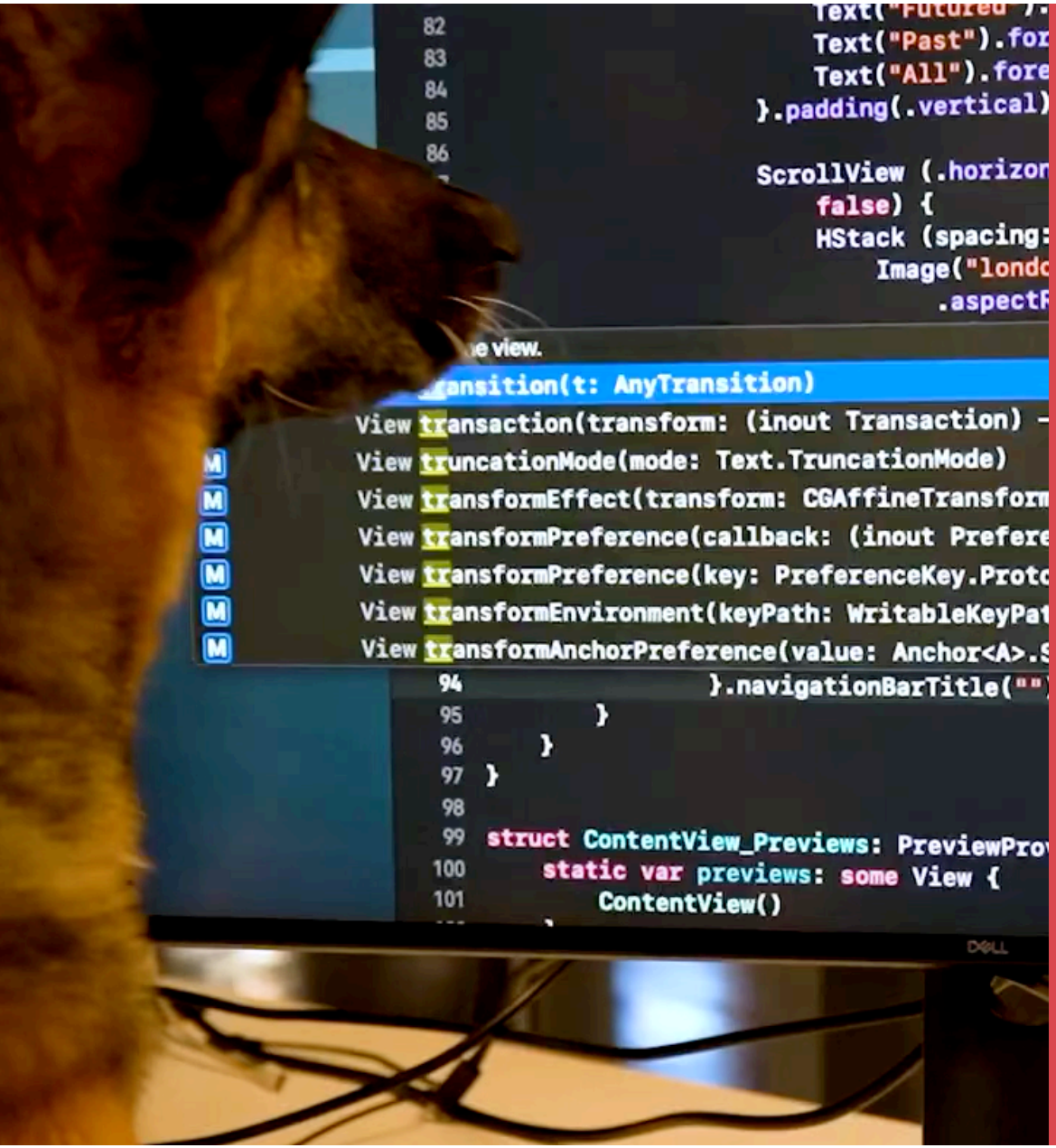
Documenting Your Project With DocC

SIMONBS.DEV


SOX
Chicago


Text("Past").for
Text("All").fore
}.padding(.vertical)
ScrollView (.horizon
false) {
HStack (spacing:
Image("londo
.aspectR
e view.
ansition(t: AnyTransition)
View transaction(transform: (inout Transaction) -
View truncationMode(mode: Text.TruncationMode)
View transformEffect(transform: CGAffineTransform
View transformPreference(callback: (inout Prefere
View transformPreference(key: PreferenceKey.Proto
View transformEnvironment(keyPath: WritableKeyPat
View transformAnchorPreference(value: Anchor<A>.S
}.navigationBarTitle("")
}
}
}
struct ContentView_Previews: PreviewPro
static var previews: some View {
ContentView()

shape

Runestone

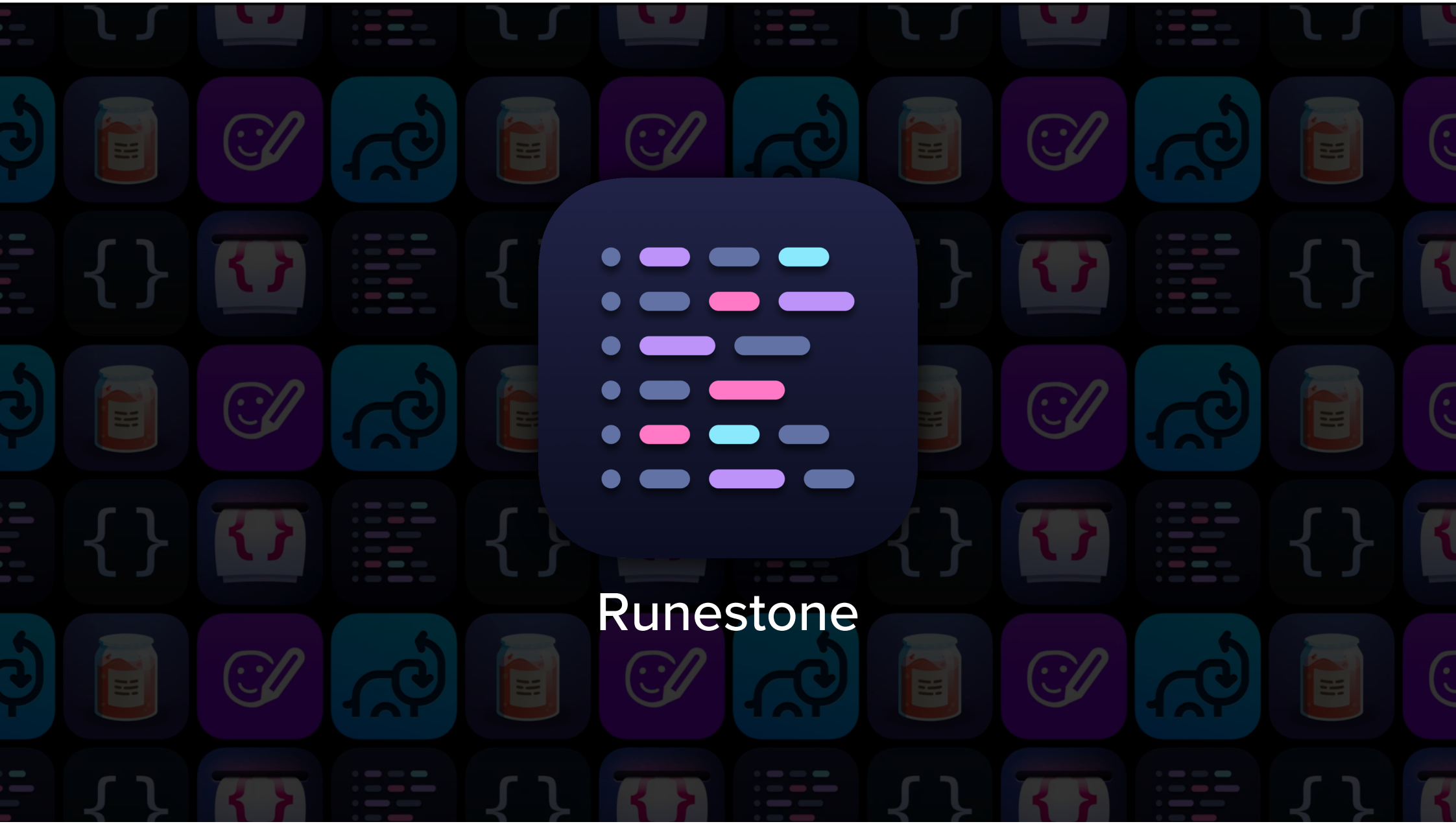
Scriptable

Jayson

Re: Toot

Data Jar

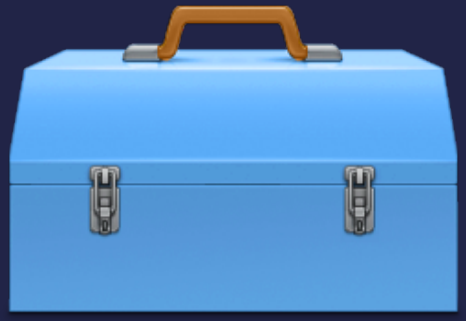
Sticky Draw


Runestone

Runestone

Runestone

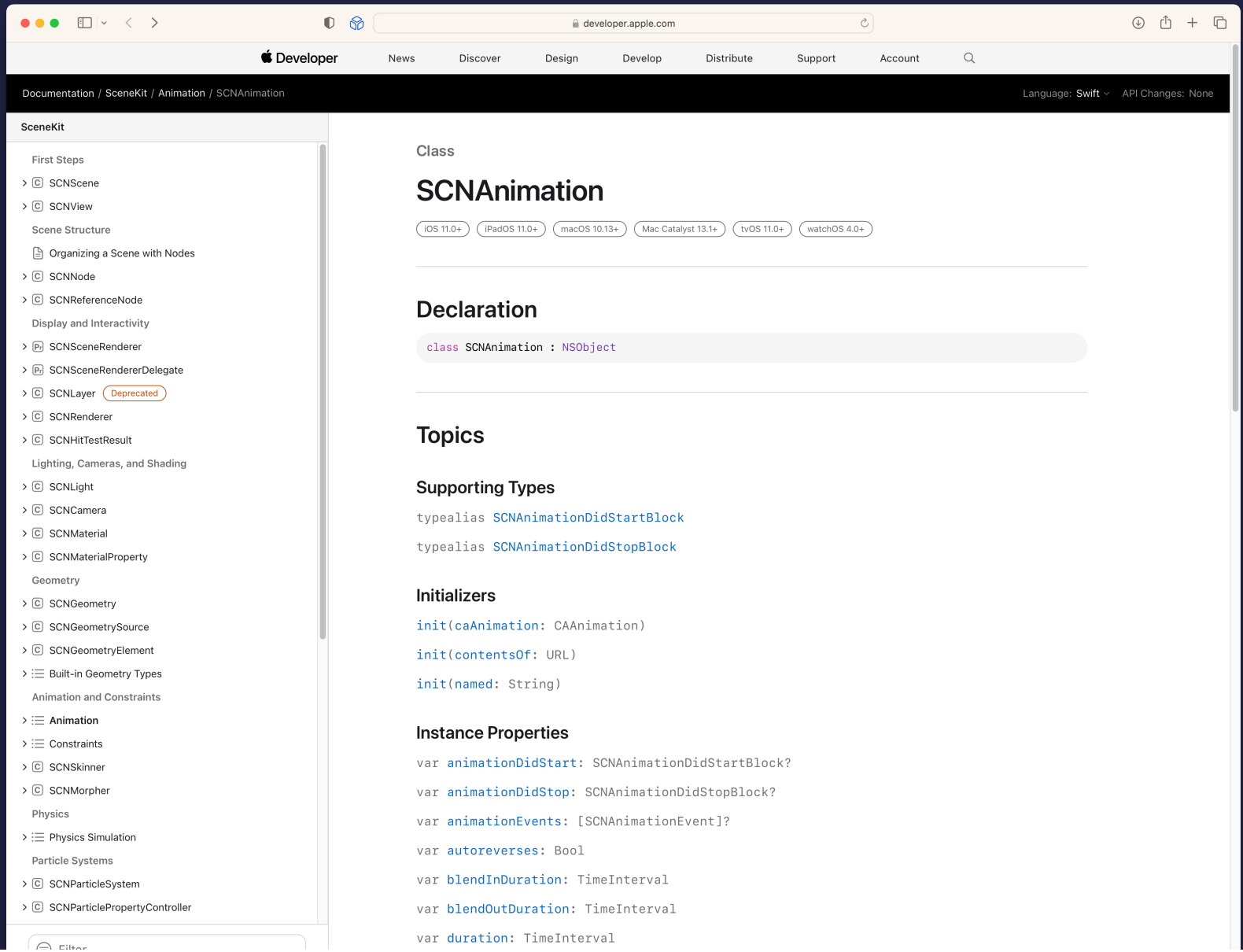
Swift Packages

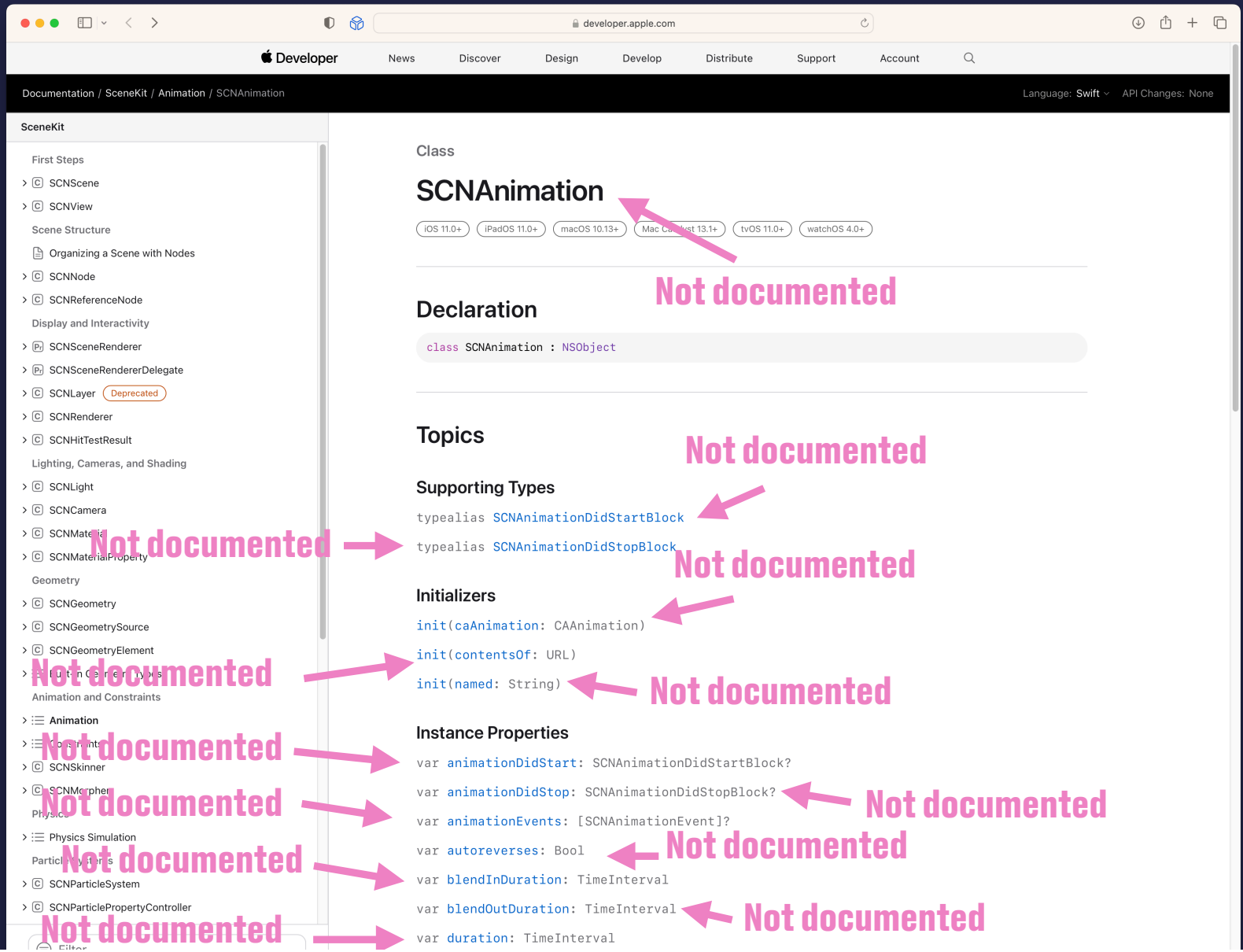


Frameworks

Apps

How can I document my project?


developer.apple.com
Developer
News
Discover
Design
Develop
Distribute
Support
Account
Documentation / SceneKit / Animation / SCNAnimation
Language: Swift
API Changes: None
SceneKit
First Steps
SCNScene
SCNView
Scene Structure
Organizing a Scene with Nodes
SCNNode
SCNReferenceNode
Display and Interactivity
SCNSceneRenderer
SCNSceneRendererDelegate
SCNLayer
Deprecated
SCNRenderer
SCNHitTestResult
Lighting, Cameras, and Shading
SCNLight
SCNCamera
SCNMaterial
SCNMaterialProperty
Geometry
SCNGeometry
SCNGeometrySource
SCNGeometryElement
Built-in Geometry Types
Animation and Constraints
Animation
Constraints
SCNSkinner
SCNMorpher
Physics
Physics Simulation
Particle Systems
SCNParticleSystem
SCNParticlePropertyController
Class
SCNAnimation
iOS 11.0+
iPadOS 11.0+
macOS 10.13+
Mac Catalyst 13.1+
tvOS 11.0+
watchOS 4.0+
Declaration
class SCNAnimation : NSObject
Topics
Supporting Types
typealias SCNAnimationDidStartBlock
typealias SCNAnimationDidStopBlock
Initializers
init(caAnimation: CAAnimation)
init(contentsOf: URL)
init(named: String)
Instance Properties
var animationDidStart: SCNAnimationDidStartBlock?
var animationDidStop: SCNAnimationDidStopBlock?
var animationEvents: [SCNAnimationEvent]?
var autoreverses: Bool
var blendInDuration: TimeInterval
var blendOutDuration: TimeInterval
var duration: TimeInterval


developer.apple.com
Developer
News
Discover
Design
Develop
Distribute
Support
Account
Documentation / SceneKit / Animation / SCNAnimation
Language: Swift
API Changes: None
SceneKit
First Steps
SCNScene
SCNView
Scene Structure
Organizing a Scene with Nodes
SCNNode
SCNReferenceNode
Display and Interactivity
SCNSceneRenderer
SCNSceneRendererDelegate
SCNLayer
Deprecated
SCNRenderer
SCNHitTestResult
Lighting, Cameras, and Shading
SCNLight
SCNCamera
SCNMaterial
SCNMaterialProperty
Geometry
SCNGeometry
SCNGeometrySource
SCNGeometryElement
Animation and Constraints
Animation
SCNSkinner
Physics Simulation
SCNParticleSystem
SCNParticlePropertyController
Class
SCNAnimation
iOS 11.0+
iPadOS 11.0+
macOS 10.13+
tvOS 11.0+
watchOS 4.0+
Not documented
Declaration
class SCNAnimation : NSObject
Topics
Not documented
Supporting Types
typealias SCNAnimationDidStartBlock
typealias SCNAnimationDidStopBlock
Not documented
Not documented
Initializers
init(caAnimation: CAAnimation)
init(contentsOf: URL)
init(named: String)
Not documented
Not documented
Not documented
Not documented
Not documented
Not documented
Instance Properties
var animationDidStart: SCNAnimationDidStartBlock?
var animationDidStop: SCNAnimationDidStopBlock?
var animationEvents: [SCNAnimationEvent]?
var autoreverses: Bool
var blendInDuration: TimeInterval
var blendOutDuration: TimeInterval
var duration: TimeInterval
Not documented
Not documented
Not documented
Not documented

developer.apple.com

Developer News Discover Design Develop Distribute Support Account

Documentation / WidgetKit

Language: Swift API Changes: Show

WidgetKit

- Widget creation
  - Creating a Widget Extension
  - Creating Lock Screen Widgets and ...
  - Migrating ClockKit complications to...
  - Building Widgets Using WidgetKit a...
  - Adding widgets to the Lock Screen ...
  - Fruta: Building a Feature-Rich App ...
  - Widget
  - WidgetBundle
  - StaticConfiguration
  - WidgetFamily
  - WidgetRenderingMode
- Configurable widgets
  - Making a Configurable Widget
  - IntentConfiguration
  - WidgetInfo
  - IntentRecommendation
- Timeline management
  - Keeping a Widget Up To Date
  - TimelineProvider
  - IntentTimelineProvider
  - TimelineProviderContext
  - TimelineEntry
  - Timeline
  - WidgetCenter
- User interface
  - SwiftUI Views
  - Introducing SwiftUI
  - AccessoryWidgetBackground
- Location services in widgets
  - Accessing Location Information in ...
- Smart stacks

Filter

Framework

WidgetKit

Show relevant, glanceable content from your app as widgets in iOS and macOS, and as watch complications.

iOS 14.0+ iPadOS 14.0+ macOS 11.0+ Mac Catalyst 14.0+

Overview

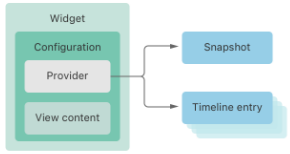
WidgetKit gives users ready access to content in your app by putting widgets on the iOS Home Screen and Today View, the macOS Notification Center, and by putting accessory widgets on the Lock Screen on iPhone and as complications in watchOS. Your widgets stay up to date so users always have the latest information at a glance. When they need more details, your widget takes them directly to the appropriate place in your app.

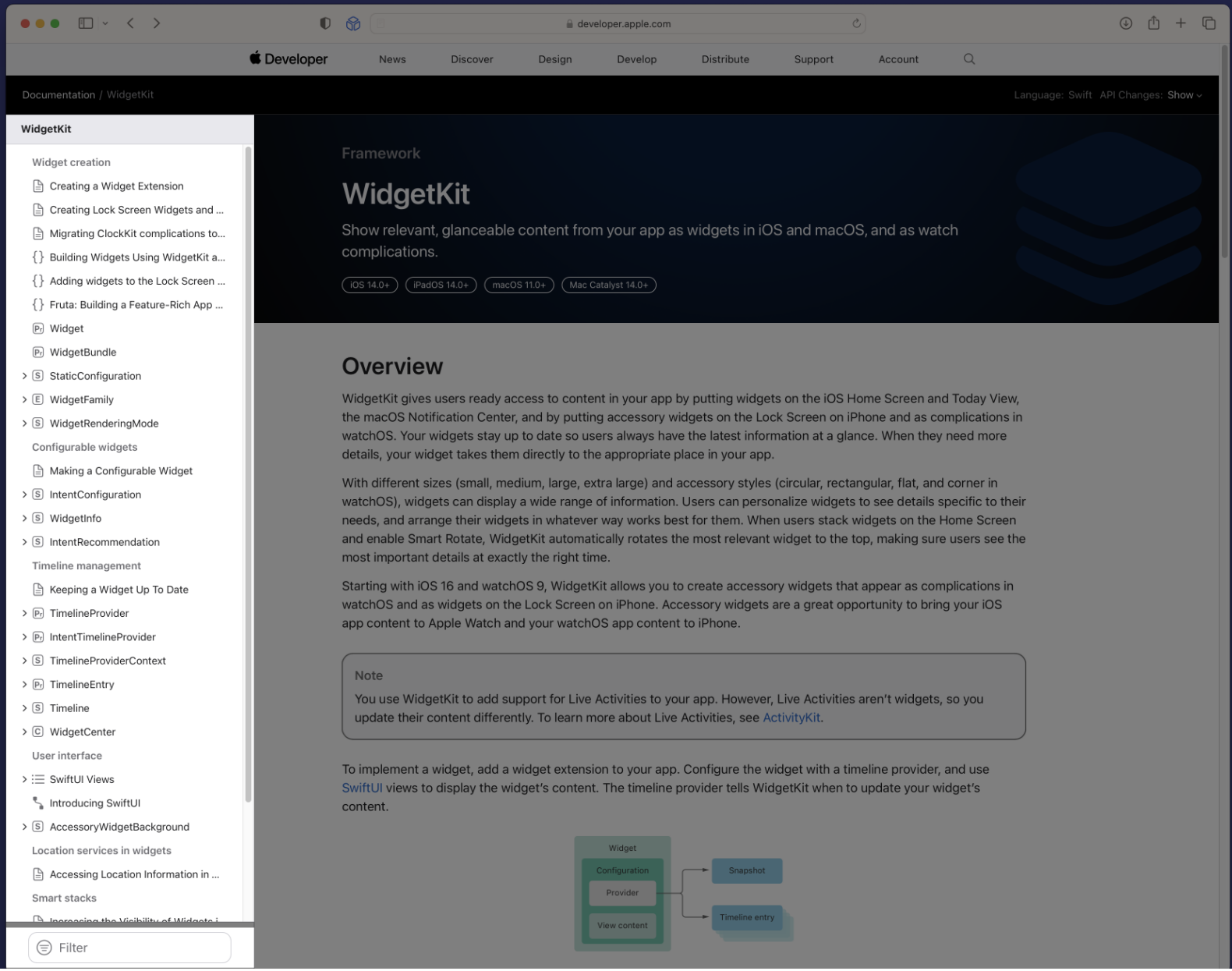
With different sizes (small, medium, large, extra large) and accessory styles (circular, rectangular, flat, and corner in watchOS), widgets can display a wide range of information. Users can personalize widgets to see details specific to their needs, and arrange their widgets in whatever way works best for them. When users stack widgets on the Home Screen and enable Smart Rotate, WidgetKit automatically rotates the most relevant widget to the top, making sure users see the most important details at exactly the right time.

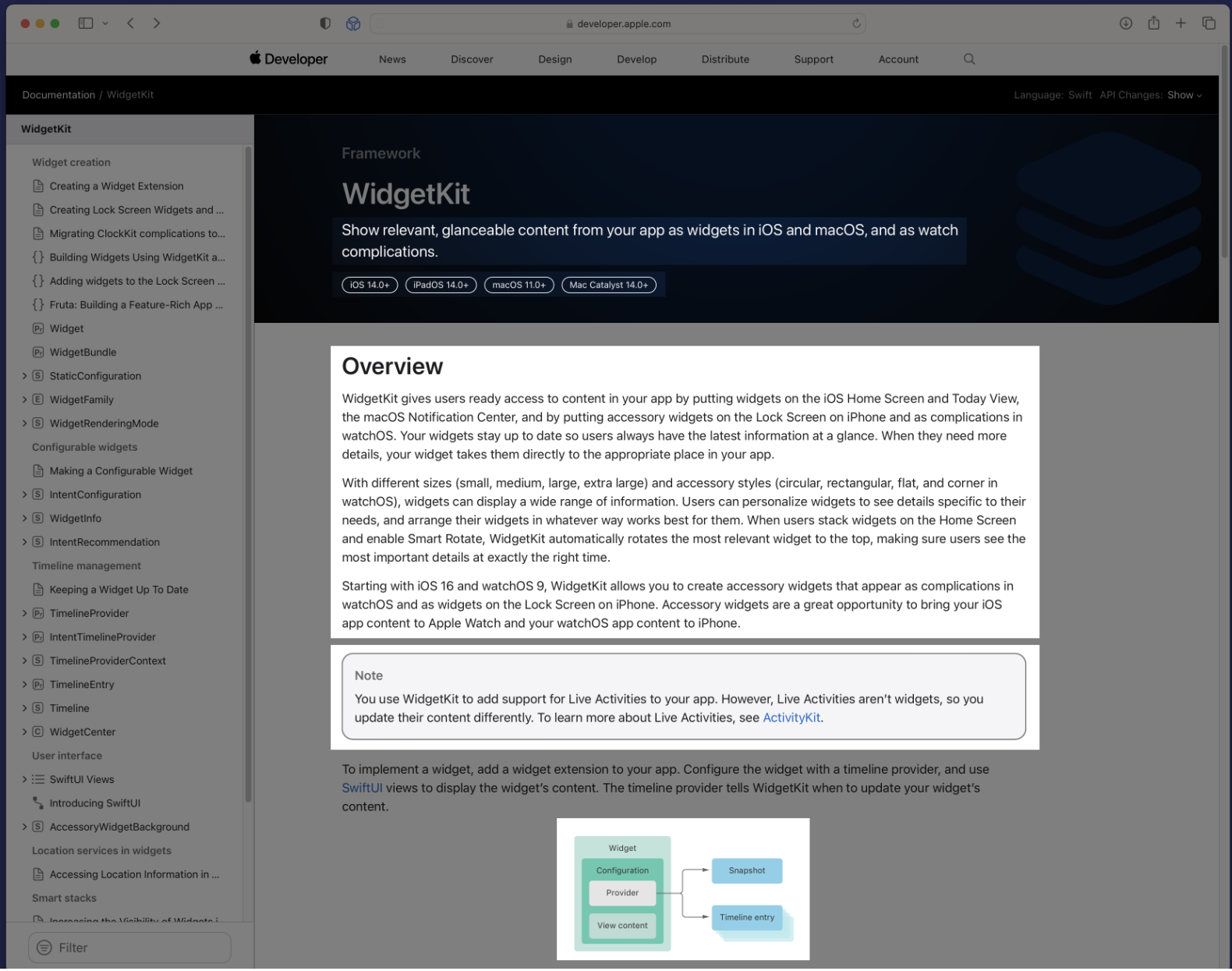
Starting with iOS 16 and watchOS 9, WidgetKit allows you to create accessory widgets that appear as complications in watchOS and as widgets on the Lock Screen on iPhone. Accessory widgets are a great opportunity to bring your iOS app content to Apple Watch and your watchOS app content to iPhone.

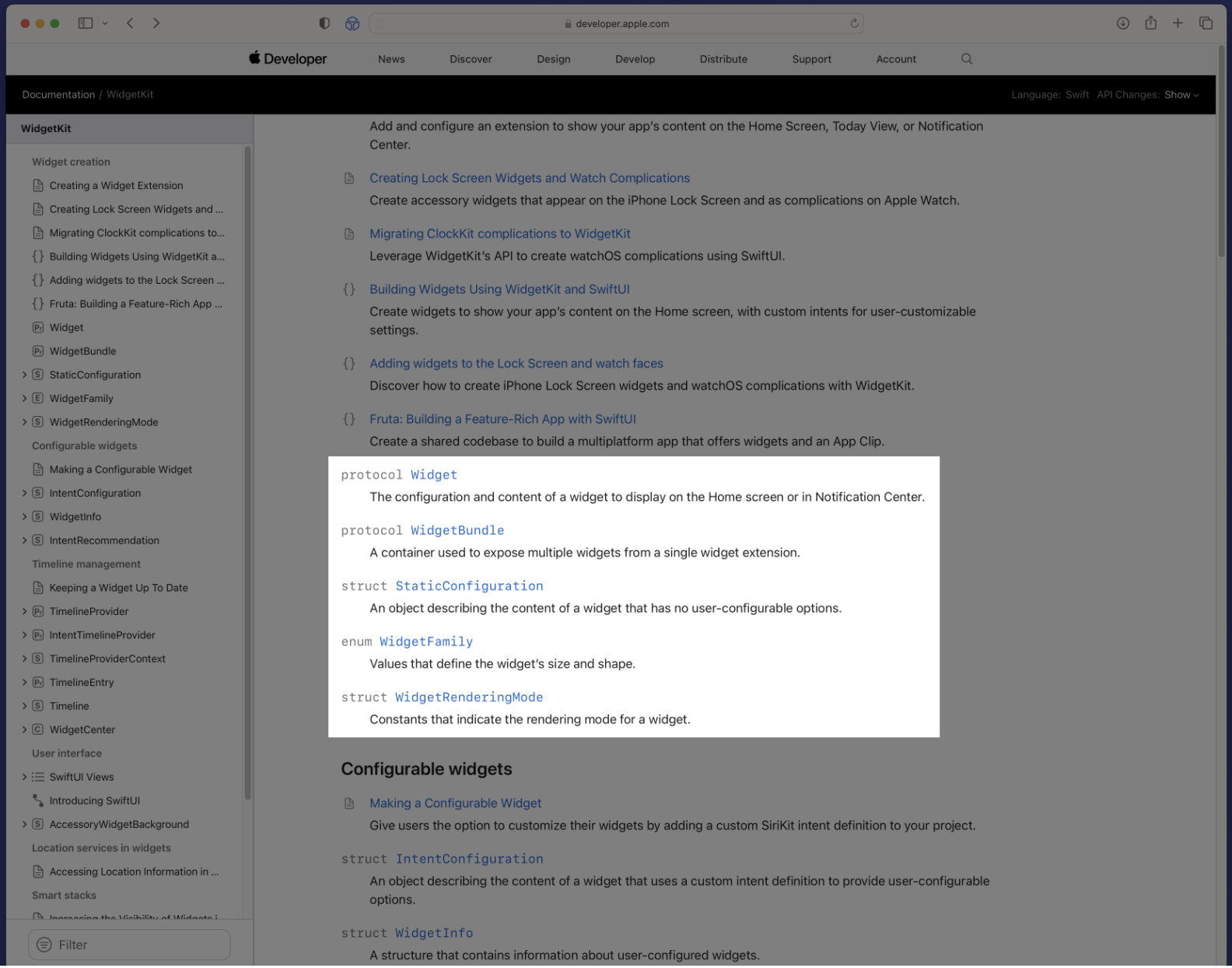
> **Note**
> You use WidgetKit to add support for Live Activities to your app. However, Live Activities aren't widgets, so you update their content differently. To learn more about Live Activities, see ActivityKit.

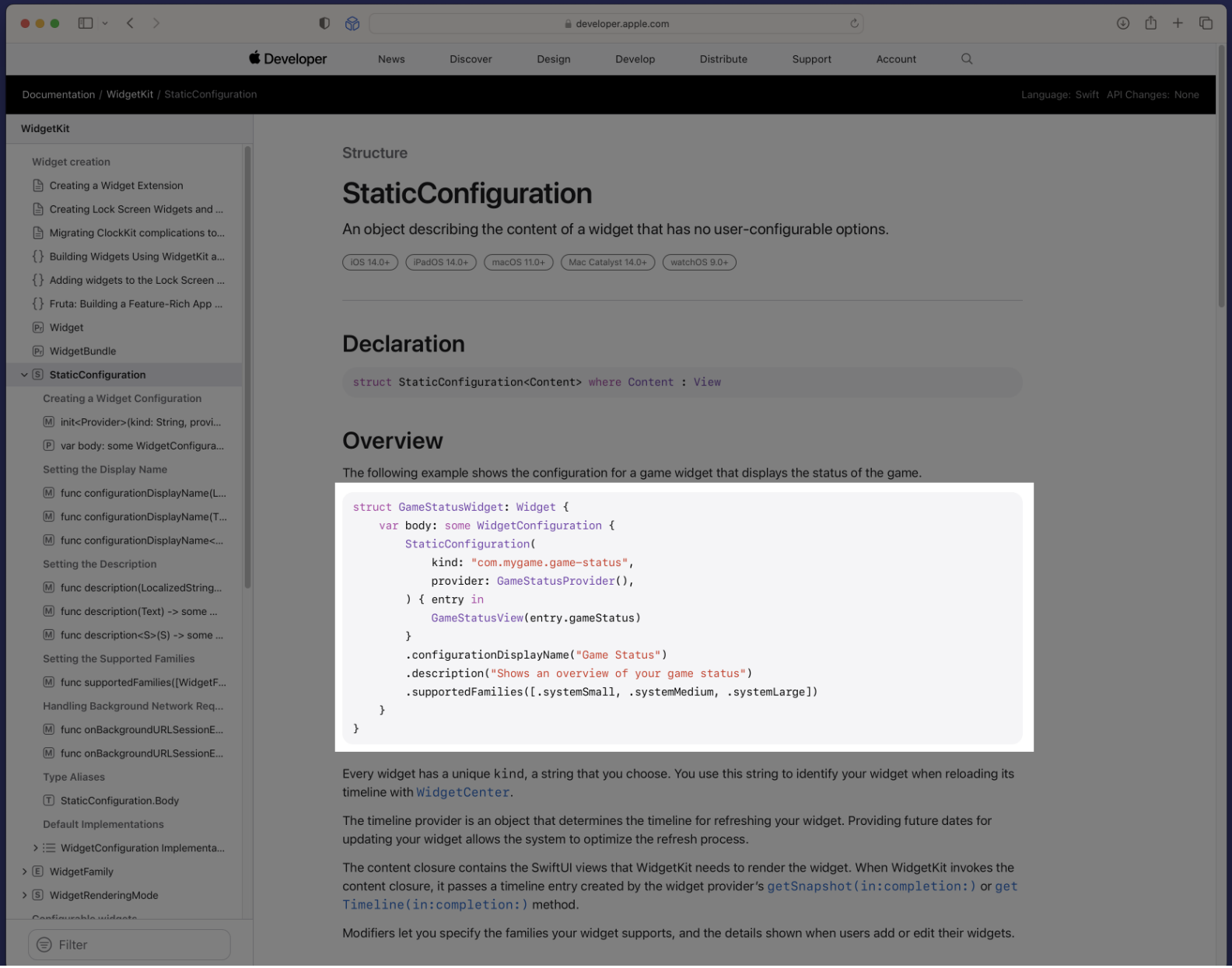
To implement a widget, add a widget extension to your app. Configure the widget with a timeline provider, and use SwiftUI views to display the widget's content. The timeline provider tells WidgetKit when to update your widget's content.




developer.apple.com
Developer
News
Discover
Design
Develop
Distribute
Support
Account
Documentation / WidgetKit
Language: Swift API Changes: Show
WidgetKit
Widget creation
Creating a Widget Extension
Creating Lock Screen Widgets and ...
Migrating ClockKit complications to...
Building Widgets Using WidgetKit a...
Adding widgets to the Lock Screen ...
Fruta: Building a Feature-Rich App ...
Widget
WidgetBundle
StaticConfiguration
WidgetFamily
WidgetRenderingMode
Configurable widgets
Making a Configurable Widget
IntentConfiguration
WidgetInfo
IntentRecommendation
Timeline management
Keeping a Widget Up To Date
TimelineProvider
IntentTimelineProvider
TimelineProviderContext
TimelineEntry
Timeline
WidgetCenter
User interface
SwiftUI Views
Introducing SwiftUI
AccessoryWidgetBackground
Location services in widgets
Accessing Location Information in ...
Smart stacks
Filter
Framework
WidgetKit
Show relevant, glanceable content from your app as widgets in iOS and macOS, and as watch complications.
iOS 14.0+
iPadOS 14.0+
macOS 11.0+
Mac Catalyst 14.0+
Overview
WidgetKit gives users ready access to content in your app by putting widgets on the iOS Home Screen and Today View, the macOS Notification Center, and by putting accessory widgets on the Lock Screen on iPhone and as complications in watchOS. Your widgets stay up to date so users always have the latest information at a glance. When they need more details, your widget takes them directly to the appropriate place in your app.
With different sizes (small, medium, large, extra large) and accessory styles (circular, rectangular, flat, and corner in watchOS), widgets can display a wide range of information. Users can personalize widgets to see details specific to their needs, and arrange their widgets in whatever way works best for them. When users stack widgets on the Home Screen and enable Smart Rotate, WidgetKit automatically rotates the most relevant widget to the top, making sure users see the most important details at exactly the right time.
Starting with iOS 16 and watchOS 9, WidgetKit allows you to create accessory widgets that appear as complications in watchOS and as widgets on the Lock Screen on iPhone. Accessory widgets are a great opportunity to bring your iOS app content to Apple Watch and your watchOS app content to iPhone.
Note
You use WidgetKit to add support for Live Activities to your app. However, Live Activities aren't widgets, so you update their content differently. To learn more about Live Activities, see ActivityKit.
To implement a widget, add a widget extension to your app. Configure the widget with a timeline provider, and use SwiftUI views to display the widget's content. The timeline provider tells WidgetKit when to update your widget's content.
Widget
Configuration
Provider
View content
Snapshot
Timeline entry


developer.apple.com
Developer
News
Discover
Design
Develop
Distribute
Support
Account
Documentation / WidgetKit
Language: Swift API Changes: Show
WidgetKit
Widget creation
Creating a Widget Extension
Creating Lock Screen Widgets and ...
Migrating ClockKit complications to...
Building Widgets Using WidgetKit a...
Adding widgets to the Lock Screen ...
Fruta: Building a Feature-Rich App ...
Widget
WidgetBundle
StaticConfiguration
WidgetFamily
WidgetRenderingMode
Configurable widgets
Making a Configurable Widget
IntentConfiguration
WidgetInfo
IntentRecommendation
Timeline management
Keeping a Widget Up To Date
TimelineProvider
IntentTimelineProvider
TimelineProviderContext
TimelineEntry
Timeline
WidgetCenter
User interface
SwiftUI Views
Introducing SwiftUI
AccessoryWidgetBackground
Location services in widgets
Accessing Location Information in ...
Smart stacks
Filter
Framework
WidgetKit
Show relevant, glanceable content from your app as widgets in iOS and macOS, and as watch complications.
iOS 14.0+
iPadOS 14.0+
macOS 11.0+
Mac Catalyst 14.0+
Overview
WidgetKit gives users ready access to content in your app by putting widgets on the iOS Home Screen and Today View, the macOS Notification Center, and by putting accessory widgets on the Lock Screen on iPhone and as complications in watchOS. Your widgets stay up to date so users always have the latest information at a glance. When they need more details, your widget takes them directly to the appropriate place in your app.
With different sizes (small, medium, large, extra large) and accessory styles (circular, rectangular, flat, and corner in watchOS), widgets can display a wide range of information. Users can personalize widgets to see details specific to their needs, and arrange their widgets in whatever way works best for them. When users stack widgets on the Home Screen and enable Smart Rotate, WidgetKit automatically rotates the most relevant widget to the top, making sure users see the most important details at exactly the right time.
Starting with iOS 16 and watchOS 9, WidgetKit allows you to create accessory widgets that appear as complications in watchOS and as widgets on the Lock Screen on iPhone. Accessory widgets are a great opportunity to bring your iOS app content to Apple Watch and your watchOS app content to iPhone.
Note
You use WidgetKit to add support for Live Activities to your app. However, Live Activities aren't widgets, so you update their content differently. To learn more about Live Activities, see ActivityKit.
To implement a widget, add a widget extension to your app. Configure the widget with a timeline provider, and use SwiftUI views to display the widget's content. The timeline provider tells WidgetKit when to update your widget's content.
Widget
Configuration
Provider
View content
Snapshot
Timeline entry

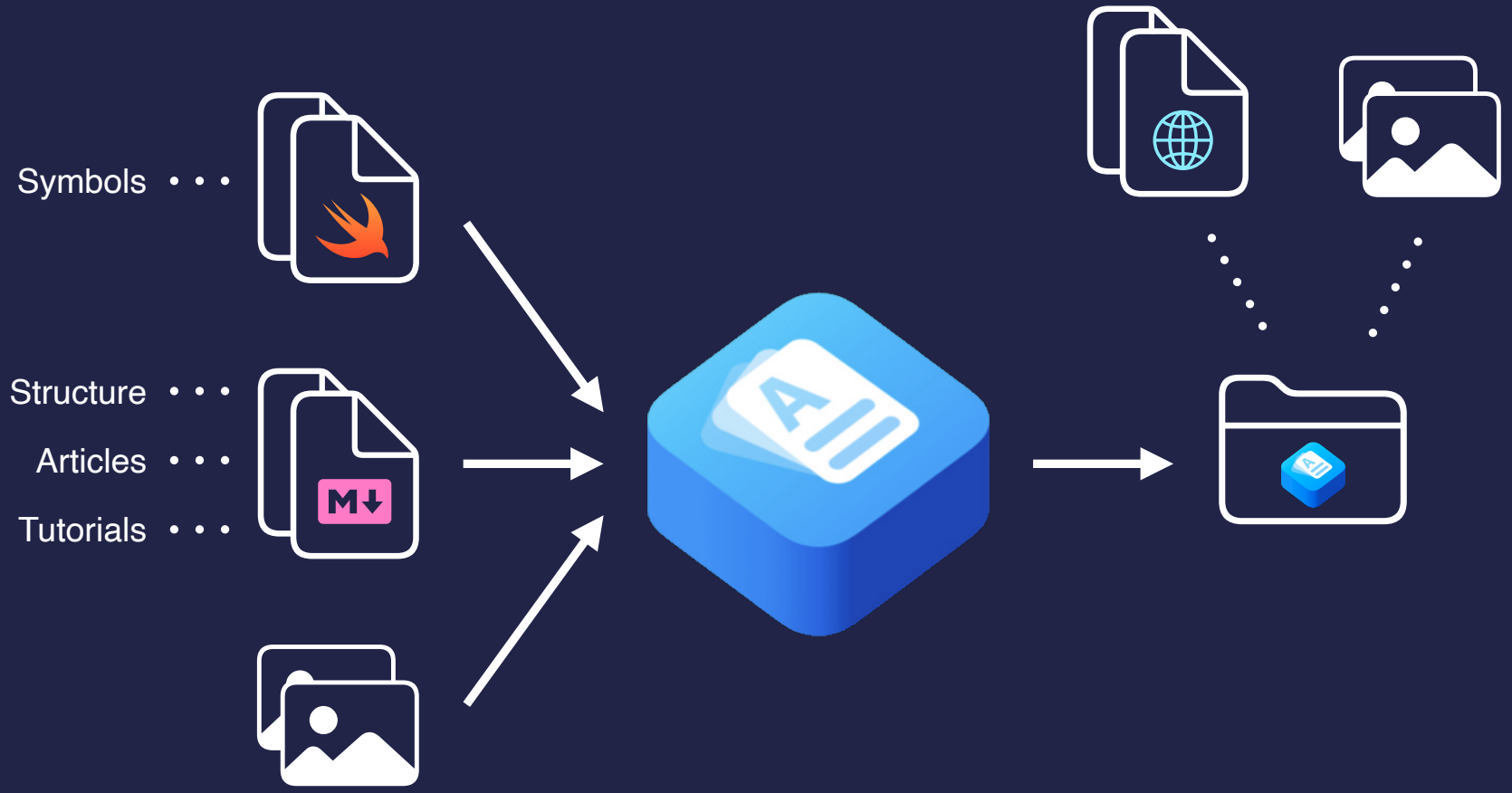

developer.apple.com
Developer
News
Discover
Design
Develop
Distribute
Support
Account
Documentation / WidgetKit
Language: Swift API Changes: Show
WidgetKit
Widget creation
Creating a Widget Extension
Creating Lock Screen Widgets and ...
Migrating ClockKit complications to...
Building Widgets Using WidgetKit a...
Adding widgets to the Lock Screen ...
Fruta: Building a Feature-Rich App ...
Widget
WidgetBundle
StaticConfiguration
WidgetFamily
WidgetRenderingMode
Configurable widgets
Making a Configurable Widget
IntentConfiguration
WidgetInfo
IntentRecommendation
Timeline management
Keeping a Widget Up To Date
TimelineProvider
IntentTimelineProvider
TimelineProviderContext
TimelineEntry
Timeline
WidgetCenter
User interface
SwiftUI Views
Introducing SwiftUI
AccessoryWidgetBackground
Location services in widgets
Accessing Location Information in ...
Smart stacks
Filter
Add and configure an extension to show your app's content on the Home Screen, Today View, or Notification Center.
Creating Lock Screen Widgets and Watch Complications
Create accessory widgets that appear on the iPhone Lock Screen and as complications on Apple Watch.
Migrating ClockKit complications to WidgetKit
Leverage WidgetKit's API to create watchOS complications using SwiftUI.
Building Widgets Using WidgetKit and SwiftUI
Create widgets to show your app's content on the Home screen, with custom intents for user-customizable settings.
Adding widgets to the Lock Screen and watch faces
Discover how to create iPhone Lock Screen widgets and watchOS complications with WidgetKit.
Fruta: Building a Feature-Rich App with SwiftUI
Create a shared codebase to build a multiplatform app that offers widgets and an App Clip.
protocol Widget
The configuration and content of a widget to display on the Home screen or in Notification Center.
protocol WidgetBundle
A container used to expose multiple widgets from a single widget extension.
struct StaticConfiguration
An object describing the content of a widget that has no user-configurable options.
enum WidgetFamily
Values that define the widget's size and shape.
struct WidgetRenderingMode
Constants that indicate the rendering mode for a widget.
Configurable widgets
Making a Configurable Widget
Give users the option to customize their widgets by adding a custom SiriKit intent definition to your project.
struct IntentConfiguration
An object describing the content of a widget that uses a custom intent definition to provide user-configurable options.
struct WidgetInfo
A structure that contains information about user-configured widgets.


developer.apple.com
Developer
News
Discover
Design
Develop
Distribute
Support
Account
Documentation / WidgetKit / StaticConfiguration
Language: Swift API Changes: None
WidgetKit
Widget creation
Creating a Widget Extension
Creating Lock Screen Widgets and ...
Migrating ClockKit complications to...
Building Widgets Using WidgetKit a...
Adding widgets to the Lock Screen ...
Fruta: Building a Feature-Rich App ...
Widget
WidgetBundle
StaticConfiguration
Creating a Widget Configuration
init<Provider>(kind: String, provi...
var body: some WidgetConfigura...
Setting the Display Name
func configurationDisplayName(L...
func configurationDisplayName(T...
func configurationDisplayName<...
Setting the Description
func description(LocalizedString...
func description(Text) -> some ...
func description<S>(S) -> some ...
Setting the Supported Families
func supportedFamilies([WidgetF...
Handling Background Network Req...
func onBackgroundURLSessionE...
func onBackgroundURLSessionE...
Type Aliases
StaticConfiguration.Body
Default Implementations
WidgetConfiguration Implementa...
WidgetFamily
WidgetRenderingMode
Filter
Structure
StaticConfiguration
An object describing the content of a widget that has no user-configurable options.
iOS 14.0+
iPadOS 14.0+
macOS 11.0+
Mac Catalyst 14.0+
watchOS 9.0+
Declaration
struct StaticConfiguration<Content> where Content : View
Overview
The following example shows the configuration for a game widget that displays the status of the game.
struct GameStatusWidget: Widget {
var body: some WidgetConfiguration {
StaticConfiguration(
kind: "com.mygame.game-status",
provider: GameStatusProvider(),
) { entry in
GameStatusView(entry.gameStatus)
}
.configurationDisplayName("Game Status")
.description("Shows an overview of your game status")
.supportedFamilies([.systemSmall, .systemMedium, .systemLarge])
}
}
Every widget has a unique kind, a string that you choose. You use this string to identify your widget when reloading its timeline with WidgetCenter.
The timeline provider is an object that determines the timeline for refreshing your widget. Providing future dates for updating your widget allows the system to optimize the refresh process.
The content closure contains the SwiftUI views that WidgetKit needs to render the widget. When WidgetKit invokes the content closure, it passes a timeline entry created by the widget provider's getSnapshot(in:completion:) or getTimeline(in:completion:) method.
Modifiers let you specify the families your widget supports, and the details shown when users add or edit their widgets.

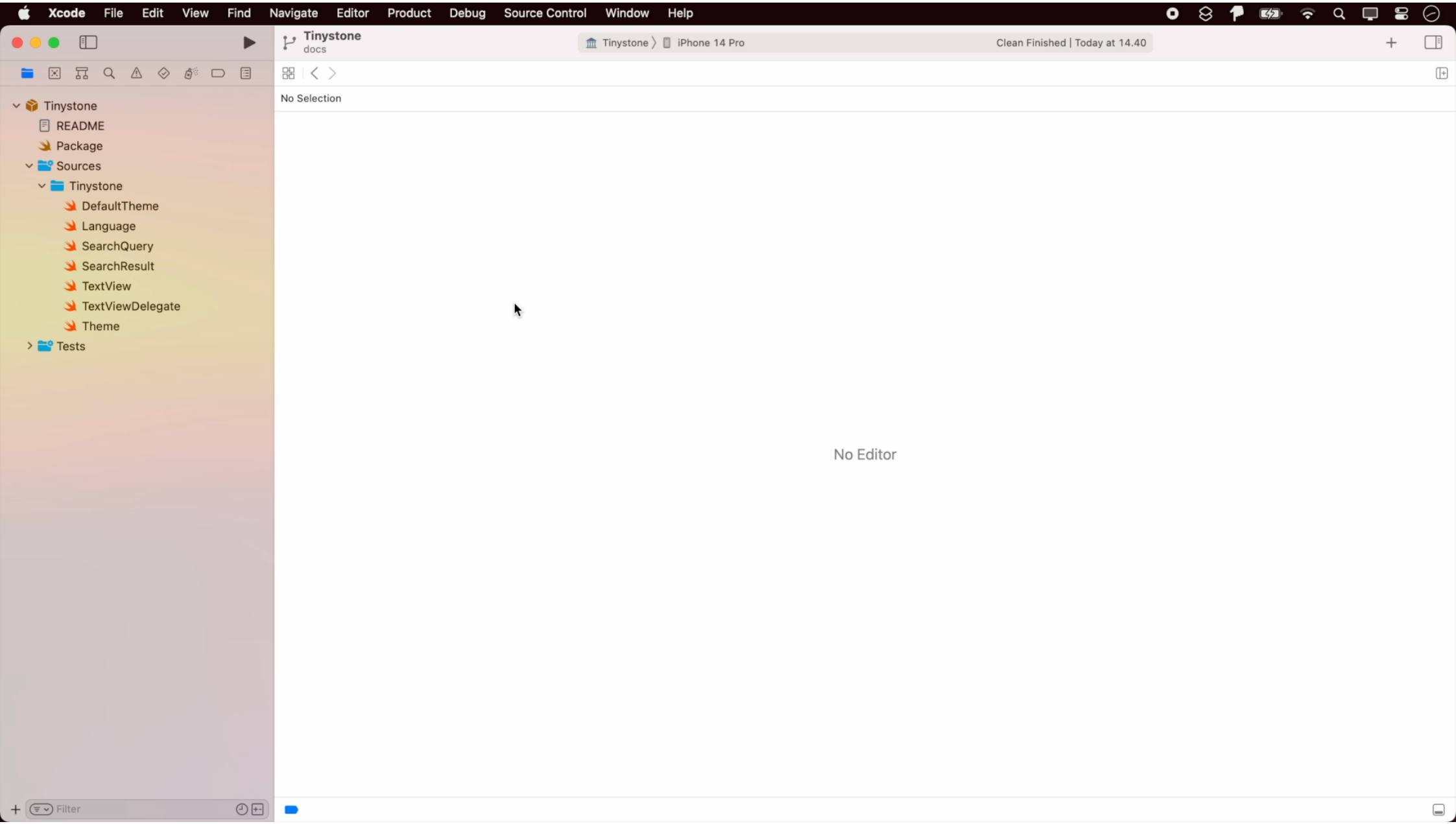
Apple's documentation is great.

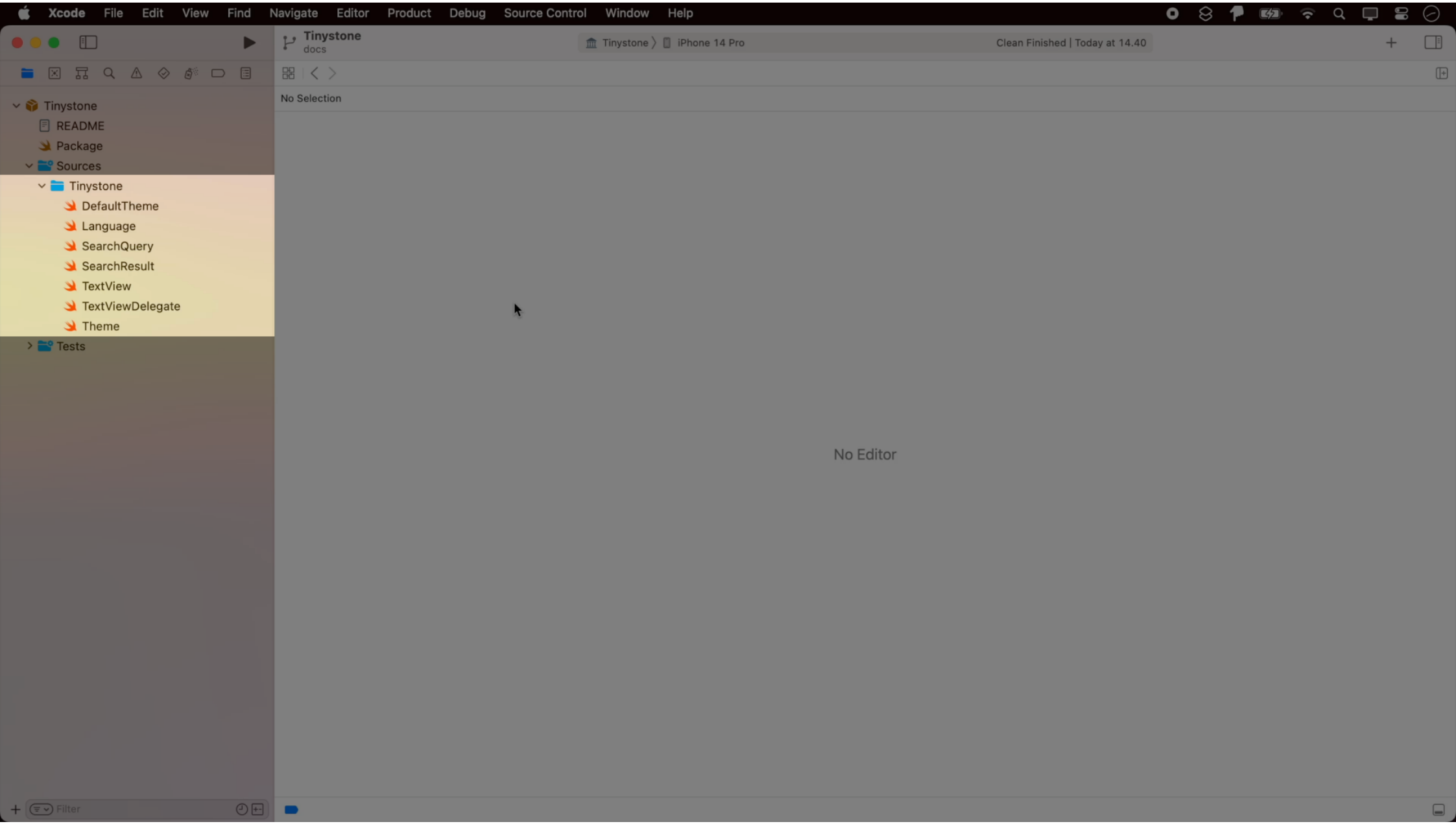
Apple's documentation is great when it exists.

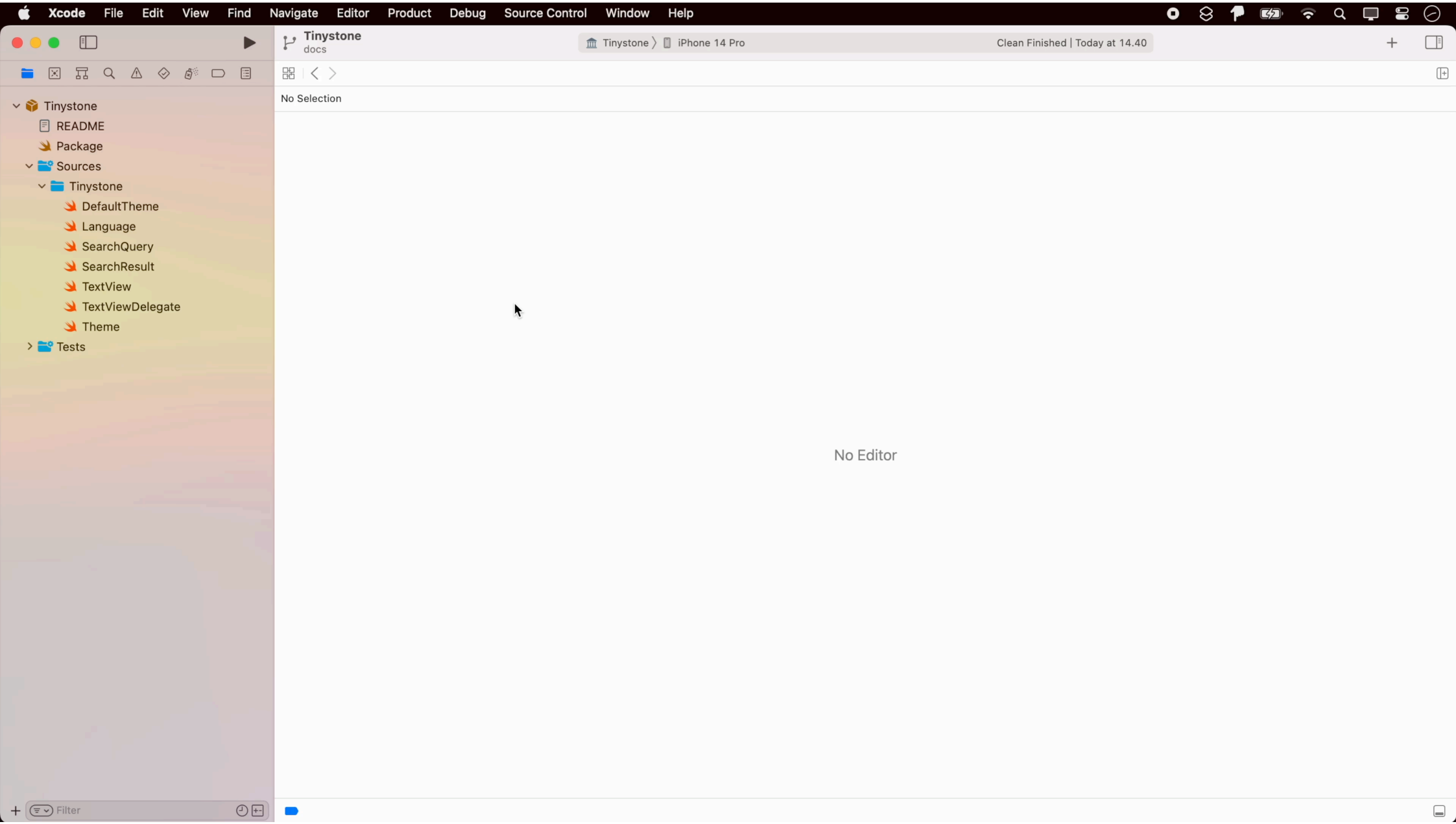

Chapter 1 Title
Symbol
Tutorial Title

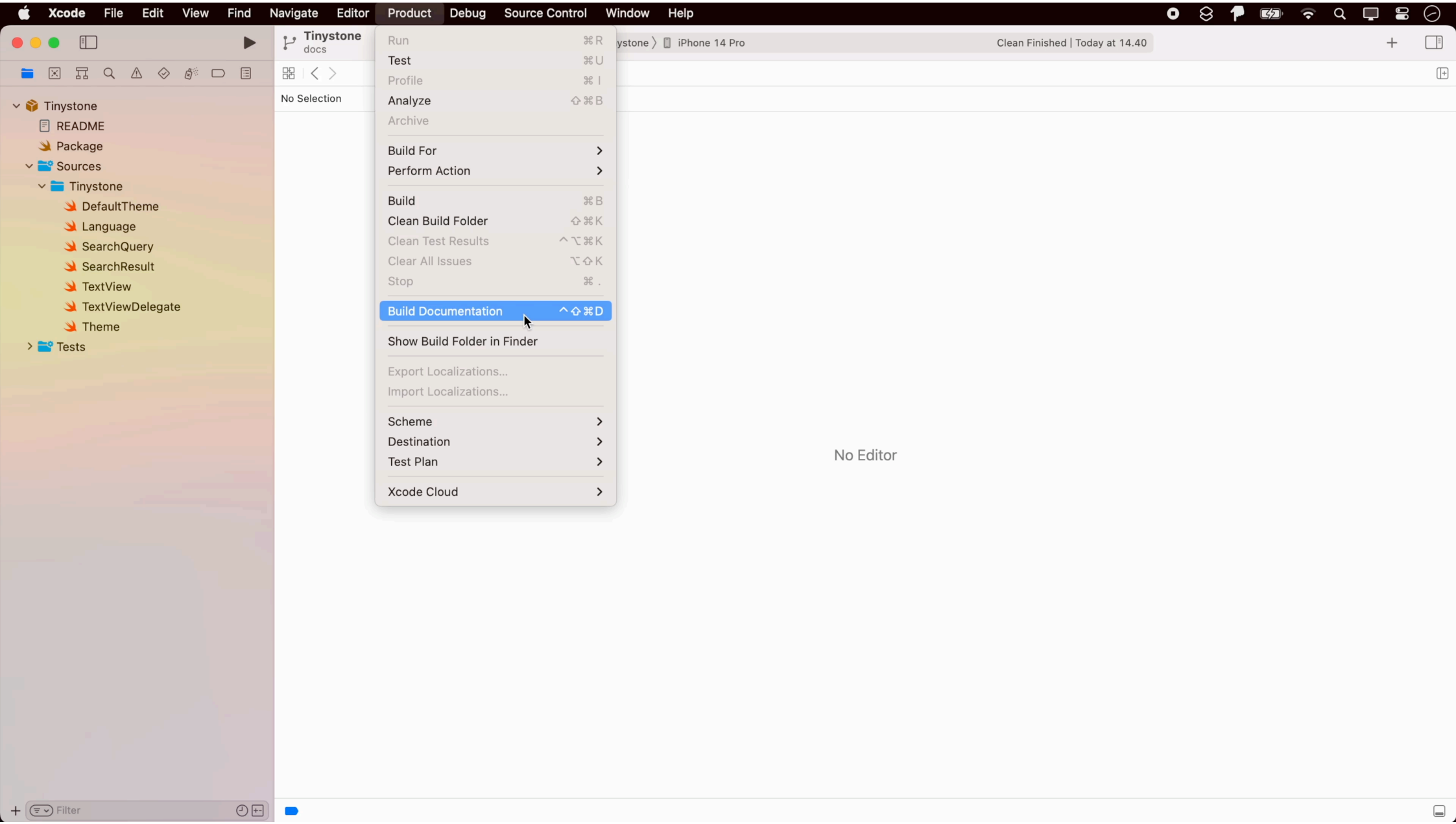

Symbols
Structure
Articles
Tutorials

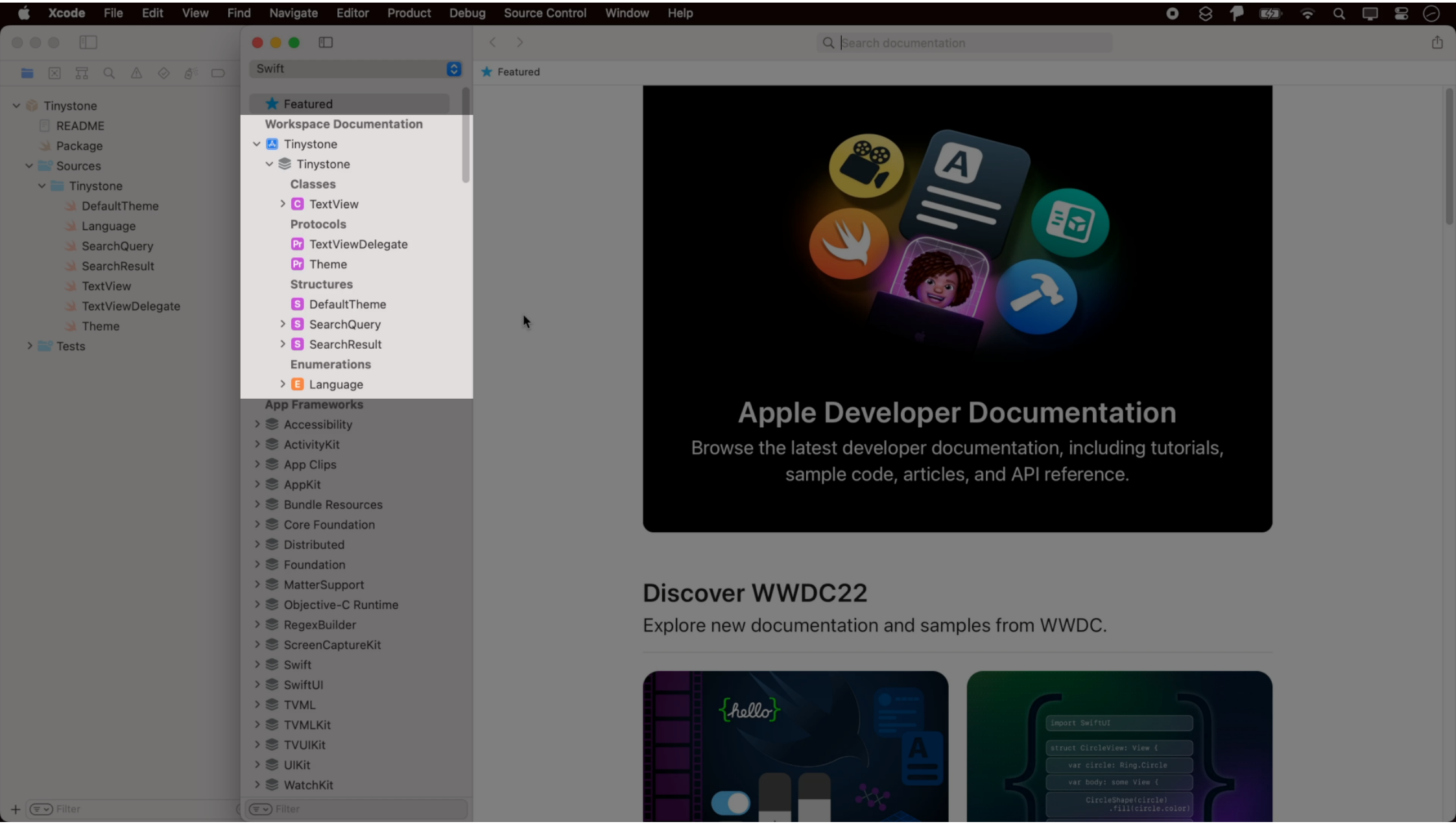
Swift, Markdown, and resources go in,
documentation comes out.

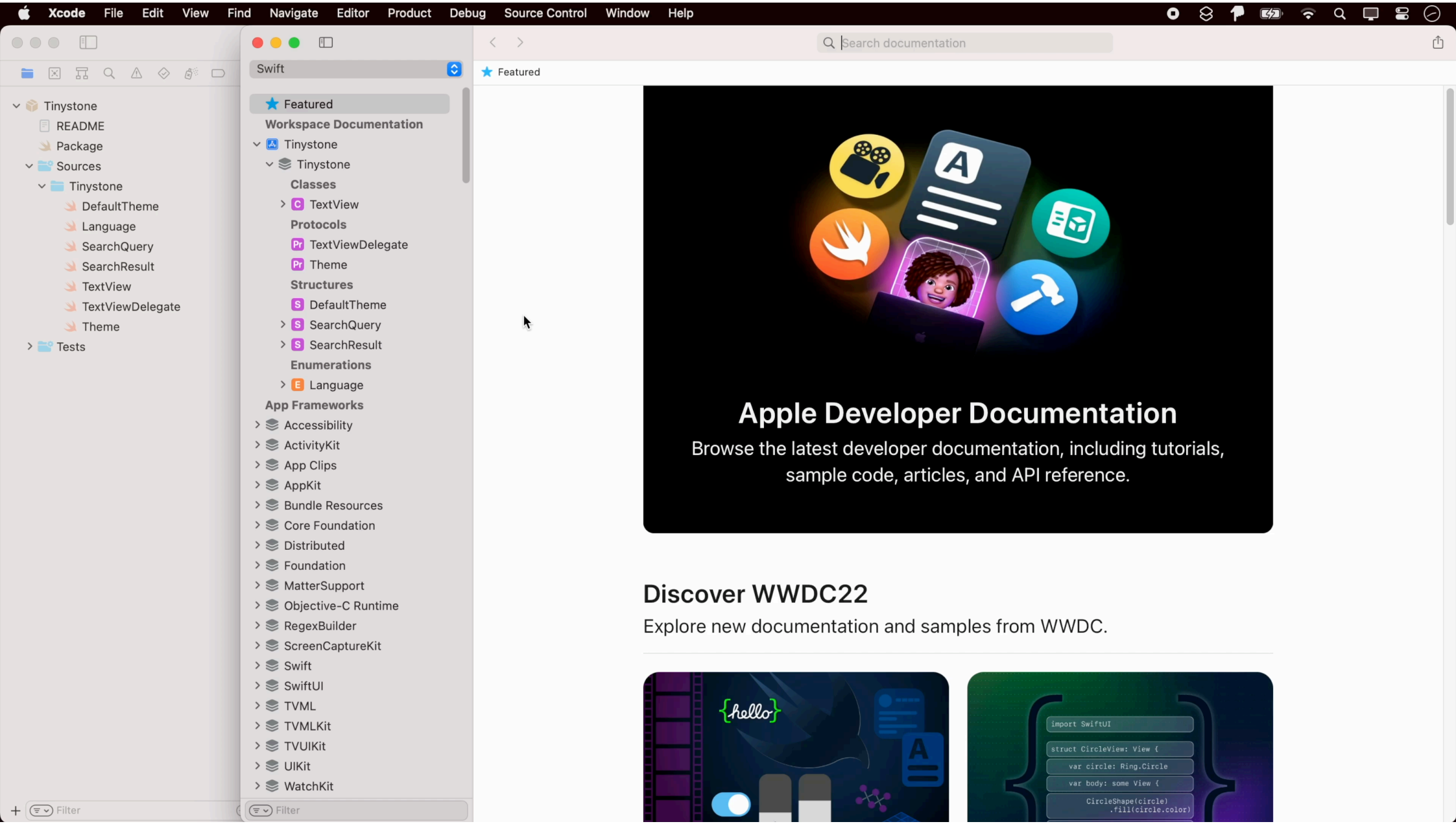

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Clean Finished | Today at 14.40
No Selection
Tinystone
README
Package
Sources
Tinystone
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
No Editor
Filter

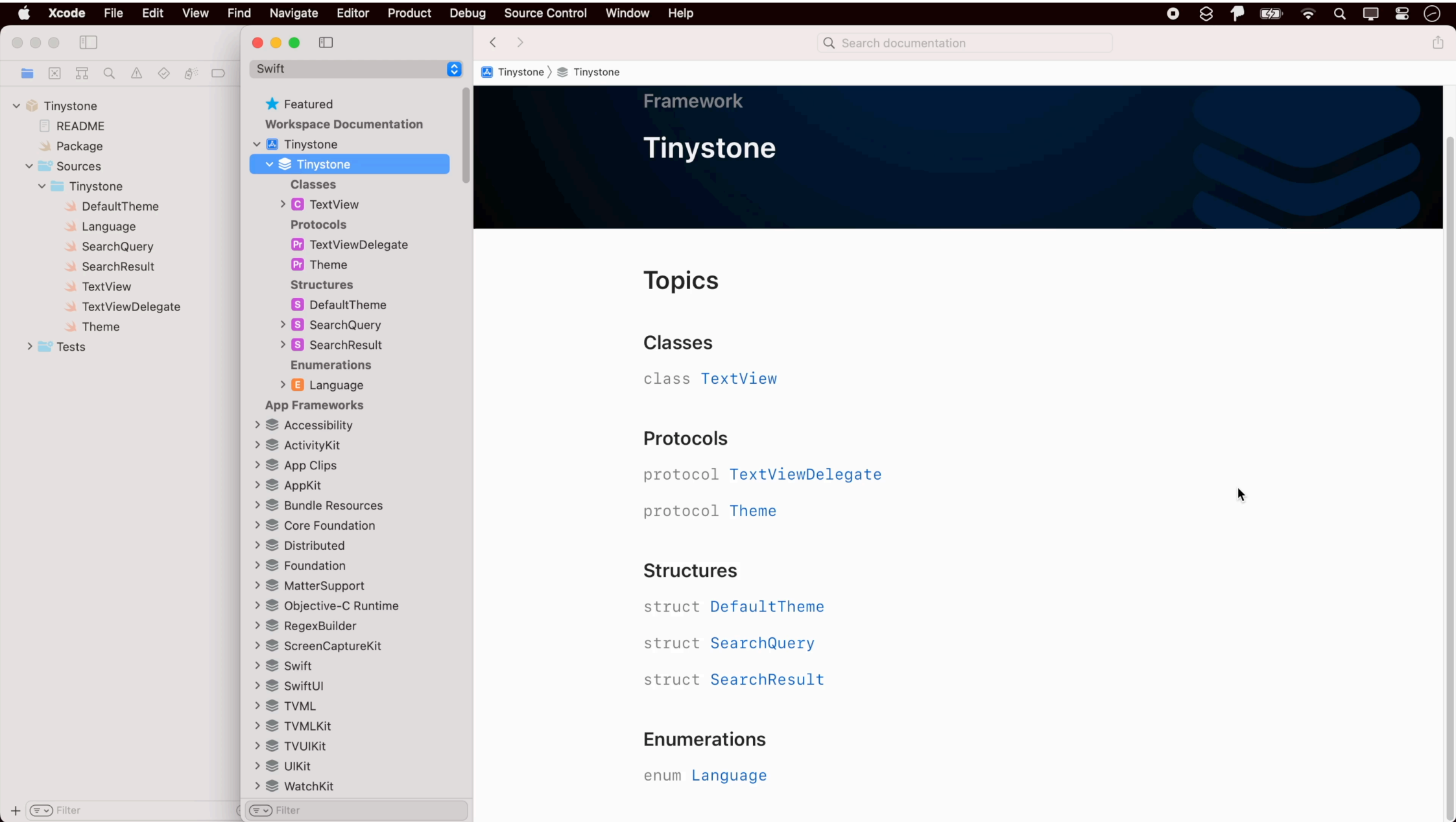

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Clean Finished | Today at 14.40
No Selection
Tinystone
README
Package
Sources
Tinystone
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
No Editor
Filter


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Clean Finished | Today at 14.40
No Selection
Tinystone
README
Package
Sources
Tinystone
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
No Editor
Filter


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Clean Finished | Today at 14.40
iPhone 14 Pro
No Selection
Tinystone
README
Package
Sources
Tinystone
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For
Perform Action
Build ⌘B
Clean Build Folder ⇧⌘K
Clean Test Results ^⌥⌘K
Clear All Issues ⌥⇧K
Stop ⌘.
Build Documentation ^⇧⌘D
Show Build Folder in Finder
Export Localizations...
Import Localizations...
Scheme
Destination
Test Plan
Xcode Cloud
No Editor
Filter


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Search documentation
Swift
Featured
Workspace Documentation
Tinystone
Tinystone
Classes
TextView
Protocols
TextViewDelegate
Theme
Structures
DefaultTheme
SearchQuery
SearchResult
Enumerations
Language
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
Tinystone
README
Package
Sources
Tinystone
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Filter
Featured
Apple Developer Documentation
Browse the latest developer documentation, including tutorials, sample code, articles, and API reference.
Discover WWDC22
Explore new documentation and samples from WWDC.
{hello}
import SwiftUI
struct CircleView: View {
var circle: Ring.Circle
var body: some View {
CircleShape(circle)
.fill(circle.color)


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
README
Package
Sources
Tinystone
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Swift
Featured
Workspace Documentation
Tinystone
Tinystone
Classes
TextView
Protocols
TextViewDelegate
Theme
Structures
DefaultTheme
SearchQuery
SearchResult
Enumerations
Language
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
Filter
Search documentation
Featured
Apple Developer Documentation
Browse the latest developer documentation, including tutorials, sample code, articles, and API reference.
Discover WWDC22
Explore new documentation and samples from WWDC.
hello
import SwiftUI
struct CircleView: View {
var circle: Ring.Circle
var body: some View {
CircleShape(circle)
.fill(circle.color)


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
README
Package
Sources
Tinystone
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Swift
Featured
Workspace Documentation
Tinystone
Tinystone
Classes
TextView
Protocols
TextViewDelegate
Theme
Structures
DefaultTheme
SearchQuery
SearchResult
Enumerations
Language
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
Filter
Search documentation
Tinystone 〉 Tinystone
Framework
Tinystone
Topics
Classes
class TextView
Protocols
protocol TextViewDelegate
protocol Theme
Structures
struct DefaultTheme
struct SearchQuery
struct SearchResult
Enumerations
enum Language

Describe Symbols

Describe Symbols

Add Structure

Describe Symbols

Add Structure

Add Supplemental Content

Describe Symbols

Add Structure

Add Supplemental Content

TextView.swift

```
func search(for query: SearchQuery) -> [SearchResult]
```

Instance Method

search(for:)

Declaration

```
func search(for query: SearchQuery) -> [SearchResult]
```

TextView.swift

```
func search(for query: SearchQuery) -> [SearchResult]
```

Instance Method

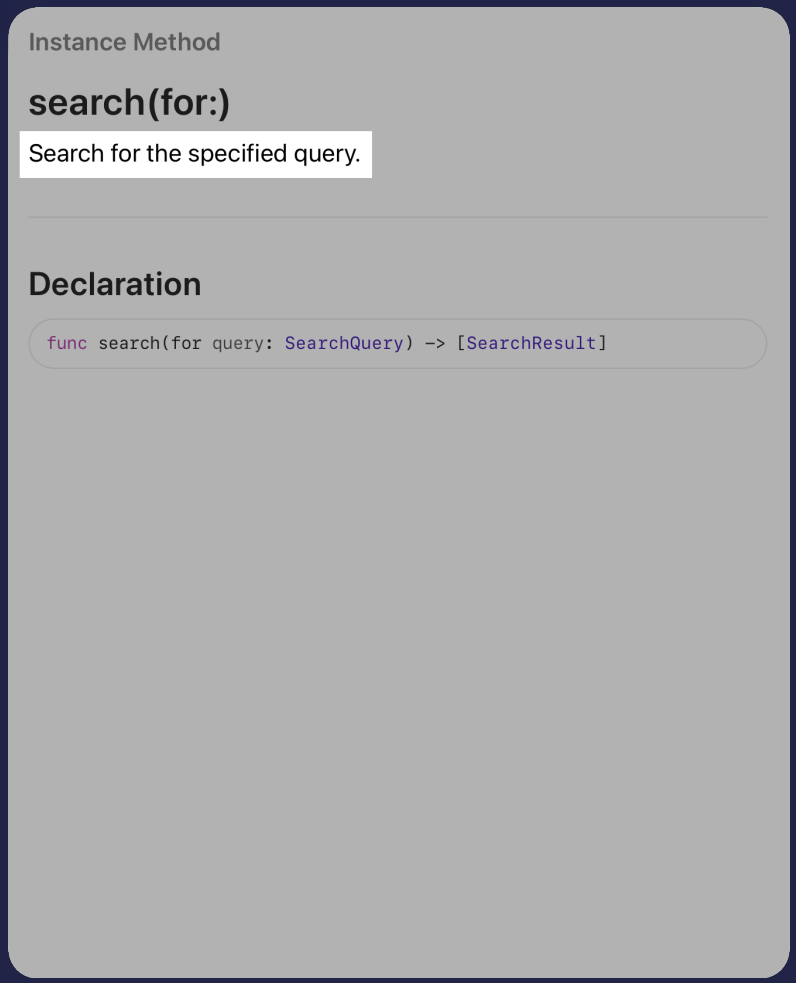
search(for:)

Declaration

```
func search(for query: SearchQuery) -> [SearchResult]
```

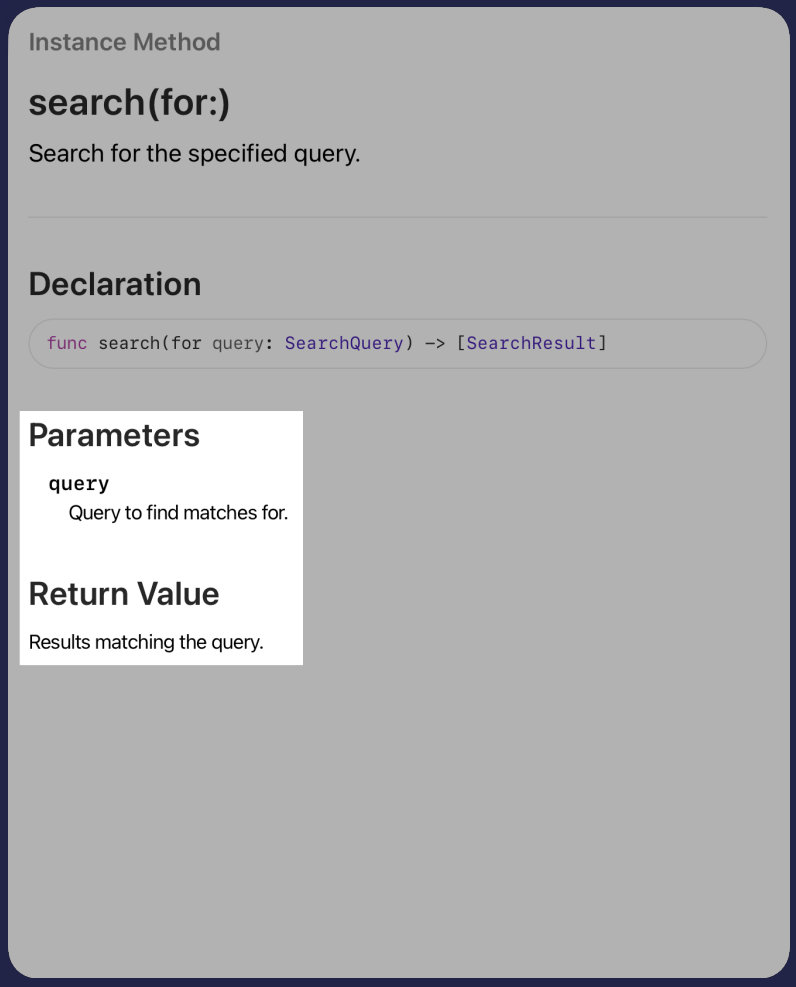
TextView.swift

```
/// Search for the specified query.
func search(for query: SearchQuery) -> [SearchResult]
```



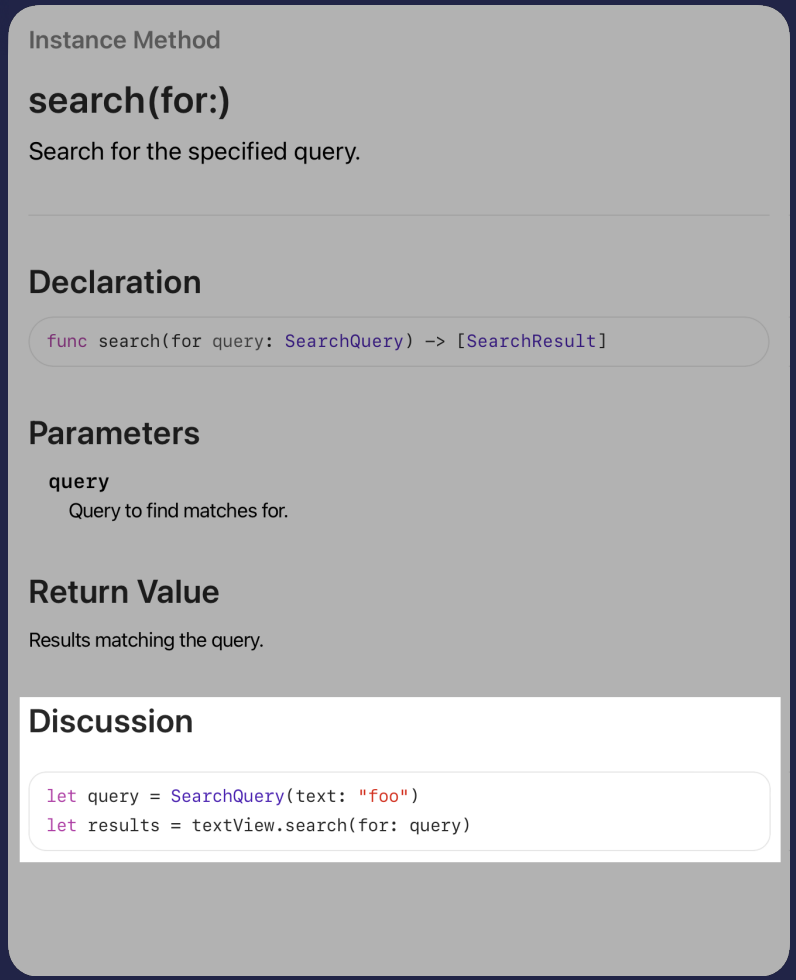
TextView.swift

```
/// Search for the specified query.
/// - Parameter query: Query to find matches for.
/// - Returns: Results matching the query.
func search(for query: SearchQuery) -> [SearchResult]
```



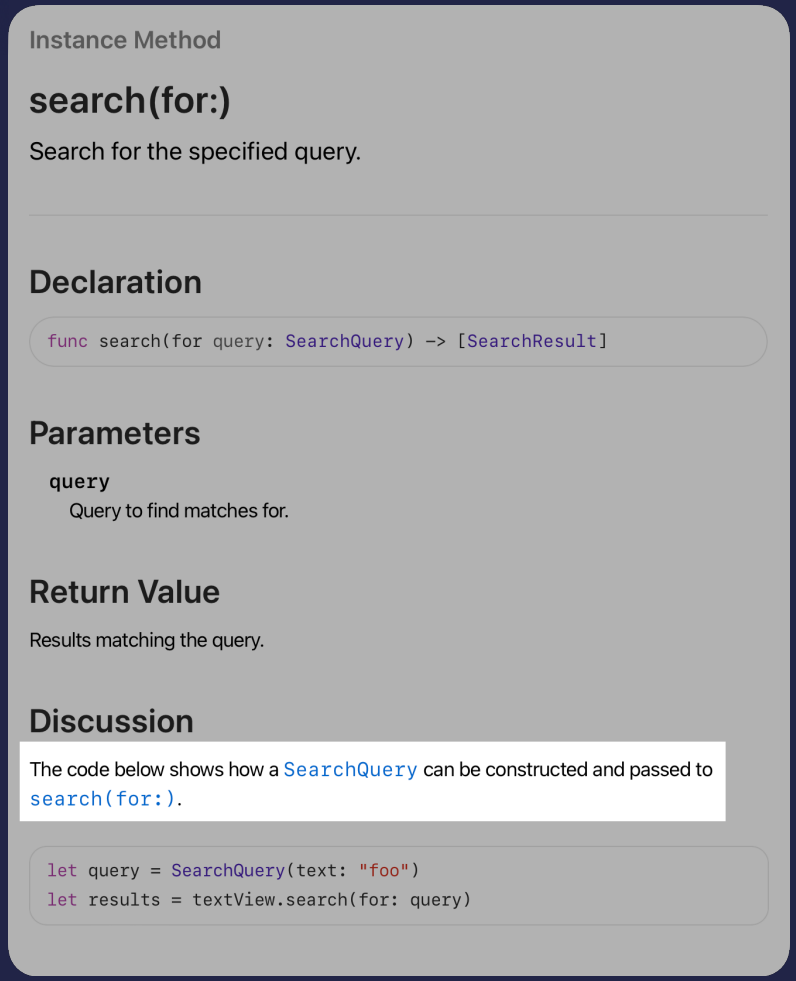
TextView.swift

```
/// Search for the specified query.
///
/// ```swift
/// let query = SearchQuery(text: "foo")
/// let results = textView.search(for: query)
/// ```
///
/// - Parameter query: Query to find matches for.
/// - Returns: Results matching the query.
func search(for query: SearchQuery) -> [SearchResult]
```



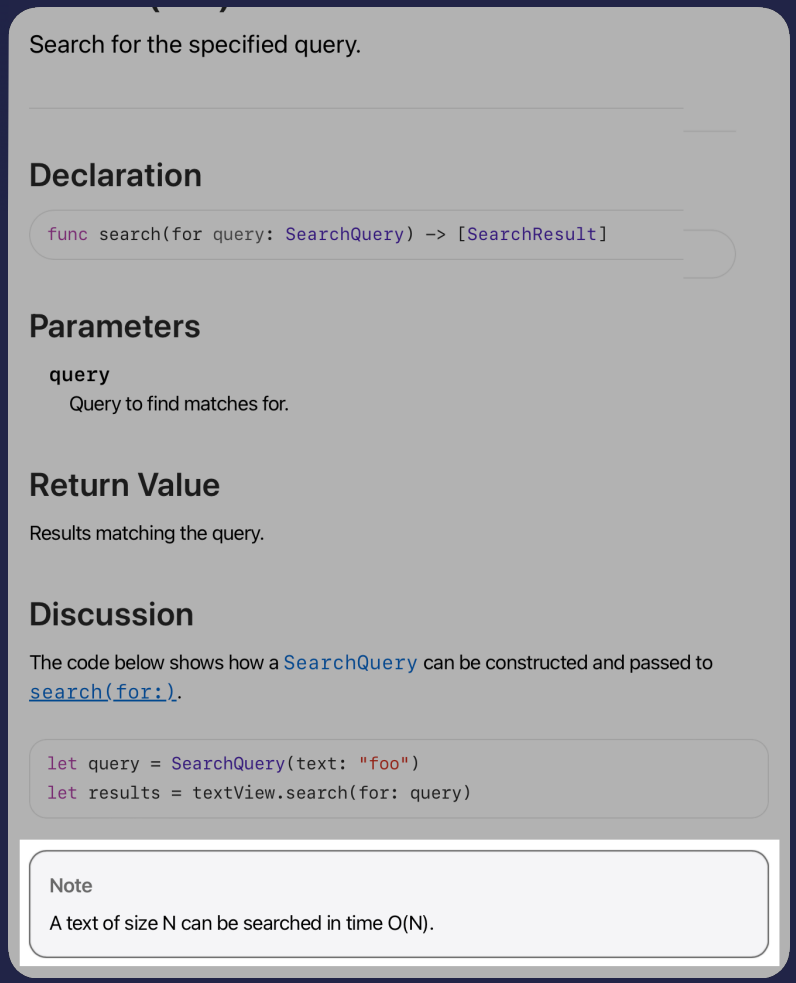
TextView.swift

```
/// Search for the specified query.
///
/// The code below shows how a ``SearchQuery`` can
/// be constructed and passed to ``search(for:)``.
///
/// ```swift
/// let query = SearchQuery(text: "foo")
/// let results = textView.search(for: query)
/// ```
///
/// - Parameter query: Query to find matches for.
/// - Returns: Results matching the query.
func search(for query: SearchQuery) -> [SearchResult]
```



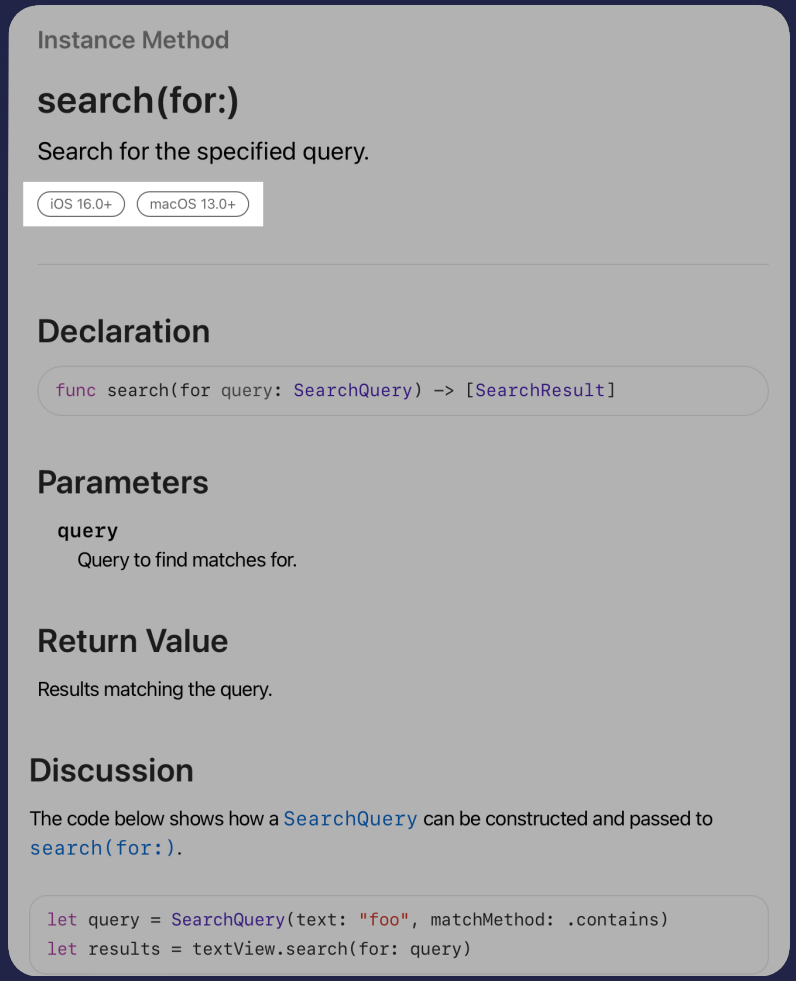
TextView.swift

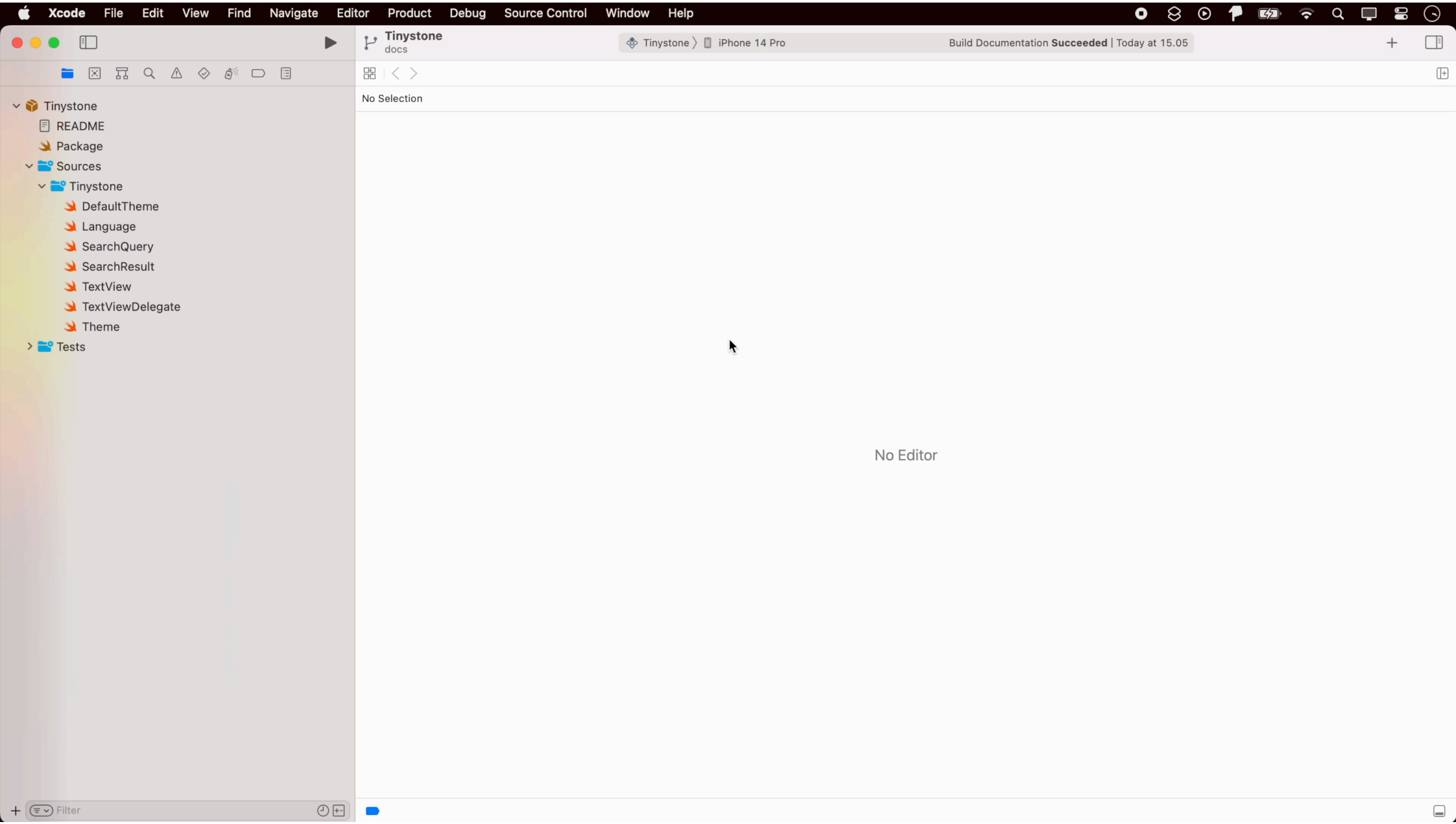
```
///
/// The code below shows how a ``SearchQuery`` can
/// be constructed and passed to ``search(for:)``.
///
/// ```swift
/// let query = SearchQuery(text: "foo")
/// let results = textView.search(for: query)
/// ```
///
/// > Note: A text of size N can be searched in time O(N).
///
/// - Parameter query: Query to find matches for.
/// - Returns: Results matching the query.
func search(for query: SearchQuery) -> [SearchResult]
```

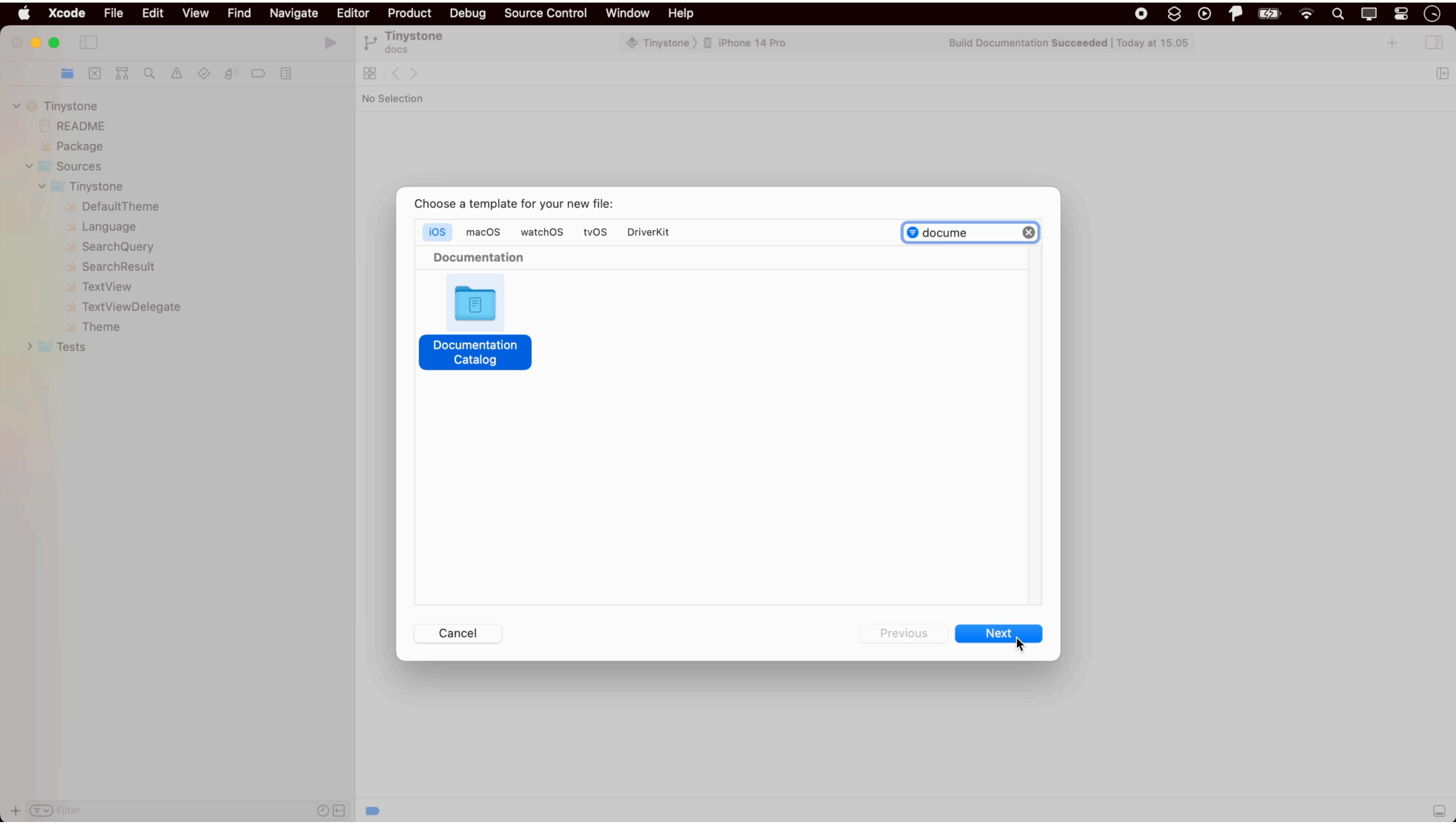


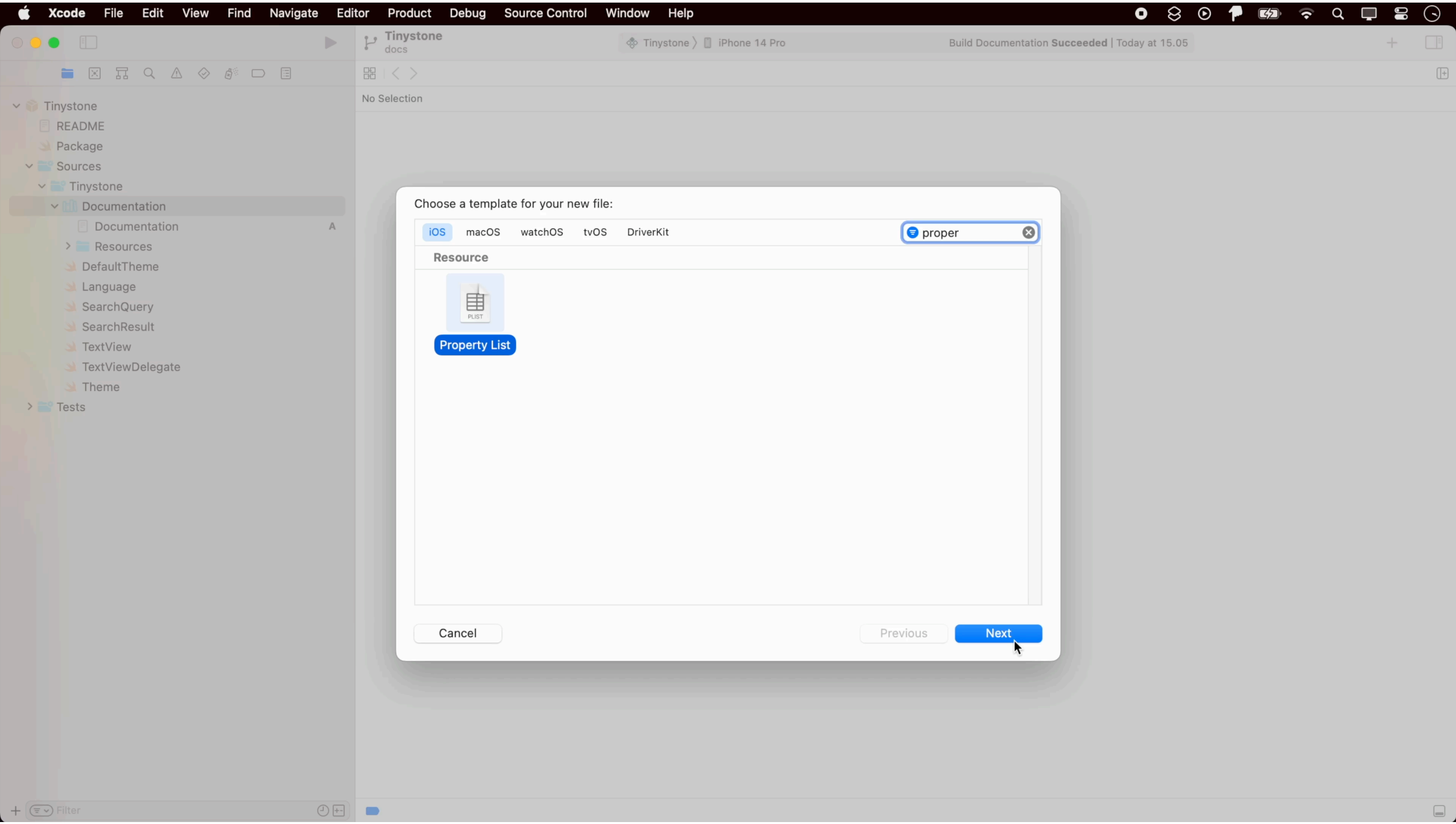
TextView.swift

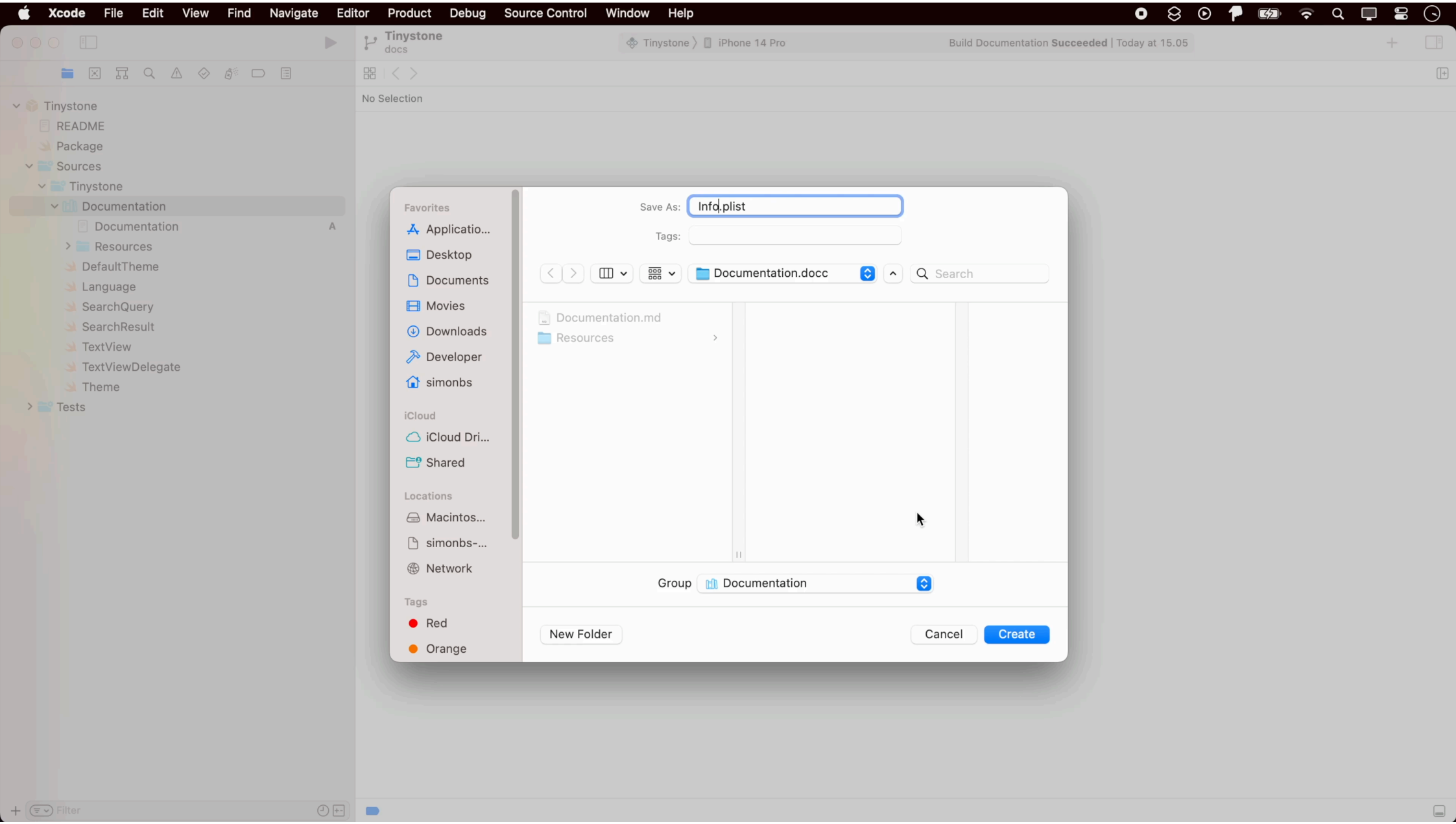
```
///
/// The code below shows how a ``SearchQuery`` can
/// be constructed and passed to ``search(for:)``.
///
/// ```swift
/// let query = SearchQuery(text: "foo")
/// let results = textView.search(for: query)
/// ```
///
/// > Note: A text of size N can be searched in time O(N).
///
/// - Parameter query: Query to find matches for.
/// - Returns: Results matching the query.
@available(iOS 16, macOS 13, *)
func search(for query: SearchQuery) -> [SearchResult]
```

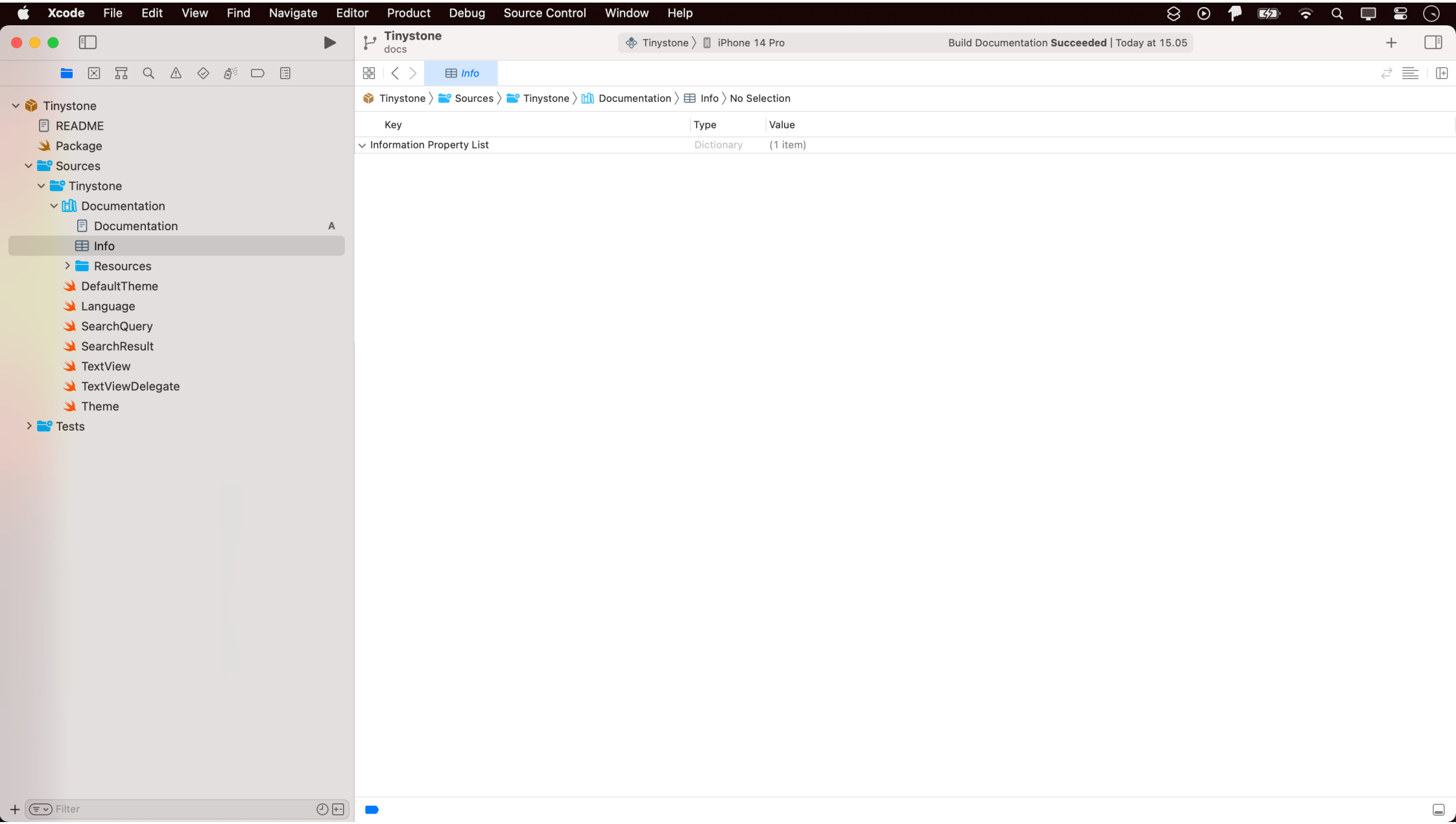


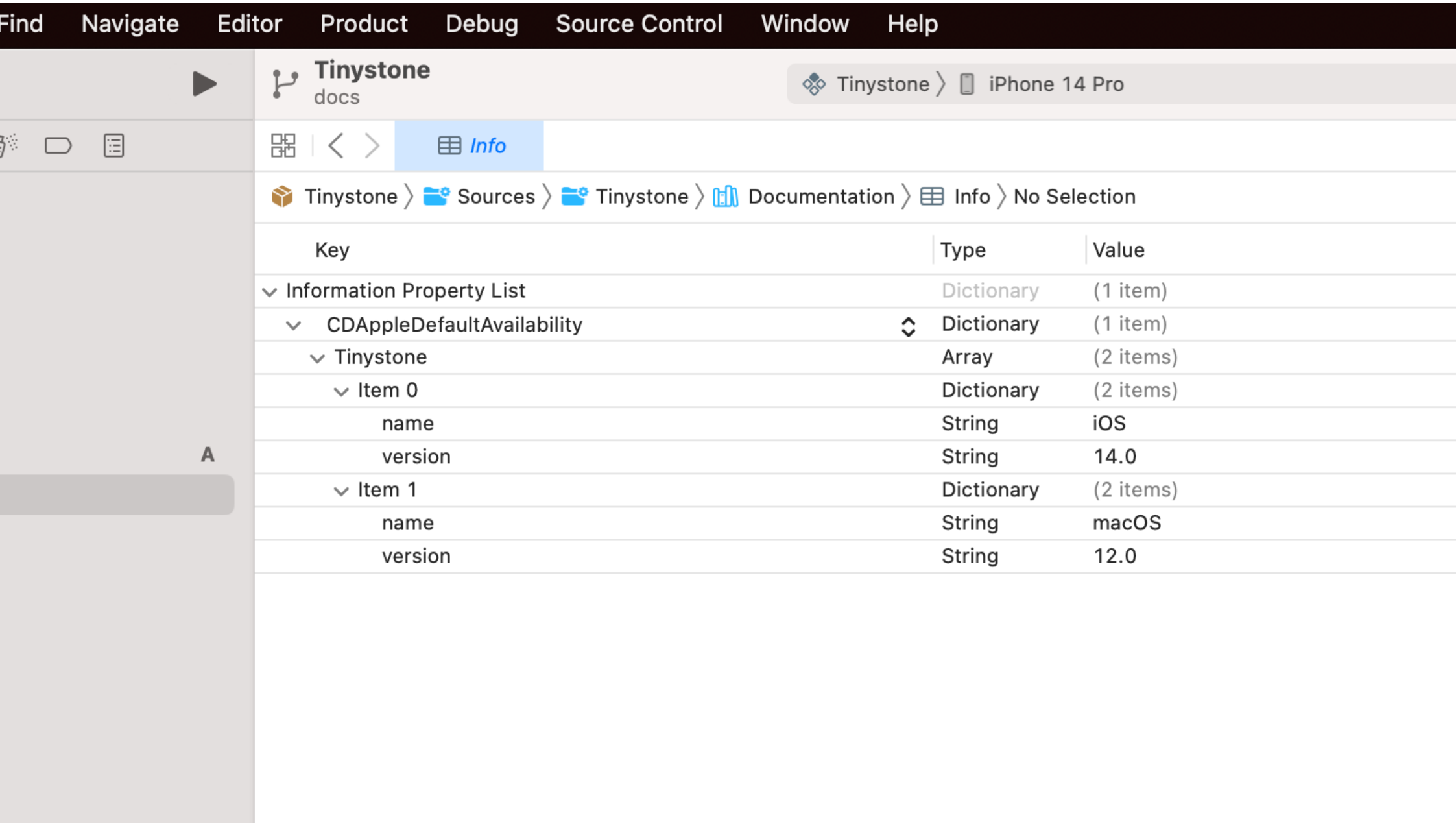

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Build Documentation Succeeded | Today at 15.05
No Selection
Tinystone
README
Package
Sources
Tinystone
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
No Editor

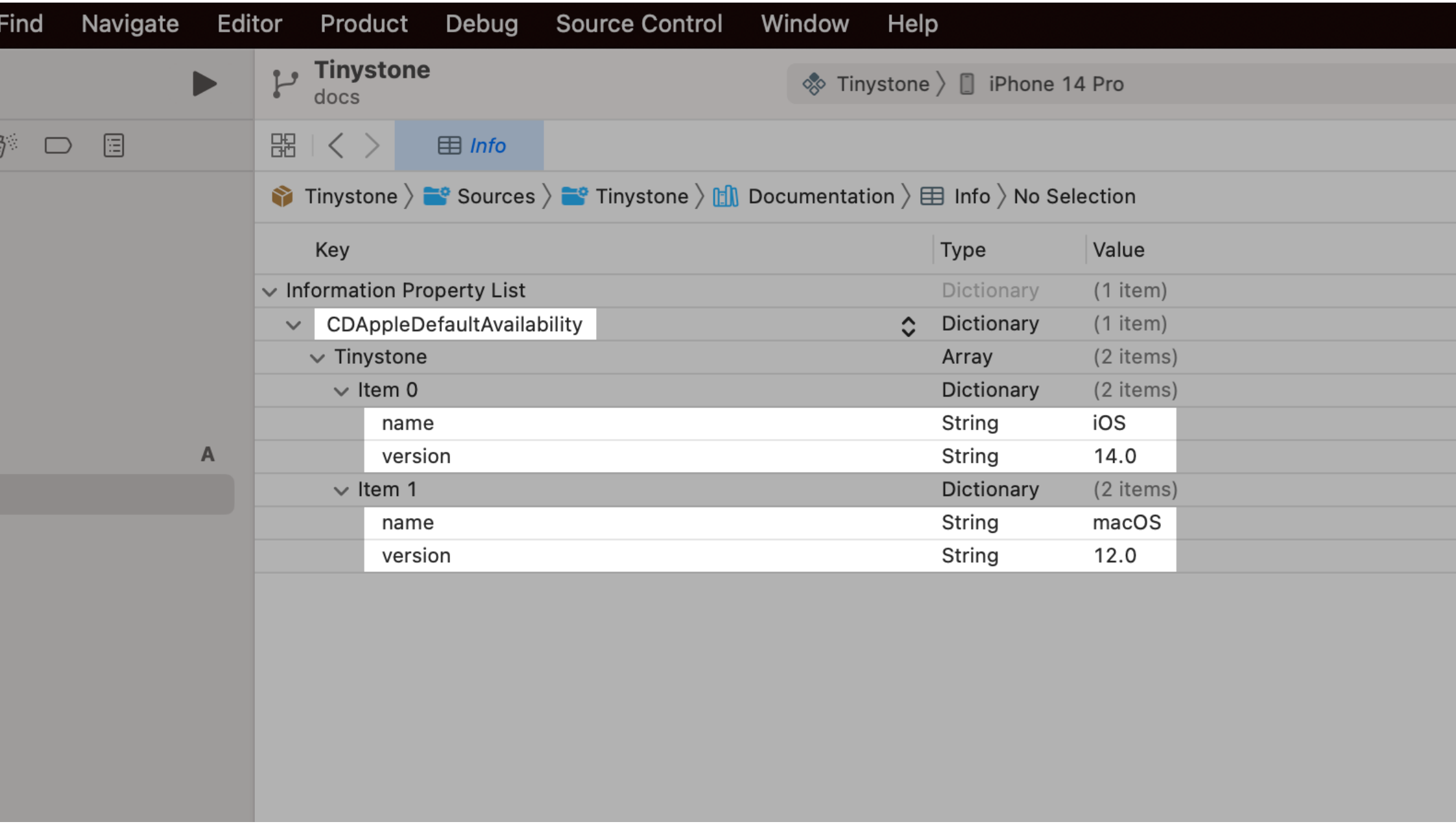

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Build Documentation Succeeded | Today at 15.05
No Selection
Tinystone
README
Package
Sources
Tinystone
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Choose a template for your new file:
iOS
macOS
watchOS
tvOS
DriverKit
docume
Documentation
Documentation Catalog
Cancel
Previous
Next

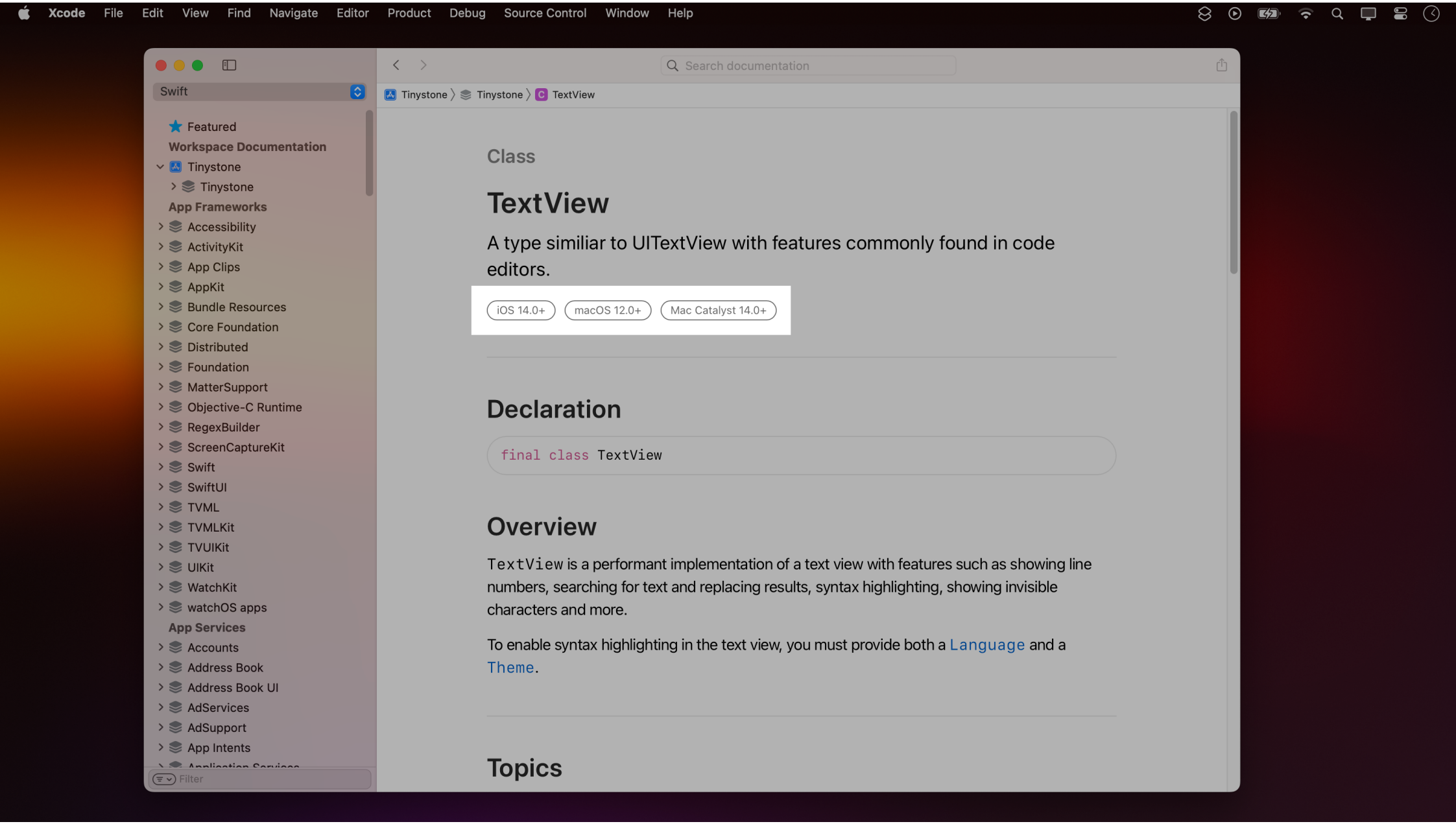

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Build Documentation Succeeded | Today at 15.05
No Selection
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Choose a template for your new file:
iOS
macOS
watchOS
tvOS
DriverKit
proper
Resource
PLIST
Property List
Cancel
Previous
Next


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone iPhone 14 Pro
Build Documentation Succeeded | Today at 15.05
No Selection
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Favorites
Applicatio...
Desktop
Documents
Movies
Downloads
Developer
simonbs
iCloud
iCloud Dri...
Shared
Locations
Macintos...
simonbs-...
Network
Tags
Red
Orange
Save As: Info.plist
Tags:
Documentation.docc
Search
Documentation.md
Resources
Group Documentation
New Folder
Cancel
Create


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Build Documentation Succeeded | Today at 15.05
Info
Tinystone 〉 Sources 〉 Tinystone 〉 Documentation 〉 Info 〉 No Selection
Key
Type
Value
Information Property List
Dictionary
(1 item)
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Info
Resources
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter

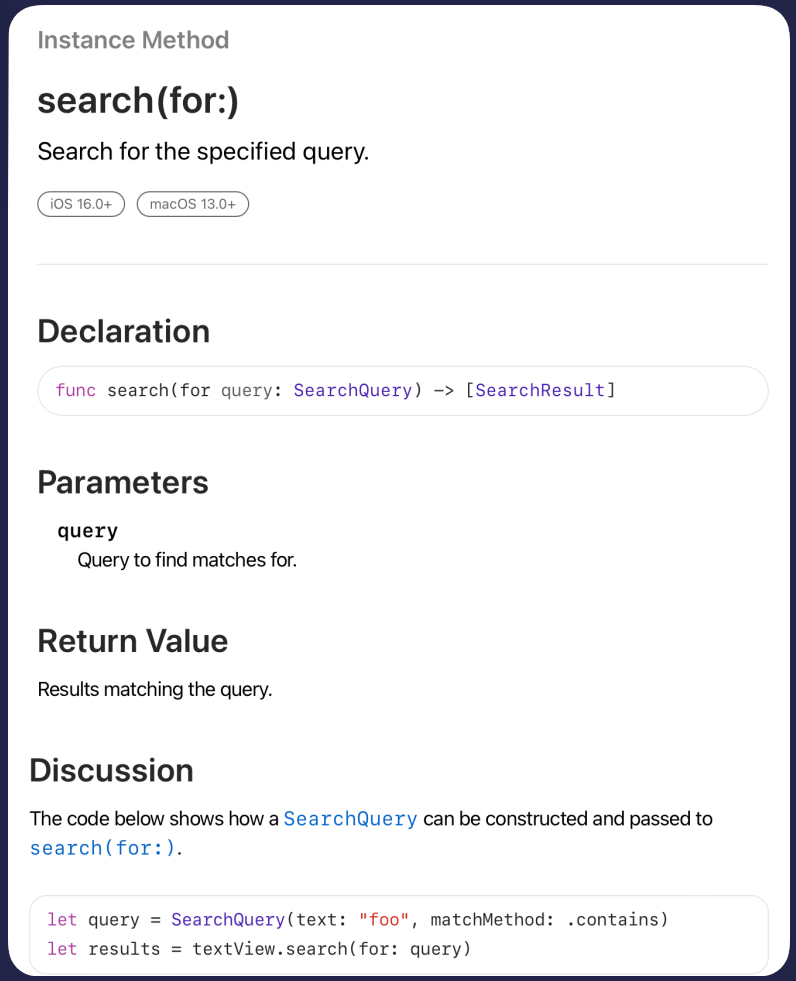

Find
Navigate
Editor
Product
Debug
Source Control
Window
Help
Tinystone
docs
Tinystone
iPhone 14 Pro
Info
Tinystone
Sources
Tinystone
Documentation
Info
No Selection
Key
Type
Value
Information Property List
Dictionary
(1 item)
CDAppleDefaultAvailability
Dictionary
(1 item)
Tinystone
Array
(2 items)
Item 0
Dictionary
(2 items)
name
String
iOS
version
String
14.0
Item 1
Dictionary
(2 items)
name
String
macOS
version
String
12.0


Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Info
Tinystone 〉 Sources 〉 Tinystone 〉 Documentation 〉 Info 〉 No Selection
Key
Type
Value
Information Property List
Dictionary
(1 item)
CDAppleDefaultAvailability
Dictionary
(1 item)
Tinystone
Array
(2 items)
Item 0
Dictionary
(2 items)
name
String
iOS
version
String
14.0
Item 1
Dictionary
(2 items)
name
String
macOS
version
String
12.0


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Swift
Featured
Workspace Documentation
Tinystone
Tinystone
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
watchOS apps
App Services
Accounts
Address Book
Address Book UI
AdServices
AdSupport
App Intents
Filter
Search documentation
Tinystone 〉 Tinystone 〉 C TextView
Class
TextView
A type similiar to UITextView with features commonly found in code editors.
iOS 14.0+
macOS 12.0+
Mac Catalyst 14.0+
Declaration
final class TextView
Overview
TextView is a performant implementation of a text view with features such as showing line numbers, searching for text and replacing results, syntax highlighting, showing invisible characters and more.
To enable syntax highlighting in the text view, you must provide both a Language and a Theme.
Topics

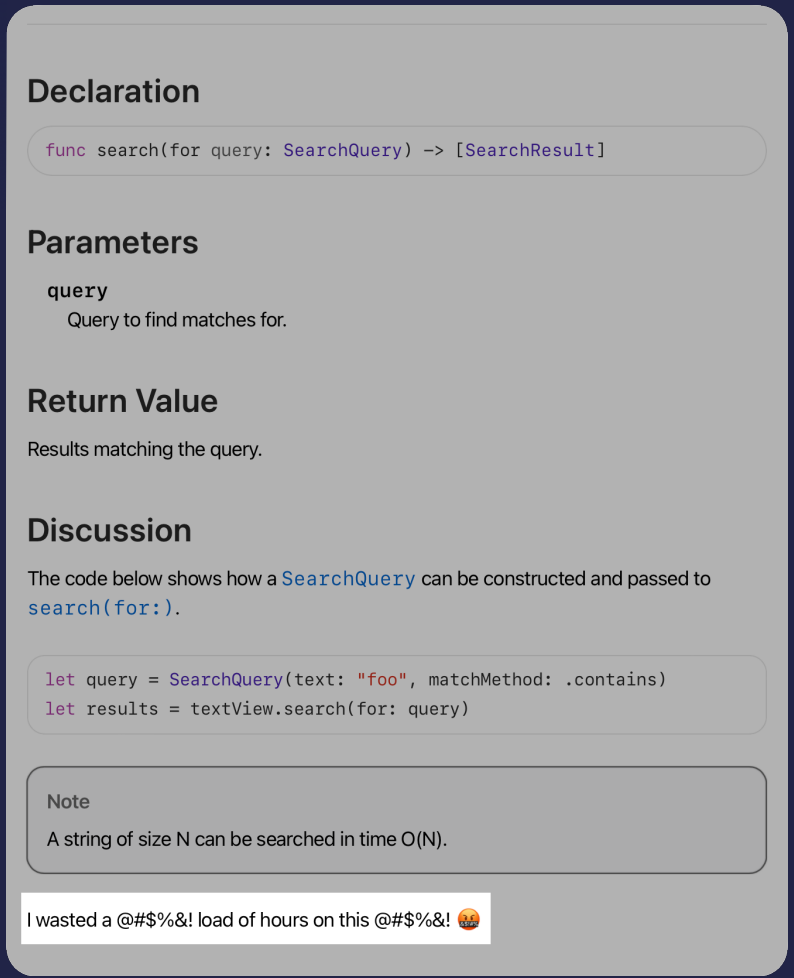
TextView.swift

```swift
///
/// The code below shows how a ``SearchQuery`` can
/// be constructed and passed to ``search(for:)``.
///
/// ```swift
/// let query = SearchQuery(text: "foo")
/// let results = textView.search(for: query)
/// ```
///
/// > Note: A text of size N can be searched in time O(N).
///
/// - Parameter query: Query to find matches for.
/// - Returns: Results matching the query.
@available(iOS 16, macOS 13, *)
func search(for query: SearchQuery) -> [SearchResult]
```



TextView.swift

```swift
/// be constructed and passed to `search(for:)`.
///
/// ```swift
/// let query = SearchQuery(text: "foo")
/// let results = textView.search(for: query)
/// ```
///
/// > Note: A text of size N can be searched in time O(N).
///
/// I wasted a @#$%&! load of hours on this @#$%&! 🤬
///
/// - Parameter query: Query to find matches for.
/// - Returns: Results matching the query.
@available(iOS 16, macOS 13, *)
func search(for query: SearchQuery) -> [SearchResult]
```



TextView.swift

```swift
/// be constructed and passed to `search(for:)`.
///
/// ```swift
/// let query = SearchQuery(text: "foo")
/// let results = textView.search(for: query)
/// ```
///
/// > Note: A text of size N can be searched in time O(N).
///
/// - Parameter query: Query to find matches for.
/// - Returns: Results matching the query.
@available(iOS 16, macOS 13, *)
func search(for query: SearchQuery) -> [SearchResult] {
  // I wasted a @#$%&! load of hours on this @#$%&! 😡
}
```

Instance Method

search(for:)

Search for the specified query.

iOS 16.0+ macOS 13.0+

Declaration

```
func search(for query: SearchQuery) -> [SearchResult]
```

Parameters

query
Query to find matches for.

Return Value

Results matching the query.

Discussion

The code below shows how a `SearchQuery` can be constructed and passed to `search(for:)`.

```
let query = SearchQuery(text: "foo", matchMethod: .contains)
let results = textView.search(for: query)
```

Describe Symbols

Add Structure

Add Supplemental Content

Describe Symbols

Add Structure

Add Supplemental Content

Framework

Tinystone

Performant code editor with syntax highlighting, line numbers, and invisible characters.

Overview

Tinystone provides a text view with support for features commonly found in code editors. This includes the following list of features.

- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as tabs, spaces, and line breaks.
- Adjust the height of lines.

Topics

Essentials

Meet Tinystone
Explore Tinystone, the performant code editor with syntax highlighting, line numbers, and invisible characters through a series of interactive tutorials.

`class` TextView
A type similiar to UITextView with features commonly found in code editors.

`protocol` TextViewDelegate
The methods for receiving editing-related messages for the text view.

Syntax Highlighting

`enum` Language
Language that can be used to syntax highlight text in the text view.

`protocol` Theme
Fonts and colors to be used by a `TextView`.

`struct` DefaultTheme
The theme used by a text view when no other theme has been specified.

Find and Replace

`struct` SearchQuery
Query to search for in the text view.

`struct` SearchResult
A match returned by performing a search query.

Landing Page

Class

TextView

A type similiar to UITextView with features commonly found in code editors.

Declaration

```
final class TextView
```

Overview

`TextView` is a performant implementation of a text view with features such as showing line numbers, searching for text and replacing results, syntax highlighting, showing invisible characters and more.

To enable syntax highlighting in the text view, you must provide both a `Language` and a `Theme`.

Topics

Essentials

`init(frame: CGRect)`
Create a new text view.

`var text: String`
The text that the text view displays.

`var textViewDelegate: TextViewDelegate?`
The object to receive editing callbacks.

`protocol TextViewDelegate`
The methods for receiving editing-related messages for the text view.

Text Layout

`var isLineWrappingEnabled: Bool`
When line wrapping is disabled, users can scroll the text view horizontally to see the entire line.

`var showInvisibleCharacters: Bool`
Enable to show invisible characters.

`var lineHeightMultiplier: CGFloat`
The line-height is multiplied with the value.

`var kern: CGFloat`
The number of points by which to adjust kern.

Syntax Highlighting

`var language: Language`
The language used for syntax highlighting text in the text view.

`var theme: Theme`
Fonts and colors used when syntax highlighting text.

Gutter

`var showLineNumbers: Bool`
Enable to show line numbers in the gutter.

`var gutterWidth: CGFloat`
Width of the gutter.

Extensions

Framework

Tinystone

Performant code editor with syntax highlighting, line numbers, and invisible characters.

Overview

Tinystone provides a text view with support for features commonly found in code editors. This includes the following list of features.

- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as tabs, spaces, and line breaks.
- Adjust the height of lines.

Topics

Essentials

Meet Tinystone

Explore Tinystone, the performant code editor with syntax highlighting, line numbers, and invisible characters through a series of interactive tutorials.

`class TextView`

A type similiar to UITextView with features commonly found in code editors.

Topics

Essentials

Meet Tinystone

Explore Tinystone, the performant code editor with syntax highlighting, line numbers, and invisible characters through a series of interactive tutorials.

`class TextView`

A type similiar to UITextView with features commonly found in code editors.

`protocol TextViewDelegate`

The methods for receiving editing-related messages for the text view.

Syntax Highlighting

`enum Language`

Language that can be used to syntax highlight text in the text view.

`protocol Theme`

Fonts and colors to be used by a `TextView`.

`struct DefaultTheme`

The theme used by a text view when no other theme has been specified.

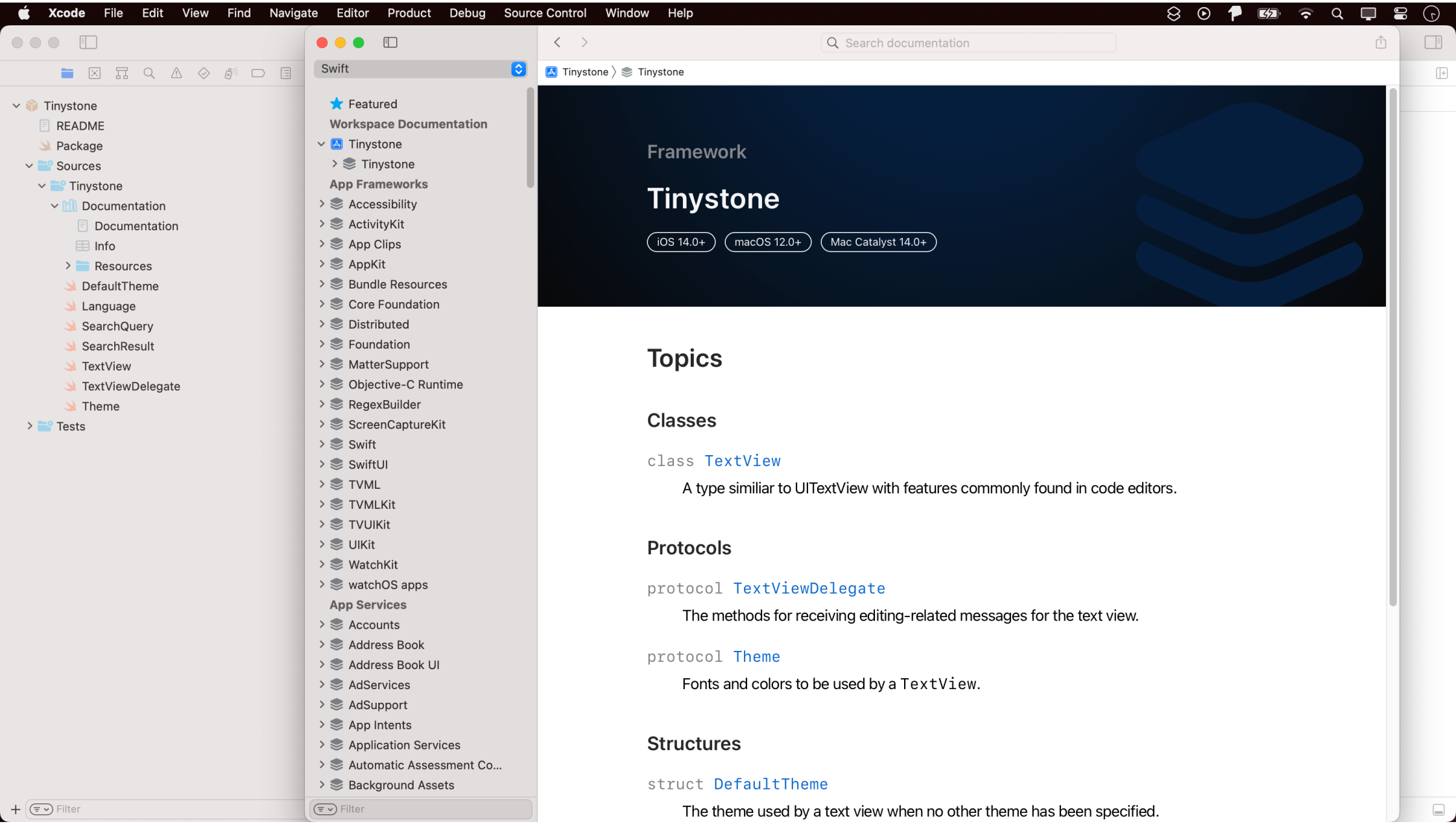
Find and Replace

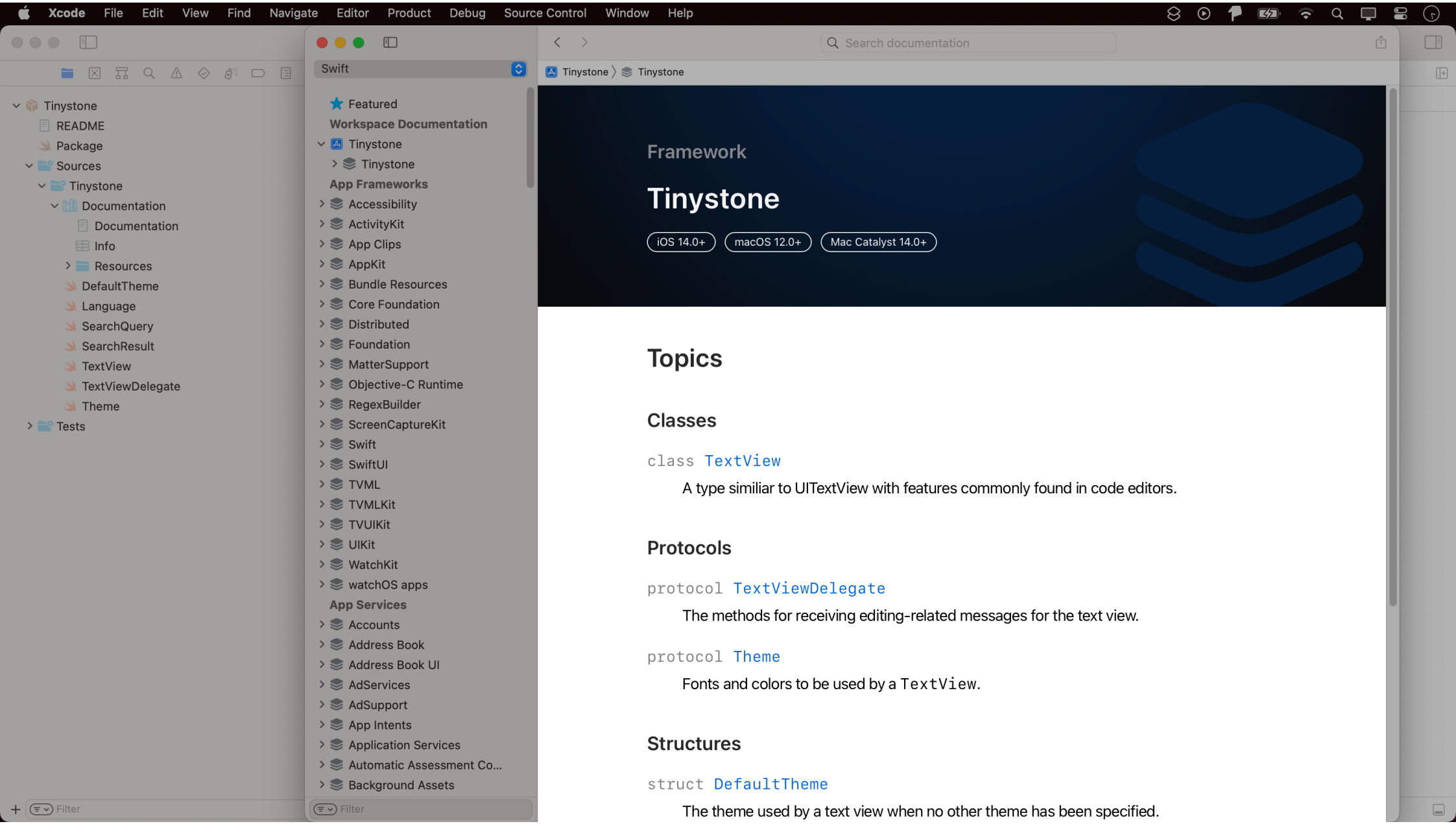
`struct SearchQuery`

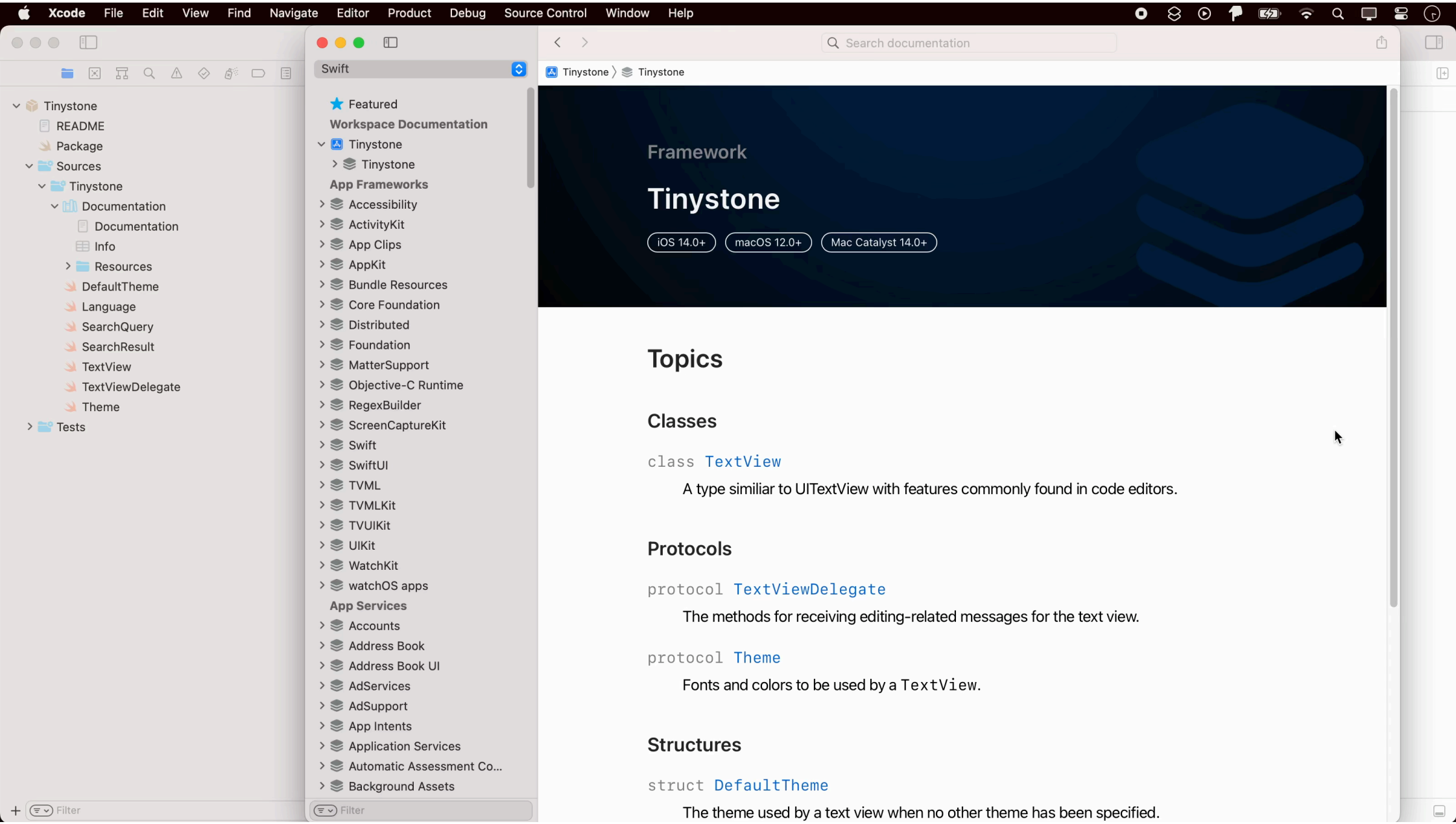
Query to search for in the text view.

`struct SearchResult`

A match returned by performing a search query.


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Info
Resources
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Swift
Featured
Workspace Documentation
Tinystone
Tinystone
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
watchOS apps
App Services
Accounts
Address Book
Address Book UI
AdServices
AdSupport
App Intents
Application Services
Automatic Assessment Co...
Background Assets
Filter
Search documentation
Tinystone
Tinystone
Framework
Tinystone
iOS 14.0+
macOS 12.0+
Mac Catalyst 14.0+
Topics
Classes
class TextView
A type similiar to UITextView with features commonly found in code editors.
Protocols
protocol TextViewDelegate
The methods for receiving editing-related messages for the text view.
protocol Theme
Fonts and colors to be used by a TextView.
Structures
struct DefaultTheme
The theme used by a text view when no other theme has been specified.


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Search documentation
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Info
Resources
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Swift
Featured
Workspace Documentation
Tinystone
Tinystone
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
watchOS apps
App Services
Accounts
Address Book
Address Book UI
AdServices
AdSupport
App Intents
Application Services
Automatic Assessment Co...
Background Assets
Filter
Tinystone Tinystone
Framework
Tinystone
iOS 14.0+
macOS 12.0+
Mac Catalyst 14.0+
Topics
Classes
class TextView
A type similiar to UITextView with features commonly found in code editors.
Protocols
protocol TextViewDelegate
The methods for receiving editing-related messages for the text view.
protocol Theme
Fonts and colors to be used by a TextView.
Structures
struct DefaultTheme
The theme used by a text view when no other theme has been specified.


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Info
Resources
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Swift
Featured
Workspace Documentation
Tinystone
Tinystone
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
watchOS apps
App Services
Accounts
Address Book
Address Book UI
AdServices
AdSupport
App Intents
Application Services
Automatic Assessment Co...
Background Assets
Filter
Search documentation
Tinystone 〉 Tinystone
Framework
Tinystone
iOS 14.0+
macOS 12.0+
Mac Catalyst 14.0+
Topics
Classes
class TextView
A type similiar to UITextView with features commonly found in code editors.
Protocols
protocol TextViewDelegate
The methods for receiving editing-related messages for the text view.
protocol Theme
Fonts and colors to be used by a TextView.
Structures
struct DefaultTheme
The theme used by a text view when no other theme has been specified.

```
# ````

Summary

## Overview

Text

## Topics

### Groups

- ``Symbol``
```

Documentation.md

```
# ````

Summary

## Overview

Text

## Topics

### Groups

- ``Symbol``
```

Documentation.md

```
# ``Tinystone``

Summary

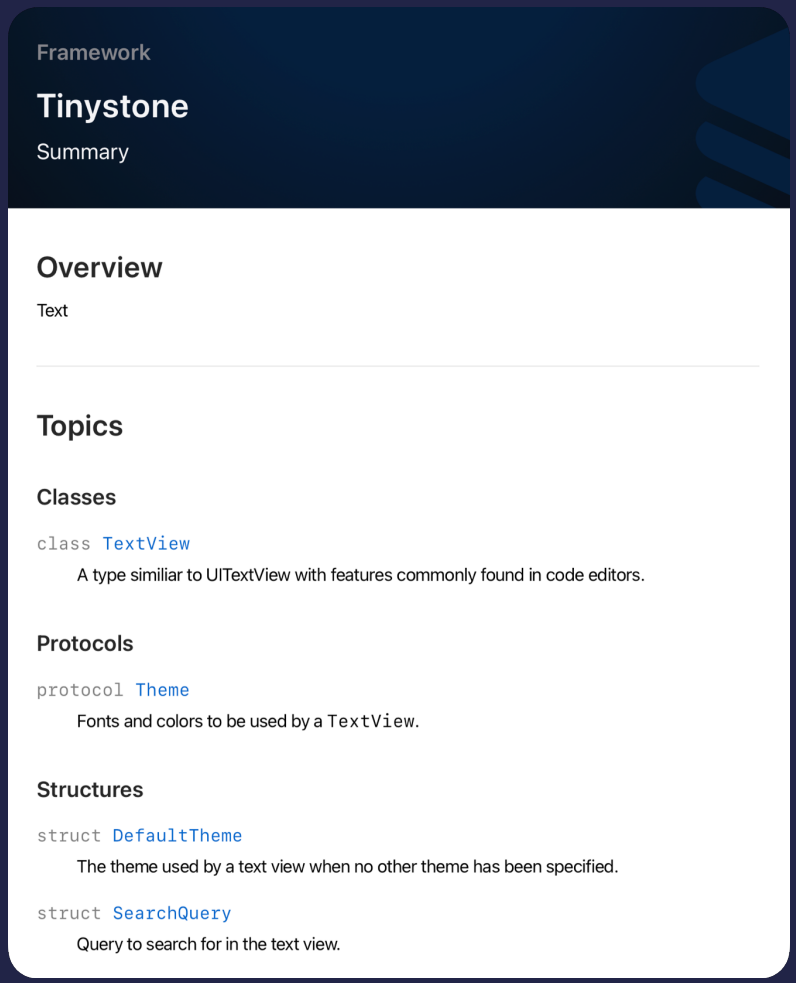
## Overview

Text

## Topics

### Groups

- ``Symbol``
```



Documentation.md

```
# ``Tinystone``

Performant code editor with syntax highlighting, line numbers, and invisible characters.

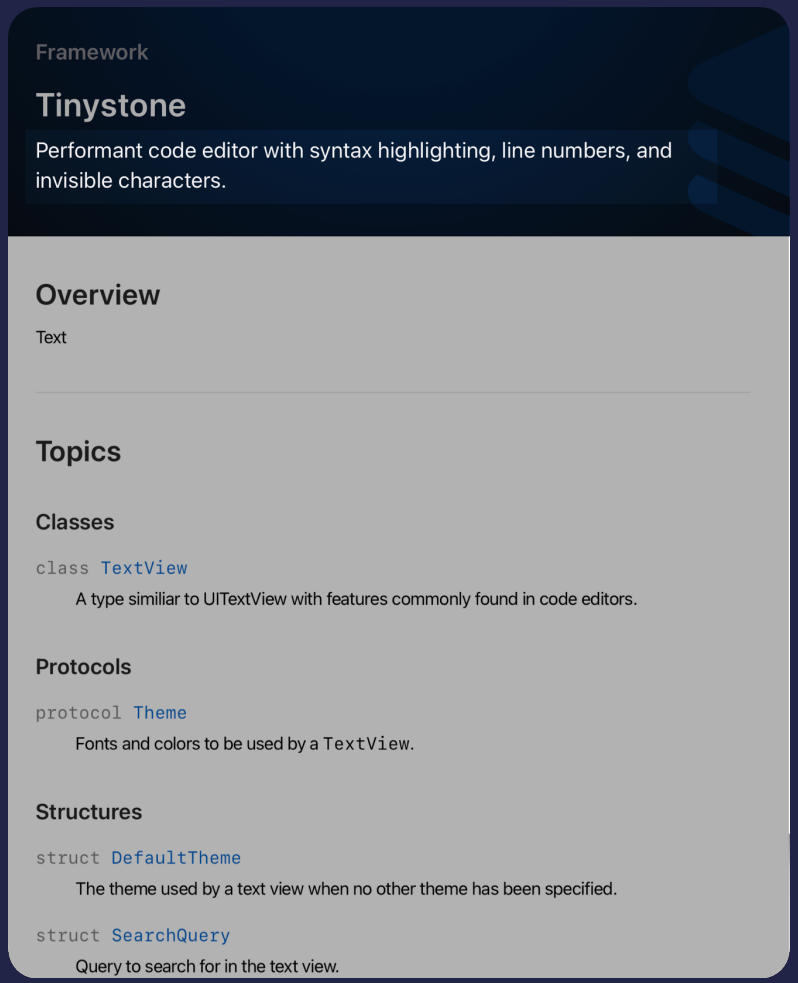
## Overview

Text

## Topics

### Groups

- ``Symbol``
```



Documentation.md

```
# ``Tinystone``

Performant code editor with syntax highlighting, line numbers, and invisible characters.

## Overview

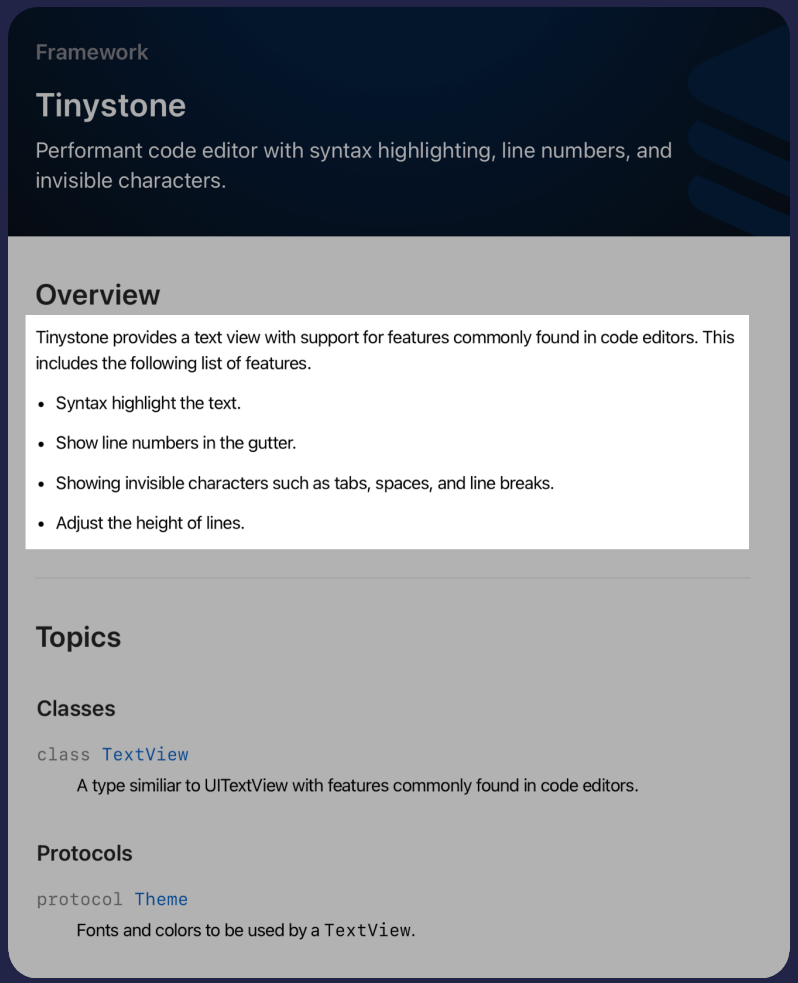
Tinystone provides a text view with support for features commonly found in code editors. This includes the following list of features.

- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as tabs, spaces, and line breaks.
- Adjust the height of lines.

## Topics

### Groups

- ``Symbol``
```



```
# ``Tinystone``

Performant code editor with syntax highlighting, line numbers,
and invisible characters.

## Overview

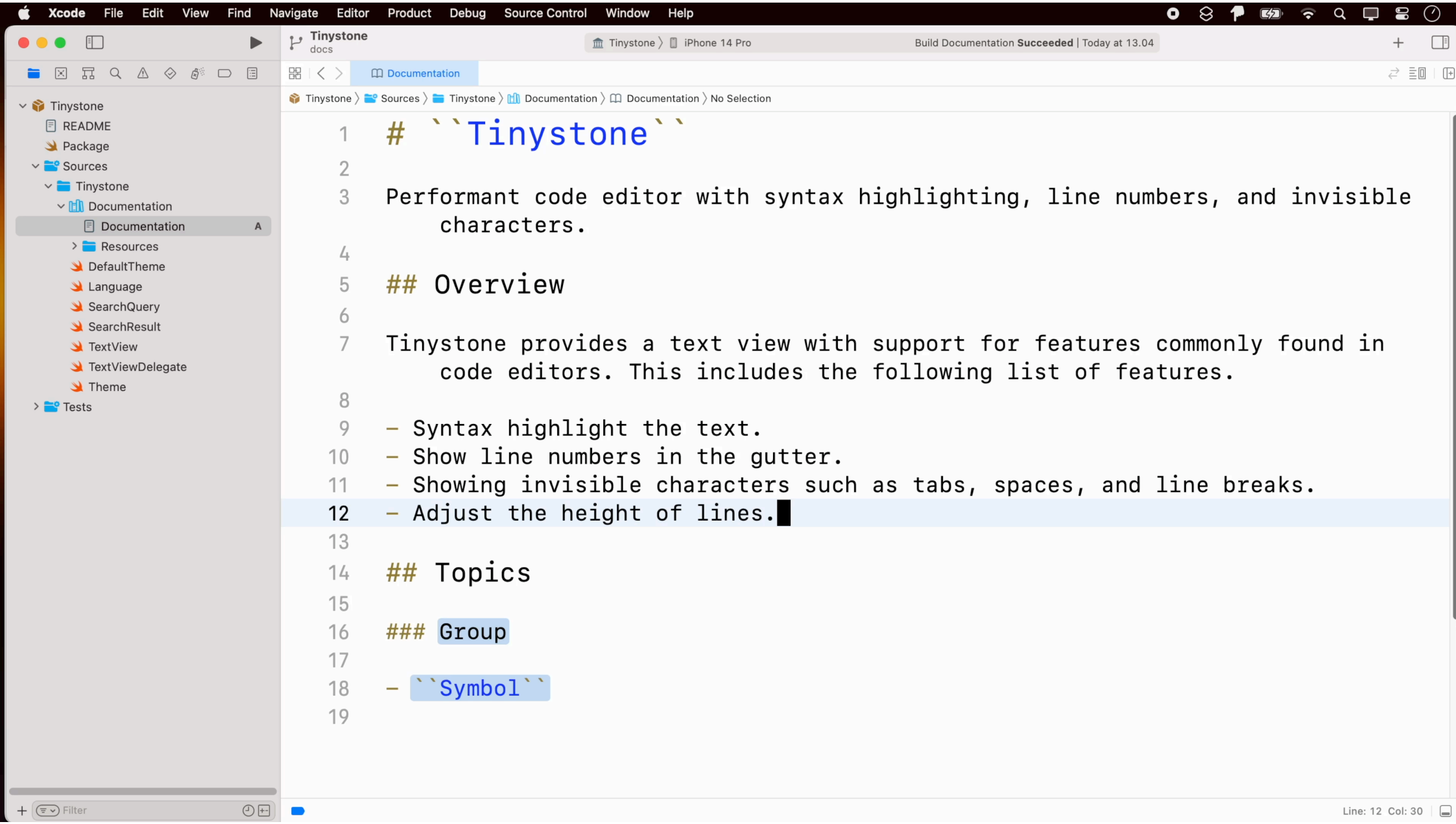
Tinystone provides a text view with support for features
commonly found in code editors. This includes the following
list of features.

- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as tabs, spaces, and line
breaks.
- Adjust the height of lines.

## Topics

### Groups

- ``Symbol``
```


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Build Documentation Succeeded | Today at 13.04
Documentation
Tinystone 〉 Sources 〉 Tinystone 〉 Documentation 〉 Documentation 〉 No Selection
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
1 # ``Tinystone``
2
3 Performant code editor with syntax highlighting, line numbers, and invisible characters.
4
5 ## Overview
6
7 Tinystone provides a text view with support for features commonly found in code editors. This includes the following list of features.
8
9 - Syntax highlight the text.
10 - Show line numbers in the gutter.
11 - Showing invisible characters such as tabs, spaces, and line breaks.
12 - Adjust the height of lines.
13
14 ## Topics
15
16 ### Group
17
18 - ``Symbol``
19
Filter
Line: 12 Col: 30

```
# ``Tinystone``

Performant code editor with syntax highlighting, line numbers,
and invisible characters.

## Overview

Tinystone provides a text view with support for features
commonly found in code editors. This includes the following
list of features.

- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as tabs, spaces, and line
breaks.
- Adjust the height of lines.

## Topics

### Groups

- ``Symbol``
```

Documentation.md

```
# ``Tinystone``

Performant code editor with syntax highlighting, line numbers, and invisible characters.

## Overview

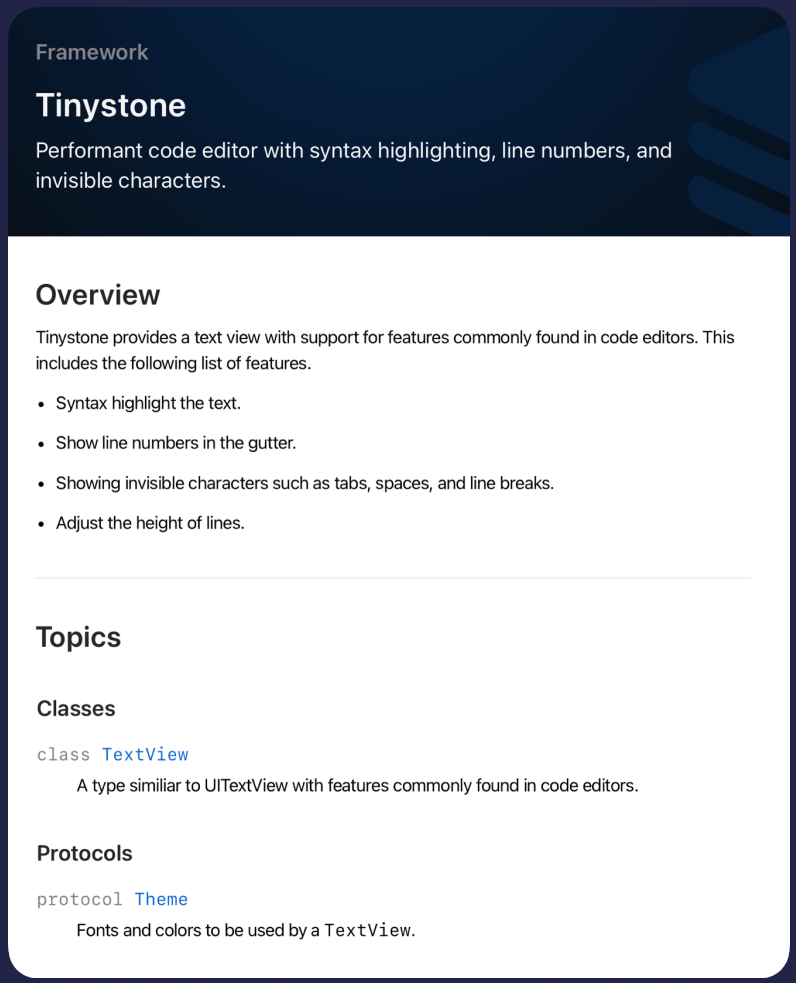
Tinystone provides a text view with support for features commonly found in code editors. This includes the following list of features.

- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as tabs, spaces, and line breaks.
- Adjust the height of lines.

## Topics

### Groups

- ``Symbol``
```



Documentation.md

```
# ``Tinystone``

Performant code editor with syntax highlighting, line numbers, and invisible characters.

## Overview

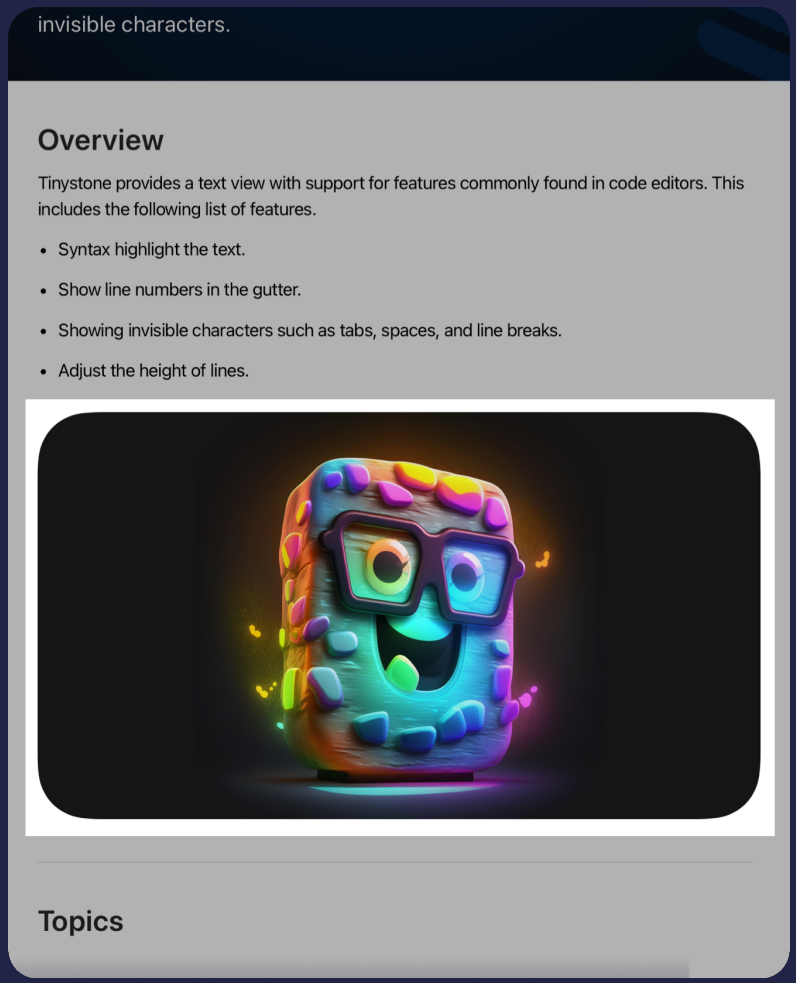
Tinystone provides a text view with support for features commonly found in code editors. This includes the following list of features.

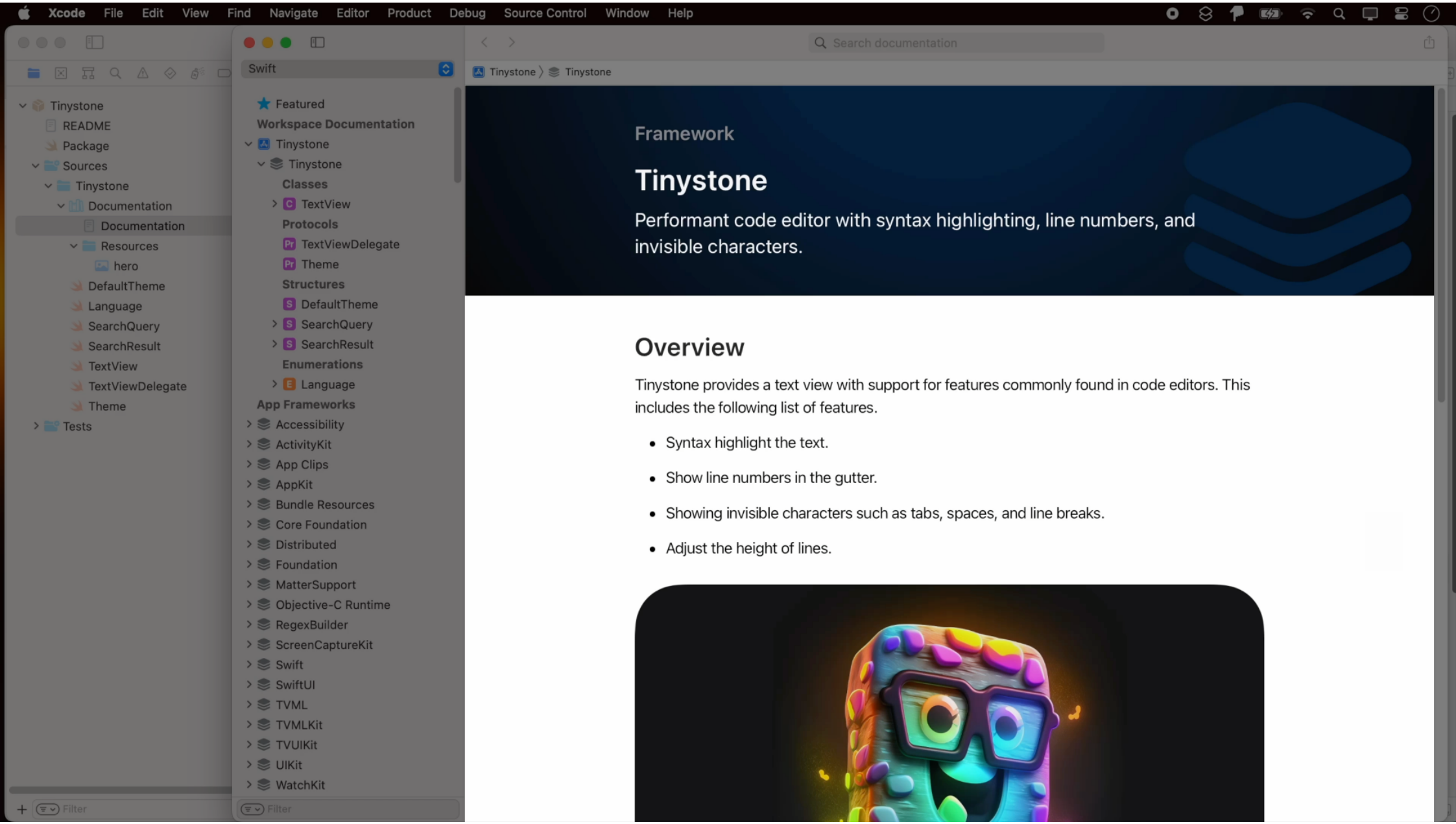
- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as tabs, spaces, and line breaks.
- Adjust the height of lines.

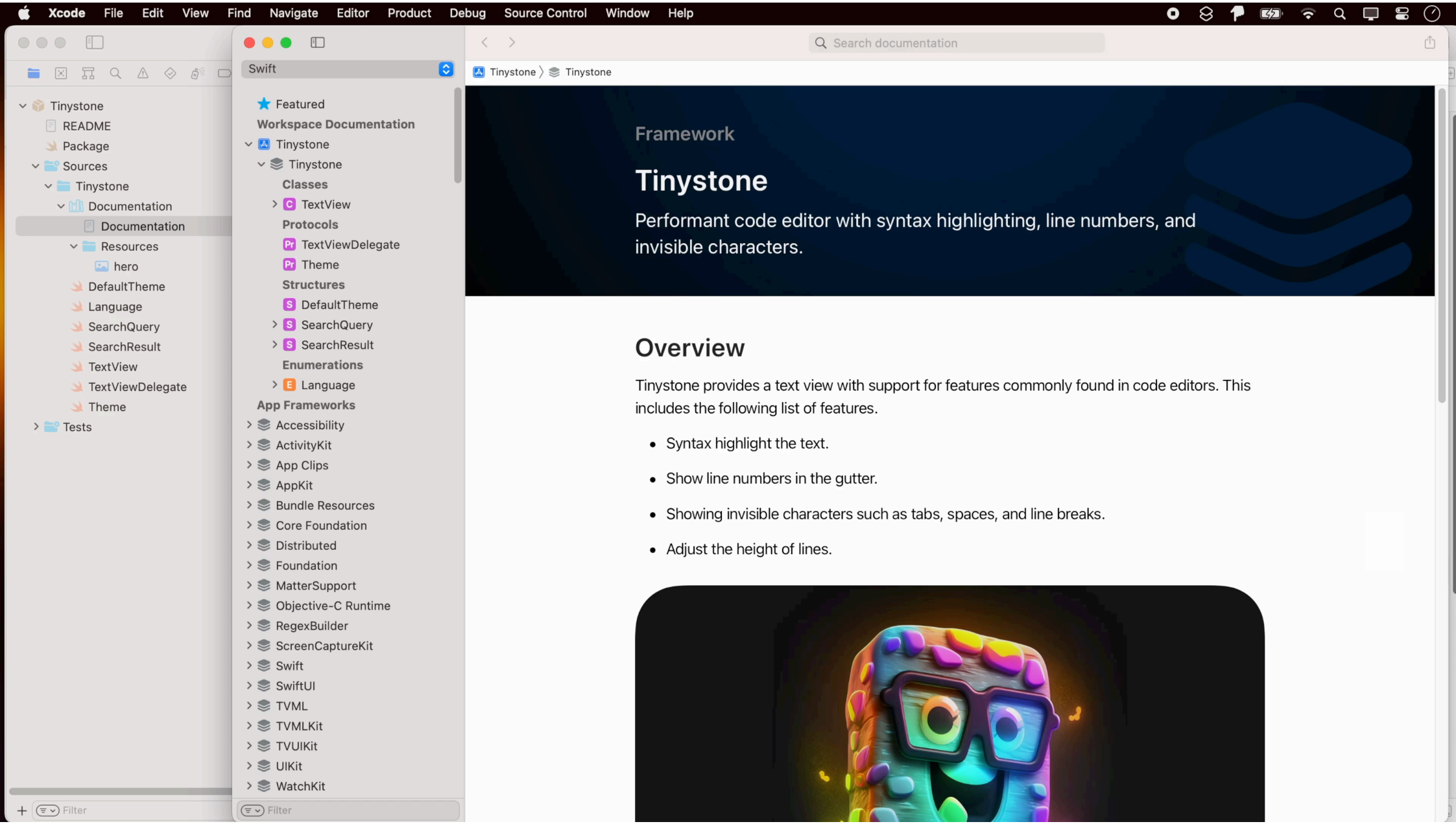
![Hero image showing a happy stone in fluorescent colors](hero.png)

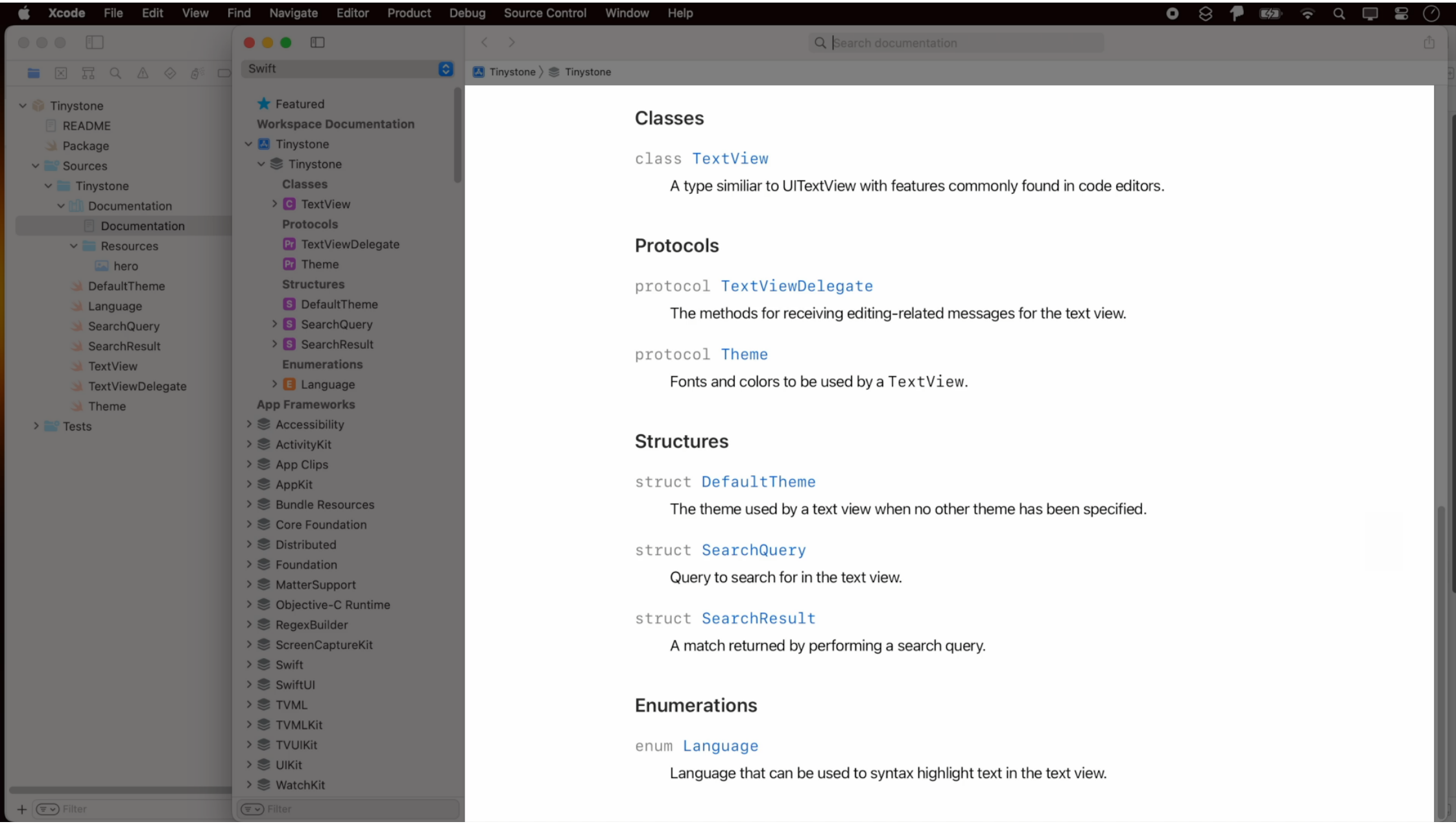
## Topics

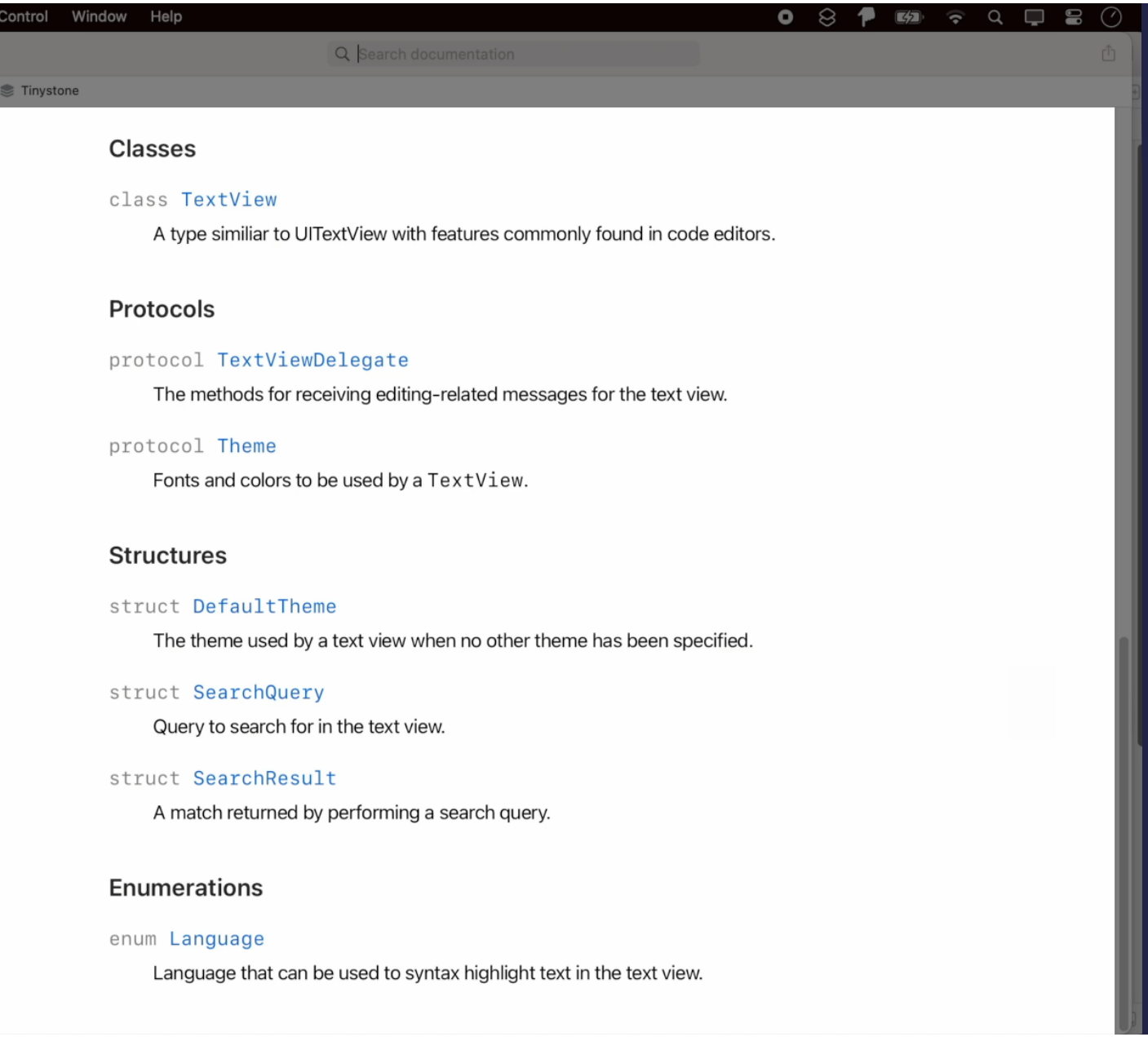
### Groups
```




Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Search documentation
Swift
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
hero
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Featured
Workspace Documentation
Tinystone
Tinystone
Classes
TextView
Protocols
TextViewDelegate
Theme
Structures
DefaultTheme
SearchQuery
SearchResult
Enumerations
Language
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
Filter
Tinystone Tinystone
Framework
Tinystone
Performant code editor with syntax highlighting, line numbers, and invisible characters.
Overview
Tinystone provides a text view with support for features commonly found in code editors. This includes the following list of features.
Syntax highlight the text.
Show line numbers in the gutter.
Showing invisible characters such as tabs, spaces, and line breaks.
Adjust the height of lines.


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
hero
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Swift
Featured
Workspace Documentation
Tinystone
Tinystone
Classes
TextView
Protocols
TextViewDelegate
Theme
Structures
DefaultTheme
SearchQuery
SearchResult
Enumerations
Language
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
Filter
Search documentation
Tinystone 〉 Tinystone
Framework
Tinystone
Performant code editor with syntax highlighting, line numbers, and invisible characters.
Overview
Tinystone provides a text view with support for features commonly found in code editors. This includes the following list of features.
Syntax highlight the text.
Show line numbers in the gutter.
Showing invisible characters such as tabs, spaces, and line breaks.
Adjust the height of lines.


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Search documentation
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
hero
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Swift
Featured
Workspace Documentation
Tinystone
Tinystone
Classes
TextView
Protocols
TextViewDelegate
Theme
Structures
DefaultTheme
SearchQuery
SearchResult
Enumerations
Language
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
Filter
Tinystone 〉 Tinystone
Classes
class TextView
A type similiar to UITextView with features commonly found in code editors.
Protocols
protocol TextViewDelegate
The methods for receiving editing-related messages for the text view.
protocol Theme
Fonts and colors to be used by a TextView.
Structures
struct DefaultTheme
The theme used by a text view when no other theme has been specified.
struct SearchQuery
Query to search for in the text view.
struct SearchResult
A match returned by performing a search query.
Enumerations
enum Language
Language that can be used to syntax highlight text in the text view.



Documentation.md

```
# ``Tinystone``

Performant code editor with syntax hig

## Overview

Tinystone provides a text view with su
following list of features.

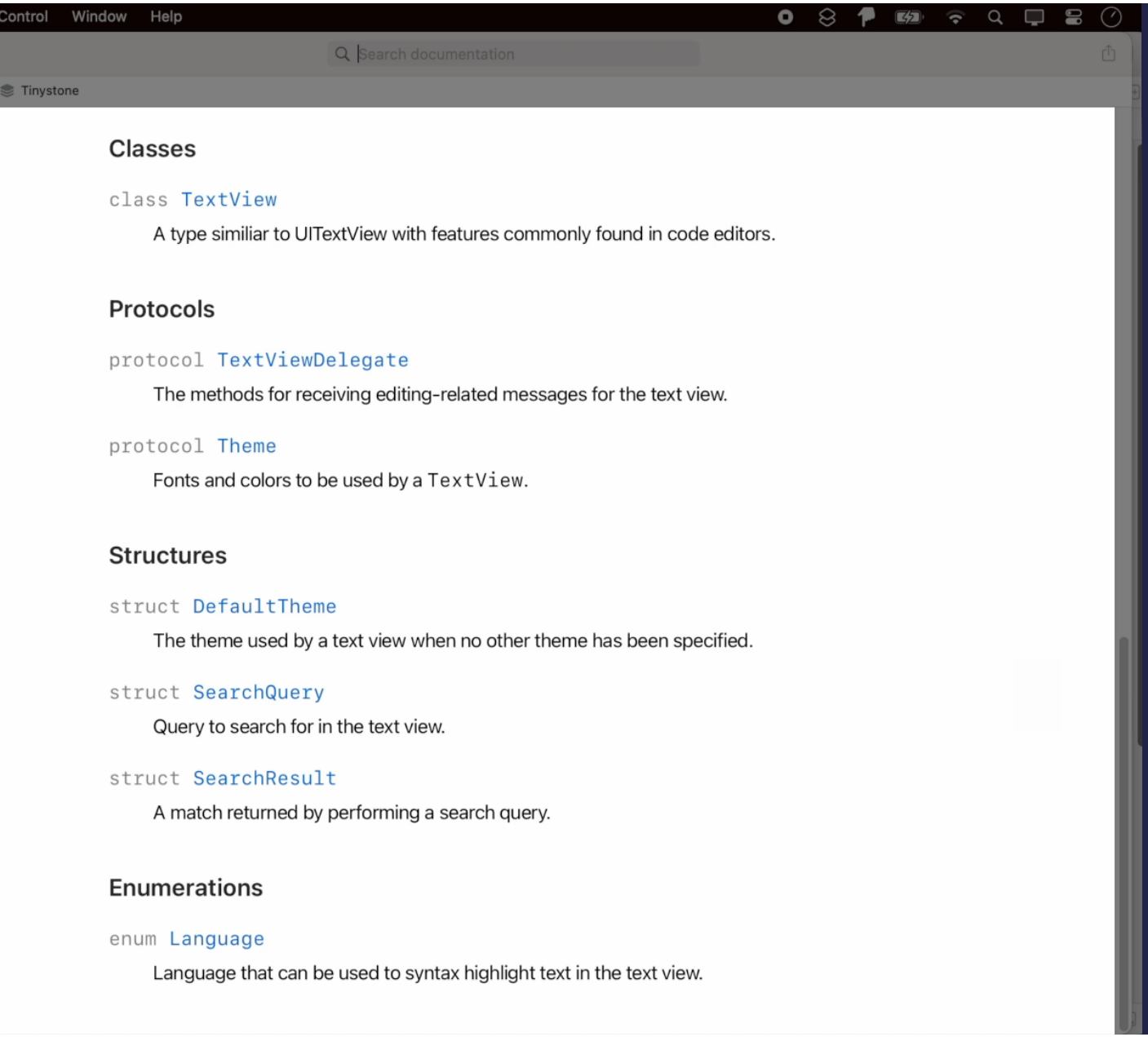
- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as
- Adjust the height of lines.

![Hero image showing a happy stone in

## Topics

### Groups

- ``Symbol``
```



Documentation.md

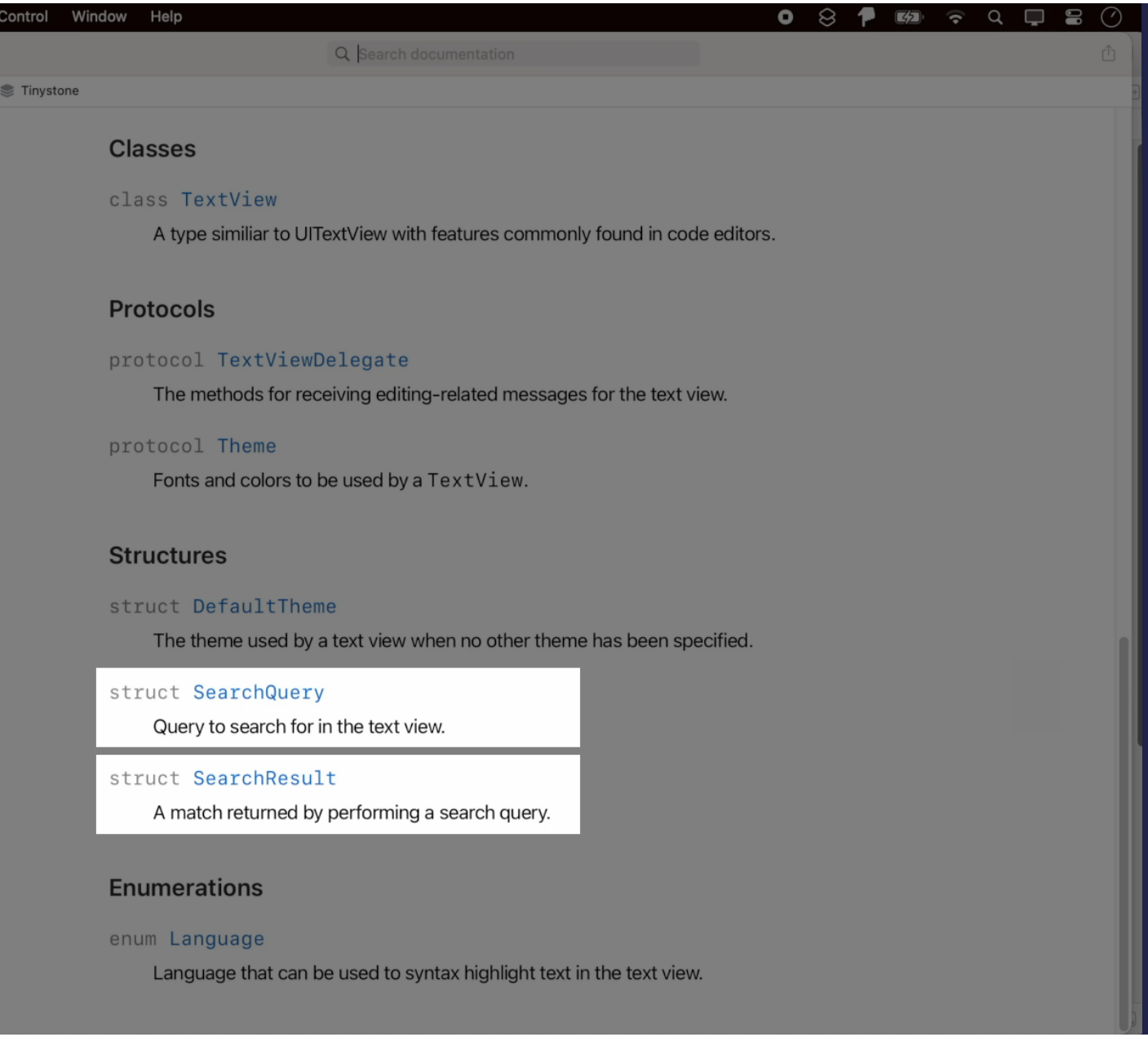
```
- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as
- Adjust the height of lines.

![Hero image showing a happy stone in

## Topics

### Groups

- ``Symbol``
```



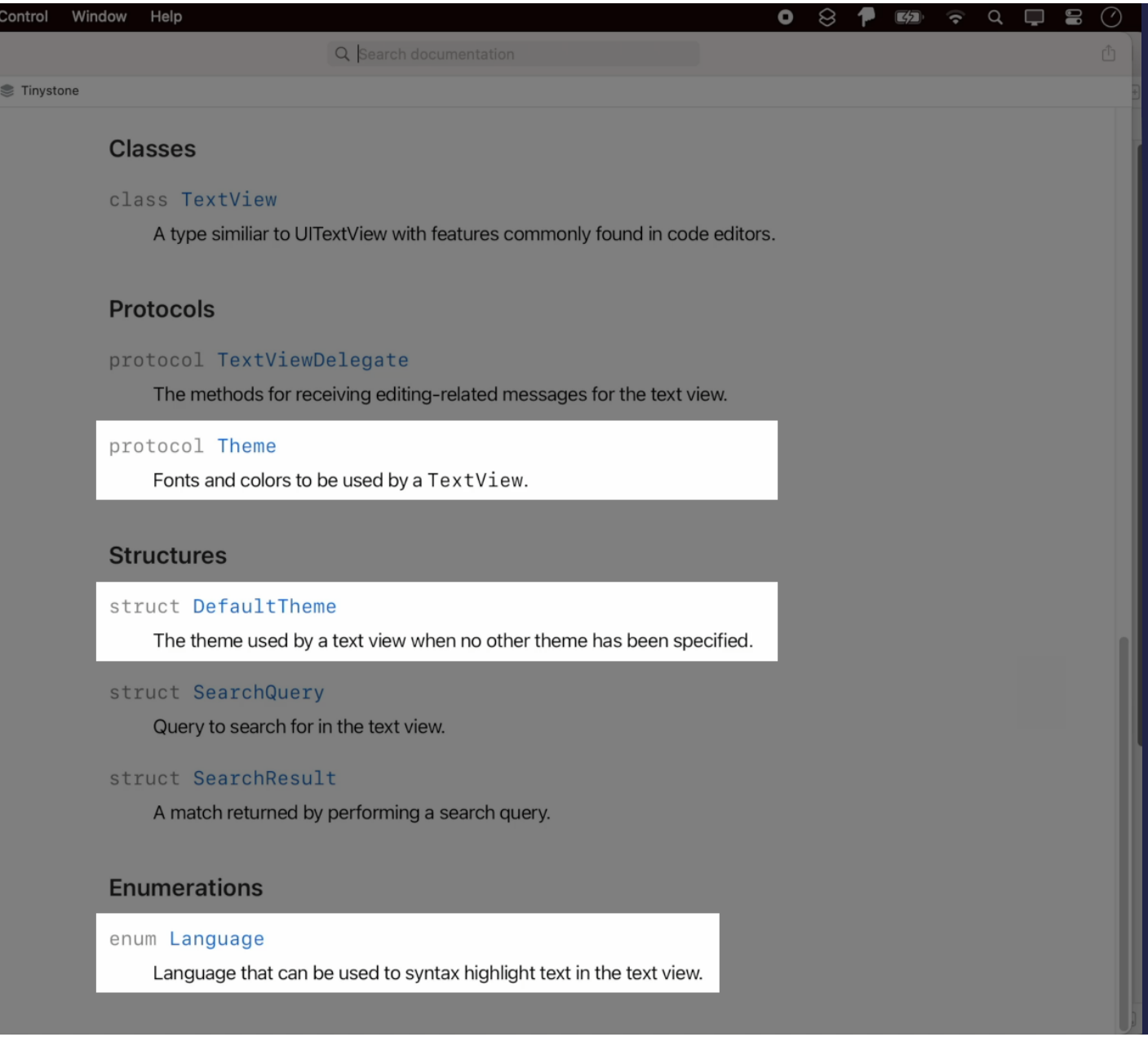
Documentation.md

```
- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as
- Adjust the height of lines.

![Hero image showing a happy stone in

## Topics

### Find and Replace
```



Documentation.md

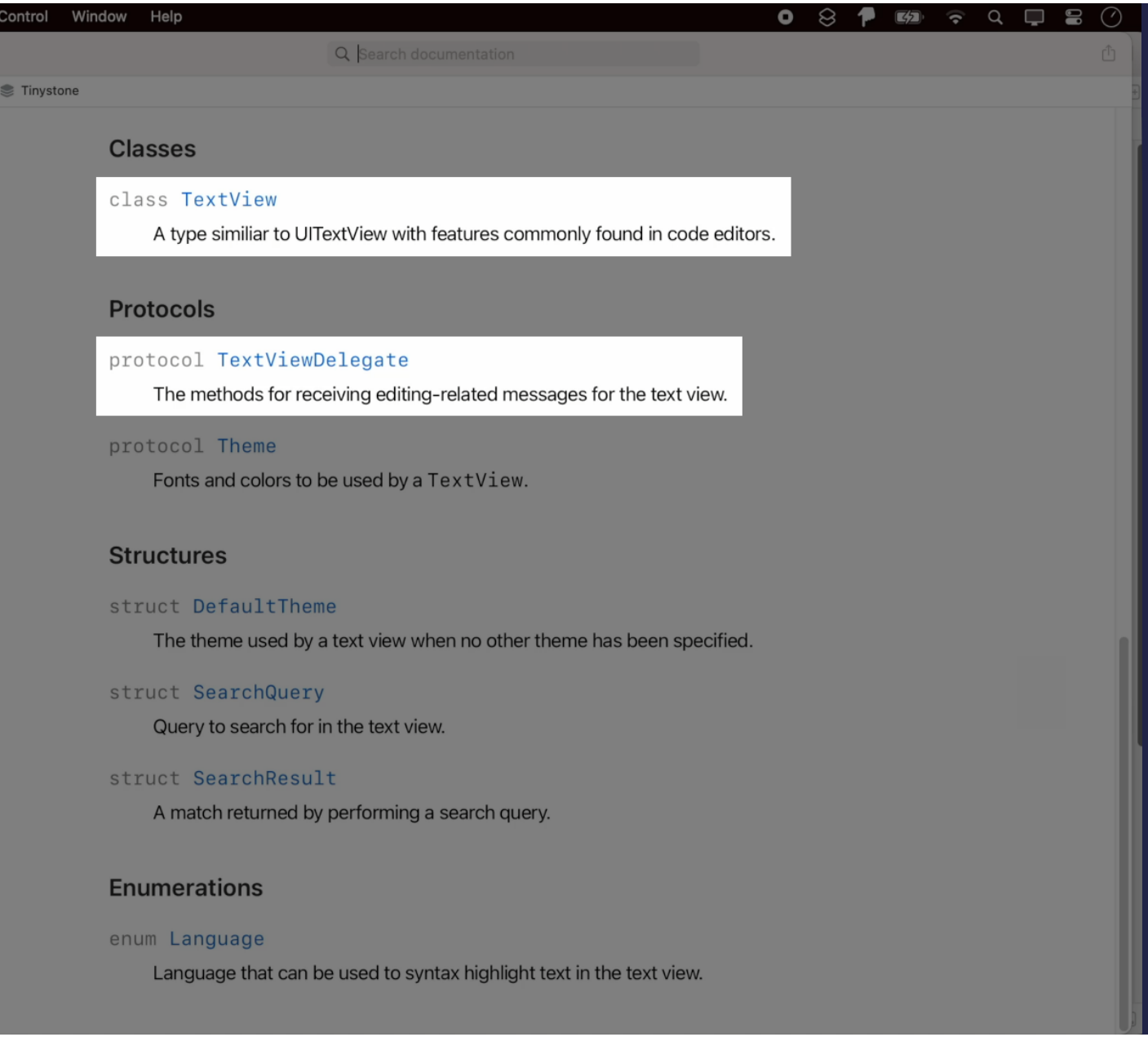
```
- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as
- Adjust the height of lines.

![Hero image showing a happy stone in

## Topics

### Syntax Highlighting

### Find and Replace
```



Documentation.md

```
- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as
- Adjust the height of lines.

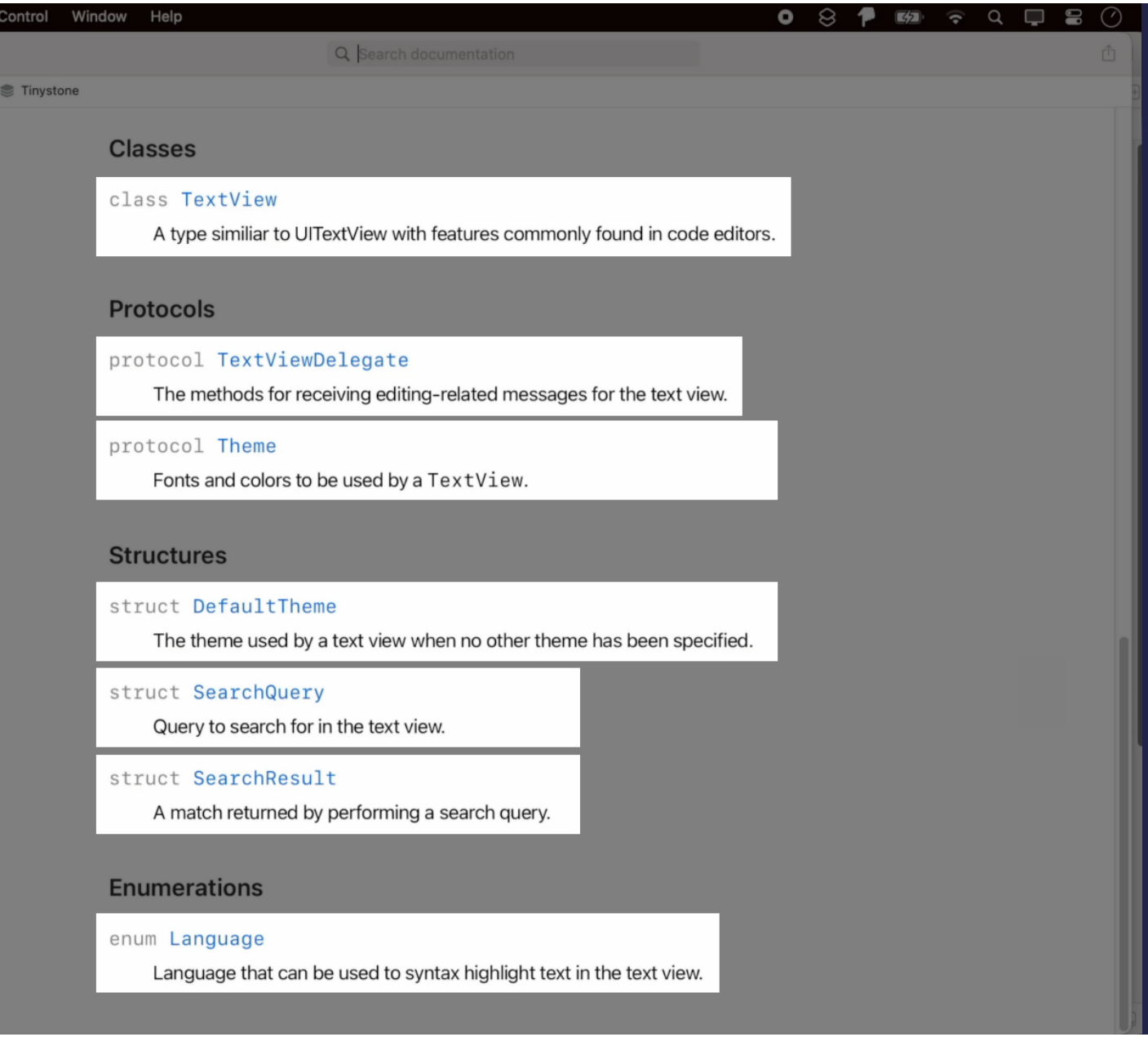
![Hero image showing a happy stone in

## Topics

### Essentials

### Syntax Highlighting

### Find and Replace
```



Documentation.md

```
- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as
- Adjust the height of lines.

![Hero image showing a happy stone in

## Topics

### Essentials

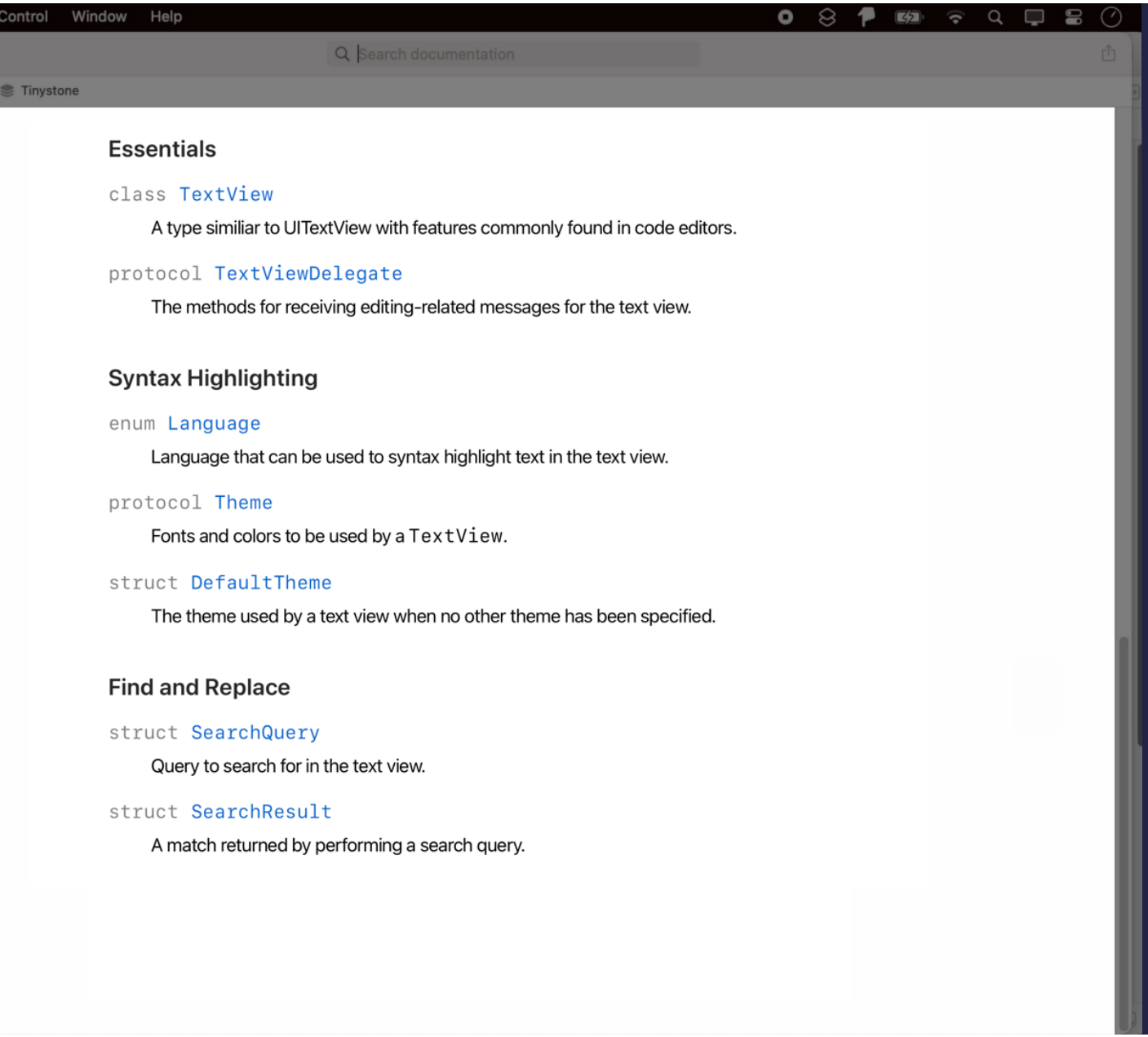
- ``TextView``
- ``TextViewDelegate``

### Syntax Highlighting

- ``Language``
- ``Theme``
- ``DefaultTheme``

### Find and Replace

- ``SearchQuery``
- ``SearchResult``
```



Documentation.md

```
- Syntax highlight the text.
- Show line numbers in the gutter.
- Showing invisible characters such as
- Adjust the height of lines.

![Hero image showing a happy stone in

## Topics

### Essentials

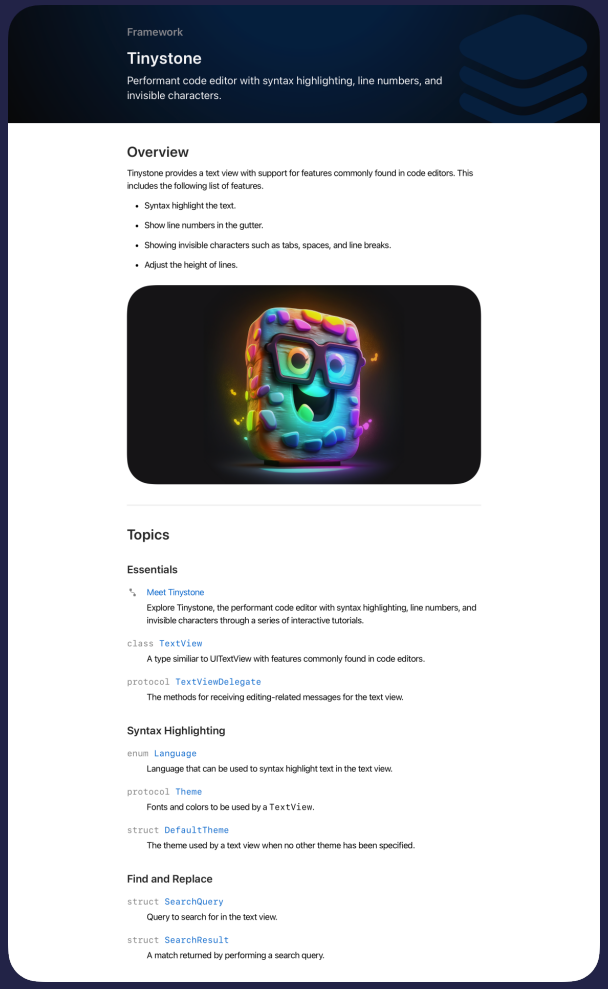
- ``TextView``
- ``TextViewDelegate``

### Syntax Highlighting

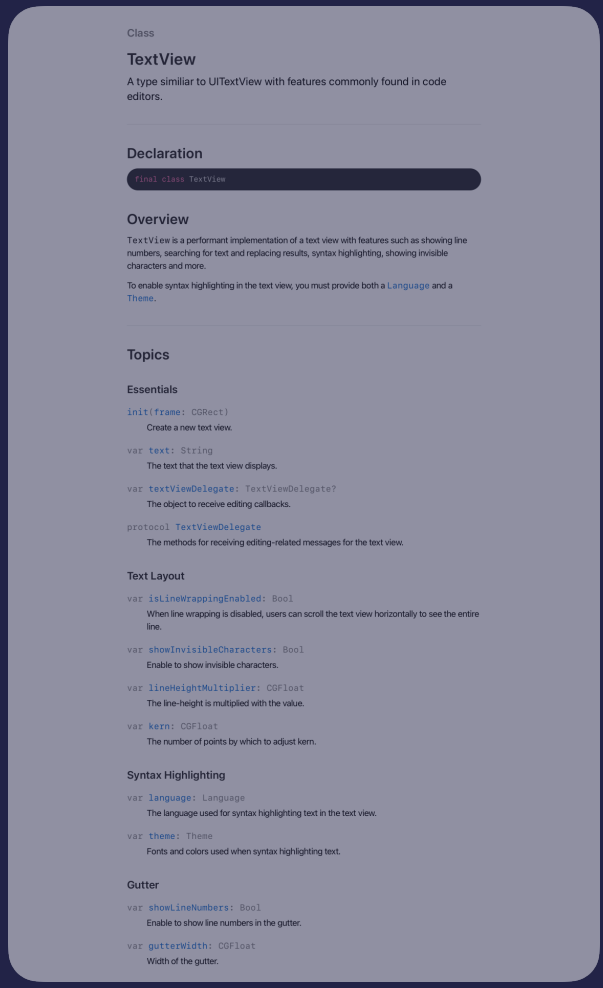
- ``Language``
- ``Theme``
- ``DefaultTheme``

### Find and Replace

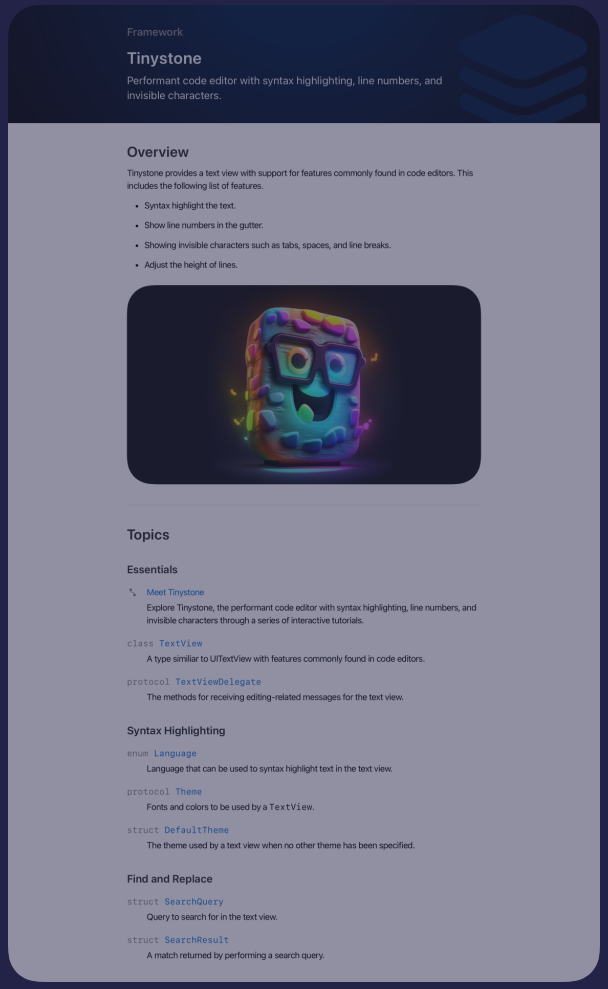
- ``SearchQuery``
- ``SearchResult``
```



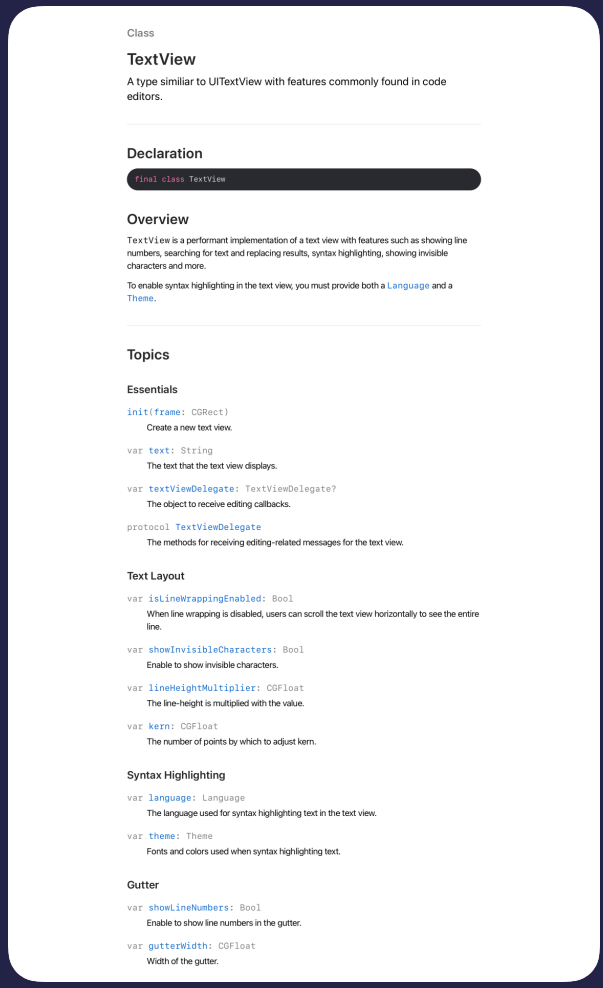
Landing Page



Extensions



Landing Page



Extensions

TextView.md

```
# ``Tinystone/TextView``

## Topics

### Essentials

- ``init(frame:)``
- ``text``
- ``textViewDelegate``
- ``TextViewDelegate``

### Text Layout

- ``isLineWrappingEnabled``
- ``showInvisibleCharacters``
- ``lineHeightMultiplier``
- ``kern``

### Syntax Highlighting

- ``language``
- ``theme``

### Gutter

- ``showLineNumbers``
- ``gutterWidth``
```

Class

TextView

A type similiar to UITextView with features commonly found in code editors.

Declaration

```
final class TextView
```

Overview

`TextView` is a performant implementation of a text view with features such as showing line numbers, searching for text and replacing results, syntax highlighting, showing invisible characters and more.

To enable syntax highlighting in the text view, you must provide both a `Language` and a `Theme`.

Topics

Essentials

`init(frame: CGRect)`
Create a new text view.

`var text: String`
The text that the text view displays.

`var textViewDelegate: TextViewDelegate?`
The object to receive editing callbacks.

`protocol TextViewDelegate`
The methods for receiving editing-related messages for the text view.

Text Layout

`var isLineWrappingEnabled: Bool`
When line wrapping is disabled, users can scroll the text view horizontally to see the entire line.

`var showInvisibleCharacters: Bool`
Enable to show invisible characters.

`var lineHeightMultiplier: CGFloat`
The line-height is multiplied with the value.

`var kern: CGFloat`
The number of points by which to adjust kern.

Syntax Highlighting

`var language: Language`
The language used for syntax highlighting text in the text view.

`var theme: Theme`
Fonts and colors used when syntax highlighting text.

Gutter

`var showLineNumbers: Bool`
Enable to show line numbers in the gutter.

TextView.md

```
# ``Tinystone/TextView``

## Topics

### Essentials

- ``init(frame:)``
- ``text``
- ``textViewDelegate``
- ``TextViewDelegate``

### Text Layout

- ``isLineWrappingEnabled``
- ``showInvisibleCharacters``
- ``lineHeightMultiplier``
- ``kern``

### Syntax Highlighting

- ``language``
- ``theme``

### Gutter

- ``showLineNumbers``
- ``gutterWidth``
```

Class

TextView

A type similiar to UITextView with features commonly found in code editors.

Declaration

```
final class TextView
```

Overview

`TextView` is a performant implementation of a text view with features such as showing line numbers, searching for text and replacing results, syntax highlighting, showing invisible characters and more.

To enable syntax highlighting in the text view, you must provide both a `Language` and a `Theme`.

Topics

Essentials

`init(frame: CGRect)`
Create a new text view.

`var text: String`
The text that the text view displays.

`var textViewDelegate: TextViewDelegate?`
The object to receive editing callbacks.

`protocol TextViewDelegate`
The methods for receiving editing-related messages for the text view.

Text Layout

`var isLineWrappingEnabled: Bool`
When line wrapping is disabled, users can scroll the text view horizontally to see the entire line.

`var showInvisibleCharacters: Bool`
Enable to show invisible characters.

`var lineHeightMultiplier: CGFloat`
The line-height is multiplied with the value.

`var kern: CGFloat`
The number of points by which to adjust kern.

Syntax Highlighting

`var language: Language`
The language used for syntax highlighting text in the text view.

`var theme: Theme`
Fonts and colors used when syntax highlighting text.

Gutter

`var showLineNumbers: Bool`
Enable to show line numbers in the gutter.

TextView.md

```
# ``Tinystone/TextView``

## Topics

### Essentials

- ``init(frame:)``
- ``text``
- ``textViewDelegate``
- ``TextViewDelegate``

### Text Layout

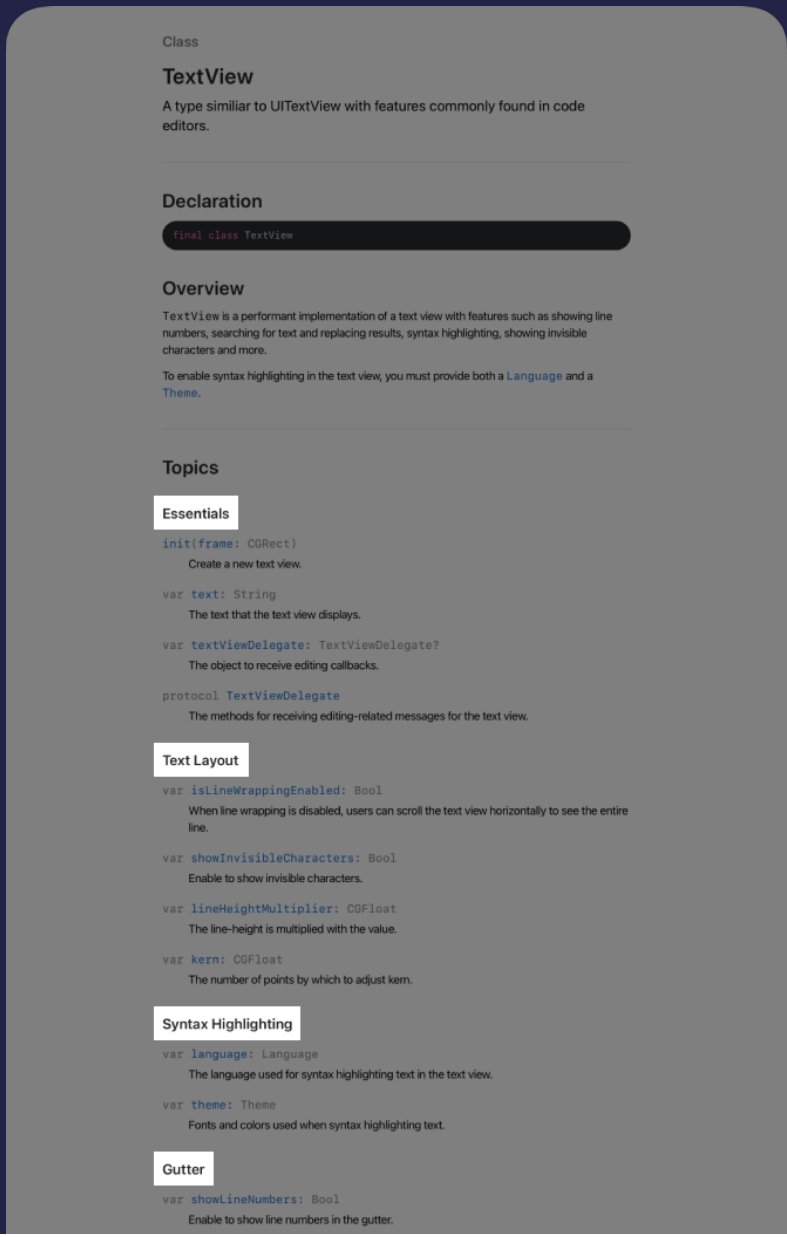
- ``isLineWrappingEnabled``
- ``showInvisibleCharacters``
- ``lineHeightMultiplier``
- ``kern``

### Syntax Highlighting

- ``language``
- ``theme``

### Gutter

- ``showLineNumbers``
- ``gutterWidth``
```



Class

TextView

A type similiar to UITextView with features commonly found in code editors.

Declaration

```
final class TextView
```

Overview

TextView is a performant implementation of a text view with features such as showing line numbers, searching for text and replacing results, syntax highlighting, showing invisible characters and more.

To enable syntax highlighting in the text view, you must provide both a Language and a Theme.

Topics

Essentials

init(frame: CGRect)
Create a new text view.

var text: String
The text that the text view displays.

var textViewDelegate: TextViewDelegate?
The object to receive editing callbacks.

protocol TextViewDelegate
The methods for receiving editing-related messages for the text view.

Text Layout

var isLineWrappingEnabled: Bool
When line wrapping is disabled, users can scroll the text view horizontally to see the entire line.

var showInvisibleCharacters: Bool
Enable to show invisible characters.

var lineHeightMultiplier: CGFloat
The line-height is multiplied with the value.

var kern: CGFloat
The number of points by which to adjust kern.

Syntax Highlighting

var language: Language
The language used for syntax highlighting text in the text view.

var theme: Theme
Fonts and colors used when syntax highlighting text.

Gutter

var showLineNumbers: Bool
Enable to show line numbers in the gutter.

Describe Symbols

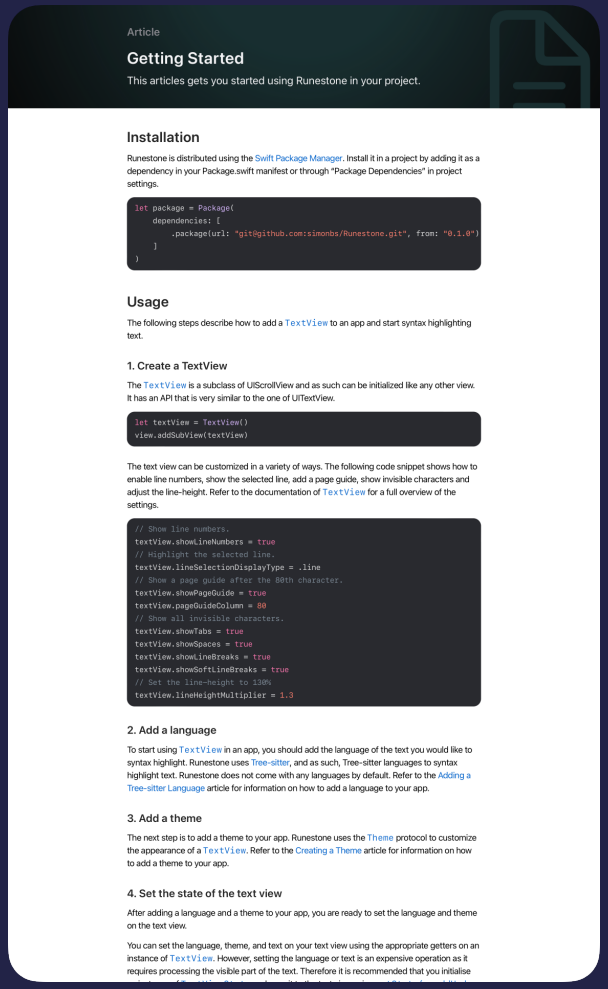
Add Structure

Add Supplemental Content

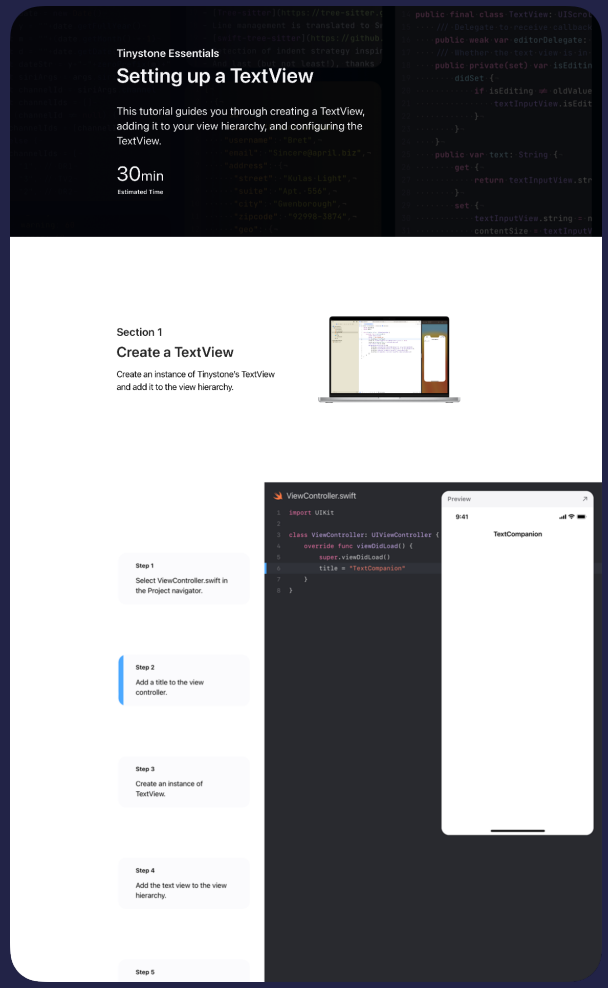
Describe Symbols

Add Structure

Add Supplemental Content



Articles



Tutorials

Getting Started.md

```
# Getting Started

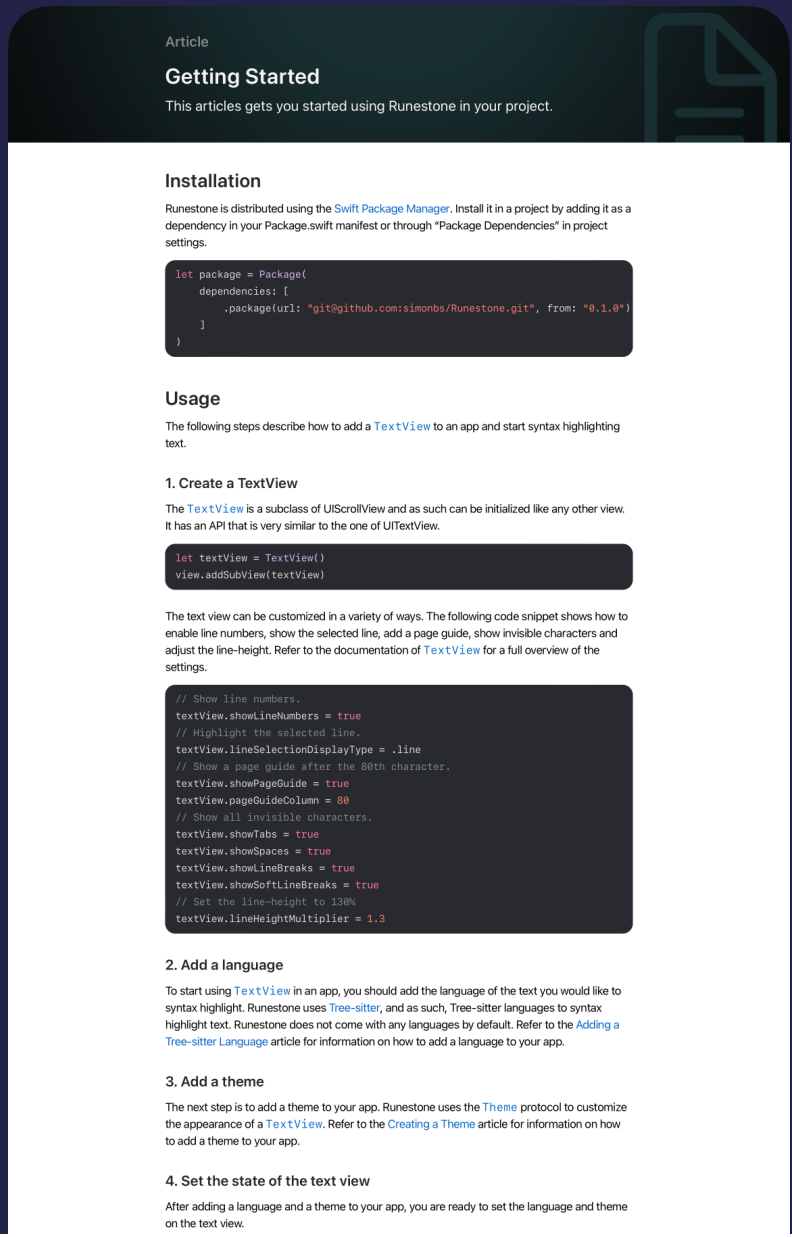
This articles gets you started using Tinystone in your
project.

## Installation

Tinystone is distributed using the [Swift Package Manager]
(https://www.swift.org/package-manager/). Install it in a
project by adding it as a dependency in your Package.swift
manifest or through "Package Dependencies" in  project
settings.

```swift
let package = Package(
    dependencies: [
        .package(url: "git@github.com:simonbs/Tinystone.git",
from: "0.1.0")
    ]
)
```

Usage
```



Article

Getting Started

This articles gets you started using Runestone in your project.

Installation

Runestone is distributed using the Swift Package Manager. Install it in a project by adding it as a dependency in your Package.swift manifest or through “Package Dependencies” in project settings.

```
let package = Package(
    dependencies: [
        .package(url: "git@github.com:simonbs/Runestone.git", from: "0.1.0")
    ]
)
```

Usage

The following steps describe how to add a `TextView` to an app and start syntax highlighting text.

1. Create a TextView

The `TextView` is a subclass of UIScrollView and as such can be initialized like any other view. It has an API that is very similar to the one of UITextView.

```
let textView = TextView()
view.addSubView(textView)
```

The text view can be customized in a variety of ways. The following code snippet shows how to enable line numbers, show the selected line, add a page guide, show invisible characters and adjust the line-height. Refer to the documentation of `TextView` for a full overview of the settings.

```
// Show line numbers.
textView.showLineNumbers = true
// Highlight the selected line.
textView.lineSelectionDisplayType = .line
// Show a page guide after the 80th character.
textView.showPageGuide = true
textView.pageGuideColumn = 80
// Show all invisible characters.
textView.showTabs = true
textView.showSpaces = true
textView.showLineBreaks = true
textView.showSoftLineBreaks = true
// Set the line-height to 130%
textView.lineHeightMultiplier = 1.3
```

2. Add a language

To start using `TextView` in an app, you should add the language of the text you would like to syntax highlight. Runestone uses Tree-sitter, and as such, Tree-sitter languages to syntax highlight text. Runestone does not come with any languages by default. Refer to the Adding a Tree-sitter Language article for information on how to add a language to your app.

3. Add a theme

The next step is to add a theme to your app. Runestone uses the `Theme` protocol to customize the appearance of a `TextView`. Refer to the Creating a Theme article for information on how to add a theme to your app.

4. Set the state of the text view

After adding a language and a theme to your app, you are ready to set the language and theme on the text view.

Documentation.md

```
# ``Tinystone``

Performant code editor with syntax highlighting, line numbers,
and invisible characters.

## Overview

Tinystone provides a text view with support for...

## Topics

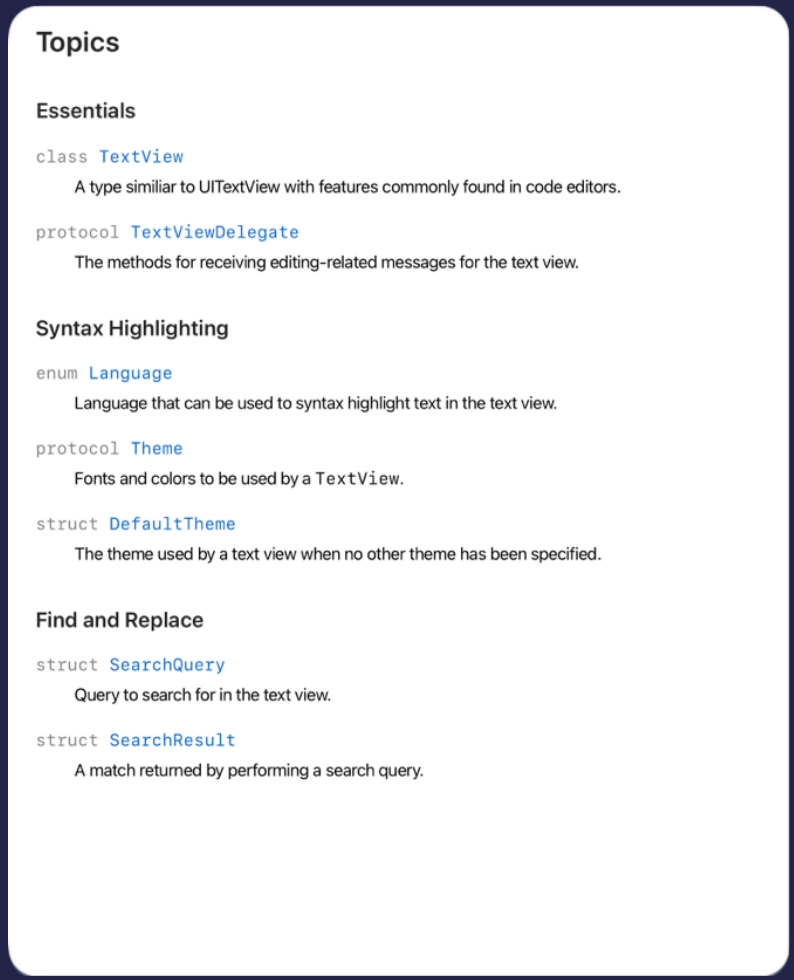
### Essentials

- ``TextView``
- ``TextViewDelegate``

### Syntax Highlighting

- ``Language``
- ``Theme``
- ``DefaultTheme``

### Find and Replace
```



Documentation.md

```
# ``Tinystone``

Performant code editor with syntax highlighting, line numbers,
and invisible characters.

## Overview

Tinystone provides a text view with support for...

## Topics

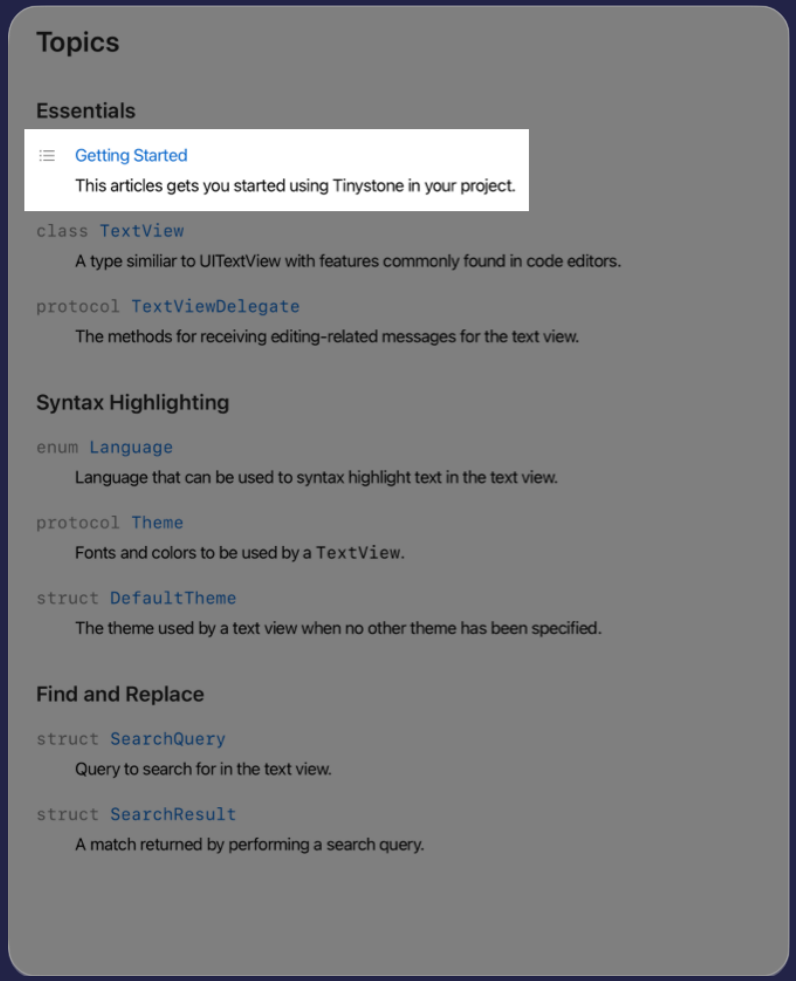
### Essentials

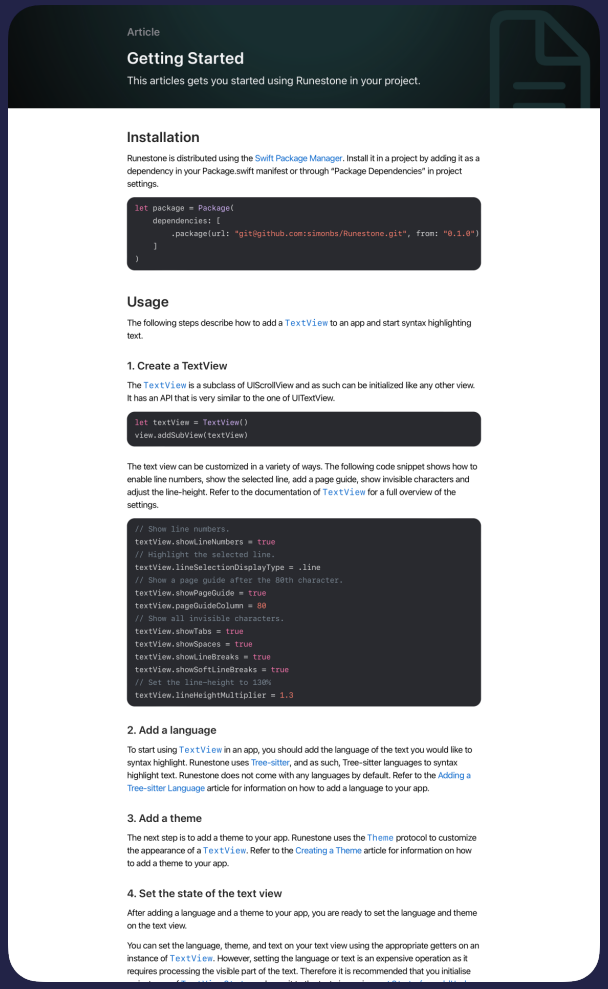
- <doc:Getting-Started>
- ``TextView``
- ``TextViewDelegate``

### Syntax Highlighting

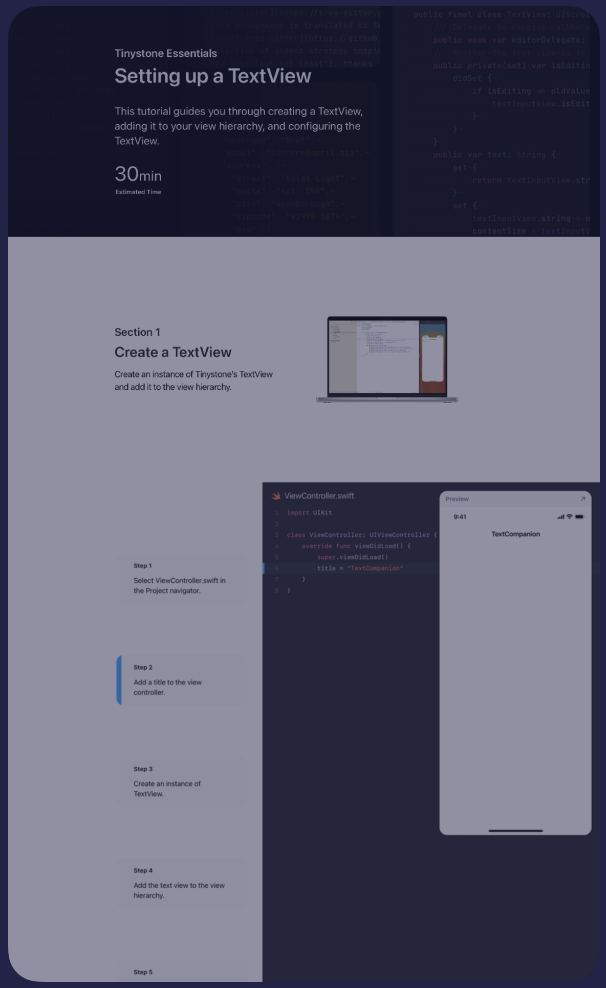
- ``Language``
- ``Theme``
- ``DefaultTheme``

### Find and Replace
```

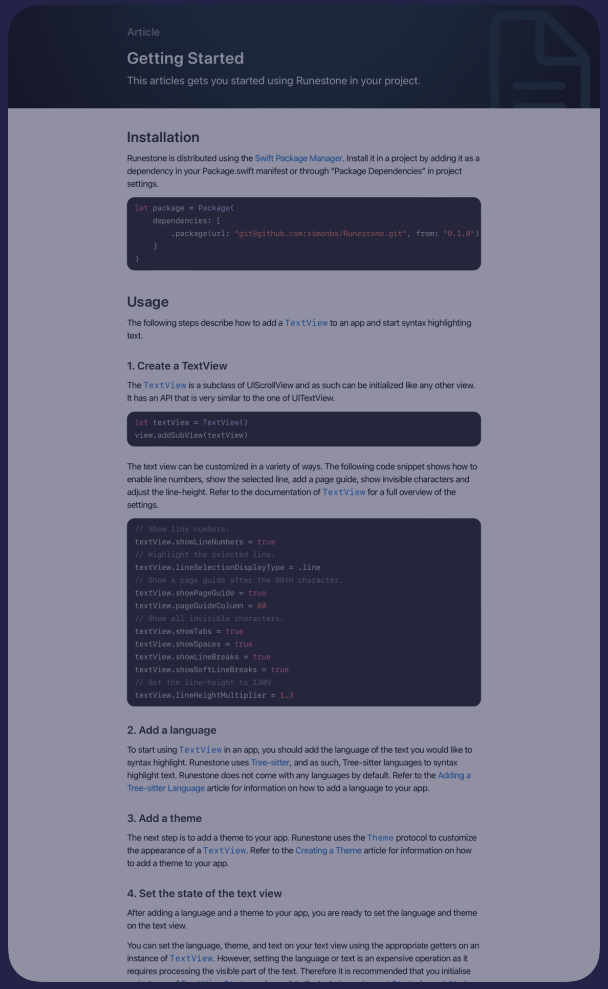




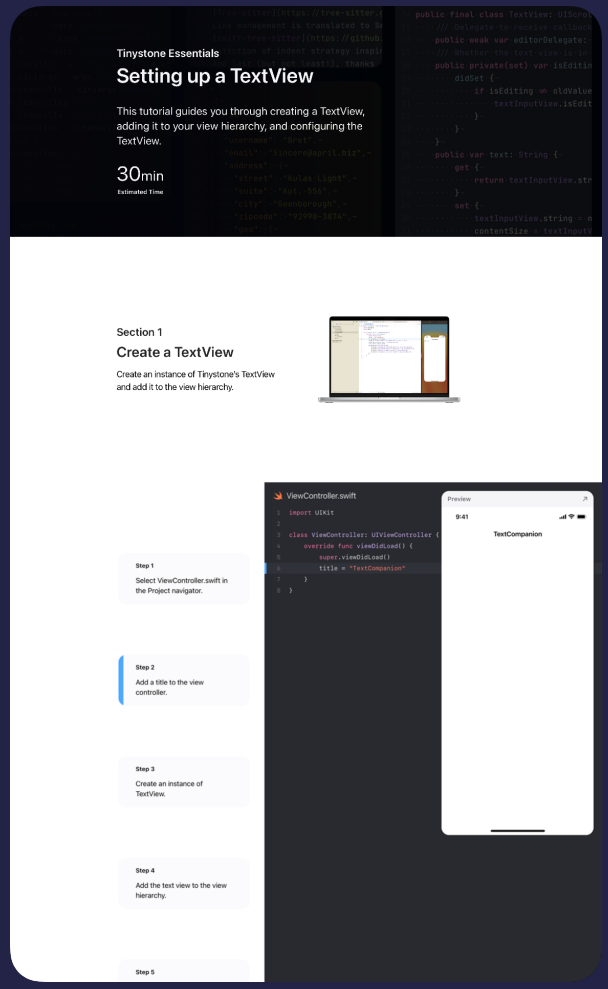
Articles



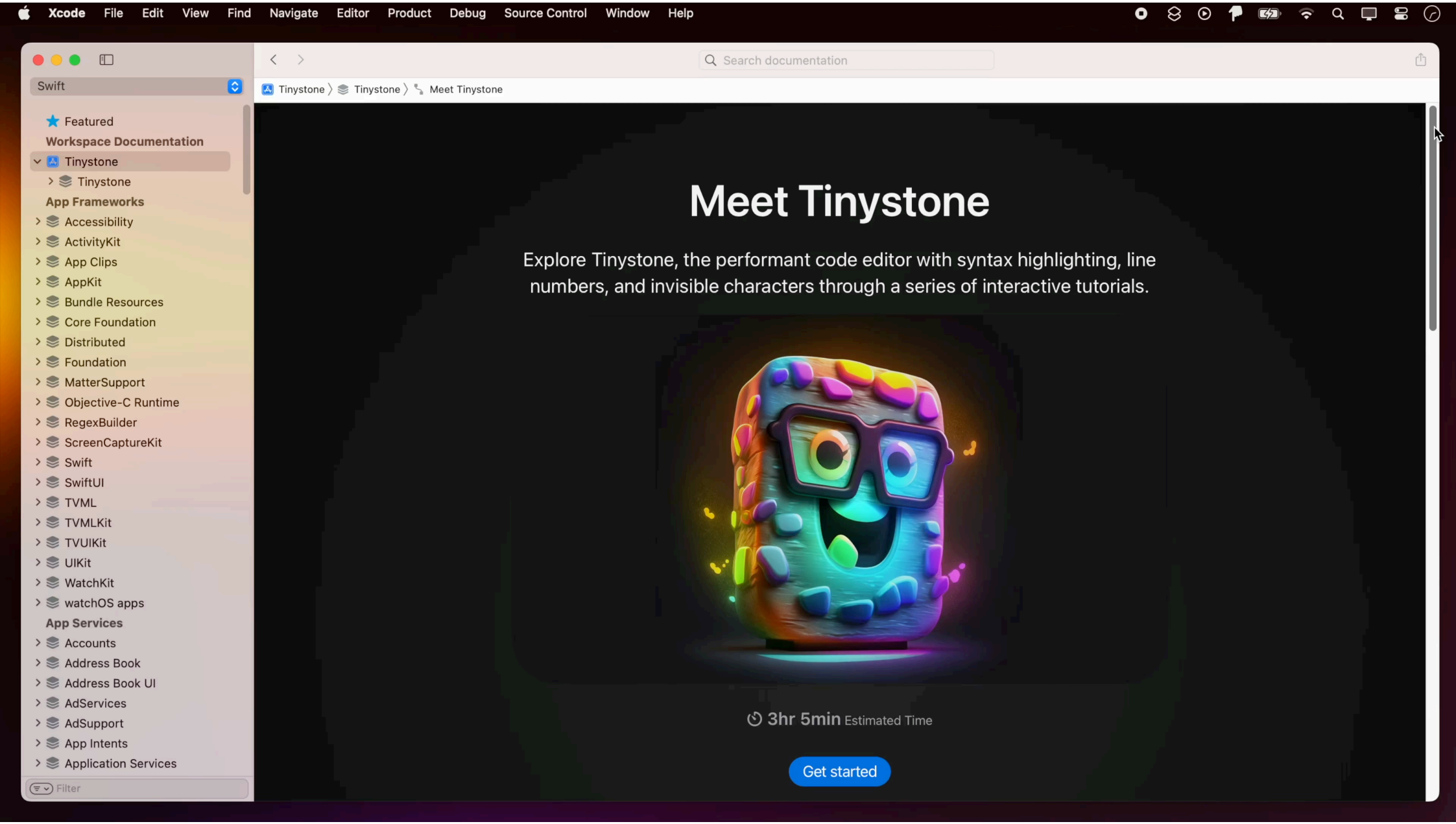
Tutorials

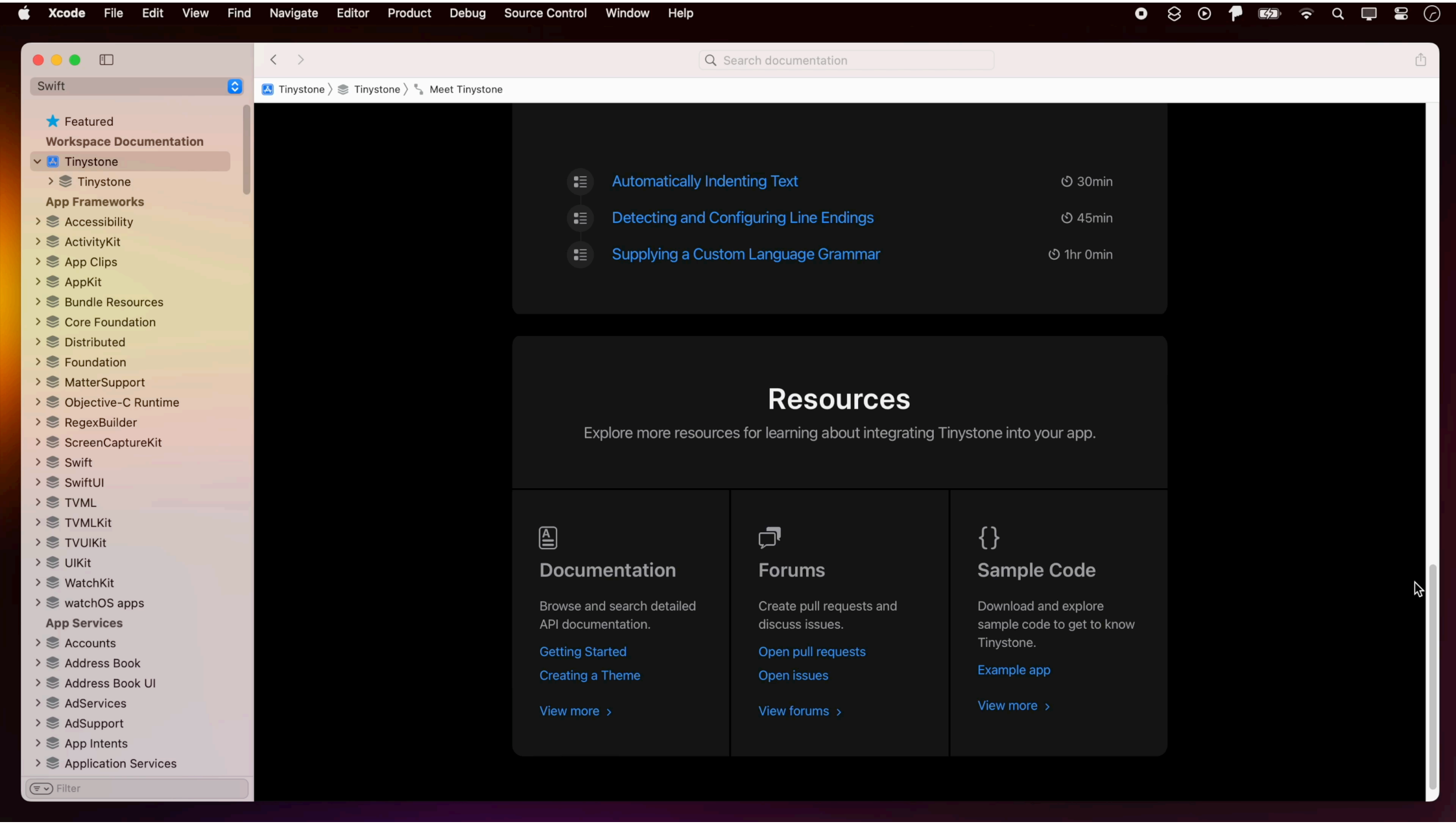


Articles



Tutorials


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Search documentation
Swift
Tinystone Tinystone Meet Tinystone
Featured
Workspace Documentation
Tinystone
Tinystone
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
watchOS apps
App Services
Accounts
Address Book
Address Book UI
AdServices
AdSupport
App Intents
Application Services
Filter
Meet Tinystone
Explore Tinystone, the performant code editor with syntax highlighting, line numbers, and invisible characters through a series of interactive tutorials.
3hr 5min Estimated Time
Get started


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Search documentation
Swift
Tinystone Tinystone Meet Tinystone
Featured
Workspace Documentation
Tinystone
Tinystone
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
SwiftUI
TVML
TVMLKit
TVUIKit
UIKit
WatchKit
watchOS apps
App Services
Accounts
Address Book
Address Book UI
AdServices
AdSupport
App Intents
Application Services
Filter
Automatically Indenting Text 30min
Detecting and Configuring Line Endings 45min
Supplying a Custom Language Grammar 1hr 0min
Resources
Explore more resources for learning about integrating Tinystone into your app.
Documentation
Browse and search detailed API documentation.
Getting Started
Creating a Theme
View more
Forums
Create pull requests and discuss issues.
Open pull requests
Open issues
View forums
Sample Code
Download and explore sample code to get to know Tinystone.
Example app
View more

DocC uses Markdown

DocC uses a dialect of Markdown

Directives

@Intro

@Steps

@Code

@Metadata

@Chapter

@TechnologyRoot

@Comment

@ContentAndMedia

@DisplayName

@TutorialReference

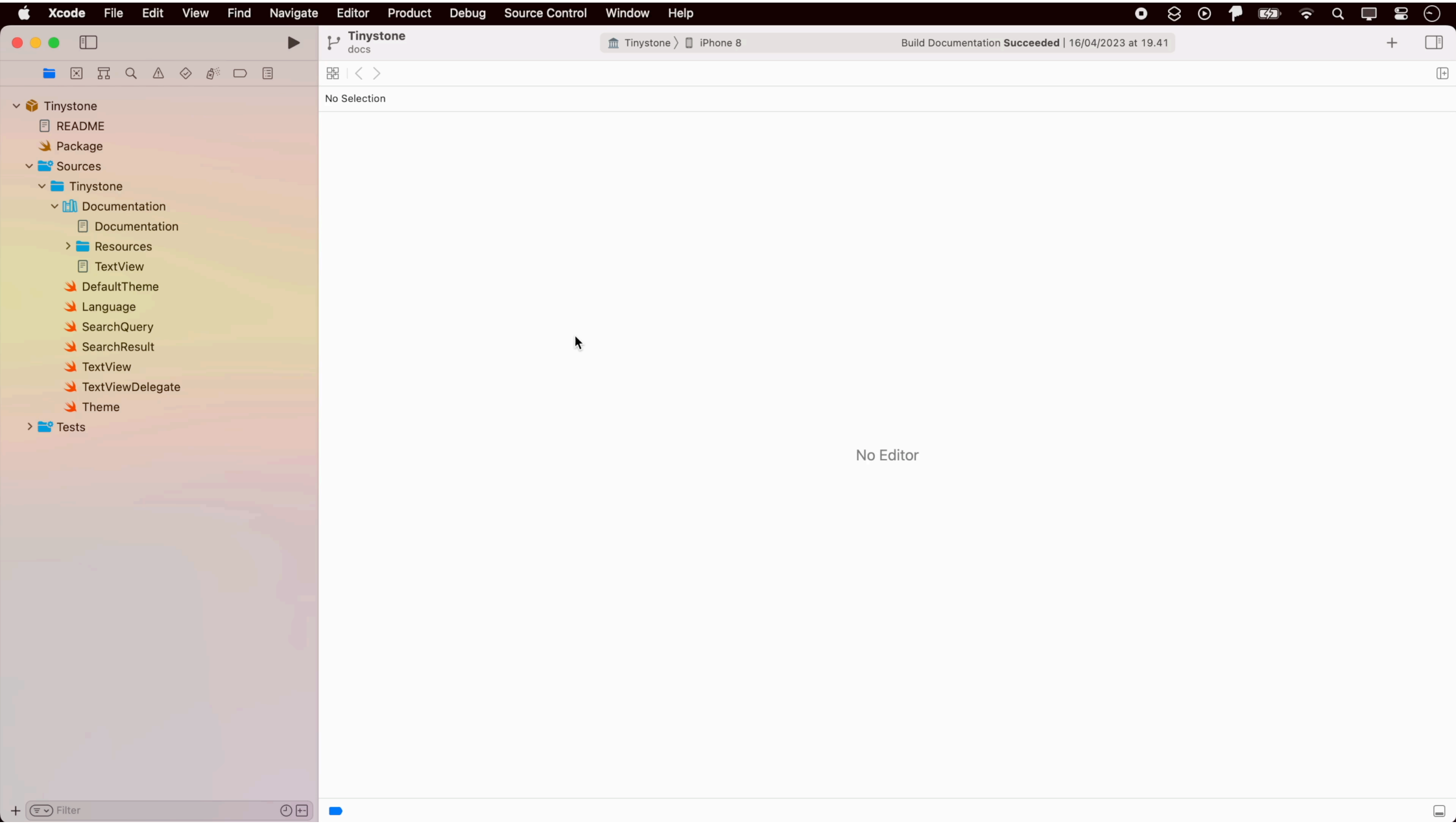
@Section

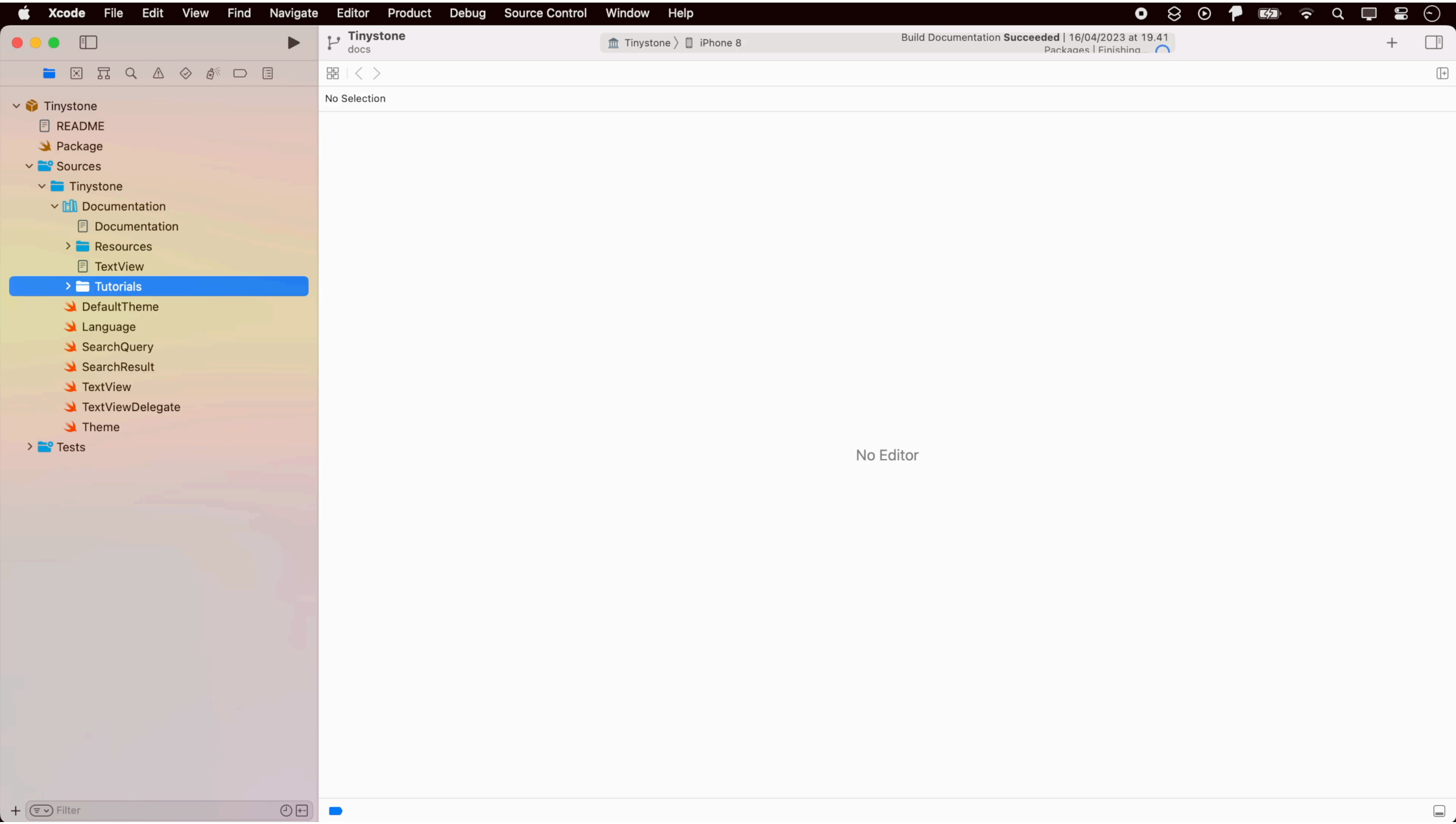
@Image

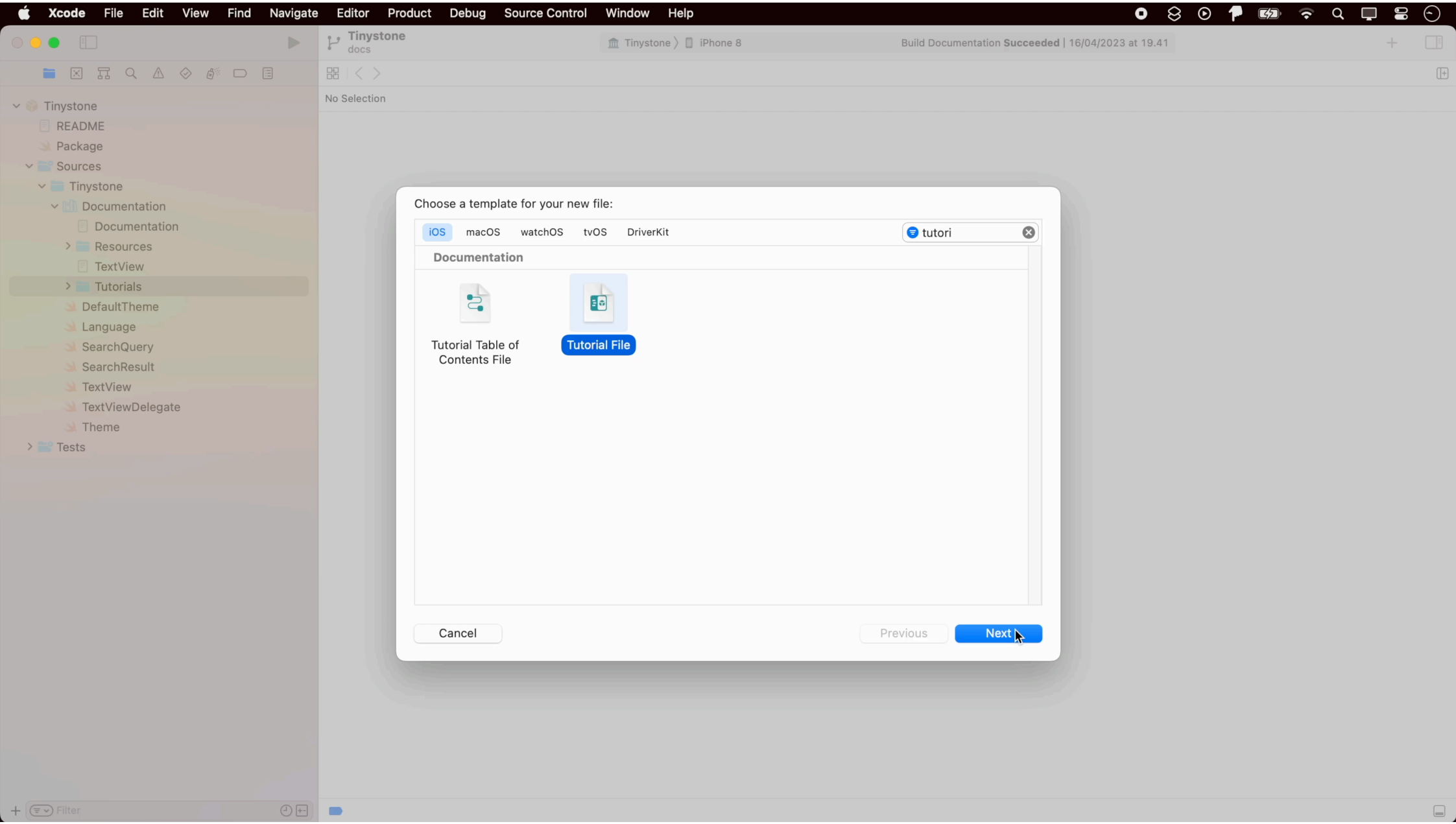
@Tutorial

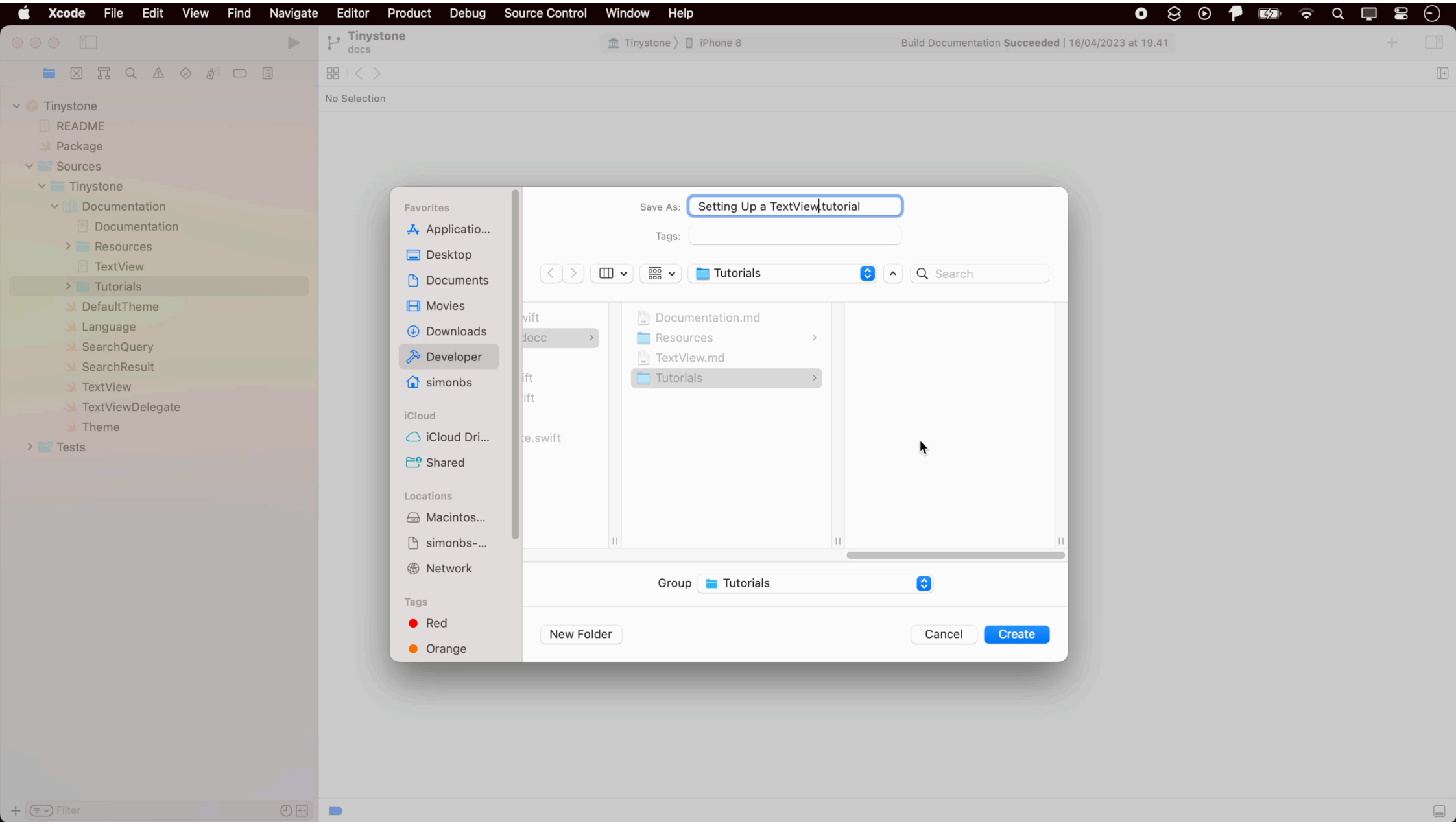
@Step

```
@Code(name: name, file: file) {
    children
}
```


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 8
Build Documentation Succeeded | 16/04/2023 at 19.41
No Selection
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
TextView
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
No Editor


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 8
Build Documentation Succeeded | 16/04/2023 at 19.41
Packages | Finishing...
No Selection
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
TextView
Tutorials
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
No Editor


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 8
Build Documentation Succeeded | 16/04/2023 at 19.41
No Selection
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
TextView
Tutorials
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Choose a template for your new file:
iOS
macOS
watchOS
tvOS
DriverKit
tutori
Documentation
Tutorial Table of Contents File
Tutorial File
Cancel
Previous
Next


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 8
Build Documentation Succeeded | 16/04/2023 at 19.41
No Selection
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
TextView
Tutorials
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
Save As: Setting Up a TextView tutorial
Tags:
Tutorials
Search
Favorites
Applicatio...
Desktop
Documents
Movies
Downloads
Developer
simonbs
iCloud
iCloud Dri...
Shared
Locations
Macintos...
simonbs-...
Network
Tags
Red
Orange
Documentation.md
Resources
TextView.md
Tutorials
Group Tutorials
New Folder
Cancel
Create

```
@Tutorial(time: number) {
   @Intro(title: "text") {
      text

      @Image(source: file, alt: "accessible description")
   }

   @Section(title: "text") {
      @ContentAndMedia {
         text

         @Image(source: file, alt: "accessible description")
      }

      @Steps {
         @Step {
            text

            @Image(source: file, alt: "accessible description")
         }

         @Step {
            text

            @Image(source: file, alt: "accessible description")
```

Setting Up a TextView.md

```
@Tutorial(time: number) {
   @Intro(title: "text") {
      text

      @Image(source: file, alt: "accessible description")
   }

   @Section(title: "text") {
      @ContentAndMedia {
         text

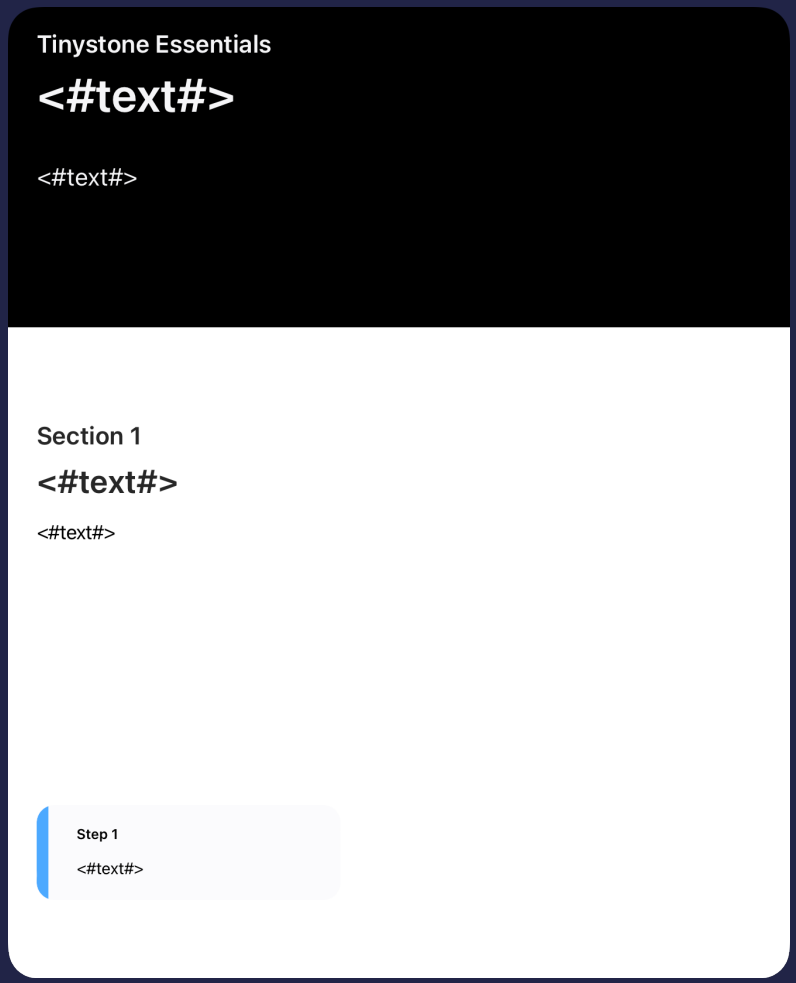
         @Image(source: file, alt: "accessible description")
      }

      @Steps {
         @Step {
            text

            @Image(source: file, alt: "accessible description")
         }

         @Step {
            text

            @Image(source: file, alt: "accessible description")
```



Setting Up a TextView.md

```
@Tutorial(time: 30) {
   @Intro(title: "text") {
      text

      @Image(source: file, alt: "accessible description")
   }

   @Section(title: "text") {
      @ContentAndMedia {
         text

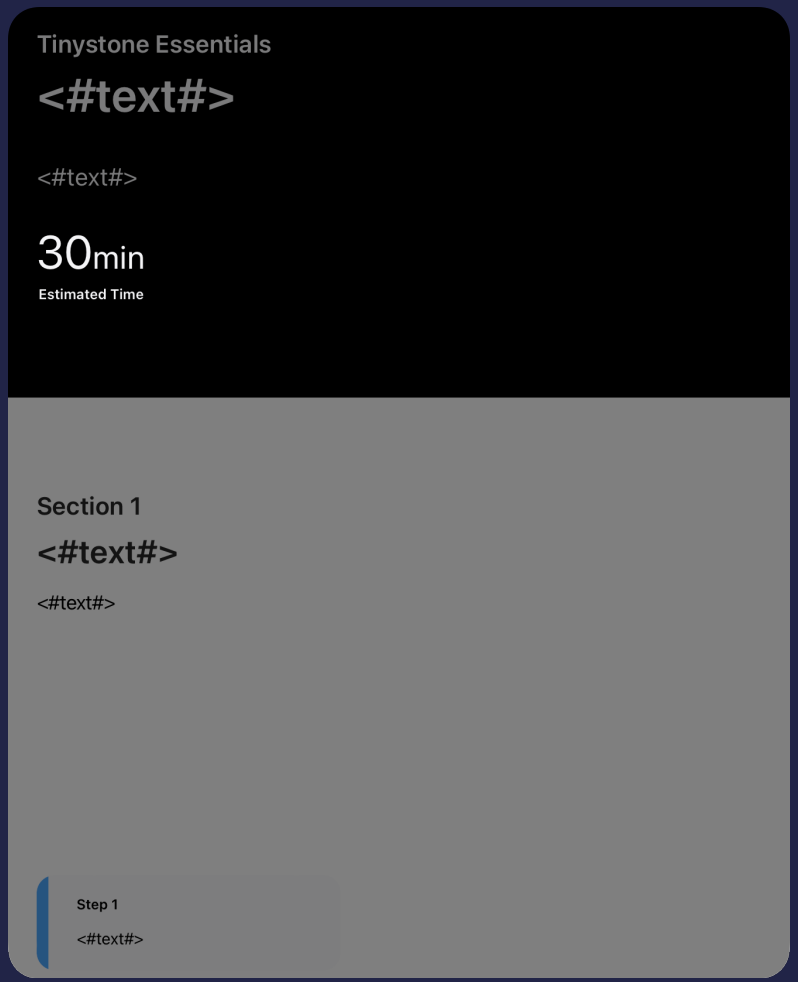
         @Image(source: file, alt: "accessible description")
      }

      @Steps {
         @Step {
            text

            @Image(source: file, alt: "accessible description")
         }

         @Step {
            text

            @Image(source: file, alt: "accessible description")
```



Setting Up a TextView.md

```
@Tutorial(time: 30) {
   @Intro(title: "Setting Up a TextView") {
      text

      @Image(source: file, alt: "accessible description")
   }

   @Section(title: "text") {
      @ContentAndMedia {
         text

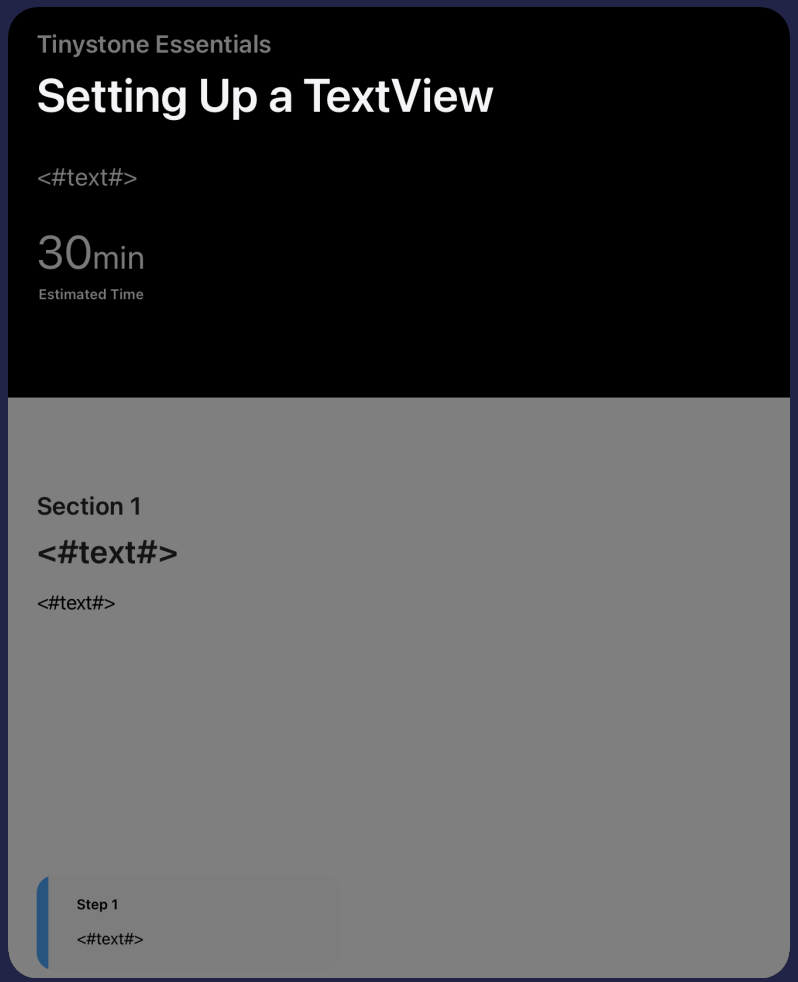
         @Image(source: file, alt: "accessible description")
      }

      @Steps {
         @Step {
            text

            @Image(source: file, alt: "accessible description")
         }

         @Step {
            text

            @Image(source: file, alt: "accessible description")
```



Setting Up a TextView.md

```
@Tutorial(time: 30) {
   @Intro(title: "Setting Up a TextView") {
      This tutorial guides you through creating a TextView and
      configuring it.

      @Image(source: file, alt: "accessible description")
   }

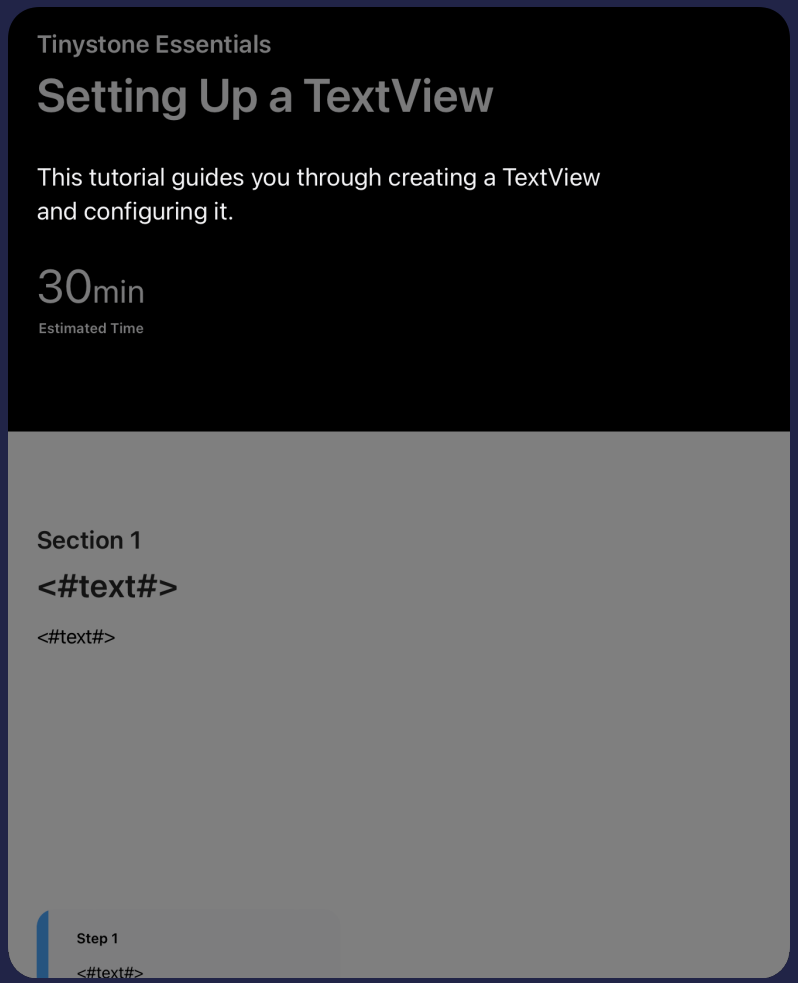
   @Section(title: "text") {
      @ContentAndMedia {
         text

         @Image(source: file, alt: "accessible description")
      }

      @Steps {
         @Step {
            text

            @Image(source: file, alt: "accessible description")
         }

         @Step {
            text
```



Setting Up a TextView.md

```
@Tutorial(time: 30) {
   @Intro(title: "Setting Up a TextView") {
      This tutorial guides you through creating a TextView and
      configuring it.

      @Image(source: header.png, alt: "Code snippets in grid")
   }

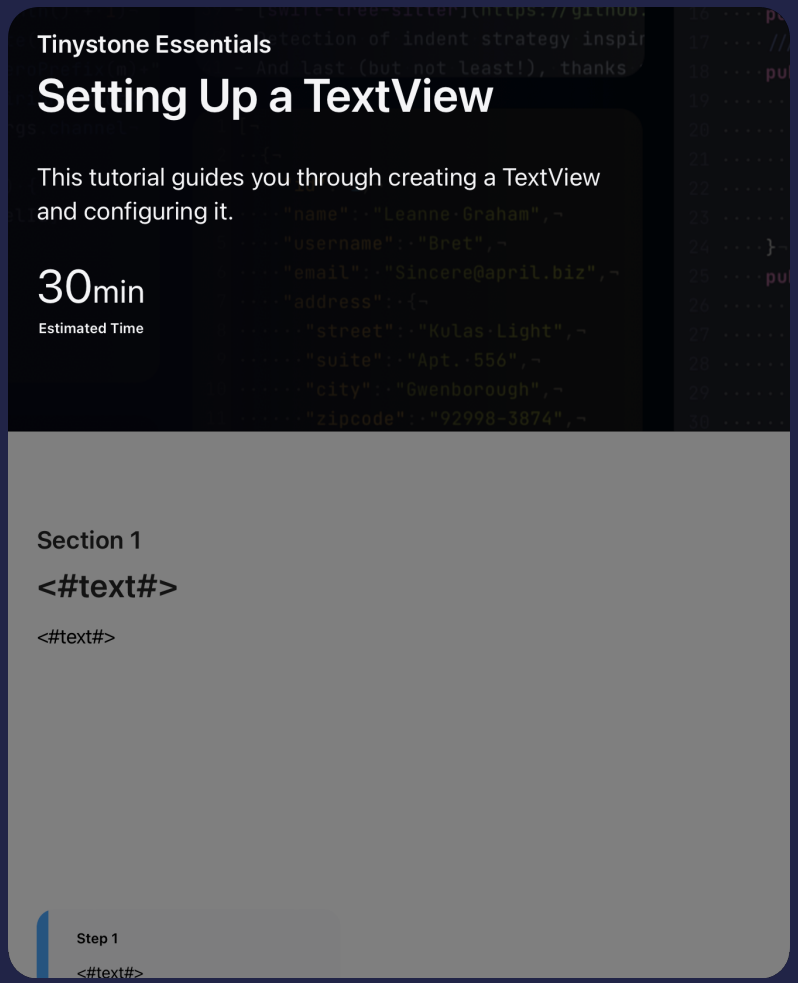
   @Section(title: "text") {
      @ContentAndMedia {
         text

         @Image(source: file, alt: "accessible description")
      }

      @Steps {
         @Step {
            text

            @Image(source: file, alt: "accessible description")
         }

         @Step {
            text
```



Setting Up a TextView.md

```
@Tutorial(time: 30) {
   @Intro(title: "Setting Up a TextView") {
      This tutorial guides you through creating a TextView and
      configuring it.

      @Image(source: header.png, alt: "Code snippets in grid")
   }

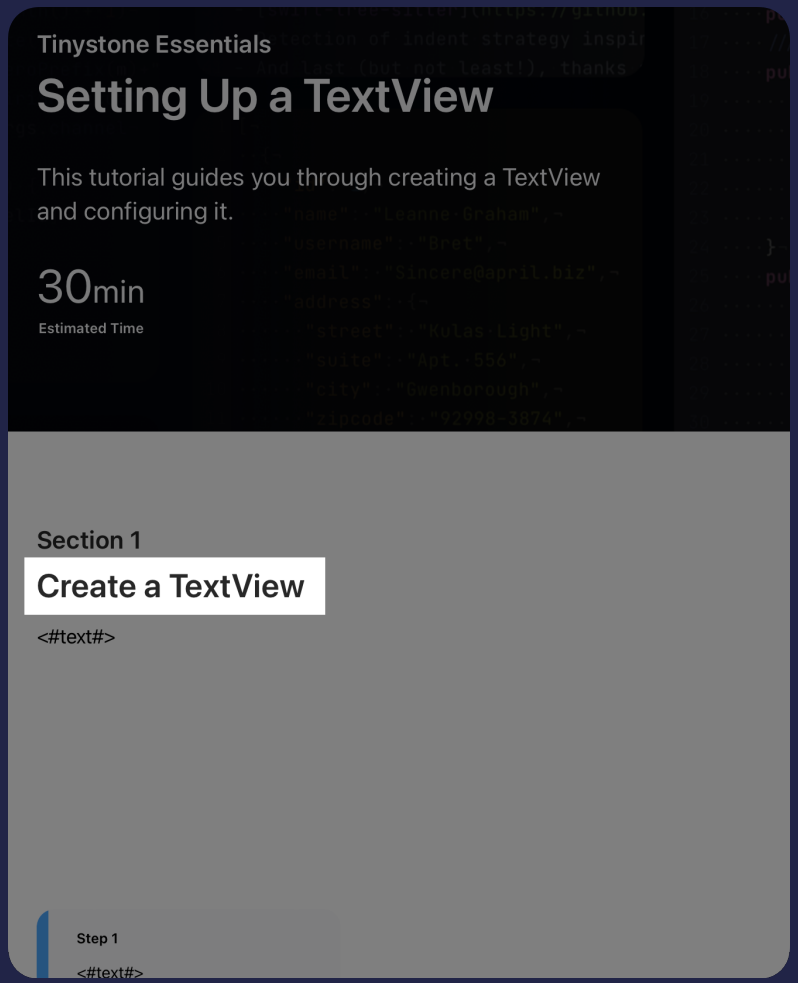
   @Section(title: "Create a TextView") {
      @ContentAndMedia {
         text

         @Image(source: file, alt: "accessible description")
      }

      @Steps {
         @Step {
            text

            @Image(source: file, alt: "accessible description")
         }

         @Step {
            text
```



Setting Up a TextView.md

```
@Tutorial(time: 30) {
   @Intro(title: "Setting Up a TextView") {
      This tutorial guides you through creating a TextView and
      configuring it.

      @Image(source: header.png, alt: "Code snippets in grid")
   }

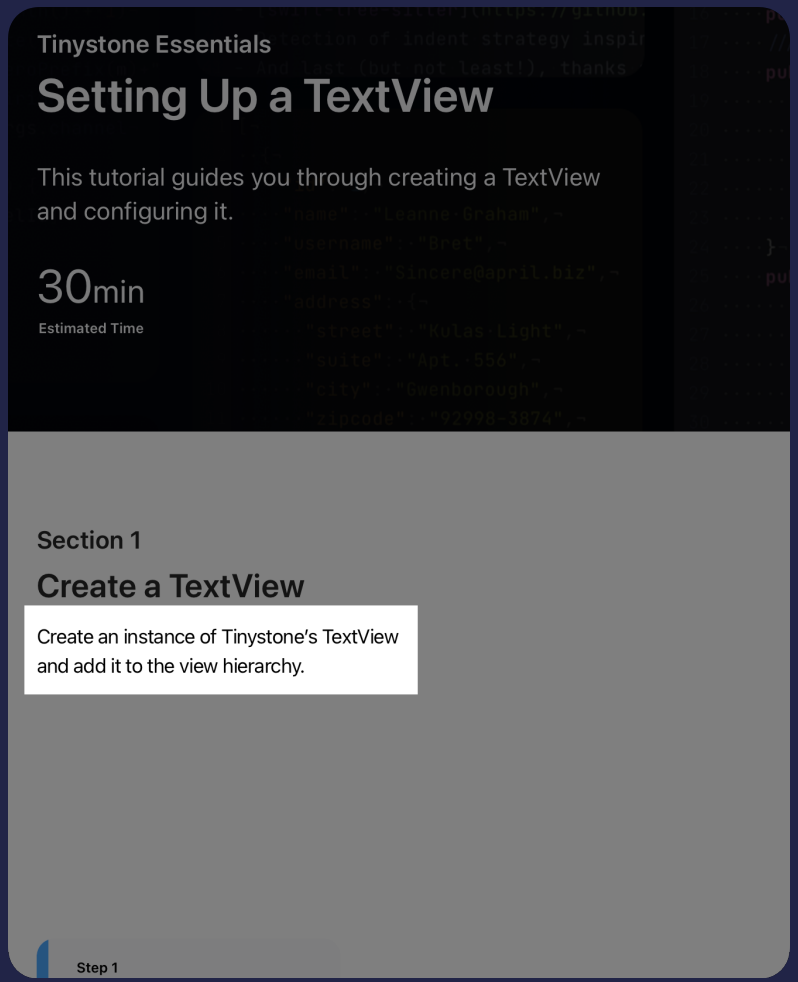
   @Section(title: "Create a TextView") {
      @ContentAndMedia {
         Create an instance of Tinystone's TextView and add it to
         the view hierarchy.

         @Image(source: file, alt: "accessible description")
      }

      @Steps {
         @Step {
            text

            @Image(source: file, alt: "accessible description")
         }

         @Step {
            text
```



Setting Up a TextView.md

```
@Tutorial(time: 30) {
   @Intro(title: "Setting Up a TextView") {
      This tutorial guides you through creating a TextView and
      configuring it.

      @Image(source: header.png, alt: "Code snippets in grid")
   }

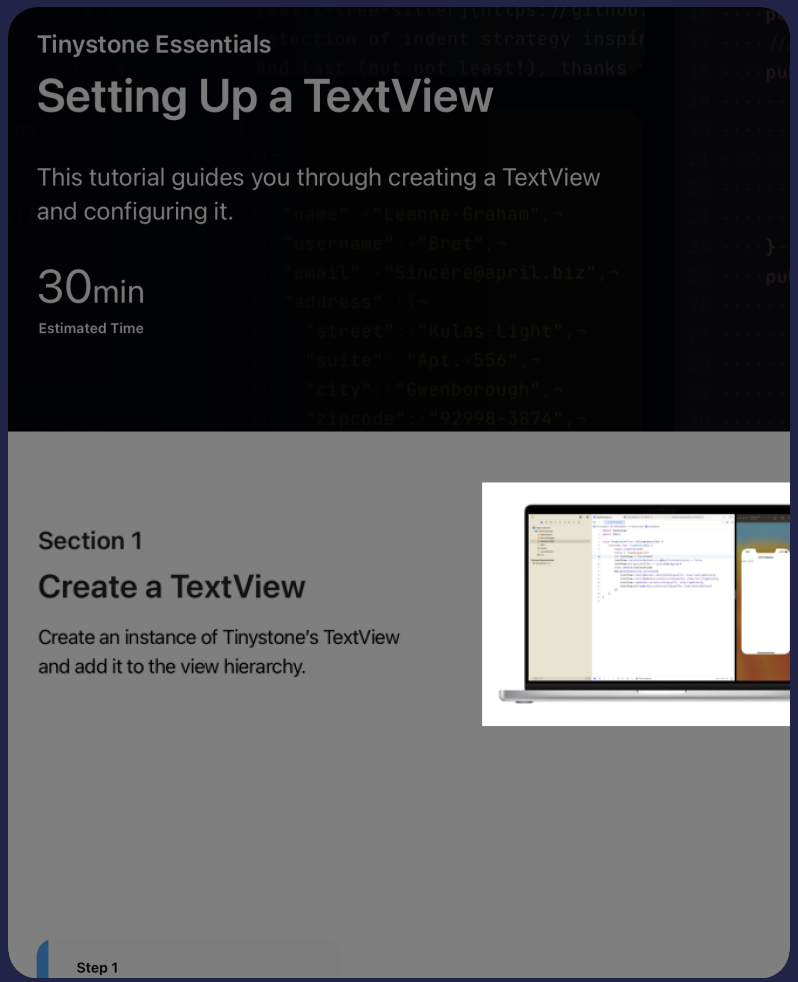
   @Section(title: "Create a TextView") {
      @ContentAndMedia {
         Create an instance of Tinystone's TextView and add it to
         the view hierarchy.

         @Image(source: section1.png, alt: "Xcode on a MacBook")
      }

      @Steps {
         @Step {
            text

            @Image(source: file, alt: "accessible description")
         }

         @Step {
            text
```



Setting Up a TextView.md

```
@Tutorial(time: 30) {
   @Intro(title: "Setting Up a TextView") {
      This tutorial guides you through creating a TextView and
      configuring it.

      @Image(source: header.png, alt: "Code snippets in grid")
   }

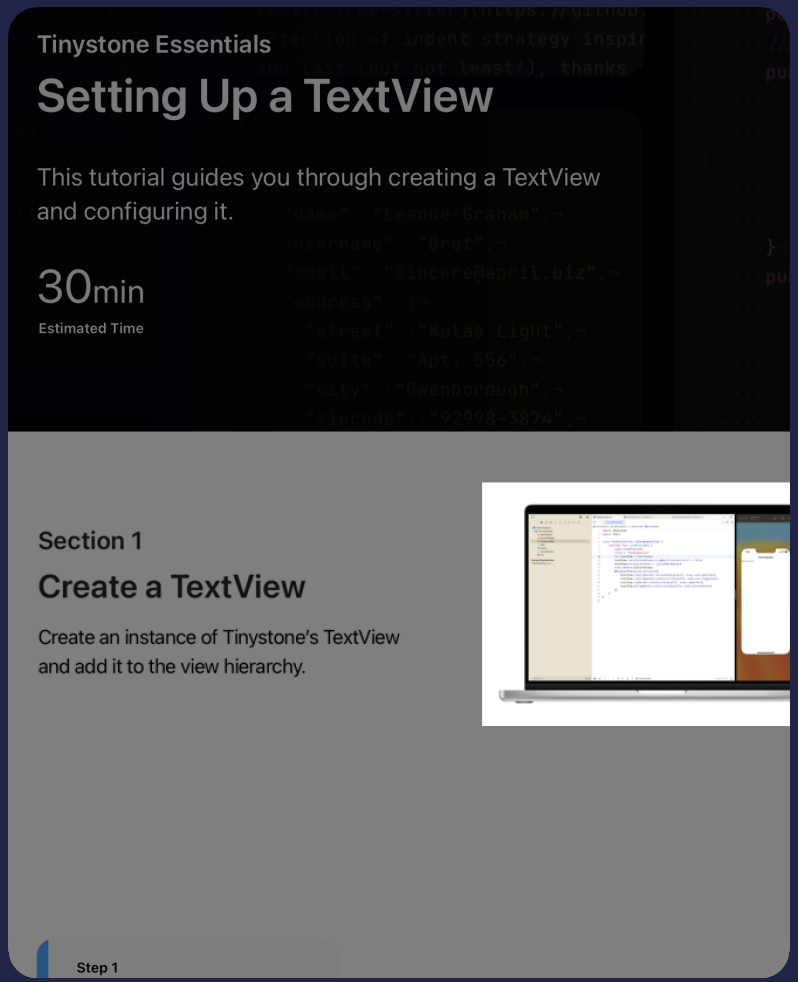
   @Section(title: "Create a TextView") {
      @ContentAndMedia {
         Create an instance of Tinystone's TextView and add it to
         the view hierarchy.

         @Image(source: section1.png, alt: "Xcode on a MacBook")
      }

      @Steps {
         @Step {
            text

            @Image(source: file, alt: "accessible description")
         }

         @Step {
            text
```



Setting Up a TextView.md

```
        Create an instance of Tinystone's TextView and add it to
        the view hierarchy.

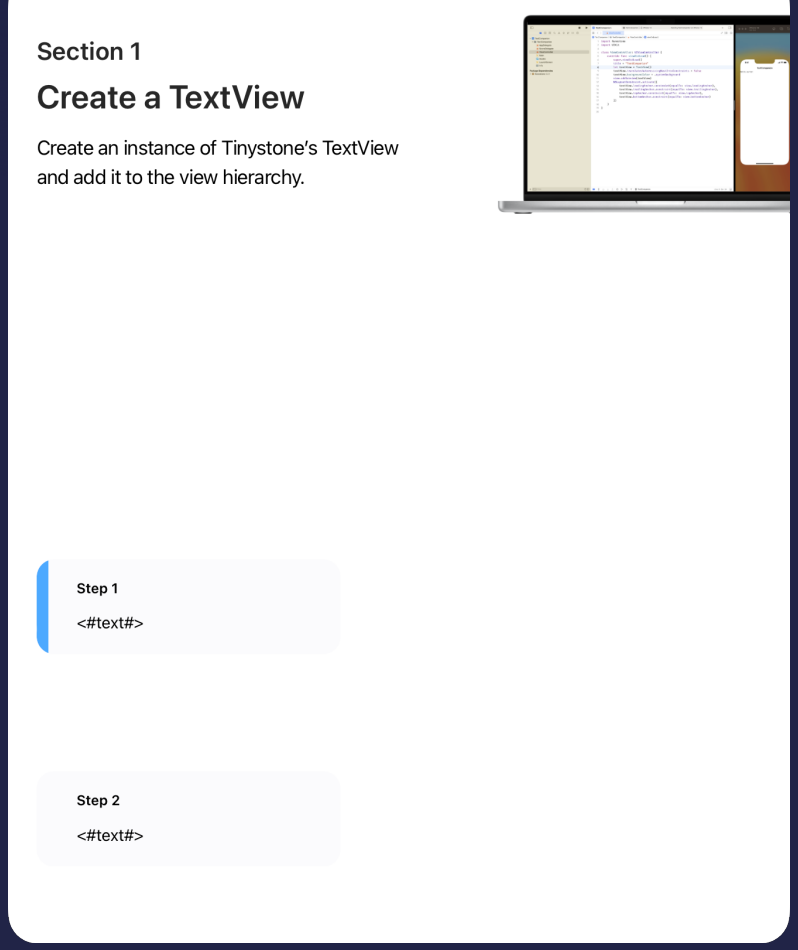
        @Image(source: section1.png, alt: "Xcode on a MacBook")
    }

    @Steps {
        @Step {
            text

            @Image(source: file, alt: "accessible description")
        }

        @Step {
            text

            @Image(source: file, alt: "accessible description")
        }
    }
  }
}
```



Setting Up a TextView.md

```
        the view hierarchy.

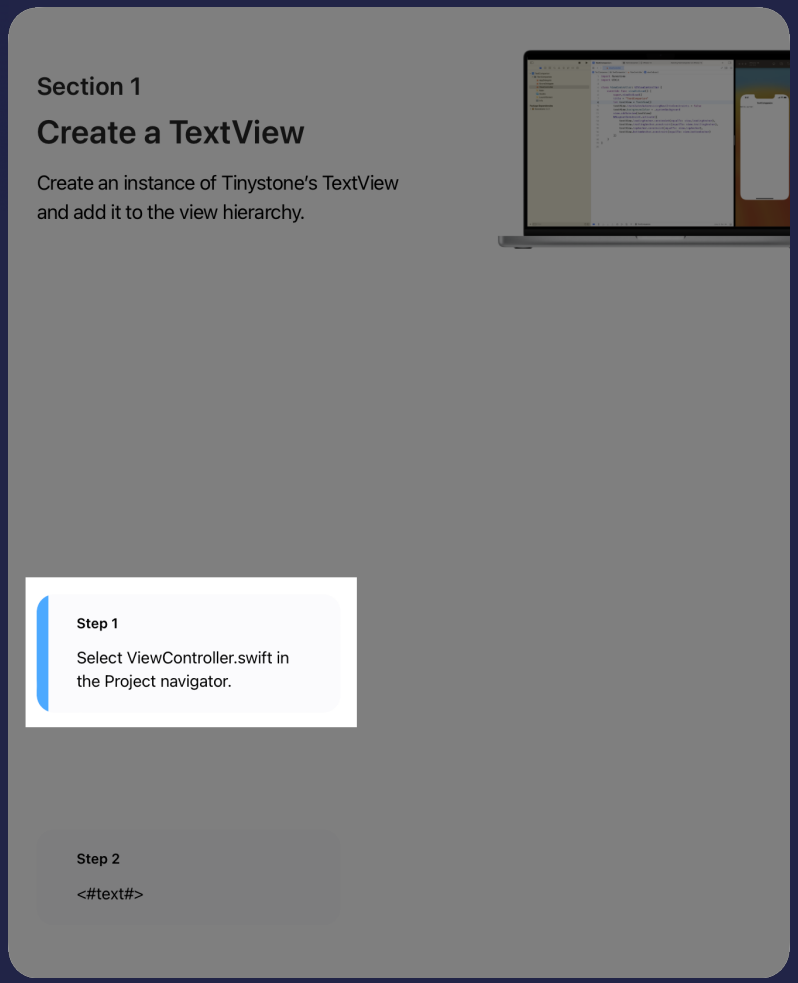
        @Image(source: section1.png, alt: "Xcode on a MacBook")
    }

    @Steps {
        @Step {
            Select ViewController.swift in the Project navigator.

            @Image(source: file, alt: "accessible description")
        }

        @Step {
            text

            @Image(source: file, alt: "accessible description")
        }
    }
  }
}
```



Setting Up a TextView.md

```
        the view hierarchy.

        @Image(source: section1.png, alt: "Xcode on a MacBook")
    }

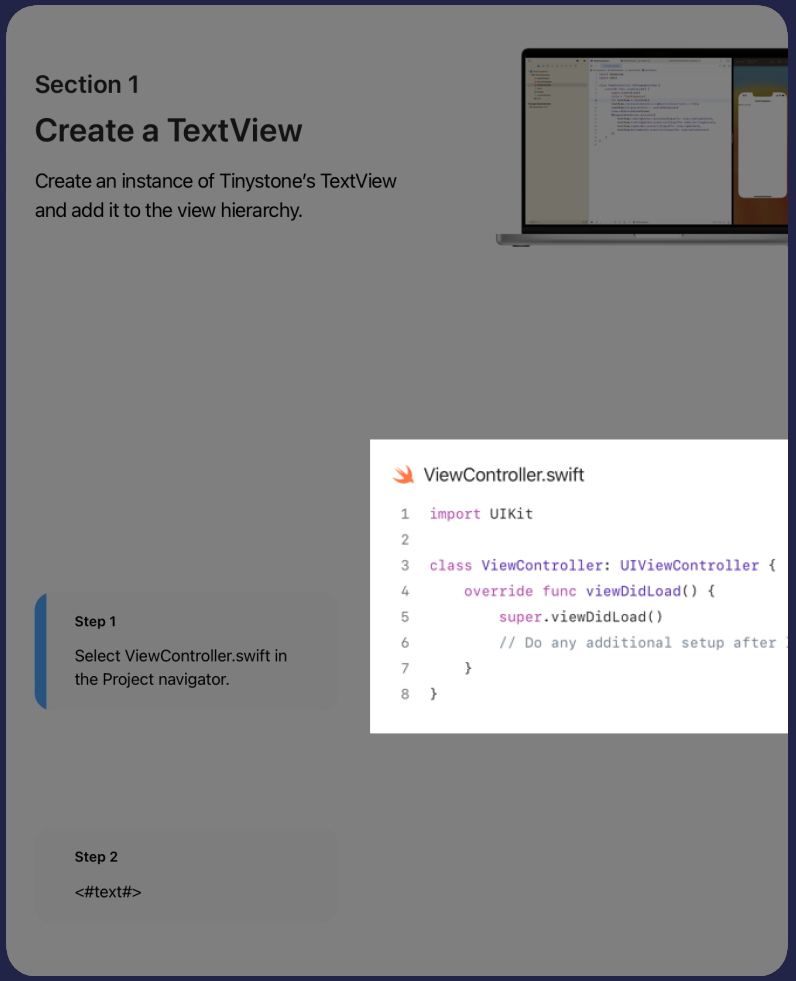
    @Steps {
        @Step {
            Select ViewController.swift in the Project navigator.

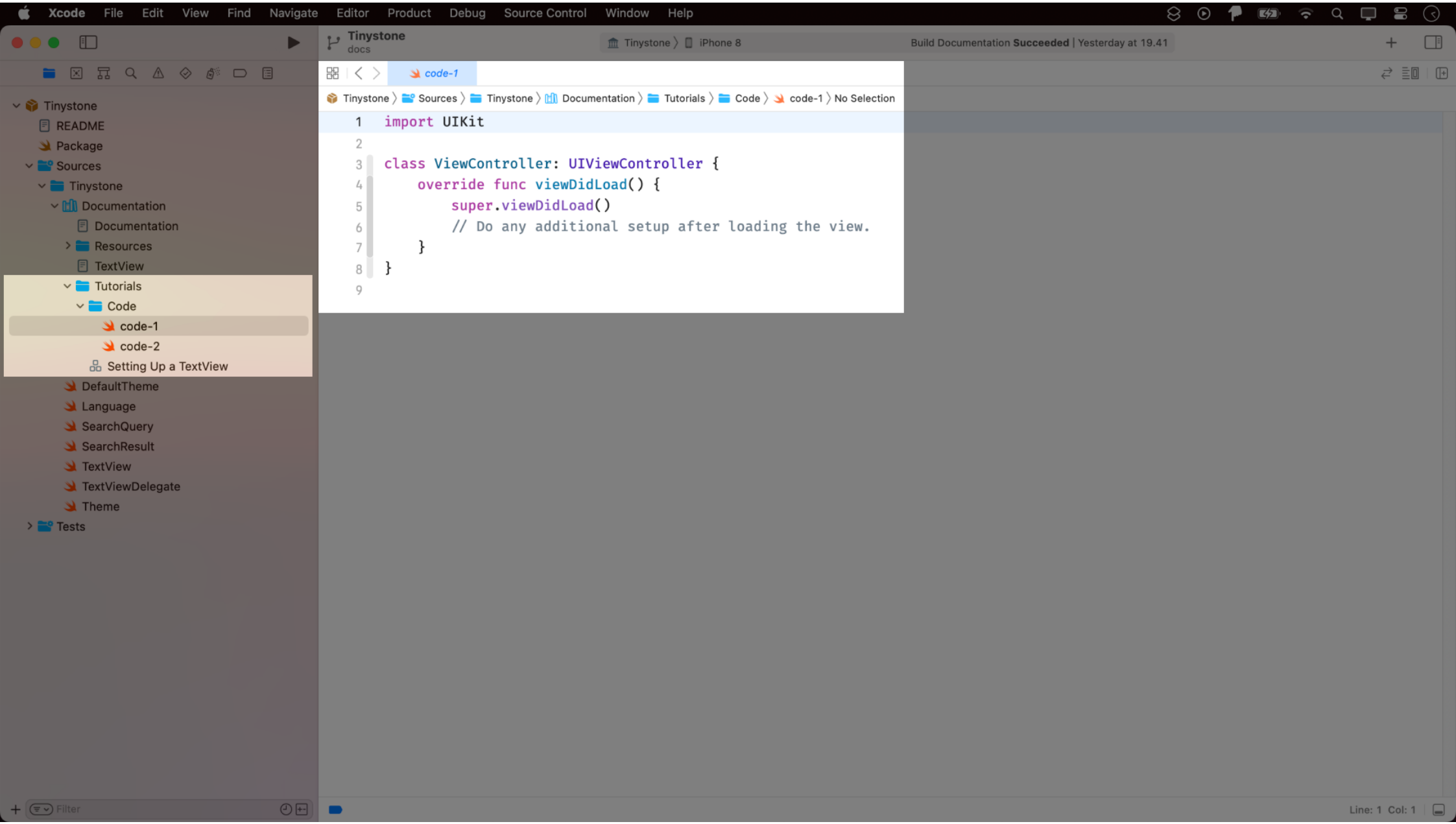
            @Code(name: "ViewContr.swift", file: code-1.swift) {

            }
        }

        @Step {
            text

            @Image(source: file, alt: "accessible description")
        }
    }
  }
}
```




Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 8
Build Documentation Succeeded | Yesterday at 19.41
code-1
Tinystone 〉 Sources 〉 Tinystone 〉 Documentation 〉 Tutorials 〉 Code 〉 code-1 〉 No Selection
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
TextView
Tutorials
Code
code-1
code-2
Setting Up a TextView
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
Filter
1 import UIKit
2
3 class ViewController: UIViewController {
4 override func viewDidLoad() {
5 super.viewDidLoad()
6 // Do any additional setup after loading the view.
7 }
8 }
9
Line: 1 Col: 1

Setting Up a TextView.md

```
        the view hierarchy.

        @Image(source: section1.png, alt: "Xcode on a MacBook")
    }

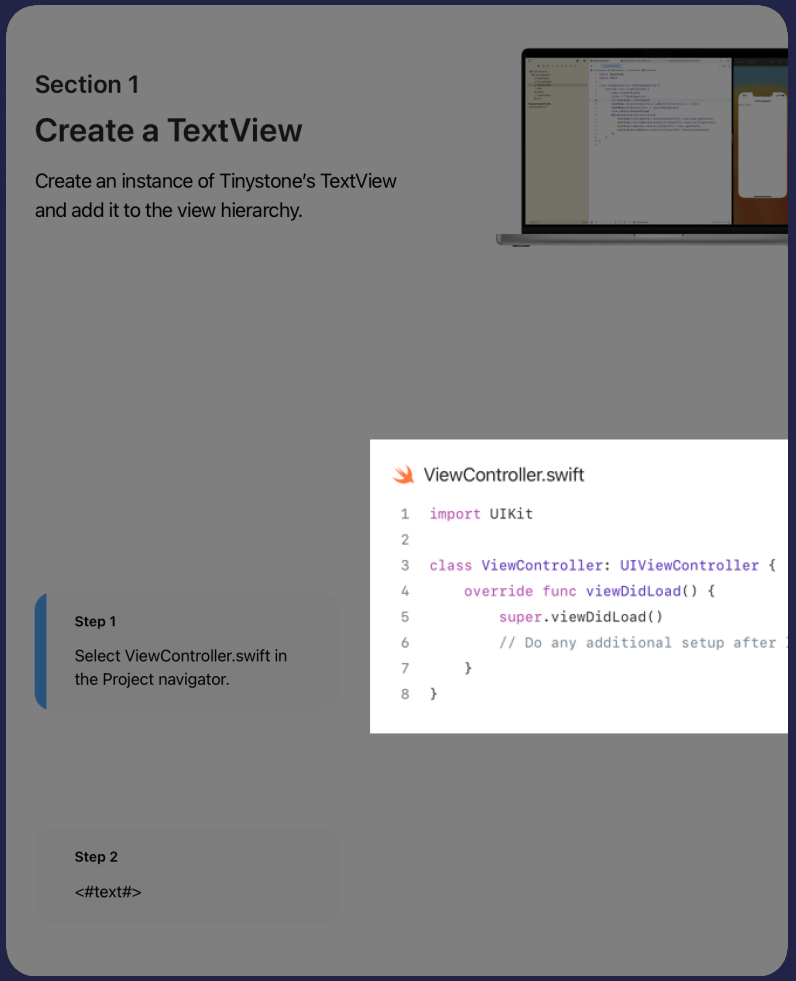
    @Steps {
        @Step {
            Select ViewController.swift in the Project navigator.

            @Code(name: "ViewContr.swift", file: code-1.swift) {

            }
        }

        @Step {
            text

            @Image(source: file, alt: "accessible description")
        }
    }
  }
}
```



Setting Up a TextView.md

```
        Create an instance of Tinystone's TextView and add it to
        the view hierarchy.

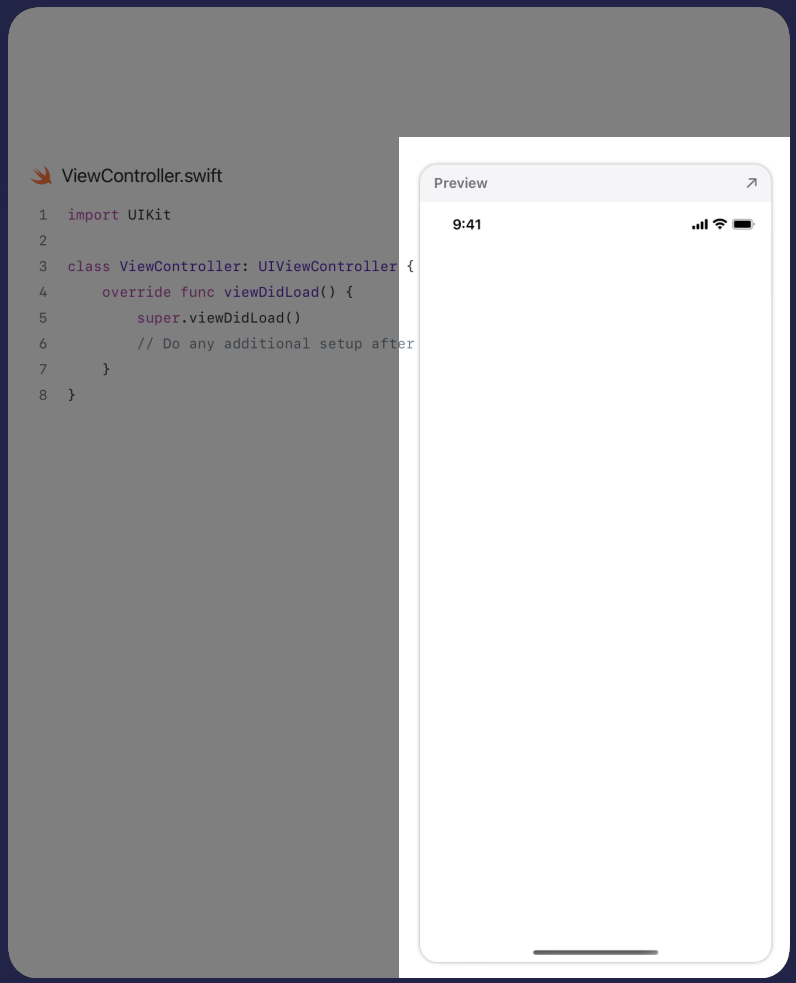
        @Image(source: section1.png, alt: "Xcode on a MacBook")
    }

    @Steps {
        @Step {
            Select ViewController.swift in the Project navigator.

            @Code(name: "ViewContr.swift", file: code-1.swift) {
                @Image(source: preview.png, "Blank screen")
            }
        }

        @Step {
            text

            @Image(source: file, alt: "accessible description")
        }
    }
  }
}
```



Setting Up a TextView.md

```
        Create an instance of Tinystone's TextView and add it to
        the view hierarchy.

        @Image(source: section1.png, alt: "Xcode on a MacBook")
    }

    @Steps {
        @Step {
            Select ViewController.swift in the Project navigator.

            @Code(name: "ViewContr.swift", file: code-1.swift) {
                @Image(source: preview.png, "Blank screen")
            }
        }

        @Step {
            Add an instance of TextView to the view.

            @Code(name: "ViewContr.swift", file: code-2.swift) {
                @Image(source: preview-2.png, "Sample text view")
            }
        }
    }
  }
}
```

Setting Up a TextView.md

```
        the view hierarchy.

        @Image(source: section1.png, alt: "Xcode on a MacBook")
    }

    @Steps {
        @Step {
            Select ViewController.swift in the Project navigator.

            @Code(name: "ViewContr.swift", file: code-1.swift) {
                @Image(source: preview.png, "Blank screen")
            }
        }

        @Step {
            Add an instance of TextView to the view.

            @Code(name: "ViewContr.swift", file: code-2.swift) {
                @Image(source: preview-2.png, "Sample text view")
            }
        }
    }
  }
}
```

Setting Up a TextView.md

```
        Create an instance of Tinystone's TextView and add it to
        the view hierarchy.

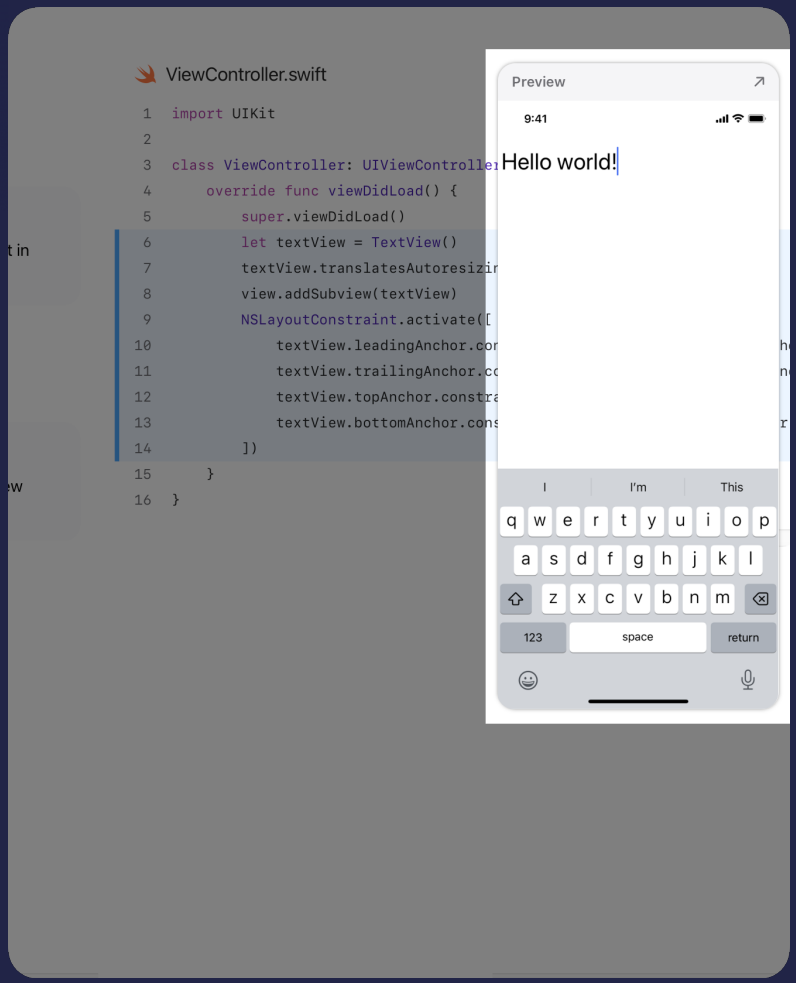
        @Image(source: section1.png, alt: "Xcode on a MacBook")
    }

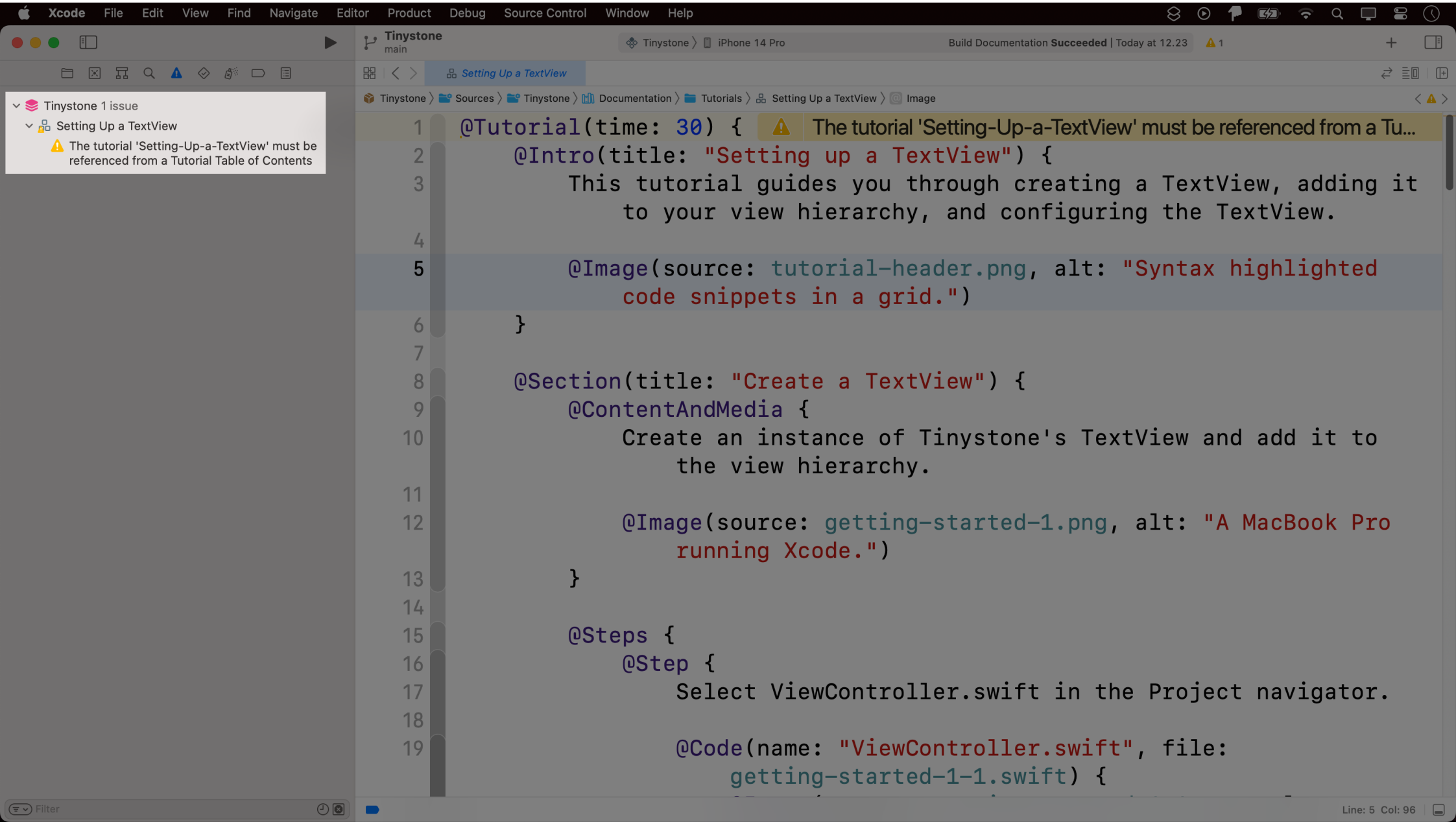
    @Steps {
        @Step {
            Select ViewController.swift in the Project navigator.

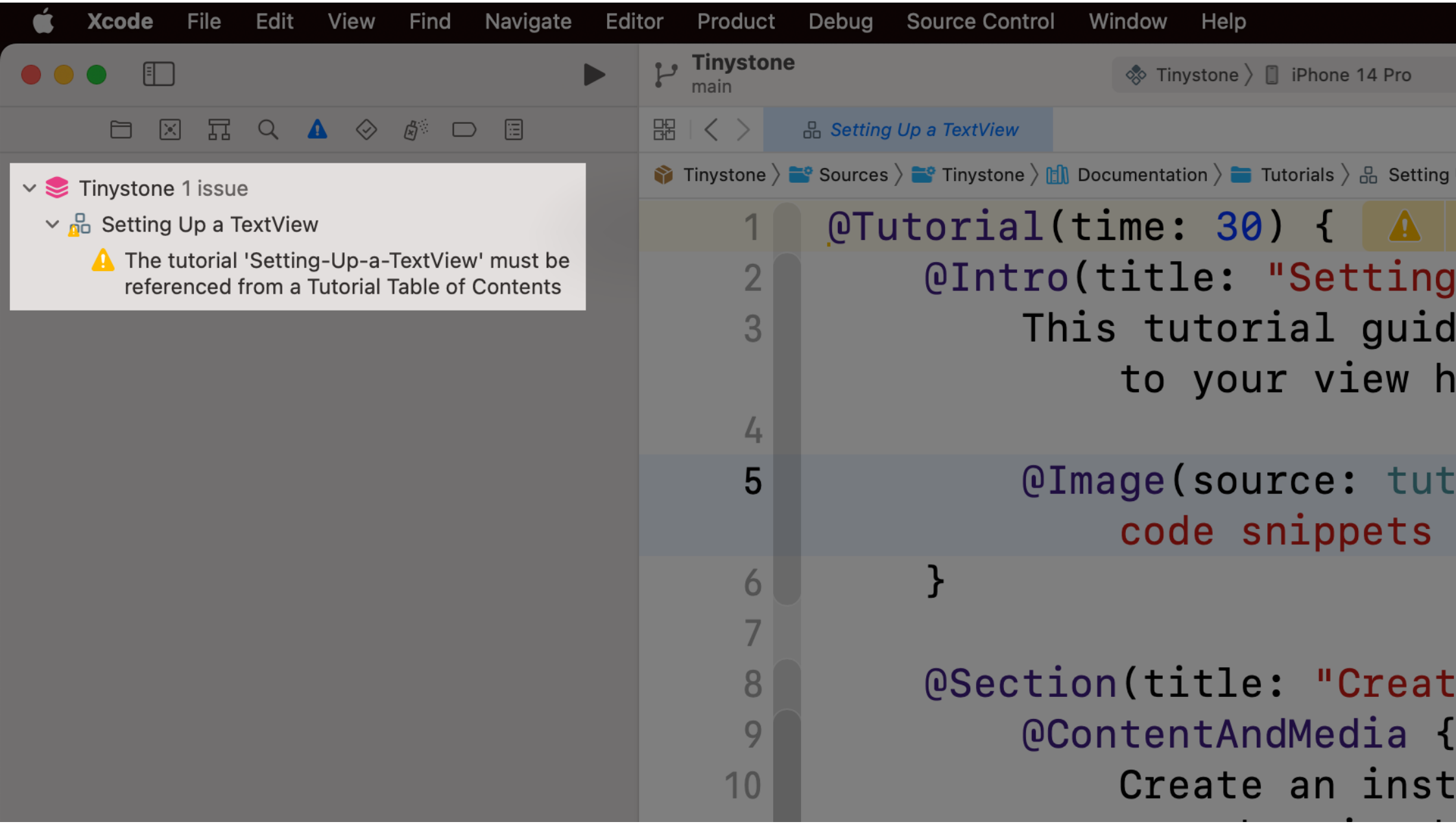
            @Code(name: "ViewContr.swift", file: code-1.swift) {
                @Image(source: preview.png, "Blank screen")
            }
        }

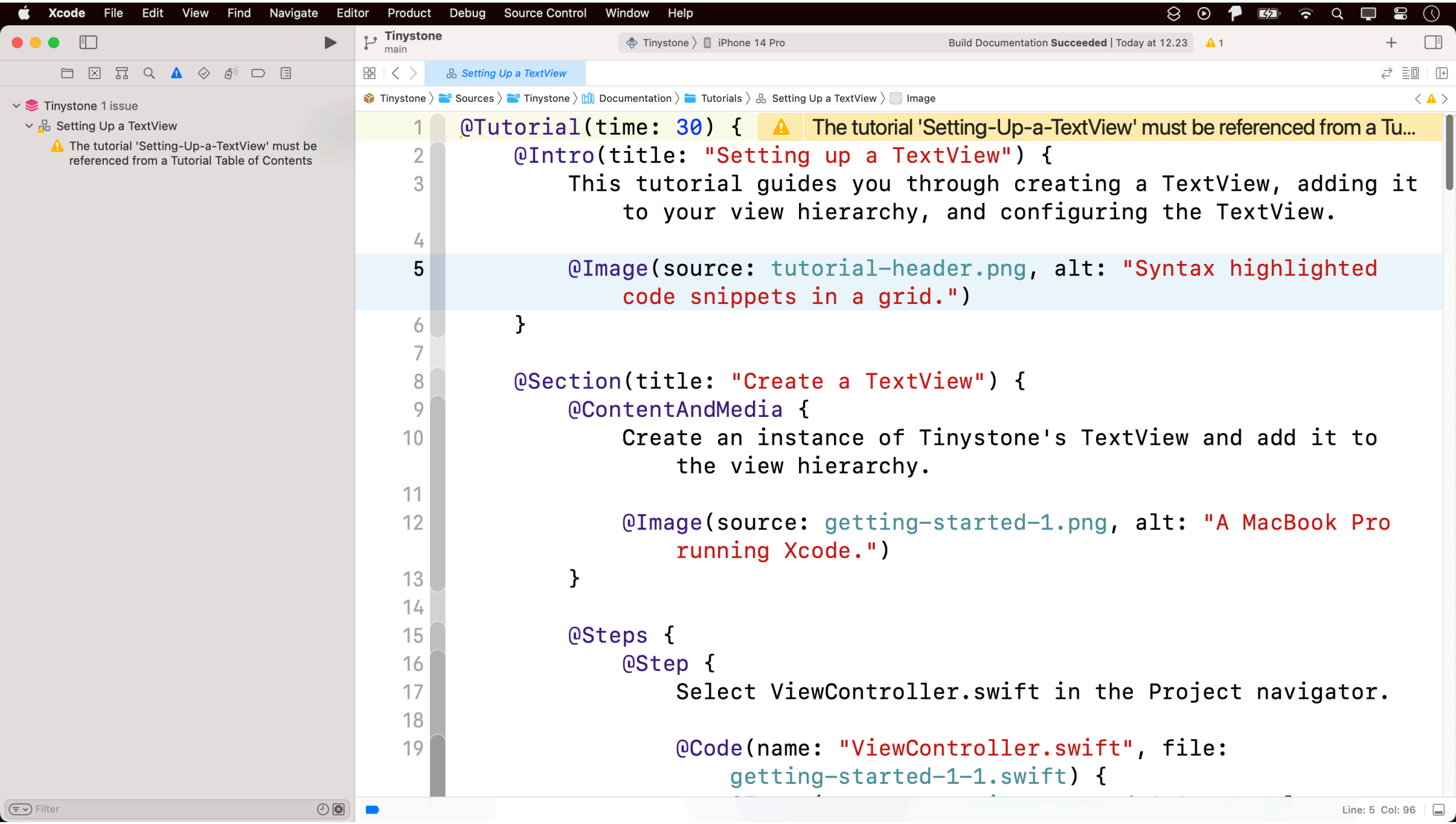
        @Step {
            Add an instance of TextView to the view.

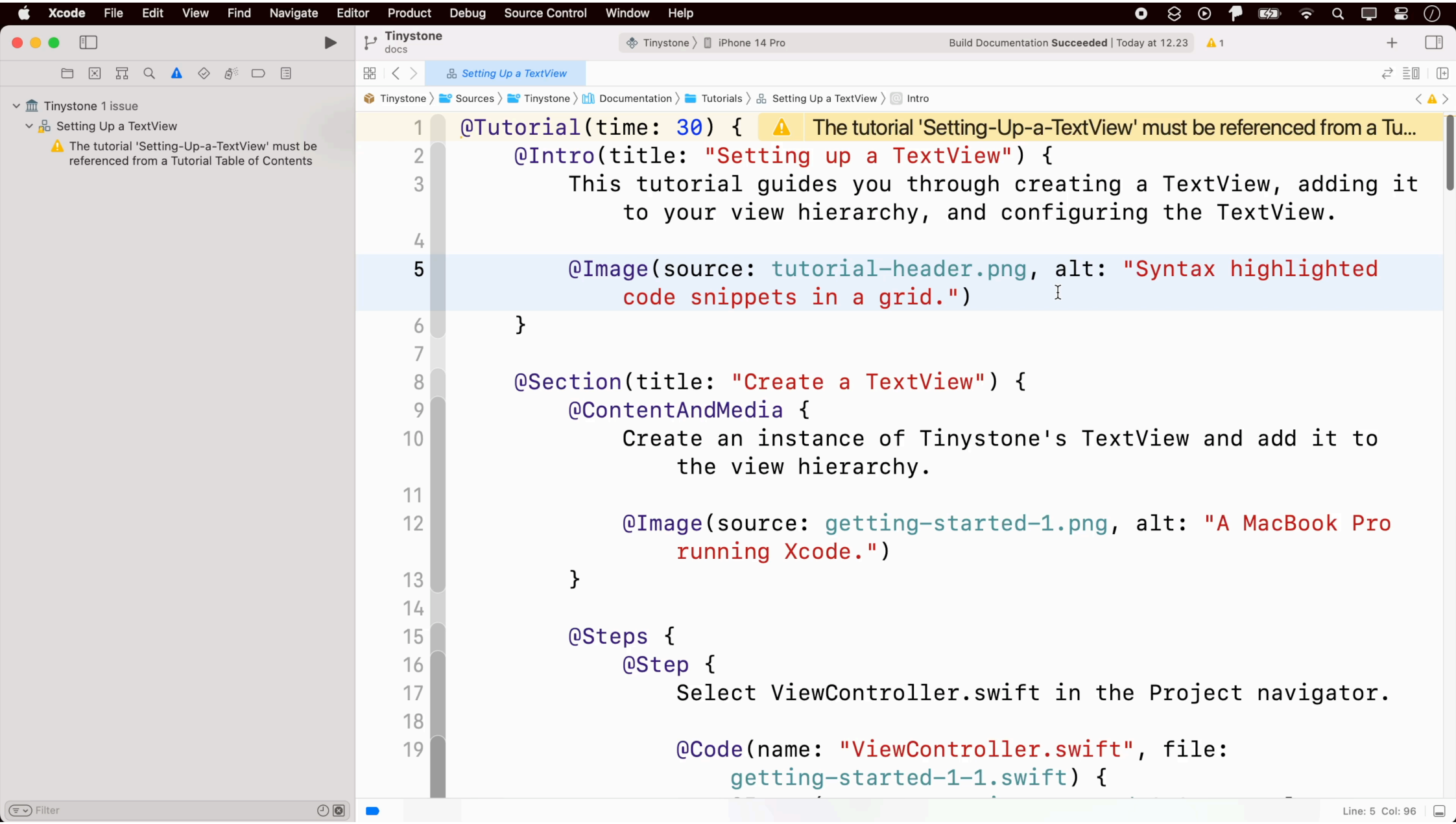
            @Code(name: "ViewContr.swift", file: code-2.swift) {
                @Image(source: preview-2.png, "Sample text view")
            }
        }
    }
  }
}
```

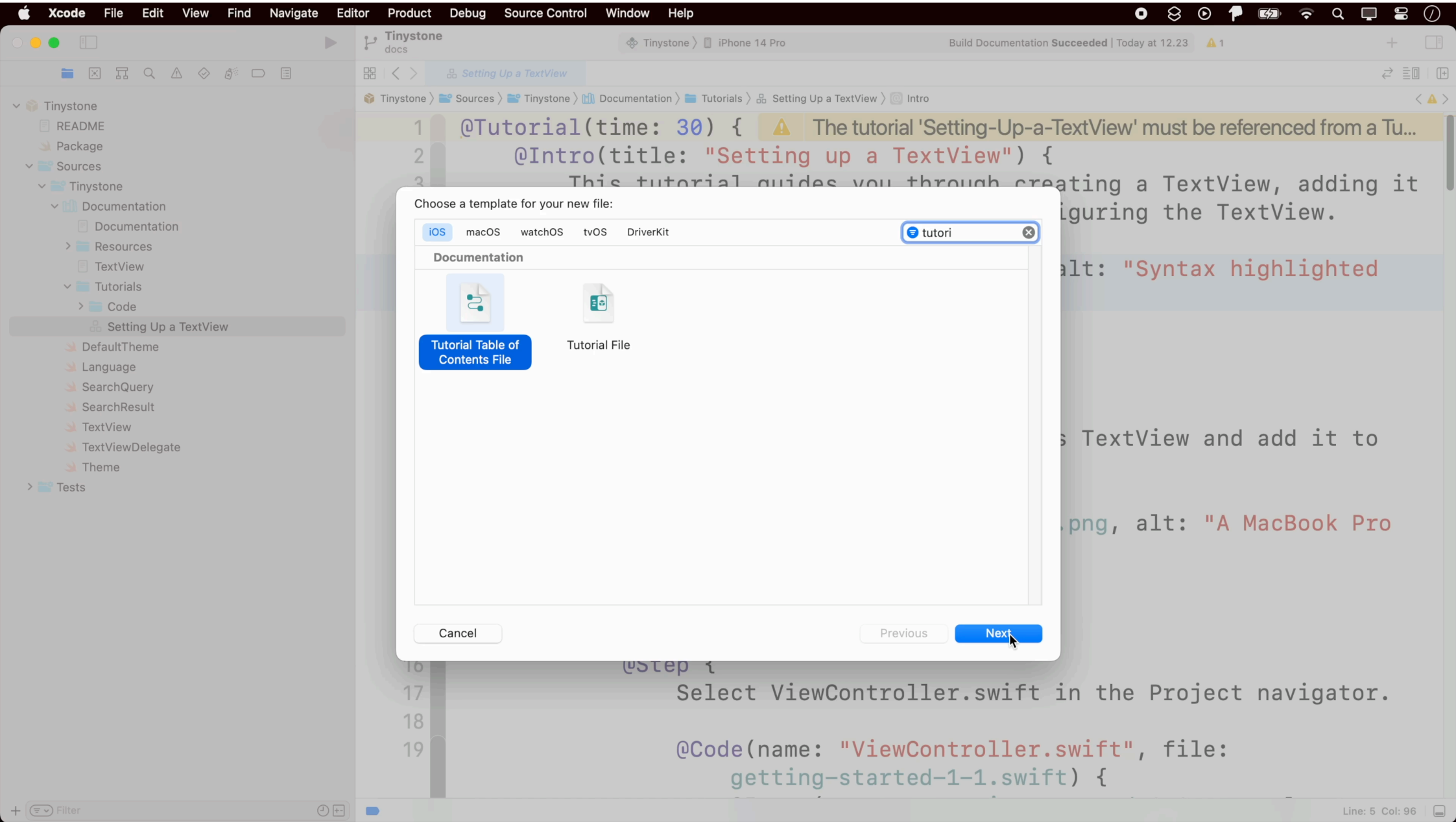


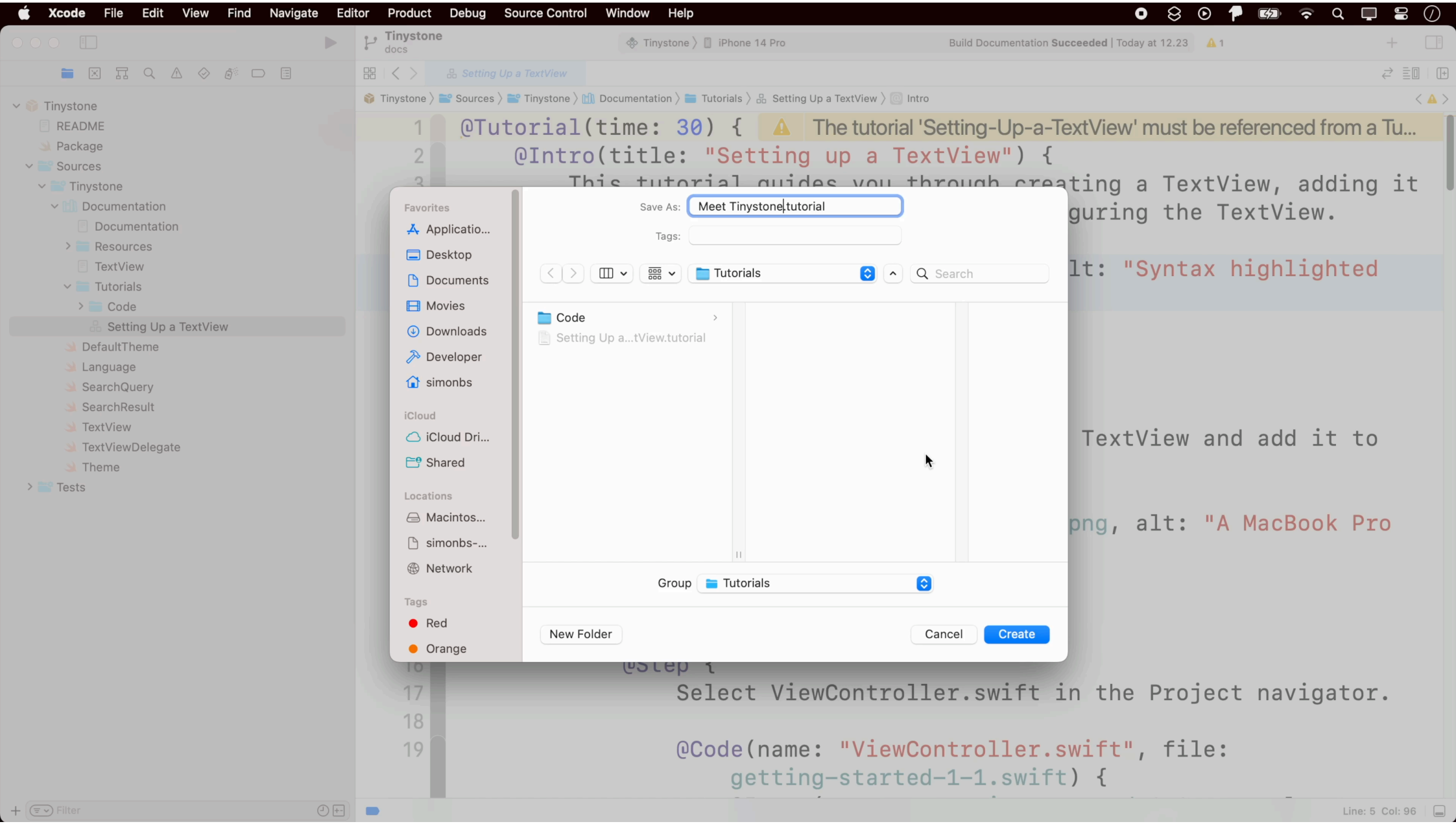

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
main
Tinystone 〉 iPhone 14 Pro
Build Documentation Succeeded | Today at 12.23
1
Setting Up a TextView
Tinystone 〉 Sources 〉 Tinystone 〉 Documentation 〉 Tutorials 〉 Setting Up a TextView 〉 Image
Tinystone 1 issue
Setting Up a TextView
The tutorial 'Setting-Up-a-TextView' must be referenced from a Tutorial Table of Contents
The tutorial 'Setting-Up-a-TextView' must be referenced from a Tu...
@Tutorial(time: 30) {
@Intro(title: "Setting up a TextView") {
This tutorial guides you through creating a TextView, adding it to your view hierarchy, and configuring the TextView.
@Image(source: tutorial-header.png, alt: "Syntax highlighted code snippets in a grid.")
}
@Section(title: "Create a TextView") {
@ContentAndMedia {
Create an instance of Tinystone's TextView and add it to the view hierarchy.
@Image(source: getting-started-1.png, alt: "A MacBook Pro running Xcode.")
}
@Steps {
@Step {
Select ViewController.swift in the Project navigator.
@Code(name: "ViewController.swift", file: getting-started-1-1.swift) {
Filter
Line: 5 Col: 96


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
main
Tinystone 〉 iPhone 14 Pro
Setting Up a TextView
Tinystone 〉 Sources 〉 Tinystone 〉 Documentation 〉 Tutorials 〉 Setting
Tinystone 1 issue
Setting Up a TextView
The tutorial 'Setting-Up-a-TextView' must be referenced from a Tutorial Table of Contents
1 @Tutorial(time: 30) {
2 @Intro(title: "Setting
3 This tutorial guid
to your view h
4
5 @Image(source: tut
code snippets
6 }
7
8 @Section(title: "Creat
9 @ContentAndMedia {
10 Create an inst


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
main
Tinystone 〉 iPhone 14 Pro
Build Documentation Succeeded | Today at 12.23
1
Setting Up a TextView
Tinystone 〉 Sources 〉 Tinystone 〉 Documentation 〉 Tutorials 〉 Setting Up a TextView 〉 Image
Tinystone 1 issue
Setting Up a TextView
The tutorial 'Setting-Up-a-TextView' must be referenced from a Tutorial Table of Contents
The tutorial 'Setting-Up-a-TextView' must be referenced from a Tu...
@Tutorial(time: 30) {
@Intro(title: "Setting up a TextView") {
This tutorial guides you through creating a TextView, adding it to your view hierarchy, and configuring the TextView.
@Image(source: tutorial-header.png, alt: "Syntax highlighted code snippets in a grid.")
}
@Section(title: "Create a TextView") {
@ContentAndMedia {
Create an instance of Tinystone's TextView and add it to the view hierarchy.
@Image(source: getting-started-1.png, alt: "A MacBook Pro running Xcode.")
}
@Steps {
@Step {
Select ViewController.swift in the Project navigator.
@Code(name: "ViewController.swift", file: getting-started-1-1.swift) {
Filter
Line: 5 Col: 96


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Build Documentation Succeeded | Today at 12.23
1
Setting Up a TextView
Tinystone 〉 Sources 〉 Tinystone 〉 Documentation 〉 Tutorials 〉 Setting Up a TextView 〉 Intro
Tinystone 1 issue
Setting Up a TextView
The tutorial 'Setting-Up-a-TextView' must be referenced from a Tutorial Table of Contents
The tutorial 'Setting-Up-a-TextView' must be referenced from a Tu...
@Tutorial(time: 30) {
@Intro(title: "Setting up a TextView") {
This tutorial guides you through creating a TextView, adding it to your view hierarchy, and configuring the TextView.
@Image(source: tutorial-header.png, alt: "Syntax highlighted code snippets in a grid.")
}
@Section(title: "Create a TextView") {
@ContentAndMedia {
Create an instance of Tinystone's TextView and add it to the view hierarchy.
@Image(source: getting-started-1.png, alt: "A MacBook Pro running Xcode.")
}
@Steps {
@Step {
Select ViewController.swift in the Project navigator.
@Code(name: "ViewController.swift", file: getting-started-1-1.swift) {
Filter
Line: 5 Col: 96


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Build Documentation Succeeded | Today at 12.23
Setting Up a TextView
Tinystone 〉 Sources 〉 Tinystone 〉 Documentation 〉 Tutorials 〉 Setting Up a TextView 〉 Intro
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
TextView
Tutorials
Code
Setting Up a TextView
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
1 @Tutorial(time: 30) {
The tutorial 'Setting-Up-a-TextView' must be referenced from a Tu...
2 @Intro(title: "Setting up a TextView") {
This tutorial guides you through creating a TextView, adding it
iguring the TextView.
alt: "Syntax highlighted
s TextView and add it to
.png, alt: "A MacBook Pro
16 @Step {
17 Select ViewController.swift in the Project navigator.
18
19 @Code(name: "ViewController.swift", file:
getting-started-1-1.swift) {
Choose a template for your new file:
iOS
macOS
watchOS
tvOS
DriverKit
tutori
Documentation
Tutorial Table of Contents File
Tutorial File
Cancel
Previous
Next
Filter
Line: 5 Col: 96


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
docs
Tinystone 〉 iPhone 14 Pro
Build Documentation Succeeded | Today at 12.23
Setting Up a TextView
Tinystone 〉 Sources 〉 Tinystone 〉 Documentation 〉 Tutorials 〉 Setting Up a TextView 〉 Intro
Tinystone
README
Package
Sources
Tinystone
Documentation
Documentation
Resources
TextView
Tutorials
Code
Setting Up a TextView
DefaultTheme
Language
SearchQuery
SearchResult
TextView
TextViewDelegate
Theme
Tests
1 @Tutorial(time: 30) {
The tutorial 'Setting-Up-a-TextView' must be referenced from a Tu...
2 @Intro(title: "Setting up a TextView") {
This tutorial guides you through creating a TextView, adding it
guring the TextView.
lt: "Syntax highlighted
TextView and add it to
png, alt: "A MacBook Pro
@Step {
17 Select ViewController.swift in the Project navigator.
18
19 @Code(name: "ViewController.swift", file:
getting-started-1-1.swift) {
Line: 5 Col: 96
Save As: Meet Tinystone tutorial
Tags:
Tutorials
Search
Favorites
Applicatio...
Desktop
Documents
Movies
Downloads
Developer
simonbs
iCloud
iCloud Dri...
Shared
Locations
Macintos...
simonbs-...
Network
Tags
Red
Orange
Code
Setting Up a...tView.tutorial
Group Tutorials
New Folder
Cancel
Create

```
@Tutorials(name: "text") {
   @Intro(title: "text") {
      text

      @Image(source: file, alt: "accessible description")
   }

   @Chapter(name: "text") {
      text

      @Image(source: file, alt: "accessible description")

      @TutorialReference(tutorial: "doc:tutorial name")
   }
}
```

Meet Tinystone.md

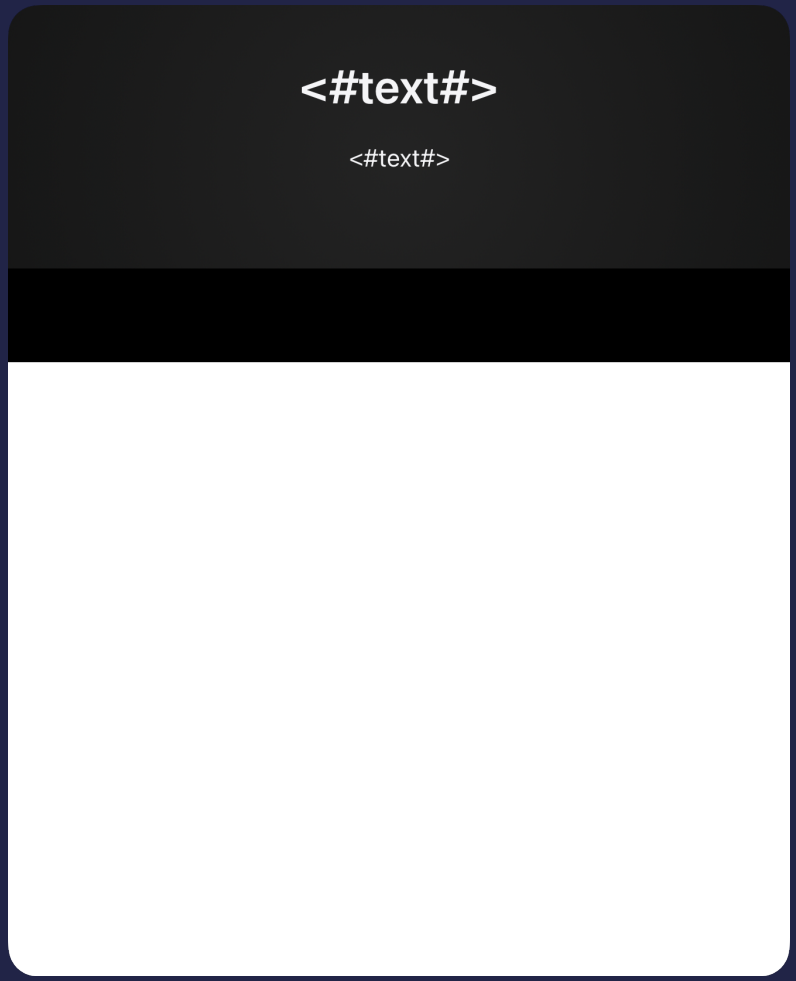
```
@Tutorials(name: "text") {
   @Intro(title: "text") {
      text

      @Image(source: file, alt: "accessible description")
   }

   @Chapter(name: "text") {
      text

      @Image(source: file, alt: "accessible description")

      @TutorialReference(tutorial: "doc:tutorial name")
   }
}
```



Meet Tinystone.md

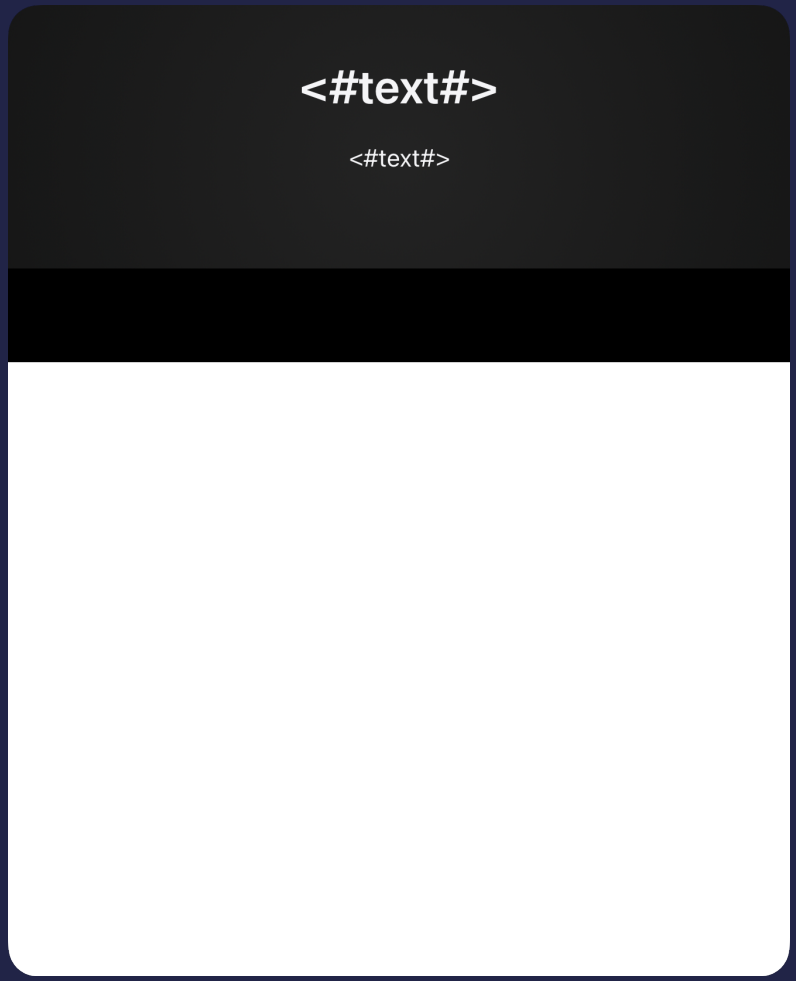
```
@Tutorials(name: "Tinystone") {
   @Intro(title: "text") {
      text

      @Image(source: file, alt: "accessible description")
   }

   @Chapter(name: "text") {
      text

      @Image(source: file, alt: "accessible description")

      @TutorialReference(tutorial: "doc:tutorial name")
   }
}
```



Meet Tinystone.md

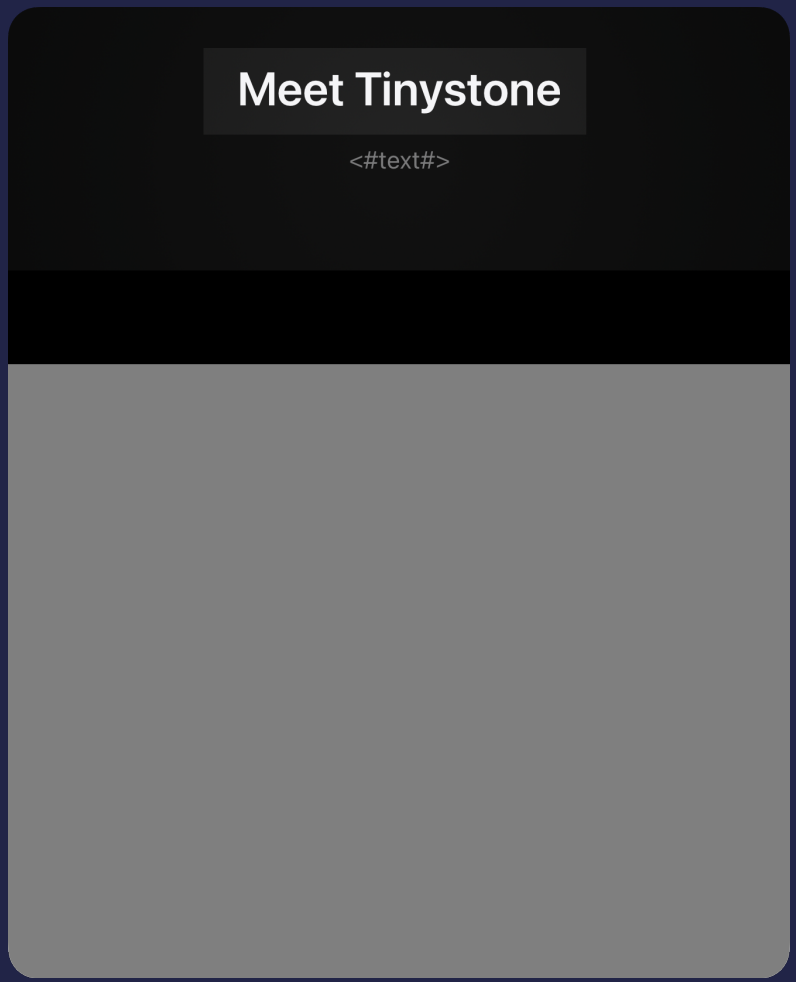
```
@Tutorials(name: "Tinystone") {
   @Intro(title: "Meet Tinystone") {
      text

      @Image(source: file, alt: "accessible description")
   }

   @Chapter(name: "text") {
      text

      @Image(source: file, alt: "accessible description")

      @TutorialReference(tutorial: "doc:tutorial name")
   }
}
```



Meet Tinystone.md

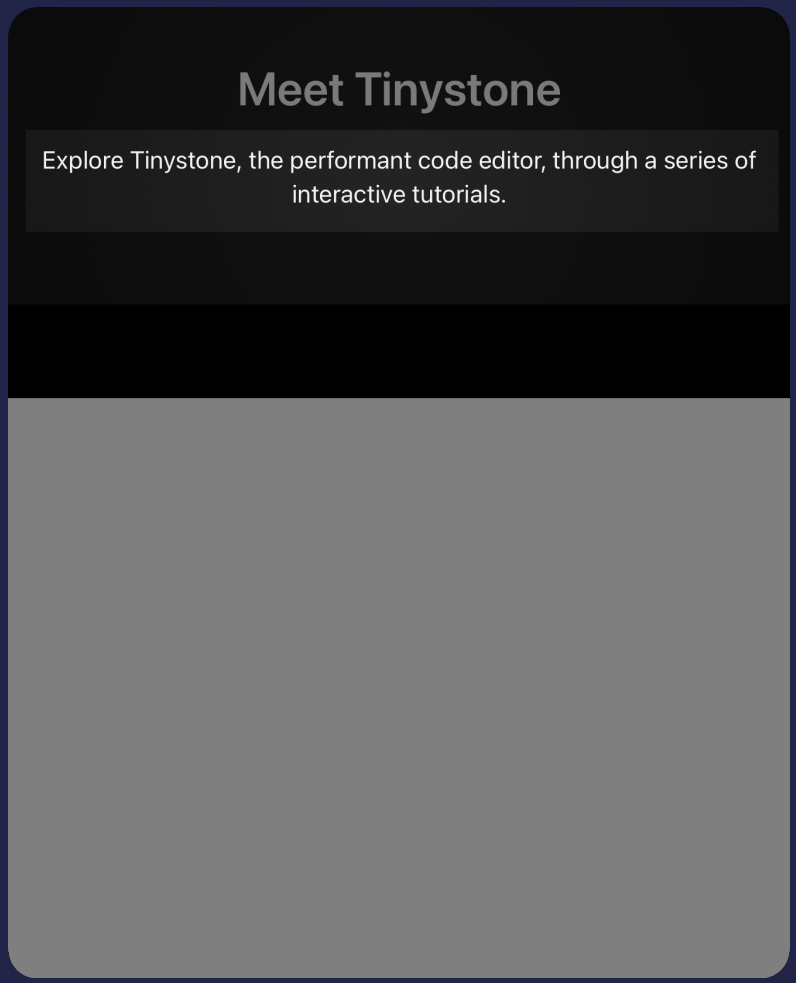
```
@Tutorials(name: "Tinystone") {
   @Intro(title: "Meet Tinystone") {
      Explore Tinystone, the performant code editor,
      through a series of interactive tutorials.

      @Image(source: file, alt: "accessible description")
   }

   @Chapter(name: "text") {
      text

      @Image(source: file, alt: "accessible description")

      @TutorialReference(tutorial: "doc:tutorial name")
   }
}
```



Meet Tinystone.md

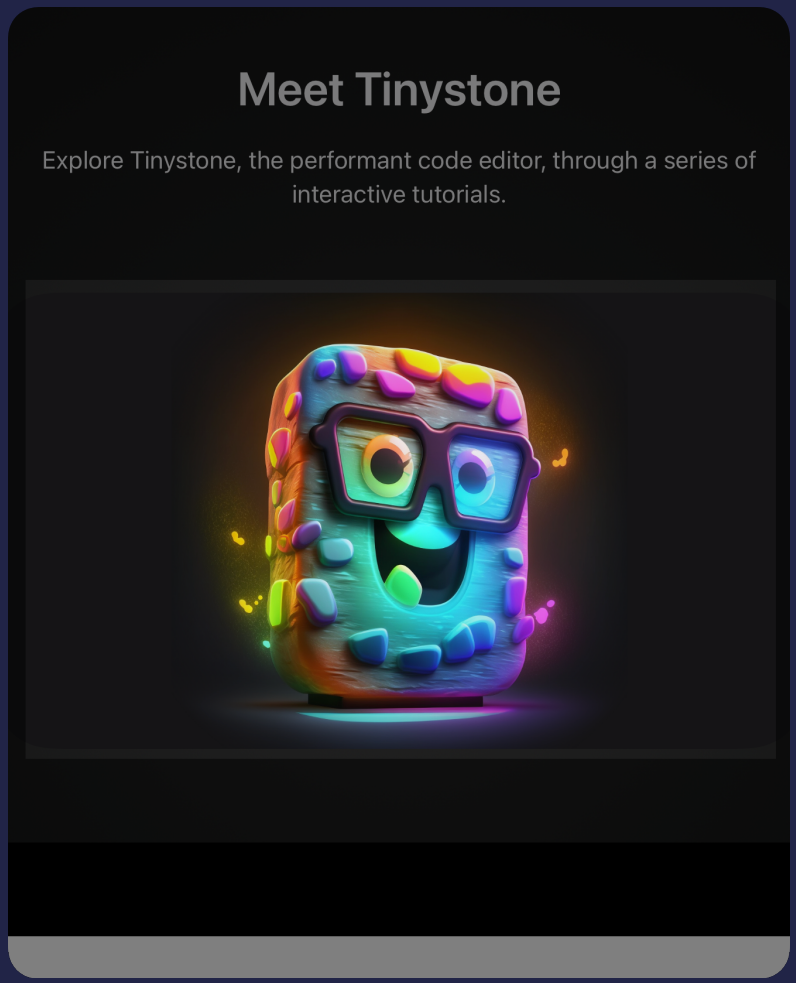
```
@Tutorials(name: "Tinystone") {
   @Intro(title: "Meet Tinystone") {
      Explore Tinystone, the performant code editor,
      through a series of interactive tutorials.

      @Image(source: hero.png, alt: "Happy stone in many colors")
   }

   @Chapter(name: "text") {
      text

      @Image(source: file, alt: "accessible description")

      @TutorialReference(tutorial: "doc:tutorial name")
   }
}
```



Meet Tinystone.md

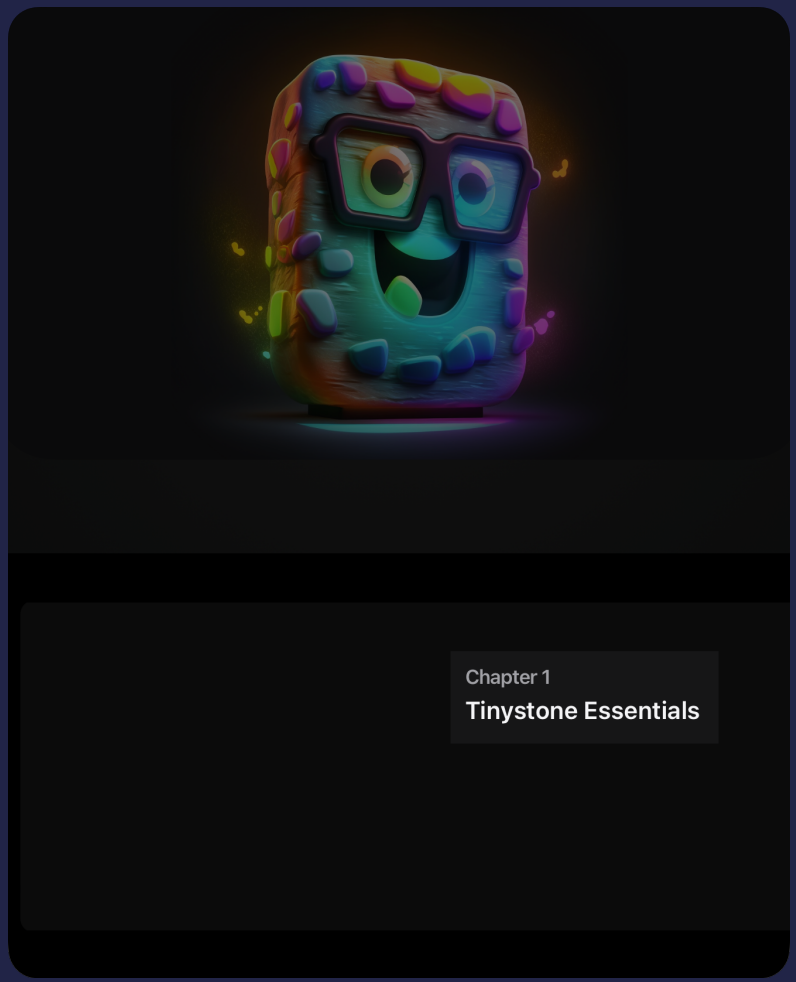
```
@Tutorials(name: "Tinystone") {
   @Intro(title: "Meet Tinystone") {
      Explore Tinystone, the performant code editor,
      through a series of interactive tutorials.

      @Image(source: hero.png, alt: "Happy stone in many colors")
   }

   @Chapter(name: "Tinystone Essentials") {
      text

      @Image(source: file, alt: "accessible description")

      @TutorialReference(tutorial: "doc:tutorial name")
   }
}
```



Meet Tinystone.md

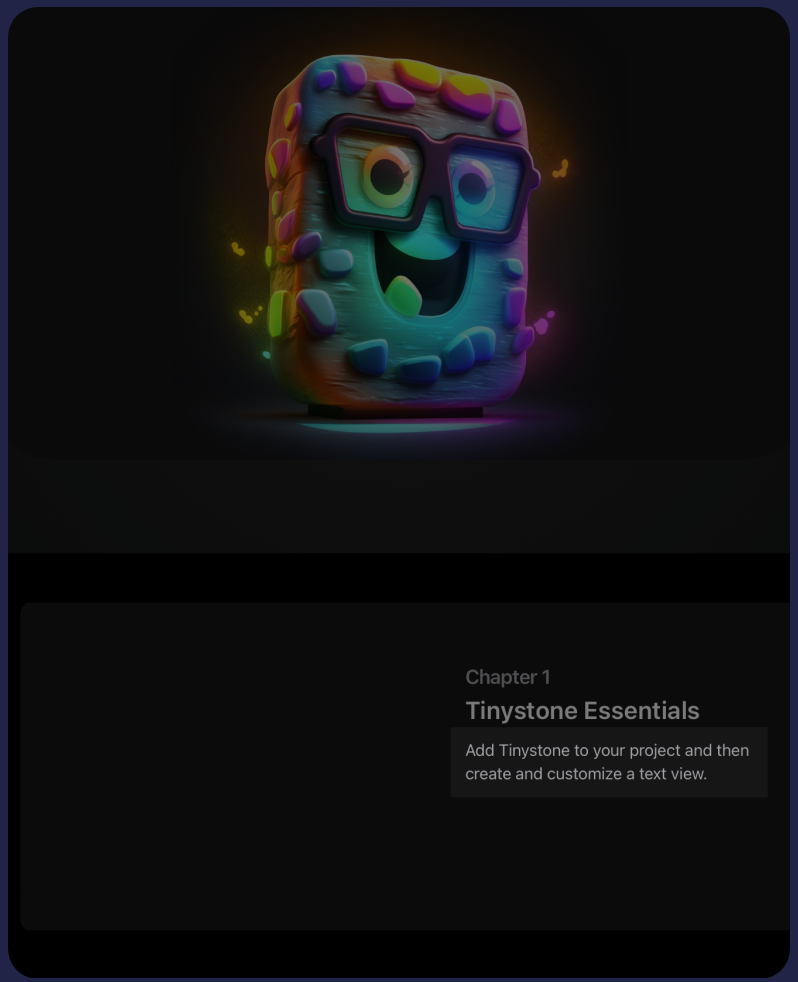
```
@Tutorials(name: "Tinystone") {
   @Intro(title: "Meet Tinystone") {
      Explore Tinystone, the performant code editor,
      through a series of interactive tutorials.

      @Image(source: hero.png, alt: "Happy stone in many colors")
   }

   @Chapter(name: "Tinystone Essentials") {
      Add Tinystone to your project and then create and
      customize a text view.

      @Image(source: file, alt: "accessible description")

      @TutorialReference(tutorial: "doc:tutorial name")
   }
}
```



Meet Tinystone.md

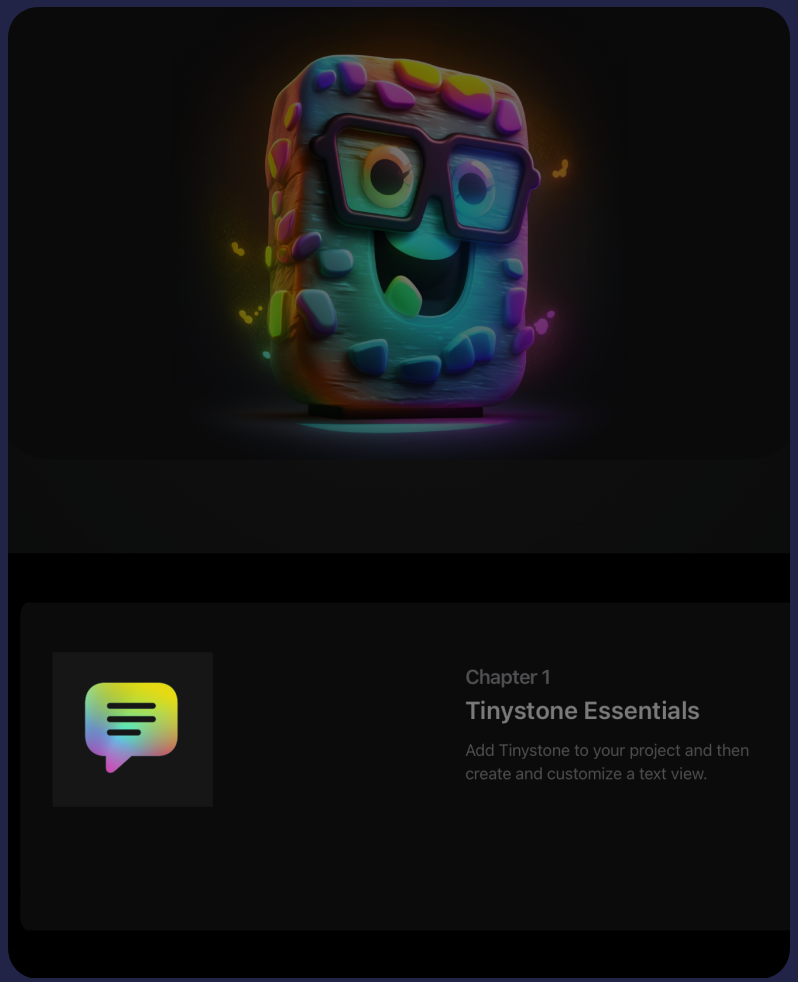
```
@Tutorials(name: "Tinystone") {
   @Intro(title: "Meet Tinystone") {
      Explore Tinystone, the performant code editor,
      through a series of interactive tutorials.

      @Image(source: hero.png, alt: "Happy stone in many colors")
   }

   @Chapter(name: "Tinystone Essentials") {
      Add Tinystone to your project and then create and
      customize a text view.

      @Image(source: chapter.png, alt: "Text forming an R")

      @TutorialReference(tutorial: "doc:tutorial name")
   }
}
```



Meet Tinystone.md

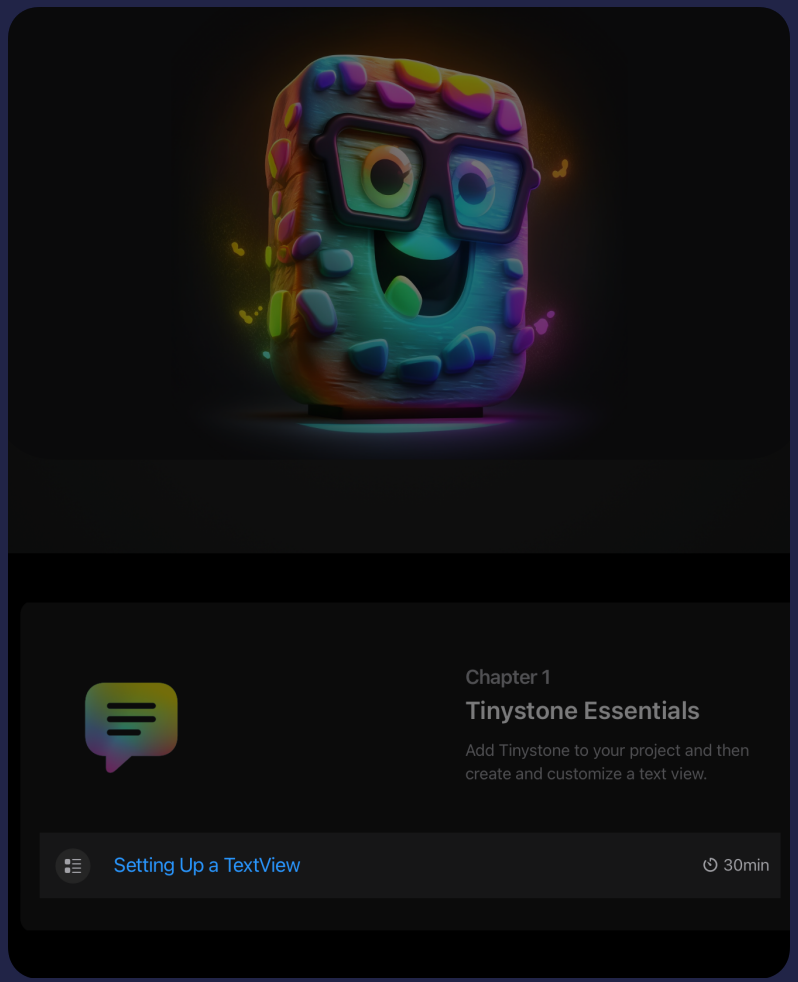
```
@Tutorials(name: "Tinystone") {
   @Intro(title: "Meet Tinystone") {
      Explore Tinystone, the performant code editor,
      through a series of interactive tutorials.

      @Image(source: hero.png, alt: "Happy stone in many colors")
   }

   @Chapter(name: "Tinystone Essentials") {
      Add Tinystone to your project and then create and
      customize a text view.

      @Image(source: chapter.png, alt: "Text forming an R")

      @TutorialReference(tutorial: "doc:Setting-Up-a-TextView")
   }
}
```




TED
LASSO
Woohoo!
FOUR WINDS

Describe Symbols

Add Structure

Add Supplemental Content

Describe Symbols

✅

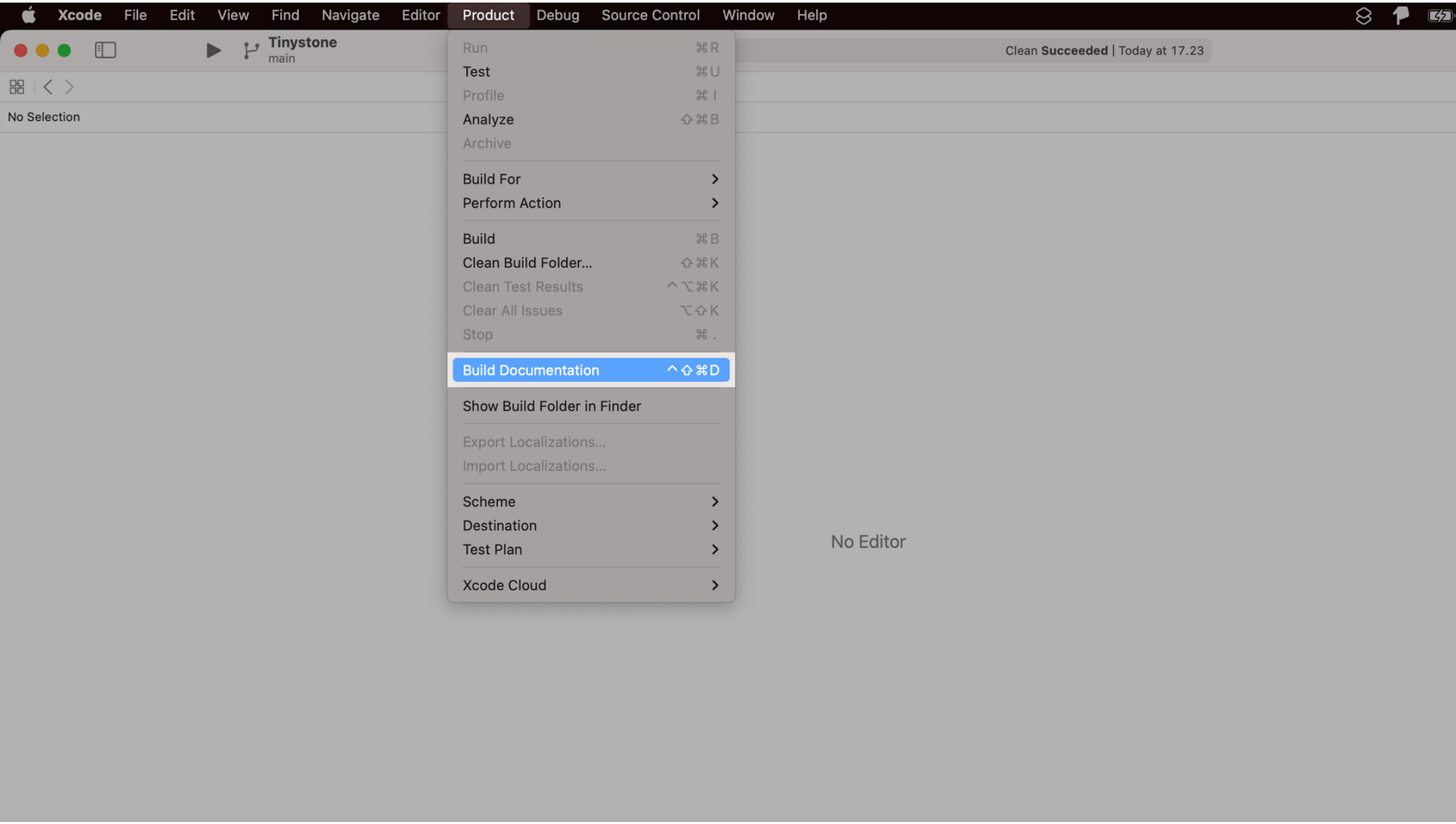
Add Structure

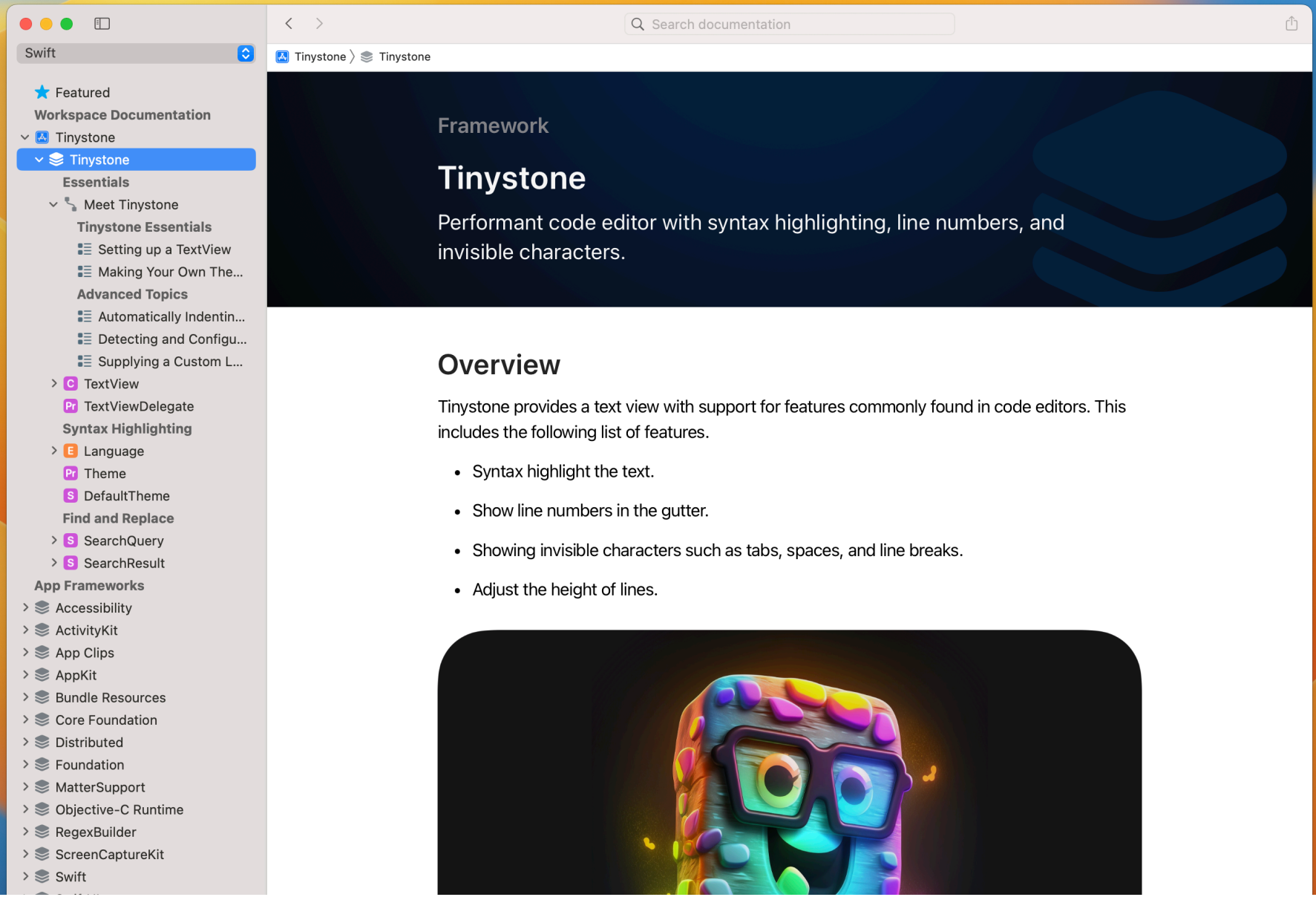
✅

Add Supplemental Content

✨

How can people read my documentation?


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tinystone
main
Clean Succeeded | Today at 17.23
No Selection
Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For
Perform Action
Build ⌘B
Clean Build Folder... ⇧⌘K
Clean Test Results ^⌥⌘K
Clear All Issues ⌥⇧K
Stop ⌘.
Build Documentation ^⇧⌘D
Show Build Folder in Finder
Export Localizations...
Import Localizations...
Scheme
Destination
Test Plan
Xcode Cloud
No Editor


Swift
Featured
Workspace Documentation
Tinystone
Tinystone
Essentials
Meet Tinystone
Tinystone Essentials
Setting up a TextView
Making Your Own The...
Advanced Topics
Automatically Indentin...
Detecting and Configu...
Supplying a Custom L...
TextView
TextViewDelegate
Syntax Highlighting
Language
Theme
DefaultTheme
Find and Replace
SearchQuery
SearchResult
App Frameworks
Accessibility
ActivityKit
App Clips
AppKit
Bundle Resources
Core Foundation
Distributed
Foundation
MatterSupport
Objective-C Runtime
RegexBuilder
ScreenCaptureKit
Swift
Search documentation
Tinystone
Tinystone
Framework
Tinystone
Performant code editor with syntax highlighting, line numbers, and invisible characters.
Overview
Tinystone provides a text view with support for features commonly found in code editors. This includes the following list of features.
Syntax highlight the text.
Show line numbers in the gutter.
Showing invisible characters such as tabs, spaces, and line breaks.
Adjust the height of lines.

GitHub Actions

+

GitHub Pages

=

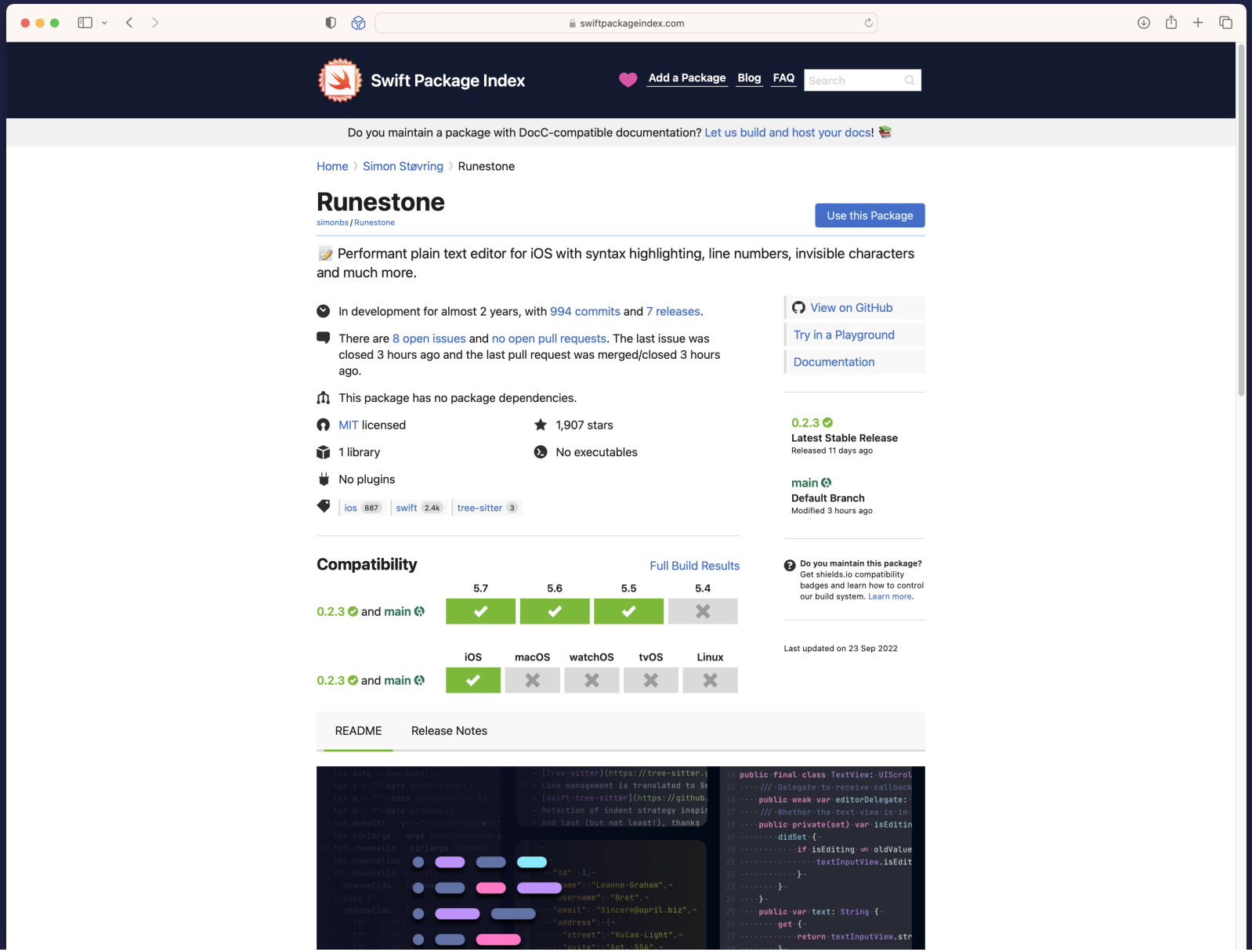
docs.runestone.app

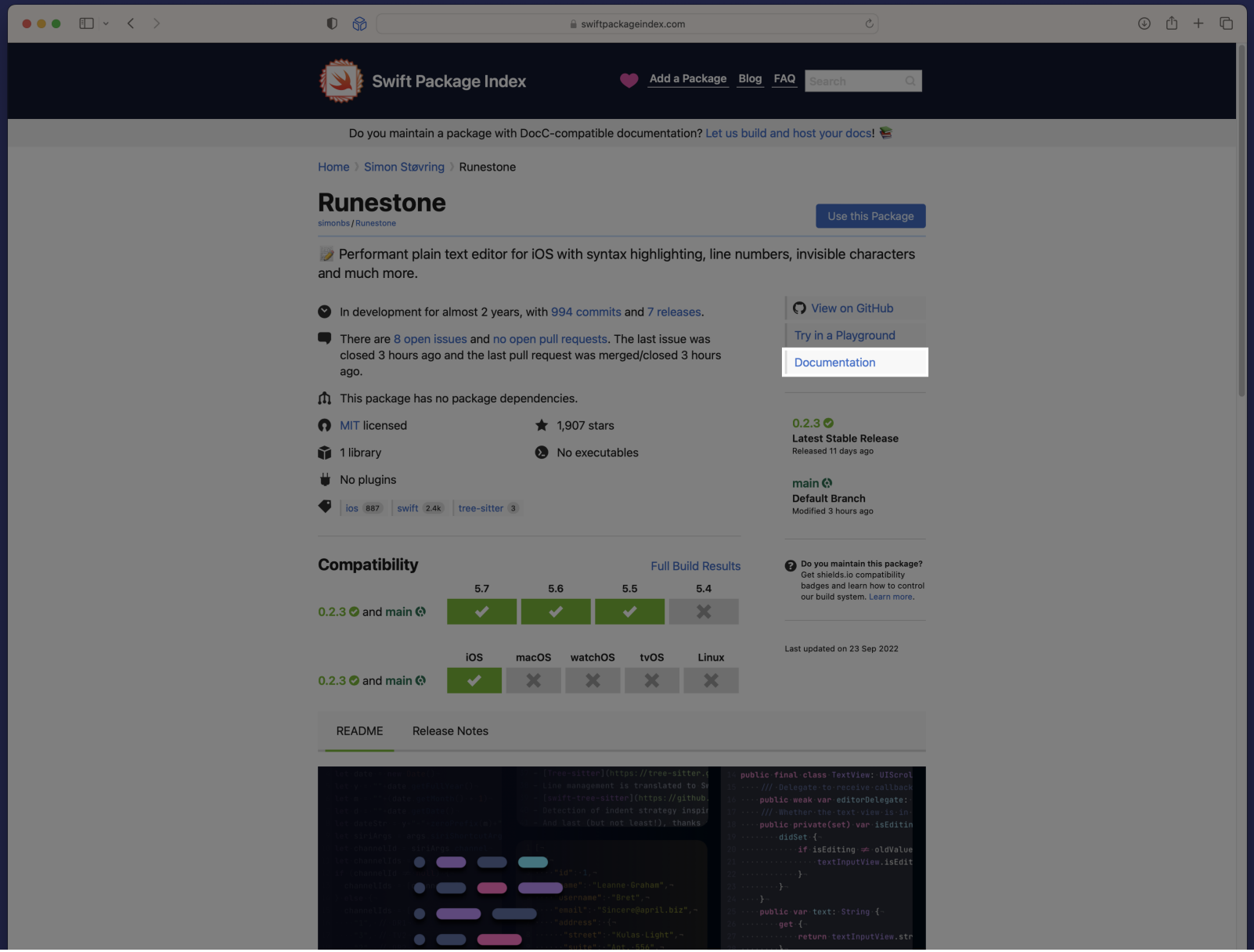
```
xcodebuild docbuild \
    -scheme Runestone \
    -destination 'generic/platform=iOS' \
    -derivedDataPath DerivedData
```

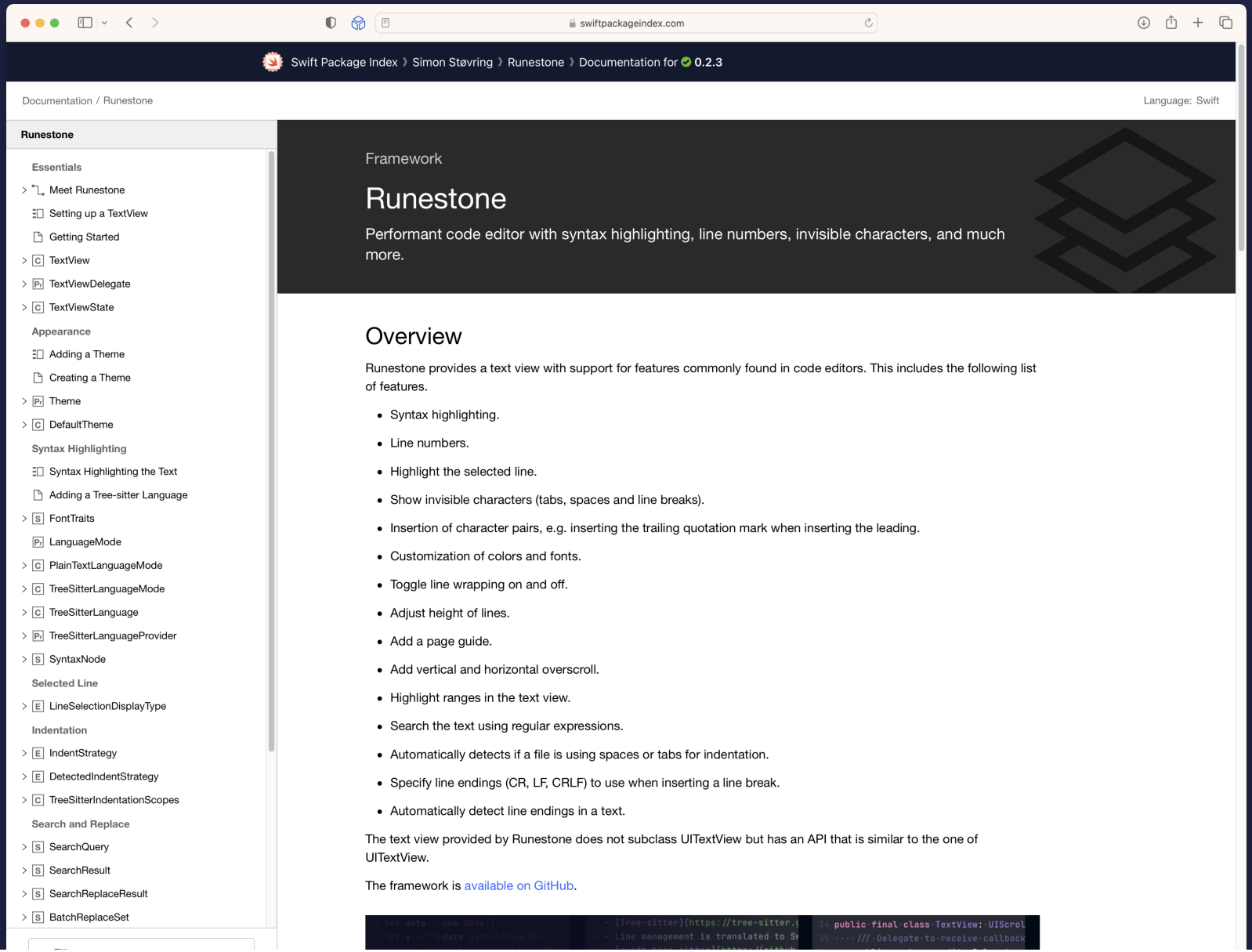
```
xcodebuild docbuild \
    -scheme Runestone \
    -destination 'generic/platform=iOS' \
    -derivedDataPath DerivedData

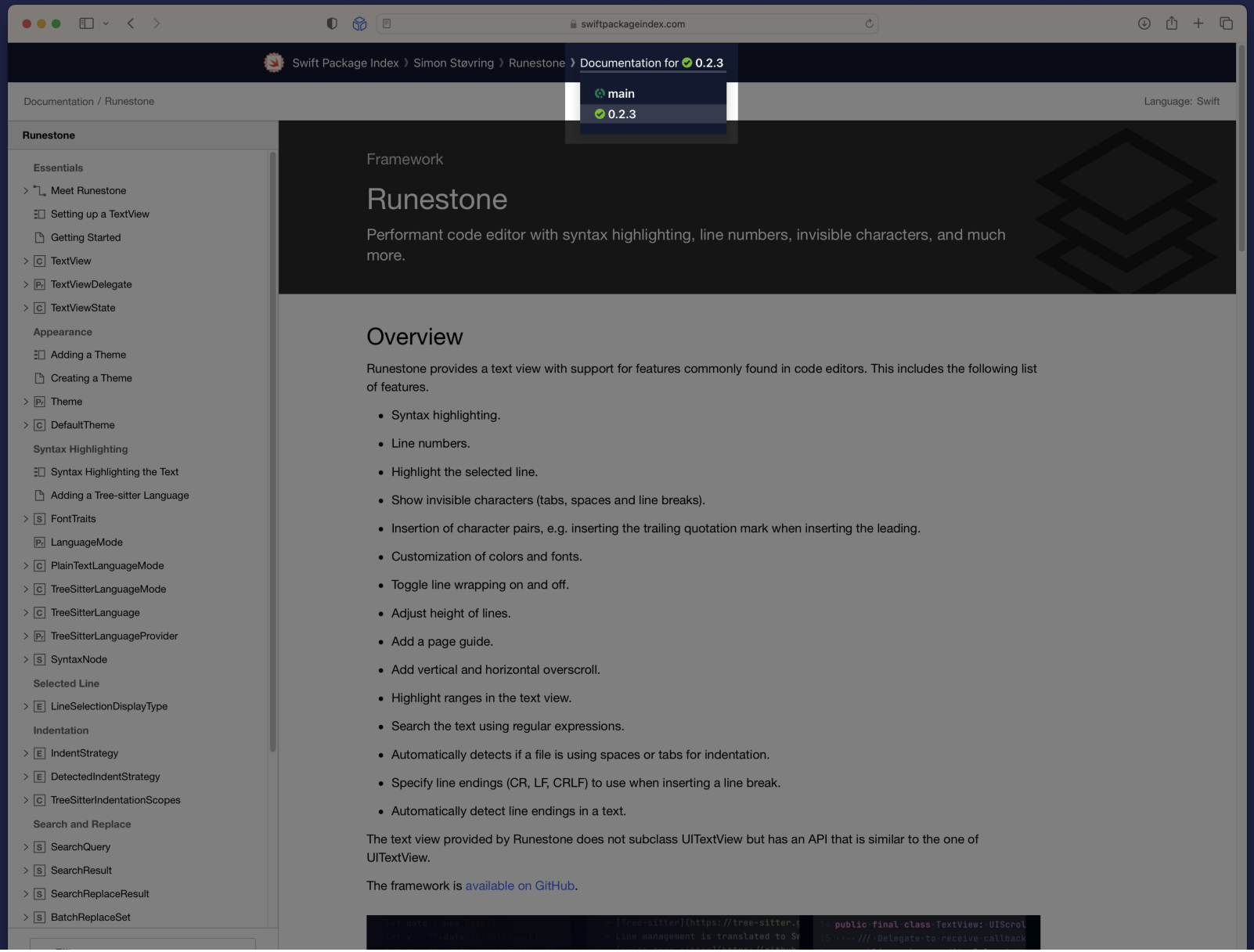

$(xcrun --find docc) process-archive transform-for-static-hosting \
    DerivedData/Build/Products/Debug-iphoneos/Runestone.doccarchive \
    --output-path _site \
    --hosting-base-path /
```

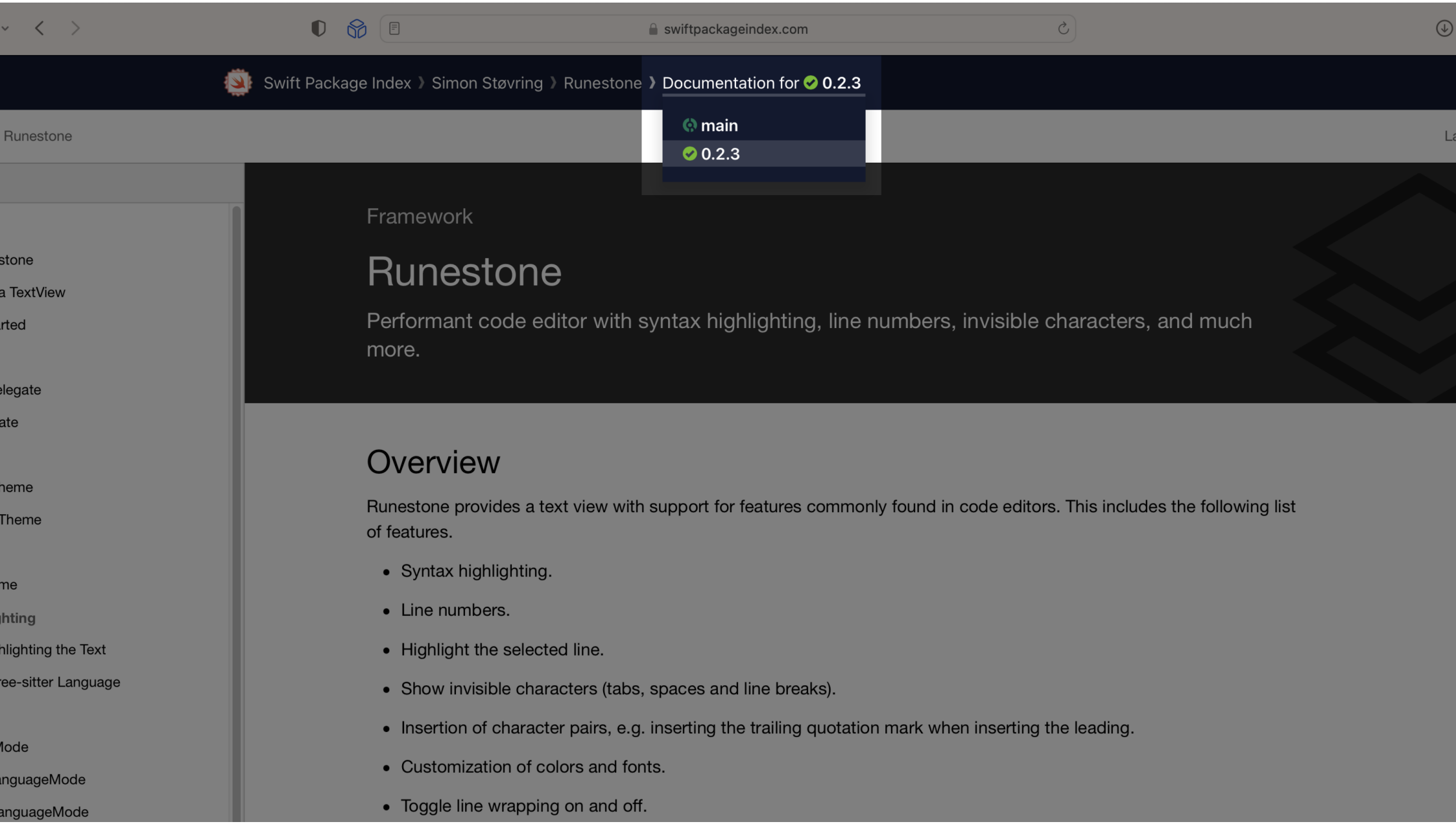
Swift Package Index.com

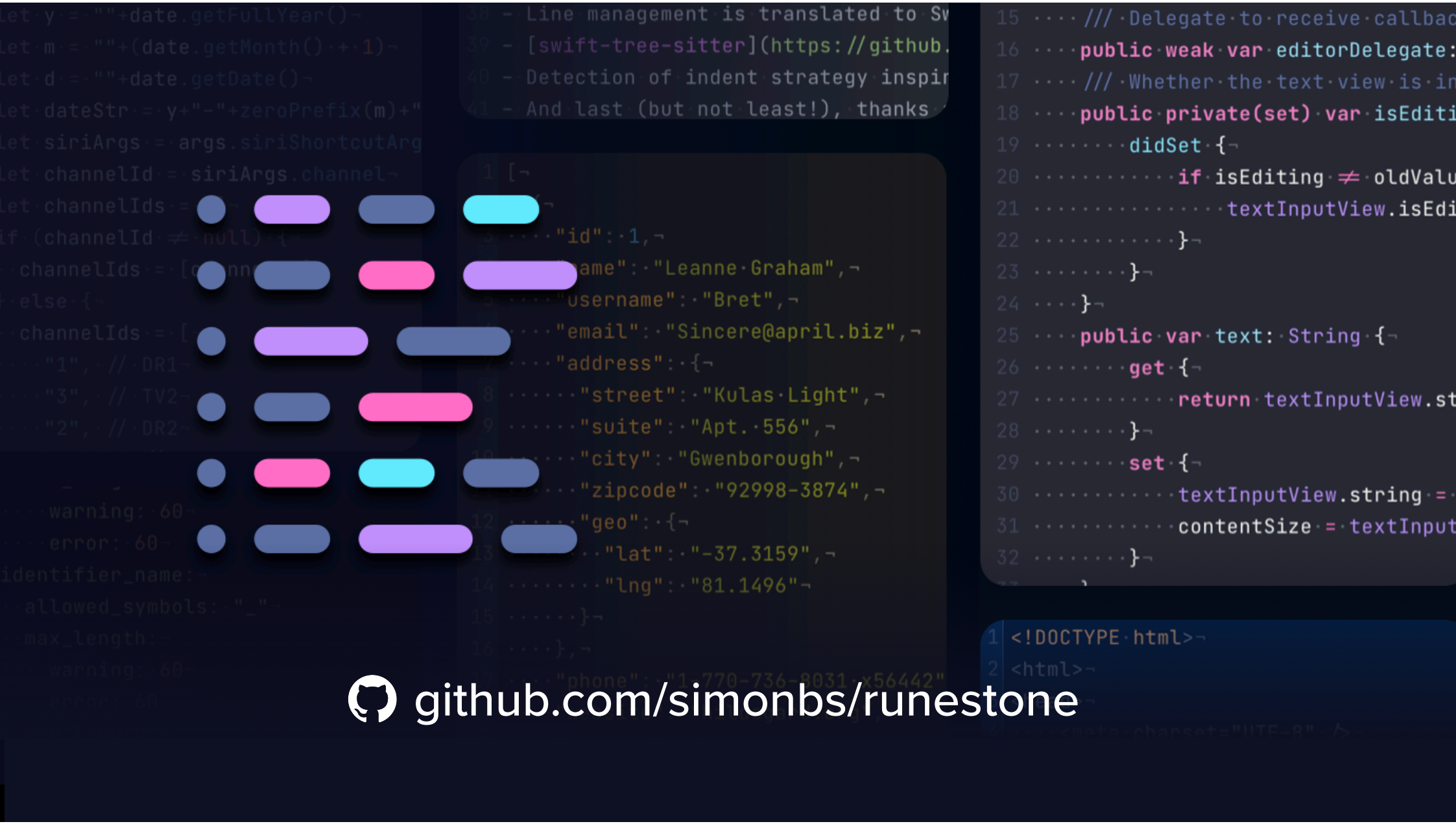

swiftpackageindex.com
Swift Package Index
Add a Package
Blog
FAQ
Search
Do you maintain a package with DocC-compatible documentation? Let us build and host your docs!
Home › Simon Støvring › Runestone
Runestone
simonbs / Runestone
Use this Package
Performant plain text editor for iOS with syntax highlighting, line numbers, invisible characters and much more.
In development for almost 2 years, with 994 commits and 7 releases.
There are 8 open issues and no open pull requests. The last issue was closed 3 hours ago and the last pull request was merged/closed 3 hours ago.
This package has no package dependencies.
MIT licensed
1,907 stars
1 library
No executables
No plugins
ios 887
swift 2.4k
tree-sitter 3
View on GitHub
Try in a Playground
Documentation
0.2.3
Latest Stable Release
Released 11 days ago
main
Default Branch
Modified 3 hours ago
Compatibility
Full Build Results
5.7
5.6
5.5
5.4
0.2.3 and main
iOS
macOS
watchOS
tvOS
Linux
0.2.3 and main
Do you maintain this package?
Get shields.io compatibility badges and learn how to control our build system. Learn more.
Last updated on 23 Sep 2022
README
Release Notes


swiftpackageindex.com
Swift Package Index
Add a Package
Blog
FAQ
Search
Do you maintain a package with DocC-compatible documentation? Let us build and host your docs!
Home
Simon Støvring
Runestone
Runestone
simonbs / Runestone
Use this Package
Performant plain text editor for iOS with syntax highlighting, line numbers, invisible characters and much more.
In development for almost 2 years, with 994 commits and 7 releases.
There are 8 open issues and no open pull requests. The last issue was closed 3 hours ago and the last pull request was merged/closed 3 hours ago.
This package has no package dependencies.
MIT licensed
1,907 stars
1 library
No executables
No plugins
ios 887
swift 2.4k
tree-sitter 3
View on GitHub
Try in a Playground
Documentation
0.2.3
Latest Stable Release
Released 11 days ago
main
Default Branch
Modified 3 hours ago
Compatibility
Full Build Results
5.7
5.6
5.5
5.4
0.2.3 and main
iOS
macOS
watchOS
tvOS
Linux
0.2.3 and main
Do you maintain this package?
Get shields.io compatibility badges and learn how to control our build system. Learn more.
Last updated on 23 Sep 2022
README
Release Notes


swiftpackageindex.com
Swift Package Index › Simon Støvring › Runestone › Documentation for 0.2.3
Documentation / Runestone
Language: Swift
Runestone
Essentials
Meet Runestone
Setting up a TextView
Getting Started
TextView
TextViewDelegate
TextViewState
Appearance
Adding a Theme
Creating a Theme
Theme
DefaultTheme
Syntax Highlighting
Syntax Highlighting the Text
Adding a Tree-sitter Language
FontTraits
LanguageMode
PlainTextLanguageMode
TreeSitterLanguageMode
TreeSitterLanguage
TreeSitterLanguageProvider
SyntaxNode
Selected Line
LineSelectionDisplayType
Indentation
IndentStrategy
DetectedIndentStrategy
TreeSitterIndentationScopes
Search and Replace
SearchQuery
SearchResult
SearchReplaceResult
BatchReplaceSet
Framework
Runestone
Performant code editor with syntax highlighting, line numbers, invisible characters, and much more.
Overview
Runestone provides a text view with support for features commonly found in code editors. This includes the following list of features.
Syntax highlighting.
Line numbers.
Highlight the selected line.
Show invisible characters (tabs, spaces and line breaks).
Insertion of character pairs, e.g. inserting the trailing quotation mark when inserting the leading.
Customization of colors and fonts.
Toggle line wrapping on and off.
Adjust height of lines.
Add a page guide.
Add vertical and horizontal overscroll.
Highlight ranges in the text view.
Search the text using regular expressions.
Automatically detects if a file is using spaces or tabs for indentation.
Specify line endings (CR, LF, CRLF) to use when inserting a line break.
Automatically detect line endings in a text.
The text view provided by Runestone does not subclass UITextView but has an API that is similar to the one of UITextView.
The framework is available on GitHub.


swiftpackageindex.com
Swift Package Index
Simon Støvring
Runestone
Documentation for 0.2.3
main
0.2.3
Documentation / Runestone
Language: Swift
Runestone
Essentials
Meet Runestone
Setting up a TextView
Getting Started
TextView
TextViewDelegate
TextViewState
Appearance
Adding a Theme
Creating a Theme
Theme
DefaultTheme
Syntax Highlighting
Syntax Highlighting the Text
Adding a Tree-sitter Language
FontTraits
LanguageMode
PlainTextLanguageMode
TreeSitterLanguageMode
TreeSitterLanguage
TreeSitterLanguageProvider
SyntaxNode
Selected Line
LineSelectionDisplayType
Indentation
IndentStrategy
DetectedIndentStrategy
TreeSitterIndentationScopes
Search and Replace
SearchQuery
SearchResult
SearchReplaceResult
BatchReplaceSet
Framework
Runestone
Performant code editor with syntax highlighting, line numbers, invisible characters, and much more.
Overview
Runestone provides a text view with support for features commonly found in code editors. This includes the following list of features.
Syntax highlighting.
Line numbers.
Highlight the selected line.
Show invisible characters (tabs, spaces and line breaks).
Insertion of character pairs, e.g. inserting the trailing quotation mark when inserting the leading.
Customization of colors and fonts.
Toggle line wrapping on and off.
Adjust height of lines.
Add a page guide.
Add vertical and horizontal overscroll.
Highlight ranges in the text view.
Search the text using regular expressions.
Automatically detects if a file is using spaces or tabs for indentation.
Specify line endings (CR, LF, CRLF) to use when inserting a line break.
Automatically detect line endings in a text.
The text view provided by Runestone does not subclass UITextView but has an API that is similar to the one of UITextView.
The framework is available on GitHub.


swiftpackageindex.com
Swift Package Index › Simon Støvring › Runestone › Documentation for 0.2.3
main
0.2.3
Runestone
Framework
Runestone
Performant code editor with syntax highlighting, line numbers, invisible characters, and much more.
Overview
Runestone provides a text view with support for features commonly found in code editors. This includes the following list of features.
Syntax highlighting.
Line numbers.
Highlight the selected line.
Show invisible characters (tabs, spaces and line breaks).
Insertion of character pairs, e.g. inserting the trailing quotation mark when inserting the leading.
Customization of colors and fonts.
Toggle line wrapping on and off.


github.com/simonbs/runestone

It has never been easier to make great documentation

 SIMONBS.DEV