A relative layout displays views in relative positions

A relative layout allows you to position views relative to their parent layout, or relative to other views in the layout.

You define a relative layout using the `<RelativeLayout>` element like this:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    ...>

    ...

</RelativeLayout>
```

The layout_width and layout_height specify what size you want the layout to be.

There may be other attributes too.

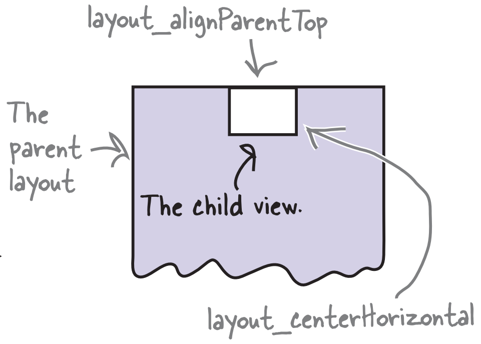
This is where you add any views.

Positioning views relative to the parent layout

If you want a view to always appear in a particular position on the screen, irrespective of the screen size or orientation, you need to position the view relative to its parent. As an example, here's how you'd make sure a button always appears in the top-center of the layout:

```
<RelativeLayout ... >
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/click_me"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

The layout contains the button, so the layout is the button's parent.



The lines of code:

```
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
```

mean that the top edge of the button is aligned to the top edge of the layout, and the button is centered horizontally in the parent layout. This will be the case no matter what the screen size, language, or orientation of your device:

Positioning views to the left or right

You can also position a view to the left or right of the parent layout. There are two ways of doing this.

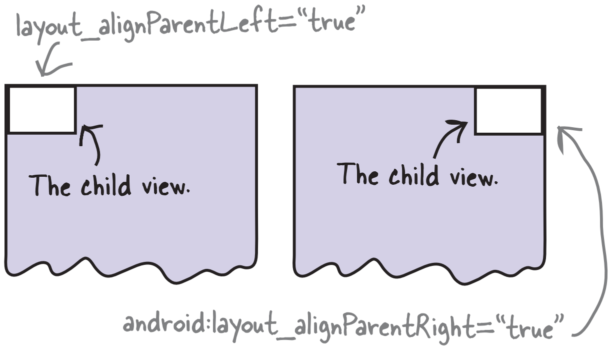
The first way is to explicitly position the view on the left or right using:

```
        android:layout_alignParentLeft="true"
```

or:

```
        android:layout_alignParentRight="true"
```

These lines of code mean that the left (or right) edge of the view is aligned to the left (or right) edge of the parent layout, regardless of the screen size, orientation, or language being used on the device.


layout_alignParentLeft="true"
The child view.
The child view.
android:layout_alignParentRight="true"

Use start and end to take language direction into account

For apps where the minimum SDK is *at least* API 17, you can position views on the left or right depending on the language setting on the device. As an example, you might want views to appear on the left for languages that are read from left to right such as English. For languages that are read from right to left, you might want them to appear on the right instead so that their position is mirrored.

To do this, you use:

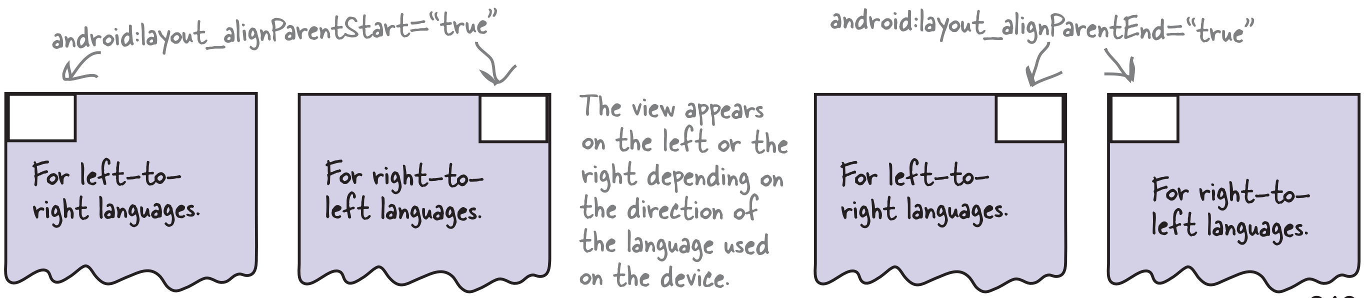
```
        android:layout_alignParentStart="true"
```

or:

```
        android:layout_alignParentEnd="true"
```

`android:layout_alignParentStart="true"` aligns the start edge of the view with that of its parent. The start edge is on the left for languages that are read from left to right, and the right edge for languages that are read from right to left.

`android:layout_alignParentEnd="true"` aligns the end edge of the view with that of its parent. The end edge is on the right for languages that are read from left to right, and the left edge for languages that are read from right to left.

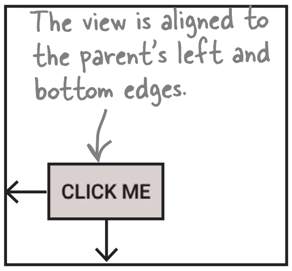

android:layout_alignParentStart="true"
For left-to-right languages.
For right-to-left languages.
The view appears on the left or the right depending on the direction of the language used on the device.
android:layout_alignParentEnd="true"
For left-to-right languages.
For right-to-left languages.

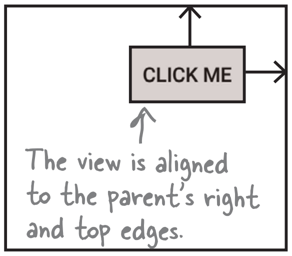
Attributes for positioning views relative to the parent layout

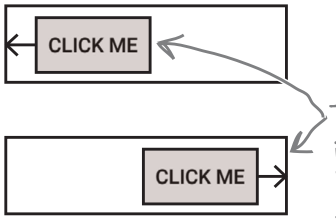
Here are some of the most common attributes for positioning views relative to their parent layout. Add the attribute you want to the view you're positioning, then set its value to `"true"`:

```
android:attribute="true"
```

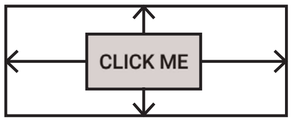
Attribute	What it does
layout_alignParentBottom	Aligns the bottom edge of the view to the bottom edge of the parent.
layout_alignParentLeft	Aligns the left edge of the view to the left edge of the parent.
layout_alignParentRight	Aligns the right edge of the view to the right edge of the parent.
layout_alignParentTop	Aligns the top edge of the view to the top edge of the parent.
layout_alignParentStart	Aligns the start edge of the view to the start edge of the parent.
layout_alignParentEnd	Aligns the end edge of the view to the end edge of the parent.
layout_centerInParent	Centers the view horizontally and vertically in the parent.
layout_centerHorizontal	Centers the view horizontally in the parent.
layout_centerVertical	Centers the view vertically in the parent.

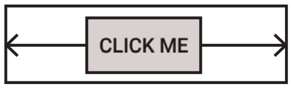


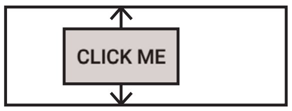




The start is on the left and the end is on the right for left-to-right languages. This is reversed for right-to-left languages.







Positioning views relative to other views

In addition to positioning views relative to the parent layout, you can also position views *relative to other views*. You do this when you want views to stay aligned in some way, irrespective of the screen size or orientation.

In order to position a view relative to another view, the view you're using as an anchor must be given an ID using the `android:id` attribute:

```
        android:id="@+id/button_click_me"
```

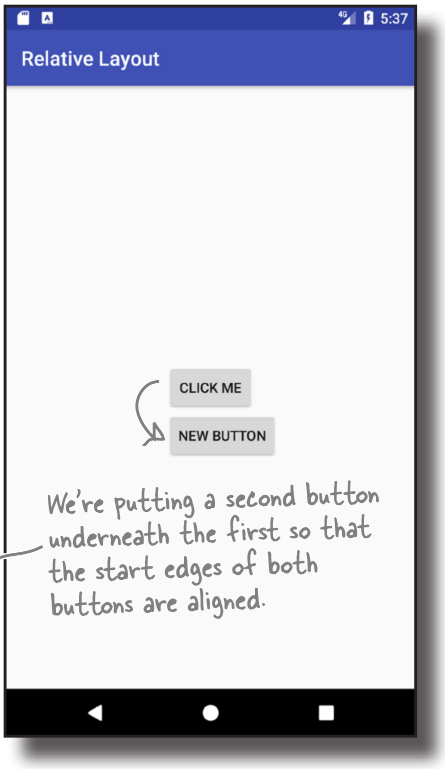
The syntax "`@+id`" tells Android to include the ID as a resource in its resource file *R.java*. You must include the "+" whenever you define a new view in the layout. If you don't, Android won't add the ID as a resource and you'll get errors in your code. You can omit the "+" when the ID has already been added as a resource.

Here's how you create a layout with two buttons, with one button centered in the middle of the layout, and the second button positioned underneath the first:

We're using this button as an anchor for the second one, so it needs an ID.

```
<RelativeLayout ... >
    <Button
        android:id="@+id/button_click_me"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Click Me" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignStart="@id/button_click_me"
        android:layout_below="@id/button_click_me"
        android:text="New Button" />
</RelativeLayout>
```



We're putting a second button underneath the first so that the start edges of both buttons are aligned.

The lines:

```
android:layout_alignStart="@id/button_click_me"
android:layout_below="@id/button_click_me"
```

When you're referring to views that have already been defined in the layout, you can use @id instead of @+id.

ensure that the second button has its start edge aligned to the start edge of the first button, and is always positioned beneath it.

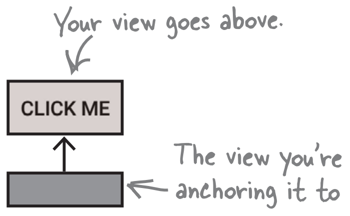
Attributes for positioning views relative to other views

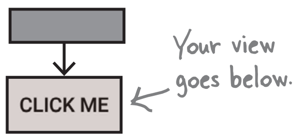
Here are attributes you can use when positioning views relative to another view. Add the attribute to the view you're positioning, and set its value to the view you're positioning relative to:

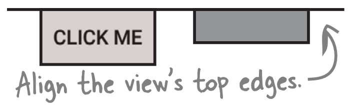
```
android:attribute="@+id/view_id"
```

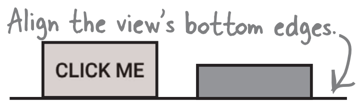
Remember, you can leave out the "+" if you've already defined the view ID in your layout.

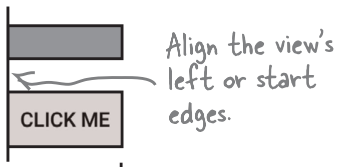
Attribute	What it does
layout_above	Puts the view above the view you're anchoring it to.
layout_below	Puts the view below the view you're anchoring it to.
layout_alignTop	Aligns the top edge of the view to the top edge of the view you're anchoring it to.
layout_alignBottom	Aligns the bottom edge of the view to the bottom edge of the view you're anchoring it to.
layout_alignLeft, layout_alignStart	Aligns the left (or start) edge of the view to the left (or start) edge of the view you're anchoring it to.
layout_alignRight, layout_alignEnd	Aligns the right (or end) edge of the view to the right (or end) edge of the view you're anchoring it to.
layout_toLeftOf, layout_toStartOf	Puts the right (or end) edge of the view to the left (or start) of the view you're anchoring it to.
layout_toRightOf, layout_toEndOf	Puts the left (or start) edge of the view to the right (or end) of the view you're anchoring it to.

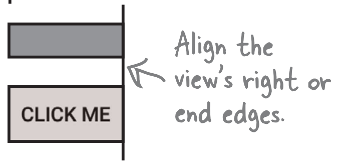

Your view goes above.
CLICK ME
The view you're anchoring it to

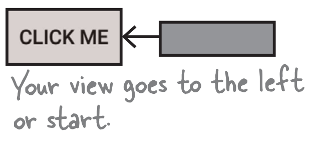

CLICK ME
Your view goes below.

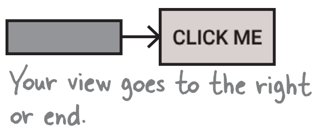

CLICK ME
Align the view's top edges.


Align the view's bottom edges.
CLICK ME


CLICK ME
Align the view's left or start edges.


CLICK ME
Align the view's right or end edges.


CLICK ME
Your view goes to the left or start.


CLICK ME
Your view goes to the right or end.