SwiftUI Tips & Tricks

@stuffmc, Cocoaheads Feb. '20

SwiftUI

&

SwiftUI "Installation" in UIKit

```
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication,
            didFinishLaunchingWithOptions launchOptions:
            [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

        window?.rootViewController = UIHostingController(rootView: Content)


open class UIHostingController<Content> : UIViewController where Content : View {
```

View != UIView

- Views are `structs`, not `classes`
- So they don't inherit...
- Always have a `var body: some View` property
- `body` returns ONE View, e.g. `Text("Is it me?")`

ONE (?!?) View?

- Enter the world of *Z/V/HStack*, *Group*, ...
- Tupple of 2, 3, ... TEN (so Max 10 in a "Group")
- Ridiculous but helpful to organize :)

UIKit in SwiftUI

UIKit in SwiftUI

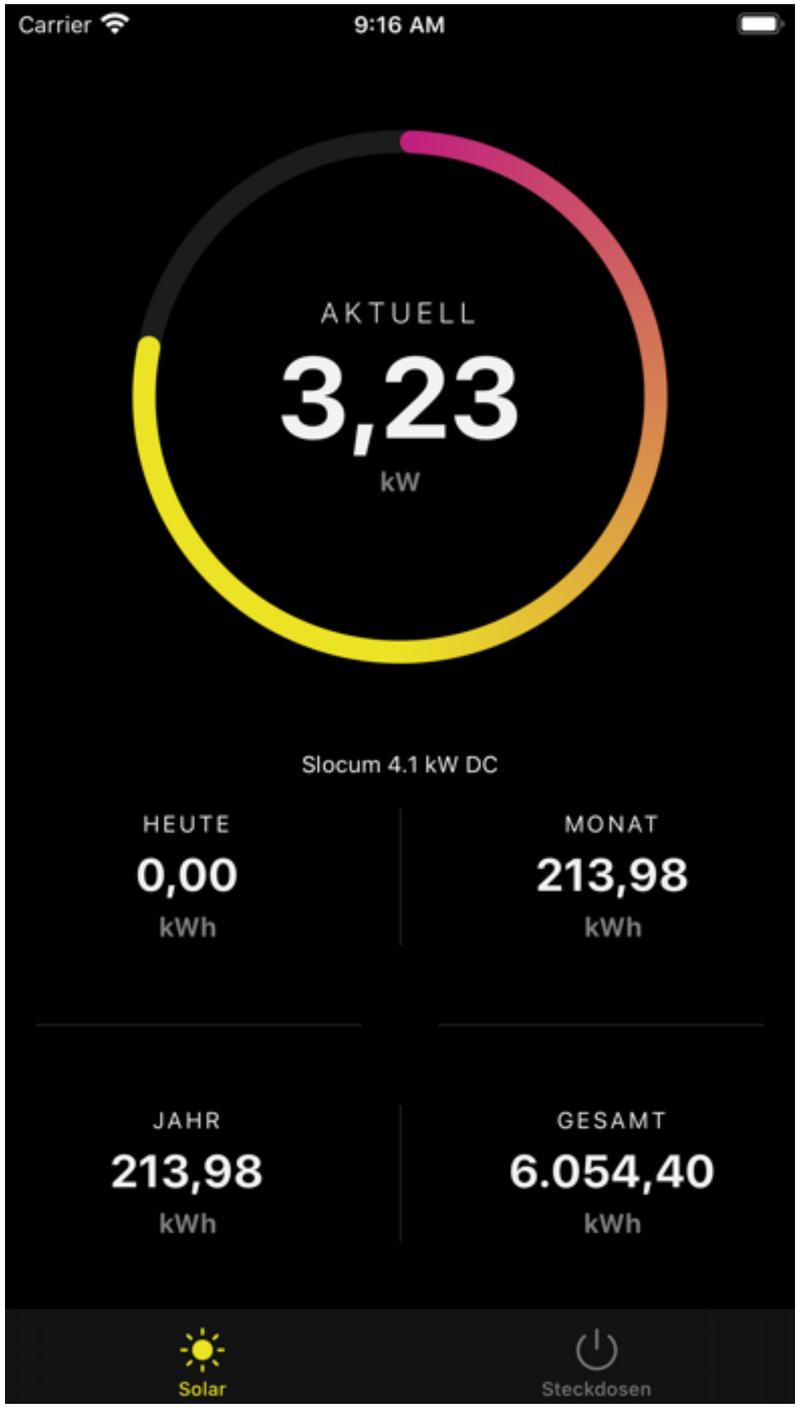
```
struct ActivityIndicator: UIViewRepresentable {
    func makeUIView(context: Context) -> UIActivityIndicatorView {
        return UIActivityIndicatorView()
    }

    func updateUIView(_ activityIndicator: UIActivityIndicatorView, context: Context) {
        activityIndicator.startAnimating()
    }
}

struct ContentView : View {
    var body: some View {
        ActivityIndicator()
    }
}
```

Expose start/stopAnimating with a Binding

```
struct ActivityIndicator: UIViewRepresentable {
    @Binding var isAnimating: Bool

    func updateUIView(_ activityIndicator: UIActivityIndicatorView, context: Context) {
        if isAnimating {
            activityIndicator.startAnimating()
        } else {
            activityIndicator.stopAnimating()
        }
    }
}
```

Previews

```
struct DashboardView_Previews : PreviewProvider {
    static var previews: some View {
        Group {
            DashboardView().environmentObject(AppEnvironment.example_with_data.solarPowerMonitor)
                .previewDevice(PreviewDevice(rawValue: "iPhone 11"))
                .previewDisplayName("iPhone 11")
                .environment(\.colorScheme, .light)

            DashboardView().environmentObject(AppEnvironment.example_with_data.solarPowerMonitor)
                .previewDevice(PreviewDevice(rawValue: "iPhone 11"))
                .previewDisplayName("iPhone 11")
                .environment(\.colorScheme, .dark)

            DashboardView().environmentObject(AppEnvironment.example_with_data.solarPowerMonitor)
                .previewDevice(PreviewDevice(rawValue: "iPhone 8"))
                .previewDisplayName("iPhone 8")
                .environment(\.colorScheme, .light)
```



ArcView

```
struct ArcView: View {
 var body: some View {
   ZStack {
    ArcShape(start: ...)
        .stroke(lineWidth: self.width)
        .foregroundColor(Color.gray)
        .opacity(0.2)
    ArcShape(start: ...)
        .stroke(self.angularGradient,
        lineWidth: self.width)
   }
 }
}
```

ArcShape

```
struct ArcShape: Shape {
 func path(in rect: CGRect) -> Path {
  (...)
  var path = Path()
  path.add(center: , radius: , start: newStart, end: , clockwise: false)
  let strokeStyle = StrokeStyle(lineWidth: width,
                                lineCap: roundCorners ? .round : .butt)
  return path.strokedPath(strokeStyle)
 }
}
```

```
public protocol Shape : Animatable, View
```

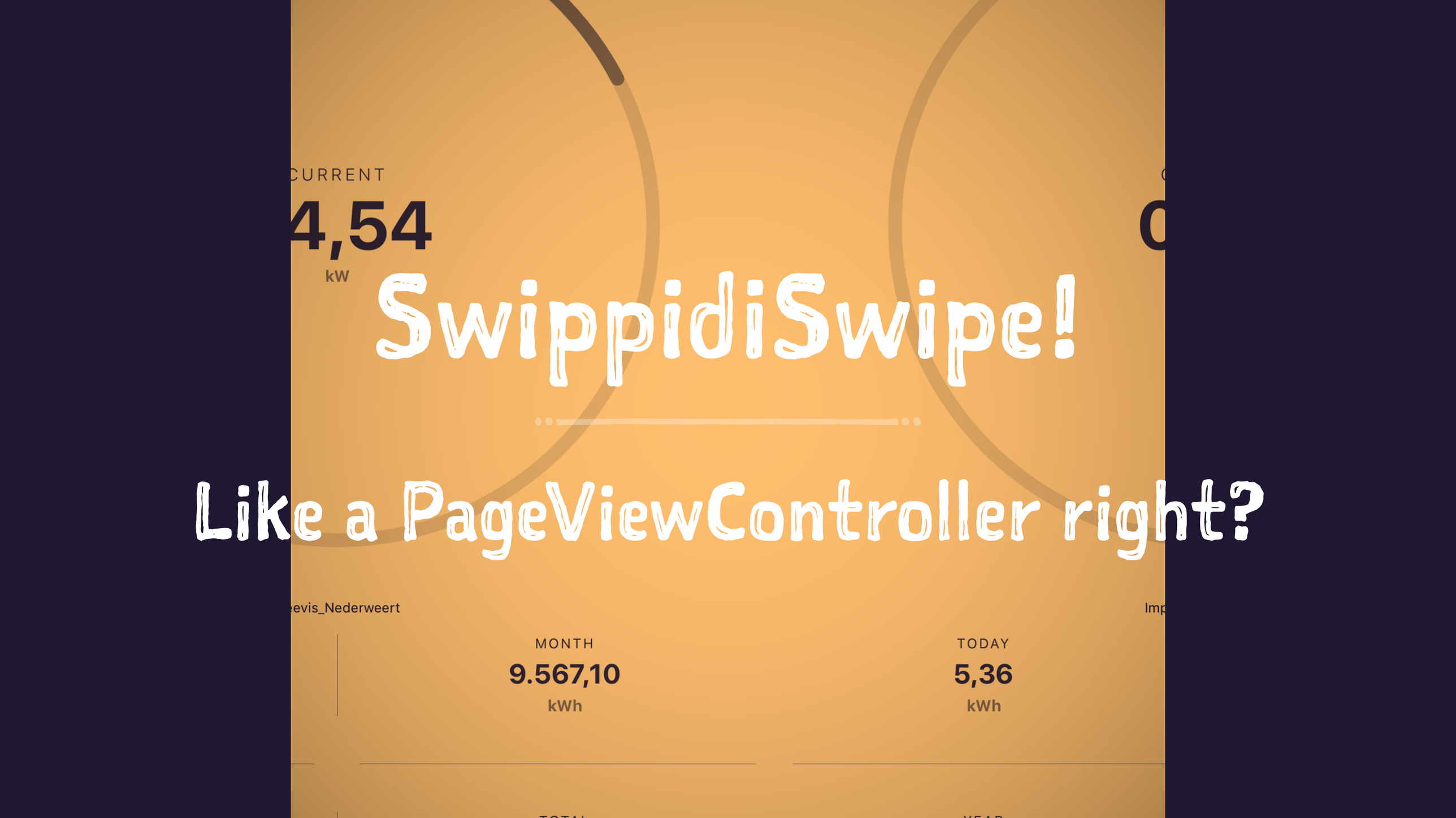
Animations with SwiftUI

```
struct ContentView: View {
    @State private var showingWelcome = false

    var body: some View {
        VStack {
            Toggle(isOn: $showingWelcome.animation(.spring())) {
                Text("Toggle label")
            }

            if showingWelcome {
                Text("Hello World")
            }
        }
    }
}
```

Animations in SolarDashboard

```
HStack(alignment: .center, spacing: self.tilePadding)  {
    /// building items into HStack
    ForEach(0..<self.items.count) { index in
            self.items[index]
                .offset(x: self.currentScrollOffset, y: 0)
                .frame(width: self.tileWidth)
    }
}
.onAppear {
    self.currentScrollOffset = self.offsetForPageIndex(self.activePageIndex)
}
.offset(x: self.stackOffset, y: 0)
.background(Color.black.opacity(0.00001)) // hack
.frame(width: self.contentWidth)
.simultaneousGesture(DragGesture(minimumDistance: 1, coordinateSpace: .local)
    .onChanged { value in
        self.dragOffset = value.translation.width
        self.currentScrollOffset = self.computeCurrentScrollOffset()
    }
    .onEnded { value in
        (...)
    }
)
```

```
HStack(alignment: .center, spacing: self.tilePadding)  {
    /// (building items into HStack)
}
(...)
.simultaneousGesture(DragGesture(minimumDistance: 1, coordinateSpace: .local)
    .onChanged { value in
        self.dragOffset = value.translation.width
        self.currentScrollOffset = self.computeCurrentScrollOffset()
    }
    .onEnded { value in
        // compute nearest index
        let velocityDiff = (value.predictedEndTranslation.width - self.dragOffset) * self.scrollDampingFactor
        let newPageIndex = self.indexPageForOffset(self.currentScrollOffset + velocityDiff)
        self.dragOffset = 0
        withAnimation(.interpolatingSpring(mass: 0.1, stiffness: 15, damping: 1.7, initialVelocity: 0)){
            self.activePageIndex = newPageIndex
            self.currentScrollOffset = self.computeCurrentScrollOffset()
        }
    }
)
```


CURRENT
4,54
kW
SwippidiSwipe!
Like a PageViewController right?
eevis_Nederweert
MONTH
9.567,10
kWh
TODAY
5,36
kWh

Live Coding

Animate ArcShape!

GeometryReader

```
struct DashboardView : View {

var body: some View {
    GeometryReader { geometry in
        PagingScrollView(activePageIndex: self.$activePageIndex,
                         itemCount: self.environment.powerValues.count,
                         pageWidth: geometry.size.width,
                         tileWidth: geometry.size.width,
                         tilePadding: 0) {
            // Show values
        }
    }
}
```

```
extension GeometryProxy {
    var center: CGPoint { CGPoint(x: size.height / 2, y: size.height / 2) }
    var radius: CGFloat { size.width / 2 }
}
```

SwiftUI Defaults

Tonsky's Blog Post

- War on Commas / Tupples
- Automatic Padding

```
static func buildBlock() -> EmptyView

static func buildBlock<Content>(Content) -> Content

static func buildBlock<C0, C1>(C0, C1)
            -> TupleView<(C0, C1)>
(...)

static func buildBlock
<C0, C1, C2, C3, C4, C5, C6, C7, C8, C9>
(C0, C1, C2, C3, C4, C5, C6, C7, C8, C9)
-> TupleView<(C0, C1, C2, C3, C4, C5, C6, C7, C8, C9)>
```

Automatic/Adaptive Padding

```
Text("abc").padding()

HStack {
    Text("★★★★★")
    Text("Avocado Toast).font(.title)
    ...
}
```

→ I think is smarter than me.

2S's T&T

Pauls' Blog Post

- Option-Cmd-P: Reload Live Preview
- Make @State private
- Make `print()` work
- Cmd-click on nav link -> Extract Subview
- Showing multiple alerts in a view

Prototype with constant bindings

```
TextField("Example placeholder", text: .constant("Hello"))
    .textFieldStyle(RoundedBorderTextFieldStyle())

Slider(value: .constant(0.5))
```

Semantic (Adaptive) colors

- `Color.red != RGB(255, 0, 0)`
- `Color.primary, Color.secondary, Color.accentColor, …`
- Environment

Combine text views

```
struct ContentView: View {
    var body: some View {
        Text("Colored ")
            .foregroundColor(.red)
        +
        Text("SwifUI ")
            .foregroundColor(.green)
        +
        Text("Text")
            .foregroundColor(.blue)
    }
}
```

Don't forget SolarDashboard

It's on the AppStoreZ (Mac & iOS)

Don't forget my book on Privacy

Apress.com / Books

Karma-based API on Platforms