

## What is material? Elevation and shadows

---

# Elevation and 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 cast shadows and reflect light.

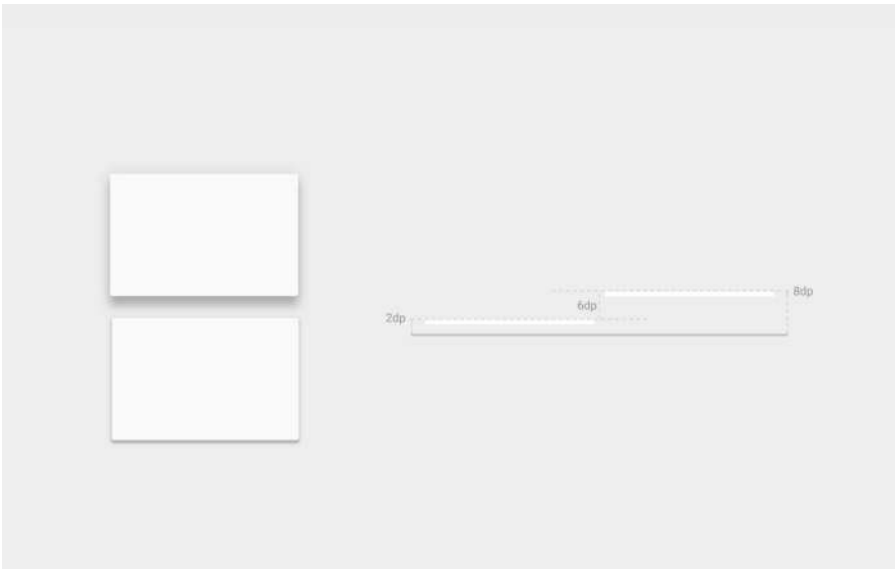
These qualities form a spatial model that is familiar to users and can be applied consistently across apps. Underpinning this spatial model are the concepts of elevation and shadow.

## 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.



Multiple elevation measurements for two objects

The images and values shown are for Android apps.



**Shadow effect**

Define shadows for custom elements

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.

Elevation (dp)	Component
24	Dialog
	Picker
32	

**Component elevations:**

- The resting elevation for a component type is consistent across apps (e.g., FAB elevation does not vary from 6dp in one app to 16dp in another app).
- Components may have different resting elevations across platforms, depending on the depth of the environment (e.g., TV has a greater depth than mobile or desktop).

**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.

23	
22	
21	
20	
19	
18	
17	
16	Nav drawer Right drawer Bottom Sheet
15	
14	
13	
12	Floating action button (FAB - pr
11	
10	
9	Sub menu (+1dp for each sub n
8	Menu Card (picked up state) Raised button (pressed state)
7	

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.

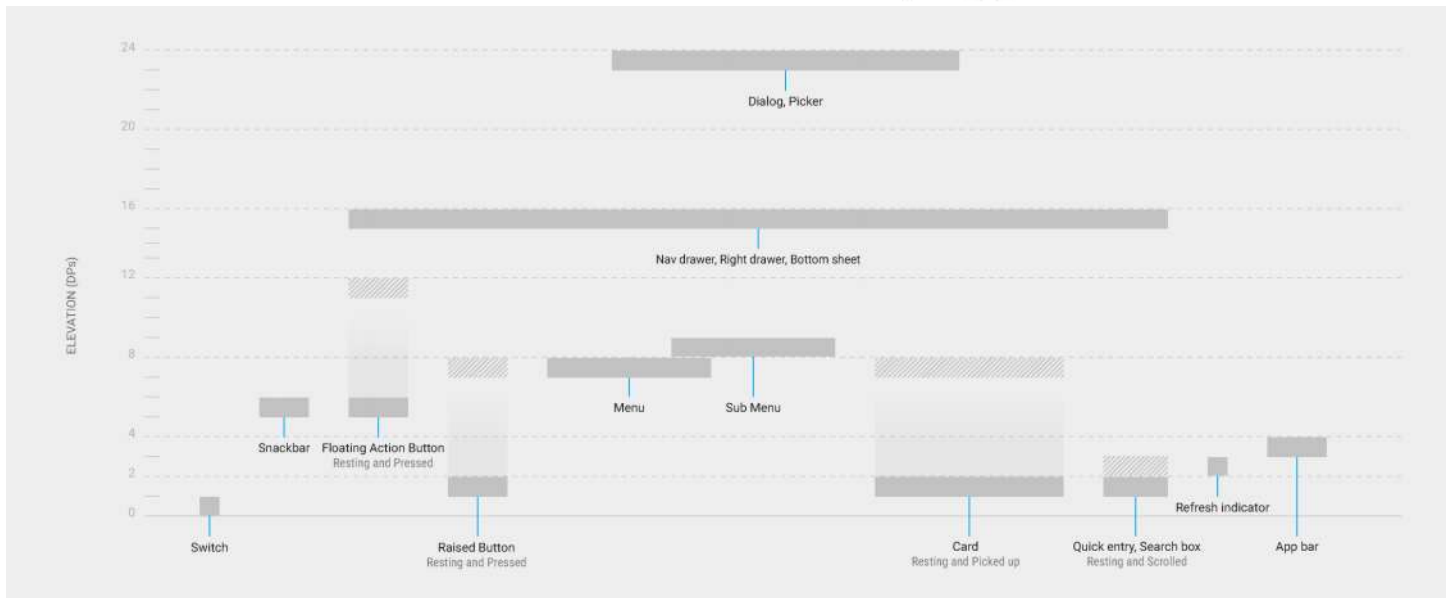
On a component level, components can move or be removed before they cause interference. For example, a floating action button (FAB) can disappear or move off-screen before a user picks up a card, or it can move if a snackbar appears.

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

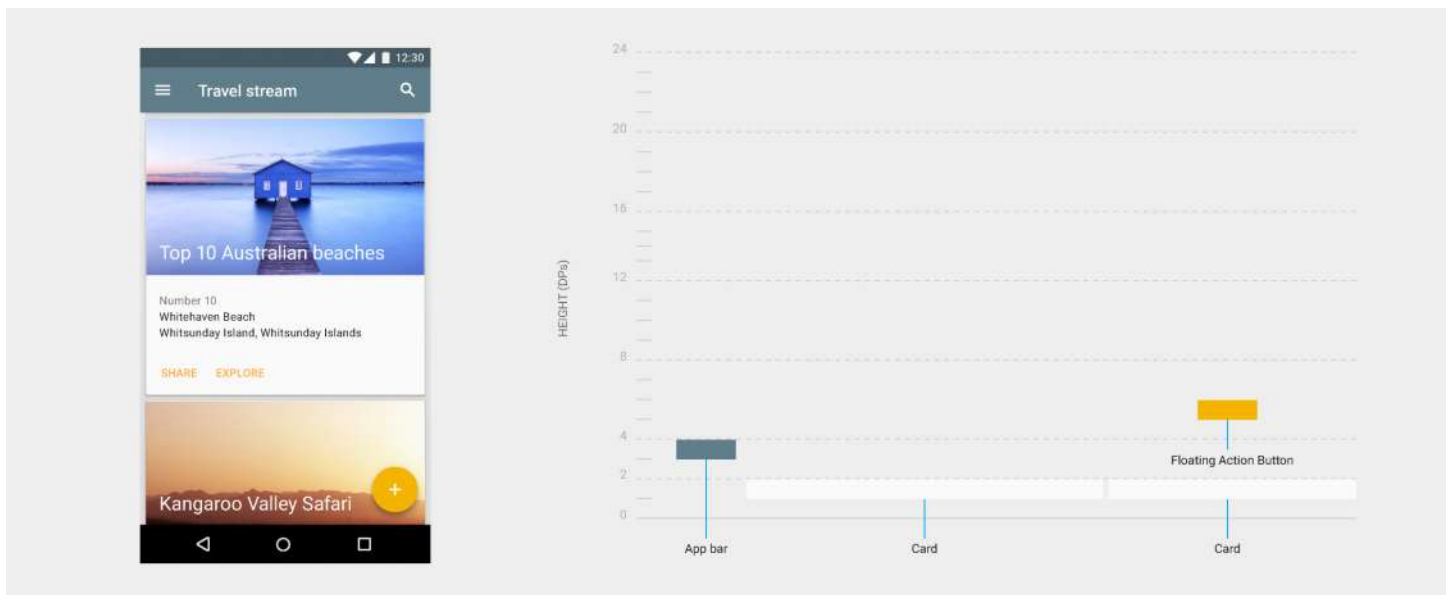
6	Floating action button (FAB - re Snackbar
5	
4	App Bar
3	Refresh indicator Quick entry / Search bar (scroll
2	Card (resting elevation) Raised button (resting elevatio Quick entry / Search bar (restin
1	Switch

Component elevation comparisons

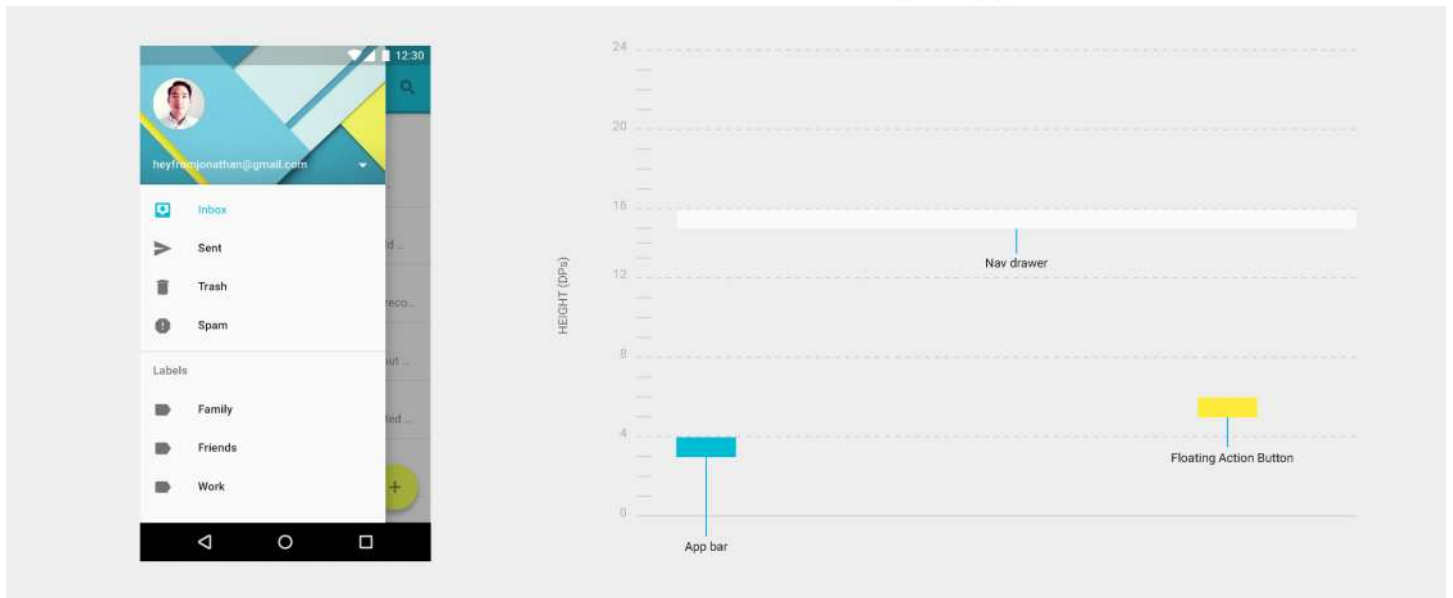
The following diagram compares various component resting elevations and dynamic elevation offsets.



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



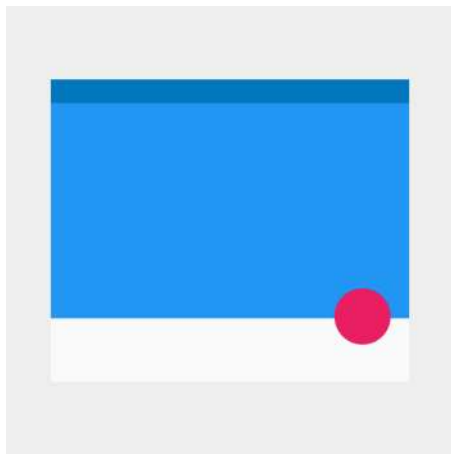
An example app layout with cards and a FAB, along with a cross-section diagram of its component elevations along its z-axis



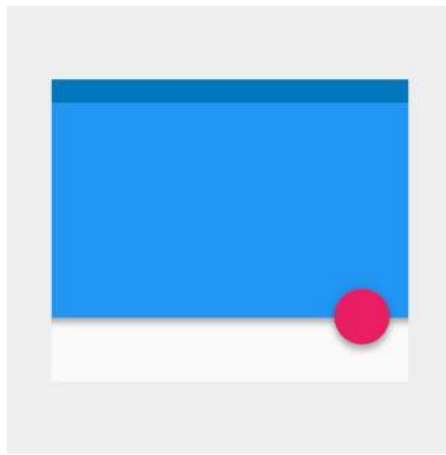
An example app layout with an open navigation drawer, along with a cross-section diagram of its component elevations along its z-axis.

## Shadows

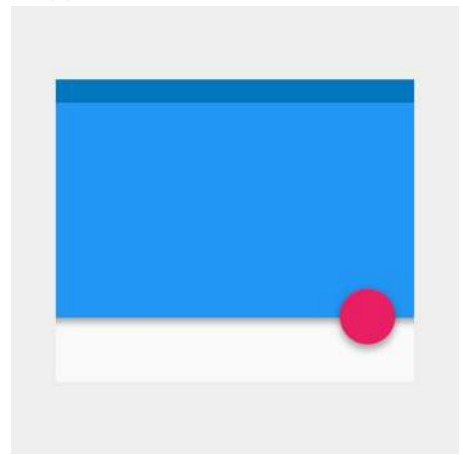
Shadows provide important visual cues about objects' depth and directional movement. They are the only visual cue amount of separation between surfaces. An object's elevation determines the appearance of its shadow.

**Don't.**

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

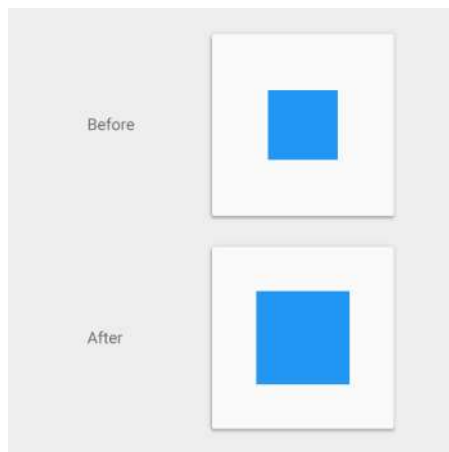
**Don't.**

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.

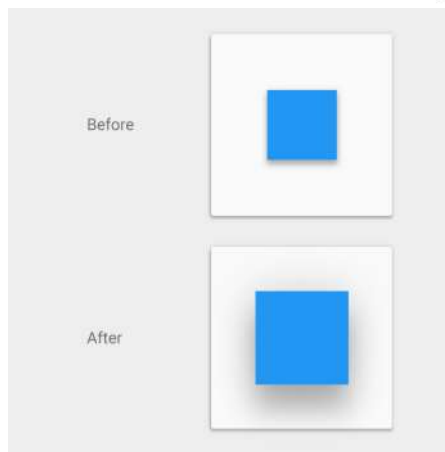
**Do.**

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

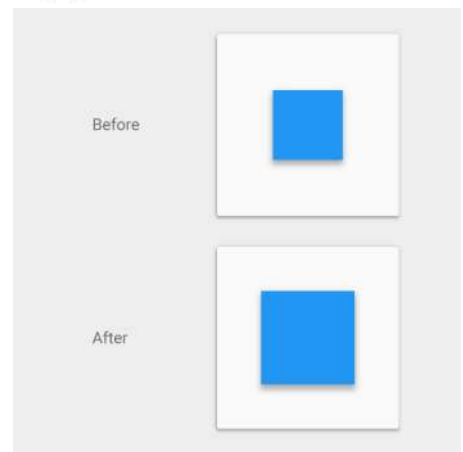
In motion, shadows provide useful cues about an object's direction of movement and whether the distance between increasing or decreasing.

**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.**

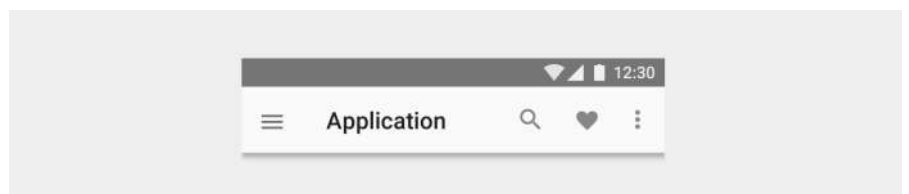
In this case, the consistent shadow helps the user understand that the object is changing shape as opposed to changing elevation.

## Component reference shadows

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

### App bar

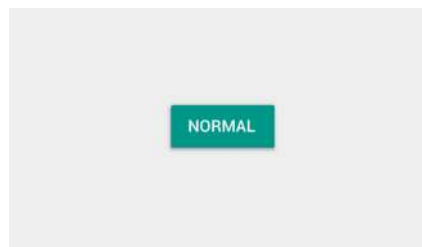
4dp



### Raised button

Resting state: 2dp

Pressed state: 8dp

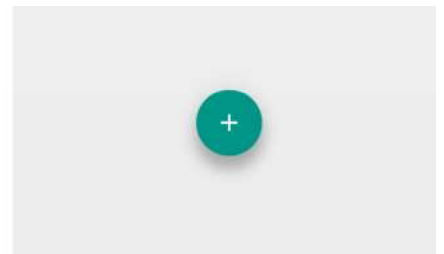
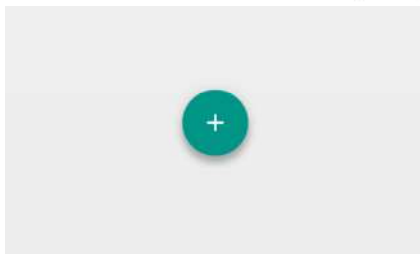




## Floating action button (FAB)

Resting state: 6dp

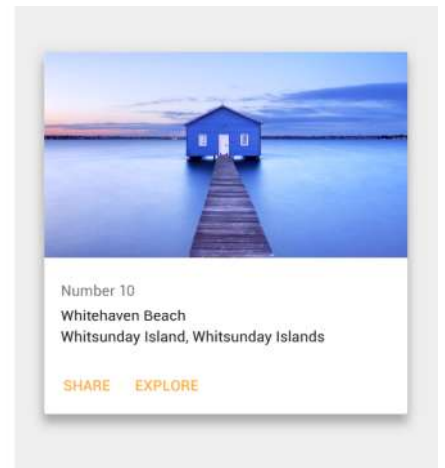
Pressed state: 12dp



## Card

Resting state: 2dp

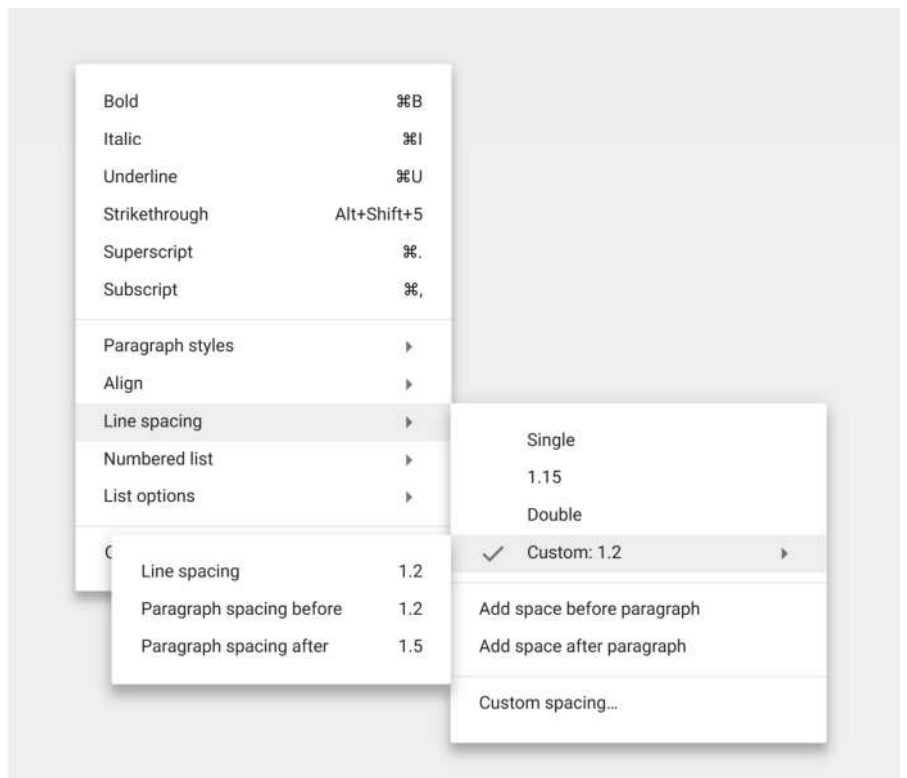
Picked-up state: 8dp



## Menus and sub menus

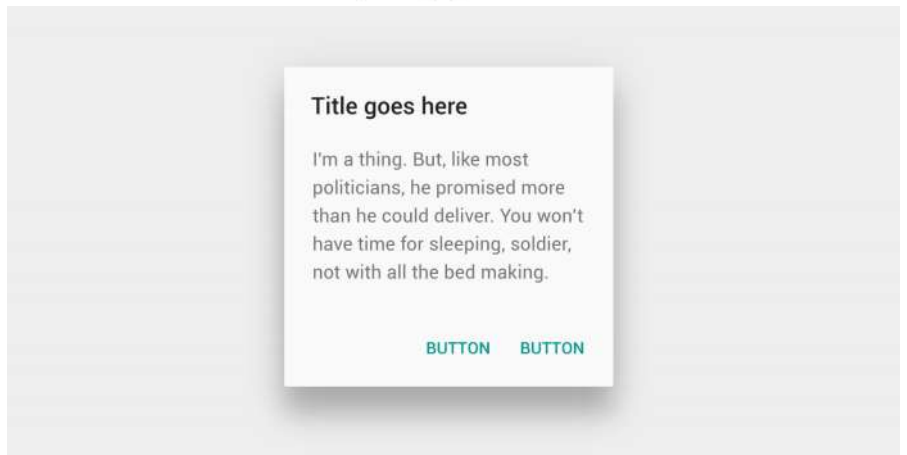
Menus: 8dp

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



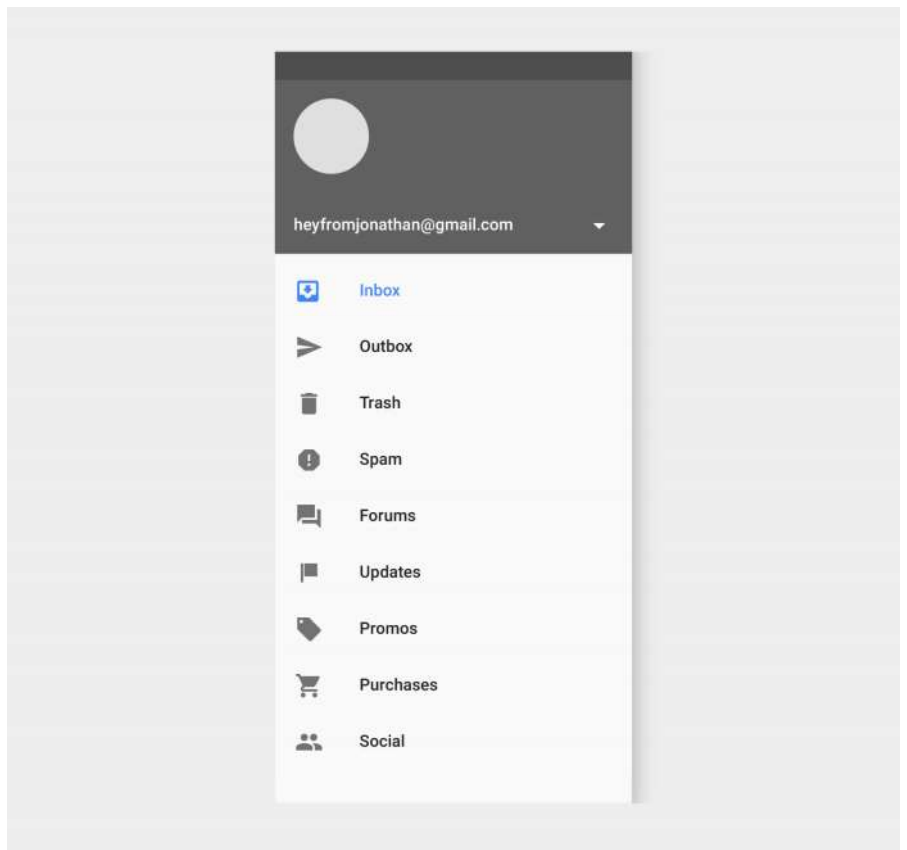
## Dialogs

24dp



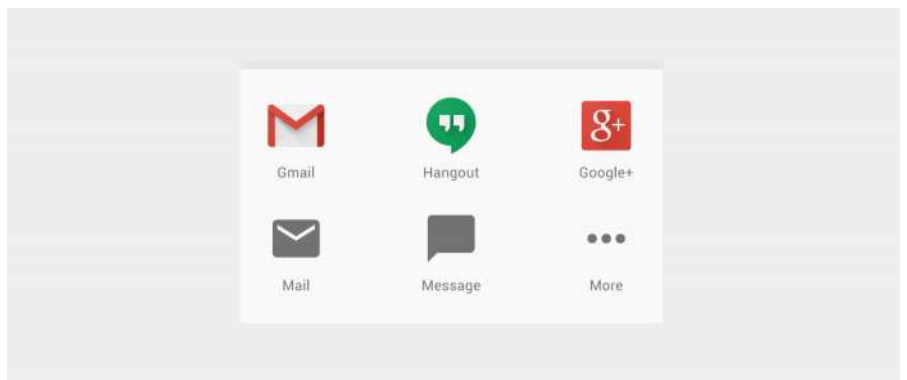
## Nav Drawer & Right drawer

16dp



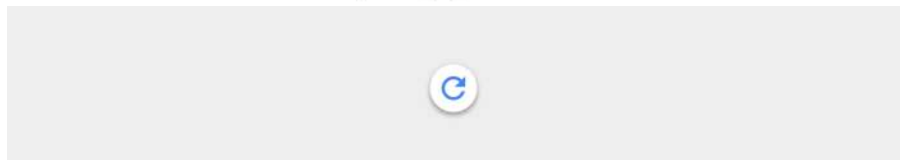
## Bottom sheet

16dp



## Refresh indicator

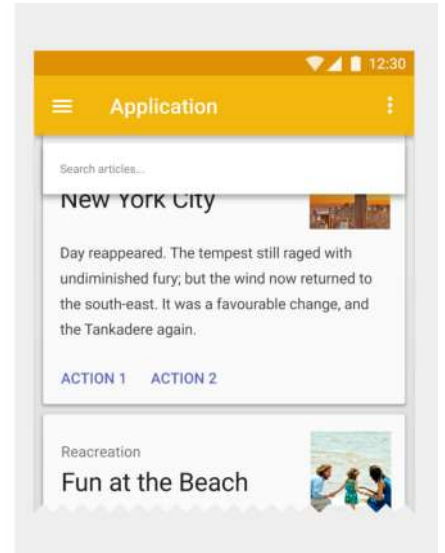
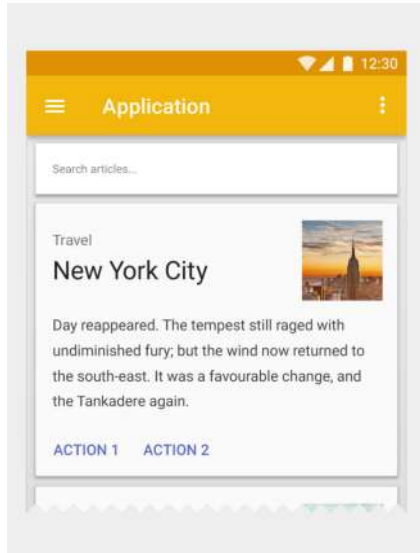
3dp



## Quick entry/Search bar

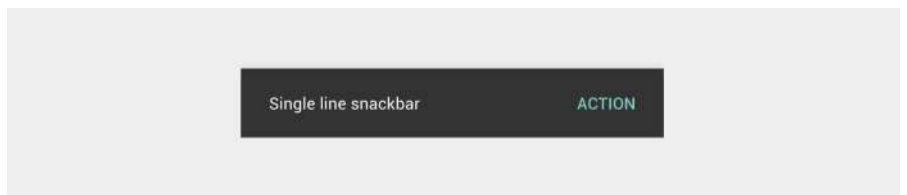
Resting state: 2dp

Scrolled state: 3dp



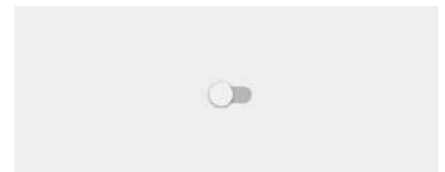
## Snackbar

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.

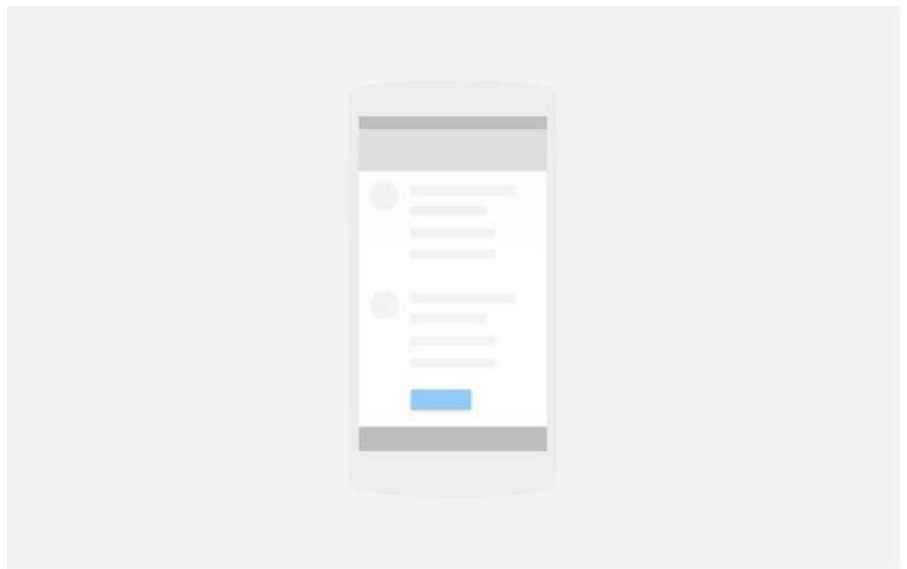
Parent-child specifics:

- 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



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



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

## 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.