Material design

Introduction

We challenged ourselves to create a visual language for our users that synthesizes the classic principles of good design with the innovation and possibility of technology and science. This is material design. This spec is a living document that will be updated as we continue to develop the tenets and specifics of material design.

Goals

Create a visual language that synthesizes classic principles of good design with the innovation and possibility of technology and science.

Develop a single underlying system that allows for a unified experience across platforms and device sizes. Mobile precepts are fundamental, but touch, voice, mouse, and keyboard are all ﬁrst-class input methods.

Principles

Material is the metaphor

A material metaphor is the unifying theory of a rationalized space and a system of motion. The material is grounded in tactile reality, inspired by the study of paper and ink, yet technologically advanced and open to imagination and magic.

Surfaces and edges of the material provide visual cues that are grounded in reality. The use of familiar tactile attributes helps users quickly understand affordances. Yet the flexibility of the material creates new affordances that supersede those in the physical world, without breaking the rules of physics.

The fundamentals of light, surface, and movement are key to conveying how objects move, interact, and exist in space and in relation to each other. Realistic lighting shows seams, divides space, and indicates moving parts.

Bold, graphic, intentional

The foundational elements of print-based design – typography, grids, space, scale, color, and use of imagery – guide visual treatments. These elements do far more than please the eye. They create hierarchy, meaning, and focus. Deliberate color choices, edge-to-edge imagery, large-scale typography, and intentional white space create a bold and graphic interface that immerse the user in the experience.

An emphasis on user actions makes core functionality immediately apparent and provides waypoints for the user.

Motion provides meaning

Motion respects and reinforces the user as the prime mover. Primary user actions are inflection points that initiate motion, transforming the whole design.

All action takes place in a single environment. Objects are presented to the user without breaking the continuity of experience even as they transform and reorganize.

Motion is meaningful and appropriate, serving to focus attention and maintain continuity. Feedback is subtle yet clear. Transitions are efﬁcient yet coherent.

Environment

Material design is a three-dimensional environment containing light, material, and cast shadows.

All material objects have x, y, and z dimensions.

All material objects have a single z-axis position.

Key lights create directional shadows, and ambient light creates soft shadows.

Material thickness

1dp

Shadows

Shadows are created by the elevation difference between overlapping material.

3D world

The material environment is a 3D space, which means all objects have x, y, and z dimensions. The z-axis is perpendicularly aligned to the plane of the display, with the positive z-axis extending towards the viewer. Every sheet of material occupies a single position along the z-axis and has a standard 1dp thickness, equivalent to one pixel of thickness on screens with a pixel density of 160.

On the web, the z-axis is used for layering and not for perspective. The 3D world is emulated by manipulating the y-axis.

Light and shadow

Within the material environment, virtual lights illuminate the scene. Key lights create directional shadows, while ambient light creates soft shadows from all angles.

Shadows in the material environment are cast by these two light sources. In Android development, shadows occur when light sources are blocked by sheets of material at various positions along the z-axis. On the web, shadows are depicted by manipulating the y-axis only. The following example shows the card with a height of 6dp.

Shadow cast by key light

Shadow cast by ambient light

Combined shadow from key and ambient lights

Material properties

Material has certain immutable characteristics and inherent behaviors.

Understanding these qualities of material will help you manipulate material in a way that’s consistent with the vision of material design.

Material characteristics

Solid

Occupies unique points in space

Impenetrable

Mutable shape

Changes in size only along its plane

Unbendable

Can join to other material

Can separate, split, and heal

Can be created or destroyed

Moves along any axis

Physical properties

Material has varying x & y dimensions (measured in dp) and a uniform thickness (1dp).

Do. The height and width of material can vary. Don't.

Material is always 1dp thick. Material casts shadows.

Shadows result naturally from the relative elevation (z-position) between material elements.

Do.

Shadows depict the relative elevation between material elements.

Don't.

Shadows are never approximated by coloring material.

Content is displayed on material, in any shape and color. Content does not add thickness to material.

Do.

Material can display any shape and color.

Content can behave independently of the material, but is limited within the bounds of the material.

Do.

Content behavior can be independent of the behavior of material.

Material is solid.

Input events cannot pass through material.

Do.

Input events only affect the foreground material.

Don't.

Input events cannot pass through material.

Multiple material elements cannot occupy the same point in space simultaneously.

Do.

Using elevation to separate material elements is one method of preventing multiple material elements from occupying the same point in space simultaneously.

Don't.

Multiple material elements cannot occupy the same point in space simultaneously.

Material cannot pass through other material.

For example, one sheet of material cannot pass through another sheet of material when changing elevation.

Don't.

Material cannot pass through other material.

Transforming material

Material can change shape.

Material can change shape.

Material grows and shrinks only along its plane.

Do.

Material grows and shrinks only along its plane.

Material never bends or folds.

Don't.

Material never bends or folds.

Sheets of material can join together to become a single sheet of material.

Multiple sheets of material can join together to become a single sheet.

When split, material can heal. For example, if you remove a portion of material from a sheet of material, the sheet of material will become a whole sheet again.

Material can split and become whole again.

Movement of material

Material can be spontaneously generated or destroyed anywhere in the environment.

Material can be spontaneously generated or destroyed.

Material can move along any axis.

Material can move along various axes.

Z-axis motion is typically a result of user interaction with material.

Z-axis motion prompted by user interaction.

Elevation & shadows

Objects in material design possess similar qualities to objects in the physical world.

In the physical world, objects can be stacked or affixed to one another, but cannot pass through each other. Objects also cast shadows and reflect light.

Material design reflects these qualities to form a spatial model that is familiar to users and can be applied consistently across apps.

Elevation

Measured from the front of one surface to the front of another, an element’s elevation indicates the distance between surfaces and the depth of its shadow.

Resting elevation

All material elements have resting elevations. While components have consistent resting elevations across apps, they may have different resting elevations across platforms and devices.

Dynamic elevation offsets

Dynamic elevation offsets are the goal elevation that a component moves towards, relative to its resting state.

Elevation (Android)

Elevation is the relative depth, or distance, between two surfaces along the z-axis.

Specifications:

Elevation is measured in the same units as the x and y axes, typically in density-independent pixels (dp). Because material elements have depth (all material is 1dp thick), elevation is measured in distance from the top of one surface to the top of another.

A child object's elevation is relative to the parent object's elevation.

The images and values shown are for Android apps.

Multiple elevation measurements for two objects

Resting elevation

All material objects, regardless of size, have a resting elevation, or default elevation that does not change. If an object changes elevation, it should return to its resting elevation as soon as possible.

Desktop resting elevation is 2dp below the listed values to accommodate mouse and non-touch environments.

Component elevations:

Components maintain consistent resting elevations across apps. For example, the floating action button’s elevation does not vary from one app to another.

Components may have different resting elevations across platforms and devices, depending on the depth of the environment. For instance, TV has a greater depth than desktop as it has a larger screen and is viewed from further away. Similarly, both TV and desktop have a greater depth than mobile.

Responsive elevation and dynamic elevation offsets

Some component types have responsive elevation, meaning they change elevation in response to user input (e.g., normal, focused, and pressed) or system events. These elevation changes are consistently implemented using dynamic elevation offsets.

Dynamic elevation offsets are the goal elevation that a component moves towards, relative to the component’s resting state. They ensure that elevation changes are consistent across actions and component types. For example, all components that lift on press have the same elevation change relative to their resting elevation.

Once the input event is completed or cancelled, the component will return to its resting elevation.

Avoiding elevation interference

Components with responsive elevations may encounter other components as they move between their resting elevations and dynamic elevation offsets. Because material cannot pass through other material, components avoid interfering with one another any number of ways, whether on a per-component basis or using the entire app layout.

On a component level, components can move or be removed before they cause interference. For example, a floating action button (FAB) can disappear or move off-screen before a user picks up a card, or it can move if a snackbar appears.

On the layout level, design your app layout to minimize opportunities for interference. For example, position the FAB to one side of a stream of cards so the FAB won’t interfere when a user tries to pick up one of cards.

Elevation (dp)

Component

24

Dialog

Picker

16

Nav drawer

Right drawer

Modal bottom Sheet

12

Floating action button (FAB - pressed)

9

Sub menu (+1dp for each sub menu)

8

Bottom navigation bar

Menu

Card (when picked up)

Raised button (pressed state)

6

Floating action button (FAB - resting elevation)

Snackbar

4

App Bar

3

Refresh indicator

Quick entry / Search bar (scrolled state)

2

Card (resting elevation) \*

Raised button (resting elevation)\*

Quick entry / Search bar (resting elevation)

1

Switch

Avoiding elevation interference

Components with responsive elevations may encounter other components as they move between their resting elevations and dynamic elevation offsets. Because material cannot pass through other material, components avoid interfering with one another any number of ways, whether on a per-component basis or using the entire app layout.

On a component level, components can move or be removed before they cause interference. For example, a floating action button (FAB) can disappear or move off-screen before a user picks up a card, or it can move if a snackbar appears.

On the layout level, design your app layout to minimize opportunities for interference. For example, position the FAB to one side of a stream of cards so the FAB won’t interfere when a user tries to pick up one of cards.

Elevation (dp)

Component

24

Dialog

Picker

16

Nav drawer

Right drawer

Modal bottom Sheet

12

Floating action button (FAB - pressed)

9

Sub menu (+1dp for each sub menu)

8

Bottom navigation bar

Menu

Card (when picked up)

Raised button (pressed state)

6

Floating action button (FAB - resting elevation)

Snackbar

4

App Bar

3

Refresh indicator

Quick entry / Search bar (scrolled state)

2

Card (resting elevation) \*

Raised button (resting elevation)\*

Quick entry / Search bar (resting elevation)

1

Switch

Component elevation comparisons

The following diagram compares component resting elevations and dynamic elevation offsets.

In this diagram, only the elevation dimensions and layout for components are accurate. Other dimensions and overall layout of components are for illustration only.

An example app layout with cards and a floating action button, along with a cross-section diagram of its component elevations along its z-axis

An example app layout with an open navigation drawer, along with a cross-section diagram of its component elevations along its z-axis.

Shadows

Shadows provide important visual cues about objects’ depth and directional movement. They are the only visual cue indicating the amount of separation between surfaces. An object’s elevation determines the appearance of its shadow.

Don't.

Without a shadow, nothing indicates that the floating action button is separate from the background surfaces.

Don't.

Crisp shadows indicate the floating action button and the blue sheet are separate elements. However, their shadows are so similar that they imply they are both at the same elevation.

Do.

Softer, larger shadows indicate the floating action button is at a higher elevation than the blue sheet, which has a crisper shadow.

In motion, shadows provide useful cues about an object’s direction of movement and whether the distance between surfaces is increasing or decreasing.

Don't.

Without a shadow to indicate elevation, it’s unclear whether this square is increasing in size or increasing its elevation.

Do.

The shadow grows softer and larger as the object’s elevation increases and grows crisper and smaller as the elevation decreases.

Do.

In this case, the consistent shadow helps the user understand that the object is changing shape as opposed to changing elevation.

Component reference shadows

The following component shadows should be used as canonical references. If there are any differences between the following reference shadows and component shadows found elsewhere within this spec, defer to these reference shadows.

App bar

4dp

Example of an app bar

Raised button

Resting state: 2dp

Pressed state: 8dp

For desktop only, raised buttons can have an elevation of:

Resting state: 0dp

Pressed state: 2dp

See raised buttons for more information

Floating action button (FAB)

Resting state: 6dp

Pressed state: 12dp

Card

Resting state: 2dp

Raised state: 8dp

For Desktop only, see Content blocks.

Menus and sub menus

Menus: 8dp

Sub menus: 9dp (+1 dp for each sub menu)

Dialogs

24dp

Nav Drawer & Right drawer

16dp

Modal bottom sheet

16dp

Refresh indicator

3dp

Quick entry/Search bar

Resting state: 2dp

Scrolled state: 3dp

Snackbar

6dp

Switch

1dp

Object relationships

Object hierarchy

How you organize objects, or collections of objects, in an app determines how they move in relation to one another. Objects can move independently of each other or be constrained by objects higher in the hierarchy.

All objects are part of a hierarchy described in terms of a parent-child relationships. The “child” in each of these relationships refers to an element that is a subordinate to its “parent” element. Objects can be children of either the system or another object.

Parent-child specifics:

Each object has one parent.

Each object may have any number of children.

Children inherit transformative properties from their parent, such as position, rotation, scale, and elevation.

Siblings are objects at the same level of hierarchy.

Exceptions

Items parented to the root, such as primary UI elements, move independently of other objects. For example, the floating action button does not scroll with content. Other elements include:

An app’s side nav drawer

The action bar

Dialogs

Interaction

How objects interact with one another is determined by their place in the parent-child hierarchy.

For example:

Children have minimal z-axis separation from their parent; other objects do not get inserted between parents and children.

In a scrolling card collection, the cards are siblings of each other, so they all move together in tandem. They are children of the card collection object that controls their movement.

Elevation

How you determine the elevation of objects – their position in z-space – depends on the content hierarchy you want to express and whether an object needs to move independently of other objects.

As the parent sheet scrolls, the raised button (its child) scrolls off screen with it.

As the card collection scrolls off-screen, its child cards scroll off-screen with it. The floating action button remains in place because its parent is not being scrolled.

What’s new

April 2017

Significant updates

Updated guidance on the Material Design color system, usability, and a brand new Color Tool for creating color palettes and testing accessibility

Text fields expands on usage basics and introduces text field boxes for increasing text field discoverability

Bidirectionality has added resource links to developer guidance on right-to-left (RTL) icons

Previous updates

December 2016

New sections

Platforms discusses when and how to adapt apps to Material Design, and when to adhere to platform conventions

App shortcut icons contains specs on how to create app shortcut icons for the home screen

Help and feedback describes how to indicate, craft, and place help content

Significant updates

Bidirectionality has updated guidance on mirroring UI content and icons in right-to-left (RTL) languages

Accessibility contains new guidance on sound, controls, labeling elements, and testing

Selection includes detailed examples on item selection, toggling, and styling

August 2016

New sections

Notifications have been revised for Android N, including new content areas, interactions, user controls, and short text input

Widgets contains guidance on user engagement, retention, and acquisition

Confirmation and acknowledgement covers when and how to acknowledge user actions

Significant updates

Navigation now includes details on how to use the Up and Back buttons as part of your navigation strategy

Using full-screen mode for Lean Back, Immersive mode, and Lights out interactions

May 2016

New sections

Motion includes detailed examples and specs on material motion principles, duration and easing, transformations, choreography, and customization.

Growth & communications contains guidance on user engagement, retention, and acquisition.

Expansion panels guidance describes lightweight containers used for creating flows and editing content.

March 2016 (for Pi Day)

New sections

Bottom navigation bars allow switching between top-level views

Split screens allow the display of two activities at once (Android only)

Significant updates

Expanded accessibility guidance on color and contrast, sound, motion, hierarchy and more

How to display password input on text fields

February 2016

Significant updates

Expanded guidance on customizing your app’s color palette, with more examples

Color specs for dialog errors, text selection, and text field labels

Sticker sheets now include text field colors, month and time pickers, and bottom toolbars (for Photoshop, Illustrator, and Sketch)

December 2015

Significant updates

Responsive UI includes a pattern on positioning and more examples of responsive patterns

Text fields adds guidance on required text fields

Whiteframes have been added for Sketch

Punctuation has new guidance on en dashes and parentheses

October 2015

New sections

Steppers describes how different types of steppers convey progress through numbered steps

Significant updates

Chips adds guidance on chip movement and deletion

Notifications includes formatting guidance for timestamps

September 2015

Significant updates

Navigation adds guidance on navigation hierarchy and structure, new navigation patterns, and combined navigation patterns

UI color application describes how transparent text maintains legibility against different background colors

Buttons contains guidance on toggle buttons and flat button behaviors

Tabs includes guidance on text sizing in tabs

August 2015

New sections

Permissions contains guidance on permissions groups, runtime permissions, request patterns, and how to address permissions when they are denied

Significant updates

Bottom sheets includes descriptions of persistent and modal bottom sheets, alternative display options, and desktop and tablet display using increments

Settings includes a usage overview

July 2015

New sections

Notifications contains guidance for tailoring notifications to users, including peeking notifications

Fingerprint contains guidance on implementing Fingerprint as an authentication option

Significant updates

Cards includes more image examples of card UI controls, sliders, and tabs

Launch screens includes guidance for implementing placeholder UIs

Settings includes additional guidance for labels and secondary text

May 2015

New sections

Adaptive UI includes responsive layout guidance, grids, breakpoints, and patterns

Empty states contains guidance for states when regular content can’t be shown

Launch screens includes branding and placeholder UI content

Navigation

Significant updates

Elevation and shadows has been renamed from “Objects in 3D space,” adding new details about elevation changes

Authentic motion describes easing terms in more detail

April 2015

New sections

Data tables

Units and measurements

Floating Action Button has been moved into its own section, adding new details about behavior and transitions

App structure

Component sticker sheet for Adobe After Effects

Data truncation and redaction

Writing guidelines for Settings

Significant updates

Typography adds further guidance on style and line height for dense and tall languages

Cards includes more specs for laying out actions and content

Dialogs contains additional layout guidance

Tabs adds guidance around label content and more complete sizing specs

Scrolling techniques adds guidance for overlapping content

Motion. Material motion

Motion in the world of material design is used to describe spatial relationships, functionality, and intention with beauty and fluidity.

Why does motion matter?

Motion shows how an app is organized and what it can do.

Motion provides:

Guided focus between views

Hints at what will happen if a user completes a gesture

Hierarchical and spatial relationships between elements

Distraction from what’s happening behind the scenes (like fetching content or loading the next view)

Character, polish, and delight

Related

Making Motion Meaningful

Motion designers from around Google share some of their most moving product features.

How does material move?

The material environment draws inspiration from real-world forces, such as gravity and friction. These forces are reflected in the way user input affects elements on screen and how elements react to each other.

Material in motion has the following characteristics:

Responsive

Material is full of energy. It quickly responds to user input precisely where the user triggers it.

Larger animations on mobile devices are 300-400ms long. Smaller animations can be as short as 150-200ms. Animations longer or shorter than these can feel sluggish or difficult to follow.

See more in Common Durations.

Ink ripples confirm user input by immediately expanding outward from the point of touch. The card lifts to indicate an active state.

See more in Radial reaction.

Show the connection between new surfaces and the element or action that creates them.

See more in Creation.

Natural

Material depicts natural movement inspired by forces in the real world.

In the real world, an element’s ability to speed up or slow down quickly is affected by weight and surface friction. In a similar way, starts and stops do not occur instantaneously in material design.

See more in Natural easing curves.

Real-world forces, like gravity, inspire an element’s movement along an arc rather than in a straight line.

See more in Movement.

Material transformations follow an arc of movement.

See more in Transforming material.

Aware

Material is aware of its surroundings, including the user and other material around it. It can be attracted to elements and respond appropriately to user intent.

See more in Choreography.

As elements transition into view, they and their surroundings are choreographed in a way that defines their relationships.

Material can push other material out of the way.

Elements may attract other elements and join with them as they approach one another.

Intentional

Material in motion guides focus to the right spot at the right time.

See more in Continuity.

A transition helps guide the user to the next step of an interaction.

Movement can communicate different signals, such as whether an action is unavailable.

Animation can bring focus to elements that need user attention.

What makes a good transition?

Successful motion design possesses the following characteristics:

Motion is quick

An interaction shouldn't keep the user waiting longer than necessary.

Do.

Animate quickly so that the user never has to wait for the animation to finish.

Don't.

Staggering and slowing the movement of many elements can lengthen the duration.

Motion is clear

Transitions should be clear, simple, and coherent. They should avoid doing too much at once.

Do.

Maintain a clear path into the next view, even while elements are choreographed as a group.

Don't.

Transitions can get confusing when multiple items need to move in different directions or cross paths.

Motion is cohesive

Material elements are unified by their speed, responsiveness, and intention. Any customizations to your app's motion experience should be consistent throughout the app.

While these apps have different functions, their similar animation experiences make them feel related.

Implications of motion

The benefits of these motion patterns are observable in the following two examples, in which an app that follows these patterns is compared with an app that does not.

Do.

During the transition, the user is guided to the next view. The surface transforms to communicate hierarchy. Loading occurs behind the scenes to reduce perceived latency.

Don't.

It is unclear how the new view is related to the old because no transition occurs and there isn’t a clear focal point. Any sense of hierarchy isn’t communicated, and loading is more apparent by the display of a circular spinner.

Duration & easing

Material in motion is responsive and natural. Use these easing curves and duration patterns to create smooth and consistent motion.

Easing curves

Standard curve

Deceleration curve

Acceleration curve

Sharp curve

Speed

When elements move between positions or states, the movement should be fast enough that it doesn't cause waiting, but slow enough that the transition can be understood. Keep transitions short as users will see them frequently.

Do.

Animate quickly so that the user doesn’t have to wait for the animation to finish.

Don't.

Don’t animate slowly as it creates unnecessary lag.

Dynamic durations

Rather than using a single duration for all animations, adjust each duration to accommodate the distance travelled, an element's velocity, and surface changes.

Objects leaving the screen may have shorter durations, as they require less attention.

Use longer durations when objects need to travel large distances or have dramatic changes in surface area.

Use shorter durations when objects travel short distances or have minor changes in surface area, so that the movement doesn't appear too slow.

Common durations

Mobile

Transitions on mobile typically occur over 300ms, within this margin of variance:

Large, complex, full-screen transitions may have longer durations, occurring over 375ms

Elements entering the screen occur over 225ms

Elements leaving the screen occur over 195ms

Transitions that exceed 400ms may feel too slow.

Larger screens

Elements travelling longer distances on larger screens have higher peak velocities than those going shorter distances over the same time period. Larger screens should have longer durations so that movements aren't too fast.

Tablets

Durations on tablet should be about 30% longer than on mobile. For example, a 300ms mobile duration would increase to 390ms on tablet.

Wearables

Durations on wearables should be about 30% shorter than those on mobile. For example, a 300ms mobile duration would be 210ms on wearables.

Because material must move greater distances on larger devices, transitions may have slightly slower durations than on smaller devices.

Desktop

Desktop animations should be faster and simpler than their mobile counterparts. These animations should last 150ms to 200ms.

Because desktop transitions may be less noticeable, they should be immediately responsive and faster than their mobile counterparts.

Complex web transitions often result in dropped frames (unless they are built for GPU acceleration). Shorter durations will make these less noticeable because the animation completes faster.

Transitions appear much faster on desktop.

Natural easing curves

These natural easing curves affect an element’s speed, opacity, and scale.

Acceleration and deceleration changes should be smooth across the duration of an animation so that movement doesn't appear mechanical.

Motion appears more natural and delightful when acceleration and deceleration occur asymmetrically.

Easing curves

Easing curves may be named differently depending on the platform or software used. These guidelines will refer to them as standard, deceleration, acceleration, and sharp curves.

Standard curve

The standard curve (also referred to as “ease in out”) is the most common easing curve. Elements quickly accelerate and slowly decelerate between on-screen locations. It applies to growing and shrinking material, among other property changes.

See more in Movement within screen bounds.

Platform

Protocol

Android

FastOutSlowInInterpolator

iOS

[[CAMediaTimingFunction alloc] initWithControlPoints:0.4f:0.0f:0.2f:1.0f]

CSS

cubic-bezier(0.4, 0.0, 0.2, 1);

After Effects

Outgoing Velocity: 40%

Incoming Velocity: 80%

Deceleration curve

Using the deceleration curve (also referred to as “ease out”) elements enter the screen at full velocity and slowly decelerate to a resting point.

During deceleration, elements may scale up either in size (to 100%) or opacity (to 100%). In some cases, when elements enter the screen at 0% opacity, they may slightly shrink from a larger size upon entry.

See more in Movement in and out of screen bounds.

Platform

Protocol

Android

LinearOutSlowInInterpolator

iOS

[[CAMediaTimingFunction alloc] initWithControlPoints:0.0f:0.0f:0.2f:1.0f]

CSS

cubic-bezier(0.0, 0.0, 0.2, 1);

After Effects

Outgoing Velocity: 0%

Incoming Velocity: 80%

Acceleration curve

Using the acceleration curve (also referred to as “ease in”) elements leave the screen at full velocity. They do not decelerate when off-screen.

They accelerate at the beginning of the animation and may scale down in either size (to 0%) or opacity (to 0%). In some cases, when elements leave the screen at 0% opacity, they may also slightly scale up or down in size.

See more in Movement in and out of screen bounds.

Platform

Protocol

Android

FastOutLinearInInterpolator

iOS

[CAMediaTimingFunction alloc] initWithControlPoints:0.4f:0.0f:1.0f:1.0f]

CSS

cubic-bezier(0.4, 0.0, 1, 1);

After Effects

Outgoing Velocity: 40%

Incoming Velocity: 0%

Sharp curve

Using the sharp curve (also referred to as “ease in out”) elements quickly accelerate and decelerate. It is used by exiting elements that may return to the screen at any time.

Elements may quickly accelerate from a starting point on-screen, then quickly decelerate in a symmetrical curve to a resting point immediately off-screen. The deceleration is faster than the standard curve since it doesn't follow an exact path to the off-screen point. Elements may return from that point at any time.

See more in Movement in and out of screen bounds.

Platform

Protocol

Android

-

iOS

-

CSS

cubic-bezier(0.4, 0.0, 0.6, 1);

After Effects

Outgoing Velocity: 40%

Incoming Velocity: 40%

Movement

Material in motion abides by forces similar to those of the real world, like gravity.

On-screen movement

Arc upward

Arc downward

In-and-out screen movement

Independent movement

Relative movement

Movement within screen bounds

The movement of an element between two points within the bounds of the screen follows a natural, concave arc. All on-screen movements use the standard curve.

Arc upward

Rising against gravity in the real world requires effort. Elements moving upward on the screen should similarly depict effort during acceleration through a slower upward movement.

Do.

When moving diagonally upward, begin with a shallow ascent and end with a steep ascent.

Don't.

Don’t start an element’s upward movement with a steep ascent.

Arc downward

Falling elements in the real world are accelerated by gravity. Elements moving downward on screen should depict less effort through a faster downward movement.

Do.

When moving diagonally downward, descent should begin steeply and end shallowly.

Don't.

Don’t start an element’s downward movement with a shallow descent.

When not to arc

Elements moving along a single axis (either horizontally or vertically, but not both) do not follow an arc. These movements are simpler and may move at a slightly faster speed.

Do.

Keep the path for single-axis movement straight.

Don't.

Don’t apply an unnatural arc to single-axis movement.

Elements entering and exiting the screen also move along a single axis.

Do.

Linear entrances are easy to follow with a clearly defined entry point.

Don't.

Don’t arc the path of an element if it is moving off-screen, as it complicates the entry point.

Movement in and out of screen bounds

Independent movement

Elements entering and exiting the screen are referred to as independent elements as they don't affect the position of other on-screen content.

Entering the screen

Elements entering the screen use the deceleration curve for a speedy entrance, indicating that they had been travelling at peak velocity.

Permanently leaving the screen

Elements permanently leaving the screen use the acceleration curve to speed off-screen over a slightly shorter duration, as they will not be returning and require less user focus.

Do.

Apply the deceleration curve to items entering the screen that do not affect the position of any other elements around it. On mobile, this transition typically occurs over 225ms. Apply the acceleration curve to these same items when they permanently leave the screen. On mobile, this transition typically occurs over 195ms.

Don't.

Don’t use the standard curve for either these cases. For entrances, slow acceleration is unnecessary as focus should be on the final resting point. For exits, slow deceleration is unnecessary as the ending point isn’t visible.

Temporarily leaving the screen

Elements temporarily leaving the screen should use the sharp curve, as they may return to the screen at any time and should appear nearby and within reach.

Do.

Apply the sharp curve to items temporarily leaving the screen that may return from the same exit point. When they return, use the deceleration curve. On mobile, this transition typically occurs over 300ms.

Don't.

Apply the acceleration curve to items permanently leaving the screen, or if they are to return from a different location. If they return, use the deceleration curve.

Relative movement

Entering or exiting elements that move other on-screen elements do so along a smooth easing curve, so that they remain minimally disruptive and avoid eye-catching, dramatic movement.

The standard curve is used for moving elements both in and out of the bounds of the screen. This curve has a slightly longer duration compared to independent elements.

Do.

Since the on-screen floating action button’s movement is influenced by the entering card, the standard curve is applied to the movement of both elements to maintain a smooth start and stop. On mobile, this transition typically occurs over 300ms.

Don't.

Using deceleration and acceleration curves for the upward and downward movement cause the on-screen floating action button to abruptly start moving upward and abruptly stop moving downward. This movement is harsh and disruptive.

Transforming material

Material can make surfaces feel alive by multiplying and dividing them, and changing their shape and size.

Rectangular transformation

Asymmetric

Symmetric

Radial transformation

Symmetrical and circular

Rectangular transformation

When material changes shape and size, its width and height change asynchronously along a motion curve. These changes are choreographed with nearby surfaces inside or around the changing element.

All surface transformations that occur on-screen, such as the ones below, use the standard curve.

Transforming width and height

Elements may be expanded or collapsed using either asymmetric or symmetrical width and height changes.

Asymmetric transformations involve the width and height changes at different rates. They work best when multiple elements or position changes are involved.

Asymmetric transformation

To expand an element’s size, begin transforming the width of an element slightly before transforming the height. To collapse an element’s size, begin transforming the height slightly before the width.

Symmetrical transformations involve width and height changes occurring at the same rate. They are better for changes to a single element that occur along a single axis.

Symmetric transformation

Transforming width and height at the same time is a less detailed animation approach best suited to simple shape changes. These transformations can have slightly shorter durations than asymmetric ones.

When an element expands asynchronously, the items it contains (such as text or images) change size at a constant aspect ratio to prevent unnatural stretching. For more information on how to transition content within a surface, see Choreography.

Content (such as a full-width image) transforms at a constant aspect ratio, even as its container (such as a larger card) transforms asynchronously along a motion curve.

Containers with full-bleed content (such as a full-bleed image) may expand synchronously.

Radial transformation

Radial transformations are symmetrical, circular visualizations that originate from a user’s point of touch. They are commonly used on circular surfaces that morph into other shapes.

Do.

Radial transformation should be used on circular surfaces that morph into rectangular surfaces, or for creating new surfaces from the point of input.

Don't.

Don’t use a radial transformation when transforming between two rectangular shapes.

Don't.

Don’t expand an oval’s width and height asynchronously.

Don't.

Don’t transform complex shapes.

Transformations can originate either from the object’s current location or from the center of the final surface being formed.

During expansion, the floating action button moves in an arc towards its destination as it expands into a card.

The center of the floating action button stays in place during the transformation, creating a subtler expansion effect.

Joining & Dividing

Joining

Material may join with other material, or it may divide into multiple pieces. As two pieces of material approach one another, their edges meet and margins overlap before the movement completes.

Dividing

When material divides into multiple pieces, the pieces begin separation at the start of the movement.

Shadows

Shadows from separated material do not appear above sibling elements.

Example of material joining and dividing

Choreography

Material in motion guides the user’s focus through interactions by sharing elements across transitions.

Surface creation

Stagger the choreography of new surfaces being created.

Radial reaction

Connect user input to surface reactions with touch ripples.

Continuity

Maintain a clear focal point during transitions by carefully selecting the number and type of elements shared across the transitions.

All content elements are shared

While a surface is expanding, a significant number of elements should remain visible during the transition.

Complex transitions should keep a single element visible (see below).

All elements from this collapsed card form the header of the expanded card.

Few or no content elements are shared

While expanding a surface, if only a single element will be present after the transition, that element should be the focal point of the transition, controlling all other elements.

Anchoring non-shared content to a single shared element’s movement.

If there are no shared elements between views, anchor all crossfading elements to the surface’s vertical movement. The surface crops the content within.

Anchoring all non-shared content to the vertical movement of the surface expansion.

No shared surfaces (Not recommended)

If there are no common surfaces shared across the transition, a new surface enters the screen from another location, creating a new focal point.

Multiple shared elements

When multiple elements remain visible during a transition, only the most important ones should be included. Some elements may disappear during the transition but reappear once the transition completes, if they are too distracting during the transition itself.

Do.

Guide the user’s focus to the next view using the most important shared element.

Don't.

Avoid creating scenes without focus, such as by using many shared elements or by letting elements cross paths during the transition.

Layout awareness

When any element doesn't completely load by the time a transition completes, allow sufficient space in the location where the element will appear. This prevents jarring layout shifts when the elements finally appear, and avoids user frustration when a touch target suddenly jumps away.

Do.

Make room for the unloaded elements during the transition and gracefully display them in place once they fully load.

Don't.

Don’t shift the layout as elements load into view, as this distracts and causes a shift in focus.

Creation

New material surfaces, and their content, may be created from nothing.

Creating new surfaces

Associate newly created surfaces to the element or action that creates them. New surfaces usually emerge from radial or rectangular expansions from the point of touch.

Do.

This menu appears from the touch point, tying the element to the point of touch.

Don't.

This menu appears far away from the touch point that triggered it, which breaks its relationship with the input method.

Choreographing surfaces

When multiple new surfaces are created at the same time, quickly stagger the appearance of each. Create a clear, smooth focal path in a single direction.

Do.

List items have a slightly staggered entrance. Grid items populate left to right, and top to bottom.

Don't.

Items shouldn’t appear within the list all at once. It’s overwhelming and eliminates a clear focus.

Don't.

Items that have a slightly staggered entrance shouldn’t appear in an order that’s confusing to follow.

Don't.

Do not wait for each item to fully animate before introducing the next item. Begin each item’s staggered entrance no more than 20ms apart.

Autonomous surface creation

Surfaces created without user input, or without a point of origin, should do so using an elegant combination of fading, position, and scaling transformations.

Do.

An automatically triggered surface appears smoothly and quickly.

Don't.

Too much animation during surface creation is distracting.

Radial reaction

Clarify the connection between user input and surface reactions using radial action.

User input

Connect user input to screen reactions by using touch ripples to both indicate the point of touch, and to confirm that touch input was received. For touch or mouse, this occurs at the point of contact.

Screen reactions occurring near the point of touch should occur sooner than reactions further away.

Use touch ripples to connect screen reactions to the point of touch.

Starting from the point of touch, the app bar changes color with the new color rippling outwards.

Images in this grid fade out based on the radial reaction around the tap point.

Creative customization

Animation may be used in a wide range of scales and contexts to unite beauty and function.

Icon types

System icons may perform dual functions.

Product icons should demonstrate polish.

Icons

System icons

An icon may serve dual functions. For example, a menu icon may smoothly transition to a playback control, and back again. This effect both informs the user of the button's function while adding an element of wonder to the interaction.

Transforming the menu icon into the back arrow signifies that the two actions are connected, and that one cannot exist while the other is present. The back button must be pressed in order to return to the top-level view with the menu icon.

Transforming the play button to a pause button signifies that the two actions are linked, and that pressing one makes the other one visible.

When pressing this floating action button, the plus sign transforms into a pencil. This indicates that the pencil is the primary creation method.

Product icons

Animated product icons should demonstrate polish and refinement from the moment the app launches.

The Inbox product icon animation displays when the user opens the app for the first time and is asked to sign in.

The Calendar product icon animation displays when the user’s calendar data is loaded for the first time.

The Photos product icon animation displays when the user opens the app for the first time and enters the warm welcome.

The Gmail product icon animation has a slightly more playful personality while still keeping a similar reveal, and is also displayed when the user opens the app for the first time and is asked to sign in.

Illustrations

Subtle animation in imagery and illustrations can add a sense of playfulness to the user experience.

When displaying an unexpected error, use it as an opportunity to delight with animation.

Animating the bicycle while Calendar schedules a goal helps entertain the user while the task finishes.

Rewarding the user with an animation when they accomplish personal goals, like reaching Inbox Zero, can create deeper engagement with app features.

Animating an illustration for a new feature can educate the user about the feature in a memorable way.

Style. Color

Color in material design is inspired by bold hues juxtaposed with muted environments, deep shadows, and bright highlights.

Color Tool

The color tool helps you create, share, and apply color palettes to your UI, as well as measure the accessibility level of any color combination.

Create color schemes

Create color schemes that include darker and lighter variations of your primary and secondary colors.

Test accessibility

Check if text is accessible on different-colored backgrounds, as measured using the Web Content Accessibility Guidelines legibility standards.

Preview your UI in color

Preview the look of your color scheme across a range of Material Design components, with editable HTML, CSS, or JavaScript in Codepen.

Color Tool

Create color palettes for your UI and test color accessibility

Color palette

The color palette

This color palette comprises primary and accent colors that can be used for illustration or to develop your brand colors. They’ve been designed to work harmoniously with each other. The color palette starts with primary colors and fills in the spectrum to create a complete and usable palette for Android, Web, and iOS. Google suggests using the 500 colors as the primary colors in your app and the other colors as accents colors.

Themes enable consistent app styling through surface shades, shadow depth, and ink opacity.

Download color swatches

0.02 MB (.zip)

Red

500 #F44336

50#FFEBEE

100#FFCDD2

200#EF9A9A

300#E57373

400#EF5350

500#F44336

600#E53935

700#D32F2F

800#C62828

900#B71C1C

A100#FF8A80

A200#FF5252

A400#FF1744

A700#D50000

Pink

500 #E91E63

50#FCE4EC

100#F8BBD0

200#F48FB1

300#F06292

400#EC407A

500#E91E63

600#D81B60

700#C2185B

800#AD1457

900#880E4F

A100#FF80AB

A200#FF4081

A400#F50057

A700#C51162

Purple

500 #9C27B0

50#F3E5F5

100#E1BEE7

200#CE93D8

300#BA68C8

400#AB47BC

500#9C27B0

600#8E24AA

700#7B1FA2

800#6A1B9A

900#4A148C

A100#EA80FC

A200#E040FB

A400#D500F9

A700#AA00FF

Deep Purple

500 #673AB7

50#EDE7F6

100#D1C4E9

200#B39DDB

300#9575CD

400#7E57C2

500#673AB7

600#5E35B1

700#512DA8

800#4527A0

900#311B92

A100#B388FF

A200#7C4DFF

A400#651FFF

A700#6200EA

Indigo

500 #3F51B5

50#E8EAF6

100#C5CAE9

200#9FA8DA

300#7986CB

400#5C6BC0

500#3F51B5

600#3949AB

700#303F9F

800#283593

900#1A237E

A100#8C9EFF

A200#536DFE

A400#3D5AFE

A700#304FFE

Blue

500 #2196F3

50#E3F2FD

100#BBDEFB

200#90CAF9

300#64B5F6

400#42A5F5

500#2196F3

600#1E88E5

700#1976D2

800#1565C0

900#0D47A1

A100#82B1FF

A200#448AFF

A400#2979FF

A700#2962FF

Light Blue

500 #03A9F4

50#E1F5FE

100#B3E5FC

200#81D4FA

300#4FC3F7

400#29B6F6

500#03A9F4

600#039BE5

700#0288D1

800#0277BD

900#01579B

A100#80D8FF

A200#40C4FF

A400#00B0FF

A700#0091EA

Cyan

500 #00BCD4

50#E0F7FA

100#B2EBF2

200#80DEEA

300#4DD0E1

400#26C6DA

500#00BCD4

600#00ACC1

700#0097A7

800#00838F

900#006064

A100#84FFFF

A200#18FFFF

A400#00E5FF

A700#00B8D4

Teal

500 #009688

50#E0F2F1

100#B2DFDB

200#80CBC4

300#4DB6AC

400#26A69A

500#009688

600#00897B

700#00796B

800#00695C

900#004D40

A100#A7FFEB

A200#64FFDA

A400#1DE9B6

A700#00BFA5

Green

500 #4CAF50

50#E8F5E9

100#C8E6C9

200#A5D6A7

300#81C784

400#66BB6A

500#4CAF50

600#43A047

700#388E3C

800#2E7D32

900#1B5E20

A100#B9F6CA

A200#69F0AE

A400#00E676

A700#00C853

Light Green

500 #8BC34A

50#F1F8E9

100#DCEDC8

200#C5E1A5

300#AED581

400#9CCC65

500#8BC34A

600#7CB342

700#689F38

800#558B2F

900#33691E

A100#CCFF90

A200#B2FF59

A400#76FF03

A700#64DD17

Lime

500 #CDDC39

50#F9FBE7

100#F0F4C3

200#E6EE9C

300#DCE775

400#D4E157

500#CDDC39

600#C0CA33

700#AFB42B

800#9E9D24

900#827717

A100#F4FF81

A200#EEFF41

A400#C6FF00

A700#AEEA00

Yellow

500 #FFEB3B

50#FFFDE7

100#FFF9C4

200#FFF59D

300#FFF176

400#FFEE58

500#FFEB3B

600#FDD835

700#FBC02D

800#F9A825

900#F57F17

A100#FFFF8D

A200#FFFF00

A400#FFEA00

A700#FFD600

Amber

500 #FFC107

50#FFF8E1

100#FFECB3

200#FFE082

300#FFD54F

400#FFCA28

500#FFC107

600#FFB300

700#FFA000

800#FF8F00

900#FF6F00

A100#FFE57F

A200#FFD740

A400#FFC400

A700#FFAB00

Orange

500 #FF9800

50#FFF3E0

100#FFE0B2

200#FFCC80

300#FFB74D

400#FFA726

500#FF9800

600#FB8C00

700#F57C00

800#EF6C00

900#E65100

A100#FFD180

A200#FFAB40

A400#FF9100

A700#FF6D00

Deep Orange

500 #FF5722

50#FBE9E7

100#FFCCBC

200#FFAB91

300#FF8A65

400#FF7043

500#FF5722

600#F4511E

700#E64A19

800#D84315

900#BF360C

A100#FF9E80

A200#FF6E40

A400#FF3D00

A700#DD2C00

Brown

500 #795548

50#EFEBE9

100#D7CCC8

200#BCAAA4

300#A1887F

400#8D6E63

500#795548

600#6D4C41

700#5D4037

800#4E342E

900#3E2723

Grey

500 #9E9E9E

50#FAFAFA

100#F5F5F5

200#EEEEEE

300#E0E0E0

400#BDBDBD

500#9E9E9E

600#757575

700#616161

800#424242

900#212121

Blue Grey

500 #607D8B

50#ECEFF1

100#CFD8DC

200#B0BEC5

300#90A4AE

400#78909C

500#607D8B

600#546E7A

700#455A64

800#37474F

900#263238

Black#000000

White#FFFFFF

Color system

Choosing a color scheme

You can customize your app’s color scheme to match your brand colors. Alternatively, you can create an entirely new color scheme using the material design color palette.

When creating a color scheme:

Use the color tool to create and apply palettes to your app

Ensure your app’s color usage meets accessibility standards, with sufficient contrast between elements

Expressing Brand in Material

A step-by-step guide to staying on-brand with Material Design

Material Design’s color system

In Material Design, a primary color refers to a color that appears most frequently in your app. A secondary color refers to a color used to accent key parts of your UI.

Using colors from the Material Design palette is optional.

This color scheme has a primary color, lighter and darker versions of that color, and a secondary color.

Beneath the region using the primary color, related information is colored with a lighter version of the primary color. The floating action button uses the secondary color to accent it.

This color scheme contains a primary color, plus darker and lighter versions of that color.

This primary color is applied to the toolbar and status bar, while also being used to accent the floating action button.

Primary color

A primary color is the color displayed most frequently across your app’s screens and components. It can also be used to accent elements, if you don’t have a secondary color.

To create contrast between elements, you can use lighter or darker tones of your primary color. The contrast between lighter and darker tones helps show division between surfaces, such as between the status bar and a toolbar.

This color scheme contains different tones of the primary color, for when lighter or darker contrast is needed.

Secondary Color

A secondary color is used to accent select parts of your UI. It can be complementary or analogous to your primary color, but it should not simply be a light or dark variation of your primary color. It should contrast with elements that surround it and be applied sparingly as an accent.

Secondary colors are best used for:

Buttons, floating action buttons, and button text

Text fields, cursors, and text selection

Progress bars

Selection controls, buttons, and sliders

Links

Headlines

Using a secondary color is optional. It’s not necessary if you use variations of your primary color to accent elements.

A secondary color scheme with varied tones.

Secondary color schemes do not have to be colorful. They only need to contrast with surrounding elements and be used sparingly throughout your UI.

Using color in your app

Large UI areas and elements should be colored with your primary color. A secondary color can be used to accent smaller areas. If you don’t have a secondary color, you can use your primary color instead for these areas.

The floating action button is accented using the secondary color, while the phone icon uses the primary color.

The switch uses the secondary color as an accent. The system bar uses the primary color, while the toolbar uses a darker variation of the primary color.

Use the primary color to accent elements, such as buttons or checkboxes.

Elements that appear rarely, such as alerts, should be distinct from other elements and not use your primary color.

To indicate that buttons and checkboxes are selected, use the primary color as an accent.

Your secondary color can be used on typical elements, such as buttons and links.

Secondary colors, and anything used to accent parts of the UI, should be used sparingly.

Do.

Use a secondary color for certain text, such as linked text.

Don't.

Don’t use bright colors for body text, even if your primary or secondary colors are bright.

See the color tool for more guidance about typographic legibility on color.

Do.

You can use your primary color as a link accent.

Do.

Use your primary or secondary color to emphasize select shorter text, such as headlines.

Do.

Use your secondary color for your primary action button.

Don't.

Don’t use the secondary color for app bars, larger areas of color, or the floating action button if it rests on a background of that color.

Do.

Use a monochromatic color as an accent to let imagery and other colorful elements take prominence.

Do.

You can use your primary color in a different tone on elements that rest on top of a primary color region (like this floating action button).

Do.

Text fields and switches can use your secondary color as an accent.

Do.

Text selection can use your secondary color as an accent.

Alternative secondary colors

If your secondary color is too light or dark to sufficiently contrast with the background color, use a lighter or darker shade of the secondary color instead.

Do.

Use a different tone of your secondary color against backgrounds that are very light or very dark.

Don't.

Don’t use a secondary color on top of a background if there is not enough contrast between the two colors.

Usability

Hierarchy

Hierarchy refers to organizing content according to different levels of importance. Color can convey how important some content is relative to other content.

For example, a brightly colored button on a colorless background makes that button stand out. Alternatively, colorless bars and buttons allow bright content to take prominence in a UI.

Bold color emphasizes the floating action button and the toolbar, giving prominence to message creation and navigation.

Colorless bars and buttons allow colorful content to take precedence, deemphasizing the status and navigation bars.

Meaning

Color may be used to communicate the meaning of various elements on a screen. A weather app may display colors that indicate current weather conditions, and a maps app may use color to show traffic conditions, with roads colored red or green.

A weather app uses color to signal the time of day.

A maps app uses color to signal traffic conditions.

State

Color can provide information about:

An element’s current state, such as if a button is enabled or disabled

An app or element’s change of state

Color that indicates a change of state should be noticeable, as subtle differences in color may be missed. It’s best to indicate a change of state in more than one way, such as displaying an icon or moving the location of an element.

The red color in the label and helper text indicate error messages.

The color changes from blue to red to indicate different states.

Contrast

Your app’s primary and secondary colors should ensure sufficient color contrast between elements so that all users can see and use your app.

To learn more about color, contrast, and accessibility design, read Material Design Accessibility.

In this UI, red signifies the time of day and green represents the foggy weather.

This is how users with deuteranopia (red/green color blindness) might see the UI shown to the left. They might not be able to see differences between text and other elements.

Additional affordances, beyond color, should be used to convey information.

Smaller text may be difficult to read for individuals with cataracts, for whom your UI may appear blurry. Elements without sufficient contrast get lost in the background.

To accommodate all users, offer enlarged text settings.

Text on backgrounds

Text should be legible on the background on which it appears. It is recommended that:

Dark gray text is used on light backgrounds

Light gray text is used on dark backgrounds

If your app has both light and dark themes, the text should be available in a contrasting color against each theme.

Legibility

Text that appears on colored backgrounds should be legible and meet accessibility standards. Both backgrounds and text must use colors and opacities that, when used together, meet these standards. The Web Content Accessibility Guidelines (WCAG 2.0) level AA requires a contrast ratio of 4.5:1 for normal text and a 3:1 ratio for large text.

Do.

Using opacity instead of shifting colors often creates better contrast and relative luminance. For example, gray text (#727272) becomes hard to read if the background color changes to magenta.

Don't.

Shifting to gray instead of decreasing contrast on black often lowers the relative luminance, making typography unreadable.

Dark text on light backgrounds

The level of opacity used for text depends on whether your background is dark or light. For dark text on light backgrounds, apply the following opacity levels:

The most important text has an opacity of 87%.

Secondary text, which is lower in the visual hierarchy, has an opacity of 54%.

Text hints (such as text fields and labels) and disabled text have even lower visual prominence with an opacity of 38%.

Dark text (#000000)

Opacity

Primary text

87%

Secondary text

54%

Disabled text, hint text

38%

Dividers

12%

White text on dark backgrounds

White text appearing on colored backgrounds should do so at an opacity of 100%.

The table relays relative levels of importance for white text on dark backgrounds.

Light text (#FFFFFF)

Opacity

Primary text

100%

Secondary text

70%

Disabled text, hint text

50%

Dividers

12%

Icons and other elements

Elements like icons benefit from having a hex value of black or white at 38% opacity so that they work on backgrounds of any color.

Dark icons (#000000)

Opacity

Active Icon

54%

Inactive Icon

38%

Light icons (#FFFFFF)

Opacity

Active Icon

100%

Inactive Icon

50%

Colored text and backgrounds

Use colored text on colored backgrounds sparingly, limiting usage to important text elements. Colored text should be used to indicate importance and selective emphasis.

See the color tool to determine if certain foreground colors used for typography meet accessibility standards against different background colors.

Do.

Use a contrasting color (such as your secondary color) on important text to contrast it against the background.

Don't.

Avoid placing a large amount of colored text on colored backgrounds.

Do.

You can use a darker or lighter tone of color to create a monochrome, yet accentuated, look.

Don't.

Don’t place low-contrast tones of the same color against each other. It’s difficult to read text with a low contrast against its background.

Themes

Themes let you apply a consistent tone to an app. The theme specifies the darkness of the surfaces, level of shadow, and appropriate opacity of ink elements. To promote greater consistency between apps, light and dark themes are available to choose from.

Download themes

1.23 MB (.ai)

Light theme

1. Status bar

2. App bar

3. Background

4. Cards/Dialogs

Light theme palette

UI application

Dark theme

1. Status bar

2. App bar

3. Background

4. Cards/Dialogs

Dark theme palette

UI application

Icons

Material icons use geometric shapes to visually represent core ideas, capabilities, or topics.

Product icons are the visual expression of a brand’s products, services, and tools.

System icons represent a command, file, device, directory, or common actions.

Sizing

Product icons are 48dp; system icons are 24dp

Icons on light backgrounds

Icon state

Opacity

Active + focused

87%

Active + unfocused

54%

Inactive

38%

Icons on dark backgrounds

Icon state

Opacity

Active + focused

100%

Active + unfocused

70%

Inactive

50%

Product icons

Product icons are the visual expression of a brand’s products, services, and tools. Simple, bold, and friendly, they communicate the core idea and intent of a product. While each product icon is visually distinct, all product icons for a given brand should be unified through concept and execution.

Use these guidelines as a starting point to ensure that your product icon colors and key elements reflect your brand identity.

Material icons

Material icons are easy to use in your web, Android, and iOS projects.

Design approach

Product icon design is inspired by the tactile and physical quality of material. Each icon is cut, folded, and lit as paper would be, but represented by simple graphic elements. The quality of the material is sturdy, with clean folds and crisp edges. The matte-like finish interacts with light through subtle highlights and consistent shadows.

Physical prototype

Lighting study

Material prototype

Color study

Product icon grid

The product icon grid has been developed to facilitate consistency and establish a clear set of rules for the positioning of graphic elements. This standardization results in a flexible but coherent system.

Grid

Keylines

Keyline shapes

Keyline shapes are the foundation of the grid. By using these core shapes as guidelines, you can maintain a consistent visual proportion across related product icons.

Square

Height: 152dp

Width: 152dp

Circle

Diameter: 176dp

Vertical rectangle

Height: 176dp

Width: 128dp

Horizontal rectangle

Height:128dp

Width:176dp

DP unit grid

Android expects product icons to be provided at 48dp, with edges at 1dp. When you create the icon, maintain the 48-unit measure, but scale it to 400% at 192 x 192 dp (the edge becomes 4dp).

Any scaling done to the original will scale the image up or down proportionally. By maintaining the unit ratio, you preserve sharp edges and correct alignment when the scale is reduced.

1:1 Unit grid

4:1 Unit grid

Geometry

Preset standards have been determined for specific keylines: circle, square, rectangle, orthogonals, and diagonals. This small palette of universal and simple elements has been developed to unify product icons and systemize their placement on the grid.

Product icon anatomy

Product icon anatomy describes the graphic elements that make up a product icon. The consistency of these elements across icons for a given brand is critical in maintaining a shared visual language. Familiarity with these elements makes it easier to understand characteristics of each logo and subtle differences between them. It will also help educate your eye to recognize the underlying structure of logo designs.

1. Finish

2. Material background

3. Material foreground

4. Color

5. Shadow

Components

Each component is positioned on top of the previous one, always viewed from straight above.

Construction perspective

An exploded perspective example illustrating the context of each component of the logo construction.

Material background

The back-most material element

Material foreground

A material element raised above, and casting a shadow upon, the material background.

Spot color

Color applied to a small portion of an element.

Flooding

Color applied to an entire element, edge-to-edge.

Tinted edge

The top edge of a material element. A tint is the mixture of a color with white, which lightens the original color.

Shaded edge

The bottom edge of a material element. Shade is the mixture of a color with a darker hue, which darkens the original color.

Contact shadow

A soft shadow around all edges of a raised material element.

Finish

A soft tint above all elements to provide surface lighting, fading from upper-left to lower-right.

Product icon metrics

Lighting

Within the material environment, virtual lights illuminate the scene and allow objects to cast shadows. A top light cast on material elements creates a contact shadow while highlighting the top and bottom edges. An angled light reinforces the sense of surface across the elements.

Top

45º angle

Shadows

For a product icon, the top light from above casts a soft shadow surrounding an element lightly on the top and left. The shadow is slightly heavier below and to the right. This shadow is always contained within the icon’s silhouette.

Drop shadow metrics

Mode: Normal

Opacity: 20%

X Offset: 0dp

Y Offset: 4dp

Blur: 4dp

Color: Refer to Tint, shade and shadow values

Edge tint and shade

The top and bottom edges of material elements provide a sense of depth and surface. Material elements have a standard 1dp thickness. All edge distances are measured from an element's interior edge.

Tint highlights the top edge of all elements. The left, right, and bottom edges do not have a tint applied.

Shade darkens the bottom edge of all elements. The left, right, and top edges do not have a shade applied.

Tinted edge

Height: 1dp

Opacity: 20%

Color: White (#FFFFFF)

Shaded edge

Height: 1dp

Opacity: 20%

Color: Refer to Tint, shade and shadow values

Finish

The finish layer is a result of the virtual 45º light source. It extends from the top-left corner to the exterior edge of the icon’s silhouette. The finish is always contained within these boundaries.

Gradient metrics

Type: Radial

Angle: 45º

Color: White (#FFFFFF)

Midpoint Location: 33%

Slider 1

Opacity: 10%

Location: 0%

Slider 2

Opacity: 0%

Location: 100%

Tint, shade, and shadow values

Each color reacts differently when tints and shades are added. The color of every edge tint, edge shade, and shadow needs to be adjusted for each color that lies behind it. To ensure color harmony, follow the appropriate value for each.

Product icon patterns

Influenced by the behavior of physical material, simple conventions provide a sense of surface and tactility. The interactions of material and color allow for numerous unique compositions.

Color

Color elements are flush to the paper’s surface.

Don’t embellish color elements with any edges or shadows.

Do.

Don't.

Layer

Layered paper elements create depth, having edges and shadows.

Be cautious with the quantity of overlapping surfaces. Having too many complicates the icon and lacks focus.

Do.

Don't.

Elevate

Elevating a key material element atop a simple background silhouette focuses attention to the center.

Don’t crop elevated material elements within another shape.

Do.

Don't.

Score

Scored material elements have the illusion of depth without losing their geometric form. Scores should be centered on symmetrical shapes.

Don’t use multiple scores, or position a score off-center.

Do.

Don't.

Fold

Folded material elements are skewed, having greater dimension. Spot colors should be avoided, so as to avoid altering or misrepresenting key elements.

Do.

Don't.

Overlap

Overlapped material elements create unique silhouettes. All elements, edges, and shadows are confined to the interior of the silhouette.

Don’t exceed more than two overlaps. Having too many complicates the icon and lacks focus.

Do.

Don't.

Accordion

Accordion folded material elements are adjoined by a connecting fold, used to add dimension to a single material element.

Don’t exceed more than two accordion folds. Having too many complicates the icon and lacks focus.

Do.

Don't.

Distort

Product icons should never be distorted or transformed. Elements should remain in their geometric form, and not be skewed, rotated, bowed, warped, or bent.

Don't.

Don't.

Human iconography

The below guidelines and examples illustrate best practices for incorporating human iconography into your UI.

Form

Construction

Keyline alignment

Keyline shapes

Square

Circle

Vertical rectangle

Horizontal rectangle

Paper vs color

Paper

Color

Gestures

Construction

Composition

Human icon rules

Do.

Do use simple shapes for background silhouettes.

Don't.

Don’t use complicated shapes for background silhouettes.

Do.

Do use the correct and consistent human form at all times.

Don't.

Don’t use an incorrect human form or add complex details.

Do.

Do use curved and straight edges for visual balance.

Don't.

Don’t use circular arm terminals nor cropped arms.

System icons

A system icon, or UI icon, symbolizes a command, file, device, or directory. System icons are also used to represent common actions like trash, print, and save.

The design of system icons is simple, modern, friendly, and sometimes quirky. Each icon is reduced to its minimal form, with every idea edited to its essence. The designs ensure readability and clarity even at small sizes.

Material design icons

Download our system icons

Design principles

Shapes are bold and geometric.

Symmetry and consistency of shapes give the icons a unique quality, while keeping them simple and bold.

Simple

Intuitive

Actionable

Consistent

Grid, proportion, and size

DP unit grid

System icons are displayed at 24dp. When creating icons, it’s important to design at 100% scale for pixel-perfect accuracy.

When the mouse and keyboard are the primary input methods, measurements may be condensed to accommodate denser layouts. For dense layouts on desktop, icons can be scaled down to 20dp.

100% scale

Dense grid at 20dp, 100% scale

800% scale

Dense grid at 20dp, 800% scale

Icon grid

The icon grid has been developed to facilitate consistency and establish a clear set of rules for the positioning of graphic elements. This standardization results in a flexible but coherent system.

Grid

Keylines

Content area

The content of an icon should remain inside of the live area. The live area is a safe zone of an image, in which graphics have sufficient display room and are unlikely to be cut off from view (such as when sidebars appear upon scrolling).

The trim area refers to the final size of a graphic file.

Content should only extend into the padding between the live and trim areas if additional visual weight is needed. Do not place any part of the icon outside of the trim area.

Dense layouts

For dense layouts on desktop, icons may be scaled down to 20dp with 2dp of padding surrounding the icon.

Live area

Icon content is limited to the 20dp x 20dp live area, with 4dp of padding around the perimeter.

Dense live area

Icon content is limited to the 16dp x 16dp live area, with 2dp of padding around the perimeter.

Padding

4px of empty space makes up the padding surrounding the 20dp x 20dp live area.

Dense padding area

The 2dp of padding surrounds the live area.

Keyline shapes

Keyline shapes are the foundation of the grid. By using these core shapes as guidelines, you can maintain a consistent visual proportion throughout the system icons.

Square

Height: 18dp

Width: 18dp

Circle

Diameter: 20dp

Vertical rectangle

Height: 20dp

Width: 16dp

Horizontal rectangle

Height: 16dp

Width: 20dp

Geometry

Preset standards have been determined for specific keylines: circle, square, rectangle, orthogonals, and diagonals. This small palette of universal and simple elements has been developed to unify Google system icons and systemize their placement on the icon grid.

Construction

Composition

System icon anatomy

1. Stroke terminal

2. Corner

3. Counter area

4. Stroke

5. Counter stroke

6. Bounding area

Corners

Consistent corner radiuses are key to unifying the overall system icon family. A 2dp corner radius is used on the silhouette form of the icon. Do not round the corners of strokes (shapes 2dp wide or less).

Interior corners should be square. Do not round the corners of interior shapes.

Exterior corners with 2dp corner radius

Interior corners

Strokes

Consistent stroke weights are key to unifying the overall system icon family. Maintain a 2dp width for all stroke instances, including curves, angles, and both interior and exterior strokes.

Consistency

Curves and angles

Stroke terminalangle

Counter stroke

Optical corrections

Extreme scenarios that call for subtle tweaks add to the legibility of an icon. Instances where complex details are unavoidable require adjusting metrics.

If optical corrections are necessary, only use the consistent geometric forms on which all other icons are based. Don’t skew or distort the forms.

Complex

The paperclip icon in this example is only using 1.5dp of the possible 2dp stroke area to fit multiple curves within the 24 x 24dp icon space.

Small scale

The microphone icon in this example is using a 1.5dp stroke to indicate microphone sound waves within the 24 x 24dp icon space.

Clearance

Adequate space around the icon is needed to allow for legibility and touch.

When the mouse and keyboard are the primary input methods, measurements may be condensed to accommodate denser layouts.

Clearance area

Icon: 24dp

Touch target: 48dp

Clearance area for dense icons

Icon: 20dp

Touch target: 40dp

Placement

Best practices

Consistency aids user comprehension of icons. Use the existing system icons whenever possible and across different applications.

Do.

Do use consistent stroke weights and squared stroke terminals.

Don't.

Don’t use inconsistent stroke weights or rounded stroke terminals.

Do.

Do make icons appear front-facing and sturdy.

Don't.

Don’t tilt, rotate, or make icons appear dimensional.

Do.

Do simplify icons for greater clarity and legibility.

Don't.

Don’t be overly literal. Avoid complex icons.

Do.

Do make icons graphic and bold.

Don't.

Don’t use delicate, thin stroke weights.

Do.

Do use geometric, consistent shapes.

Don't.

Don’t use gestural or loose organic shapes.

Do.

Do position icons “on pixel” – meaning the X and Y coordinates are integers and do not contain decimals.

Icons should have equal width and height (e.g. 24x24) to avoid distorting the icon.

Don't.

Don’t place the icon on a coordinate that is not “on pixel”.

Don’t distort the icon by having unequal width and height values.

Human iconography

Human icon anatomy

1. Head

2. Neck

3. Upper body

4. Arm

5. Leg

Full body

Upper body

Form

Full body example

Upper body example

Cropped example

Detailed parts example

Full body

Full body examples

Visual Alignment

Upper body

Upper body examples

Visual Alignment

Contained

Cropped examples

Detailed parts

Detailed part examples

Human icon rules

Do.

Do use consistent stroke weights and squared arm/leg terminals.

Don't.

Don’t use inconsistent stroke weights nor rounded arms/legs.

Do.

Do align elements to simplify the silhouette for clarity.

Don't.

Don’t crop portions of arms/legs.

Do.

Do fully embed elements within a shape when contained.

Don't.

Don’t break the container’s boundary with elements.

Do.

Do add human elements when they help amplify the meaning of an icon.

Don't.

Don’t add human elements when they increase the complexity of an icon.

Do.

Do use the most simple shapes to represent human characteristics.

Don't.

Don’t apply human characteristics to inanimate objects.

Color

The standard opacity for an active icon on a light background is 54% (#000000). An inactive icon, which is lower in the visual hierarchy, should have an opacity of 38% (#000000).

The standard opacity for an active icon on a dark background is 100% (#FFFFFF). An inactive icon, which is lower in the visual hierarchy, should have an opacity of 50% (#FFFFFF).

App shortcut icons

App shortcuts give users quick, easy access to up to four of your app’s actions. Each action can also be added to the home screen.

Shortcut actions are displayed depending upon the location of your app’s icon. For example, app icons near the left edge of the screen will display actions wherever there is available space.

These guidelines will help you create app shortcuts using standard icons, single avatars, and group avatars.

Four shortcuts maximum may be shown at once. You may determine the order of the shortcuts.

Shortcut icon

App shortcut icons are circular. They contain a standard system icon, or at least one avatar. Icon content must remain inside of the trim area (the total area of the graphic).

Sizing

Live area: 44dp circle

Total area: 48dp circle

2dp of padding must surround the 44x44dp live area

Color

Live area: Solid fill of Material Grey 100 (#F5F5F5)

Shadow

Shortcut icons do not include shadows

Live area

All icon content should remain in the 44dp live area, with a solid background color fill of Material Grey 100 (or #F5F5F5).

Total area

2dp of padding must surround the live area, making the total icon size 48dp.

Standard shortcut icons

Standard shortcut icons have a Material system icon centered within the live area.

Sizing

Live area: 44dp circle

Total area: 48dp circle

System icon: 24dp

Color

The system icon should have the same color as the app’s primary color or app icon (with enough contrast against the circular background).

File format

Standard icons should be provided in SVG, which allows icons to be scaled automatically. You may also use vector drawables, tinted bitmaps, or layer lists.

Do not use non-vector drawables, as they have specific dimensions that do not scale in this context.

System icon size

The system icon should have equal width and height: 24x24dp.

Position in live area circle

The 24dp icon should be centered vertically and horizontally within the live area circle.

Do.

The color of the icon should have enough contrast against the Material Grey 100 background.

Don't.

Don’t use colors with insufficient contrast against the grey background.

Single avatars

Single avatars contain one image. They must be created at XXXHPDI.

Sizing

Live area: 44dp circle

Avatar area: 44dp circle

Total area: 48dp circle

File format

Provide PNG files at all densities.

Live area circle & avatar area

The avatar should fill the entire live area circle. These assets must be created in XXXHPDI.

Total area

2dp of padding must surround the 44dp live area circle for a total area of 48dp.

Group avatars

Group avatars contain 2-4 images on a circular background. They must be created at XXXHPDI.

Sizing

Live area: 44dp circle

Avatar area: 30dp

Total area: 48dp

Color

Live area: Solid fill Material Grey 100 (#F5F5F5)

File format

Provide PNG files at all densities.

Live area circle

The live area circle should have a color fill of Material Grey 100 (#F5F5F5).

Avatar area

Avatars must fit within the 30x30 dp space and be centered, vertically and horizontally, within the live area.

Total area

2dp of padding must surround the 44dp live area for a total area of 48dp.

Imagery

Imagery is more than decoration. It’s a powerful tool to help you communicate and differentiate your product.

Bold, graphic, and intentional imagery helps to engage the user.

Whether the mood is subdued and muted or bright and colorful, the following principles and best practices can help you successfully incorporate imagery and bring your apps to life, no matter what the visual brand.

Principles

Relevant

Informative

Delightful

Integrating with UI

Use appropriate resolution

Use scrims to protect text

Use cases

Avatars and thumbnails

Hero images

Galleries

Principles

When using illustration and photography to enhance the user experience, choose images that express personal relevance, information, and delight.

Personal relevance

Imagery can reflect the context and the world the user inhabits.

Information

Images can convey specific information that makes comprehension easy and immediate.

Delight

Portraying context with aesthetic beauty will make your product unique and add to user delight.

Appreciate context

Ensure images are dynamic and context-relevant. Use predictive visuals to substantially improve the user experience.

Use imagery that is context-relevant.

Be immersive

When necessary, it’s acceptable to obscure or allude to integrated heros and thumbnails through color and content overlays.

Background imagery can be obscured when engaging with foreground content.

Best practices

Use multiple mediums

Both illustration and photography can live within the same product. Photography automatically implies a degree of specificity and should be used to showcase specific entities and stories. Illustration is effective for representing concepts and metaphors where specific photography might be alienating.

Do.

For specific entities, look first to photographic representation.

Do.

When total specificity is not available or applicable, illustration conveys approximation of content to aid comprehension.

Stay away from stock

Use imagery to express a distinctive voice and exemplify creative excellence.

For specific entities or branded content, use specific imagery. For more abstract content, be interpretive. Photographic stock and clipart is neither specific nor interpretive.

Do.

Strive for images that represent genuine stories.

Don't.

Stock images can feel inauthentic.

Do.

For specific representations, use specific imagery.

Don't.

Don't default to literal stock photography.

Have a point of focus

Have an iconic point of focus in your imagery. Focus ranges from a single entity to an overarching composition. Ensure that a clear concept is conveyed to the user in a memorable way.

Do.

Use color and composition to give images a clear focus.

Don't.

Avoid making the user hunt for the meaning in the image.

Do.

The most powerful iconic images consist of a few meaningful elements, with minimal distractions.

Don't.

When the point of focus is obscured, the iconic quality of the image is lost.

Do.

A clear focus communicates the concept at a glance.

Don't.

A lack of focus makes the image meaningless.

Build narratives

Create an immersive story and a sense of context to humanize your app. Define the mood through visual storytelling. Is the mood aspirational? Somber? Celebratory?

Do.

An image that tells a story is infinitely more interesting and informative.

Don't.

An image that lacks a story loses an opportunity to convey mood, brand, and context.

Do.

Properly contextualized concepts convey your message and brand more effectively.

Don't.

A representation of an entity that is dissociated from context is uninteresting.

UI integration

Resolution

Make sure your images are appropriately sized for displays and across platforms. Material design emphasizes large images. Ideally, the assets should not appear pixelated. Test appropriate resolution sizes for specific ratios and devices.

Do.

Appropriately sized imagery

Don't.

Degraded imagery

Introduce scale

Introduce alternative scales to create levels of visual importance.

Within a gallery context, introduce thumbnails of various sizes to convey content hierarchy.

Encourage multiple containers living in the same ecosystem.

Text protection

To make typography legible on top of imagery, apply text protection in the form of scrims. Scrims are lightweight, translucent material layers.

1. Dark end of gradient: Opacity values depend on context

2. Center point of gradient: 3/10ths closer the darker end

3. End point of gradient: Opacity of 0

To avoid banding (the formation of distinct stripe shapes), the gradient should be long, 3x standard app bar height, with the centerpoint about 3/10 towards the darker side of the gradient. This gives the gradient a natural falloff and avoids a sharp edge.

The gradient’s opacity should be chosen based on its environment. Some imagery can use darker gradients, such as the image displayed here with a 60% gradient.

Gradients could be as low as 10-20% in some contexts.

In this image, the gradient is 60% black.

This is the gradient which has been applied to the image at the left.

Do.

Dark scrims should ideally be between 20%-40% opacity depending on the content. Light scrims should ideally range from 40%-60% opacity depending on the content.

Don't.

Don't obscure imagery when you use a scrim.

Do.

For larger real estate, target specific areas for text protection instead of blanketing the entire image.

Don't.

Don’t ‘over-scrim’ large images.

Do.

Color overlays are different than text-protection scrims and can be used as a design element. When creating complementary color overlays, use the Palette API to define your color scheme based on content.

Do.

You can use your brand’s color palette decoratively to enhance imagery.

Avatars and thumbnails

Avatars and thumbnails represent entities or content, either literally through photography or conceptually through illustration. Generally, they are tap targets that lead to a primary view of the entity or content.

Avatars can be used to represent people. For personal avatars, offer personalization options. As users may choose not to personalize an avatar, provide delightful defaults. When used with a specific logo, avatars can also be used to represent brand.

Thumbnails allude to more information – letting the user peek into content – and assist navigation. Thumbnails let you include imagery in tight spaces.

Avatars make an app feel more personal – and occupy minimal space.

A brand avatar communicates information at a glance.

Hero images

Hero images are images that are usually anchored in a prominent position, above the fold, such as a banner at the top of the screen. They serve to draw in a user, provide context about the content, or reinforce the brand.

Related

Extract prominent colors from an image

Feature

A feature image is a bold, primary point of focus within a heterogeneous layout.

Integrated hero image

Integrated hero images create a setting for heterogeneous pieces of content within a layout. They are not the primary points of focus.

Gallery

Gallery images are bold visual hero images within homogenous layouts in either a grid or as a single image.

Photo Grid

Gallery image

Typography

Roboto and Noto are the standard typefaces on Android and Chrome.

Roboto is the standard typeface on Android.

Noto is the standard typeface for all languages on Chrome and Android for all languages not covered by Roboto.

Script types

English and English-like (Latin, Greek, and Cyrillic)

Dense (Chinese, Japanese, and Korean)

Tall (South and Southeast Asian and Middle Eastern languages)

App bar

Title style, Medium 20sp

Buttons

English: Medium 14sp, all caps

Dense: Medium 15sp, all caps

Tall: Bold 15sp

Subheading

English: Regular 16sp (device), 15sp (desktop)

Dense: Regular 17sp (device), 16sp (desktop)

Tall: Regular 17sp (device), 16sp (desktop)

Body 1

English: Regular 14sp (device), 13sp (desktop)

Dense: Regular 15sp (dense), 14sp (desktop)

Tall: Regular 15sp (device), 14sp (desktop)

Text contrast ratios

Minimum: 4.5:1

Preferred: 7:1

Language categorization

Language scripts can be divided into three categories:

English and English-like: Latin (except Vietnamese), Greek, and Cyrillic scripts, supported by both Roboto and Noto. Roboto has been extended to completely cover all Latin, Greek, and Cyrillic characters as defined in Unicode 7.0. The number of supported characters has doubled from previous releases, from about 2000 to about 4000 characters.

Tall: Language scripts that require extra line height to accommodate larger glyphs, including South and Southeast Asian and Middle-Eastern languages, like Arabic, Hindi, Telugu, Thai, Vietnamese. Noto supports these languages with two weights.

Dense: Language scripts that require extra line height to accommodate larger glyphs, including Chinese, Japanese, and Korean. Noto supports these languages with seven weights.

See the full list in the language categories table.

Typeface

Roboto

Roboto has been refined extensively to work across the wider set of supported platforms. It is slightly wider and rounder, giving it greater clarity and making it more optimistic.

Examples of Roboto

Roboto A-Z and numerals

Roboto font weights

Roboto has six weights: Thin, Light, Regular, Medium, Bold, and Black.

Roboto font weights

Noto

Noto’s vertical metrics are compatible with Roboto.

Noto Sans examples for English and CJK (Chinese, Japanese, and Korean)

Noto Sans examples for Thai and Hindi (Devanagari)

Noto font weights

Noto Sans CJK (Chinese, Japanese, and Korean) has seven weights: Thin, Light, DemiLight, Regular, Medium, Bold, and Black. The weight of Noto Sans CJK Regular is the same as Roboto Regular.

Noto Sans CJK font weights

Noto fonts for Thai, Devanagari, and all other major living languages have Regular and Bold weights.

Noto Sans Thai and Hindi (Devanagari) font weights

Google font directory

A directory of open-source web fonts

Hinted fonts

Hints are the instructions embedded in a font on how to modify (distort) a glyph to look better on low-resolution displays. As a tradeoff, a hinted font consumes more space than the unhinted version.

Both Roboto and Noto have hinted and unhinted versions. Google recommends:

Use the unhinted versions on Android and on Mac OS X, which doesn’t implement hints.

Use hinted fonts on Chrome OS, Windows, and Linux.

Font stack

For both Android and web properties, the font stack should specify Roboto, Noto, and then sans-serif.

Styles

Too many type sizes and styles at once can wreck any layout. A typographic scale has a limited set of type sizes that work well together along with the layout grid.

These sizes and styles were developed to balance content density and reading comfort under typical usage conditions. Type sizes are specified with sp (scaleable pixels) to enable large type modes for accessibility.

English and English-like scripts

Latin, Greek, and Cyrillic.

The basic set of styles are based on a typographic scale of 12, 14, 16, 20, and 34.

Examples using Display style, English

Examples using Headline style, English

Across form factors, text that appears in the app bar should use the Title style, Medium 20sp.

Examples using Title style, English

In certain scenarios, use the larger subheading style instead of the smaller Body style. Some of those scenarios include when information is presented as small snippets or when titles are paired with lines of Body-styled text.

Examples using subheading style, English

Examples using Body style, English

Examples using Body style, English

Button style (Medium 14sp, all caps) is used for all buttons. Button text should be all caps in languages that have capitalization. For languages that don’t have capitals, consider using color text for flat buttons to make them stand out from normal text.

Examples using Button style, English

Dense scripts

Chinese, Japanese, and Korean.

Weight: Since Noto CJK has seven weights that match Roboto, use the same weight settings as English.

Font size: For Title through Caption styles, the font size is 1px larger than that specified for English. For styles larger than Title, the English type size is suitable.

Example, dense language script

Example of a subheading, Japanese

Example of Body, Japanese

Example of Body 1, Japanese

Example of a subheading, Traditional Chinese

Example of Body, Traditional Chinese

Example of Body 1, Traditional Chinese

Tall scripts

South and Southeast Asian and Middle Eastern languages, including Arabic, Hindi, and Thai.

Weight: Use Regular weight, as Medium weight is unavailable in Noto. Google recommends avoiding Bold weight, based on feedback from native speakers that Bold is too heavy.

Font size: For Title through Caption styles, font size is 1px larger than that specified for English. For styles larger than Title, the English type size is suitable.

Example, tall language scripts

Example of a subheading, Hindi

Example of Body, Hindi

Example of Body1, Hindi

Example of a subheading, Thai

Example of Body, Thai

Example of Body1, Thai

Line height

To achieve proper readability and appropriate pacing, line heights have been determined based on each style’s individual size and weight. Line wrapping only applies to Body, Subhead, Headline, and the smaller Display styles. All other styles should exist as single lines.

English and English-like scripts

English and English-like type and leading

Pairing examples

Isolated examples – Increased line height

Dense and tall scripts

For all styles, line height is 0.1em larger than the English-like languages. English and English-like languages mostly use a portion of the em box, often the lower portion below the x-height. Chinese, Japanese, and Korean (CJK) ideographic characters use the entire em box. Characters in tall languages often have long descenders and/or ascenders. To achieve the same design intention as English for CJK and to avoid potential text clipping between two lines next to each other for tall languages, the line height needs to be larger than in English for tall and dense languages.

Line height for tall and dense languages

Line height pairing example, Thai and Hindi

Line height pairing example, Chinese and Japanese

Other typographic guidelines

Colors & contrast

A text color that is too similar to the background color is hard to read. Text with too much contrast can also be hard to read. This is especially true of light-colored text against dark backgrounds.

Text should maintain a minimum contrast ratio of at least 4.5:1 (calculated based on luminance values) for legibility. A ratio of 7:1 is preferred.

These color combinations also consider contrast ratios for users who perceive color differently.

Contrast over light background

Contrast over dark background

Contrast over image

Contrast over illustration

Large and dynamic type

For the best user experience, use dynamic type instead of relying only on smaller type sizes or allowing truncation of larger-size text.

Large type applied correctly can make apps more interesting, differentiate layouts, and help users to decode content quickly.

Dynamic type enables large type when the length of the text in a layout is unknown. Dynamic sizes are selected from a typographic scale based on available space and letter size estimates.

Examples of dynamic type

UI application of dynamic type

Line breaking

The images show best practices for line breaks.

Do.

Don't.

Avoid leaving large gaps and orphans on a line. Try not to leave very short words such as prepositions at the end of a line. Well-considered line breaks can avoid hyphenation of words altogether.

Tracking and kerning

Line length

Consider this advice on readability and line length from the Baymard Institute:

“You should have around 60 characters per line if you want a good reading experience. Having the right amount of characters on each line is key to the readability of your text.”

"Too wide – if a line of text is too long, the user’s eye will have a hard time focusing on the text. This is because the length makes it difficult to get an idea of where the line starts and ends. Furthermore it can be difficult to continue from the correct line in large blocks of text.”

"Too narrow – if a line is too short, the eye will have to travel back too often, breaking the reader’s rhythm. Too short lines also tend to stress people, making them begin on the next line before finishing the current one (hence skipping potentially important words).”

Source: “Readability: the Optimal Line Length,”

http://baymard.com/blog/line-length-readability

Ideal line lengths for English body text. The numbers represent the number of characters per line.

Ideal line lengths for short lines of text in English. The numbers represent the number of characters per line.

Language categories reference

For ease of internationalization, Google has categorized languages into three categories: English and English-like; tall; and dense.

English and English-like: Latin (except Vietnamese), Greek, Cyrillic, Hebrew, Armenian and Georgian.

Tall: Language scripts that require extra line height to accommodate larger glyphs, including South and Southeast Asian and Middle Eastern languages, like Arabic, Hindi, Telugu, Thai, Vietnamese.

Dense: Language scripts that require extra line height to accommodate larger glyphs but have different metrics from tall scripts. Includes Chinese, Japanese, and Korean.

Code

Description

Category

af

Afrikaans

English-like

am

Amharic

English-like

ar

Arabic (Modern Standard)

Tall

az

Azerbaijani

English-like

bg

Bulgarian

English-like

bn

Bengali

Tall

ca

Catalan

English-like

cs

Czech

English-like

cy

Welsh

English-like

da

Danish

English-like

de

German

English-like

el

Greek

English-like

en

English (US)

English-like

en-GB

English (UK)

English-like

es

Spanish (European)

English-like

es-419

Spanish (Latin American)

English-like

et

Estonian

English-like

eu

Basque

English-like

fa

Persian

Tall

fi

Finnish

English-like

fil

Filipino

English-like

fr

French (European)

English-like

fr-CA

French (Canadian)

English-like

gl

Galician

English-like

gu

Gujarati

Tall

hi

Hindi

Tall

hr

Croatian

English-like

hu

Hungarian

English-like

hy

Armenian

English-like

id

Indonesian

English-like

is

Icelandic

English-like

it

Italian

English-like

iw

Hebrew

English-like

ja

Japanese

Dense

ka

Georgian

English-like

kk

Kazakh

English-like

km

Khmer

Tall

kn

Kannada

Tall

ko

Korean

Dense

ky

Kirghiz

English-like

lo

Lao

Tall

lt

Lithuanian

English-like

lv

Latvian

English-like

mk

Macedonian

English-like

ml

Malayalam

Tall

mn

Mongolian

English-like

mr

Marathi

Tall

ms

Malay

English-like

my

Burmese (Myanmar)

Tall

ne

Nepali

Tall

nl

Dutch

English-like

no

Norwegian (Bokmål)

English-like

pa

Punjabi

Tall

pl

Polish

English-like

pt

Portuguese (Brazilian)

English-like

pt-PT

Portuguese (European)

English-like

ro

Romanian

English-like

ru

Russian

English-like

si

Sinhala

Tall

sk

Slovak

English-like

sl

Slovenian

English-like

sq

Albanian

English-like

sr

Serbian (Cyrillic)

English-like

sr-Latn

Serbian (Latin)

English-like

sv

Swedish

English-like

sw

Swahili

English-like

ta

Tamil

Tall

te

Telugu

Tall

th

Thai

Tall

tr

Turkish

English-like

uk

Ukrainian

English-like

ur

Urdu

Tall

uz

Uzbek

English-like

vi

Vietnamese

Tall

zh-CN

Chinese (Simplified, Mandarin)

Dense

zh-HK

Chinese (Mandarin, Hong Kong)

Dense

zh-TW

Chinese (Traditional, Mandarin)

Dense

zu

Zulu

English-like

Writing

Text should be understandable by anyone, anywhere, regardless of their culture or language.

Clear, accurate, and concise text makes interfaces more usable and builds trust.

In addition to these guidelines, be sure to consult guidelines for specific UI elements, such as Errors, Dialogs, Settings, and Data formats.

Guidelines

Tone

Capitalization

Punctuation

UI button text

Style

Google guidelines generally follow the Associated Press (AP) style guidelines.

Text should be simple, concise, and direct

Do.

Language

Addressing users

Your UI may address the user using either:

Second person, “you” or “your”: Use this conversational style for most situations, as though the app is speaking directly to the user.

First person, “I” or “my”: In some cases, you may need to use this form of address to emphasize the user's ownership of content or actions.

Quickly open the camera without unlocking your screen

Your places

Addressing the user with the second person using “you” or “your”

I agree to follow the Google Terms of Service

My Account

Addressing the user with the first person using “I” or “my”

Avoid mixing "me"/"my" with "you"/"your.” It can cause confusion to see both forms of addressing the user in the same context.

Change your preferences in My Account

Don't.

Avoid the pronoun “we”

Focus on the user and what they can do with your app, rather than what you or your app is doing for the user.

Get started with these popular posts on Google+

Do.

To get you started, we’re showing you popular posts on Google+

Don't.

One exception is when a person takes an action for a user, such as reviewing an appeal or responding to a suggestion. Here, the use of “we” is appropriate.

We’ll review your appeal and respond within a few days

Do.

Your appeal will be reviewed, and you will receive a response within a few days

Don't.

Be concise

Write in small, scannable segments to facilitate navigation and discovery.

Send money to anyone in the US who has an email address. It’s fast, easy, and free.

Do.

Send (and receive) money with friends and family in the US with an email address. It’s a two-step process with little-to-no latency and there aren’t any charges for the recipients of the money.

Don't.

Keep your sentences and phrases short, with as few concepts as possible.

Read the instructions that came with your phone

Do.

Consult the documentation that came with your phone for further instructions

Don't.

Write in the present

Use the present tense to describe product behavior. Avoid using the future tense to describe the way a product always acts.

When you need to write in the past or future, use simple verb forms.

Write simply and directly

Use simple, direct language that is easy for users to understand.

Common introductory phrases may be omitted.

Save changes?

Do.

Would you like to save your changes?

Don't.

Message sent

Do.

Message has been sent

Don't.

Register to vote

Do.

You must register before you can vote

Don't.

Write for all levels of readers

Pick common words that are clearly and easily understandable to both beginning and advanced English readers.

Turn on Location History

Do.

Enable Location History

Don't.

Avoid industry-specific terminology or names invented for UI features.

Preparing video…

Do.

Buffering…

Don't.

“Ok Google” isn’t supported on your phone

Do.

“Ok Google” is only supported on dual-core devices

Don't.

Refer users to the labels on UI elements, not the type of element (such as menu or button).

Click Continue

Do.

Click the Continue button

Don't.

Use consistent words in all parts of a feature

Use verbs in a consistent manner across the description of an action.

Remove photo

Do.

Delete photo

Don't.

Remove photo?

Do.

The dialog title is short.

Remove photo from page?

Don't.

The dialog title is longer than necessary.

Begin with the objective

If a sentence describes both an objective and the action needed to achieve that objective, start the sentence with the objective.

To remove a photo from this album, drag it to the trash

Do.

Drag a photo to the trash to remove it from this album

Don't.

Reveal detail as needed

It's not necessary to describe every detail in the first interaction. Reveal increasing detail about features as the user explores them and needs the information.

Remove downloaded book?

Do.

Are you sure you want to remove this downloaded book? You won’t be able to access it unless you’re online.

Don't.

Never say “never”

Avoid “never” and other absolutes.

Your circle names aren’t shared

Do.

We’ll never share your circle names

Don't.

Text for buttons and related elements

Buttons appear in dialogs and as blue or underlined text.

Button

Usage

[ Action ]

See above guideline about using consistent verbs.

Back

Allows multi-step processes

Cancel

Cancels an action

Dismiss

Causes a message or dialog to disappear without any consequences

Done

Confirms the completion of a multi-step process

Got it

Causes a message or dialog to disappear without any consequences (similar to OK)

Learn more

Takes the user to additional content

Next

Takes the user to the next step of a multi-step process

No thanks

Allows a user to decline

Not now

Let’s a user postpone an action or decision. Use only when the call to action in the dialog is essential to the functionality of the product, for legal reasons, or for another urgent reason.

Do not use “Not now” as a mechanism to avoid providing a “No thanks” option.

OK

Allows the user to confirm an action that’s relevant to the task at hand

Skip

Gives the user a way to avoid an interruption and proceed with a task

Tone

Be friendly, respectful, and focus on the user

Your app’s text should complement its design: intuitive, efficient, casual, and trustworthy.

MyApp isn’t responding

Do you want to close it?

Do.

Sorry!

Activity in MyAppActivity (in the MyApp app) is not responding

Don't.

Be humble

Reveal what a feature does, without bragging or over-promising.

All your savings in one place

Do.

Great deals at places you’ll love

Don't.

More restaurant reviews

Do.

All restaurant reviews

Don't.

Be inviting

Focus on the benefits of each feature. Omit implementation details, caveats, and restrictions when features are introduced.

To save power, switch Location mode to Battery saving mode

Do.

Manually control GPS to prevent other apps from using it

Don't.

Be positive

Present information in a positive light: it’s reassuring.

Use 24 characters or fewer for file names

Do.

Your file name must be less than 25 characters

Don't.

Try again

Do.

The action failed

Don't.

Be essential

Communicate essential details, so that users can focus on their own tasks. Sometimes the most effective UI contains no text at all.

Signing in...

Your phone is contacting Google. This can take up to five minutes.

Do.

Signing in...

Your phone needs to communicate with Google servers to sign in to your account. This may take up to five minutes.

Don't.

Capitalization & punctuation

Use sentence-style caps

Titles, headings, labels, and menu items should use sentence-style capitalization (capitalizing only the first word of a sentence).

Avoid capitalizing all letters except where the material spec requires them, such as the Button style.

Search settings

Do.

Search Settings

Don't.

SEARCH SETTINGS

Don't.

Skip periods and unnecessary punctuation

To help readers scan text at a glance, avoid using periods and other unnecessary punctuation.

Periods

Avoid using periods in solitary sentences in these elements:

Labels

Hover text

Bulleted lists

Dialog body text

Periods should be used for:

Text with multiple sentences (including lists)

Any sentence followed by a link

Share your photos with friends. Learn more

Do.

Place periods after sentences followed by a link.

Share your photos with friends. Learn more.

Don't.

Sentences followed by a link should place the period before the link, not after.

To narrow your targeting from all countries and territories, add a location

Do.

Undo bulk edit?

If you undo this bulk edit, everything you changed will go back to its previous state

Do.

Skip periods after solo sentences of body text.

Undo bulk edit?

If you undo this bulk edit, everything you changed will go back to its previous state.

Don't.

Don’t place periods after body text if there is only a single sentence present.

Colons

Skip colons after labels.

Share with

Do.

Share with:

Don't.

Use contractions

Don't make a sentence harder to understand just to follow this rule. For example, "do not" can give more emphasis than "don't" when needed.

it’s, can’t, wouldn’t, you’re, you’ve, haven’t, don’t

Do.

it is, cannot, would not, it’ll, should’ve

Don't.

Avoid exclamation points

Avoid exclamation points: they tend to come across as shouting. Some exceptions include greetings or congratulatory messages.

Welcome!

Do.

Learn about the new features of Calendar!

Don't.

Good job!

Do.

“1, 2, 3” not “one, two, three”

Use numerals in place of words for numbers.

One exception is when mixing uses of numbers, such as "Enter two 3s."

You have 3 messages

Do.

You have three messages

Don't.

Punctuation

Omit punctuation after phrases and labels to create a cleaner and more readable interface.

Use punctuation to add clarity or be grammatically correct.

Glyph/character

HTML entity

Unicode

Description

Periods

.

Omit for fragments and single sentences in affordances like toasts, snackbars, and labels. Do punctuate full sentences in body text, for example, in the body of dialogs.

Place inside quotation marks (unless you’re telling the reader what to enter and it’s ambiguous whether to include the period).

Commas

,

Place inside of quotation marks.

Use the serial comma, except when using an ampersand (&) in a list of three or more items, such as “you, me & my dog.”

Exclamation points

!

Avoid exclamation points as they may come across as shouting in most cases.

Colons

:

&#58;

\u003A

Omit from labels, for example, in the labels for fields in a form.

Quotation marks

“

”

‘

’

&ldquo;

&rdquo;

&lsquo;

&rsquo;

\u201C

\u201D

\u2018

\u2019

Use real quotation marks, not the inch and foot symbols.

The right single quotation mark symbol is also used for apostrophes.

Never use the generic quotes ", ' or free-standing accents `, ´ (\u0022, \u0027, \u0060, \u00B4). These are never right for quotation marks, apostrophes, or primes.

Primes

′

&prime;

&Prime;

\u2032

\u2033

Use prime (′) only in abbreviations for feet, arcminutes, and minutes. For example: 3° 15′

Use double-prime (″) only in abbreviations for inches, arcminutes, and minutes. For example: 3° 15′ 35″

Don’t use generic quotes ", ' or free-standing accents `, ´ for primes.

Ellipses

…

&hellip;

Use to indicate an action in progress ("Downloading…") or incomplete or truncated text. No space before the ellipses.

Omit from menu items or buttons that open a dialog or start some other process.

Midline ellipses (three-bullet glyphs) are also used to represent numeric truncation and the redaction of sensitive data, such as credit cards.

OS keystroke: Option-;

Double angle brackets

>>

<<

&laquo;

&raquo;

\u00AB

\u00BB

Omit from links or buttons that open another page or move to the next or previous step in a process.

Em dash

—

&mdash;

\u2014

Avoid using em dashes. Use en dashes instead.

OS keystroke: Shift+Option -

En dash

–

&ndash;

\u2013

Use an en dash instead of a hyphen to indicate a range, without spaces (in English).

Avoid using dashes to separate text. If you must use dashes for this purpose – like this – use an en dash surrounded by spaces.

Example of use with a range:

8:00 AM–12:30 PM

3–5 kg

OS keystroke: Option+ -

Hyphen

-

Use hyphens to represent negative numbers.

For example:

-5

-$100

Use hyphens to avoid ambiguity in adjective-noun or noun-participle pairs.

For example:

anti-inflammatory

5-mile walk

Parentheses

( )

Use parentheses only to define acronyms or jargon.

For example:

“Secure web connections are based on a technology called SSL (the secure sockets layer).”

Global writing

People of all ages, cultures, and education levels rely on the English versions of products. Simple, clear English makes it easy for anyone to understand your product.

Example of Japanese writing

Example of Thai writing

Be aware of using terms like “left” and “right” in RTL languages

Interfaces for languages with right-to-left scripts may be mirrored when a product is localized.

In this example of a right-to-left (RTL) interface in Arabic, the interface is mirrored relative to English.

Example of a left-to-right (LTR) interface in English

Specifying gender

Sometimes the gender of the user isn’t known. In English, the gender-neutral word "they" may be used in this case. But many languages refer to gender in different ways.

When referring to gender in your app:

Don't combine gender options, such as "his/her" or "she/he.”

If a specific gender is required, such as text referring to a person’s name, provide clear translation instructions and use the International Components for Unicode (ICU) APIs or equivalent libraries.

Principles

Material design is guided by print-based design elements – such as typography, grids, space, scale, color, and imagery – to create hierarchy, meaning, and focus that immerse the user in the experience. Material design adopts tools from the field of print design, like baseline grids and structural templates, encouraging consistency across environments by repeating visual elements, structural grids, and spacing across platforms and screen sizes. These layouts scale to fit any screen size, which simplifies the process of creating scalable apps.

How paper works

In material design, the physical properties of paper are translated to the screen. The background of an application resembles the flat, opaque texture of a sheet of paper, and an application’s behavior mimics paper’s ability to be re-sized, shuffled, and bound together in multiple sheets.

The surfaces that comprise applications are referred to in this spec as material, or sheets of material.

Elements outside of applications, such as system bars, are separate from the app content beneath them, and are not treated as material.

For detailed information about material, see Material Properties.

Seams

Two sheets of material that share an entire common edge are called seams. While joined by a seam, they move together.

Seams in two sheets of material

Steps

Two overlapping sheets of material, with different z-positions (or depths), form a step. They move independently of each other.

Steps in two sheets of overlapping material

Floating action button

Floating action button

A floating action button is a circular sheet separate from a toolbar. It represents a single promoted action.

It can straddle a step if it relates to the content creating that step.

Floating action buttons straddling steps

A floating action button can straddle a seam if it relates to the content of both sheets.

Do not introduce a decorative seam to provide an anchoring point for an action.

Floating action buttons straddling seams

For detailed information about floating action buttons, see Floating Action Buttons.

Units & measurements

Some units have different meanings in different contexts. This chapter discusses the usage of device-independent pixels, scaleable pixels, as well as concepts like pixel density.

Pixel density

The number of pixels that fit into an inch is referred to as “pixel density.” High-density screens have more pixels per inch than low-density ones. As a result, UI elements (such as a button) appear physically larger on low-density screens and smaller on high-density screens.

Screen resolution refers to the total number pixels in a display.

screen density = screen width (or height) in pixels / screen width (or height) in inches

Related

Supporting Multiple Screens

Optimize your design for different screen densities.

More Resource Types

Units of measure supported by Android.

High-density screen

Low-density screen

Density-independent pixels (dp)

"Density independence" refers to the uniform display of UI elements on screens with different densities.

Density-independent pixels (pronounced “dips”) are flexible units that scale to uniform dimensions on any screen. When developing an Android application, use dp to display elements uniformly on screens with different densities.

A dp is equal to one physical pixel on a screen with a density of 160. To calculate dp:

dp = (width in pixels \* 160) / screen density

When writing CSS, use px wherever dp or sp is stated. Dp only needs to be used in developing for Android.

Screen density

Screen width in pixels

Screen width in density-independent pixels

120

180 px

240 dp

160

240 px

240

360 px

If you have three screens, all 1.5 inches wide, with varying screen densities, the screen width will still be 240dp for all of them.

Scaleable pixels (sp)

When developing for Android, scaleable pixels (sp) serve the same function as dp, but for fonts. The default value of an sp is the same as the default value for dp.

The primary difference between an sp and a dp is that sp preserves a user's font settings. Users who have larger text settings for accessibility will see the font size matched to their text size preferences.

Designing layouts for dp

When designing layouts for the screen, calculate an element’s measurements in dp:

dp = (width in pixels \* 160) / density

For example, a 32 x 32 px icon with a screen density of 320 equals 16 x 16 dp.

Image scaling

Images can be scaled to look the same across different screen resolutions by using these ratios:

Screen resolution

dpi

Pixel ratio

Image size (pixels)

xxxhdpi

640

4.0

400 x 400

xxhdpi

480

3.0

300 x 300

xhdpi

320

2.0

200 x 200

hdpi

240

1.5

150 x 150

mdpi

160

1.0

100 x 100

Metrics & keylines

Baseline grids

All components align to an 8dp square baseline grid for mobile, tablet, and desktop. Iconography in toolbars align to a 4dp square baseline grid.

Example of baseline grid

Example of baseline grid

Type aligns to a 4dp baseline grid. See detailed information on typography.

Example of typography in a baseline grid

Keylines and spacing

The following templates and examples contain keylines, spacing guidance, and sample screens for elements on mobile, tablet, and desktop.

Templates

Mobile layout template

2.23 MB (.ai)

Tablet layout template

1.39 MB (.ai)

Desktop Layout Template

89.40 MB (.ai)

Mobile

List

A two-column, left-aligned list with a 56dp floating action button.

Keylines and margins

Screen edge left and right margins: 16dp

Content associated with an icon or avatar left margin: 72dp

Horizontal margins on mobile: 16dp

Vertical spacing

Status bar: 24dp

Toolbar: 56dp

Subtitle: 48dp

List item: 72dp

Keylines and margins

Screen edge left and right margins: 16dp

Content left margin from screen edge: 72dp

Vertical spacing

Status bar: 24dp

Toolbar: 56dp

Title and list items: 72dp

Subtitle: 48dp

Space between content areas: 8dp

Detail view

A detail card with a 56dp floating action button.

Keylines and margins

Screen edge left and right margins: 16dp

Content left margin from screen edge: 72dp

Right-side icons align 32dp from the right edge to coordinate with the floating action button.

Vertical spacing

Status bar: 24dp

Toolbar: 56dp

Space between content areas: 8dp

List item: 72dp

Navigation drawer

A side navigation menu with icons, avatars, and text aligned on the left. Other icons align on the right.

Keylines and margins

Screen edge left and right margins: 16dp

Content associated with an icon or avatar left margin: 72dp

Side nav width: The screen width minus the height of the action bar. Here, the width is 56dp from the right screen edge.

Vertical spacing

Account menu and list items: 48dp

Space between content areas: 8dp

Navigation right margin: 56dp

Tablet

List with detail view

Left-aligned list with a right-aligned floating action button

Keylines and margins

Screen edge left and right margins: 24dp

Content left margin from screen edge: 80dp

Card left and right padding: 24dp

Card content left padding: 104dp

Vertical spacing

Status bar and space above list: 24dp

List item: 64dp

Space between content areas: 8dp

List item: 72dp

List with detail view

Left-aligned list with a left-aligned floating action button

Keylines and margins

Screen edge left and right padding: 24dp

Icons’ vertical center distance from screen edge: 52dp

Nav item left padding from screen edge: 104dp

Content left margin from screen edge: 80dp

Card left and right padding: 32dp

Card nav item left padding: 96dp

Vertical spacing

Status bar: 24dp

Toolbar and list item: 64dp

Space between content areas: 8dp

Subtitle, list item, and slider: 48dp

Title: 80dp

Desktop

Keylines and spacing blocks for desktop inherit grid rules from tablet and mobile, depending on window size.

Desktop full screen

Resized windows

Horizontal window

Vertical window

Ratio keylines

The proportion of an element’s width to its height (called the aspect ratio) applies to both UI elements and screen size. It is written as width:height.

These aspect ratios are recommended:

16:9

3:2

4:3

1:1

3:4

2:3

For example:

A 1:1 aspect ratio means an element has equal height and width.

A 360dp wide image with a 2:3 aspect ratio has a height of 540.

Determine the width or height of your element for a chosen aspect ratio using the below formulas. The aspect ratio is always expressed as a fraction. For example, 3:2 is treated as 3/2.

Width = Aspect ratio \* Height

Height = Width/Aspect ratio

Screen width

Example of screen width on mobile

Element width

Example of element width on mobile

Sizing by increments

An increment is a measurement used to determine the size and position of other elements in the app.

For example, you can define an increment as the height of a standard element, such as the action bar. If the action bar is 56dp tall, you may define one increment as equal to 56 x 56dp. Determine the size of other elements by how many 56dp increments wide or tall an element should be.

If an element is 8 increments wide, using 56dp as the increment size, the width is equal to 448dp.

Incremental keylines apply mostly to desktop, often to tablet, and infrequently to mobile. The number of increments varies based on window size.

This example card width has a horizontal increment of 8x.

The height of the example extended app bar is 2x the increment, and the width of the right panel is 5x the increment.

Increments can work across many elements in material design UI.

Touch target size

To ensure balanced information density and usability, touch targets should be at least 48 x 48 dp. In most cases, there should be 8dp or more space between them.

Size elements at least 48dp high and wide to ensure a physical size of about 9mm regardless of screen size. The recommended target size for touchscreen objects is 7-10mm.

Avatar: 40dp

Icon: 24dp

Touch target on both: 48dp

Touch target height: 48dp

Button height: 36dp

Example of touch targets

Example of touch targets and buttons

Structure

UI regions

For guidance on arranging elements in z-space to achieve the correct behavior and shadows, see the Environment and Elevations and shadows sections.

Mobile structure

This structure includes a permanent app bar and floating action button. An optional bottom bar can be added for additional functionality or action overflow. Side nav menus overlay all other structural elements.

Mobile structure

Top left to right: Side nav, app bar/primary toolbar, content area (below the app bar/primary toolbar), and right nav

On the bottom: bottom bar

Tablet structure

This structure shows a permanent app bar with a floating action button. The app bar absorbs elements from the tablet and mobile bottom bars. An optional bottom bar can be added for additional functionality or action overflow. A side nav overlays all other structural elements. A right nav menu can be accessed temporarily or pinned for permanent display.

Tablet structure

Top left to right: Side nav, app bar/primary toolbar, content canvas (below the app bar/primary toolbar), and right nav

On the bottom: bottom bar

Desktop structure

The desktop structure contains a permanent app bar with a floating action button. The app bar absorbs elements from the tablet and mobile bottom bars. Where possible, the window controls are absorbed into the app bar.

Side navigation menus can take up the full height of the screen size (including under the app bar) and be accessed temporarily or pinned for permanent display. Side nav menus, as well as the content canvas, can have their own secondary toolbars for tabs, palettes, or secondary actions.

Desktop structure

Top left to right: App bar/primary toolbar

Second row from left to right: Toolbar, secondary toolbar, and toolbar

Third row from left to right: side nav, content canvas, and right nav

On the bottom: floating action button

UI regions

Define a primary horizontal or vertical divider.

Vertical divider

Horizontal divider

Avoid slicing up the interface into too many regions which can cause L shapes. Instead, use whitespace to delineate secondary areas.

Do.

Use whitespace.

Don't.

Avoid creating too many regions.

Break edges with cards and floating action buttons.

Card breaking an edge

Floating action button breaking an edge

Use cards to organize content when specific behaviors are needed or if groupings of information need more separation than what whitespace or dividers can provide.

Cards

Cards

Toolbars

Toolbars are versatile and can be used in many different ways.

Full-width, default height app bar

Full-width, extended height app bar

Column-width toolbars at multiple levels of hierarchy

Flexible toolbar and card toolbar

Floating toolbar

Detached toolbar palette

Bottom toolbar that launches to a shelf and clings to the top of the keyboard or other bottom component

Bottom toolbar shelf

App bar

The app bar, formerly known as the action bar in Android, is a special kind of toolbar that’s used for branding, navigation, search, and actions.

The nav icon at the left side of the app bar can be:

A control to open a navigation drawer.

An up arrow for navigating upward through your app’s hierarchy.

Omitted entirely if no navigation is required from this screen.

The title in the app bar reflects the current page. It can be an app title, page title, or a page filter.

Icons on the right side of the app bar are app-related actions. The menu icon opens the overflow menu, which contains secondary actions and menu items like help, settings, and feedback.

App bar structure

On the left: Nav icon, title, and filter icon

On the right: Action and menu icons

Light

Dark

Colored

Transparent

Title color

In an app bar, all icons should be the same color.

The title can have a distinct color from the icons if increased visual hierarchy is needed. Distinct title colors work best on backgrounds with enough contrast for both white and black glyphs.

Single color (default)

Distinct title color

Metrics

Default heights:

Mobile Landscape: 48dp

Mobile Portrait: 56dp

Tablet/Desktop: 64dp

For extended app bars, the height is equal to the default height plus content increment(s).

App bar height: 56dp

App bar left and right padding: 16dp

App bar icon top, bottom, left padding: 16dp

App bar title left padding: 72dp

App bar title bottom padding: 20dp

The app bar with significantly increased height for prominence

App bar height: 128dp

Action area height: 56dp

Title area height: 80dp

Title area bottom padding: 8dp

Description area height: 72dp

Description area bottom padding: 16dp

Incremental keyline is set by the app bar height, with a height of 64dp, which determines the keyline increment.

App bar left and right content padding: 24dp

App bar icon top and bottom padding: 20dp

Prominent app bar height: 128dp

App bar left and right icon padding: 24dp

App bar left content padding: 80dp

Density

When the mouse and keyboard are the primary input methods, measurements may be condensed to accommodate denser layouts.

Dense app bar on desktop

App bar height: 48dp

App bar left and right content padding: 24

App bar left content padding: 80dp

Icon height with touch target: 40dp

Dense prominent app bar

Dense prominent app bar height: 96dp

App bar left and right content padding: 24dp

App bar left content padding: 80dp

Menus

A menu is a temporary sheet of paper that always overlaps the app bar, rather than behaving as an extension of the app bar.

Example of an app bar

Example of menu in an app bar

System bars

Status bar/window bar

On Android, the status bar contains notification icons and system icons. On Chrome, the top bar contains the window controls: minimize, full screen, and close. In a Chrome app, the top bar can disappear, and the window controls are then brought into the app bar.

Android status bar

Chrome window bar

Metrics:

Android status bar height: 24dp

Chrome window height: 32dp

Android status bar on top of the app bar

Chrome window bar on top of the app bar

Chrome window controls inside app bar

Full screen

Some content is best experienced in full-screen mode, such as videos, games, books, and slides. Full-screen mode increases engagement with content by minimizing visual distraction from controls and protecting users from escaping the app accidentally.

Full-screen mode comes in several types:

Lean back

Immersive

Lights Out

Lean back

Lean back mode is best suited to watching content with limited screen interaction, such as a video.

Interaction: Touch the screen anywhere to bring back the system bars.

Touch the screen anywhere to bring back the system bars

Immersive

Immersive mode is best when users need to interact heavily with the screen, such as playing a game or interacting with an image gallery. You may show and hide your app’s controls along with the system bars as needed.

Interaction: Swipe from any edge of the screen to make the system bars appear. The first time an app goes full-screen, this swipe gesture is explained.

Edge swipe exception: Apps that use the swipe-from-edge gesture to perform actions should also display the system bars when that gesture is used. For example, a drawing app that uses an edge swipe (such as to draw a line) should also display the system bars semi-transparently for a few seconds any time that gesture occurs.

Swipe from the any edge of the screen with a hidden bar to bring back system bars.

Lights Out

In Lights Out mode, the Action Bar and Status Bar fade away and become unavailable after a few seconds of inactivity. The Navigation Bar is still available and responds to touches but appears dimmed.

Dark status bar

By default, the color of the status or window bar is a darker shade of the app bar color. It can also use color from another element in the layout or be translucent.

Color is based on a sample taken from the content

Translucent status bar, 20% Black #000000

Dark status bar

Status bar color in a darker tone of the app bar color

Light status bar

The light status bar, with dark icons, better harmonizes with light content and can be used as an alternative to the dark status bar.

Light status bar color is based on sample taken from content

Translucent light status bar, 70% White #FFFFFF

Light status bar, default background fill is #E0E0E0

Light status bar in darker tone of a light app bar color

Android navigation bar

The navigation bar in Android houses the device navigation controls: Back, Home, and Overview. It also displays a menu for apps written for Android 2.3 or earlier.

Height: 48dp

Dark

Light

Color variants

The navigation bar can be opaque, translucent, or transparent.

Translucent

Translucent over complex image

Transparent

Transparent over even-toned image

Chrome OS shelf

The shelf houses the app launcher, application icons, and system settings on Chrome OS.

Height: 56dp

Example of Chrome OS shelf

Side nav

If present, side nav bars can be pinned for permanent display or float temporarily as overlays. Side navs can be positioned on the left or right side of the screen.

The content appearing in the left nav is ideally navigation- or identity-based. Right nav content should be secondary to the main content on a page.

See also Navigation drawer.

Mobile screen

Side nav menu

Structure

Side nav bars may be pinned for permanent display, or they can float temporarily as overlays. Temporary nav drawers overlay the content canvas; whereas pinned nav panels are situated alongside or below the content canvas.

Screen size can determine whether a panel is pinned or an overlay. With sufficient space, a panel can be pinned and content adjusted responsively. With insufficient space, a panel has to be an overlay.

Metrics

Mobile:

Width = Screen width − 56 dp

Maximum width: 320dp

Maximum width applies only when using a left nav. When using a right nav, the panel can cover the full width of the screen.

Desktop: Maximum width for a left nav is 400dp. The right nav can vary depending on content.

Left nav on mobile

Right nav on mobile

Left nav on desktop

Right nav on desktop

Whiteframes

Whiteframes provide a variety of layout structures using a consistent approach to surfaces, layering, and shadows.

For download and other information about whiteframes, see Resources.

Carded content that expands and collapses

Overlayed content details with focused app bar on mobile

Left side nav and one-up stream on mobile

Source list

Full-screen background image with inset search field and carded search results

Expandable footer drawer

Responsive UI

Responsive layouts in material design adapt to any possible screen size. This UI guidance includes a flexible grid that ensures consistency across layouts, breakpoint details about how content reflows on different screens, and a description of how an app can scale from small to extra-large screens.

Breakpoints

For optimal user experience, material design user interfaces should adapt layouts for the following breakpoint widths: 480, 600, 840, 960, 1280, 1440, and 1600dp.

Resizer

Preview material design breakpoints with this interactive viewer across desktop, mobile, and tablet

1. Summary and detail view content in layouts

Layouts under 600dp wide may fill the screen with a single level of content hierarchy (either summary or detail content, but not both).

Layouts over 600dp wide may place two levels of content hierarchy on the screen (both summary and detail content).

2. Max screen widths

Layouts over 1600dp wide may let the layout grow until it hits a max width. At this point, the grid may do one of the following:

Become center aligned with increased margins

Remain left aligned while the right margin grows

Continue to grow while revealing additional content

Breakpoint system

These breakpoints describe column and width specifications for different screens, devices, and orientations.

For some measurements, the values remain the same even if a device is rotated. For that reason the smallest width in either orientation is the defining value.

\* 16dp when the smallest width of the device is <600

\*\* Desktop breakpoints are 16dp below the listed values to accommodate variations in browser chrome.

Breakpoint (dp)

Handset / Tablet Portrait

Handset / Tablet Landscape

Window

Columns

Gutter

0

small handset

xsmall

4

16

360

medium handset

xsmall

4

16

400

large handset

xsmall

4

16

480

large handset

small handset

xsmall

4

16

600

small tablet

medium handset

small

8

16/24\*

720

large tablet

large handset

small

8

16/24\*

840

large tablet

large handset

small

12

16/24\*

960

small tablet

small

12

24

1024\*\*

large tablet

medium

12

24

1280\*\*

large tablet

medium

12

24

1440\*\*

large

12

24

1600\*\*

large

12

24

1920\*\*

xlarge

12

24

Grid

Material design’s responsive UI is based on a 12-column grid layout. This grid creates visual consistency between layouts, while allowing flexibility across a wide variety of designs. The number of grid columns varies based on the breakpoint system.

This animation shows how surfaces and panels can align to influence the 12-column grid.

Margins and Gutters

The responsive grid focuses on consistent margin and gutter widths, rather than column width. Material design margins and columns follow an 8dp square baseline grid. Margins and gutters can be 8, 16, 24, or 40dp wide.

Margins and gutters don’t need to be equal. For example, it’s acceptable to use 40dp margins and 24dp gutters in the same layout.

This animation shows interactions of the following margin and gutter width variations:

8dp margins and gutters

16dp margins and gutters

24dp margins and gutters

40dp margins and gutters

40dp margins and 24dp gutters

Full-width vs Centered

Full-width grids use fluid columns and breakpoints to determine when a layout needs to change.

Centered grids use fixed columns and reflow the layout when all columns (plus a defined margin) no longer fit on screen.

Full-width grids

Centered grids

Panel Influence on the Grid

As defined in Navigation patterns, the side nav may be permanent, persistent, or temporary. These behaviors apply to any panel, not just a side nav.

Permanent

A permanent panel exists outside of the responsive grid. The panel appears at a defined breakpoint (when the screen can accommodate the panel) and squeezes content. There are no controls to show/hide the panel.

The effects of a permanent panel on the responsive grid

Side panel effects on the grid

This animation happens in two phases:

A persistent side panel appears, squeezing both content and the grid. Content is accessible while the panel remains visible. The panel hides upon toggling.

A temporary side panel appears, pushing grid content off-screen. Touching either outside the panel, or an item inside the panel, hides the panel.

The effects of a persistent panel on the responsive grid

Temporary overlay

A temporary panel does not affect the grid or content when off-screen. When toggled to be visible, it can be hidden again by touching anywhere outside the panel or an item inside the panel.

The effects of a temporary panel on the responsive grid

Surface behaviors

When a screen size changes, the UI adapts using the following surface-specific behaviors.

Visibility

Behavior

Definition

Permanent

When screen space is available, a surface is always visible.

Persistent

Surface visibility can be toggled between visible and hidden. When visible, interacting with other elements on the screen does not change visibility.

Temporary

Surface visibility can be toggled between visible and hidden. When visible, interacting with other elements on the screen toggles the surface to become hidden or minimized.

Width

Behavior

Definition

Fixed

Element width stays the same when screen size changes.

Fluid

Element width grows as screen size changes.

Sticky

Element width is fixed until influenced by another element or breakpoint.

Squeeze

Element width contracts as a panel is revealed

Push

Element width stays the same, its position changes horizontally as a panel appears, and it may be partially occluded by a screen’s edge.

Overlay

Element width and position stays the same as a panel appears over content.

Descriptors

Behavior

Definition

Above, Below

The y position of an element.

Over, Under

The z position of an element in motion.

In Front, Behind

The static z position of an element.

Left, Right, Centered

The x position of an element

Top, Bottom

The y position of an element and its position relative to a screen edge.

Flat, Raised

The z position, and shadow of an element. A flat element will have no shadow.

Inset, Full Bleed

The padding of an image or element

Summary, Detail

A content summary, and the full expansion of the summary

Patterns

As screen space increases, the following responsive patterns may be applied:

Reveal

Transform

Divide

Reflow

Expand

Position

Reveal

UI hidden by smaller screens may be revealed with increased available space.

Elements, such as a side nav, may become visible.

A simple UI may reveal more powerful or complex options.

Content on a small screen that only appears after tapping an element may be revealed by default when more space is available.

Transform

A UI element may transform from one format to another.

Side navigation may transform into tabs.

A list may transform into a grid list.

Menu items may transform into icons in a toolbar.

Divide

A layered UI may be divided into newly available space.

Side navigation, list content, and detailed content divide to fill a single view.

A left panel, list content, and a right panel divide within a single view as space increases.

Tabbed sibling content is divided within the same view.

Reflow

A UI may reflow into available space.

Elements from a single-column format may reflow to fill the content area in various combinations.

Horizontal tabs may reflow into a vertical list.

Elements may reflow within a component based on a new screen ratio or device orientation.

Expand

The UI may expand across more space.

Content cards may expand to fill the new space.

Dialogs may expand proportionally with content or in specific increments.

Position

The position of UI components may change to more appropriate locations.

A bottom sheet on a small screen may reposition itself as a menu.

A floating action button (FAB) may move to a more visible location relative to other UI elements.

Split screen

Android only

Split-screen mode allows two activities to be visible on screen at the same time.

Usage

User interaction required

Apps are not allowed to invoke split-screen mode by themselves; the user must activate it.

Launching new activities

When an app launches a new activity, it will launch within the same portion of the screen by default. Apps will be resized to fit the split-screen view as necessary (unless your app isn’t compatible with the split-screen format).

When split-screen mode is active, apps may launch new activities in the opposite portion of the screen if the following criteria are met:

The original and new activities are related to each other and usable while in split-screen mode

The task the app helps accomplish warrants displaying two activities at once, and hiding the other app that was open

It is clear that the app will enter split-screen mode, such as with a button stating “Open in new window”

Split screen shown in portrait mode on mobile (on the left) and landscape mode on tablet (on the right)

Behavior

Split-screen mode remains active until the user cancels it or switches to an incompatible app.

Resizing split screen

Users may move and resize each screen in split-screen mode by dragging the divider between the two split screens.

Edge swipe gestures

When split-screen mode is active, the edge swipe gesture will likely not work as intended. Because split-screen mode relies on swiping to resize each split screen, if your app also relies on edge swipes to perform actions, then it’s possible that the swipe will either trigger screen resizing or an action in your app.

To avoid this, edge swipes should not be the only way to perform actions in your app. There should be an alternative method to perform each action.

For example, the navigation drawer enables an edge swipe to open the drawer, but it is also accessible by pressing the menu icon.

Layout

To support split-screen usage, viewable content should be scaled to an appropriate size and density.

Primary controls should be adapted for split-screen mode. For example, navigation tabs may be collapsed into a menu.

Responsive UI

Apps in split-screen mode should elegantly adjust across device sizes and orientations.

Changing a device’s orientation should not cause the UI to change unexpectedly. For example, an app displaying a video in one of the split screens (in portrait mode) should not begin playback in full-screen mode if the device rotates to landscape mode.

Apps may use the same or different layouts for mobile and tablet:

Apps with similar layouts for mobile and tablet may switch between the tablet and mobile UIs when the app is resized, as the transition will not be jarring.

Apps with completely different layouts for mobile and tablet should avoid using the mobile UI on tablet in split-screen mode. Instead, the existing tablet UI should be adapted to fit the smaller size to ensure that users have a consistent experience on both devices.

An app may use the same layout across mobile (left) and tablet (right).

An app may use different layouts across mobile (left) and tablet (right).

Design for condensed sizes

To simplify adapting your app for the various sizes of split-screen mode, it is recommended to design for the smallest size first.

Create a layout that works at 220dp wide or tall by condensing elements or removing non-essential ones. The layout may be scaled upward from there.

When the split screen is active, the aspect ratio of an app in portrait orientation is 16:9 on mobile.

Bottom navigation

Bottom navigation bars make it easy to explore and switch between top-level views in a single tap.

Tapping on a bottom navigation icon takes you directly to the associated view or refreshes the currently active view.

Bottom navigation is primarily for use on mobile. To achieve a similar effect for desktop, use side navigation.

Usage

Three to five top-level destinations

Destinations requiring direct access

Color

Tint the active icon with the app’s primary color. Use black or white if the bottom navigation bar is already colored.

Specs

Width of each action: The width of the view divided by the number of actions (with a max of 168dp and a minimum of 80dp)

Height: 56dp

Icon: 24 x 24dp

Usage

Bottom navigation provides quick navigation between top-level views of an app. It is primarily designed for use on mobile.

Larger displays, like desktop, may achieve a similar effect by using side navigation. For instance, the compact “rail” treatment displays navigational icons by default.

The bottom navigation bar on mobile

Left navigation on a larger display, such as tablet or desktop

When to use

Bottom navigation should be used for:

Three to five top-level destinations of similar importance (alternative: a persistent navigation drawer accessible from anywhere in the app)

Destinations requiring direct access from anywhere in the app (alternative: tabs for only one or two destinations)

Do.

The bottom navigation bar exposes the three to five top-level destinations of an app.

Don't.

If there are fewer than three destinations, consider using tabs instead.

If your top-level navigation has more than six destinations, provide access to destinations not covered in bottom navigation through alternative locations, such as a navigation drawer.

Do.

Views are fixed in a bottom navigation bar.

Don't.

Avoid scrollable content in the bottom navigation bar.

Do.

Use up to five top-level destinations in a bottom navigation bar.

Don't.

Avoid using more than five destinations in bottom navigation as tap targets will be situated too close to one another.

Bottom navigation and tabs

Be cautious when combining bottom navigation with tabs, as the combination may cause confusion when navigating an app. For example, tapping across both tabs and bottom navigation could display a mixture of different transitions across the same content.

Style

Icons and text

Because bottom navigation actions are presented as icons, they should be used for content that can be suitably communicated with icons.

Style each action according to the following conditions:

When the view is in focus, display that view’s icon and text label

When there are only three actions, display both icons and text labels at all times

If there are four or five actions, display inactive views as icons only

Color

Tint the current bottom navigation action (including the icon and any text label present) with the app’s primary color.

Do.

Use the app’s primary color to indicate the view in focus.

Don't.

Avoid using different colored icons and text labels.

If the bottom navigation bar is colored, make the icon and text label of the current action black or white.

Do.

Use black or white iconography if the bottom navigation bar is colored.

Don't.

Avoid pairing colored icons with a colored bottom navigation bar.

Text Labels

Text labels provide short, meaningfully definitions to bottom navigation icons. Avoid long text labels as these labels do not truncate or wrap.

Do.

Use short labels

Don't.

Avoid labels with wrapping text

Don't.

Avoid truncating text labels as doing so may prevent comprehension.

Don't.

Avoid shrinking text labels to fit on a single line.

Behavior

Tapping on a bottom navigation icon takes you directly to the associated view, or refreshes the currently active view.

Each bottom navigation icon must lead to a destination, and may not open menus or other pop-ups.

On Android, the Back button does not navigate between bottom navigation bar views.

Tapping on the active action in the bottom navigation bar will navigate the user to the top of the view.

Navigation through the bottom navigation bar should reset the task state.

The bottom navigation bar can appear and disappear dynamically upon scrolling:

Scrolling downward hides the bottom navigation bar

Scrolling upward reveals it

The bottom navigation bar can disappear and reappear to allow immersive content views.

The bottom navigation bar remains in view when navigating through the app’s hierarchy.

Using swipe gestures on the content area does not navigate between views.

Do.

Transition between active and inactive views using a cross-fade animation.

Don't.

Avoid using lateral motion to transition between views.

Specs

Fixed bottom navigation bar

To calculate the width of each bottom navigation action, divide the width of the view by the number of actions:

Width of action = Width of entire view / Number of actions

For example, if a view is 360dp wide, and there are three actions, each action should be 120dp wide.

Desktop and larger views

Alternatively, make all bottom navigation actions the width of the largest action.

Fixed bottom navigation bar on mobile

Fixed bottom navigation bar on mobile

Width minimum and maximum (these values include padding):

Maximum: 168dp

Minimum: 80dp

Height:

56dp

Icon:

24 x 24dp

Content alignment:

Text and icon are centered horizontally within the view.

Padding:

6dp above icon (active view), 8dp above icon (inactive view)

10dp under text

12dp left and right of text

Text label:

Roboto Regular 14sp (active view)

Roboto Regular 12sp (inactive view)

80dp min width

12dp padding

56dp height

12dp left and right of text

10dp under text

168dp max width

12dp padding

Example of bottom navigation bar

Fixed bottom navigation bar on landscape mobile

56dp height

24 x 24dp icon

6dp above icon (active view)

8dp above icon (inactive view)

10dp under text

Fixed bottom navigation bar on tablet

56dp height

24 x 24dp icon

6dp above icon (active view)

8dp above icon (inactive view)

10dp under text

Shifting bottom navigation bar

Width minimum and maximum (this includes padding):

Active view

Maximum: 168dp

Minimum: 96dp

Inactive view

Maximum: 96dp

Minimum: 56dp

Height:

56dp

Icon:

24 x 24dp

Content alignment:

Text and icon are centered horizontally within view.

Padding:

6dp above icon (active view), 16dp above and below icon (inactive view)

10dp under text

Text label:

Roboto Regular 14sp (active view)

Shifting bottom navigation bar on mobile

56dp height

24 x 24dp icon

6dp above icon (active view)

10dp under text

6dp below icon

16dp below icon with no title

Shifting bottom navigation bar on mobile

Active view: 96dp min width

Inactive view: 56dp min width

Text labels may use the maximum width on the active view.

Active view: 168dp max width

Inactive view: 96dp max width

Text labels may use the maximum width on the active view.

Shifting bottom navigation bar on landscape mobile

56dp height

24 x 24dp icon

6dp above icon (active view)

10dp under text

6dp below icon

16dp below icon with no title

Shifting bottom navigation bar on tablet

56dp height

24 x 24dp icon

6dp above icon (active view)

10dp under text

6dp below icon

16dp below icon with no title

Elevation

Because snackbars have a lower (6dp) elevation, they appear behind the bottom navigation bar (8dp elevation).

Bottom sheets, navigation drawers, and keyboards appear in front of the bottom navigation bar, temporarily covering it.

Snackbars appear behind the bottom navigation bar.

Bottom sheets and keyboards appear in front of the bottom nav bar, temporarily covering it.

Orthographic view of app structure

Bottom sheets

Bottom sheets slide up from the bottom of the screen to reveal more content.

Modal bottom sheets are primarily for mobile and can also present deep-linked content from other apps.

Persistent bottom sheets integrate with the app to display supporting content.

Interaction

Bottom sheets are displayed only as a result of a user-initiated action.

Elevation

Modal bottom sheets: Higher than the app

Persistent bottom sheets: Same elevation as the app

Alternatives

Simple dialogs

Menus

Usage

There are two major types of bottom sheets:

Modal bottom sheets are alternatives to menus or simple dialogs. They can also present deep-linked content from other apps. They are primarily for mobile.

Persistent bottom sheets present in-app content.

Elevation distinguishes modal from persistent bottom sheets. Modal bottom sheets rest at a higher elevation than the app’s content; whereas persistent bottom sheets rest at the same elevation as the app and integrate with its content.

On larger screens, where space is less constrained, using alternative surfaces and components such as simple dialogs and menus may be more appropriate than bottom sheets.

Modal bottom sheets slide in over an app’s content.

Persistent bottom sheets are an integral part of an app’s layout.

Persistent Bottom Sheets

Persistent bottom sheets display in-app content that supplements the main view. It remains visible even when not actively in use, resting at the same elevation as an app and integrating with its content.

Usage

To introduce new content on a unique surface

To display content equal in value to the primary content

Introducing a persistent bottom sheet into a layout with a floating action button (FAB) may cause the FAB to move vertically.

A persistent bottom sheet enables place details to remain on-screen while a map is panned around.

Playback controls remain visible on a persistent bottom sheet regardless of user navigation in the main view.

Mobile

Mobile persistent bottom sheets are full width in both portrait and landscape view.

Tablet/Desktop

Persistent bottom sheets are either full width or inset, as determined by the width of the content they contain, overall UI, underlying content, and stylistic choices.

A full-width persistent bottom sheet on tablet

An inset persistent bottom sheet on tablet

On desktop, a persistent bottom sheet may transform into another surface or sheet of material.

On desktop, content ordinarily presented in a persistent bottom sheet might move onto a new sheet of material. On larger screens it may be more appropriate to display bottom sheet content on a sheet of material positioned on the left side of the interface.

Modal bottom sheets

Modal bottom sheets are alternatives to menus, or simple dialogs, and can display deep-linked content from another app. They appear above other UI elements and must be dismissed in order to interact with the underlying content. When a modal bottom sheet slides into the screen, the rest of the screen dims, giving focus to the bottom sheet.

A modal bottom sheet with a set of actions in a list

A modal bottom sheet with a set of actions in a grid

Usage

Modal bottom sheets may:

Present actions in a list or grid as an alternative to menus or simple dialogs.

Display a contextual menu, when there is no obvious entry point for a menu.

Prioritize the visibility of the elements they contain.

Modal bottom sheets can display long menu item names, menu items with subtext, and icons associated with menu items.

When displaying menu items, fully-expanded modal bottom sheets maintain a minimum 8dp distance from the bottom of the app bar.

The bottom sheet may associate icons with menu items.

Menu items with subtext

Deep linking

Modal bottom sheets may be used to provide deep links to content or controls from another app. These span the full screen width.

A deep-linked modal bottom sheet displays a definition of a term without the user having to open the dictionary app.

The app on the right displays a bottom sheet containing content from the app on the left. This allows the user to view content from another app without leaving their current app.

Bottom sheets may contain links that perform deep navigation within another app. These links may:

Allow the user to travel multiple levels deep within another app

Return to the level on which they started

However, these deep links may not allow the user to navigate upward in the hierarchy: they may only stay at the initial level, go deeper, or travel back to the initial level.

Alternatively, deep links may navigate the user away from the bottom sheet entirely, into another view.

To enable upward navigation from the bottom sheet’s deep links, offer an explicit link to open the application via an overflow menu. Actions within the deep-linked bottom sheet may result in opening the parent application, such as with an “add contact” action.

Do.

Fully-expanded modal bottom sheets provide an X in the app bar.

Don't.

When navigating downwards into the content hierarchy, it would be appropriate to show an Up arrow.

Mobile

Modal bottom sheets are primarily a mobile component, where they span full screen width in both portrait and landscape views.

Do.

For modal bottom sheets, don’t initially overlap the app bar. Allow the user to tap/swipe to dismiss.

Do.

When displaying a long list, modal bottom sheets can scroll internally and expand past the 16:9 keyline.

Don't.

The height of the bottom sheet should be dictated by the amount of content it contains.

Don't.

For modal bottom sheets, don’t initially overlap the app bar. Retain an area for the user to tap outside the bottom sheet to dismiss it.

Tablet/Desktop

Consider alternatives to modal bottom sheets on larger screens. Because bottom sheets may appear far from the place where the user clicked or touched to summon them, it splits the user’s attention across two different parts of the screen.

Because desktop space is less constrained, it can support several alternatives to modal bottom sheets.

Bottom sheet alternative

Benefit

Menu

Keeps action choices close to the point of interaction

Simple dialogs

Strongly interrupts the user’s current context

Inline expansion within the same view

Is not as interruptive as a menu or dialog

Showing all options as a persistently accessible element

Does not require a button press to access them

Opening a new browser window

Flexible sizing and layout independence

Menus keep the choice of actions close to the point of interaction. Placing these menu options into a bottom sheet would not be recommended.

Make modal bottom sheets on large screens use appropriate layouts for the amount of extra space.

A grid list bottom sheet on tablet

A long list bottom sheet on tablet

Behavior

Bottom sheets can be dismissed by swiping the bottom sheet down, by touching an explicit control such as an X in the app bar, or by touching the system back button (Android).

Modal bottom sheets can also be dismissed by touching outside of the bottom sheet.

Dismiss by swiping the bottom sheet down

Dismiss by touching outside of the bottom sheet surface

Dismiss by touching the “X” dismiss action (if there is an app bar)

Specs

The following specs are provided for mobile apps.

Font and color

Text: Roboto Regular 16sp, #000 87%

Title (optional): Roboto Regular 16sp, #000 54%

Default bottom sheet background fill: #FFF

Transparent overlay fill: #000 20%

List-style bottom sheet

Screen edge left & right margins: 16dp

Top and bottom margins: 8dp

List-item height: 48dp

List icons: 24x24dp, middle vertical alignment

Text associated with icon: 32dp horizontal padding

Keylines for list-style bottom sheet in a mobile app

List-style bottom sheet in a mobile app

List-style sheet with header

Screen edge left & right margins: 16dp

Space below the top list item: 7dp

Divider line: 1dp

Space above second bottom sheet: 8dp

List-item height: 48dp

List icons: 24x24dp, middle vertical alignment

Text associated with icon: 32dp horizontal padding

List title height: 56dp

Keylines for list-style sheet with header

List-style sheet with header

Grid-style bottom sheet with icons

Margin on all sides of grid: 24dp

Space between grid list rows: 24dp

Grids list icons: 48x48dp, equally spaced across each row

Grid list label: 16dp tall, center-aligned below each icon

Margin below icon: 8dp

Keylines for grid-style bottom sheet that contains a standard set of actions for other apps

Grid-style bottom sheet that contains a standard set of actions for other apps

Grid-style bottom sheet with header

Screen edge left & right margins: 24dp

Space below list row: 8dp

Divider line: 1dp

Space above and below divider line: 8dp

List icons: 48x48dp, middle vertical alignment

Grid list label: 16dp tall, center-aligned below each icon

List title height: 56dp

Keylines for grid-style bottom sheet with header

Keylines for grid-style bottom sheet with header

Bottom sheets slide up from the bottom of the screen and initially cover a portion of the screen.

The initial height of a bottom sheet is relative to the height of the list items (or grid rows) it contains. A bottom sheet should not initially have a height beyond its 16:9 ratio keyline, depending on how much content it contains. Bottom sheets may be swiped up to fill the height of the screen, with content that then scrolls internally.

The initial height of a bottom sheet should not extend beyond the 16:9 ratio keyline.

The minimum height of a bottom sheet is relative to the height of the list items it contains.

Specs for desktop and tablet apps

Modal and Inset persistent bottom sheets on desktop and tablet use these minimum and maximum widths.

The width of a bottom sheet is determined by the structure of the overall interface, which is typically based on increments. On a page grid, the maximum width of the bottom sheet is relative to the page grid and content it rests on.

For example, on a 960dp wide screen, if you use the height of the app bar (64dp) as your increment, it would result in fifteen total increments wide. A minimum of one increment should separate your bottom sheet from the screen edge, and the bottom sheet itself should be at least six increments wide.

Screen width

Minimum distance from screen edge (in increments)

Minimum sheet width (in increments)

960dp

1 increment

6 increments

1280dp

2 increments

8 increments

1440dp

3 increments

9 increments

Breakpoints for various desktop screen sizes

Buttons

Buttons communicate the action that will occur when the user touches them.

Material buttons trigger an ink reaction on press. They may display text, imagery, or both. Flat buttons and raised buttons are the most commonly used types.

Additional button types include:

Persistent footer buttons are flat buttons that may be used in screen footers or dialogs.

Dropdown buttons display multiple selections.

Toggle buttons group related options. Icon toggles allow a single choice to be selected or deselected.

Flat buttons

Flat buttons are text-only buttons.

They may be used in dialogs, toolbars, or inline.

They do not lift, but fill with color on press.

Raised buttons

Raised buttons are rectangular-shaped buttons.

They may be used inline.

They lift and display ink reactions on press.

Elevation

Flat buttons: 0dp

Raised buttons: 2dp

Button types

There are three standard types of buttons:

Floating action button: A circular material button that lifts and displays an ink reaction on press.

Raised button: A typically rectangular material button that lifts and displays ink reactions on press.

Flat button: A button made of ink that displays ink reactions on press but does not lift.

Floating action button

Raised button

Flat button

Example of a floating action button

Example of a raised button

Example of a flat button

Choosing button style

Choosing a button style depends on the primacy of the button, the number of containers on screen, and the screen layout.

Function: Is it important and ubiquitous enough to be a floating action button?

Dimension: Choose raised or flat depending on the container it will be in and how many z-space layers you have on screen. There should not be many layers of objects on the screen.

Layout: Use primarily one type of button per container. Only mix button types when you have a good reason to, such as emphasizing an important function.

Usage

Button types

The type of button used should be suited to the context in which it appears.

Context

Button type

Dialogs

Use flat buttons in dialogs.

Inline

Raised buttons or flat buttons can be used for inline buttons.

Always available

If your app requires actions to be persistent and readily available, you can use the floating action button or persistent footer buttons.

Recommended button placement

Standard dialogs

Button alignment on screen: right

Place the affirmative button on the right, the dismissive button on the left.

Forms

Button alignment on screen: Left

Place the affirmative button on the left, the dismissive button on the right.

Cards

Buttons are best placed on the left side of a card to increase their visibility. However, as cards have flexible layouts, buttons may be placed in a location suited to the content and context, while maintaining consistency within the product.

Non-standard dialogs and modal windows

Button placement in non-standard dialogs and modal windows depends on the complexity of the content they contain.

For dialogs with relatively simple content, it is recommended to place buttons on the right side of a dialog, with the affirmative button to the right of the dismissive button.

For lengthy or complex forms, it is recommended to place buttons on the left of the form, with the affirmative button to the left of the dismissive button.

Style

Typography

Button text should be capitalized in languages that have capitalization. For other languages, colored text on flat buttons distinguishes them from normal text.

Accessibility

To ensure usability for people with disabilities, give buttons a height of 36dp and give touchable targets a minimum height of 48dp.

Button height: 36dp

Touchable target height: 48dp

Corner radius

Buttons should have a 2dp corner radius.

Do.

Buttons should have a 2dp corner radius

Density

When mouse and keyboard are the primary input methods, button measurements can be slightly reduced to accommodate dense UIs.

Default Button

Button text: 14pt Roboto medium

Button height: 36dp

Button text left and right padding: 16dp

Corner radius: 2dp

Dense Button

Button text: 13pt Roboto medium

Button height: 32dp

Button text left and right padding: 16dp

Corner radius: 2dp

Size and padding

Examples of flat buttons in dialogs

Flat Button

Height: 36dp

Minimum width: 88dp

Touch target height: 48dp

Corner radius: 2dp

Horizontal margin: 8dp

Horizontal padding: 8dp

Flat buttons

Usage

Flat buttons are printed on material. They do not lift, but fill with color on press.

Use flat buttons in the following locations:

On toolbars

In dialogs, to unify the button action with the dialog content

Inline, with padding, so the user can easily find them

Flat buttons are appropriate in dialogs.

Flat buttons minimize distraction from content.

Do.

Correct use of flat buttons.

Don't.

These raised buttons appear too prominently.

Specs

Flat Light/Light theme

Minimum width: 88dp

Height: 36dp

Corner radius: 2dp

Pressed: 40% #999999

Disabled text: 26% #000000

Flat Dark/Dark theme

Minimum width: 88dp

Height: 36dp

Corner radius: 2dp

Pressed: 25% #CCCCCC

Disabled text: 30% #FFFFFF

Dialog text padding: 24dp

Dialog button container height: 52dp

Dialog buttons: 8dp padding from container edges, aligned to the right of container (for LTR scripts)

Flat buttons in dialogs

Dialog buttons

Button height: 36dp

Button top margin: 24dp

Button padding: 8dp

Button alignment: Right edge for LTR scripts

Button width: Varies based on button text length

Flat buttons with 12% opacity on focus

Behavior

Flat buttons

Raised buttons

Usage

Raised buttons add dimension to mostly flat layouts. They emphasize functions on busy or wide spaces.

Use raised buttons to give more prominence to actions in layouts with a lot of varying content.

Raised buttons can help delineate sections of content on a page.

Do.

Raised buttons stand out more than flat buttons.

Don't.

Don’t use flat buttons in UIs where they would be difficult to see.

Raised buttons behave like a piece of material resting on another sheet – they lift and fill with color on press.

Raised Light/Light theme

Minimum width: 88dp

Height: 36dp

Corner radius: 2dp

Disabled text: 26% #000000

Disabled button: 12% #000000

Focused:12% #000000 shade over the color

Raised Dark/Dark theme

Minimum width: 88dp

Height: 36dp

Corner radius: 2dp

Normal color: 500

Pressed color: 700

Disabled text: 30% #FFFFFF

Disabled button: 12% #FFFFFF

Focused:12% #000000 shade over the color

Button Elevation

Raised buttons have a default elevation of 2dp.

On desktop, raised buttons can gain this elevation on hover.

Dark/Light theme

Minimum width: 88dp

Height: 36dp

Elevation: 0dp

Disabled text: 30% #FFFFFF

Disabled button: 12% #FFFFFF

Behavior

Raised buttons

Persistent footer buttons

If your app requires actions to be persistent and readily available to the user, consider using the floating action button or persistent footer buttons.

Persistent footer buttons are 48dp tall.

Example of persistent footer buttons

Do not use raised buttons within persistent button areas.

Don't.

A persistent footer button may be used for scrollable dialogs when a divider is added.

Persistent footer button in dialog

Persistent footer button in scrollable dialog

Dropdown buttons

Mobile dropdown buttons

Dropdown button

A dropdown button selects between multiple selections. The button displays the current state and a down arrow. Available states may be shown as a list of strings, a palette, or icons, for example.

When a user interacts with the button, a menu covers the button and displays the possible states. Pressing a state dismisses the menu and updates the button to display this new state.

Scrolling within the dropdown behaves the same way a menu scrolls.

Closed dropdown button, open dropdown menu, and selected dropdown menu

Generic dropdown button with a list menu

Generic overflow dropdown button

The generic overflow dropdown button displays an arrow or menu button by default. When the button is pressed, the menu appears. Pressing an option on the menu navigates to further settings for that option.

Segmented dropdown button

A segmented dropdown has two sections: the current state and the dropdown arrow icon. Pressing the current state will cause that state’s action to fire within the screen. Pressing the dropdown arrow will display all the state options. Selecting one will change the displayed state.

Editable segmented dropdown button

This button has a segmented dropdown, where the displayed state is text editable, such as a font size picker. Pressing the current state causes both that state’s action to fire and makes the displayed state editable. Pressing the arrow displays all the state options.

Generic, segmented, and editable overflow dropdown buttons in normal, hover, focus, pressed, and selecting states

Animation of dropdown button

Desktop dropdowns

Desktop app bar specs

Desktop dropdown

Top and bottom padding: 16dp

Left padding: 24dp

Right margin: 16dp

Horizontal space between font styles: 24dp

Toggle buttons

Toggle buttons may be used to group related options. Arrange layout and spacing to convey that certain toggle buttons are part of a group.

Focus and pressed states may reinforce that toggles are part of a group. For example, when one button segment is focused, focus may be displayed simultaneously on the rest of the segment.

Toggle button requirements:

Have at least three toggle buttons in a group

Label buttons with text, an icon, or both

The following combinations are recommended:

Multiple and unselected

Exclusive and unselected

Exclusive only

Exclusive selection

Text justification toggle buttons present options for left, right, center, full, and justified text with only one item available for selection at a time. Selecting one option deselects any other.

No options are selected

Multiple selection

Logically-grouped options, like Bold, Italic, and Underline, allow multiple options to be selected.

One option is selected

Icon toggles

Icons are appropriate for toggle buttons that allow a single choice to be selected or deselected, such as adding or removing a star to an item.

They are best located in app bars, toolbars, action buttons or toggles.

Icon toggles may display bounded or unbounded ink surface reaction ripples beyond their touch-target bounds.

Icon toggles in normal, hover, focused, pressed, and inactive states

The icon toggle focus indicator color and opacity are related to the color of the icon.

Buttons: Floating Action Button

A floating action button represents the primary action in an application.

A floating action button is used for a promoted action.

Shaped like a circled icon floating above the UI, it changes color upon focus and lifts upon selection. When pressed, it may contain more related actions.

Usage

Only one floating action button is recommended per screen to represent the most common action.

Behavior

It animates onto the screen as an expanding piece of material, by default.

Sizes

Default: 56 x 56dp

Mini: 40 x 40dp

Floating action button

Floating action button

Floating action buttons are used for a promoted action. They are distinguished by a circled icon floating above the UI and have motion behaviors that include morphing, launching, and a transferring anchor point.

Floating action buttons come in two sizes:

Default size: For most use cases

Mini size: Only used to create visual continuity with other screen elements

The button size should change from the default (56dp) to the mini size (40dp) when the screen width is 460dp or less.

Floating action button

Mini floating action button

The floating action button should be placed 16dp minimum from the edge on mobile and 24dp minimum on tablet/desktop.

Floating action button

Interior icon: 24 x 24dp

Floating action button circle: 56 x 56dp

Mini floating action button

Interior icon: 24 x 24dp

Floating action button circle: 40 x 40dp

The floating action button changes color on focus and lifts upon being selected.

Floating action button gaining focus

Floating action button being selected

Not every screen needs a floating action button. A floating action button represents the primary action in an application.

Do.

The primary action is to touch images in a gallery, so no button is needed.

Do.

The primary action is to add files.

Only one floating action button is recommended per screen to increase its prominence. It should represent only the most common action.

Don't.

Don't.

Qualities

Make floating action buttons positive actions like Create, Favorite, Share, Navigate, and Explore.

Do.

Avoid using floating action buttons for minor and destructive actions, including the following:

Archive or Trash

Nonspecific actions

Alerts or errors

Limited tasks like cutting text

Controls that should be in a toolbar, like volume control or changing a font color

Floating action buttons don’t contain app bar icons or status bar notifications. Don’t layer badges or other elements over a floating action button.

Don't.

Use the circle-shaped icon consistently to enforce the primary action pattern across apps.

Do.

Don't.

Don’t give the floating action button extra dimension.

Do.

Don't.

Behavior

The floating action button animates onto the screen as an expanding piece of material, by default. The icon within it may be animated.

Floating action buttons may move differently than other UI elements because of their relative importance.

Examples of moving floating action buttons

Lateral screens

A floating action button that spans multiple lateral screens (such as tabbed screens) should briefly disappear, then reappear if its action changes.

The button should stay on screen if its action remains the same across screens (and translate to a new position, if necessary.)

Lists

Lists underneath floating action buttons should have enough padding beneath them so their content isn’t blocked by the button.

Do.

Do.

Tabbed screens

On tabbed screens, the floating action button should not exit the screen in the same direction as the content. This prevents:

Floating action buttons from appearing functional when they aren’t

The perception that the floating action button is at the same the elevation as the content

Don't.

Don't.

Transitions

Transitions

The floating action button is a unique example of a primary use case in an app. Take advantage of its visibility to create delightful transitions for a primary UI element.

Common transitions include Trigger, Toolbar, Speed dial, and Morphing. This is not an exhaustive list. Floating action buttons are designed to be flexible. Experiment with transitions that best suit your app and the screens on which the button sits.

Trigger

The floating action button can simply trigger an action or navigate somewhere. The animation of the touch ripple expands outwards as the force that causes changes to the UI.

A floating action button can simply trigger an action.

A floating action button recentering a map view

Apps typically use a single floating action button. However, this example uses two floating action buttons because they perform equally important yet distinct actions.

Toolbar

The floating action button can transform into a toolbar upon press or from a toolbar upon scroll. The toolbar can contain related actions, text and search fields, or any other items that would be useful at hand.

A floating action button transforming into a toolbar

A floating action button transforming into a toolbar.

A toolbar that disappears on scroll is particularly useful for screens where the full toolbar is needed upon initial entry or while at the top or bottom of a long list. This saves screen real estate when the user has signaled through scrolling that they’re interested in looking at the main content.

A toolbar transforming into a floating action button upon scroll

A toolbar transforming into a floating action button upon scroll

If a floating action button morphs into a toolbar, that toolbar should contain related actions.

Do.

In this example, the button lets the user select the media type to add.

Don't.

Don’t include unrelated or confusing actions.

Speed dial

The floating action button can fling out related actions upon press. The button should remain on screen after the menu is invoked. Tapping in the same spot should either activate the most commonly used action or close the open menu.

A floating action button flinging out related actions

A floating action button flinging out related actions

The floating action button can transform into a single sheet of material which contains all the actions.

A floating action button transforming into a single sheet of material

A floating action button transforming into a single sheet of material

As a general rule, have at least three options upon press but not more than six, including the original floating action button target. If you have two options – i.e. your floating action button only flings out one other choice – choose which action is most important. If you have more than six, users may have trouble reaching the furthest option.

Reduce decision fatigue by giving users the best, most distinct, and fewest options.

Don't.

A floating action button should fling out at least three options.

Don't.

A floating action button should fling out no more than six options.

Don’t put overflow menus in the floating action button actions. There should be at most two taps from the initial screen to get to the intended destination.

Do.

Options should be specific destinations.

Don't.

Options should not include an overflow menu.

Put overflow actions in the overflow menu in toolbars, not in floating action buttons.

Do.

Overflow menus belong in toolbars.

Do.

Overflow menus belong in toolbars.

Don't.

Floating action button are not overflow menus.

Don't.

Floating action buttons are not overflow menus.

If the hallmark of the app is adding file types, a floating action button can transform into related actions after it is first touched. However, if the actions that appear are unrelated to the button, place the actions into an overflow menu.

Do.

A floating action button can transform into related actions.

Don't.

A floating action button should not transform into unrelated actions.

A floating action button can contain a list of contacts. The list shouldn’t contain unrelated actions.

Do.

A floating action button can transform into a list of contacts.

Don't.

A floating action button should not transform into unrelated actions.

Morph

The floating action button can transform into sheets of material that are part of the app structure. This dramatic transformation accentuates the action the button enables.

When morphing the floating action button, transition between starting and ending positions in a logical way. For example, do not pass through other sheets of material.

The morph animation should be reversible and transform the new sheet of material back into the floating action button.

A floating action button can morph into a sheet of material that is part of the app structure.

A floating action button can morph into a sheet of material that is part of the app structure.

Full screen

The floating action button can transform into a new sheet of material that spans the entire screen.

This type of dramatic transformation is typically associated with creating new content. As a result, it does not tend to have an method of undoing the transformation or a reversible animation.

The floating action button can transform into a new sheet of material that spans the entire screen.

The floating action button can transform into a new sheet of material that spans the entire screen.

Large Screens

Large screens

A floating action button can attach to an extended app bar.

Do.

A floating action button can be attached to a toolbar or structural element within a sheet (as long as it’s not blocking other elements).

Do.

A floating action button can be attached to the edge of a sheet.

Do.

Don’t have more than one floating action button per screen.

Don't.

Don’t associate floating action buttons with every element on a screen.

Don't.

Cards

A card is a sheet of material that serves as an entry point to more detailed information.

Cards may contain a photo, text, and a link about a single subject. They may display content containing elements of varying size, such as photos with captions of variable length.

A card collection is a layout of cards on the same plane.

Usage

Cards display content composed of different elements whose size or supported actions vary.

Supported gestures

Swipe

Pick-up-and-move

Related components

Grid lists

For developers

Android cards

Polymer cards

Usage

Cards are a convenient means of displaying content composed of different elements. They’re also well-suited for showcasing elements whose size or supported actions vary, like photos with captions of variable length.

Example of a card

A card collection is coplanar, or a layout of cards on the same plane.

Example of a card collection

Example of a card collection

When to use

Use a card layout when displaying content that:

As a collection, comprises multiple data types, such as images, movies, and text

Does not require direct comparison (a user is not directly comparing images or text)

Supports content of highly variable length, such as comments

Contains interactive content, such as +1 buttons or comments

Would otherwise be in a grid list but needs to display more content to supplement the image

Do.

1. Cards have rounded corners.

2. Cards can have multiple actions.

3. Cards can be dismissible and rearranged.

Don't.

This is a tile, not a card.

1. Tiles have square corners.

2. Tiles have no more than two actions.

Do.

A quickly scannable list, instead of cards, is an appropriate way to represent homogeneous content that doesn't have many actions.

Don't.

The use of cards here distracts the user from being able to quickly scan. These list items are also not dismissable, so having them on separate cards is confusing.

Do.

Grid tiles are a clean and lightweight way to present a gallery of images.

Don't.

Cards are unnecessary in a gallery of images (homogeneous content).

Content

Cards provide context and an entry point to more robust information and views, and their content and quantity can vary greatly. Cards within a card collection can each contain a unique data set, such as a checklist with an action, a note with an action, and a note with a photo.

Don't overload cards with extraneous information or actions.

Content hierarchy

Use hierarchy within the card to direct users’ attention to the most important information. For example, place primary content at the top of the card, or use typography to emphasize primary content.

Images can reinforce other content in a card. However, their size and placement within the card depends on whether images are the primary content or are being used to supplement other content on the card.

Background images

Text is most legible when placed on a solid color background with a sufficient contrast ratio to the text. Text placed on image backgrounds should preserve text legibility.

This card collection contains cards with varied layouts.

This card collection contains cards with varied content types and layouts.

Typography can emphasize primary content.

This card collection contains cards with varied layouts and content hierarchy.

Behavior

Cards have a constant width and variable height. The maximum height is limited to the height of the available space on a platform, but it can temporarily expand (for example, to display a comment field).

Cards do not flip over to reveal information on the back.

Supported gestures

Card gestures should be consistently implemented within a card collection.

Supported gestures include:

The swipe gesture may be used on a per-card basis. Limit swipe gestures within a view so that they don’t overlap with one another. For example, a swipeable card should not contain a swipeable image carousel, so that only a single action occurs when the card is swiped.

The pick-up-and-move gesture may be used if it is important for the user to be able to sort cards within a collection. But consider if filtering or sorting would better organize the content.

Card collection filtering and sorting

Card collections can be programmatically sorted or filtered by date, file size, alphabetical order, or other parameters.

The first item in the collection is positioned at the top left.

The order proceeds left to right and top to bottom.

Card sorting from left to right, top to bottom.

Scrolling

Card collections only scroll vertically.

Card content that exceeds the maximum card height is truncated and does not scroll, but the card can be expanded. Once expanded, the card may exceed the maximum height of the view. In this case, the card will scroll with the card collection.

Card content that exceeds the maximum card height is truncated and does not scroll

Do.

Cards may be expanded to reveal more content, without using scrolling.

Don't.

On mobile, avoid placing scrollable space within a card, as it could cause two sets of scroll bars to be displayed, if one is already present.

On desktop, card content can expand and scroll internally

Card focus

When traversing through focus points on a card, all focusable elements are visited before moving to the next card.

For interfaces that depend on focus traversal for navigation (D-pad and keyboard), cards should have either a primary action or open a new view containing primary and supplemental actions.

Action selected during focus traversal

Expanded supporting text made visible, with the focus then placed on a supplemental action

Actions

The primary action in a card is typically the card itself.

Supplemental actions can vary from card to card in a collection, depending on the content type and expected outcome; for example, playing a movie versus opening a book. Within cards in a collection, position actions consistently.

Supplemental actions

Supplemental actions within the card are explicitly called out using icons, text, and UI controls, typically placed at the bottom of the card.

Limit supplemental actions to two actions, in addition to an overflow menu.

Card with action area, rich media and supporting text

Card with action areas 1 and 2, optional header, rich media, supporting text, and supplemental actions

Card with action areas 1, 2, 3 and 4, optional header, rich media, and supplemental actions

Card with action areas 1 and 2, optional header, supporting text and supplemental actions

UI controls

UI controls, like a slider, placed inline with primary content can modify the view of the primary content. For example, a slider to choose a day, stars to rate content, or a segmented button to select a date range.

This card contains UI controls within the action area block

This card contains segmented buttons within the action area block.

This card contains a slider control within the action area block.

This card contains tabs within the action area block.

Overflow menu (optional)

An overflow menu is typically placed in the upper-right corner of a card, but can be placed in the lower-right corner if doing so improves content layout and legibility.

Take care not to overload an overflow menu with too many actions.

This card contains an overflow menu in the top right.

Additional actions

Inline links within text content are strongly discouraged.

Although cards can support multiple actions, UI controls, and an overflow menu, use restraint and remember that cards are entry points to more complex and detailed information.

Do.

Use cards as an entry point to more detailed information.

Don't.

Cards provide context and an entry point to more robust information and views. Don't overload cards with extraneous information or actions. Inline links within text content are strongly discouraged.

Content blocks

Cards can be constructed using blocks of content which include:

An optional header

A primary title

Rich media

Supporting text

Actions

These blocks can be organized to promote different types of content. For example, numbers may be emphasized by increasing their typographic scale.

On tablet/desktop, cards should follow the 24dp keyline. See Metrics & Keylines for more information.

Primary title top padding: 24dp

Primary title bottom padding: 16dp

Action button row padding: 8dp

Supporting text top padding: 16dp

Supporting text bottom padding: 24dp

Content block types

Rich media

16:9 or 1:1 aspect ratio (recommended)

Actions

Padding: 8dp

Primary title/text

Title: 24sp or 14sp

Subtitle: 14sp

Left and right padding: 16dp on mobile

On tablet/desktop, cards should follow the 24dp keyline. See Metrics & Keylines for more information.

Top padding: 16dp or 24dp (when a large primary title is present)

Bottom padding: 16dp (if there are additional actions or supporting text) or 24dp (no actions or supporting text)

Supporting text

Supporting text: 14sp

Left and right padding: 16dp on mobile

On tablet/desktop, cards should follow the 24dp keyline. See Metrics & Keylines for more information.

Top padding: 16dp

Bottom padding: 24dp (16dp if there are additional actions or text below)

Bullet points (but not their text), images, and buttons may extend outside of the 16dp padding.

Card collections

Card margins on mobile

Padding from edge of screen to card: 8dp

Space between cards: 8dp

Responsive UI

Card gutters and margins may vary on larger screen sizes, as long as they follow Material Design 8dp grid. Card margins and gutters can be 8, 16, 24, or 40dp wide.

Margins and gutters don’t need to be equal. For example, both 40dp margins and 24dp gutters can be used in the same layout.

This animation shows interactions of the following margin and gutter width variations:

8dp margins and gutters

16dp margins and gutters

24dp margins and gutters

40dp margins and gutters

40dp margins and 24dp gutters

Elevation

Card resting elevation: 2dp

Card raised elevation: 8dp

Cards have a default elevation of 2dp.

On desktop, cards can have a resting elevation of 0dp and gain an elevation of 8dp on hover.

Card resting elevation of 2dp, and raised elevation of 8dp

On desktop, cards have a resting elevation of 0dp (left), and an elevation of 8dp on hover (right)

(desktop only)

Content block combinations

The following examples illustrate some possible combinations of content blocks.

Media area

16:9 ratio

Supporting text

Text: 14sp

Left and right padding: 16dp

Top padding: 16dp or 24dp (when a primary title is present)

Bottom padding: 16dp (if there are additional actions or supporting text) or 24dp (if there are no actions or supporting text)

Avatar, Title, and Subtitle area

Height: 72dp

Left and right padding: 16dp

Top and bottom padding: 16dp

Media area

16:9 ratio

Supporting text

Text: 14sp

Left and right padding: 16dp

Top and bottom padding: 16dp

Actions

Padding: 8dp

Avatar, Title, and Subtitle area

Height: 72dp

Left and right padding: 16dp

Top and bottom padding: 16dp

Media area

16:9 ratio

Actions

Padding: 8dp

Padding between actions: 4dp

Media area

16:9 ratio

Primary text

Text: 24sp

Top padding: 24dp

Bottom padding: 16dp

Right and left padding: 16dp

Subtext

Text: 14sp

Bottom padding: 16dp

Right and left padding: 16dp

Actions

Padding: 8dp

Media area

16:9 ratio

Primary text

Text: 24sp

Top padding: 24dp

Bottom padding: 16dp

Right and left padding: 16dp

Subtext

Text: 14sp

Bottom padding: 16dp

Right and left padding: 16dp

Actions

Padding: 8dp

Expanded supporting text

Text: 14sp

Top padding: 16dp

Bottom padding: 24dp

Right and left padding: 16dp

Primary text

Text: 24sp

Top padding: 24dp

Bottom padding: 16dp

Right and left padding: 16dp

Subtitle

Top padding: 12dp

Right and left padding: 16dp

Subtext

Text: 14sp

Bottom padding: 16dp

Right and left padding: 16dp

Supporting text

Text: 14sp

Top padding: 16dp

Bottom padding: 16dp

Right and left padding: 16dp

Actions

Padding: 8dp

Media area

16:9 ratio

Actions

Padding: 8dp

Media area

1:1 ratio

Primary text

Text: 24sp

Top padding: 24dp

Right and left padding: 16dp

Subtext

Text: 14sp

Bottom padding: 16dp

Right and left padding: 16dp

Actions

Padding: 8dp

Media area

1:1 ratio

Primary text

Text: 24sp

Actions

Padding: 8dp

Media area

80dp

1x

Top padding: 16dp

Primary text

Text: 24sp

Top padding: 24dp

Right and left padding: 16dp

Subtext

Text: 14sp

Right and left padding: 16dp

Actions

Padding: 8dp

Media area

1.5x (Increment size based on 80dp media area)

Top padding: 16dp

Primary text

Text: 24sp

Top padding: 24dp

Right and left padding: 16dp

Subtext

Text: 14sp

Right and left padding: 16dp

Actions

Padding: 8dp

Media area

2x (Increment size based on 80dp media area)

Top padding: 16dp

Primary text

Text: 24sp

Top padding: 24dp

Right and left padding: 16dp

Subtext

Text: 14sp

Right and left padding: 16dp

Actions

Padding: 8dp

Media area

3x (Increment size based on 80dp media area)

Padding: 16dp

Actions

All around padding: 8dp + 16dp

Dividers in cards

Dividers may be used to separate content areas in cards that require distinct visual separation. Dividers may also indicate seams in places where material may expand.

Content areas that can be expanded should use full-width dividers. Dividers can indicate seams in material where the material will expand when content is expanded.

This example uses a full-bleed divider. Similar to expansion of lists, this card uses a full-width divider to denote expansion.

Content areas that need more distinct visual separation should use dividers.

This example uses a full-bleed divider as a way to visually separate the slider content and list from the action below.

Use inset dividers to separate related content.

This example uses an inset divider to separate related the restaurant information from the booking section.

Chips

Chips represent complex entities in small blocks, such as a contact.

A chip may contain entities such as a photo, text, rules, an icon, or a contact.

Contact chips may represent contact information in a compact way.

Behavior

Selecting a chip opens a full detail view.

Chips may be deleted if they display a delete icon.

Specs

Height: 32dp

Label: 13sp Roboto Regular, 87% black

Usage

A chip may contain a photo, short title, and brief information.

Types of chips

Chips can be used for various types of entities, including free form text, predefined text, rules, or contacts. Chips may also contain icons.

Chips with text

Chip label

13sp Roboto Regular

87% black

Chips with text and an icon

Behavior

Opening chips

Touching a chip opens a full detailed view (either in a card or full screen) or a menu of options related to that chip.

Deleting chips

Chips can be deletable or non-deletable. Display a delete icon if a chip is deletable.

Users may press the delete icon or use the keyboard “Delete” button to remove a chip.

Contact chips

Contact information that users have for people may be represented in a compact way using contact chips. The chips are invoked and inserted into a text field (usually the “To” field) when the user starts typing a contact’s name, sees the contact’s addresses, and selects the correct one. Contact chips can be added directly to a text field from a menu of contacts.

Contact chips efficiently confirm that the user will be sending their message to the correct person.

The user confirms the name(s) of the email recipient(s) on the contact chips.

The user confirms which email address to select on the contact chip.

The contact chip has different states: Normal, Focused, Pressed, and Activated.

Specs

Single-line chips

Deletable chip

Height: 32dp

Label left padding: 12dp

Remove icon

Size: 24dp x 24dp

Color & opacity: 54% black

Margin: 4dp

On desktop, the “Remove” icon is persistent.

Non-deletable chip

Height: 32dp

Label left and right padding: 12dp

Contact chip

Height: 32dp

Label left padding: 8dp

Label right padding: 12dp

Contact chips

Closed contact chip

The contact name text is Roboto Regular 14sp.

Upon focus, the chip rises to an elevation of 2dp. When pressed, it expands to show alternative addresses for the contact.

Open contact chip

Contact name text: Roboto Regular 16sp

Address text: Roboto Regular 14sp

Elevation of the open contact chip is 8dp.

On press, the contact chip closes automatically.

By default, the top field is activated and focused.

Contact chip

Height: 32dp

Title top, bottom, and left padding: 8dp

Title right padding: 12dp

Focused and activated contact chip

Height: 40dp

Top and bottom padding: 16dp

Unfocused contact chips

Top and bottom padding: 20dp

Data tables

Data tables display sets of raw data. They usually appear in desktop enterprise products.

Data tables may be embedded on a surface, such as a card. They can include:

A corresponding visualization

The ability to query and manipulate data

Data tables on cards may display navigation and data manipulation tools at the top and bottom.

Supported interactions

Row hover (desktop)

Row selection

Column sorting

Column hover (desktop)

Long header titles

Text editing

Menus

Related components

Cards

Structure

A data table contains a header row at the top that lists column names, followed by rows for data.

Checkboxes should accompany each row if the user needs to select or manipulate data.

Data tables may include:

Three or more columns of data

A corresponding visualization

The ability for users to query and manipulate data

Basic data table

Table content

13sp Roboto Regular

87% black

Column headers

12sp Roboto Medium

54% black

Text alignment

Right-aligned numeric columns

Left-aligned text columns

Interaction

Row hover (Desktop)

Display a background in a table row if a user hovers over any part of that row. If there are separate hover states on individual table cells, display both the hover state of the cell and row at the same time.

Hover background

Grey 200 (#EEEEEE)

Hovering over a row

Row selection

When a row is selected, display a background color on the row.

Selected row background

Grey 100 (#F5F5F5)

Checkbox

Use secondary app color

A selected table row

Sorted column

If column sorting is enabled, sort the most important data by default and display a sorted state in the column header. If the user clicks on a column that is already sorted, reverse the sort order and rotate the sort icon.

Sorted column name

12sp Roboto Medium

87% black

Sort icon

16dp container

87% black

Table with an ascending sorted column

Column name hover (Desktop)

If there is a need to provide definitions to column headers, display a tooltip on hover. If sorting is enabled, display a light sort icon upon hover, which indicates that the column is sortable.

Tooltip

Follow the guidelines for Tooltips

Sort icon

16dp container

38% black

Hovering over a column name

Long header titles

Sometimes, column names don’t fit in a container with the recommended 56dp of padding in between columns. There are two options to handle this:

Display the full column name and enable horizontal scrolling in the table container.

Shorten the column name and display it in full on hover.

Long column names truncated with an ellipse

Hovering over a truncated column name

Inline text editing

Tables may require basic text editing (e.g. for editing existing text content, or adding comments). Include editable fields within a table and denote them using placeholder text. You can use a simple edit dialog with just a text field, or display a full dialog component on click.

Placeholder text

13sp Roboto Regular

38% black

No divider line on text field

An editable table cell with placeholder text

Small edit dialog

Align edges of dialog with nearest divider lines, or table edges

24dp of left and right padding

Include a single text field, applying the app theme where applicable

User confirms text by pressing Enter

A simple, one-field edit dialog

Large edit dialog

Align edges of dialog with nearest divider lines, or table edges

Follow the dialog guidelines, applying the app theme where applicable

User confirms text by clicking the Save button

A complex, flexible edit dialog

Inline edit icon

18dp container

54% black

(Optional) Icon-based edit affordance. Include icons at the edge of the cell on the opposite side of the content.

Inline menus

Instead of editing text, users may need to select from a predefined list of options. In this scenario, embed a menu component directly in the table.

Inline menu

13sp Roboto Regular

87% black

Follow spacing and sizing guidelines for menus

A table with inline menus

An expanded inline menu

Tables within cards

Tables may be embedded within a card, with table navigation and data manipulation tools displayed at the top and bottom.

Table card with header and footer

Card title

20sp Roboto Regular

87% black

Card action icons (header and footer)

24dp container

54% black

Footer pagination labels

12sp Roboto Regular

54% black

Alternative headers

Some table cards may require headers with actions instead of titles. In this case you may either:

Display persistent actions

Display a contextual header that activates when items are selected

Alternate headers for a table card

Action buttons

Use borderless buttons

Enable upon item selection

Colored header

Use 50-value of secondary app color for background fill

Display the number of items selected

Display available contextual icons upon item selection

Specs

Vertical spacing

Vertical spacing of a data table card and its last row

64dp card header height

56dp height for last row

48dp data row height

Header & footer padding

Horizontal padding between elements in a table card

24dp of padding around the perimeter of table cards

32dp in between footer control sets

40dp in between the footer row count and its label

Column padding

Padding in between columns

Use a minimum of 56dp of padding in between columns. The widest item in the column (including data and column name) should delineate the column border.

Checkbox icons have a width and height of 18dp within a 24dp icon container.

Dialogs

Dialogs inform users about a specific task and may contain critical information, require decisions, or involve multiple tasks.

Dialogs contain text and UI controls. They retain focus until dismissed or a required action has been taken. Use dialogs sparingly because they are interruptive.

Some dialog types include:

Alerts are urgent interruptions that inform about a situation and require acknowledgement.

Simple menus display options for list items, whereas simple dialogs can provide details or actions about a list item.

Confirmation dialogs require users to explicitly confirm a choice.

Behavior

Dialogs should never be obscured, either by other elements or the screen edge. Dialogs always retain focus until dismissed or a required action has been taken.

Full-screen dialogs (Mobile only)

Full-screen dialogs are best suited to complex tasks, or require an input method editor, as they group a series of tasks together before they can be saved.

Behavior

Beyond standard dialogs

Dialogs are a sub-type of modal windows, and the examples covered here are for standard material system dialogs. (Other modal window constructions aren’t covered here because they have too much variation, such as branded buttons for purchasing flows, non-standard UI form elements, or unique layouts.)

Reduce interruption

Use dialogs sparingly because they are interruptive. Their sudden appearance forces users to stop their current task and focus on the dialog content. Not every choice, setting, or detail warrants interruption. Alternatives to dialogs include menus or inline expansion, both of which maintain the current context.

Dialog prominence

Dialogs should never be obscured by other elements or appear partially on screen. Dialogs always retain focus until dismissed or a required action has been taken, such as choosing a setting.

Dialogs should avoid:

Opening dialogs from within a dialog

Containing scrolling content

Full-screen dialog exception

Full-screen dialogs may open additional dialogs, such as pickers, because their design accommodates additional layers of material without significantly increasing the app’s perceived z-depth or visual noise.

Example of dialog content

Example of a full-screen dialog

Scrollable content exception

Some dialog content requires scrolling, such as lists of ringtones.

For scrollable lists of options, the dialog title remains pinned to the top. This ensures that a selected item remains visible with the title, regardless of the item’s position in the list.

Otherwise, the title scrolls off with the content. Actions always remain in place when content scrolls.

Dialogs are separate from the underlying parent material and do not scroll with it.

Displaying additional content

To disclose additional content in a dialog, do so using inline expansion within the content area. Or consider alternative components that are optimized for large amounts of content.

Dismissing dialogs

Dialogs may be dismissed either by tapping outside of the dialog, or tapping the system back button (on Android). Alternatively, the user’s ability to dismiss a dialog may be disabled, so that one of the actions must be chosen before proceeding.

Make the dialog title fixed when viewing a scrollable list of options ensures that both the title and the selected item are simultaneously visible.

Alerts

Alerts are urgent interruptions, requiring acknowledgement, that inform the user about a situation.

Disambiguation from Snackbars: Snackbars present optional information after an action, such as confirming the discarding of a draft. They often allow a user to undo an action just taken.

Alerts without title bars

Most alerts don't need titles.

They summarize a decision in a sentence or two by either:

Asking a question (e.g. "Delete this conversation?")

Making a statement related to the action buttons

Do.

The affirmative action text “Discard” clearly indicates the outcome of the decision.

Don't.

The dismissive action text “No” answers the question, but does not suggest what will happen afterwards. A better action pair would be an explicit “Cancel” and “Delete.”

Alerts with title bars

Use title bar alerts only for high-risk situations, such as the potential loss of connectivity. Users should be able to understand the choices based on the title and button text alone.

If a title is required:

Use a clear question or statement with an explanation in the content area, such as "Erase USB storage?".

Avoid apologies, ambiguity, or questions, such as “Warning!” or “Are you sure?”

Do.

This dialog poses a specific question, concisely elaborates on its impact, and provides clear actions.

Don't.

This dialog poses an ambiguous question and its scope of impact is unclear.

Simple menus

Mobile and tablet only

Simple menus display options for list items, and they immediately commit choices upon selection. See Simple Menus for more details.

Disambiguation: Simple dialogs display detailed options for list items or provide related actions. Simple dialogs can display the same content as simple menus. However, simple menus are preferred because they are less disruptive to the user’s current context.

Examples of a simple menu

Simple dialogs

Simple dialogs can provide additional details or actions about a list item. For example, they can display avatars, icons, clarifying subtext, or orthogonal actions (such as adding an account).

Touch mechanics:

Choosing an option immediately commits the option and closes the menu

Touching outside of the dialog, or pressing Back, cancels the action and closes the dialog

Reduce interruption

Simple dialogs are more interruptive than simple menus and should be used sparingly.

Example of a simple dialog

On mobile, a dialog’s width is based on a multiple of a unit. That unit can be either:

An increment

A specific distance from the screen edge

No explicit cancel button

Simple dialogs do not have explicit buttons that accept or cancel an operation.

Don't.

This simple dialog has an explicit “Cancel” button.

Don't.

This simple dialog has an explicit “Cancel” button.

Specifications

Row heights can vary in simple dialogs.

Simple dialogs are displayed centered vertically and horizontally on the screen.

The dialog's distance from the screen edge is at least 40dp on the left and right, and at least 24dp on the top and bottom.

The dialog's content is 24dp from the dialog edge.

Avoid text wrapping

If any text in a simple menu wraps to another line, use a simple dialog instead.

Do.

This simple dialog has varying row heights.

Confirmation dialogs

Confirmation dialogs require users to explicitly confirm their choice before an option is committed. For example, users can listen to multiple ringtones but only make a final selection upon touching “OK.”

Tapping “Cancel” in a confirmation dialog, or pressing “Back,” cancels the action, discards any changes, and closes the dialog.

Avoid dialogs launching dialogs

Confirmation dialogs should avoid launching additional simple dialogs or simple menus, as they add complexity and appear to increase an app’s elevation. If they are needed to complete a task, consider using a full-screen dialog instead.

The ringtone choice in the following confirmation dialog will not be set until the user taps “OK.”

Example of a confirmation dialog with controls positioned on the left side of text.

Confirm a single value

Confirmation dialogs can use layouts other than lists, such as a date picker, but remain focused on specifying a single value (picking the date, but not picking the time and date).

The date choice is set by the user tapping a date and the “OK” button.

The user selects the hour by moving the clock hand and tapping “OK.”

Cancel and confirmation buttons

Confirmation dialogs provide both an explicit confirmation button and explicit cancel button. Tapping the cancel button, Back button, or leaving the confirmation dialog will discard changes.

Do.

Provide explicit confirmation and cancel buttons.

Don't.

A single dialog button makes the system Back action ambiguous: does Back cancel or confirm?

Full-screen dialogs

Mobile only: Due to limited space, full-screen dialogs may be more appropriate for mobile devices than dialogs used on devices with larger screens.

Usage

Full-screen dialogs group a series of tasks (such as creating a calendar entry) before they may be committed or discarded. No selections are saved until “Save” is touched. Touching the “X” discards all changes and exits the dialog.

Full-screen dialogs enable complex layouts, minimize the appearance of stacked sheets of material (dialogs above dialogs), and temporarily reset the app’s perceived elevation to a higher baseline. They allow tasks to launch simple menus or simple dialogs as part of a complex operation.

Full-screen dialogs may be used for content or tasks that meet any of these criteria:

The dialog includes components (like pickers or form fields) that require an input method editor (IME), such as a keyboard.

When changes are not saved in real time

When there is no draft capability in the app

When performing batch operations or queuing changes prior to submitting them

The full-screen dialog supports a simple dialog used to pick dates.

Date picker opened from full-screen dialog

Actions

Place confirmation and dismissive actions for full-screen dialogs at the top of the screen.

Confirmation

The confirmation action in the top right of the screen uses descriptive verbs, such as: save, send, share, update or create. Don’t use vague actions for confirming action, such as: done, ok or close.

The confirmation action is disabled until all mandatory fields in the dialog are met.

Discard

Both the discard action (the “X” at the top left of the screen) and the Back button close the full-screen dialog and discard changes.

If no changes have been made, the dialog closes and no discard confirmation is required.

If the user has made any changes, they are prompted to confirm the discard action.

Don't.

Don’t use vague terms like “Close” for confirmation actions.

Do.

Prompt users to confirm the discard action if they have made any changes.

Navigation

The “X” used in a full-screen dialog differs from an up arrow, which indicates the view’s state is constantly being saved. For example, an up arrow used in Settings indicates all changes are committed immediately without explicit confirmation or cancel actions.

The up arrow in this Settings example indicates that any changes will be immediately saved upon selection.

Touching the “X” in this Settings example will discard all changes. Changes will be saved only upon touching Save.

Titles

Full-screen dialog titles don’t use dynamic type.

Titles should be succinct. They can wrap to a second line if necessary, and should then be truncated.

If the full-screen dialog uses titles of variable length or anticipates long titles (for example, because certain words are longer in different languages), place title text in the content area of the dialog instead of the app bar.

Don't.

Avoid using titles of variable length in the app bar.

Do.

Place long titles in the content area of the full-screen dialog.

Specs

Dialogs contain content, actions, and an optional title.

Optional title

The title briefly describes the type of choice being made. Titles should always be displayed in their entirety and only used when necessary. For example, a title may indicate to which part of a task the dialog relates, or identify what will be affected by the decision.

Content

Dialog content typically consists of text and/or UI control elements. It is focused on a specific task, such as confirming item deletion or choosing a setting.

This dialog contains a title, content, and actions.

Actions

Dialogs present a focused and limited set of actions, which are generally affirmative or dismissive.

Affirmative actions are placed on the right side and continue the process. Affirmative actions may be destructive, like “Delete” or “Remove.”

Dismissive actions are placed directly to the left of affirmative actions and return the user to the originating screen or step in the process.

Dismissive and affirmative action text can be “Cancel”/”OK” or specific active verbs or verb phrases that indicate the outcome of the decision.

Don't.

Dismissive actions are always placed directly to the left of affirmative actions.

Dialog actions should present a clear choice directly related to the dialog’s title and content.

Don't.

Avoid presenting users with ambiguous or unclear choices. In this example, “Cancel” doesn't make sense in relation to the title because there's no clear action being proposed.

Acknowledgement actions

In situations where users are required to acknowledge the dialog’s content to proceed, an alert may contain only one action. Use this type of alert sparingly as it is interruptive. Consider other methods of communicating the information to users, such as in-line with a view’s content.

Number of actions

Dialogs should not include more than two actions. A third action, such as “Learn more,” navigates away from the dialog, potentially leaving the task unfinished.

Avoid using a “Learn more” action to access help documentation; in-line expansion within the dialog should be used instead. If more extensive information is needed, provide it prior to entering the dialog.

Don't.

The “Learn more” action navigates away from this dialog, leaving it in an indeterminate state.

Color

Dialog actions use system colors by default, but they should reflect your product's color palette. Use a contrasting color, such as the palette’s accent color, to distinguish dialog actions from dialog content.

Languages without capitalization

For languages without capitalization (such as Chinese, Japanese or Korean), it is important to maintain consistent placement, spacing, and colors for actions to distinguish them from regular text.

The consistent placement of actions and text color helps distinguish actions from regular text even when the affirmative action is disabled.

Affirmative actions are more likely to be disabled until a choice is made. Dismissive actions are never disabled.

Content guidelines

Padding around content area: 24dp

Padding between title and body text: 20dp

Padding around buttons: 8dp

Button height: 36dp

Action area height: 52dp

Dialog elevation: 24dp

Content padding

Within the content area, the 24dp of padding below the content helps separate it from the actions.

Dialog content bottom padding: 24dp

Button height: 36dp

Button margin: 8dp

Content padding for a dialog in a scrolled state

Button width and padding

Button height: 36dp

Minimum button width: 64dp

Internal button padding: 8dp

Padding between buttons: 8dp

Detail of button widths and padding

Button height: 36dp

Button area height: 52dp

Left button padding: 24dp

Right button padding: 8dp

Padding between buttons: 8dp

Detail of button area

In a scrolled state, a stroke delineates the dialog’s content area from actions.

Side-by-side buttons

Side-by-side buttons are recommended when the text of each label does not exceed the maximum button width, such as the commonly used OK/Cancel buttons.

Use the following formula to calculate maximum button width for a given dialog:

The maximum width for buttons in a dialog =

(Dialog width - 8dp - 8dp - 8dp)/2

For example:

The maximum width for buttons in a 280dp wide dialog =

(280dp - 8dp - 8dp - 8dp)/2 = 128dp

Button height: 36dp

Padding between text and action area: 24dp

Padding around buttons is: 8dp

Stacked full-width buttons

When text labels exceed the maximum button width, use stacked buttons to accommodate the text. Affirmative actions are stacked above dismissive actions.

Touchable target height for each button: 48dp

Padding between text and touch target: 24dp

Padding below touch target to dialog edge: 8dp

Padding between button text right edge and dialog edge: 16dp

Simple dialog keylines

Vertical keyline at 24dp from the left and right edges. Content associated with an icon or avatar aligns 80dp from the left edge.

Keylines for a simple dialog

Simple dialog content guidelines

It is recommended that simple dialogs have titles, but titles can be omitted based on your product’s needs.

Title

Top padding: 24dp

Bottom padding: 20dp

Text size: Roboto Medium 20sp

Text line height: 28dp

Content

Row height of single-line list with avatars: 56dp

Bottom edge padding: 8dp

redlines for a simple dialog

Full-screen dialog titles

Full-screen dialog titles can wrap to a second line if necessary, and then should be truncated. Titles should be succinct, but in some situations, such as when words are longer in different languages, titles may need to wrap.

App bar height with a single line of text: 56dp

App bar height with two lines of text: 80dp

Title text keyline: 72dp

Title text: 20sp

Title text top and bottom padding: 20dp

Dismissive action padding from left edge: 16dp

Affirmative action text: 14sp

Affirmative action text padding on the left and right: 16dp

Detail of a full-screen dialog app bar

Full-screen dialog with an app bar containing a single line of text.

Note that this image is for illustration purposes only. Long titles should be placed in the content area of the full-screen dialog.

Dividers

A divider is a thin, lightweight rule that groups content in lists and page layouts.

Dividers help organize page content and hierarchy into individual tiles.

Full-bleed dividers emphasize separate content areas and sections that require more distinct visual separation. Alternatives to full-bleed dividers include white space, subheaders, or inset dividers.

Image-based content in grid lists does not need dividers. Grid lists adequately separate content using white space and subheaders.

Usage

For lists and to separate content

Type of dividers

Full-bleed dividers are used to separate distinct content sections.

Inset dividers are used to separate related content.

Specs

Thickness: 1dp

Opacity: 12% black or 12% white

Placement: Along the bottom edge of content tiles

Usage

Dividers help users understand how content is organized by establishing a rhythm and hierarchy on a page. But a heavy use of dividers can lead to visual noise and dilute their impact.

Full-bleed dividers emphasize separate content areas and sections, but if such strong divisions are not required, consider using white space, subheaders, or inset dividers.

Items without anchors

When lists don’t have an anchoring element such as an avatar or icon, spacing alone isn’t always enough to separate tiles. In this case, full-bleed dividers can help create rhythm and separate individual tiles.

Full-bleed dividers separate email summaries.

Full-bleed dividers separate the From, To, Subject, and Message sections of the email screen.

Image-based content

Because the grid itself creates visual distinction, grid lists do not need dividers to separate subheaders from content. In this case, the white space and the subheaders separate the sections adequately.

The white space and the subheaders separate the photos and albums.

Do.

Judicious use of inset dividers demarcates major sections of content.

Do.

Use inset dividers in conjunction with anchoring elements such as avatars.

Don't.

Overuse of full-bleed dividers creates visual noise and ultimately diminishes the meaning of the dividers.

Don't.

Inset dividers help cut down on visual noise and should be used with anchor elements, but overuse reduces their meaning.

Types of dividers

Full-bleed dividers

Full-bleed dividers separate distinct content sections (e.g., biographic details from contact information) or distinct content elements (e.g., list items) in both lists and page layouts.

Full-bleed dividers can also indicate seams in material where the material will expand when content is expanded.

Example of full-bleed divider

Example of full-bleed divider

Inset dividers

Inset dividers separate related content, such as sections in a list of contacts or emails in a conversation.

Inset dividers should be used in conjunction with anchoring elements such as icons or avatars aligned with the Title Key Line.

Example of inset divider

Example of inset divider

Subheaders and dividers

When using a divider with a subheader, place the divider above the subheader to reinforce the relationship between the subheader and the content.

Example of divider above subheader

Example of divider above subheader

Specs

Dividers are 1dp thick, with an opacity of either 12% black in light themes or 12% white in dark themes.

Dividers are placed along the bottom edge of the content tiles, independent of the grid.

The divider sits within the baseline of the tile.

Expansion panels

Expansion panels contain creation flows and allow lightweight editing of an element.

An expansion panel is a lightweight container that may either stand alone or be connected to a larger surface, such as a card.

They may be used for a variety of tasks, such as:

To edit a setting

To create a tool for ad campaigns

Usage

Editing

Creating flows

Behavior

Expand

Collapse

Focus

Contents

Usage

Behavior

Specs

Usage

An expansion panel is a lightweight container that may either stand alone or be connected to a larger surface, such as a card.

Editing

Expansion panels are best used for lightweight editing of an element, such as selecting a value for a setting.

For example, an app that contains tables may use expansion panels to adjust settings on table elements.

Creation flows

Expansion panels may be displayed in a sequence to form creation flows.

For example, a tool for creating ad campaigns may display a series of expansion panels to collect campaign details.

Various expansion panels

List content

15sp Roboto Regular

87% black

Secondary content

12sp Roboto Regular

54% black

Divider

1dp border

12% black

Expand icon

24dp x 24dp

38% black

Behavior

Collapsed panel

A collapsed panel displays summary information. Upon selecting the collapsed panel, it expands to display the full expansion panel.

Example of a collapsed panel

Expanded panel

Upon selection, a collapsed panel expands, allowing users to add or edit information.

Helper text may be added to the panel to assist the user.

Example of an extended panel

Focus state

An expansion panel may use a focus state (by using a grey background on the list item) to provide focus to individual list items.

On a keyboard, users may traverse through list items by using the Tab key.

Background

Grey 200

Expand icon on hover and pressed

24dp x 24dp

54% black

Specs

Label

Font: Roboto Regular 15sp

Padding

Top and bottom padding: 16dp

Left and right padding: 24dp

Grid lists

Grid lists are an alternative to standard list views.

A grid list consists of a repeated pattern of cells arrayed in a vertical and horizontal layout.

Grid lists are best used on similar data types. They help improve the visual comprehension of the content they contain.

Types

Image-only

Single-line text (or with icon)

Two-line text (or with icon)

Actions

Vertical scrolling

Filtering

Alternatives

Lists

Cards

Usage

A grid list is best suited to presenting homogenous data, typically images, and is optimized for visual comprehension and differentiating between similar data types.

Example of grid list

A grid list consists of a repeated pattern of cells arrayed vertically and horizontally within the grid list.

Tiles hold content and can span one or more cells vertically or horizontally.

Example of grid list

Example of a cell and tile

If the text in tiles needs to be prominent enough to distinguish between primary content pieces, consider using a different container, like a list or cards, optimized for displaying text and facilitating reading comprehension.

Lists: Optimized for reading comprehension, particularly when comparing a set of data containing multiple data types.

Cards: Used for content with inconsistent formatting, such as photos with captions of variable length, or data sets with heterogeneous content, such as a mixed collection of photos and videos and books.

Content

Content in tiles

Tile content consists of primary content and secondary content. Primary content is the main differentiating element, typically an image. Secondary content can be an action or text.

Provide a default image for tiles that lack an image for primary content.

Example of tile content with primary and secondary actions

Actions in tiles

Actions on both primary and secondary content – such as play, zoom in, delete, or select – are immediate and typically do not result in a submenu of options (action overflow) within the grid list.

Actions can open a subsequent view, such as a card.

Primary actions

Fill the entire tile and therefore are not represented via icons or text.

Are consistent throughout tiles in a specific grid list. For example, the primary action for all tiles in a single grid list could be to view details for an image.

Secondary actions or content

Are represented in tiles with icons or text.

Are consistent throughout tiles in a specific grid list.

Are placed in a consistent location within the tiles of a specific grid list, but that consistent location may vary between grid lists (at corners or edges). For example, all titles in one grid list could be located at the bottom left corner, while all tiles in another grid list might place titles in the top left corner.

Example of secondary action with action and text positions

Behavior

Scrolling

Grid lists typically scroll only vertically.

Horizontally scrolling grid lists are discouraged because the scrolling interferes with typical reading patterns, affecting comprehension. One notable exception is a horizontally-scrolling, single-line grid list of images, such as a gallery, which is compatible with typical reading patterns.

Cut off grid tiles in the view’s initial scroll position to communicate the scroll direction for content overflow.

Do.

Indicate content overflow by cutting off grid tiles.

Don't.

Avoid aligning grid tiles along a view edge in the view’s initial scroll position. This positioning doesn’t effectively communicate that there is more content available.

Gestures

Per-tile swipe actions are not permitted. Pick-up-and-move actions are discouraged.

Tile filtering and sorting

Content in a grid list can be programmatically filtered or sorted by date, file size, alphabetical order, or other parameters.

The first item in the grid list is positioned at the top left of the grid list, and the order proceeds left to right and top to bottom.

Dimensions and resizing

Resizing a grid list causes the tiles to re-sort as horizontal space becomes available. Full-bleed grid list tiles resize to fit the screen width.

A grid list does not transform into a list when horizontal space contracts. Grid lists and lists are separate structures for emphasizing different data types. Grid lists prioritize images over text and lists prioritize text over images.

To maintain a consistent rhythm throughout the grid list, truncate text content that is too long for the tile’s width. Alternatively, increase the grid size so that the tiles can accommodate the longer titles.

Responsive design

Full-screen grid lists should use fluid image ratios with minimum and maximum widths, derived using the Ratio keylines. They should retain fixed heights, margins, and padding.

Centered grid lists have fluid margins with a minimum width. They maintain fixed image widths, heights, and padding.

Specs

Grid list header/footers

Single-line header/footer

Height: 48dp

Text padding: 16dp

Default font size: 16sp

Secondary action is flush right to the footer.

Two-line header/footer

Height: 68dp

Text padding: 16dp

Default font size for each line: 16sp/12sp or 14sp/14sp

Type 14/14

Two-line bar (with icon) height: 68dp

Two-line bar height: 68dp

Single-line bar height: 48dp

Type 16/12

Two-line bar (with icon) height: 68dp

Two-line bar height: 68dp

Single-line bar height: 48dp

Text truncation in header/footers.

Image-only grid list

Grid list padding: 4dp or 1dp

Tiles in grid lists can span multiple columns.

Carefully consider whether secondary text is needed in grid lists that use multi-column tiles, as larger tiles can develop significant empty space.

Image-only grid list

Image-only grid list photo album

Single-line grid list

Text only

Height: 48dp

Text padding: 16dp

Default font size: 16sp

Grid list padding: 4dp or 1dp

Single-line grid list footer with 16dp padding

Single-line grid list header with 16dp padding

Single-line grid list photo album

Text with icon

Height: 48dp

Text padding: 16dp

Default font size: 16sp

Grid list padding: 4dp or 1dp

The secondary action can be flush right or flush left within the footer or header.

Single-line grid list footer with 16dp padding and icon

Single-line grid list header with 16dp padding and icon

Single-line grid list footer with icon in photo album

Single-line grid list footer with icon

Single-line grid list header with icon

Two-line grid list

Text only

Height: 68dp

Text padding: 16dp

Default font size for each line: 16sp/12sp or 14sp/14sp

Grid list padding: 4dp or 1dp

Two-line grid list footer with 16dp of padding

Two-line grid list, 4dp example

Two-line grid list header with 16dp of padding

Two-line grid list, 1dp example

Text with icon

Height: 68dp

Text padding: 16dp

Default font size for each line: 16sp/12sp or 14sp/14sp

The secondary action can be flush right or flush left within the footer or header.

Grid list padding: 4dp or 1dp

Two-line text with icon footer

Text padding: 16dp

Two-line text with icon header

Text padding: 16dp

Example of text with icon

Lists

Lists present multiple line items vertically as a single continuous element.

Lists are made up of a continuous column of rows. Each row contains a tile. Primary actions fill the tile, and supplemental actions are represented by icons and text.

Lists are best suited for similar data types.

Actions

Vertical scrolling

Filtering and sorting

Gestures

Swipe

Drag and drop

Alternatives

Cards

Grid lists

Usage

A list consists of a single continuous column of tessellated sub-divisions of equal width called rows that function as containers for tiles.

Tiles hold content, and can vary in height within a list.

List

Row and tile

Lists are best suited to presenting a homogeneous data type or sets of data types, such as images and text. They are optimized for reading comprehension while differentiating either between similar data types, or qualities within a single data type.

List alternatives:

If more than three lines of text need to be shown in list tiles, use cards instead.

If the primary distinguishing content consists of images, use a grid list.

Density

When the mouse and keyboard are the primary input methods, measurements may be condensed to accommodate denser layouts.

Typical list scanning pattern

Behavior

Scrolling

Lists scroll only vertically.

Gestures

The swipe action of each tile should be consistent within lists.

Tiles may be moved between a list and drop target (similar to moving a file into a folder) and picked up and manually reordered within a list.

Filtering and sorting

List tiles can be sorted or filtered by date, file size, alphabetical order, or other parameters.

Do.

Alphabetical sorting makes information easier to find.

Don't.

Don’t display information in random order; if the UI doesn’t allow the user to sort a list, your app should use a logical sort order.

Content

Collections of list tiles present related content in a consistent format. Lists use hierarchy to prioritize a type or set of content, which helps users find the most important information. For example, in an email inbox app, an avatar and text snippet are emphasized over a time stamp.

List tiles may contain up to three lines of text, and text length may vary between tiles in the same list. To display more than three lines of text, use a card.

Place the most distinguishing content on the left of the tile and the least distinguishing content on the right.

Specifications:

The majority of space on a list tile should be dedicated to the primary action.

Place the most distinguishing content towards the left of the tile.

In tiles with multi-line content, place the most distinguishing content in the first line.

Place supplemental actions on the right side.

The primary action takes up the majority of the space on the tile. The supplemental action goes on the right side of the tile.

Actions

Primary and supplemental actions (such as play, zoom in, delete, and select) may open a subsequent view, such as a card. They do not have a submenu or an action overflow menu.

Primary actions

Fill the entire tile and are not represented solely by an individual icon or text button

Are consistent for all tiles in a specific list, such as the primary action in a list of music to play songs

Supplemental actions

Are represented by icons, secondary text, etc

Have consistent functionality throughout a list, such as an icon that indicates if someone is online

Have a consistent location in a list’s tiles

Repeating actions

Avoid displaying a repeated supplemental action in tiles, such as a share action in every tile.

Toggles, such as stars or pins, are exceptions because they provide unique information about each individual tile.

Specs

In a single-line list, each tile contains a single line of text. The amount of text can vary between tiles within the same list.

Single-line list specs

Icon left padding: 16dp

List item left padding: 72dp

List item right padding: 16dp

Example of a single-line list

In a two-line list, each tile contains a maximum of two lines of text. The amount of text can vary between tiles within the same list.

Two-line list specs

Icon left padding: 16dp

List item left padding: 72dp

List right padding: 16dp

Example of a two-line list

In a three-line list, each tile contains a maximum of three lines of text.

The amount of text can vary between tiles within the same list.

Three-line list specs

Icon left padding: 16dp

List item left padding: 72dp

List right padding: 16dp

Example of a three-line list

Single-line list

Text only

Add 8dp of padding at the top and bottom of a list, except for lists with subheaders, which have their own padding.

Bottom padding is measured from the baseline.

Single-line list

Example of a single-line list

Single-line list specs

Font: Roboto Regular 16sp

Height: 48dp

Left and right padding from screen edge: 16dp

Top padding: 8dp

Dense single-line list specs

Font: Roboto Regular 13sp

Height: 40dp

Left and right padding from screen edge: 16dp

Top padding: 4dp

Icon with text

Add 8dp of padding at the top and bottom of a list, except for lists with subheaders, which have their own padding.

Bottom padding is measured from the baseline.

Single-line list with icon

Example of a single-line list with icon

Single-line list with icon specs

Font: Roboto Regular 16sp

Height: 48dp

Icon padding, left: 16dp

Text padding, left: 72dp

Top padding: 8dp

Dense single-line list with icon specs

Font: Roboto Regular 13sp

Height: 40dp

Icon padding, left: 16dp

Text padding, left: 72dp

Top padding: 4dp

Avatar with text

Bottom padding is measured from the baseline.

Add 8dp of padding at the top and bottom of a list, except for lists with subheaders, which have their own padding.

Single-line item with avatar

Example of a single-line item with avatar

Single-line item with avatar specs

Font: Roboto Regular 16sp

Height: 56dp

Left avatar padding: 16dp

Text padding, left: 72dp

Top padding: 8dp

Dense single-line item with avatar specs

Font: Roboto Regular 13sp

Height: 48dp

Left avatar padding: 16dp

Text padding, left: 72dp

Top padding: 4dp

Avatar with text and icon

Right icon padding: 16dp

Add 8dp of padding at the top and bottom of a list, except for lists with subheaders, which have their own padding.

Bottom padding is measured from the baseline.

Single-line item with avatar and icon

Example of a single-line item with avatar and icon

Single-line item with avatar and icon specs

Font: Roboto Regular 16sp

Height: 56dp

Left avatar padding: 16dp

Right icon padding: 16dp

Text padding, left: 72dp

Top padding: 8dp

Dense single-line item with avatar and icon specs

Font: Roboto Regular 13sp

Height: 48dp

Left avatar padding: 16dp

Right icon padding: 16dp

Text padding, left: 72dp

Top padding: 4dp

Two-line list

Text only

Primary text font: Roboto Regular 16sp

Secondary text font: Roboto Regular 14sp

Tile height: 72dp

Text padding, left: 16dp

Text padding, top and bottom: 20dp

Bottom padding is measured from the baseline.

Add 8dp of padding at the top and bottom of a list. One exception is at the top of a list with a subheader, because subheaders contain their own padding.

Two-line list

Example of a two-line list

Two-line list specs:

Font: Roboto Regular 16sp

Height: 72dp

Left and right padding from screen edge: 16dp

Top padding: 8dp

Dense two-line list specs:

Font: Roboto Regular 13sp

Height: 60dp

Left and right padding from screen edge: 16dp

Top padding: 4dp

Icon with text

Primary text font: Roboto Regular 16sp

Secondary text font: Roboto Regular 14sp

Tile height: 72dp

Left icon padding: 16dp

Text padding, left: 72dp

Text padding, top and bottom: 20dp

Bottom padding is measured from the baseline.

Add 8dp of padding at the top and bottom of a list, except for lists with subheaders, which have their own padding.

Two-line list with icon

Example of a two-line list with icon

Two-line list with icon specs:

Font: Roboto Regular 16sp

Height: 72dp

Icon left padding: 16dp

Text left padding: 72dp

Padding above list: 8dp

Text right padding: 16dp

Dense two-line list with icon specs:

Font: Roboto Regular 13sp

Tile height: 60dp

Icon left padding: 16dp

Text left padding: 72dp

Padding above list: 4dp

Text right padding: 16dp

Avatar with text

Primary text font: Roboto Regular 16sp

Secondary text font: Roboto Regular 14sp

Tile height: 72dp

Left avatar padding: 16dp

Text padding, left: 72dp

Text padding, top and bottom: 20dp

Bottom padding is measured from the baseline.

Center-align icon with text area.

Add 8dp of padding at the top and bottom of a list, except for lists with subheaders, which have their own padding.

Two-line list with avatar

Example of a two-line list with avatar

Two-line list with avatar specs:

Font: Roboto Regular 16sp

Tile height: 72dp

Icon left padding: 16dp

Text left padding: 72dp

Padding above list: 8dp

Text right padding: 16dp

Dense two-line list with avatar specs:

Font: Roboto Regular 13sp

Tile height: 60dp

Icon left padding: 16dp

Text left padding: 72dp

Padding above list: 4dp

Text right padding: 16dp

Avatar with text and icon

Primary text font: Roboto Regular 16sp

Secondary text font: Roboto Regular 14sp

Tile height: 72dp

Left avatar padding: 16dp

Text padding, left: 72dp

Right icon padding: 16dp

Bottom padding is measured from the baseline.

Add 8dp of padding at the top and bottom of a list, except for lists with subheaders, which have their own padding.

Avatar with text and icon

Example of an avatar with text and icon

Avatar with text and icon

Font: Roboto Regular 16sp

Tile height: 72dp

Icon left padding: 16dp

Text left padding: 72dp

Padding above list: 8dp

Text right padding: 16dp

Dense avatar with text and icon

Font: Roboto Regular 13sp

Tile height: 60dp

Icon left padding: 16dp

Text left padding: 72dp

Padding above list: 4dp

Text right padding: 16dp

Three-line list

Text only

Primary text font: Roboto Regular 16sp

Secondary text font: Roboto Regular 14sp

Tile height: 88dp

Text padding, left: 16dp

Text padding, top: 16dp

Text padding, bottom: 20dp

Bottom padding is measured from the baseline.

Add 8dp of padding at the top and bottom of a list, except for lists with subheaders, which have their own padding.

Three-line list

Example of a three-line list

Three-line list specs:

Font: Roboto Regular 16sp

Tile height: 88dp

Text left padding: 16dp

Text right padding: 16dp

Padding above list: 8dp

Dense three-line list specs:

Font: Roboto Regular 13sp

Tile height: 76dp

Text left padding: 16dp

Text right padding: 16dp

Padding above list: 4dp

Icon with text

Primary text font: Roboto Regular 16sp

Secondary text font: Roboto Regular 14sp

Tile height: 88dp

Left icon padding: 16dp

Text left padding: 72dp

Text padding, top: 16dp

Text padding, bottom: 20dp

Bottom padding is measured from the baseline.

Top-align icon with primary text.

Add 8dp of padding at the top and bottom of a list, except for lists with subheaders, which have their own padding.

Three-line list with icon

Example of a three-line list with icon

Three-line list with icon specs:

Font: Roboto Regular 16sp

Tile height: 88dp

Icon left padding: 16dp

Text left padding: 72dp

Text padding right: 16dp

Padding above list: 8dp

Dense three-line list with icon specs:

Font: Roboto Regular 13sp

Tile height: 76dp

Icon left padding: 16dp

Text left padding: 72dp

Text padding right: 16dp

Padding above list: 4dp

Avatar with text

Primary text font: Roboto Regular 16sp

Secondary text font: Roboto Regular 14sp

Tile height: 88dp

Left avatar padding: 16dp

Left text padding: 72dp

Text padding, top: 16dp

Text padding, bottom: 20dp

Bottom padding is measured from the baseline.

Top-align avatar with primary text.

Add 8dp of padding at the top and bottom of a list, except for lists with subheaders, which have their own padding.

Three-line list with avatar

Example of a three-line list with avatar

Three-line list with avatar specs:

Font: Roboto Regular 16sp

Tile height: 88dp

Padding above list: 8dp

Padding left of avatar: 16dp

Text left padding: 72dp

Text right padding: 16dp

Dense three-line list with avatar specs:

Font: Roboto Regular 13sp

Tile height: 76dp

Padding above list: 4dp

Padding left of avatar: 16dp

Text left padding: 72dp

Text right padding: 16dp

Avatar with text and icon

Primary text font: Roboto Regular 16sp

Secondary text font: Roboto Regular 14sp

Tile height: 88dp

Left avatar padding: 16dp

Text left padding: 72dp

Text padding, top: 16dp

Text padding, bottom: 20dp

Right icon padding: 16dp

Bottom padding is measured from the baseline.

Top-align avatar and icon with primary text.

Add 8dp of padding at the top and bottom of a list, except for lists with subheaders, which have their own padding.

Three-line list with avatar and icon

Example of a three-line list with avatar and icon

Three-line list with avatar and icon specs:

Font: Roboto Regular 16sp

Tile height: 88dp

Left avatar padding: 16dp

Text left padding: 72dp

Padding above list: 8dp

Right and left padding by the icon: 16dp

Dense three-line list with avatar and icon specs:

Font: Roboto Regular 13sp

Tile height: 76dp

Left avatar padding: 16dp

Text left padding: 72dp

Padding above list: 4dp

Right and left padding by the icon: 16dp

Lists: Controls

List controls display information and actions for list items.

List controls provide information, indicate a state, or present actions for list items. List controls appear as icons to the left or right of list text.

List control placement:

States and primary actions are placed on the left side of a list tile.

Secondary actions and info are placed on the right side of the tile.

Types

Checkbox

Switch

Reorder

Expand/collapse

Leave-behinds

Menu controls

Check

Inline information

Nested menu indicator

Usage

List controls fall under four categories:

State

Primary action (including text strings)

Secondary action

Secondary info

Distinguishing elements of list tiles need to be on the left for left-to-right interfaces, and vice versa. States and primary actions are placed on the left side of a list tile. Text within a list item should be considered part of the primary action target.

Don’t place two icons or actions next to one another, such as a checkbox next to an avatar.

If the primary action of the list item is navigational, don’t use an icon. The list item itself and its context should be sufficient to communicate the destination.

Secondary actions and info should be placed on the right side of the title. Secondary actions are always a separate target from the primary action, as users increasingly expect every icon to trigger an action.

States and primary actions are placed on the left side of a list tile. The secondary action and secondary info should be placed on the right side of the title.

Types of list controls

Checkbox

A checkbox can either be a primary action or a secondary action.

Type: primary action/state

Desktop checkboxes should appear only on hover and focus.

The checkbox is the primary action and the state indicator for the list item.

Type: secondary action

Separate target

When controlling a family of variables, consider using switches instead.

The checkbox is the secondary action for the list item.

Switch

Type: secondary action

Separate target

The switch is the secondary action and a separate target.

Reorder

Type: secondary action

Usually a separate target, depending on which mode list it is in.

Allows dragging of the list item to other locations within the list. It usually appears in list editing mode.

The reorder icon is the secondary action for the list item.

Expand/collapse

Type: secondary action

Separate target

Expands and collapses a list view vertically to show and hide details of existing list items.

By touching the list control, the user expands the list.

The list is expanded.

Collapsed single-line item

Expanded single-line item

Leave-behinds

Type: Other

A leave-behind is an informative hint as to what swiping a list item away will do to that item. The leave-behind can transform into an action.

Swiping on a list item from either direction will reveal an icon indicating the action. After swiping, a follow-up action can appear as a text button within the space of the list item.

Swiping on a list item from right to left will reveal an icon indicating the action.

Swiping on a list item from left to right will reveal an icon indicating the action.

Types of menu controls

A menu is a special type of list. In menus, use controls that are appropriate for dropdown menus, overflow menus, and so on. Don’t use these controls in regular lists. Use the regular list controls instead.

Check

Type: state

Not a separate target.

Menus only. Indicates that the list item has been selected. The selection is done through a different control.

The check indicates that the list item has been selected.

Inline information

Type: secondary info

Not a separate target.

Menus only. Inline information is a small snippet of text related to the line title that can provide information or a tip, like a keyboard shortcut. It cannot be truncated.

The inline information for Bold, Italic and Underline provide keyboard shortcuts.

Nested menu indicator

Type: primary action

Not a separate target.

Menus only. The glyph indicates that a secondary menu will fly out upon hover, focus, and/or selection.

The glyphs for the menu items indicate that a secondary menu will fly out upon hover, focus, and/or selection.

Menus

Menus display a list of choices on a transient sheet of material.

Menus appear upon interaction with a button, action, or other control. They display a list of choices, with one choice per line.

Menu items may be disabled if not applicable to a certain context. Contextual menus dynamically change their available menu items based on the current state of the app.

Types

Simple menus (mobile and tablet)

Cascading menus (desktop)

Elevation

Menus appear above other in-app elements.

Behavior

Scrollable

Alternatives

Simple dialogs

Usage

A menu is a temporary piece of material that appears upon interaction with a button, action, pointer, or other control. It contains at least two menu items.

Each menu item consists of a discrete option or action that can affect the app, the view, or selected elements within a view.

Menus should not be used as a primary method for navigation within an app.

The action overflow in the app bar will emit a menu.

This action overflow menu contains five menu items: “Refresh”, “Settings”, “Send feedback”,“Help,” and “Signout”.

Menu labels

The label of a button or control should concisely and accurately reflect the items within the menu. Menu bars typically use single words as labels, like “file,” “format,” and “edit.” Other contexts may require longer labels.

Disabled menu options

Menus display a consistent set of menu items. A menu item may be enabled or disabled based on the current state of the application.

Do.

The title “View” accurately describes the unifying characteristic of the menu items it contains.

Don't.

The title “Stuff” is not descriptive and does not help users predict what menu items they will find in the Stuff menu.

Contextual menus

Contextual menus dynamically change their available menu items based on the current state of the application.

Menu items that are irrelevant to the current context may be removed.

Menu items which are relevant but need certain conditions to be met may be disabled. For example, the Copy menu option becomes enabled when text is selected.

Single menu-item states

Some app states may result in a contextual menu with only a single menu item. For example, when highlighting text on a web page, Android display only the Copy menu item, as users cannot cut or paste text.

Contextual menu

Menus are scrollable

If the height of a menu prevents all menu items from being displayed, the menu can scroll internally. One example is when viewing a menu on a phone in landscape orientation.

Internally scrolling menu

Cascading menus (Desktop only)

For menus that cascade, position the menus vertically and horizontally based on their proximity to screen edges.

Developer note: Implementations of this component may vary by platform. By using standard platform implementations, you will receive any related future improvements.

Cascading menu on desktop

Cascading dropdown menu on desktop

Dropdown menu

Action overflow menu

Menu items

Single-line display

Each menu item is limited to a single line of text (a single word or short phrase) that describes the action it will perform when selected.

Menu items may also contain:

Icons and helper text, like keyboard shortcuts

Controls like checkmarks to indicate multiple selected items or states (See List controls)

Menu ordering

Menus with static content should have the most frequently used menu items placed at the top of the menu.

Menus with dynamic content may have other behavior, such as placing previously used fonts at the top of the menu. The order can change based on user actions.

Menu nesting

Menu items can reveal nested submenus. Ideally, limit nesting to one level deep, as it can be difficult to navigate multiple nested submenus.

Examples of menu items

Disabled actions

Displaying actions as disabled, rather than removing them, lets the user know they exist under the right conditions.

For example, Redo is disabled when there is nothing to redo. Cut and Copy are disabled until content is selected.

Examples of disabled actions

Behavior

Location

Menus appear above all other in-app UI elements.

Example of menu appearing over all other in-app UI elements.

Menus are positioned over their emitting elements such that the currently selected menu item appears on top of the emitting element.

Do.

The currently selected menu item appears over the emitting element.

Do not display a duplicate of the selected menu item.

Don't.

Do not display a duplicate of the selected menu item.

Positioning the menu below the emitting element separates it from its context.

Don't.

Menus are positioned over their emitting element.

Dismissing menus

Dismiss a menu by tapping outside of the menu, or by tapping the emitting button (if visible).

Selecting a menu item should also dismiss the menu. An exception is when a menu allows for multiple items to be chosen, for example, by using checkmarks.

Simple Menus

Mobile or tablet

Use simple menus in lists to display the options for a specific list item.

Vertical alignment

When close to a screen edge, simple menus vertically realign to make all menu items are completely visible.

Disambiguation: In contrast to simple menus, simple dialogs can present additional detail related to the options available for a list item or provide navigational or orthogonal actions related to the primary task. Although they can display the same content, simple menus are preferred over simple dialogs because simple menus are less disruptive to the user’s current context.

Item selection

Choosing an option immediately commits the option and closes the menu.

Cancelling selection

Touching outside of the menu, or pressing the system Back button, cancels the action and closes the menu.

Example of a simple menu

When opened, simple menus attempt to vertically align the currently selected menu item with the list item. The currently selected menu item is highlighted.

Do.

The currently selected menu item aligns vertically over the list item.

Do.

The currently selected menu item aligns vertically over the list item.

Do not arbitrarily position the first menu item over the list item.

Don't.

The first menu item is arbitrarily positioned over the list item, regardless of the currently selected menu item.

When close to a screen edge, simple menus reposition their vertical alignment so that all menu items are completely visible.

Do.

The currently selected menu item normally would be positioned over the list item, but because doing so in this case would cause part of the menu to appear off-screen, the entire menu is repositioned vertically.

Simple menus appear over their emitting element, not below.

Do.

Simple menus appear over their emitting element.

Don't.

Simple menus do not appear below their emitting element.

Menu width varies depending on string length, and on mobile is defined as a multiple of a 56dp unit.

Simple menus always maintain a 16dp margin (phone) or 24dp margin (tablet) to the left and right edges of the screen.

Simple menu with a 16dp margin.

Each unit is 56dp wide.

Minimum width on mobile = 2 \* 56dp = 112dp

Simple menu with 16dp margins on the left and right.

Each unit is 56dp wide.

Maximum width on mobile (in both portrait and landscape) = 5 \* 56dp = 280dp

If text in a simple menu wraps to a second line, use a simple dialog instead. Simple dialogs can have rows with varying heights.

Do.

Use a simple dialog if any text in a simple menu wraps to a second line.

Don't.

Don’t truncate text in a simple menu. Use a simple dialog instead.

Menus show a persistent scroll bar when content is scrollable.

The maximum height of a simple menu should be one or more rows less than the view height. This ensures a tappable area outside of the simple menu with which to dismiss the menu.

Do.

Display a scroll bar by default for scrollable content.

Don't.

Without a scroll bar, it’s unclear if there are additional menu items available.

Don’t duplicate the selected menu item.

Do.

Each menu item appears only once.

Don't.

Don’t display a duplicate of the selected element.

Simple menus are always left-aligned with the start of the list item text and do not reposition horizontally based on the touch location.

Do.

Simple menus are left-aligned regardless of touch location.

Don't.

Simple menus do not reposition based on the touch location.

Specs

Specs are provided for various sizes and types of menus and for different platforms. Add 8dp padding at the top and bottom of a menu.

The type style used for menus is defined by the Material typographic scale.

By default, menus use: 16sp (mobile devices), 15px (desktop)

Dense interfaces can use: 14sp (mobile devices), 13px (desktop)

Mobile

Menu item height: 48dp

Menu item text left padding: 16dp

Menu item text bottom padding: 20dp

Top padding: 8dp

Bottom padding: 8dp

Typography: 16sp

Various widths

Menus of various widths: min width 1.5 x, 3x, 6x and 7x

Desktop

Default desktop menu

Menu item height: 32px

Menu item text left padding: 24px

Top padding: 8px

Bottom padding: 8px

Typography: 15px

Dense desktop menus

Menu item height: 24px

Menu item text left padding: 24px

Top padding: 4px

Bottom padding: 4px

Typography: 13px

Cascading menu

Cascading menu on desktop

Cascading redlines

Default Cascading menu

Cascading menu top padding: 16px

Cascading menu left padding: 24px

Cascading menu item height: 32px

Menu item icon right padding: 32px

Menu sub-item left padding: 64px

Typography: 15px

Dense Cascading menus can use 13px typography

Pickers

Pickers provide a simple way to select a single value from a pre-determined set.

Date pickers use a dialog window to select a single date on mobile.

Time pickers use a dialog to select a single time (in the hours:minutes format) on mobile. They adjust to a user’s preferred time setting.

Alternatives

Segmented dropdown buttons

Usage

On mobile, pickers are best suited for display in a confirmation dialog.

For inline display, such as on a form, consider using compact controls such as segmented dropdown buttons.

Date pickers

A dialog picker is used to select a single date on mobile.

The selected day is indicated by a filled circle. The current day is indicated by a different color and type weight.

Date and year picker: Light theme, portrait

Swipe left to right to navigate through the months. Touch the year in the title bar to transition to the year view.

Date picker navigation

The picker view adapts to the device’s orientation.

Date picker: Light theme, landscape

A dark theme picker can complement apps that have a dark theme.

Date picker: Dark theme, portrait

Landscape view of the dark theme picker.

Date picker: Dark theme, landscape

Time pickers

A time picker adjusts to a user’s preferred time setting, i.e. the 12-hour or 24-hour format.

A dialog picker is used to select a single time (hours:minutes) on mobile.

The selected time is indicated by the filled circle at the end of the clock hand.

Time picker: Light theme, portrait

Tap between the hour and minute label in the title bar to switch between the hour or minute view. Additionally, tap on the AM/PM label in the title bar to switch between AM or PM (in 12-hour clock mode).

Time picker navigation

The picker view adapts to the device’s orientation.

Time picker: Light theme, landscape

A dark theme picker can complement apps that have a dark theme.

Time picker: Dark theme, portrait

Landscape view of the dark theme picker

Time picker: Dark theme, landscape

Progress & activity

Progress and activity indicators are visual indications of an app loading content.

A single visual indicator should be used to represent each type of operation. For example, a refresh operation should display either a refresh bar or an activity circle, but not both.

Determinate indicators display how long an operation will take.

Indeterminate indicators visualize an unspecified wait time.

Types

Linear

Circular

Behavior

Loading content in phases

Loading additional content

Types of indicators

When indicators are determinate they indicate how long an operation will take when the percentage complete is detectable.

When indicators are indeterminate they request that the user wait while something finishes when it’s not necessary to indicate how long it will take.

Both linear and circular progress indicators may be either determinate or indeterminate.

Linear

A linear progress indicator should always fill from 0% to 100% and never decrease in value. It should be represented by bars on the edge of a header or sheet that appear and disappear.

For multiple operations happening in sequence, use the indicator to represent the progress as a whole, and not each individual operation.

Linear progress indicators

Buffering progress indicator on a video

Query indicator on a webpage

Circular

Circular indeterminate and determinate progress indicators

Circular with integration

A circular loader may be integrated with a floating action button.

Animation of a circular loader with integration

Behavior

Loading in phases

One-phased loads

Ink (copy and images) loads within an existing, unchanging container.

Two-phased loads

The paper container is generated, then the ink (copy and images) load within it.

Loading content for the first time

Load and display all content at once.

Loading additional content

Example 1: Card expansion

An indeterminate linear indicator is recommended for a card expanding on larger surfaces such as desktop.

Example 2: Scroll up to load more

An indeterminate circular indicator with an initial radial ink reaction is recommended when loading a list from below.

Example 3: Swipe down to refresh

An indeterminate circular indicator with a radial ink reaction is recommended when refreshing a list from above.

Selection controls

Selection controls allow the user to select options.

Three types of selection controls are covered in this guidance:

Checkboxes allow the selection of multiple options from a set.

Radio buttons allow the selection of a single option from a set.

Switches allow a selection to be turned on or off.

Color

Selection controls use an app’s accent color.

Themes

Selection controls are available in both dark and light themes.

Checkbox

Checkboxes allow the user to select multiple options from a set.

If you have multiple options appearing in a list, you can preserve space by using checkboxes instead of on/off switches.

If you have a single option, avoid using a checkbox and use an on/off switch instead.

Light theme

Dark theme

Light

On: Swatch 500, Opacity 100%

Off: #000000, Opacity 54%

Disabled: #000000, Opacity 26%

Light theme for checkboxes in hover, focused, pressed, disabled, and disabled focused states.

Dark

On: Swatch 500, Opacity 100%

Off: #FFFFFF, Opacity 70%

Disabled: #FFFFFF, Opacity 30%

Dark theme for checkboxes in hover, focused, pressed, disabled, and disabled focused states.

Radio button

Radio buttons allow the user to select one option from a set. Use radio buttons for exclusive selection if you think that the user needs to see all available options side-by-side.

Otherwise, consider a dropdown, which uses less space than displaying all options.

Light theme for radio buttons

Dark theme for radio buttons

Light

On: Swatch 500, Opacity 100%

Off: #000000, Opacity 54%

Disabled: #000000, Opacity 26%

Light theme for radio buttons in hover, focused, pressed, disabled, and disabled focused states.

Dark

On: Swatch 500, Opacity 100%

Off: #FFFFFF, Opacity 70%

Disabled: #FFFFFF, Opacity 30%

Dark theme for radio buttons in hover, focused, pressed, disabled, and disabled focused states.

Switch

On/off switches toggle the state of a single settings option. The option that the switch controls, as well as the state it’s in, should be made clear from the corresponding inline label. Switches take on the same visual properties of the radio button.

The on/off slide toggle with the text “on” and “off” included within the asset is deprecated. Use the switch shown here instead.

Light theme

Dark theme

Light

Thumb On: Swatch 500, Opacity 100%

Track On: Swatch 500, Opacity 50%

Thumb Off: Grey 50, #FAFAFA, Opacity 100%

Track Off: #000000, Opacity 38%

Thumb Disabled: Grey 400, #BDBDBD, Opacity 100%

Track Disabled: #000000, Opacity 12%

Light theme for switches in hover, focused, pressed, disabled, and disabled focused states.

Dark

Thumb On: Swatch 200, Opacity 100%

Track On: Swatch 200, Opacity 50%

Thumb Off: Grey 400, #BDBDBD, Opacity 100%

Track Off: #FFFFFF, Opacity 30%

Thumb Disabled: Grey 800, #424242, Opacity 100%

Track Disabled: #FFFFFF, Opacity 10%

Dark theme for switches in hover, focused, pressed, disabled, and disabled focused states.

Use the outer radial reaction only on form factors that favor finger touch, where interaction may obstruct the element completely. For desktop usage with a mouse, you do not need this extra indication.

Radial reaction on mobile

No radial reaction on desktop

Sliders

Sliders let users select from a range of values by moving the slider thumb.

Sliders are ideal components for adjusting settings that reflect intensity levels, such as volume, brightness, or color saturation.

Sliders may have icons on both ends of the bar that reflect the value intensity. Place the smallest value for the slider range on the left and the largest value on the right.

Continuous sliders

Continuous sliders allow users to select a value along a subjective range. They do not require a specific value to make adjustments, although they may, in some instances, offer an editable numeric value.

Discrete sliders

Discrete sliders allow users to select a specific value from a range.

Continuous slider

Use continuous sliders for subjective settings that do not require a specific value for the user to make meaningful adjustments.

Light

Thumb on: Swatch 500, Opacity 100%

Track on: Swatch 500, Opacity 100%

Thumb off: #000000, Opacity 26%

Track off: #000000, Opacity 26%

Focused and Click Thumb off: #000000, Opacity 38%

Focused and Click Track off: #000000, Opacity 38%

Disabled (disconnected): #000000, Opacity 26%

Light theme

Dark

Thumb on: Swatch 200, Opacity 100%

Track on: Swatch 200, Opacity 100%

Thumb off: #FFFFFF, Opacity 30%

Track off: #FFFFFF, Opacity 30%

Disabled (disconnected): #FFFFFF, Opacity 30%

Dark theme

Various slider behaviors

Spacing for icons to the left and/or right of the bar:

Example of spacing for icons in a slider.

Examples of sliders in normal, hover, focus, click, and disabled states.

With an editable numeric value

Use for settings for which users need to set the exact value. Upon pressing the thumb, the text box becomes editable for text entry and updates the value automatically with thumb movement.

Example of a slider with an editable numeric value

Example of a slider with an editable numeric value in normal, hover, focus, click, and disabled states.

Discrete slider

The discrete slider thumb snaps to evenly spaced tick marks along the slider rail. Use for objective settings that require specific values for the user to make meaningful adjustments. Each tick mark should change the setting to a level that’s discernible to the user. The values are predetermined and aren’t user-editable.

With a numeric value label

Use for settings for which users need to know the exact value of the setting.

Light theme

Dark theme

Animation of a discrete slider

Snackbars & toasts

Snackbars provide brief feedback about an operation through a message at the bottom of the screen.

Snackbars contain a single line of text directly related to the operation performed. They may contain a text action, but no icons.

Toasts (Android only) are primarily used for system messaging. They also display at the bottom of the screen, but may not be swiped off-screen.

Usage

Only one snackbar may be displayed at a time. Each snackbar may contain a single action, neither of which may be “Dismiss” or “Cancel.”

Behavior

Snackbars animate upwards from the bottom edge of the screen.

Snackbar specs

Action button: Roboto Medium 14sp, all-caps text

Mobile height: 48dp (single-line), 80dp (multi-line)

Desktop snackbar height: 48dp

Usage

Show only one snackbar on screen at a time.

Placement

Snackbars appear above most elements on screen, and they are equal in elevation to the floating action button. However, they are lower in elevation than dialogs, bottom sheets, and navigation drawers.

Behavior

Upon entrance, snackbars animate upwards from the bottom edge of the screen. When they appear, they do not block input.

They exit by being swiped off-screen or automatically disappear after a timeout or user interaction elsewhere (such as summoning a new surface or activity).

Very short text strings

Snackbars should contain a single line of text directly related to the operation performed. They may contain a text action. They cannot contain icons.

Do.

Don't.

Transient

Snackbars automatically time out from the screen. For usability reasons, snackbars should not contain the only way to access a core use case. They should not be persistent or be stacked, as they are above other elements on screen.

On Android, when an unrelated dialog or popup occurs while the snackbar is up, the snackbar timeout will reset upon the window focus being regained. This is to ensure that the user will be able to read the snackbar for the full intended duration.

Do.

Don't.

0-1 actions, not dismiss or cancel

If an action is present, comply with dialog spacing and affordance rules. For two or more actions, use a dialog, not a snackbar, even when one of the actions is a dismiss action. If the action described in the snackbar is important enough to block the use of the screen, it should be a dialog.

Do.

Don't.

Don’t block the floating action button

Move your floating action button vertically to accommodate the snackbar height.

Do.

Don't.

Consecutive snackbars

There is only ever one snackbar displayed. When a second snackbar is triggered while the first is displayed, the first should start the contraction motion downwards before the second one animates upwards.

Two consecutive snackbars with different content.

Two consecutive snackbars with the same content.

Specs

Mobile

Single-line snackbar height: 48dp

Multi-line snackbar height: 80dp

Text: Roboto Regular 14sp

Action button: Roboto Medium 14sp, all-caps text

Default background fill: #323232 100%

A mobile snackbar

A mobile snackbar

Animation of a mobile snackbar

Mobile in landscape with a snackbar

Specs of a mobile snackbar

Tablet/desktop

Single-line snackbar height: 48dp tall

Minimum width: 288dp

Maximum width: 568dp

2dp rounded corner

Text: Roboto Regular 14sp

Action button: Roboto Medium 14sp, all-caps text

Default background fill: #323232 100%

Alignment: Centered or left-aligned 24dp from the left and bottom edges of the screen

A center aligned tablet/desktop snackbar

A center-aligned tablet/desktop snackbar

A left-aligned tablet/desktop snackbar

A left-aligned tablet/desktop snackbar

Specs of tablet/desktop snackbar

Snackbar height: 48dp

Snackbar left and right padding: 24dp

Snackbar top and bottom padding: 14dp

Snackbar button left padding: 48dp

Animation of a tablet/desktop snackbar

Steppers

Steppers convey progress through numbered steps.

Steppers display progress through a sequence of logical and numbered steps. They may also be used for navigation.

Steppers may display a transient feedback message after a step is saved.

Types of Steps

Editable

Non-editable

Mobile

Optional

Types of Steppers

Horizontal

Vertical

Linear

Non-linear

Usage

Steppers display progress through a sequence by breaking it up into multiple logical and numbered steps.

Steppers should be used when a field determines a subsequent field.

Active stepper circle

24dp x 24dp

12sp Roboto Regular

Inherits primary color

Default color is Google Blue 500

Active step

14sp Roboto Medium

87% black

Connector line

1dp border

Grey 400 color value

Inactive stepper circle

24dp x 24dp

12sp Roboto Regular

100% white

38% black

Inactive steps

14sp Roboto Regular

38% black

Avoid using steppers to break up sections in a short form, or multiple times on one page.

Don't.

Don’t embed steppers within steppers or use multiple steppers on one page.

Types of steps

Editable steps

Editable steps allow users to return later to edit a step. These are ideal for workflows that involve editing steps within a session.

Editable steps can be edited within a session.

Non-editable steps

Non-editable steps should be used when:

Users cannot edit a step later

Step editing poses a distraction risk to form completion

Non-editable steps cannot be edited once completed.

Mobile steps

Mobile step text

Steps displayed at the top in text

Mobile step dots

Use dots when the number of steps isn’t large.

Mobile step progress bar

Use a progress bar when there are many steps, or if there are steps that need to be inserted during the process (based on responses to earlier steps).

Optional steps

Optional steps within a linear flow should be marked as optional.

Mark optional steps in linear flows as optional.

Optional steps

12sp Roboto Regular

54% black

Types of steppers

Horizontal steppers

Horizontal steppers are ideal when the contents of one step depend on an earlier step.

Avoid using long step names in horizontal steppers.

On horizontal steppers, the step name and numbers appear on a horizontal bar, which can be fixed to the top of the page when scrolling down.

Vertical steppers

Vertical steppers are designed for narrow screen sizes. They are ideal for mobile.

Vertical steppers can be used in mobile as-is. Simply ensure that the contents for each step are responsive.

Linear steppers

Linear steppers require users to complete one step in order to move on to the next.

Each linear step must be completed before proceeding to the next one.

Non-linear steppers

Non-linear steppers allow users to enter a multi-step flow at any point.

Non-linear steps may be completed in any order.

Inactive stepper circle

24dp x 24dp

12sp Roboto Regular

Text 100% white

38% black

Inactive steps

14sp Roboto Regular

54% black

Alternative labels

Alternative label placement

Alternative label placement with an optional step

Stepper feedback

Steppers may display a transient feedback message after a step is saved. Stepper feedback should only be used if there is a long latency between steps.

This stepper displays a transient feedback message after a step is saved.

Specs

Standard stepper

Step height: 72dp

Icon top, bottom, and left padding: 24dp

Icon right padding: 8dp

Optional step

Step height: 72dp

Icon left and right padding: 8dp

Label right padding: 8dp

Multi-line error state

Alternative label

Step height: 104dp

Icon left and right padding: 8dp

Label top padding: 16dp

Icon and label padding from edge: 24dp

Alternative label placement with optional step

Icon left and right padding: 8dp

Step left and right padding: 16dp

Label bottom padding: 24dp

Icon and label padding from screen edge: 24dp

Alternate label placement error state

Non-linear steppers

Hover and pressed states

Hover

Hover

Pressed

Pressed

Background

6% black

Stepper circle

24dp x 24dp

12sp Roboto Regular

Text 100% white

38% black

Vertical steppers

Vertical stepper, with sub-steps

Icon left padding: 24dp

Icon alignment with label: Center vertical

Icon top and bottom padding: 8dp

Label top padding: 24dp

Label bottom padding: 16dp

Vertical space between steps: 48dp

Button height: 48dp

Button top padding: 16dp

Example of a vertical stepper

Mobile steppers

Icon top and bottom padding: 8dp

Label top padding: 24dp

Label button padding (no sublabel): 16dp

Vertical space below inactive step: 40dp

Vertical space below active step: 48dp

Button height: 48dp

Subheaders

Subheaders are list tiles that delineate sections of a list or grid list.

Subheaders may be displayed inline with tiles or associated with content. They are typically related to filtering or sorting criteria.

Upon scrolling, subheaders remain pinned to the top of the screen until pushed off screen by the next subheader.

Subheader types

List

Grid

Menu

Font

Roboto Medium 14sp

List subheaders

Specs

Tile height: 48dp

Subheader font: Roboto Medium 14sp

Color

Subheader color can either be the secondary grey text value (54% black) or the primary color of the app.

In long lists of complex information, consider using colored subheaders to delineate content. Android settings contain an example of how color can be used on subheaders.

Grey subheader

Padding: 16dp

Subheader tile height: 48dp

Grey subheader in email inbox

Alignment

A subheader should be left-aligned with an avatar/icon in a list, unless there is a promoted action in that space.

If there is a floating action button left-aligned with the avatar/icon in a list, align the subheader with the text content.

With a left-aligned floating action button, the subheader is aligned with the text content.

Subheader tile height: 48dp

Floating action button tile width: 72dp

The subheader is in the primary color of the app and aligned with the text because of the left-aligned floating action button.

Grid subheaders

Tile height: 48dp

Subheader font: Roboto Medium 14sp

Subheader is aligned 16dp from the left edge

The grid subheader is aligned 16dp from the left edge. The height of the grid subheader tile is 48dp.

Grid subheaders for the months of the year in a photo album

Menu subheaders

Tile height: 48dp

Subheader font: Roboto Medium 14sp

Subheader is aligned 16dp from the left edge

Menu subheader specs

Subheader padding from left edge: 16dp

Tile height: 48dp

Example of menu subheader

Tabs

Tabs make it easy to explore and switch between different views.

Tabs enable content organization at a high level, such as switching between views, data sets, or functional aspects of an app.

Present tabs as a single row above their associated content. Tab labels should succinctly describe the content within.

Because swipe gestures are used for navigating between tabs, don't pair tabs with content that also supports swiping.

Types

Fixed

Scrollable

Tab labels

Tab labels may include icons and text.

Color

Apply your app’s accent color, or a contrasting color, to text fields and the text field cursor.

Text fields

Text fields allow users to input, edit, and select text.

Text fields typically reside in forms but can appear in other places, like dialog boxes and search.

Usage

Text fields validate input, help users fix errors, autocomplete words, and provide suggestions.

Field types

Single-line

Multi-line

Text area

Input types

Formatted inputs

Nested menus & pickers

Text fields

Text fields allow users to input, edit, and select text.

Text fields typically reside in forms but can appear in other places, like dialog boxes and search.

Usage

Text fields validate input, help users fix errors, autocomplete words, and provide suggestions.

Field types

Single-line

Multi-line

Text area

Input types

Formatted inputs

Nested menus & pickers

Principles

Text fields should have the following characteristics:

Identifiable

Using a tappable touch target, text fields should indicate that users can enter information.

Findable

It should be easy to find a text field among other elements.

Legible

Text fields should indicate their state – whether enabled or disabled, empty or filled, valid or invalid – with clear label, input, and assistive text.

Layout

Text fields contain the following elements:

Label

Text field labels display the type of input a field requires. Every text field should have a label.

Labels are aligned with the input line and always visible. They can be resting (when a field is inactive and empty) or floating.

alt-text

Resting label text

Top padding: 16dp

Bottom padding: 8dp

alt-text

Floating label text

Top padding above label: 16dp

Bottom padding below label: 8dp

Bottom padding below input: 8dp

Input line

The input line indicates where to enter text, displayed below the label.

When a field is active or contains an error, the line’s color and thickness vary.

alt-text

Labels rest above the line when the text field is empty.

alt-text

Labels float upward when the text field is in focus or filled.

Cursor

The cursor indicates the user’s current input position.

alt-text

The cursor is inset with the label until the user enters input.

Input text

Text entered into a text field is called input text.

The first letter of input text (and each sentence in a field) should be capitalized where appropriate. Such as text that:

Names something, like an alarm

Contains phrases, like a text message

alt-text

Input text in an active field

alt-text

Input text

Top padding: 8dp

Bottom padding: 8dp

Input text can be used with autocomplete to help users who have limited literacy or who write in a foreign language. For example, autocomplete can:

Suggest input as it’s typed (refreshing suggestions with each keystroke)

Fill a field with default input text

Pressing the return button accepts the current autocomplete suggestion.

alt-text

Reduce the opacity of suggested text, while keeping typed text at full opacity.

Placeholder text (Hint text)

Placeholder text rests in the input field until the user starts entering text. It may contain an action or an example, such as a phone number or email address.

alt-text

Placeholder text can be displayed in an empty text field until input is entered.

alt-text

Placeholder text

Padding above label: 16dp

Padding above placeholder text: 8dp

Padding below placeholder text: 8dp

Helper text

Helper text gives context about a field’s input, such as how the input will be used.

It should be visible either persistently or only on focus.

Specs:

Left justified

On a single line if possible, or with text wrapping (if multiple lines)

alt-text

Helper text can provide additional context below the text field.

alt-text

Helper text

Padding above helper text: 8dp

alt-text

Helper text can support placeholder text in guiding the user.

Error message

When input isn’t accepted, text fields can display an error message below the input line, with instructions on how to fix the error. Until the error is fixed, the error replaces the helper text.

An error message should appear on a single line, if possible.

alt-text

An error message helps guide the user to fix invalid input.

alt-text

Error message

Padding above the error message: 8dp

An error message is preceded by the word “Error” (or an error icon) to make the error state more distinct to colorblind users.

alt-text

The term “Error:” (including the colon) precedes the error message

alt-text

Error icon inline with the error message

Required field

To indicate that a field is required, display an asterisk (\*) next to its label. At the bottom of the form, mention that asterisks indicate required fields.

alt-text

Required asterisk indicator in an empty text field

alt-text

Required asterisk indicator in a filled text field.

To indicate that a field is optional, display the word “optional” in parentheses next to the field label.

alt-text

Optional indicator text in an empty text field

alt-text

Optional indicator text in a filled text field

In forms with some required fields, indicate all of the required ones. However, if there are fewer optional fields, indicate those instead.

alt-text

Name and zip code are the only required fields in the form.

alt-text

Description is the only optional field in the form.

Character or word counter

Use character or word counters where there is a character or word limit.

Specs:

Right justified

Displayed as a ratio of characters used and the character limit (formatted as: characters used / character limit)

alt-text

The character counter shows that 5 out of 10 possible characters have been entered.

alt-text

Character counter

Padding below the text input line: 8dp

Icon signifier

Icons describe the type of input a text field requires. They are displayed to the left of the text field.

Icons can also be touch targets for nested components. For example, a calendar icon may be tapped to reveal a date picker.

alt-text

A calendar icon indicates that the text field is for entering a date.

alt-text

Icon signifier

Width: 24dp

Padding between icon signifier and label: 16dp

Voice input icon

A microphone icon signifies that users can input characters using voice.

Voice input icons are right justified, and always enabled.

alt-text

Clicking on the icon allows users to input characters using voice.

alt-text

Voice input icon

Width: 24dp

Padding below icon: 8dp

Dropdown icon

A dropdown arrow indicates that a text field contains nested selections.

alt-text

Clicking on the dropdown arrow expands the nested selection component.

alt-text

Dropdown arrow container

Width: 24dp

Height: 24dp

Padding below icon: 8dp

Clear button

Clear buttons let users empty an entire input field.

They are right justified in the field.

alt-text

The clear icon is enabled when there are characters entered in the field.

alt-text

Clear icon

Width: 24dp

Height: 24dp

Padding below icon: 8dp

In forms with many or crowded fields, text fields may be made shorter.

Specs:

Input text: 13sp

Reduce space between the label and input text

Reduce space between the input line and help text by 50%

alt-text

Normal spacing

Padding above label: 16dp

Padding below label: 8dp

Padding above text input line: 8dp

Padding below text input line: 8dp

alt-text

Dense spacing

Padding above label: 8dp

Padding below label: 4dp

Padding above text input line: 8dp

Padding below text input line: 4dp

States

Text fields have two major states: enabled or disabled.

In the enabled state, these user interactions are available:

Idle

Hover

Pressed

Focused

Text field input can be either:

Empty or filled

Valid or invalid

alt-text

Text field states in light and dark themes

Text field states

Enabled

Enabled text fields are empty by default.

Specs

Labels have 8dp of padding above the input line

Label text is the same size as input text

Accessibility

The line should have a 3:1 contrast ratio to improve identifiability, the same AA accessibility standard as large text.

Light theme

Bottom line

Thickness: 1dp

Color: 42% black (3:1 contrast ratio)

Label text

Type: Roboto 16sp

Color: 54% black (4.5:1 contrast ratio)

Spacing: 16dp top padding, 8dp bottom padding, inset with bottom line

Helper text (optional)

Type: Roboto 12sp

Color: 54% black (4.5:1 contrast ratio)

Spacing: 16dp left padding, 8dp top padding

Dark theme

Bottom line

Thickness: 1dp

Color: 70% white (7:1 contrast ratio)

Label text

Type: Roboto 16sp

Color: 70% white (7:1 contrast ratio)

Spacing: 16dp top padding, 8dp bottom padding, inset with bottom line

Helper text (optional)

Type: Roboto 12sp

Color: 70% white (7:1 contrast ratio)

Spacing: 16dp left padding, 8dp top padding

Disabled

Disabled text fields are uneditable. They have a dotted input line and less opacity so that they appear less tappable.

Light theme

Bottom line

Thickness: 1dp

Color: 42% black

Dotted

Label text

Type: Roboto 12sp

Color: 42% black (3:1 contrast ratio)

Spacing: 8dp bottom padding, inset with bottom line

Input text

Type: Roboto 16sp

Color: 42% black (3:1 contrast ratio)

Spacing: 8dp top padding, inset with bottom line

Dark theme

Bottom line

Thickness: 1dp

Color: 70% white

Dotted

Label text

Type: Roboto 12sp

Color: 50% white (4.5:1 contrast ratio)

Spacing: 8dp bottom padding, inset with bottom line

Input text

Type: Roboto 16sp

Color: 50% white (4.5:1 contrast ratio)

Spacing: 8dp top padding, inset with bottom line

User interactions

Hover

Hover states give the input line more opacity, helping make text fields more noticeable on desktop.

Light theme

Bottom line

Thickness: 2dp

Color: 87% black (15:1 contrast ratio)

Label text

Type: Roboto 16sp

Color: 54% black (4.5:1 contrast ratio)

Spacing: 8dp bottom padding, inset with bottom line

Helper text (optional)

Type: Roboto 12sp

Color: 54% black (4.5:1 contrast ratio)

Spacing: 8dp top padding, inset with bottom line

Dark theme

Bottom line

Thickness: 2dp

Color: 100% white (13:1 contrast ratio)

Label text

Type: Roboto 16sp

Color: 70% white (7:1 contrast ratio)

Spacing: 8dp bottom padding, inset with bottom line

Helper text (optional)

Type: Roboto 12sp

Color: 70% white (7:1 contrast ratio)

Spacing: 8dp top padding, inset with bottom line

Press

Upon press, the text field label shrinks and floats above the field, revealing the cursor and any placeholder text.

To signal that the field is active, the primary color of the UI is applied to the input line, label text, and cursor.

Light theme

Bottom line

Thickness: 2dp

Color: 100% primary color (Dark A700)

Motion: Ripple & wash

Label text

Type: Roboto 12sp

Color: 87% primary color (Dark A700)

Spacing: 8dp bottom padding, inset with bottom line

Placeholder text (optional)

Type: Roboto 16sp

Color: 42% black (3:1 contrast ratio)

Spacing: 8dp bottom padding, inset with bottom line

Helper text (optional)

Type: Roboto 12sp

Color: 54% black (4.5:1 contrast ratio)

Spacing: 8dp top padding, inset with bottom line

Dark theme

Bottom line

Thickness: 2dp

Color: 100% primary color (Light A200)

Motion: Ripple & wash

Label text

Type: Roboto 12sp

Color: 100% primary color (Light A200)

Spacing: 8dp bottom padding, inset with bottom line

Placeholder text (optional)

Type: Roboto 16sp

Color: 50% white (4.5:1 contrast ratio)

Spacing: 8dp bottom padding, inset with bottom line

Helper text (optional)

Type: Roboto 12sp

Color: 70% white (7:1 contrast ratio)

Spacing: 8dp top padding, inset with bottom line

Focus

As the user types into the text field, input text replaces any placeholder text. Autocompleted text is displayed to the right of the cursor, using the same font size and color as placeholder text.

Light theme

Bottom line

Thickness: 2dp

Color: 100% primary color (Dark A700)

Label text

Type: Roboto 12sp

Color: 87% primary color (Dark A700)

Spacing: 16dp top padding and 8dp bottom padding

Input text

Type: Roboto 16sp

Color: 87% black (15:1 contrast ratio)

Spacing: 8dp bottom padding, inset with bottom line

Helper text (optional)

Type: Roboto 12sp

Color: 54% black (4.5:1 contrast ratio)

Spacing: 8dp top padding, inset with bottom line

Dark theme

Bottom line

Thickness: 2dp

Color: 100% primary color (Light A200)

Label text

Type: Roboto 12sp

Color: 100% primary color (Light A200)

Spacing: 16dp top padding and 8dp bottom padding

Input text

Type: Roboto 16sp

Color: 100% white (13:1 contrast ratio)

Spacing: 8dp bottom padding, inset with bottom line

Helper text (optional)

Type: Roboto 12sp

Color: 70% white (7:1 contrast ratio)

Spacing: 8dp top padding, inset with bottom line

Text field input

Idle & filled

When the user shifts focus to another element, entered input text remains visible.

Light theme

Bottom line

Thickness: 1dp

Color: 42% black (3:1 contrast ratio)

Label text

Type: Roboto 12sp

Color: 54% black (4.5:1 contrast ratio)

Spacing: 16dp top padding and 8dp bottom padding

Input text

Type: Roboto 16sp

Color: 87% black (15:1 contrast ratio)

Spacing: 8dp bottom padding, inset with bottom line

Helper text (optional)

Type: Roboto 12sp

Color: 54% black (4.5:1 contrast ratio)

Spacing: 8dp top padding, inset with bottom line

Dark theme

Bottom line

Thickness: 1dp

Color: 70% white (7:1 contrast ratio)

Label text

Type: Roboto 12sp

Color: 70% white (7:1 contrast ratio)

Spacing: 16dp top padding and 8dp bottom padding

Input text

Type: Roboto 16sp

Color: 100% white (13:1 contrast ratio)

Spacing: 8dp bottom padding, inset with bottom line

Helper text (optional)

Type: Roboto 12sp

Color: 70% white (7:1 contrast ratio)

Spacing: 8dp top padding, inset with bottom line

Text field content

Error

If a text field is invalid, an error message is displayed when the user shifts focus to another element. To indicate an error, the input line and label use a color with a warmer hue, such as red or orange.

Specs

Display an error message below the text field, replacing any helper text

Briefly state the reason for the error and how to fix it

Place the word “Error” (or an error icon) before the error message

Remove the error once fixed by the user

Light theme

Bottom line

Thickness: 2dp

Color: 100% red (A400)

Label text

Type: Roboto 12sp

Color: 100% red (A400)

Spacing: 16dp top padding and 8dp bottom padding, inset with bottom line

Input text

Type: Roboto 16sp

Color: 87% black (15:1 contrast ratio)

Spacing: 16dp left padding, 8dp bottom padding

Error text

Type: Roboto 12sp

Color: 87% red (A400)

Spacing: 8dp top padding, inset with bottom line

Dark theme

Bottom line

Thickness: 2dp

Color: 100% red (A400)

Label text

Type: Roboto 12sp

Color: 100% red (A400)

Spacing: 16dp top padding and 8dp bottom padding, inset with bottom line

Input text

Type: Roboto 16sp

Color: 100% white (13:1 contrast ratio)

Spacing: 16dp left padding, 8dp bottom padding

Error text

Type: Roboto 12sp

Color: 100% red (A400)

Spacing: 8dp top padding, inset with bottom line

Field types

There are several types of text fields:

Single-line

Multi-line

Text area

Single-line fields

When the text entered is longer than the input line, it automatically scrolls left as the cursor reaches the right edge of the field.

Input text scrolls to the left as the cursor reaches the right end of the single-line text field.

alt-text

Single-line text fields

alt-text

Single-line text fields

Multi-line fields

Multi-line text fields wrap text onto a new line by expanding the bottom of the field, shifting screen elements downward.

Multi-line text fields allow users to see everything they input at once.

Input text wraps to a new line and makes the field taller as the cursor reaches the right end of the field.

alt-text

Multi-line text field

alt-text

Multi-line text field

Text area

Text areas are taller than text fields and wrap overflow text onto a new line. They scroll vertically when the cursor reaches the bottom of the field.

Input text scrolls up as the cursor reaches the bottom right of the text area.

alt-text

The message box is the text area

alt-text

The message box is the text area

Text areas should have a visible border, using the same color and thickness as the input line.

alt-text

Light and dark states for text areas

Input types

Formatted inputs

Text fields can be formatted to indicate the types of data a field accepts, using placeholder text, preset layout, and character limits. Text formatting should be displayed only when the field is pressed, focused, or filled.

Characters can be formatted using:

Grouped characters

Prefixes and suffixes

Password redaction

Grouped characters

Input text can be grouped in different ways, with special characters added in appropriate places, such as hyphens added to a phone number. Once the first special character is reached, the cursor automatically jumps to the next group.

alt-text

Input text grouped and formatted for a phone number

alt-text

Input text grouped and formatted for a credit card number

Prefixes & suffixes

Prefixes and suffixes can be used to clarify units or to add input in advance. Prefixes are left justified in the text field; suffixes are right justified. Text fields can have both prefixes and suffixes.

alt-text

Prefix for dollar currency

alt-text

Suffix for weight

alt-text

Suffix for email domain

alt-text

Suffix for time zone

Password redaction

Password input is disguised by default. Midline ellipses are displayed to represent each character of a password entered (such as •••••••).

When the visibility icon is displayed with a text field, it indicates whether or not that field’s input is visible. A text field’s visibility may be toggled on or off using the icon.

alt-text

The visibility icon is displayed with password characters visible when visibility is enabled.

alt-text

The invisibility icon is displayed with password characters hidden when visibility is disabled.

Other inputs

Menus and pickers

Other components such as dropdown menus and pickers can be nested in text fields to make input easier.

alt-text

Text field dropdown menu for selecting a US state

alt-text

Text field picker for selecting a date

Field variations

Draw additional focus to text fields using these style variations:

Labels as headings

Solo fields

Full-width fields

Labels as headings

When a text field is the primary action on a screen, its label can be removed or become a heading.

Placeholder text appears in the idle state and is replaced with input text as the user enters text.

alt-text

Label text can be removed from the field and displayed as a headline in onboarding or query flows.

alt-text

Label text can be removed from the field and displayed as a headline in dialogs.

Solo fields

When a text field is used for a single action such as search, the text field may be elevated, with associated icons placed within it. An input line is not required.

On desktop, a raised search button should appear to the right of the search field.

alt-text

Search action as a raised text field on desktop

alt-text

Search action as a raised text field on mobile

alt-text

The app bar acts as a text input field

alt-text

The app bar has a nested text input field

Full-width field

Full-width text fields are useful for in-depth tasks or entering complex information.

alt-text

Full-width text field

alt-text

Full-width text field with both single-line and multi-line text behavior

alt-text

Full-width field with normal spacing

Height: 56dp

Top padding: 20dp

Bottom padding: 20dp

alt-text

Full-width field with dense spacing

Height: 48dp

Top padding: 16dp

Bottom padding: 16dp

Text field boxes

Text field boxes increase text field identifiability and scannability by using a transparent rectangular fill to enclose the label and input text.

Text field boxes use color to increase the discoverability of the field, while preserving the input line.

States

A text field box uses a rectangular fill as a primary affordance, masking the ends of the input line. A ripple motion occurs in the hover and press states.

Text field states with a rectangular affordance

Rectangular fill

Height: 56dp

Corner radius: 4dp

Color (light theme): 6% black

Color (dark theme): 10% white

Bottom line

Thickness: 2dp

Masked by rounded corners of rectangle

Idle & empty

Left padding: 16dp

Right padding: 16dp

Padding above label: 8dp

Hover

Left padding: 16dp

Padding above label: 20dp

Padding below label: 20dp

Press

Padding above label: 8dp

Padding below label: 8dp

Padding below placeholder: 8dp

Spacing for focus, idle & filled, error, and disabled states are the same as the press state.

Dense layouts

Text field boxes make dense layouts easier to read.

Normal spacing

Height: 56dp

Padding above label: 8dp

Padding below label: 8dp

Padding below text field: 8dp

Padding above helper text: 8dp

Dense spacing

Height: 44dp

Padding above label: 8dp

Padding below label: 4dp

Padding below text field: 8dp

Padding above helper text: 4dp.

Single-line text field box

Text scrolling behavior in a single-line field

Single-line text field with label as heading

Multi-line text field box

Text wrapping and scrolling behavior in a multi-line field

Multi-line field in pressed state

Idle and filled multi-line field

Text box area

Text wrapping and scrolling behavior in the text box area

Empty and idle text area

Text area in pressed state

Menus & pickers

Dropdown list in a rectangular text field

Icon and date picker in a rectangular text field

Toolbars

Toolbar actions appear above the view affected by their actions.

Toolbars appear a step above the content they affect. They may constrain their width to accommodate material passing over them.

Do not separate a toolbar’s actions by another sheet of material, unless it is transient, such as a menu or dialog.

Elevation

Above app content

Scrolling

Affected content scrolls underneath toolbars.

Dismissed content

Sheets may temporarily cover the toolbar while moving offscreen.

Usage

Toolbars appear a step above the sheet of material affected by their actions. When sheets scroll underneath toolbars, they are clipped and cannot pass through to the opposite side.

Toolbars are located a step above another sheet.

Sheets can also cover the toolbar as they move offscreen.

The toolbar maintains its seam until it moves off of the screen.

Finally, the second sheet can cover the toolbar as it moves.

The second sheet covers the toolbar as it moves.

A toolbar’s left and right actions are never split by another sheet of material, with the exception of temporary materials such as menus or dialogs.

Instead, toolbars can constrain their width to make space for sheets of material passing over them. The actions on the left and right cluster to stay on the same side of a toolbar.

Do.

Constrain width to less than the full length of its material.

Don't.

Never allow a sheet of material to be split by another persistent sheet of material.

Toolbars can share a seam with another sheet.

In cases where they share a seam that transforms into a step, it is called a waterfall.

In the waterfall variant of clipping, the toolbar starts out as seamed and then lifts to form a step.

Tooltips

Tooltips are text labels that appear when the user hovers over, focuses on, or touches an element.

Tooltips identify an element when they are activated. They may contain brief helper text about its function. For example, they may contain text information about actionable icons.

Tooltip labels do not receive input focus.

Summoned by:

Hovering over an element with a cursor

Focusing on an element with a keyboard (usually the tab key)

Upon touch

Usage

Use tooltips for interactive imagery.

Do.

Don't.

Tooltips don’t display rich information including images and formatted text.

Tooltips are different than ALT-attributes, which are intended primarily for static images.

Tooltips don’t have directional arrows; instead, they rely on motion emanating from the source to convey direction.

Do.

Don't.

Interaction

A tooltip is triggered by tapping and holding an item. Keep the tooltip displayed as long as the user continues to hold the element.

Timing

On lift, display the tooltip for 1.5 seconds.

If the user takes another action before that time ends, the tooltip will disappear.

Motion details

The tooltip entrance occurs over 150ms, using the Deceleration Curve. It also exits over 150ms, using the Acceleration Curve.

Tooltips (desktop)

Text: Roboto Medium 10sp

Color: Grey 700

Opacity: 90%

Cursor/keyboard tooltips

Tile height: 22dp

Left and right text padding: 8dp

Top margin: 14dp

Example of cursor/keyboard tooltip

Tooltips (mobile)

Text: Roboto Medium 14sp

Color: Grey 700

Opacity: 90%

Touch UI tooltips

Tile height: 32dp

Left and right text padding: 16dp

Top margin: 24dp

Example of Touch UI tooltips

Example of Touch UI tooltips

Example of Touch UI tooltips in a normal state

Example of Touch UI tooltips in a press and hold state

Widgets

Widgets display glanceable views of an app's most important data and functionality.

Usage

Home screen widgets display your app’s new and interesting content in a consolidated form. They link to richer detail within the app.

Users can move and, if supported, resize widgets across their home screen panels.

App widgets

How to publish an app widget

Types of widgets

Information widgets

Information widgets display a few elements of importance to a user and track how that information changes over time, such as weather or sports scores. Tapping the widget launches the associated app into a detail view.

An information widget

Collection widgets

Collection widgets display multiple elements of the same type, such as a collection of articles from a news app. They focus on two interactions:

Browsing a collection

Opening an element’s detail view

Collection widgets can scroll vertically.

Collection widgets

Control widgets

Control widgets display frequently used functions. These functions may be triggered from the home screen without opening the app. For example, music app widgets allow the user to play, pause, or skip music tracks from outside the music app.

Control widgets may or may not progress to a detail view.

A control widget

Hybrid widgets

Many widgets are hybrids that combine elements of the different types above. Center your widget around one of these types and add elements of others as needed.

For example, a music player widget combines a control widget with elements of an information widget. The result keeps the user informed about which track is currently playing.

A hybrid widget

Behavior

Navigation

Your widgets should provide navigation links to frequently used areas of your app, including:

Functions that allow the user to create new content, such as a new document or message

Access to the top level of your app

Widget resizing

Resizing allows users to adjust the height or width of a widget. This allows users to influence the layout of widgets on home panels.

Your app may be completely resizable or constrained to horizontal or vertical size changes.

A long press and subsequent release sets resizable widgets into resize mode. Users can use the drag handles or the widget corners to set the desired size.

Scrollable widgets

List or grid-based collection widgets usually expand or contract the vertical scrolling area. Regardless of the widget's size, the user can still scroll all elements into view.

Determine how much of your app's information should surface. For smaller sizes concentrate on the essential and then add more contextual information as the widget grows.

Non-scrollable widgets

Information widgets are not scrollable. All content and layout must dynamically fit as sized by the user.

Responsive resizing

Widgets should accommodate different spacing requirements across devices, including cell number, size, and spacing variations.

Configuring widgets

Android widgets display their configuration choices once placed on a home panel.

Configuration best practices:

Keep configuration light and don't present more than 2-3 configuration elements

Present configuration choices using dialogs, rather than full-screen, to retain the user's context

Once setup, widgets do not typically show a "Setup" or "Configuration" button.

Patterns

Confirmation & acknowledgement

When a user invokes an action in your app, confirm or acknowledge that action through text.

Usage

Confirming and acknowledging actions can help alleviate uncertainty about things that have happened or will happen. It also prevents users from making mistakes they might regret.

Confirming involves asking the user to verify that they want to proceed with an action.

Acknowledging is displaying text to let the user know that the action they chose has been completed.

Not all actions warrant a confirmation or an acknowledgment.

Confirmation

A confirmation asks the user to verify that they truly want to proceed with the action they just invoked. It may be paired with a warning or critical information related to that action.

Modal dialog titles for confirmations should be meaningful and echo the requested action.

Confirmation isn’t necessary when the consequences of an action are reversible or negligible. For example, if a check mark shows an image has been selected, further confirmation is unnecessary.

The user has requested to delete an album from their library. An alert appears to confirm this action and inform the user that the album will no longer be available from any device.

After initiating Android Beam, the user is prompted to touch the content to be shared. If they decide not to proceed, they simply move their phone away.

Acknowledgement

Acknowledgement removes uncertainty about implicit operations that the system is taking. It may be paired with an option to undo the action.

The draft of an email is saved if the user navigates back or up from the compose screen. An acknowledgment in the form of a toast appears, then fades after a few seconds.

After the user deletes a conversation from the list view, an acknowledgment appears with an undo option. It remains until the user takes an unrelated action, such as scrolling the list.

Data formats

Different types of numeric and linguistic data are represented using data formats.

Date and time values may be formatted to depict ranges, time zones, abbreviations, and more, as appropriate to the context.

Data redaction and truncation guidelines describe how to represent truncated numbers and disguise sensitive data.

Formats represented

Time

Month, day, year

Approximating time

Absolute time

Time zones

Abbreviations

Rounded numbers

000 • • • 0000

12:00 UTC

Data truncation and time display examples

Date & time

Date and time

How you format the date and time depends on a few factors:

If the time or date is the current one

Whether you are using the 12-hour or 24-hour clock

How specific you need to be

Element

Description

Example

Time

Within the current day, display the time using uppercase AM or PM, without periods. If you are using the 24-hour clock, display the time without AM/PM.

Many non-English languages use lowercase am and pm.

2:00 PM

14:00

Month, day, and year

Within the current calendar year, display the date without the year. Otherwise, display the date and year.

January 14

14 January 2012

Approximate time

Approximate time rounds down to the largest and most recent date or time unit.

In 5 minutes

3 days ago

Absolute time

When approximate time isn’t appropriate, display the specific date and/or time.

Today, 10:00 AM

Time and date ranges

Date and time ranges are formatted based on the following:

The year is the same on both sides of the range

It’s the current year on both sides of the range

Whether both times have the same AM or PM

Range element

Description

Example

En dash

Separate a range of dates or times with an en dash, without spaces. Add spaces when spelling out months, or to remove ambiguity.

8:00 AM–12:30 PM

6 Jan – 2 Feb

Year

Show the year on both the start and end, unless both dates have the current year.

Dec 6, 2013–Jan 2, 2014

January 4–6, 2014

AM/PM

Use a single AM or PM at the end of the range, if both times have the same AM/PM.

8:00–10:30 AM

Time zones

As time zones vary across regions, time zone usage should be both clear and relevant.

Types of time zones

Description

Example

Single-digit hours

Remove the leading 0 for single digit hours.

UTC+5:00

Single times zones

Use two-letter time zones when referring to a single time zone. Remove the S (for Standard) or D (for Daylight).

Open 4–10 PM ET

Combination time zones

When specifying a date for a nationwide audience, include Standard or Daylight acronyms (such as EDT) to avoid confusion. Some U.S. states, such as Arizona, don't use daylight savings time.

The first game is on April 13, 2015 at 4 PM EDT.

Date and time modifications by context

Depending on the context, show either date or time, or both date and time.

Context

Description

Example

Future

Include time to a future day or date.

10 Jan, 08:00

Past

When referring to a past time, display both date and time.

Reminded Jan 5, 7:16 AM

Distant past

Omit the time for events in the distant past.

3 Jan

Weekday

When referring to a day of the week, such as for Calendar invites, display the abbreviated day separated by a comma.

Mon, Jan 10, 8:00 AM

Duration

Show the duration of a recording, like audio or video, in the format H:MM:SS. Omit hours or seconds if they don’t apply.

Use the same format across the same context. A video labelled “3:15” referring to hours and minutes, should not use that time format later on that page to refer to minutes and seconds.

0:30

1:01:05

Abbreviations

Where space is limited, use abbreviations for months and days.

Time period

Description

Example

Months

Months can be abbreviated or numeric. Many languages do not abbreviate month names.

Jan 6

3/12/12

Date format

The US date format is month/day/year, but other countries use day/month/year.

March 12, 2012 written as 03/12/12 could be misinterpreted to mean December 3, 2012 across different users.

Days of the week

Days of the week may be abbreviated using the first letter of each day.

S for Sunday

M for Monday

Day and time

Abbreviated days of the week may be combined with a time.

Sun, 2:00 PM

Sun, 14:00

Rounded numbers

Abbreviated numerical units, which remove the “:00,” are appropriate for timestamps, labels on graphs, durations, etc.

8 AM

2 hr 32 min ago

Talking about date and time

Display date and time information as people normally speak to each other, when space is available.

Refer to yesterday or tomorrow in those terms

If the day is in the upcoming week, display the day of the week (such as Tuesday)

Describe a time of day in other ways that promote understanding, such as, “Store open 9:00 AM–Midnight” or “Reminder for tomorrow afternoon”

Data redaction and truncation

Redacting numbers

Midline ellipses [• • •] are bulleted characters that partially hide sensitive data. They indicate when a number, such as a credit card or Social Security number, is not entirely visible.

When to use 3 ellipses

To redact most numbers, like Social Security numbers, use three midline ellipses [• • •].

You may use this format across typefaces and platforms:

1. Place two regular spaces before the first midline ellipse

2. Then add a thin space between each midline ellipse and after the final one.

Or, use the Unicode string instead: U0020,U0020,U2022,U2006,U2022,U2006,U2022,U2006

SSN • • • 5678

Do.

When to use 4 ellipses

Credit and debit card data must be redacted using four midline ellipses [• • • •].

Written in Unicode:

U0020,U0020,U2022,U2006,U2022,U2006,U2022,U2006,U2022,U2006

Visa • • • • 1234

Do.

Ellipses for credit card redaction

Redacting letters

Baseline ellipses [...] represent letters, words, or phrases that aren’t shown. They also represent when a name or email address is only partially visible.

Usage

Type the ellipses as three periods in a row (or Unicode character U2026) after the last letter shown.

PayPal krestovozdvizhensky@goog…

Do.

Redacted email address with three baseline ellipses.

Empty states

Empty states occur when an item’s content can’t be shown.

A list that doesn’t contain any items, or a search that doesn’t display any results, are examples of empty states. Although these states aren’t typical, they should be designed to prevent user confusion.

For content that cannot be displayed because of a system failure, see app errors.

Alternatives to empty states

Starter content

Educational content

Best match

Content for empty states

Text tagline

Non-interactive image

Displaying empty states

The most basic empty state displays a non-interactive image and a text tagline.

Use an image that:

Is subtle and neutral with respect to the background

Conveys the purpose and potential of the app in a lively way, such as your app's icon

Include a tagline that:

Has a positive tone

Is consistent with your brand

Conveys the purpose of the app without appearing to be actionable

Do.

The image is neutral and clearly part of the background. The tagline conveys the purpose of the app without coming across as a call to action.

Don't.

The image is bright and the tagline is worded like a call to action, which may encourage users to touch either the image or the tagline to start a hangout.

Avoiding completely empty states

There are several situations in which you can provide users with alternatives to truly empty states.

Starter content

The most compelling way for new users to learn and get excited about your app is by using it. Consider providing starter content that will allow users to explore your app right away.

Recommendations:

Use content that has broad appeal and demonstrates primary features.

Give users the ability to delete and replace this content.

If possible, provide content that's personalized.

A book reading app might provide all users with a few free popular books to immediately start exploring the app.

Educational content

If the purpose of the screen isn't easily conveyed through an image and a tagline, consider showing educational content instead.

Recommendations:

Help users understand what they'll be able to do on this screen once it has content.

Make it possible to dismiss or skip this content.

Keep it brief.

Before a user watches a video in the movies app, a dismissible card is shown to explain the features and benefits of the service.

Best match

If nothing exactly matches the user's query, are there any results for a query spelled slightly differently? If so, then show the results, as they may help a user find what they're after.

With this approach, clearly convey in a heading above the results that this content shouldn't be mistaken for a match to actual query results.

Example of best matches

Offering best matches is a great way to handle a misspelled query without explicitly placing blame on the user.

Errors

Errors occur when an app fails to complete an expected action.

Some examples of errors include:

When user input is not understood

An app failing to load

Incompatible operations are run concurrently

Types of errors

User input errors

App errors

Incompatible state errors

Specific error patterns

Forms

General usage errors

Sync errors

Connectivity

Permissions

Contents

Usage

User input errors

App errors

Incompatible state errors

Usage

Errors occur when an app fails to complete an action, such as:

The app does not understand user input

The system or app fails

A user intends to run incompatible operations concurrently

Minimize errors by designing apps that make it easy for users to input information flexibly. Apps should accept common data formats that use affordances to improve user understanding.

To address errors:

Clearly communicate what is happening

Describe how a user can resolve it

Preserve as much user-entered input as possible

User input errors

Help users fix input errors as soon as they are detected. Disable the submission of a form if errors are detected, and if detected only after form submission, clearly explain the error and how to fix it.

Text field input

Helper text may be included before, during, or after a user interacts with each field on a form.

Show error text only after user interaction with a field. If the user inputs incorrect data, helper text may transform into error text.

Minimize form text to the essentials. Not every text field needs helper and/or error text.

Specification:

Place 16dp of vertical space between text fields and the below error text.

Example of a text field input

Example of a text field input with error text

Form submission

The Submit button should be enabled by default.

If you are performing inline form validation, and the field with the error is clearly marked, the submit button may be disabled until the error is corrected.

Color

Error text should reflect your product’s color palette. It’s recommended that you use a contrasting color for error states, such as a warmer hue like red or orange.

Error text should be legible, with noticeable contrast against its background color.

Sites such as webaim.org offer color contrast checking tools.

Fonts for light backgrounds

Errors: Roboto Regular 12sp

Hint and helper text: #000000 with 38% opacity

Light theme for these states: normal with hint text, normal with error text, normal with helper text, and normal with helper as error text

Fonts for dark backgrounds

Errors: Roboto Regular 12sp

Hint and helper text: #FFFFFF with 50% opacity

Dark theme for these states: normal with hint text, normal with error text, normal with helper text, and normal with helper as error text

Helper and error text

Example with helper text

Example with error text

Example with helper text

Example with error text

Examples of normal with helper text and normal with helper and error text

Input and error text

Tile height: 64dp

Padding above input text: 16dp

Padding below input text: 8dp

Padding above and below error text: 4dp

Error with floating text label

Example of floating text label with input text

Example of floating text label with error text

Examples of normal with hint text/label and normal with input text, label and error

Specs for floating text label

Tile height: 80dp

Padding above label text: 8dp

Padding above and below input text: 8dp

Padding above and below error text: 4dp

Text field input - Over/under character or word count

A character counter may be displayed before, during, and after user interaction with a field. Consider not displaying the counter until the user approaches the character limit.

Counter font is Roboto Regular 12sp

Counter fields have 16dp of additional bottom padding

Single-line with character counter

Example of a single-line input label with character counter and user input text

Example of a single-line input label with character counter and error text

Multi-line with character counter

Example of a multi-line input label with character counter and user input text

Example of a multi-line input label with character counter and error text

Examples of these states: single-line input label with user input text, single-line input label with error text, single-line input label with user input text, and multi-line input label with error text

Text field input

Height of entire input area: 88dp

Single-line input label top padding: 16dp

Top and bottom padding for input text for a single-line field: 8dp

Padding above counter: 8dp

Incompatible values

Show errors for incompatible values during or after a user interacts with a text field.

If two or more fields have incompatible inputs:

In the text field, indicate a fix is needed. Add an error message below.

Display a message at the top of the form, or screen, summarizing the fixes needed and any additional explanation.

Errors detected upon form submission

Reload the form with consolidated error messages and scroll position at the top. Error messages for individual fields may be resolved as the user works through the form.

Example of a form

Example of errors detected after attempted form submission

Incomplete form

Empty form fields should be indicated by both the text field and error message below.

Display incomplete form errors to indicate a user has skipped a field after they have advanced through a form. If unable to detect user progress through the form, display an error after the user has attempted to submit the form.

Multiple errors before form submission

Individually label error messages as the user works through the form.

Example of a form

Example of an incomplete form

Single-line list error

Example of a single-line list error

App errors

App errors occur independent of user input.

General app error

When an error occurs, an app should display loading indicators until the error message appears.

Features not available may be indicated as disabled in the UI. For example, a button not may be displayed in a disabled state, paired with text explaining it is not available. Not every error requires a new component to pop up.

If possible, give your user an action that will help them address the error.

Alert dialog: app feedback about an error that is blocking normal operation

The snackbar contains app feedback about a peripheral error. Snackbars are transient. Don't use them for critical, persistent, or bulk errors.

Sync error/failure to load

When sync is down or content has failed to load, the user should be able to interact with as much of the rest of the app as possible.

Empty state for the screen/content failing to load

Container/component specific error with action

Connectivity

When connectivity is down, the user should be able to interact with as much of the rest of the app as possible.

If appropriate, present a link to help a user accomplish their task. Only offer links that you can actually support. For example, don't offer an option like "Try again" in cases where you can detect that the operation will fail.

Snackbar with action to retry

Empty state for a screen that is only available online

Incompatible state errors

Incompatible state errors occur when users attempt to run operations that conflict, such as making a call while in airplane mode or taking a screenshot from a restricted work account. Help prevent users from putting themselves into these situations by clearly communicating the states they are selecting and the implications for the rest of their experience. When these errors are triggered, do not imply that they are the user’s fault.

General incompatibility

Clarify the reason for and origination of the error.

For example, screenshots and premium features are not allowed while in a restricted mode.

Snackbar + special mode indicator

Offline by choice

Display an unobtrusive, persistent indicator when users are offline but try to do tasks that require being online.

Examples:

Placing a call while in airplane mode

Music availability while offline

Snackbar

Indicator that the device has been placed into airplane mode.

Permission requested

If your app requires user permission before proceeding with an action, include the permission request in the app flow instead of treating it as an error.

If permissions are necessary before the first run of an app, consider including them into your app’s first-run experience.

Examples:

An app’s permissions have changed.

In-app purchases have been disabled.

Dialog

Fingerprint

Android only

Fingerprint detection may be used to unlock a device, sign in to apps, and authenticate purchases using Google Play and some third-party apps.

To authenticate purchases using Fingerprint, display the Fingerprint authentication dialog.

Fingerprint is not as secure as a strong PIN or password. Authentication alternatives include a user’s account password, an app PIN, and device credentials.

When to use

Upon opening the app

During your app’s purchase flow

In your app settings

After enrollment

Icon

Fingerprint icon size: 24dp

Circle surrounding icon: 40dp

Enrollment

Before invoking Fingerprint in your app, you must get consent from the user that they want to use Fingerprint as an alternate way of authenticating themselves.

Possible times to invite users to try Fingerprint include:

Upon opening the app

During your app’s purchase flow

In your app settings

After enrollment

Upon opening the app

The user is offered the option to enroll their fingerprint right after logging in.

The user is offered the option to enroll their fingerprint right after creating their account.

During your app’s purchase flow

The user is offered the option to enroll their fingerprint right after making a purchase with an existing account.

In your app settings

The user is offered the option to enable Fingerprint as their lock within an app’s settings.

After enrollment

After enrollment, present the Fingerprint dialog whenever you need authentication from the user.

Authentication

Your app’s users can set up the option to log in or authenticate purchases using Fingerprint.

When a user action needs to be authenticated, display the Fingerprint authentication dialog in place of your login screen.

Title

Use the title to describe the action being performed, such as “Sign in.” The title should not be used to introduce Fingerprint.

Secondary text

Use the phase “Confirm fingerprint.” This wording maintains consistency with Android Settings.

You may combine the above text with the associated user action, such as “Confirm fingerprint to complete purchase.”

Confirm fingerprint dialog

Specifications for confirm fingerprint dialog:

Dialog left and right padding: 24dp

Dialog top padding: 24dp

Title bottom margin: 20dp

Subtitle bottom margin: 28dp

Fingerprint icon right margin: 16dp

Touch sensor height: 40dp

Touch sensor call to action top and bottom padding: 12dp

Button container height: 52dp

Button container bottom padding: 8dp

Button height: 36p

Button right and left padding: 8dp

Behavior

Default state

Ask the user to put their finger on the sensor.

Default state asking the user to put their finger on the sensor

Success state

Once the fingerprint is recognized, change the dialog to a success message and the image to the fingerprint icon with the check mark.

Specification:

Use your app’s primary color for the success text and icon, or Teal 500 (#009688)

Success states should avoid:

Displaying two success states simultaneously

Extensive transitions between default and success states

Success message stating that the fingerprint has been recognized

Failure state

Upon failure, provide a clear indication that the user’s fingerprint was not recognized and that they should try again, using the error icon either with or without a status message.

Specification:

Use your app’s color assigned to failure states, or Deep Orange 600 (#F4511E)

Failure states should avoid:

Extensive transitions between default and failure states

Error message stating the fingerprint was not recognized.

Exiting

Provide alternative ways to exit the dialog. At minimum, provide an affordance to close the Fingerprint dialog, such as a “Cancel” button.

Authentication alternatives

Fingerprint should not be the only way to authenticate. Provide alternative authentication methods, such as:

User’s account password

App PIN

Device credentials

Fingerprint icon

Usage and size

The Fingerprint icon should be displayed at the standard system icon size, 24dp, within a 40dp circle.

40dp circle

24dp system icon

Default icon

Users will be asked to look for this icon for places where they can use Fingerprint.

Specifications:

Circular background color: #607D8B

Works on white backgrounds

Default icon

Dialog box and default icon

Tinted icon

The circle surrounding the icon can be customized with a color that provides appropriate contrast against the lines of the Fingerprint icon.

Contrast and tinted icons

Don't.

Don’t use a multi-colored Fingerprint icon.

UIs with light backgrounds

Use the Fingerprint icon with a dark circle background.

Specifications:

Color: #FFFFFF

Opacity: 100%

Dark circle background with the Fingerprint icon for a light background

UIs with dark backgrounds

Use the Fingerprint icon with a light circle background.

Specifications:

Color: #000000

Opacity: 54%

Icon with no circular background

Apps requiring a more versatile Fingerprint icon may eliminate the circular background.

Icon with no circular background

Dialog box with icon with no circular background

Don't.

Don’t tilt or distort the fingerprint icon.

Reauthentication

Reauthentication applies when users need to confirm their identity before continuing.

Reauthentication uses:

To prevent unauthorized viewing of sensitive information

To guard against unauthorized purchases

When cryptographic keys expire and users are asked to reconfirm their identity

Reauthentication dialog

Alternatively, if a user forgets about Fingerprint authentication, you can ask users to use their backup password and remind them of the Fingerprint option next time.

Dialog asking user for their backup password

Gestures

Mobile only

Gestures include touch mechanics (what your fingers do on the screen) and touch activities (results of specific gestures).

A touch mechanic may cause different outcomes, depending on the context in which it is used. For example, a long press may select an element like a list item.

A touch activity may be achieved through combining multiple touch mechanics. For example, by using the touch mechanics of pinch open, double-touch, or double-touch and dragging, a user may zoom in on a view.

Types of drag, swipe, and fling activities

Scroll

Reveal upon scroll

Pan

Dismiss

Swipe to refresh

Edge swipe

Paging swipe

Overscroll collapse

Menu open

Tilt

Help & feedback

Help content providers answers to user’s questions and concerns.

Help & feedback

Help content providers answers to user’s questions and concerns.

Usage

Help content provides answers to common user questions about your app. Users can send comments, report bugs, and ask questions that are not already answered by the help content.

Placement

Make it easy for users to find help content in your app.

Help content can be made accessible from various points in your app’s navigation, with options listed below. It is usually placed in the navigation drawer (or overflow menu) under the label “Help” or “Send feedback.”

Complex apps

If your app is complex, place “Help” both in the navigation drawer and the app bar.

To make it easy for users to get help, particularly if your app is complex, place “Help” in both the navigation drawer and app bar. Otherwise, place “Help” in the navigation drawer only, or the overflow menu.

Navigation Drawer

Place “Help” as the last item in the navigation drawer, with “Send feedback” directly above it.

If “Sign out” appears in the navigation drawer, it should be the last item in the list.

The navigation drawer showing “Help” and “Send feedback”

Overflow menu

Place “Help” and “Feedback” (or “Send feedback”) in the overflow menu when there is no navigation drawer.

If “Sign out” appears in the overflow menu, it should be the last item in the list.

Do.

Overflow menu showing “Help” and “Send feedback”

App bar

To make it easier for users to find help for urgent issues, such as payments and refunds, place a Help icon in the app bar.

Desktop applications may also place a Help icon in the app bar, as there is more space in the desktop UI.

Help icon promoted in the app bar

Help screen

A “Send feedback” button may be included in the help screen.

Content viewed less often (such as “About,” “Privacy,” and “Terms of service”) may be placed in an action overflow menu.

A help screen with frequently asked questions and answers and an action button to send feedback.

Behavior

Relevant help topics

Help content should be relevant to the user’s current location in an app. For example, if a user is viewing their account information, the help content displayed should include information about accounts.

Immediate access

Take the user directly to help content upon selecting “Help” in your app.

Do.

Once a user selects “Help,” they should be taken directly to the “Help” menu.

Don't.

After a user has selected “Help,” do not present a screen asking the user to choose between “Help” and other options.

Unsolicited help

Occasionally, help content appears when users interact with an app for the first time, even though the user has not requested help. By offering unsolicited help, users don’t have to search the “Help” menu for answers on how to use new features. This kind of help often appears:

To promote feature discovery, when a feature can’t easily be activated from a menu or button

When a new gesture is introduced that wasn’t in the app before

This type of help should be limited to new features and gesture education, as it could be interruptive and distracting.

Icons

Help and Feedback icons are displayed at 24dp.

Tooltip icons are 18dp.

On light backgrounds, icons have an opacity of 54%. On dark backgrounds they have an opacity of 100%.

Light theme

Opacity

Active icon

54%

Inactive icon

38%

Dark theme

Opacity

Active icon

100%

Inactive icon

50%

Help

Do.

Use the standard material icon for Help.

Don't.

Don’t use icon variations, such as those with speech bubbles or circle outlines.

Send feedback

Do.

Use the standard material icon for Send feedback.

Don't.

Don’t use icon variations, such as chat or bug icons.

Help Tooltips

Do.

Use the standard material icon for Help Tooltips.

Don't.

Don’t use icon variations, including the Help icon, or icons with speech bubbles or outlines.

Writing

Help content lets users find answers to questions or problems that are unique to your app.

Consider these guidelines when writing help content:

Give key information

Keep explanations as short as possible. Avoid giving details that aren’t relevant to typical usage. Answer one question or concern at a time.

Make it easy to read

Make help content easy to read by formatting text with bold headings, lists, tables, and space between paragraphs as needed.

In particular, when referring to elements that users need to select, such as buttons or links, bold the label names when referring to them in help content.

Headings clearly outline an article’s content. They help readers scan the page to quickly find the information they need.

Use simple language

Avoid using technical terms in help content where possible.

Show images

When providing step-by-step instructions, show relevant images or icons to explain what the user needs to do.

Launch screens

The launch screen is a user’s first experience of your application.

A placeholder UI displays core structural elements such as the status and app bars until the app has loaded.

Branded launch screens display your logo or other elements that improve brand recognition.

Usage

A placeholder UI is best for apps with very short load times.

Types of launch screens

Because launching your app while displaying a blank canvas increases its perceived loading time, consider using a placeholder UI or a branded launch screen.

A placeholder UI is the most seamless launch transition – appropriate for both app launches and in-app activity transitions.

Branded launch screens provide momentary brand exposure, freeing the UI to focus on content.

Branded launch screen

Placeholder UI

Placeholder UI

For apps that have very short load times, or don’t have a strong branding need, a placeholder UI presents the fewest possible transitions and greatest perceived responsiveness.

On launch, core structural elements such as the status bar, app bar, and bottom sheets are displayed without content until the app has loaded.

Example of placeholder UI

Branded launch

Branded launch screens are a good way to portray a brand while the app is loading. Take advantage of the loading time to display your logo or other elements that improve brand recognition. Avoid using text on this screen except your logo and, if applicable, a tagline.

Branded launch

Launch screens should be used for initial, cold launch from the home screen, and should not be displayed if the application is running, or if the launch comes from another application.

Examples of branded launch screens

Loading images

Images, including illustrations and photographs, may load in three phases.

Illustrations and photographs may load and transition in three phases by staggering opacity, exposure, and saturation levels.

Behavior

Progressive fade-in

Duration

Animation

Duration

Use longer durations for loading images, and shorter durations for transitions.

Usage

Illustrations and photographs may load and transition in three phases at staggered durations, rather than relying on opacity changes alone. You may combine image opacity, exposure, and saturation changes to suit your needs.

Exposure adjustments

Images should begin loading with low contrast levels and desaturated color. Once image opacity reaches 100%, display the image with full-color saturation.

Opacity, exposure, and saturation recommendations for loading images

Opacity: 0% start, 100% end

Exposure: Gamma, Black output (for darker images)

Saturation: 0 or 20% start, 100% end

Full opacity and exposure

Color saturation

Larger screens

This process is ideal for larger screens, such as loading the Chrome OS wallpaper.

Animation showing loading on larger screens for Chrome OS wallpaper

Behavior

Progressive fade-in

Visualize the image fading in, like a print during the photo development process.

Don’t adjust the levels in a way that will overexpose the lighter values.

Do.

Don't.

Duration

A longer duration is recommended for loading images, and a shorter duration is recommended for transitions.

Load and transition animation

Animation

Add a small position shift to loading images. The background image shown here in the account switcher animates in horizontally.

Video showing adding animation

Navigation

Navigation guides users through different parts of your app.

Organize your app's structure according to the content and tasks you want users to see. Focus attention on important destinations by displaying them in tabs or in the side navigation, and de-emphasize inessential content by displaying it in less prominent locations.

Navigation patterns

Embedded navigation

Tabs

Bottom navigation bar

Navigation drawer

Nested navigation

Expanding navigation drawer

Cascading navigation drawer (desktop)

Gestural

Combination navigation patterns

In-context navigation

Side navigation and tabs

Nested navigation drawer and tabs (desktop)

Usage

Navigation through your app should be intuitive and predictable. Both new and returning users should be able to figure out how to move through your app with ease. When users move from one view to the next, they observe scenes containing imagery, actions, and content. These scenes work together to tell a story about the content they contain, leading users down paths that emphasize certain actions.

For information on movements between states, see navigational transitions.

Implementing Effective Navigation

Communicate relationships & context

Group content into logical and relatable chunks that show clear relationships between items.

Illuminate actions

Guide users from one scene to the next using paths that encourage specific choices or freer exploration.

Focus attention

Tailor your app’s navigation to promote important content and tasks.

This image shows relationships between email items by grouping them into these clusters: Promos, Social, Finance.

Use simple design and lightweight structure when user needs are simple.

Defining your navigation

Navigation organizes content so that it’s easier to find important destinations in your app. Navigation may include frequently visited places, contain settings, or encourage specific actions.

To determine the type of navigation best suited to your app, identify your app’s users, typical paths they might take through your app, and actions you want them to perform.

For instance, if your app is a restaurant guide, your users may want to make reservations, post menus, or write reviews. By identifying the goals most users will have, you’ll be able to better define a navigation structure that meets their priorities.

Take inventory

Identify your app’s users and their potential roles, such as consumer, business owner, or journalist. Identify the most common tasks they may want to perform.

Identify your key users and the use cases they might want to perform.

Prioritize

Assign priority levels of high, medium, or low to common user tasks. Give prominence in the UI to paths and destinations with high priority levels and frequent use.

Adjust priority levels as user needs change over time.

When designing navigation for a restaurant app, the highest priority tasks may include viewing restaurant details or finding new restaurants.

Sequence

Identify the different paths that users take through your app and use those paths to define your navigation:

List frequent destinations prominently in your navigation

Group related tasks together and use those groupings to structure your navigation

Whether a consumer wants to find a new restaurant or view a favorite one, both paths lead viewing restaurant details. Make frequently used paths to these destinations (such as search or favorites) easily available.

Deconstruct

Divide complex, broad, or vague use cases into smaller activities. These smaller tasks may be frequently used, more easily understood, or better meet user goals.

For example, dividing search into smaller activities that differentiate searching by name, location, or popularity may reveal the need to surface those smaller activities as part of navigation.

The broader activity of “Find a restaurant” may actually encapsulate smaller activities, such as: Browse nearby, Search by name, and Browse popular. These smaller tasks may be easier for users to perform searches and should be integrated within your navigation.

Up and Back buttons

App bars provide basic navigation through the Up and Back buttons.

The Up button

The Up button returns users to the previous screen they viewed. It navigates upward in the app’s hierarchy until the app’s home screen is reached.

The Up button

The Up button in a photo gallery

The Back button

The Back button navigates in reverse chronological order through the history of recently viewed screens.

Whereas the Up button ensures the user remains in your app, the Back button may take the user back through recent screens outside of your app.

The Back button also:

Dismisses floating windows (such as dialogs or popups)

Dismisses contextual action bars, and removes the highlight from the selected items

Hides the on-screen keyboard (IME)

Tasks and Back Stack

Activities performed by users are arranged in a stack in the order in which each activity is opened

Make navigation more predictable by inserting the navigation path the user took to the app's topmost screen

The Back button

The Back button in a photo gallery

Changing views within a screen

Navigation history is not affected by these types of screen changes:

Switching views using tabs, swiping, or a dropdown

Filtering or sorting a list

Changing display characteristics (such as zooming)

Exception: Navigation history is created by following links from related detail views (not unified by a list), such as browsing Play Store albums by the same artist.

Navigating between apps (Android)

Navigate directly from one app to another to create a consistent experience for common actions. For example, an app may activate the Camera app to capture a photo.

Activities, tasks, and intents

In Android, an activity includes an app’s screen and the associated actions available to the user. Your app is a collection of activities you create and reuse from other apps.

A task is the sequence of activities a user follows to accomplish a goal.

An intent allows an app to signal it would like another app's assistance in performing an action, such as “Share.” Apps can indicate which intents to respond to through activities.

Launching an app from the Home screen begins a new task. This task is extended to include activities of navigating through albums and viewing photo details. If the user triggers the Share action, a bottom sheet is revealed with available sharing activities from different apps (all of which have been registered to handle the Share intent).

Hierarchy

Navigation organizes content by placing it within a hierarchy. Scenes at the top of the hierarchy are the most prominent and accessible, connecting users to scenes lower in the hierarchy.

Home

The home scene is the entrance to the app. It introduces the app and its navigation.

The home scene may vary the content it displays. A stateful home contains indications of a user’s previous interaction with the app.

Parent and child

A higher level of navigational hierarchy is called a parent, and the level (or levels) below are referred to as children.

For example, the home screen is the parent to all other screens in an app.

Navigation

When you navigate more deeply into an app, you descend from a higher level of hierarchy to a lower one.

Moving from a parent scene to a child scene is descending navigation. Moving from child to parent is ascending navigation.

Siblings

Scenes that have the same parent are called siblings. Apps that have multiple use cases, all of equal priority, usually contain many sibling scenes.

Lateral navigation is movement between siblings.

Collections

Collections include multiple items that share the same parent.

Links

Links allow users to move quickly between scenes that aren’t navigationally adjacent.

Crosslinks are links that take users anywhere within your app.

External links are links from sources outside your app, including notifications and home screen widgets. They take users directly to scenes deep within your app.

Patterns

Navigation should best accommodate the needs of the majority of your app’s users. Apps with complex and deep hierarchies (that require switching between multiple views) have different navigation requirements than apps with a single primary view.

The following navigation patterns are designed to:

Promote engaging and responsive layouts

Anchor navigation and actions

Highlight important destinations and de-emphasize infrequent ones

Promote a primary action with the floating action button

Embedded navigation

Apps with simple navigation may embed navigation inside app content, but doing so reduces the available space to display that content.

Recommended for:

Apps with a strong primary view, and few alternate views

Apps that perform common tasks in the main view

Infrequently used apps

This is a mobile example of an app with embedded navigation. Common tasks are performed in a strong primary view.

Tabs

Tabs allow users to quickly move between a small number of equally important views.

Appropriate for these hierarchies:

Parents with embedded child views

A group of sibling views

Recommended for:

Frequent switching between views

Apps with few top-level views

Promoting awareness of alternate views

Mobile example with three tabs

Tablet example of centered tabs

Desktop example of left-aligned fixed tabs with a title and extended header

Bottom navigation bar

A bottom navigation bar allows users to quickly move between a small number of top-level views.

Appropriate for these hierarchies:

Parents with embedded child views

A group of parent views

Recommended for:

Frequent switching between views

Apps with few top-level views

Promoting awareness of alternate views

Mobile devices, as bottom navigation is located in a more ergonomic location

Mobile example with fixed text labels present at all times

Tablet example with text labels present at all times

Mobile example with shifting text and labels

Tablet example with shifting text and labels

Navigation drawer

When there is insufficient space to support tabs, side navigation is a good alternative. Side navigation can display many navigation targets at once. A drawer remains hidden until invoked by the user.

Apps with a single “home” should list the most frequently accessed destinations at the top of side navigation.

Side navigation may appear either with or without a navigation drawer.

Appropriate for these hierarchies:

Lateral navigation

Parents with siblings or peers

Recommended for:

Apps with many top-level views

Enabling quick navigation between unrelated views

Deep navigation structures

Reducing visibility of infrequent destinations

Mobile example of a closed left nav

Tablet example of an open left nav

Mobile example of an open left nav

Desktop example of an open and pinned-in-place left nav

Nested Navigation

When you have multiple levels of navigation, sibling views should be nested underneath their parent.

On desktop, a secondary level of navigation may be nested within the navigation drawer.

Appropriate for these hierarchies:

Lateral navigation

Parents with siblings or peers

Recommended for:

Deep navigation structures with many views

Enabling quick navigation between unrelated views

Desktop example of an expanding left nav

Expanding navigation drawer

If you have a deep navigational hierarchy, you may expand that hierarchy within the navigation drawer. Upon selecting a level, the level of navigation below is revealed. Selecting a collapsed section expands that level in-line, hiding all levels outside of it.

Desktop example of an expanding left nav

Cascading navigation drawer (Desktop only)

If you have a deep navigational hierarchy, you may use cascading menus to expand navigation drawer content horizontally. Each collection of views is presented in its own panel, and a level closes when a subsection is selected.

Desktop example of a cascading left nav

Gestural

Gestural navigation allows users to use swipe gestures to navigate between sibling or peer views. Supported gestures include: touching and dragging the screen horizontally (left/right), vertically (up/down), or while zooming in or out.

Recommended for:

Naturally ordered relationships, such as pages representing consecutive calendar days

Views with few siblings

Views with similar content types

See Navigational transitions for more information on hierarchical journeys between scenes.

When in Inbox, you are able to pull down on a message to return back to the parent view.

Combined patterns

In-context navigation

In-context navigation (used with tabs or navigation drawers) allows nimble movement between related sets of data.

In-context navigation integrates into an app’s content and is typically less linear in nature. When moving through in-context navigation, navigating to a new scene will display navigational controls specific to that view.

Appropriate for these hierarchies:

Parents with siblings

Recommended for:

Showing large sets of child scenes

Focusing attention on a specific set of tasks

Apps with a strong primary view, and few alternates

Example uses:

Link a song to an artist

Move between recent items and a complete history

Connect a user’s post to their profile page

The up arrow is used to return to the previous level.

Mobile example of navigation integrated into content

Mobile example of an up arrow

Tablet/desktop example of navigation integrated into content

Tablet/desktop example of an up arrow

Side nav and tab combinations

Products with two levels of navigation may pair a left navigation drawer with tabs.

Mobile example using two levels of navigation: a navigation drawer and tabs

Mobile example of an open left nav

Tablet example using two levels of navigation: a navigation drawer and tabs

Tablet example of an open left nav in front of tabs

Desktop example using two levels of navigation: a navigation drawer and tabs

Nested navigation drawer + tabs (Desktop only)

If your product has three levels of navigation, pair a navigation drawer with nested navigation and tabs.

Desktop example of a nested left nav with tabs

Navigation drawer

The navigation drawer slides in from the left and contains the navigation destinations for your app.

The nav drawer spans the height of the screen, with everything behind it visible but darkened by a scrim.

The navigation drawer follows the keylines and metrics for lists.

Resting elevation

16dp

Navigation drawer types

Permanent

Persistent

Mini variant

Temporary

Specs

Typography

Name: Roboto Medium, 14sp, #FFFFFF

Email address: Roboto Regular, 14sp, #FFFFFF

List item: Roboto Medium, 14sp, 87% #000000

Subheader: Roboto Medium, 14sp, 54% #000000. It aligns to the 16dp keyline.

Keylines and margins

Icons align at screen left and right margins: 16dp

Icon values: 54% #000000

Content associated with an icon or avatar left margin: 72dp

Side nav width: Equal to the screen width minus the height of the action bar. In the example shown above, the nav drawer is 56dp from the right edge of the screen.

Maximum width: The maximum width of the nav drawer is 280dp on mobile and 320dp on tablet. This is calculated by multiplying the standard increment by five (the standard increment is 56dp on mobile and 64dp on tablet).

Use 16dp horizontal margins on mobile and 24dp on tablet

Vertical spacing

Status bar: 24dp

Subtitle: 56dp

Space between content areas: 8dp

Subtitles and list items: 48dp

Add 8dp padding at the top and bottom of every list grouping. One exception is at the top of a list with a subheader, because subheaders contain their own padding.

Content

Elevation

The nav drawer spans the full height of the screen, including behind the status bar, at a resting elevation of 16dp. Everything behind the drawer is still visible but darkened by a scrim.

Nav drawer on Android

Selection state

After a list item is selected, that item changes color to match the app’s primary color (or #000000 100%) to indicate selection. Additionally, the touch ripple highlights the row of that list item.

If the color of the touch ripple/highlight doesn’t contrast enough with your primary color, use a darker tint of the primary color.

List item selected

Touch ripple/highlight

Touch ripple/highlight

Touch ripple/highlight

Touch ripple/highlight

Touch ripple/highlight

Dividers

All dividers in the nav drawer are full-bleed within the drawer with 8dp padding above and below each divider.

Example of a divider

8dp vertical spacing

Scrolling

The navigation drawer scrolls in the same way a view scrolls.

Navigation drawer before scrolling

Navigation drawer during scrolling

Settings and support

Settings and support are located at the bottom of the scrolling list, inline with the rest of the list content. They refer to Help, Feedback, or Help & feedback, depending on what your product offers.

Settings and support are located at the bottom of the scrolling list.

Behavior

Permanent

Recommended default for desktop

Permanent navigation drawers are always visible and pinned to the left edge, at the same elevation as the content or background. They cannot be closed.

The threshold for pinning is calculated using these minimum values:

1. Side nav width

2. Content width

3. Content padding

Types of permanent navigation drawers

The structure and behavior of the overall interface determines which type of permanent navigation drawer to use:

Full-height navigation drawer: Apps focused on information consumption that use a left-to-right hierarchy

Navigation drawer clipped under the app bar: Apps focused on productivity that require balance across the screen

Floating navigation drawer: Apps that require less hierarchy

Example of a permanent, full-height drawer

Example of a permanent, clipped drawer

Example of a permanent, floating drawer

Example of a permanent, floating drawer in card

Persistent

Persistent navigation drawers can toggle open or closed. The drawer sits on the same surface elevation as the content. It is closed by default and opens by selecting the menu icon, and stays open until closed by the user. The state of the drawer is remembered from action to action and session to session.

When the drawer is outside of the page grid and opens, the drawer forces other content to change size and adapt to the smaller viewport.

Persistent navigation drawers are acceptable for all sizes larger than mobile.

Example of a closed persistent drawer

Example of an open persistent drawer

Persistent navigation drawers are not recommended for apps with multiple levels of hierarchy that require using an up arrow for navigation.

Don't.

Mini variant

In this variation, the persistent navigation drawer changes its width. Its resting state is as a mini-drawer at the same elevation as the content, clipped by the app bar. When expanded, it appears as the standard persistent navigation drawer.

Recommended for:

Apps sections that need quick selection access alongside content

Example of a mini drawer

Example of open persistent drawer

Temporary

Temporary navigation drawers can toggle open or closed. Closed by default, the drawer opens temporarily above all other content until a section is selected.

Recommended for:

Tablet

Required for:

Mobile

Example of a closed temporary drawer on mobile

Example of an open temporary drawer on mobile

The mini variant is also an acceptable closed state for a temporary navigation drawer.

Example of a closed temporary drawer on tablet

Example of an open temporary drawer on tablet

Example of a closed temporary drawer on desktop

Example of an open temporary drawer on desktop

Navigational transitions

Navigational transitions are movements between states in an app, such as from a high-level view to a detailed view.

Most, but not all, transitions are hierarchical in nature. These moments should appropriately reflect the user’s journey through each state of an app.

Hierarchy and elevation

When elements lift up and expand, the elevation change indicates a shift in focus from parent to child elements.

Sibling transitions occur between elements at the same level of hierarchy without elevation changes.

Parent to child

Exploring deeper levels, or screens, of an app is a hierarchical journey that starts at a parent screen. From there, a user can explore multiple possible sub-screens, which are children to the parent screen.

In material design, elevation changes indicate changes in focus from parent to child elements.

Elevation and expansion

The surface that the user touches should lift up and expand into place from its origin. This expansion and motion highlights movement away from the parent towards a destination (a child element) in a natural movement using the material motion curve.

Parent-to-child transition animation

Animation of parent-to-child transition

Sibling to sibling

Sibling transitions are transitions that occur between elements at the same level of hierarchy.

Static elevation

When a user navigates through tabs, for example, no elevation changes occur. The content and surface of each tab stays at the same elevation level. New content slides in from the right while sibling content moves off-screen to the left.

Sibling transition

Animation of sibling-to-sibling transition

Notifications

Android only

Notifications provide short, timely, and relevant information about your app when it’s not in use.

Notifications are intended to inform users about events in your app. These two types of notifications are the most effective:

Communication from other users

Well-timed and informative task reminders

Notification components

Header area

Content area

Action rail

How notifications may be noticed

Showing a status bar icon

Appearing on the lock screen

Pulsing the device's LED

Playing a sound or vibrating

Peeking onto the current screen

Anatomy of a notification

Android N highlights

In Android Nougat (Android N), notifications have been designed to make it easier for users to scan and use a notification’s most important elements:

Primary content

Content is the most prominent element of a notification. Secondary information, such as a timestamp, is smaller and consolidated above the primary content.

People

Avatars appear on the right side of a notification.

Actions

Expandable notifications are revealed by tapping an indicator icon. Actions are displayed using text only (which accommodates more characters) on a separate background color and location.

Collapsed notification on Android M (above) and Android N (below)

Collapsed notification on Android M (top)

The avatar and icon are on the left and the timestamp is on the right.

Collapsed notification on Android N (bottom)

The app icon, app name, timestamp, and expansion indicator icon appear on the top left. The avatar is on the right.

Expanded notification on Android M (above) and Android N (below)

Header area

New for Android N, the header area is comprised of:

1. The app icon: The app icon is a small two-dimensional representation of your app's identity. It appears in monochrome in the status bar. If your app sends a wide variety of notifications, you may replace your app's identity icon with a symbol that reflects the content type. For example, Google Now uses a cloud icon for weather notifications.

2.The app name: Starting in Android N, the name of the app automatically appears in the notification.

3.Header text (optional): Header text is usually only needed if an app sends notifications from multiple sources, such as the account name for users with multiple accounts.

4. Timestamp (optional): By default, a timestamp does not appear, but it may be added if it's important to display when a notification was sent, such as the time of a missed call.

5. Expand indicator: This appears if the notification can be expanded.

The header area is new for Android N.

Content area

The content area is comprised of:

1. A content title: A brief headline for the notification

2. Content text: Supporting information

3. Large icon (optional): An image may be added to reinforce the notification in a meaningful way, such as a message that includes a photo of the sender

Android N's notifications are wider, resulting in a less cluttered presentation of the main content.

Action area

When expanded, a notification may display up to three actions on a strip at the bottom.

Starting in Android N, actions are shown without icons in order to accommodate more text. An icon should still be provided because devices with earlier versions of the OS continue to rely on it, as will Android Wear and Android Auto devices.

The action area in an Android N notification.

Usage

When not to use a notification

Notifications should not be the primary communication channel with your users, as frequent interruptions may cause irritation. The following cases do not warrant notification:

Cross-promotion, or advertising another product within a notification, which is strictly prohibited by the Play Store

An app that a user has never opened

Messages that encourage the user to return to an app, but provide no direct value, such as "Haven't seen you in a while"

Requests to rate an app

Operations that don't require user involvement, like syncing information

Error states from which the app may recover without user interaction

Don't.

Don’t send holiday or birthday greetings in the form of notifications.

Don't.

Don’t interrupt the user in the middle of a task for the sole purpose of asking if you're doing a good job.

Behavior

Notification arrival

When a notification arrives, it gets added to the notification drawer. You may specify if its delivery includes:

Making a sound or vibration

Peeking onto the current screen to grab the user's attention

Users may choose to block notifications. Starting in Android N, they may also prevent an app’s notifications from making sounds, vibrating, or peeking.

When a notification arrives, an icon usually appears in the status bar. This signals to the user that there's something to see in the notification drawer.

If it's important to grab a user's attention in the middle of a task, an app can have the notification peek onto the current screen.

Notification drawer

The notification drawer in Android typically shows notifications in reverse-chronological order, with adjustments influenced by:

The app's stated notification priority

Whether the notification recently alerted the user with a sound or vibration

Any people attached to the notification and whether they are starred contacts

Stale notifications

The notification drawer is intended to show the user information relevant to this very moment in time. If a notification you sent earlier is no longer relevant, automatically dismiss it.

In this notification drawer, a text that just arrived appears at the top. A lower priority notification about the weather appears at the bottom.

Notification interactions

Users may interact with a notification in the following ways:

1. To navigate to a destination

To navigate, a user may tap a notification. If the notification appears on a locked screen, the user will need to double-tap it and then enter their PIN, pattern, or password.

When the user taps a notification, they should be taken to a screen in your app that relates directly to that notification and lets them take immediate action. For example, if the notification says it's their turn in a two-player game, tapping the notification should take them directly to that game.

2. To see an expanded view

If offered, an expand indicator appears in the header. A user may tap the indicator or swipe down the notification body to expand it.

An expanded notification

3. To dismiss it (if permitted)

A user may dismiss a notification by swiping it left or right.

Ongoing notifications that indicate a continuing process in the background, such as music playing, may not be dismissed with a swipe.

A notification being dismissed

4. To choose how similar notifications should be delivered in the future

A user may adjust how notifications for an app are delivered by touching and holding an individual notification. In Android N, the user may also gently swipe the notification left or right and then tap the settings icon to access these controls.

Notification settings

Multiple notifications

For apps that may generate multiple notifications of the same type, Android offers two different approaches to represent them: summarizing and bundling.

Summarizing

Instead of displaying multiple notifications, you can create one notification that summarizes them all. For example, a messaging app might have a summary notification that says "3 new messages.” Upon expansion, it could show a snippet for each message.

Multiple notifications summarized, shown in both collapsed view and expanded view.

Bundling

Starting in Android N, you may present multiple notifications from your app in a grouped and hierarchical manner:

A parent notification displays summary information for all child notifications.

If the user expands the parent notification, all child notifications are revealed.

The user may expand a child notification to reveal its entire content.

When child notifications are presented, the system automatically removes any redundant header information. For example, if a child notification's app icon is the same as the parent notification’s app icon, then the child notification's header does not include an icon.

Child notifications should be able to stand on their own because the system may choose to show them outside of the bundle when they arrive.

Bundled notifications, shown in both collapsed and expanded views

Expanded views

You may display more information in a notification through an expanded view, without navigating away from the notification.

Collapsed and expanded notifications

A notification may offer up to three actions when expanded. It should not duplicate the action taken from tapping on the notification body.

Don't.

Don’t include text actions that duplicate the behavior of tapping on the notification body.

Do.

When Android's Clock app has a timer running, this notification lets the user pause or add a minute directly from the notification.

Enable typing into notifications

In Android N, apps may enable the user to type directly into a notification. Users may type small amounts of text, such replying to a text message or jotting a brief note.

For long-form typing, navigate users to your app, where they benefit from more space to view and edit text.

If you're using this functionality for a messaging app, keep the notification around after the user has sent the reply, and wait until the conversation has reached a lull before automatically dismissing it.

Users of Android's Messenger app can reply directly to any message without leaving the notification

Types of notifications

Notifications are considered either transactional or non-transactional.

Transactional

Transactional notifications provide content that a user must receive at a specific time in order to do one of the following:

Enable human-to-human interaction

Function better in daily life

Control or resolve transient device states

Non-transactional

If none of the above situations describe your notification, then it is non-transactional.

Transactional notifications can help users...

Examples

Enable human-to-human interaction

Incoming phone calls or text messages

Notification of a user's turn in a two-player game

Function better in daily life

Calendar event about to take place

A reminder or alarm set up by the user

Delayed flight

Delivered order

Control or resolve temporary device states

Music playing

Turn-by-turn navigation

Stopwatch running

Screenshot taken

Non-transactional opt-out and opt-in

Non-transactional notifications should be optional, as they may not appeal to all users. You can make them optional in one of two ways:

Opt-out: Users receive opt-out notifications by default, but they may stop receiving them by turning off a setting.

Opt-in: Users only receive opt-in notifications by turning on a setting in your app.

Opt-out

Avoid the opt-out approach unless your notifications meet both of the following criteria:

They provide concrete value to the user

They clearly relate to the user's context (such as current location, current date or time, past history, or expressed interests)

Example notification

Reason to avoid using opt-out notifications here

A random tip on how to use an app

Provides concrete user value but isn't contextual

A prompt to rate content recently used in the app

Contextual, but doesn't provide concrete user value

This notification from YouTube takes the opt-out approach. It offers both context (a YouTube channel the user has subscribed to) and value (a new video from that channel). The "Options" action navigates the user to where they can opt out of future notifications for this channel.

Opt-in

The opt-in approach is more conservative. Because users explicitly choose to receive these notifications, it's more likely they'll be glad to see them. However, they must visit settings to find out how to receive them. You may inform users about these notifications from other places in your app, such as an onboarding flow or a dismissible card.

This app takes the opt-in approach. A card at the top of the app lets users know that they can receive notifications for breaking news stories. If the user wants to opt in, they choose Yes. Otherwise, they choose No Thanks and remain opted out. This choice is also available in the app's settings.

Settings and priority

Notification settings

Include a setting in your app for turning all notifications on or off. Additionally, consider offering granular settings, such as letting users choose notification sounds or receiving specific notification types.

Enable users to navigate directly to notification settings from the Android Settings app by creating a link there using the appropriate AndroidManifest tags.

Assign appropriate priority levels

Android allows you to assign a priority level to each notification to influence how the Android system will deliver it. The higher the priority, the more interruptive it will be — for example, MAX and HIGH priority notifications will peek onto the user's current screen, no matter what they are doing.

Prioritize with the utmost consideration for the user's time and attention. When an unimportant notification is disguised as urgent, it tricks users into unnecessary anxiety.

To give users relief from overly interruptive notifications, users can quickly silence or block an app's notifications directly on the notification itself.

Priority

Use

MAX

For critical and urgent notifications that alert the user to a condition that is time-critical or needs to be resolved before they can continue with a time-critical task.

HIGH

Primarily for important communication, such as messages or chats.

DEFAULT

For all notifications that don't fall into any of the other priorities described here.

LOW

For information and events that are valuable and/or contextually relevant to the user, but aren't urgent or time-critical.

MIN

For nice-to-know background information that isn't aimed specifically at the user. For example, weather or nearby places of interest.

Pre-defined categories

Assign your notification to the most suitable predefined category. Android may use this information to make ranking and filtering decisions.

Category

Description

CATEGORY\_CALL

Incoming call (voice or video) or similar synchronous communication request

CATEGORY\_MESSAGE

Incoming direct message (SMS, instant message, etc.)

CATEGORY\_EMAIL

Asynchronous bulk message (email)

CATEGORY\_EVENT

Calendar event

CATEGORY\_PROMO

Promotion or advertisement

CATEGORY\_ALARM

Alarm or timer

CATEGORY\_PROGRESS

Progress of a long-running background operation

CATEGORY\_SOCIAL

Social network or sharing update

CATEGORY\_ERROR

Error in background operation or authentication status

CATEGORY\_TRANSPORT

Media transport control for playback

CATEGORY\_SYSTEM

System or device status update. Reserved for system use.

CATEGORY\_SERVICE

Indication of running background service

CATEGORY\_RECOMMENDATION

A specific, timely recommendation for a single thing. For example, a news app might recommend a news story the user might want to read next.

CATEGORY\_STATUS

Ongoing information about device or contextual status

Notification sounds

When you assign a priority of DEFAULT or higher to a notification, you may optionally provide a sound to play when the notification is delivered. However, you should deliver your notification silently unless one or more of the following are true:

The notification helps the user uphold a time-sensitive social expectation, such as an incoming phone call or a reminder for a work meeting in five minutes.

The notification tells the user about something they should do or know at that moment in order to function better in their daily life, such as a notification that a flight is delayed.

The user has explicitly requested that this notification make a sound upon arrival (if offered as an option in settings, and off by default).

Lock screen

Android evaluates each notification's visibility level to figure out what can safely be shown. A user may choose to show notifications when their screen is locked. They may also conceal any content that an app marks as sensitive.

Sensitive content on lock screens

Because notifications are visible on the lock screen, user privacy is an important consideration. For each notification you create, set a visibility level to either public, private, or secret.

Public notifications are fully visible on secure lock screens, while secret notifications are hidden.

Private notifications fall in the middle: they show only basic information about the existence of the notification, including its icon and name of the app that posted it. The content is replaced with a default message that says "Contents hidden.” You can replace this message with a redacted version of the content that is still useful but does not reveal personal information, such as "2 new messages.”

This user has chosen to show all notification content on the lock screen.

This user has chosen to hide sensitive content from the lock screen. Because these apps gave their notifications a visibility level of "private,” they show "Contents hidden" by default.

Style

Clear and concise text

Android truncates content titles to a single line (even when expanded).

Content titles should:

Be under under 30 characters long

Contain the most important information

Avoid variables – unless they contain a number or short text string, or are preceded by static text

Exclude the app's name, which already appears in the header

Don't.

The content title shows the app name, which is redundant with the header area and uses up available characters.

Content text should:

Avoid exceeding the 40-character limit

Not repeat what's in the content title

Do.

The content title shows the most important information.

The large icon

In Android N, the large icon is only meant for specific situations in which imagery meaningfully reinforces the notification's content, including:

Communication from another person, such as the image of someone sending a message

The source of content if it's different than the app sending the notification, such as the logo from a YouTube channel to which the user is subscribed

Meaningful symbols about the notification, such as an arrow symbol for driving directions

Large icons should be circular when showing a person, but square in all other cases.

Don't.

The large icon is not intended for branding.

Do.

The large icon is meant to reinforce the content of the notification in a meaningful way, such as a person's photo attached to a message notification.

Templates

Google uses the following notification templates in its suite of Android apps, which you may adapt for your own needs.

Standard

The standard template works well for most notifications, allowing for a succinct message, a large icon when applicable, and actions.

Standard template for collapsed and expanded notifications

Big text

Use this template when you have longer text to display. It lets the user preview more of the text when the notification is expanded.

Big text template for collapsed and expanded notifications

Big picture

Use this template when the notification contains a picture. The large icon offers a thumbnail of the picture, and the user can get a bigger preview by expanding the notification.

Big picture template for collapsed and expanded notifications

Progress

Use this template for activities initiated by the user that need time to complete and can be canceled at any time. (Non-cancelable activities don't warrant notifications.)

Progress template for collapsed and expanded notifications

Media

Use this template to let the user control media currently playing from your app. The collapsed view displays up to three actions, and the large icon can be used for a related image, such as an album cover. The expanded view displays up to five actions with a larger image, or six actions if no image is displayed.

Media template for collapsed and expanded notifications

Permissions

Permission requests should be simple, transparent, and understandable.

Apps should clarify why each permission request is needed, either through the feature name or an explanation provided.

Runtime permissions are requested at the moment a user needs to perform an action in an app.

Denied permissions should provide feedback and options.

Types of permissions

Educate before asking

Ask up-front

Ask in context

Educate in context

Provide an immediate benefit

Only ask for relevant permissions

Usage

Permission groups

Permissions are divided into nine groups, allowing users to grant all permissions encompassed by a single action. For instance, granting permission to Contacts includes the ability to both view and edit Contacts.

Permission

Description

Calendar

Managing calendars

Camera

Taking photos and recording videos

Contacts

Managing contacts

Location

Current device location

Microphone

Audio recording

Phone

Dialing and managing phone calls

Body Sensors

Heart rate and similar data

SMS

Sending and viewing messages

Storage

Accessing photos, media, and files

Implied permission

A user may perform an action that demonstrates clear intent, such as:

Taking a picture

Selecting a contact

Starting a call or text message

In these cases, the user’s action clearly demonstrates their intent, and no permission dialog is needed or authorized.

Intents and Intent Filters

Runtime permissions

Apps may request permission to access information or use device capabilities at any time after installation. When a user needs to perform an action in an app, such as using the device camera, the app may request permission at that moment.

Users may also allow or deny the permissions of any app from Android Settings anytime after installation.

Example of an app requesting permission

Request patterns

Your permissions strategy depends on the clarity and importance of the permission type you are requesting. These patterns offer different ways of introducing permissions to the user.

Critical permissions should be requested up-front. Secondary permissions may be requested in-context.

Permissions that are less clear should provide education about what the permission involves, whether done up-front or in context.

Educate before asking

If your app has a “warm welcome,” use it to explain what your app does and why unexpected permissions will be requested.

Ask up-front

Only ask for critical and obvious permissions on first launch.

Because users expect a messaging app to request SMS permissions, requesting it up-front makes sense.

Ask in context

Wait until a feature is invoked to request permission.

Users are more likely to allow a permission when they want to use the feature it enables.

Educate in context

Explaining a permission in context helps gauge user interest and improve comprehension of the permission.

Provide an immediate benefit

Provide instant gratification to make accepting a permission feel justified.

Only ask for relevant permissions

When more than one permission is needed for a feature to work, ask for those permissions only and nothing extra.

Denied permissions

Provide feedback whenever a permission is denied. Because denying permissions may prevent a feature from working as intended, whenever a permission is denied it should be explained to the user.

Permission denials occur in one of two ways:

A permission request is denied by the user

A permission is silently denied without warning because a user once selected “Don’t ask again” from a previous permission request

To ensure that features requiring permission always behave as intended, the app should state that permission is needed and provide a way to allow it.

A snackbar displays the need for permission in order to use an audio feature.

Critical permissions

If the app can no longer run because a critical permission has been denied, explain why that permission must be allowed and offer a button to open Settings so the user can allow it.

A screen explains that a permission is needed for an app to function, with a link to allow that permission in Settings.

Scrolling techniques

Scrolling techniques affect how content scrolls in relation to the app bar.

These following patterns describe the elevations at which content should scroll, how to size flexible space, and when to anchor specific elements.

App bar scrollable regions

Status bar

Toolbar

Tab bar/search bar

Flexible space

App bar scrollable regions

When designing scrolling behavior, app bars contain four main regions (referred to as blocks) that comprise the scrolling structure:

Status bar

Tool bar

Tab bar/search bar

Flexible space: space to accommodate a desired aspect ratio for images or extended app bars

App bar components can include: status bar, navigation bar, tab/search bar, and flexible space

Example of a status bar, navigation bar, tab/search bar, and flexible space

Behavior

Standard app bar

Specs:

The standard app bar height is 56dp on mobile and 64dp on larger screen sizes.

The app bar has two scrolling options:

The app bar can scroll off-screen with the content and return when the user reverse scrolls.

The app bar can stay fixed at the top with content scrolling under it.

The standard app bar

Status bar height: 24dp

Toolbar height: 56dp / 64dp

Animation of toolbar off-screen during scrolling

App bar with tabs

The standard app bar component that can include the following blocks: a toolbar, tab bar, or flexible space.

Tabs may have one of these behaviors:

The tab bar stays anchored at the top, while the toolbar scrolls off.

The app bar stays anchored at the top, with the content scrolling underneath.

Both the toolbar and tab bar scroll off with content. The tab bar returns on reverse-scroll, and the toolbar returns on complete reverse scroll.

Status bar, toolbar, and tab bar

Status bar height: 24dp

Toolbar height: 56dp / 64dp

Tab bar height: 48dp

Animation showing the toolbar scrolling off and the tab and app bars stay in place.

Flexible space

Because the app bar is flexible, it can be extended to accommodate larger typography or images. To extend the app bar, add a flexible space block.

Flexible space may be displayed one of two ways:

The flexible space shrinks until only the toolbar remains. The title shrinks to 20sp in the navigation bar. When scrolling to the top of the page, the flexible space and the title grow into place again.

The whole app bar scrolls off. When the user reverse scrolls, the toolbar returns anchored to the top. When scrolling all the way back, the flexible space and the title grow into place again.

Status bar, toolbar, and flexible space

Status bar height: 24dp

Toolbar height: 56dp / 64dp

Animation showing flexible space during scrolling

Flexible space with image

Use flexible space to accommodate images in the app bar with the desired aspect ratio.

In this example, the aspect ratio is 4:3. When scrolling, the content pushes up the image, which shrinks the flexible space. At the end of the transformation, the image gets tinted with the primary color, independent of scrolling.

Status bar, toolbar, and flexible space

Status bar height: 24dp

Toolbar height: 56dp / 64dp

Animation showing flexible space and image during scrolling

Flexible space with overlapping content

Content can overlap the app bar.

The app bar has two scrolling options:

The app bar is initially positioned behind the content. Upon upward scroll, the app bar should scroll faster than the content, until the content no longer overlaps it. Once anchored in place, the app bar lifts up for content to scroll underneath.

The app bar can scroll off-screen with the content and return when the user reverse scrolls.

In this interaction, the app bar cannot include tabs.

Flexible space

Status bar: 24dp

Toolbar: 56dp/64dp

Animation showing flexible space and overlapping content during scrolling

Z-space diagram, side view

Search

Search allows users to locate app content quickly.

When an app supports large amounts of information, users should be able to quickly locate content by searching for it.

Search options

Voice search

Search suggestions

Autocompletion

Types of search

Persistent search

Expandable search

In-app search

When an app supports large amounts of information, users should be able to quickly locate content by searching.

Basic search involves:

Opening a search text field

Entering and submitting a query

Displaying a set of search results

However, the search experience can be enhanced by providing:

Voice search

Historical search suggestions based on recent user queries, before a query is completed

Auto-completed search suggestions matching actual results in your application data

There are two major patterns for in-app search: persistent search and expandable search.

Persistent search

Use persistent search when search is the primary focus of your app.

Behavior:

The search text field is presented inside of an inset search box, ready to receive focus. The user can touch the microphone icon to initiate a voice search.

When in focus, the search field expands to show historical search suggestions. If needed, the onscreen keyboard will also appear.

Choosing any of the suggestions submits the search. Touching the up arrow releases the focus from search, dismissing suggestions and the on-screen keyboard.

Persistent search box

Persistent search text field in focus

As the user enters a query, the search suggestions shift to auto-completion. As the user types, the suggestions are filtered and sorted. Choosing a suggestion or pressing the return key submits the search.

The X action in the search box clears the query.

When displaying search results, the search box remains visible, but is not focused by default. The onscreen keyboard is dismissed so more results can be shown.

Search results are formatted as cards to match the inset appearance of the search box and to accommodate different types of results.

Auto-complete

Search results

Expandable search

Use expandable search when search is not the primary focus of your app.

Behavior:

Display a magnifying glass icon in the toolbar instead of a search text box.

Touching the search icon causes the toolbar to transform, clearing other content and displaying a search text field. If voice search is supported, the microphone icon also appears.

The search text field automatically receives focus, and, if needed, the onscreen keyboard will appear. Historical search suggestions can be shown beneath the toolbar. Choosing any of the suggestions submits the query.

Touching the up arrow closes search and restores the original presentation of the toolbar.

Expandable search

Search text field in focus

As the user enters a query, the search suggestions shift to auto-completion. As the user types, the suggestions are filtered and sorted. Choosing a suggestion or pressing the return key submits the search.

The X action in the search field clears the query.

When displaying search results, the search version of the toolbar remains visible, but is not in focus by default. The onscreen keyboard is dismissed so more results can be shown.

Search results are formatted as cards and appear in the main body of the page beneath the toolbar.

Auto-complete

Search results

Selection

Selection enables user choices through gestures and visual cues.

Checkboxes indicate completion, such as checking items off a list.

Text selection is indicated by highlighting a segment of text.

Types of selection

Item selection

Text selection

Selection gestures

A long-press, touch, or mousedown that’s held in the same position for a moment

A two-finger touch (touch devices only)

Usage

The selection gesture

The global gesture to select data is a long-press, touch, or mousedown that’s held in the same position for a moment. A two-finger touch may also trigger selection on touch devices.

Checkboxes (Recommended for desktop)

Checkboxes, or custom analogs, should only be used to indicate completion, such as checking items off lists.

Checkbox behaviors:

When hovering over an item, display a single checkbox for that item.

After an item is selected, display checkboxes for all remaining items in that set.

Avoid persistently displaying checkboxes as part of each item.

Item selection

Item selection allows users to apply actions to selected items.

Selecting items

On touch devices, select an item by pressing and holding it.

To exit selection mode, deselect all items or perform an action on selected items using the toolbar.

On touch devices, select an item by pressing and holding it.

Toggling selection on and off

To toggle an item between selected and unselected states, tap the item.

To select multiple items, long press and drag the cursor.

Once an item is selected, tap the item to toggle its selection.

Users may long press and drag to select items using the cursor.

Desktop

To toggle selection on desktop, hover over an item to reveal a check mark which can be clicked.

On desktop, users may click and drag across checkboxes to select multiple items.

Indicating selection

To indicate that an item is selected, display a check mark and scrim overlapping the item, or its associated avatar.

Checkmarks replace icons or avatars if there is space.

Use compact checkmarks to minimize the hiding the amount of content displayed.

.

Text selection

Text selection is indicated by highlighting a contiguous segment of text.

On mobile, a selection handle is added to both the beginning and end of a selection. Actions related to the text appear in a dropdown menu positioned immediately above, but not overlapping, the selection.

Adjusting selected text

The user may adjust which text is selected by:

Dragging the initial selection to expand it

Dragging either of the selection handles to expand or reduce the selection

Touching (or clicking) repeatedly within the selection to expand it (Selection will toggle between a single word, paragraph, or all text)

Keyboard shortcuts (see table)

Keyboard shortcut

Result

Shift+Left/Right Arrow

Character-by-character selection

Shift+Up/Down Arrow

Line-by-line selection

Ctrl/Command+A

Select all

Text selection keyboard shortcuts can be used to alter the selection bounds.

Text selection handles

Example of text selection handles

Example of text selection handles

Color and contrast

Text selection elements may reflect your color palette by using your accent color.

Selected text should be legible, with noticeable contrast against its background color. Alternatively, selected text may be indicated by displaying outlines, text treatments, motion, or checkmark icons.

Learn more about contrast (or try a color contrast analyzer) at webaim.org.

Text selection color

Text

Background

Text selection can be customized using your palette accent color.

Text selection – Light theme

Text selection – Light theme

Text selection – Dark theme

Text selection – Dark theme

Text selection toolbar

The floating toolbar appears above the selection area with primary actions of Cut, Copy, Paste, and More. When the user selects the More icon, the toolbar collapses and transforms into a secondary menu to reveal additional actions.

The font for the floating toolbar text is Roboto Medium 14sp, all caps.

Dropdown menus for Cut, Copy, Paste, and More appear above the selection area.

More

When the user selects More, the menu begins transitioning to show the overflow items.

Overflow menu items: Share, Search, and Translate

Text selection

Dropdown menu height: 44dp

Dropdown menu text top padding: 12dp

Dropdown menu text top padding: 14dp

Dropdown menu text left padding: 24dp

Dropdown menu space between menu text items: 32dp

Selected text top padding: 8dp

Expansion icon right padding: 16dp

Expansion icon left padding: 22dp

Expansion icon top and bottom padding: 10dp

Text selection handles: 22 x 22 에

Settings

Application settings let users indicate their preferences for how an app or service should behave.

Place app settings under the “Settings” label in your app’s navigation.

Controls that belong in settings should capture user preferences and be infrequently accessed. They should either affect most users or provide critical support to a minority of users.

Placement options

Side navigation

Toolbar menu

Settings components

Labels

Secondary text

Switch or checkbox settings

Status

Placement

All of an app’s settings should be reached via the “Settings” label. Do not use synonyms such as “Options” or “Preferences.”

Side navigation

If side navigation such as a navigation drawer exists, include “Settings” below all other items (except Help & Feedback).

Apps without navigation

Place Settings in the toolbar menu below all other items (except Help & Feedback).

Settings in the navigation drawer

Settings in the toolbar menu

Usage

Settings should be well-organized, predictable, and contain a manageable number of options.

Provide an overview

A user should be able to quickly understand all available settings and their current values.

If there are many settings, prioritize the ones most likely to interest users. Less important settings can be grouped and moved to their own screen.

Important settings appear at the top of the list

Select appropriate settings

Controls that belong in settings:

Capture a user preference

Get accessed infrequently

Are used by the majority of users

Are used by a minority of users, but are essential to supporting their needs

Settings should not:

Be frequently accessed actions. (Move these items to a toolbar.)

Contain information about the app, such as a version number or licensing information. (These should move to a Help screen.)

Manage user accounts. (Present these actions within the main flow of your app, such as within a side nav.)

Choose polite defaults

The initial value for each setting should:

Represent the default most users would choose

Be neutral and pose little risk

Use less battery or mobile data

Only interrupt when important

Grouping settings

For large lists of settings, cluster settings into multiple shorter lists. Arrange them according to the total number of settings in the Settings panel in your app.

7 or fewer

Don’t group at all.

9 to 10

Group related settings under one or two section dividers. For settings that can't be grouped:

If important, list them at the top without a section divider.

Otherwise, list them at the bottom with a section divider called "Other," in order of importance.

11 to 15

Group related settings under two to four section dividers. Combine sets of two related settings into a single setting.

For example, two related settings with checkboxes could combine into a single multiple-choice setting.

16 or more

Group four or more related settings under a subscreen. Use consistent terminology, making sure that the subscreen’s title matches the label of the setting which opens it.

Grouped settings for interruptions

Related settings grouped under a subscreen

Labels and secondary text

Labels

Make your labels brief and meaningful.

Labels should:

Capitalize only the first word of each label, and proper nouns.

Put the most important text of your label first.

Rephrase negative words like "Don't" or "Never" into neutral terms such as "Block."

Use impersonal labels like "Notifications" instead of "Notify me." Exception: If referring to the user is necessary for understanding the setting, use the second person ("you") rather than the first person ("I").

Labels should avoid:

Generic terms, such as: Set, Change, Edit, Modify, Manage, Use, Select, or Choose

Repeating words from the section divider or subscreen title

Technical jargon, unless it's widely understood by your target audience

Secondary text

Secondary text is optional. If the label is sufficient on its own, don't add a secondary text description.

Notifications

Show notifications

Don't.

Avoid repeating words from the label in the secondary text.

Switch or checkbox settings

Settings that require precise descriptions may add a single-line description underneath the settings label. Descriptions should:

Convey what happens when a setting is enabled

Use words that don't already appear in the label

Enable NFC

Allow data exchange when the phone touches another device

Do.

Start the description with a verb.

Settings that require longer explanations may add a description on a second screen.

Underneath the label, show the status text of the switch as "On" or "Off.”

Add the switch itself and a longer description to a second screen.

All other settings

For non-switch settings, secondary text should show the current status of a setting only.

Sleep

After 10 minutes of inactivity

Do.

Screen timeout

Adjust the delay before the screen automatically turns off

Don't.

Writing guidelines

Be direct and understandable.

Vibrate on touch

Do.

Use tactile feedback

Don't.

Indicate status with specific details placed within the appropriate context.

Sleep

After 10 minutes of inactivity

Do.

Screen timeout

Adjust the delay before the screen automatically turns off

Don't.

Use keywords that describe settings accurately.

Screen lock

Pattern

Do.

If the user hasn’t set up a screen lock, the secondary text should say “None.”

Change screen lock

Change or disable pattern, PIN, or password security

Don't.

Use familiar acronyms when there aren’t better alternatives. Convey how and why an unfamiliar setting exists.

NFC

Allow data exchange when the phone touches another device

Do.

NFC

Use Near Field Communication to read and exchange tags

Don't.

Swipe to refresh

Swipe to refresh manually refreshes screen content with a user action or gesture.

There are two methods for updating content in an app:

The preferred method is to automatically update content using sync, which keeps app content automatically updated.

Swipe to refresh is a swipe gesture available at the beginning of lists, grid lists, and card collections that are sorted by recent content.

Manual refreshing can supplement syncing. It maintains the current scroll position, as when checking for new mail in Gmail.

Icon

The refresh indicator is represented by a circular spinner, an icon with a curved arrow that spins in a circular motion.

Position

The swipe to refresh gesture is available at the top or bottom of content collections.

The refresh indicator’s resting position is centered horizontally relative to refreshing content.

Threshold

The refresh indicator must pass a threshold before an app refreshes, as indicated by the circular spinner’s opacity, speed, and translation changes.

Usage

Swipe to refresh is a swipe gesture available at the beginning of lists, grid lists, and card collections where the most recent content appears (Index 0).

It’s best to use this gesture with dynamic content that has frequent updates surfacing from a consistent location, where users have a high probability of seeing new content after initiating the gesture.

Refresh indicator

The refresh indicator appears only in conjunction with a refresh gesture or action. Syncing does not display a refresh indicator.

Effects

Changes may not be immediately obvious to users when this gesture is used in views that can change significantly upon refresh. For example, the refresh may non-sequentially delete, reorder, modify, and insert items or change only off-screen items.

Swipe to refresh layout

Refresh the contents of a view via a vertical swipe gesture

Animation showing swipe to refresh

Swipe to refresh should not be used in the following situations:

Navigation drawers

Home screen widgets

Pannable content

Don't.

Navigation drawers (if present in an app) contain navigation destinations, not dynamic content.

Don't.

Home screen widgets should update content automatically.

Don't.

Pannable content, like in maps, have no primary direction or content origin from which users can presume the swipe to refresh gesture will originate.

Positioning

Gesture positioning

Typically, this gesture is available at the top of content collections, but it can also be at the bottom (for example, in chat applications).

Refresh indicator resting position

The refresh indicator, when resting, is centered horizontally relative to refreshing content.

Do.

Don't.

Don't.

Refresh indicator active position

The refresh indicator is located near the top of refreshing content.

Its exact vertical resting position can be adjusted to promote visual harmony with the underlying layout. For example, the indicator may fall on a material edge or grid line, as long as it’s located near the top of the refreshing content.

Do.

Don't.

Behavior

As the refresh indicator translates and/or scales into view, the circular spinner fades in while rotating.

The refresh indicator remains visible until the refresh activity completes and any new content is visible, or the user navigates away from the refreshing content.

Do.

Keeping the refresh indicator in view until the activity is completed provides feedback on the status of the refresh activity.

Don't.

Scrolling the refresh indicator off-screen hides the status of the refresh activity and may imply that the refresh activity is associated with a specific component within the view, such as a card, instead of the entire view.

Refresh indicator transitions

When another surface is in front of the material with refreshing content, the refresh indicator translates from underneath that surface and is clipped until it is fully visible.

The refresh indicator increases in size as it translates when:

The refreshing material is at a higher elevation than all other surfaces

A content refresh is initiated via an app bar or overflow menu action

Refreshing content that is at a higher elevation than other surfaces

Refreshing content that is coplanar with another surface

Refreshing content that is below another surface in z-space

Refreshing content via app bar action

Threshold requirements

To ensure intentional usage of the swipe to refresh gesture, the refresh indicator must pass a threshold before the app will refresh. This threshold is indicated by a number of cues:

The circular spinner reaches 100% opacity.

The rotation of the circular spinner slows down.

The rate of translation of the refresh indicator slows down.

After passing the threshold, completing the gesture will initiate a refresh.

Reversing the gesture past the threshold will cancel the refresh action.

Growth & communications

Introduction

The growth and communications guidelines contain best practices and components to help users quickly and intuitively understand what they can do with your app, including onboarding, feature discovery, and gesture education.

The following guidelines aim to:

Help users derive more value from your app by introducing them to features and functionality at contextually relevant moments

Improve app engagement and retention metrics

To ensure a user experience that respects user attention, these recommendations address targeting, timing, volume, and frequency.

First seven days

Onboarding: Self-Select model

User education: Basic guidance on how to use the app

Next 30 days and beyond

Feature discovery: Tips for features and functionality the user hasn’t tried

Onboarding

Onboarding is a virtual unboxing experience that helps users get started with an app’s UI. It should be brief and enhance the use of the app.

Usage

Your onboarding should have specific goals suited to the user’s level of familiarity with the app.

User context

Goals of the warm welcome

The user has already installed the app and doesn’t need to see more marketing for it

The user may be eager to try the app without reading an instruction manual

The user is not yet familiar with the app’s UI or ready to learn about it

Welcome users and excite them about the experience ahead

Help users implicitly or explicitly understand how the app can be used in their lives

Drive users to take actions that increase engagement and retention in the first seven days

Onboarding models

Onboarding is one point in a longer journey that begins in the app store and ends with either:

A first-run experience (FRE), or

A user’s first experience using an app

When designing your onboarding, consider the screens that came before it and those that will come after it.

Show onboarding to first-time users. Don’t show it to returning users.

Material design includes three onboarding models:

1. Self-Select

Allow users to customize their experiences.

2. Quickstart

Start the user directly in the app.

3. Top User Benefits

Display a brief autoplay carousel (or animatic) highlighting up to three benefits of using the app.

The onboarding best suited to your app depends on if your app uses a common, recognizable UI style and and how easy it is to set up.

Self-Select

Quickstart

Top User Benefits

When to use

The UI can be customized

Your app has setup and consent requirements

You’ve already identified the behaviors that correspond to increased engagement (in the first session) or increased retention (in the first seven days)

Your app is tackling a new challenge or providing a new kind of benefit

To announce new uses or major UI changes

When not to use

Your app is tackling a new challenge or providing a new kind of benefit

To announce new uses or major UI changes

Your app is tackling a new challenge or providing a new kind of benefit

Your app’s UI and benefits use patterns familiar to most users

Your app has no major changes

Combinations

Don’t combine Self Select with Top User Benefits

Ok to include setup before first-run experience

Don’t combine Top User Benefits with Self Select

It’s okay for your app to show setup after or as part of onboarding

Self-Select

The Self-Select model allows users to customize their first-run experience by making a short series of choices.

This experience provides implicit education, giving the user a sense of control and vested interest in the screens to come.

Sign-in screen

Self-Select model

Present the right choices

The choices you give users affect the success of your onboarding.

Choices should:

Be meaningful and noticeable

Provide new information

Be short

Meaningful and noticeable

Offer choices that have a meaningful and noticeable impact on the user experience. These implicitly teach users how to interact with your UI.

Do.

Selecting what to bundle will have a meaningful impact on the user experience.

Don't.

Knowing how often users check their stream doesn’t impact their experience.

Ask what you don’t know

Don’t ask for preferences that can be garnered from normal usage.

Do.

Normal usage won’t easily clarify what a user wants to bundle. It is valuable to ask the user their preferences.

Don't.

Don't ask users to make selections that will become evident with normal use of the app and that won't meaningfully change the first-run experience.

Keep it short

Limit choices to one screen, or make multiple screens feel connected to each other.

Each screen should have fewer than ten choices.

Do.

A single Self-Select screen

Don't.

Many Self-Select screens

When designing onboarding, think about how the onboarding process connects to a user’s first-run experience. After onboarding, the users should land on a screen that makes it easy to act on what they just learned.

Design

Design your Self-Select screens to relate to what your app does. Apps focused on content consumption may ask for topics of interest, whereas apps with feeds may ask which topics to bundle.

Some common Self-Select design patterns include:

Bundled List

Grid View

List

Quickstart

In the Quickstart model, users land directly in the UI without any onboarding model shown (other than sign in and setup).

The Quickstart model:

Enables users to quickly get started with the core functionality of the app

Often prioritizes the first key action

Can also provide an optional way to learn more or ask for help

Best practices

Give users something to do

Rather than leaving users on a blank screen, your UI should encourage interaction.

Do.

Provide options to get the user started.

Don't.

Don’t leave users with nothing to do.

Offer education

If it appears as though the user is unclear how to use the app towards the end of the average first-run experience (90% of the way through the first session), display a UI prompt offering the opportunity to learn how to use the app.

Do.

Offer the user the opportunity to learn more about the app.

Don't.

Don’t force education upfront.

Prioritize the first key action

Choose the action most closely linked to engagement in the first seven days. Alternatively, introduce core functionality as tips for actions that a user hasn’t tried.

Do.

Nudge users to take the first key action.

Don't.

Don’t leave users with nothing to do.

Top User Benefits

The Top User Benefits onboarding model contains a brief autoplay carousel, or animated storyboard, that highlights up to three of the primary benefits from using an app.

Choosing the right benefits

The Top User Benefits model should demonstrate up to three primary benefits from using the app. These benefits should position the app as relevant and personal during a moment that matters, rather than give instructions or describe features.

When identifying which benefits to present, consider:

Problems that the app solves

The primary benefits the app creates

The app’s “toothbrush features” (meaning, a feature you would use once or twice a day)

Integration options

Auto-rotating carousel

A maximum of three illustrations should auto-rotate every two to three seconds and display pagination navigation. Auto-advance the first screen more quickly so that it’s clear this isn’t a single screen. The auto-advance feature should be disabled if the user touches the carousel.

Display a “Get Started” button throughout the animation, and loop through the animation continuously until the “Get Started” action is tapped.

The screen should be swipe-enabled, either in a forward or backward direction.

The button and pagination navigation are fixed. The typography is dynamic and on a separate field than the illustration, but at the same elevation.

Video

The most complex version of a Top User Benefits experience includes a highly produced video.

This should include a “Get Started” button, without pagination circles.

Video implementation

Best practices

Maintain visual continuity

Maintain visual continuity throughout characters, environments, style, typography, and button colors.

Do.

Using consistent visuals and colors throughout the experience unify the story by creating a uniform canvas for the button and circles.

Simplify

Simplify the visuals to the essentials needed to convey a concept.

Do.

The metaphor of saving files to the cloud is visually illustrated.

Don't.

There is no point of focus in this image.

Don’t be UI literal

Don’t show app UI if users haven't experienced it yet. Show the user benefit first.

You may display education about the specific UI in a later context.

Do.

This illustration helps convey the benefits of directions.

Don't.

Showing the actual app’s UI makes it unclear if the images are an illustration or an interactive element.

Design onboarding that connects to a first-run experience. The UI seen after onboarding should make it easy for users to act on what they just learned.

Design

The design of Top User Benefits should compliment the writing. If there’s an idea that can be better expressed through words, use text rather than imagery.

These layouts are designed to allow an illustration with a 1:1 aspect ratio to consistently fit on screens across all platforms. Ensure that the background and text color meet minimum contrast ratios for accessibility.

Mobile and tablet portrait

Place center-aligned copy and interactions beneath the illustration.

Mobile portrait

Tablet portrait

Mobile and tablet landscape

Place left-aligned copy and interactions against the right edge of the illustration, vertically centered.

Mobile landscape

Tablet landscape

Desktop

Place illustration, copy, and interactions in a centered card. Display “Next” and “Previous” buttons beside this card and pagination indicators beneath it.

Desktop

Warm welcome experiences for wearables and TV use different methods of interaction, and should be treated differently.

Platform Redlines

Mobile and tablet portrait

Copy and UI are center-aligned on the screen and built from the bottom of the screen upward with 24dp padding.

Headline: Type 24sp, Leading 32sp

Subhead 1: Type 15sp, Leading 24sp

32sp line height

56dp padding between the top of button and the center of copy (allowing room for 1-3 lines of text)

24dp vertical padding

Mobile and tablet landscape

Align copy and UI to the left edge of the illustration, vertically centered.

Headline: Type 24sp, Leading 32sp

Subhead 1: Type 15sp, Leading 24sp

32sp line height

56dp padding between the top of the button and the center of copy (allowing room for 1-3 lines of text)

24dp vertical padding

Desktop

Headline: Type 24sp, Leading 32sp

Subhead: Type 16sp, Leading 24sp

Line Height: 32sp

Padding between top of button and center of copy: 56dp (allowing for 1-3 lines of text)

Vertical padding from image to pagination dots: 24dp

Horizontal spacing from image to arrow: 48에

Feature discovery

Provide value and encourage return visits by introducing users to new features and functionality at contextually relevant moments.

Design patterns

Single-step actions

Prompt users to try features or functionality that can be completed in a single tap, such as saving a location in a map app.

Tap target

Hint text

Multiple-step actions

Prompt users to try features or functionality that require more than one tap to complete, such as turning on automatic backup in a photo app.

Guided flow

Targeting & triggering

Feature discovery prompts have more impact when they are presented to the right users at contextually relevant moments. When presented to the wrong user at the wrong time, they can be intrusive and annoying.

Targeting

Display features that are relevant to specific users.

Target the right users

Prioritize the right features

Show feature prompts to users who have not tried those features.

Avoid showing advanced features to more casual users.

Present general features to all users, such as undoing sent mail in Gmail.

Feature discovery should continue over time.

Prioritize features that are relevant to user context, not just features that are new.

Triggering

Time the display of feature discovery prompts appropriately.

Don’t present upon opening an app

Contextually relevant moments

Natural pauses

Users may be less likely to find a feature discovery prompt helpful upon opening an app if they’re opening it to take a specific action.

Avoid displaying feature discovery prompts at interruptive moments.

Present feature discovery prompts at moments when they will help the user better complete the action they’re taking.

For example, let a user know they can crop an image only after they’ve added an image.

If you can’t set feature discovery prompts to display when triggered by specific actions, display them during natural pauses in the user experience.

For example, in a mail app, wait until the user has read or sent a message to display a feature discovery prompt.

Volume & frequency

Volume and frequency

A user’s interaction with a message signals their satisfaction with it and informs when to show the next message.

Volume

Frequency

Limit the number of feature discovery messages you present in your UI. For example, don’t display more than one per session.

If the user dismisses a message, don’t show it or a similar message again for a more substantial period of time.

If the user accepts a message, it’s a good signal that the user found it useful. It’s okay to display a similar message in a shorter period of time.

Design

Feature discovery scenarios

Feature discovery scenarios and app UIs have specific material design components.

Feature discovery scenario

Design component

Prompts for actions that can be completed in a single tap

Tap target, hint text

Prompts for actions that require more than one interaction

Guided flow

Tap target

A feature discovery prompt focuses user attention on a specific UI element. It contains a tap target, indicated by a circular shape, which is surrounded by a larger area with a distinct background color.

The prompt contains a single action, without additional actions present.

Users may dismiss the prompt by using a swipe gesture or tapping outside of the prompt’s background area.

The feature discovery prompt contains a tap target, surrounded by a large background area which is colored using the app’s primary color.

Mobile

The prompt may be modified to accommodate different screen locations and display sizes.

On mobile, tap targets are located at a higher elevation than other UI elements and may span the width of the screen.

When the prompted action’s tap target is located within 88dp of the screen top or bottom (not including the status and nav bars), place the tap target directly on top of the promoted action.

When the prompted action is located toward the center of the screen, the prompt background should be adjusted as follows:

Tap targets located right of the screen’s center: The prompt background should be offset 20dp to the left

Tap targets located left of the screen’s center: The prompt background should be offset 20dp to the right

If the tap target is in the center of screen: The prompt background will be offset automatically to the right or left depending on language

Desktop and tablet

On larger screens, feature discovery prompts are generally limited to the surface on which the tap target rests, such as a card. The tap target should be the size of the tappable surface.

For tap targets not in the center of the screen, offset the prompt background from the center of the surface.

If the tap target appears on a surface smaller than 280dp tall and 360dp wide, allow the tap target and prompt background to extend beyond that surface’s edges (at a higher elevation).

Tap targets located within 88dp of the surface’s edge should center the prompt’s background around the target.

For tap targets located in the center of the screen, offset the prompt background area by 20dp.

For all other tap targets, place the prompt background in a location suitable to other screen elements.

For tap targets located within the top or bottom (144dp) or left or right (200dp) screen edge, center the prompt background around the tap target, above other elements.

Feature discovery tap targets on small surfaces, outside of the top or bottom (144dp) or left or right (200dp) screen edges, may use prompts be unconfined by any surface.

Tap targets located right of the screen’s center: The prompt background should be offset 20dp to the left

Tap targets located left of the screen’s center: The prompt background should be offset 20dp to the right

If the tap target is in the center of screen: The prompt background will be offset automatically to the right or left depending on language

Color

The tap target may use the primary color of your app or toolbar.

The background of the tap target has an opacity of 96%.

The primary color is applied to the tap target background.

The accent color is applied to the tap target background.

Motion

Motion helps inform users where the tap target originates.

Appear

The tap target originates on the screen from the highlighted icon using a standard material motion curve.

Idle

While idle, the highlighted action grows by 10% over one second and then shrinks. A material touch ripple emanates from the action as it reaches its largest size.

Interaction

Tapping the highlighted action causes the tap target to expand and then fade with an ease out.

Dismiss

Tapping outside the tap target causes it to shrink and fade with an ease out.

Motions for appear, idle, and interaction

Motions for appear, idle, and dismiss

Guided flows

Guided flows walk users through tasks that require multiple steps to complete.

Guided flow

Guided flows may contain up to three design components. Use the components that work best for the specific tasks in your UI.

1. Tap target

Tap targets introduce the task ahead by highlighting the user benefit.

2. Ripple

Ripples guide users through subsequent steps.

The color of the ripple should match the color of the tap target.

3. Completion toast

Upon completion of the guided flow, provide a toast confirming that the user has completed the task.

Include an action that brings the user back to where the tap target appeared.

The tap target is used to introduce the guided flow.

The ripples are used for following steps.

The toast is used to reaffirm that the task has been completed.

Hint Text

For smaller, simpler prompts that request user input, use hint text instead of tap targets.

Example of hint text

Gesture education

Gesture education helps users understand how to interact with a specific element, like a card, for the first time.

Design patterns

Partial surface

Prompt users to perform a gesture that affects a portion of the screen, such as swiping a card.

“Hint motion” shows a preview of how sample cards move when performing the action marked by the gesture indicator.

Entire screen

Prompt a user to perform a gesture that affects the entire screen, such as pinching to zoom in.

The snackbar uses text to describe the swipe action that triggers movement between tabs. The gesture indicator visualizes the action while the gesture itself is partially performed in the background using hint motion.

Targeting and triggering

Gesture education should happen as the user needs it. It doesn’t need to be condensed into a user’s first moments with an app. It’s smart and contextual, helping users interact with an element or surface in a way they have not done so previously.

Whether the gesture affects only part of the screen, or the entire screen, similar targeting and triggering guidelines are used.

Targeting

Triggering

Show gesture education only to users who have not performed the gesture.

Gesture education occurs:

During the first-run experience (FRE)

In later sessions, if a user has not performed the gesture

After a major change to the UI

If a certain gesture is core to the user experience, present it at a contextually relevant moment later in the first-run experience.

If a gesture is not critical to the experience, show it in a later session at a contextually relevant moment. For example, swiping the toolbar in Chrome to switch tabs.

Volume and frequency

Volume

Limit the number of gesture education prompts displayed in your app and UI.

If a gesture is critical to your user experience, display up to two prompts during the first-run experience.

Frequency

Most gesture education prompts should be shown only once.

If a gesture is critical to your user experience, display it up to two times in distinct sessions.

Design & writing

Sample UI

Design

The sample UI should mimic the movement of the real UI it represents.

The gesture indicator should pause then disappear once a tap gesture is detected. The gesture being taught should be the only way to dismiss the sample UI. For example, only a left or right swipe will dismiss a card describing a swipe gesture.

Writing

Describe the result of the gesture in short and clear text.

This type of interaction may be used for a gesture that affects a portion of the screen, such as swiping a card.

Sample gesture

Design

Full-screen gestures should explain what the gesture will do using a snackbar.

When the user taps the screen, the page content should return to its default state, with the gesture indicator removed. Continue to display the bottom sheet until the user interacts with an element, such as tapping a button.

Writing

Describe the result of the gesture in short and clear text.

This type of interaction may be used for a gesture that affects the entire screen, such as pinch-to-zoom.

Accessibility

Accessible design allows users of all abilities to navigate, understand, and use your UI successfully.

Principles

A well-designed product is accessible to users of all abilities, including those with low vision, blindness, hearing impairments, cognitive impairments, or motor impairments. Improving your product’s accessibility enhances the usability for all users. It’s also the right thing to do.

Material design’s built-in accessibility considerations will help you accommodate all of your users. This section primarily applies to mobile UI design. For more information on designing and developing fully accessible products, visit the Google accessibility site.

Clear

Help users navigate your app by designing clear layouts with distinct calls to action. Every added button, image, and line of text make the screen more complicated. Simplify your app’s UI with:

Clearly visible elements

Sufficient contrast and size

A clear hierarchy of importance

Key information discernable at a glance

Robust

Design your app to accommodate a variety of users. A user may have a short attention span, be new to your product, or use a text-only screen reader (a program that uses a speech synthesizer to read text aloud or kinesthetically using a braille display). Your app should make it easy for each user to:

Navigate: Give users confidence in knowing where they are in your app and what is important.

Understand important tasks: Reinforce important information through multiple visual and textual cues. Use color, shape, text, and motion to communicate what is happening.

Access your app: Include appropriate content labelling to accommodate users who experience a text-only version of your app.

Specific

Support assistive technologies specific to your platform, just as you support the input methods of touch, keyboard, and mouse. For example, ensure your Android app works with Google’s screen reader, TalkBack.

Assistive technology helps increase, maintain, or improve the functional capabilities of individuals with disabilities, through devices like screen readers, magnification devices, wheelchairs, hearing aids, or memory aids.

Color and contrast

Use color and contrast to help users see and interpret your app’s content, interact with the right elements, and understand actions.

Accessible color palette

Choose primary, secondary, and accent colors for your app that support usability. Ensure sufficient color contrast between elements so that users with low vision can see and use your app.

Contrast ratios

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

Contrast ratios represent how different a color is from another color, commonly written as 1:1 or 21:1. The higher the difference between the two numbers in the ratio, the greater the difference in relative luminance between the colors.

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

Small text should have a contrast ratio of at least 4.5:1 against its background.

Large text (at 14 pt bold/18 pt regular and up) should have a contrast ratio of at least 3:1 against its background.

Correct color contrast

Do.

These lines of text follow the color contrast ratio recommendations and are legible against their background colors.

Incorrect color contrast

Don't.

These lines of text do not meet the color contrast ratio recommendations and are difficult to read against their background colors.

Icons or other critical elements should also use the above recommended contrast ratios.

Correct color contrast for icons

Do.

These icons follow the color contrast ratio recommendations and are legible against their backgrounds.

Incorrect color contrast for icons

Don't.

These icons do not follow the color contrast ratio recommendations and are difficult to discern against their backgrounds.

Logos and decorative elements

While decorative elements (such as logos or illustrations) don’t have to meet contrast ratios, they should be distinguishable if they possess important functionality.

Decorative logo

Do.

Decorative logos that are distinguishable don’t have to meet contrast ratios.

Distorted decorative logo

Don't.

It is not necessary to distort your logo to meet contrast ratios.

Other visual cues

For users who are colorblind, or cannot see differences in color, include design elements in addition to color that ensure they receive the same amount of information.

Colorblindness takes different forms (including red-green, blue-yellow, and monochromatic). Use multiple visual cues to communicate important states. Use elements such as strokes, indicators, patterns, texture, or text to describe actions and content.

Correct text field example

Do.

The text field error state is communicated through multiple cues: title color, text field stroke, and an error message below the field.

Incorrect text field example

Don't.

The text field error state is only communicated with a colored stroke, which would be missed by a user who cannot perceive color.

Sound and motion

Sound

Give visual alternatives to sound, and vice versa. Provide closed captions, a transcript, or another visual alternatives to critical audio elements and sound alerts.

Allow users to navigate your app using sound by adding descriptive labels to UI elements. When using a screen reader such as TalkBack and navigating by touch exploration, labels are spoken aloud when users touch UI elements with their fingertips.

The following sounds should be avoided:

Unnecessary sounds that play over a screen reader, such as background music that autoplays when entering a web page. If there is background sound, ensure users can safely pause or stop it.

Extra sounds added to native elements (screen readers will be able to interpret native elements correctly).

Labeling User Interface Elements

Add audible descriptions to input controls and other elements

Motion

Material design uses motion to guide focus between views, which helps reduce user distraction. Surfaces transform into focal points for the user to follow, and unimportant elements are removed.

To allow users with motion and vision sensitivities to use interfaces comfortably, adhere to the material design motion spec, which supports the following motion guidelines from the W3C:

Enable content that moves, scrolls, or blinks automatically to be paused, stopped, or hidden if it lasts more than than five seconds.

Limit flashing content to three times in a one-second period to meet flash and red flash thresholds.

Avoid flashing large central regions of the screen.

Timed controls

Controls in an app may be set to disappear after a certain amount of time. For example, five seconds after starting a video, playback controls may fade from the screen.

High-priority controls

Avoid using timers on controls that perform a high-priority functions, as users may not notice these controls if they fade away too quickly. For example, TalkBack reads controls out loud if they are focused on, and placing them on timers may prevent the controls from completing their task.

For controls that enable other important functions, make sure that the user can turn on the controls again or perform the same function in other ways. Learn more in hierarchy and focus.

Vocalizing text automatically

Read about placing text in a live region.

Style

Layout

Material design’s touch target guidelines enable users who aren’t able to see the screen, or who have motor-dexterity problems, to tap elements in your app.

Touch targets

Touch targets are the parts of the screen that respond to user input. They extend beyond the visual bounds of an element. For example, an icon may appear to be 24 x 24 dp, but the padding surrounding it comprises the full 48 x 48 dp touch target.

Touch targets should be at least 48 x 48 dp. A touch target of this size results in a physical size of about 9mm, regardless of screen size. The recommended target size for touchscreen elements is 7-10mm. It may be appropriate to use larger touch targets to accommodate a larger spectrum of users, such as children with developing motor skills.

Touch target spacing

In most cases, touch targets should be separated by 8dp of space or more to ensure balanced information density and usability.

Touch target specs

Avatar: 40dp

Icon: 24dp

Touch target on both: 48dp

Examples

Examples of touch targets

Touch target height

Touch target height: 48dp

Button height: 36dp

Examples

Examples of touch targets and buttons

Grouping items

Keeping related items in proximity to one another is helpful for those who have low vision or may have trouble focusing on the screen.

Correct slider value example

Do.

The slider value is in close proximity with the slider control.

Incorrect slider value

Don't.

The slider value is placed too far away from the control. A user of screen magnification may not be able to view both the slider and the value without panning back and forth.

Fonts

To improve readability, users might increase font size. Mobile devices and browsers include features to allow users to adjust font size system-wide. To enable system font size in an Android app, mark text and their associated containers to be measured in scaleable pixels (sp) .

Make sure to allot enough space for large and foreign language fonts. See Line Height for information on the recommended sizes of foreign language fonts.

Hierarchy and focus

Apps should give users feedback and a sense of where they are in the app. Navigation controls should be easy to locate and clearly written. Visual feedback (such as labels, colors, and icons) and touch feedback show users what is available in the UI.

Navigation should have clear task flows with minimal steps. Focus control, or the ability to control keyboard and reading focus, should be implemented for frequently used tasks.

Enabling Focus Navigation

Use focus controls for navigation

Screen readers

Screen readers give users multiple ways to navigate a screen, including:

Touch interface screen readers allow users to run their finger over the screen to hear what is directly underneath. This provides the user a quick sense of an entire interface. Or the user can quickly move to a UI element from muscle memory. In TalkBack, this feature is called “explore by touch.” To select an item, the user must double tap.

Users may also move focus by swiping backwards or forwards on screen to read pages linearly, from top to bottom. This allows users to hone in on certain elements. In TalkBack, this is called linear navigation.

Users may switch between both “explore by touch” and “linear navigation” modes. Some assistive technologies allow users to navigate between page landmarks, such as headings, when these landmarks use the appropriate semantic markup.

Hardware or software directional controllers (such as a D-pad, trackball, or keyboard) allow users to jump from selection to selection in a linear fashion.

Hierarchy

Place items on the screen according to their relative level of importance.

Important actions: Place important actions at the top or bottom of the screen (reachable with shortcuts).

Related items: Place related items of a similar hierarchy next to each other.

Correct hierarchy example

Do.

By placing important actions at the top of the screen, they are given more importance in the hierarchy.

Incorrect hierarchy example

Don't.

When important actions are embedded within other content, it’s unclear what the most important elements are on the page.

Focus order

Input focus should follow the order of the visual layout, from the top to the bottom of the screen. It should traverse from the most important to the least important item. Determine the following focus points and movements:

The order in which elements receive focus

The way in which elements are grouped

Where focus moves when the element in focus disappears

Clarify where the focus exists through a combination of visual indicators and accessibility text.

Grouping

Group similar items under headings that communicate what the groupings are. These groups organize content spatially.

Transitions

Focus traversal between screens and tasks should be as continuous as possible.

If a task is interrupted and then resumed, place focus on the element that was previously focused.

Example of focus order

The green circles indicate the order in which onscreen elements should receive focus.

Implementation

By using standard platform controls, your app will automatically contain the markup and code needed to work well with a platform’s assistive technology. Adapt your app to meet each platform's accessibility standards and assistive technology (including shortcuts and structure) to give users an efficient experience.

Example of a standard platform dialog

Do.

This screen is using the standard platform dialog.

Example of a non-standard platform dialog

Don't.

This screen is using a non-standard platform dialog to perform a standard dialog task. This implementation would require extra coding and testing to make this non-standard control work well with assistive technology.

Test your design with the platform accessibility settings turned on (both during and after implementation).

Other design considerations:

Use scalable text and a spacious layout to accommodate users who may have large text, color correction, magnification, or other assistive settings turned on.

Keyboard/mouse interfaces should have every task and all hover information accessible by keyboard-only.

Touch interfaces should allow screen readers and other assistive technology devices to read all parts of your interface. The text read aloud should be both meaningful and helpful.

Example of scalable text

Do.

This UI has scaled well with magnification and large text.

Example of poorly scaled text

Don't.

This UI has not scaled well with magnification and large text. Portions of content are overlapping or cut off.

Label visual UI elements

Screen-reader users need to know which UI elements are tappable on-screen. To enable screen readers to read the names of components out loud, add the contentDescription attribute to components such as buttons, icons, and tabs containing icons that have no visible text.

Labeling UI Elements

Use the contentDescription attribute to set labels.

1. Label the Search icon

2. Label the Microphone icon

1. Label the Edit icon

2. Label the Chat icon

Help documentation

Any features with special accessibility considerations should be included in help documentation. Make help documentation relevant, accessible, and discoverable. As an example, review this guide on how to use a screen reader with Google Drive.

Testing and research

Following these accessibility guidelines will help improve the accessibility of your app, but does not guarantee a fully accessible experience. It is recommended that you also:

Test your app for full task completion, beginning to end, with various assistive technologies turned on. For example, turn on Explore by Touch in TalkBack and change the speed at which text is spoken out loud.

Have users with impairments test your app.

Consider how individual elements should be accessible while also fitting together in a coherent user flow.

Make sure the major tasks you want your users to complete are possible for everyone.

Talk to your users, particularly those who use assistive technology, to learn about their needs, what they want out of your app, which tools they use, and how they use them. Become familiar with these tools so you can give them the best experience.

People use assistive technology in different ways.

Writing

Clear and helpful accessibility text is one of the primary ways to make UIs more accessible. Users with limited or no eyesight benefit from explicit verbal descriptions. Accessibility text refers to text that is used by screen reader accessibility software, such as TalkBack on Android, VoiceOver on iOS, and JAWS on desktop. Screen readers read all text on screen aloud, including both visible and nonvisible alternative text.

Accessibility text includes both visible text (including labels for UI elements, text on buttons, links, and forms) and nonvisible descriptions that don’t appear onscreen (such as alternative text for buttons without text labels). Sometimes, an onscreen label may be overridden with accessibility text to provide more information for the user.

Both visible and nonvisible text should be helpfully descriptive and independently meaningful, as some users navigate by using all the headings or links on a page. Test your app with a screen reader to identify areas that are missing or need better accessibility text.

Be succinct

Keep content and accessibility text short and to the point. Screen reader users hear every UI element read aloud. The shorter the text, the faster the screen reader users can navigate it.

Switch to heyfromjonathan@gmail.com

Do.

Write clear and short accessibility text.

Account switcher. Switch to account heyfromjonathan@gmail.com

Don't.

Don’t write long accessibility text.

Avoid including control type or state in text

Screen readers may automatically announce a control’s type or state through a sound or by speaking the control name before or after the accessibility text.

Search

Do.

Use short descriptions.

Search field

Don't.

Don’t write the control type.

Developer note: If the control type or state is not being read correctly, the control’s accessibility role may be improperly set or be a custom control. Every element should have an associated accessibility role on a website or be coded to be announced properly. This means a button should be set as a button, and a checkbox as a checkbox, so that the control’s type or state is communicated correctly to the user. If you extend or inherit from a native UI element, you will get the correct role. If not, you can override this information for accessibility on each platform (ARIA for web, AccessibilityNodeInfo for Android).

On Android, set the class name field of the control’s AccessibilityNodeInfo to "android.widget.Button".

Building Accessible Custom Views

Download over Wi-Fi only

Do.

Use short descriptions.

Download over Wi-Fi is selected

Don't.

Don’t write out the state.

Indicate what an element does

Use action verbs to indicate what an element or link does, not what an element looks like, so a visually impaired person can understand. Link text should:

Specify the task that tapping the link will perform

Avoid vague descriptions, such as “click here”

Ensure an element has the same description everywhere it’s used.

Action verb example

Do.

The description read aloud indicates the action represented by the icon.

Incorrect example of a description

Don't.

Describing what the icon looks like doesn’t make it clear what the action does.

Correct example of accessible text

Do.

Accessible text for a navigation menu could be “Show/hide navigation menu” (preferred) or “Show/Hide main menu” (acceptable).

Incorrect example of accessible text

Don't.

When read aloud, the text “side drawer” doesn’t indicate what action will occur.

Elements with state changes

For icons that toggle between values or states, announce the icon according to how it is presented to the user.

If the icon is a property of an item, make it a checkbox so that screen readers verbalize the current state, such as “on” or “off.”

If the icon is an action, write the text label to specify the action that occurs if the icon is selected, such as “Add to wishlist.”

How elements should be used affects how they are displayed. For example, if a star icon represents the action of adding something to a wishlist, the app should verbalize “Add to wishlist” or “Remove from wishlist.”

Don’t mention the exact gesture or interaction

Don’t tell users how to physically interact with a control, as they may be navigating with a keyboard or other device, not with their fingers or a mouse. Accessibility software will describe the correct interaction for the user.

Correct example of a command.

Do.

The command “voice search” describes the user task (search) paired with the input method (voice).

Incorrect example of a command

Don't.

The command “Tap” is inaccurate as it is not the only way to activate this control (it could be selected via a keyboard press, switch device, or braille display). As the main user task in this example is to search, that action should be mentioned instead of “speak”.

Confirm actions

Use dialogs, toasts, or snackbars (Android) to confirm or acknowledge user actions that are destructive (like “Delete” or “Remove”) or difficult to undo.

For actions that are confirmed through visual means, such as a grid rearranging itself when an item is deleted, a toast is not necessary. In these cases, add accessibility text to provide acknowledgement.

Provide hint speech

Hint speech provides extra information for actions that aren't clear. For example, Android's “double-tap to select” feature prompts the user to tap twice when landing on an item without taking action. Android TalkBack will also announce any custom actions associated with an element. Use hint speech sparingly and only for complex UI.

Bidirectionality

UIs for languages that are read from right-to-left (RTL), such as Arabic and Hebrew, should be mirrored to ensure content is easy to understand.

UI mirroring overview

The main difference between left-to-right (LTR) and right-to-left (RTL) language scripts is the direction in which content is displayed:

LTR languages display content from left to right

RTL languages display content from right to left

RTL content also affects the direction in which some icons and images are displayed, particularly those depicting a sequence of events.

In general, the passage of time is depicted as left to right for LTR languages, and right to left for RTL languages.

Element

LTR

RTL

Text

Sentences are read from left to right.

Sentences are read from right to left.

Timeline

An illustrated sequence of events progresses left to right.

An illustrated sequence of events progresses right to left.

Imagery

An arrow pointing left to right indicates forward motion: →

An arrow pointing right to left indicates forward motion: ←

The passage of time is depicted from left to right for LTR languages, and from right to left for RTL languages.

When a UI is changed from LTR to RTL (or vice-versa), it’s often called mirroring. An RTL layout is the mirror image of an LTR layout, and it affects layout, text, and graphics.

When a UI changes from one direction to another, these items are not mirrored:

Numbers

Untranslated text (even if it’s part of a phrase)

Text should always be in the correct direction for the language it’s in. For example, any LTR words, such as a URL, will continue to be shown in an LTR format, even if the rest of the UI is in RTL.

Do.

Text and numbers should always be in the correct direction for the language.

Don't.

LTR text shouldn’t be displayed in reverse order.

When a UI is mirrored, these changes occur:

Text fields icons are displayed on the opposite side of a field

Navigation buttons are displayed in reverse order

Icons that communicate direction, like arrows, are mirrored

Text (if it is translated to an RTL language) is aligned to the right

These items are not mirrored:

Icons that do not communicate direction, such as a camera

Numbers, such as the clock and phone numbers

Charts and graphs

An English UI in LTR

An Arabic UI in RTL, with numbers presented in LTR

Text editing menu in LTR

Text editing menu in RTL

1. Icons related to bidirectionality are mirrored to reflect the start and end of a line of text

LTR screen

On an LTR screen, the tab for “Item One” is aligned to the left, and users swipe to the left to see more tabs.

Touch target height: 48dp

Screen edge margin before first tab: 72dp

Tab labels bottom padding: 20dp

Tab labels right and left padding: 12dp

RTL screen

On an RTL screen, the tab for “Item One” is aligned to the right, and users swipe to the right to see more tabs.

Touch target height: 48dp

Screen edge margin before first tab: 72dp

Tab labels bottom padding: 20dp

Tab labels right and left padding: 12dp

LTR screen

RTL screen

Title, icons, and UI elements are displayed flowing from right to left

1. Back button points to the right

2. Text is right-aligned

3. Primary and secondary buttons are mirrored to match reading direction

4. Checkbox appears to the right of text

5. Icons that do not communicate direction are not changed

6. Placement of units may vary across languages

7. Progress bars fill in the same direction as content is read

LTR

Navigation, overflow menu, and icons displayed left-to-right

Icon padding from screen edge: 16dp

Title distance from screen edge: 72dp

Padding below title: 20dp

Navigation bar height: 56dp

Overflow menu padding: 16dp

RTL

Navigation, overflow menu, and icons switch sides in the RTL layout

Icon padding from screen edge: 16dp

Title distance from screen edge: 72dp

Padding below title: 20dp

Navigation bar height: 56dp

Overflow menu padding: 16dp

LTR

Padding and margin around icons and text for LTR

List item height: 72dp

Icon margin from left screen edge: 16dp

List item distance from left screen edge: 72dp

RTL

When mirroring the layout, padding and margin around icons and text also switch placement to match RTL layouts.

List item height: 72dp

Icon margin from right screen edge: 16dp

List item distance from right screen edge: 72dp

RTL mirroring guidelines

When text, layout, and iconography are mirrored to support right-to-left (RTL) UIs, anything that relates to time should be depicted as moving from right to left. For example, in a RTL layout, forward points to the left, and backwards points to the right.

For detailed developer guidance on RTL icons, check out:

RTL Material Design icons guide

See how to mirror icons

RTL Sketch plug-in

Flip existing designs and assets into an RTL format

When to mirror

The most important icons for mirroring are back and forward buttons. Back and forward navigation buttons are reversed.

LTR back button

RTL back button

LTR forward button

RTL forward button

An icon that shows forward movement should be mirrored.

In a LTR UI, a bicycle facing the right typically communicates a sense of moving forward.

In a RTL UI, a bicycle pointing to the left similarly communicates a sense of moving forward.

Most RTL countries do not mirror slashes. Leave images with slashes as-is for RTL locales.

Do.

The LTR slash can indicate an off state for both LTR and RTL languages.

Don't.

An RTL off state doesn’t need an RTL slash.

A volume icon with a slider at its right side should be mirrored. The slider should progress RTL, and the sound waves should emerge from the right.

LTR volume with slider

RTL volume with speaker icon and slider mirrored

Sometimes, both the horizontal and circular direction of time are implied in an icon. For example, the redo and undo buttons in Google Docs have both a horizontal direction and a circular direction.

In LTR, these point to the same direction in both circular and horizontal representations of time. In RTL, choose whether to show circular or horizontal direction.

LTR redo and undo button from the toolbar in Google Docs

Icons that contain representations of text need careful mirroring.

Text is right-aligned in RTL. If there is a paragraph indent at the beginning of a paragraph, an unfinished line at the end of the paragraph, or a ragged right side, the icons need to be mirrored.

LTR chat icon

RTL chat icon

When not to mirror

While the linear representation of time is mirrored in RTL, the circular direction of time is not. Clocks still turn clockwise for RTL languages. A clock icon or a circular refresh or progress indicator with an arrow pointing clockwise should not be mirrored.

The refresh icon shows time moving forward; the direction is clockwise. The icon is not mirrored.

The history icon points backwards in time; the direction is counterclockwise. The icon is not mirrored.

Some icons refer to physical objects that are not mirrored in the right-to-left world.

For example, physical keyboards look the same everywhere in the world, so they should not be mirrored.

Keyboard icon

Headset icon

Certain icons might seem directional but they actually represent holding an object with one’s right hand.

For example, the search icon typically has its handle at the bottom right side, because the majority of users are right-handed.

The majority of users in RTL-writing countries are also right-handed, so such icons should not be mirrored.

Search icon

Local cafe icon

The passage of time

Anything depicting the passage of time should be mirrored.

Do not mirror media playback buttons and the media progress indicator as they refer to the direction of tape being played, not the direction of time.

Since media playback buttons and the progress indicator reflect the direction of the tape, they are not mirrored.

Do.

Media controls for playback are always LTR.

Don't.

Do not mirror media playback or progress bars. The direction of these elements represents the direction of the tape, not the direction of time.

Localization

Text in graphics

Graphics that include text usually require localization.

Numbers

Numbers, including icons containing numbers, must be localized for languages that use different numerals. For example, Bengali, Marathi, Nepali, and some Arabic-speaking locales use different forms of numbers.

An icon in LTR containing a number

A RTL icon in Arabic

Mirroring

Sometimes content may need to be mirrored, even if the UI is not mirrored. For example, when a user edits an RTL paragraph inside a LTR document, the toolbar buttons for the RTL text should be in RTL.

In this RTL paragraph inside a LTR document, the buttons for indenting and lists should be RTL even though the primary UI direction is LTR.

1. Paragraph aligned right

2. Icons flipped

3. Hebrew text direction is RTL

Platform adaptation

Material Design supports design and usability best practices across platforms to help create beautiful user experiences.

When to adapt

Material Design was built on a foundation of best practices in both traditional and web design, informed by user experience research and cognitive science. The design guidelines that developed from these findings are intended to be universally applied across all platforms and devices.

Design conventions can differ from platform to platform. These differences in convention can affect the user's ability to understand the UI or complete certain tasks. In these cases, it is recommended to adapt to platform-specific conventions. In areas where design differences are minimally disruptive, adapting to the platform is optional.

The following guidelines indicate when you should adapt to native platform conventions, and when you have the option to adapt. Platform conventions are constantly evolving, and Material Design is evolving with them to increase the quality of our design patterns.

Platform recommendations

Toolbars

Toolbars are commonly used to frame the context of a screen.

It is recommended to use a platform’s default text alignment for toolbar titles, unless multiple action buttons are present.

Android/Web

Titles are always left-aligned.

iOS

Titles are center-aligned by default.

Android

When multiple actions, or even no actions, appear on the right side, toolbar titles are always left-aligned.

iOS

When multiple action buttons are on the right side, or at the home screen of an application, titles may be left-aligned.

Iconography

System icons are used to represent the most universally available actions for the platform.

Android/Web

The back button contains a thin arrow with a stem.

iOS

On iOS, the back arrow is thicker, and without a stem.

Android/Web

The action overflow menu icon (indicated by the “More…” symbol) contains three vertical dots.

iOS

The action overflow menu icon (indicated by the “More…” symbol) contains three horizontal dots.

Controls

Controls should clearly indicate how users should interact with them.

Android

Use Material switches, checkboxes, and radio buttons.

iOS

Native platform switches may be used as they have matching functionality and appearance as Material switches.

Use switches instead of check boxes and check mark lists instead of radio buttons, as these are the graphics expected on iOS.

Web

You may use Material switches, checkboxes, and radio buttons.

Gestures

Edge swipe

An edge swipe starts from outside of the screen to reveal off-screen content.

Edge swipes can conflict with other swipe gestures, such as horizontal swipes through pages or table rows. To avoid these kinds of conflicts, an edge swipe should perform the same behavior as any other swipe that exists over a content area.

Android

When conflicting gestures are not present, performing an edge swipe from the left reveals off-screen content, such as a navigation drawer.

iOS

When conflicting gestures are not present, performing an edge swipe from the left navigates you back through an app’s hierarchy.

Do.

Web

Browsers often use edge gestures for their feature set. Edge swipes should not be used on the web.

Don't.

Web

Edge swipes can not be relied on across browsers.

Typography

Typography communicates both text content and branding. In both cases, text must be accessible and resizable.

If your app already uses typography for branding, it is recommended to do so in moderation.

Android

The default typeface on Android is Roboto. On Android, text size should be specified in scaleable pixels to allow type to be resized using accessibility features.

iOS

The default typeface on iOS is San Francisco. Using this typeface is the easiest way to implement accessibility features like Dynamic Type. Using other typefaces may require making adjustments to get the same accessibility features.

Web

While Arial is the most broadly available font on the web, modern browsers provide accessibility and scaling support for web fonts.

Material Design’s standard typographic guidelines for Roboto on the web can help simplify your font usage. Alternatively, use custom fonts and establish your own type hierarchy that ensures legibility.

Android

Material Design has Android-only guidance for several interactions.

The following guidance refers specifically to these interactions on Android:

Notifications

Snackbars and toasts

Fingerprint

Split screen

Color palettes

This downloadable zip file contains Adobe Photoshop and Adobe Illustrator color swatches that you can install. Included in the zip file is a Read-Me.pdf with instructions on how to install the .aco files into Photoshop and .ase files into Illustrator.

For more information on color in material design, see Color.

Devices

Device metrics

Resizer tool

Resizer is an interactive viewer that shows how digital products respond to material design breakpoints across desktop, mobile, and tablet.

Layout templates

Mobile

The mobile layout template includes 10 diverse screens. Use the layers palette to toggle on and off keylines and spacing rules.

Tablet

The tablet layout template includes 14 various screens. Use the layers palette to toggle on and off keylines and spacing rules.

Desktop

The desktop layout template includes four variations of four different window sizes. Use the layers palette to toggle on and off keylines and spacing rules.

Whiteframes

Whiteframes provide a variety of layout structures using a consistent approach to surfaces, layering, and shadows.

Roboto & Noto fonts

The latest version of Roboto is designed for both mobile and web usage.

Noto is the standard typeface on Android for all languages not covered by Roboto.

Sticker sheets & icons

Components

The sticker sheets contain the latest refinements to the components, integrated into a comprehensive sticker sheet that is easier to use.

The sticker sheet includes the elements that make up layouts, including light and dark symbols for status bars, app bars, bottom toolbars, cards, dropdown menus, search field, dividers, side navs, dialogs, floating action buttons, and other components.

The sticker sheet elements align to an 8dp grid. Type baselines fall on a 4dp grid.

In Illustrator, the text box offset is set to Leading (Type > Area Type Options). The sticker sheet also contains character styles that you can use for any text box.

.

System icons

Material design system icons are used across different apps, such as icons used for media playback, communication, content editing, and connectivity.

The download is a sticker sheet with the system icon grid.

Product icons

The download includes 4 sticker sheets for Product Icons. The sticker sheets provide guidance on applying layers and shapes within the Product Icon grid.

Icon library

Shadows

Shadows provide important visual cues about objects’ depth and directional movement. An object’s elevation determines the appearance of its shadow. For guidance on elevation and shadows in material design view Shadows and elevation guidance in the design guidelines.

For developer documentation on shadows in Android or Polymer:

Sketch

Sketch Material elevations

The table to the right is a listing of shadow values for Sketch.

1dp elevation

Key Shadow

Ambient shadow

Umbra

Penumbra

Opacity:

14

12

20

X:

0

0

0

Y:

0

2

1

Blur:

2

2

3

Spread:

0

0

0

2dp elevation

Key Shadow

Ambient shadow

Umbra

Penumbra

Opacity:

14

12

20

X:

0

0

0

Y:

0

3

1

Blur:

4

4

5

Spread:

0

0

0

3dp elevation

Key Shadow

Ambient shadow

Umbra

Penumbra

Opacity:

14

12

20

X:

0

0

0

Y:

3

3

1

Blur:

3

4

8

Spread:

0

0

0

4dp elevation

Key Shadow

Ambient shadow

Umbra

Penumbra

Opacity:

14

12

20

X:

0

0

0

Y:

2

4

1

Blur:

4

5

10

Spread:

0

0

0

6dp elevation

Key Shadow

Ambient shadow

Umbra

Penumbra

Opacity:

14

12

20

X:

0

0

0

Y:

6

1

3

Blur:

10

18

5

Spread:

0

0

0

8dp elevation

Key Shadow

Ambient shadow

Umbra

Penumbra

Opacity:

14

12

20

X:

0

0

0

Y:

8

3

4

Blur:

10

14

15

Spread:

1

3

0

9dp elevation

Key Shadow

Ambient shadow

Umbra

Penumbra

Opacity:

14

12

20

X:

0

0

0

Y:

9

3

5

Blur:

12

16

6

Spread:

1

2

0

12dp elevation

Key Shadow

Ambient shadow

Umbra

Penumbra

Opacity:

14

12

20

X:

0

0

0

Y:

12

5

7

Blur:

17

22

8

Spread:

2

4

0

16dp elevation

Key Shadow

Ambient shadow

Umbra

Penumbra

Opacity:

14

12

20

X:

0

0

0

Y:

16

6

8

Blur:

24

30

10

Spread:

2

5

0

24dp elevation

Key Shadow

Ambient shadow

Umbra

Penumbra

Opacity:

14

12

20

X:

0

0

0

Y:

24

9

11

Blur:

38

46

15

Spread:

3

8

0

Illustrator

Illustrator Material elevations

The table to the right is a listing of shadow values for Illustrator.

Shadows in Illustrator

1.65 MB (.ai)

1dp elevation

Key Shadow

Ambient shadow

Opacity:

26%

8%

X:

0

0

Y:

.92

0

Blur:

.5

.67

2dp elevation

Key Shadow

Ambient shadow

Opacity:

26%

8%

X:

0

0

Y:

1.83

0

Blur:

1

1.33

3dp elevation

Key Shadow

Ambient shadow

Opacity:

26%

8%

X:

0

0

Y:

2.75

0

Blur:

1.5

2

4dp elevation

Key Shadow

Ambient shadow

Opacity:

26%

8%

X:

0

0

Y:

3.67

0

Blur:

2

2.67

6dp elevation

Key Shadow

Ambient shadow

Opacity:

26%

8%

X:

0

0

Y:

5.5

0

Blur:

3

4

8dp elevation

Key Shadow

Ambient shadow

Opacity:

26%

8%

X:

0

0

Y:

7.33

0

Blur:

4

5.33

9dp elevation

Key Shadow

Ambient shadow

Opacity:

26%

8%

X:

0

0

Y:

8.25

0

Blur:

4.5

6

12dp elevation

Key Shadow

Ambient shadow

Opacity:

26%

8%

X:

0

0

Y:

11

0

Blur:

6

8

16dp elevation

Key Shadow

Ambient shadow

Opacity:

26%

8%

X:

0

0

Y:

14.67

0

Blur:

8

10.67

24dp elevation

Key Shadow

Ambient shadow

Opacity:

26%

8%

X:

0

0

Y:

22

0

Blur:

12

16

Photoshop

Photoshop Material elevations

The table to the right is a listing of shadow values for Photoshop.

Shadows in Photoshop

1.09 MB (.psd)

1dp elevation

Key Shadow

Ambient shadow

Opacity:

24%

16%

Angle:

90

90

Distance:

2

1

Spread:

6

2

Size:

3

4

2dp elevation

Key Shadow

Ambient shadow

Opacity:

24%

16%

Angle:

90

90

Distance:

3

1

Spread:

6

2

Size:

5

7

3dp elevation

Key Shadow

Ambient shadow

Opacity:

24%

16%

Angle:

90

90

Distance:

5

2

Spread:

6

2

Size:

7

10

4dp elevation

Key Shadow

Ambient shadow

Opacity:

24%

16%

Angle:

90

90

Distance:

7

3

Spread:

6

2

Size:

10

13

6dp elevation

Key Shadow

Ambient shadow

Opacity:

24%

16%

Angle:

90

90

Distance:

10

4

Spread:

6

2

Size:

15

20

8dp elevation

Key Shadow

Ambient shadow

Opacity:

24%

16%

Angle:

90

90

Distance:

13

5

Spread:

6

2

Size:

20

27

9dp elevation

Key Shadow

Ambient shadow

Opacity:

24%

16%

Angle:

90

90

Distance:

18

9

Spread:

6

2

Size:

27

36

12dp elevation

Key Shadow

Ambient shadow

Opacity:

24%

16%

Angle:

90

90

Distance:

20

8

Spread:

6

2

Size:

30

41

16dp elevation

Key Shadow

Ambient shadow

Opacity:

24%

16%

Angle:

90

90

Distance:

28

11

Spread:

6

2

Size:

40

53

24dp elevation

Key Shadow

Ambient shadow

Opacity:

24%

16%

Angle:

90

90

Distance:

39

15

Spread:

6

2

Size:

60

81