Article

Supporting multiple languages in your app

Internationalize your app's strings, images, and other resource types to prepare for the translation process.

Overview

Multilingual apps are apps that can run in more than one language and region. Making your app multilingual doesn't only enable your app to run natively in more regions of the world, it gives your customers a better overall experience, while bringing your app to a wider audience.

To make your app multilingual, you first need to *internationalize* it. This process involves preparing assets in your app so that a localizer can translate them into different languages and regional conventions. For example, dates in some countries appear in a day-month-year format, while in others, the month comes first.

After you internationalize your app, you need to *localize* it. Localization is the process of translating your assets into other languages for various regions. During this process, you export the relevant strings and other resources from your app, and give them to a localizer for translation. The localizer gives you back translated versions of your assets, which you import into your app. You then test the translations in your app to make sure everything works.

Internationalize your code

The first step to making your app multilingual is to internationalize your code to handle different languages and regional conventions.

This process involves writing your code in such a way that your app can automatically extract the language resources it needs based on the current language settings of your user's device. The Foundation framework, along with other Apple frameworks, supports this internationalization process.

When writing code for internationalization, consider the following:

- **User-facing text**. People are more comfortable using apps when the text appears in the language and region of their device. Use localized versions of the string formatters to prepare your app's text for localization. For more information about user-facing text, see Discover String Catalogs.
- **Dates, currencies, and numbers**. Different regions have different formats for dates, currencies, and numbers. Use `DateFormatter` and `NumberFormatter` in the Foundation framework to translate these strings correctly.
- **Pluralization**. Languages have different grammatical rules for handling plurals of nouns and units. Use a string catalog to localize formatted strings that contain variable amounts. For more information about pluralization, see Localizing and varying text with a string catalog.
- **Device type**. The device your app runs on affects the text it displays. The display on an Apple Watch differs from that on a Mac. Vary the text you present depending upon the device your app is running on. For more information about localizing text for a device type, see Localizing and varying text with a string catalog.
- **Grammatical agreement**. Many languages rely on gender for their grammar. Without knowing the subject's gender or pronoun preferences, some localized strings may have grammatical errors, resulting in a poor user experience. Use the automatic grammar agreement APIs in Foundation, such as `TermOfAddress`, to represent grammatical gender in localized text. For more information about grammatical agreement, see Unlock the power of grammatical agreement.
- **Text direction**. European languages read left to right, and languages like Arabic and Hebrew read right to left. Use the layout tools in SwiftUI and Xcode to control text and UI element orientation, and to flip image direction when necessary. For more information about text direction, see Get it right (to left).
- **Tall languages**. Languages like Arabic, Hindi, and Thai require significantly more vertical space for their characters than Latin languages do. Additionally, Chinese, German, Japanese, and Korean have language-specific conventions for wrapping and hyphenation. To prevent clipping of words and letters, and to ensure proper spacing of text, use Dynamic Type. For more

information about Dynamic Type, see Scaling fonts automatically and What's new with text and text interactions.

- **Sounds, images and assets**. App assets like sounds, images, and colors can vary across language and region. Use an asset catalog to localize colors, images, and sounds in your app. For more information about adding resources to asset catalogs, see Adding resources to localizations and Localizing assets in a catalog.

Localize your assets

After you internationalize your app, it's ready for localization. To localize your assets, export localizable text from Xcode using standard file formats and submit them to a localization team for translation into your app's supported languages.

To localize your app, do the following:

- Export your app assets.
- Translate those assets into other languages and regional conventions.
- Import the translated assets into your project.

For more information about exporting and importing app assets, see Discover String Catalogs.

Test your translations

After you localize your app's assets, you need to test the translations. Run your app in each language, and for each region you support, to thoroughly test the localized assets. For more information about testing, see Testing localizations when running your app.

See Also

Essentials

Localizing and varying text with a string catalog

Use a string catalog to translate text, handle plurals, and vary the text your app displays on specific devices.

Using generated localizable symbols in your code

Add keys directly to your string catalog that you can reference in your code using Xcode generated localizable symbols.

{} Localizing Landmarks

Add localizations to the Landmarks sample code project.