


# Accessibility & Material Design

## Material Design [↗](#)

Material 3 is the latest version of Google's open-source design system. Design and build beautiful, usable products with Material 3.

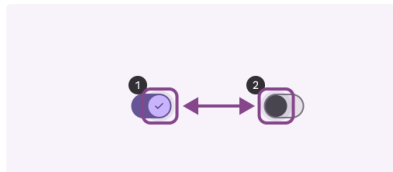
 [Material Design](#) includes in-depth UX guidance and UI component implementations for Android, Flutter, and the Web.

### Assistive technology

Assistive technology helps increase, or improve the functional capabilities of individuals with **disabilities**. People can live more independently by engaging with technology through devices like **keyboards**, **screen readers**, and braille displays, as well as tracking input, magnifiers (tools used to enlarge objects), and voice input.

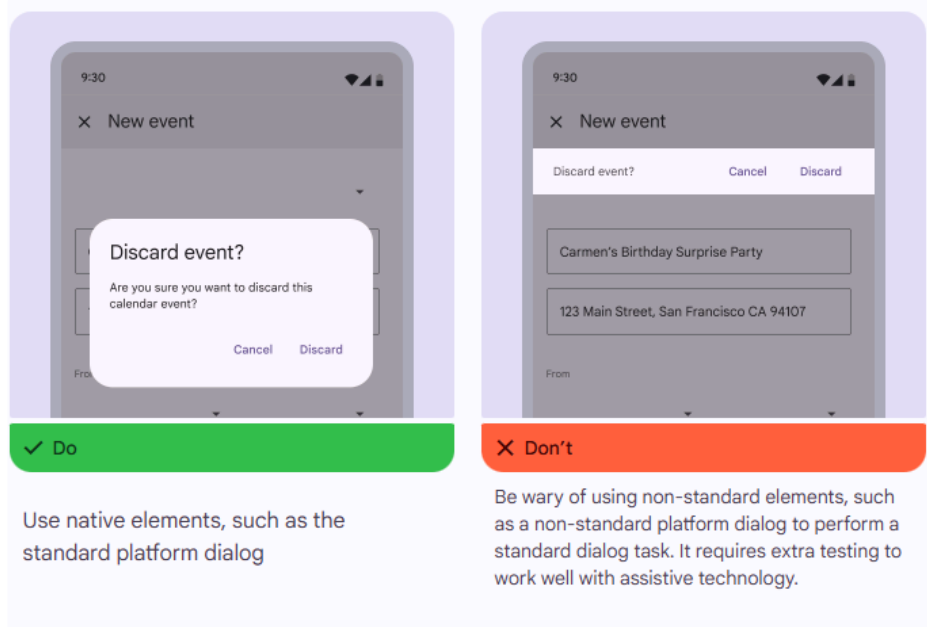
### Accessibility markup [↗](#)

Accessibility markup is an integral part of creating documentation for design specs.



### Implementing accessibility [↗](#)

By using standard platform controls and semantic HTML (on the web), apps automatically contain the markup and code needed to work well with a platform's assistive technology.



## Designing [🔗](#)

### Color & contrast [🔗](#)

Color can help communicate mood, tone, and critical information. Primary, secondary, and accent colors ([accent colors](#)) be selected to support usability. Sufficient color contrast between elements can help users with low vision see and use your app.

Higher contrast makes the imagery easier to see.

The contrast ratio between a color and its background ranges from 1-21 based on its luminance (the intensity of light emitted) according to the World Wide Web Consortium (W3C).

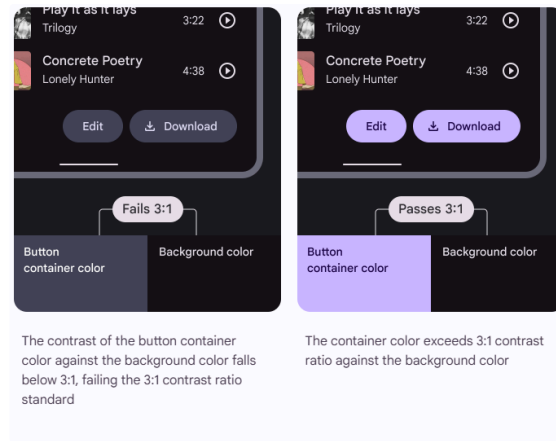
The W3C recommends the following contrast ratios for body text and image text

| Text type  | Color contrast ratio         |
|--|------------------------------|
| Large text (at 14 pt bold/18 pt regular and up) and graphics | 3:1 against the background   |
| Small text   | 4.5:1 against the background |

Disabled states do not need to meet contrast requirements.

### Clustering elements [🔗](#)

Some non-text elements, such as button containers, should meet a contrast ratio of 3:1 between their container color and the color of their background.



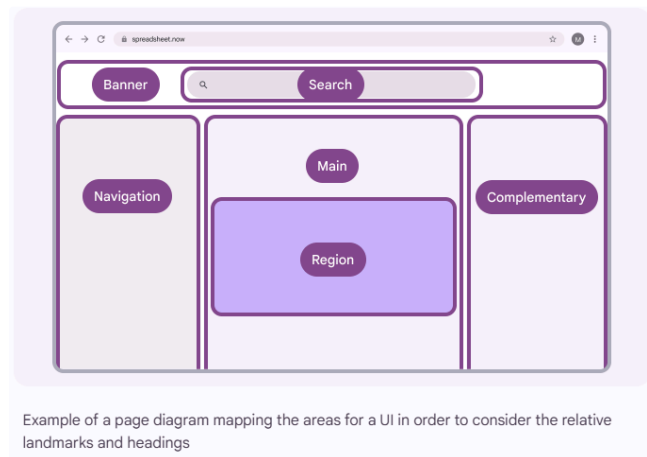
[Learn more about color contrast for accessibility](#)

## Web landmarks and headings [↗](#)

Web landmarks and headings are essential for improving navigation and comprehension, especially for users relying on assistive technologies (AT) like screen readers. These technologies process content in a linear and hierarchical manner, making clear structures critical for accessibility.

### 1. Identifying landmarks and headings [↗](#)

Landmarks are large blocks of content that establish the high-level structure of your layout.



## Target sizes [↗](#)

Touch and pointer target sizes

**touch target** refers to the **clickable or tappable area** of an interactive element on the screen. For example, an icon may appear to be 24 x 24dp, but the padding surrounding it comprises the full 48 x 48dp touch target.

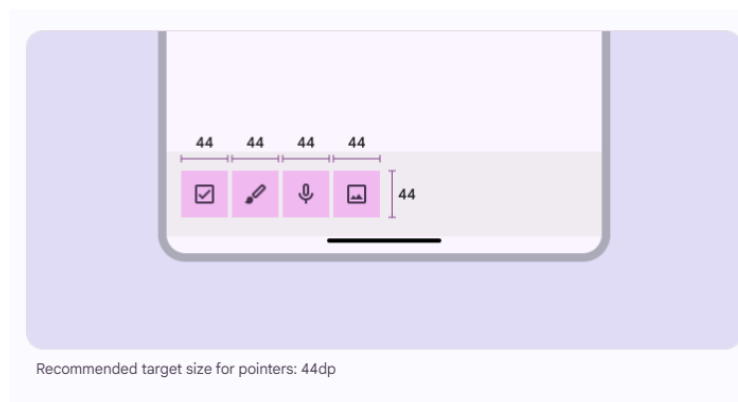
For most platforms, consider making touch targets at least 48 x 48dp. ( "dp" stands for **density-independent pixel**)



Larger touch targets make it easier for users to accurately tap the intended element, reducing accidental touches.

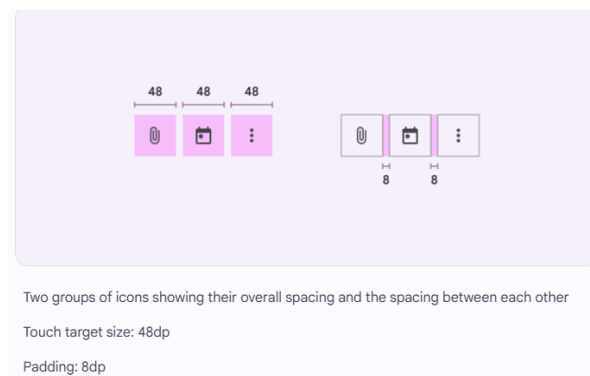
### Pointer targets

Pointer targets are similar to touch targets, but are implemented by motion-tracking pointer devices such as a **mouse** or a stylus. Consider making pointer targets minimums 44 x 44dp.



### Target spacing

In most cases, targets separated by 8dp of space or more promote balanced information density and usability



# Flow [🔗](#)

## Focus order & key traversal [🔗](#)

To support navigation by keyboard, screen reader, or other assistive technology, goals should be achievable by using **tab, arrow, and other common navigation keys**.

Simplify your flows by:

strategically **ordering** tab stops

reducing overall page complexity

### Focus Order

- **Definition:** Focus order is the sequence in which interactive elements (like buttons and links) are selected when you navigate **using the keyboard** (usually by pressing the tab key).

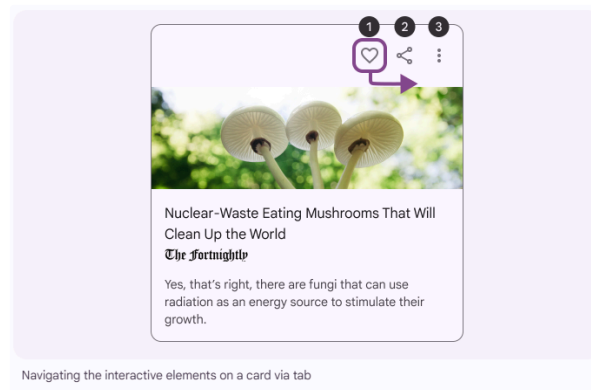
**Example:** When you press the tab key, the focus might move from the header menu, to a search bar, to a button, following a logical flow.

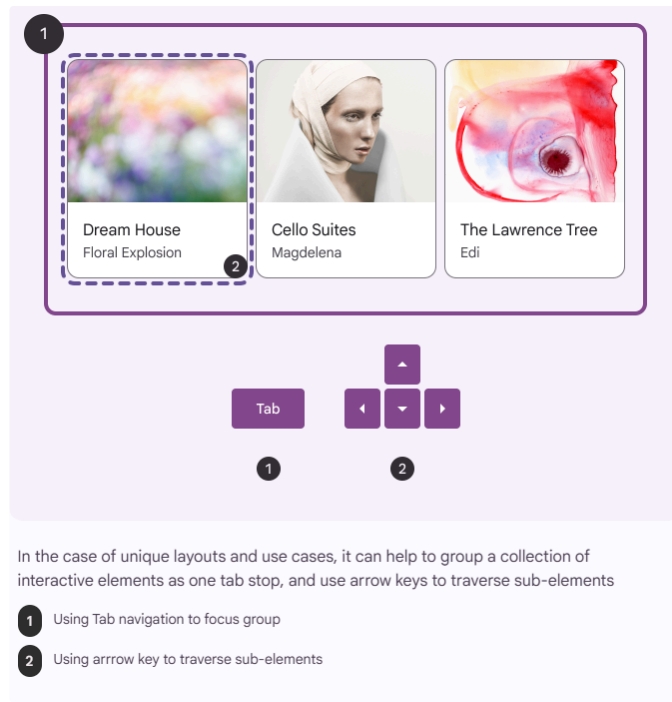
### key traversal [🔗](#)

**Definition:** Key traversal refers to the way you move focus between interactive elements using keyboard keys (e.g., tab, arrow keys).

**Example:**

- **Tab Key:** Moves focus to the next interactive element.
- **Shift + Tab:** Moves focus to the previous interactive element.
- **Arrow Keys:** Move focus within menus or lists.





## Use defaults [🔗](#)

The default order follows the DOM and generally flows from left to right; top to bottom.

Determining user flows

### 1. Group product use cases

Group product use cases into primary and secondary user journeys.

### 2. Define initial focus and component-level focus

- **Initial Focus:** Set where the user's interaction **starts** when a screen loads or a component with multiple parts is used.
- **Focus Movement:** Generally, the **tab** key is used to move the focus between interactive elements like buttons or links.

## Keyboard shortcuts [🔗](#)

Keyboard shortcuts should use a combination of **two or more keys** by default.

Include a tutorial, list, or help center page of all custom keyboard shortcuts in your product. For example, Cmd+Z (Ctrl+Z) to undo deleting an event in Google Calendar.

If a keyboard shortcut is activated with a **single key**, provide users with a way to take at least one of these actions:

1. (Most preferred) **Remap:** Combine the key with another key (e.g., "Ctrl+A" instead of just "A").
2. (Preferred). **Contextual Activation** refers to enabling a keyboard shortcut only when a specific component or part of the user interface is currently in focus.

### Example 1: Text Editing Application

**Context:** In a text editor, the shortcut **Ctrl+B** is used to bold text.

**Contextual Activation:** **Ctrl+B** only works when the text editor area is focused. If the focus is on the file menu or any other non-editable area, the shortcut is inactive.

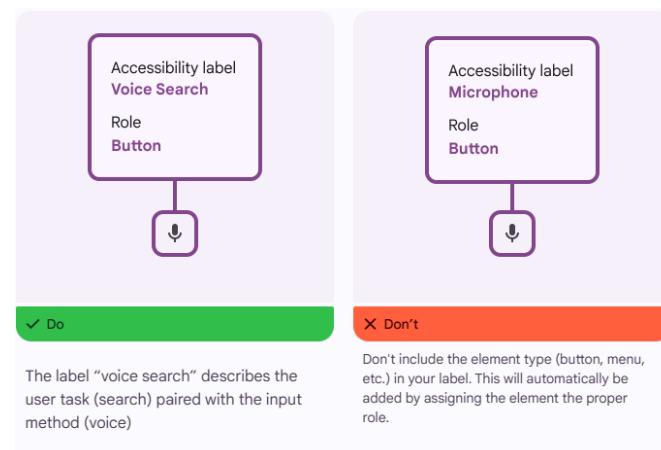
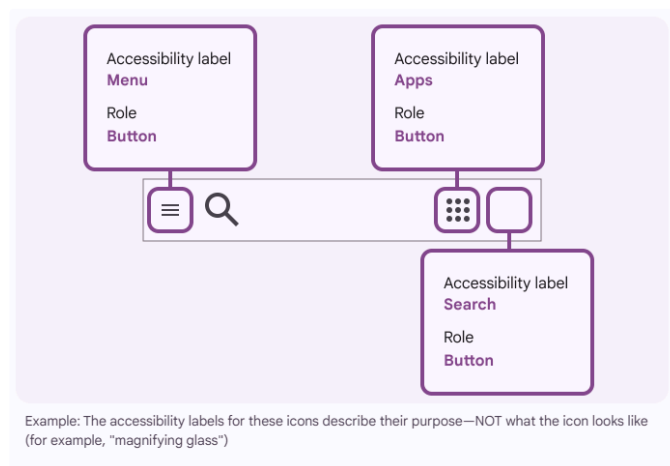
## Elements [↗](#)

### Labeling elements [↗](#)

Elements can be defined and labeled to enhance understanding of their function and reduce confusion for those navigating with assistive technology. Add accessibility labels to define roles and indicate decorative elements.

#### Visual elements that need labels

- **Interactive icons or buttons** with no visible text or not enough context in the text (for example, an edit button with a pencil icon)
- **Interactive images**
- Visual cues (including progress bars and error handling)
- Meaningful icons (such as status icons)
- Meaningful images (for example, diagrams, substantive photos, and illustrations)




## Writing and text [↗](#)

### Captions [↗](#)

Captions are the text that appear below an image. They explain contextual information about the image to help the reader understand how it relates to the content. Both sighted and screen reader users rely on captions for descriptions of images.

1



2

A DJ performs a set under lights and lasers for a crowd of screaming fans at the American EDM festival.

3

The general consensus is that artists using digital turntables are just pushing random buttons when they perform live. But is that true? In the DMC Championships it's obvious that some artists are

1

Image

2


Caption

3

Adjacent text

## Alternative text (Alt text) [🔗](#)

Alt text helps translate a visual UI into a text-based UI.

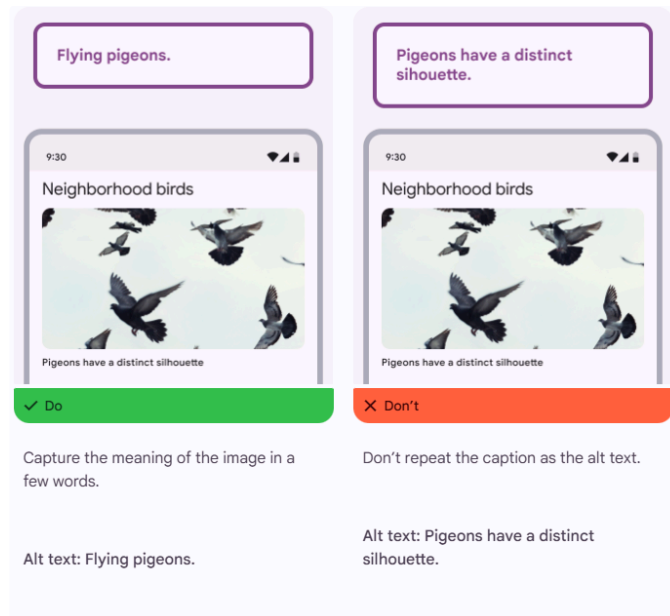


Alt text: A DJ performs a set under lights and lasers

✓ Do

Use alt text to convey what the image is showing in an informative, short phrase.  
Alt text example: A DJ performs a set under lights and lasers



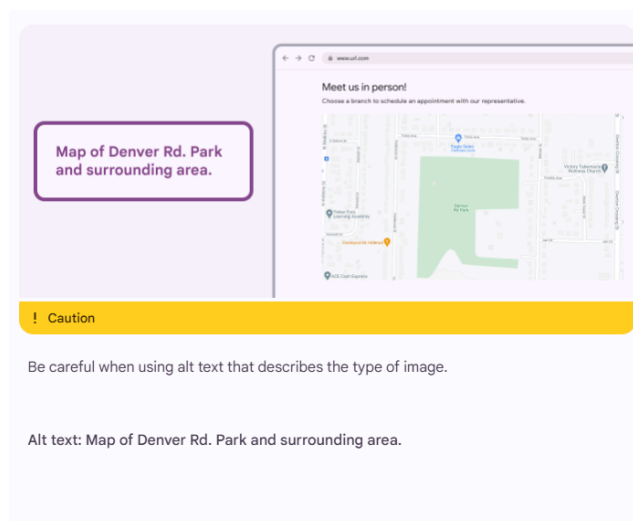


## Types of imagery [↗](#)

### When to describe the type of image [↗](#)

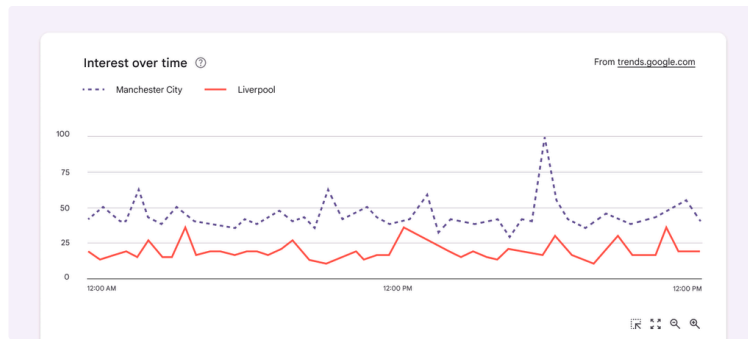
Occasionally it benefits users to name the type of image. This can include:

- Chart
- Infographic
- Map
- Graph
- Screenshot
- Headshot
- Diagram



## Charts and graphs [↗](#)

A general formula for chart alt text would be: "Summary of [data type] + [reason for showing the chart]."



Summary of interest in Manchester City, which is almost double that of Liverpool. Manchester interest peaks at 100 at 6pm. Liverpool peaks at 40 at 3am and 12:30pm.

✓ Do

Capture the meaning of the chart, along with key data points

Alt text: Summary of interest in Manchester City, which is almost double that of Liverpool. Manchester interest peaks at 100 at 6pm. Liverpool peaks at 40 at 3am and 12:30pm.

Interest in Manchester City: 12am=40, 12pm=56, 6pm=100, 9pm=45. Liverpool interest 12am=20, 12:30pm=40, 6pm=30, 9pm=20.

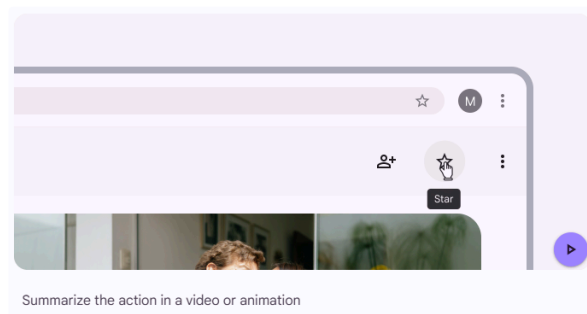
✗ Don't

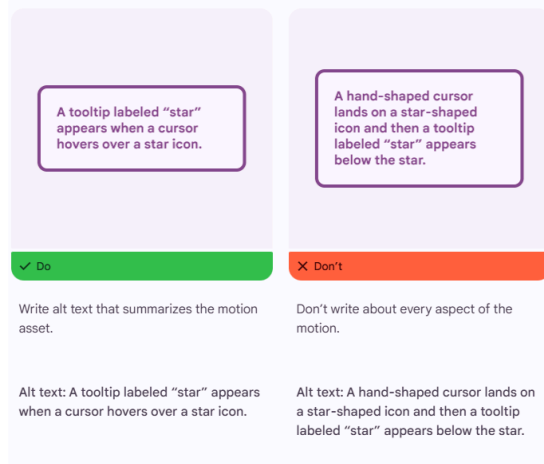
Avoid copying data points

Alt text: Interest in Manchester City: 12am=40, 12pm=56, 6pm=100, 9pm=45. Liverpool interest 12am=20, 12:30pm=40, 6pm=30, 9pm=20.

## Video and motion alt text [🔗](#)

Alt text on informative GIFs or motion assets should highlight the important points. Think of a heading or title you might give it, and that's likely your alt text.





## Global writing [↗](#)

Global writing is easier to read, understand, and translate. [🌐 Global writing – Material Design 3](#)

## Style guide [↗](#)

UX writing best practices [🌐 Style guide – Material Design 3](#) [↗](#)

Word choice [🌐 Style guide – Material Design 3](#) [↗](#)

Grammar and punctuation [🌐 Style guide – Material Design 3](#) [↗](#)

Read more about Alt [🌐 Material Design](#)

## Text Truncation (text shortening) [↗](#)

### Information Accessibility:

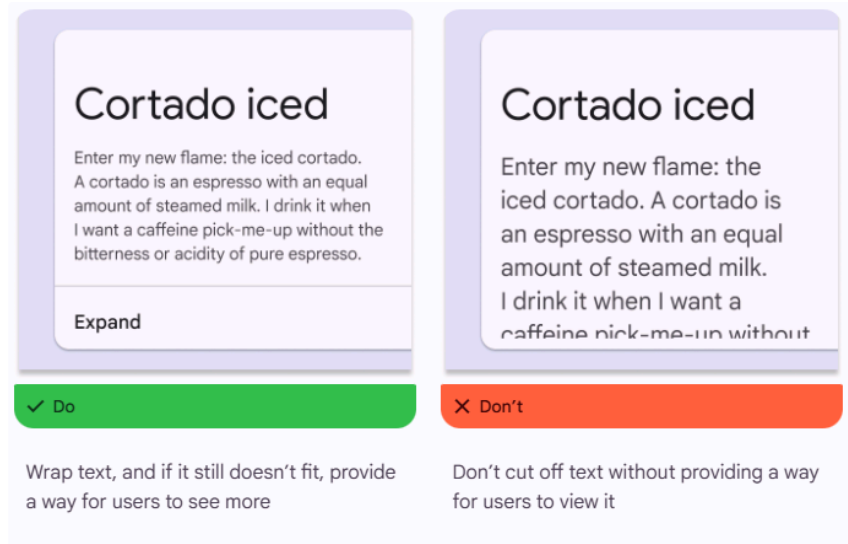
- Ensure that information is always accessible to readers, even if text gets **cut off or wrapped**.

### Background:

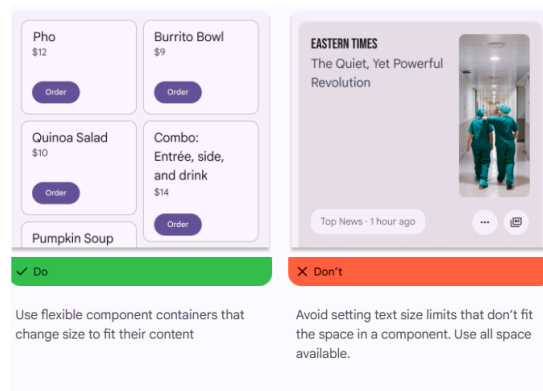
- **Adaptability:** Text should remain readable and content should not be lost if users increase text size, change spacing, or translate to longer languages.
- **Flexible Designs:** Create designs that can handle different screen sizes or zoom levels.

### Common Methods: [↗](#)

1. **Text Wrapping:** Allow text to flow onto the next line.



2. **Component Resizing:** Increase the height or width of text boxes to fit larger text. (Some components can extend vertically or horizontally for more text)



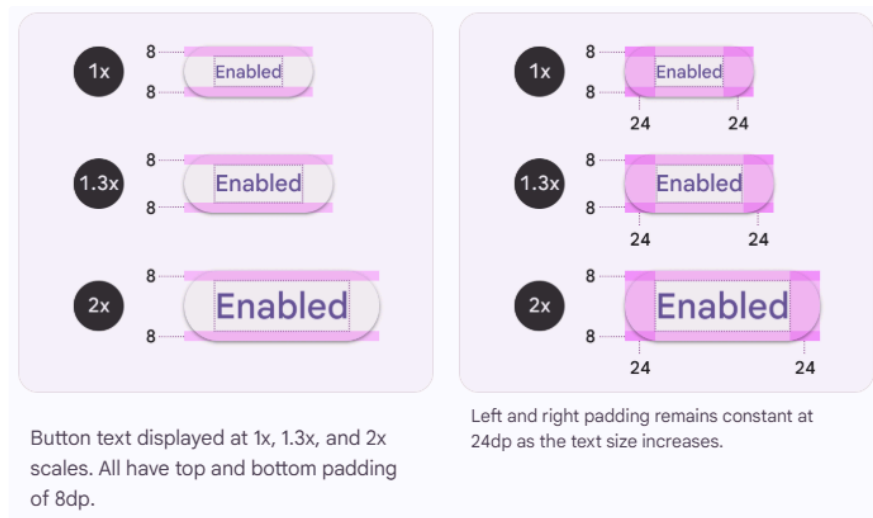
3. **Ellipses and Hover/Link:** Use "..." to indicate cut-off text, with the option to see the full text by hovering or clicking a link.

## Text resizing [🔗](#)

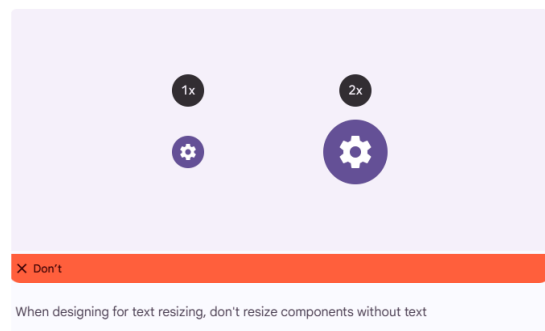
UIs should support a minimum text increase of **200%**.

**Most components behave the same when text is resized:**

- Text and line height scale up proportionally, multiplied by scale value
- Padding remains constant at 1x the default size
- Spacing between elements in a component remain constant at 1x the default size



Components that don't include text, like progress indicators, checkboxes, or radio buttons, aren't affected by text resizing.



## Methods [↗](#)

Avoid common text resizing issues by

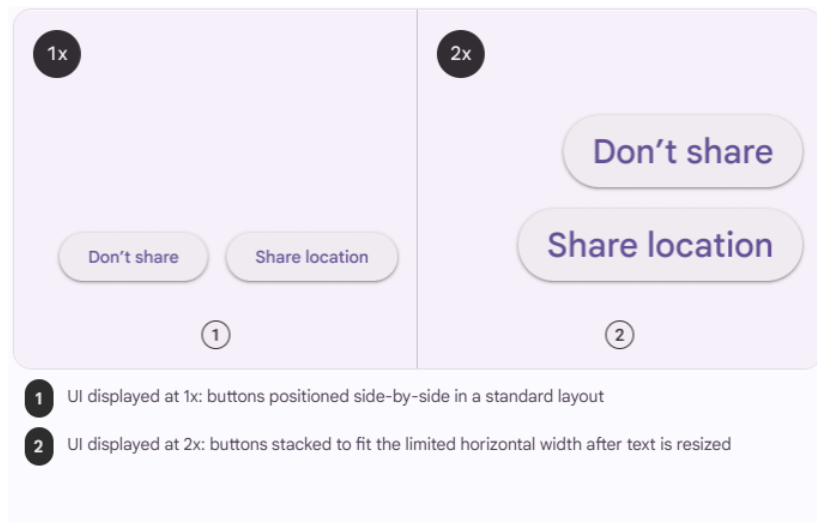
### 1. Increase container size

Resizing containers can prevent text from overlapping, clipping, or truncating.

Consider how **text** might **reflow** in a way that allows the eye to follow the end of one line to the beginning of the following line.

### 2. Reflow the layout

Consider reflowing the layout, especially when components grow very long. To accommodate larger text, components can be stacked on top of one another, rather than fixed side-by-side.



### 3. Enable content to scroll

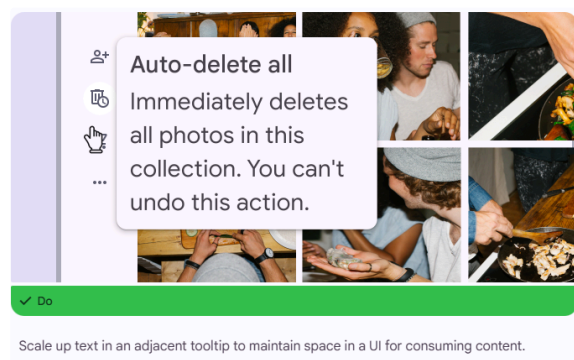
When long strings of enlarged text don't fit on one screen, consider adding a scrollbar to provide access to more content.

Vertical scrolling is preferable to horizontal. Users should only be asked to scroll in one direction, rather than both vertically and horizontally.

### 4. Use touch & hold tooltips to provide enlarged labels

Some components, such as app bars and navigation bars, position text in spaces with stricter space and character limits. In these situations, you can add a **tooltip** to display enlarged content in the UI.

**In this case**, the text size in the component remains displayed at 1x while the scaled up text is displayed in a tooltip on **touch & hold**.



**Tooltips** are the best choice for displaying enlarged text in:

Top app bar

Navigation bar

Navigation rail

Tabs, when fixed to the top of a screen and don't move off-screen upon scrolling