
Splash Screens
Going deep for Flutter apps

FLUTTER

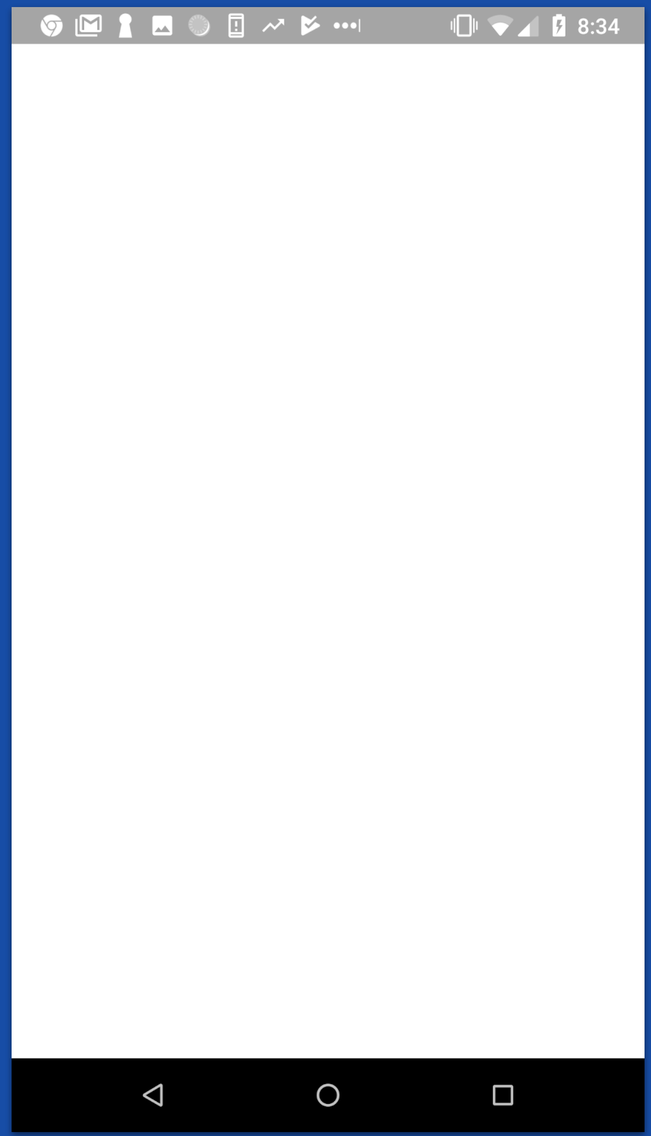
Every Flutter app has one

Default Flutter project provides it for Android & iOS

See it in the App on:

- Cold starts
- Warm starts (some)

We can/should change it !

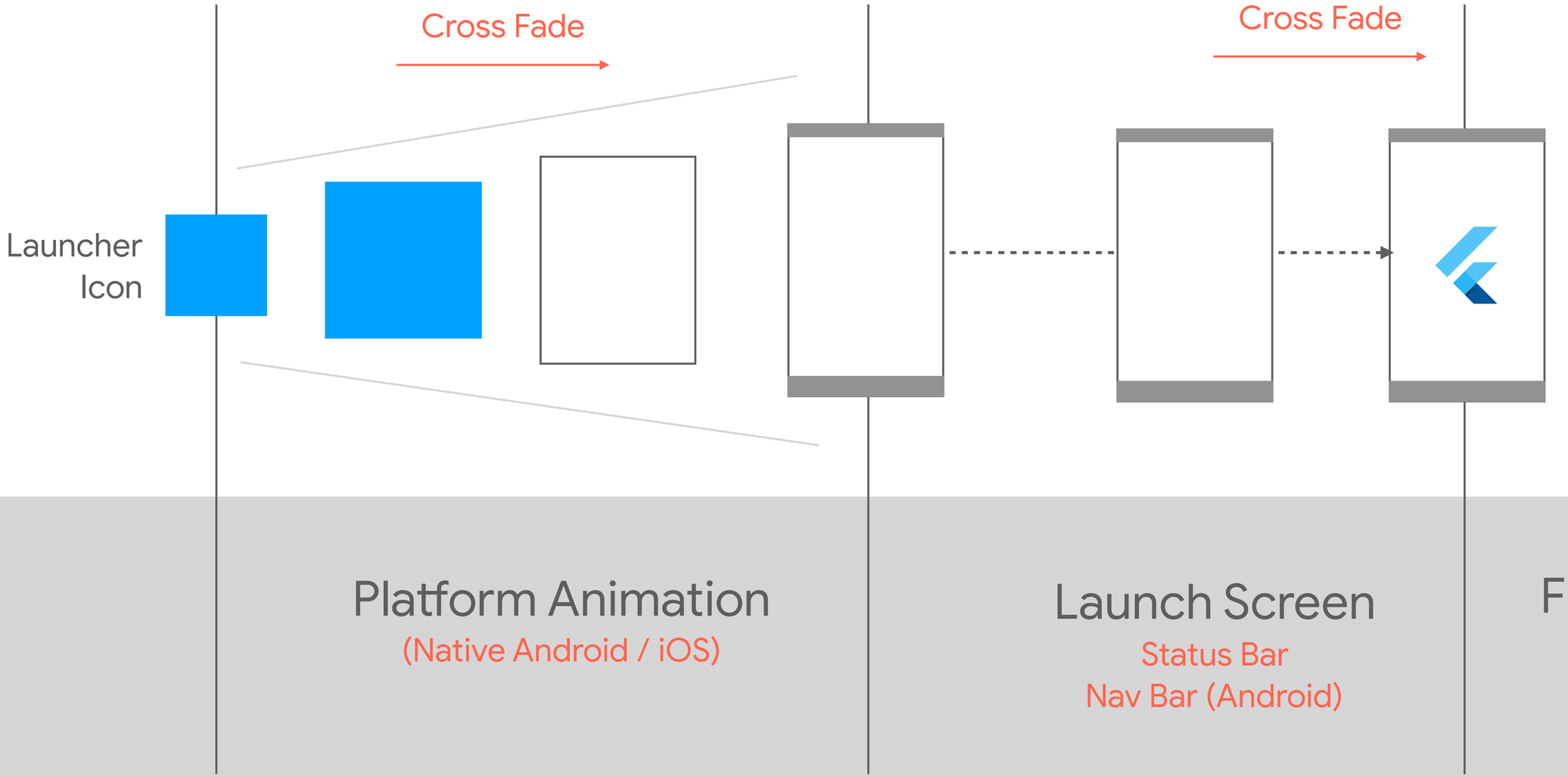


Official Flutter guidance says ...

Uses **native platform mechanisms** to draw launch screens while the Flutter framework loads

Persists until runApp() in the main() function of your Flutter app renders the first frame

WHATS HAPPENING



GUIDANCE

iOS

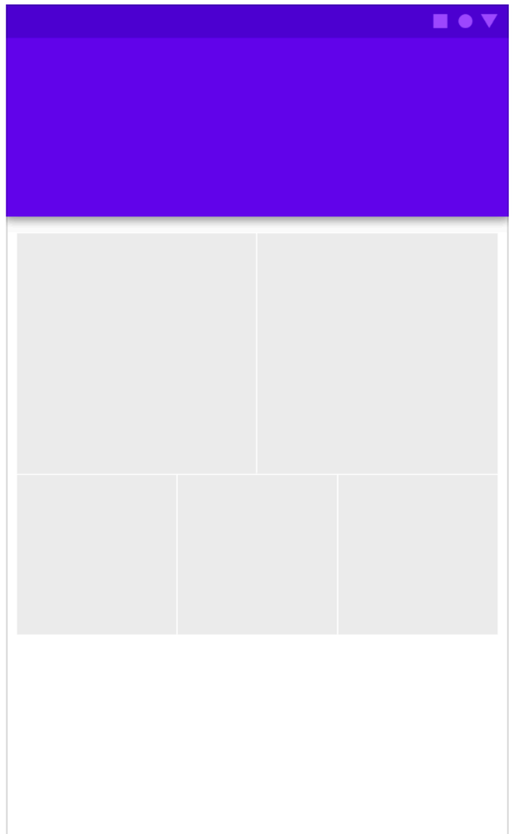
Communication > Launch screen > Usage

MATERIAL DESIGN

Types of launch screens

There are two types of launch screens:

- Placeholder UI launch screens display a non-interactive preview of the app's actual UI. This launch screen is appropriate for both app launches and activity transitions within an app.
- Branded launch screens provide momentary brand exposure.



Developer Discover Design Develop Distribute Sup

Human Interface Guidelines Overvie

- iOS
- App Architecture
- User Interaction
- System Capabilities
- Visual Design
- **Icons and Images**
  - Image Size and Resolution
  - App Icon
  - Custom Icons
  - **Launch Screen**
  - System Icons
- Bars
- Views
- Controls
- Extensions

macOS
tvOS
watchOS
- **Technologies**

Launch Screen

A launch screen appears instantly when your app starts up ar app's first screen, giving the impression that your app is fast a screen isn't an opportunity for artistic expression. It's solely in perception of your app as quick to launch and immediately re supply a launch screen.

Launch screens

user’s first experience of your app

shouldn’t be displayed if an app is running

decrease the sense of a long load time

can be displayed upon launch instead of a blank screen

types of Launch Screens;
Placeholder or
Branded

avoid using text

WHEN

WHY

WHAT

iOS

Launch screens

appears instantly when your app starts up

quickly replaced with the app's first screen

solely to enhance the perception as quick to launch

every app **must** supply a launch screen

nearly identical to the first screen ie. placeholder
isn’t a branding opportunity
static images (Not Recommended)

avoid including text

CUSTOM LAUNCH SCREEN

x2 Methods

PNG multiple images by pixel density

Vector images

iOS

x2 Methods

PNG multiple images by pixel density

Vector images

Flutter documentation approach on flutter.dev and all Flutter projects are setup for my default:

In the file /android/app/src/main/AndroidManifest.xml this line:

```
android:theme="@style/LaunchTheme"
```

- Is present by default
- Means there is a style (in **styles.xml** file) has a theme called **LaunchTheme**
- You don’t need to change anything here

/android/app/src/main/AndroidManifest.xml


```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.flutter_app9">

    <!-- io.flutter.app.FlutterApplication is an android.app.Application that
         calls FlutterMain.startInitialization(this); in its onCreate method.
         In most cases you can leave this as-is, but you if you want to provide
         additional functionality it is fine to subclass or reimplement
         FlutterApplication and put your custom class here. -->
    <application
        android:name="io.flutter.app.FlutterApplication"
        android:label="flutter_app9"
        android:icon="@mipmap/ic_launcher">
        <activity
            android:name=".MainActivity"
            android:launchMode="singleTop"
            android:theme="@style/LaunchTheme"
            android:configChanges="orientation|keyboardHidden|keyboard|screenSize|locale
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">
            <!-- This keeps the window background of the activity showing
                 until Flutter renders its first frame. It can be removed if
                 there is no splash screen (such as the default splash screen
                 defined in @style/LaunchTheme). -->
            <meta-data
                android:name="io.flutter.app.android.SplashScreenUntilFirstFrame"
                android:value="true" />
            <intent-filter>
```

Set to false .... get a BLACK screen

In the file /android/app/src/main/res/values/styles.xml this line

```
<style name="LaunchTheme" parent="@android:style/Theme.Black.NoTitleBar">
```

- Theme called **LaunchTheme** is based on the Android default theme of **Theme.Black.NoTitleBar** which as name suggests:
  - Has no Action bar (just Status & Navigation bars)
  - Black Navigation bar

- Many other Android default themes you, or define your own to change Status & Navigation bar colouring

Other key line in /android/app/src/main/res/values/styles.xml means

```
<item name="android:windowBackground">@drawable/launch_background</item>
```

- Window (main window between Status bar and Navigation bar) is a drawable called **launch_background**
- You don’t need to change anything here

/android/app/src/main/res/values/styles.xml

Themes will impact Status / Nav bars !!!

```
<resources>
    <style name="LaunchTheme" parent="@android:style/Theme.Black.NoTitleBar">
        <!-- Show a splash screen on the activity. Automatically removed when
             Flutter draws its first frame -->
        <item name="android:windowBackground">@drawable/launch_background</item>
    </style>
</resources>
```

File /android/app/src/main/res/drawable/launch_background.xml is named after the drawable in **styles.xml**

```
<item android:drawable="@android:color/white" />
```

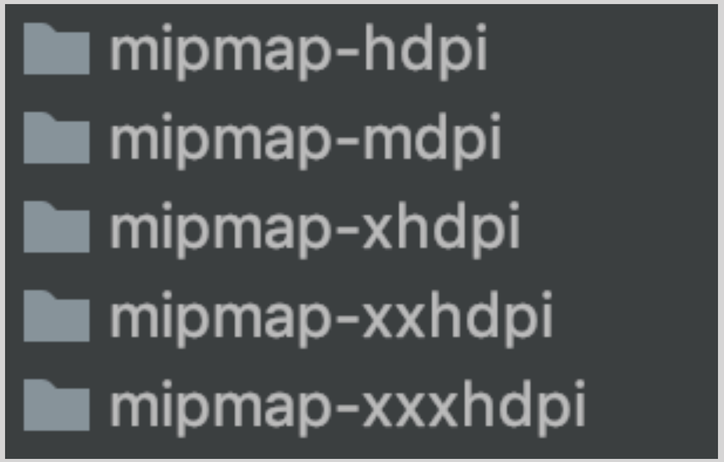
- Makes the screen the Android default colour of **white**
- Can define your own colors but you will have to add a **colors.xml** file for those colour definitions.

Other key lines in /android/app/src/main/res/drawable/launch_background.xml are these:

```
<!-- You can insert your own image assets here -->
<item>
    <bitmap
        android:gravity="center"
        android:src="@mipmap/launch_image" />
</item>
```

- Uncomment the **<item>** tags
- You will need an image called **launch_image.png** (note the .png is dropped in code)
- Add the **launch_image.png** file into the multiple **mipmap-....** directories with the correct resolutions

Default mipmap-... directories in /android/app/src/main/res/ are these



- Add a **launch_image.png** file of different densities to these directories
- Android densities: developer.android.com/training/multiscreen/screendensities

/android/app/src/main/res/drawable/launch_background.xml


```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- Modify this file to customize your launch splash screen -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@android:color/white" />

    <!-- You can insert your own image assets here -->
    <!-- <item>
        <bitmap
            android:gravity="center"
            android:src="@mipmap/launch_image" />
    </item> -->
</layer-list>
```

Suggested to use a launch_image
@mipmap means 4x versions of differing densities
Image entered

VECTOR DRAWABLE METHOD

A *better* method using Android Vector Drawables not covered in the Flutter documentation:

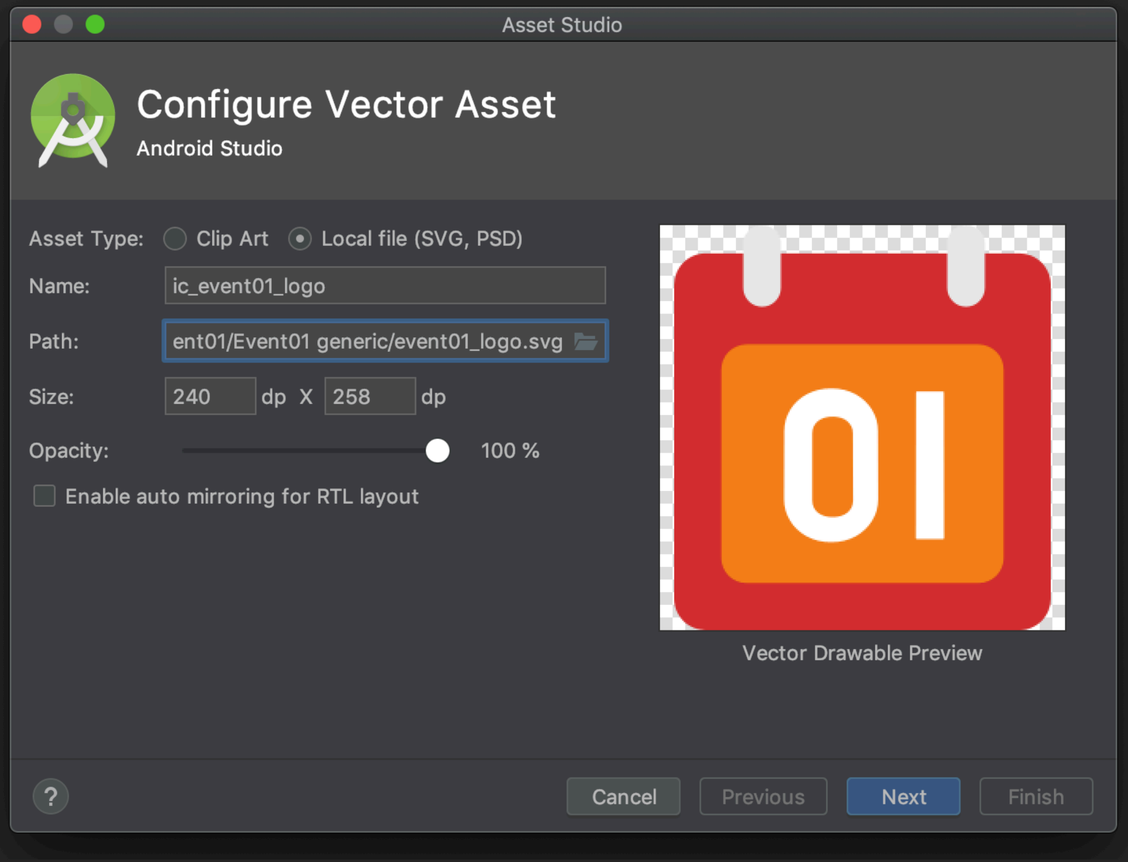
- Uses one vector drawable image, rather than **many** bitmap PNG images of varying densities
- Scales automatically to any device pixel density
- Vector drawable image is smaller than many bitmap PNG images

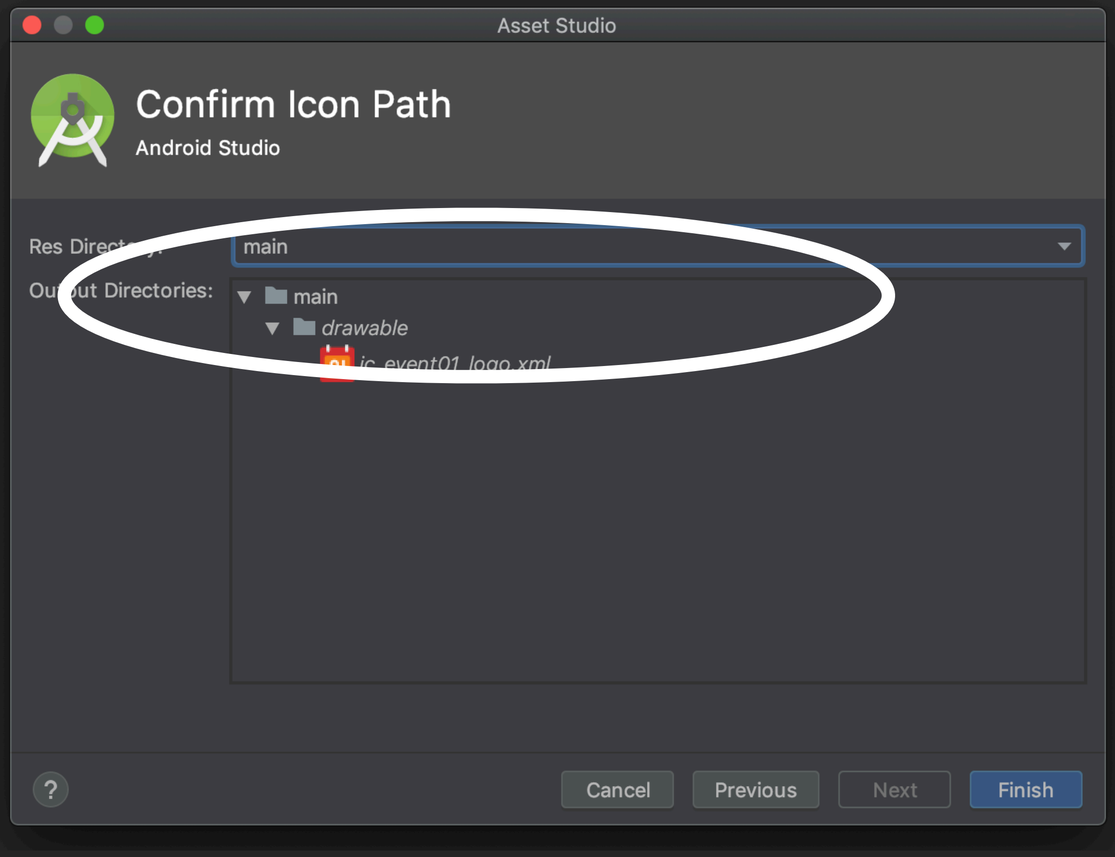
VECTOR DRAWABLE METHOD

To create your vector drawable image from an SVG image in your project using Android Studio (v3.x)

Right click on /android/app/src/main/res folder in your project and select New / Vector Asset

Use Path to select your image SVG file and Name for the vector drawable file that will be produced





Update the /android/app/src/main/res/drawable/launch_background.xml file. Make sure the **<bitmap>** section of code is commented out:

```
<!-- You can insert your own image assets here -->
<!-- <item>
    <bitmap
        android:gravity="center"
        android:src="@mipmap/launch_image" />
</item> —>
```

Add this **<item>** section:

```
<item
    android:gravity="center"
    android:drawable="@drawable/ic_event01_logo"/>
```

The vector drawable image file name MUST be correct !

VECTOR DRAWABLE METHOD

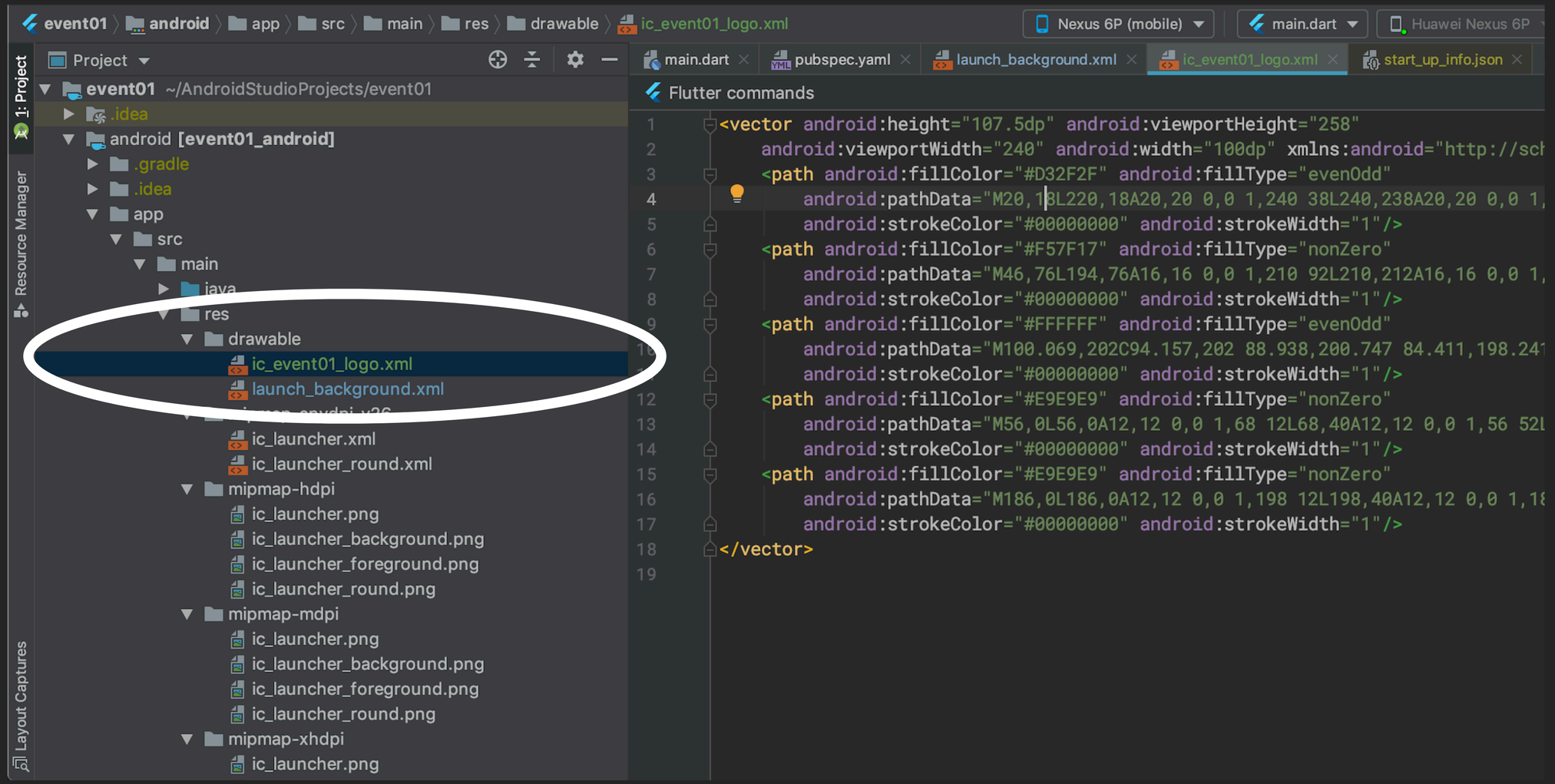
launch_background.xml layout file now looks like this

Don't use the <bitmap> tag like for PNGs as it is a drawable

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Modify this file to customize your launch splash screen -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:drawable="@android:color/white" />

<!--    This is Bitmap like PNG for the logo on the Launch Screen-->
    <!-- You can insert your own image assets here -->
<!--    <item>-->
<!--        <bitmap-->
<!--            android:gravity="center"-->
<!--            android:src="@mipmap/launch_image" />-->
<!--    </item>-->

<!--    This is using Vector Drawable from an SVG for the logo on Splash screen-->
    <item
        android:gravity="center"
        android:drawable="@drawable/ic_event01_logo"/>

</layer-list>
```

VECTOR DRAWABLE METHOD

Puts the vector drawable xml file into the /res/drawable folder of project by default



iOS

Flutter documentation approach on flutter.dev and all Flutter projects are setup for my default:

In your project navigate to /ios/Runner/Assets.xcassets/LaunchImage.imageset and copy in PNG images named:

- LaunchImage.png
- LaunchImage@2x.png
- LaunchImage@3x.png
- LaunchImage@4x.png (is not standard but it is a good idea to future-proof your app)

/ios/Runner/Assets.xcassets/LaunchImage.imageset/Contents.json

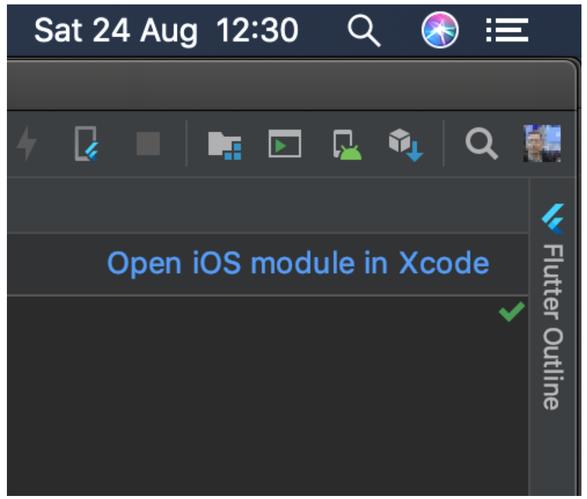
```
{
  "images" : [
    {
      "idiom" : "universal",
      "filename" : "LaunchImage.png",
      "scale" : "1x"
    },
    {
      "idiom" : "universal",
      "filename" : "LaunchImage@2x.png",
      "scale" : "2x"
    },
    {
      "idiom" : "universal",
      "filename" : "LaunchImage@3x.png",
      "scale" : "3x"
    }
  ],
  "info" : {
    "version" : 1,
    "author" : "xcode"
  }
}
```

By default Contents.json looks like this.

If you use different filenames than LaunchImage you'll also have to update the Contents.json file in the same directory

If you add a 4x image, also update this file.

Xcode can also be used instead. You can fully customize your launch screen storyboard in Xcode. In your project navigate to /ios/Runner.xcworkspace and click the Open iOS module in Xcode link (top right corner of Android Studio):



Or in a Terminal session, in the root of your project folder run the command:

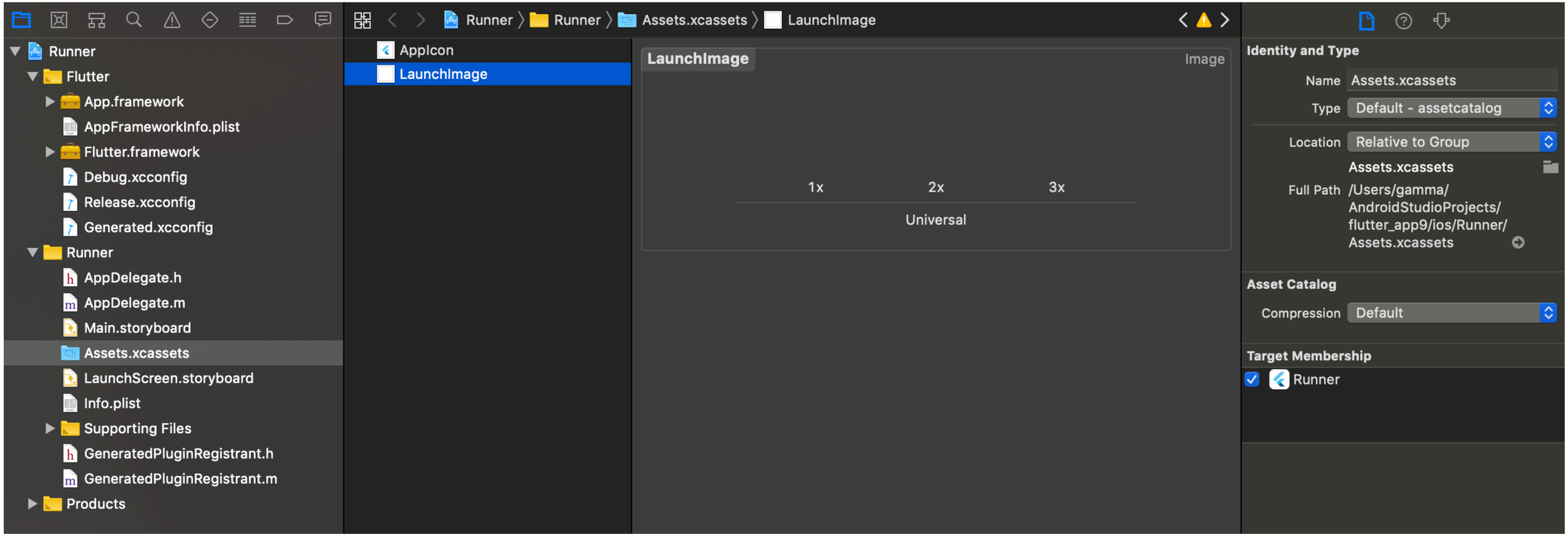
open ios/Runner.xcworkspace

MULTIPLE PNG METHOD

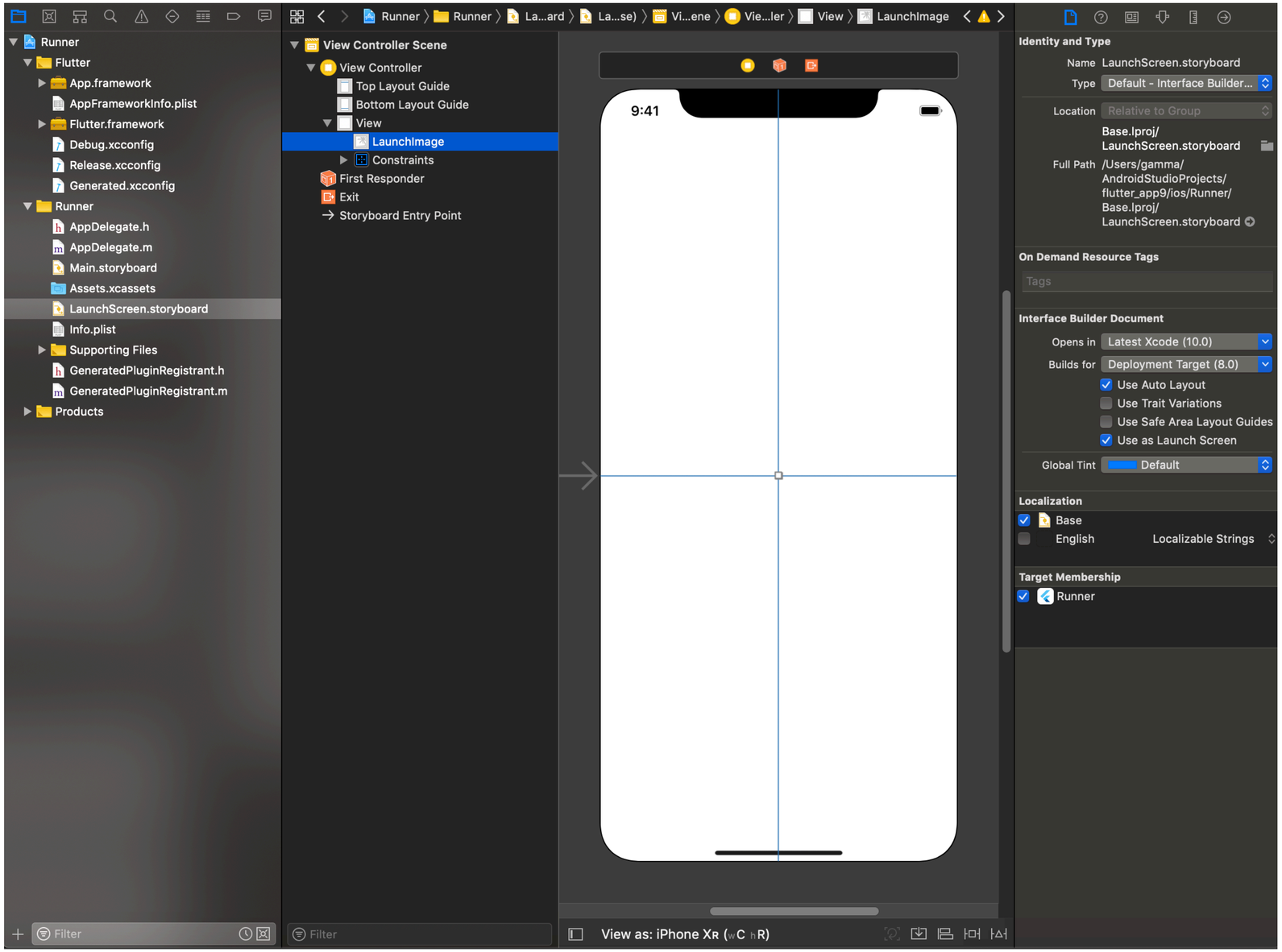
iOS

In Xcode, navigate to Runner/Runner and select Assets.xcassets and the LaunchImage

Drag and drop each PNG image file into Xcode over the 1x, 2x or 3x boxes:



MULTIPLE PNG METHOD



In Xcode, you can also do any customization using the Interface Builder in LaunchScreen.storyboard

We'll do this **next** when looking at how to use PDF vector images for iOS launch screens!

iOS

A *better* method using PDF Vector Images not covered in the Flutter documentation.

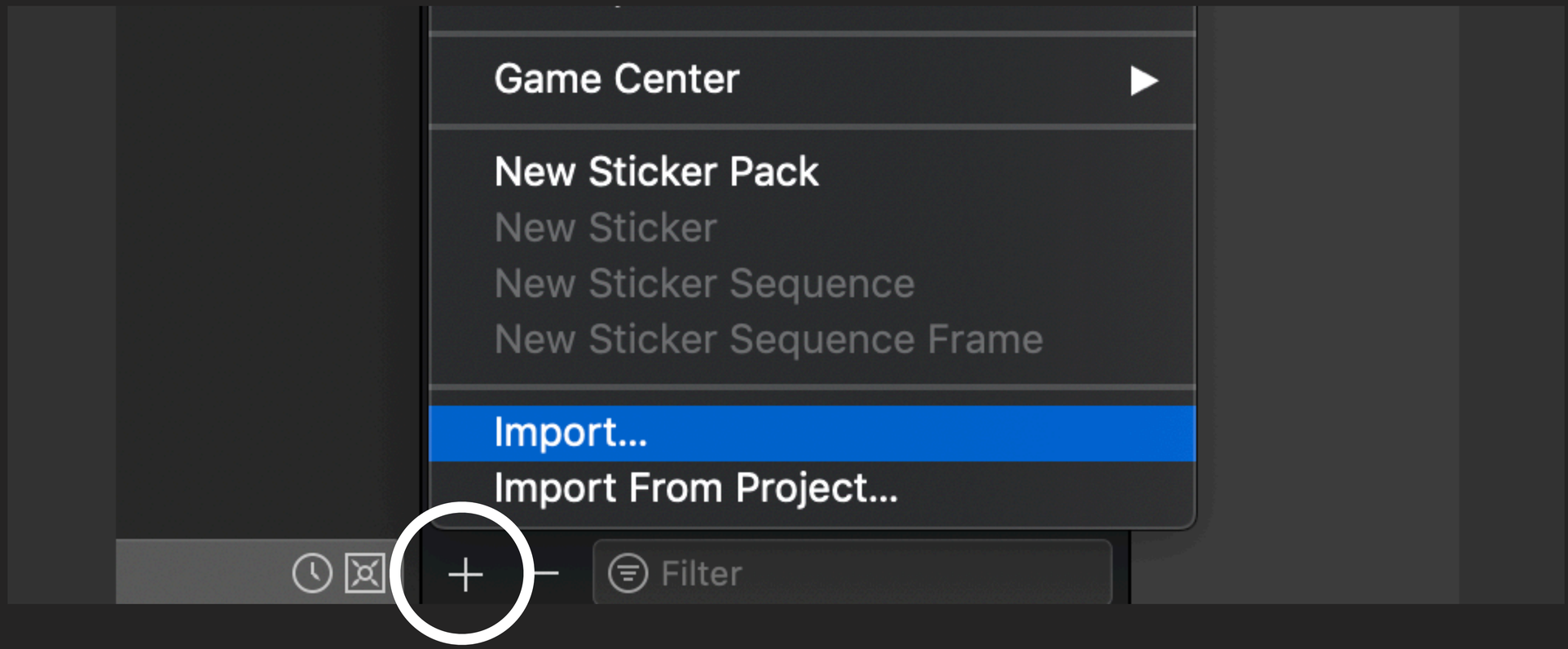
You will need a **PDF Vector** formatted copy of your image !!!

I used Sketch to create the image I wanted and then Exported as PDF. Others apps are Pixelmator or Autodesk Graphic can create Vector PDF images.

iOS

Open Xcode for your project and navigate to Runner / Runner / Assets.xcassets

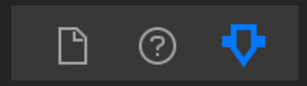
Click + at bottom, select Import, then select your PDF vector image file

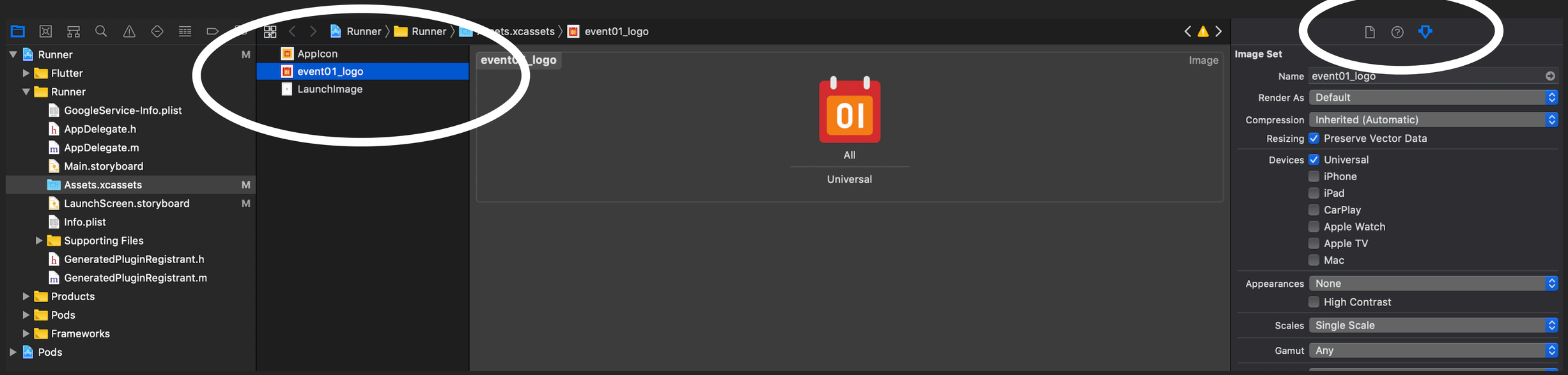


PDF VECTOR METHOD

Now you should see your new vector PDF image e.g. event01_logo

Select your image

Select the image **Attributes** with this icon 



The

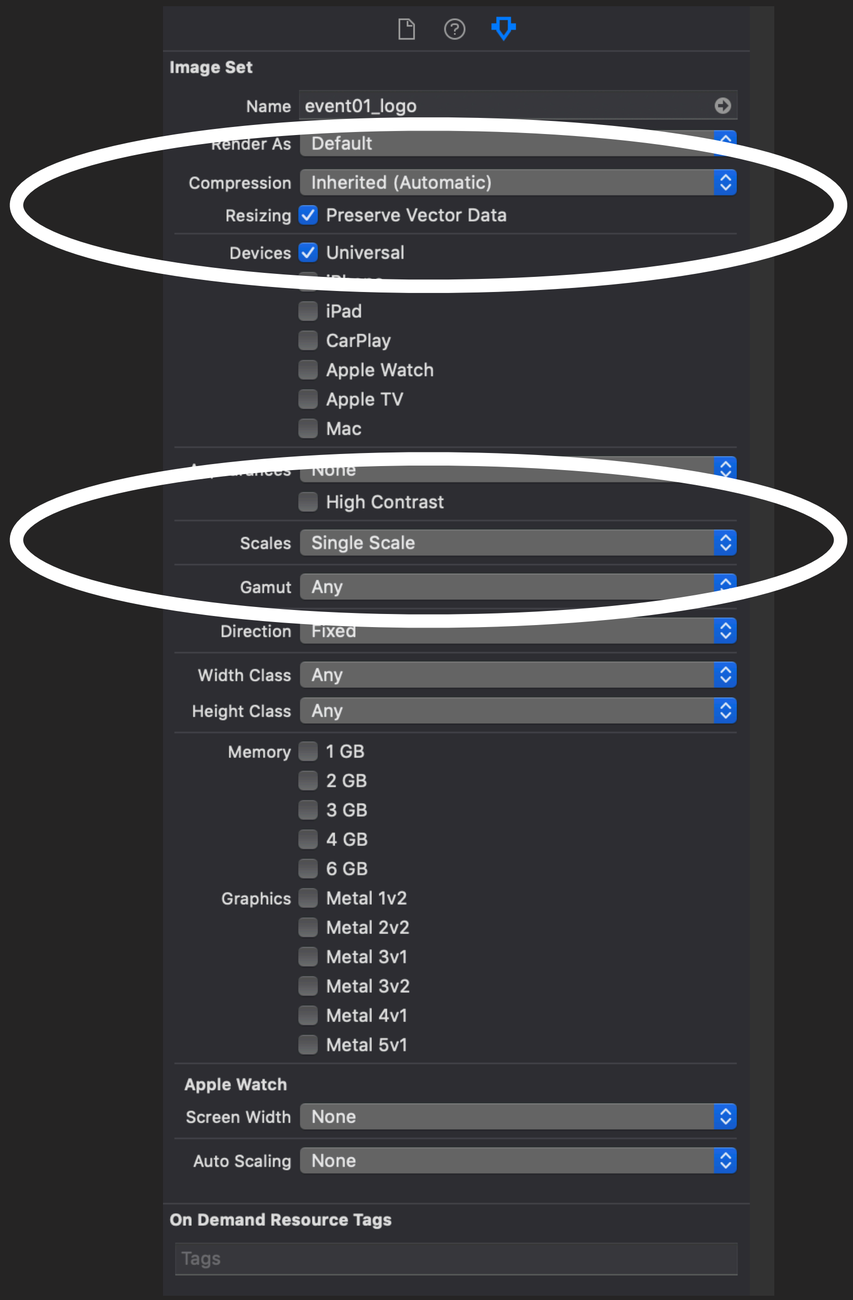
PDF VECTOR METHOD

Set these attributes:

- Preserve Vector Data
- Scales - set to Single Scale

If you are using Interface Builder the scaled image will still look blurry in the Xcode preview but it is fine at runtime in the simulator and on a device.

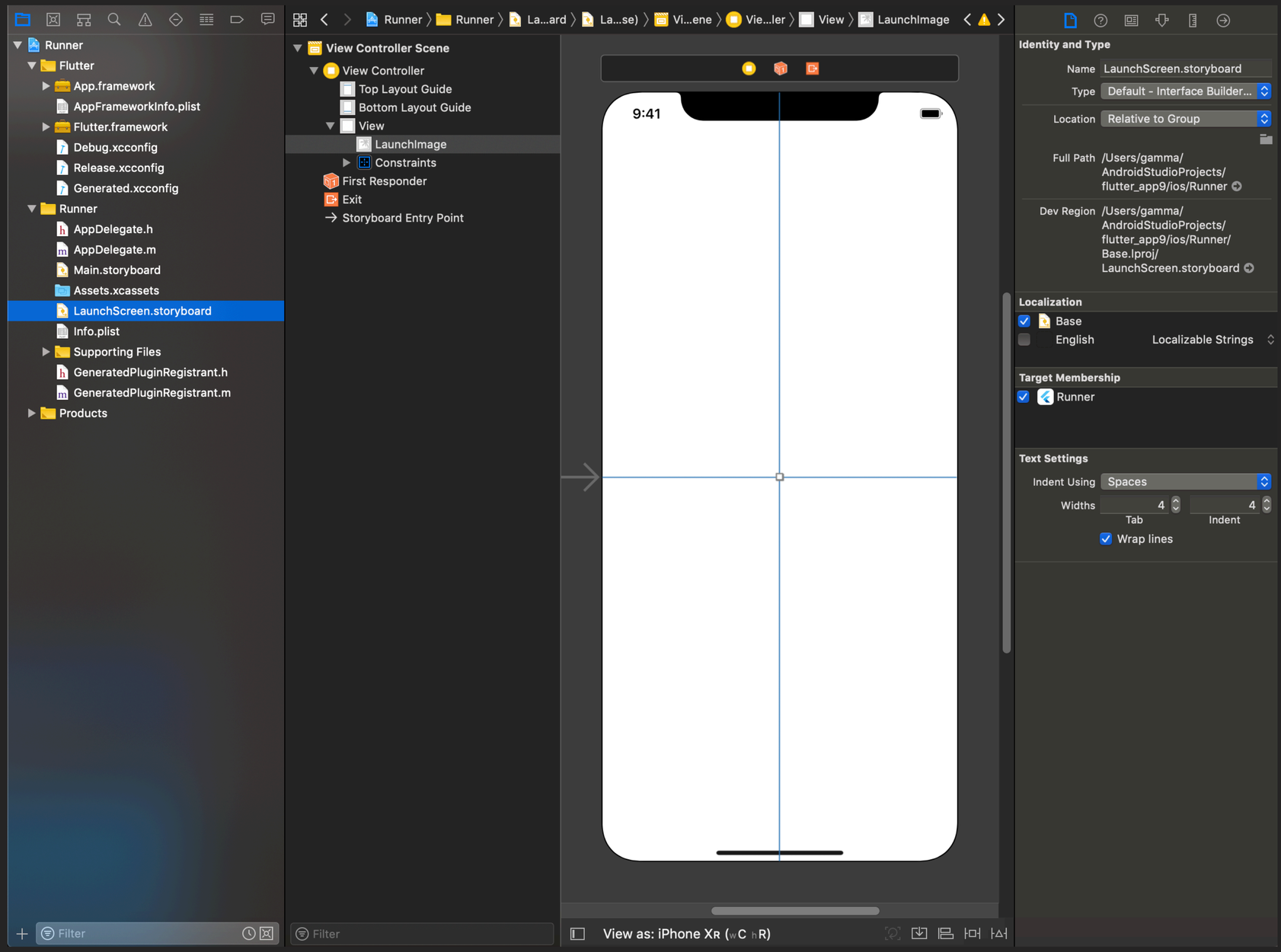
Requires Xcode 11 or higher.



PDF VECTOR METHOD

Now we have the PDF vector image in Xcode select LaunchScreen.storyboard which launches the Interface Builder in Xcode.

Expand the View Controller Scene which is using the LaunchImage by default

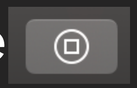


iOS

PDF VECTOR METHOD

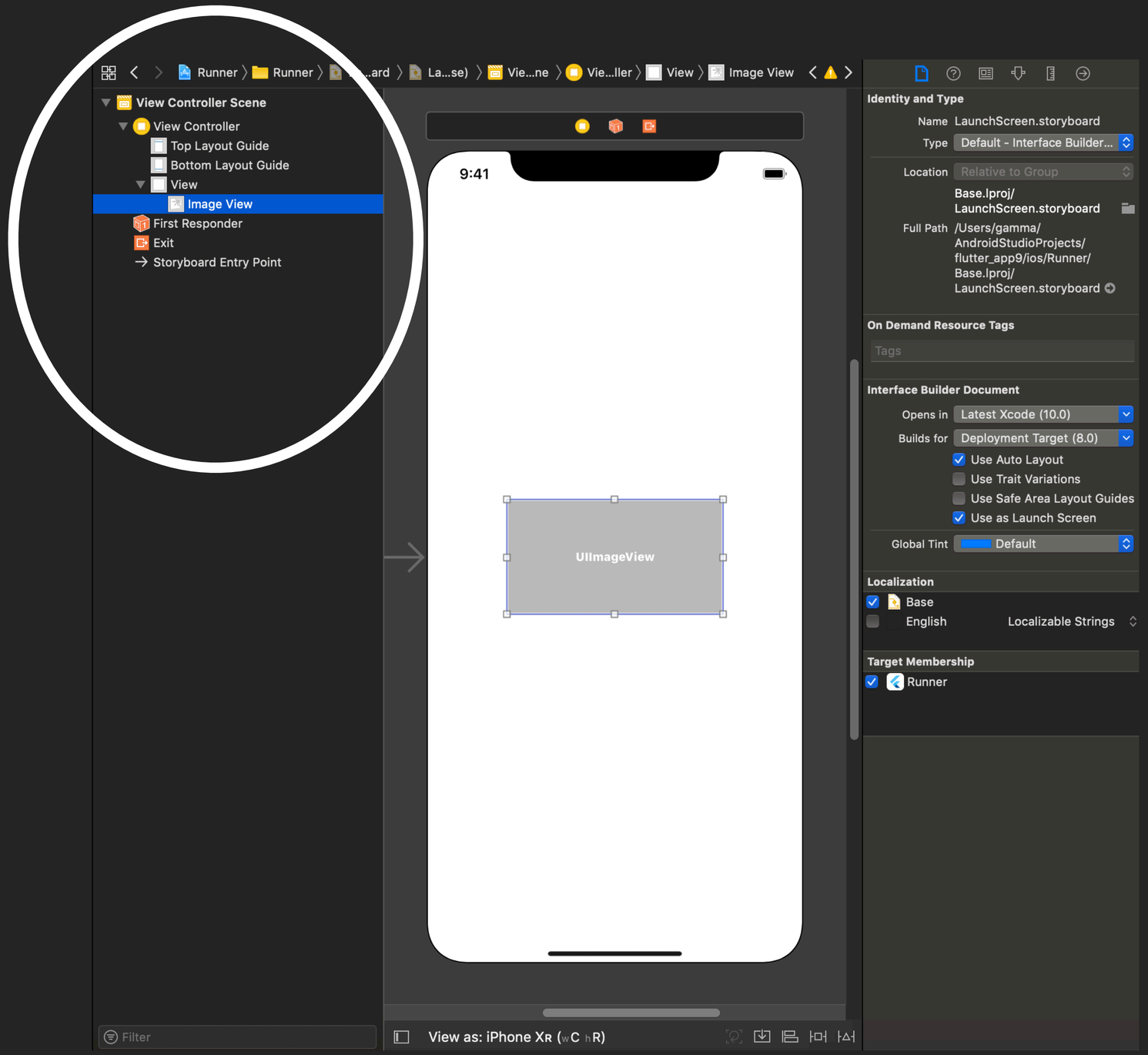
With LaunchImage selected press delete to remove it

Select View

Use  and search for *image*

Select Image View and press enter

iOS

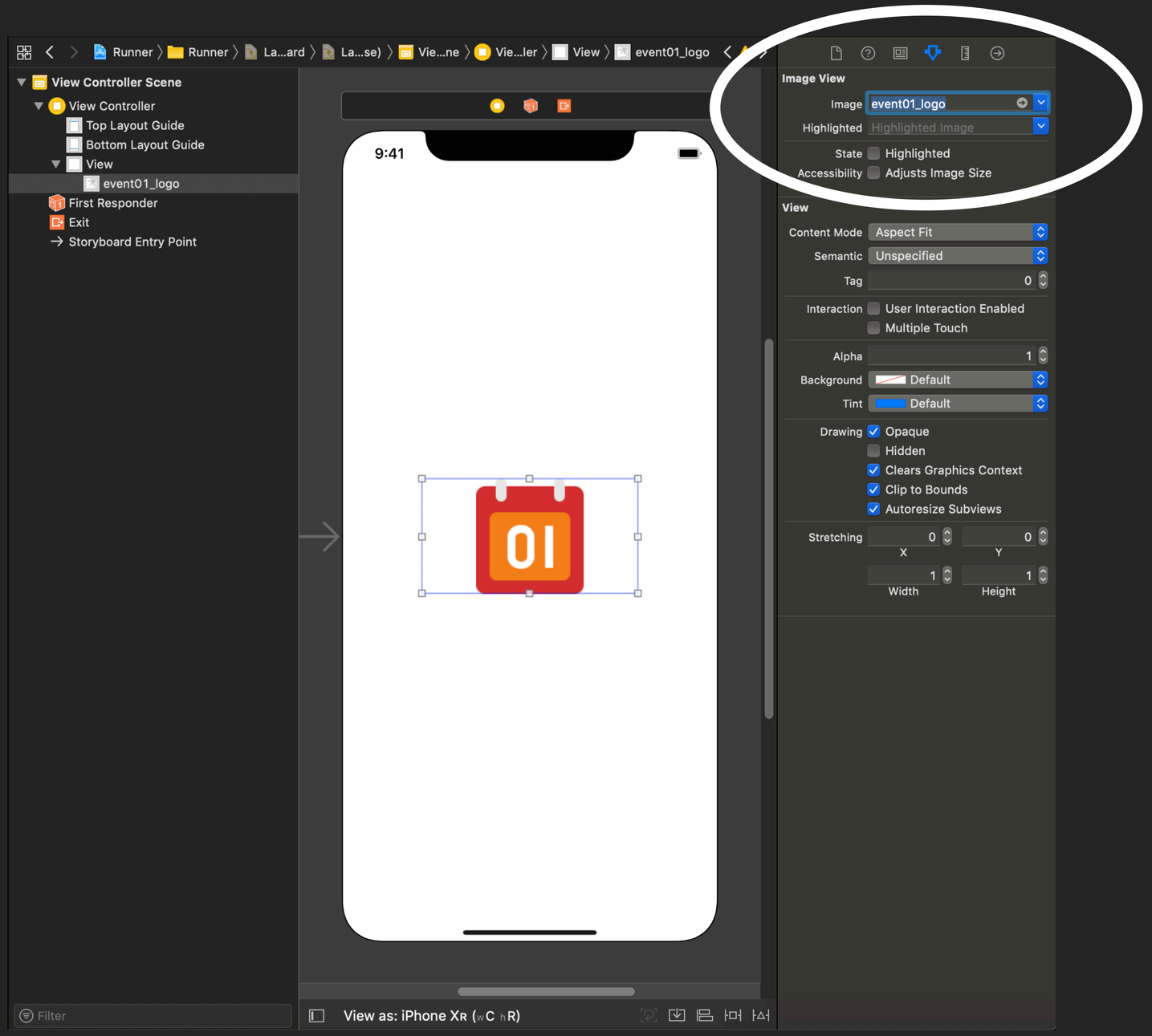


PDF VECTOR METHOD

With ImageView selected

Click the Attributes icon

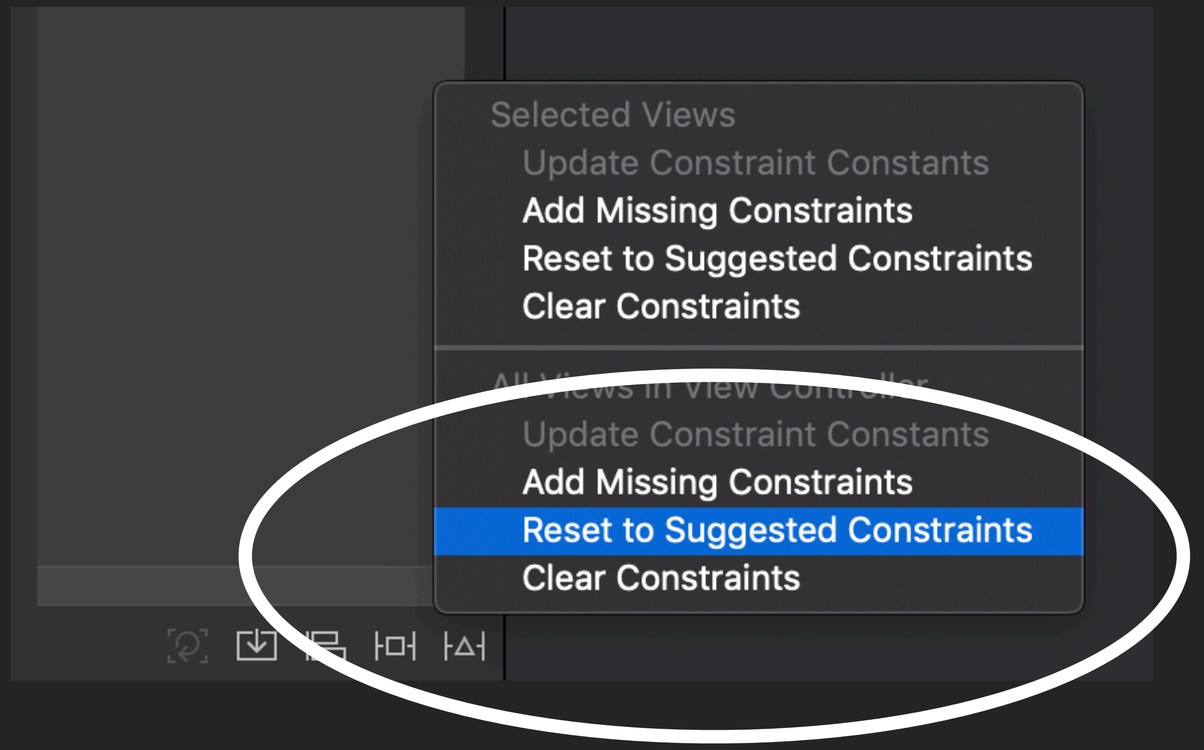
Now use Image to change the images to your PDF vector image



iOS

PDF VECTOR METHOD

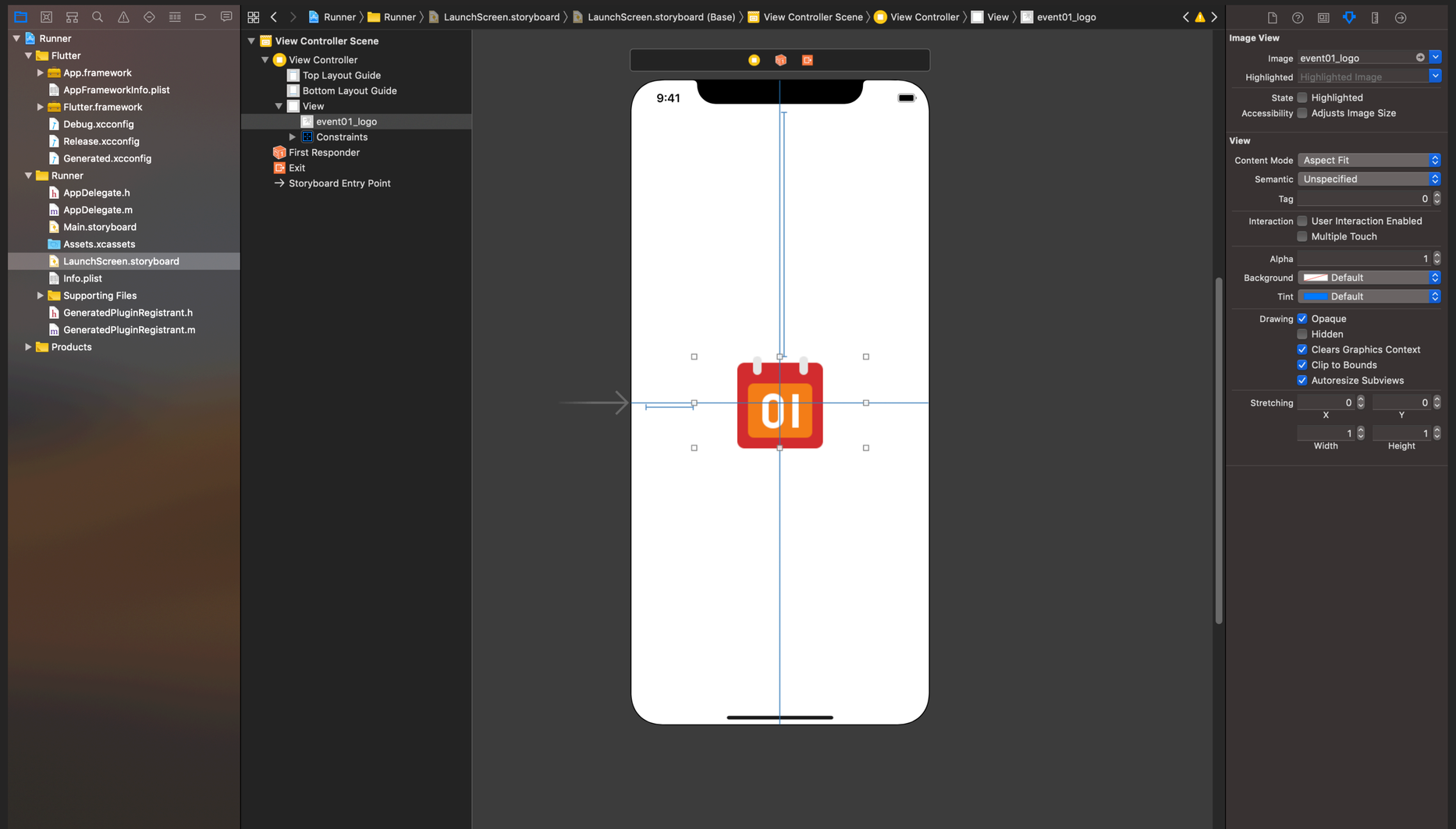
Use the controls at the bottom to add Suggested Constraints that centre the image on screen



PDF VECTOR METHOD

iOS

Should look something like this. You can now close Xcode.



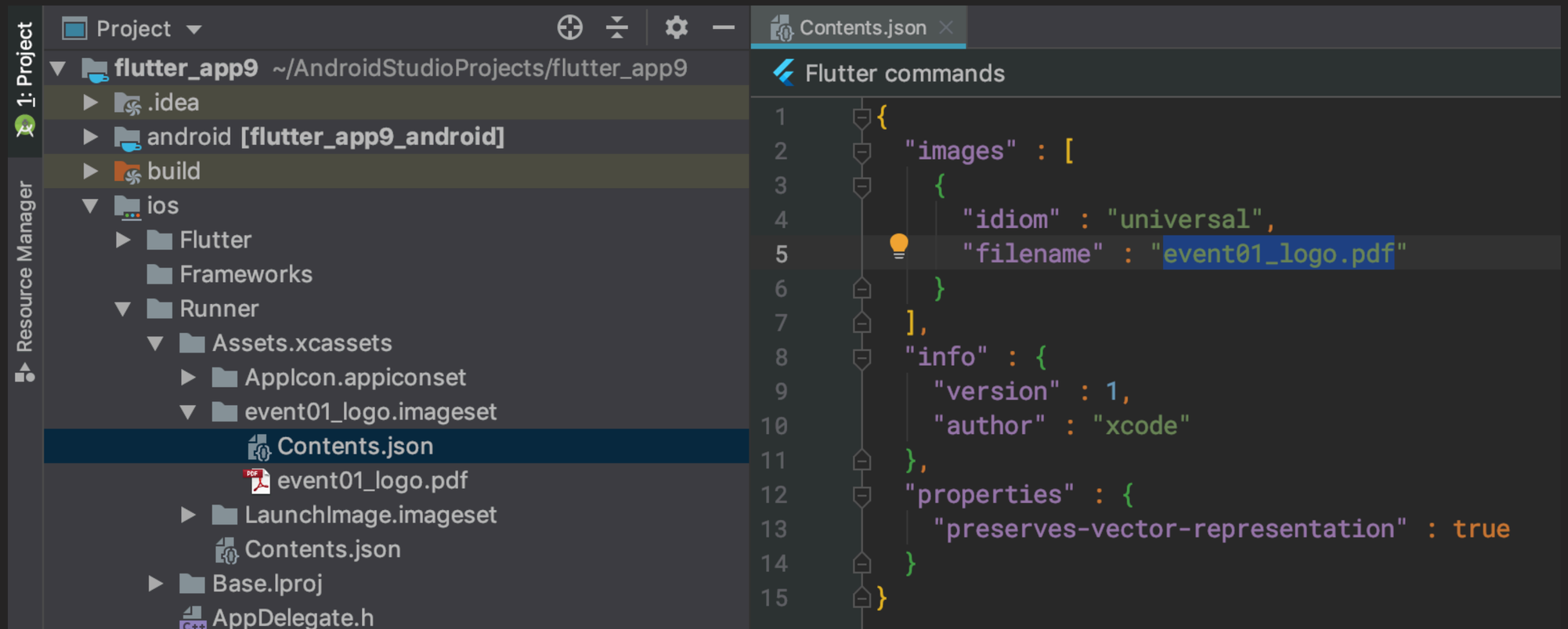
PDF VECTOR METHOD

iOS

In your project you should now have a new folder named:

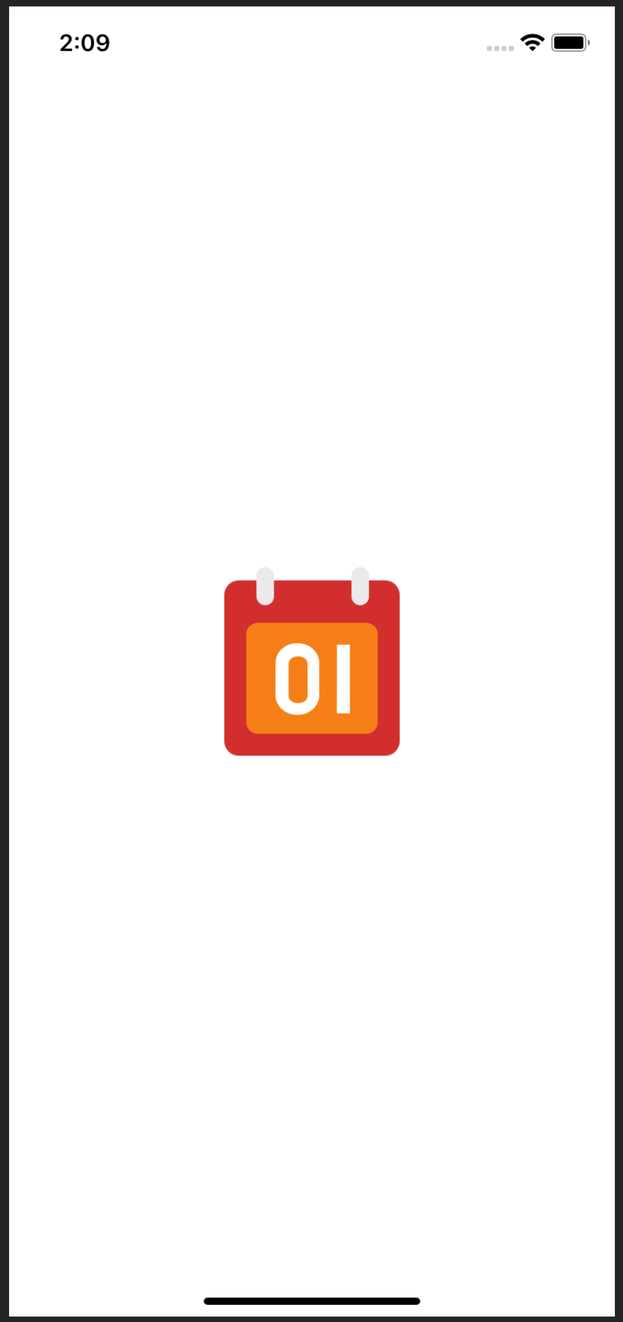
/ios/Runner/Assets.xcassets/*your_image_name*.imageset

Should look something like this with a Contents.json and your PDF vector image file



PDF VECTOR METHOD

Now when you run your app you should see your image on the Launch Screen centered and crisp because it is a vector image !



flutter run —trace-startup —profile

Run in terminal in the root directory of your project

See output file in your project for the details /build/start_up_info.json

277 ms Android app startup

583 ms Until finish loading Flutter framework & paint first frame of app

```
[gamma flutter_splash_testing $ flutter run --trace-startup --profile                ]
Initializing gradle...                                            1.1s
Resolving dependencies...                                         1.9s
Launching lib/main.dart on Nexus 6P in profile mode...
Running Gradle task 'assembleProfile'...
Running Gradle task 'assembleProfile'... Done                    35.7s
Built build/app/outputs/apk/profile/app-profile.apk (10.4MB).
Installing build/app/outputs/apk/app.apk...                       6.1s
Tracing startup on Nexus 6P.
D/OpenGLRenderer(17753): HWUI GL Pipeline
I/OpenGLRenderer(17753): Initialized EGL, version 1.4
D/OpenGLRenderer(17753): Swap behavior 2
Error -32601 received from application: Method not found
Waiting for application to render first frame...                  277ms
D/vndksupport(17753): Loading /vendor/lib64/hw/android.hardware.graphics.mapper@2.0-impl.so from current namespace instead of spha
l namespace.
D/vndksupport(17753): Loading /vendor/lib64/hw/gralloc.msm8994.so from current namespace instead of sphal namespace.
Time to first frame: 583ms.
Saved startup trace info in build/start_up_info.json.
Application finished.
```

LAUNCH TO FIRST SCREEN

DO Lazy Load everything on first screen ie. Onboarding or Login & Registration

DON’T Create another Splash screen INSIDE your Flutter app

DON’T - Hold the launch screen

- for a fixed X seconds to cover up background work
- until **all** setup/background tasks are done ... only then render

EXAMPLES

Thank you ...

RESOURCES ANDROID

Material Design guidance on Launch Screens:
https://material.io/design/communication/launch-screen.html#usage

Flutter official guidance on Launch Screens which is very brief and outlines the multiple PNG image method:
flutter.dev/docs/development/ui/assets-and-images#updating-the-launch-screen

Flutter guidance on measuring app startup time to Flutter's first frame:
flutter.dev/docs/development/ui/assets-and-images#updating-the-launch-screen

Android Developers app startup time reference:
https://developer.android.com/topic/performance/vitals/launch-time

Android Developers App Startup Time, Detect and Diagnose problems:
developer.android.com/topic/performance/vitals/launch-time#ddp

RESOURCES IOS

Apple Human Interface Guidelines on Launch Screens:
https://developer.apple.com/design/human-interface-guidelines/ios/icons-and-images/launch-screen/

Flutter official guidance on Launch Screens which is very brief and outlines the multiple PNG image method:
flutter.dev/docs/development/ui/assets-and-images#updating-the-launch-screen

Flutter guidance on measuring app startup time to Flutter's first frame:
flutter.dev/docs/development/ui/assets-and-images#updating-the-launch-screen

Xcode Interface Builder doco:
https://developer.apple.com/xcode/interface-builder