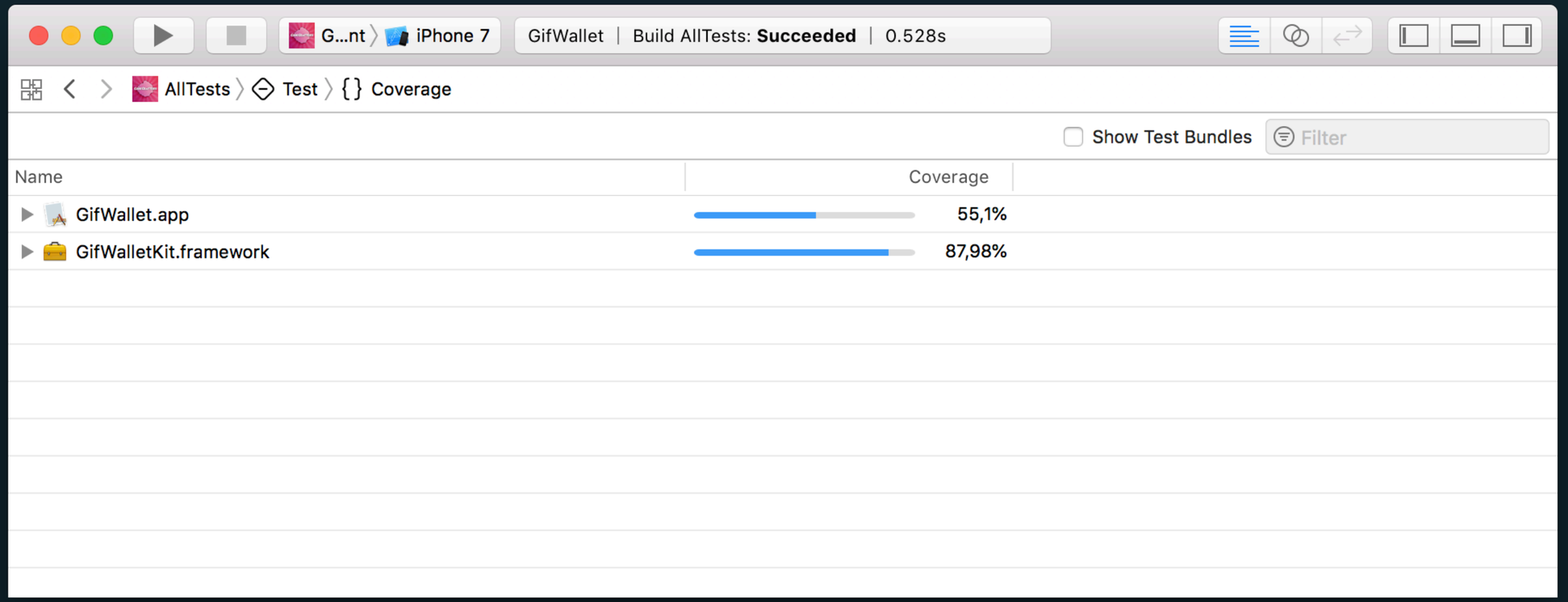
UITests

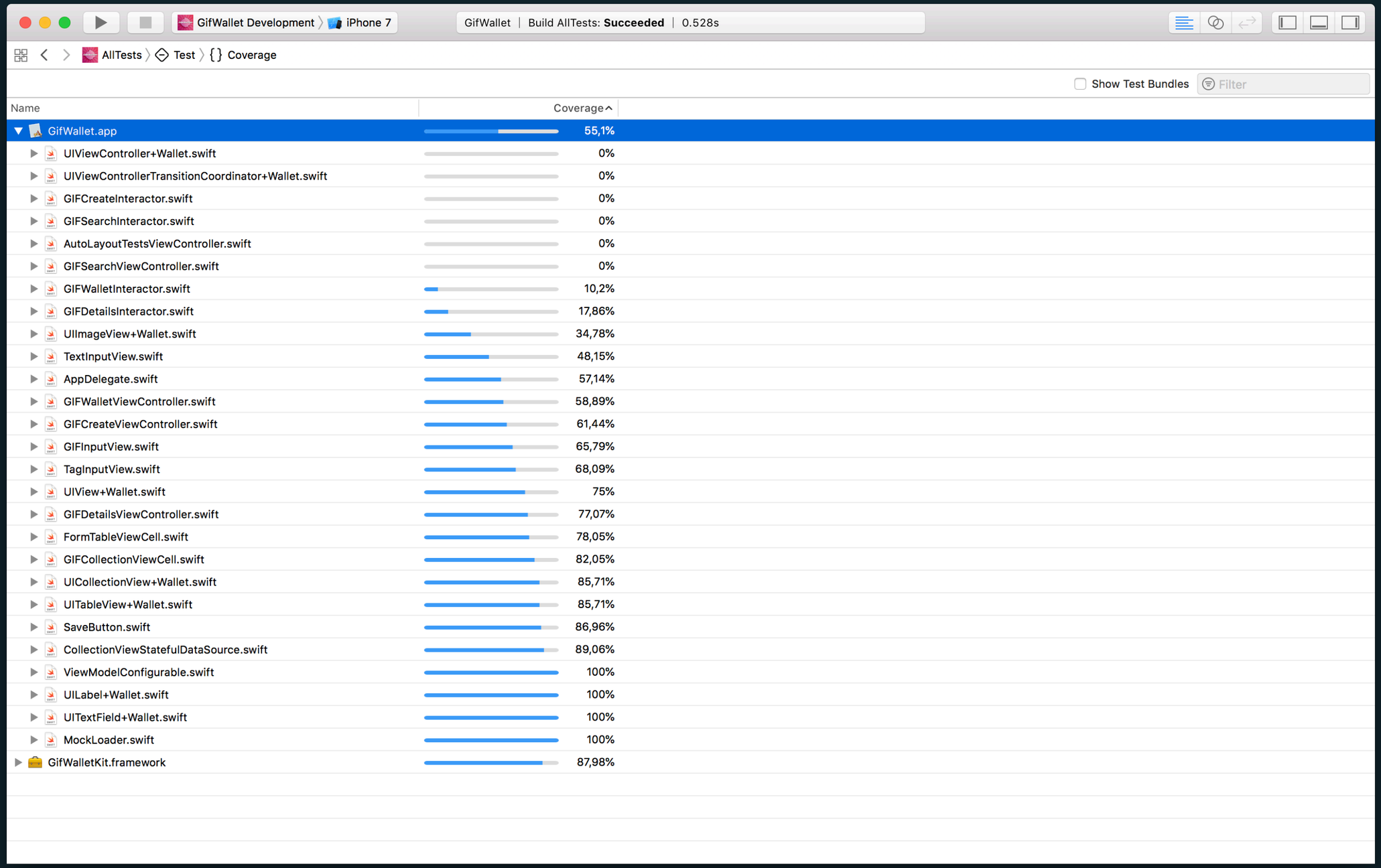
Tests

During this workshop, we've built:

— Unit Tests
— Snapshot Tests

However...


G...nt
iPhone 7
GifWallet | Build AllTests: Succeeded | 0.528s
AllTests
Test
Coverage
Show Test Bundles
Filter
Name
Coverage
GifWallet.app
55,1%
GifWalletKit.framework
87,98%

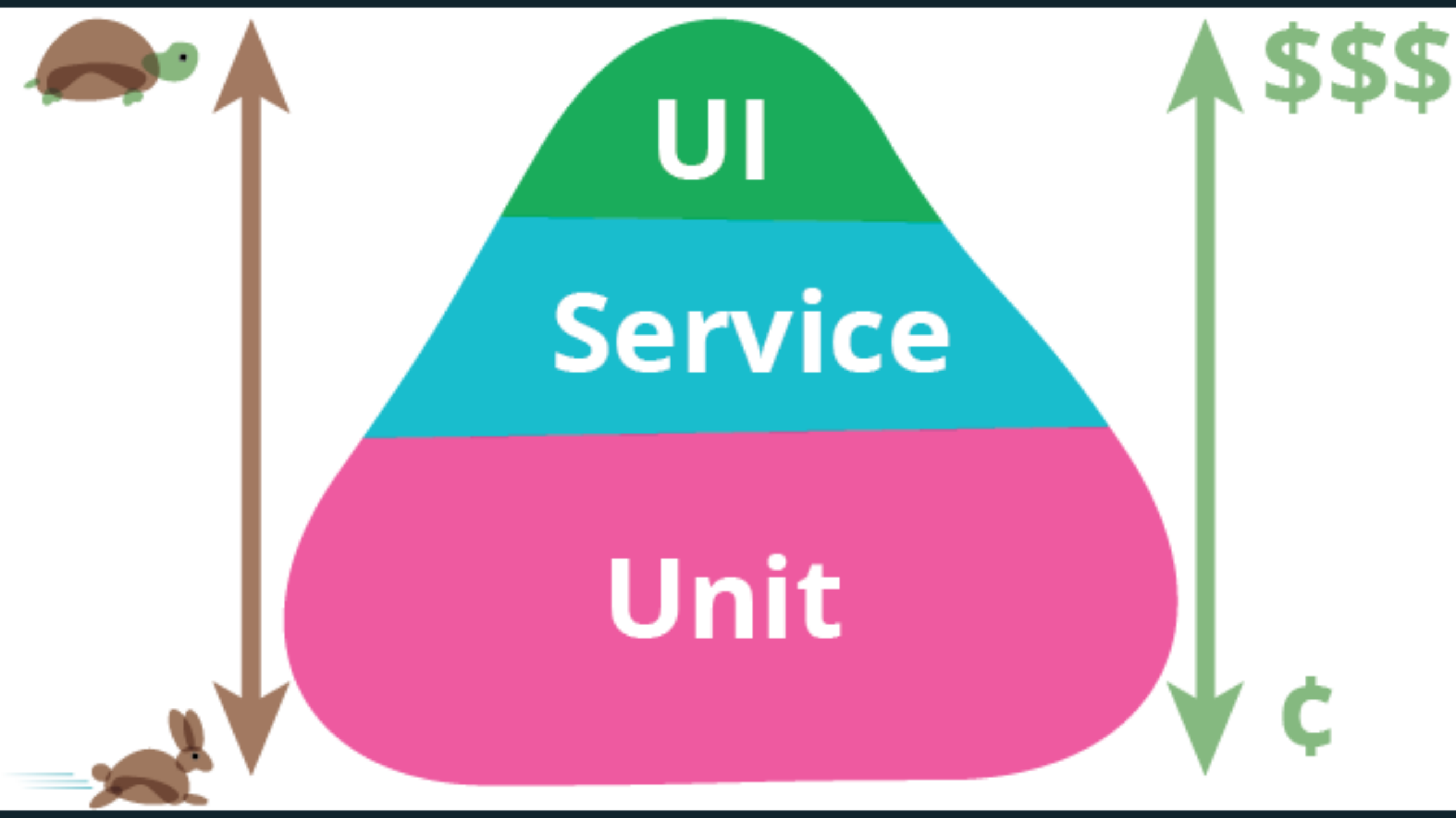

GifWallet Development 〉 iPhone 7
GifWallet | Build AllTests: Succeeded | 0.528s
AllTests 〉 Test 〉 { } Coverage
Show Test Bundles
Filter
Name
Coverage
GifWallet.app 55,1%
UIViewController+Wallet.swift 0%
UIViewControllerTransitionCoordinator+Wallet.swift 0%
GIFCreateInteractor.swift 0%
GIFSearchInteractor.swift 0%
AutoLayoutTestsViewController.swift 0%
GIFSearchViewController.swift 0%
GIFWalletInteractor.swift 10,2%
GIFDetailsInteractor.swift 17,86%
UIImageView+Wallet.swift 34,78%
TextInputView.swift 48,15%
AppDelegate.swift 57,14%
GIFWalletViewController.swift 58,89%
GIFCreateViewController.swift 61,44%
GIFInputView.swift 65,79%
TagInputView.swift 68,09%
UIView+Wallet.swift 75%
GIFDetailsViewController.swift 77,07%
FormTableViewCell.swift 78,05%
GIFCollectionViewCell.swift 82,05%
UICollectionView+Wallet.swift 85,71%
UITableView+Wallet.swift 85,71%
SaveButton.swift 86,96%
CollectionViewStatefulDataSource.swift 89,06%
ViewModelConfigurable.swift 100%
UILabel+Wallet.swift 100%
UITextField+Wallet.swift 100%
MockLoader.swift 100%
GifWalletKit.framework 87,98%

This is what UITests are for

User Interface Testing

— UI testing gives you the ability to find and interact with the UI of your app in order to validate the properties and state of the UI elements.

— Built with `XCTest` and `Accessibility`


UI
Service
Unit
$$$
¢

APIs

UI tests are based on the implementation of three new classes:

— `XCUIApplication`

— `XCUIElement`

— `XCUIElementQuery`

APIs

Best way to start it via recording

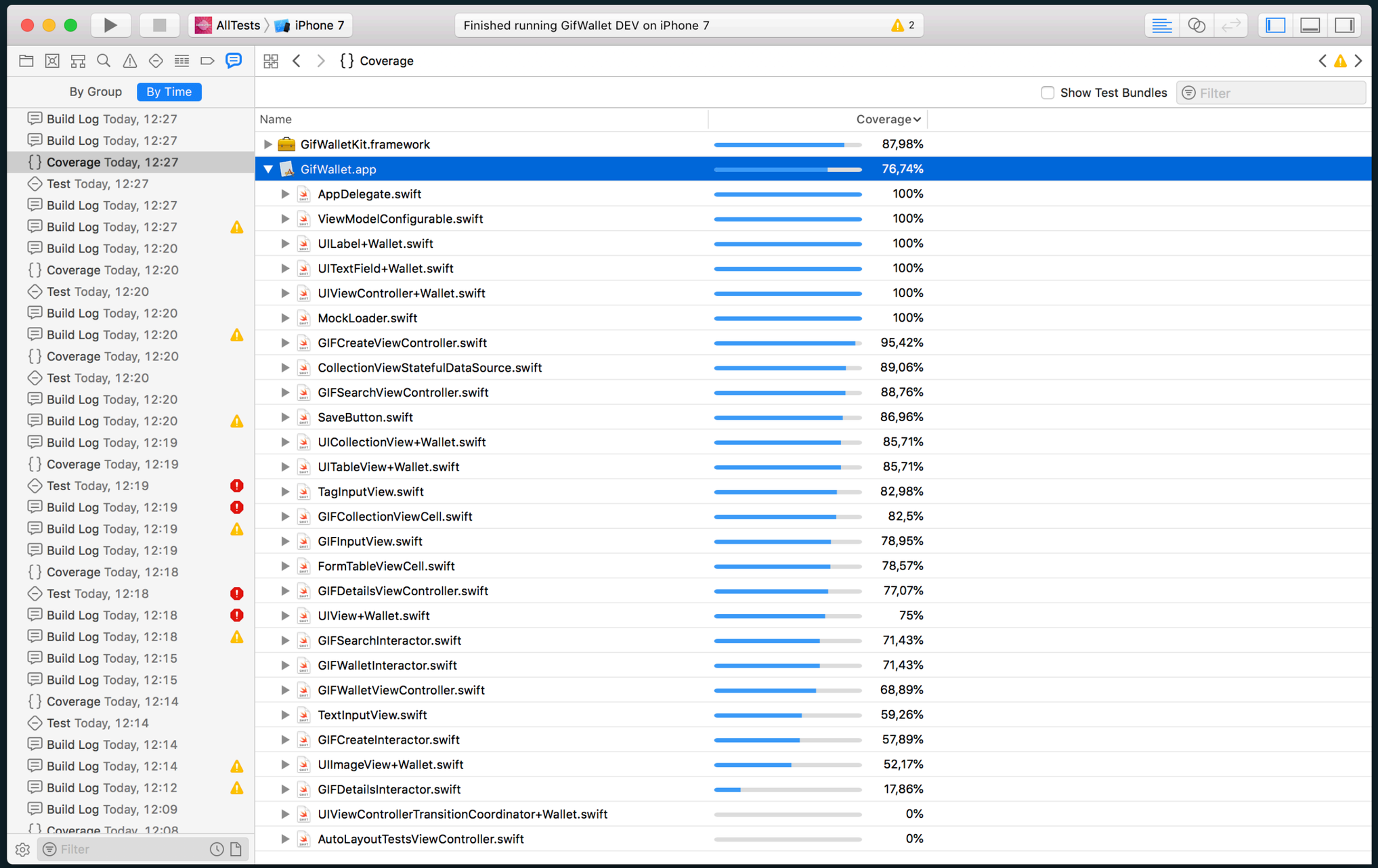
Demo

Takeaways

- UITests are executed on a different executable, not on your app.
  - Don't have access to your app's code.
  - Test preparation is done differently
- Elements are accessed via Accesibility queries
  - Slow 🐌.

Takeaways

- Ideal to run nightly or on every PR.
  - Definetly not for every build.
  - Run them on a different scheme.
- You'll get accessibility for free.
- You'll improve your coverage.


AllTests
iPhone 7
Finished running GifWallet DEV on iPhone 7
2
Coverage
By Group
By Time
Show Test Bundles
Filter
Name
Coverage
Build Log Today, 12:27
Build Log Today, 12:27
Coverage Today, 12:27
Test Today, 12:27
Build Log Today, 12:27
Build Log Today, 12:27
Build Log Today, 12:20
Coverage Today, 12:20
Test Today, 12:20
Build Log Today, 12:20
Build Log Today, 12:20
Coverage Today, 12:20
Test Today, 12:20
Build Log Today, 12:20
Build Log Today, 12:20
Build Log Today, 12:19
Coverage Today, 12:19
Test Today, 12:19
Build Log Today, 12:19
Build Log Today, 12:19
Build Log Today, 12:19
Coverage Today, 12:18
Test Today, 12:18
Build Log Today, 12:18
Build Log Today, 12:18
Build Log Today, 12:15
Build Log Today, 12:15
Coverage Today, 12:14
Test Today, 12:14
Build Log Today, 12:14
Build Log Today, 12:14
Build Log Today, 12:12
Build Log Today, 12:09
GifWalletKit.framework 87,98%
GifWallet.app 76,74%
AppDelegate.swift 100%
ViewModelConfigurable.swift 100%
UILabel+Wallet.swift 100%
UITextField+Wallet.swift 100%
UIViewController+Wallet.swift 100%
MockLoader.swift 100%
GIFCreateViewController.swift 95,42%
CollectionViewStatefulDataSource.swift 89,06%
GIFSearchViewController.swift 88,76%
SaveButton.swift 86,96%
UICollectionView+Wallet.swift 85,71%
UITableView+Wallet.swift 85,71%
TagInputView.swift 82,98%
GIFCollectionViewCell.swift 82,5%
GIFInputView.swift 78,95%
FormTableViewCell.swift 78,57%
GIFDetailsViewController.swift 77,07%
UIView+Wallet.swift 75%
GIFSearchInteractor.swift 71,43%
GIFWalletInteractor.swift 71,43%
GIFWalletViewController.swift 68,89%
TextInputView.swift 59,26%
GIFCreateInteractor.swift 57,89%
UIImageView+Wallet.swift 52,17%
GIFDetailsInteractor.swift 17,86%
UIViewControllerTransitionCoordinator+Wallet.swift 0%
AutoLayoutTestsViewController.swift 0%
Filter

Takeaways

— Environment variables can be passed using `launchArguments`

```
let app = XCUIApplication()
app.launchArguments.append("UITests")
app.launch()
...
if ProcessInfo.processInfo.arguments.contains("UITests") {
}
```

Takeaways

— Use this to create mock Interactors instead

```
let presenter: GIFWalletPresenterType = {
    if ProcessInfo.processInfo.arguments.contains("UITests") {
        return GIFWalletViewController.MockDataPresenter()
            } else {
        return GIFWalletViewController.Presenter(dataStore: dataStore)
    }
}()
```