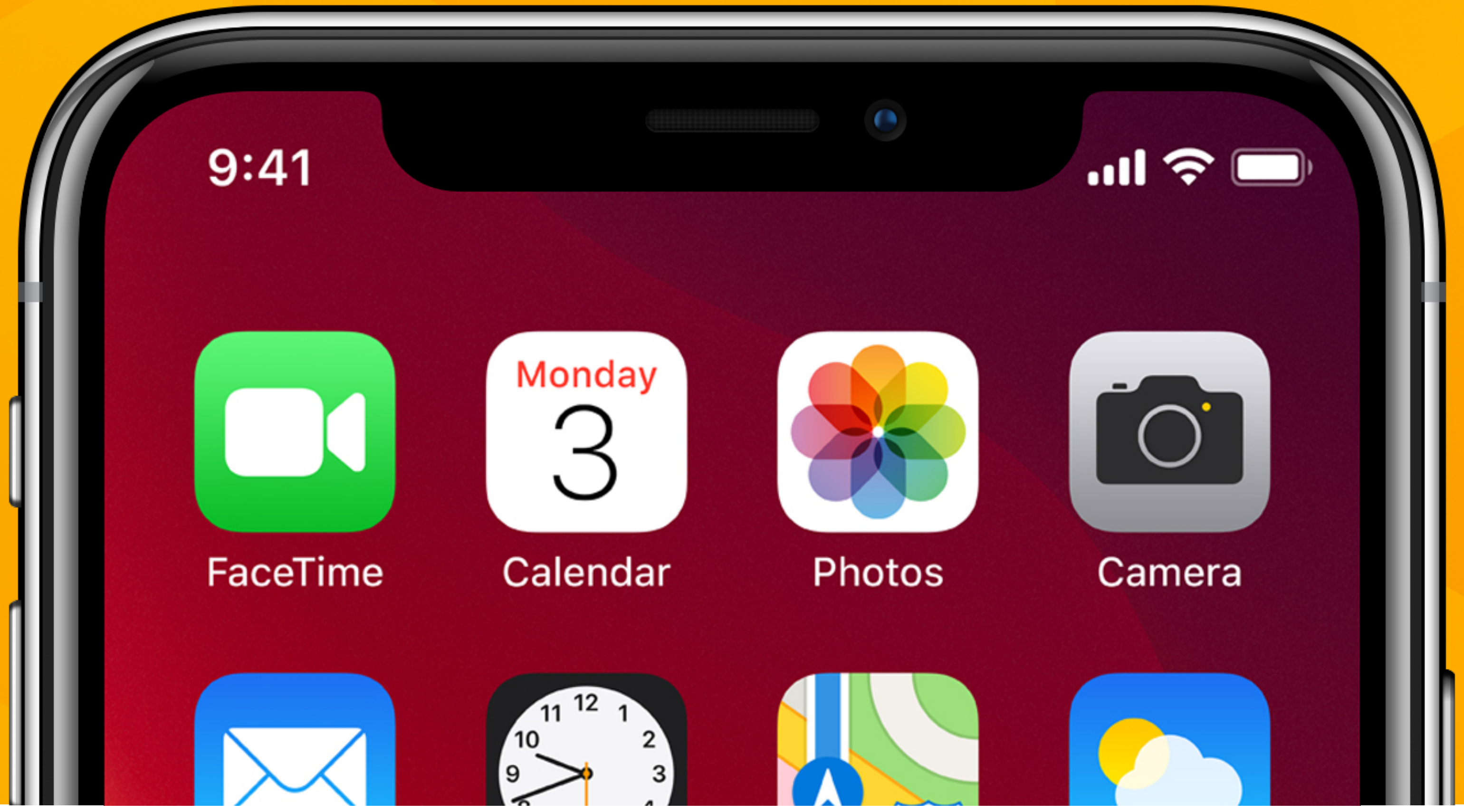


iOS Application Development

Lecture 10: Introducing SwiftUI

Prof. Dr. Jan Borchers
Media Computing Group
RWTH Aachen University

WS ’22/’23 • hci.rwth-aachen.de/ios

Recap

- Swift Generics
  - How to specify type constraints?
  - How to use generic types in protocols?
- Diffable Data Sources
  - How to create smooth animations in `CollectionViews`?
- CollectionViews
  - What are the three components?

A Brief History of SwiftUI

- 2014 Apple releases the Swift language as successor to Objective-C
- 2015 Development of SwiftUI begins at Apple
- 2019 SwiftUI is introduced officially with iOS 13
- 2022 Apple ships several new parts and entire apps on iOS and macOS using SwiftUI

SwiftUI: the Big Messages

1. Object-Oriented Programming is Dead, Long Live **Declarative Programming!**
2. **MVVM** is the corresponding modern improvement over MVC
3. Modern universal languages can describe UIs like **domain-specific languages**
4. You can design a UI **graphically and in code simultaneously**
5. The best app languages must **evolve together** with a UI library and IDE
6. Declarative Programming simplifies development across **mobile and desktop**
7. SwiftUI is a current **case study of a paradigm shift** across a major OS family

Hello SwiftUI!

Hello Aachen
Hello Aachen 〉 iPhone 14 Pro
Hello Aachen: Ready | Today at 17:52
ContentView.swift
Hello Aachen 〉 Hello Aachen 〉 ContentView.swift 〉 No Selection
Hello Aachen
Hello Aachen
Hello_AachenApp.swift
ContentView.swift
Assets.xcassets
Preview Content
Preview Assets.xcassets
Content View
Hello, Aachen!
Filter
Line: 14 Col: 1

```swift
import SwiftUI

struct ContentView: View {
    var body: some View {
        Text("Hello, Aachen!")
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

Looking at the Code: the Shortest SwiftUI App

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        Text("Hello, Aachen!")
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

Composing Views

- The **body** property can only return **one** view
- To compose views, they need to be embedded into layout views like **`VStack`**
- Their initializers use **trailing closures** for multiple child views (max. 10)
- Note that this makes the code begin to look like a hierarchical UI **layout tree!**
  - *"Modern universal languages can describe UIs like **domain-specific languages**"*

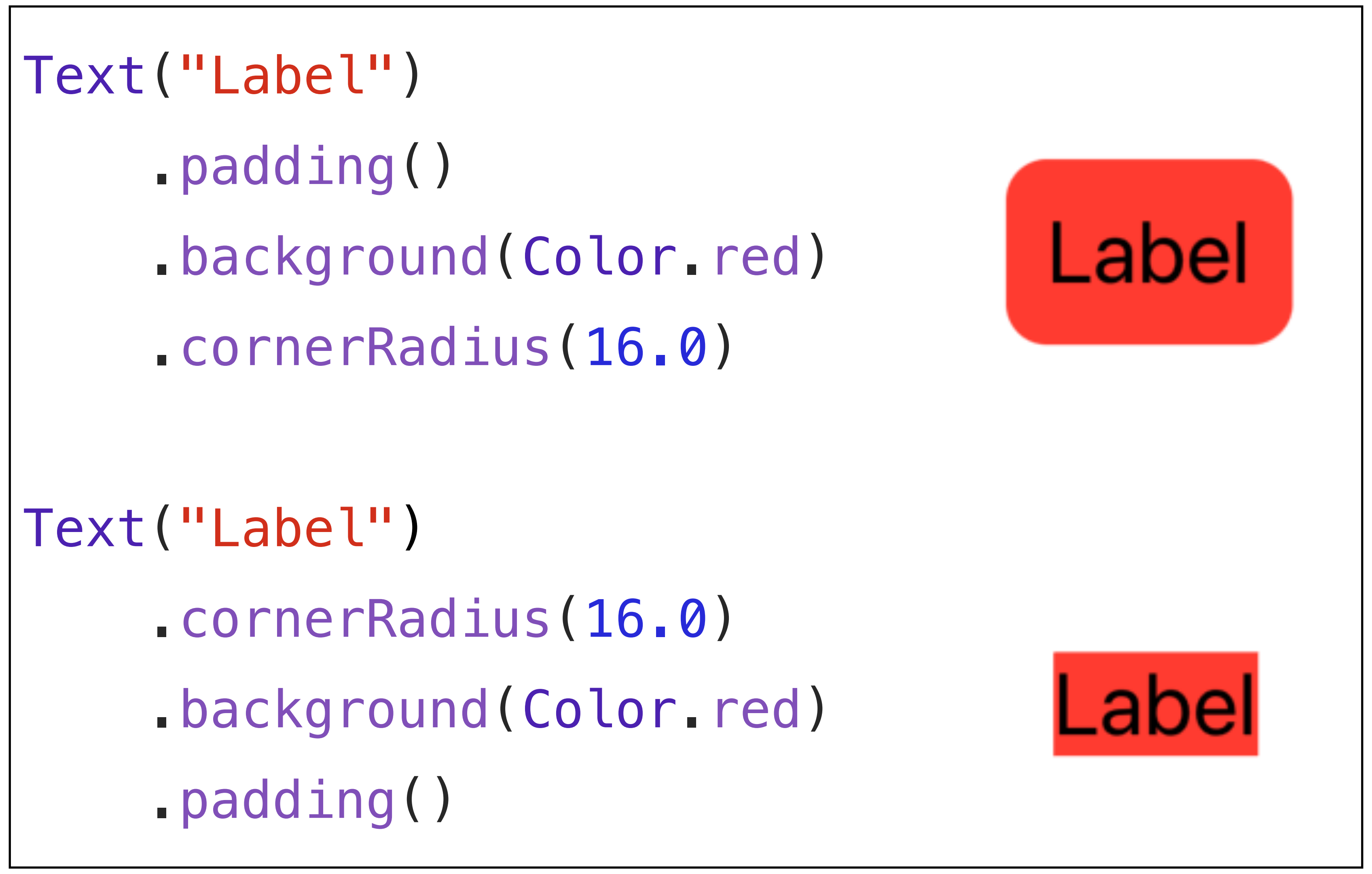
```
import SwiftUI

struct ContentView: View {
    var body: some View {
        VStack {
            Image(systemName: "globe")
            Text("Hello, Aachen!")
        }
    }
}
…
```

Modifiers

Modifiers

- Modifiers allow us to adjust Views
- They are View methods returning another View
- Have (optional) parameters
  - E.g., `spacing` for VStack
- Order matters
  - Executed first to last
- If applied to containers, they are also applied to children (unless property is overridden)



```
Text("Label")
    .padding()
    .background(Color.red)
    .cornerRadius(16.0)

Text("Label")
    .cornerRadius(16.0)
    .background(Color.red)
    .padding()
```

Common Modifiers

- .font
  - Applies font to all text in a view
  - Predefined fonts such as `.largeTitle`
- .foregroundColor
- .background
  - Sets the background to a style
  - Adds a layer behind the view
  - Must conform to `ShapeStyle`
- .frame
  - Positions view within an invisible frame having the specified size constraints
  - .frame(maxWidth: .infinity) extends view to device edges
- .padding
  - Adds space around a view

Xcode Preview and Inspector

Preview

- Lets you preview your layout in the **Canvas,** without launching the simulator

- Changes instantly while editing code

- Provides dummy data to test your layout
    - Useful if data is not static

- Can preview different devices and different modes (dark mode, dynamic text size,…)

```swift
import SwiftUI

struct ContentView: View {

    var myText: String = ""

    var body: some View {
        VStack {
            Text(myText)
                .font(.largeTitle)
                .foregroundColor(Color.orange)
                .padding([.top, .leading, .bottom], 20.0)
                .padding([.trailing], 10.0)
                .bold()
            Image(systemName: "globe")
                .imageScale(.large)
                .foregroundColor(.accentColor)
        }
        .padding()
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView(myText: "Hello, iOS!”) // dummy data
    }
}
```

Common Preview Options

- `.preferredColorScheme`
  - Sets the color scheme (e.g., dark mode)
- `.previewDevice`
  - Allows us to set the device
- `.environment`
  - Sets properties of the used environment such as a dynamic type size or truncation mode

Attribute Inspector

- Powerful tool to adjust properties of views
- Set, change, enable, or disable modifiers and other properties
- Changes affect the code and vice versa
- Smartly adapts the code
  - E.g., combines `.top` and `.bottom` padding to `.vertical`
- "*You can design a UI* ***graphically and in code simultaneously***"
- "*The best app languages must* ***evolve together*** *with a UI library and IDE*"