# @wordpress/i18n


- Installation
- Usage
- API
  - createI18n
  - defaultI18n
  - getLocaleData
  - hasTranslation
  - isRTL
  - resetLocaleData
  - setLocaleData
  - sprintf
  - subscribe
  - _n
  - _nx
  - _x
  - __
- Contributing to this package


Internationalization utilities for client-side localization. See How to Internationalize Your Plugin for server-side documentation.

# Installation

Install the module:

```
npm install @wordpress/i18n --save
```

*This package assumes that your code will run in an **ES2015+** environment. If you're using an environment that has limited or no support for such language features and APIs, you should include [the polyfill shipped in `@wordpress/babel-preset-default`] in your code.*

# Usage

```
import { sprintf, _n } from '@wordpress/i18n';

sprintf( _n( '%d hat', '%d hats', 4, 'text-domain' ), 4 );
// 4 hats
```

For a complete example, see the Internationalization section of the Block Editor Handbook.

# API

## createI18n

Create an i18n instance

*Parameters*

- *initialData* `[LocaleData< TextDomain >]`: Locale data configuration.
- *initialDomain* `[TextDomain]`: Domain for which configuration applies.
- *hooks* `[Hooks]`: Hooks implementation.

*Returns*

- `I18n< TextDomain >`: I18n instance.

## defaultI18n

Default, singleton instance of `I18n`.

## getLocaleData

Returns locale data by domain in a Jed-formatted JSON object shape.

*Related*

- http://messageformat.github.io/Jed/

*Parameters*

- *domain* `[ string | undefined ]`: Domain for which to get the data.

*Returns*

- `LocaleData`: Locale data.

# hasTranslation

Check if there is a translation for a given string (in singular form).

*Parameters*

- *single* `string`: Singular form of the string to look up.
- *context* `string`: Context information for the translators.
- *domain* `string`: Domain to retrieve the translated text.

*Returns*

- `boolean`: Whether the translation exists or not.

# isRTL

Check if current locale is RTL.

**RTL (Right To Left)** is a locale property indicating that text is written from right to left. For example, the `he` locale (for Hebrew) specifies right-to-left. Arabic (ar) is another common language written RTL. The opposite of RTL, LTR (Left To Right) is used in other languages, including English ( `en`, `en-US`, `en-GB`, etc.), Spanish ( `es` ), and French ( `fr` ).

*Returns*

- `boolean`: Whether locale is RTL.

# resetLocaleData

Resets all current Tannin instance locale data and sets the specified locale data for the domain. Accepts data in a Jed-formatted JSON object shape.

*Related*

- http://messageformat.github.io/Jed/

*Parameters*

- *data* `[LocaleData]`: Locale data configuration.
- *domain* `[string | undefined]`: Domain for which configuration applies.

# setLocaleData

Merges locale data into the Tannin instance by domain. Accepts data in a Jed-formatted JSON object shape.

*Related*

- http://messageformat.github.io/Jed/

*Parameters*

- *data* `[LocaleData ]`: Locale data configuration.
- *domain* `[string | undefined]`: Domain for which configuration applies.

# sprintf

Returns a formatted string.

*Related*

- https://www.npmjs.com/package/@tannin/sprintf

*Parameters*

- *format* `T | TransformedText< T >`: The format of the string to generate.
- *args* `DistributeSprintfArgs< T >`: Arguments to apply to the format.

*Returns*

- `TransformedText< T >`: The formatted string.

# subscribe

Subscribes to changes of locale data

*Parameters*

- *callback* `SubscribeCallback`: Subscription callback

*Returns*

- `UnsubscribeCallback`: Unsubscribe callback

# _n

Translates and retrieves the singular or plural form based on the supplied number.

*Related*

- https://developer.wordpress.org/reference/functions/_n/

*Parameters*

- *single* `Single`: The text to be used if the number is singular.
- *plural* `Plural`: The text to be used if the number is plural.
- *number* `number`: The number to compare against to use either the singular or plural form.
- *domain* `string | undefined`: Domain to retrieve the translated text.

*Returns*

- `TransformedText<Single | Plural>`: The translated singular or plural form.

# _nx

Translates and retrieves the singular or plural form based on the supplied number, with gettext context.

*Related*

- https://developer.wordpress.org/reference/functions/_nx/

*Parameters*

- *single* `Single`: The text to be used if the number is singular.
- *single* `Single`: The text to be used if the number is singular.
- *plural* `Plural`: The text to be used if the number is plural.
- *number* `number`: The number to compare against to use either the singular or plural form.
- *context* `string`: Context information for the translators.
- *domain* `[string | undefined]`: Domain to retrieve the translated text.

*Returns*

- `TransformedText<Single | Plural>`: The translated singular or plural form.

## _x

Retrieve translated string with gettext context.

*Related*

- https://developer.wordpress.org/reference/functions/_x/

*Parameters*

- *text* `Text`: Text to translate.
- *context* `string`: Context information for the translators.
- *domain* `string | undefined`: Domain to retrieve the translated text.

*Returns*

- `TransformedText<Text>`: Translated context string without pipe.

## __

Retrieve the translation of text.

*Related*

- https://developer.wordpress.org/reference/functions/__/

*Parameters*

- *text* `Text`: Text to translate.
- *domain* `string | undefined`: Domain to retrieve the translated text.

*Returns*

- `TransformedText<Text>`: Translated text.

# Contributing to this package

This is an individual package that’s part of the Gutenberg project. The project is organized as a monorepo. It’s made up of multiple self-contained software packages, each with a specific purpose.

The packages in this monorepo are published to npm and used by WordPress as well as other software projects.

To find out more about contributing to this package or Gutenberg as a whole, please read the project’s main contributor guide.

---

□□□ #2
□ □□□ □□□ 2026-10-31 18:19:17 UTC
□ □□□ □□□ 2026-10-31 18:19:18 UTC