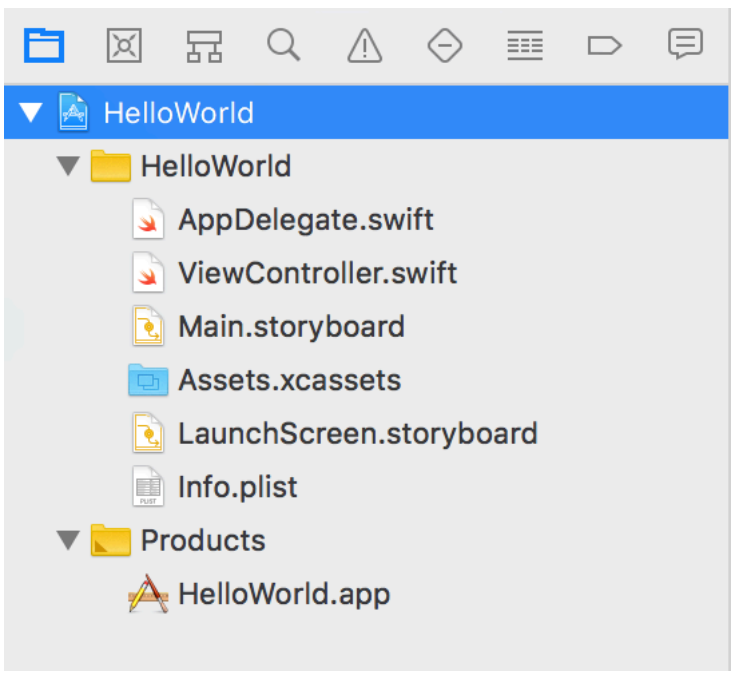
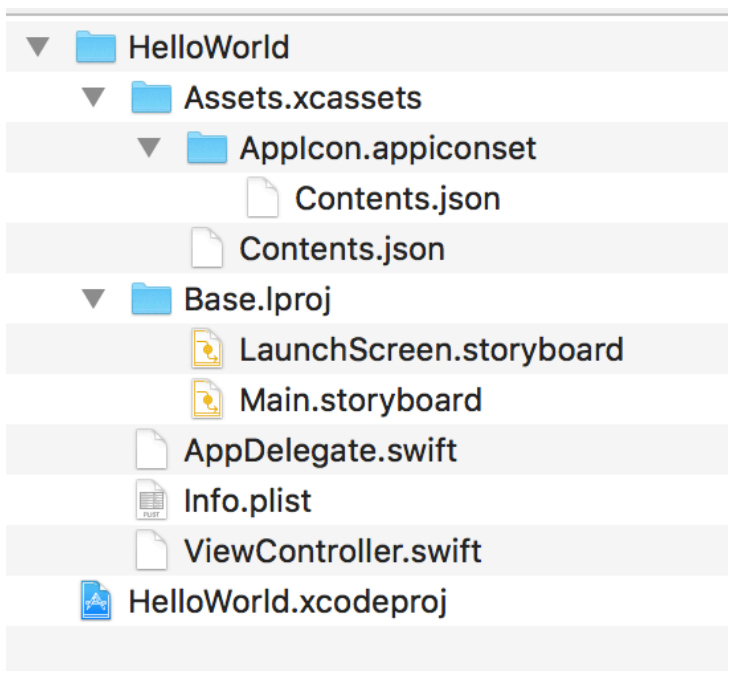
CHAPTER THREE : XCODE PROJECT AND FILES

Before moving on to our HelloWorld app, in order to actually work on the app, we have to understand the structure of the project. Some of the concepts might be a little bit abstract and theoretical, but these are the files that we will be working with all the time so a good understanding of their functions is really helpful when you have to deal with some complex structures in big projects. So have some patience and let's nail this chapter.

I. PROJECT STRUCTURE

If you paid attention to the **Navigator** in the last chapter when I told you to go from the root to Main.storyboard, you might have already noticed about all these files that are created for us by Xcode. What exactly are they?





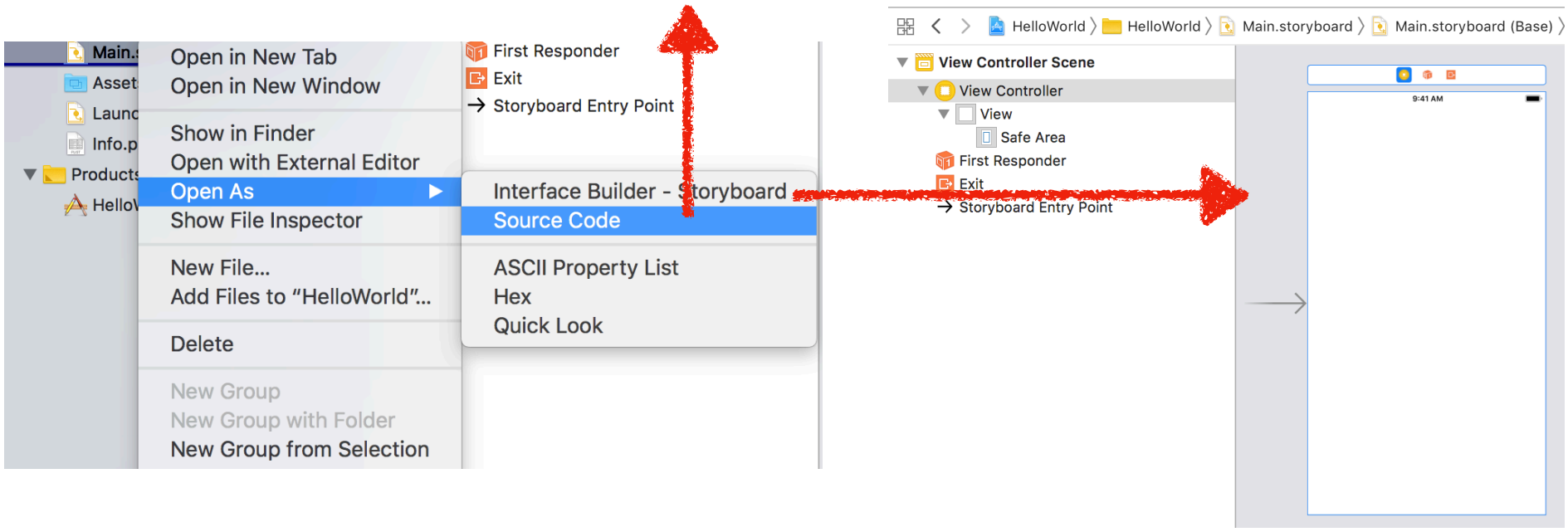
If you use finder to go to the root folder of your project, you will see a slightly different structure. For the most of the time, we only work on the one in our Xcode. The additional files you see directly with Finder are the files for Xcode to organize and present your project, and we are not supposed to modify them manually, ever!

II. STORYBOARDS

Main.storyboard:

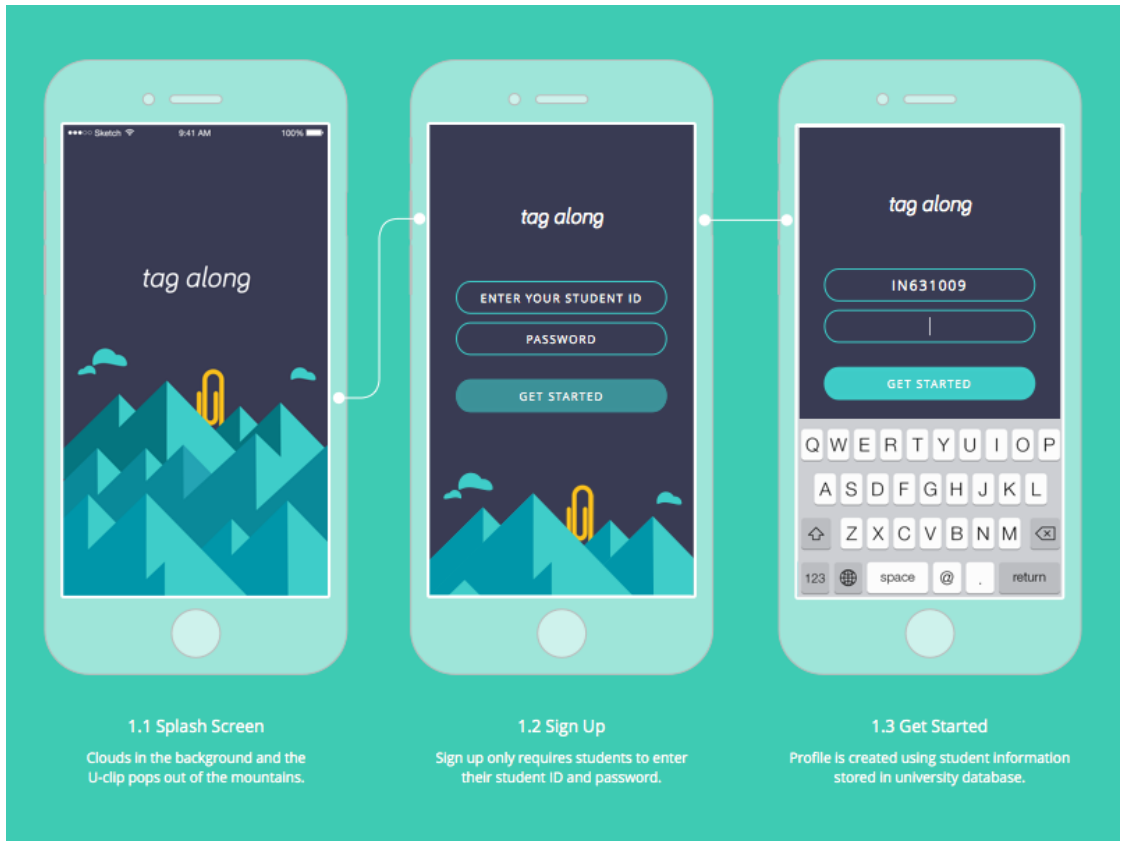
This is the file that represents our main user interface. It is an XML file and we(almost always) visualize it in a graphical way. But keep in mind that you can also visualize it in XML. Right click **Main.storyboard**, then choose **Open As > Source Code** or **Interface Builder**. They are just two different representation of the same structure. Before we can use storyboards in Xcode, developers have to write code to build the interface. Now with storyboards, we can build much more complex UIs in a more efficient and more pleasant way. Some iOS engineers still prefer building their apps without this feature, but for beginners, we should stick to the storyboard and if you are interested in the old way, you can try it out.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0" toolsVersion="14460.31"
       targetRuntime="iOS.CocoaTouch" propertyAccessControl="none" useAutolayout="YES" useTraitCollections="YES"
       useSafeAreas="YES" colorMatched="YES" initialViewController="BYZ-38-t0r">
3      <device id="retina4_7" orientation="portrait">
4          <adaptation id="fullscreen"/>
5      </device>
6      <dependencies>
7          <deployment identifier="iOS"/>
8          <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="14460.20"/>
9          <capability name="Safe area layout guides" minToolsVersion="9.0"/>
10         <capability name="documents saved in the Xcode 8 format" minToolsVersion="8.0"/>
11     </dependencies>
12     <scenes>
13         <!--View Controller-->
14         <scene sceneID="tne-QT-ifu">
15             <objects>
16                 <viewController id="BYZ-38-t0r" customClass="ViewController" customModule="HelloWorld"
                       customModuleProvider="target" sceneMemberID="viewController">
17                     <view key="view" contentMode="scaleToFill" id="8bC-Xf-vdC">
18                         <rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
19                         <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
20                         <color key="backgroundColor" red="1" green="1" blue="1" alpha="1" colorSpace="custom"
                               customColorSpace="sRGB"/>
21                         <viewLayoutGuide key="safeArea" id="6Tk-OE-BBY"/>
22                     </view>
23                 </viewController>
24                 <placeholder placeholderIdentifier="IBFirstResponder" id="dkx-z0-nzr" sceneMemberID="firstResponder"/>
25             </objects>
26         </scene>
27     </scenes>
28 </document>
29
```



LaunchScreen.storyboard:

Remember when you open an app for the first time or update an app and open it, your UI is a little bit different? This is called a walkthrough screen or an on-boarding screen. It gives you a general idea of how the UI is designed via a walkthrough. Most apps have very nice LaunchScreens these days. This is where you want to leave a good impression to your users.

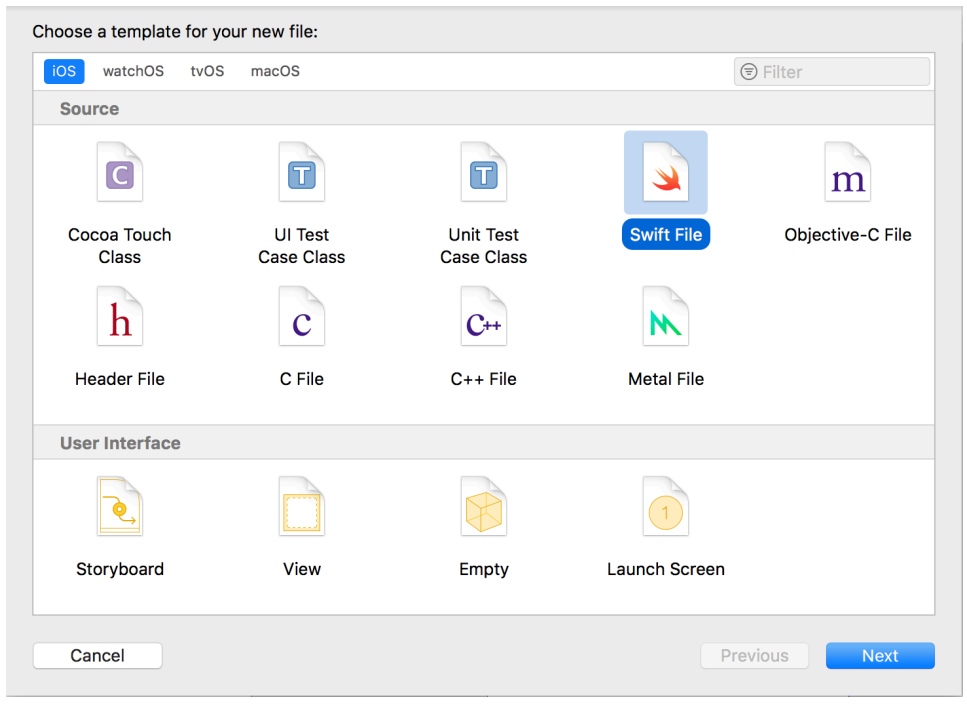


Here is an example of an app walkthrough. We will also build our own walkthrough in later chapters.

III. SWIFT FILES

In order to control the storyboards and provide them with useful and accurate information, we need some extra code to represent and control the data. By choosing **Single View App**, Xcode automatically generates a `ViewController.swift`. As its name suggests, this controls the view defined in our storyboards. We can also define our own Swift files by going to **File>New>File** or use **cmd+N**. Choose Swift File or Cocoa Touch Class to get a new Swift file.

In `AppDelegate.swift`, we define the methods of UIResponder and UIApplication-Delegate to make adjustments when the state of our app is changed. You will learn more about this in later chapters for Application Lifecycle.



IV. OTHER SUPPORTING FILES

Info.plist:

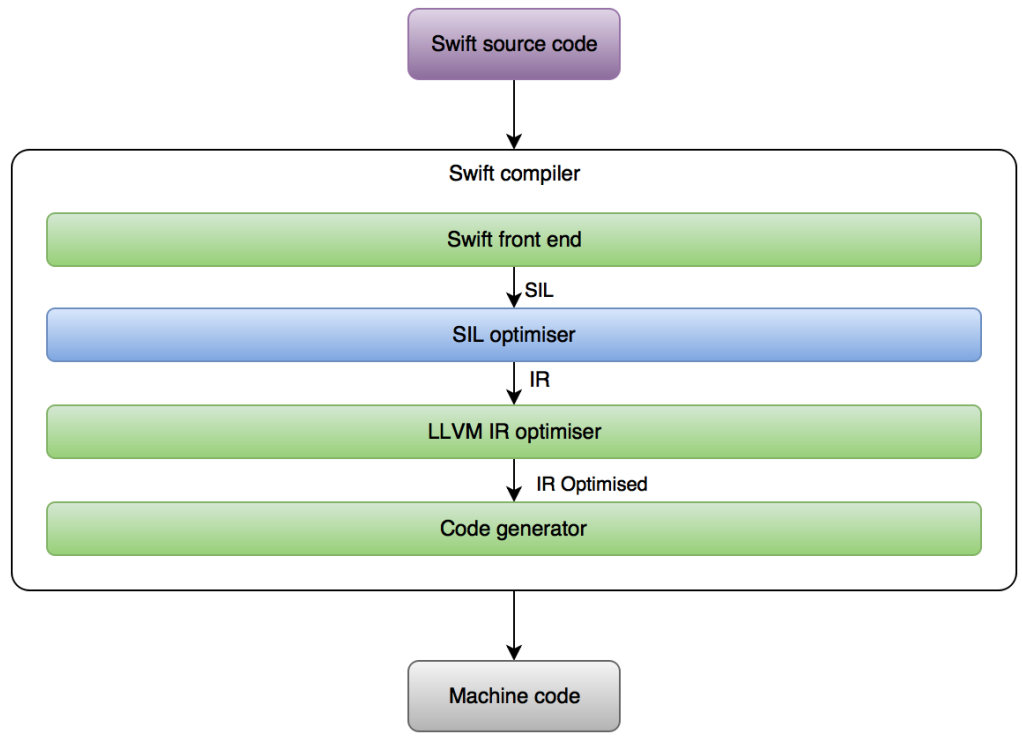
A property list is a dictionary of keys and values that can be stored in your file system with a `.plist` file extension. They are used as a lightweight and portable means to store a small amounts of data. Every application is expected to have an Info.plist file so the system can read it and work according to it's contained values. If you don't completely understand this, don't worry. We don't need to work on this file in this chapter and I will explain how it works in more detail when we do need to work on it.

Assets.xcassets:

This is the folder for all the supporting files like images, audio files and so on. A strait forward example for the usage of this folder is the asset `AppIcon`. You can add icon images of different sizes for different screen resolutions. You can also store some audio files for some custom sound effects in this folder.

Products:

When you clicked run to test your app in a simulator, did you notice that in the status bar, Xcode first compiles your project, then runs on a device or a simulator. At the end of the last chapter, I asked if we need different codes for different devices. The answer is obviously no because the code that we write don't actually run on a device. It is compiled first and packed in to a product. What's actually running is only the product. During compilation, we can generate different products with the same source code for different software environments (iOS 10, 11, 12) and different devices(iPhone, iPad).



EXERCISE #1

XML stands for eXtensible Markup Language. If you are familiar with web development and HTML, XML shouldn't be new to you. Just like you define the content and the layout of a web page, you can also organize the storyboards in a very similar way.

untitled.html

```
<!DOCTYPE html>
<html>
<head>
    <title>My Page</title>
</head>
<body>
    <H1>My Study Plan</H1>
    <br>

    <p>Things to learn to build an iOS app</p>
    <ul style="list-style-type:disc;">
        <li>Swift</li>
        <li>Xcode</li>
        <li>Objective C</li>
    </ul>
</body>
</html>
```

file:///Users/Carlistle/Desktop/untitled.html

My Study Plan

Things to learn to build an iOS app

- Swift
- Xcode
- Objective C

In `untitled.html`, we define DOM elements in XML and we can visualize them in our browser like Safari. Now try to analyze `Main.storyboard` in its XML form and see if you can understand in a lower level how the storyboards are implemented for Xcode.

```
<?xml version="1.0" encoding="UTF-8"?>
<document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0" toolsVersion="14460.31"
    targetRuntime="iOS.CocoaTouch" propertyAccessControl="none" useAutolayout="YES" useTraitCollections="YES"
    useSafeAreas="YES" colorMatched="YES" initialViewController="BYZ-38-t0r">
    <device id="retina4_7" orientation="portrait">
        <adaptation id="fullscreen"/>
    </device>
    <dependencies>
        <deployment identifier="iOS"/>
        <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="14460.20"/>
        <capability name="Safe area layout guides" minToolsVersion="9.0"/>
        <capability name="documents saved in the Xcode 8 format" minToolsVersion="8.0"/>
    </dependencies>
    <scenes>
        <!--View Controller-->
        <scene sceneID="tne-QT-ifu">
            <objects>
                <viewController id="BYZ-38-t0r" customClass="ViewController" customModule="HelloWorld"
                    customModuleProvider="target" sceneMemberID="viewController">
                    <view key="view" contentMode="scaleToFill" id="8bC-Xf-vdC">
                        <rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
                        <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
                        <color key="backgroundColor" red="1" green="1" blue="1" alpha="1" colorSpace="custom"
                            customColorSpace="sRGB"/>
                        <viewLayoutGuide key="safeArea" id="6Tk-OE-BBY"/>
                    </view>
                </viewController>
                <placeholder placeholderIdentifier="IBFirstResponder" id="dkx-z0-nzr" sceneMemberID="firstResponder"/>
            </objects>
        </scene>
    </scenes>
</document>

```

EXERCISE #2

Go to `Assets.xcassets` and select AppIcon. Try to add your own image file for this HelloWorld app. Test it out in different devices, and think about why do we need many different sizes of our icon