

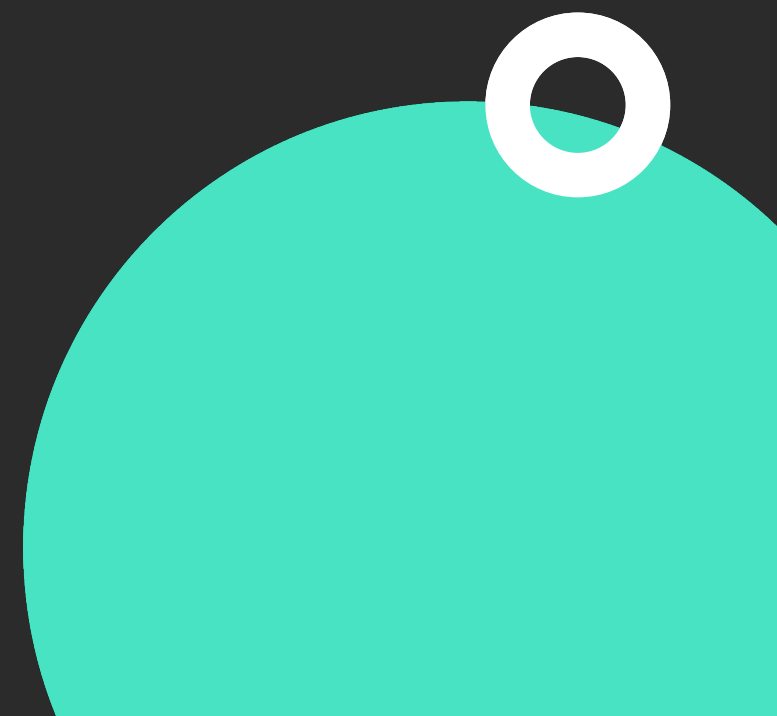


# Accessibility in Android Jetpack Compose: Validation, Announcements and Animations



Quintin Balsdon

**in** [In/qbalsdon](#)



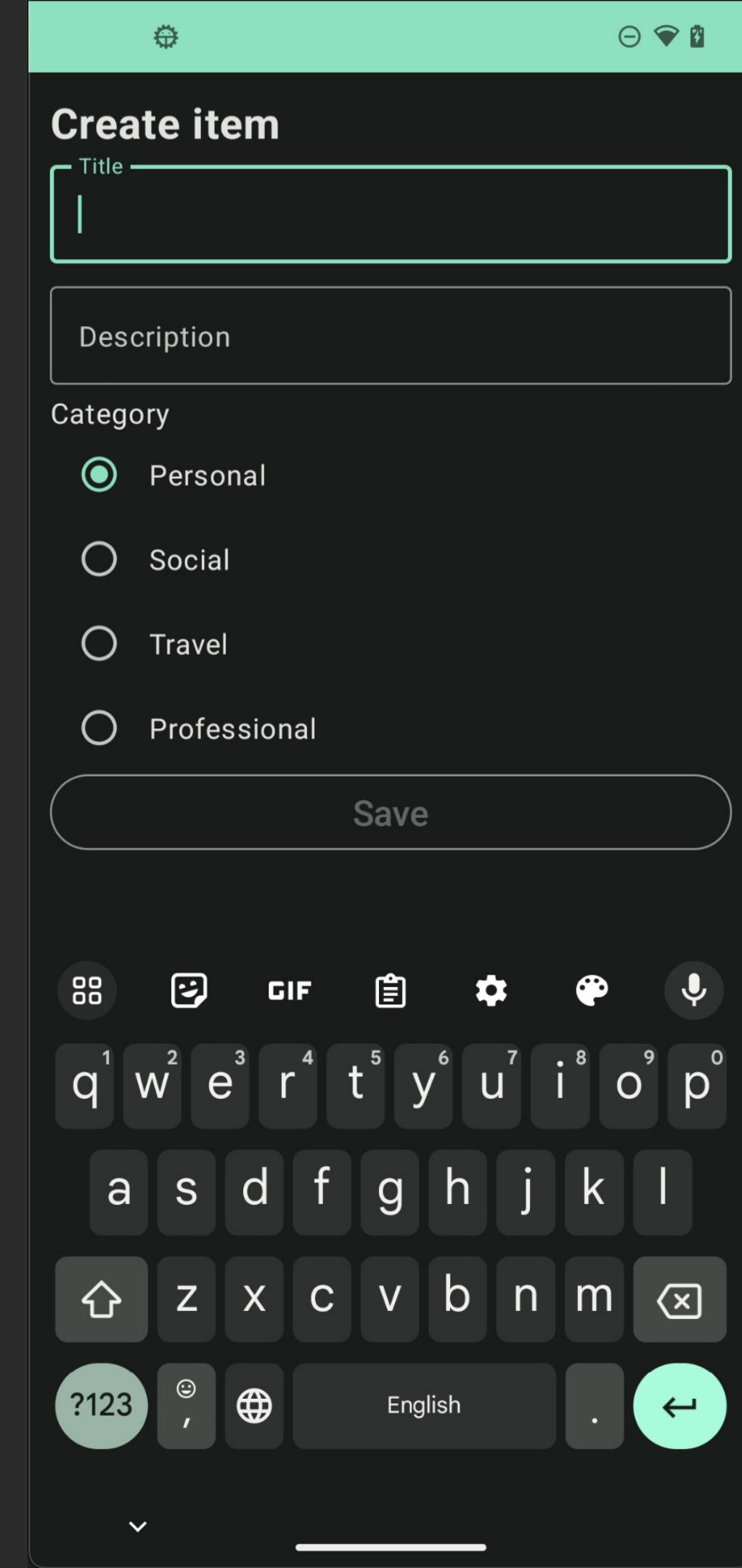
# Section Overview

- Error states and validation
- Live regions and announcements
- Ignoring content
- Animations and timing



# Error states

- The goal of a user interface should be:
  - Allow the user to **complete a task**
  - Show them **what** they need, **when** they need it
  - Users should not be surprised, or have to guess
- Many different approaches – consider your user **first**



**Create item**

Title

Description

Category

☒ Personal

☐ Social

☐ Travel

☐ Professional

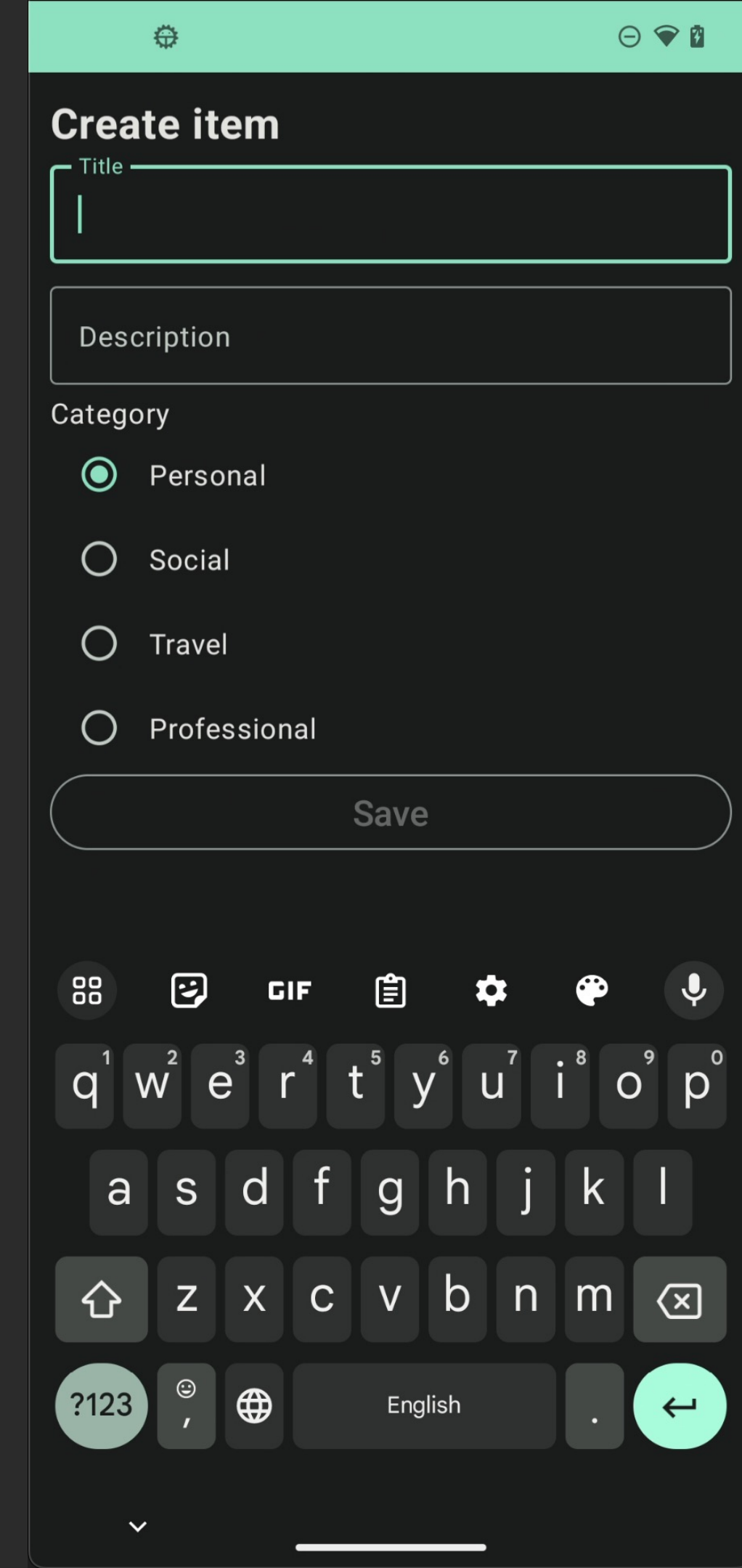
**Save**

?123 , . English ↵



# Error states: Current issues

- Save button disabled until valid state reached
- Users unaware of validation rules
- Perception problems for users
  - Not paying attention
  - Not differentiate color well
  - Who are blind
  - Use a magnifier
- No examples for users to follow



**Create item**

Title

Description

Category

☒ Personal

☐ Social

☐ Travel

☐ Professional

Save

Keyboard icons: App Store, Messages, GIF, Clipboard, Settings, App Store, Microphone

q<sup>1</sup> w<sup>2</sup> e<sup>3</sup> r<sup>4</sup> t<sup>5</sup> y<sup>6</sup> u<sup>7</sup> i<sup>8</sup> o<sup>9</sup> p<sup>0</sup>

a s d f g h j k l

⬆ z x c v b n m ⬆

?123 , ' & English . ↵



```
val localView = LocalView.current
// ...
event = {
    localView.announceForAccessibility("My announcement")
    // ...
}

modifier = Modifier
    .semantics {
        liveRegion = LiveRegionMode.Polite
        // Assertive
    }
```

## Making announcements: Two approaches

- Accessibility Announcement
- Live region



# Ignoring content

- Elements can be ignored if
  - They are purely decorative
  - there is enough context somewhere else

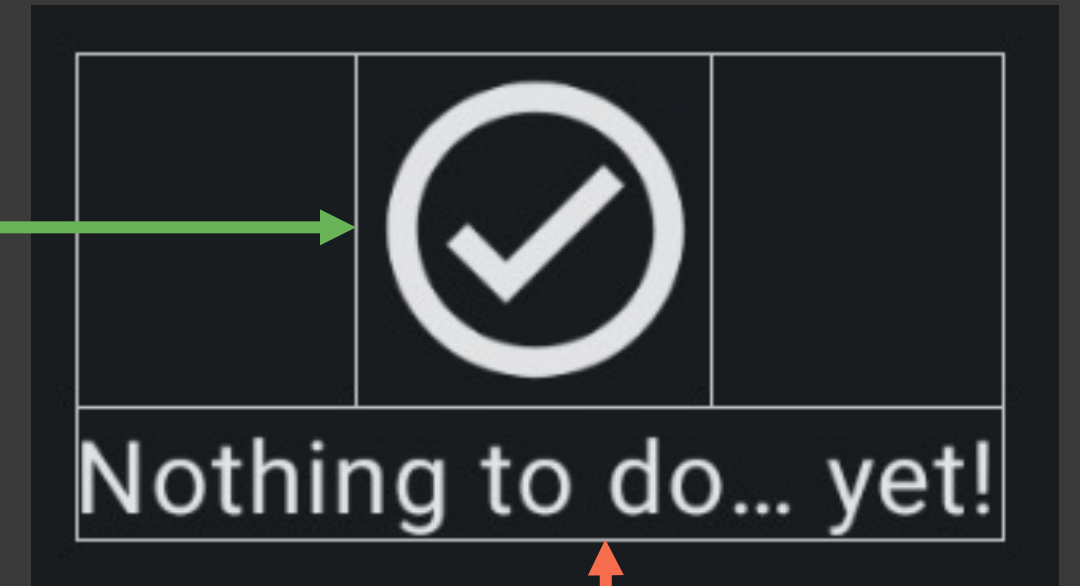
**Bonus section:** writing good content descriptions

- Explain the **key purpose** of the image
- If there is text, make sure that is included
- **Never** use it as an “easter egg”
- NASA is very good at content descriptions (or alt text)

“Against the blackness of space, Earth is seen rising over the lunar surface, which is gray and full of craters in the foreground. The lower part of the globe is in shadow, but the rest of Earth is mostly blue water with swirling white clouds and a few patches of brown land.”



```
Icon(  
    modifier = Modifier.size(56.dp) ,  
    imageVector = Icons.Outlined.CheckCircle ,  
    contentDescription = null)  
Text(text = stringResource(id = R.string.empty_list))
```



## Ignoring decorative elements: Icons and images

- The content description of the Icon can be set to null because the icon is grouped with a descriptive
- If this label updated periodically, but not regularly, it's the perfect candidate for a live region



```
Column(  
    ...  
) {  
    AnimatedLoadingIndicator(modifier.clearAndSetSemantics { })  
    Text(...)  
}
```

## Ignoring more complex elements

- This clears the semantics of all the descendant nodes and sets new semantics, in this case, none.





# Animation and timing

- **Respect** a user's decision as to whether they want to see animations
- Guidelines for animations / video:
  - Avoid **flashing**, or warn users
  - Allow users to **pause** or **hide** them in the app
- Timing controls
  - Check if users have set the “time to take action” setting
- Test with animations off!



```
const val ANIMATION_DEFAULT = 1.0f
...
val localContext = LocalContext.current
var animationsOn by remember {
    mutableStateOf(
        Settings.Global.getFloat(
            localContext.contentResolver,
            Settings.Global.ANIMATOR_DURATION_SCALE,
            ANIMATION_DEFAULT) == ANIMATION_DEFAULT
    )
}
```

## Checking for animation settings

- Always check API levels – this was added in 17
- You may need to restart your app



# Section Summary

- Error states
- Validation
- When to ignore content
- Dealing with animations and timing





# Final Section

Up Next

