Using Basic Widgets

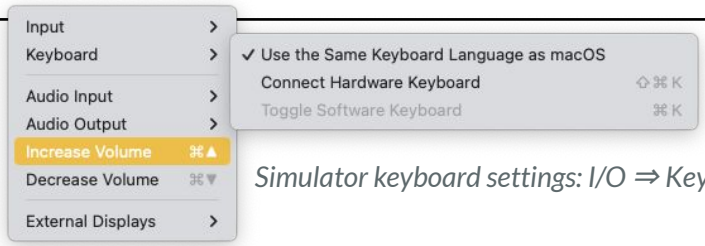
Commonly Used Widgets

- Input Widgets
  - TextField, SecureField, Toggle Switch, Slider, Pickers (Date, Color, general pickers)
  - These widgets require **2-way variable binding**
    - Changes made in your code to the variable are reflected on the UI
    - Changes on the UI are captured by your @State variable
    - Use $ prefix when supplying your @State variable to the widget
    - Look for arguments with `Binding<XXXX>` type in SwiftUI documentation
- Layout Container Widgets

Input Widgets

TextField (with 2-way binding)

```
struct MyView: View {
  @State private var name = “”
  var body: some View {
    TextField(“Enter your name”, text:$name)
    SecureField(“Password”, text: $pwd)
  }
}
```



Simulator keyboard settings: I/O ⇒ Keyboard

TextField + Customized Keyboard Type

```
struct MyView: View {
  @State private var name = “”
  var body: some View {
    TextField(“Enter your name”, text: $name)
      .keyboardType(.emailAddress)
  }
}
```

UIKeyboardType:

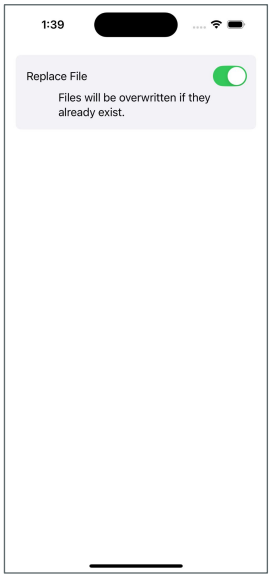
- .default
- .asciiCapable
- .numbersAndPunctuation
- .URL
- .numberPad
- .phonePad
- .namePhonePad
- .emailAddress
- *any many more*

TextField + Keyboard Focus

```
struct MyView: View {
  @State private var name = “”
  @FocusState private var nameHasFocus: Bool
  var body: some View {
    TextField(“Enter your name”, text: $name)
      .focused($nameHasFocus)
    Button(“OK”) {
      // Dismiss the virtual keyboard
      nameHasFocus = false
    }
  }
}
```

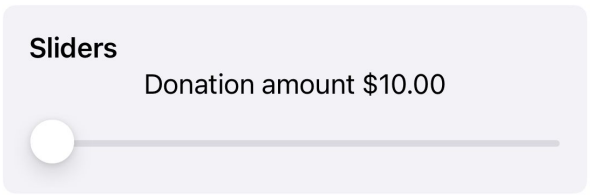
Toggle Switch

```
struct MyView: View {
  @State private var overWrite = false
  var body: some View {
    Toggle(isOn: $overWrite) {
      Text("Replace File")
    }
    if overWrite {
      Text("Files will be overwritten if they already exist.")
    } else {
      Text("Existing files will not be overwritten.")
    }
  }
}
```



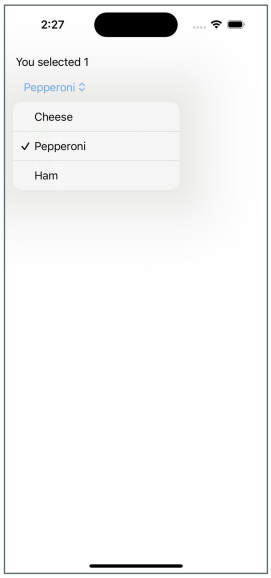
Slider

```
struct MyView: View {
  @State private var donateAmount = 0.0
  var body: some View {
    VStack {
       let amt = donateAmount.formatted(.currency(code: "USD"))
       Text("Donation amount \(amt)")
       Slider(value: $donateAmount,
              in: 0...5000)
    }
  }
}
```



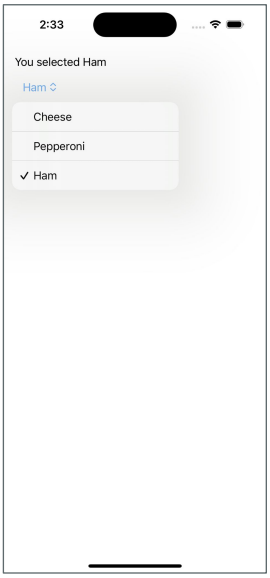
Picker (select by index)

```
struct myView: View {
  // The selection array does not have to be a @State
  let toppings = ["Cheese", "Pepperoni", "Ham"]
  @State private var selectedIndex: Int = 0
  var body: some View {
    VStack {
      Text("You selected \(selectedIndex)")
      Picker("Choose a topping", selection: $selectedIndex) {
        ForEach(0 ..< toppings.count, id: \.self) { k in
          Text(toppings[k]).tag(k)
        }
      }
      .pickerStyle(MenuPickerStyle())
    }
  }
}
```



Picker (select by entry)

```
struct myView: View {
  // The selection array does not have to be a @State
  let toppings = ["Cheese", "Pepperoni", "Ham"]
  @State private var selectedTopping: String = toppings[0]
  var body: some View {
    VStack {
      Text("You selected \(selectedTopping)")
      Picker("Choose a topping", selection: $selectedTopping) {
        ForEach(0 ..< toppings.count, id: \.self) { k in
          Text(toppings[k]).tag(toppings[k])
        }
      }
      .pickerStyle(MenuPickerStyle())
    }
  }
}
```



Layout Container Widgets

Common Layout Containers

Jetpack Compose	SwiftUI	Usage
Column	VStack	Arrange child views along the Y-axis
Row	HStack	Arrange child views along the X-axis
Box	ZStack	Arrange child views along the Z-axis (First child furthest from your eyes, Last child nearest from your eyes)

Column & Alignment

```
VStack (alignment: .leading) {
  Image(systemName: "cup.and.saucer")
  Text("A cup of coffee")
}
```


```
VStack (alignment: .center) {
  Image(systemName: "cup.and.saucer")
  Text("A cup of coffee")
}
```


```
VStack (alignment: .trailing) {
  Image(systemName: "cup.and.saucer")
  Text("A cup of coffee")
}
```