This is a useful article about swiftUI One of the most interesting aspects of SwiftUI, at least from an architectural perspective, is how it essentially treats views as data. After all, a SwiftUI view isn't a direct representation of the pixels that are being rendered on the screen, but rather a description of how a given piece of UI should work, look, and behave.

That very data-driven approach gives us a ton of flexibility when it comes to how we structure our view code — to the point where one might even start to question what the difference actually is between defining a piece of UI as a view type, versus implementing that same code as a modifier instead.

Take the following FeaturedLabel view as an example — it adds a leading star image to a given text, and also applies a specific foreground color and font to make that text stand out as being "featured":

struct FeaturedLabel: View { var text: String

var body: some View {
    HStack {
        Image(systemName: "star")
        Text(text)
    }
    .foregroundColor(.orange)
    .font(.headline)
}
} While the above may look like a typical custom view, the exact same rendered UI could just as easily be achieved using a "modifier-like" View protocol extension instead — like this:

extension View { func featured() -> some View { HStack { Image(systemName: "star") self } .foregroundColor(.orange) .font(.headline) } } Here's what those two different solutions look like side-by-side when placed within an example ContentView:

struct ContentView: View { var body: some View { VStack { // View-based version: FeaturedLabel(text: "Hello, world!")

        // Modifier-based version:
        Text("Hello, world!").featured()
    }
}
} One key difference between our two solutions, though, is that the latter can be applied to any view, while the former only enables us to create String-based featured labels. That's something that we could address, though, by turning our FeaturedLabel into a custom container view that accepts any kind of View-conforming content, rather than just a plain string:

Regarding the design of our application, I have created some rough sketches through swift by exploring the different interfaces as well as through canva. Our gropu has been working on the UI for the app on canva and we have figured out the wireframing. I will attatch the images on the document that I submit on. Above is a very interesting article that I have found regaridng Swift and how it can be very impactful in the wa we structure our code. I don't have anything regarding the code we do in class because I haven't really messed with it, but I have been thinking of ways that I can change the tictactoe app into candy crush like we have bee talking about. I maight implement these changes for my next journal. This is all that I have bee doing regarding my journey through learning swift.