



Imperativ war gestern

Mit deklarativen UI-Frameworks
in die Zukunft

Thomas Künneth, MATHEMA

<https://www.thomaskuenneth.eu/>

questions@thomaskuenneth.eu

 @tkuenneth

https://github.com/tkuenneth/imperative_vs_declarative_uis



(c) Thomas Künne

- Viele UI-Frameworks sind objektorientiert
- Zur Entwicklungszeit wird jede UI-Komponente durch eine Klasse repräsentiert
- Zur Laufzeit ist Oberfläche ein Objektgraph



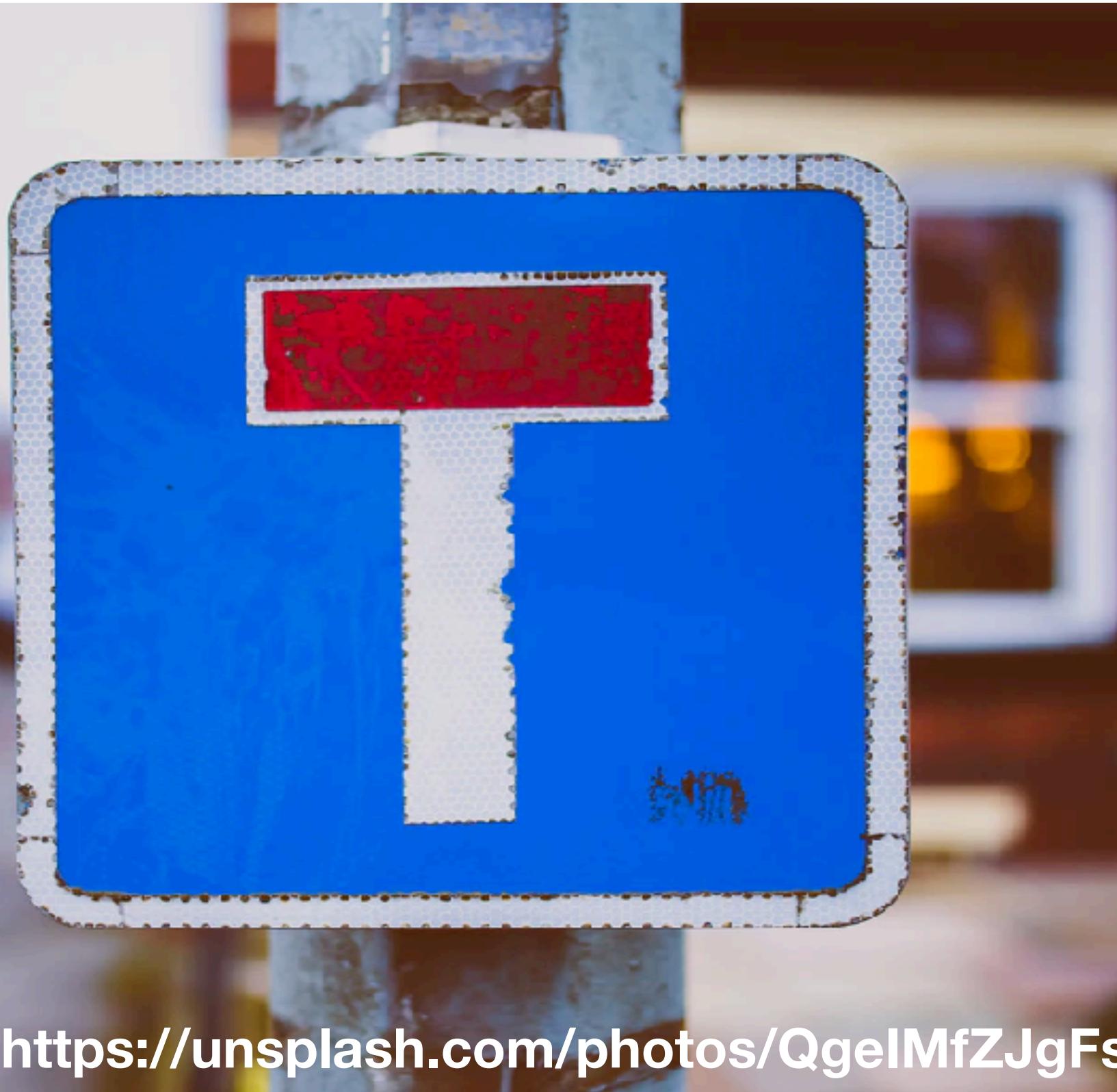
<https://unsplash.com/photos/E75ZuAlpCzo>

- Änderungen der Oberfläche durch Manipulation der Objekteigenschaften
- Für Interaktion werden Callbacks / Listener registriert
- Holen und ggf. Halten von Referenzen auf benötigte Objekte

- Spezialisierung durch Vererbung

```
java.lang.Object  
java.awt.Component  
java.awt.Container  
javax.swing.JComponent  
javax.swing.AbstractButton  
javax.swing.JButton
```

- Verhaltensänderungen durch Überschreiben
 - ✓ Praktisch bei kleinen Anpassungen
 - ! Frameworks mögen Ableitungen nicht immer



<https://unsplash.com/photos/QgelMfZJgFs>

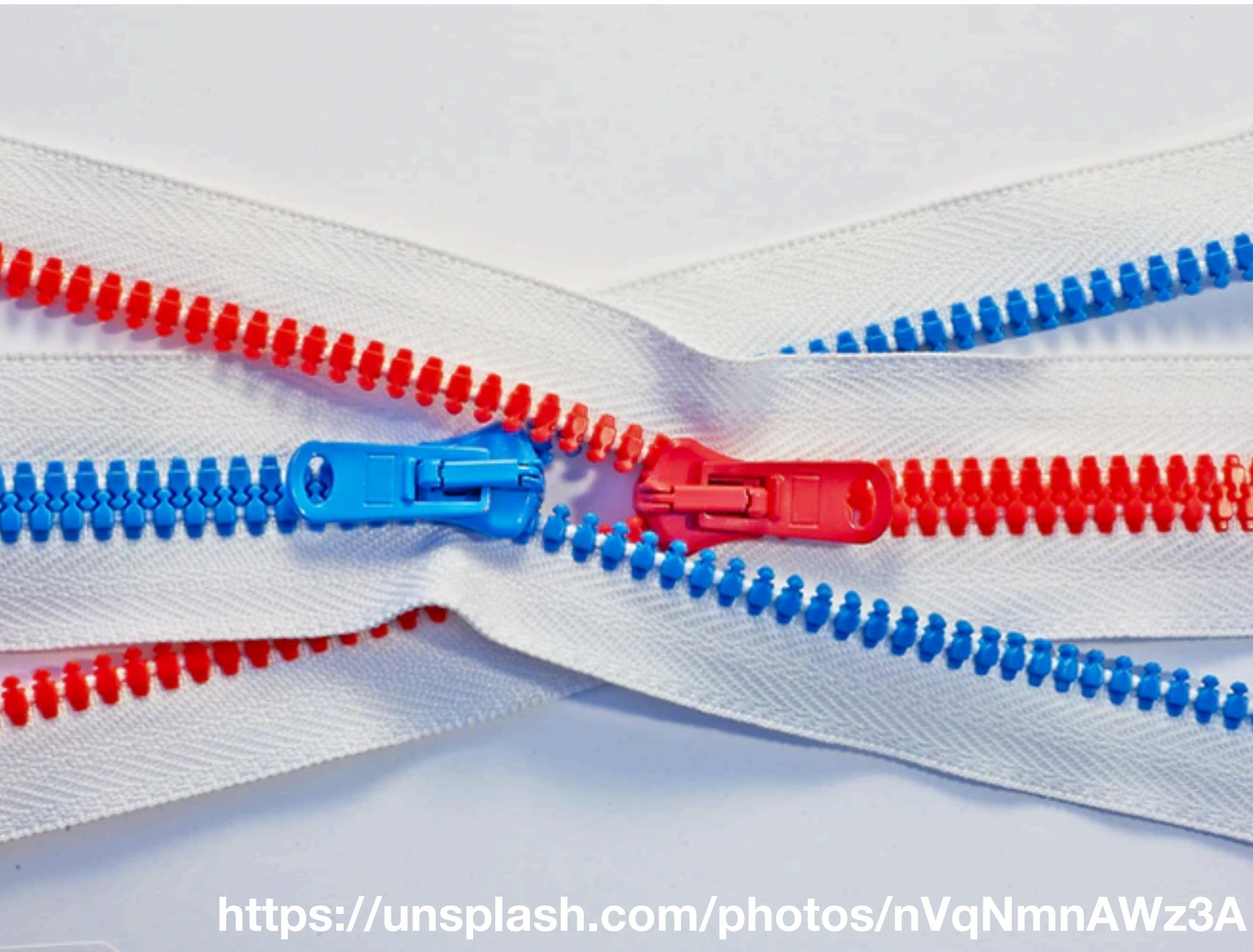


- Enge Kopplung von Daten und Logik
- Code zum Bau der Oberflächen bläht Quelltexte auf
- Zusammenhänge schon im Kleinen nicht leicht zu erfassen



(c) Thomas Künneth

- Trennung von Oberfläche und Code
- Verschlankt Quelltexte
- Macht Oberflächen leichter...
 - designbar
 - wiederverwendbar



<https://unsplash.com/photos/nVqNmNnAWz3A>

- Oberflächenbeschreibung in eigener Datei
- Wird zu Objektgeflecht oder Datenstrukturen entfaltet
- Referenzen auf Objekte oder Strukturen holen und halten
- Nach Bedarf Objekte oder Strukturen manipulieren



https://unsplash.com/photos/62H_swdrc4A

- Wurde von einigen sehr frühen grafischen Oberflächen so gehandhabt
- GEM (Graphics Environment Manager) von Digital Research
 - Oberflächenbeschreibung in Resourcendateien (.rsc)
 - Laden mit `rsrc_load()`
 - `rsrc_gaddr()` liefert Adresse im Speicher



https://unsplash.com/photos/62H_swdrC4A



- Wirkt elegant
- Haben viele moderne UI-Frameworks übernommen:
 - Android: XML
 - JavaFX: FXML
 - macOS/iOS: Storyboard
 - WPF, Xamarin: XAML



<https://unsplash.com/photos/466ENaLuhLY>



- Entwicklung zeigt:
Löst nur Vermischung von
Oberflächenbeschreibung und Code
- Um Code nachhaltig zu verbessern,
sind zusätzlich Entwurfsmuster nötig
- Je nach Framework:
MVP, MVVM, MVC, ...



<https://unsplash.com/photos/i--IN3cvEjg>

The screenshot shows a dual-pane code editor interface. The left pane displays a JavaScript file named `CounterDemoJavaScript.js`, and the right pane displays a Java file named `CounterDemoSwing.java`. Both files implement a counter logic.

JavaScript File (`CounterDemoJavaScript.js`):

```
function updateText(counter) {
    if (counter == 0) {
        document.getElementById("counter").innerHTML =
            "<h3>Noch nicht geklickt</h3>";
    } else {
        document.getElementById("counter").innerHTML =
            `<h1>${counter}</h1>`;
    }
}
```

Java File (`CounterDemoSwing.java`):

```
private void updateUI(JLabel label, int counter) {
    if (counter == 0) {
        label.setFont(font1);
        label.setText("Noch nicht geklickt");
    } else {
        label.setFont(font2);
        label.setText(Integer.toString(counter));
    }
}
```

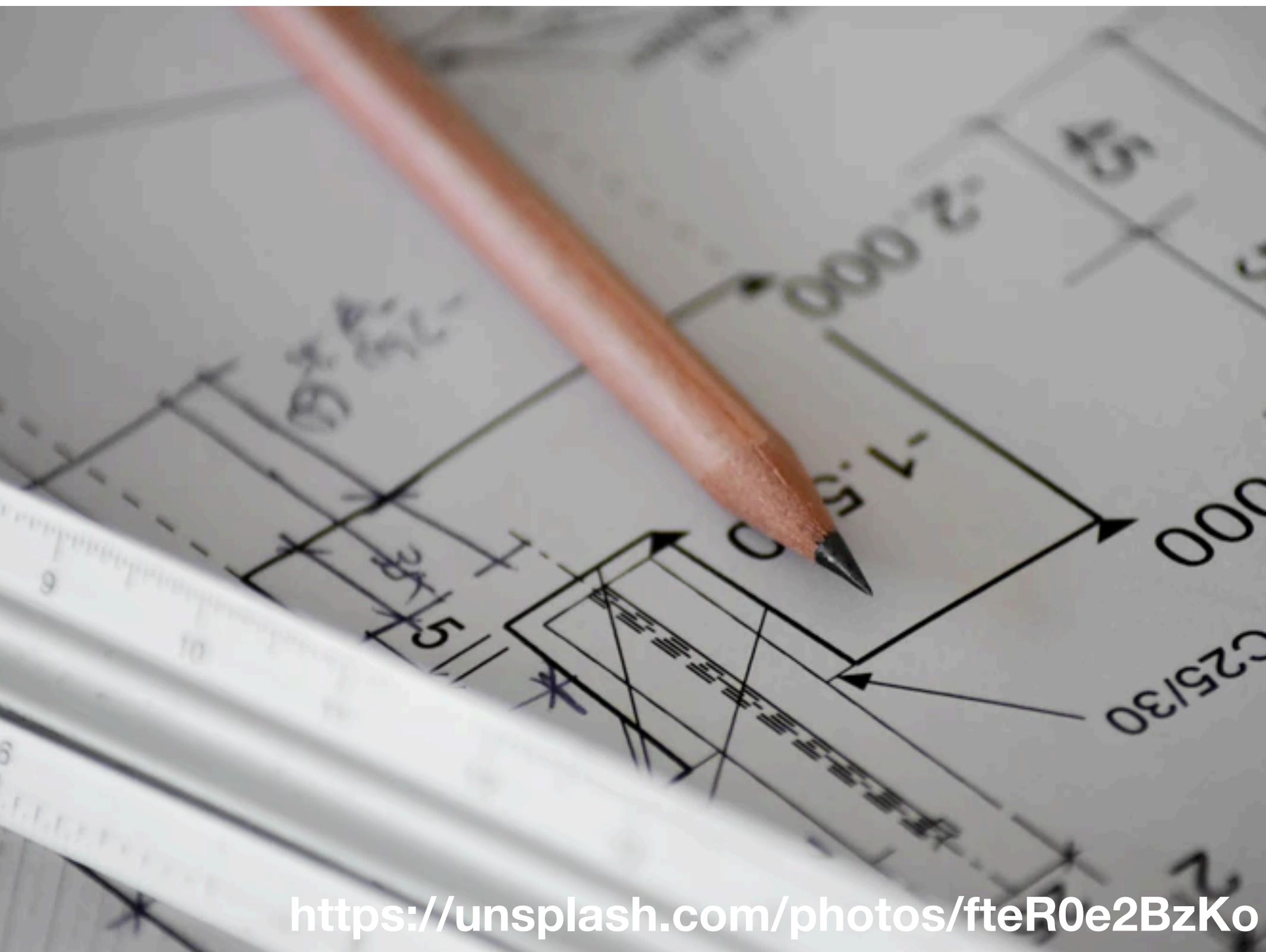
The Java code uses a different approach to update the UI compared to the JavaScript code. It uses a single method `updateUI` to handle both the initial state (when counter is 0) and the updated state (when counter is greater than 0). The UI is updated by setting the font and text of a `JLabel` component.

- Jede Änderung erfolgt durch Ändern von Objekt- bzw. Strukturattributen
- Entwickler muss Weg von Ist- zu Sollzustand kennen
- Und implementieren
- Je größer die Änderung, desto aufwendiger



(c) Thomas Künneth

- Entwickler vom Wissen um „Wegbeschreibung“ entlasten
- Also nicht mehr...
 - Wie muss ich Farbe von xyz ändern, damit...?
 - Welche Objekte muss ich hinzufügen oder löschen, damit...?



- Stattdessen:
 - Wie soll Oberfläche **jetzt** aussehen?
 - Welche Elemente sind zu sehen?
 - Wie sind diese beschaffen?
- Weg dorthin Sache des Frameworks
- Vorreiter Web: React mit Virtual DOM



(c) Thomas Künneth

- Flutter (2015)
- SwiftUI (2019)
- Jetpack Compose (2019)



(c) Thomas Künneth



- Keine nativen UI-Elemente
- Material Design und Cupertino Widgets



- Gesamte Oberfläche wird aus **Widgets** zusammengesetzt
- Bau der Oberfläche wird durch **Zustandsänderungen** ausgelöst
- Unterscheidung zwischen zustandslosen und -behafteten Widgets



<https://unsplash.com/photos/bu-6kNWQj6U>

- Widgets sind Klassen
- Um Aussehen zu ändern, nicht ableiten sondern kombinieren



- Zustandslose Widgets überschreiben
`build()`
- Zustandsbehaftete überschreiben
`createState()`

(c) Thomas Künne

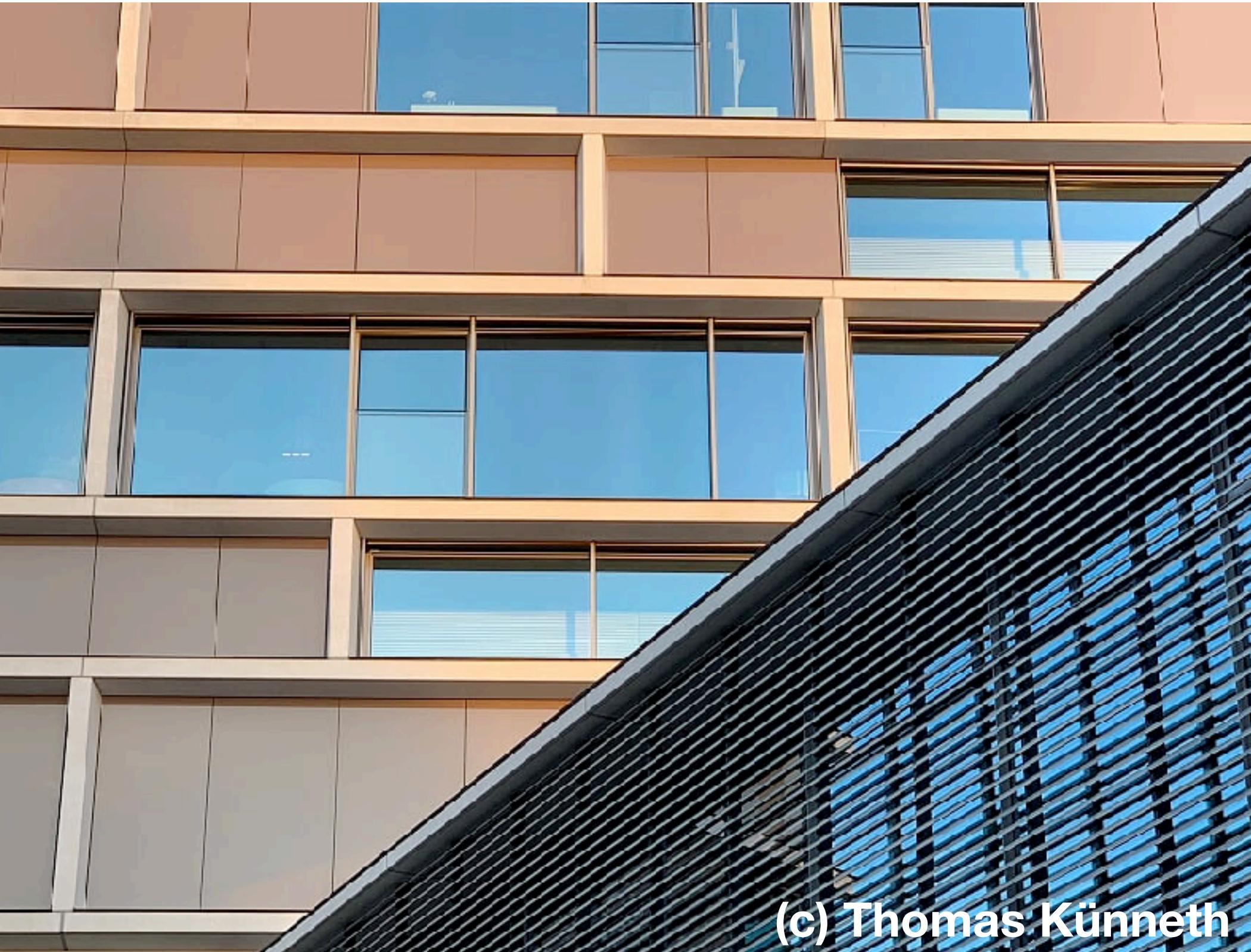
The screenshot shows a development environment for a Flutter application named 'counterdemoflutter'. The main.dart file is open in the editor, displaying the following code:

```
counterdemoflutter - main.dart
void main() => runApp(CounterDemo());
class CounterDemo extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: _title,
      home: CounterDemoHomePage(title: _title),
    ); // MaterialApp
}
class CounterDemoHomePage extends StatefulWidget {
  CounterDemoHomePage({Key key, this.title}) : super(key: key);
  final String title;
  @override
  _CounterDemoHomePageState createState() => _CounterDemoHomePageState();
}
class _CounterDemoHomePageState extends State<CounterDemoHomePage> {
  int _counter = 0;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ), // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text('You have pushed the button this many times:'),
            Text(_counter.toString(), style: TextStyle(fontSize: 24)),
            TextButton(onPressed: () => setState(() => _counter += 1), child: Text('Increment')),
          ],
        ),
      ),
    );
}

```

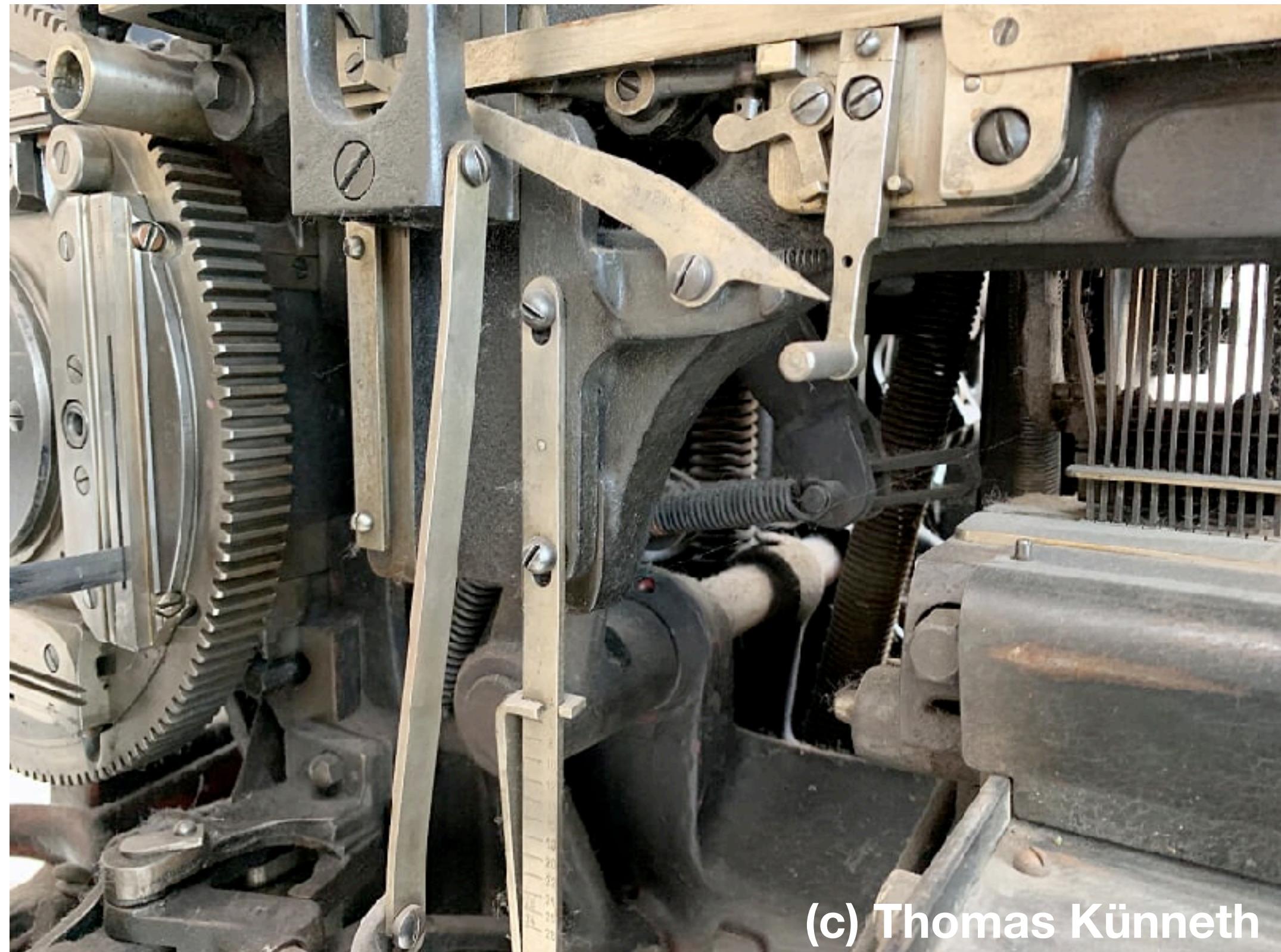
The code defines a main application structure with a CounterDemo class extending StatelessWidget. It uses a MaterialApp to wrap the CounterDemoHomePage. The CounterDemoHomePage is a StatefulWidget with a _CounterDemoHomePageState. The state contains an integer counter and a build method that returns a Scaffold with an AppBar and a Centered Column containing a Text for the counter value and a TextButton to increment it.

- Alles ist eine View
- Views bilden Hierarchien
- Grundlegende Layouts mit HStack, VStack und ZStack



(c) Thomas Künneth

- Views werden nicht direkt referenziert
- UI-Änderungen nach Zustandsänderungen
- Daten werden mit @State gekennzeichnet



(c) Thomas Künneth

- Modifier ersetzen Views (z.
B. `.font(.largeTitle)`)
- Ersetzungen sind auch auf Eltern-
Views möglich
- Kinder können Eltern „überschreiben“



(c) Thomas Künne

The screenshot shows the Xcode IDE with the project "CounterDemoSwiftUI" open. The file "ContentView.swift" is selected in the left sidebar, which displays the following Swift code:

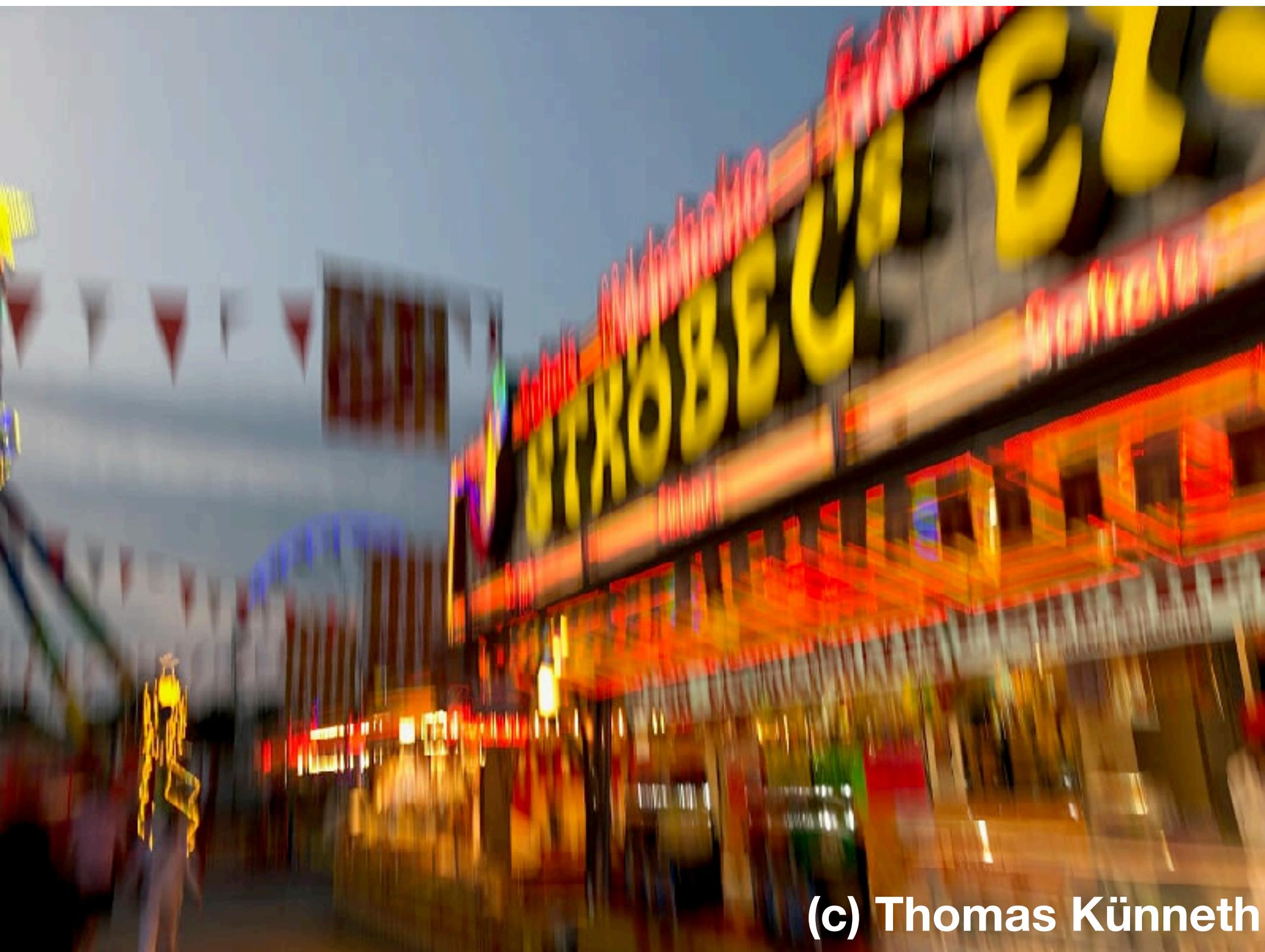
```
1 import SwiftUI
2
3 struct ContentView: View {
4     @State var counter = 0
5
6     var body: some View {
7         VStack {
8             if (counter == 0) {
9                 Text("Noch nicht geklickt")
10                .font(Font.system(size: 14))
11                .italic()
12                .addFrame(withHeight: 100)
13            } else {
14                Text("\(counter)")
15                .font(Font.system(size: 72))
16                .bold()
17                .addFrame(withHeight: 100)
18            }
19            Button("Klick") {
20                self.counter += 1
21            }
22        }
23    }
24}
25
26
27 extension Text {
28     func addFrame(withHeight height: CGFloat) ->
29         some View {
30         self.frame(height: height).fixedSize()
31     }
32 }
33
34 struct ContentView_Previews: PreviewProvider {
35     static var previews: some View {
36         ContentView()
37     }
38 }
```

The right side of the interface shows a preview of the application running on a "Phone SE (2nd generation)" device. The screen displays the text "Noch nicht geklickt" in a smaller, italicized font at the top, and a large, bold "Klick" button below it.



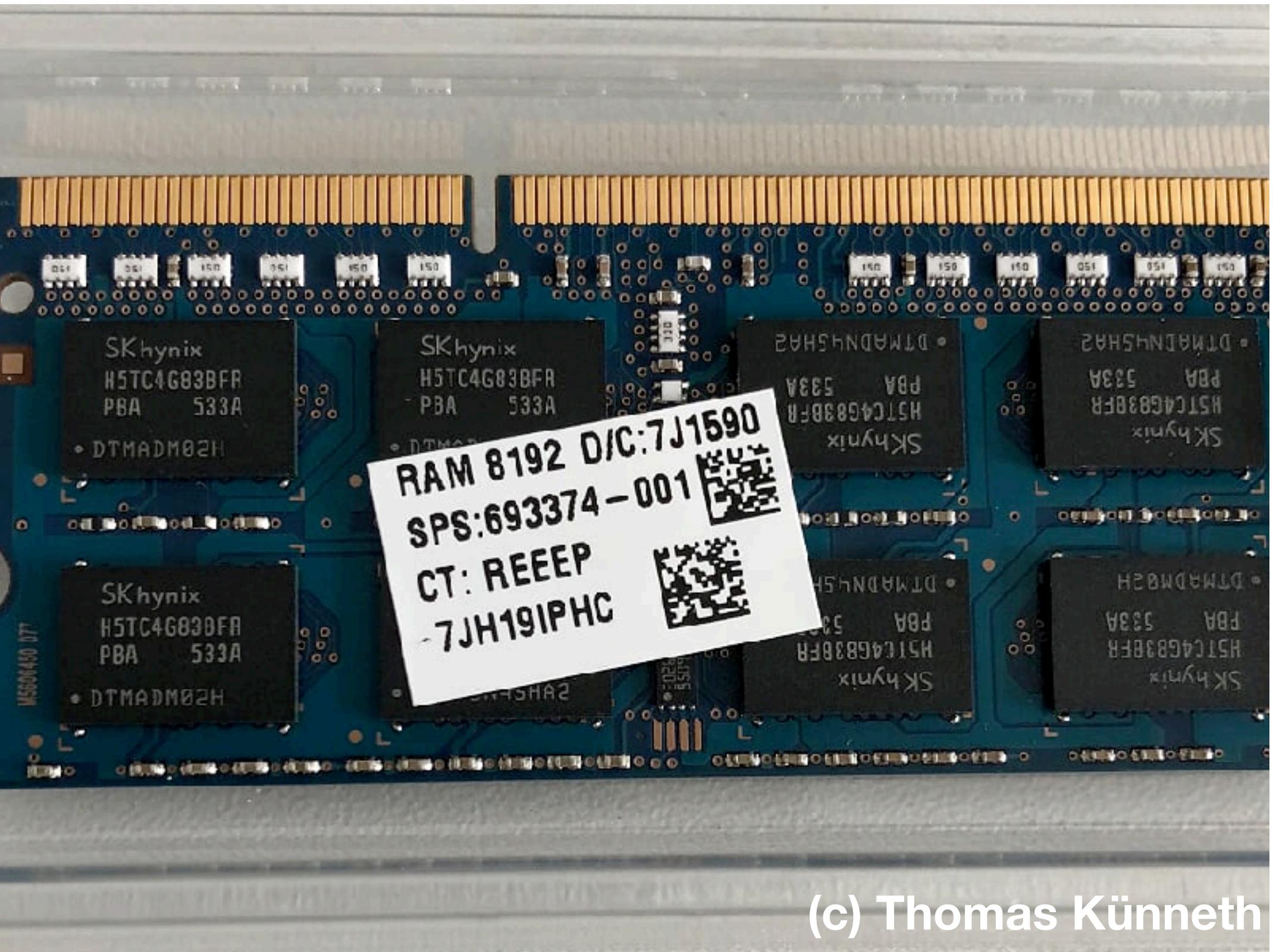
(c) Thomas Künne

- Basiert auf Material Design
- Hierarchien
- Einfache Layouts mit Row und Column



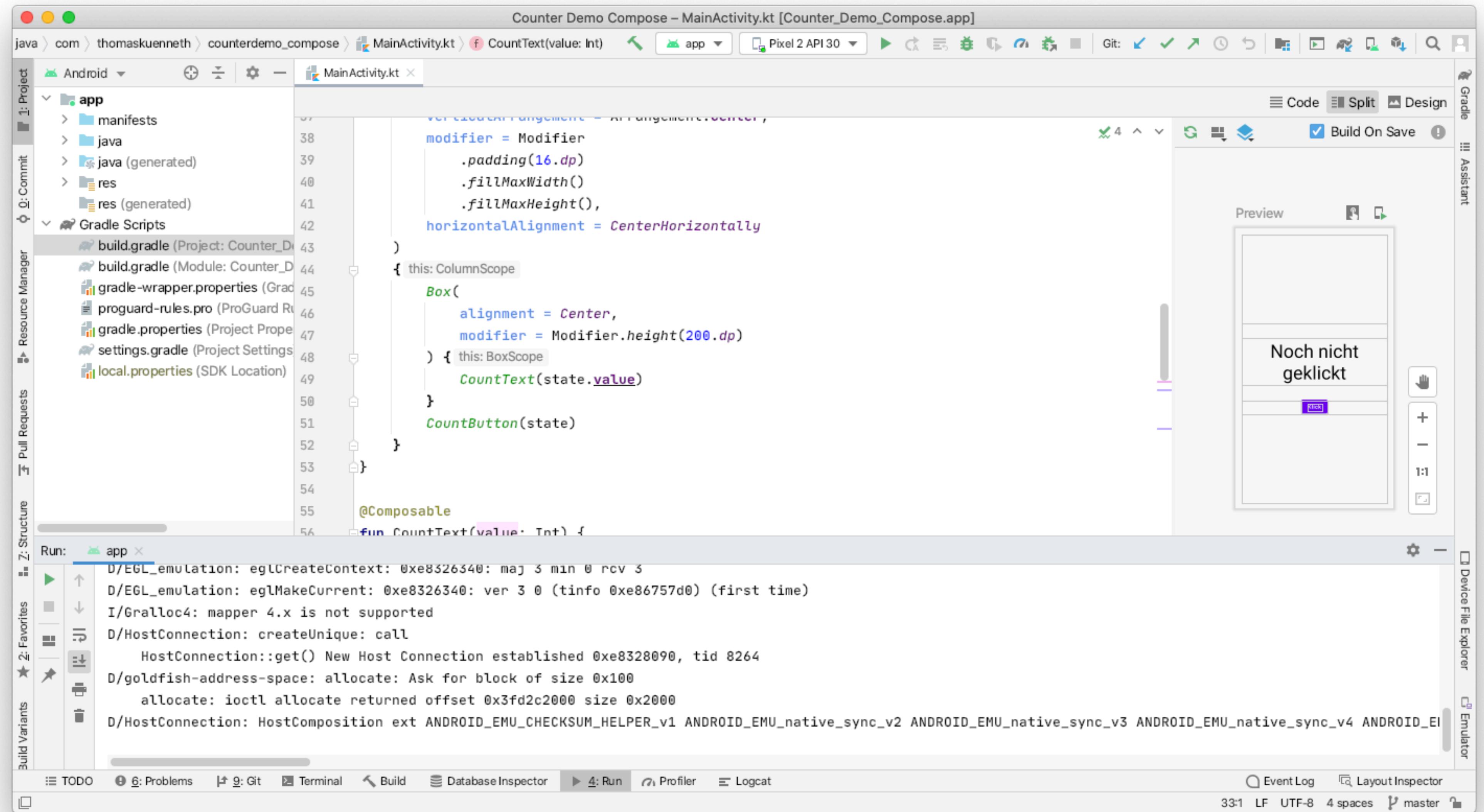
(c) Thomas Künne

- Grundbausteine sind **composable functions**
- Werden mit @Compose annotiert



(c) Thomas Künne

- Zustand mit remember { } merken
- Änderungen führen zum Aufruf des Composables

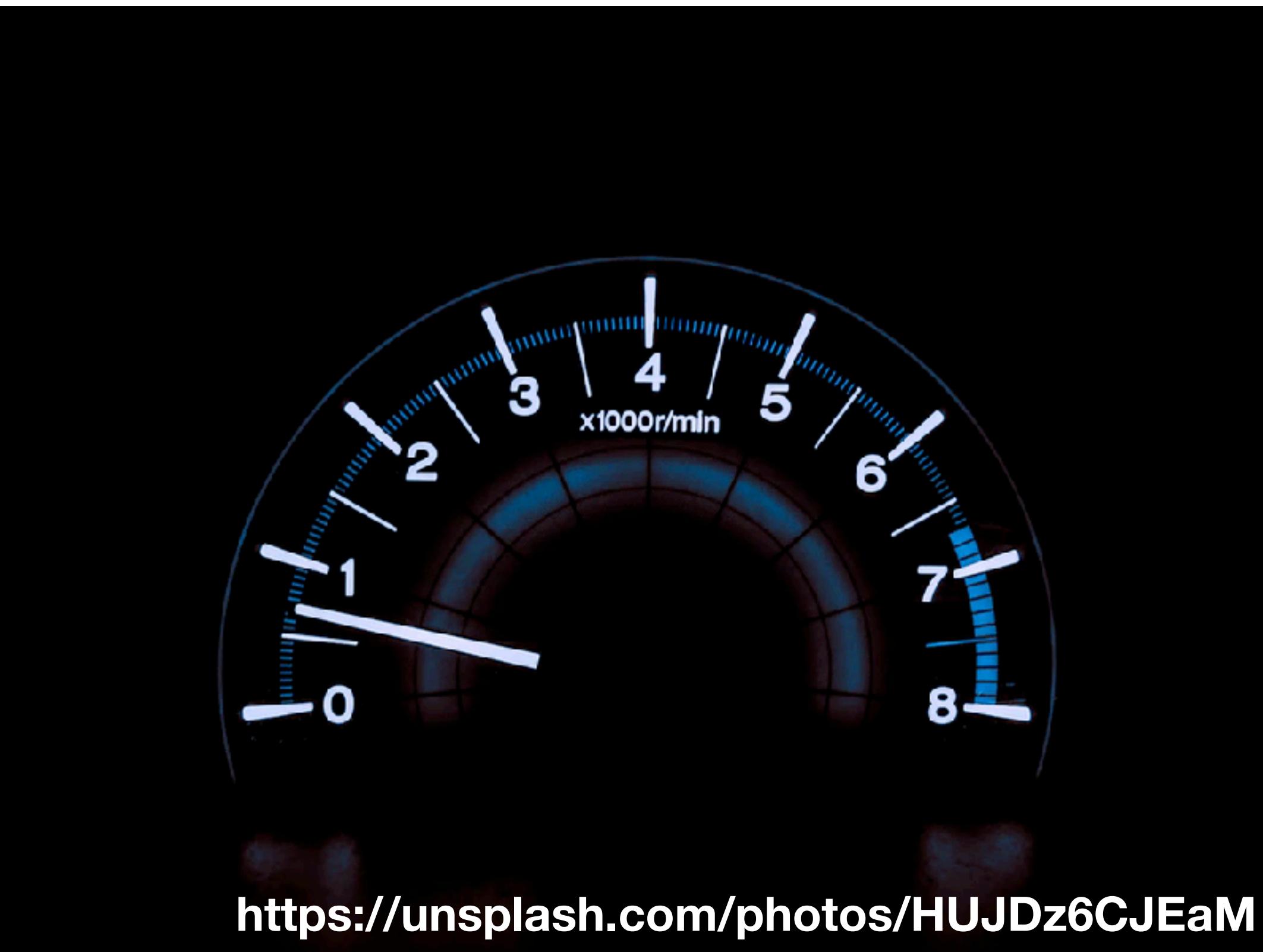


- Frameworks entwickeln sich noch recht schnell weiter
- Oberflächengestaltung wieder mehr „Entwicklersache“
- Weniger „Feintuning“ zentrale Idee hinter deklarativen UIs



https://unsplash.com/photos/ij5_qCBpIVY

- Schnelle Entwicklungszyklen
 - Schlanker Code
 - Vorschau in IDE oder Hot Reload
 - Statische Codeanalyse auch für Oberflächen



<https://unsplash.com/photos/HUJDz6CJEaM>

- Entwickler müssen umdenken
- Vermeiden von Fehlern durch Reduzieren von Boilerplate-Code
- Nach kurzer Eingewöhnung hohe Produktivität



<https://unsplash.com/photos/j06gLuKK0GM>



Vielen Dank

<https://www.thomaskuenneth.eu/>
questions@thomaskuenneth.eu

 @tkuenneth