Internationalization

- What is internationalization?
- Why is internationalization important?
- How to internationalize your theme?
  - Text Domain
    - style.css theme header
      - Domain Path
    - Add text domain to strings
    - Loading Translations
      - Language Packs
- Internationalizing your theme

What is internationalization?

Internationalization is the process of developing your theme, so it can easily be translated into other languages. Internationalization is often abbreviated as `i18n` (because there are 18 letters between the letters i and n).

Why is internationalization important?

WordPress is used all over the world, in countries where English is not the main language. The strings in the WordPress plugins need to be coded in a special way so that can be easily translated into other languages. As a developer, you may not be able to provide localizations for all your users; however, a translator can successfully localize the theme without needing to modify the source code itself.

How to internationalize your theme?

For the text in the theme to be able to be translated easily the text should not be hardcoded in the theme but be passed as an argument through one of the localization functions in WordPress.

The following example could not be translated unless the translator modified the source code which is not very efficient.

```
<h1>Settings Page</h1>
```

By passing the string through a localization function it can it can be easily parsed to be translated.

```
<h1><?php _e( 'Settings Page' ); ?></h1>
```

WordPress uses gettext libraries to be able to add the translations in PHP. In WordPress you should use the WordPress localization functions instead of the native PHP gettext-compliant translation functions.

Text Domain

The text domain is the second argument that is used in the internationalization functions. The text domain is a unique identifier, allowing WordPress to distinguish between all of the loaded translations. The text domain is only needed to be defined for themes and plugins.

Themes that are hosted on WordPress.org the text domain must match the slug of your theme URL (`wordpress.org/themes/<slug>`). This is needed so that the translations from translate.wordpress.org work correctly.

The text domain name must use dashes and not underscores and be lowercase. For example, if the theme's name `My Theme` is defined in the `style.css` or it is contained in a folder called `my-theme` the text domain should be `my-theme`.

The text domain is used in three different places:

1. In the `style.css` theme header
2. As an argument in the localization functions
3. As an argument when loading the translations using `load_theme_textdomain()` or `load_child_theme_textdomain()`

style.css theme header

The text domain is added to the `style.css` header so that the theme meta-data like the description can be translated even when the theme is not enabled. The text domain should be same as the one used when loading the text domain.

Example:

```
/*
* Theme Name: My Theme
* Author: Theme Author
* Text Domain: my-theme
*/
```

Domain Path

The domain path is needed when the translations are saved in a directory other than `languages` . This is so that WordPress knows where to find the translation when the theme is not activated. For example, if .mo files are located in the languages folder then Domain Path will be `/languages` and must be written with the first slash. Defaults to the `languages` folder in the theme.

Example:

```
/*
* Theme Name: My Theme
* Author: Theme Author
* Text Domain: my-theme
* Domain Path: /languages
*/
```

Add text domain to strings

The text domain should be added as an argument to all of the localization functions for the translations to work correctly.

Example 1:

```
__( 'Post' )
```

should become

```
__( 'Post', 'my-theme' )
```

Example 2:

```
_e( 'Post' )
```

should become

```
_e( 'Post', 'my-theme' )
```

Example 3:

```
_n( '%s post', '%s posts', $count )
```

should become

```
_n( '%s post', '%s posts', $count, 'my-theme' )
```

The text domain should be passed as a string to the localization functions instead of a variable. It allows parsing tools to differentiate between text domains. Example of what not to do:\

```
__( 'Translate me.' , $text_domain );
```

\

Loading Translations

The translations in WordPress are saved in `.po` and `.mo` files which need to be loaded. They can be loaded by using the functions `load_theme_textdomain()` or `load_child_theme_textdomain()`.
This loads `{locale}.mo` from your theme's base directory or `{text-domain}-{locale}.mo` from the WordPress theme language folder in `/wp-content/languages/themes/`.

As of version 4.6 WordPress automatically checks the language directory in `wp-content` for translations from translate.wordpress.org. This means that plugins that are translated via translate.wordpress.org do not require `load_plugin_textdomain()` anymore.
If you don't want to add a `load_plugin_textdomain()` call to your plugin you should set the `Requires at least:` field in your readme.txt to 4.6.\

To find out more about the different language and country codes, see the list of languages.

Watch Out

- Name your MO file as `{locale}.mo` (e.g. de_DE.po & de_DE.mo) if adding the translation to the theme folder.

- Name your MO file as `{text-domain}-{locale}.mo` (e.g my-theme-de_DE.po & my-theme-de_DE.mo) if you are adding the translation to the WordPress theme language folder.

Example:

```
function my_theme_load_theme_textdomain() {
    load_theme_textdomain( 'my-theme', get_template_directory() . '/languages' );
}
add_action( 'after_setup_theme', 'my_theme_load_theme_textdomain' );
```

This function should ideally be run within the theme's `function.php`.

Language Packs

If you're interested in language packs and how the import to translate.wordpress.org is working, please read the Meta Handbook page about Translations.

Internationalizing your theme

Now that your translations are loaded, you can start writing every string in your theme with Internationalization functions.

Check the Internationalization page on the Common APIs Handbook for more information and best practices.

□□□ #2
□ □□□ □□□ 2026-10-31 18:36:37 UTC
□ □□□ □□□ 2026-10-31 18:36:38 UTC