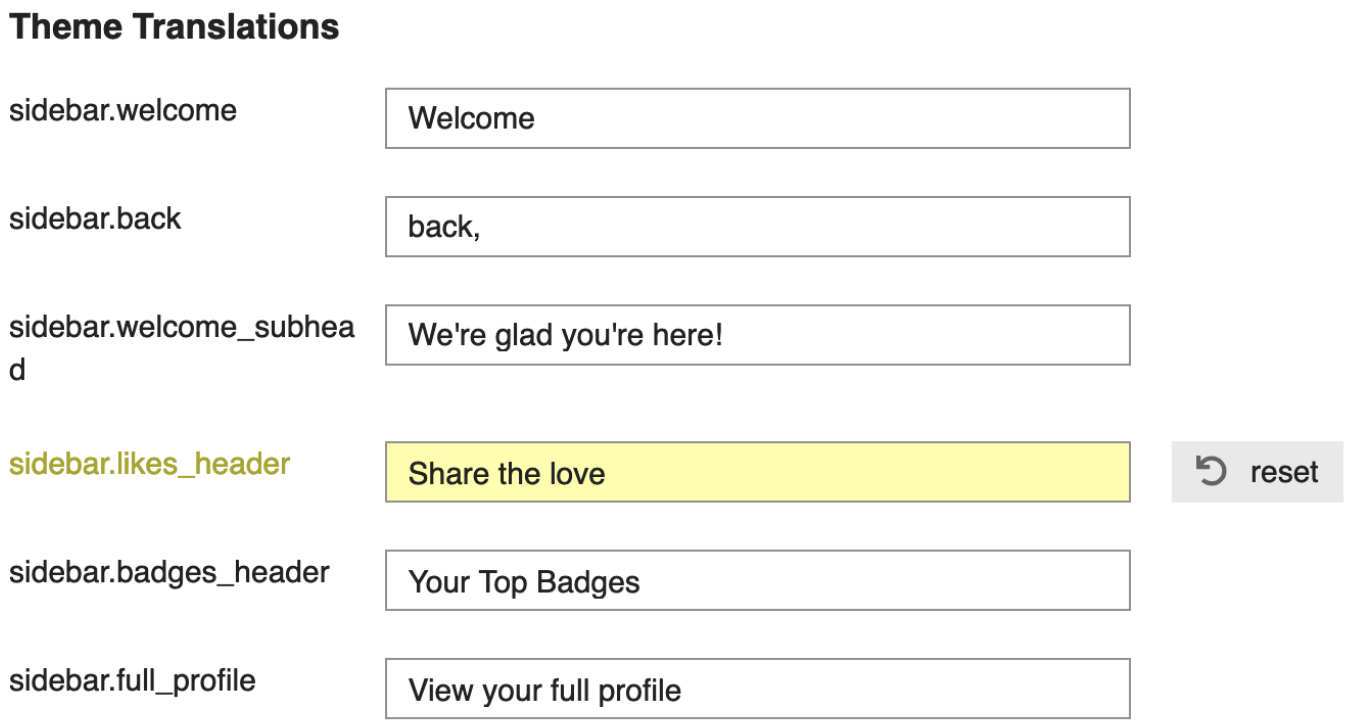
Themes can supply translation files in a format like `/locales/{locale}.yml`. These files should be valid YAML, with a single top level key equal to the locale being defined. These can be defined using the `discourse_theme` CLI, importing a `.tar.gz`, installing from a GIT repository, or via the editor on theme-creator.discourse.org.

An example locale file might look like

```
en:
  theme_metadata:
    description: "This is a description for my theme"
    settings:
      theme_setting_name: "This is a description for the setting `theme_setting_name`"
      another_theme_setting_name:
        description: "This is a description for the setting `another_theme_setting_name`"
  sidebar:
    welcome: "Welcome"
    back: "back,"
    welcome_subhead: "We're glad you're here!"
    likes_header: "Share the Love"
    badges_header: "Your Top Badges"
    full_profile: "View your full profile"
```

Administrators can override individual keys on a per-theme basis in the /admin/customize/themes user interface. Fallback is handled in the same way as core, so it is ok to have incomplete translations for non-english languages will make use of the english keys.

Administrators can also manage these translations in **Appearance → Site texts**. Select a theme or component to search its locale-file strings, edit a translation in the selected language, or revert an override to the locale-file default. The selection is linkable using `/admin/customize/site_texts?theme_id=THEME_ID`. Site texts identifies these entries as `js.theme_translations.THEME_ID.KEY`; edits use the same theme translation overrides as the theme editor.



In the background, these translations are stored alongside the core translations, under a theme-specific namespace. For example:

```
theme_translation.{theme_id}.sidebar.welcome
```

You should never hardcode the theme_id in your theme code. To dynamically build the translation key, use the `themePrefix` helper:

```
import { i18n } from "discourse-i18n";
import { themePrefix } from "virtual:theme";

// In JS code:
const result = i18n(themePrefix("my_translation_key"));
console.log("From Javascript", result);

// In a template tag:
<template>{{i18n (themePrefix "blah")}}</template>
```

For a complete example of using translations in a theme, check out @awesomerobot's Fakebook theme: https://github.com/awesomerobot/Fakebook

---

□□□ #3
□ □□□ □□□ 2026-10-31 13:30:58 UTC
□ cocobook-sync □□□ 2026-11-06 07:04:04 UTC