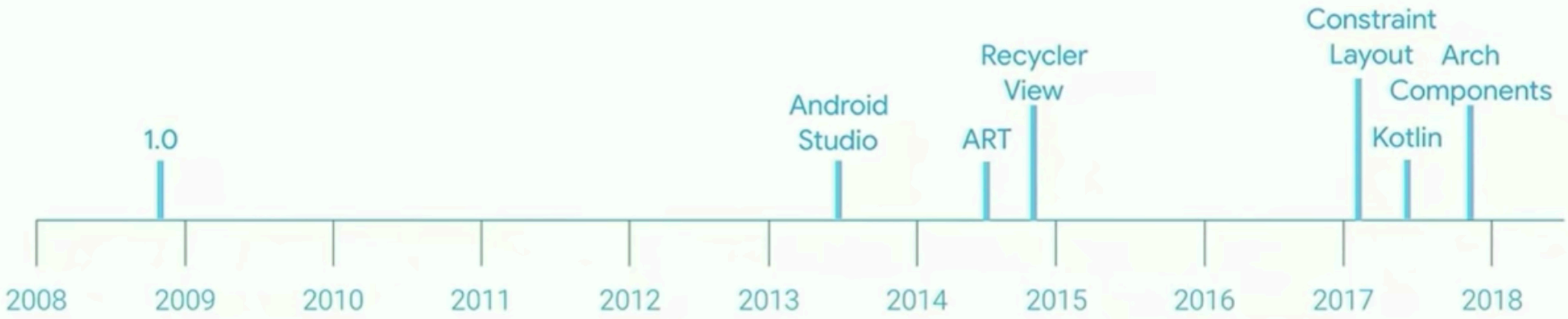
Lecture #13
Jetpack Compose & SwiftUI

Mobile Applications 2019-2020

Happy New Year!

2020


LIN
С РОЖДЕСТВОМ
ХРИСТОВЫМ



A brief history of Android

HTC Sooner

HTC Sooner

OMAP 850 processor

HTC Sooner

OMAP 850 processor

64 MB of RAM

HTC Sooner

OMAP 850 processor

64 MB of RAM

64 MB of on-board memory

HTC Sooner

OMAP 850 processor

64 MB of RAM

64 MB of on-board memory

a “generous”
320 x 240 pixel resolution

HTC Sooner

OMAP 850 processor

64 MB of RAM

64 MB of on-board memory

a “generous”
320 x 240 pixel resolution

NO touch input

T-Mobil G1

a “generous”
320 x 240 pixel resolution

T-Mobil G1

a “generous”
320 x 240 pixel resolution

TFT capacitive touchscreen

T-Mobil G1

TFT capacitive touchscreen

192MB RAM, 256MB

a “generous”
320 x 240 pixel resolution

T-Mobil G1

TFT capacitive touchscreen

192MB RAM, 256MB

Qualcomm MSM7201A

a “generous”
320 x 240 pixel resolution

T-Mobil G1

TFT capacitive touchscreen

192MB RAM, 256MB

Qualcomm MSM7201A

a “generous”
320 x 240 pixel resolution

320 x 480 pixels

Pixel 4

Pixel 4

Qualcomm SM8150 Snapdragon 855

Pixel 4

Qualcomm SM8150 Snapdragon 855

64GB, 6GB RAM

Pixel 4

Qualcomm SM8150 Snapdragon 855

64GB, 6GB RAM

1080 x 2280 pixels

Pixel 4

Qualcomm SM8150 Snapdragon 855

64GB, 6GB RAM

1080 x 2280 pixels

Pixel 4

Qualcomm SM8150 Snapdragon 855

64GB, 6GB RAM

1080 x 2280 pixels

Architectural Components approach to UI

- What are developers asking for?
- What would the framework team for developers to be doing?
- How to make things easer?

Among Top Answers

Among Top Answers

- Unbundle the UI toolkit!

Among Top Answers

- Unbundle the UI toolkit!
- aka move android.widget to Jetpack.

Among Top Answers

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

Some Improvements

- View.java

Some Improvements

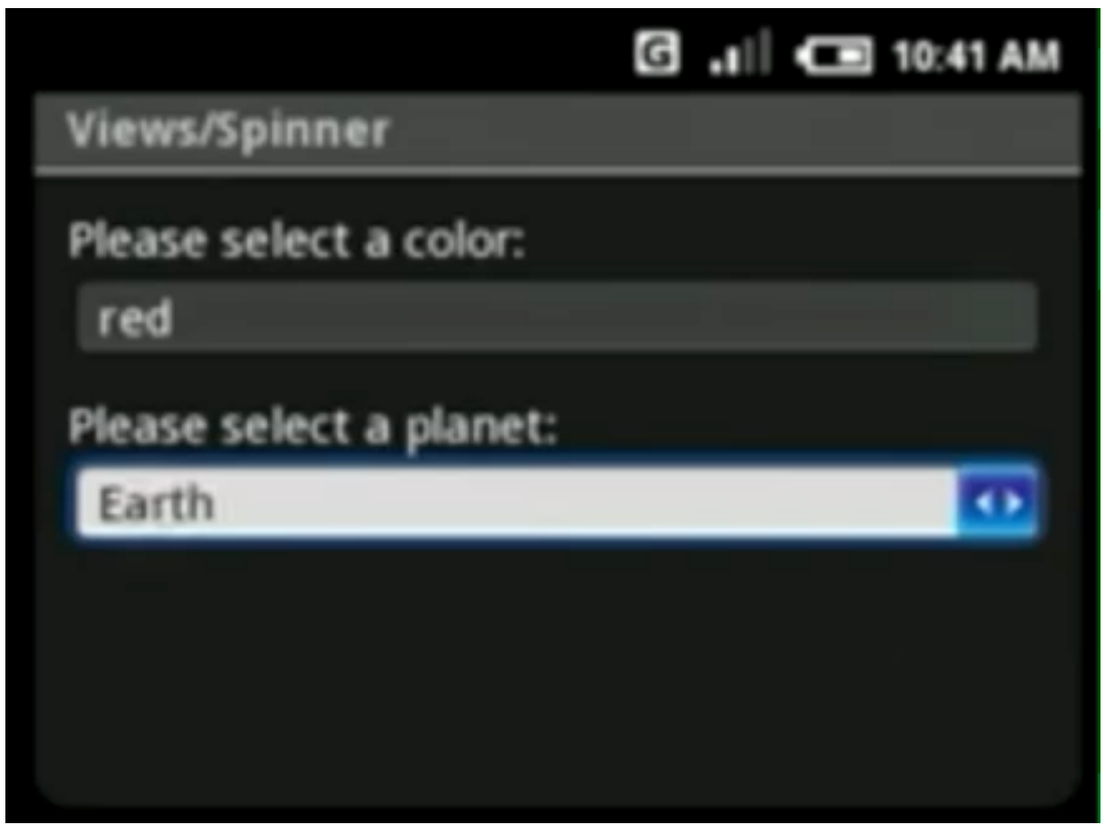
- View.java

```
29231
29232
29233
29234
29235
29236
29237
29238
29239
29240
```

```
mListenerInfo.mUr
if (mParent insta
    ((ViewGroup)
}
}
}
}
}
}
```

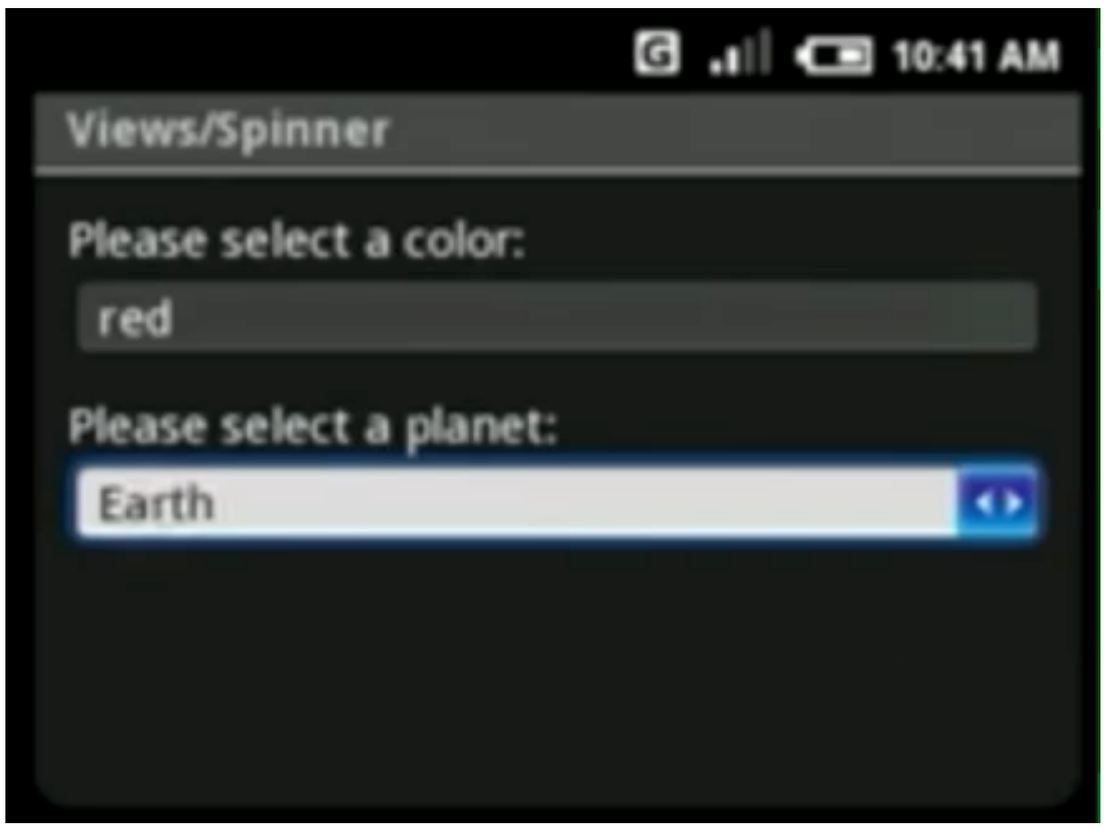
Some Improvements

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



Some Improvements

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

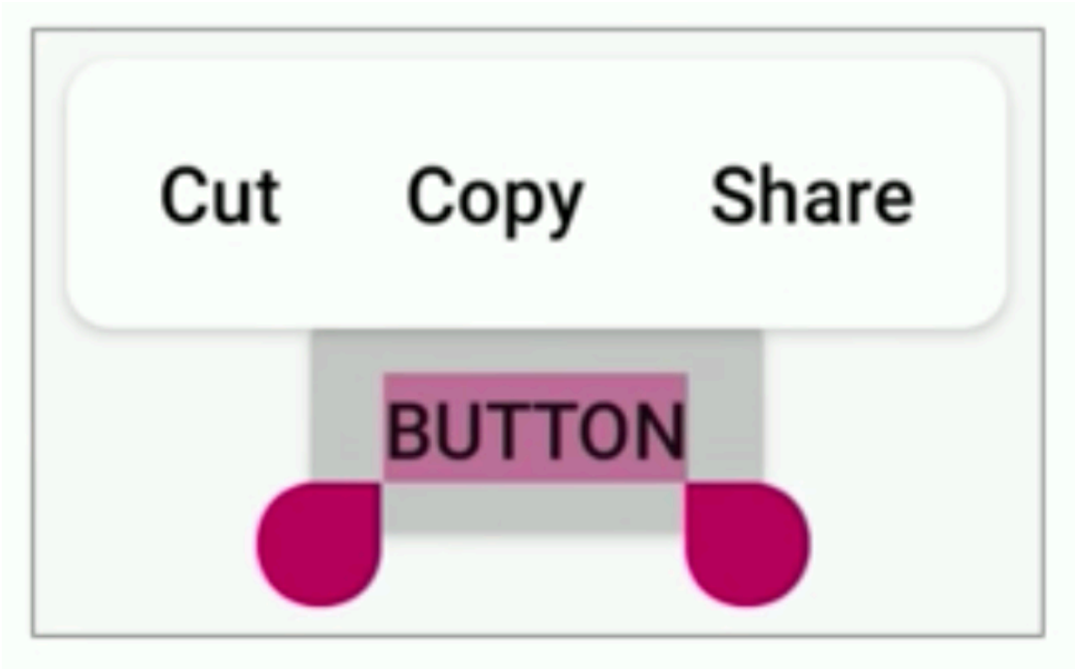


Some Improvements

public class Button **extends** TextView

Some Improvements

public class Button **extends** TextView



Too Much Code

Too Much Code

- My{Activity|Fragment}.{kt|java}

Too Much Code

- My{Activity|Fragment}.{kt|java}
- main_{activity|fragment}.xml

Too Much Code

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

Too Much Code

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

Goals

Goals

- Unbundle from platform releases.

Goals

- Unbundle from platform releases.
- Fewer choices when trying to build an UI element.

Goals

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

Goals

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

Can we go back to a simpler Time?

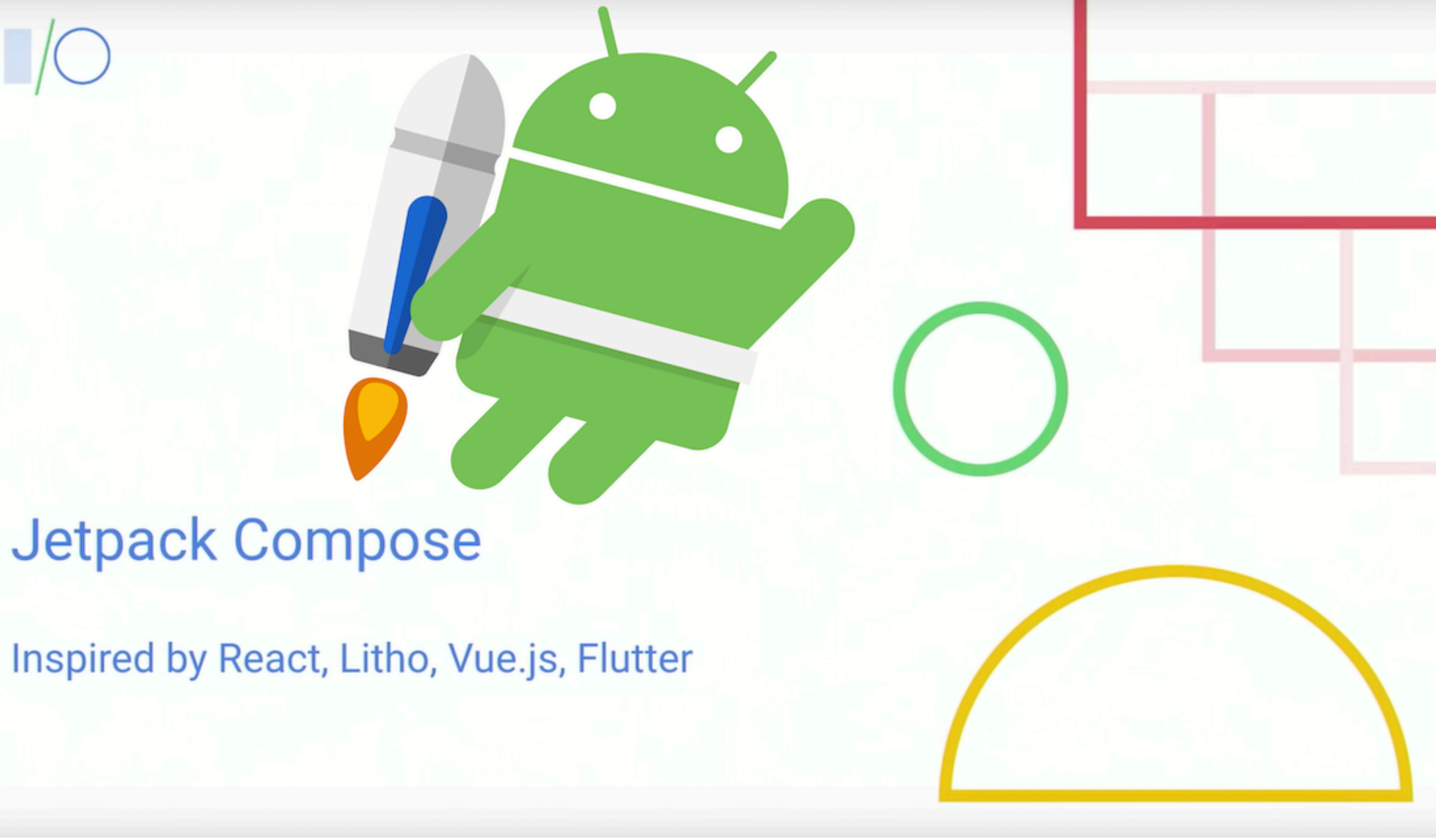
```
fun main(){
  print("Hello World!")
}
```

Can we go back to a simpler Time?

```
fun Greeting(){
  print(“Hello World!")
}
```

Can we go back to a simpler Time?

```
fun Greeting(){
  Text(“Hello World!")
}
```



developer.android.com/jetpack/compose

UI as a Function!

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

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

UI as a Function!


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

What is Compose?

What is Compose?

- 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 developement.

Use in Existing App

```
class MainActivity : AppCompatActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    setContent {
      Greeting("World!")
    }
  }
}
```

Composable UI Widget

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

Intercept & Rewrite

```
@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

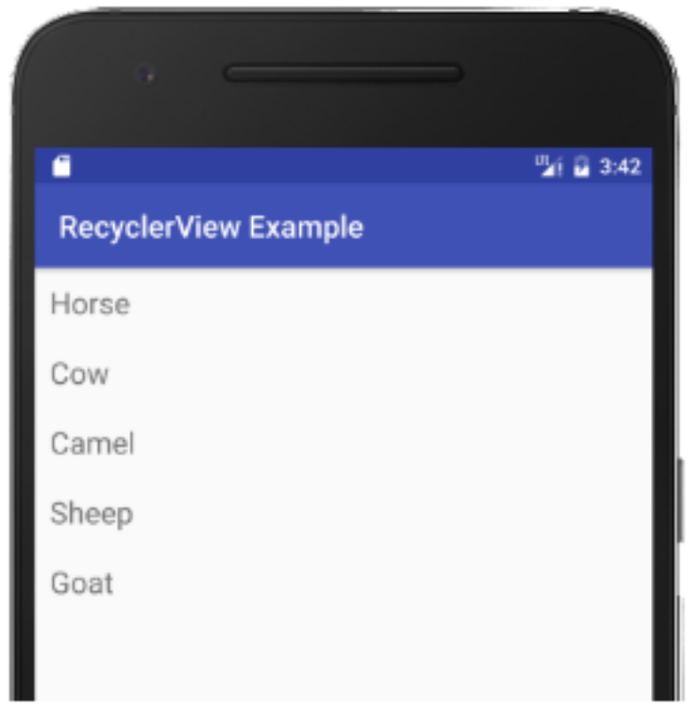
- Composable functions are defined in terms of other composable functions.

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

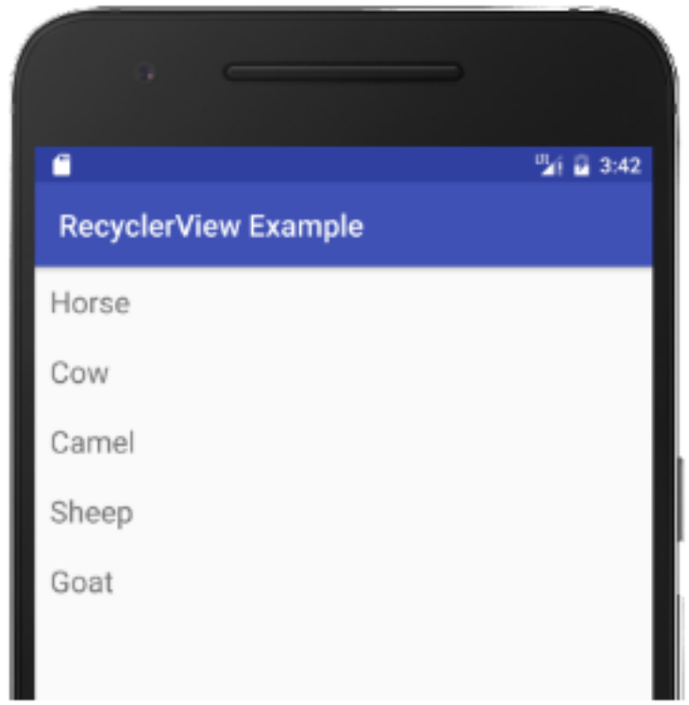
```
@Composable
fun StoryWidget(story: Story) {
  Padding(8.dp){
    Column{
      Title(story.title)
      Image(story.image)
      Text(story.content)
    }
  }
}
```

RecyclerView Adapter

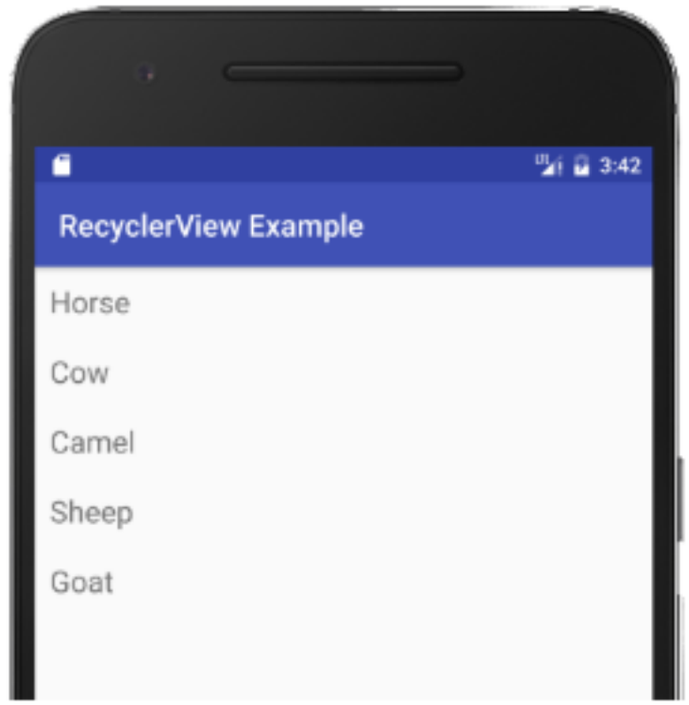
```
public class MyRecyclerViewAdapter extends RecyclerView.Adapter<MyRecyclerViewAdapter.ViewHolder> {

    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
```



```
    }

    // binds the data to the TextView in each row
    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        String animal = mData.get(position);
        holder.myTextView.setText(animal);
    }

    // total number of rows
    @Override
    public int getItemCount() {
        return mData.size();
    }


    // stores and recycles views as they are scrolled off screen
    public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
        TextView myTextView;

        ViewHolder(View itemView) {
            super(itemView);
            myTextView = itemView.findViewById(R.id.tvAnimalName);
            itemView.setOnClickListener(this);
        }
```



```
        itemView.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        if (mClickListener != null) mClickListener.onItemClick(view, getAdapterPosition());
    }
}

// convenience method for getting data at click position
String getItem(int id) {
    return mData.get(id);
}

// allows clicks events to be caught
void setClickListener(ItemClickListener itemClickListener) {
    this.mClickListener = itemClickListener;
}

// parent activity will implement this method to respond to click events
public interface ItemClickListener {
    void onItemClick(View view, int position);
}
}
```



```
@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)
    }
  }
}
```

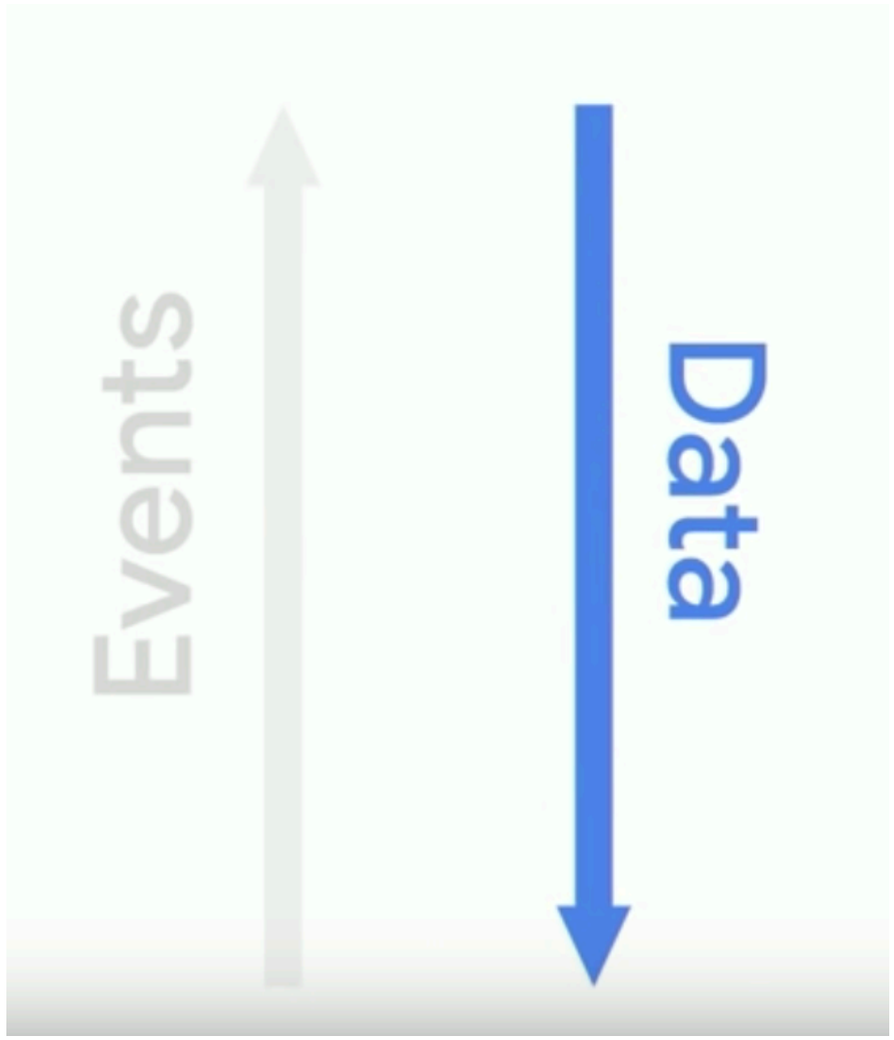
```
@Composable
fun NewsFeed(stories: LiveDate<List<Story>>) {
  ScrollingList(stories.observer()) { story ->
    StoryWidget(story)
  }
}
```

```
@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)
      }
    }
  }
}
```

```
@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
)
```

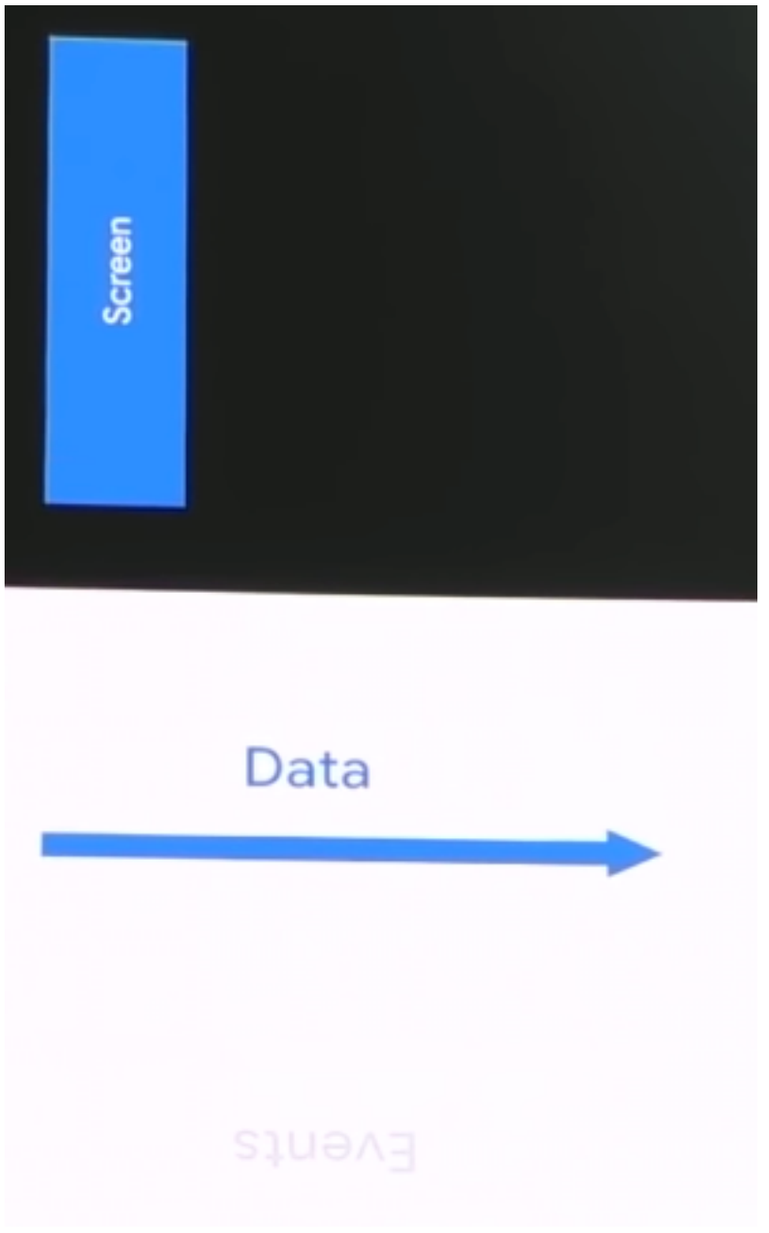
```
@Composable
fun StoryWidget(story: Story) {
  val image = asyncLoad(defaultPlaceholder) {
    loadImage(story.imageUri)
  }
  Card(cornerRadius = 4.dp, elevation = 4.dp) {
    Column {
      Image(image)
      Padding(16.dp) {
        Text(story.headline)
      }
    }
  }
}

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

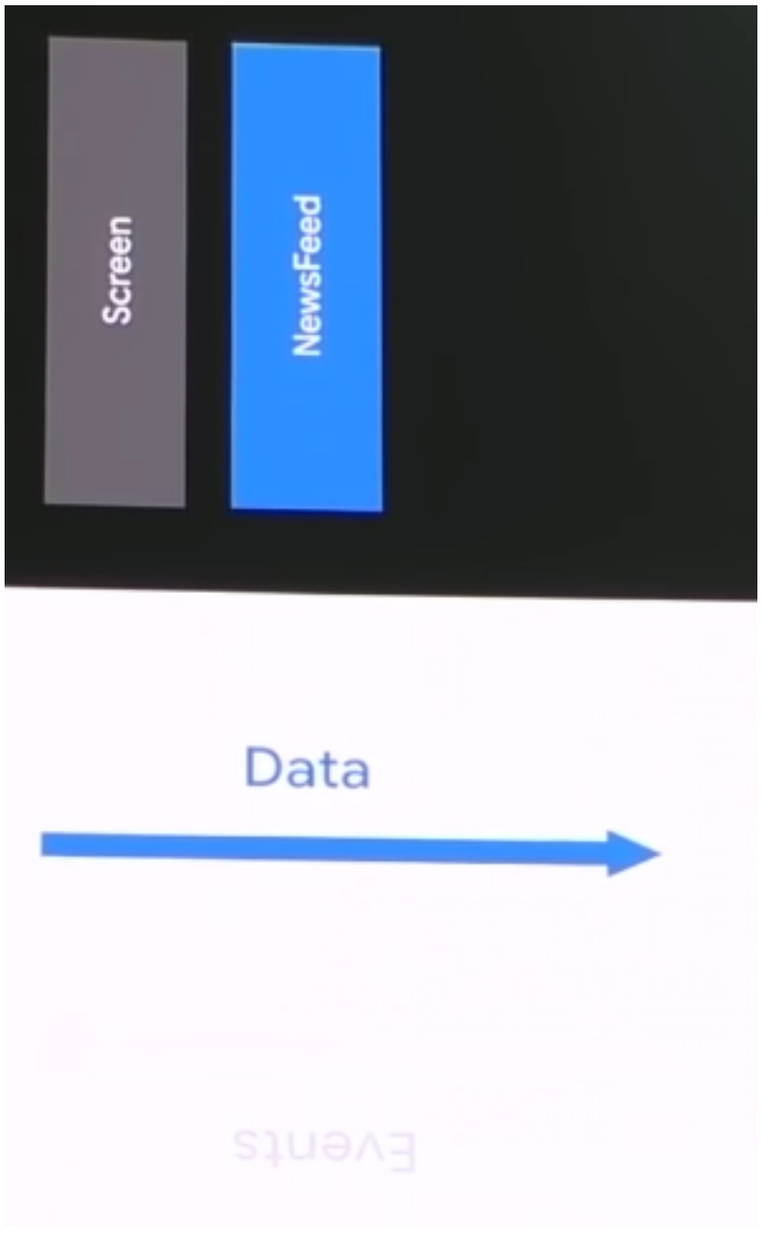
Top-down Data Flow

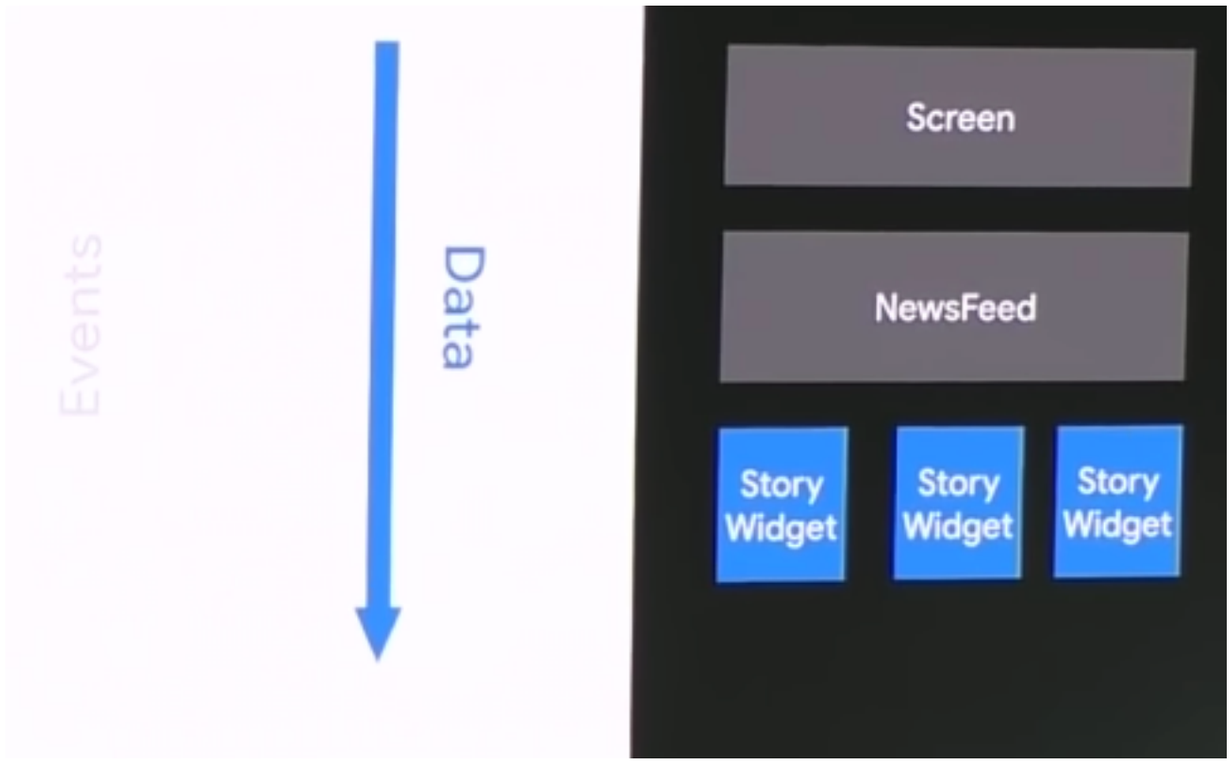


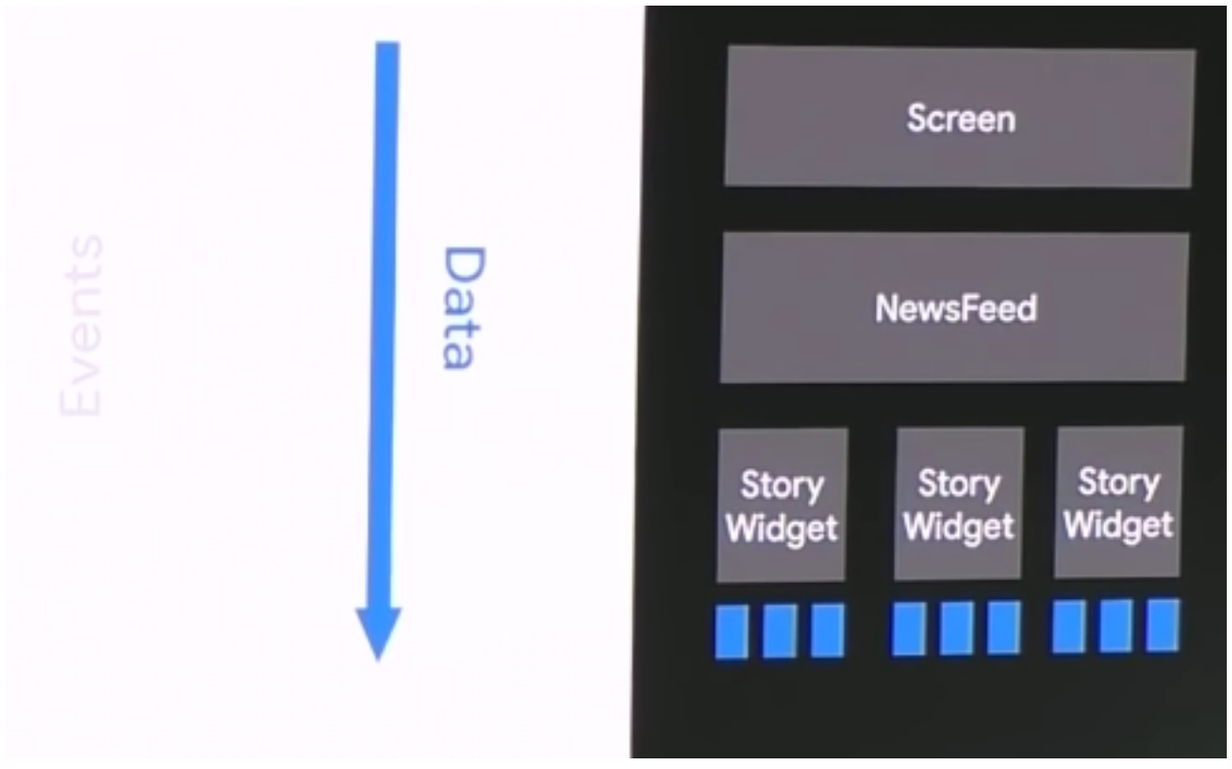

```
@Composable
fun NewsFeed(stories: List<Story>) {
  for (story in stories) {
    StoryWidget(story)
  }
}

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

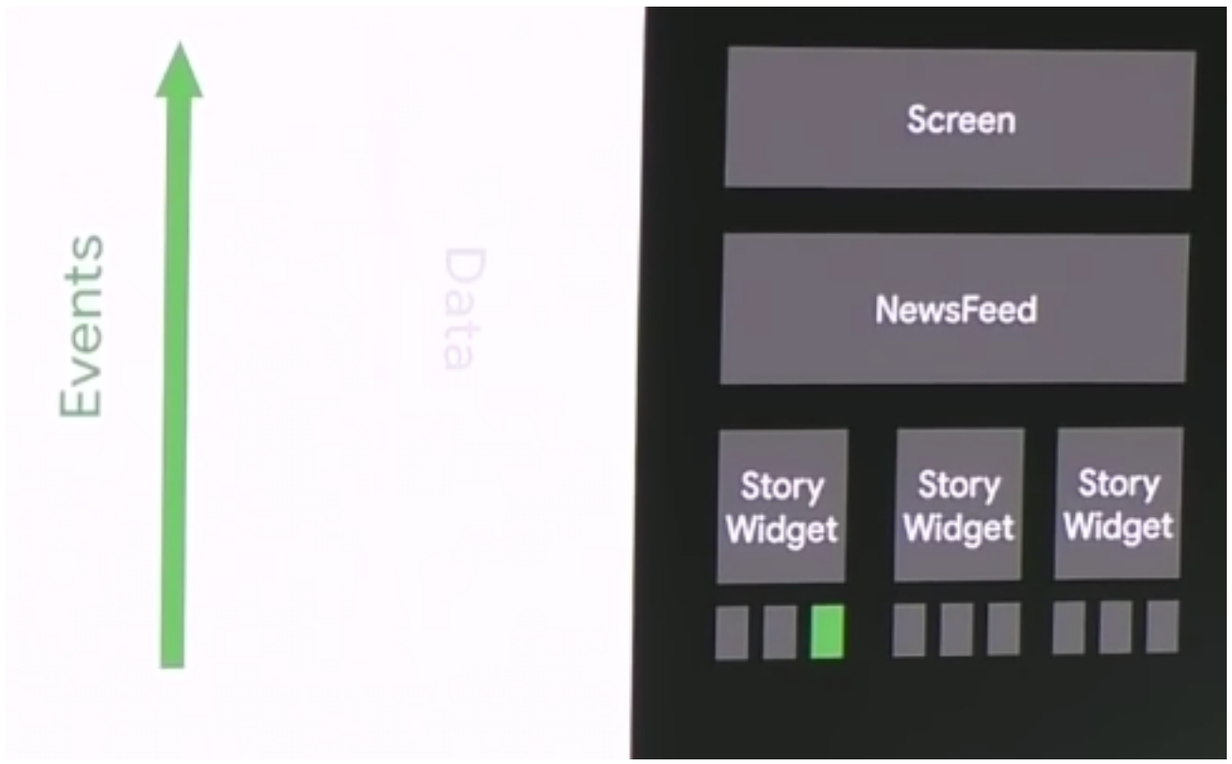
```
@Composable
fun NewsFeed(stories: List<Story>,
             onSelected: (Story) -> Unit) {
  ScrolingList(stories) { story ->
    StoryWidget(story, onClick = { onSelected(story) })
  }
}

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

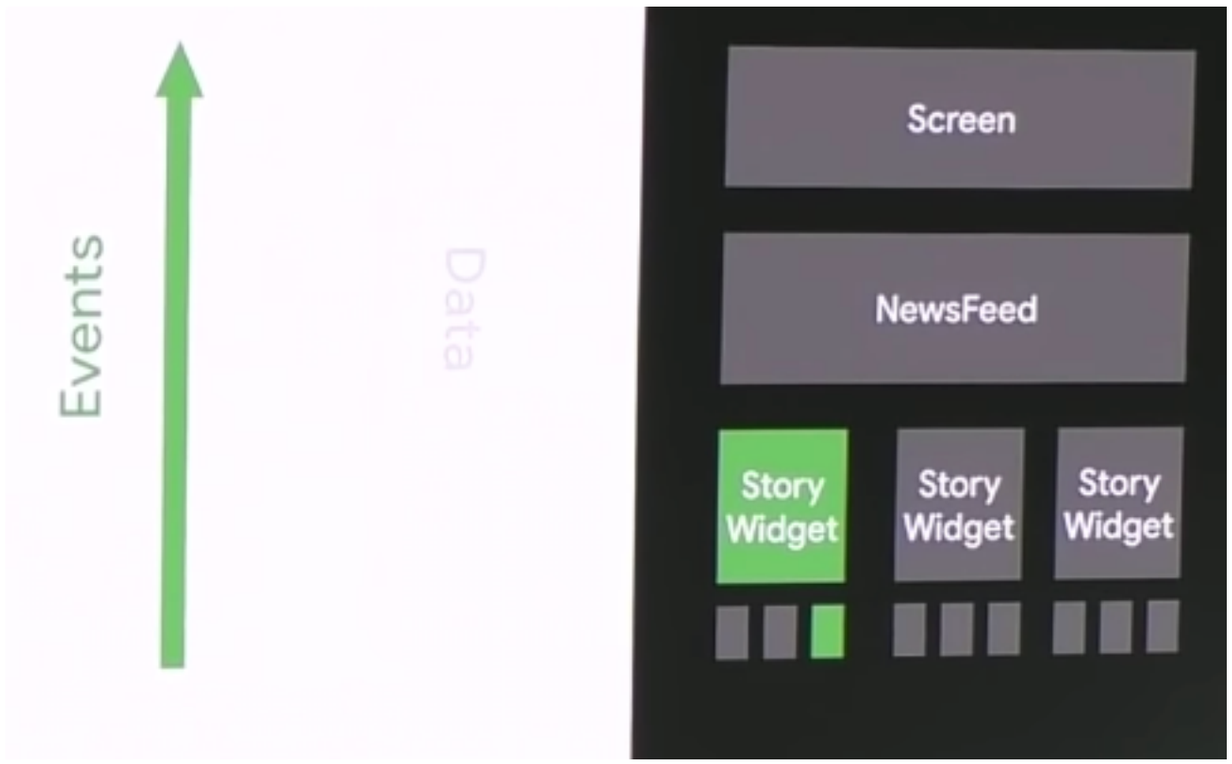

Screen
Data
Events

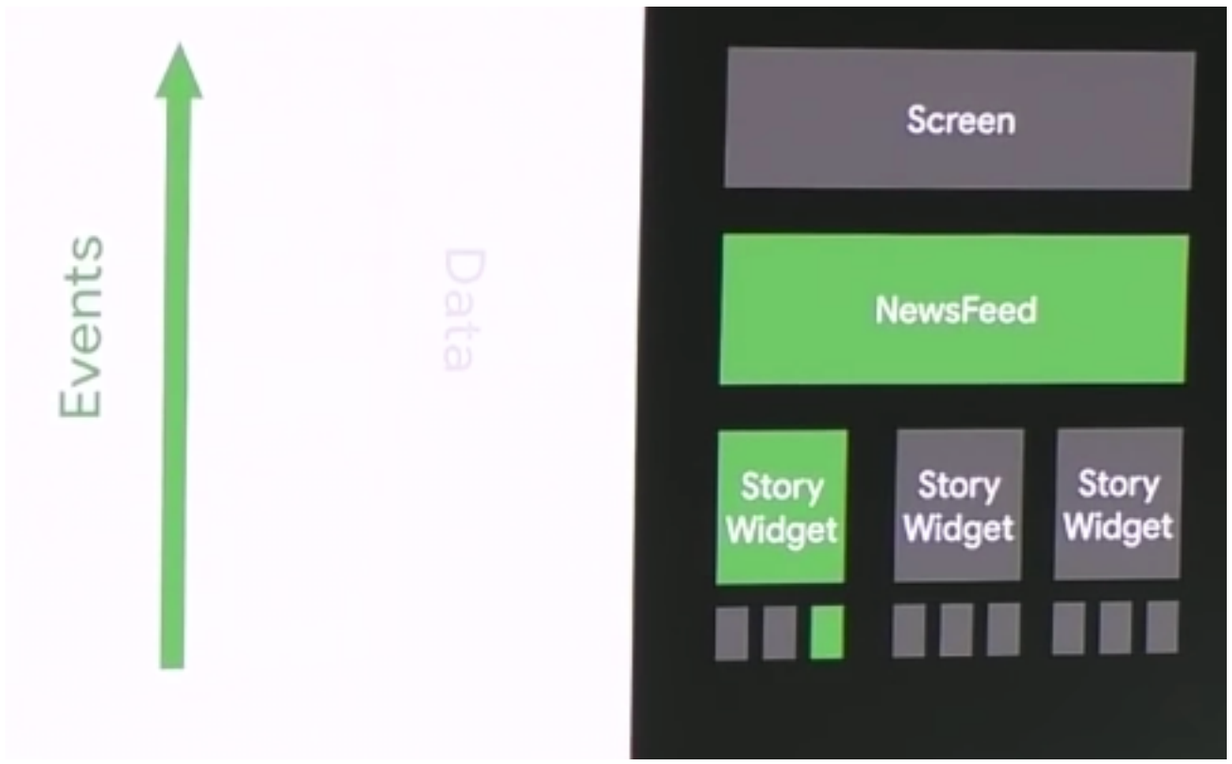

Screen
NewsFeed
Data
Events

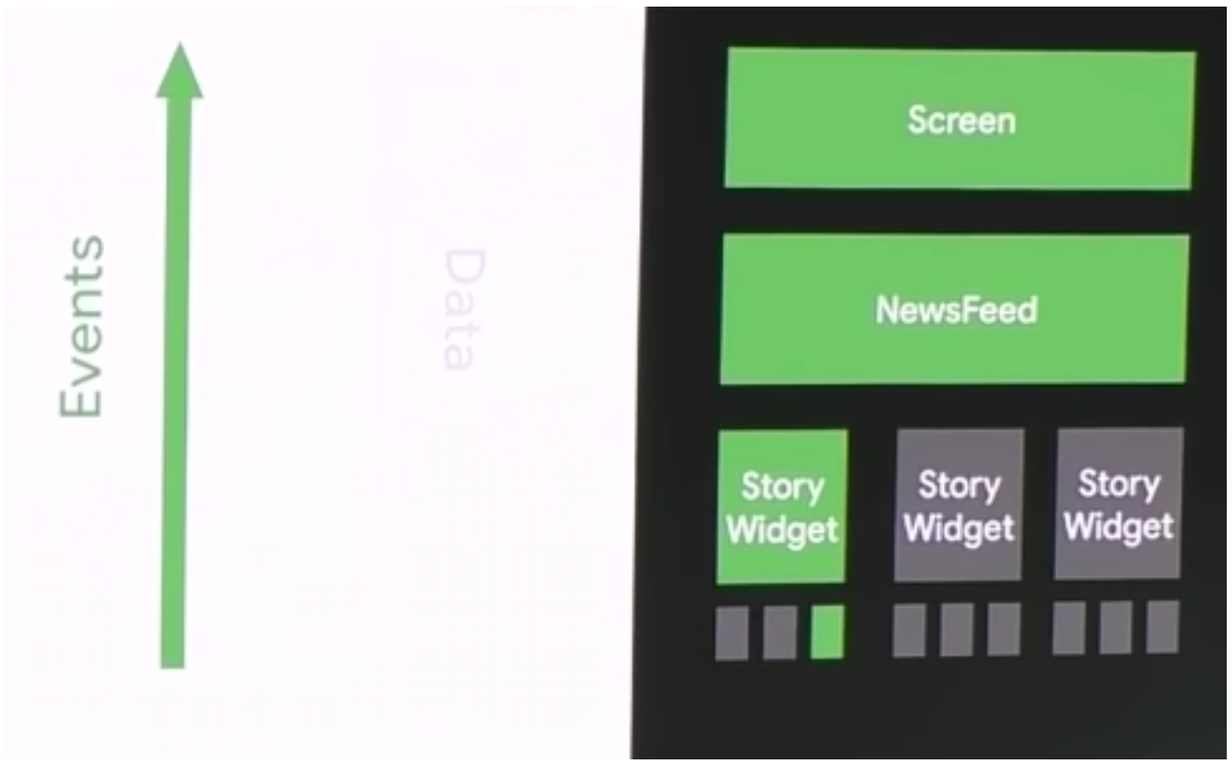

Events
Data
Screen
NewsFeed
Story Widget
Story Widget
Story Widget

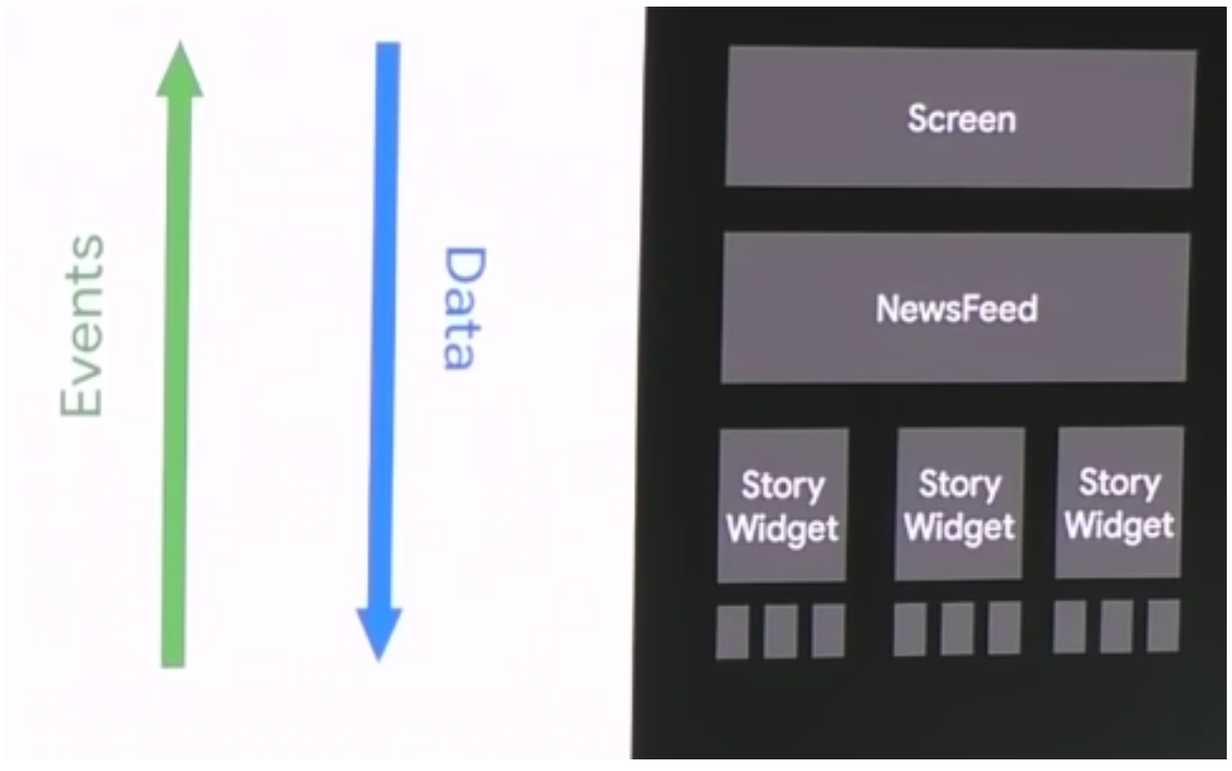

Events
Data
Screen
NewsFeed
Story Widget
Story Widget
Story Widget


Events
Data
Screen
NewsFeed
Story Widget
Story Widget
Story Widget

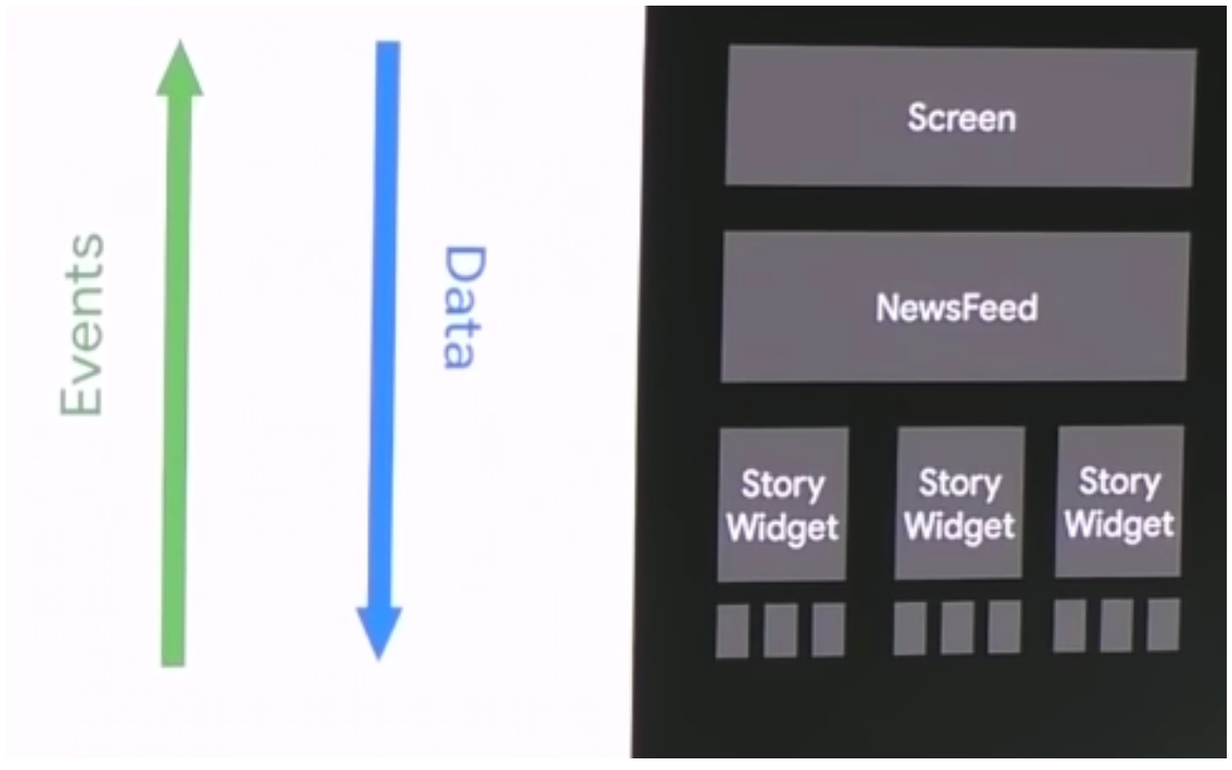

Events
Data
Screen
NewsFeed
Story Widget
Story Widget
Story Widget


Events
Data
Screen
NewsFeed
Story Widget
Story Widget
Story Widget


Events
Data
Screen
NewsFeed
Story Widget
Story Widget
Story Widget


Events
Data
Screen
NewsFeed
Story Widget
Story Widget
Story Widget

Since Source of Truth



Multiple Sources 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 ~~s~~ of Truth

- Let the app be the source.

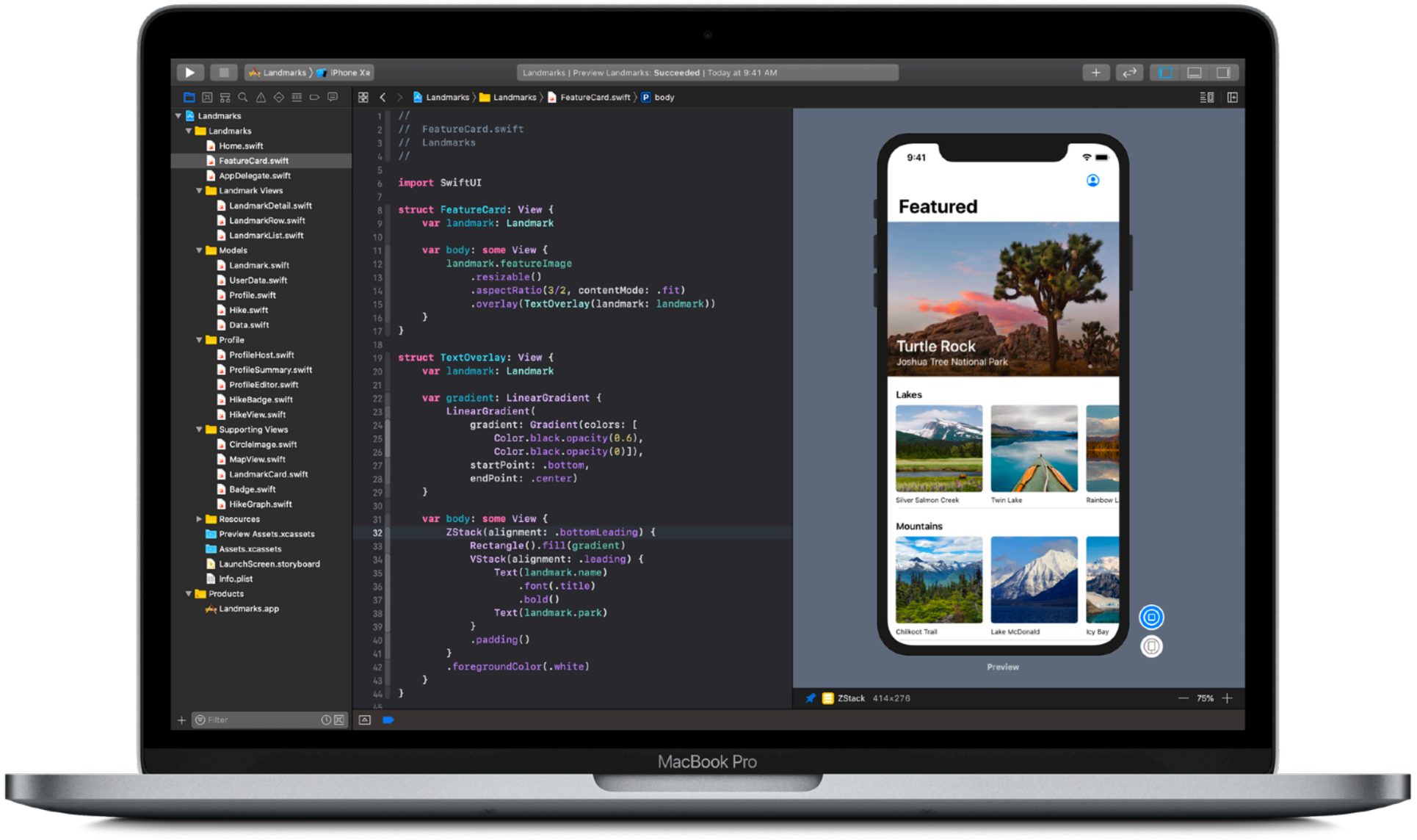
```
@Composable
fun FoodPreferences(data: UserPreferences) {
  val options = listOf(MILK, EGGS, BREAD)
  Checkbox(
    options = options,
    selected = data.favoriteFood,
    onChange = { selected -> data.favoriteFood = selected }
  )
}
```

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

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

Introducing SwiftUI

Better apps. Less code.

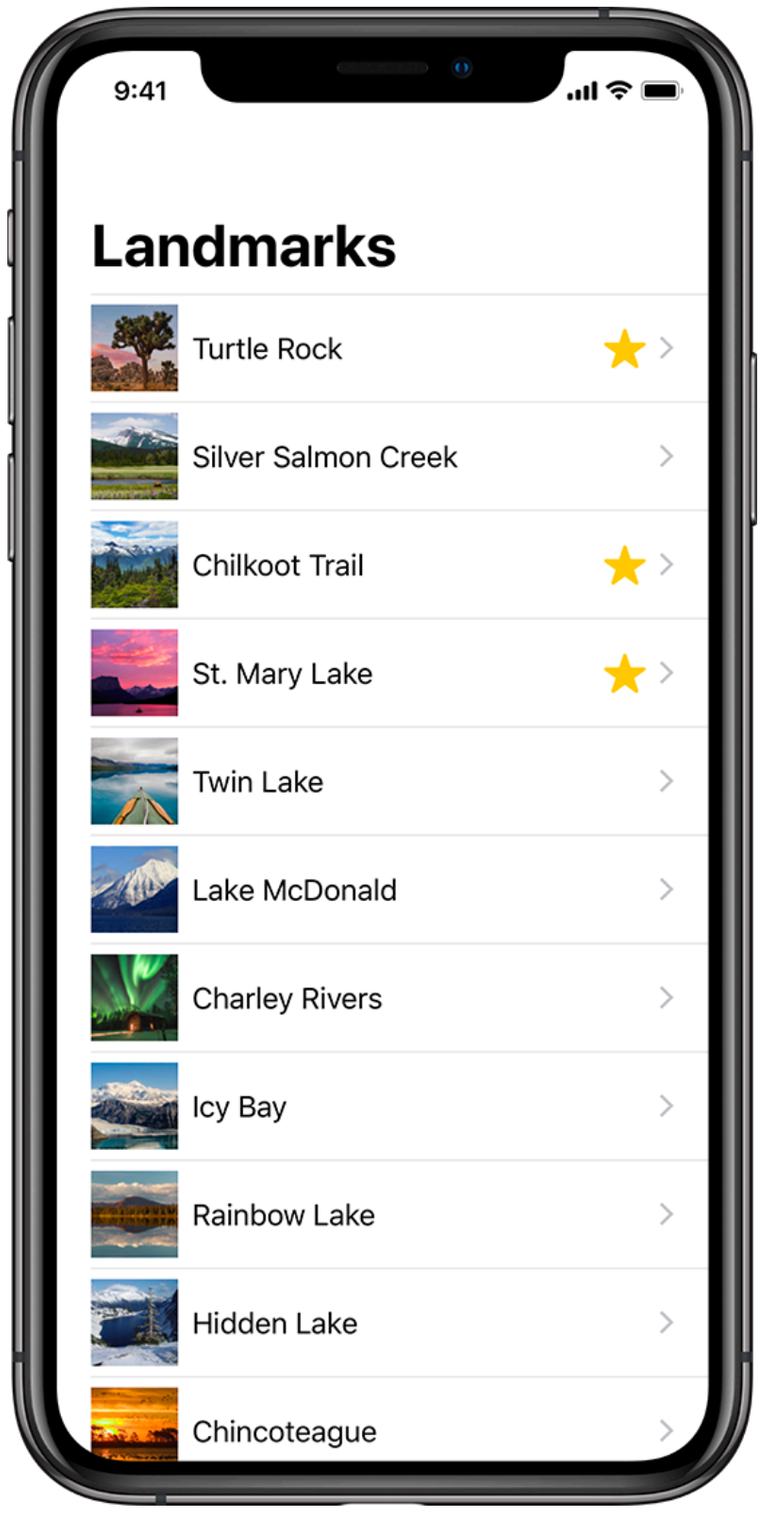


developer.apple.com/xcode/swiftui/

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.

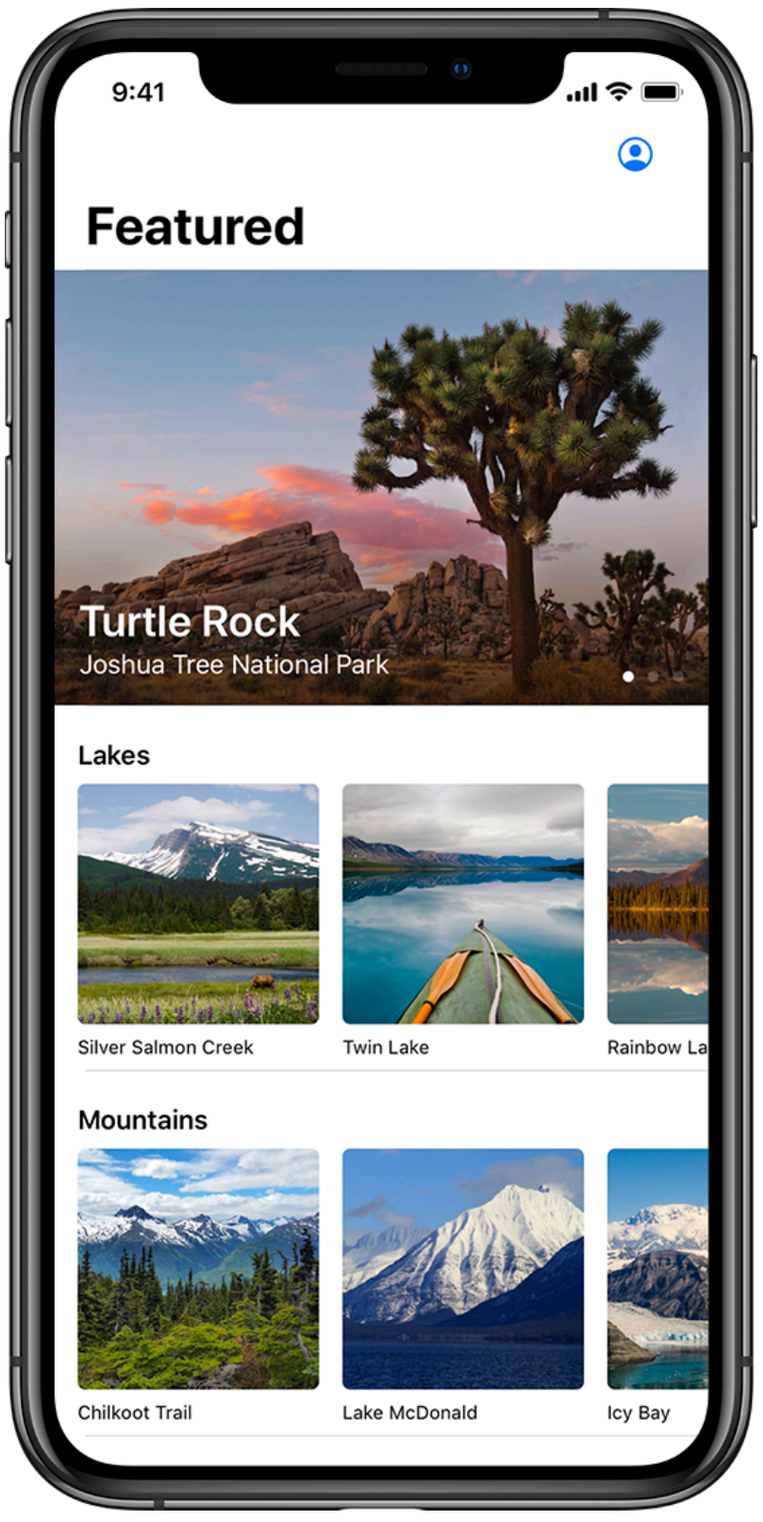
```
List(landmarks) { landmark in
  HStack {
    Image(landmark.thumbnail)
    Text(landmark.name)
    Spacer()

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



Build reusable components

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

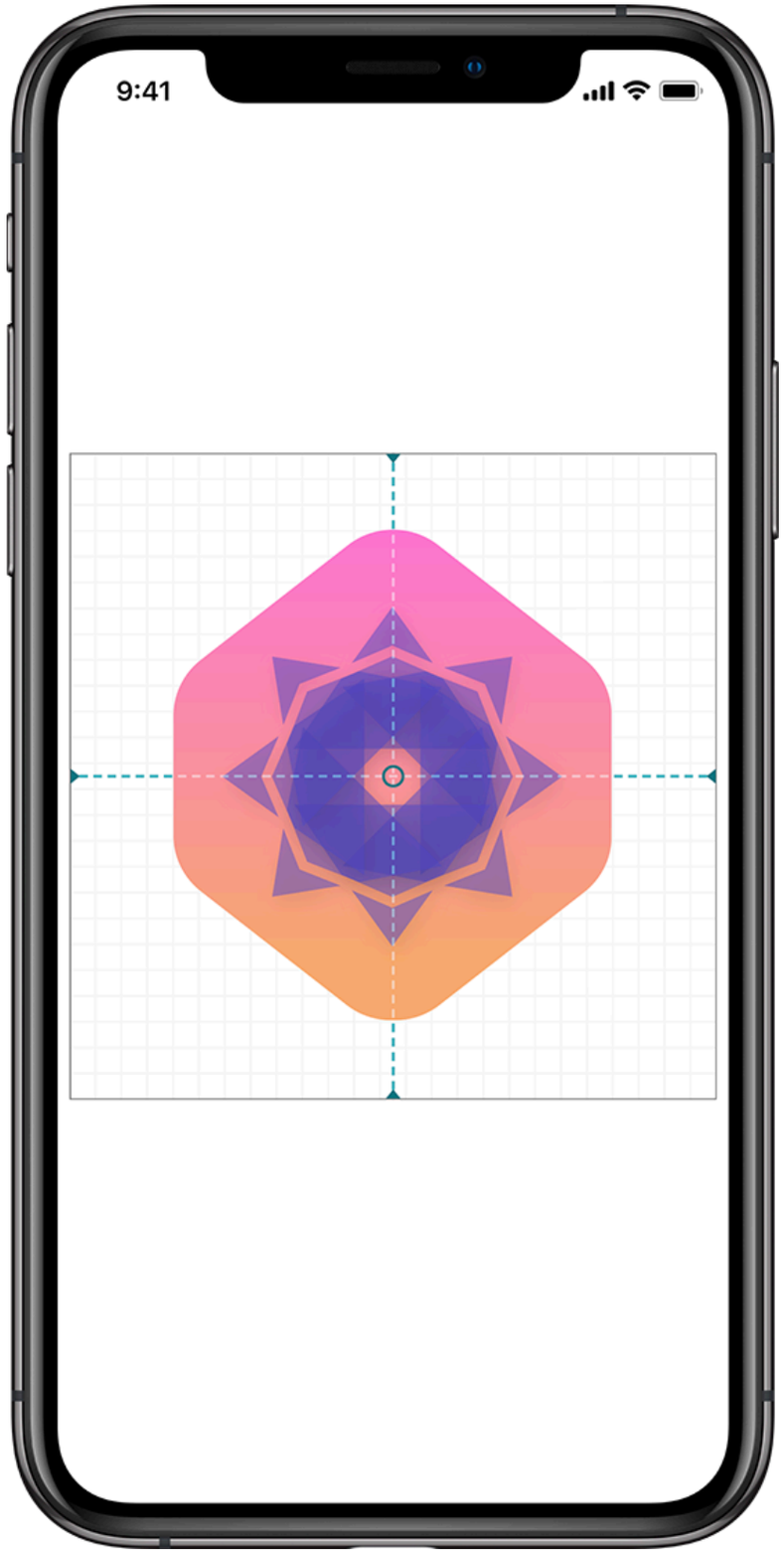
```
struct FeatureCard: View {
    var landmark: Landmark

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



Simplify Animations

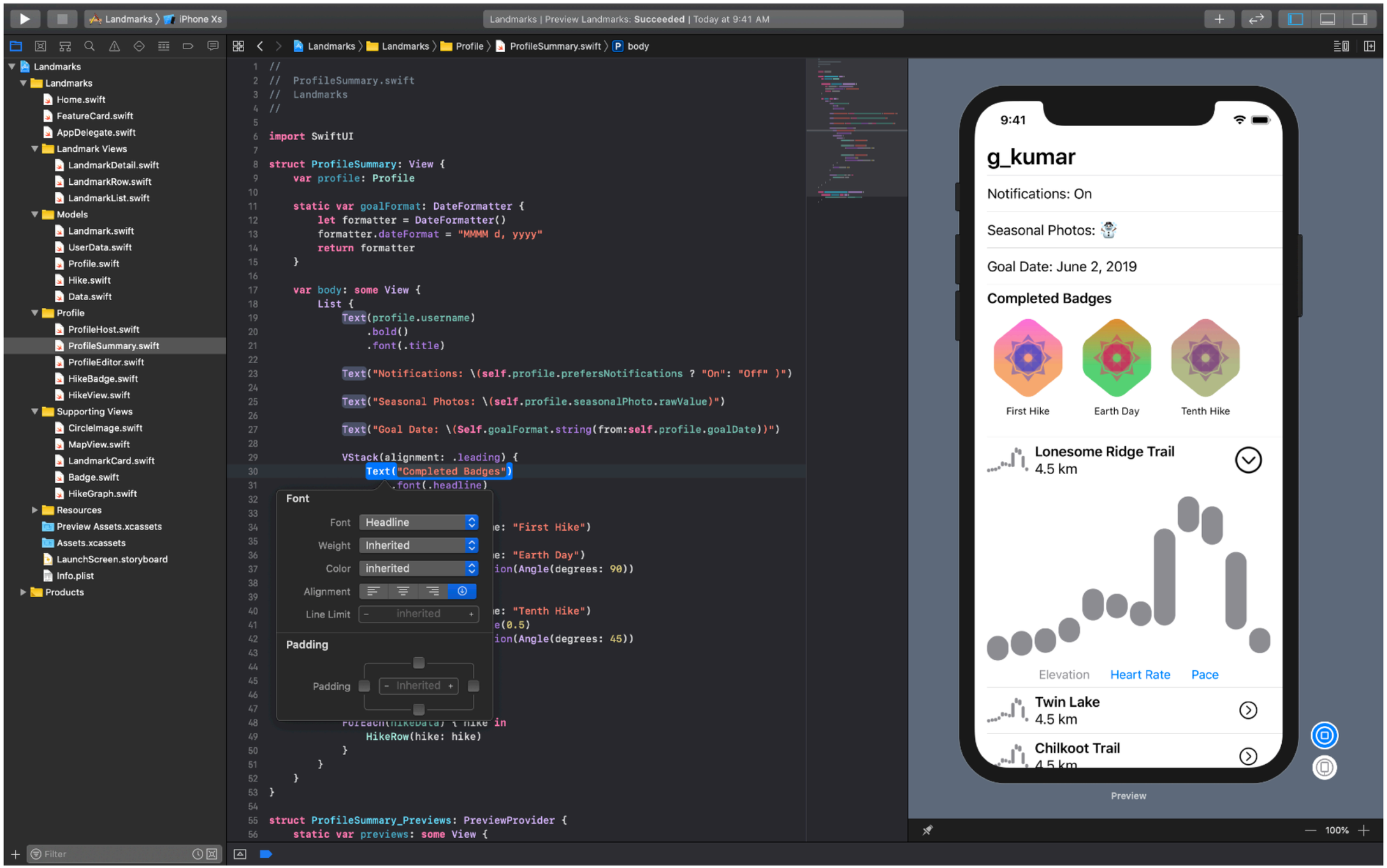
- Creating smooth animations is as easy as adding a single method call. SwiftUI automatically calculates and animates transitions when needed.

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



DEMO

Live in Xcode



developer.apple.com/xcode/swiftui/

Lecture outcomes

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