## Playgrounds - Get Started with Apps

## Playgrounds - Keep Going with Apps

## Playgrounds - Keep Going with Apps (files)

CreatureDetail

CreatureEditor

CreatureRow

CreatureZoo

CreatureZooExtension

DancingCreatures

MyApp

NavigationExperiment

NavigationSplitViewExperiment

ResizableFont

SlidingRectangle

StoryEditor

TestView

## Playgrounds - Get Started with Apps - Files

### IntroView - Get Started with Apps

```swift
import SwiftUI

struct IntroView: View {
    var body: some View {
        HStack {
            Text("test")
            Image("FriendAndGem")
                .resizable()
                .scaledToFit()
        }
        Text("example text")
        Text("string")
    }
}

struct IntroView_Previews: PreviewProvider {
    static var previews: some View {
        VStack {
            IntroView()
        }
    }
}
```

### ExperimentView - Get Started with Apps

```swift
import SwiftUI

struct ExperimentView: View {
    var body: some View {
        VStack {
            FriendDetailView()

            HStack {
                Image("Blu")
                    .resizable()
                    .scaledToFit()

                VStack {
                    Text("a")
                        .font(.caption)
                    Text("a")
                        .font(.caption)
                }
            }
            HStack {
                Image("Hopper")
            }
        }
    }
}

struct ExperimentView_Previews: PreviewProvider {
    static var previews: some View {
        VStack {
            ExperimentView()
        }
    }
}
```

FriendDetailView - Get Started with Apps

```swift
import SwiftUI

struct FriendDetailView: View {
    var body: some View {
        VStack {
            HStack {
                Image("Friend")
                    .resizable()
                    .scaledToFit()

                VStack {
                    Text("Friend")
                        .font(.largeTitle)
                    Text("a")
                        .font(.caption)
                }
            }

        }
    }
}

struct FriendDetailView_Previews: PreviewProvider {
    static var previews: some View {
        VStack {
            FriendDetailView()
        }
    }
}
```

### MyApp - Get Started with Apps

```
import SwiftUI
import Guide

@main
struct MyApp: App {
    var body: some Scene {
        WindowGroup {
            IntroView()
        }
    }
}
```

## Playgrounds - Keep Going with Apps

### Modifying State - Keep Going with Apps

```
import SwiftUI

struct ConditionalViews: View {

    @State var isOn = false
    @State var isReady = false

    var body: some View {
        VStack {
            Button("Toggle Ready") {
                isReady.toggle()
            }
            if isReady {
                Text("Ready!")
            } else {
                Text("Not Ready")
            }

            if isOn {

                Circle()
                    .frame(maxHeight: 200)
                    .foregroundColor(.yellow)
                Text("On")
            } else {
                Circle()
                    .frame(maxHeight: 200)
                    .foregroundColor(.green)
                Text("Off")

            }

            Button("Press Me") {

                isOn.toggle()
            }
        }
    }
}

struct ConditionalViews_Previews: PreviewProvider {
    static var previews: some View {
        VStack {
            ConditionalViews().assess()
        }
    }
}
```

## Using a Conditional Modifier - Keep Going with Apps

```
import SwiftUI

struct ConditionalCircle: View {
    @State var isOn = false

    var body: some View {
        VStack {

            Circle()
                .frame(maxHeight: 200)
                .foregroundColor( isOn ? .purple : .mint)
                .shadow(color: isOn ? .indigo : .orange, radius: 20)
                .scaleEffect(isOn ? 1 : 0.75)
                .animation(.default, value: isOn)
            Button("Press Me") {
                isOn.toggle()
            }
        }
    }
}
```

## Built In Views - Keep Going with Apps

```
import SwiftUI

struct Bindings: View {
```

```swift
    @State var isOn = false
    @State var color = Color.primary

    var body: some View {
        VStack {

            Toggle("Press Me", isOn: $isOn)

            ColorPicker("Pick", selection: $color)

            Image(systemName: isOn ? "battery.100" : "battery.25")
                .font(.system(size: 150))
                .foregroundColor(color)

            Text("test text")
                .font(.largeTitle)
                .foregroundColor(color)
                .padding()

        }
        .padding()
    }
}
```

Practice with Built in Views - Keep Going with Apps

```swift
import SwiftUI

struct StoryEditor: View {
    @State var name: String = ""
    @State var hobby: String = ""
    @State var favoriteFood: String = ""

    var body: some View {
        VStack {
            Text("Hello, my name is \(name), my favorite hobby is \(hobby) and I
can't stop eating \(favoriteFood)!")

            TextField("Enter name", text: $name)
            TextField("Enter hobby", text: $hobby)
            TextField("Enter favorite food", text: $favoriteFood)

        }
        .padding()
    }
}

struct StoryEditor_Previews: PreviewProvider {
    static var previews: some View {
        StoryEditor()
    }
}
```

```
import SwiftUI

struct SlidingRectangle: View {
    @State var sliderAmount: Double = 0

    var body: some View {
        VStack {
            Slider(value: $sliderAmount)
            Rectangle()
                .frame(width: sliderAmount * 300)
                .foregroundColor(.blue)
        }
        .padding()

    }
}
```

## Navigating in SwiftUI - Keep Going with Apps

NavigationExperiment - Navigating in SwiftUI

```
import SwiftUI

struct NavigationExperiment: View {
    var body: some View {
        VStack {
            NavigationLink("Link 1") {
                Text("This is the destination")
                    .navigationTitle("Destination")
            }

            NavigationLink("Link 2") {
                Text("Destination 2")
                    .navigationTitle("Destination 2")
            }
        }
    }
}

struct NavigationExperiment_Previews: PreviewProvider {
    static var previews: some View {
        NavigationExperiment().assess()
    }
}
```

NavigationSplitViewExperiment - Navigating in SwiftUI

```
import SwiftUI

struct NavigationSplitViewExperiment: View {
    var body: some View {
        VStack {
            NavigationSplitView {
                List {
                    NavigationLink{
                        SlidingRectangle()
                            .navigationTitle("Sliding rectangle")
                    } label: {
                        HStack {
                            Text("Tap to navigate")
                            Spacer()
                            Image(systemName: "arrow.forward.circle")
                                .font(.largeTitle)
                        }
                    }
                    NavigationLink("Second link") {
                        Text("2")
                    }
                    NavigationLink {
                        Image(systemName: "arrow.forward.circle")
                    } label: {
                        Text("New navigation link")
                    }
                }
            } detail: {
                Text("a")
            }
        }
    }
}

struct NavigationSplitViewExperiment_Previews: PreviewProvider {
    static var previews: some View {
        NavigationSplitViewExperiment().assess()
    }
}
```

MyApp - Navigating in SwiftUI

```
import SwiftUI
import Guide

@main
struct MyApp: App {

    var body: some Scene {
        SPCAssessableWindowGroup(app: self, assessmentCandidates: [CreatureZoo()])
{
            NavigationStack {
            ContentView()
                    .navigationTitle("My Creatures")
            }

        }
    }
}
```

ContentView - Navigating in SwiftUI

```
import SwiftUI
import Guide

struct ContentView: View {

    var body: some View {
        SPCAssessableGroup(view: self) {
            List {
                Text("ContentView")

                Section("Dance") {
                    NavigationLink {
                        DancingCreatures()
                            .navigationTitle("Dancing creatures")
                    } label: {
                        Text("make some creatures dance around")
                    }
                }
            }
        }
    }
}
```

## Sharing Data Between Views - Keep Going with Apps

CreatureZoo - Sharing Data Between Views

```
import SwiftUI

class CreatureZoo : ObservableObject {

    @Published var creatures = [
        Creature(name: "Gorilla", emoji: "🦍"),
        Creature(name: "Peacock", emoji: "🦚"),
        Creature(name: "Squid", emoji: "🦑"),
        Creature(name: "T-Rex", emoji: "🦖"),
        Creature(name: "Ladybug", emoji: "🐞"),
    ]
}

struct Creature : Identifiable {
    var name : String
    var emoji : String

    var id = UUID()
    var offset = CGSize.zero
    var rotation : Angle = Angle(degrees: 0)
}
```

ContentView - Sharing Data Between Views

```swift
import SwiftUI
import Guide

struct ContentView: View {
//    @StateObject var data = CreatureZoo()
    @EnvironmentObject var data : CreatureZoo

    var body: some View {
        SPCAssessableGroup(view: self) {
            List {
                Text("ContentView")

                Section("Dance") {
                    NavigationLink {
                        DancingCreatures()
                            .navigationTitle("Dancing Creatures")
                    } label: {
                        Text("Make some creatures dance around")
                    }
                }
                ForEach(data.creatures) { creature in
                    CreatureRow(creature: creature)
                }
//                ForEach(data.creatures) { creature in
//                    HStack {
//                        Text(creature.name)
//                            .font(.title)
//                        Spacer()
//                        Text(creature.emoji)
//                            .resizableFont()
//                            .frame(minWidth: 125)
//                    }
//                }

            }
        }
    }
}
```

MyApp - Sharing Data Between Views

```swift
import SwiftUI
import Guide

@main
struct MyApp: App {
    @StateObject var data = CreatureZoo()

    var body: some Scene {
        SPCAssessableWindowGroup(app: self, assessmentCandidates: [CreatureZoo()])
{
            NavigationStack {
            ContentView()
```

```
                    .navigationTitle("My Creatures")
            }
            .environmentObject(data)
        }
    }
}
```

## Create a New View to Share Data - Keep Going with Apps

DancingCreatures - Create a New View to Share data

```
import SwiftUI
import Guide

struct DancingCreatures: View {
    @EnvironmentObject var data : CreatureZoo

    var body: some View {
        SPCAssessableGroup(view: self) {
            VStack {
                ZStack {
                    ForEach(data.creatures) { creature in
                        Text(creature.emoji)
                            .resizableFont()
                            .offset(creature.offset)
                            .rotationEffect(creature.rotation)
                            .animation(.spring(response: 0.5, dampingFraction:
0.5), value: creature.rotation)
                            .animation(.default, value: creature.offset)
                            .animation(.default.delay(data.indexFor(creature) /
10), value: creature.offset)

                    }
                }
                .onTapGesture {
                    data.randomizeOffsets()
                }
            }
        }
    }
}

struct DancingCreatures_Previews: PreviewProvider {
    static var previews: some View {
        DancingCreatures().environmentObject(CreatureZoo())
    }
}
```

## Add and Delete Creatures - Keep Going with Apps

CreatureEditor - Add and Delete Creatures

```
import SwiftUI
import Guide

struct CreatureEditor: View {
    @State var newCreature : Creature = Creature(name: "", emoji: "")
    @EnvironmentObject var data : CreatureZoo
    @Environment(\.dismiss) var dismiss

    var body: some View {
        SPCAssessableGroup(view: self) {
            VStack(alignment: .leading) {
                Form {
                    Section("Name") {
                        TextField("Name", text: $newCreature.name)

                    }

                    Section("Emoji") {
                        TextField("Emoji", text: $newCreature.emoji)
                    }

                    Section("Creature Preview") {
                        CreatureRow(creature: newCreature)
                    }
                }

            }
            .toolbar {
                ToolbarItem {
                    Button("Add") {
                        data.creatures.append(newCreature)
                        dismiss()
                    }
                }
            }
        }
    }
}

struct CreatureEditor_Previews: PreviewProvider {
    static var previews: some View {
        NavigationStack() {
            CreatureEditor().environmentObject(CreatureZoo())
        }
    }
}
```

ContentView - Add and delete creatures

```
import SwiftUI
import Guide

struct ContentView: View {
//    @StateObject var data = CreatureZoo()
    @EnvironmentObject var data : CreatureZoo
```

```swift
    var body: some View {
        SPCAssessableGroup(view: self) {
            List {
                Text("ContentView")

                Section("Dance") {
                    NavigationLink {
                        DancingCreatures()
                            .navigationTitle("Dancing Creatures")
                    } label: {
                        Text("Make some creatures dance around")
                    }
                }
                ForEach(data.creatures) { creature in

                    NavigationLink {
                        CreatureDetail(creature: creature)
                            .navigationTitle(creature.name)
                    } label: {
                        CreatureRow(creature: creature)
                    }

                }

//                ForEach(data.creatures) { creature in
//                    HStack {
//                        Text(creature.name)
//                            .font(.title)
//                        Spacer()
//                        Text(creature.emoji)
//                            .resizableFont()
//                            .frame(minWidth: 125)
//                    }
//                }
                .onDelete { indexSet in
                    data.creatures.remove(atOffsets: indexSet)
                }

            }
            .toolbar {
                ToolbarItem {
                    NavigationLink("Add") {
                        CreatureEditor()
                            .navigationTitle("Add Creature")
                    }
                }
            }
        }
    }
}
```

## Add a CreatureDetail View - Keep Going with Apps

CreatureDetail - Add a CreatureDetail View

```
import SwiftUI
struct CreatureDetail: View {
    let creature : Creature

    @State var isScaled = false
    @State var color = Color.white
    @State var shadowRadius : CGFloat = 0.5
    @State var angle = Angle(degrees: 0)

    var body: some View {
        VStack {
            Text(creature.emoji)
                .resizableFont()
                .colorMultiply(color)
                .shadow(color: color, radius: shadowRadius * 40)
                .rotation3DEffect(isScaled ? Angle(degrees: 0) : Angle(degrees:
360), axis: (x: 5, y: 2, z: 1))
                .scaleEffect(isScaled ? 1.5 : 1)
                .animation(.spring(response: 0.5, dampingFraction: 0.5,
blendDuration: 0.5), value: isScaled)

            Button("Scale") {
                isScaled.toggle()
            }

            ColorPicker("Choose a Color", selection: $color)

            HStack {
                Text("Shadow")
                Slider(value: $shadowRadius)
            }

        }
        .padding()
    }
}

struct CreatureDetail_Previews: PreviewProvider {
    static var previews: some View {
        CreatureDetail(creature: CreatureZoo().creatures.randomElement() ??
Creature(name: "Panda", emoji: "🐼")).assess()
    }
}
```

ContentView - Add a CreatureDetail View

```
import SwiftUI
import Guide

struct ContentView: View {
//    @StateObject var data = CreatureZoo()
    @EnvironmentObject var data : CreatureZoo
    var body: some View {
        SPCAssessableGroup(view: self) {
            List {
                Text("ContentView")

                Section("Dance") {
                    NavigationLink {
                        DancingCreatures()
                            .navigationTitle("Dancing Creatures")
                    } label: {
                        Text("Make some creatures dance around")
                    }
                }
                ForEach(data.creatures) { creature in

                    NavigationLink {
                        CreatureDetail(creature: creature)
                            .navigationTitle(creature.name)
                    } label: {
                        CreatureRow(creature: creature)
                    }

                }

//                ForEach(data.creatures) { creature in
//                    HStack {
//                        Text(creature.name)
//                            .font(.title)
//                        Spacer()
//                        Text(creature.emoji)
//                            .resizableFont()
//                            .frame(minWidth: 125)
//                    }
//                }
                .onDelete { indexSet in
                    data.creatures.remove(atOffsets: indexSet)
                }

            }
            .toolbar {
                ToolbarItem {
                    NavigationLink("Add") {
                        CreatureEditor()
                            .navigationTitle("Add Creature")
                    }
                }
            }
        }
    }
}
```

## Playgrounds Keep Going with Apps (files)

### Bindings - Keep Going with Apps

```swift
import SwiftUI

struct Bindings: View {
    @State var isOn = false
    @State var color = Color.primary

    var body: some View {
        VStack {
            Toggle("Press Me", isOn: $isOn)
            ColorPicker("Pick", selection: $color)

            Image(systemName: isOn ? "battery.100" : "battery.25")
                .font(.system(size: 150))
                .foregroundColor(color)

            Text("test text")
                .font(.largeTitle)
                .foregroundColor(color)
                .padding()

        }
        .padding()
    }
}

struct Bindings_Previews: PreviewProvider {
    static var previews: some View {
        Bindings().assess()
    }
}
```

## ConditionalCircle - Keep Going with Apps

```swift
import SwiftUI

struct ConditionalCircle: View {
    @State var isOn = false

    var body: some View {
        VStack {
            Circle()
                .frame(maxHeight: 200)
                .foregroundColor( isOn ? .purple : .mint )
                .shadow(color: isOn ? .indigo : .orange, radius: 20)
                .scaleEffect(isOn ? 1 : 0.75)
                .animation(.default, value: isOn)

            Button("Press Me") {
                isOn.toggle()
            }
        }
    }
}
```

```
struct SwiftUIView_Previews: PreviewProvider {
    static var previews: some View {
        ConditionalCircle().assess()
    }
}
```

## ConditionalViews - Keep Going with Apps

```
import SwiftUI

struct ConditionalViews: View {
    @State var isOn = false
    @State var isReady = false

    var body: some View {
        VStack {
            Button("Toggle Ready") {
                isReady.toggle()
            }
            if isReady {
                Text("Ready!")
            } else {
                Text("Not Ready")
            }

            if isOn {
                Circle()
                    .frame(maxHeight: 200)
                    .foregroundColor(.yellow)
                Text("On")
            } else {
                Circle()
                    .frame(maxHeight: 200)
                    .foregroundColor(.green)
                Text("Off")

            }
            Button("Press Me") {
                isOn.toggle()
            }
        }
    }
}

struct ConditionalViews_Previews: PreviewProvider {
    static var previews: some View {
        VStack {
            ConditionalViews().assess()
        }
    }
}
```

ContentView - Keep Going with Apps

```swift
import SwiftUI
import Guide

struct ContentView: View {
//    @StateObject var data = CreatureZoo()
    @EnvironmentObject var data : CreatureZoo

    var body: some View {
        SPCAssessableGroup(view: self) {
            List {
                Text("ContentView")

                Section("Dance") {
                    NavigationLink {
                        DancingCreatures()
                            .navigationTitle("Dancing Creatures")
                    } label: {
                        Text("Make some creatures dance around")
                    }
                }
                ForEach(data.creatures) { creature in

                    NavigationLink {
                        CreatureDetail(creature: creature)
                            .navigationTitle(creature.name)
                    } label: {
                        CreatureRow(creature: creature)
                    }

                }
//                ForEach(data.creatures) { creature in
//                    HStack {
//                        Text(creature.name)
//                            .font(.title)
//                        Spacer()
//                        Text(creature.emoji)
//                            .resizableFont()
//                            .frame(minWidth: 125)
//                    }
//                }
                .onDelete { indexSet in
                    data.creatures.remove(atOffsets: indexSet)
                }

            }
            .toolbar {
                ToolbarItem {
                    NavigationLink("Add") {
                        CreatureEditor()
                            .navigationTitle("Add Creature")
                    }
                }
            }
        }
    }
}
```

## CreatureDetail - Keep Going with Apps

```
import SwiftUI
struct CreatureDetail: View {
    let creature : Creature

    @State var isScaled = false
    @State var color = Color.white
    @State var shadowRadius : CGFloat = 0.5
    @State var angle = Angle(degrees: 0)

    var body: some View {
        VStack {
            Text(creature.emoji)
                .resizableFont()
                .colorMultiply(color)
                .shadow(color: color, radius: shadowRadius * 40)
                .rotation3DEffect(isScaled ? Angle(degrees: 0) : Angle(degrees:
360), axis: (x: 5, y: 2, z: 1))
                .scaleEffect(isScaled ? 1.5 : 1)
                .animation(.spring(response: 0.5, dampingFraction: 0.5,
blendDuration: 0.5), value: isScaled)

            Button("Scale") {
                isScaled.toggle()
            }

            ColorPicker("Choose a Color", selection: $color)

            HStack {
                Text("Shadow")
                Slider(value: $shadowRadius)
            }

        }
        .padding()
    }
}

struct CreatureDetail_Previews: PreviewProvider {
    static var previews: some View {
        CreatureDetail(creature: CreatureZoo().creatures.randomElement() ??
Creature(name: "Panda", emoji: "🐼")).assess()
    }
}
```

## CreatureEditor - Keep Going with Apps

```
import SwiftUI
import Guide
```

```swift
struct CreatureEditor: View {
    @State var newCreature : Creature = Creature(name: "", emoji: "")
    @EnvironmentObject var data : CreatureZoo
    @Environment(\.dismiss) var dismiss

    var body: some View {
        SPCAssessableGroup(view: self) {
            VStack(alignment: .leading) {
                Form {
                    Section("Name") {
                        TextField("Name", text: $newCreature.name)

                    }

                    Section("Emoji") {
                        TextField("Emoji", text: $newCreature.emoji)
                    }

                    Section("Creature Preview") {
                        CreatureRow(creature: newCreature)
                    }
                }

            }
            .toolbar {
                ToolbarItem {
                    Button("Add") {
                        data.creatures.append(newCreature)
                        dismiss()
                    }
                }
            }
        }
    }
}

struct CreatureEditor_Previews: PreviewProvider {
    static var previews: some View {
        NavigationStack() {
            CreatureEditor().environmentObject(CreatureZoo())
        }
    }
}
```

CreatureRow - Keep Going with Apps

```swift
import SwiftUI

struct CreatureRow: View {
    var creature : Creature

    var body: some View {
        HStack {
            Text(creature.name)
                .font(.title)

            Spacer()

            Text(creature.emoji)
                .resizableFont()
                .frame(minWidth: 125)
        }


    }
}

struct CreatureRow_Previews: PreviewProvider {
    static var previews: some View {
        CreatureRow(creature: Creature(name: "Dodo Bird", emoji: "🦤"))
    }
}
```

CreatureZoo - Keep Going with Apps

```
import SwiftUI

class CreatureZoo : ObservableObject {
    @Published var creatures = [
        Creature(name: "Gorilla", emoji: "🦍"),
        Creature(name: "Peacock", emoji: "🦚"),
        Creature(name: "Squid", emoji: "🦑"),
        Creature(name: "T-Rex", emoji: "🦖"),
        Creature(name: "Ladybug", emoji: "🐞"),
    ]
}

struct Creature : Identifiable {
    var name : String
    var emoji : String

    var id = UUID()
    var offset = CGSize.zero
    var rotation : Angle = Angle(degrees: 0)
}
```

CreatureZooExtension - Keep Going with Apps

```
import SwiftUI

extension CreatureZoo {
    func randomizeOffsets() {
        for index in creatures.indices {
            creatures[index].offset = CGSize(width: CGFloat.random(in: -200...200),
height: CGFloat.random(in: -200...200))
```

```
                creatures[index].rotation = Angle(degrees: Double.random(in: 0...720))
            }
        }

        func synchronizeOffsets() {
            let randomOffset = CGSize(width: CGFloat.random(in: -200...200), height:
    CGFloat.random(in: -200...200))
            for index in creatures.indices {
                creatures[index].offset = randomOffset
            }
        }

        func indexFor(_ creature: Creature) ->  Double {
            if let index = creatures.firstIndex(where: { $0.id == creature.id }) {
                return Double(index)
            }
            return 0.0
        }
    }
```

DancingCreatures - Keep Going with Apps

```
import SwiftUI
import Guide

struct DancingCreatures: View {
    @EnvironmentObject var data : CreatureZoo

    var body: some View {
        SPCAssessableGroup(view: self) {
            VStack {
                ZStack {
                    ForEach(data.creatures) { creature in
                        Text(creature.emoji)
                            .resizableFont()
                            .offset(creature.offset)
                            .rotationEffect(creature.rotation)
                            .animation(.spring(response: 0.5, dampingFraction:
0.5), value: creature.rotation)
                            .animation(.default, value: creature.offset)
                            .animation(.default.delay(data.indexFor(creature) /
10), value: creature.offset)

                    }
                }
                .onTapGesture {
                    data.randomizeOffsets()
                }
            }
        }
    }
}

struct DancingCreatures_Previews: PreviewProvider {
    static var previews: some View {
        DancingCreatures().environmentObject(CreatureZoo())
    }
}
```

## MyApp - Keep Going with Apps

```
import SwiftUI
import Guide

@main
struct MyApp: App {
    @StateObject var data = CreatureZoo()

    var body: some Scene {
        SPCAssessableWindowGroup(app: self, assessmentCandidates: [CreatureZoo()])
{
            NavigationStack {
            ContentView()
                    .navigationTitle("My Creatures")
            }
            .environmentObject(data)
        }
    }
}
```

## NavigationExperiment - Keep Going with Apps

```
import SwiftUI

struct NavigationExperiment: View {
    var body: some View {
        VStack {
            NavigationLink("Link 1") {
                Text("This is the destination")
                    .navigationTitle("Destination")
            }

            NavigationLink("Link 2") {
                Text("Destination 2")
                    .navigationTitle("Destination 2")
            }
        }
    }
}

struct NavigationExperiment_Previews: PreviewProvider {
    static var previews: some View {
        NavigationExperiment().assess()
    }
}
```

## NavigationSplitViewExperiment - Keep Going with Apps

```
import SwiftUI

struct NavigationSplitViewExperiment: View {
    var body: some View {
        VStack {
            NavigationSplitView {
                List {
                    NavigationLink{
                        SlidingRectangle()
                            .navigationTitle("Sliding rectangle")
                    } label: {
                        HStack {
                            Text("Tap to navigate")
                            Spacer()
                            Image(systemName: "arrow.forward.circle")
                                .font(.largeTitle)
                        }
                    }
                    NavigationLink("Second link") {
                        Text("2")
                    }
                    NavigationLink {
                        Image(systemName: "arrow.forward.circle")
                    } label: {
                        Text("New navigation link")
                    }
                }
            } detail: {
                Text("a")
            }
        }
    }
}

struct NavigationSplitViewExperiment_Previews: PreviewProvider {
    static var previews: some View {
        NavigationSplitViewExperiment().assess()
    }
}
```

## ResizableFont - Keep Going with Apps

```
import SwiftUI

extension View {
    func resizableFont(maximumFontSize: Double = 125, minimumScaleFactor: Double =
0.01) -> some View {
        self.modifier(FlexibleFontModifier(maximumFontSize: maximumFontSize,
minimumScaleFactor: minimumScaleFactor))
    }
}
```

```
struct FlexibleFontModifier: ViewModifier {
    var maximumFontSize: Double
    var minimumScaleFactor : Double

    func body(content: Content) -> some View {
        content
            .font(.system(size: maximumFontSize))
            .minimumScaleFactor(minimumScaleFactor)
    }
}
```

SlidingRectangle - Keep Going with Apps

```
import SwiftUI

struct SlidingRectangle: View {
    @State var sliderAmount: Double = 0

    var body: some View {
        VStack {
            Slider(value: $sliderAmount)
            Rectangle()
                .frame(width: sliderAmount * 300)
                .foregroundColor(.blue)
        }
        .padding()

    }
}

struct SlidingRectangle_Previews: PreviewProvider {
    static var previews: some View {
        SlidingRectangle().assess()
    }
}
```

StoryEditor - Keep Going with Apps

```
import SwiftUI

struct StoryEditor: View {
    @State var name: String = ""
    @State var hobby: String = ""
    @State var favoriteFood: String = ""

    var body: some View {
        VStack {
            Text("Hello, my name is \(name), my favorite hobby is \(hobby) and I
can't stop eating \(favoriteFood)!")

            TextField("Enter name", text: $name)
            TextField("Enter hobby", text: $hobby)
            TextField("Enter favorite food", text: $favoriteFood)

        }
        .padding()
    }
}

struct StoryEditor_Previews: PreviewProvider {
    static var previews: some View {
        StoryEditor()
    }
}
```

TestView - Keep Going with Apps

```
import SwiftUI

struct TestView: View {
    @State var isOn = false

    var body: some View {
        VStack {
            Button("Press Me") {
                isOn.toggle()
            }

            Circle()
                .frame(maxHeight: 200)
                .foregroundColor(isOn ? .yellow : .black)
                .shadow(color: isOn ? .red : .green, radius: 8)
                .animation(.default, value: isOn)


        }
    }
}

struct TestView_Previews: PreviewProvider {
    static var previews: some View {
        TestView()
    }
}
```