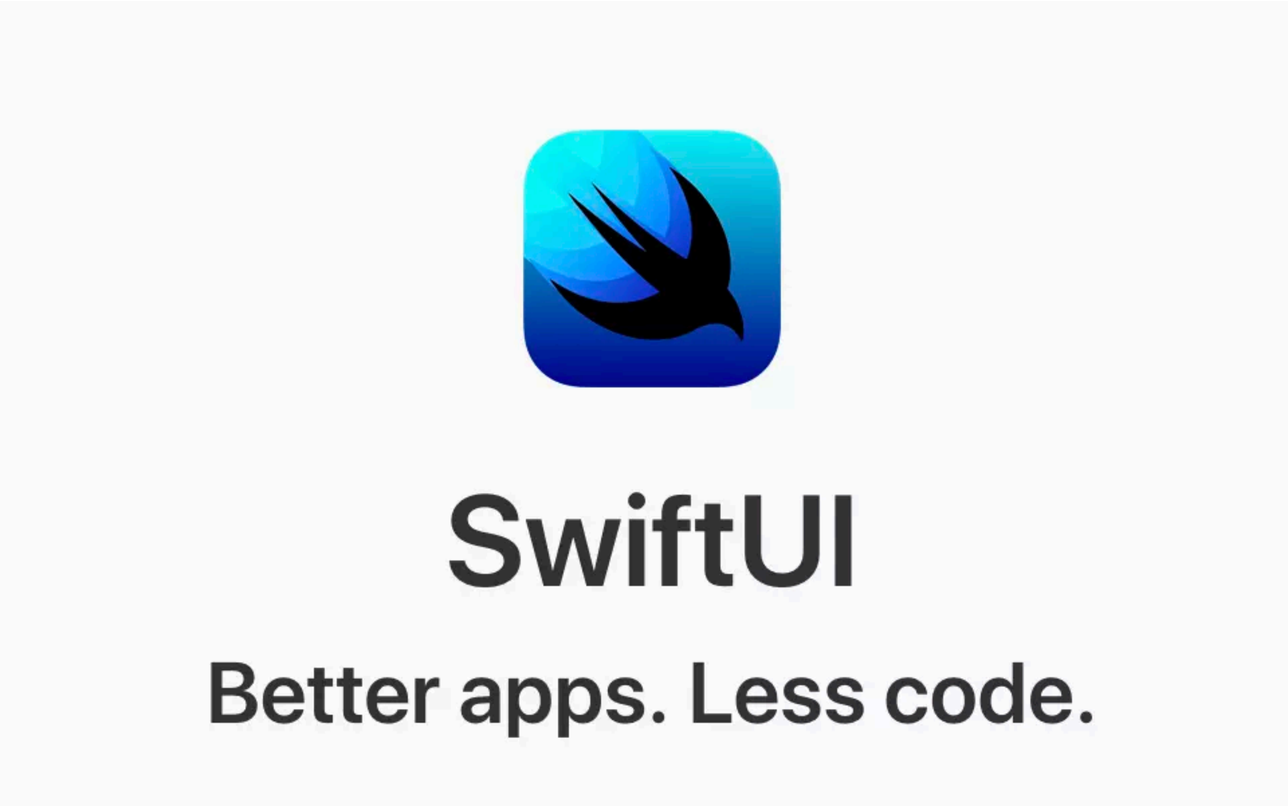
SwiftUI

极客时间

扫码试看/订阅

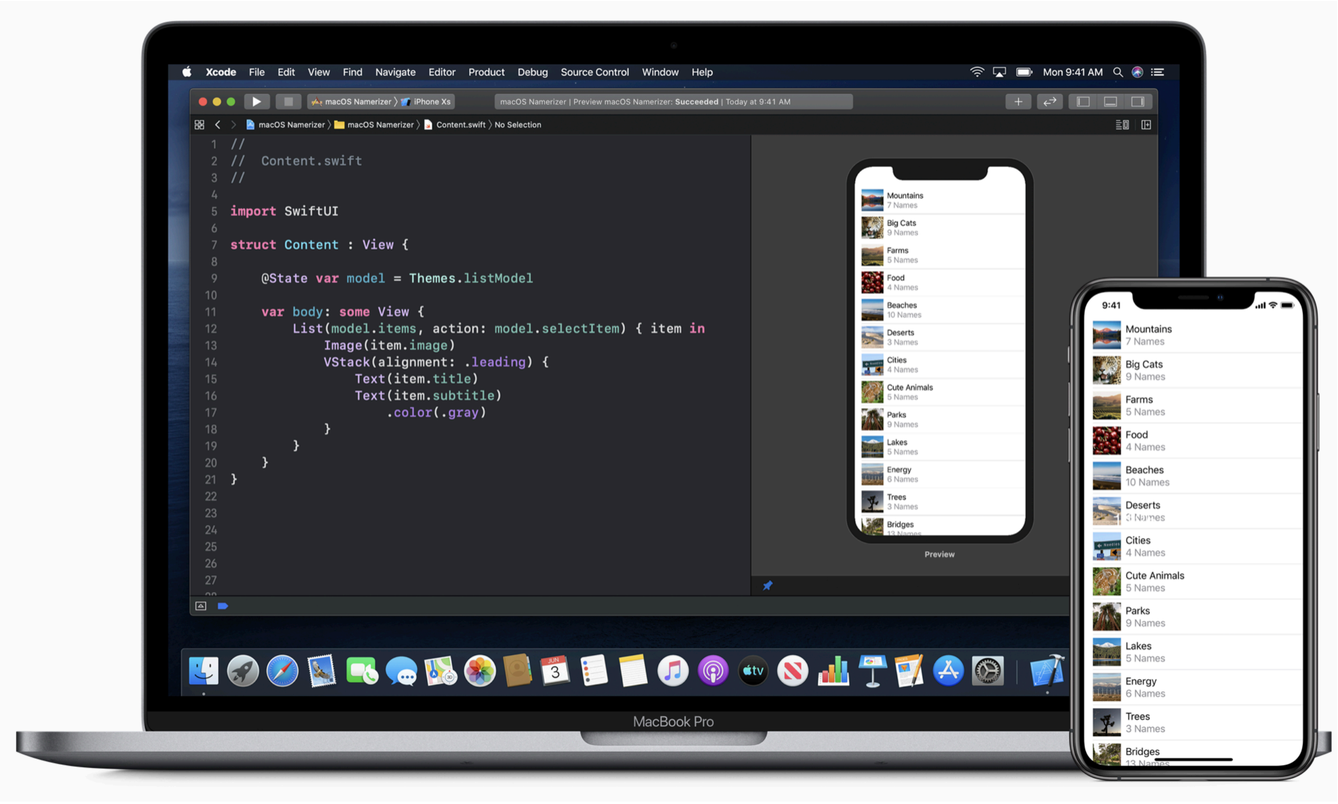
《Swift核心技术与实战》视频课程

SwiftUI



SwiftUI

- SwiftUI 是一种基于 Swift 的强大能力，简单创新的构建用户界面的方法，并且可以运行在苹果所有的平台上。



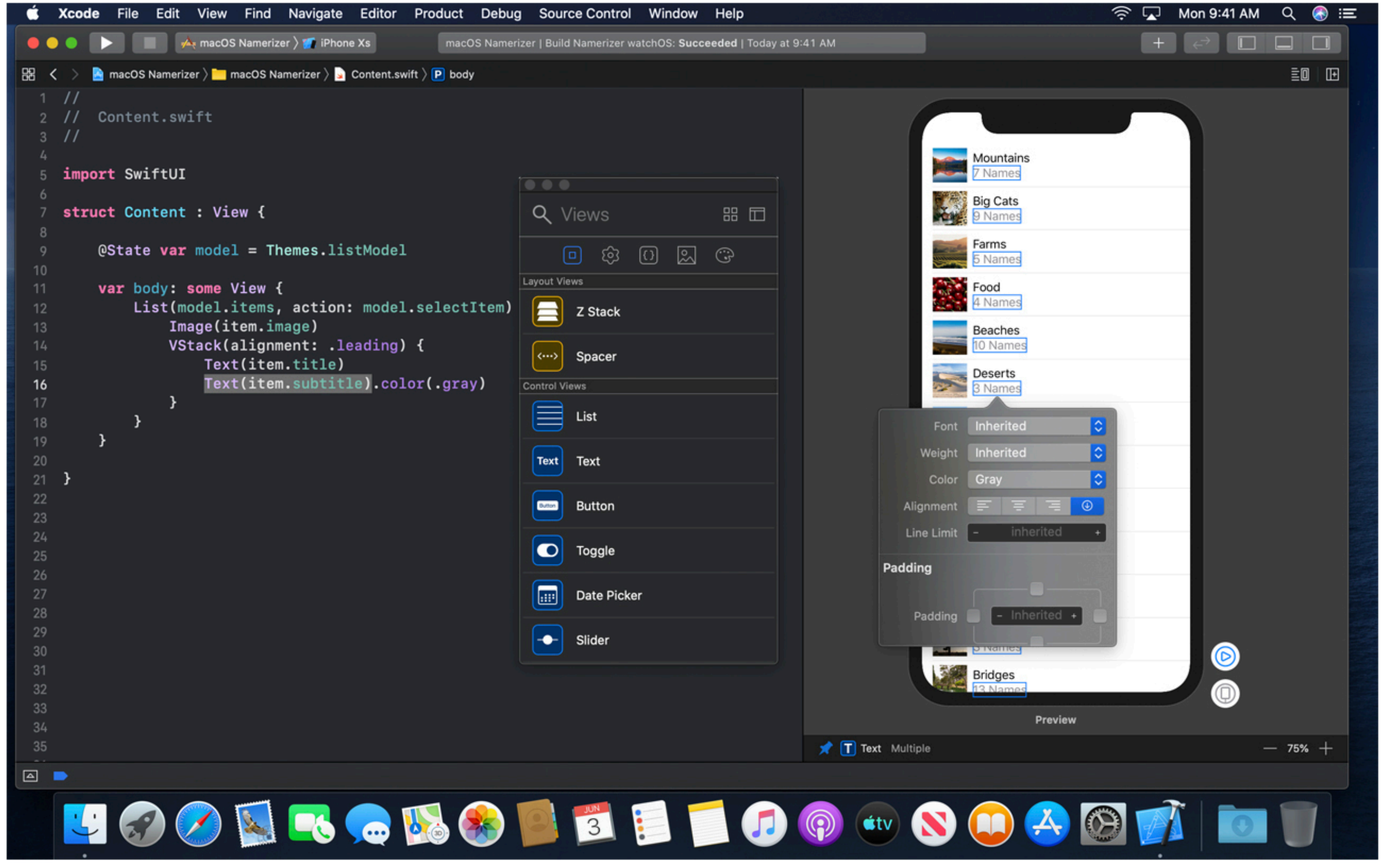
SwiftUI - 声明式语法

- SwiftUI 采用声明式语法，因此你可以简单声明你的用户界面。

```
import SwiftUI

struct Content : View {

    @State var model = Themes.listModel

    var body: some View {
        List(model.items, action: model.selectItem) { item in
            Image(item.image)
            VStack(alignment: .leading) {
                Text(item.title)
                Text(item.subtitle)
                    .color(.gray)
            }
        }
    }
}
```

SwiftUI - 设计工具

- Xcode11 提供了强大的设计工具，可以通过简单的拖拽用 SwiftUI 生成用户界面。

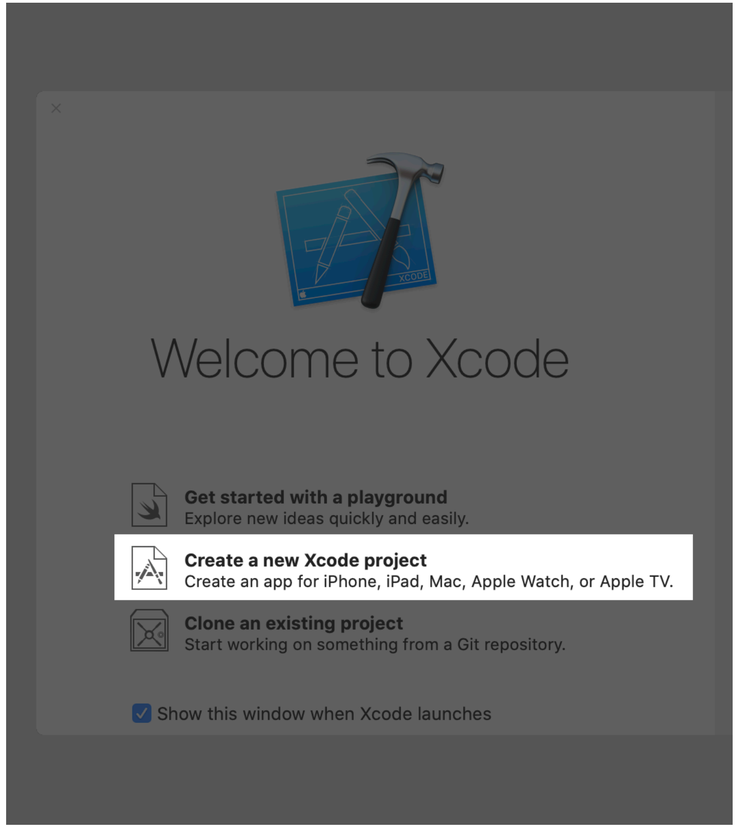


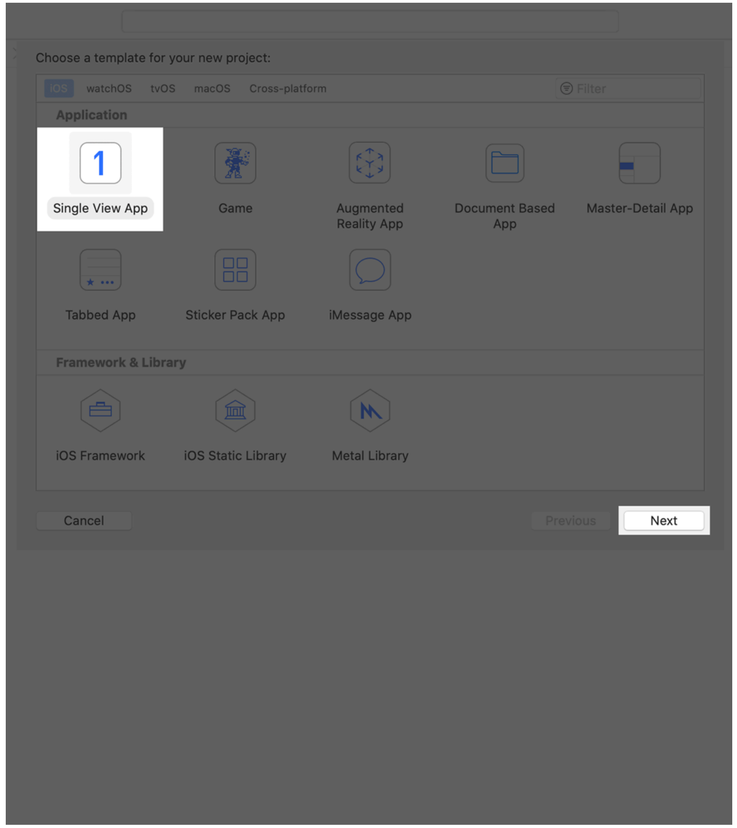
SwiftUI

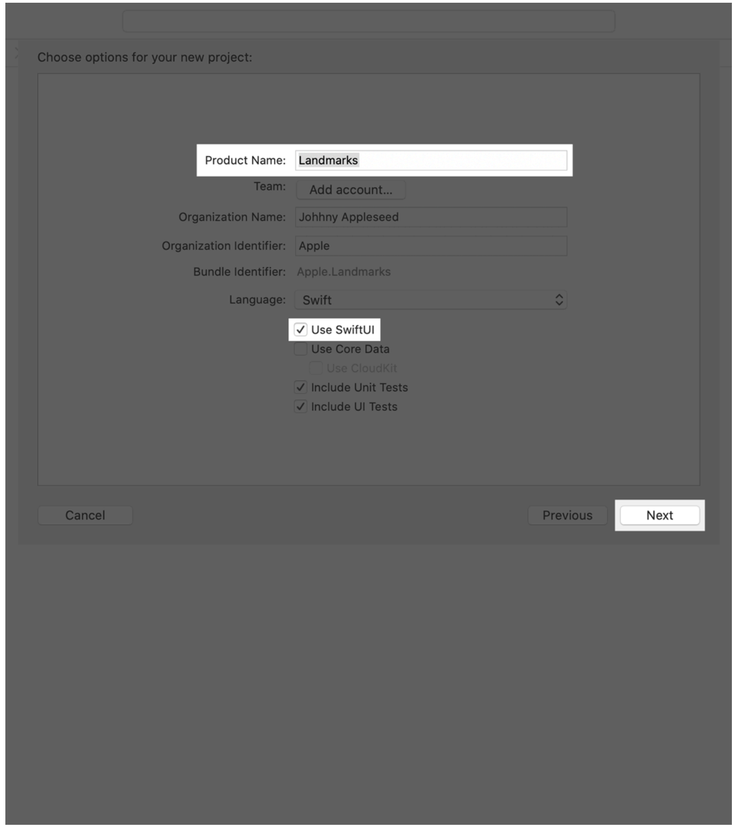
- 只需要描述一次的布局-为你的视图声明任何状态的内容和布局，一旦状态发生改变， SwiftUI 会自动更新视图的渲染。
- 构建可复用的组件-将小型、独立视图组合到更大，更复杂的界面中。在任何为 Apple 平台所设计的应用之间，共享您的自定义视图。
- 精简动画-创建平滑的动画就像调用单个方法一样简单。 SwiftUI 会在必要时自动计算并过渡动画。

SwiftUI 使用指南

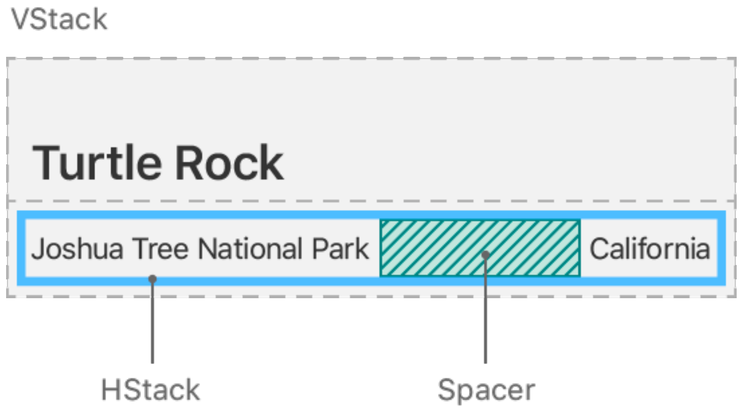
创建项目


Welcome to Xcode
Get started with a playground
Explore new ideas quickly and easily.
Create a new Xcode project
Create an app for iPhone, iPad, Mac, Apple Watch, or Apple TV.
Clone an existing project
Start working on something from a Git repository.
Show this window when Xcode launches


Choose a template for your new project:
iOS
watchOS
tvOS
macOS
Cross-platform
Filter
Application
Single View App
Game
Augmented Reality App
Document Based App
Master-Detail App
Tabbed App
Sticker Pack App
iMessage App
Framework & Library
iOS Framework
iOS Static Library
Metal Library
Cancel
Previous
Next


Choose options for your new project:
Product Name: Landmarks
Team: Add account...
Organization Name: Johhny Appleseed
Organization Identifier: Apple
Bundle Identifier: Apple.Landmarks
Language: Swift
Use SwiftUI
Use Core Data
Use CloudKit
Include Unit Tests
Include UI Tests
Cancel
Previous
Next

Stacks

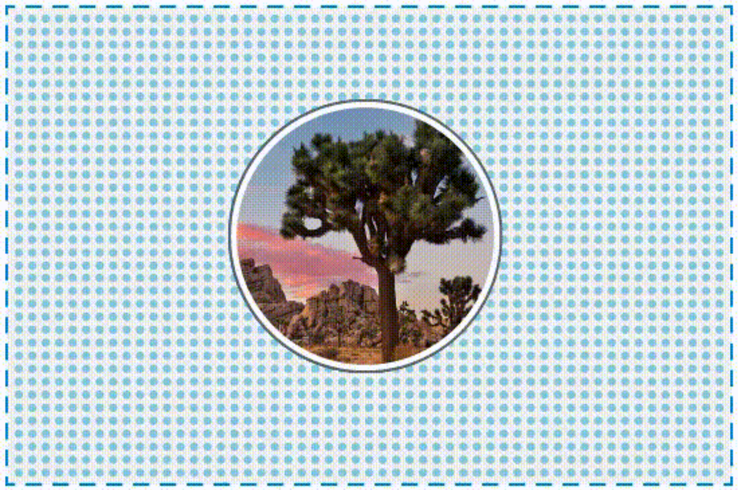

VStack
Turtle Rock
Joshua Tree National Park
California
HStack
Spacer

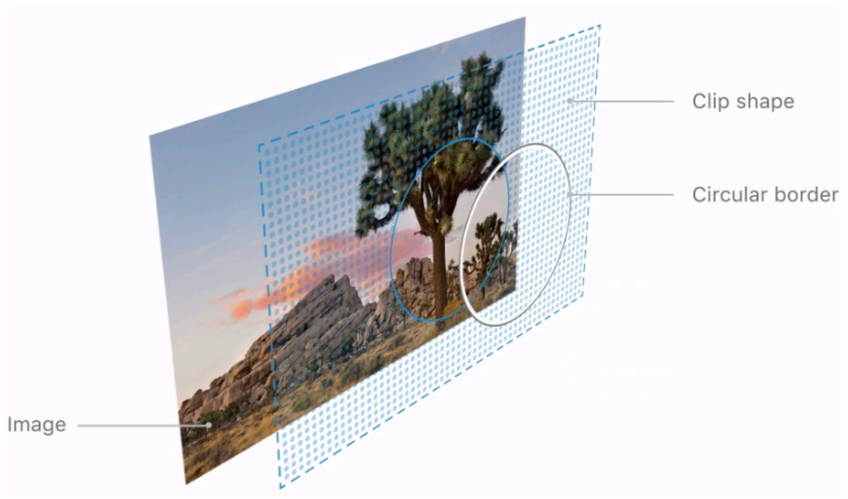
SwiftUI 构建组件

地标页


09:41
S3801
G110
G45
G101
CHANGPING
SHUNYI
S28
G7
S11
G103
Legal
圆明园
皇家园林
北京

Image 组件



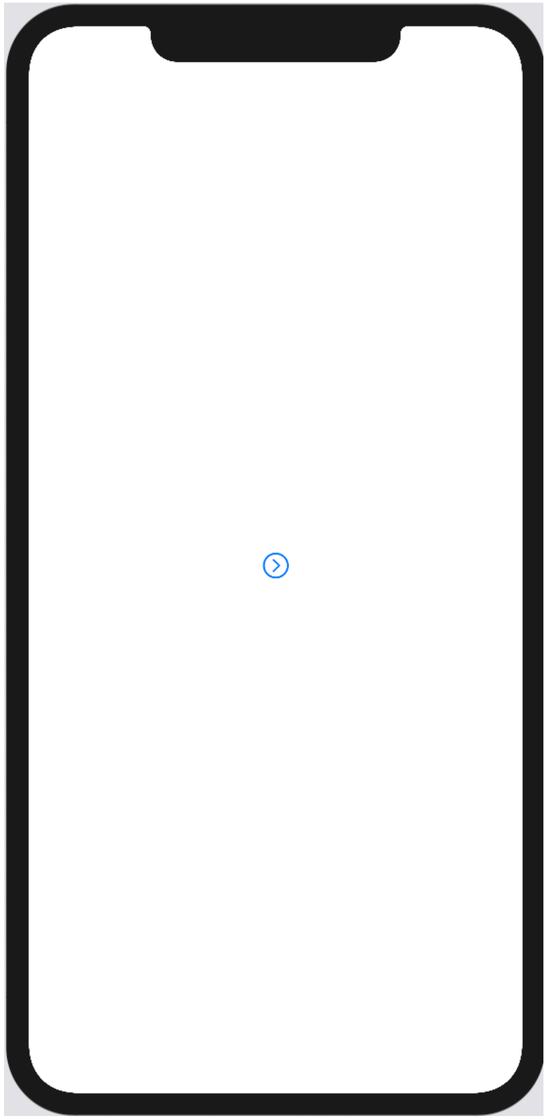


动画

SwiftUI - 动画

- 在 SwiftUI 中，你可以将任意的改变过程封装进一个 withAnimation 块中。默认，SwiftUI 会对这种改变采用 fade in/out 的方式进行动画

SwiftUI 动画



极客时间

SwiftUI 探秘

@propertyWrapper

- 通过 property Wrapper 机制，对一些类似的属性的实现代码做同一封装。
- 通过 @propertyWrapper 可以移除掉一些重复或者类似的代码。

@state

- 通过@State SwiftUI 实现了值的绑定、动态查找和 View 的自动重新绘制。

```
/// value.
@available(iOS 13.0, OSX 10.15, tvOS 13.0, watchOS 6.0, *)
@propertyWrapper public struct State<Value> : DynamicProperty {

    /// Initialize with the provided initial value.
    public init(wrappedValue value: Value)

    /// Initialize with the provided initial value.
    public init(initialValue value: Value)

    /// The current state value.
    public var wrappedValue: Value { get nonmutating set }

    /// Produces the binding referencing this state value
    public var projectedValue: Binding<Value> { get }
}
```

@state

```
@State private var showDetail = false
```

```
var showDetail = State<Bool>(initialValue: false)
var model: Bool{
  get{
    return showDetail.wrappedValue
  }
  set{
    showDetail.wrappedValue = newValue
    view.render()//这里触发view的刷新 执行view.body, 伪码
  }
}
```

课后题

- 查看源码，了解@Binding, @ObservedObject, @EnvironmentObject 等装饰器的作用。

极客时间

扫码试看/订阅

《Swift核心技术与实战》视频课程