iOS 工程師面試

SwiftUI

1. 什麼是 SwiftUI 以及它與 UIKit 的不同之處？
    - SwiftUI 是 Apple 的現代框架，用於構建用戶界面，提供宣告式語法，與 UIKit 的命令式方法相比，簡化了 UI 的創建和更新。
2. 解釋 SwiftUI 中的宣告式 UI 概念。
    - 宣告式 UI 描述了所需的結果，而不是實現它的步驟。SwiftUI 根據宣告的狀態構建和更新 UI。
3. 如何在 SwiftUI 中創建自定義視圖？
    - 創建一個新的結構體，遵循 `View` 協議，並在 `body` 屬性中定義其內容。
4. 使用 SwiftUI 相對於 UIKit 的好處有哪些？
    - 好處包括宣告式語法、更簡單的狀態管理，以及統一的界面，適用於 macOS、iOS 和其他 Apple 平台。
5. 如何在 SwiftUI 中處理狀態管理？
    - 使用 `@State` 進行本地狀態管理，使用 `@ObservedObject` 進行可觀察類，使用 `@EnvironmentObject` 進行全局狀態。
6. 解釋 `@State` 和 `@Binding` 的區別。
    - `@State` 用於本地狀態管理，而 `@Binding` 用於在視圖之間共享狀態。
7. 如何在 SwiftUI 中使用 `@EnvironmentObject`？
    - `@EnvironmentObject` 用於訪問通過視圖層次結構傳遞的對象。
8. `@ObservedObject` 和 `@StateObject` 的目的是什麼？
    - `@ObservedObject` 觀察對象的變化，而 `@StateObject` 管理對象的生命週期。
9. 如何在 SwiftUI 中處理視圖動畫？
    - 使用動畫修飾符，如 `.animation()` 或 `withAnimation {}` 來動畫化 UI 變化。
10. `ViewBuilder` 和 `@ViewBuilder` 的區別是什麼？
    - `ViewBuilder` 是用於構建視圖的協議，而 `@ViewBuilder` 是用於返回視圖的函數屬性包裝器。

CocoaPods 和依賴項

11. 什麼是 CocoaPods 以及它在 iOS 開發中的用途？
    - CocoaPods 是 Swift 和 Objective-C Cocoa 專案的依賴管理器，簡化了庫的集成。
12. 如何安裝 CocoaPods？
    - 通過 Ruby gem 安裝：`sudo gem install cocoapods`。

13. 什麼是 Podfile 以及如何配置它？
    - Podfile 列出專案依賴項。通過指定 pods 和它們的版本來配置。
14. 如何使用 CocoaPods 將依賴項添加到專案中？
    - 將 pod 添加到 Podfile 並運行 `pod install`。
15. `pod install` 和 `pod update` 的區別是什麼？
    - `pod install` 按指定安裝依賴項，而 `pod update` 更新到最新版本。
16. 如何解決不同 pods 之間的衝突？
    - 使用兼容的 pod 版本或在 Podfile 中指定版本。
17. 什麼是 Carthage 以及它與 CocoaPods 的不同之處？
    - Carthage 是另一個依賴管理器，它構建和鏈接庫而不深度集成到專案中。
18. 如何管理不同構建配置的不同 pods？
    - 在 Podfile 中根據構建配置使用條件語句。
19. 什麼是 podspec 文件以及它的用途？
    - podspec 文件描述了 pod 的版本、源、依賴項和其他元數據。
20. 如何排除 CocoaPods 的問題？
    - 檢查 pod 版本，清理專案，並參考 CocoaPods 問題追蹤器。

UI 布局

21. 如何在 iOS 中創建響應式佈局？
    - 使用 Auto Layout 和約束使視圖適應不同的屏幕尺寸。
22. 解釋 `Stack View` 和 `Auto Layout` 的區別。
    - Stack Views 簡化了在行或列中佈局視圖，而 Auto Layout 提供了對位置的精確控制。
23. 如何在 iOS 中使用 `UIStackView`？
    - 將視圖添加到 Stack View 並配置其軸、分佈和對齊。
24. `frame` 和 `bounds` 在 iOS 中的區別是什麼？
    - `frame` 定義了視圖相對於其父視圖的位置和大小，而 `bounds` 定義了視圖自己的坐標系統。
25. 如何處理 iOS 中不同屏幕尺寸和方向？
    - 使用 Auto Layout 和尺寸類來適應各種設備和方向的 UI。
26. 解釋如何在 iOS 中使用 `Auto Layout` 約束。
    - 設置視圖之間的約束來定義它們的關係和位置。

27. `leading` 和 `trailing` 在 Auto Layout 中的區別是什麼？
    - leading 和 trailing 適應文本方向，而 left 和 right 不適應。
28. 如何在 iOS 中創建自定義佈局？
    - 子類化 `UIView` 並覆寫 `layoutSubviews()` 以手動定位子視圖。
29. 解釋如何使用 `UIPinchGestureRecognizer` 和 `UIRotationGestureRecognizer`。
    - 將手勢識別器附加到視圖並在委派方法中處理它們的操作。
30. 如何處理不同設備類型（iPhone、iPad）的佈局變化？
    - 使用尺寸類和適應性佈局來調整不同設備的 UI。

Swift

31. Swift 和 Objective-C 的主要區別是什麼？
    - Swift 更安全、更簡潔，並支持現代語言功能，如閉包和泛型。
32. 解釋 Swift 中的可選值概念。
    - 可選值表示可以是 `nil` 的值，表示值的缺失。
33. `nil` 和 `optional` 的區別是什麼？
    - `nil` 是值的缺失，而可選值可以保存值或是 `nil`。
34. 如何在 Swift 中處理錯誤？
    - 使用 `do-catch` 塊或使用 `throw` 來傳播錯誤。
35. 解釋 `let` 和 `var` 的區別。
    - `let` 宣告常量，而 `var` 宣告可以修改的變量。
36. 類和結構體在 Swift 中的區別是什麼？
    - 類支持繼承並是引用類型，而結構體是值類型。
37. 如何在 Swift 中創建一個枚舉？
    - 使用 `enum` 關鍵字定義枚舉及其情況，這些情況可以有關聯值。
38. 解釋 Swift 中的協議導向編程概念。
    - 協議定義方法、屬性和要求，遵循類型必須實現。
39. 協議和委派之間的區別是什麼？
    - 協議定義方法，而委派實現特定交互的協議方法。
40. 如何在 Swift 中使用泛型？
    - 使用泛型類型來編寫靈活、可重用的代碼，適用於任何數據類型。

網絡

41. 如何在 iOS 中處理網絡請求？
    - 使用 URLSession 進行網絡任務，或使用 Alamofire 等庫進行更高層次的抽象。
42. 什麼是 URLSession？
    - URLSession 處理網絡請求，提供數據任務、上傳任務和下載任務。
43. 如何在 Swift 中處理 JSON 解析？
    - 使用 `Codable` 協議將 JSON 數據解碼為 Swift 結構體或類。
44. 解釋同步和異步請求的區別。
    - 同步請求阻塞調用線程，而異步請求不會。
45. 如何在後台線程中管理網絡請求？
    - 使用 GCD 或 OperationQueue 在主線程之外執行請求。
46. 什麼是 Alamofire 以及它與 URLSession 的不同之處？
    - Alamofire 是一個第三方網絡庫，簡化了 HTTP 請求，與 URLSession 相比。
47. 如何處理網絡錯誤和重試？
    - 在完成處理程序中實現錯誤處理，並考慮對暫時錯誤的重試機制。
48. 解釋如何使用 `URLSessionDataDelegate` 方法。
    - 實現委派方法來處理請求進度、身份驗證等。
49. GET 和 POST 請求的區別是什麼？
    - GET 檢索數據，而 POST 將數據發送到伺服器以創建或更新資源。
50. 如何保護網絡通信？
    - 使用 HTTPS 加密傳輸中的數據並正確處理證書。

最佳實踐和問題解決

51. 如何確保專案中的代碼質量？
    - 使用檢查工具、編寫單元測試，並遵循編碼標準。
52. 解釋如何調試 SwiftUI 視圖。
    - 使用 Xcode 的調試工具、預覽畫布和打印語句來識別問題。
53. 你用來優化應用程序性能的策略有哪些？
    - 使用 Instruments 進行應用程序分析，優化數據獲取，並減少 UI 層數。
54. 如何在 Swift 中處理內存管理？

- 使用 ARC（自動引用計數）並避免保留循環。

55. 解釋如何處理舊代碼的重構。
    - 識別代碼異味，編寫測試，並逐步重構。
56. 你對 CI/CD 管道的經驗是什麼？
    - 使用 Jenkins、GitHub Actions 或 Fastlane 設置管道，進行自動構建和部署。
57. 你如何保持最新的 iOS 開發？
    - 關注 Apple 的開發者資源，參加會議，並參與開發者社區。
58. 解釋你解決專案中困難錯誤的一次經歷。
    - 描述識別、隔離和修復問題的過程。
59. 你的版本控制方法是什麼？
    - 使用 Git 進行分支、提交和有效協作。
60. 你如何處理專案中的截止日期和壓力？
    - 優先處理任務，有效溝通，並高效管理時間。