ChatGPT I18N Framework Proposal | 01.24.2024

Prepared by Christopher Youm

Proposal Goals

1. Presenting my I18n technical knowledge to the ChatGPT Growth Team and proposing further industry best practices.
2. Presenting a company-specific I18n framework strategy instead of constructing and integrating the framework.

Personal Note

I am drafting this proposal as part of the technical documentation for my application to the FullStack i18N Engineer, Growth position.

I developed an in-house I18n framework (C#, Java, Objective-C, JS, and PHP) for mobile and Facebook games during my time at Zynga. As one of the founding engineers of the I18n team at Zynga, I actively contributed to the I18n framework and internal translation management tools.

When I first came across the job post for the FullStack i18N Engineer, Growth position, the initial question that crossed my mind was 'why.' Why does ChatGPT aim to develop its own robust I18n framework?

I assume ChatGPT's frontend is constructed with ES6 (I noticed Next.js footprints—webpack, Tailwind, etc., in the web application), and there are already well-established industry-standard (though not perfect) I18n frameworks like i18next.

However, there are instances where having an in-house I18n framework and maintaining it becomes necessary. This could be due to factors such as the I18n framework needing

to support an extensive array of languages, ensuring consistent functionality across different tech specifications for frontend and backend (like React for frontend, Python for backend), or dealing with multiple frontends (such as React for web, Swift for iOS, Java for Android).
I will propose general requirements for the I18n framework, offering an overview of the considerations necessary for cross-product standards.

I18N Functions

Articles

Definite: the apple, the apples
Indefinite: an orange, some oranges

Gender for names

Most Latin based European languages are gender sensitive for the name of users - feminine and masculine.

Singular and Plurals

1 Apple VS 2 apples
The numbers - 1 and 2 should be dynamic, and based on the detected numbers, the following noun - apple, needs to be dynamically handled.
Also, in French, the 0 is singular, but in English, it is a plural. This will require plural rules per language.
More complication comes with Slavic and Arabic, English only has two plural forms - singular and plural, but Russian has three - singular, genitive singular and genitive plural. Arabic has five - singular, duel, plural, sound plural and broken plural.

Grammatical cases

There are four cases in German: nominative (subject), accusative (direct object), dative (indirect object), and genitive (possessive). Determiners and/or adjectives preceding any given noun in a German sentence take

'grammar flags' (a.k.a. strong and weak declensions) that signal to us which case the noun is in.

Contraction(shortened form)
A contraction is when two words are combined to create a new unit. In German, contractions are often created by combining a preposition with a definite article. For example, "in dem" becomes "im".

Possessive
To show possession in German with a name, you can use the genitive case and an "s" without an apostrophe. For example, "Lauras Federtasche" means "Laura's pencil case". In English, we use “‘s” - Jane’s, Kim’s

Formatters

- **Date(default/long/short/menual)**
  Default: 01/25/2024
  Long: Thursday, 25 January 2024 at 14:23:16 GMT+11
  Short: Thu, 25 Jan 2024 at 14:23
  Menual: YYYY-MM-DD -> 2024-01-25
- **Time(duration, relative)**
  Duration: ISO 8601 duration format (HH:MM:SS -> 01:30:25)
  Relative:
  - Lorem in 3 days
  - Lorem 3 hours ago
  - Lorem 3 minutes ago, etc
- **Number(long/short)**
  Long: 1,000
  Short: 1K
- **Currency(long/short)**
  Long: $2,000 -> the currency symbol -”$ needs to be dynamic based on the passed locale.

> Short: $2K -> the currency symbol -”$ needs to be dynamic based on the passed locale, also the number format should be short formatter with abbreviation.

Language detection and fallback

Language detection rules should be the primary aspect of I18n. It is crucial to maintain a consistent user experience across all clients, including web and mobile, as well as all products, including marketing websites. In the event of a system error or missing locale, there should be a fallback (e.g., en-GB -> en-US). Priorities are also set to detect the user's native language (for the web: browser language setting -> cookie -> local storage -> ChatGPT setting).

Pseudo Localization

It is good to have PL test functionality in the framework.

Missing string fallback

For the best practice and optimal user experience, translation fallback is necessary. For example, if ChatGPT is localized into Korean, but some UI texts are not provided by translation agencies and are missing in production, it should automatically fallback to English texts.

Font management(web VS mobile)

Based on the I18n and L10n strategy, there are two options for font management: using a custom font for the local language or using an all-in-one font for all locales. For web products, it's common to use the Arial Unicode font, which includes 51,180 glyphs. However, for a mobile app, there is a possibility to build a custom font from the Arial Unicode font that only contains glyphs found in the entire translation files.

String file management(web VS mobile, .json VS .po VS .string)

Having one translation source file per language across all products and clients (ChatGPT web app and mobile app) will facilitate the entire I18n processes, including localization QA, debugging, requesting translation, and delivering translation.

LTR and RTL handling(Hebrew script, Arabic script)

RTL languages like Hebrew and Arabic always require special attention. If the native API of the web and mobile client programming language does not support RTL correctly, a pre-process or post-process in the translation delivery process may be needed for re-ordering.

, narrow spacing, line break handling(Thai, Korean, Simp/Trad Chinese)

Giving control of string presentation to translators is important, as it improves readability for users. Translators can insert , narrow space, line break tokens in the string, and managing them when the strings are being displayed on the UI programmatically is what the I18n framework needs.

Dynamic font resizing and truncation

Managing overflowing texts in small UI components is also important. Resizing and truncating for two-byte languages (Korean, Japanese, Simplified/Traditional Chinese, Thai) is different from one-byte languages (English or Latin character-based languages).

Alphabetical sorting(Asian, Latin, Slavic, Arabic)

Engineers' lives got easier with the locale specific sorting functions in modern programming languages. ;)

Other Requirements

Company standard
I18n framework, once built and shipped on at least one product, needs to become a company-wide standard software. All developers and leads should be aware that there is an I18n framework that they need to integrate, not only for ChatGPT but across the entire company.

Continuous education
After integrating the I18n framework, all developers need to begin writing human-readable UI strings in a specific format. They also need to understand how to externalize content and request translations from the I18n team. This can only be achieved through continuous education.

Translation management system
If there is an I18n team, then I assume there are already discussions about TM (not Translation Memory, but Translation Management system). This is a middle layer between translation vendors and ChatGPT production, which will involve a collection of pre/post processes, delivery, and various scheduling automations. At the beginning of I18n, it may not be necessary, but once there are three or more products that need to be internationalized, then it will become a significant part of the I18n team. This is another substantial aspect, and I hope to have a chance to discuss this further.