

# Mobile Applications Development 2 (MAD2)

Diploma in IT

Teaching Team:  
Mr Ng Poh Oon  
Mr Charles Keck

MAD2 Oct 2019

# Chapter 6

## Alert View and CocoaPods

Objectives

Dependency Manager  
CocoaPods

Alert and  
Indicator

Summary

Discussion 1

# Objectives

To be able to understand:

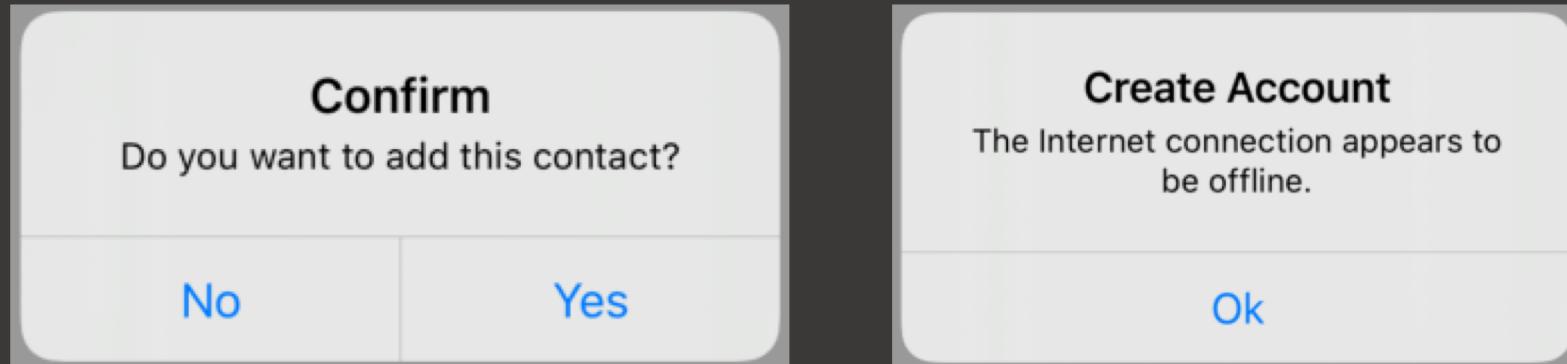
- Alert View with UIAlertController
  - Alert style and ActionSheet style
- Indicator with UIActivityIndicatorView view
- Dependancy Manager, Libraries & Frameworks : CocoaPods



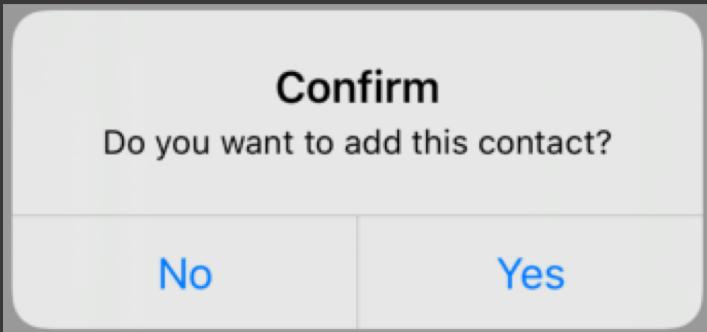
# Alert View

# Alert View with UIAlertController

- Used to **display an alert message** to the user
- Helps to provide good user experience
  - Visual feedback
  - Confirmation for user actions
  - Data validation message
- Configurable buttons



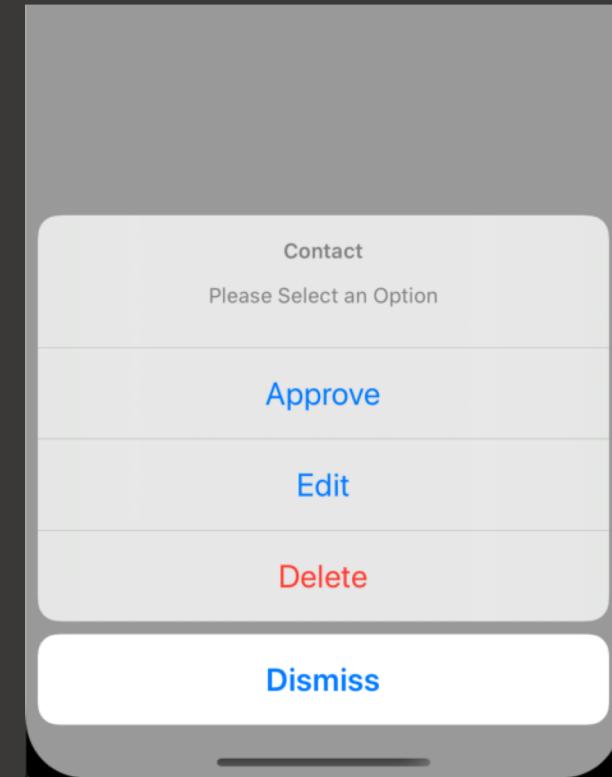
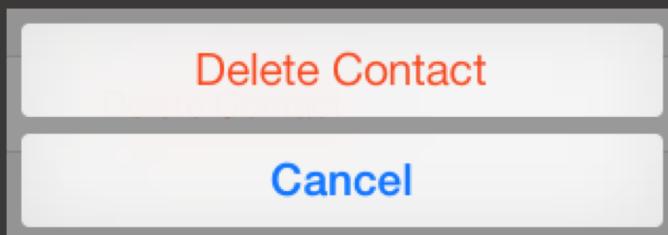
# Creating Alert View



```
@IBAction func showAlert(_ sender: Any) {  
  
    let alertView = UIAlertController(title: "Confirm",  
                                    message: "Do you want to add this contact?",  
                                    preferredStyle: UIAlertController.Style.alert)  
  
    alertView.addAction(UIAlertAction(title: "No",  
                                    style: UIAlertAction.Style.default,  
                                    handler: { _ in //to be discussed  
}))  
    alertView.addAction(UIAlertAction(title: "Yes",  
                                    style: UIAlertAction.Style.default,  
                                    handler: { _ in //to be discussed  
}))  
    self.present(alertView, animated: true, completion: nil)  
}
```

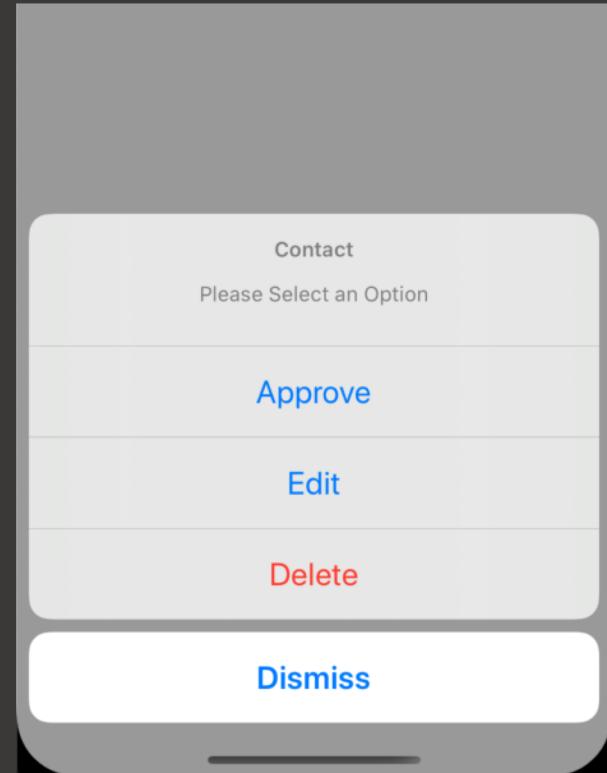
# ActionSheet with UIAlertController

- Like the Alert view, this is a dialog to present the user with a set of alternatives for how to proceed with a given task
- Contains an optional title and one or more buttons



# Creating ActionSheet

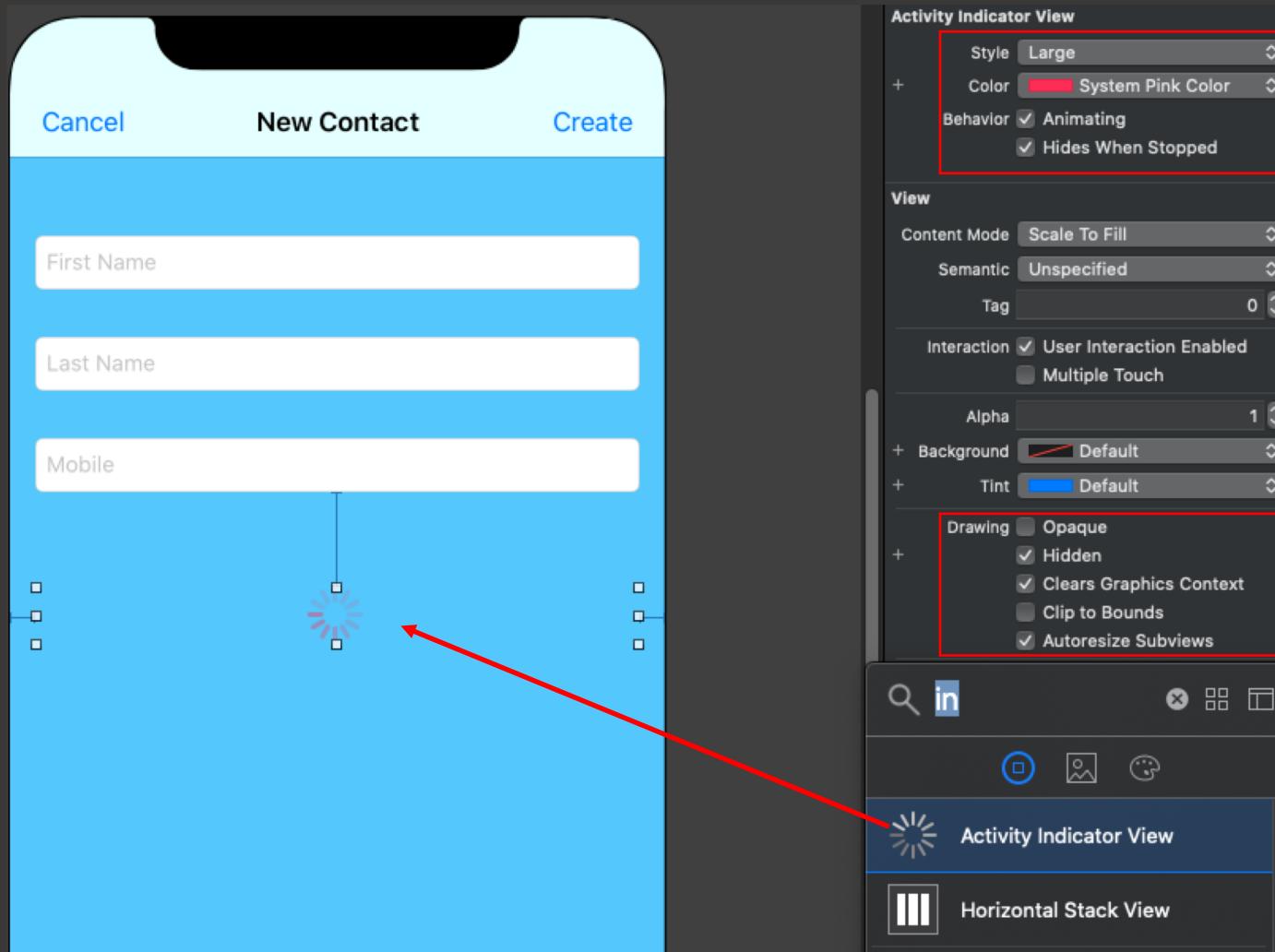
```
@IBAction func showAction(_ sender: Any) {
    let alertAction = UIAlertController(title: "Contact",
                                        message: "Please Select an Option",
                                        preferredStyle: .actionSheet)
    alertAction.addAction(UIAlertAction(title: "Approve",
                                       style: .default,
                                       handler: { (_) in print("User click Approve button")
                                       }))
    alertAction.addAction(UIAlertAction(title: "Edit",
                                       style: .default,
                                       handler: { (_) in print("User click Edit button")
                                       }))
    alertAction.addAction(UIAlertAction(title: "Delete",
                                       style: .destructive,
                                       handler: { (_) in print("User click Delete button")
                                       }))
    alertAction.addAction(UIAlertAction(title: "Dismiss",
                                       style: .cancel,
                                       handler: { (_) in print("User click Dismiss button")
                                       }))
    self.present(alertAction, animated: true, completion: {
        print("completion block")
    })
}
```



# UIActivityIndicator

- Visual feedback to inform user to “Please wait”
- Tells user something is happening and that the app is still running
- Creates a positive user experience

# UIActivityIndicatorView



MAD2 Oct 2019

# UIActivityIndicatorView

```
@IBOutlet weak var activityIndicator: UIActivityIndicatorView!

@IBOutlet func onClicked(_ sender: Any) {
    activityIndicator.isHidden = false
    activityIndicator.startAnimating()

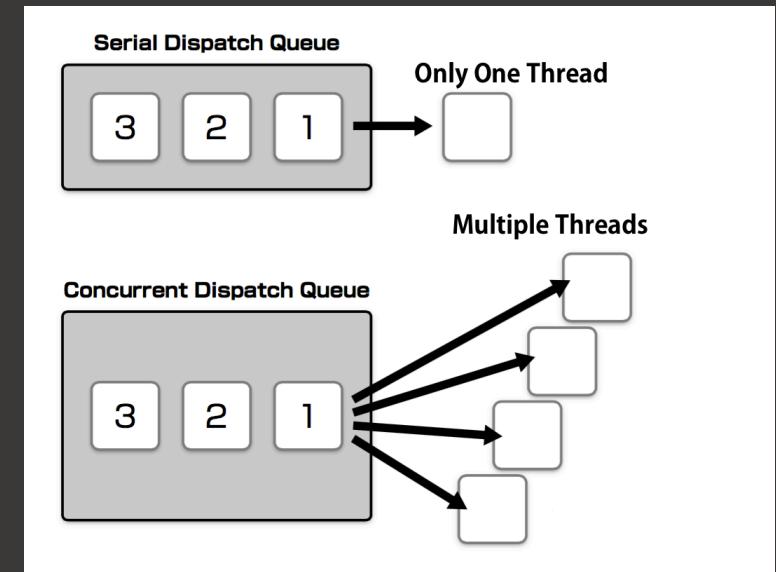
    DispatchQueue.global(qos: .userInitiated).async {
        self.testThread()
    }
}

func testThread(){
    for i in 1...3 {
        Thread.sleep(forTimeInterval: 3.0) //Sleep for 3 seconds
        print("\(i)")
    }
    DispatchQueue.main.async {
        self.onCompletion()
    }
}

func onCompletion(){
    activityIndicator.stopAnimating()
    activityIndicator.isHidden = true
}
```

# DispatchQueue

- Asynchronous operation (not in syllabus)
  - Managing the execution of tasks serially or concurrently on main thread or on background thread



<https://developer.apple.com/documentation/dispatch/dispatchqueue>

<https://medium.com/@michaellong/how-to-chain-api-calls-using-swift-5s-new-result-type-and-gcd-56025b51033c>

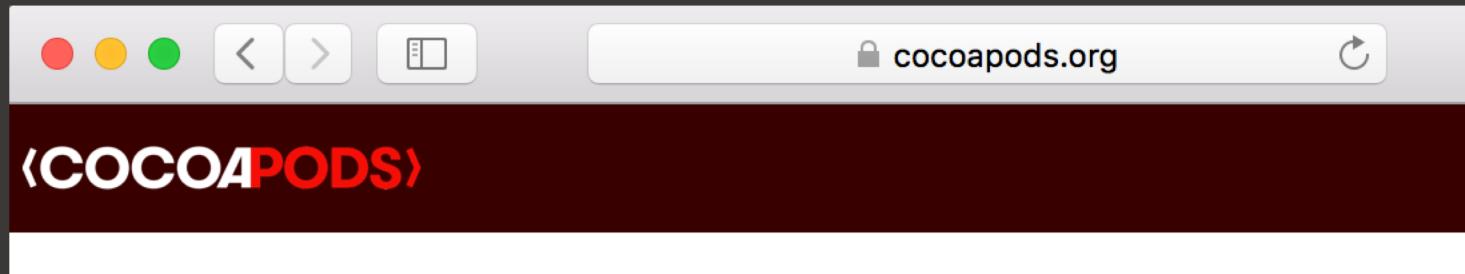


# Libraries & Frameworks

## CocoaPods

# What is CocoaPods?

- CocoaPods is a **dependency manager** for Swift and Objective-C Cocoa projects.
- It has over **53 thousand libraries** and is used in over 3 million apps.



# Setting up

- <https://www.youtube.com/watch?v=iEAjvNRdZa0>



# Search

The screenshot shows a web browser window on a Mac OS X desktop. The URL in the address bar is `cocoapods.org`. The search query is `QRCode`, which has resulted in 20 Swift-only pods. A red box highlights the first result, `QRCodeReader.swift`, version 7.2.0, maintained by Yannick Loriot. The pod's documentation includes a red box containing the command `pod 'QRCodeReader.swift', '~> 7.2'`.

For your Podfile

```
pod 'QRCodeReader.swift', '~> 7.2'
```

**QRCodeReader.swift**  
7.2.0

By Yannick Loriot [yannickloriot](#)

[yannickl/QRCodeReader.swift](#)

OVERVIEW CHANGELOG

Documented ✓  
Tested ✗  
Language Swift  
License MIT  
Last Release Oct 2016  
Supports SPM ✓

Maintained by [Yannick Loriot](#).

Downloads

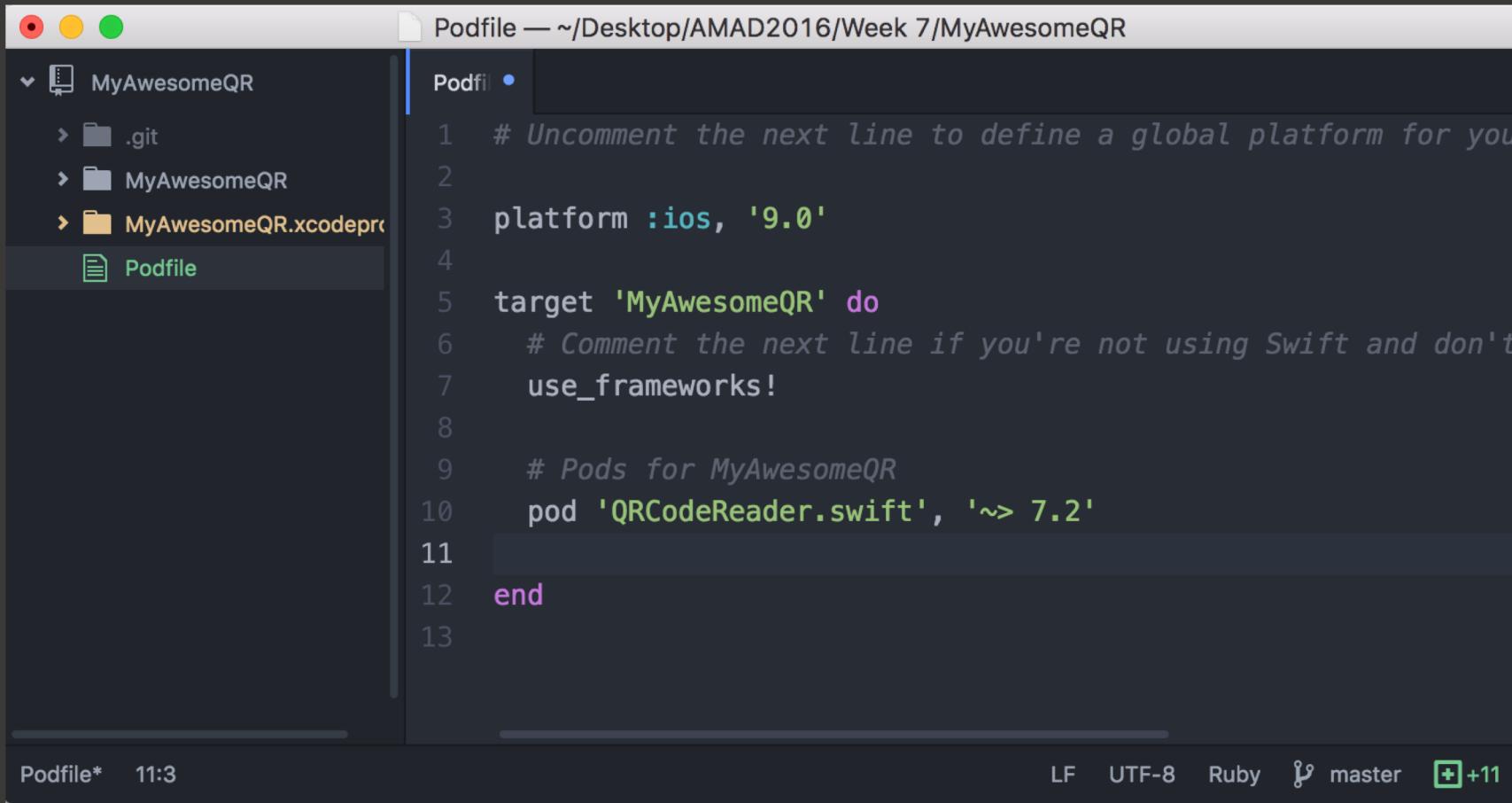
# Steps

- Use of “Ruby” GEM ← A dependency manager

```
~> cd /usr/local/bin  
  
~> sudo gem update --system  
  
~> sudo gem install -n /usr/local/bin cocoapods  
  
~> pod setup
```

```
MyProject$ pod init
```

# Atom or any text editor



The screenshot shows the Atom text editor interface with a dark theme. The title bar reads "Podfile — ~/Desktop/AMAD2016/Week 7/MyAwesomeQR". The left sidebar shows the project structure: "MyAwesomeQR" folder containing ".git", "MyAwesomeQR", "MyAwesomeQR.xcodeproj", and "Podfile". The main editor area displays the following code:

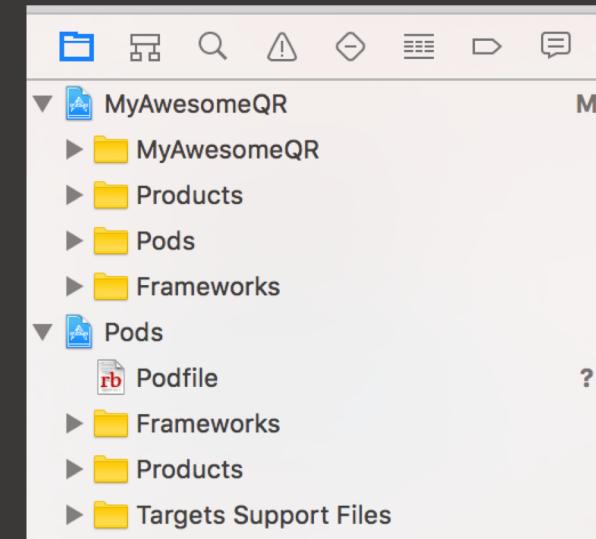
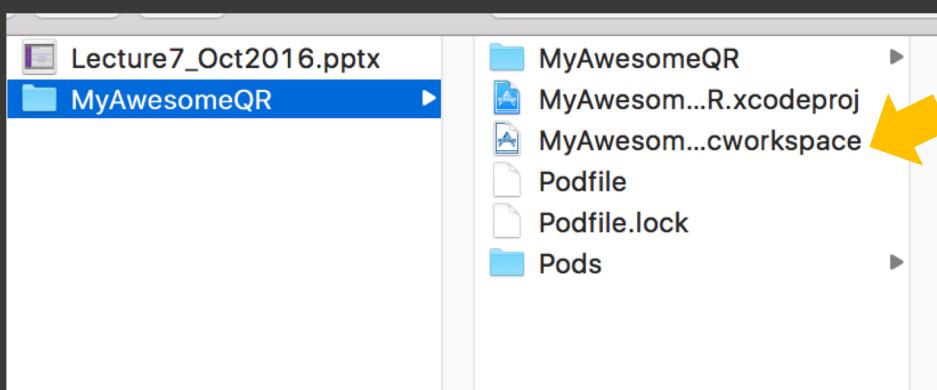
```
Podfile •
1 # Uncomment the next line to define a global platform for your
2 # entire repository
3 platform :ios, '9.0'
4
5 target 'MyAwesomeQR' do
6   # Comment the next line if you're not using Swift and don't
7   # use_frameworks!
8
9   # Pods for MyAwesomeQR
10 pod 'QRCodeReader.swift', '~> 7.2'
11
12 end
13
```

The status bar at the bottom shows "Podfile\*" 11:3, LF, UTF-8, Ruby, master, and a +11 icon.

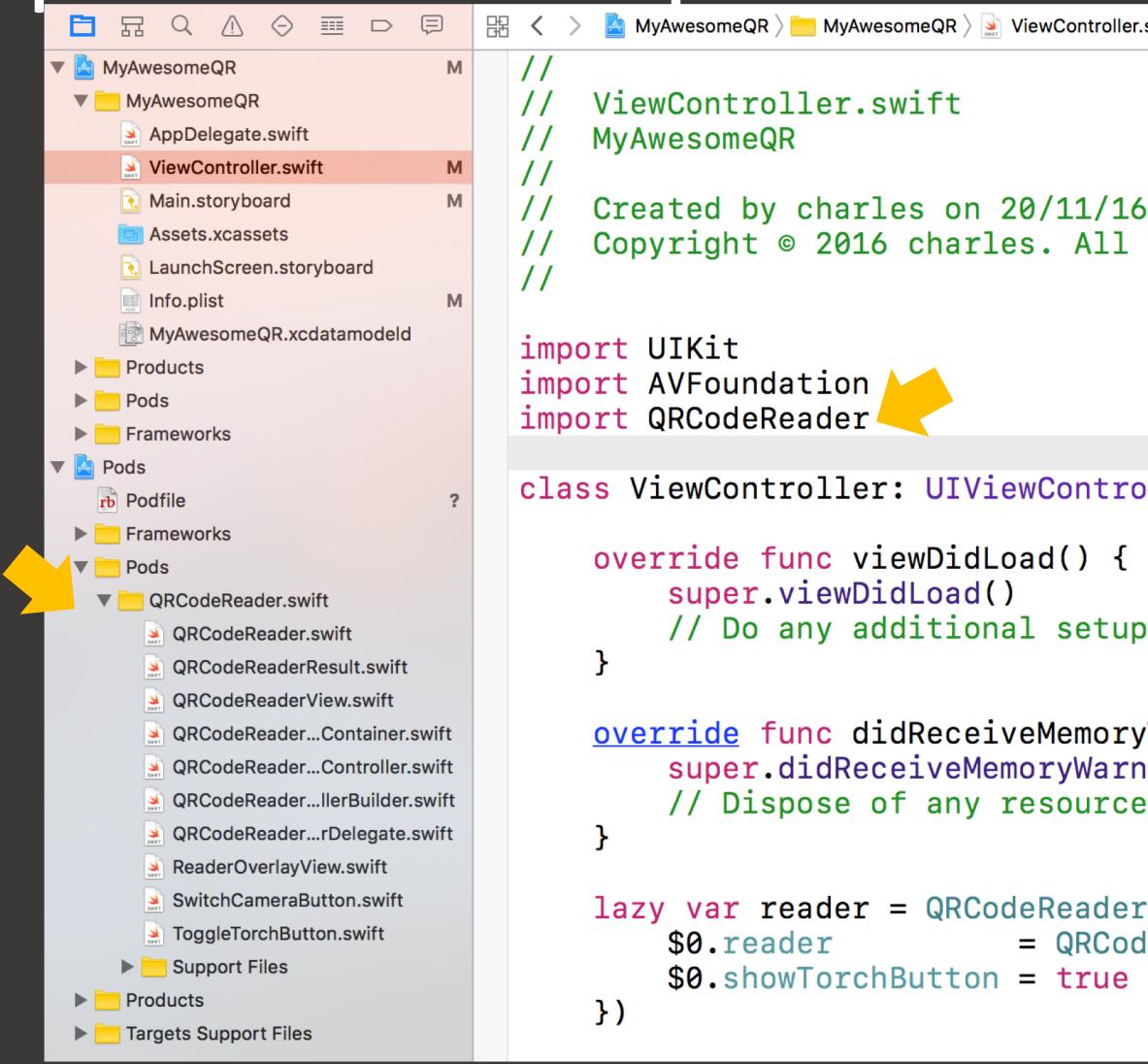
# Steps

```
MyProject$ pod install
```

```
Analyzing dependencies  
Downloading dependencies  
Generating Pods project
```



# Opening up the xcworkspace



The screenshot shows the Xcode interface with the project 'MyAwesomeQR' open. The left sidebar displays the file structure:

- MyAwesomeQR (target)
- MyAwesomeQR (group)
  - AppDelegate.swift
  - ViewController.swift (selected)
  - Main.storyboard
  - Assets.xcassets
  - LaunchScreen.storyboard
  - Info.plist
  - MyAwesomeQR.xcdatamodeld
- Products
- Pods
  - Podfile
  - Frameworks
  - Pods (group)
    - QRCodeReader.swift (selected)
    - QRCodeReader.swift
    - QRCodeReaderResult.swift
    - QRCodeReaderView.swift
    - QRCodeReader...Container.swift
    - QRCodeReader...Controller.swift
    - QRCodeReader...llerBuilder.swift
    - QRCodeReader...rDelegate.swift
    - ReaderOverlayView.swift
    - SwitchCameraButton.swift
    - ToggleTorchButton.swift
  - Support Files
- Products
- Targets Support Files

MAD2 Oct 2019

# Alternative Dependency Manager

- Carthage
  - Carthage is another simple dependency manager for the Cocoa application. It downloads and build the dependencies **but will not change Project file Or Xcode project** build setting like CocoaPods. We have to manually drag '.framework' binaries to "Linked Frameworks and Libraries".

```
$ brew install carthage
```

*<https://github.com/Carthage/Carthage>*

# Carthage

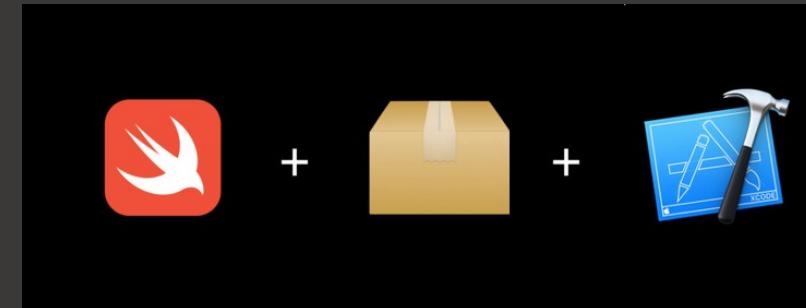
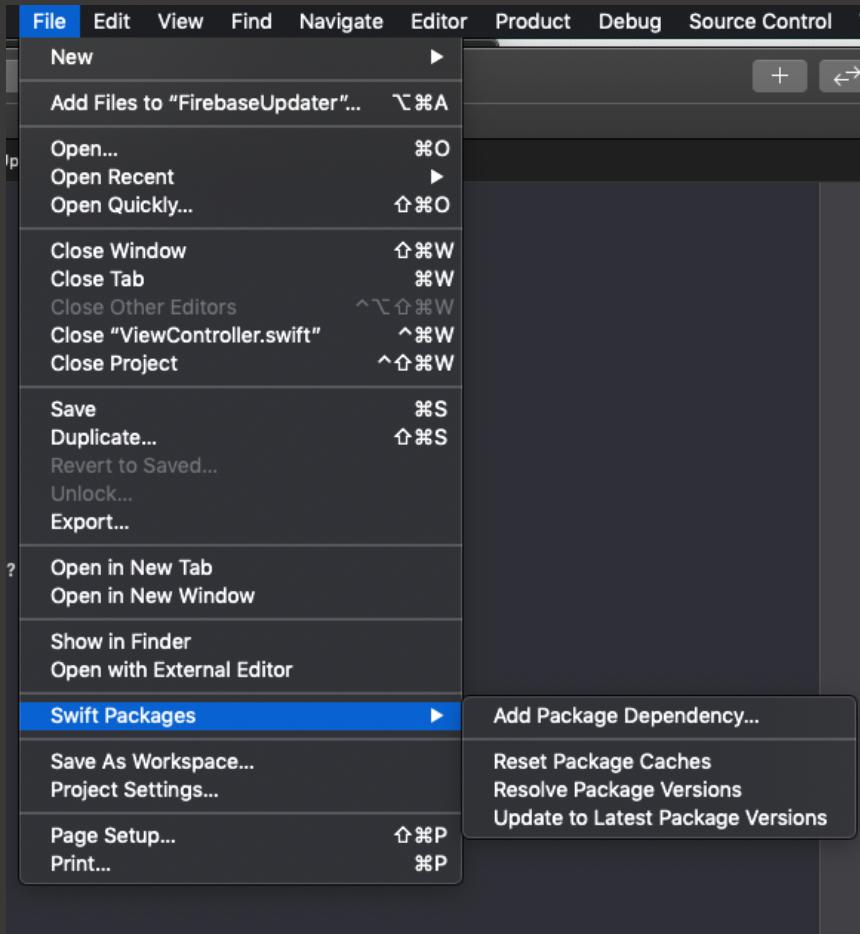
- Pros
  - Carthage won't touch your Xcode settings or project files. It just download and build the dependencies so you have proper control on what you are doing.
  - Decentralised
  - Supports submodules
- Cons
  - Unstable and Slow
  - Small Community, not many contributors
  - Lot of manual steps to perform on Xcode to get everything setup

# Alternative Dependency Manager

- Swift Package Manager
  - The Swift Package Manager is a tool for managing the distribution of Swift code. It's integrated with the Swift build system to automate the process of downloading, compiling, and linking dependencies.
  - It's Official Package Manager for Swift
- Swift Package Manager is
  - Command Line Based tool (before Xcode 11)
  - Being Cross-Platform, Swift Package Manager doesn't need Xcode to create package
  - It's decentralised
  - Swift Package Manager is open-source and source code is available on Github

```
$ swift build -- version  
Apple Swift Package Manager – Swift 4.2.0 (swiftpm-14460.2)
```

# Xcode 11 Swift Package Manager



<https://wwdcbysundell.com/2019/xcode-swiftpm-first-look/>

MAD2 Oct 2019

# Summary

Understand Alert View with UIAlertController

Understand the Dependency Manager concepts, Libraries & Frameworks : CocoaPods