Building Compose Apps for Everyone

Making your Jetpack Compose Apps Accessible

@hitherejoe

15%

The percentage of the population, **worldwide**, with some form of disability

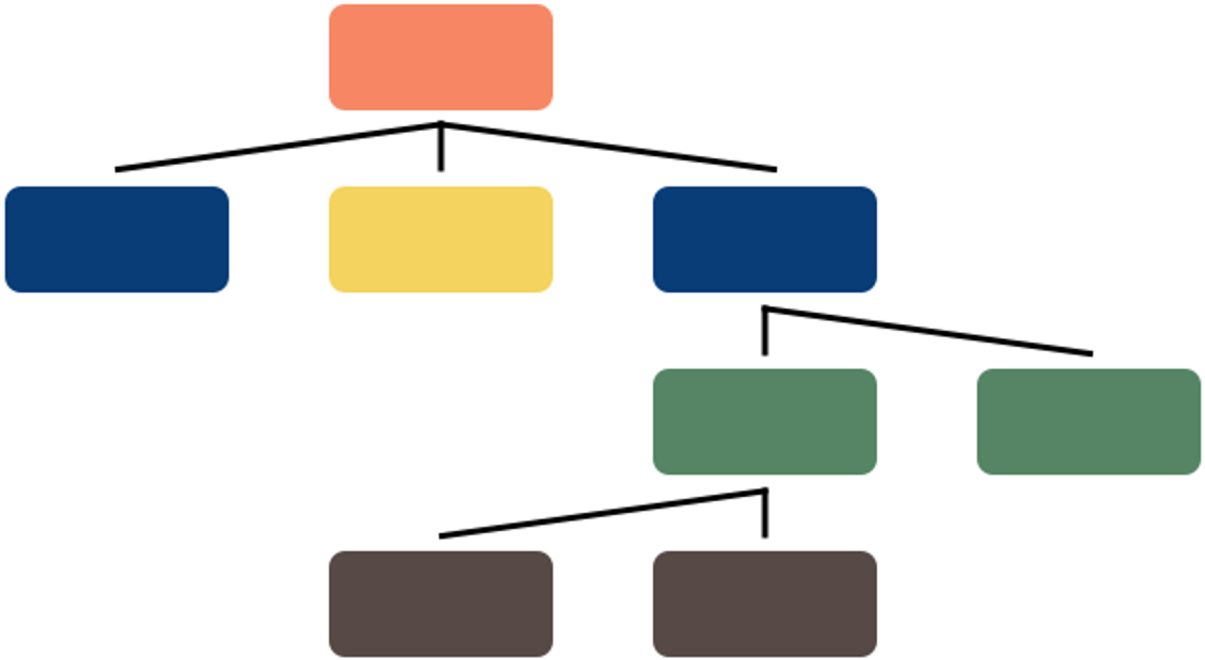
17%

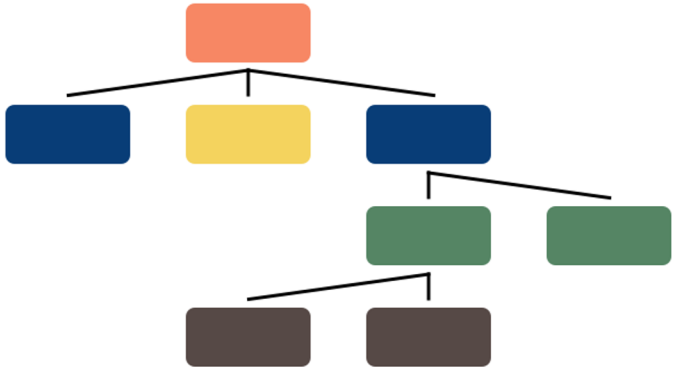
The percentage of people, **in the UK**, who were born with their disabilities

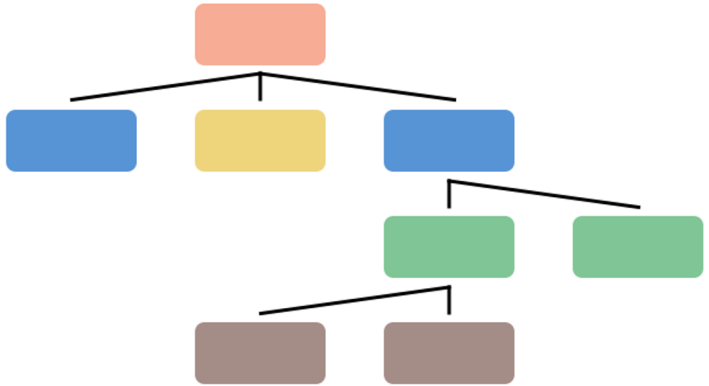
How can we make composables accessible?

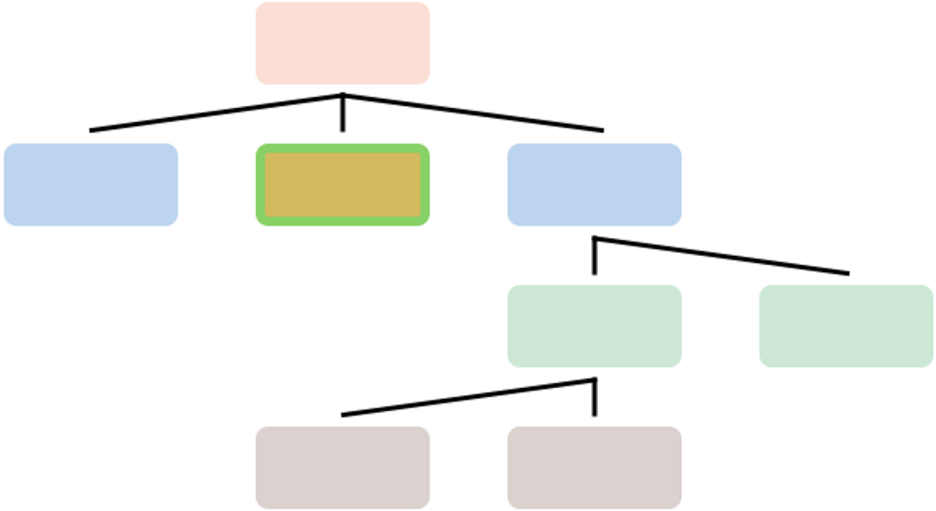
@hitherejoe

Semantics

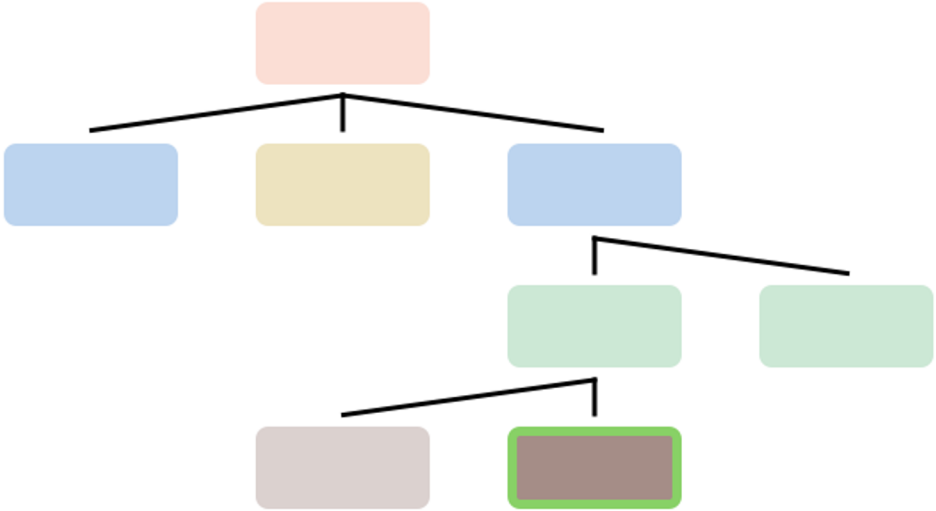








"Click to add the PS5
to your shopping cart"



“Stock notifications are currently off. Click to enable stock notifications”

```
[
 text = "Buy"
 contentDescription = "Buy the PS5"
 role = Role.Button
 selected = false
]
```

```
[
 text = "Buy"
 contentDescription = "Buy the PS5"
 role = Role.Button
 selected = false
]
```

```
[
 text = "Buy"
 contentDescription = "Buy the PS5"
 role = Role.Button
 selected = false
]
```

Button @composable

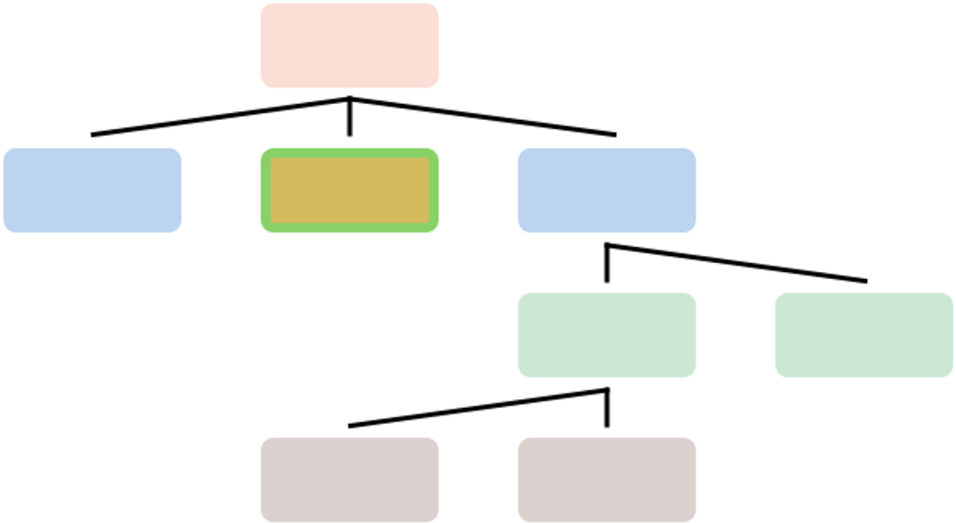
```
[
 text = "Buy the PS5"
 role = Role.Button
]
```

Icon @composable

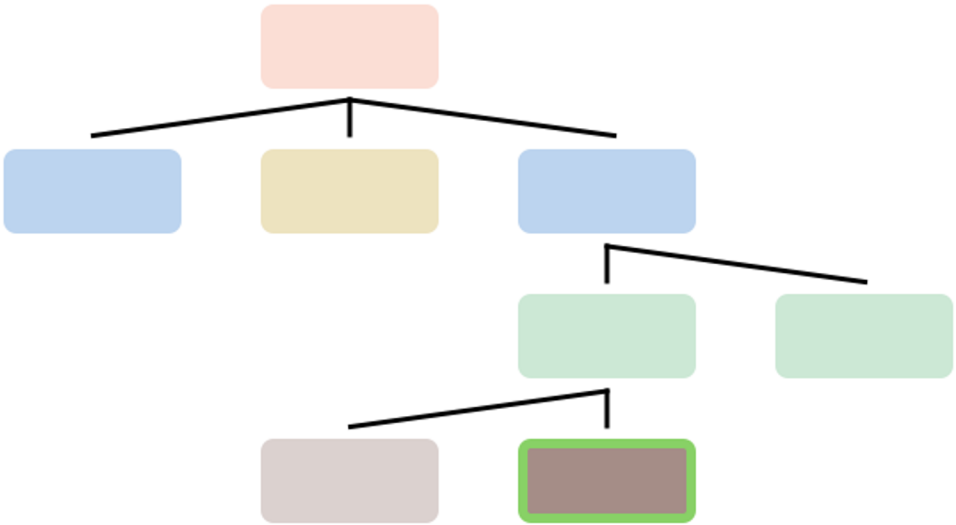
```
[
 role = Role.Image
]
```

Switch @composable

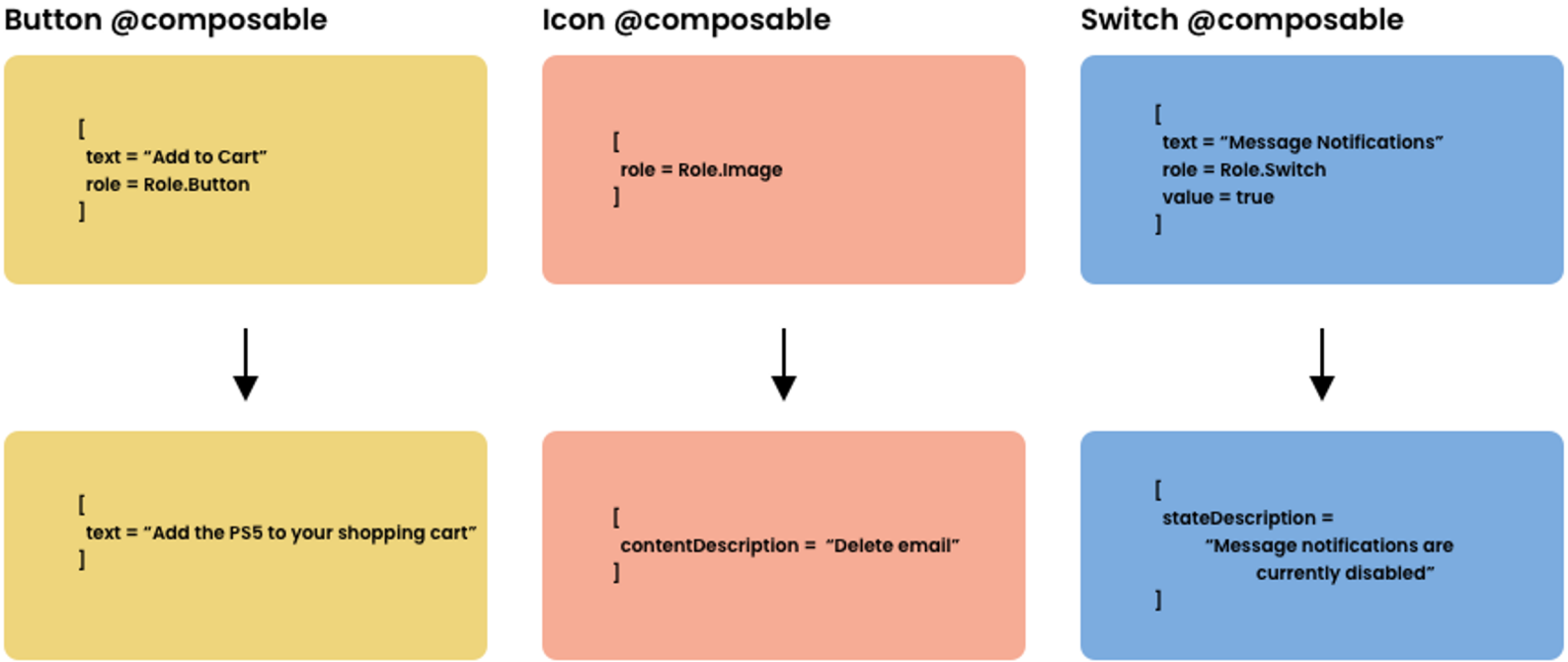
```
[
 text = "Message Notifications"
 role = Role.Switch
 value = true
]
```



“Image”



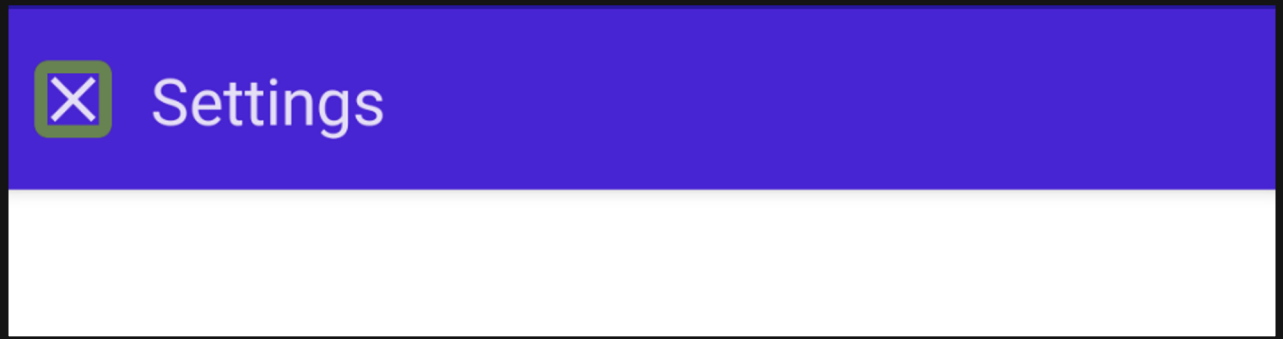
“ ”


Button @composable
[
text = "Add to Cart"
role = Role.Button
]
[
text = "Add the PS5 to your shopping cart"
]
Icon @composable
[
role = Role.Image
]
[
contentDescription = "Delete email"
]
Switch @composable
[
text = "Message Notifications"
role = Role.Switch
value = true
]
[
stateDescription =
"Message notifications are
currently disabled"
]

Approaching Accessibility

Touch Targets

```
Icon(
    modifier = Modifier.clickable {
                // Handle click
    },
    imageVector = Icons.Default.Close,
    contentDescription = stringResource(R.string.cd_close_settings)
)
```


Settings

```
Icon(
    modifier = Modifier
        .clickable {
            // Handle click
        }
        .padding(18.dp)
        .size(30.dp),
    imageVector = Icons.Default.Close,
    contentDescription =
        stringResource(R.string.cd_close_settings)
)
```


Settings

```
IconButton(onClick = {
    // Handle click
}) {
    Icon(
        imageVector = Icons.Default.Close,
        contentDescription =
            stringResource(R.string.cd_close_settings)
    )
}
```


Settings

Touch Targets

Dos

- Ensure touchable elements have a size of at least 48dp

Touch Targets

Dos

- Ensure touchable elements have a size of at least 48dp

- Default to using material implementations where possible

Touch Targets

Dos

- Ensure touchable elements have a size of at least 48dp
- Default to using material implementations where possible

Don’ts

- Sacrifice touchable areas for design requirements

Touch Targets

Dos

- Ensure touchable elements have a size of at least 48dp
- Default to using material implementations where possible

Don’ts

- Sacrifice touchable areas for design requirements
- Reduce / override default touch targets provided by material components

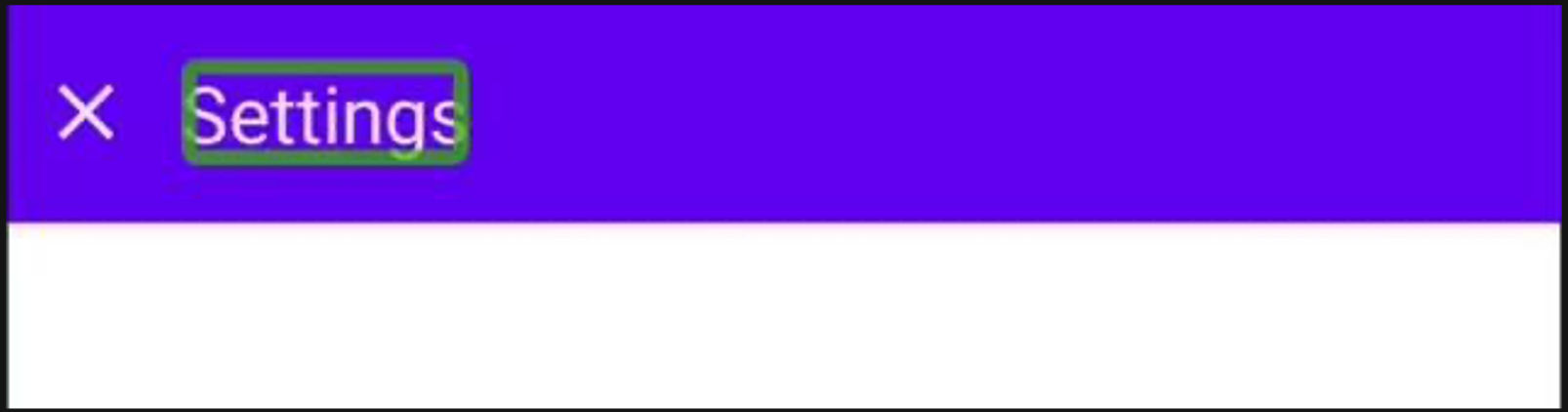
@hitherejoe

Content Descriptions

```
Icon(
    imageVector = Icons.Default.Close,
    contentDescription = null
)
```


Settings

```
Icon(
    imageVector = Icons.Default.Close,
    contentDescription =
        stringResource(id = R.string.cd_close_settings)
)
```


Settings

```
Canvas(
    modifier = Modifier
        .semantics {
            contentDescription = “Add to cart”
        },
    onDraw = {
        drawRoundRect(...)
        drawLine(...)
        drawLine(...)
    }
)
```

Content Descriptions

Dos

- Use the content description to provide **meaningful descriptions** for composables

Content Descriptions

Dos

- Use the content description to provide **meaningful descriptions** for composables

- Use null descriptions when composables are purely decorative

Content Descriptions

Dos

- Use the content description to provide **meaningful descriptions** for composables
- Use null descriptions when composables are purely decorative

Don’ts

- Default to applying null to content description arguments

Content Descriptions

Dos

- Use the content description to provide **meaningful descriptions** for composables
- Use null descriptions when composables are purely decorative

Don’ts

- Default to applying null to content description arguments
- Use content descriptions for the sake of having content descriptions

Click Labels

```
Card(
    modifier = Modifier
        .fillMaxWidth()
        .clickable {
            handleCardClick()
        }
) { ... }
```

```
Card(
    modifier = Modifier
        .fillMaxWidth()
        .clickable(
            onClickLabel = "Open Item One Article"
        ) {
            handleCardClick()
        }
) { ... }
```

Click Labels

Dos

- Use click labels to **provide interaction descriptions** for composed content

Click Labels

Dos

- Use click labels to **provide interaction descriptions** for composed content

- If also using content descriptions, ensure these compliment each other

Click Labels

Dos

- Use click labels to **provide interaction descriptions** for composed content
- If also using content descriptions, ensure these compliment each other

Don’ts

- Rely on content descriptions for action descriptions

Click Labels

Dos

- Use click labels to **provide interaction descriptions** for composed content
- If also using content descriptions, ensure these compliment each other

Don’ts

- Rely on content descriptions for action descriptions
- Modify text semantics to account for actions

Text Semantics

```
Text(text = "£60 p/m")
```

```
Text(
    modifier = Modifier.semantics {
      text = AnnotatedString("£60 per month")
    },
    text = "£60 p/m"
)
```

Text Semantics

Dos

- Use text semantics to **improve the representation** of composed content

Text Semantics

Dos

- Use text semantics to **improve the representation** of composed content

Don’ts

- Use the other semantic properties to replicate the same behaviour

Text Semantics

Dos

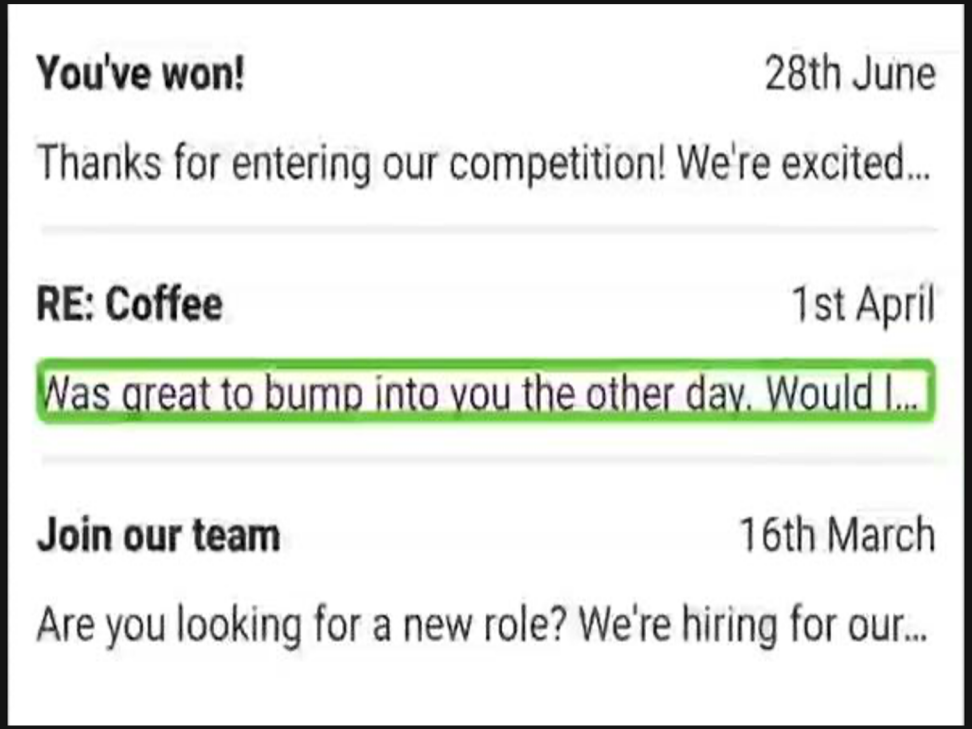
- Use text semantics to **improve the representation** of composed content

Don’ts

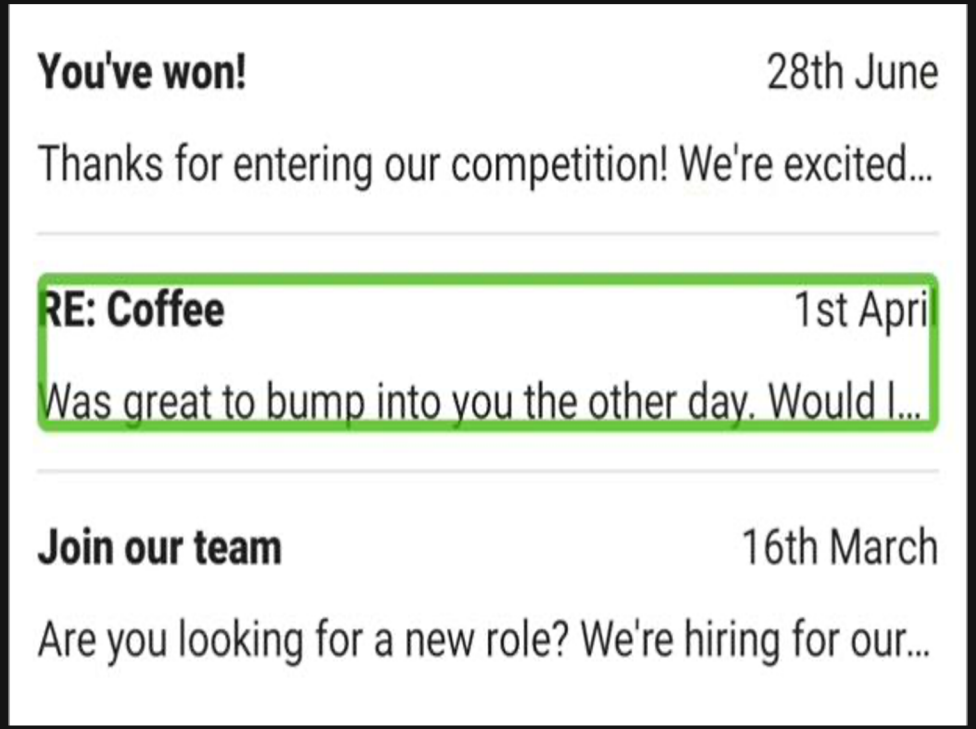
- Use the other semantic properties to replicate the same behaviour
- Use text semantics to remove composable representation

Merge Descendents

```
Column {
    Row {
        Text()
        Text()
    }
    Text()
}
```


You've won!
28th June
Thanks for entering our competition! We're excited...
RE: Coffee
1st April
Was great to bump into you the other day. Would I...
Join our team
16th March
Are you looking for a new role? We're hiring for our...

```
Column(
    modifier = Modifier.semantics(
        mergeDescendants = true
    ) { }
) {
    Row {
        Text()
        Text()
    }
    Text()
}
```


You've won!
28th June
Thanks for entering our competition! We're excited...
RE: Coffee
1st Apri
Was great to bump into you the other day. Would l...
Join our team
16th March
Are you looking for a new role? We're hiring for our...

Merge Descendants

Dos

- Use merge descendants to **group related content** into a single descendant

Merge Descendants

Dos

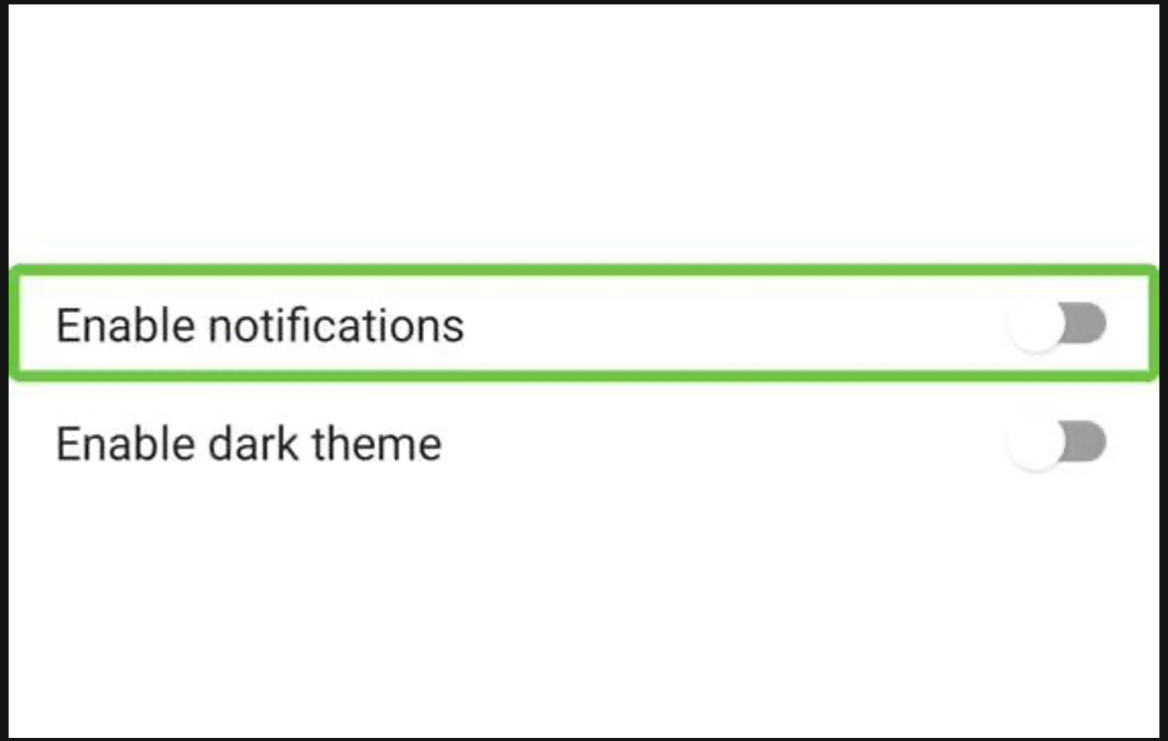
- Use merge descendants to **group related content** into a single descendant

Don’ts

- Use merge descendants for the sake of simplifying navigation between descendants

State Descriptions

```
Row(
    modifier = Modifier
        .toggleable(
            value = enabled.value,
            onValueChange = {
                enabled.value = !enabled.value
            },
            role = Role.Switch
        )
)
```


Enable notifications
Enable dark theme

```
val description = if (enabled) {
    stringResource(R.string.enabled)
} else stringResource(R.string.not_enabled)

Row(
    modifier = Modifier
        .semantics {
            stateDescription = description
        }
        .toggleable(
           ...
        )
)
```

State Descriptions

Dos

- Use state descriptions to **improve the state representation** of composed content

State Descriptions

Dos

- Use state descriptions to **improve the state representation** of composed content

Don’ts

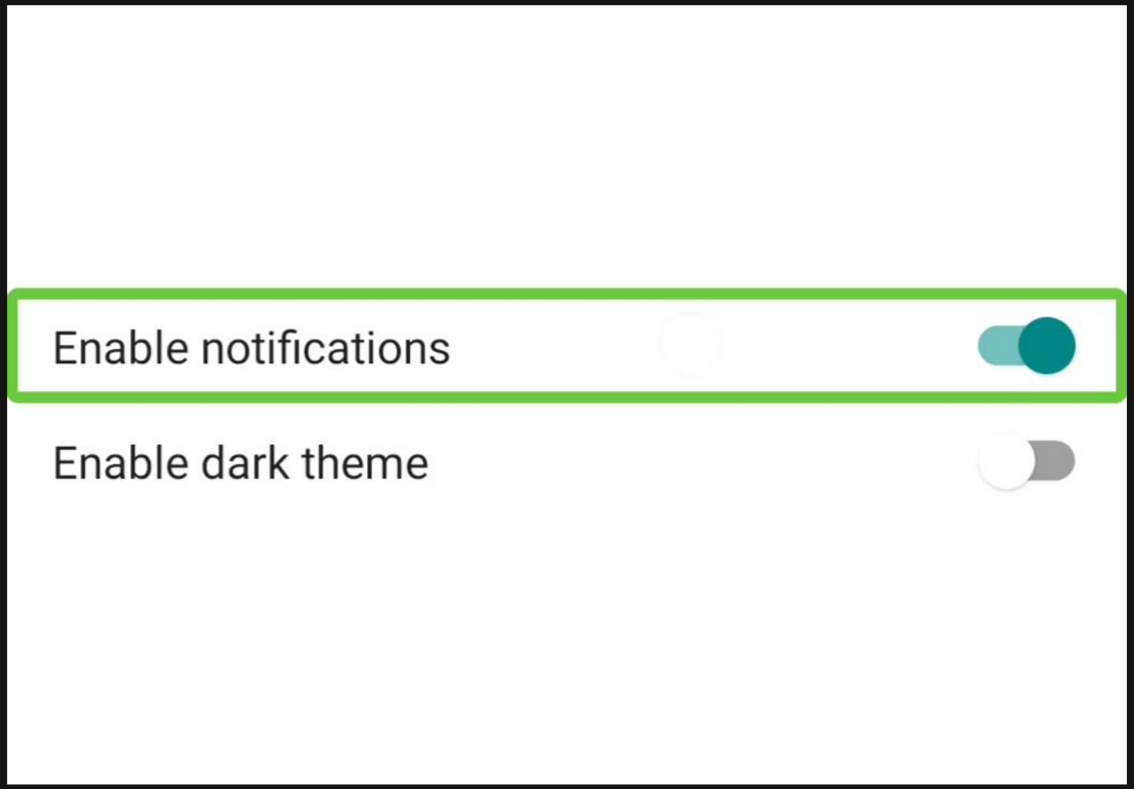
- Use state descriptions to describe actions for a component

@hitherejoe

Toggle Components

```
Row {
    val enabled = remember { mutableStateOf(false) }
    Text(...)
    Switch(
        checked = enabled.value,
        onCheckedChange = {
            enabled.value = !enabled.value
        }
    )
}
```

```
val enabled = remember { mutableStateOf(false) }
Row(
    modifier = Modifier
        .toggleable(
            value = enabled.value,
            onValueChange = {
                enabled.value = !enabled.value
            },
            role = Role.Switch
        )
) {
    Text(...)
    Switch(
        checked = enabled.value,
        onCheckedChange = null
    )
}
```


Enable notifications
Enable dark theme

Toggle Components

Dos

- Lift out toggleable state to the parent container of a toggle component

Toggle Components

Dos

- Lift out toggleable state to the parent container of a toggle component

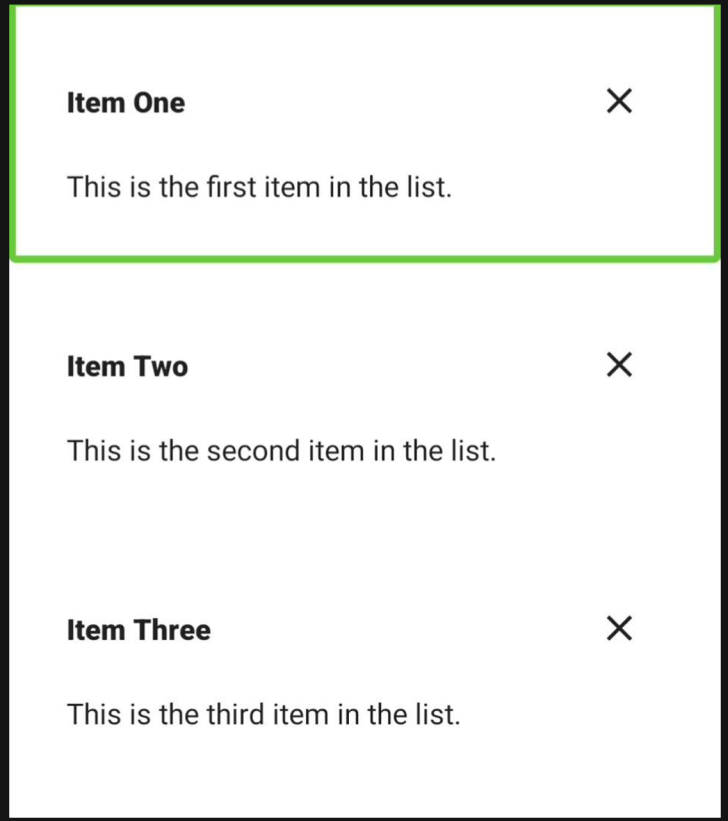
Don’ts

- Forget to set the role for the toggleable component!

@hitherejoe

Accessibility Actions

```
Column(
    modifier = Modifier
        .clickable(
            onClickLabel =
                stringResource(R.string.cd_read_email)
        ) { ... }
) {
    Row(...) {
        IconButton(
            onClick = { ... }
        ) {
            Icon(
                imageVector = Icons.Default.Close,
                contentDescription =
                    stringResource(R.string.cd_delete_email)
            )
        }
        ...
    }
)
```


Item One
This is the first item in the list.
Item Two
This is the second item in the list.
Item Three
This is the third item in the list.

```
Column(
    modifier = Modifier.semantics {
        customActions = listOf(
            CustomAccessibilityAction(label = stringResource(R.string.cd_read_email)) {
                // handle action
                true
            },
            CustomAccessibilityAction(label = stringResource(R.string.cd_delete_email)) {
                true
            }
        )
    }
) {
    Row(...) {
        Text(...)
        IconButton(
            modifier = Modifier.clearAndSetSemantics { },
            onClick = { ... }
        ) {
            Icon(
                imageVector = Icons.Default.Close,
                contentDescription = stringResource(R.string.cd_delete_email)
            )
        }
    }
}
```

Accessibility Actions

Dos

- Use accessibility actions to provide access to actions on within a list of items

Accessibility Actions

Dos

- Use accessibility actions to provide access to actions on within a list of items

Don’ts

- Disregard accessibility for the UI components that trigger those actions

@hitherejoe

Headings

```
Text(
    text = "Learn Jetpack Compose",
    fontWeight = FontWeight.Bold
)
```

Learn Jetpack Compose

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean in turpis dolor. Aliquam quis nunc id felis faucibus eleifend id ut sem. Etiam nec metus mattis ante suscipit egestas vel non sem. Phasellus commodo vulputate diam eget congue. Curabitur ut nulla felis. Duis rutrum tincidunt rhoncus. Nam ut arcu feugiat, ultrices sapien sed, imperdiet erat. Proin porttitor, diam et volutpat tempor, dolor est sagittis odio, eu aliquet ex lacus vitae ligula. Curabitur et venenatis justo

What is Jetpack Compose?

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean in turpis dolor. Aliquam quis nunc id felis faucibus eleifend id ut sem. Etiam nec metus mattis ante suscipit egestas vel non sem. Phasellus commodo vulputate diam eget congue. Curabitur ut nulla felis. Duis rutrum tincidunt rhoncus. Nam ut arcu feugiat, ultrices sapien sed, imperdiet erat. Proin porttitor, diam et volutpat tempor, dolor est sagittis odio, eu aliquet ex lacus vitae ligula. Curabitur et venenatis justo

Let's learn about state

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean in turpis dolor. Aliquam quis nunc id felis faucibus eleifend id ut sem. Etiam nec metus mattis ante suscipit egestas vel non sem. Phasellus commodo vulputate diam eget congue. Curabitur ut nulla felis. Duis rutrum tincidunt rhoncus. Nam ut arcu feugiat, ultrices sapien sed, imperdiet erat. Proin porttitor, diam et volutpat tempor, dolor est sagittis odio, eu aliquet ex lacus vitae ligula. Curabitur et venenatis justo

```
Text(
    modifier = Modifier.semantics {
        heading()
    },
    text = "Learn Jetpack Compose",
    fontWeight = FontWeight.Bold
)
```

Headings

Dos

- Use headings to breakup content into navigable sections

Headings

Dos

- Use headings to breakup content into navigable sections

- Think about how you structure the content to utilise visual heading

Headings

Dos

- Use headings to breakup content into navigable sections
- Think about how you structure the content to utilise visual heading

Don’ts

- Use headings for elements that are not section breaks

Headings

Dos

- Use headings to breakup content into navigable sections
- Think about how you structure the content to utilise visual heading

Don’ts

- Use headings for elements that are not section breaks
- Use headings for visual decorations

@hitherejoe

Custom Composables

```
var enabled by remember {
    mutableStateOf(false)
}
Row(
    modifier = Modifier.clickable {
        enabled = !enabled
    }
) {
    Icon(imageVector = ..., contentDescription = null)
    Text(text = stringResource(R.string.enable_feature))
}
```

```
var enabled by remember {
    mutableStateOf(false)
}
Row(
    modifier = Modifier.toggleable(
        value = enabled,
        role = Role.Checkbox,
        onValueChange = {
            enabled = !enabled
        }
    )
) {
    Icon(imageVector = ..., contentDescription = null)
    Text(text = stringResource(R.string.enable_feature))
}
```

```
Canvas(
    modifier = Modifier
        .clickable {
            // Handle click
        },
    onDraw = {
        drawRoundRect(...)
        drawLine(...)
        drawLine(...)
    }
)
```

```
val ps5ContentDescription = stringResource(R.string.buy_ps5)
Canvas(
    modifier = Modifier
        .clickable(
            onClickLabel = "Add to cart"
        ) {
            // Handle click
        }
        .semantics {
            role = Role.Button
        },
    onDraw = {
        ...
    }
)
```

What next?

Further Reading

- Google Jetpack Compose Accessibility Codelab
  - developer.android.com/codelabs/jetpack-compose-accessibility

- Compose Academy Blog
  - compose.academy/blog

- Compose Accessibility Best Practices
  - https://tinyurl.com/76shxzyw

- Understanding Semantics
  - https://tinyurl.com/24ktnfh9

compose.academy/practicaljetpackcompose

Thank you!

@hitherejoe