Scripting with SwiftUI

```
#!/usr/bin/swift
```

What is SwiftUI?

- User interface language for Apple platforms
  - Declarative syntax
- User interface toolkit
  - Xcode design tools

Very Brief History

- Swift was introduced in WWDC 2014
- SwiftUI was introduced in WWDC 2019
- Current version: “SwiftUI 3”
- WWDC 2022?

SwiftUI Basics

- Views

  `Text, Button, Image, …`

- Modifiers

  `.font(.title), .foregroundColor(.blue), …`

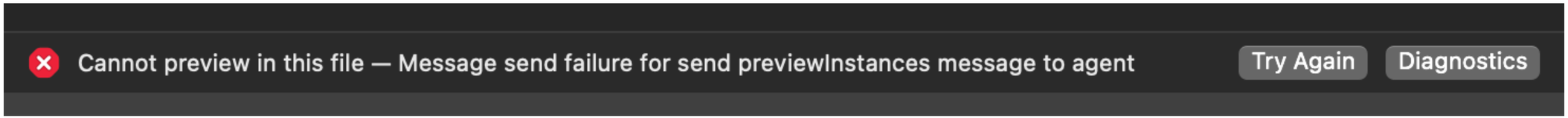
- Property wrappers

  `@State, @Binding, @ObservedObject, …`

Xcode SwiftUI demo

- New SwiftUI project
- Different ways of building the UI
  - Live preview
  - Inspector
  - Library

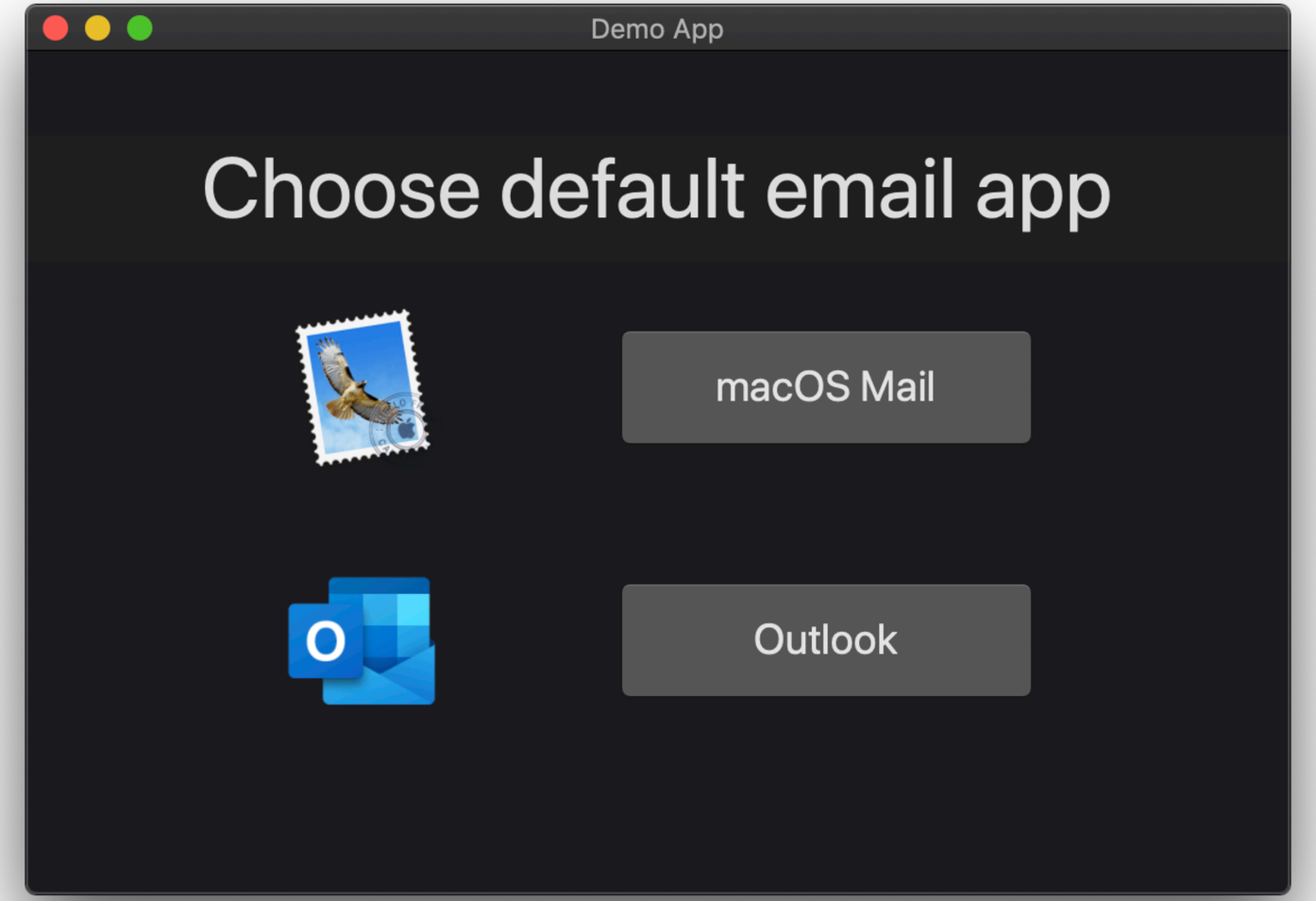
Possible live preview issues



FIX: Disable **Automatically Refresh Canvas** feature

Xcode > Editor > Canvas > Automatically Refresh Canvas

Scripting in Swift 2019 demo app



Scripting in Swift 2019 demo app

- Choose default email application
- Built using AppKit
- Static window size
- Fixed UI component positioning

Scripting with SwiftUI 2022 demo app

- Choose default email application
- Built using SwiftUI
- Resizable window
- Adaptive UI component positioning
- Refactored business logic

Requirements for SwiftUI scripts

- Xcode 13 and Xcode Command Line Tools installed
- https://github.com/jlehikoinen/ScriptingWithSwiftUIDemo
  - AppDelegate code block (see EmptyWindow.swift)

Demo GUI scripts

- EmptyWindow.swift
  - No UI components, no Launch Services functionality
- DefaultMailAppOnlyUI.swift
  - Only UI components, no Launch Services functionality
- DefaultMailApp.swift
  - UI + Launch Services functionality for changing default email app

Code overview

- AppProperties model
- Enumerations
- LaunchServicesHelper
- AppDelegate

Code overview

- SwiftUI code
  - ContentView
  - @State property wrapper
  - Launch Services helper methods
  - Custom ButtonStyles
  - App icon highlight animation

Refactoring

- “Real” SwiftUI app
  - Usually has separate view models
  - Views should contain only UI related code
  - UI code is refactored into smaller pieces, e.g. separate row view
    - Makes view related code cleaner and more structured

Script vs. Xcode project

- All code in one file vs. code structured in groups and several files
- Xcode pros:
  - Autocompletion
  - Live error & warning messages
  - Live UI preview

Script vs. Xcode project

- Script vs. notarized app bundle
  - With script you can bypass:
    - Code Signing
    - Notarization
    - App Sandbox
    - Hardened Runtime
- Script might be viable alternative if deploying an app bundle is “impossible”
  - Command Line Tools for Xcode (~700 MB) & script vs. compiled app

Scripting with SwiftUI

=

Developing with Xcode + delivering the SwiftUI code in one file

Links

- Apple SwiftUI links:
  - https://developer.apple.com/tutorials/swiftui/
  - https://developer.apple.com/videos/play/wwdc2021/10062/
  - https://developer.apple.com/documentation/swiftui/building_a_great_mac_app_with_swiftui
- Other SwiftUI links:
  - https://www.woodys-findings.com/posts/scripting-swiftui-1-coding-color-picker-view
  - https://scriptingosx.com/swiftagain/
  - https://github.com/atrinh0/wwdc22