





Info.plist and the Android manifest: five things you may have missed

Thomas Künneth

2 / 50

 Kotlin Multiplatform Wizard

English   

New Project



Templates Gallery

Project Name



KotlinProject


Project ID


org.example.project



 Android 



With Compose Multiplatform UI framework based on Jetpack Compose



 iOS 


 Share UI via Compose Multiplatform UI framework

 Do not share UI - Use SwiftUI

 Desktop 

 Web 

 Server 

 Include tests

↓ DOWNLOAD

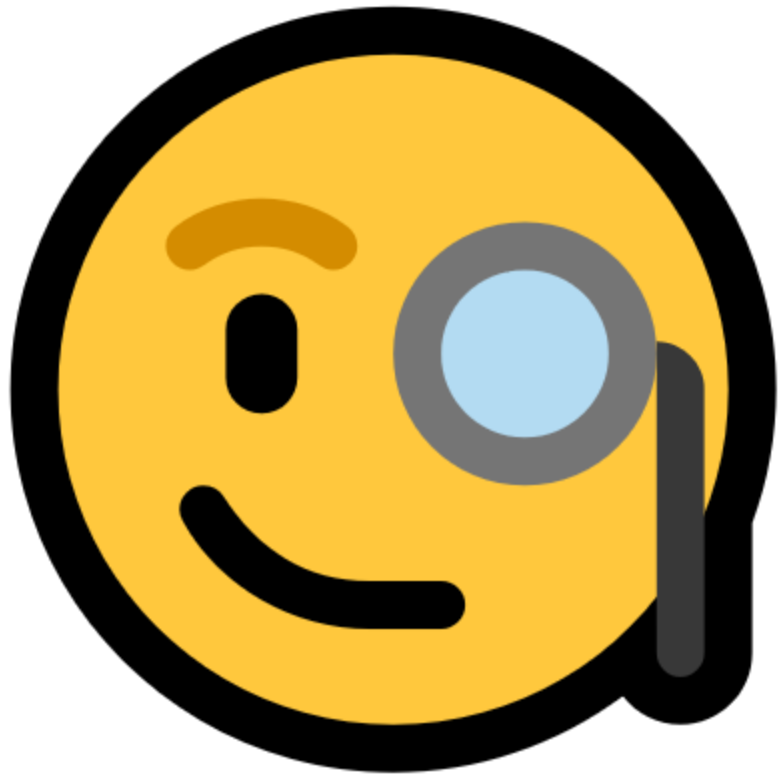
- The *Android manifest* declares the **core components** of an app (Activities, Services, Broadcast Receivers, and Content Providers) along with all required **permissions** and **features**
- The *iOS Info.plist* contains the **unique bundle ID**, **name**, and **version**, supported features, and **privacy-sensitive permission requests** to iOS
- Different file format, different content, but still similar purpose
- Vital on both platforms

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@android:style/Theme.Material.Light.NoActionBar">

        <activity
            android:exported="true"
            android:launchMode="singleTask"
            android:name="com.example.android.MainActivity"
            android:screenOrientation="portrait"
            android:windowSoftInputMode="adjustResize">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```



Who has made significant changes to any of the two files in their CMP/KMP projects?

Who believes their files not only meet the requirements but also contain everything that should be in?

Enhances visual performance, but may also lead to increased power consumption and higher CPU/GPU usage, potentially affecting battery life and device thermals



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4 <dict>
5   <key>CADisableMinimumFrameDurationOnPhone</key>
6   <true/>
7 </dict>
8 </plist>
```

Allows the app to render at higher frame rates than the standard 60 frames per second (fps) on supported devices

<https://github.com/JetBrains/compose-multiplatform/issues/3634>

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>LSApplicationCategoryType</key>
  <string>public.app-category.utilities</string>
  <key>CADisableMinimumFrameDurationOnPhone</key>
  <true/>
</dict>
</plist>

```

Category that best describes your app for the App Store

Information Property List	+	Dictionary	↕	(2 items)
App Category	↕	String	↕	Utilities
CADisableMinimumFrameDurationOnPhone	↕	Boolean	↕	YES

`android:appCategory`

Categories are used to cluster multiple apps together into meaningful groups, such as when summarizing battery, network, or disk usage

Only define this value for apps that fit well into one of the specific categories

1

Make informed decisions by understanding what a value means and why it is set

Decide if your app benefits from it being present

Not being present initially does not mean it's not important



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android">
3
4      <application
5          android:allowBackup="true"
6          android:icon="@mipmap/ic_launcher"
7          android:label="@string/app_name"
8          android:roundIcon="@mipmap/ic_launcher_round"
9          android:supportsRtl="true"
10         android:theme=
11             "@android:style/Theme.Material.Light.NoActionBar"
12     >
13         <activity
14             android:exported="true"
15             android:configChanges="orientation|screenSize|
16                                     screenLayout|keyboardHidden|
17                                     mnc|colorMode|density|fontScale|
18                                     fontWeightAdjustment|keyboard|
19                                     layoutDirection|locale|
20                                     mcc|navigation|
21                                     smallestScreenSize|touchscreen|uiMode"
22             android:name=".MainActivity">
23             <intent-filter>
24                 <action android:name="android.intent.action.MAIN" />
25                 <category android:name="android.intent.category.LAUNCHER" />
26             </intent-filter>
27         </activity>
28     </application>
29
30 </manifest>
```

Lists configuration changes that the activity handles itself. When a configuration change occurs at runtime, the activity shuts down and restarts by default, but declaring a configuration with this attribute prevents the activity from restarting. Instead, the activity remains running and its `onConfigurationChanged()` method is called.

- Current version of KMP wizard no longer sets `android:configChanges`
- Responsibility of the project to decide what to do
- Which configuration changes should your app handle by itself?
See talk by Alex Vanyo: Handling configuration changes in Compose:
<https://www.droidcon.com/2025/07/23/handling-configuration-changes-in-compose/>



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/r
3
4      <application
5          android:allowBackup="true"
6          android:icon="@mipmap/ic_launcher"
7          android:label="@string/app_name"
8          android:roundIcon="@mipmap/ic_launcher_round"
9          android:supportsRtl="true"
10         android:theme="@android:style/Theme.Material.Light
11         <activity
12             android:exported="true"
13             android:name=".MainActivity">
14                 <intent-filter>
15                     <action android:name="android.intent.acti
16
17                     <category android:name="android.intent.ca
18                 </intent-filter>
19             </activity>
20         </application>
21
22 </manifest>
```

2

Handling configuration changes on Android require careful consideration

Only you know what your app needs

It's not up to the KMP wizard to provide a complete or bullet-proof manifest / Info.plist – just a start



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android">
3
4      <application
5          android:allowBackup="true"
6          android:icon="@mipmap/ic_launcher"
7          android:label="@string/app_name"
8          android:roundIcon="@mipmap/ic_launcher_round"
9          android:supportsRtl="true"
10         android:theme="@android:style/Theme.Material.Light.NoActionBar">
11         <activity
12             android:exported="true"
13             android:name=".MainActivity">
14             <intent-filter>
15                 <action android:name="android.intent.action.MAIN" />
16
17                 <category android:name="android.intent.category.LAUNCHER" />
18             </intent-filter>
19         </activity>
20     </application>
21
22 </manifest>
```

Sets a light theme
as the initial theme
for your app

- Jetpack Compose is great at handling themes including light and dark mode
- Resource-based themes are still relevant for
 - system-provided splash screens
 - embedded `AndroidViews`
- Without a change, the splashscreen background will appear in light mode
- Many solutions, a simple one is including Jetpack Appcompat and setting `android:theme="@style/Theme.AppCompat.DayNight.NoActionBar`

- On iOS, you can set the app's default light or dark mode behavior using `UIUserInterfaceStyle` (Appearance in Xcode)
- Possible values are `Light`, `Dark`, `Automatic` (default when not set)

3

Jetpack Compose makes a lot of things way easier (configuration changes, themes / styles)

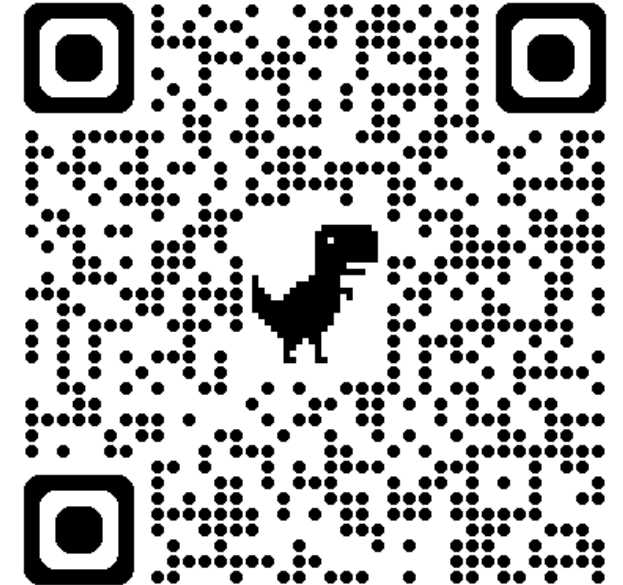
Still, your app may have not immediately obvious ties to the old world (splash screens, embedded `AndroidViews`)

Make sure to reflect them properly in your manifest file

- Similar to Android, certain actions like accessing files, contacts, calendars, or the camera, can be granted or denied by the user on iOS
- Like on Android, requesting a permission is done during runtime
- A description describing why the permission is requested, must be present in Info.plist



```
1 package dev.tkuenneth.infoplistmanifestdemo.infoplistmanifestdemo
2
3 enum class CameraPermission { Unknown, Granted, Denied }
4
5 expect fun checkCameraPermission(): CameraPermission
6
7 expect fun requestCameraPermission(callback: (CameraPermission) -> Unit)
```



<https://github.com/tkuenneth/InfoPlistManifestDemo>

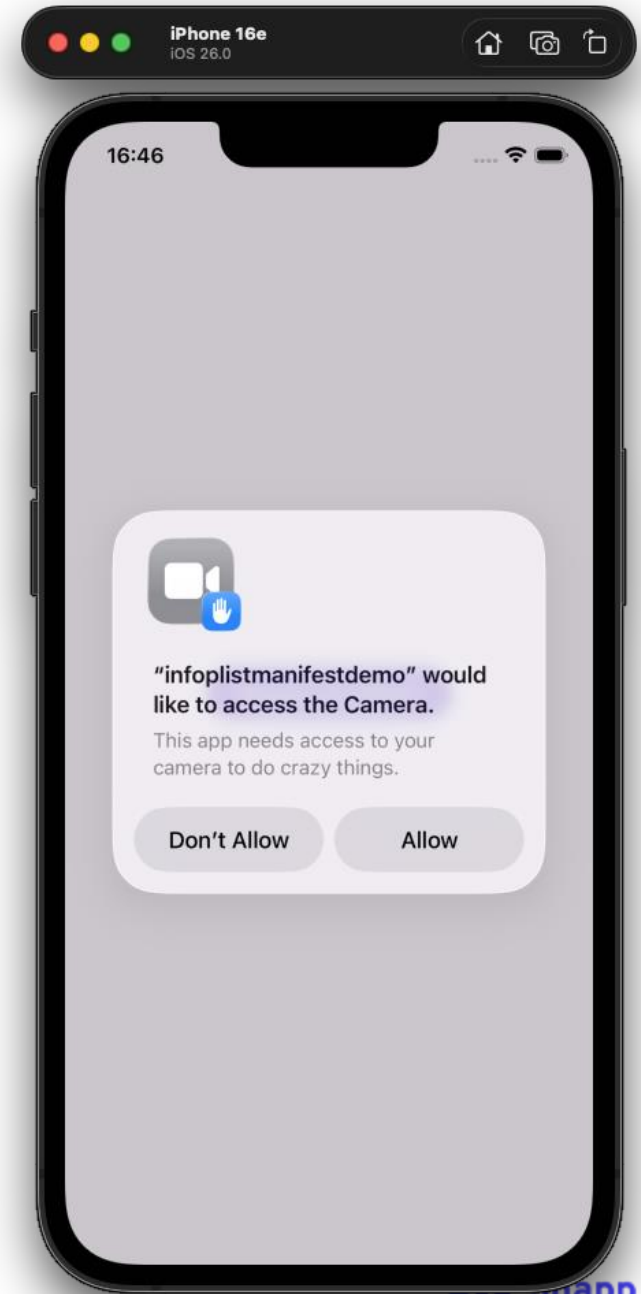


```
1 actual fun checkCameraPermission(): CameraPermission {
2     return when (AVCaptureDevice.authorizationStatusForMediaType(AVMediaTypeVideo)) {
3         AVAuthorizationStatusAuthorized -> CameraPermission.Granted
4         AVAuthorizationStatusRestricted, AVAuthorizationStatusDenied -> CameraPermission.Denied
5         else -> CameraPermission.Unknown
6     }
7 }
8
9 actual fun requestCameraPermission(callback: (CameraPermission) -> Unit) {
10     AVCaptureDevice.requestAccessForMediaType(AVMediaTypeVideo) { granted ->
11         callback(if (granted) CameraPermission.Granted else CameraPermission.Denied)
12     }
13 }
```



```
1  @Composable
2  fun App() {
3      MaterialTheme {
4          Box(
5              modifier = Modifier.background(MaterialTheme.colorScheme.surface).fillMaxSize(),
6              contentAlignment = Alignment.Center,
7          ) {
8              var cameraPermission by remember { mutableStateOf(checkCameraPermission()) }
9              Button(onClick = {
10                  when (cameraPermission) {
11                      CameraPermission.Unknown -> requestCameraPermission {
12                          cameraPermission = it
13                      }
14
15                      else -> {}
16                  }
17              }) {
18                  Text(
19                      stringResource(
20                          when (cameraPermission) {
21                              CameraPermission.Unknown -> Res.string.request_permission
22                              CameraPermission.Granted -> Res.string.take_picture
23                              CameraPermission.Denied -> Res.string.open_settings
24                          }
25                      )
26                  )
27              }
28          }
29      }
30 }
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
<plist version="1.0">
<dict>
  <key>LSApplicationCategoryType</key>
  <string>public.app-category.utilities</string>
  <key>AVCam</key>
  <string></string>
  <key>CADisableMinimumFrameDurationOnPhone</key>
  <true/>
  <key>NSCameraUsageDescription</key>
  <string>This app needs access to your camera to do crazy things.</string>
</dict>
</plist>
```

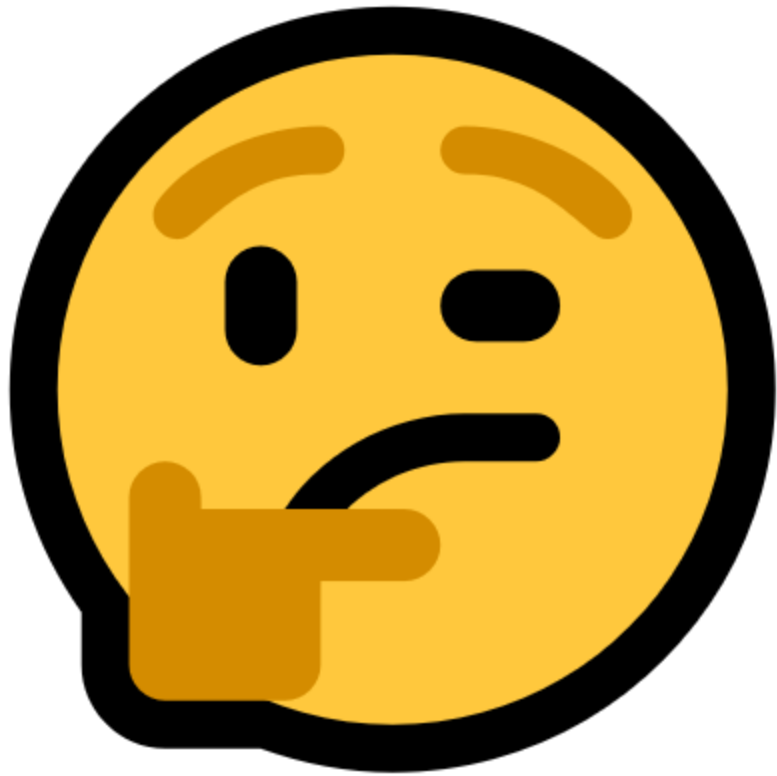


4

Info.plist contains messages for permission dialogs

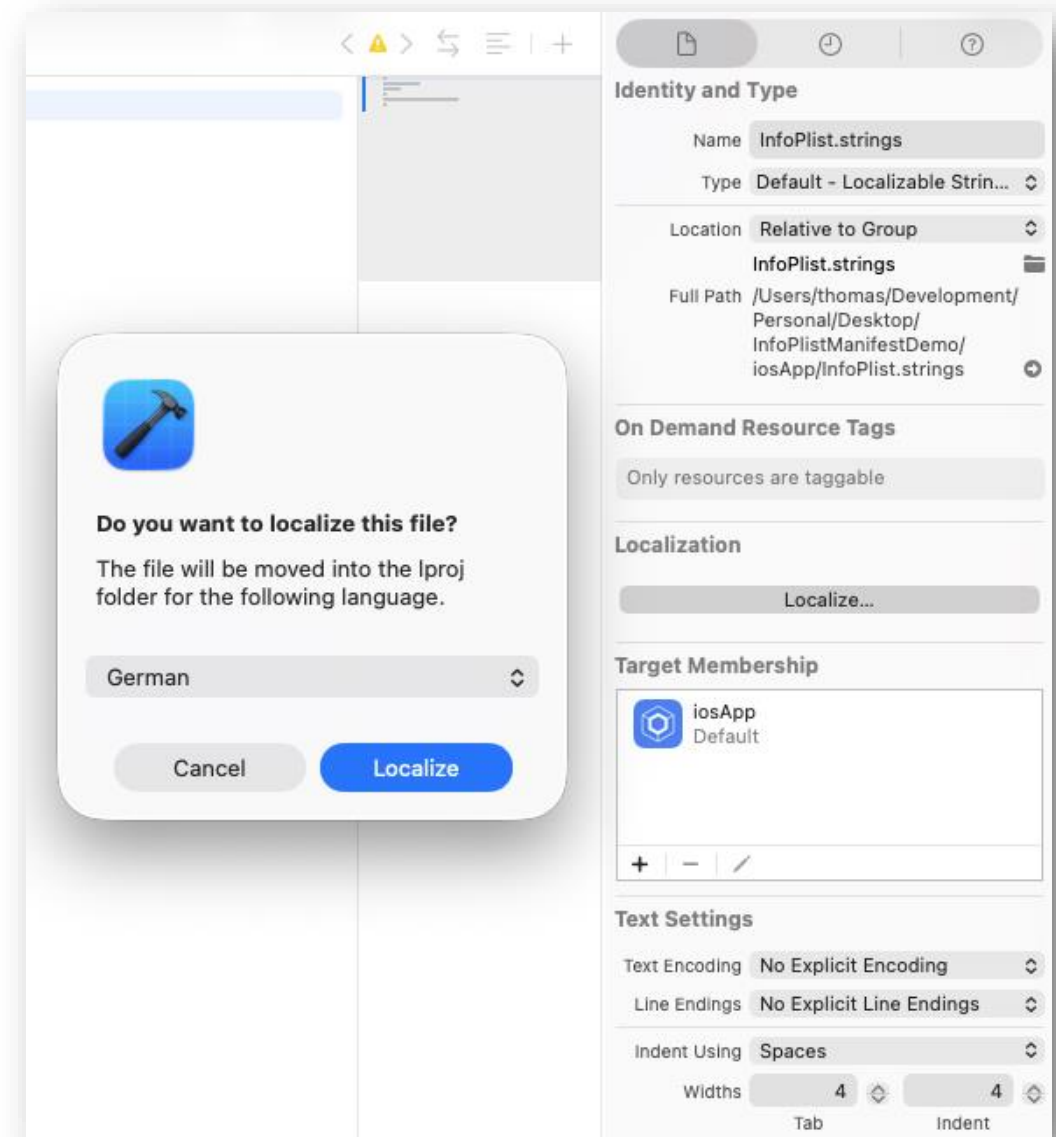
They must be present; otherwise the app will crash

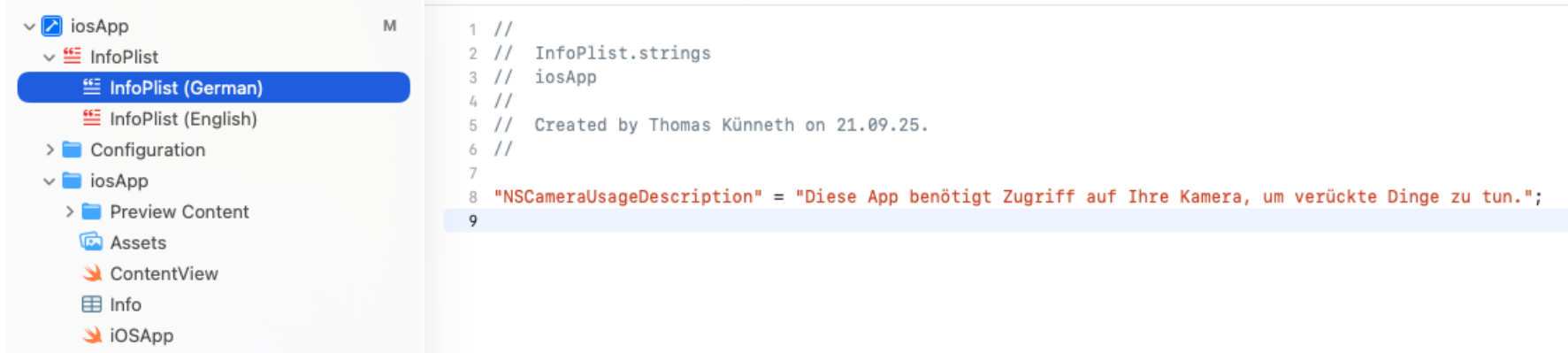
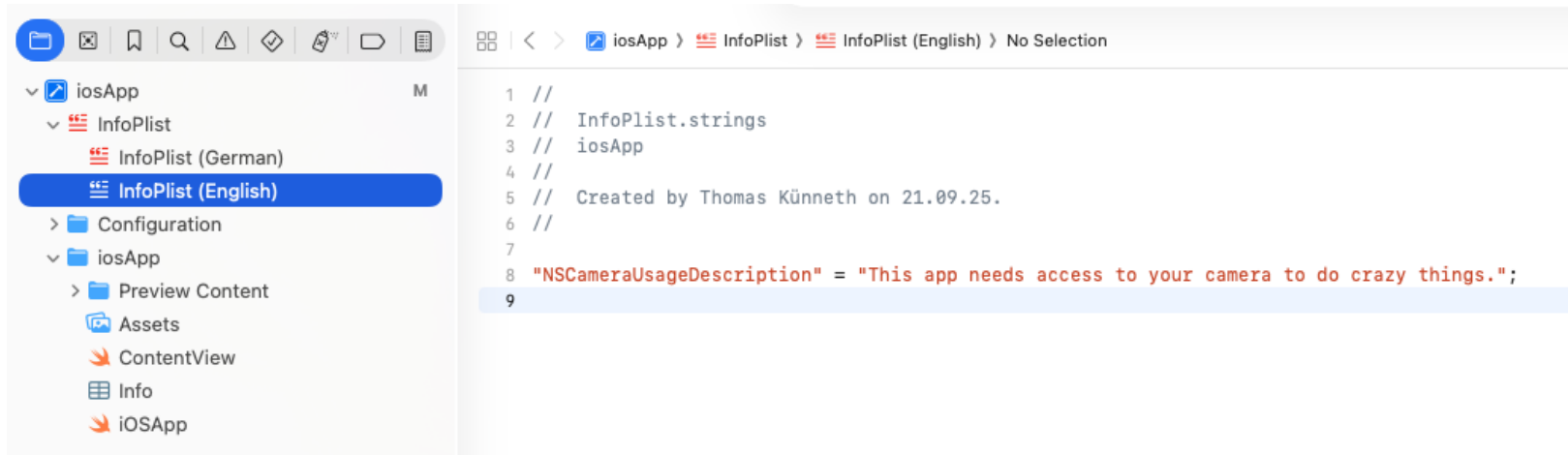
Like on Android, clearly and honestly state the reason for requesting a permission



What about internationalization?

- Step 1: Add languages to your project
- Step 2: Create an InfoPlist.strings file
- Step 3: Localize this file
- Step 4: Add your translated messages





5

Texts in Info.plist can be internationalized

Doing may feel a little more complicated than necessary 😊

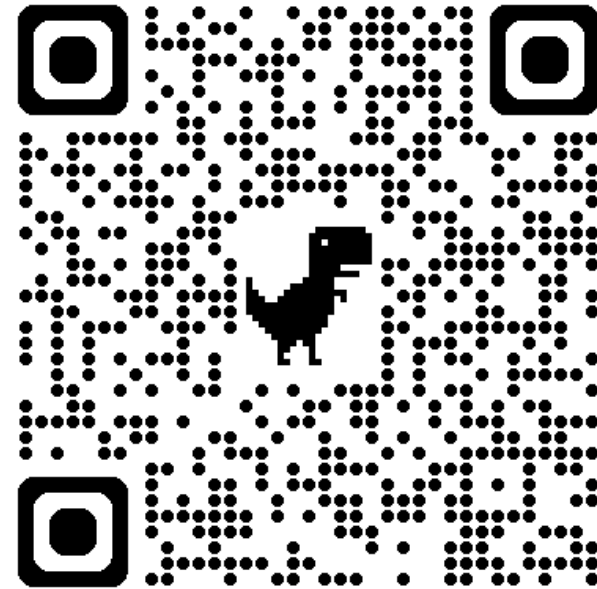
We can't just add the files and directories because they are referenced in project.pbxproj

Thank you!

 @tkuenneth

 @tkuenneth

 @tkuenneth.dev



<https://github.com/tkuenneth/InfoPlistManifestDemo>