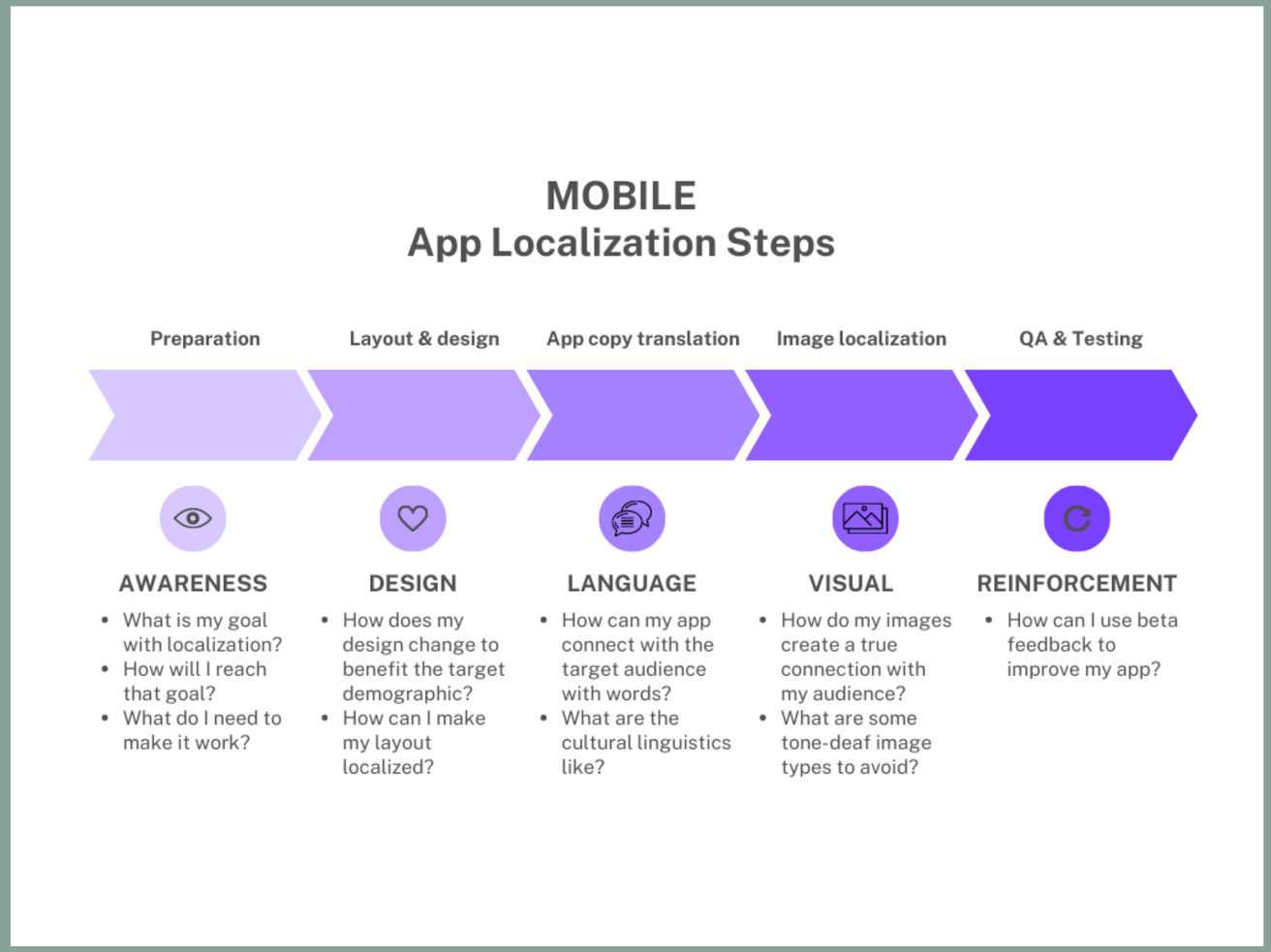
MDI 115 – 1

# Localizing Your SwiftUI iPadOS Apps

Gabriela Sánchez Alcaraz

# Introduction to Localization

## Why Localization Matters in Development



# Introduction to Localization

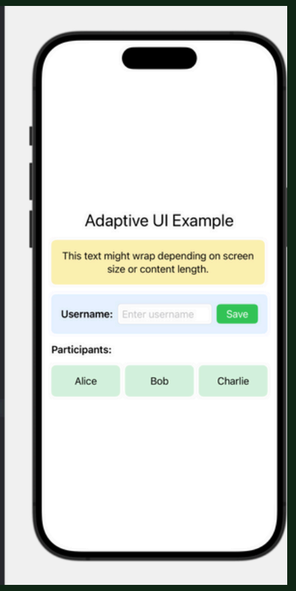
## Why Localization Matters in Development

- The App Store is a World Market: Your app can be downloaded in over 175 countries.
- Enhanced User Experience: Users prefer apps in their native language. It feels more intuitive, trustworthy, and professional.
- Increased Downloads & Revenue: Localizing your app can significantly expand your user base and boost engagement.
- Competitive Advantage: A well-localized app stands out from the competition

# Adaptive Layouts in SwiftUI

**SwiftUI is built for flexibility. Avoid fixed sizes!**

```swift
import SwiftUI

struct ContentView: View {
    @State private var username: String = ""
    @State private var participants: [String] = ["Alice", "Bob", "Charlie"]

    var body: some View {
        GeometryReader { geometry in
            ScrollView {
                VStack(alignment: .leading, spacing: 16) {
                    Text("Adaptive UI Example")
                        .font(.title)
                        .frame(maxWidth: .infinity, alignment: .center)

                    Text("This text might wrap depending on screen size or content length.")
                        .lineLimit(nil)
                        .multilineTextAlignment(.center)
                        .padding()
                        .background(Color.yellow.opacity(0.3))
                        .cornerRadius(8)

                    HStack {
                        Text("Username:")
                            .bold()
                        TextField("Enter username", text: $username)
                            .textFieldStyle(RoundedBorderTextFieldStyle())

                        Button(action: {
                            let trimmed = username.trimmingCharacters(in: .whitespacesAndNewlines)
                            if !trimmed.isEmpty {
                                participants.append(trimmed)
```



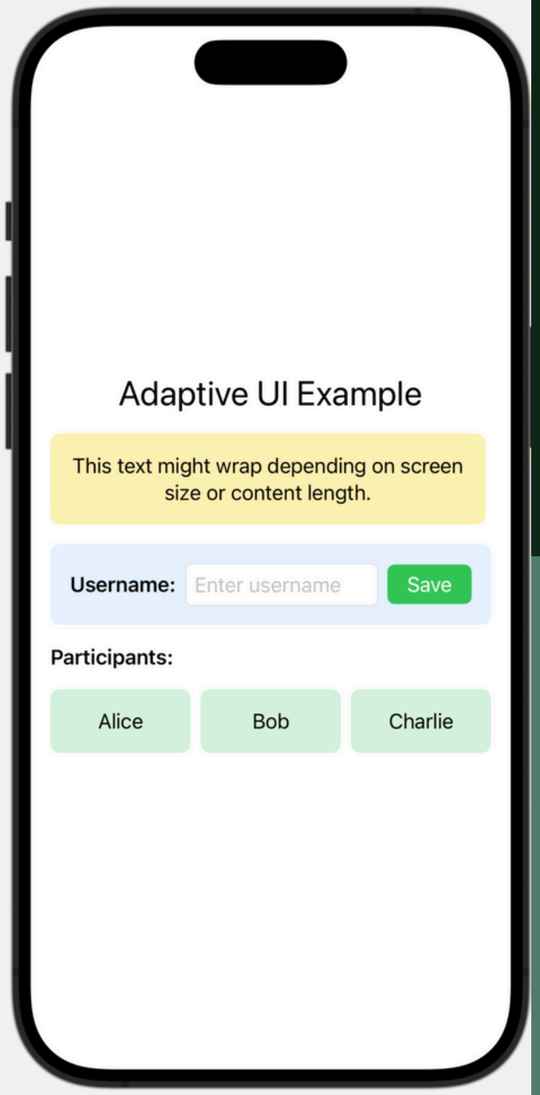

## Stacks

(HStack, VStack, ZStack): These are naturally adaptive.

## Grids

(Grid, LazyVGrid): Great for arranging content that can reflow.

```swift
import SwiftUI

struct ContentView: View {
    @State private var username: String = ""
    @State private var participants: [String] = ["Alice", "Bob", "Charlie"]

    var body: some View {
        GeometryReader { geometry in
            ScrollView {
                VStack(alignment: .leading, spacing: 16) {
                    Text("Adaptive UI Example")
                        .font(.title)
                        .frame(maxWidth: .infinity, alignment: .center)

                    Text("This text might wrap depending on screen size or content length.")
                        .lineLimit(nil)
                        .multilineTextAlignment(.center)
                        .padding()
                        .background(Color.yellow.opacity(0.3))
                        .cornerRadius(8)

                    HStack {
                        Text("Username:")
                            .bold()
                        TextField("Enter username", text: $username)
                            .textFieldStyle(RoundedBorderTextFieldStyle())

                        Button(action: {
                            let trimmed = username.trimmingCharacters(in: .whitespacesAndNewlines)
                            if !trimmed.isEmpty {
                                participants.append(trimmed)
```



# Key Terminology

## Understanding Localization Terms

### Internationalization (I18n)

- **The Foundation.** Designing your app so it can be adapted to various languages and regions without engineering changes.
- Think: Flexible architecture.

Examples:

- Using adaptive UI layouts.
- Not hardcoding text.
- Handling dates, numbers, and currencies correctly.

### Localization (L10n)

- **The Adaptation.** The process of actually translating and adapting your app for a specific language and region (a "locale").
- Think: Specific content.

Examples:

- Translating text strings to Spanish.
- Displaying dates as DD/MM/YYYY for the UK.
- Changing colors or images to be culturally appropriate.

# EXAMPLE

# EXAMPLE

**Eight hilarious localization fails in advertising**

Going beyond translation, localization can make or break marketing campaigns. The process of adapting a product or content to a specific country or target market, localization requires a deep understanding an...

Gengo / Oct 23, 2019

# Internationalization: Building a "World-Ready" App

## Understanding the process in app development

## What is Internationalization?

Internationalization (i18n) involves designing applications to support multiple languages and regions seamlessly, ensuring developers don't need to redesign the app for different locales or cultural contexts.

---

Never Hardcode User-Facing Text:
**Text("Hello, World!") vs. Text(LocalizedStringKey("welcome_message"))**

---

Design Flexible Layouts:
**Text length varies dramatically between languages. "Save" in English is "Speichern" in German.**

---

Externalize Resources:
**Keep strings, images, and other assets separate from your executable code.**

# Understanding Localization

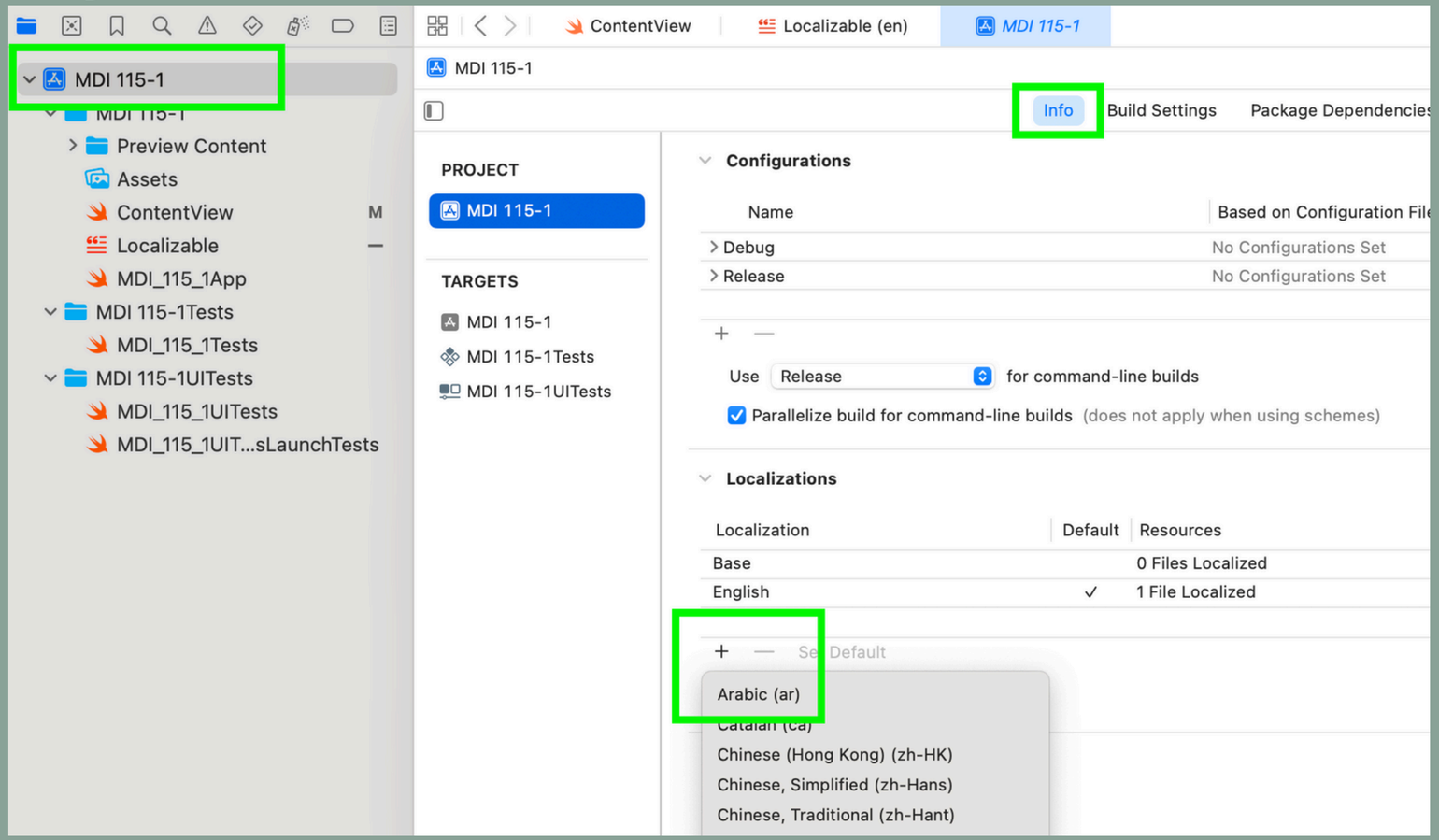
## The foundation of app internationalization
## Defining Localization

**Localization** is the process of adapting app content to meet the specific language, region, and cultural expectations of users, ensuring a more relatable and engaging experience for diverse audiences.

- Localization Catalog (.xcstrings): The modern, recommended way to manage all your localizable content.
  - A single, structured file for all languages.
  - Shows translation status (New, Stale, Translated).
  - Allows you to add comments for translators.

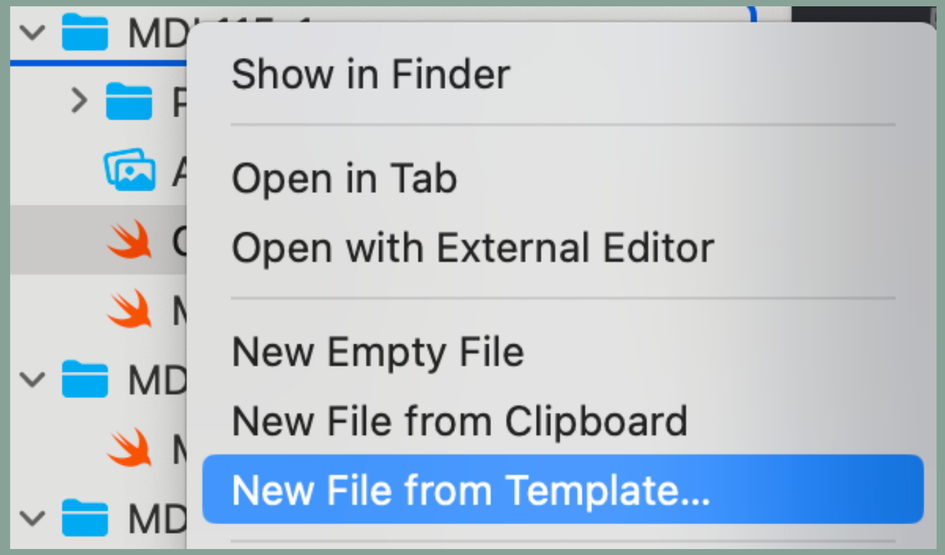
# Settip up your languages with Regions

## Project

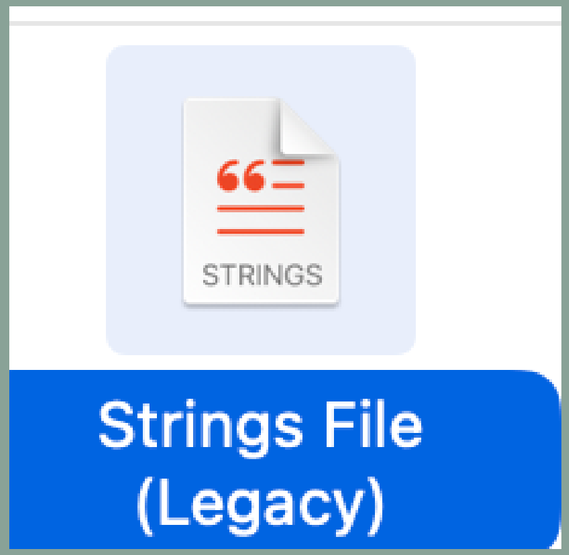



1. Select your main project.
2. Click on Info and select your project Settings
3. Add a new Localization (language + region)

# Strings File



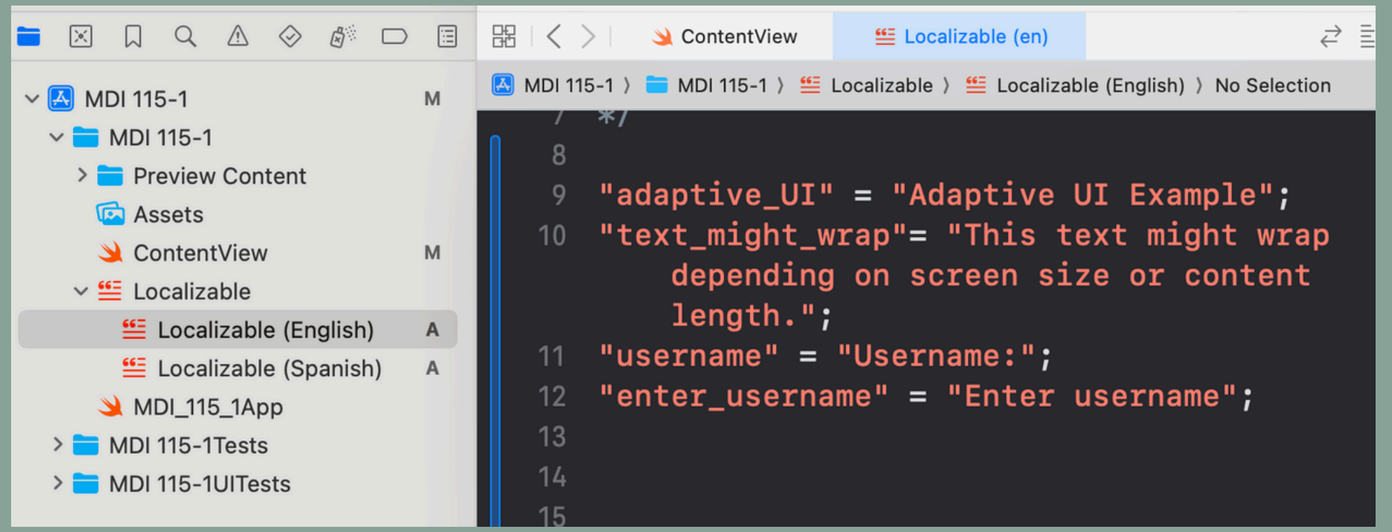

**Step 1**
**Add a new file**



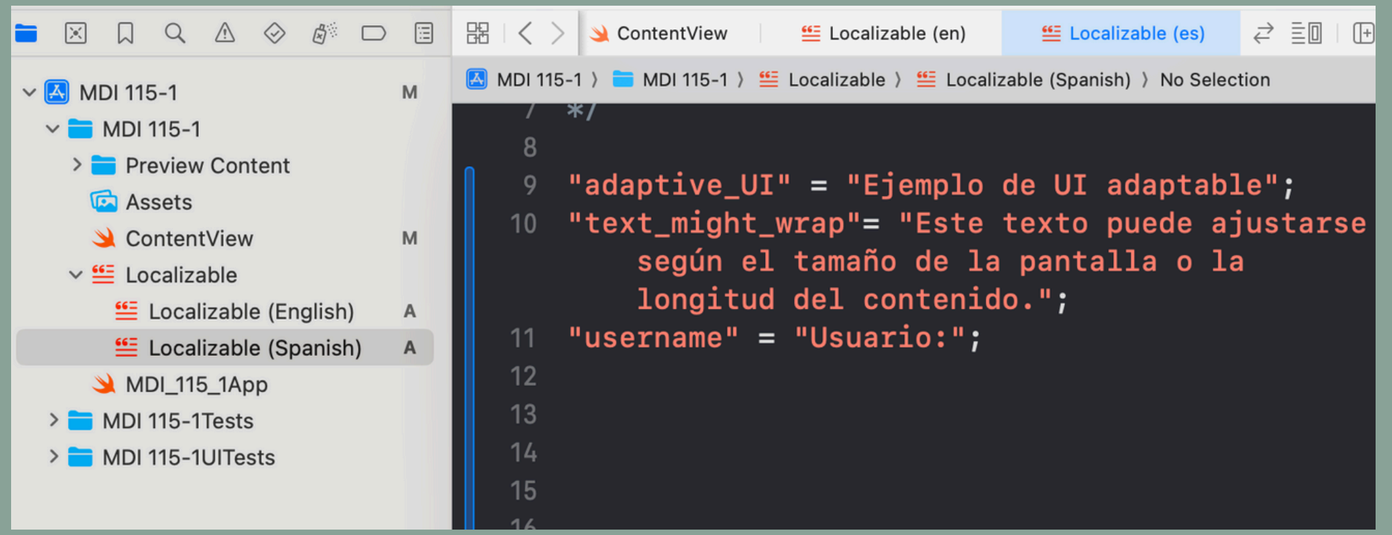

**Step 2**
**Search for Strings File**

# Strings File

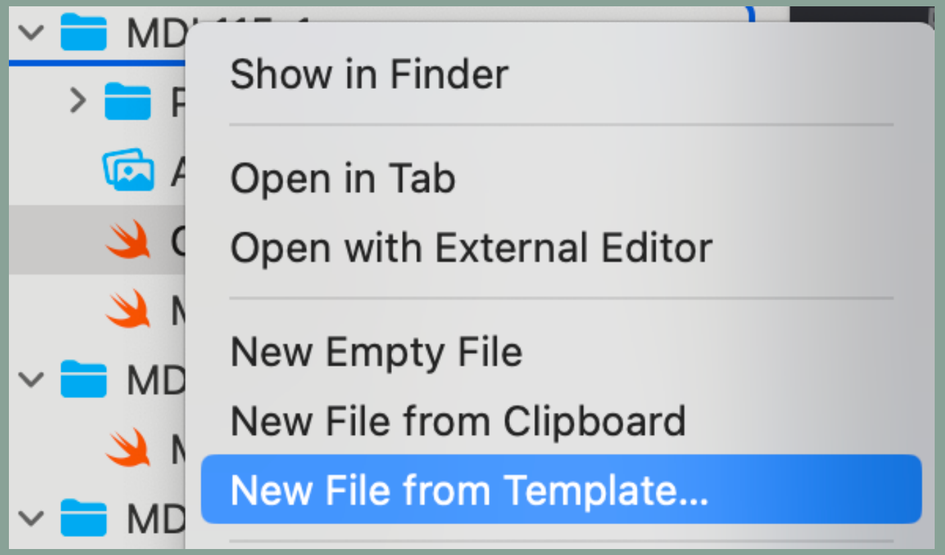
**Step 3:**
**Change all your english strings to the localizable file (English)**




**Step 4:**
**Add the translation into another localizable file (Spanish)**

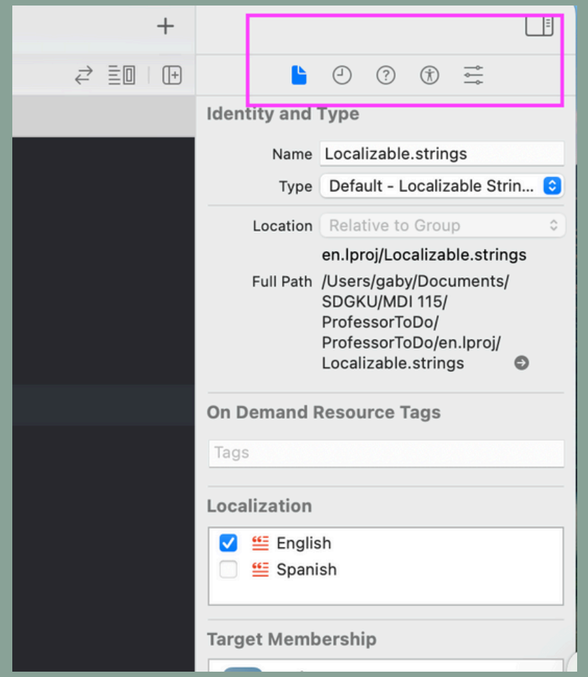


# Catalog File (new)




**Step 1:**
**Add a new File**

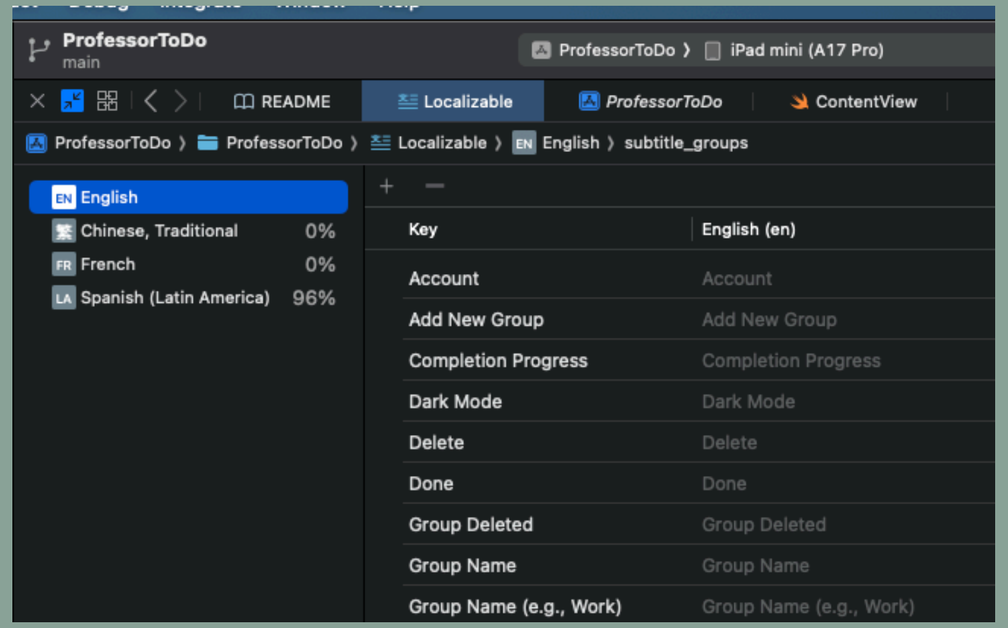

**Step 2**
**Search for String Catalog**



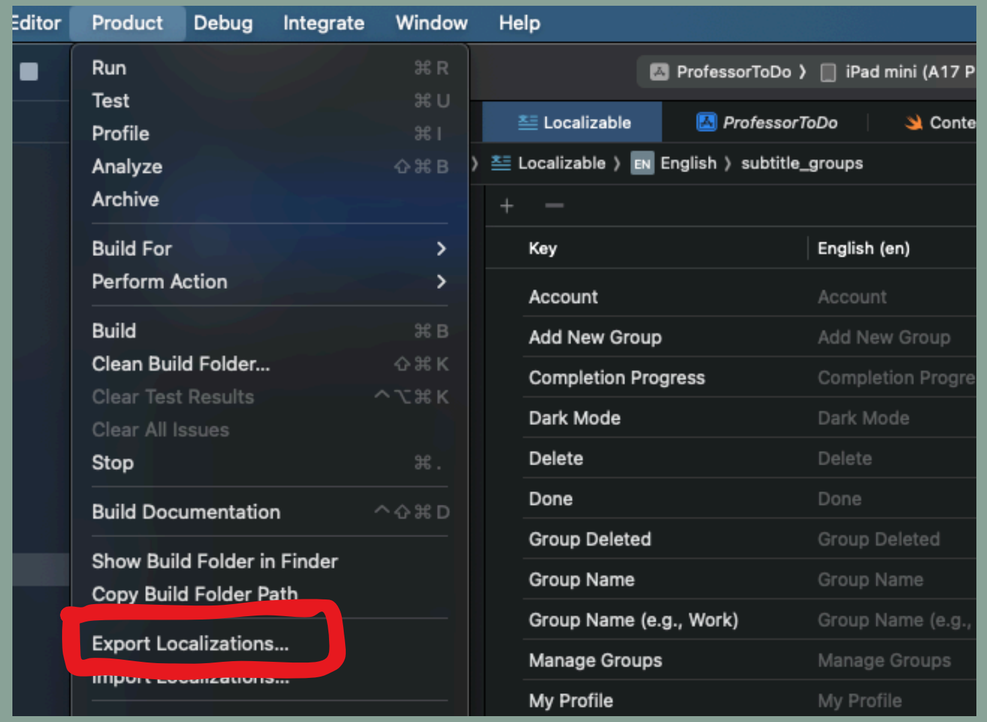

**Step 2.1 (extra)**
**If you don't see your new language check that it is being added to your project target on the top-right corner of your project.**

# String Catalog

**Step 3:**
**Build you app (Command+b)**
**It should automatically show all your text**



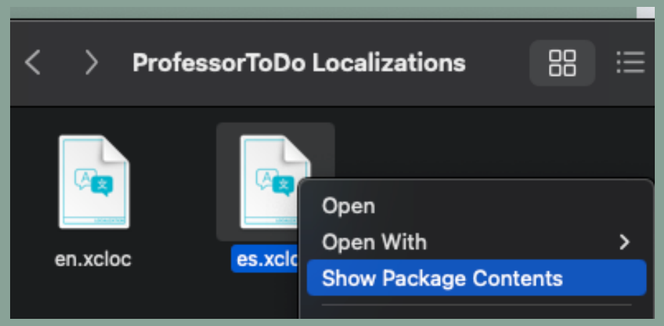

**Step 4:**
**Export you files to be translated**
**Product > Export Localization**

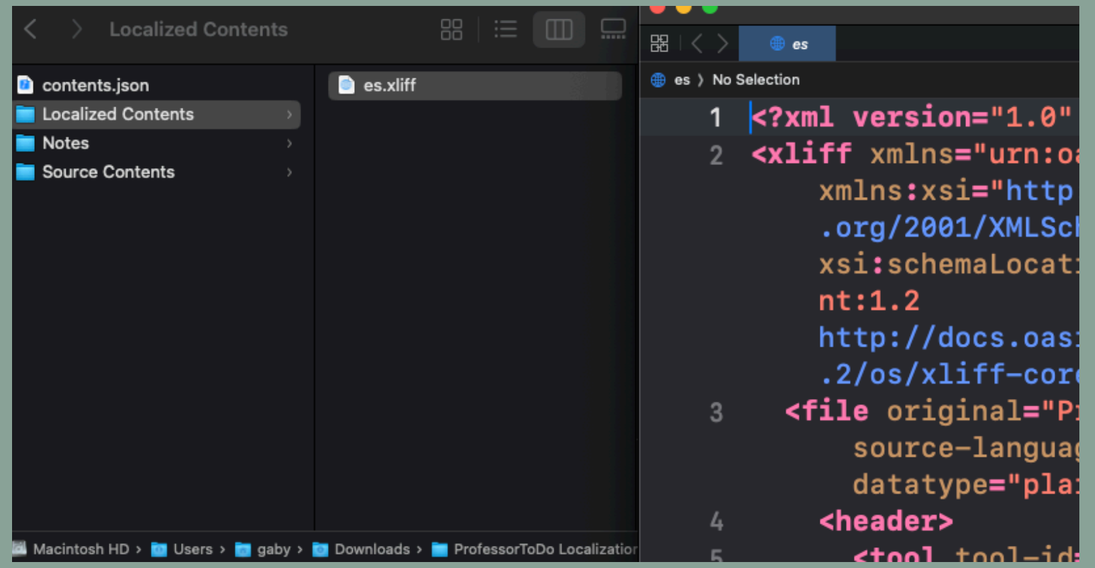


# String Catalog

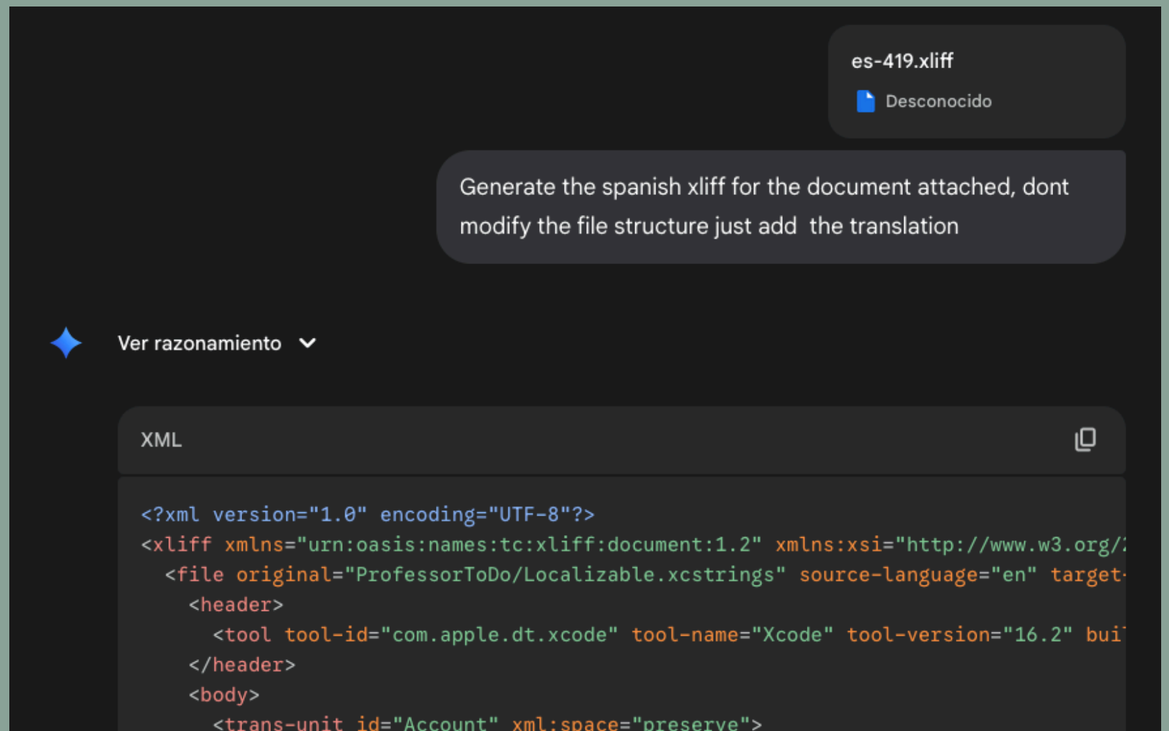
## Step 5:

1. Open the folder downloaded.
2. Right click on the language to be translated, select “Show Package Content” and open the .xliff file



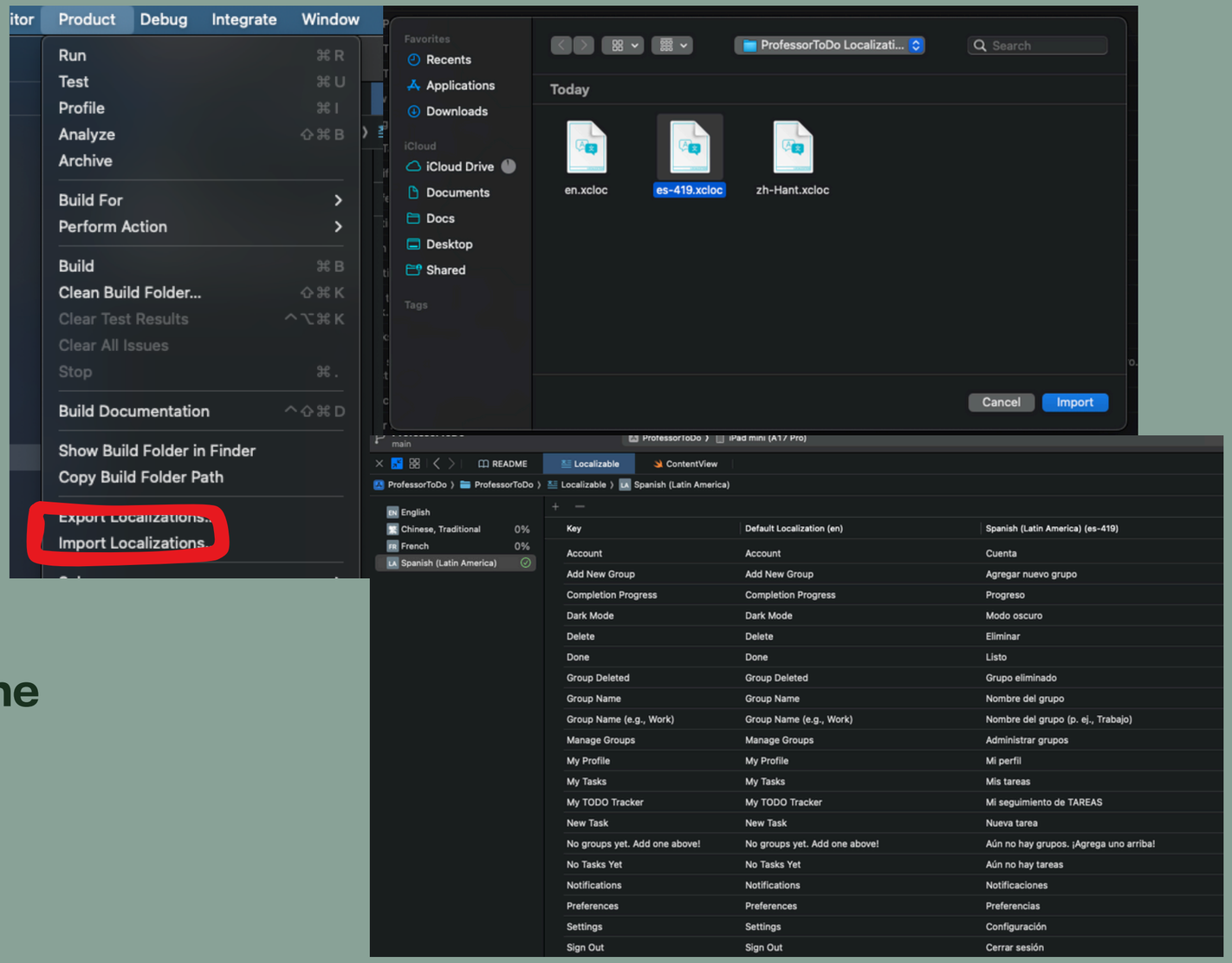




## Step 6:
## Go to an AI Tool and get help translating your keywords



# String Catalog

**Step 7:**
**Paste your new content inside the .xliff file, save it and go back to XCode**



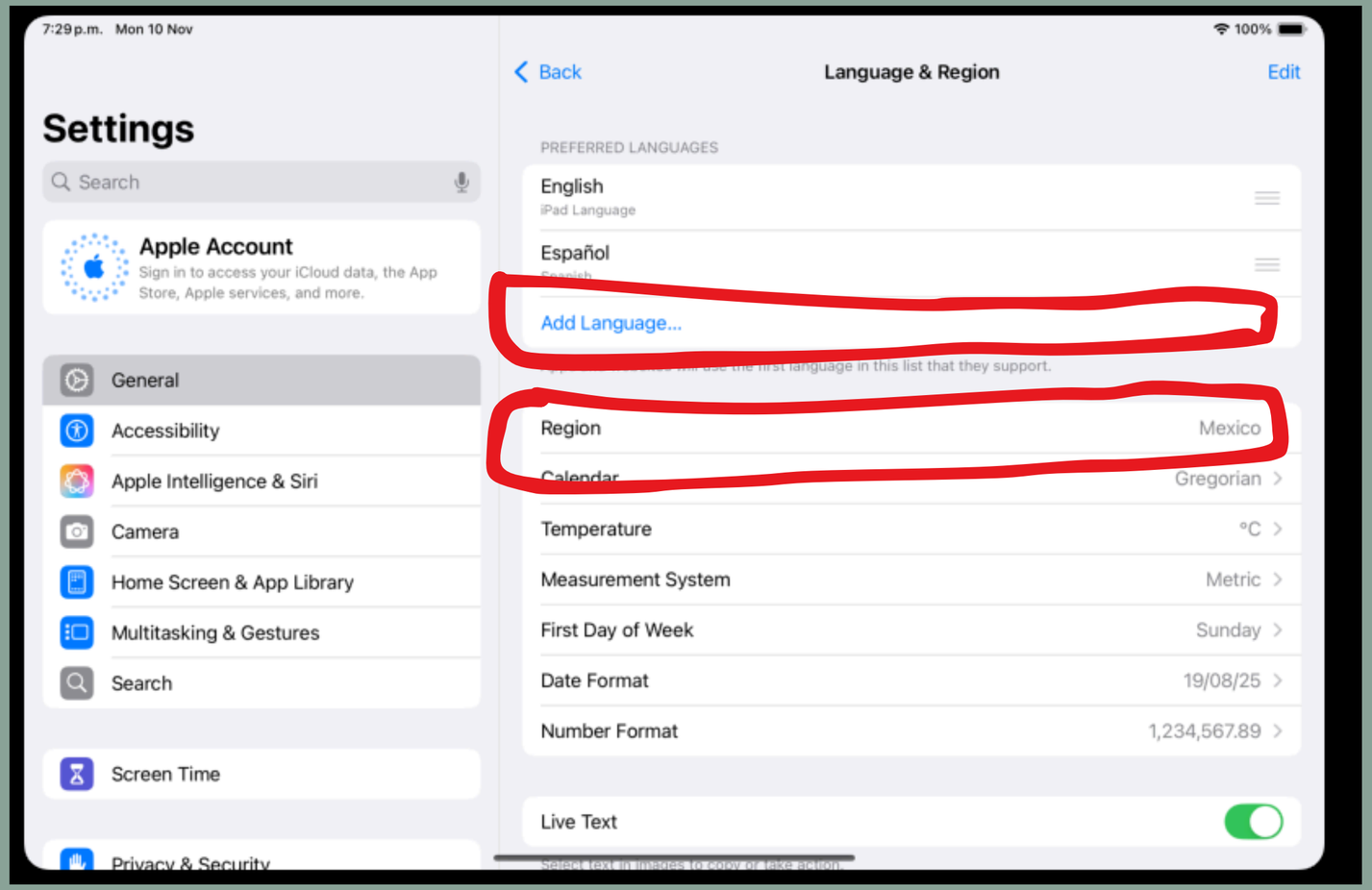

**Step 8:**
**Product > Import Localization > Select the language of type .xloc**

**See the translation on the String Catalog**

# String Catalog

**Step 9:**
**Run your App, on the simulator:**
**Settings > Language and Redgion**

**Change both the Region and Language to see the correct change**



```
var body: some View {
    VStack(alignment: .leading) {
        // Full date and time, localized
        Text(now, style: .date)
        // "October 17, 2025" in en-US
        // "17 de octubre de 2025" in es-ES

        Text(now, style: .time)
        // "1:53 PM" in en-US
        // "13:53" in es-ES

        // Or combine them!
        Text("Event Time: \(now.formatted(date: .long, time: .short
```

# Date and Number Formatting

## Never format data manually! A user's locale determines the correct format.

Proper date and number formatting is essential for localization. SwiftUI provides tools to ensure that apps display **contextually appropriate** information according to user preferences and regional settings.

### Let the System Do the Work:

- **Locale: Represents a user's language, region, and cultural conventions.**
- **DateFormatter: The classic way to configure specific date/time styles.**
- **SwiftUI's .formatted() API: The modern and preferred way.**

# UI/UX Considerations

## Designing for Diverse Global Audiences

When developing for a global audience, several **key design considerations** must be addressed:

- Accommodate text expansion in different languages
- Implement flexible layouts that adjust to varying text lengths
- Support bidirectional text for right-to-left languages
- Use Locale-aware formatters for date, time, currency, and number formats
- Ensure cultural customization of colors, icons, and images to resonate with local users
- Prioritize accessibility in localized contexts to meet diverse user needs

Adaptive SwiftUI layouts should utilize flexible elements such as **GeometryReader** to create responsive interfaces that adjust seamlessly. Implementing **dynamic type support** allows text to scale according to user preferences, enhancing readability. By considering these factors during the design phase, developers can create inclusive and user-friendly experiences for a wide range of audiences, ultimately leading to greater engagement and satisfaction.

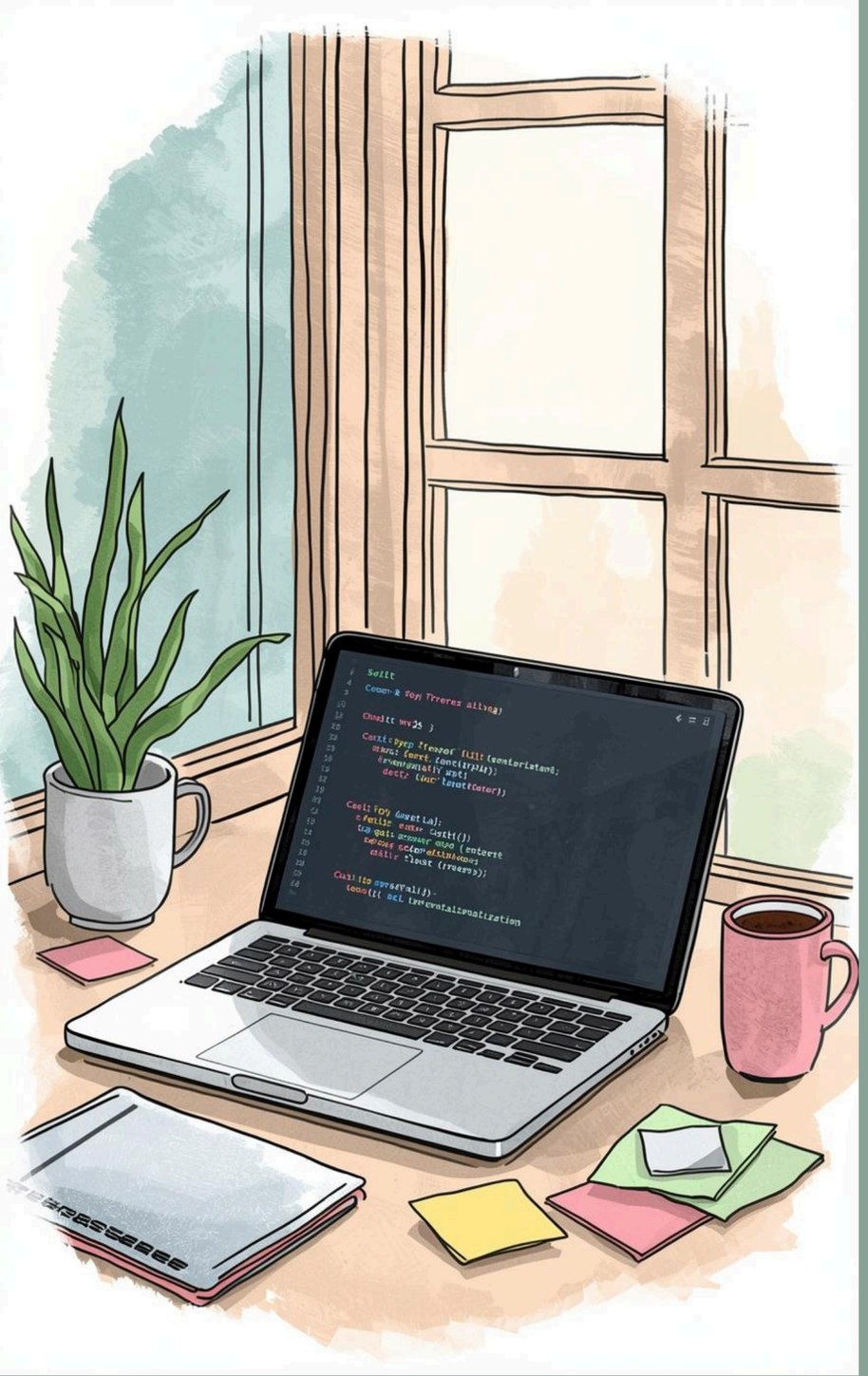
# Culturally relevant icons matter

**Culturally Relevant Icons Matter**

- Icons are not a universal language. A symbol's meaning can change drastically between cultures.
- Using locally recognized symbols improves usability and avoids misunderstandings.
- Goal: Ensure your visual language is as clear and respectful as your written language.

## Examples to Consider:

- Thumbs Up 👍: Positive gesture in the US, but offensive in parts of the Middle East and West Africa.
- Mailbox 📮: A US-style mailbox is unfamiliar in Japan or the UK. An envelope icon (✉️) is often a safer international choice.
- Owl 🦉: A symbol of wisdom in Western cultures, but considered a bad omen in some parts of Asia.



# Assignment

## Goal: Create a simple iPad app that demonstrates your understanding of localization and internationalization.

### Requirements:

1. **Localized Text:**
   - The app must contain at least two different languages to be translated (e.g. English & Spanish, English & Japannes
2. **Locale-Aware Date:**
   - Display the current date and/or time on your ToDo App, can be on the profile or whenever a new task is added to show **timeStamp**
   - The format must automatically update to match the selected language (e.g., October 18, 2025 for English-US, 18 de octubre de 2025 for Spanish-MX).
3. **Language Switcher:**
   - Switch Between languages from the phone settings
   - The entire UI (text and date format) must update immediately when the language is changed.