Build a Simple User Interface

The user interface for an Android app is built using a hierarchy of *layouts* ([ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html) objects) and *widgets* ([View](https://developer.android.com/reference/android/view/View.html) objects). Layouts are invisible containers that control how its child views are positioned on the screen. Widgets are UI components such as buttons and text boxes.



**Figure 2.** Illustration of how ViewGroup objects form branches in the layout and contain View objects

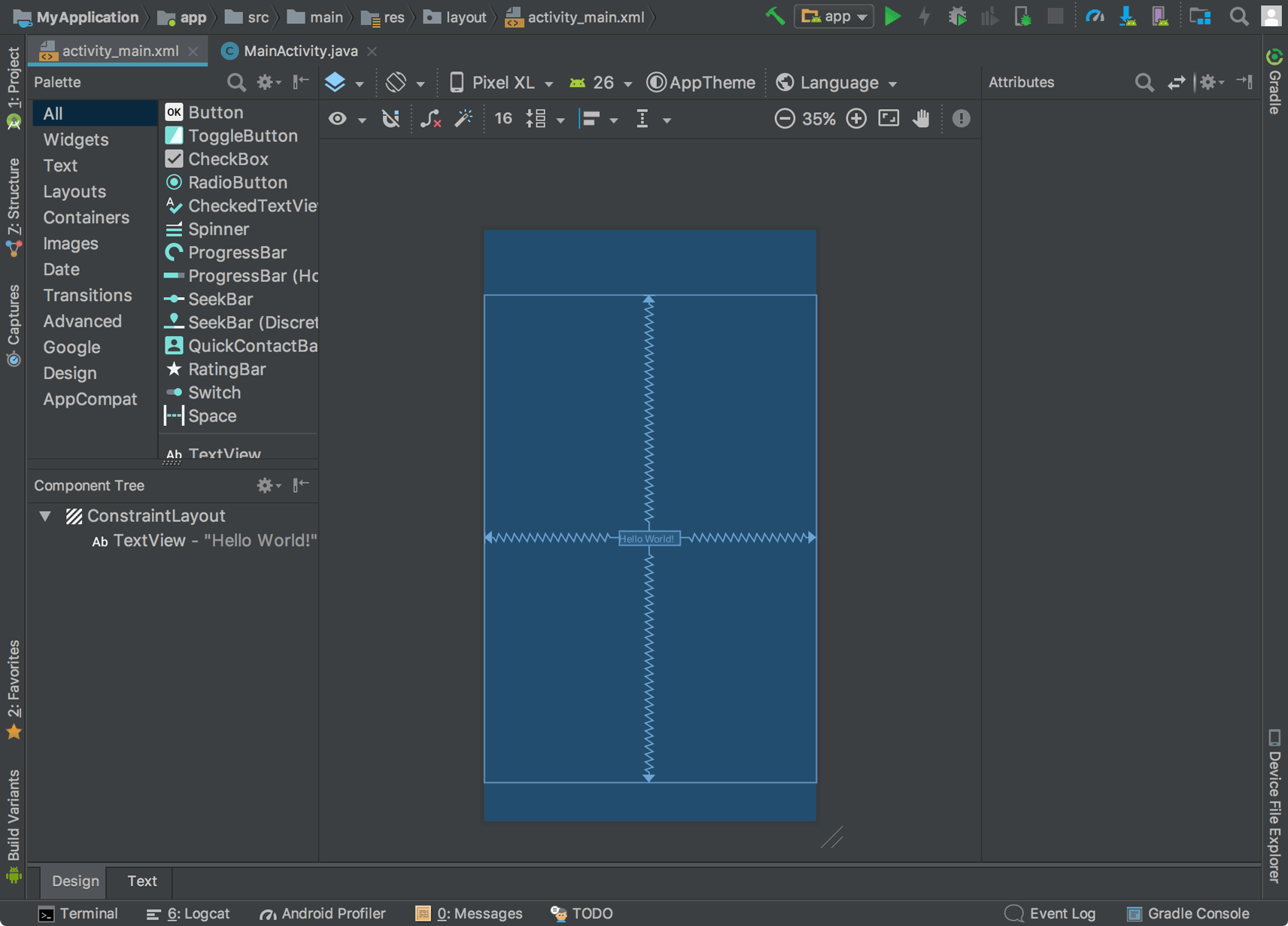
Android provides an XML vocabulary for ViewGroup and View classes, so most of your UI is defined in XML files. However, instead of teaching you to write some XML, this lesson shows you how to create a layout using Android Studio's Layout Editor, which makes it easy to build a layout by drag-and-dropping views.

## Open the Layout Editor

To get started, set up your workspace as follows:

1. In Android Studio's Project window, open **app > res > layout > activity\_main.xml**.
2. To make more room for the Layout Editor, hide the **Project** window by selecting **View > Tool Windows > Project** (or click **Project** https://developer.android.com/studio/images/buttons/window-project.png on the left side of Android Studio).
3. If your editor shows the XML source, click the **Design** tab at the bottom of the window.
4. Click **Select Design Surface** https://developer.android.com/studio/images/buttons/layout-editor-design.png and select **Blueprint**.
5. Click **Show** https://developer.android.com/studio/images/buttons/layout-editor-show-constraints.png in the Layout Editor toolbar and make sure **Show Constraints** is checked.
6. Make sure Autoconnect is off. The tooltip in the toolbar should read **Turn On Autoconnect** https://developer.android.com/studio/images/buttons/layout-editor-autoconnect-on.png (because it's now off).
7. Click **Default Margins** https://developer.android.com/studio/images/buttons/layout-editor-margin.png in the toolbar and select **16** (you can still adjust the margin for each view later).
8. Click **Device in Editor** https://developer.android.com/studio/images/buttons/layout-editor-device.png in the toolbar and select **5.5, 1440 × 2560, 560dpi (Pixel XL)**.

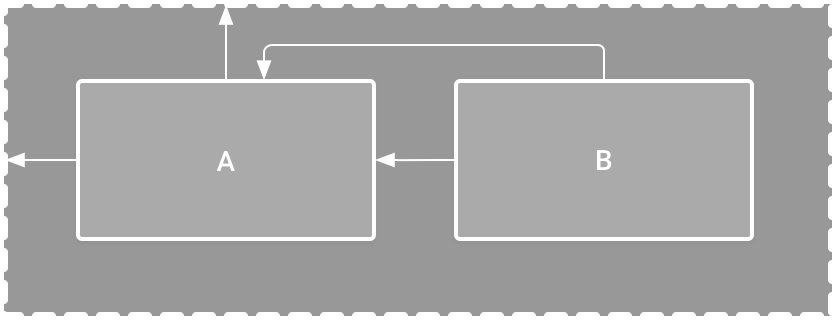
Your editor should now look as shown in figure 3.



**Figure 3.** The Layout Editor showing activity\_main.xml

The **Component Tree** window on the bottom-left side shows the layout's hierarchy of views. In this case, the root view is a ConstraintLayout, containing just one TextView object.

ConstraintLayout is a layout that defines the position for each view based on constraints to sibling views and the parent layout. In this way, you can create both simple and complex layouts with a flat view hierarchy. That is, it avoids the need for nested layouts (a layout inside a layout, as shown in figure 2), which can increase the time required to draw the UI.



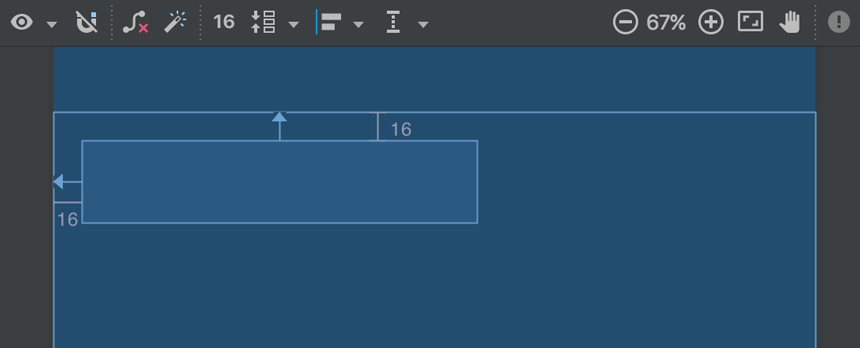
**Figure 4.** Illustration of two views positioned inside ConstraintLayout

For example, you can declare the following layout (in figure 4):

* View A appears 16dp from the top of the parent layout.
* View A appears 16dp from the left of the parent layout.
* View B appears 16dp to the right of view A.
* View B is aligned to the top of view A.

In the following sections, you'll build a layout similar to this.

## Add a text box



**Figure 5.** The text box is constrained to the top and left of the parent layout

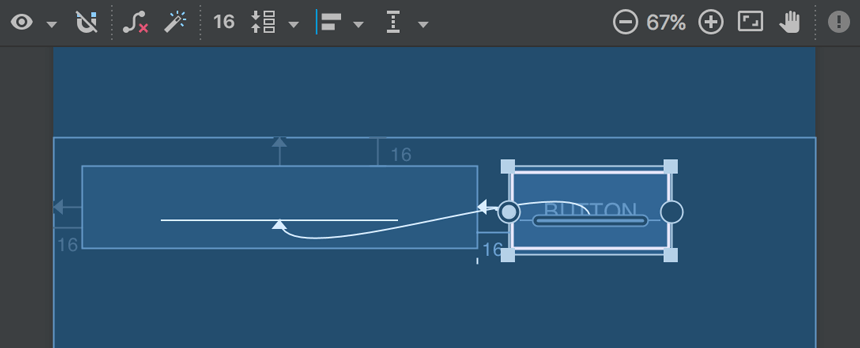
1. First, you need to remove what's already in the layout. So click **TextView** in the **Component Tree** window, and then press Delete.
2. In the **Palette**, click **Text** to show the available text controls.
3. Drag **Plain Text** into the design editor and drop it near the top of the layout. This is an [EditText](https://developer.android.com/reference/android/widget/EditText.html) widget that accepts plain text input.
4. Click the view in the design editor. You can now see the resizing handles on each corner (squares), and the constraint anchors on each side (circles).

For better control, you might want to zoom in on the editor using the buttons in the Layout Editor toolbar.

1. Click-and-hold the anchor on the top side, and then drag it up until it snaps to the top of the layout and release. That's a constraint—it specifies the view should be 16dp from the top of the layout (because you set the default margins to 16dp).
2. Similarly, create a constraint from the left side of the view to the left side of the layout.

The result should look like the screenshot in figure 5.

## Add a button



**Figure 6.** The button is constrained to the right side of the text box and its baseline

1. In the **Palette**, click **Widgets**.
2. Drag **Button** into the design editor and drop it near the right side.
3. Create a constraint from the left side of the button to the right side of the text box.
4. To constrain the views in a horizontal alignment, you need to create a constraint between the text baselines. So click the button, and then click **Edit Baseline** https://developer.android.com/studio/images/buttons/layout-editor-action-baseline.png, which appears in the design editor directly below the selected view. The baseline anchor appears inside the button. Click-and-hold on this anchor and then drag it to the baseline anchor that appears in the text box.

The result should look like the screenshot in figure 6.

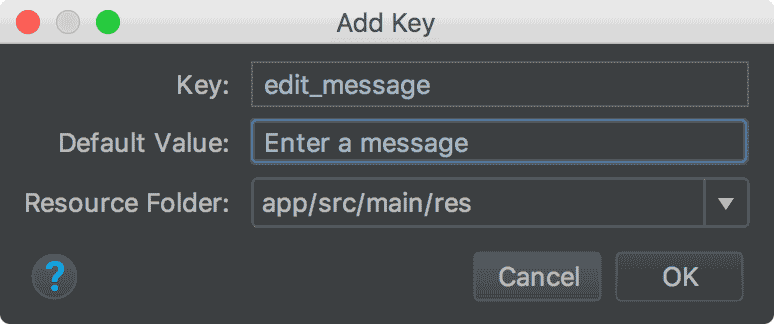
**Note:** You can also create a horizontal alignment using the top or bottom edges, but the button includes padding around its image, so the visual alignment is wrong if you align these views that way.

## Change the UI strings

To preview the UI, click **Select Design Surface** https://developer.android.com/studio/images/buttons/layout-editor-design.png in the toolbar and select **Design**. Notice that the text input is pre-filled with "Name" and the button is labeled "Button." So now you'll change these strings.

1. Open the **Project**window and then open **app > res > values > strings.xml**.

This is a [string resources](https://developer.android.com/guide/topics/resources/string-resource.html) file where you should specify all your UI strings. Doing so allows you to manage all UI strings in a single location, which makes it easier to find, update, and localize (compared to hard-coding strings in your layout or app code).

1. Click **Open editor** at the top of the editor window. This opens the [Translations Editor](https://developer.android.com/studio/write/translations-editor.html), which provides a simple interface for adding and editing your default strings, and helps keep all your translated strings organized.
2. 

**Figure 7.** The dialog to add a new string

Click **Add Key** https://developer.android.com/studio/images/buttons/add-sign-green-icon.png to create a new string as the "hint text" for the text box.

* 1. Enter "edit\_message" for the key name.
  2. Enter "Enter a message" for the value.
  3. Click **OK**.

1. Add another key named "button\_send" with a value of "Send".

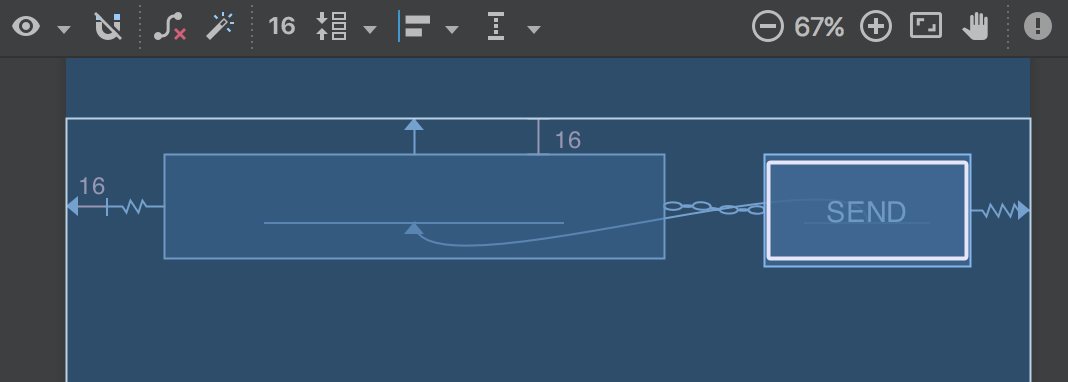
Now you can set these strings for each view. So return to the layout file by clicking**activity\_main.xml**in the tab bar, and add the strings as follows:

1. Click the text box in the layout and, if the **Attributes** window isn't already visible on the right, click **Attributes** https://developer.android.com/studio/images/buttons/window-properties.png on the right sidebar.
2. Locate the **text** property (currently set to "Name") and delete the value.
3. Locate the **hint** property and then click **Pick a Resource** https://developer.android.com/studio/images/buttons/pick-resource.png to the right of the text box. In the dialog that appears, double-click on **edit\_message** from the list.
4. Now click the button in the layout, locate the **text** property (currently set to "Button"), click **Pick a Resource** https://developer.android.com/studio/images/buttons/pick-resource.png, and then select **button\_send**.

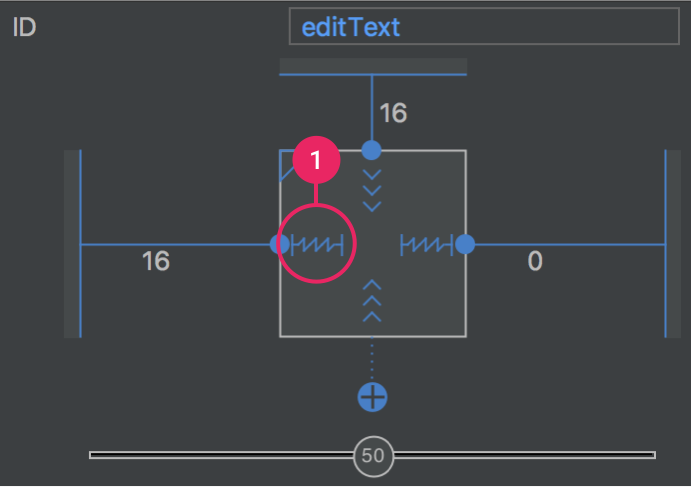
## Make the text box size flexible

To create a layout that's responsive to different screen sizes, you'll now make the text box stretch to fill all remaining horizontal space (after accounting for the button and margins).

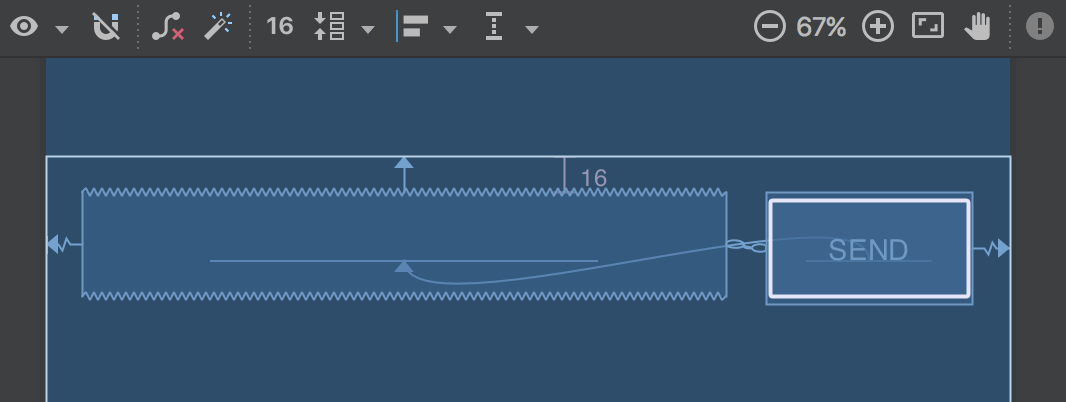
Before you continue, click **Show** https://developer.android.com/studio/images/buttons/layout-editor-design.png in the toolbar and select **Blueprint**.



**Figure 8.** The result of choosing **Create Horizontal Chain**



**Figure 9.** Click to change the width to **Match Constraints**



**Figure 10.** The text box now stretches to fill the remaining space

1. Select both views (click one, hold Shift, and click the other), and then right-click either view and select **Chain > Create Horizontal Chain**. The layout should appear as shown in figure 8.

A [chain](https://developer.android.com/training/constraint-layout/index.html#constrain-chain) is a bidirectional constraint between two or more views that allows you to lay out the chained views in unison.

1. Select the button and open the **Attributes** window. Using the view inspector at the top of the **Attributes** window, set the right margin to 16.
2. Now click the text box to view its attributes. Click the width indicator twice so that it is set to **Match Constraints**, as indicated by callout 1 in figure 9.

"Match constraints" means that the width expands to meet the definition of the horizontal constraints and margins. Therefore, the text box stretches to fill the horizontal space (after accounting for the button and all margins).

Now the layout is done and should appear as shown in figure 10.

If it seems your layout did not turn out as expected, click below to see what your the XML should look like and compare it to what you see in the **Text** tab. (If your attributes appear in a different order, that's okay.)