

---

# MATERIAL COMPONENTS

---

## TEMA 1

- Autor: Alberto Velázquez Rapado
- Asignatura: Diseño de Interfaces
- Curso: 2º DAW

## CONTENIDO

---

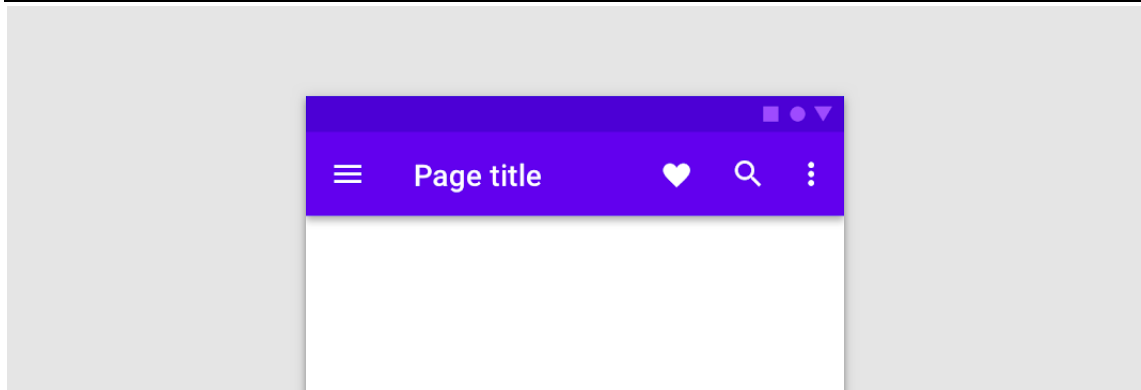
App bars: top.....	4
Usage .....	4
Principles.....	4
Types .....	4
Anatomy .....	5
Banners .....	6
Usage .....	6
Principles.....	6
Anatomy .....	7
Buttons.....	8
Usage .....	8
Principles.....	8
Types .....	8
Anatomy .....	10
Buttons: floating action button .....	11
Usage .....	11
Principles.....	11
Anatomy .....	11
Cards .....	12
Usage .....	12
Principles.....	12
Anatomy .....	12
Checkboxes .....	14
Usage .....	14
Principles.....	14
Chips.....	15
Usage .....	15
Principles.....	15
Types .....	16
Anatomy .....	18
Data Tables .....	19
Usage .....	19
Principles.....	19
Anatomy .....	20

Dialogs .....	21
Usage .....	21
Principles.....	21
Types .....	21
Anatomy .....	23
Image Lists .....	25
Usage .....	25
Principles.....	25
Types .....	25
Anatomy .....	26
Lists .....	28
Usage .....	28
Principles.....	28
Types .....	28
Anatomy .....	29
Menus.....	31
Usage .....	31
Principles.....	31
Types .....	31
Anatomy .....	32
Nagivation drawer .....	35
Usage .....	35
Principles.....	35
Types .....	36
Anatomy .....	38
Progress Indicators .....	39
Usage .....	39
Principles.....	39
Types .....	39
Anatomy .....	40
Radio Buttons .....	41
Usage .....	41
Principles.....	41
Sliders .....	42
Usage .....	42

Principles.....	42
Types .....	42
Anatomy .....	43
Snackbars .....	45
Usage .....	45
Principles.....	45
Anatomy .....	45
Switches.....	46
Usage .....	46
Principles.....	46
Tabs.....	47
Usage .....	47
Principles.....	47
Types .....	47
Anatomy .....	48
Text fields.....	49
Usage .....	49
Principles.....	49
Types .....	49
Anatomy .....	50
Tooltips .....	51
Usage .....	51
Principles.....	51

## APP BARS: TOP

---



### Usage

The top app bar provides content and actions related to the current screen. It's used for branding, screen titles, navigation, and actions.

It can transform into a contextual action bar.

### Principles

#### *Persistent*

Top app bars appear at the top of each screen in an app, and can disappear upon scroll.

#### *Guiding*

Top app bars provide a reliable way to guide users through an app.

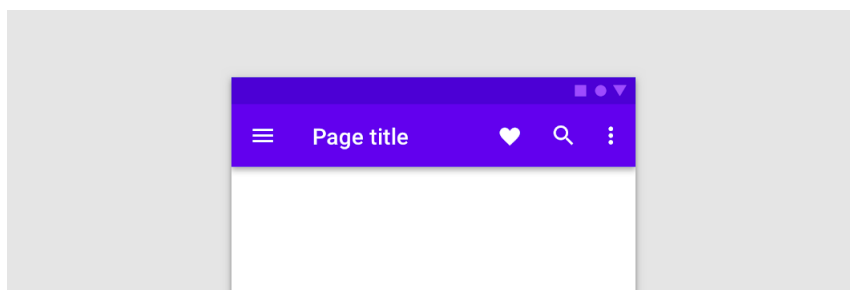
#### *Consistent*

Top app bars have a consistent position and content to increase familiarity.

### Types

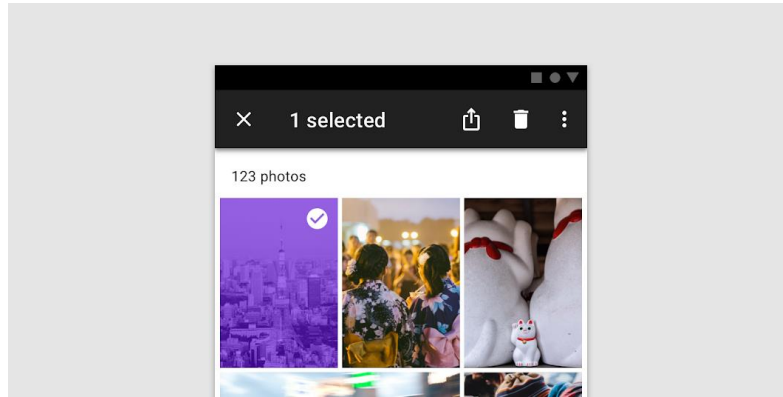
#### *Type 1: Regular*

A regular top app bar



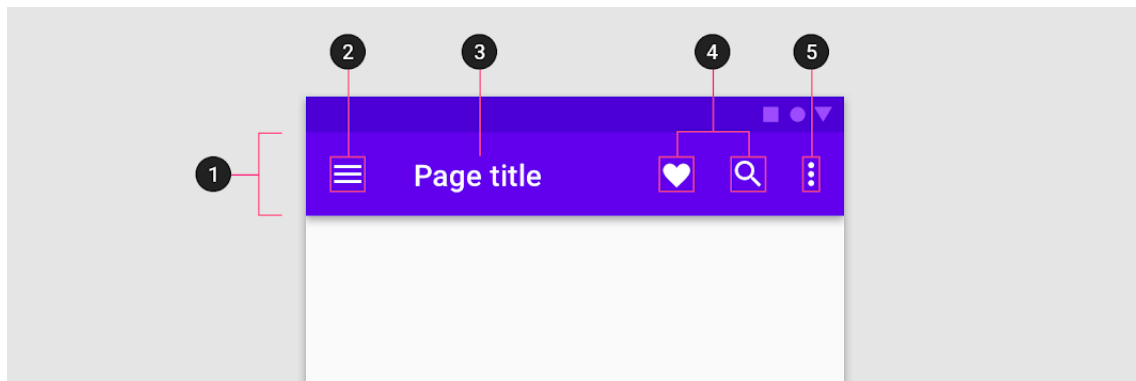
### *Type 2: Contextual action bar*

Contextual action bars provide actions for selected items. A top app bar can transform into a contextual action bar, remaining active until an action is taken or it is dismissed.

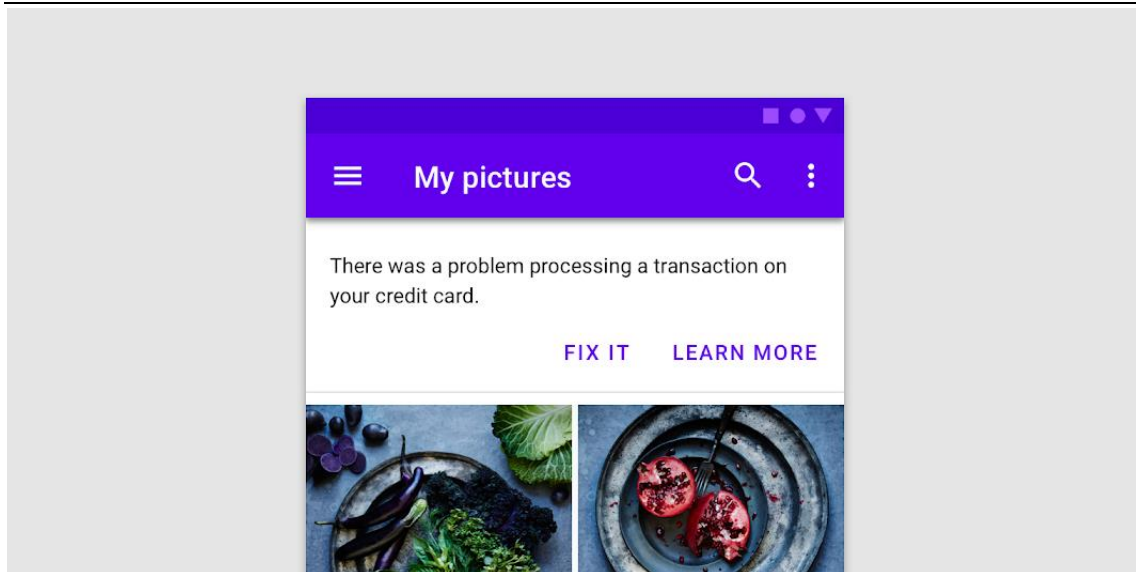


### **Anatomy**

1. Container
2. Navigation icon (optional)
3. Title (optional)
4. Action items (optional)
5. Overflow menu (optional)



## BANNERS



### Usage

A banner displays an important, succinct message, and provides actions for users to address (or dismiss the banner). It requires a user action to be dismissed.

Banners should be displayed at the top of the screen, below a top app bar. They're persistent and nonmodal, allowing the user to either ignore them or interact with them at any time. Only one banner should be shown at a time.

### Principles

#### *Appropriately interruptive*

Banners are interruptive, but their level of interruption should match the information they contain and the context in which they appear.

#### *Clear*

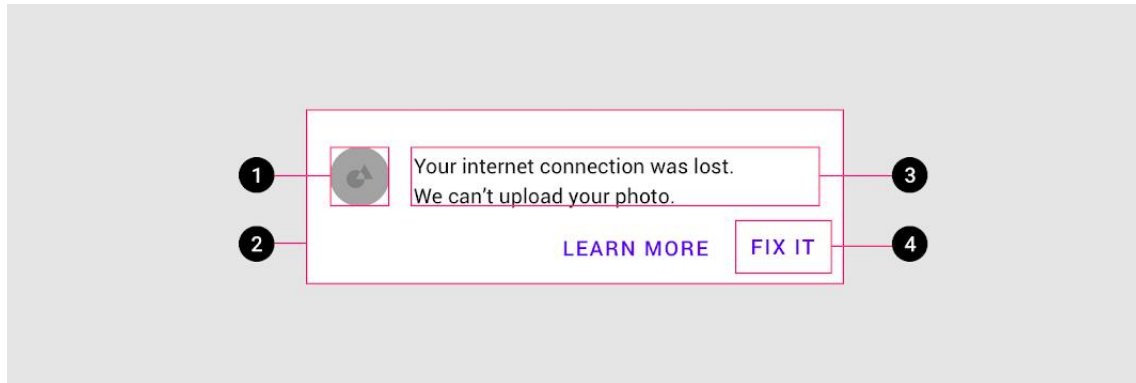
Banners communicate a succinct message and what will happen if users interact with them.

#### *Focused*

Banners contain a single message and specific actions a user can take.

## Anatomy

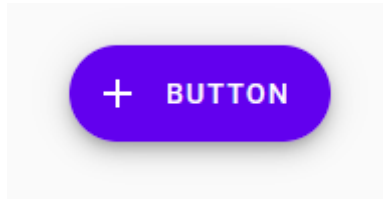
1. Supporting illustration (optional)
2. Container
3. Text
4. Buttons





## BUTTONS

---



### Usage

Buttons communicate actions that users can take. They are typically placed throughout your UI, in places like:

- Dialogs
- Modal windows
- Forms
- Cards
- Toolbars

### Principles

#### *Identifiable*

Buttons should indicate that they can trigger an action

#### *Findable*

Buttons should be easy to find among other elements, including other buttons.

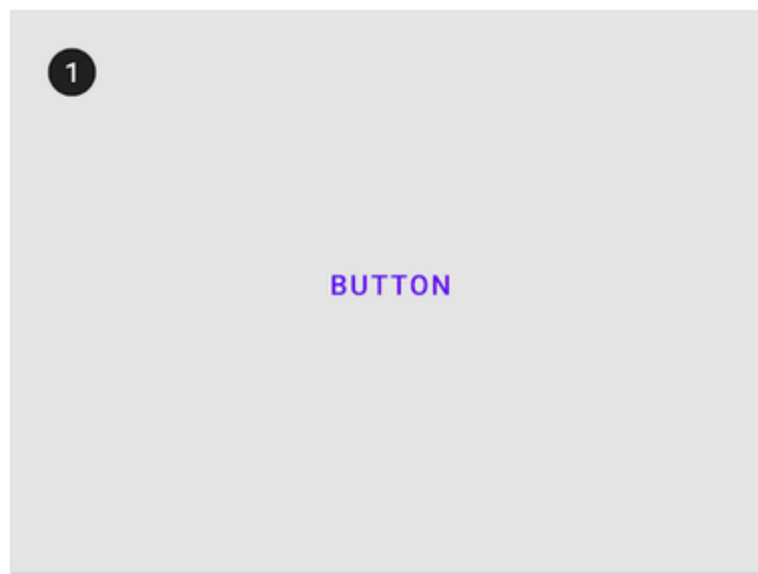
#### *Clear*

A button's action and state should be clear.

### Types

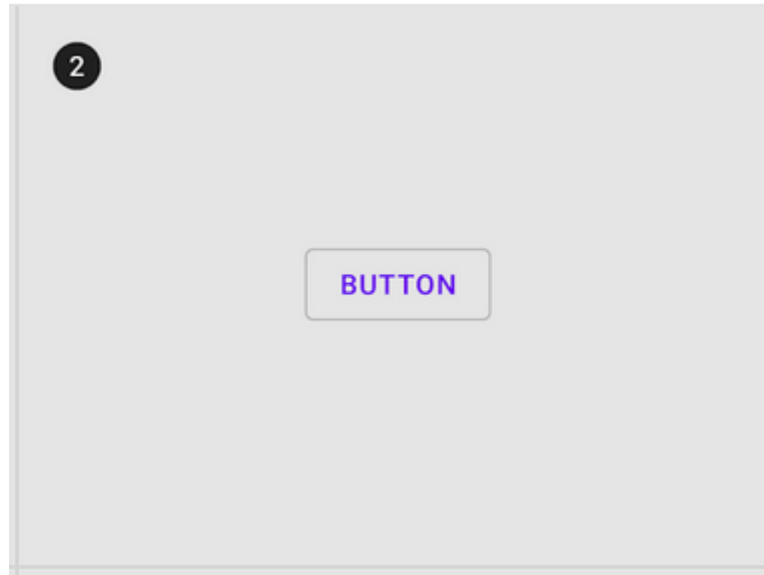
#### *Type 1. Text button (low emphasis)*

Text buttons are typically used for less important actions.

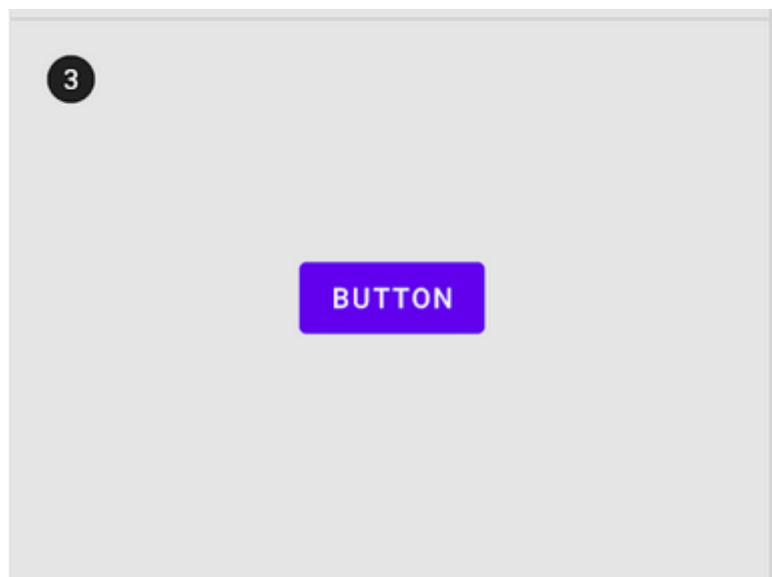


*Type 2. Outlined Button (medium emphasis)*

Outlined buttons are used for more emphasis than text buttons due to the stroke.

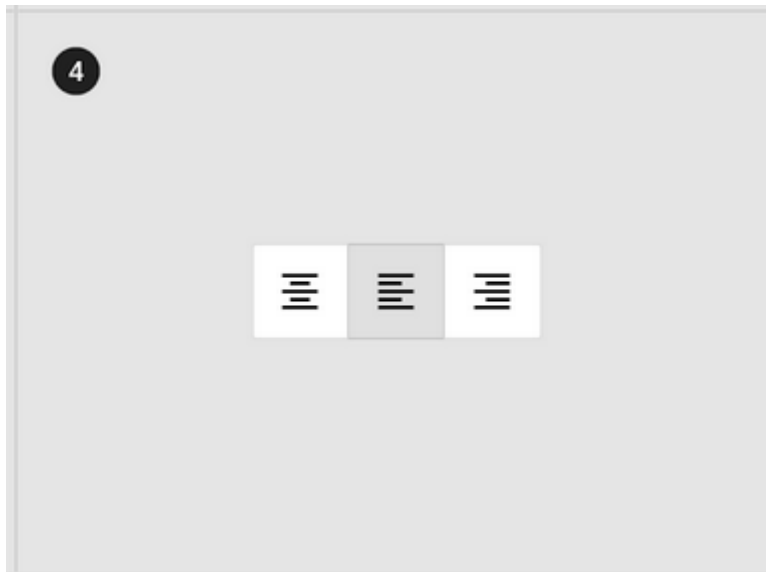
*Type 3. Contained button (high emphasis)*

Contained buttons have more emphasis, as they use a color fill and shadow.



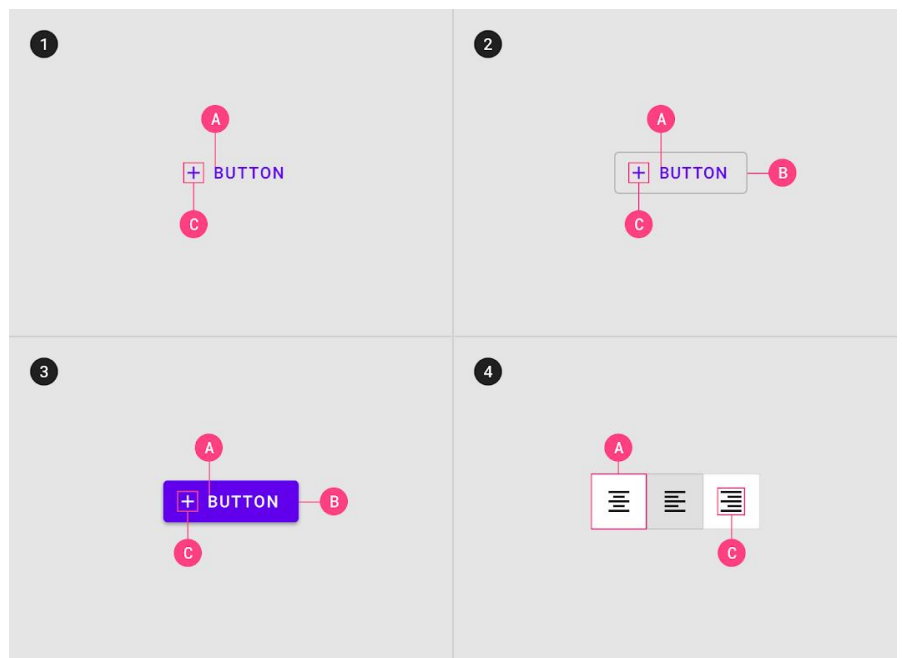
### Type 4. Toggle button

Toggle buttons group a set of actions using layout and spacing. They're used less often than other button types



### Anatomy

1. Text button
  - Text label
  - Icon (optional)
2. Outlined button
  - Text label
  - Container
  - Icon (optional)
3. Contained button
  - Text label
  - Container
  - Icon (optional)
4. Toggle button
  - Text label
  - Icon (optional)



## BUTTONS: FLOATING ACTION BUTTON

---

### Usage

A floating action button (FAB) performs the primary, or most common, action on a screen. It appears in front of all screen content, typically as a circular shape with an icon in its center. FABs come in three types: regular, mini, and extended.

Only use a FAB if it is the most suitable way to present a screen's primary action.

### Principles

#### *Primary*

A FAB represents the primary action on a screen.

#### *Constructive*

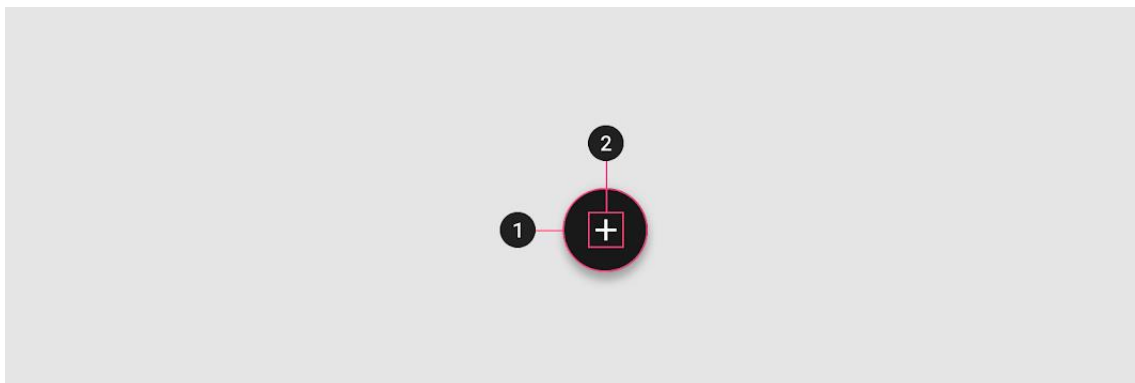
A FAB should perform a constructive action (such as create, share, or explore).

#### *Contextual*

A FAB should be relevant to the screen on which it appears.

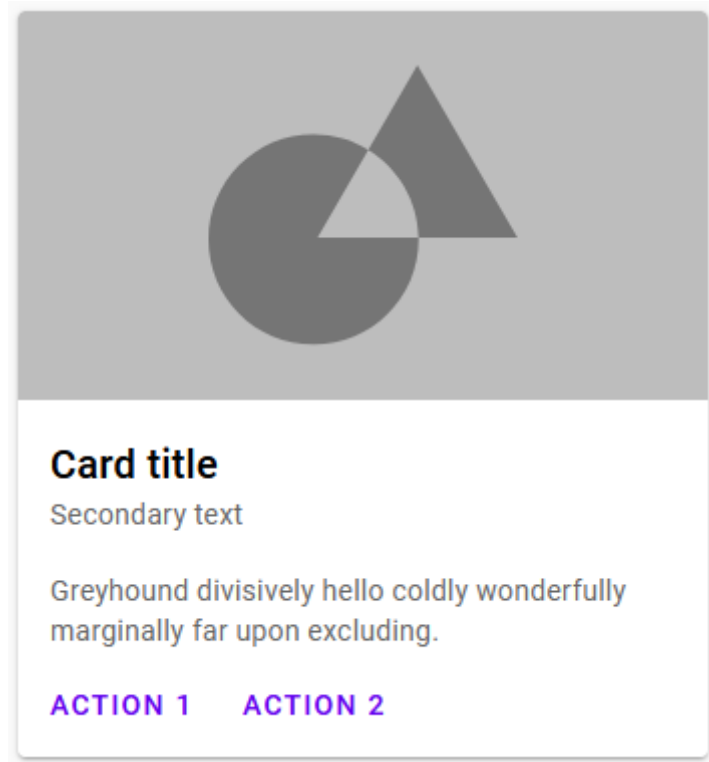
### Anatomy

1. Container
2. Icon



## CARDS

---



### Usage

Cards are surfaces that display content and actions on a single topic.

They should be easy to scan for relevant and actionable information. Elements, like text and images, should be placed on them in a way that clearly indicates hierarchy.

### Principles

#### *Contained*

A card is identifiable as a single, contained unit.

#### *Independent*

A card can stand alone, without relying on surrounding elements for context.

#### *Individual*

A card cannot merge with another card, or divide into multiple cards.

### Anatomy

#### 1. Container

Card containers hold all card elements, and their size is determined by the space those elements occupy. Card elevation is expressed by the container.

#### 2. Thumbnail [optional]

Cards can include thumbnails to display an avatar, logo, or icon.

### 3. Header text [optional]

Header text can include things like the name of a photo album or article.

### 4. Subhead [optional]

Subhead text can include text elements such as an article byline or a tagged location.

### 5. Media [optional]

Cards can include a variety of media, including photos, and graphics, such as weather icons.

### 6. Supporting text [optional]

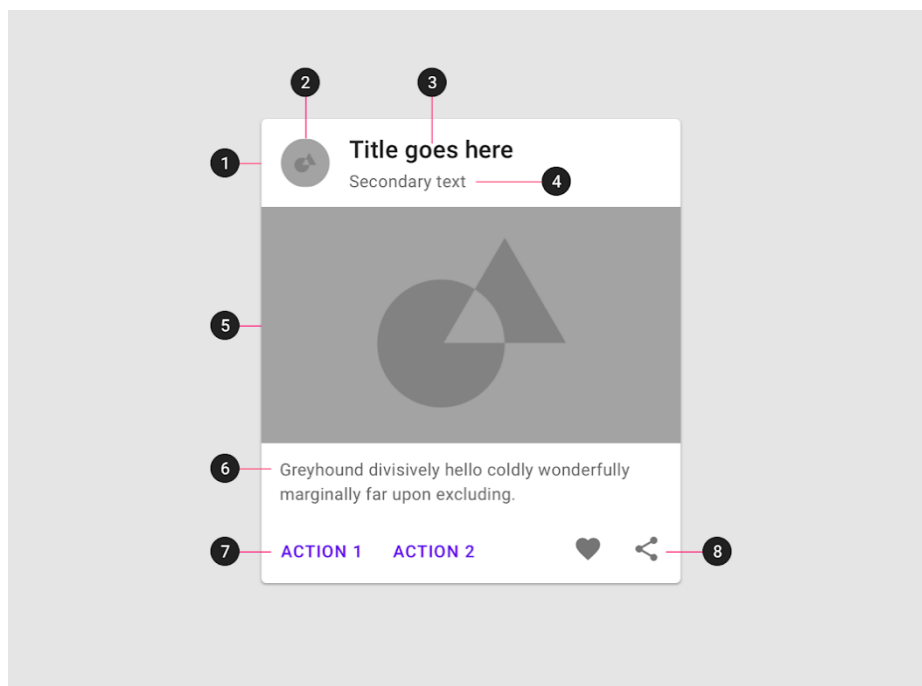
Supporting text include text like an article summary or a restaurant description.

### 7. Buttons [optional]

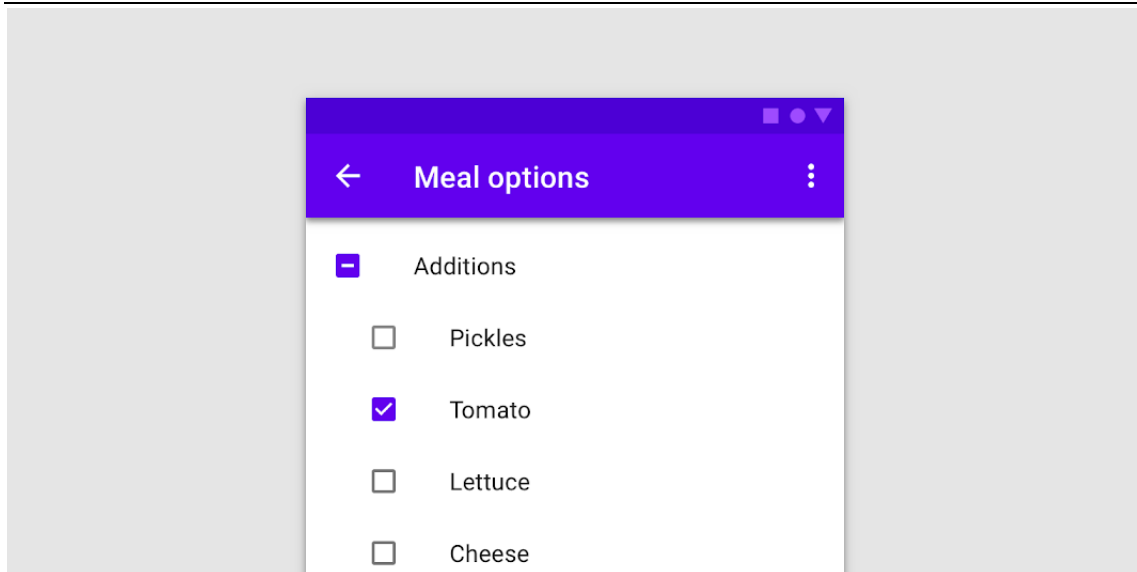
Cards can include buttons for actions.

### 8. Icons [optional]

Cards can include icons for actions.



## CHECKBOXES



### Usage

Use checkboxes to:

- Select one or more options from a list
- Present a list containing sub-selections
- Turn an item on or off in a desktop environment

### Principles

#### *Familiar*

Checkboxes have been in user interfaces for a long time and should be used as expected.

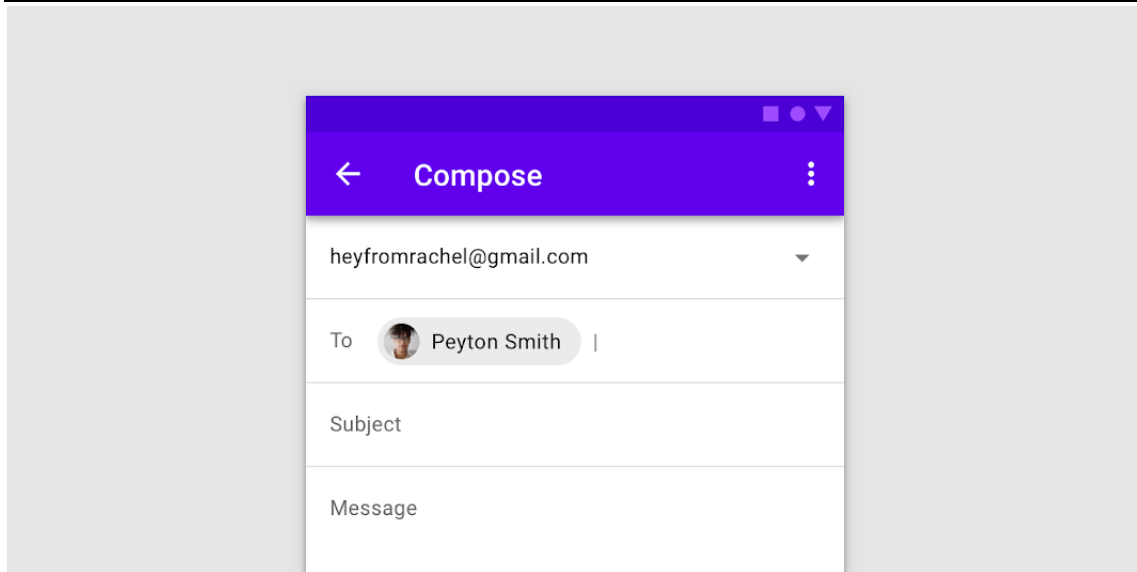
#### *Scannable*

It should be visible at a glance if a checkbox has been selected, and selected items should be more visually prominent than unselected items.

#### *Efficient*

Checkboxes make it easy to compare available options.

## CHIPS



### Usage

Chips allow users to enter information, make selections, filter content, or trigger actions. While buttons are expected to appear consistently and with familiar calls to action, chips should appear dynamically as a group of multiple interactive elements.

### Principles

#### *Compact*

Chips are compact components that represent discrete information.

#### *Relevant*

Chips should have a clear and helpful relationship to the content or task they represent.

#### *Focused*

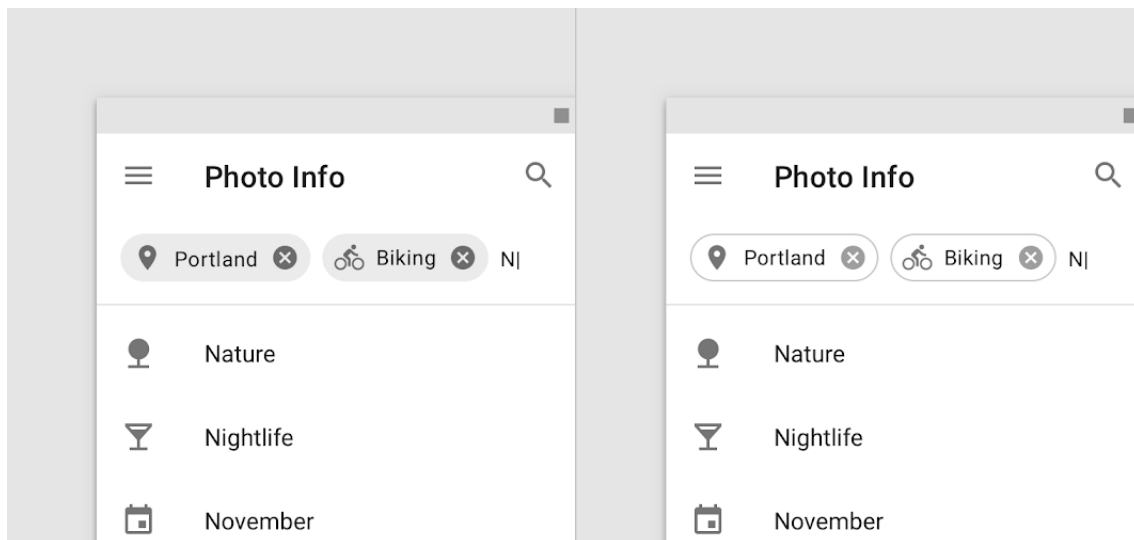
Chips should make tasks easier to complete, or content easier to sort.



## Types

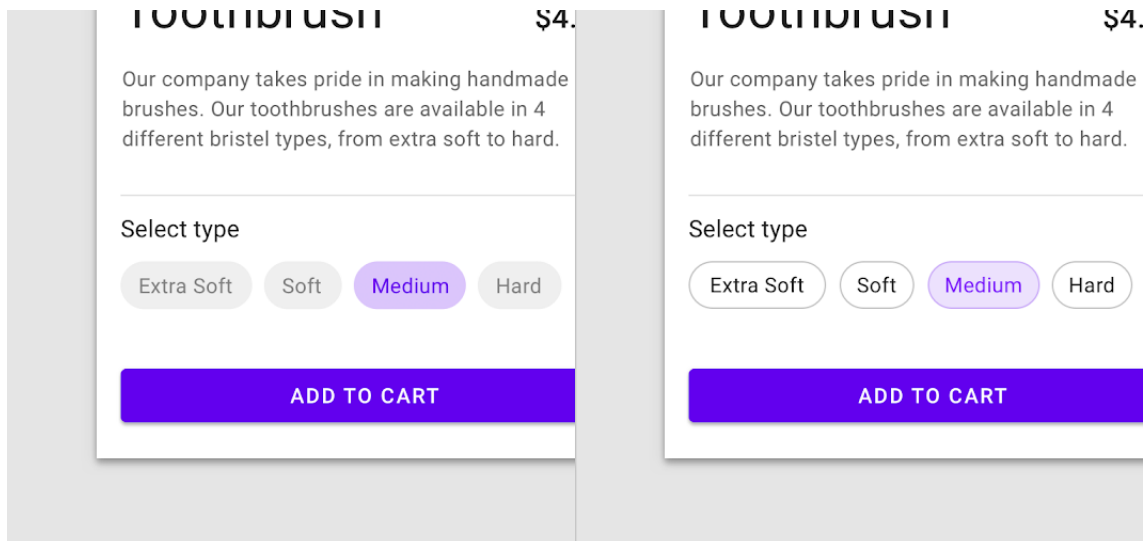
### *Type 1: Input chips*

Input chips represent information used in fields, such as an entity or different attributes.



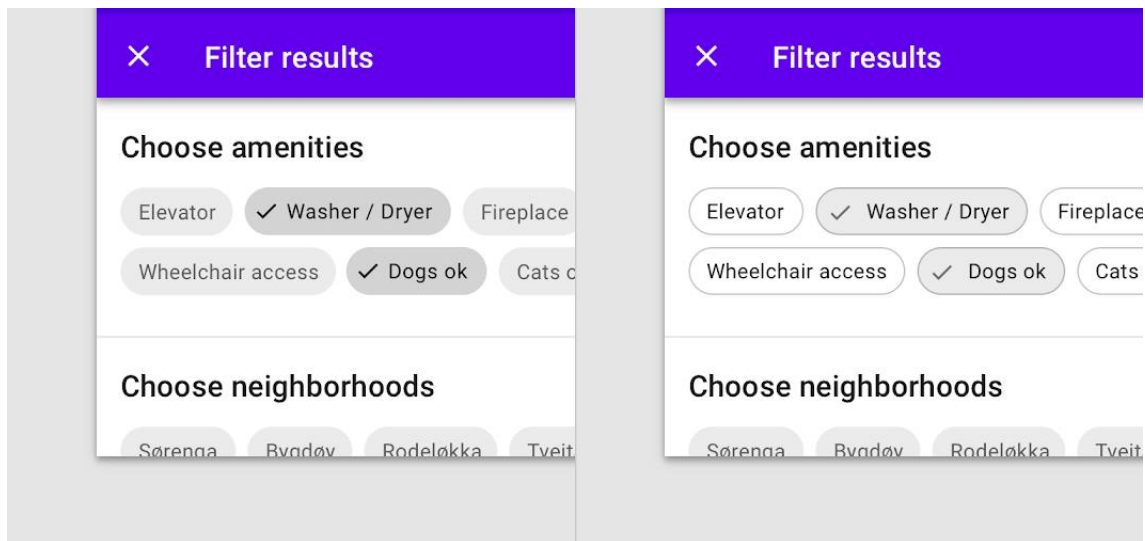
### *Type 2: Choice chips*

In sets that contain at least two options, choice chips represent a single selection.



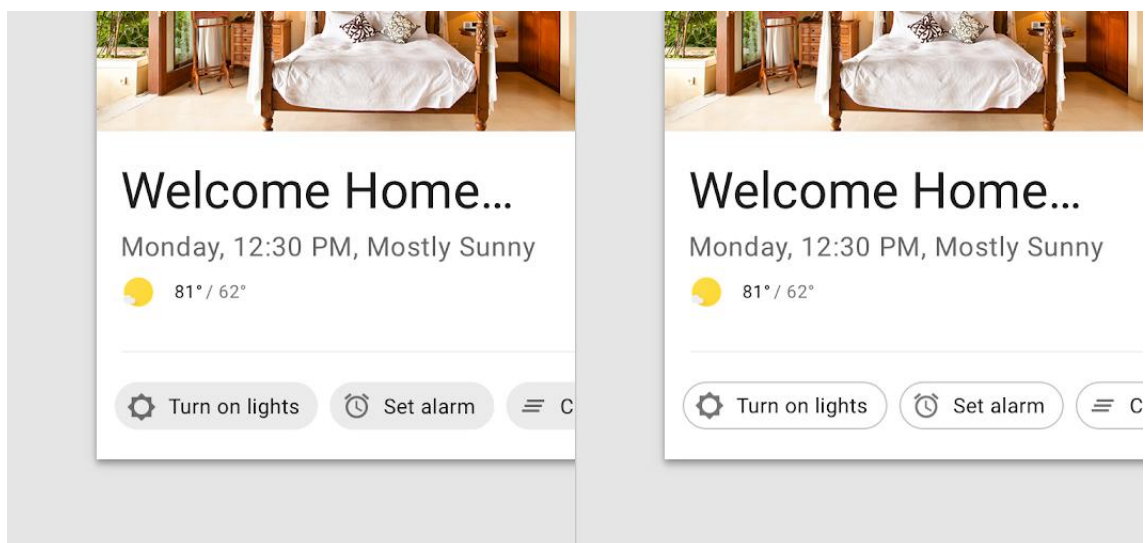
### *Type 3: Filter chips*

Filter chips represent filters for a collection.



### *Type 4: Action chips*

Action chips trigger actions related to primary content.



## Anatomy

### 1. Container

Chip containers hold all chip elements, and their size is determined by those elements. A container can also be defined by a stroke.

### 2. Thumbnail [optional]

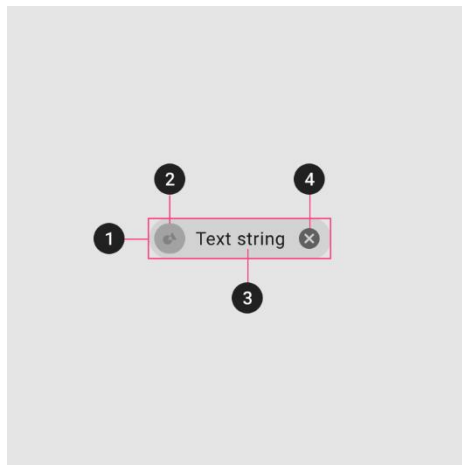
Thumbnails identify entities (like individuals) by displaying an avatar, logo, or icon.

### 3. Text

Chip text can be an entity name, description, tag, action, or conversational.

### 4. Remove icon [optional]

Input chips can include a Remove icon.



## DATA TABLES

---

Header A	Header B	Header C	Header D
Cell A1	Cell B1	Cell C1	1
Cell A2	Cell B2	Cell C2	2
Cell A3	Cell B3	Cell C3	3
Cell A4	Cell B4	Cell C4	4

### Usage

Data tables display information in a grid-like format of rows and columns. They organize information in a way that's easy to scan so that users can look for patterns and develop insights from data.

Data tables can contain:

- Interactive components (such as chips, buttons, or menus)
- Non-interactive elements (such as badges)
- Tools to query and manipulate data

### Principles

#### *Organized*

Information should be organized in a meaningful way, such as hierarchy or alphabetization.

#### *Interactive*

Data tables should allow user interaction so that a data display is customizable and interactive.

#### *Intuitive*

Data tables should be easy to use, with a logical structure that makes content easy to understand.

## Anatomy

1. Header row
2. Rows
3. Pagination
4. Row checkbox
5. Sort button
6. Container

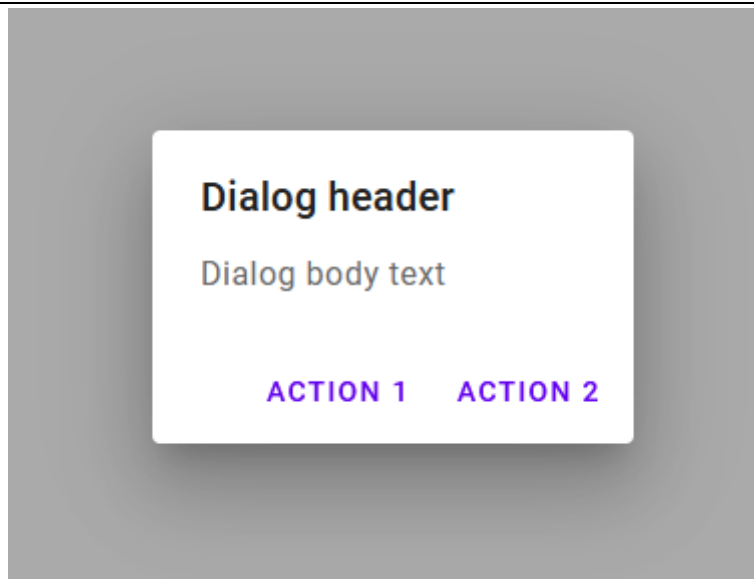
The diagram illustrates the components of a table interface. Callout 1 points to the header row. Callout 2 points to the data rows. Callout 3 points to the pagination controls. Callout 4 points to a checkbox in a data row. Callout 5 points to a sort button in the header row. Callout 6 points to the table container.

<input type="checkbox"/>	Status	Signal Name	Severity	Stage	Sort	Schedule	Team Lead
<input type="checkbox"/>	No signal	Astrid: NE shared managed	Medium	Triaged		0:33	Chase Nguyen
<input type="checkbox"/>	Offline	Cosmo: prod shared ares	Huge	Triaged		0:39	Brie Furman
<input type="checkbox"/>	Online	Phoenix: prod shared lyra-lists	Minor	Not triaged		3:12	Jeremy Lake
<input type="checkbox"/>	Online	Sirius: NW prod shared locations	Negligible	Triaged		13:18	Angelica Howards
<input type="checkbox"/>	Online	Sirius: prod independent account	Negligible	Triaged		22:06	Diane Okuma

Rows per page: 75 1-10 of 100 |< < > >|

## DIALOGS

---



### Usage

A dialog is a type of modal window that appears in front of app content to provide critical information or ask for a decision. Dialogs disable all app functionality when they appear, and remain on screen until confirmed, dismissed, or a required action has been taken.

Dialogs are purposefully interruptive, so they should be used sparingly.

### Principles

#### *Focused*

Dialogs focus user attention to ensure their content is addressed.

#### *Direct*

Dialogs should be direct in communicating information and dedicated to completing a task.

#### *Helpful*

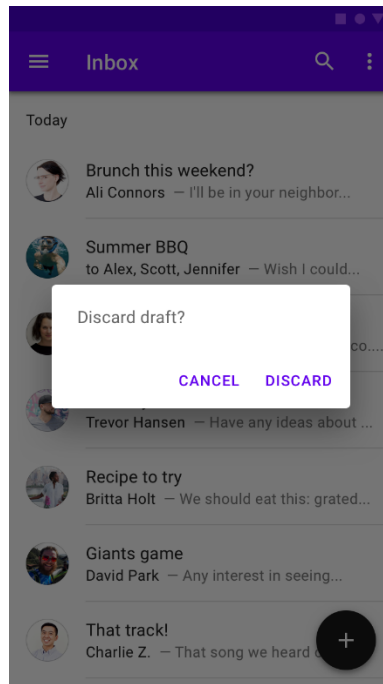
Dialogs should appear in response to a user task or an action, with relevant or contextual information.

### Types

We have 4 different types of dialogs.

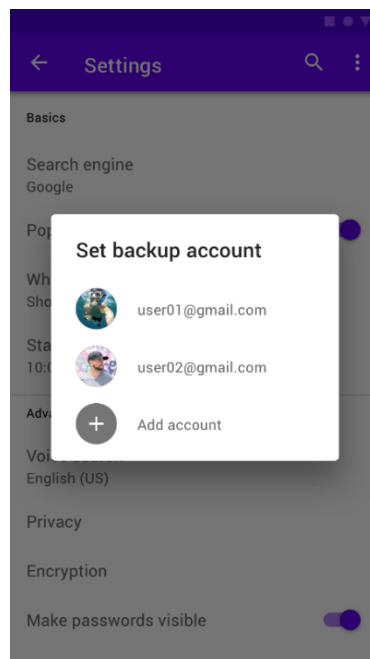
#### *Type 1: Alert Dialog*

Alert dialogs interrupt users with urgent information, details, or actions.



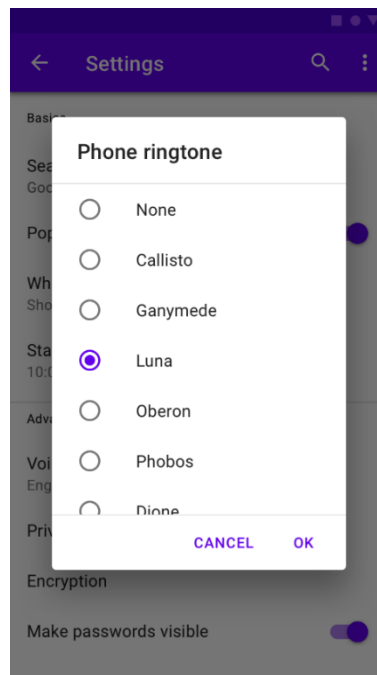
### *Type 2: Simple dialog*

Simple dialogs display a list of items that take immediate effect when selected.



### *Type 3: Confirmation dialog*

Confirmation dialogs require users to confirm a choice before the dialog is dismissed.



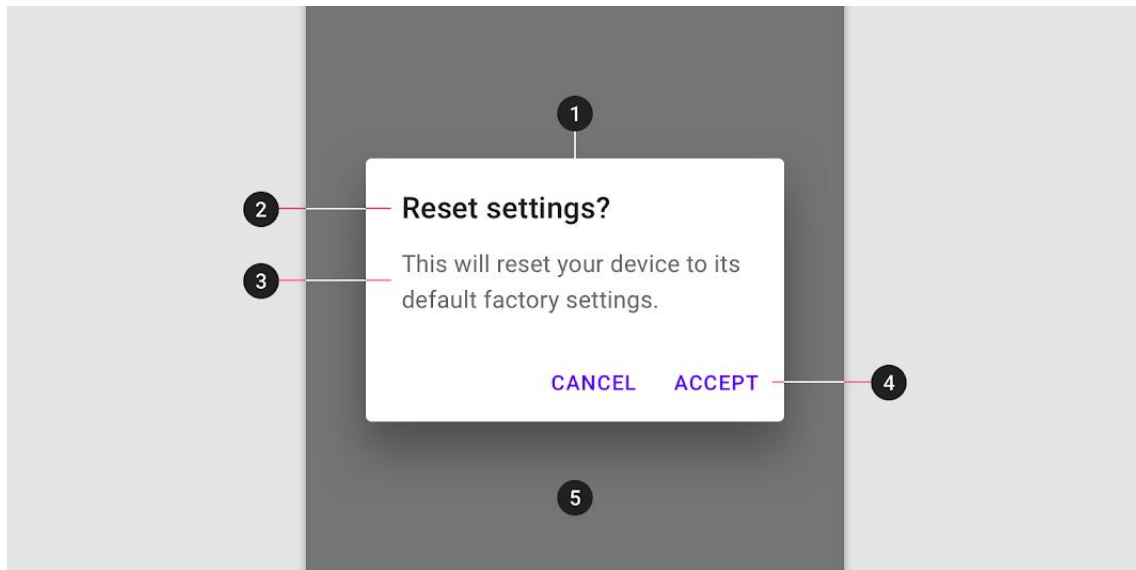
#### *Type 4: Full-screen dialog*

Full-screen dialogs fill the entire screen, containing actions that require a series of tasks to complete.

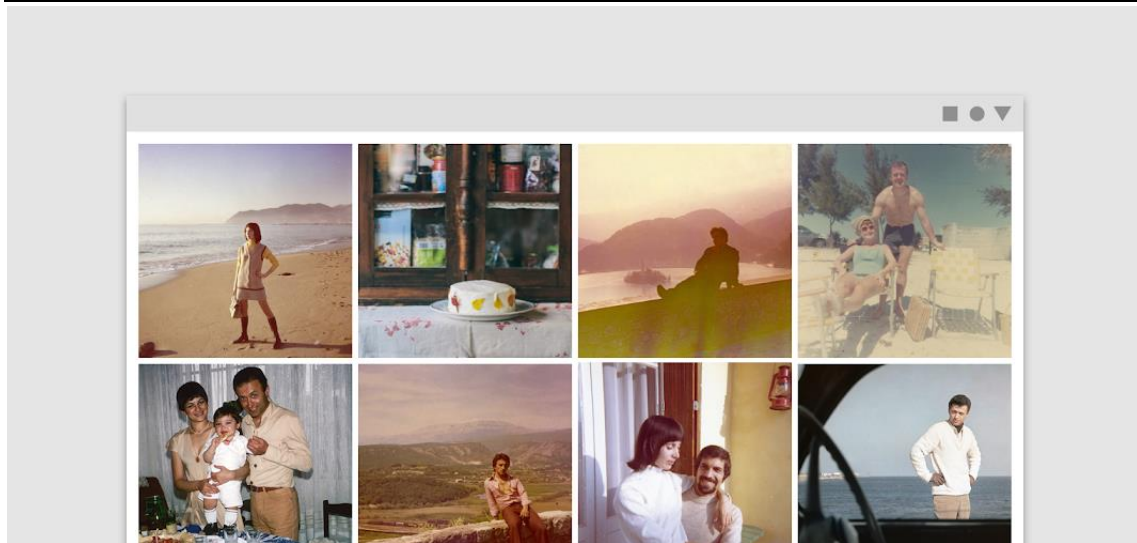
### **Anatomy**

1. Container
2. Title (optional)
3. Supporting text
4. Buttons
5. Scrim





## IMAGE LISTS



### Usage

Image lists represent a collection of items in a repeated pattern. They help improve the visual comprehension of the content they hold.

### Principles

#### *Visual*

Image lists allow users to scan content based on images.

#### *Comparable*

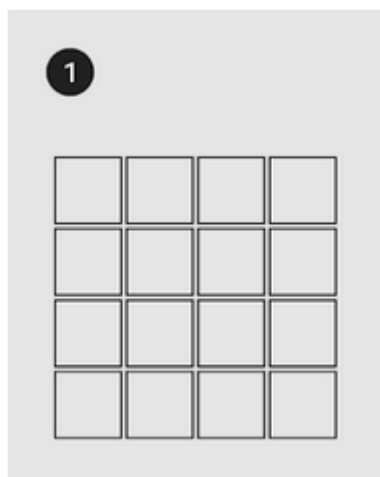
Image lists allow users to easily compare items within a collection.

#### *Integrated*

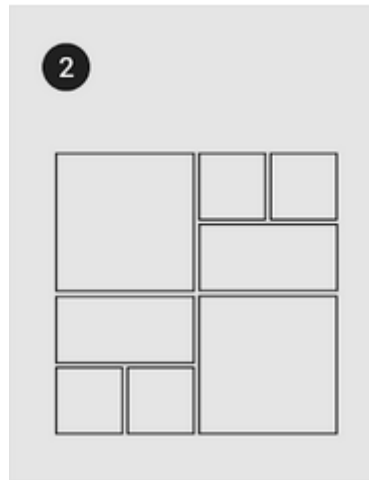
Image lists are responsively integrated with the surrounding content and layout.

### Types

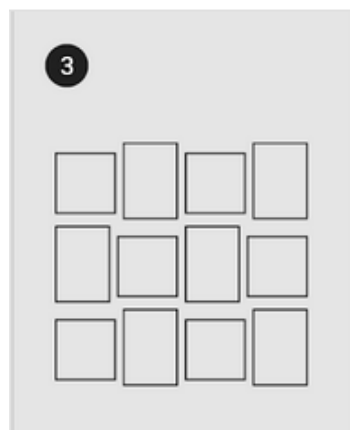
Type 1: **Standard image lists** are best for items of equal importance. They have a uniform container size, ratio, and padding.



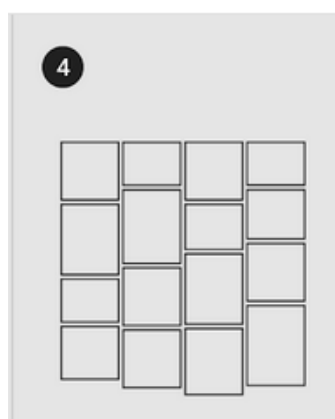
Type 2: **Quilted image lists** emphasize certain items over others in a collection. They create hierarchy using varied container sizes and ratios.



Type 3: **Woven image lists** facilitate the browsing of peer content. They display content in containers of varying ratios to create a rhythmic layout.



Type 4: **Masonry image lists** facilitate the browsing of uncropped peer content. Container heights are sized based on the image size.



## Anatomy

### 1. Image container

The image container displays an image list item's image or illustration.

## 2. Text labels (optional)

Text labels display one line of text related to an image list item.

## 3. Actionable iconography (optional)

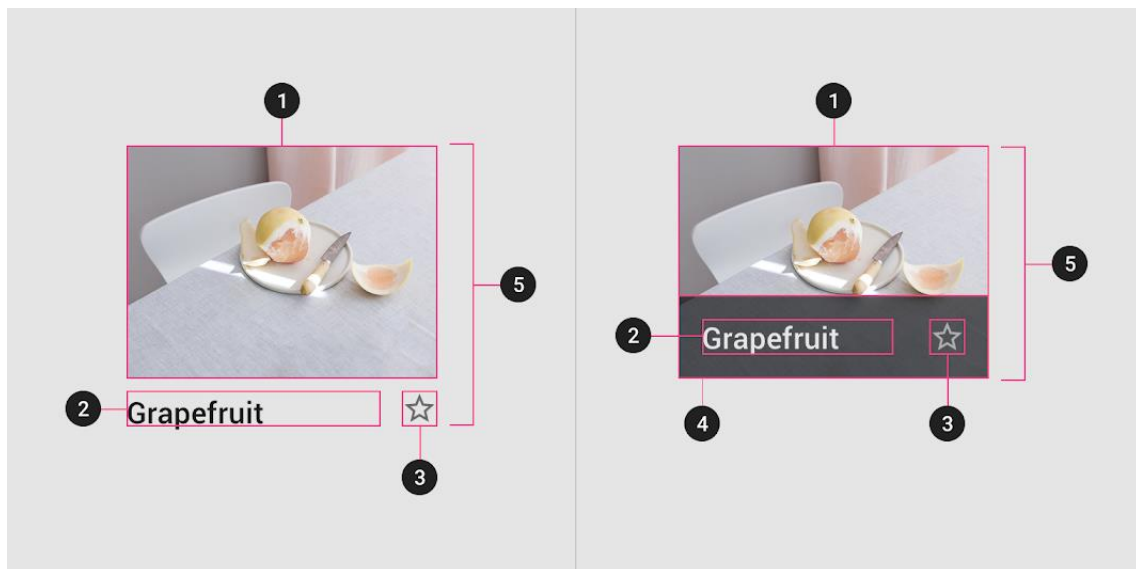
Actionable iconography can represent related actions.

## 4. Text protection (optional)

Text protection is a semi-opaque scrim placed in front of imagery to keep text above it legible.

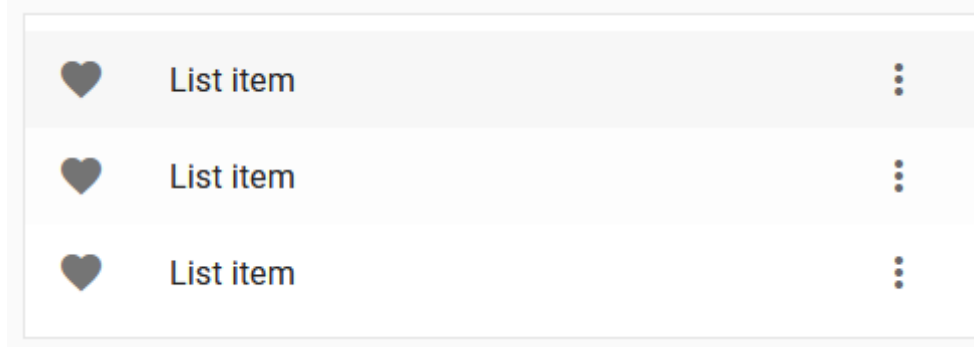
## 5. Image list item

Image list items represent individual items in an image list.



## LISTS

---



### Usage

Lists are a continuous group of text or images. They are composed of items containing primary and supplemental actions, which are represented by icons and text.

### Principles

#### *Logical*

Lists should be sorted in logical ways that make content easy to scan, such as alphabetical, numerical, chronological, or by user preference.

#### *Actionable*

Lists present content in a way that makes it easy to identify a specific item in a collection and act on it.

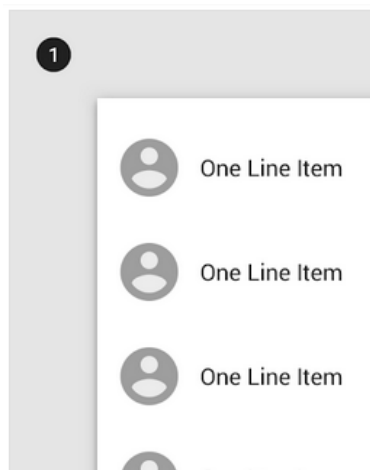
#### *Consistent*

Lists should present icons, text, and actions in a consistent format.

### Types

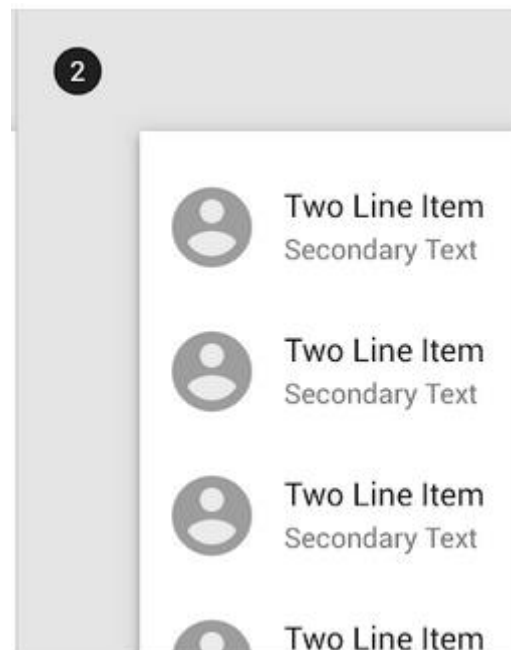
#### *Type 1: Single-line list*

Single-line list items contain a maximum of one line of text.

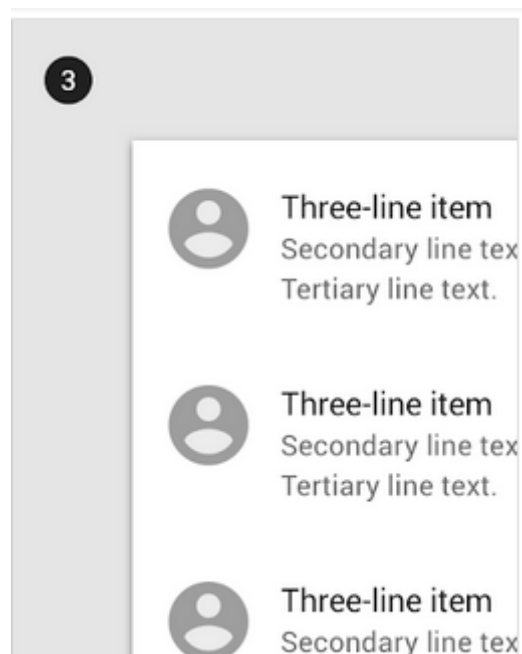


**Type 2: Two-line list**

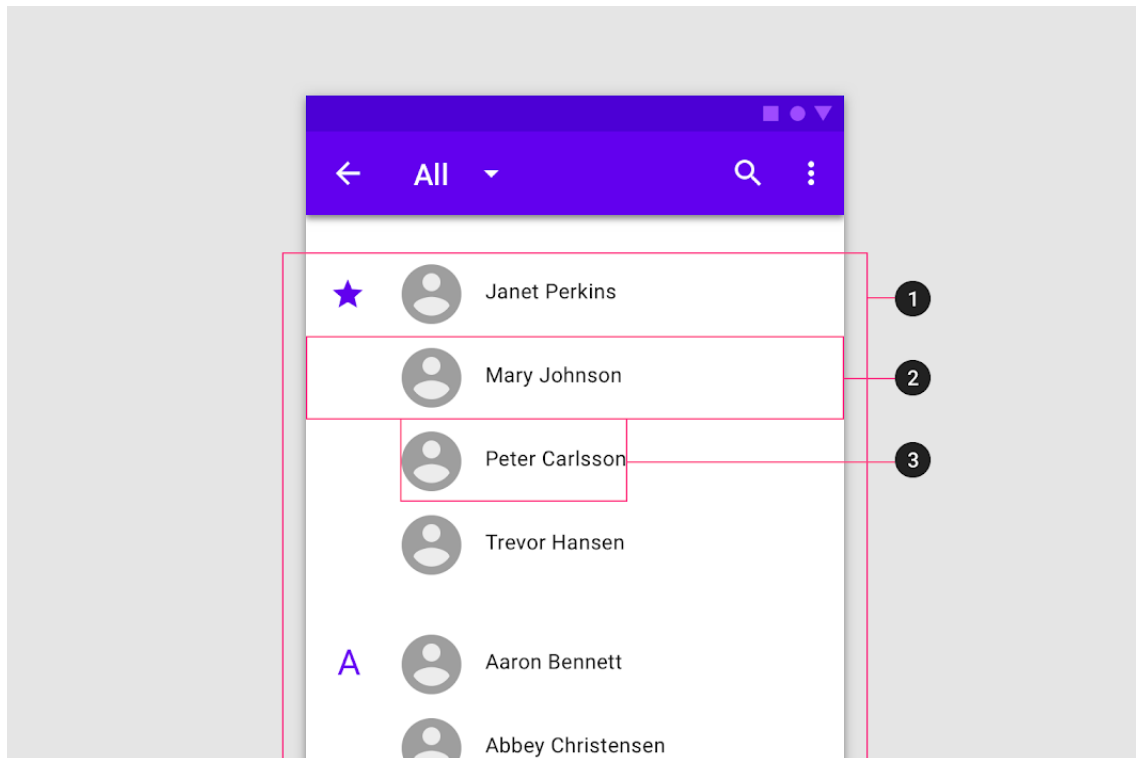
Two-line list items contain a maximum of two lines of text.

**Type 3: Three-line list**

Three-line list items contains a maximum of three lines of text.

**Anatomy**

1. List
2. Row
3. List item content



## MENUS

---



### Usage

A menu displays a list of choices on a temporary surface. They appear when users interact with a button, action, or other control.

### Principles

#### *Nimble*

Menus should be easy to open, close, and interact with.

#### *Contextual*

Menu content should be suited to user needs.

#### *Scannable*

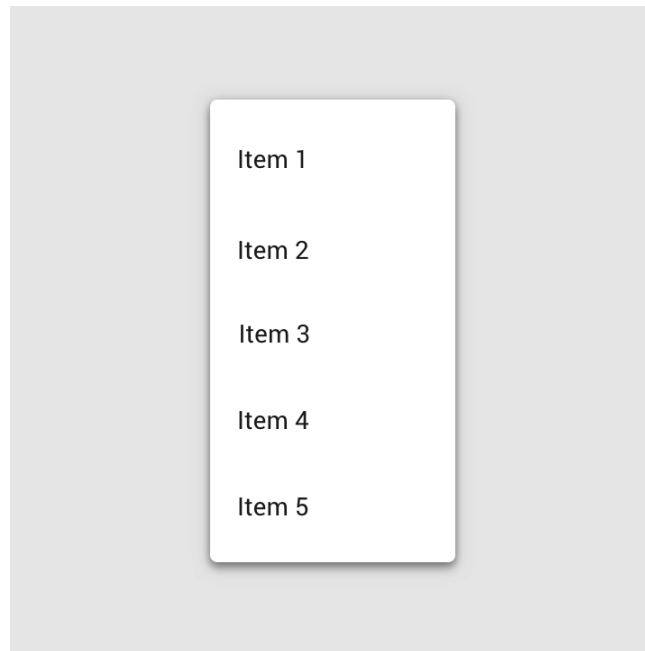
Menu items should be easy to scan.

### Types

#### *Type 1: Dropdown menus*

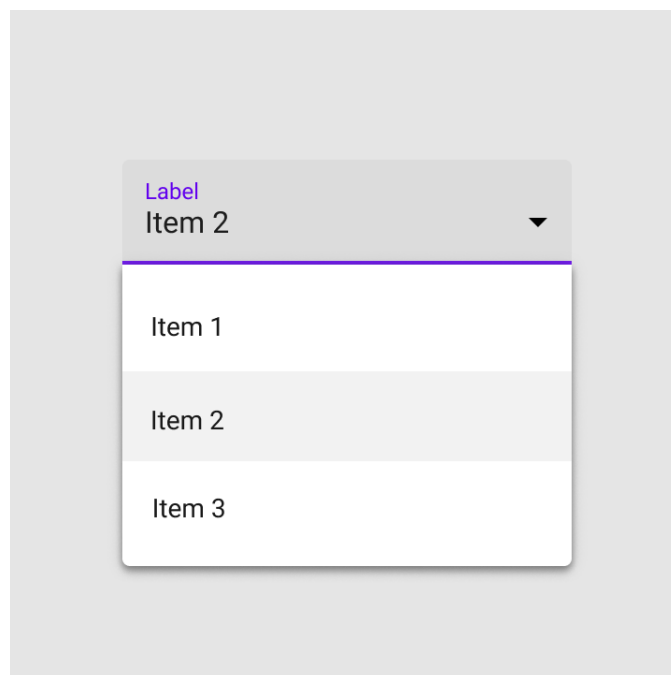
Dropdown menus display a list of options, triggered by an icon, button, or action. Their placement varies based on the element that opens them.





### *Type 2: Exposed dropdown menus*

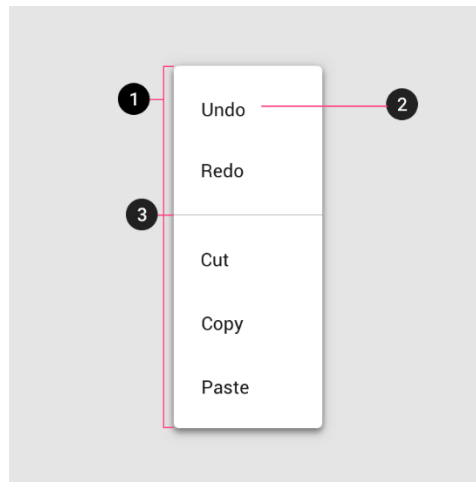
Exposed dropdown menus display the currently selected menu item above the list of options. Some variations can accept user-entered input.



## **Anatomy**

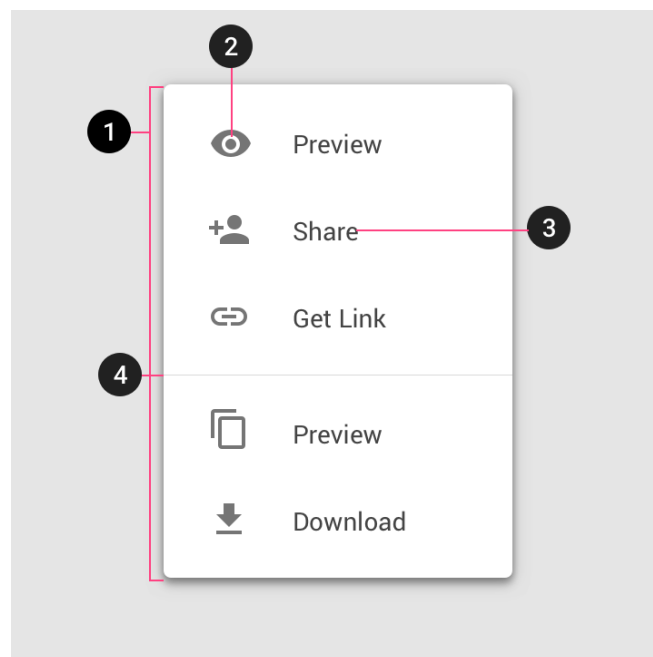
### *Text list*

1. Container
2. Text
3. Divider



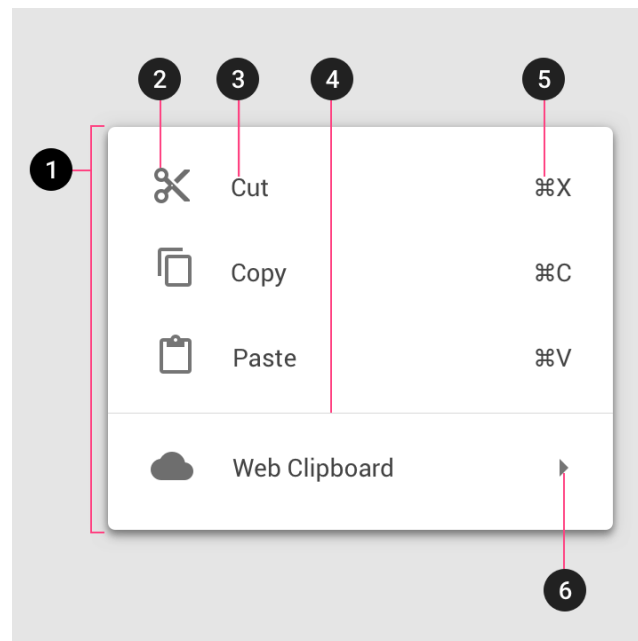
### *Text and icon list*

1. Container
2. Leading icon
3. Text
4. Divider



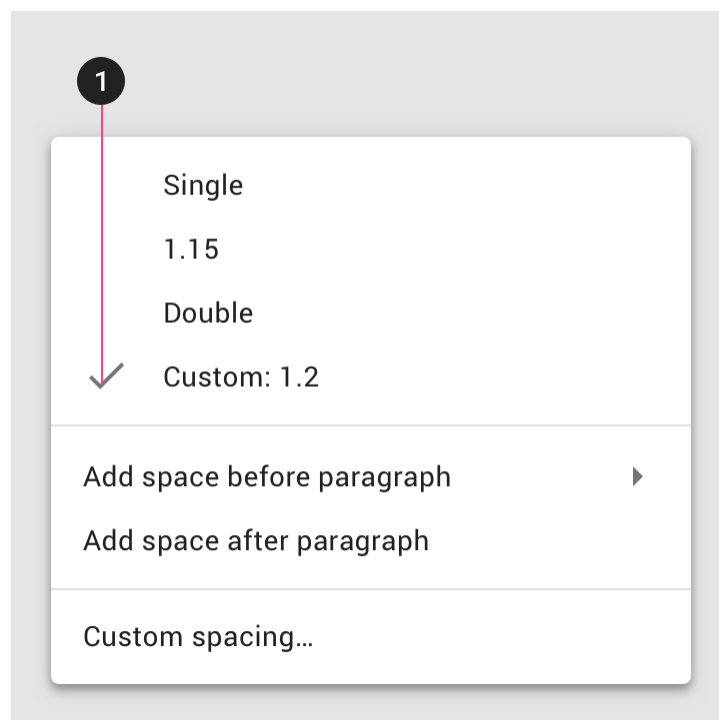
### *Text, icon, and keyboard command list*

1. Container
2. Leading icon
3. Text
4. Divider
5. Command
6. Cascading menu indicator

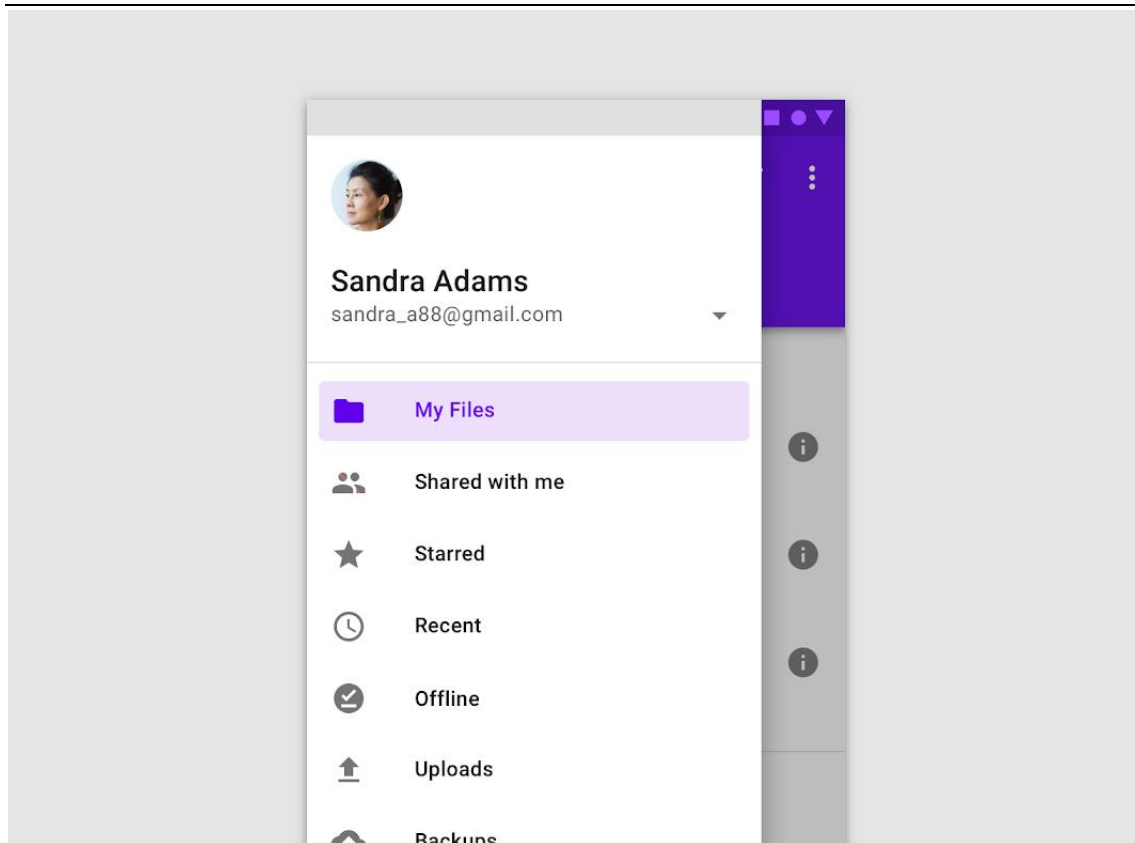


### *Text with selection state list*

#### 1. Selection state



## NAVIGATION DRAWER



### Usage

Navigation drawers provide access to destinations and app functionality, such as switching accounts. They can either be permanently on-screen or controlled by a navigation menu icon.

Navigation drawers are recommended for:

- Apps with five or more top-level destinations
- Apps with two or more levels of navigation hierarchy
- Quick navigation between unrelated destinations

### Principles

#### *Identifiable*

The placement and list-style content of navigation drawers clearly identify them as navigation.

#### *Organized*

Navigation drawers order destinations according to user importance, with frequent destinations first and related ones grouped together.

#### *Contextual*

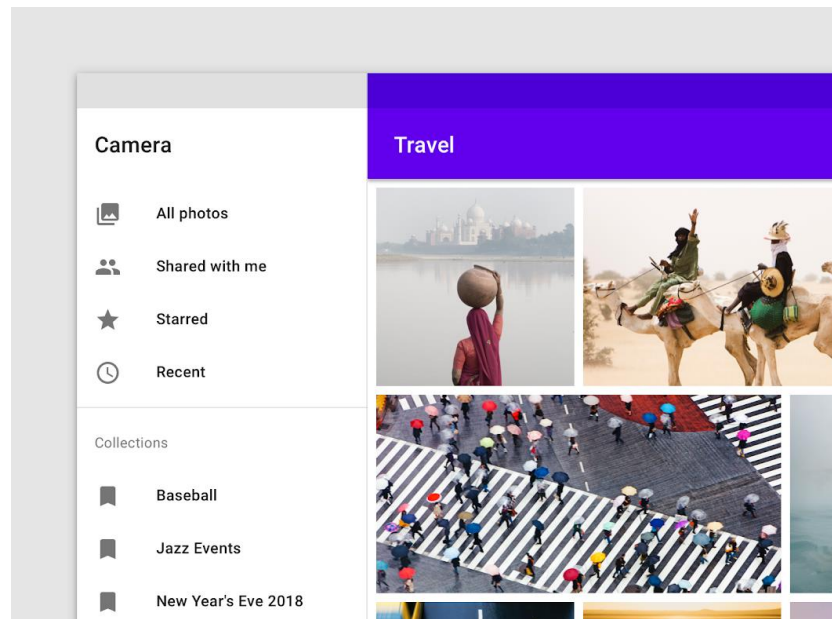
Navigation drawers can be shown or hidden to accommodate different app layouts.

## Types

### *Type 1: Standard drawer*

Standard navigation drawers allow users to simultaneously access drawer destinations and app content. They are often co-planar with app content and affect the screen's layout grid.

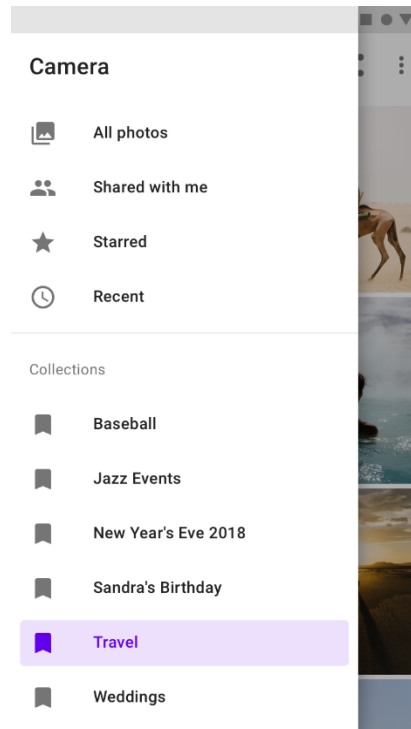
Standard drawers can be permanently visible or opened and closed by tapping a navigation menu icon. They can be used on tablet and desktop only. On mobile, modal drawers are used instead.



### *Type 2: Modal drawer*

Modal navigation drawers use a scrim to block interaction with the rest of an app's content. They are elevated above most app elements and don't affect the screen's layout grid.

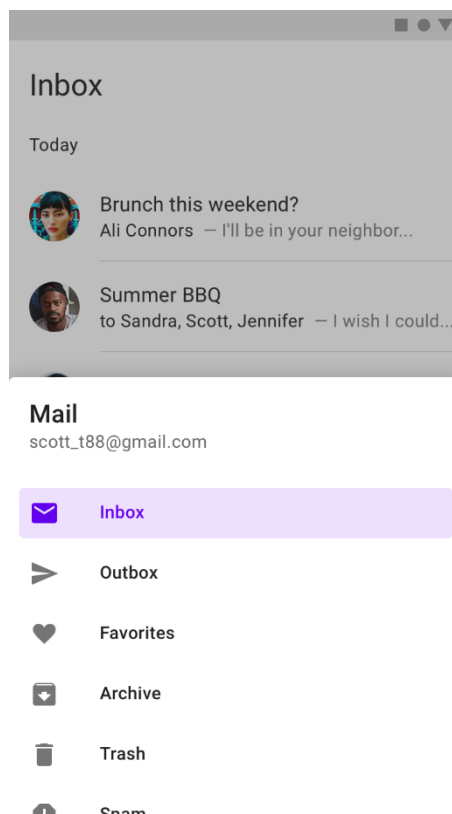
They are primarily for use on mobile, where screen space is limited. They can be replaced by standard drawers on tablet and desktop.



### *Type 3: Bottom drawer*

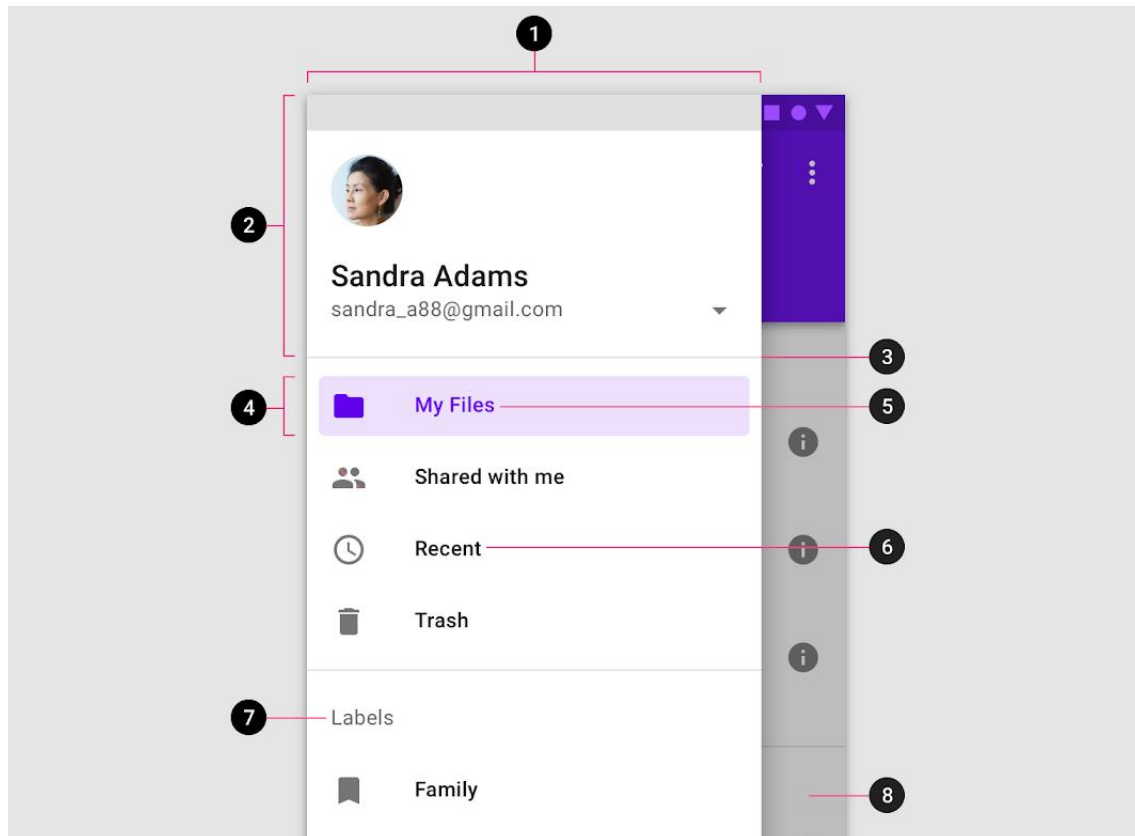
Bottom navigation drawers are a specialized type of modal drawer for use with a bottom app bar.

For increased reachability from the bottom app bar's menu icon, they open from the bottom of the screen rather than the side.



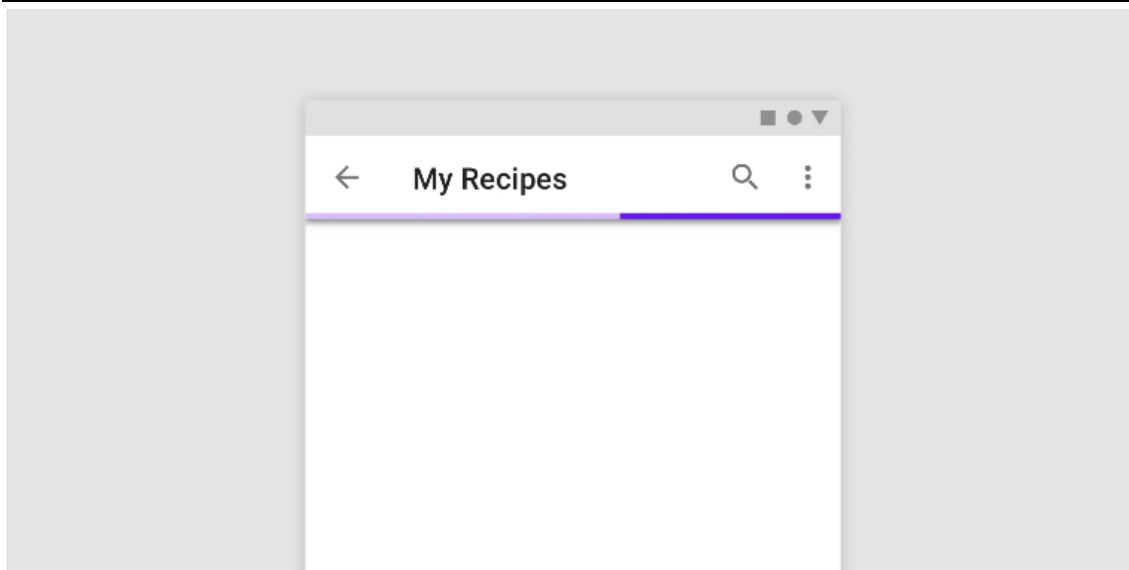
## Anatomy

1. Container
2. Header (optional)
3. Divider (optional)
4. Active text overlay
5. Active text
6. Inactive text
7. Subtitle
8. Scrim (modal only)



## PROGRESS INDICATORS

---



### Usage

Progress indicators inform users about the status of ongoing processes, such as loading an app, submitting a form, or saving updates. They communicate an app's state and indicate available actions, such as whether users can navigate away from the current screen.

### Principles

#### *Informative*

Progress indicators look and animate in ways that reflect the status of a process. They are never simply decorative.

#### *Animated*

Progress indicators use animation to capture attention and inform users of an activity's progress.

#### *Consistent*

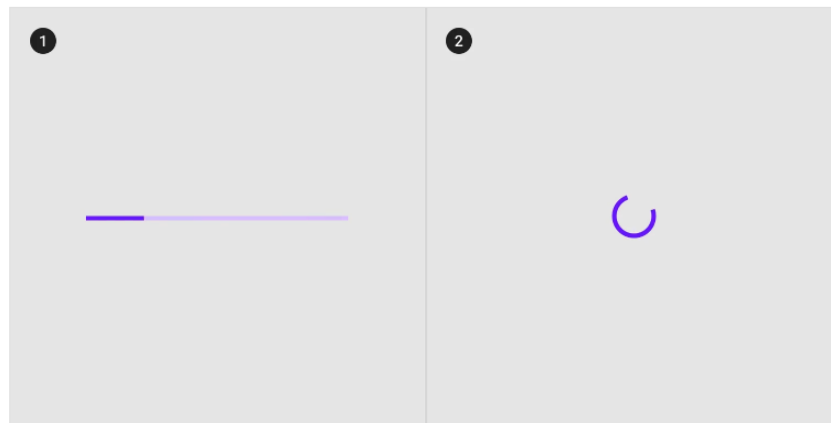
Progress indicators should be applied to all instances of a process (such as loading) in a consistent format (linear or circular).

### Types

#### *Linear and circular*

Material Design offers two visually distinct types of progress indicators: linear and circular progress indicators. Only one type should represent each kind of activity in an app. For example, if a refresh action displays a circular indicator on one screen, that same action shouldn't use a linear indicator elsewhere in the app.





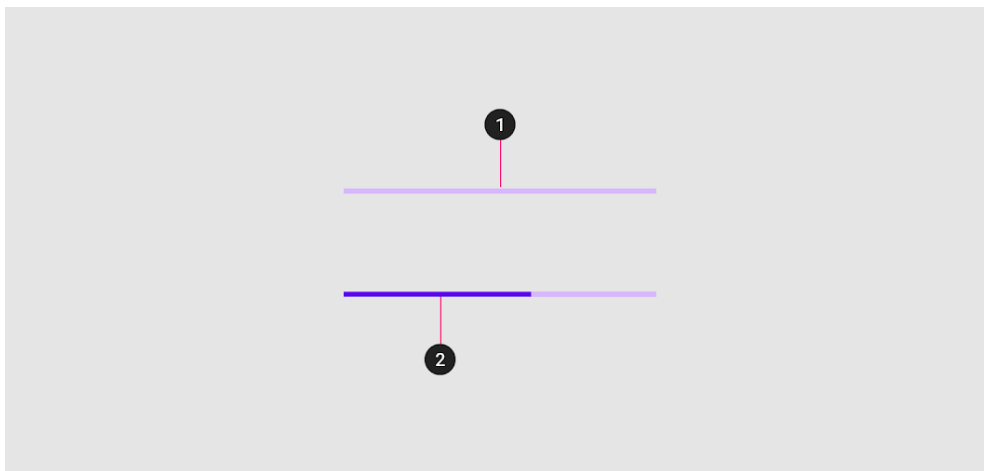
## Anatomy

### 1. Track

The track is a fixed width rule, with set boundaries for the indicator to travel along.

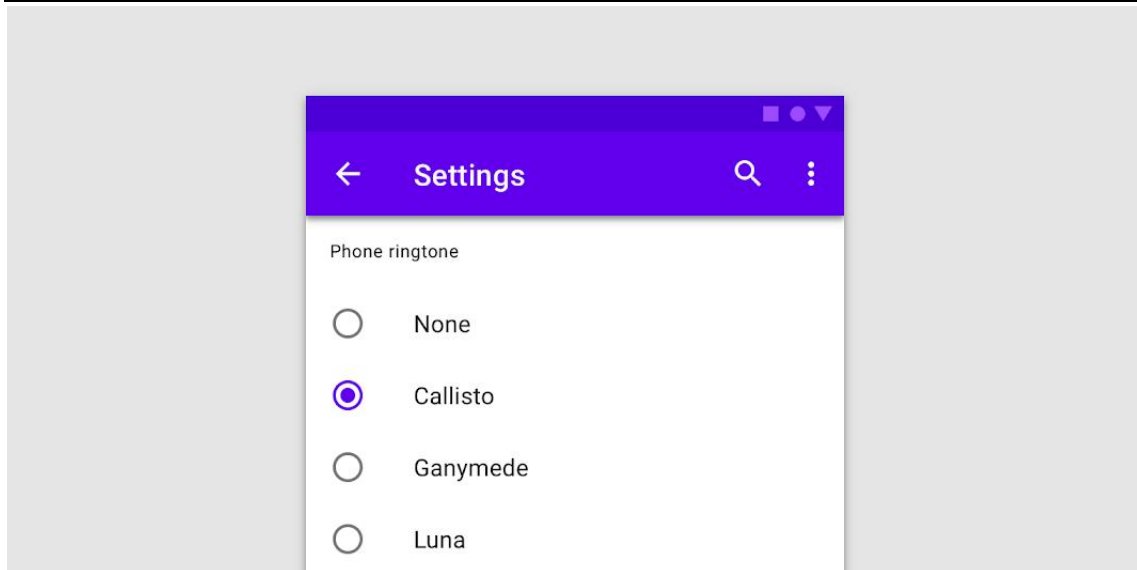
### 2. Indicator

The indicator animates along the length of the track.



## RADIO BUTTONS

---



### Usage

Use radio buttons to:

- Select a single option from a list
- Expose all available options

If available options can be collapsed, consider using a dropdown menu instead, as it uses less space.

### Principles

#### *Familiar*

Radio buttons have been in user interfaces for a long time and should be used as expected.

#### *Scannable*

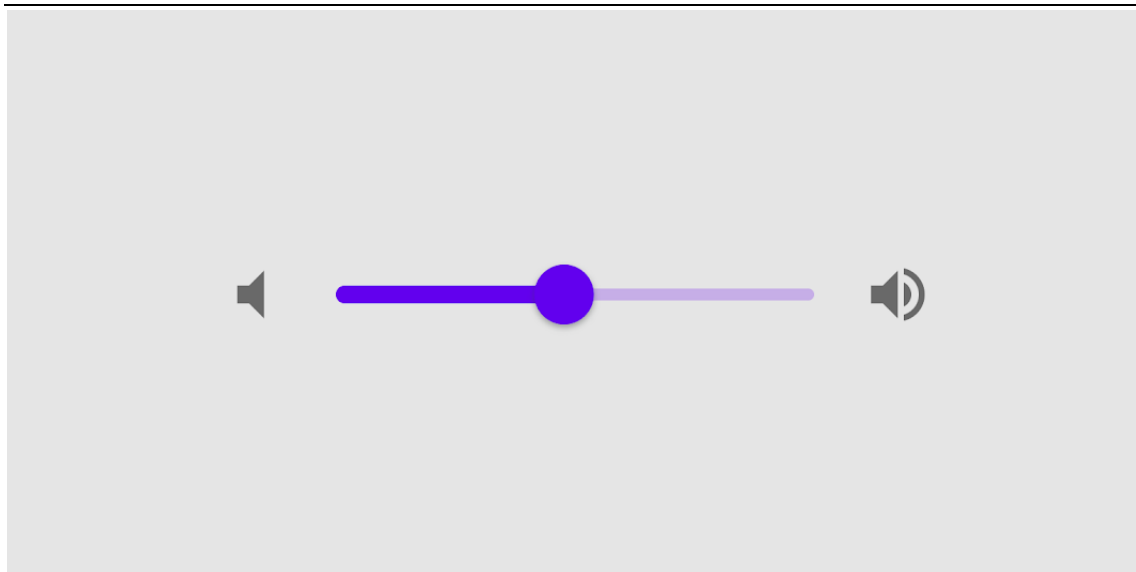
It should be visible at a glance if a radio button has been selected, and selected items should be more visually prominent than unselected items.

#### *Efficient*

Radio buttons make it easy to compare available options.

## SLIDERS

---



### Usage

Sliders allow users to view and select a value (or range) from the range along a bar. They're ideal for adjusting settings such as volume and brightness, or for applying image filters.

Sliders can use icons on both ends of the bar to represent a numeric or relative scale. The range of values or the nature of the values, such as volume change, can be communicated with icons.

### Principles

#### *Adjustable*

Sliders should be used for making selections from a range of values.

#### *Immediate*

When interacting with a slider, changes should be reflected back to a user immediately.

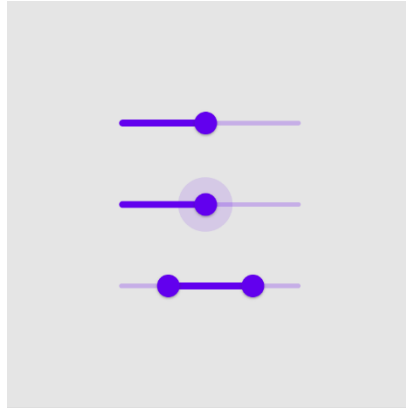
#### *Accessible*

Sliders should present the full range of choices that are available to a user.

### Types

#### *Type 1: Continuous sliders*

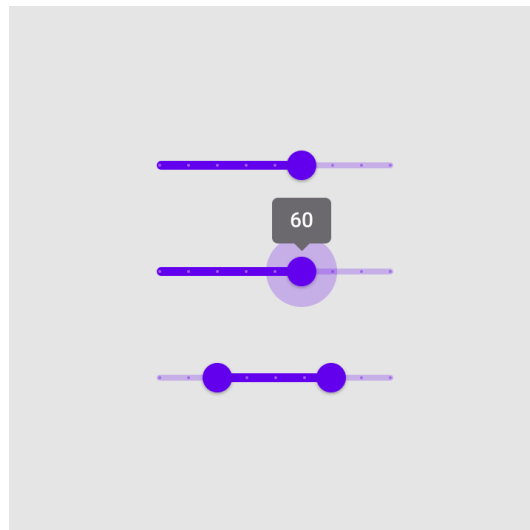
Continuous sliders allow users to set and select a value along a subjective range.



### *Type 2: Discrete sliders*

Discrete sliders can be adjusted to a specific value by referencing its value indicator.

Allowed selections may be organized and indicated with tick marks that a slider thumb will snap to.



## **Anatomy**

### **1. Track**

The track shows the range that is available for a user to select from. For left-to-right (LTR) languages, the smallest value appears on the far left end of the track and the largest value is on the far right. For right-to-left (RTL) languages this orientation is reversed.

### **2. Thumb**

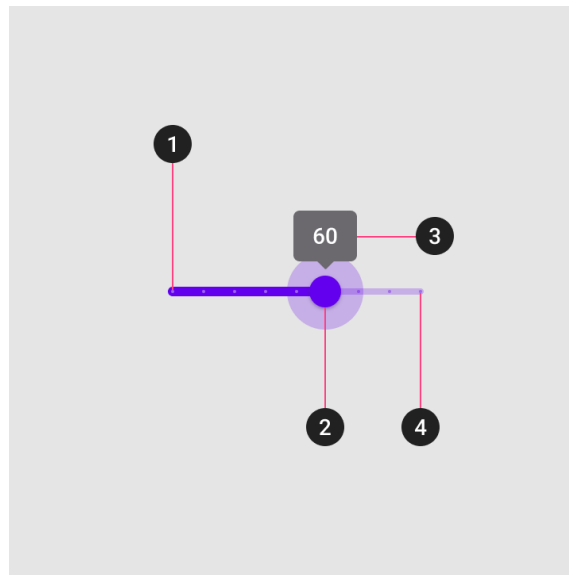
The thumb is a position indicator that can be moved along the track, displaying the selected value of its position.

### **3. Value label (optional)**

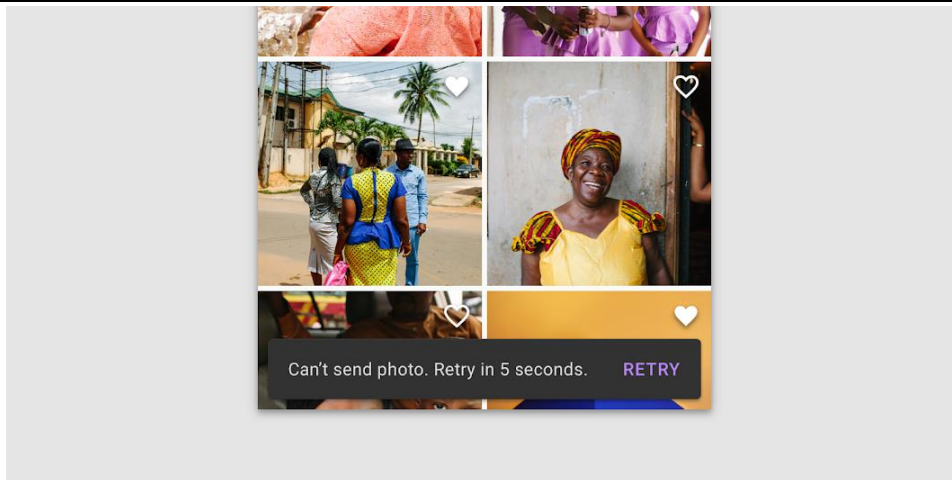
A value label displays the specific numeric value that corresponds with the thumb's placement.

#### 4. Tick mark (optional)

Tick marks along a track represent predetermined values that the user can move the slider to.



## SNACKBARS



### Usage

Snackbars inform users of a process that an app has performed or will perform. They appear temporarily, towards the bottom of the screen. They shouldn't interrupt the user experience, and they don't require user input to disappear.

### Principles

#### *Informational*

Snackbars provide updates on an app's processes.

#### *Temporary*

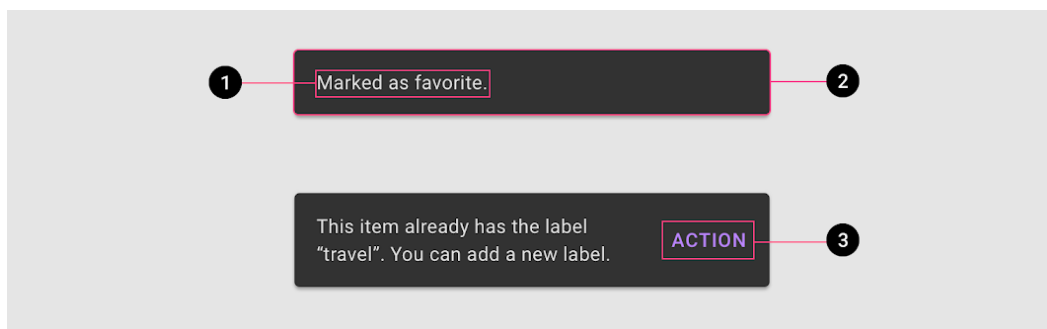
Snackbars appear temporarily, and disappear on their own without requiring user input to be dismissed.

#### *Contextual*

Snackbars are placed in the most suitable area of the UI.

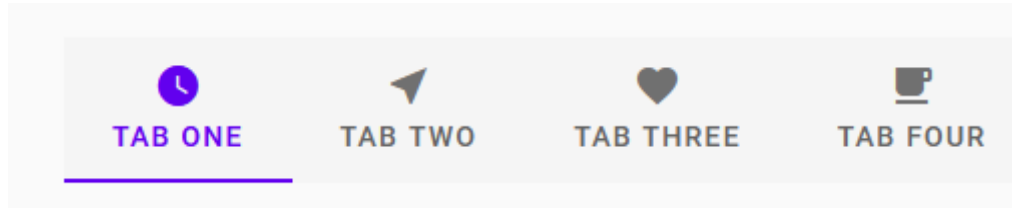
### Anatomy

1. Text label
2. Container
3. Action (optional)



## SWITCHES

---



### Usage

Switches are the preferred way to adjust settings on mobile.

Use switches to:

- Toggle a single item on or off, on mobile and tablet
- Immediately activate or deactivate something

### Principles

#### *Familiar*

Switches have been in user interfaces for a long time and should be used as expected.

#### *Scannable*

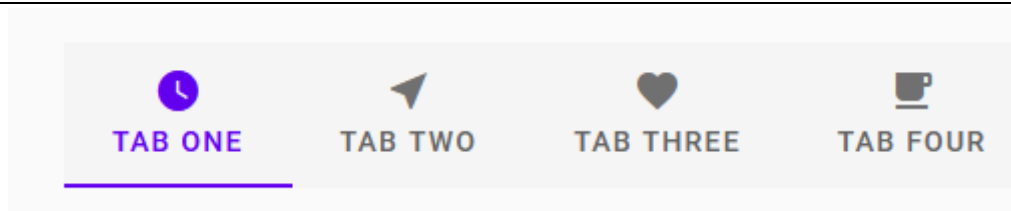
It should be visible at a glance if a switch has been selected, and selected items should be more visually prominent than unselected items.

#### *Efficient*

Switches make it easy to compare available options.

## TABS

---



### Usage

Tabs organize and allow navigation between groups of content that are related and at the same level of hierarchy.

### Principles

#### *Scalable*

As tabs can horizontally scroll, a UI can have as many tabs as needed.

#### *Informative*

Tabs organize content into categories to help users easily find different types of information.

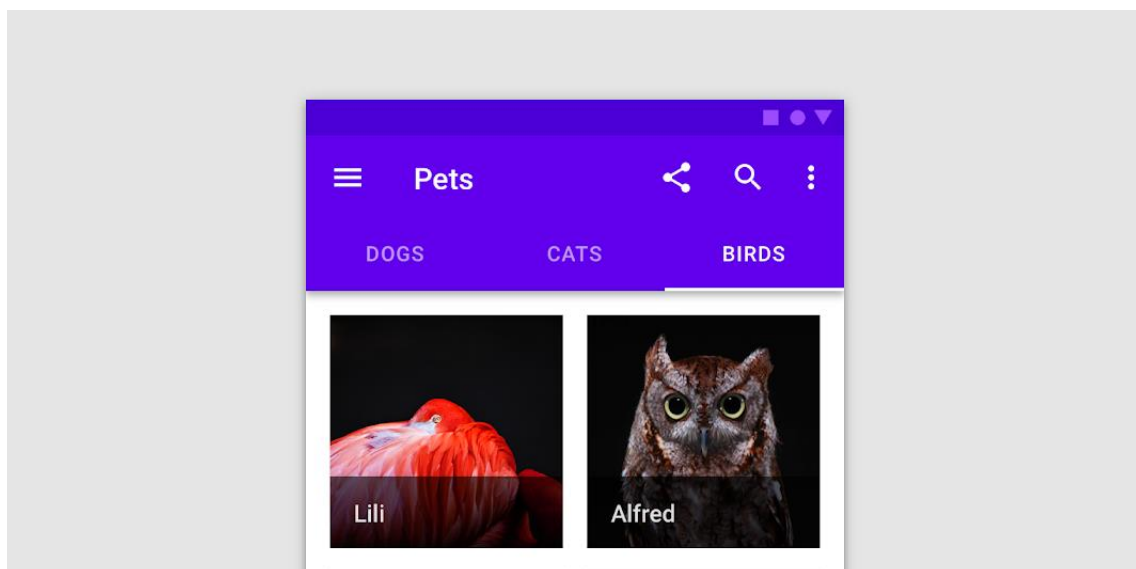
#### *Peers*

Tabs are displayed next to each other as peers, in categories of equal importance.

### Types

#### *Type 1: Fixed tabs*

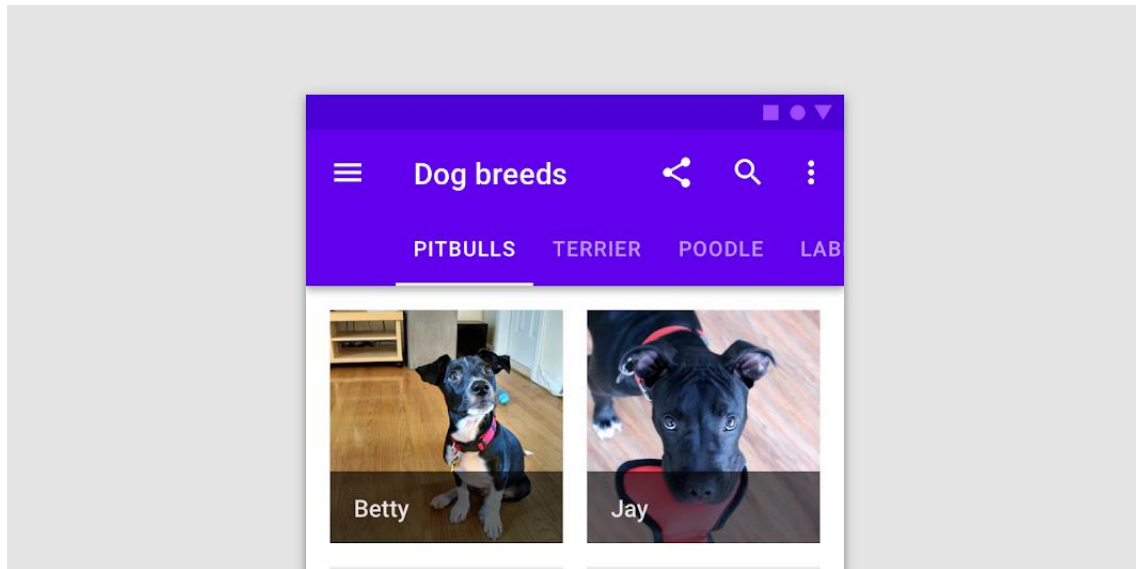
Fixed tabs display all tabs on one screen, with each tab at a fixed width. The width of each tab is determined by dividing the number of tabs by the screen width. They don't scroll to reveal more tabs; the visible tab set represents the only tabs available.





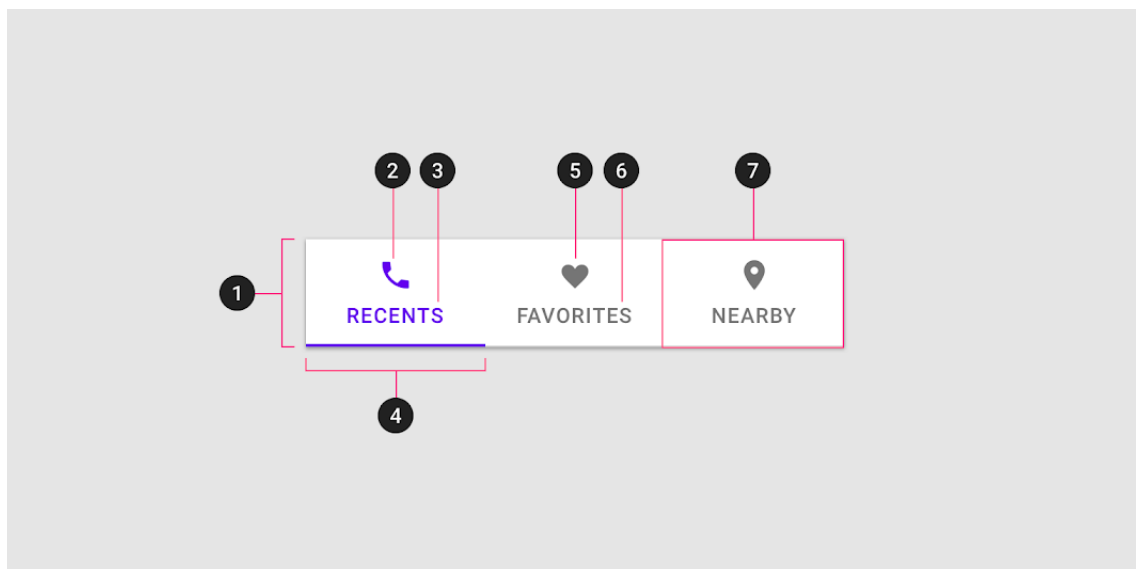
### Type 2: Scrollable tabs

Scrollable tabs are displayed without fixed widths. They are scrollable, such that some tabs will remain off-screen until scrolled



### Anatomy

1. Container
2. Active icon (Optional if there's a label)
3. Active text label (Optional if there's an icon)
4. Active tab indicator
5. Inactive icon (Optional if there's a label)
6. Inactive text label (Optional if there's an icon)
7. Tab item



## TEXT FIELDS

---



### Usage

Text fields allow users to enter text into a UI. They typically appear in forms and dialogs.

### Principles

#### *Discoverable*

Text fields should stand out and indicate that users can input information.

#### *Clear*

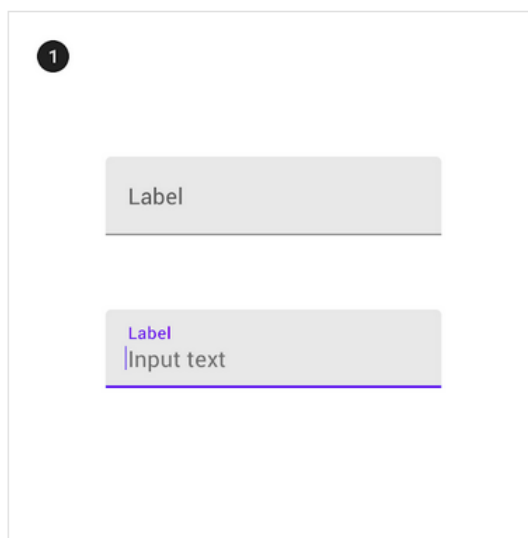
Text field states should be clearly differentiated from one another.

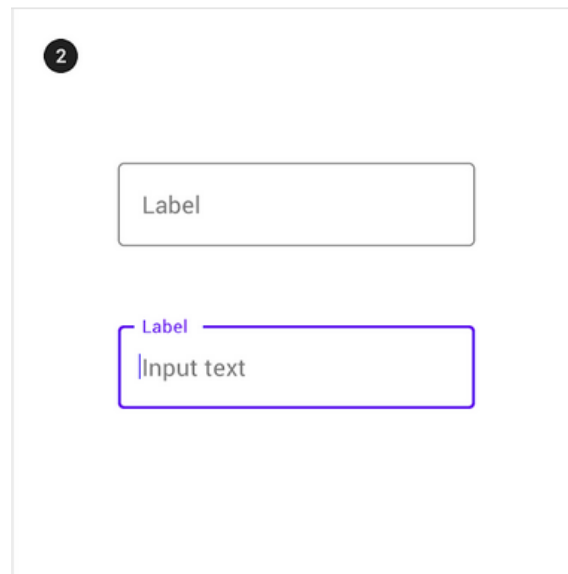
#### *Efficient*

Text fields should make it easy to understand the requested information and to address any errors.

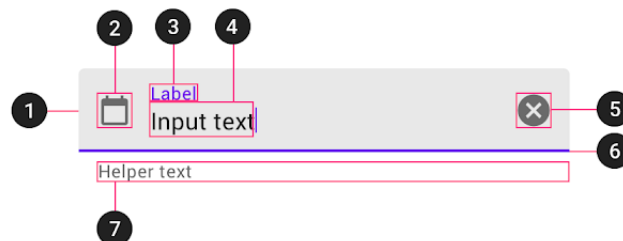
### Types

#### *Type 1. Filled text fields*

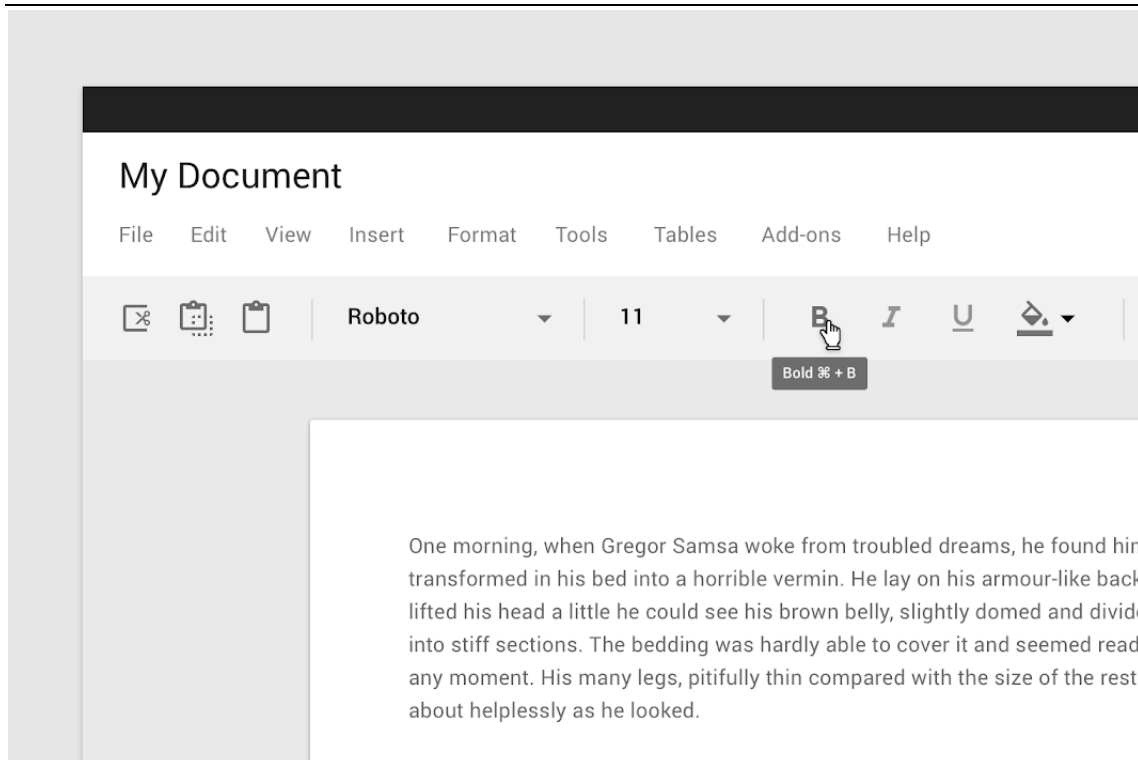


*Type 2: Outlined text fields***Anatomy**

1. Container
2. Leading icon (optional)
3. Label text
4. Input text
5. Trailing icon (optional)
6. Activation indicator
7. Helper text (optional)



## TOOLTIPS



### Usage

When activated, tooltips display a text label identifying an element, such as a description of its function.

### Principles

#### *Transient*

Tooltips appear on hover, focus, or touch, and disappear after a short duration.

#### *Paired*

Tooltips are always paired nearby the element with which they are associated.

#### *Succinct*

Tooltips only include short, descriptive text.