twitch.tv/dancojocar

youtube.com/dancojocar

Happy New Year!

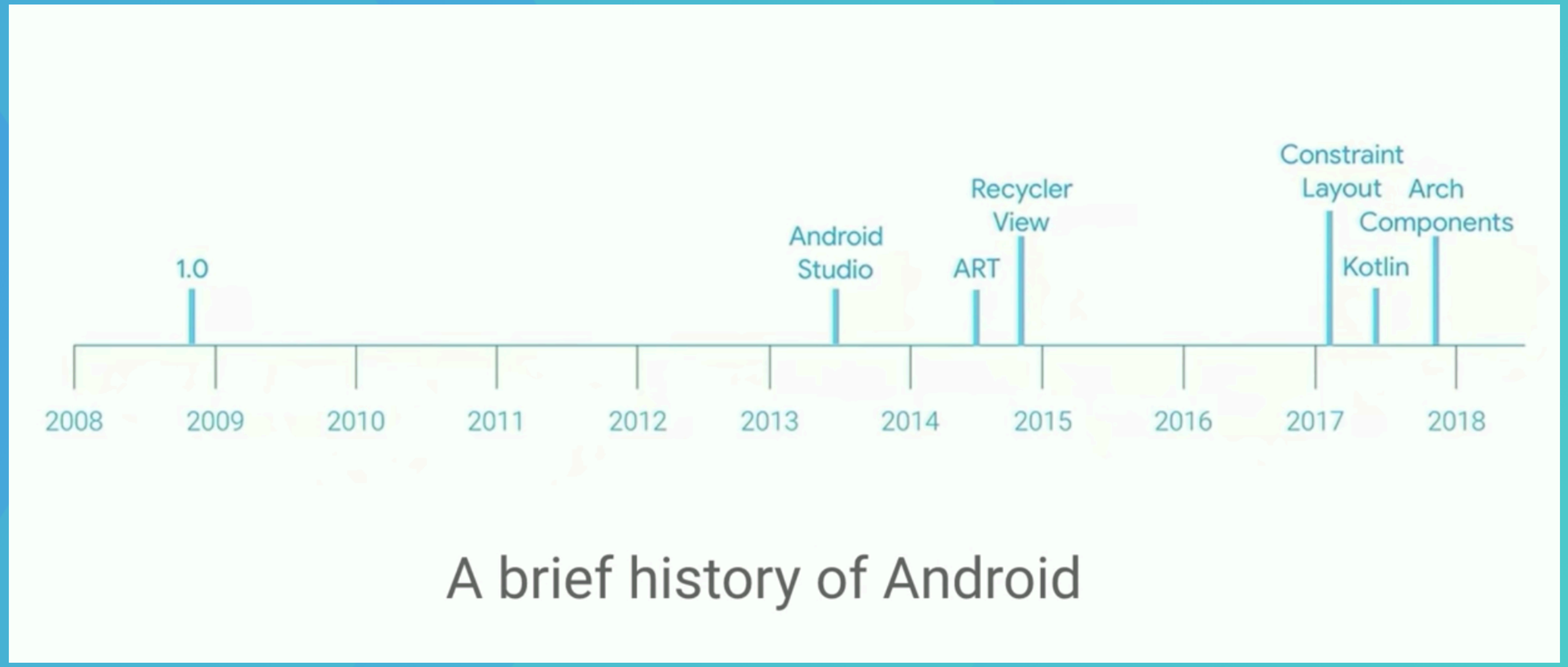
Lecture #11
Jetpack Compose & SwiftUI

2021

Mobile Applications 2020-2021

twitch.tv/dancojocar

youtube.com/dancojocar



A brief history of Android

twitch.tv/dancojocar

youtube.com/dancojocar

HTC Sooner

MAP 850 processor

64 MB of RAM

B of on-board memory

A “generous”
320 x 240 pixel resolution

NO touch input

twitch.tv/dancojocar

youtube.com/dancojocar

T-Mobil G1

A “generous”
320 x 240 pixel resolution

Qualcomm MSM7201A

192MB RAM, 256MB

TFT capacitive touchscreen

320 x 480 pixels

twitch.tv/dancojocar

youtube.com/dancojocar

Pixel 5

Qualcomm SM7250 Snapdragon 765G (7 nm)

twitch.tv/dancojocar

youtube.com/dancojocar

Architectural Components approach to UI

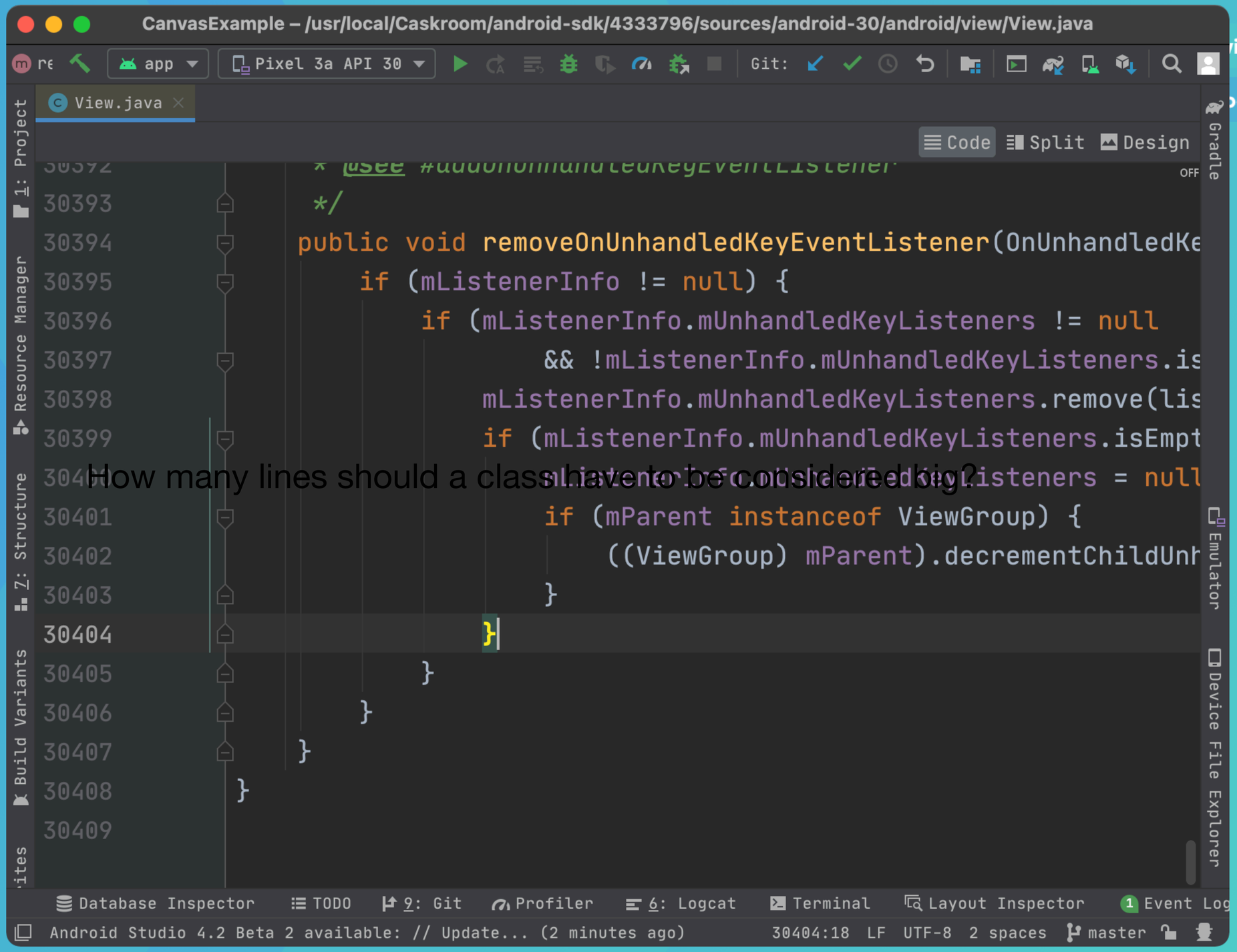
- What are developers asking for?
- What should the framework team do to help developers?
- How to make things easier?

Among Top Answers

twitch.tv/dancojocar

youtube.com/dancojocar

- Unbundle the UI toolkit!
- aka move android.widget to Jetpack.
- … what else can be fixed along the way?



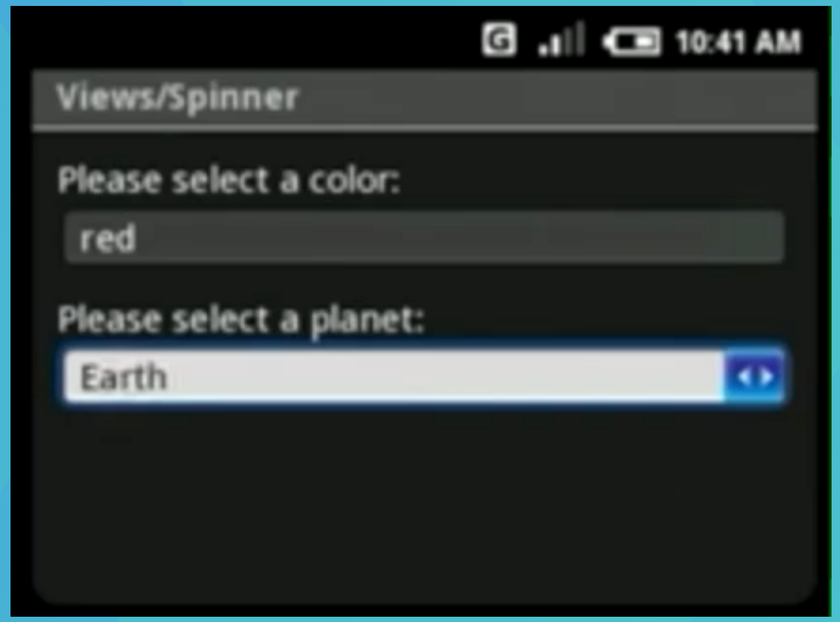

```
30393         */
30394        public void removeOnUnhandledKeyEventListener(OnUnhandledKe
30395            if (mListenerInfo != null) {
30396                if (mListenerInfo.mUnhandledKeyListeners != null
30397                        && !mListenerInfo.mUnhandledKeyListeners.is
30398                    mListenerInfo.mUnhandledKeyListeners.remove(lis
30399                    if (mListenerInfo.mUnhandledKeyListeners.isEmpt
30401                        if (mParent instanceof ViewGroup) {
30402                            ((ViewGroup) mParent).decrementChildUnh
30403                        }
30404                    }
30405                }
30406            }
30407        }
30408    }
30409
```

Room for Improvements

twitch.tv/dancojocar

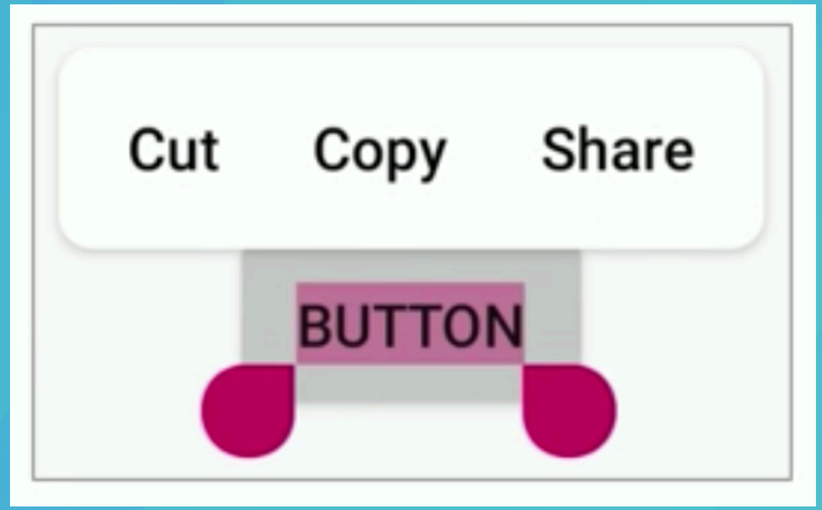
youtube.com/dancojocar

public class Spinner **extends** AbsSpinner **implements** OnClickListener



Room for Improvements

public class Button **extends** TextView



Too Much Code

twitch.tv/dancojocar

youtube.com/dancojocar

- My{Activity|Fragment}.{kt|java}
- main_{activity|fragment}.xml
- … stuff in res folder.
- Some more stuff in styles.xml

Goals

twitch.tv/dancojocar

youtube.com/dancojocar

- Unbundle from platform releases.
- Fewer choices when trying to build a UI element.
- Clarify state ownership and event handler.
- Write less code.

Can we go back to a simpler time?

twitch.tv/dancojocar

youtube.com/dancojocar

```
fun main(){
  print("Hello World!")
}
```

Can we go back to a simpler time?

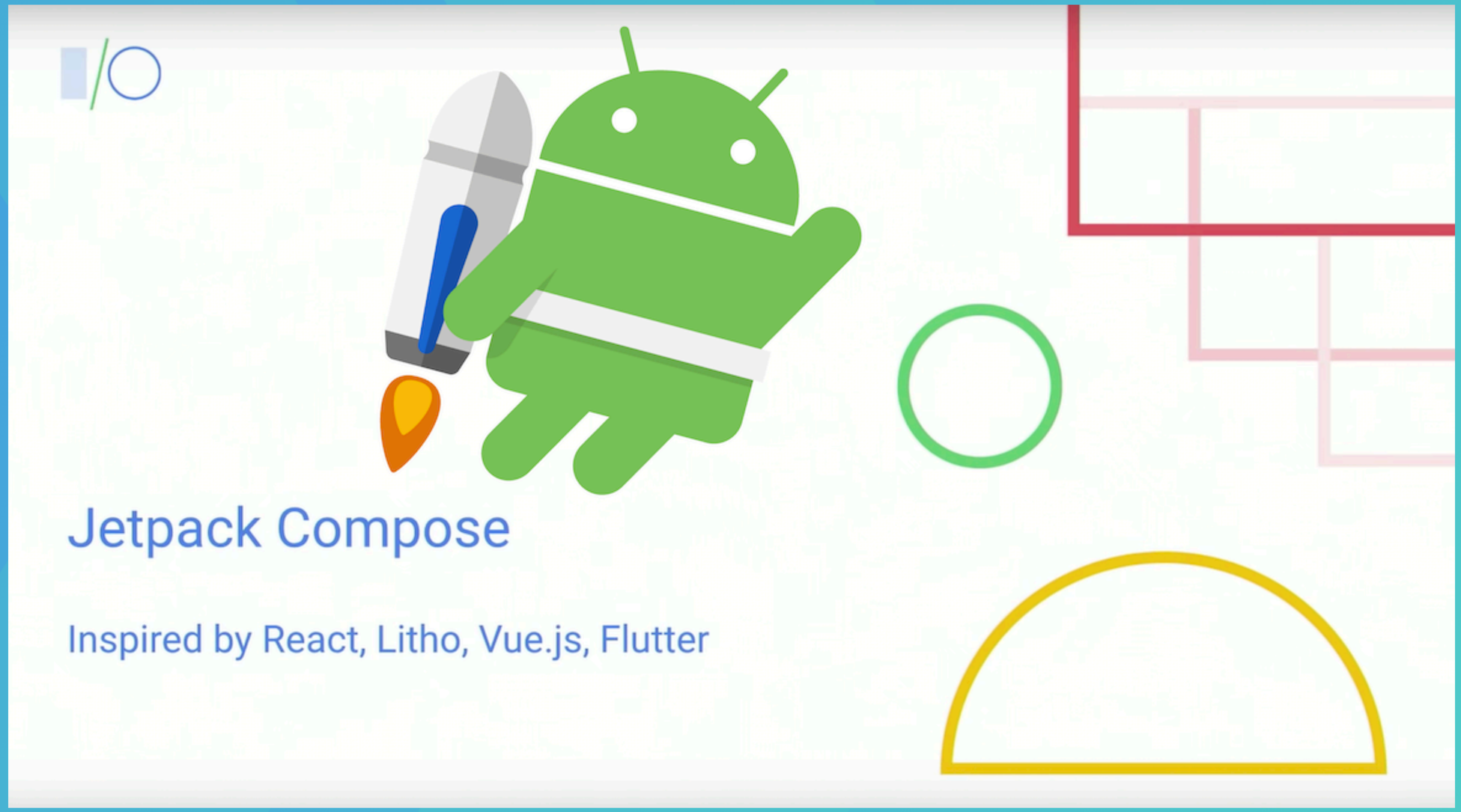
twitch.tv/dancojocar

youtube.com/dancojocar

```
fun Greeting(){
  println("Hello World!")
}
```

twitch.tv/dancojocar

youtube.com/dancojocar



developer.android.com/jetpack/compose

twitch.tv/dancojocar

youtube.com/dancojocar

developer.android.com/jetpack/compose

UI as a Function!

twitch.tv/dancojocar

youtube.com/dancojocar

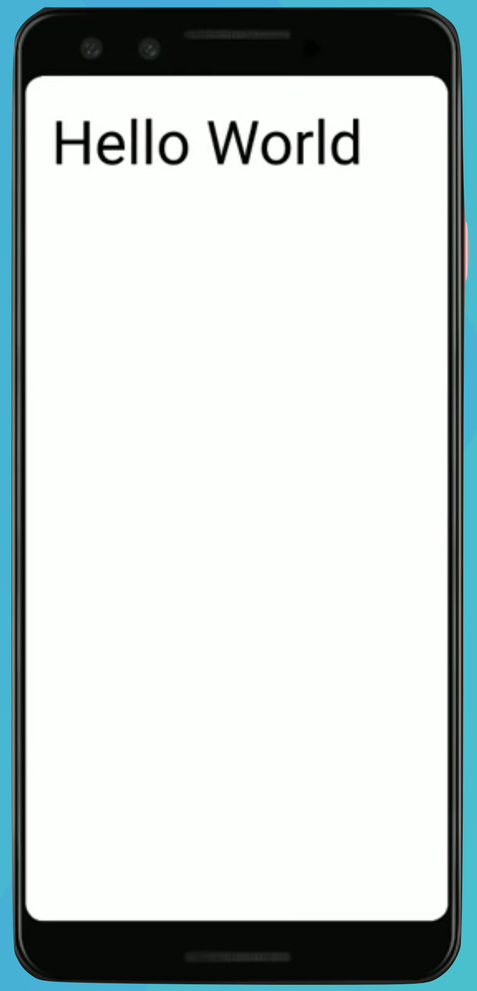
- Take data as input.
- Emit UI hierarchy when invoked.

```
@Composable
fun Greeting(name: String){
  Text("Hello $name")
}
```

UI as a Function!

twitch.tv/dancojocar

youtube.com/dancojocar




```
@Composable
fun Greeting(name: String){
  Text("Hello $name")
}
```

What is Compose?

twitch.tv/dancojocar

youtube.com/dancojocar

- A new set of Jetpack UI widgets.
  - Not widgets/fragments
  - Are much smaller => Functions
- A Kotlin compiler plugin.
- Fully compatible with the existing view system.
- Experimental, still in development.

Use in Existing App

```
class MainActivity : AppCompatActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContent {
      Greeting("World!")
    }
  }
}
```

twitch.tv/dancojocar

youtube.com/dancojocar

Composable UI Widget

```
@Composable
fun Greeting(name: String){
  Text("Hello $name")
}
```

Intercept & Rewrite

twitch.tv/dancojocar

youtube.com/dancojocar

```
@Composable
fun Greeting(name: String){
  for (i in 1..10) {
    Text("Hello $name")
  }
}
```

```
@Composable
fun Greeting(names: List<String>){
  for (name in names) {
    Text("Hello $name")
  }
}
```

```
@Composable
fun Greeting(names: List<String>){
  if (names.isEmpty()){
    Text("No soup for you!")
  } else {
    for (name in names) {
      Text("Hello $name")
    }
  }
}
```

Composable Building Blocks

twitch.tv/dancojocar

youtube.com/dancojocar

- Composable functions are defined in terms of other composable functions.

```
@Composable
fun NewsFeed(stories: List<Story>) {
  for (story in stories) {
    StoryWidget(story)
  }
}

@Composable
fun StoryWidget(story: Story) {
  Padding(8.dp){
    Column{
      Title(story.title)
      Image(story.image)
      Text(story.content)
    }
  }
}
```

twitch.tv/dancojocar

youtube.com/dancojocar

RecyclerView Adapter

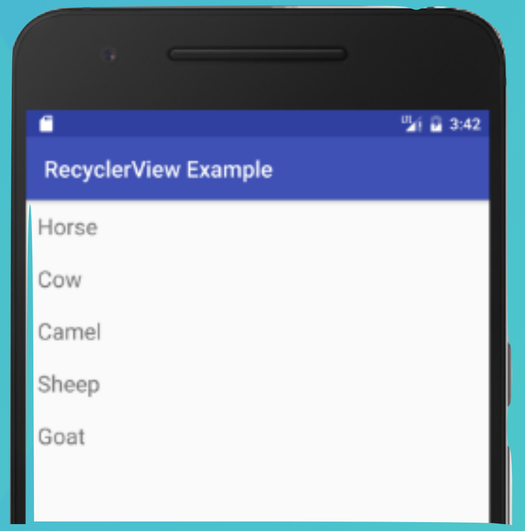
```
public class MyRecyclerViewAdapter extends RecyclerView.Adapter<MyRecyclerViewAdapter.ViewHolder> {

    private List<String> mData;
    private LayoutInflater mInflater;
    private ItemClickListener mClickListener;

    // data is passed into the constructor
    MyRecyclerViewAdapter(Context context, List<String> data) {
        this.mInflater = LayoutInflater.from(context);
        this.mData = data;
    }

    // inflates the row layout from xml when needed
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = mInflater.inflate(R.layout.recyclerview_row, parent, false);
        return new ViewHolder(view);
    }

    // binds the data to the TextView in each row
    @Override
```



twitch.tv/dancojocar

youtube.com/dancojocar

```
@Composable
fun NewsFeed(stories: List<Story>) {
  ScrollingList(stories) { story ->
    StoryWidget(story)
  }
}

@Composable
fun StoryWidget(story: Story) {
  Padding(8.dp){
    Column{
      Title(story.title)
      Image(story.image)
      Text(story.content)
    }
  }
}
```

```
@Composable
fun NewsFeed(stories: LiveDate<List<Story>>) {
  ScrollingList(stories.observer()) { story ->
    StoryWidget(story)
  }
}
```

twitch.tv/dancojocar

youtube.com/dancojocar

```
@Composable
fun StoryWidget(story: Story) {
  val image = asyncLoad(defaultPlaceholder) {
    loadImage(story.imageUri)
  }
  Card(cornerRadius = 4.dp, elevation = 4.dp) {
    Column {
      Image(image)
      Padding(16.dp) {
        Text(story.headline)
      }
    }
  }
}
```

twitch.tv/dancojocar

youtube.com/dancojocar

```
@Composable
fun StoryWidget(story: Story) {
  val image = asyncLoad(defaultPlaceholder) {
    loadImage(story.imageUri)
  }
  Card(cornerRadius = 4.dp, elevation = 4.dp) {
    Column {
      Image(image)
      Padding(16.dp) {
        Text(story.headline)
      }
    }
  }
}
```

```
data class Story(
    var imageUri: Uri,
    var headling: String
)
```

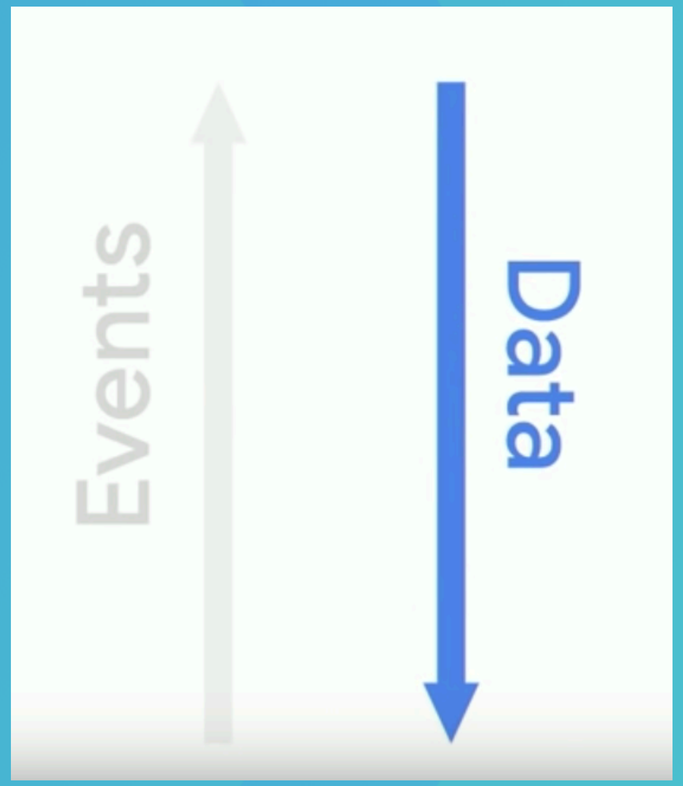
twitch.tv/dancojocar

youtube.com/dancojocar

```
@Composable
fun StoryWidget(story: Story) {
  val image = asyncLoad(defaultPlaceholder) {
    loadImage(story.imageUri)
  }
  Card(cornerRadius = 4.dp, elevation = 4.dp) {
    Column {
      Image(image)
      Padding(16.dp) {
        Text(story.headline)
      }
    }
  }
}
```

```
@Model
data class Story(
    var imageUri: Uri,
    var headling: String
)
```

Top-down Data Flow




```
@Composable
fun NewsFeed(stories: List<Story>) {
  for (story in stories) {
    StoryWidget(story)
  }
}

@Composable
fun StoryWidget(story: Story) {
  Padding(8.dp){
    Column{
      Title(story.title)
      Image(story.image)
      Text(story.content)
    }
  }
}
```

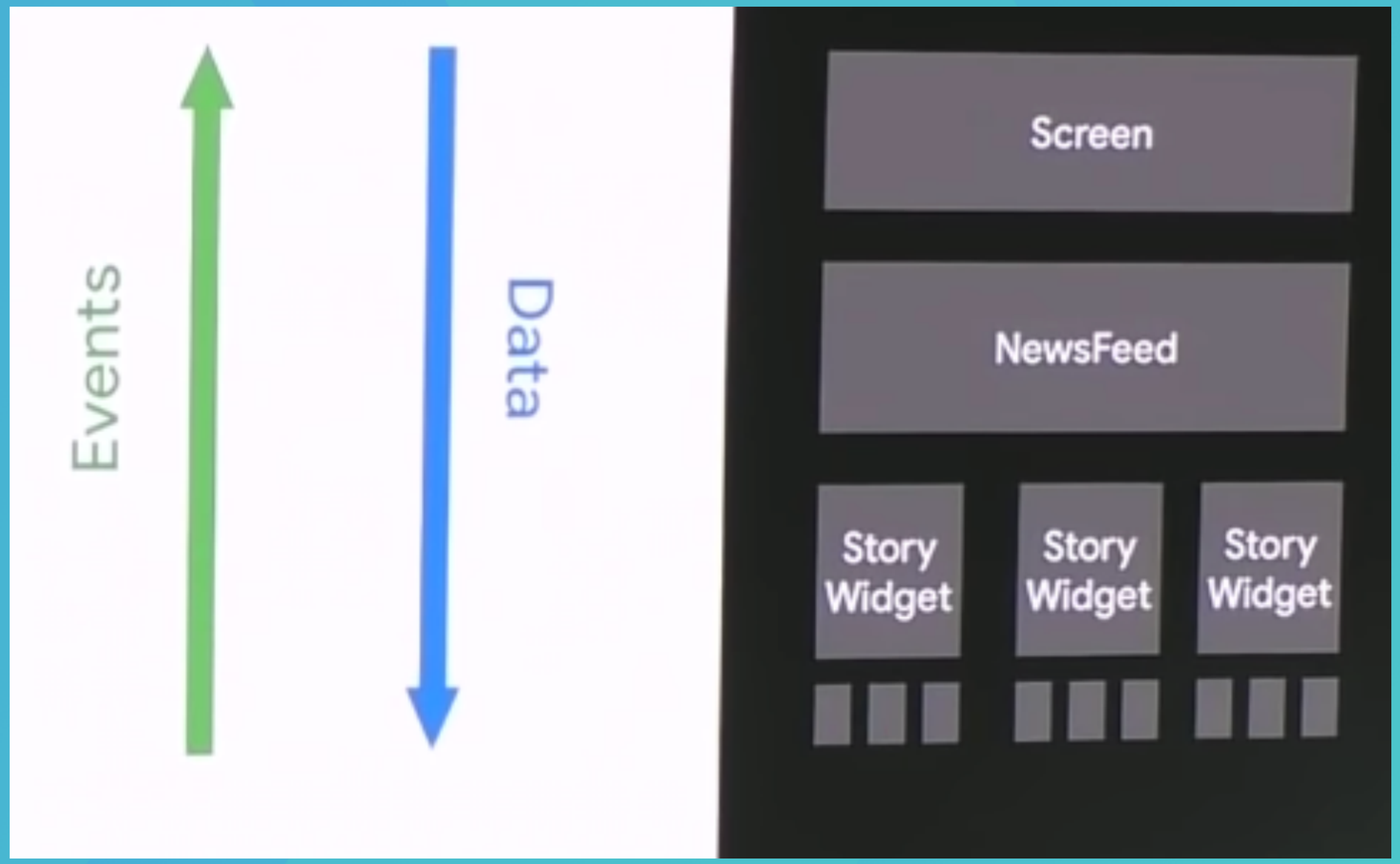
twitch.tv/dancojocar

youtube.com/dancojocar

```
@Composable
fun NewsFeed(stories: List<Story>,
             onSelected: (Story) -> Unit) {
  ScrolingList(stories) { story ->
    StoryWidget(story, onClick = { onSelected(story) })
  }
}

@Composable
fun StoryWidget(story: Story, onClick: () -> Unit) {
  Clickable(onClick) {
    Column {
      Padding(8.dp) {
        Title(story.headling)
        Image(story.image)
        Text(story.content)
      }
    }
  }
}
```

Single Source of Truth



Multiple Sources of Truth

- The app has a state.
- The checkbox has a state.
- The checkbox doesn’t ask the app before letting the user change its value.

~~Multiple~~ Sources ~~of~~ of Truth

- Let the app be the source.

```
@Composable
fun FoodPreferences(data: UserPreferences) {
  val options = listOf(MILK, EGGS, BREAD)
  Checkbox(
      options = options,
      selected = data.favoriteFood,
      onChange = { selected -> data.favoriteFood = selected }
  )
}
```

```
@Composable
fun NamePreferences(data: UserPreferrences) {
  EditableText(
      text = data.name,
      onChange = { newValue -> data.name = newValue }
  )
}
```

twitch.tv/dancojocar

youtube.com/dancojocar


```
@Composable
fun PhonePreferences(data: UserPreferrences) {
  EditableText(
      text = data.phoneNumber,
      onChange = { newValue ->
        val filtered = filterInvalidPhoneCharacters(newValue)
        val formatted = formatPhone(newValue)
        data.phoneNumber = formatted
      }
  )
}
```

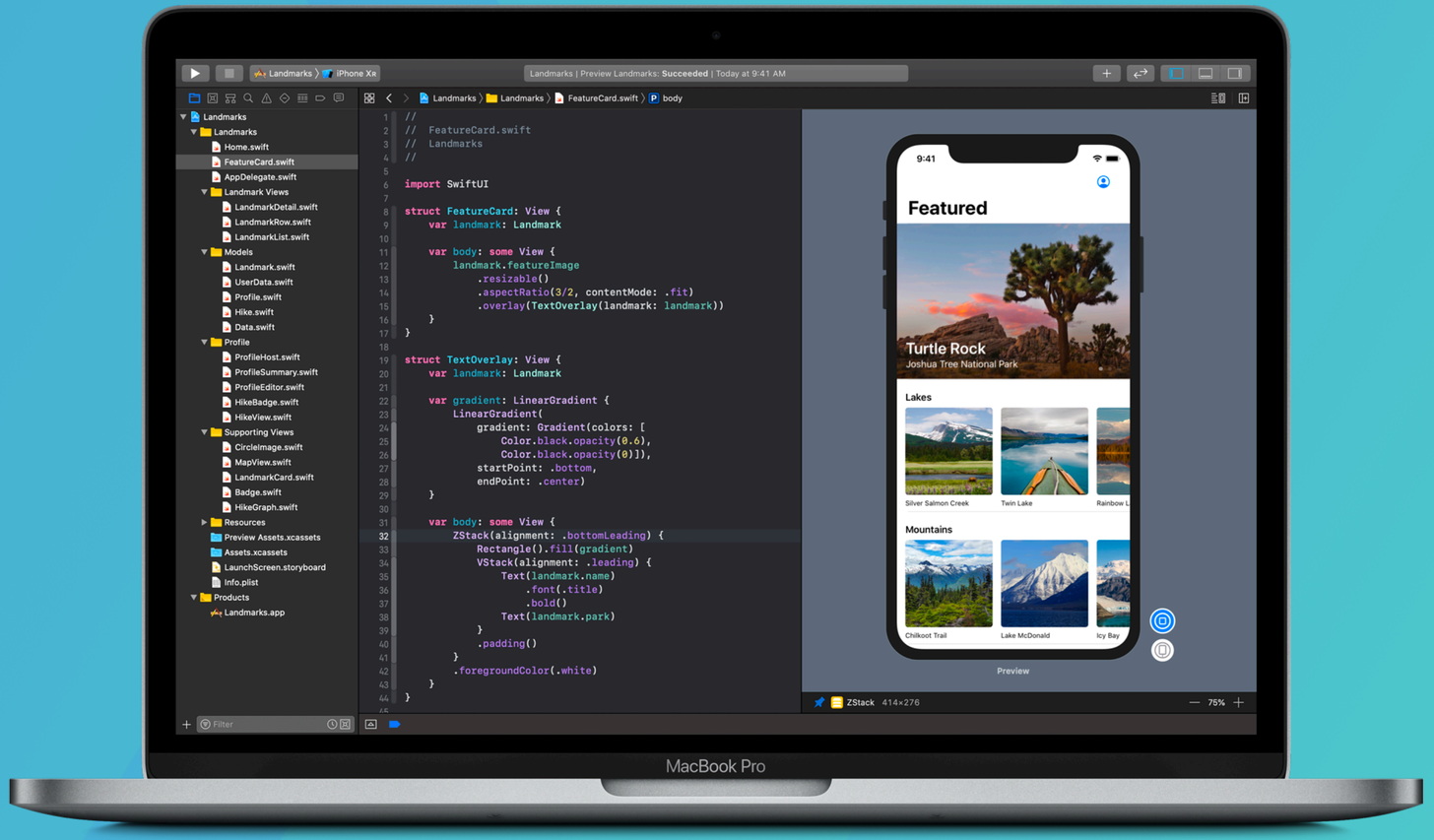
developer.android.com/jetpack/compose

twitch.tv/dancojocar

youtube.com/dancojocar

Introducing SwiftUI

Better apps. Less code.



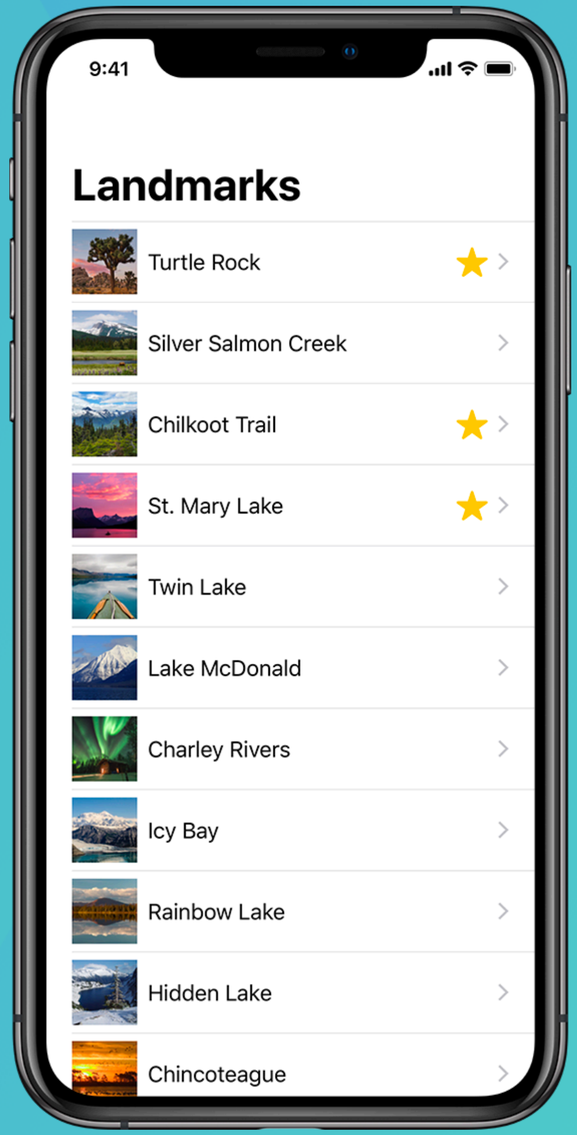
developer.apple.com/xcode/swiftui

Describe Layout Just Once

- Declare the content and layout for any state of your view. SwiftUI knows when that state changes, and updates your view's rendering to match.

```
List(landmarks) { landmark in
    HStack {
        Image(landmark.thumbnail)
        Text(landmark.name)
        Spacer()

        if landmark.isFavorite {
            Image(systemName: "star.fill")
                .foregroundColor(.yellow)
        }
    }
}
```



twitch.tv/dancojocar

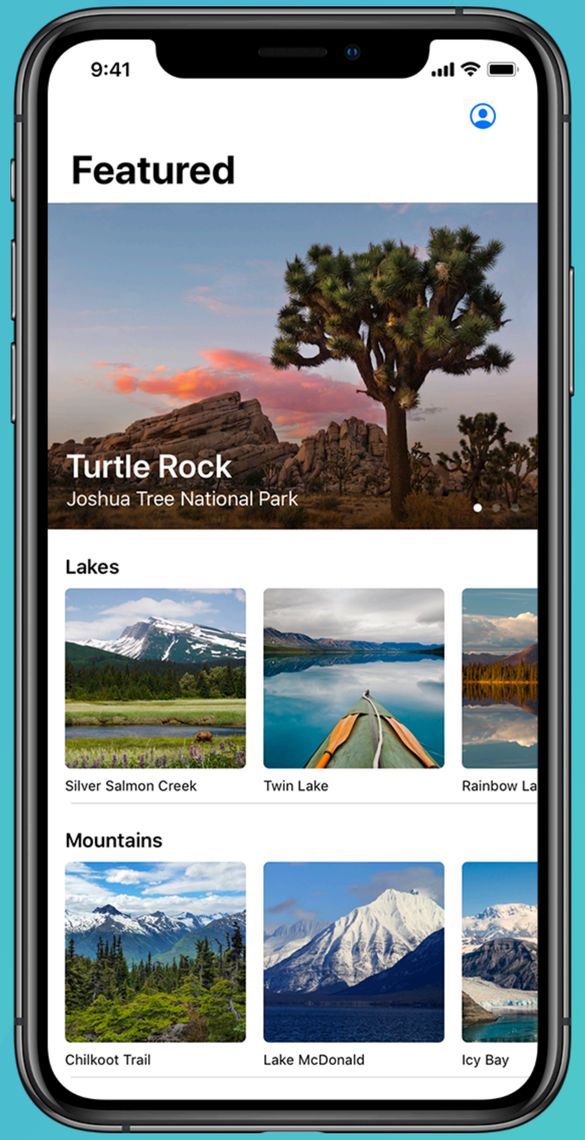
youtube.com/dancojocar

Build reusable components

- Combine small, single-responsibility views into larger, more complex interfaces. Share your custom views between apps designed for any Apple platform.

```
struct FeatureCard: View {
  var landmark: Landmark

  var body: some View {
    landmark.featureImage
      .resizable()
      .aspectRatio(3/2, contentMode: .fit)
      .overlay(TextOverlay(landmark))
  }
}
```



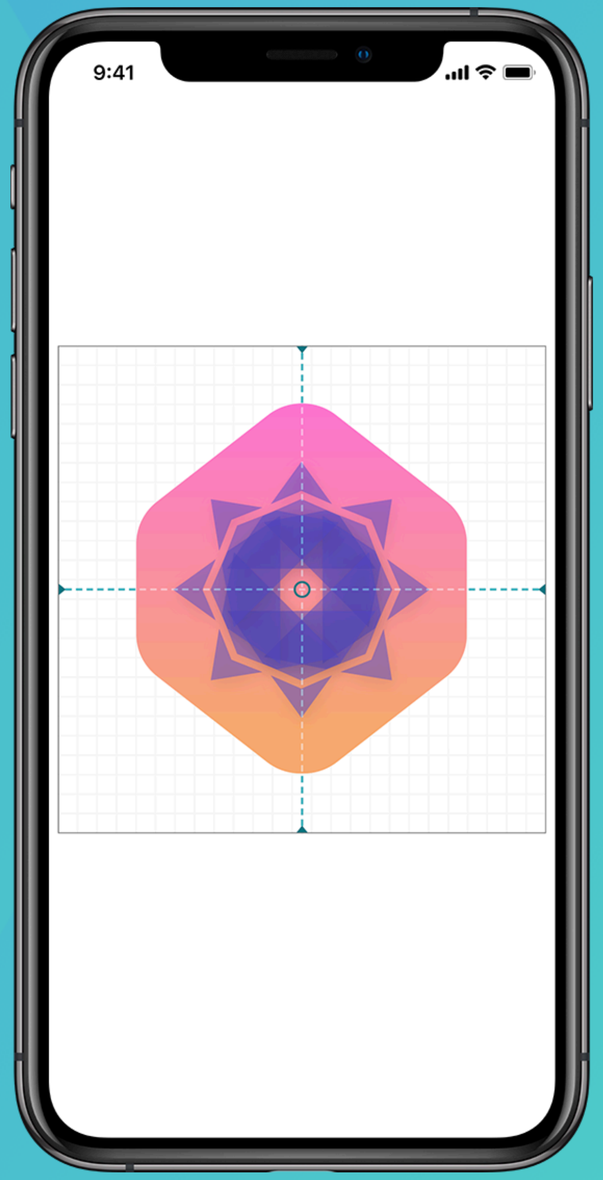
twitch.tv/dancojocar

youtube.com/dancojocar

Simplify Animations

- Creating smooth animations is as easy as adding a single method call. SwiftUI automatically calculates and animates transitions when needed.

```
VStack {
    Badge()
        .frame(width: 300, height: 300)
        .animation(.easeInOut())
    Text(name)
        .font(.title)
        .animation(.easeInOut())
}
```

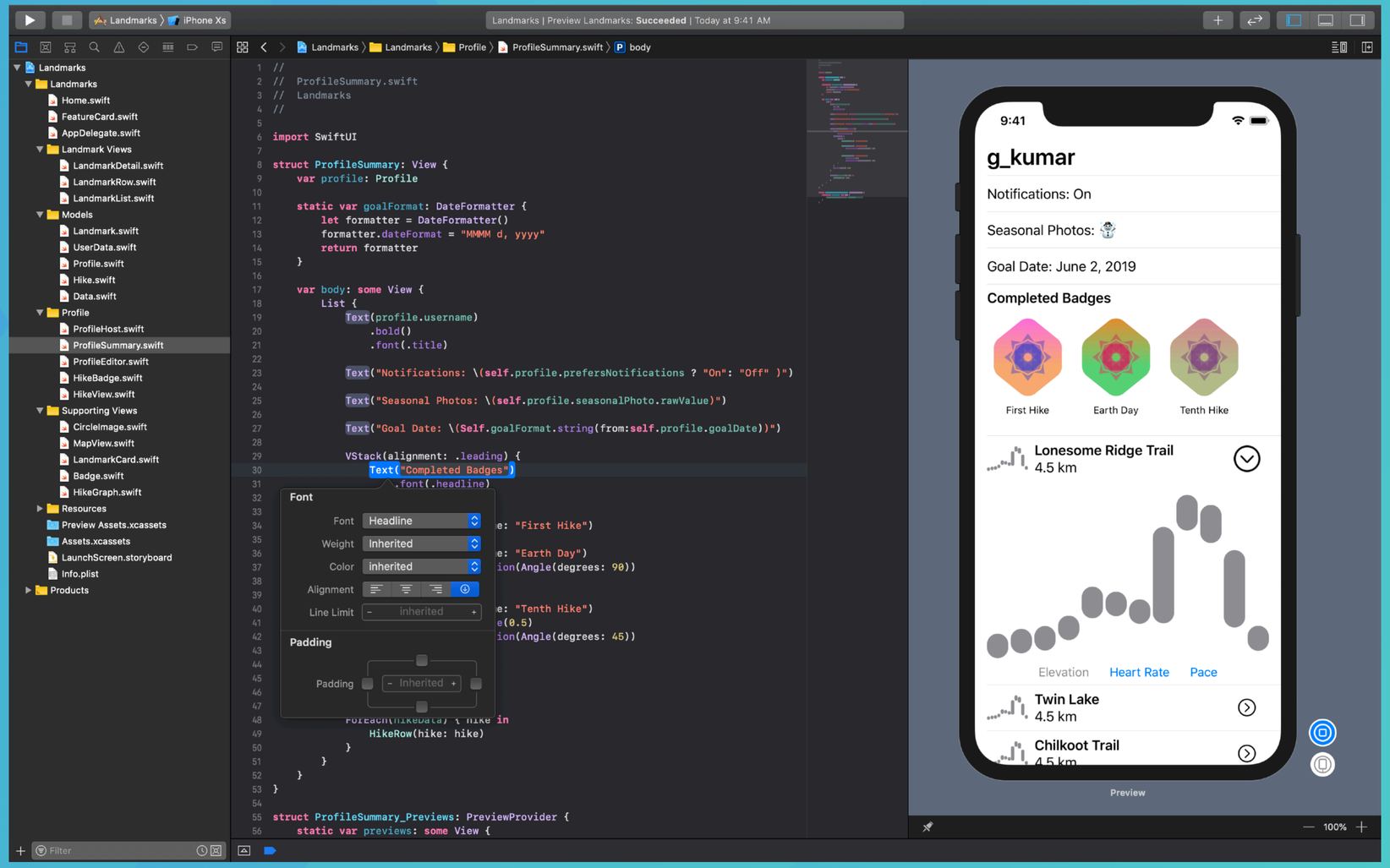


Live in Xcode

twitch.tv/dancojocar

youtube.com/dancojocar

DEMO



developer.apple.com/xcode/swiftui

Lecture outcomes

- Learn about Jetpack Compose.
- Build composable components.
- Learn about SwiftUI.