Great Developer Habits

Josh Tidsbury, Technology Evangelism
craft
\ˈkraft\ 

1. skill in planning, making, or executing
2. to make or produce with care, skill, or ingenuity
Hidden details matter.
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
Storyboards
Modern
Build Phases
- Target 'Gateway' - Remove Unnecessary Build Phases
  The Gateway target may contain unnecessary build phases, such as xcodebuild or info script files. This will remove those files from the target's build phases.

Build Settings
- Target 'Gateway' - Update iOS Deployment Target
  An iOS Deployment Target greater than 9.0 is not supported by this version of Xcode. This will update the value for Target 'Gateway' to 9.0.
- Target 'Gateway' - Adopt 'PRODUCT_BUNDLE_IDENTIFIER' build setting
  Target 'Gateway' will have its Info.plist's CFBundleIdentifier key set to "$PRODUCT_BUNDLE_IDENTIFIER", and the "PRODUCT_BUNDLE_IDENTIFIER" build setting will be set to.
- Target 'Gateway' - Enable Weak References in Manual Retain Release
  Enabling weak references in manual retain release is recommended. This will set the CLANG_ENABLE_OBJC_WEAK setting to "YES" for the "Release" configuration.
- Target 'Gateway' - Enable Weak References in Manual Retain Release
  Enabling weak references in manual retain release is recommended. This will set the CLANG_ENABLE_OBJC_WEAK setting to "YES" for the "Release" configuration.

Project 'Gateway' - Update iOS Deployment Target
An iOS Deployment Target greater than 9.0 is not supported by this version of Xcode. This will update the value for Project 'Gateway' to 9.0.

Project 'Gateway' - Enable Recommended Warnings
This will enable the following recommended compiler warnings:
- Strict Conformance of Allorability
- Duplicate Method Definitions
- Empty Logic Blocks
- Enable Strict checking of objc_msgSend Calls
- Implicit enum Conversions
- Implicit Constant Conversions
- Implicit Conversion to S20 Bit Type
- Implicit Enum Conversions
- Implicit Integer to Pointer Conversions
- Implicit Non-Literal Null Conversions
Organize
Organize

Functional organization with groups
Organize

Functional organization with groups

Mirror project structure and file structure
Organize

Functional organization with groups

Mirror project structure and file structure

Break apart large storyboards
Organize

Functional organization with groups

Mirror project structure and file structure

Break apart large storyboards

Modernize your project file
Organize

Functional organization with groups
Mirror project structure and file structure
Break apart large storyboards
Modernize your project file
Throw away code scraps
Organize

Functional organization with groups
Mirror project structure and file structure
Break apart large storyboards
Modernize your project file
Throw away code scraps
Address the root cause of warnings
Source Control:  □ Create Git repository on my Mac

Xcode will place your project under source control.
Track
Track

Use source control
Track

Use source control

Keep commits small and isolated
Use source control

Keep commits small and isolated

Write useful commit messages
Track

Use source control

Keep commits small and isolated

Write useful commit messages

Utilize branches for bug and feature work
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
“I don’t need comments, my code is self-documenting”
// A constant string id value

let id = "2ADA155F-1529-4D2D-98C4-0E4BD06940E8"
// The permanent identifier for this application when interacting
// with the CMS, provided by the vendor of the CMS.
let id = "2ADA155F-1529-4D2D-98C4-0E4BD06940E8"
// The permanent identifier for this application when interacting
// with the CMS, provided by the vendor of the CMS.
let cmsApplicationIdentifier = "2ADA155F-1529-4D2D-98C4-0E4BD06940E8"
Class

UIViewController

An object that manages a view hierarchy for your UIKit app.

Declaration

class UIViewController : UIResponder

Overview

The UIViewController class defines the shared behavior that is common to all view controllers. You rarely create instances of the UIViewController class directly. Instead, you subclass UIViewController and add the methods and properties needed to manage the view controller’s view hierarchy.

A view controller’s main responsibilities include the following:

- Updating the contents of the views, usually in response to changes to the underlying data.
- Responding to user interactions with views.
- Resizing views and managing the layout of the overall interface.
- Coordinating with other objects—including other view controllers—in your app.

A view controller is tightly bound to the views it manages and takes part in handling events in its view hierarchy. Specifically, view controllers are UIResponder objects and are inserted into the responder chain between the view controller’s root view and that view’s superview, which typically belongs to a different view controller. If none of the view controller’s views handle an event, the view controller has the option of handling the event or passing it along to the superview.

View controllers are rarely used in isolation. Instead, you often use multiple view controllers, each of which owns a portion of your app’s interface. For example, one view controller might display a subset of items while a different view controller displays the selected item from that subset. Only the views from one view controller are visible at a time. A view controller may present a different view controller to display a set of views, such as a list contained for other view controllers’ content and animates views however it wishes.

Open in Developer Documentation
import UIKit

class ViewController: UIViewController {

 Summary
 An object that manages a view hierarchy for your UIView.

 Declaration
 class UIViewController: UIResponder

 Discussion
 The UIViewController class defines the shared behavior that is common to all view
 controllers. You rarely create instances of the UIViewController class directly.
 Instead, you subclass UIViewController and add the methods and properties
 needed to manage the view controller's view hierarchy.

 A view controller's main responsibilities include the following:
 - Updating the contents of the views, usually in response to changes to the
   underlying data.
 - Responding to user interactions with views.
 - Reusing views and managing the layout of the overall interface.
 - Coordinating with other objects—including other view controllers—in your app.

 A view controller is tightly bound to the views it manages and takes part in handling
 events in its view hierarchy. Specifically, view controllers are UIResponder objects and
 are inserted into the responder chain between the view controllers' root view and that
 view's superview, which typically belongs to a different view controller. If none of the
 view controller's views handle an event, the view controller has the option of handling
 the event or passing it along to the superview.

 View controllers are rarely used in isolation. Instead, you often use multiple view
 controllers, each of which has a portion of your app's user interface. For example, one
 view controller might display a table of items while a different view controller displays
 the relevant form from that table. Usually only the table from one view controller is
 visible at a time. A view controller may present a different view controller to display a
 new set of views, or it may act as a container for other view controllers' contents and
 animate views however it wants.
Comments are critical for future understanding
Comments are critical for future understanding

Good comments provide background and reasoning
Comments are critical for future understanding

Good comments provide background and reasoning

Use descriptive variable and constant names
Comments are critical for future understanding

Good comments provide background and reasoning

Use descriptive variable and constant names

Include documentation
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
“And while you’re doing that, you might as well add a unit test to make sure that the round trip between the Struct and the dictionary representation keeps working.”
“And while you’re doing that, you might as well add a unit test to make sure that the round trip between the Struct and the dictionary representation keeps working.”
Unit Tests
Unit Tests

Write unit tests
Unit Tests

Write unit tests

Run unit tests before committing code
Unit Tests

Write unit tests

Run unit tests before committing code

Build a foundation for continuous integration
import UIKit

class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

    @IBAction func buttonPressed(_ sender: UIButton) {
        // Do something when the button is pressed.
    }

    @IBOutlet weak var label: UILabel?

}
import UIKit

class ViewController: UIViewController {

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.
}

// And in your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:

// In your storyboard file (if you are using it)

// In your code:
Analyze
Analyze

Simulate poor networks with Network Link Conditioner
Analyze

Simulate poor networks with Network Link Conditioner

Use sanitizers and checkers
Analyze

Simulate poor networks with Network Link Conditioner

Use sanitizers and checkers

Measure performance and efficiency with Debug Gauges
Analyze

Simulate poor networks with Network Link Conditioner
Use sanitizers and checkers
Measure performance and efficiency with Debug Gauges
Investigate issues with Instruments
Understand each change
Build it
Run tests
Proofread
let colour = ...
let colour = ...
let colou = ...
let colo = ...
let color = ...
Evaluate
Evaluate

Include code review as part of your practice
Include code review as part of your practice

Understand each line
Evaluate

Include code review as part of your practice

Understand each line

Build it
Evaluate

Include code review as part of your practice

Understand each line

Build it

Run tests
Evaluate

Include code review as part of your practice

Understand each line

Build it

Run tests

Proofread for style, spelling, and syntax
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
```swift
public func setFavorite(_ status: Bool, for identifier: Activity.Identifier, notify: Bool = true, syncToCloudKit: Bool = true, lastModified: Date = Date()) {
    var favorite = favoritesByIdentifier[identifier]
    var needSave = false

    // If we don't have an entry for this identifier then create one now.
    if favorite == nil {
        favorite = FavoriteEntry(activityIdentifier: identifier, isFavorite: status)
        needSave = true
    } else if favorite!.isFavorite != status {
        // Otherwise, if the status is changing, then update the timestamp and record the new status.
        favorite!.isFavorite = status
        needSave = true
    }

    // Update the last modified date, if needed.
    if favorite!.lastModified != lastModified {
        favorite!.lastModified = lastModified
        needSave = true
    }

    // If something changed, then store the updated favorite and kick off any related work, if needed.
    if needSave { ...
```

/// Sets whether the specified `identifier` is a favorite or not. This function will kick off
/// any necessary work to save the information to disk, post notifications, or initiate a
/// sync with CloudKit asynchronously.

/// Note: The `notify` and `syncToCloudKit` options will *always* cause the asynchronous
/// work to post a notification and/or sync to CloudKit regardless of whether the state of
/// the favorite is actually changing.

/// Parameters:
/// - status: whether the item is a favorite or not
/// - identifier: the identifier of the item
/// - notify: if true, will post the `WwCFavoritesDidChange` notification
/// - syncToCloudKit: if true, will initiate an async operation to push the state to CloudKit
/// - lastModified: the date to record as the last modified date. Defaults to the current date.

/// This is useful when syncing from CloudKit and we want to record a date that matches the server.
Decouple
Decouple

Determine functional segments and break them out
Decouple

Determine functional segments and break them out

Scale your work across multiple apps
Decouple

Determine functional segments and break them out

Scale your work across multiple apps

Improve efficiency with extensions
Decouple

Determine functional segments and break them out

Scale your work across multiple apps

Improve efficiency with extensions

Share your efforts with the broader community
Decouple

Determine functional segments and break them out

Scale your work across multiple apps

Improve efficiency with extensions

Share your efforts with the broader community

Documentation is critical
Have a plan.
Manage
Manage

Use community and open source projects responsibly
Manage

Use community and open source projects responsibly
Understand dependencies thoroughly
Manage

Use community and open source projects responsibly

Understand dependencies thoroughly

Ensure that privacy is respected
Manage

Use community and open source projects responsibly

Understand dependencies thoroughly

Ensure that privacy is respected

Have a plan if a dependency goes away or is no longer maintained
The Last 10%
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
Organize
Track
Document
Test
Analyze
Evaluate
Decouple
Manage
Simply beautiful app

This is the most elegant and attention to detail.
More Information

developer.apple.com/wwdc19/239