Usability – Accessibility

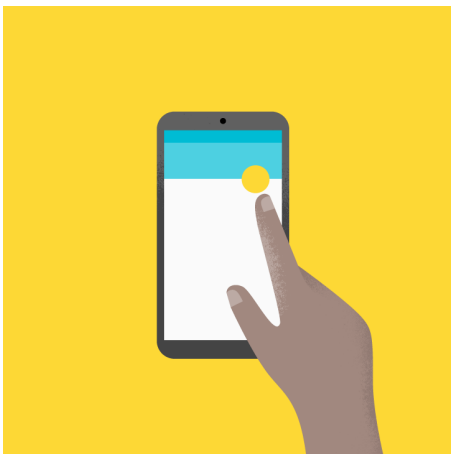
Accessibility

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 imp cognitive impairments, or motor impairments. Improving your product's accessibility enhances the usability for ever uses it. 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 prin to mobile UI design. For more information on designing and developing fully accessible products, visit the Google a site.

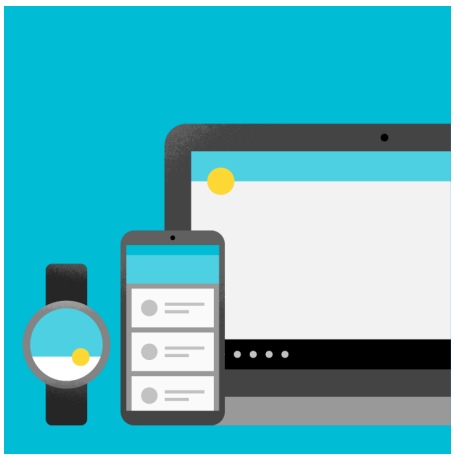


Clear

Help users navigate your app by designing clear layouts with distinct

Robust

Design your app to accommodate a variety of users. A user may have a



Specific

Support assistive technologies specific to your platform, just as you

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

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 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 unde actions.

Accessible color palette

Choose primary, secondary, and accent colors for your app that support usability. Ensure sufficient color contrast be
elements so that users with low vision can see and use your app.

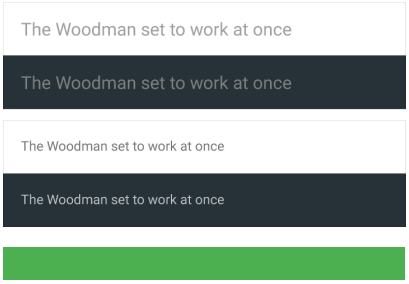
Contrast ratios

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.

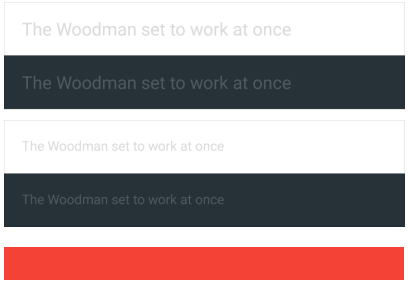
The W3C recommends the following contrast ratios for body text and image text:

- 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.



Do.

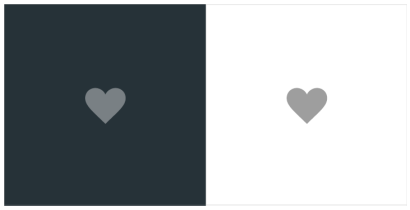
These lines of text follow the color contrast ratio recommendations and are legible against their background colors.



Don't.

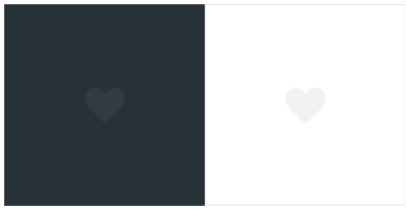
These lines of text do not meet the color contrast ratio recommendations and are difficult to read against their background colors.

Icons or other critical elements should also use the above recommended contrast ratios.



Do.

These icons follow the color contrast ratio recommendations and are legible against their backgrounds.



Don't.

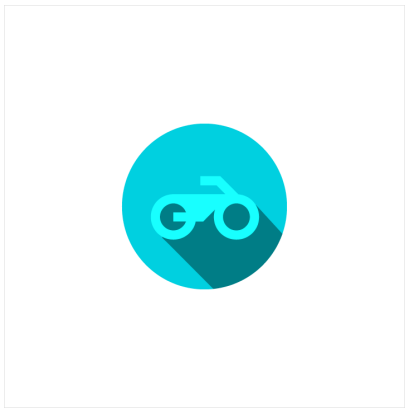
These icons do not follow the color contrast ratio recommendations and are difficult to discern against their backgrounds.

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.

Do.

Decorative logos that are distinguishable don't have to meet contrast ratios.



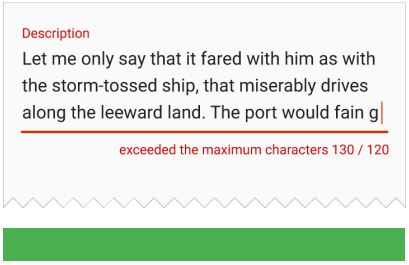
Don't.

It is not necessary to distort your logo to meet contrast ratios.

Other visual cues

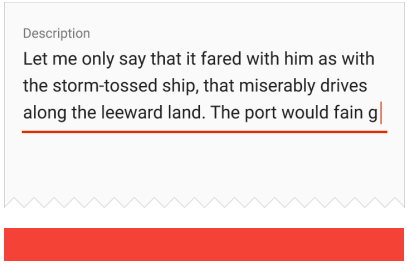
For users who are colorblind, or cannot see differences in color, include design elements in addition to color that ensure they receive the same amount of information.

Colorblindness takes different forms (including red-green, blue-yellow, and monochromatic), so it's important to use multiple visual cues to communicate important states. Use elements such as strokes, indicators, patterns, texture, or text to describe actions and content.



Do.

The text field error state is communicated through multiple cues: title color, text field stroke, and an error message below the field.



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

Give visual alternatives to sound, and vice versa. Provide closed captions, a transcript, or another visual alternative t
audio elements and sound alerts.

The following sounds should be avoided:

- Unnecessary sounds that play over a screen reader, such as background music that autoplays when entering a w
- Extra sounds added to native elements (Screen readers will be able to interpret native elements correctly.)

Motion

Material design uses motion to guide focus between views, which helps reduce user distraction. Surfaces transform
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 mot
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
  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.

Layout

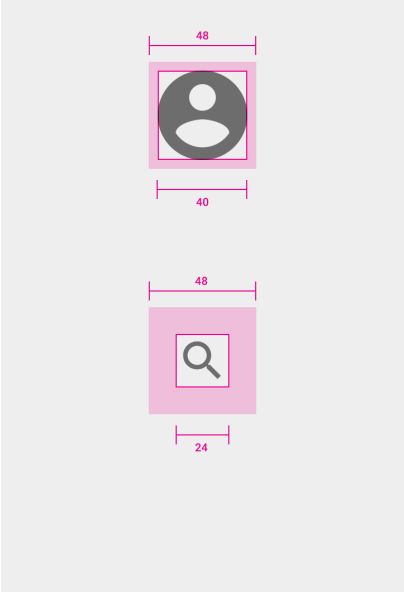
Material design's touch target guidelines describe

how to allow users with lower motor dexterity, and those who aren't able to see the screen, to tap on the correct targets in your app.

Touch targets

Touch targets include the area that responds to user input. Touch targets extend beyond the visual bounds of an element: An element like an icon may be appear to be 24 x 24dp but the padding surrounding it comprises the full 48 x 48dp touch target.

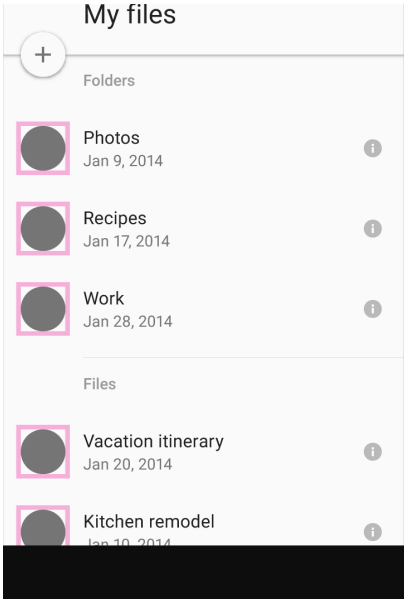
Touch targets should be at least 48 x 48dp, separated by 8dp of space or more, to ensure balanced information density and usability. A touch target of 48 x 48dp results in a physical size of about 9mm, regardless of screen size. The recommended target size for touchscreen objects is 7-10mm.



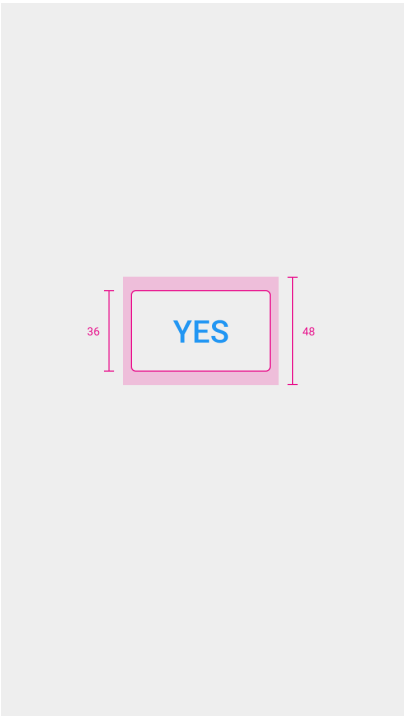
Avatar: 40dp

Icon: 24dp

Touch target on both: 48dp

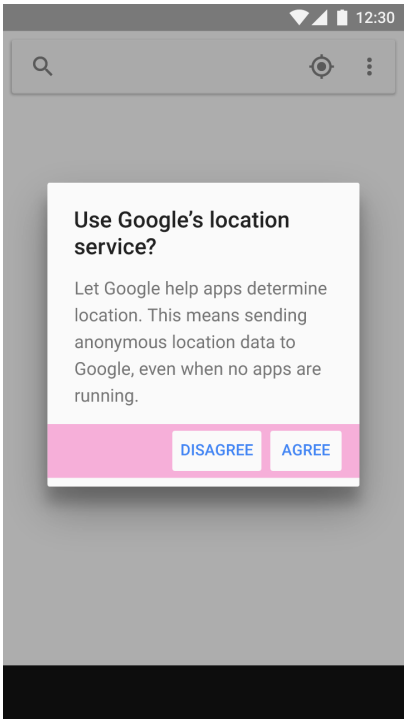


Examples of touch targets



Touch target height: 48dp

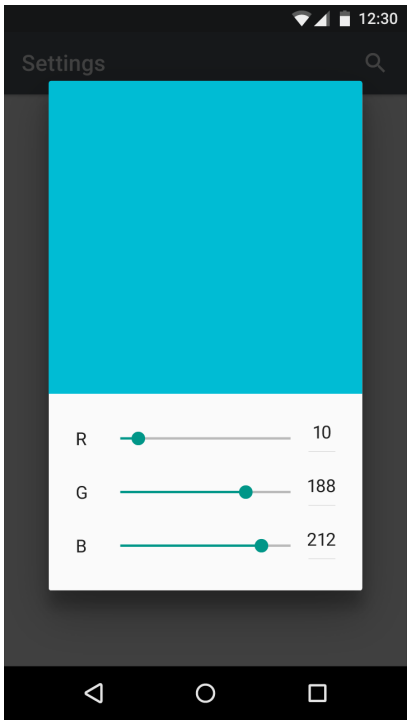
Button height: 36dp



Examples of touch targets and buttons

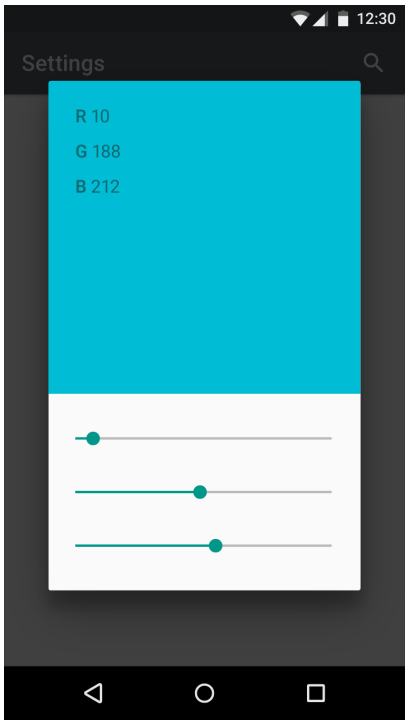
Grouping items

Keeping related items in proximity to one another is helpful for those who have low vision or may have trouble focusing on the screen.



Do.

The slider value is in close proximity with the slider control.



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.

Writing

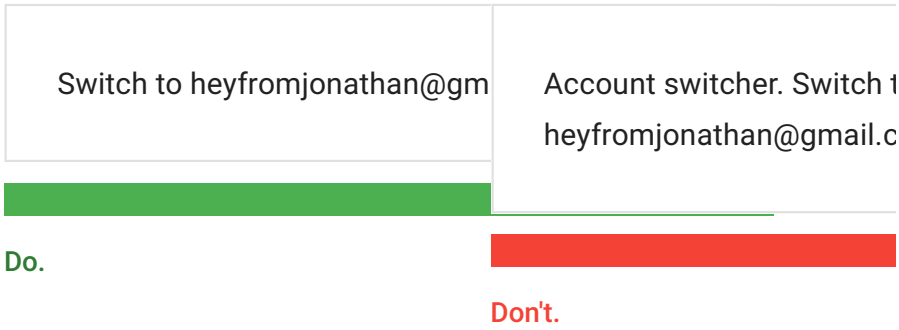
Clear and helpful **accessibility text** is one of the primary ways to make UIs more accessible. Accessibility text refers used by screen reader accessibility software, such as TalkBack on Android, VoiceOver on iOS, and JAWS on desktop readers read all text on screen aloud, including both visible and nonvisible alternative text.

Accessibility text includes both visible text (including labels for UI elements, text on buttons, links, and forms) and n descriptions that don't appear onscreen (such as alternative text for buttons without text labels). Sometimes, an on 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 naviga the headings or links on a page. Test your app with a screen reader to identify areas that are missing or need better text.

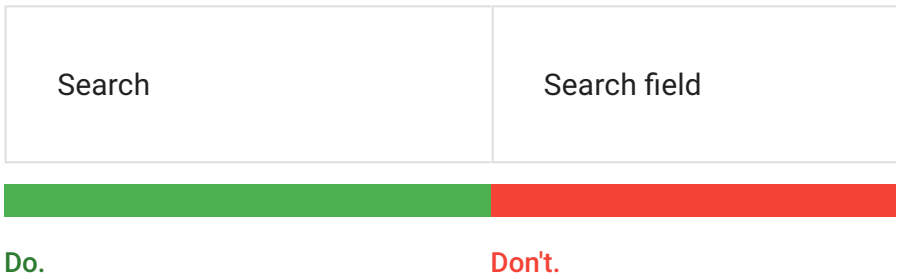
Be succinct

Keep content and accessibility text short and to the point. Screen reader users hear every UI element read aloud, so the shorter the text, the faster they can navigate.



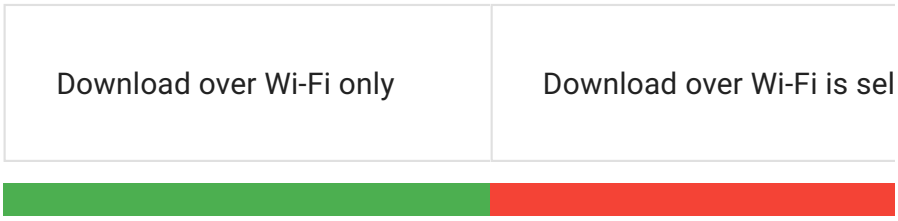
Avoid including control type or state in text

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.



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 check box 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".



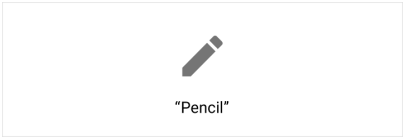
Indicate what an element does

Use action verbs to indicate what an element does, not what it looks like, so a visually impaired person can understand.

Ensure an element is given the same description in all instances where it's used.

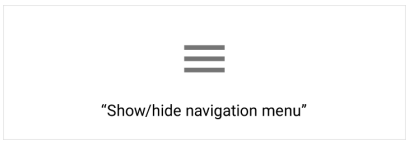
Do.

The description read aloud indicates the action represented by the icon.



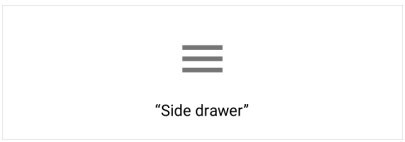
Don't.

Describing what the icon looks like doesn't make it clear what the action does.



Do.

Accessible text for a navigation menu could be "Show/hide navigation menu" (preferred) or "Show/Hide main menu" (acceptable).

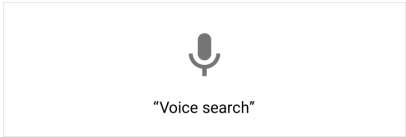


Don't.

When read aloud, the text "side drawer" doesn't indicate what action will occur.

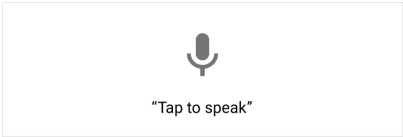
Don’t mention the exact gesture or interaction

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.



Do.

The command “voice search” describes the user task (search) paired with the input method (voice).



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 is to search in this example, 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 “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 toa 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” fea the user to tap twice when landing on an item without taking action. Android TalkBack will also announce any custo associated with an element. Use hint speech sparingly and only for complex UI.

Android Custom Actions

Hierarchy and focus

Use a clear and intuitive hierarchy to allow users of assistive technologies to navigate your design efficiently.

Screen readers

Screen readers give users multiple ways to navigate a screen, such as:

- Touch interface screen readers allow users to **run their finger over the screen to hear what is directly underneat** provides the user a quick sense of a whole interface, or to go quickly to a UI element from muscle memory. In Ta mode is called “explore by touch.” To select an item, the user must double tap.
- Users can also **move focus on screen by swiping** backwards or forwards. This reads pages linearly, from top to l that a user may hone in on a certain onscreen element. In TalkBack, this is called “linear navigation.”

Users may switch between both “explore by touch” and “linear navigation” modes. Some assistive technologies allo navigate between page landmarks, such as headings, when these landmarks use the appropriate semantic markup.

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.

Do.

The hierarchy of this screen is clear.

Don't.

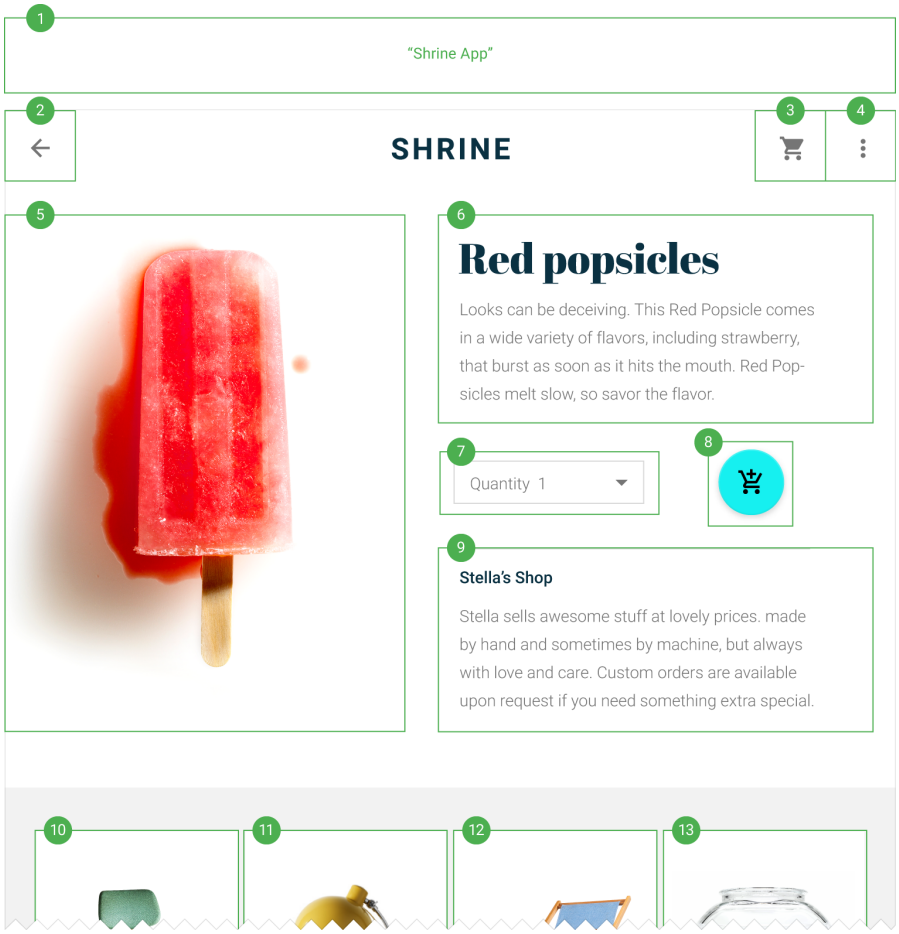
The hierarchy of this screen is jumbled.

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. In addition, 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.



The green circles indicate the order in which onscreen elements should r

Grouping

Group similar items under headings that communicate what the groupings are. These groups organize content spatially.

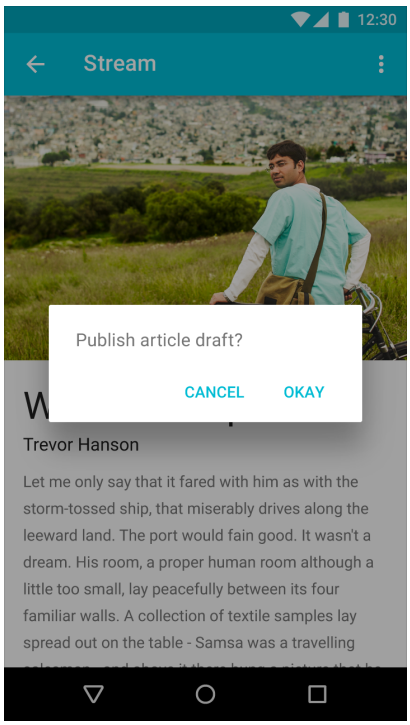
Transitions

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.

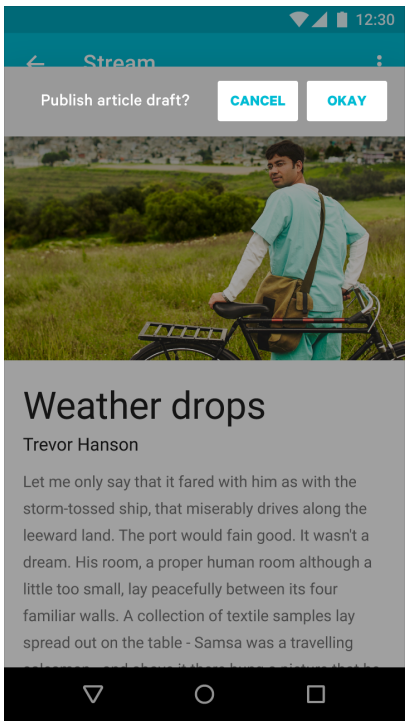
Implementation

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.



Do.

This screen is using the 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.

Your design should be tested with the platform accessibility settings turned on (both during and after implementation).

Other design considerations to include:

- **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.

Do.

This UI has scaled well with magnification and large text.

Don't.

This UI has not scaled well with magnification and large text. Portions of or cut off.

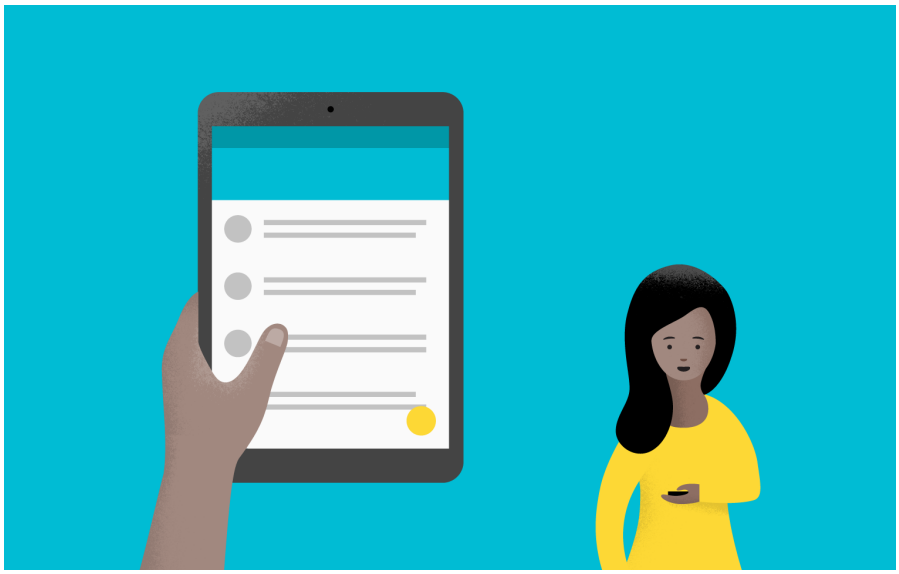
Any features with special accessibility considerations should be included in your help documentation. Make help documentation relevant, accessible, and discoverable. As an example, review this guide on how to use a screenreader with Google Drive.

Google Drive help for screen reader users is easily found on the internet.

Following these accessibility guidelines as a checklist 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.
- Have users with impairments test your app.
- Consider how individual elements should be accessible while also fitting together in coherent a 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.



Each user is unique - learn how they use their assistive technology.