Mastering Xcode Previews

Anton Vladimirov, Developer Tools
Nate Chandler, Developer Tools
Rate This Dessert

Churro Cupcake

Description text for the dessert that is being judged that makes it sound like something you'd want to eat.

Callout, Secondary

Rate It
Rate This Dessert

Churro Cupcake

Description text for the dessert that is being judged that makes it sound like something you'd want to eat.

Rate It
Rate This Dessert

Churro Cupcake

Description text for the dessert that is being judged that makes it sound like something you'd want to eat.

Rate it
Churro Cupcake
Description text for the dessert that is being judged that makes it sound like something you'd want to eat.
Churro Cupcake

Description text for the dessert that is being judged that makes it sound like something you’d want to eat.
Rate This Dessert

Churro Cupcake

Description text for the dessert that is being judged.

- - - - -

Slammin'

Rate it
Rate This Dessert

Churro Cupcake
Description text for the dessert that is being judged that makes it sound like something you'd want to eat.

🌟🌟🌟🌟
Rate it
import SwiftUI

struct ContentView: View {
    let dessert: Dessert

    var body: some View {
        VStack {
            Text("Rate This Dessert")
                .font([.largeTitle])
                .resizable() .frame(width: 350, height: 350)
                .clipShape(RoundedRectangle(cornerRadius: 10.0))
                .shadow(radius: 4.0)
            VStack(alignment: .leading, spacing: 10) {
                Text(dessert.name)
                .font([.headline])
                .leadingMargin(.none)
                .foregroundColor(.secondary)
                Text(dessert.description)
                .lineLimit(nil)
                .callOut()
                .foregroundColor(.secondary)
                .frame(minWidth: 0, maxWidth: .infinity, alignment: .leading)
                .padding(.horizontal, 35.0)
            }
            Spacer().frame(height: 20)
            Spacer()
            StarRatingView(rating: dessert.rating)
            Text("Rate it")
                .foregroundColor(.white)
                .padding(.vertical)
                .padding(.horizontal, 150)
                .background(Color(hue: 0.598, saturation: 0.749, brightness: 0.81, opacity: 0.6))
        }
        .padding(.vertical)
    }
}

#if DEBUG
struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView(dessert: Dessert.sample)
    }
}
#endif
How?
```swift
struct CellView : View {
    var body: some View {
        Text("Hello")
    }
}
```
struct CellView : View {
    var body: some View {
        Text("Hello")
    }
}
struct CellView : View {
    var body: some View {
        // dynamically replaced
        // with __preview__body
    }
}

extension CellView {
    var __preview__body: some View {
        Text("Hello")
    }
}
```swift
struct CellView : View {
    var body: some View {
        // dynamically replaced
        // with __preview__body
    }
}
```

```swift
extension CellView {
    var __preview__body: some View {
        Text("Hello")
    }
}
```
struct AnimalCellPreview : PreviewProvider {
    static var previews: some View {
        AnimalCell(.redFox)
    }
}
struct AnimalCellPreview: PreviewProvider {
    static var previews: some View {
        AnimalCell(.redFox)
    }
}
struct AnimalCellPreview: PreviewProvider {
    static var previews: some View {
        AnimalCell(.redFox)
    }
}
struct AnimalCellPreview : PreviewProvider {
    static var previews: some View {
        AnimalCell(.redFox)
    }
}
struct AnimalCell : View {
    let model: AnimalCellModel

    var body: some View {
        HStack {
            Text(model.image)
            HStack {
                Text(model.commonName)
                Text(model.family)
                Text(model.scientificName)
            }
        }
    }
}

struct AnimalCellPreview : PreviewProvider {
    static var previews: some View {
        AnimalCell(.redFox)
    }
}
struct AnimalCell: View {
    let model: AnimalCellModel

    var body: some View {
        HStack {
            Text(model.image)
            HStack {
                Text(model.commonName)
                Text(model.family)
                Text(model.scientificName)
            }
        }
    }
}

struct AnimalCellPreview: PreviewProvider {
    static var previews: some View {
        AnimalCell(.redFox)
    }
}
Menagerie

Cheetah
Felidae
Acinonyx jubatus

Jaguar
Felidae
Panthera onca

Tiger
Felidae
Panthera tigris

Leopard
Felidae
Panthera pardus

Snow Leopard
Felidae
Panthera uncia

Lion
Felidae
Panthera leo

Mountain Lion
Felidae
Puma concolor
Demo

Anton Vladimirov, Developer Tools
Nate Chandler, Developer Tools
Menagerie

Cheetah
Felidae
Acinonyx jubatus

Jaguar
Felidae
Panthera onca

Red fox
Canidae
Vulpus vulpus

Tiger
Felidae
Panthera tigris

Leopard
Felidae
Panthera pardus

Snow Leopard
Felidae
Panthera uncia

Lion
Felidae
Panthera leo

Red fox
Vulpus vulpus

The red fox is an adorable but invasive species. They rarely live solitarily, preferring familial groupings. As the name suggests, their coloration is typically reddish brown with white in some areas. Labeled by the taxonomical classification vulpus vulpus, they are a member of the family canidae.
Red fox
Vulpes

Vulpus vulpus

The red fox is an adorable but invasive species. They rarely live solitarily, preferring familial groupings. As the name suggests, their coloration is typically reddish brown with white in some areas. Labeled by the taxonomical classification vulpus vulpus, they are a member of the family canidae.
Demo

Nate Chandler, Developer Tools
// AnimalCellModel — AnimalCell's View Model

struct AnimalCellModel {
    let commonName: String
    let class: String
    let scientificName: String
    let image: String?
}
// AnimalCell

struct AnimalCell : View {
    let model: AnimalCellModel

    var body: some View {
        HStack {
            Image(model.image)
            VStack {
                Text(model.commonName).font(.title).fontWeight(.bold)
                Text(model.familyName)
                Text(model.scientificName).italic()
            }
        }
    }
}
struct AnimalCellPreviews : PreviewProvider {
    static let models: [AnimalCellModel]
    static var previews: some View {
        ForEach(models.identified(.self)) { model in
            AnimalCell(model: model)
        }
    }
}
// Animal

class Animal {
    let commonName: String
    let class: String
    let genus: String
    let species: String
    weak var clade: Clade?
    let genome: Genome
    let imageName: String
}

// Animal

class Animal {
    let commonName: String
    let class: String
    let genus: String
    let species: String
    weak var clade: Clade?
    let genome: Genome
    let imageName: String
}

// Animal

class Animal {
    let commonName: String
    let class: String
    let genus: String
    let species: String
    weak var clade: Clade?
    let genome: Genome
    let imageName: String
}

Red fox

Canidae

Vulpus vulpus

red_fox.png
extension AnimalCellModel {

    init(for animal: Animal) {
        self.commonName = animal.commonName
        self.scientificName = "\(animal.genus) \(animal.species)"
        ...
    }
}

extension AnimalCellModel {
  init(for animal: Animal) {
    self.commonName = animal.commonName
    self.scientificName = "\(animal.genus) \(animal.species)"
    ...
  }
}
extension AnimalCellModel {

    init(for animal: Animal) {
        self.commonName = animal.commonName
        self.scientificName = "\(animal.genus) \(animal.species)"
    }

}

class AnimalCellModelTest : XCTestCase {

    func testRedFox() {
        let fox: Animal = ...
        let foxModel = AnimalCellModel(for: fox)
        XCTAssertEqual(foxModel.commonName, "Red fox")
        XCTAssertEqual(foxModel.scientificName, "Vulpus vulpus")
    }

}
extension AnimalCellModel {

    init(for animal: Animal) {
        self.commonName = animal.commonName
        self.scientificName = "\(animal.genus) \(animal.species)"
        ...
    }
}

class AnimalCellModelTest : XCTestCase {

    func testRedFox() {
        let fox: Animal = ...
        let foxModel = AnimalCellModel(for: fox)
        XCTAssertEqual(foxModel.commonName, "Red fox")
        XCTAssertEqual(foxModel.scientificName, "Vulpus vulpus")
        ...
    }
}

class AppDelegate : UIApplicationDelegate {
    func application(_ application: UIApplication, didFinishLaunchingWithOptions options: [UIApplication.LaunchOptionsKey : Any]?) -> Bool
}

class SceneDelegate : UIResponder, UIWindowSceneDelegate {
    func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions: UIScene.ConnectionOptions)
}

Multitasking and the Application Lifecycle

WWDC 2019
Summary

Write a preview
Use preview pinning
Use development assets
Use previews in your UIKit/AppKit/WatchKit code
More Information

developer.apple.com/wwdc19/233