

To make the world's information universally accessible and useful

Android for Everyone

Accessibility in Mobile

...



SierraOBryan
S I E R R A O B R Y A N (S H E / H E R)

 @_sierraOBryan

Make Tech Inclusive.
Make Inclusive Tech.

It's a team effort.

Types of Disabilities

- Motor Impairments
 - May use a hardware device - Accessibility Switch - to control the app or accessibility menu
- Cognitive Impairments
 - May use Action Blocks to set up routines
- Visual Impairments
 - May use increased text size, Braille keyboard, or TalkBack
- Deaf and Hard of Hearing
 - May use Closed Captioning, Live Transcribe or Live Captioning

If you open your [accessibility settings](#), you'll find even more options that folks might use on their device

Permanent

Temporary

Situational

Touch



One arm



Arm injury



New parent

See



Blind



Cataract



Distracted driver

Hear



Deaf



Ear infection



Bartender

Speak



Non-verbal



Laryngitis



Heavy accent

Sierra

I don't want to touch my phone
when I'm trying to "cook"

I forget to blink when I'm
wearing contacts

I like to stay up too late watching TV
while everyone else is sleeping

I mumble

[Learn more](#)

Inclusive
A Microsoft Design Toolkit

@_sierraobryan

What are the Web Content Accessibility Guidelines and why do they matter for mobile?

[Learn more](#)

Modifiers and Semantics in Jetpack Compose

Modifiers

- Decorate or add behavior to Compose UI elements
- Categories include actions, animations, focus, behaviors, size, shape, etc
- Order of modifiers matters

[View All Modifiers](#)

```
MostComposables(  
    specificArg = value,  
    modifier = Modifier  
        .size( .. )  
        .shape( .. )  
)
```


Semantics

Compose uses semantics properties to pass information to accessibility services. Semantics properties provide information about UI elements that are displayed to the user.

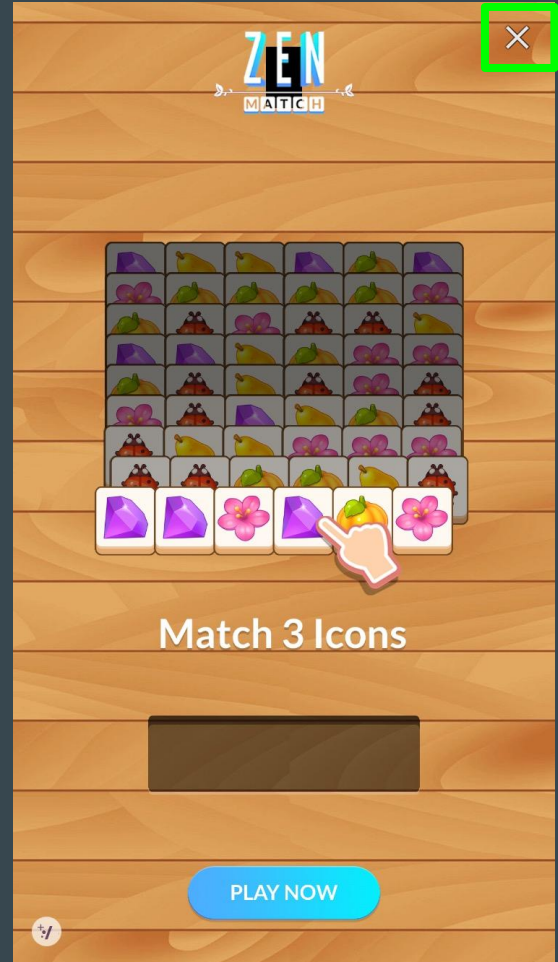
[Semantics Documentation](#)

```
Modifier.semantics(  
    mergeDescendants: Boolean,  
    properties: SemanticsPropertyReceiver.() -> Unit  
)
```

Why do I care about touch target size?

Have you ever played an free mobile game...

@_sierraobryan



Touch Targets [Learn More](#)

The recommended size for each interactive UI element's focusable area, or **touch target size**, is at least 48px by 48px (focusable != visible)

```
<EditText
    android:id="@+id/element_id"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:minHeight="48dp"
    ...
/>
```

Touch Targets [Learn More](#)

The recommended size for each interactive UI element's focusable area, or **touch target size**, is at least 48px by 48px (focusable != visible)

```
TextField(  
    value = value,  
    modifier = modifier  
        .fillMaxWidth()  
        .heightIn(min = 48.dp)  
)
```

Setting minimums allows the UI to be flexible for different screen sizes, localization, and changing text sizes.

```
heightIn(min : Dp, max: Dp)
```

Touch Targets [Learn More](#)

The recommended size for each interactive UI element's focusable area, or **touch target size**, is at least 48px by 48px (focusable != visible)

```
<ImageView
    android:layout_width="48dp"
    android:layout_height="48dp"
    android:padding="8dp"
    android:onClick="onClick"
    ...
/>
```

Touch Targets [Learn More](#)

The recommended size for each interactive UI element's focusable area, or **touch target size**, is at least 48px by 48px (focusable != visible)

```
Icon(  
    imageVector = icon,  
    contentDescription = contentDescription,  
    modifier = Modifier  
        .clickable { ... }  
        .padding(8.dp)  
        .size(36.dp)  
)
```

Different order, same result

```
Modifier  
    .clickable { }  
    .size(48.dp)  
    .padding(8.dp)
```

Touch Targets [Learn More](#)

The recommended size for each interactive UI element's focusable area, or **touch target size**, is at least 48px by 48px (focusable != visible)

```
IconButton(  
    onClick = { onClick() },  
    modifier = Modifier  
) { content() }
```

```
modifier.clickable(...).then(IconButtonSizeModifier)
```

```
val IconButtonSizeModifier = Modifier.size(48.dp)
```

Taking it further [Learn More](#)

Adding custom click labels

Replace “**double tap to activate**” with “**double tap to navigate home**”

```
Row (
    modifier = Modifier
        .fillMaxWidth()
        .clickable (
            onClickLabel = "navigate home",
            role = Role.Button
        ) { onClick() }
) { content() }
```

Role is a **Semantic Property**.


```
Row(  
    modifier = Modifier  
        .clickable { openEmail() }  
        .semantics {  
            customActions = listOf(  
                CustomAccessibilityAction(  
                    label = "Mark as read",  
                    action = { markAsRead() }  
                ),  
                CustomAccessibilityAction(  
                    label = "Delete Email",  
                    action = { deleteEmail() }  
                )  
            )  
        }  
    ) { content() }
```

Adding Custom Actions [Learn More](#)

```
Row(  
    modifier = Modifier  
        .clickable { openEmail() }  
        .semantics {  
            customActions = listOf(...)   
        }  
)  
{  
    Button(  
        onClick = { ... }  
        modifier = Modifier.clearAndSetSemantics { }  
    ) { }  
}
```

Although action is defined in the semantics, we also need the same action in the Button args

Adding Custom Actions [Learn More](#)

Why do I care about content descriptions?

You might be already using them without thinking about it!

@_sierraobryan



Accessibility Labels [Learn more](#)

Unique, Localized, Concise, Descriptive

@Composable

```
fun Image(  
    painter: Painter,  
    contentDescription: String?,  
    modifier: Modifier = Modifier,  
    alignment: Alignment = Alignment.Center,  
    contentScale: ContentScale = ContentScale.Fit,  
    alpha: Float = DefaultAlpha,  
    colorFilter: ColorFilter? = null  
)
```

There's a lot to think about with Labels

Does it need a label? It depends but generally...

Text: No

Button: No

TextField: Include a hint

ImageButton: Yes

Image: Maybe

Can it be skipped?

Is it a decorator?

Should it be grouped?

Do they make
more sense
together?

How will it be read?

Does it include
numbers or
abbreviations?

Accessibility Labels [Learn more](#)

Localized, Concise, descriptive

```
Text(  
    text = "WCAG",  
    modifier = Modifier.semantics {  
        contentDescription =  
            "Web content Accessibility Guidelines"  
    }  
)
```

A [TtsSpan](#) can help! A TtsSpan is a special type of span that can pass in metadata to give contextual information about the string. This information can help Text to Speech correctly pronounce a text element

Accessibility Labels [Learn more](#)

Unique, Localized, Concise, Descriptive

```
Column(  
    modifier = Modifier.weight(1f)  
        .semantics(mergeDescendants = true) {}  
) {  
    Text(text = commit.commitMessage)  
    Text(text = "Author: ${commit.author}")  
    Text(text = "Date: ${commit.date}")  
}
```

We can also change the focus order by using the FocusRequester with the FocusOrder Modifier.

Accessibility Labels [Learn more](#)

Unique, Localized, Concise, Descriptive

```
Column(modifier = Modifier
    .fillMaxWidth()
    .semantics {
        contentDescription = "Container"
    }
) {
    Text(text = "Text")
    TextField(label = { Text(text = "Label") })
}
```


Accessibility Labels [Learn more](#)

Unique, Localized, Concise, Descriptive

```
Text(  
    text = "WCAG Overview",  
    modifier = Modifier.semantics {  
        heading()  
    }  
)
```

Accessibility Labels [Learn more](#)

Unique, Localized, Concise, Descriptive

```
TextField(  
  ...  
  placeholder = { Text(text = "Placeholder") },  
  label = { Text(text = "Label") },  
  ...  
)
```

We should only use placeholder
OR label on a single TextField

Color Contrast [Learn more](#)

AA Compliance requires at least a requires **4.5 : 1** for regular text and **3 : 1** for large text

Slides: 5.88 : 1

7.62 : 1

3.92 : 1

2.38 : 1

How do I meet these requirements and stick to my theme?

Use [Material Color Palettes](#)!

How do I check my colors? There are lots of [tools](#) online!

WCAG 3 will use a new color contrast method called the Advanced Perceptual Contrast Algorithm.

[Learn More](#)

I'm accessible text!

I am also accessible text!

But I am not accessible text

Testing [Learn more](#)

```
@RunWith(AndroidJUnit4::class)
@LargeTest
class MyWelcomeWorkflowIntegrationTest {
    init {
        AccessibilityChecks.enable()
    }
}
```

Testing [Learn more](#)

In Jetpack Compose, semantics are used for both accessibility and testing.

We can use SemanticsMatcher to build our UI tests.

Consistency

What's next?

The Web Content Accessibility Guidelines v3 are set to be published in 2023

what does that mean for mobile?

Draft 1 was published in late Jan 2021 and continues to have updates

where do I learn more?

Android Accessibility by Tutorials! by Victoria Gonda

I'm just getting started with Accessibility + Compose

what should I do next?

Jetpack Compose: Accessibility and the new Code Lab

To make the world's information universally accessible and useful



Thank you!

...

Where to find me?



@_sierraObryan

sierraobryan.dev

