Safari Extension Project Structure

Safari Web Extension with Xcode wrapper, shared JavaScript code, and native Swift container app.

#safari #extension #browser #swift #xcode #macos #ios

PNG PDF Copy Prompt

Project Directory

```
MyExtension/
  MyExtension.xcodeproj/ Xcode project
    project.pbxproj
  MyExtension/ Container app (…
    MyExtensionApp.swift App entry point
    ContentView.swift SwiftUI view
    Assets.xcassets/ App icons
    Info.plist
  MyExtension Extension/ Extension code
    Resources/ Web extension f…
      manifest.json Extension manif…
      background.js Background scri…
      content.js Content script
      content.css
      popup/
        popup.html
        popup.js
        popup.css
      images/
        icon-48.png
        icon-96.png
        icon-128.png
        toolbar-icon.pdf Vector
      _locales/
        en/
          messages.json
    SafariWebExtensionHandler.swift Native messaging
    Info.plist
  Shared (Extension)/ Shared resources
    ToolbarItemIcon.pdf
  README.md
  .gitignore
```

Why This Structure?

Safari extensions require an Xcode project with a native Swift container app. The actual extension code lives in `Resources/` using standard WebExtension APIs. This structure is generated by Xcode's Safari Extension template but organized for maintainability.

Key Directories

`MyExtension Extension/Resources/` - Web extension code (JS, HTML, CSS)

`manifest.json` - Standard WebExtension manifest

`SafariWebExtensionHandler.swift` - Bridge for native messaging

`MyExtension/` - Container app required by App Store

Safari Manifest

```
// manifest.json
{
  "manifest_version": 2,
  "name": "My Extension",
  "version": "1.0",
  "browser_specific_settings": {
    "safari": { "strict_min_version": "15.4" }
  },
  "background": { "scripts": ["background.js"], "persisten
  "content_scripts": [{
    "matches": ["<all_urls>"],
    "js": ["content.js"]
  }],
  "browser_action": { "default_popup": "popup/popup.html"
}
```

Getting Started

1. Open Xcode and create new Safari Web Extension project
2. Select macOS and/or iOS targets
3. Add extension code in `Resources/` folder
4. Build and run to install in Safari
5. Enable in Safari > Preferences > Extensions

When To Use This

- Building extensions specifically for Safari
- Need native macOS/iOS capabilities
- Publishing to Mac/iOS App Store
- Porting Chrome/Firefox extension to Safari
- Corporate Safari-only deployment

Safari Platform Notes

`Manifest V2` - Safari still uses V2 (V3 support coming)

`App Store required` - Must distribute via Mac/iOS App Store

`Native messaging` - Swift handler for native functionality

`Universal binary` - Single build for Intel and Apple Silicon

Best Practices

- Use `browser.*` APIs with polyfill for cross-browser compat
- Test on both macOS and iOS Safari
- Container app should explain extension functionality
- Request minimal permissions for App Store approval
- Use PDF icons for crisp rendering at all sizes

Trade-offs

`Xcode required` - Must use macOS and Xcode for development

`App Store review` - Subject to Apple review process and guidelines

`Manifest V2 only` - No service workers, use event pages