# Winmed Translations

| **Version Number** | **Date** | **Author/Owner** | **Description of Change** |
| --- | --- | --- | --- |
| 1 | 2021/08/10 | Romesh Jayawardene | Initial version |
| 2 | 2021/08/11 | Romesh Jayawardene | Changes for NextJS |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Why i18next?**

There are many popular translation libraries, and many will get the job done. With all these topics there’s a lot of personal preference. I18next is a well established and well maintained and supported framework, it supports hooks and HOC, string interpolation and comes with the ability to utilize code splitting and loading the languages at runtime. This can prevent users needing to download all languages which they might not need. It is also possible to split certain areas of the app which could be ‘string heavy’ into their own JSON files and only load them if needed.

# Documentation:

<https://github.com/vinissimus/next-translate>

# Introduction:

We are now using i18next to translate all hard coded strings in the app. The initialisation is configured in the next.config.js file.

From now on when needing to output a string in the app please add it to the json file stored in:

***locales/en/common.json***

with a suitable **camelCase key** and the value as the string which is to be used.

Later on, the relevant CZ and SK translations will be filled in by translators.

# SECTION 1 - FRONTEND CHANGES

To use the string in the app there are one of three ways depending on the type of file using it:

# For Functional components:

Import useTranslation, to allow us to use the useTranslation hook

import useTranslation from 'next-translate/useTranslation'

const { **t** } = **useTranslation**('common')

t(“somethingWentWrong”)

This will now look at the corresponding json file for the language the User has their browser set to, look up the value for the key “somethingWentWrong” and output the corresponding value to correctly translate the string dynamically.

*// -> "Something went wrong"*

# For Files which are not React:

You can simple import ‘t’ as follows:

import getT from 'next-translate/getT'

const t = await getT(locale, 'common')

And use it just the same.

# Special considerations

1. **Put the entire content of the string in a single translatable key-value pair**

It is important **not** to simply concatenate strings like:

{t('hello')} {t('welcome')}

As other languages will structure sentences in different orders and what makes sense in one language might not in another. Instead, the correct usage should be:

{t('hello welcome')}

# Strings which are built up dynamically:

In the json pass in a {{value}} which is a placeholder for the value which will change:



Then in the component we can dynamically supply these values to be interpolated as the second argument to the translate function:



# Case differences in different occurrences of the same string

We need to respect case differences in strings as they appear in different files. **When and only when** a single string occurs in different cases, different translation strings needs to be created for each unique case. E.g. in en.json, for the term “complete”, there can be occurrences of titlecase and lowercase of the same string. In that case, we need to write the en.json as follows:

"products": {

"titleCase": "Products",

"lowerCase": "Products"

},

And in the referencing code, refer to the correct usage as :

t(“products.titleCase”);

This is the concept of **nesting** being used in i18next library.

The reason for this is that in many languages, the way you would write title case, uppercase, lowercase etc would be different. More info can be found here: <https://github.com/meetup/sassquatch2/issues/144#issuecomment-118958029>

# Formatting date/time values

Date/time values need to be presented in the current user’s locale. The react-moment library, which has been used in the application, has support for internationalization of date/time values. For a year/month/date format, we should use “L” or “LL” as the format string, and for year/month/date/hour/minute format we should use “LLL” as the format string.

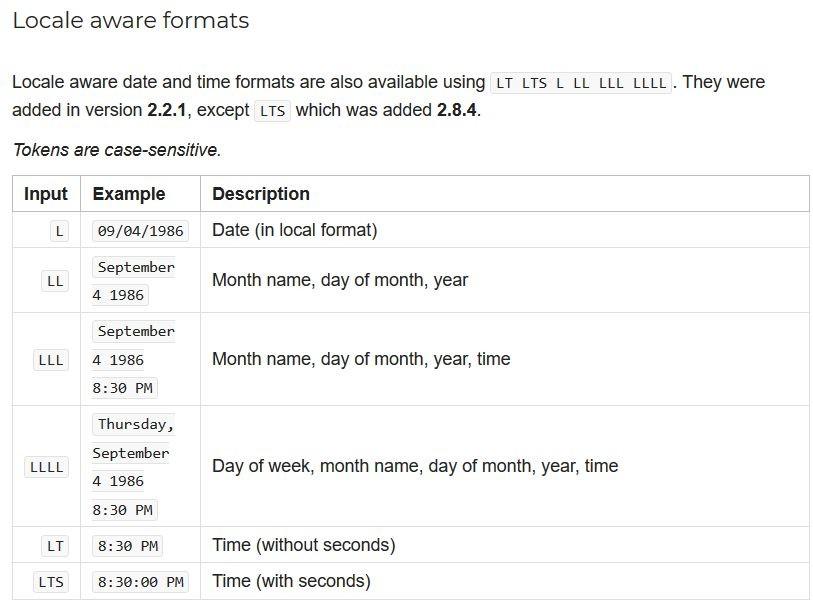
Also, since there can be custom formats in English, we can use the isEnglish() function to check if the currently selected language is English, and present the date/time format accordingly. E.g:

| import { isEnglish } from |  | |
| --- | --- | --- |
| "../../../../translations/translationUtil";  <**Moment** *className*="questions-moment" *format*={**isEnglish**() ? | | |
| "MM/DD/YYYY" : "L"}>{date}</**Moment**>  <**Moment** *format*={**isEnglish**() ? "MMMM D, YYYY" : | | |
| "LL"}>{created\_ts}</**Moment**> | |  |





Following section from the momentjs docs gives more details: [https://momentjs.com/docs/#/parsing/string-format/](https://momentjs.com/docs/%23/parsing/string-format/)



# Custom styling for sub-section of translatable string

There are instances where a word or two within a translatable string needs to be styled.

E.g. made bold or italic. In such an instance, we can use the Trans component and follow the directions in this link: <https://react.i18next.com/latest/trans-component>. IMPORTANT: the drawback of using the Trans component is that **the translation is not automatically refreshed when changing the language on the page.** So this is **only**

**suitable for places like modals,** where access to the language change dropdown is not available on the page. It is not suitable in general.

E.g., within your component, the code can be as follows to bold one word:

| <**Trans**> | | | |
| --- | --- | --- | --- |
| Our |  | | |
| {" "} | |  | |
| <strong>Reasoner™</strong> | | | |
| {" "} | |  | |
| suggests these columns matches | | | |
| </**Trans**> | | |  |
|  | | |

And within the en.json:





Note that the key is not in camelCase, but the entire string has been added as-is, with a placeholder of <1>, </1> to represent the styling tags.