



5. Lesson 13/03/23

Layouts and Event handling

Goals & Contents

Goals:

- Clarify some of the notions seen in practice
- Implement a “realistic app”

Contents:

- The layout editor and *ConstraintLayout*
- Event handling

Apps Development Summary

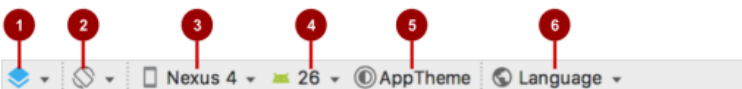
1. Design the UI
2. Implement the Behaviour (snippet of code to respond to the user interactions)

Associate event handlers to UI widgets or update the info on the widgets.

The layout editor and Constraint Layout

- Already used these tools in practice
- We will see a more complete overview of the available features

Layout editor main toolbar



1. Select Design Surface:

- Select **Design** to display a color preview of the UI elements in the layout.

- Select **Blueprint** to show only outlines of the elements.
- Select **Design + Blueprint** to both of them.

2. Orientation in Editor:

- Select **Portrait or Landscape** to show the preview in a vertical or horizontal orientation. The orientation setting lets you preview the layout orientation without running the app.

3. **Device in Editor:** select the device type (phone, tablet, Android TV...)

4. **API Version in Editor:** select the version Android to use the preview

5. **Theme in Editor:** select a theme to apply to the preview

6. **Locale in Editor:** select the language and locale for the preview

Preview layouts

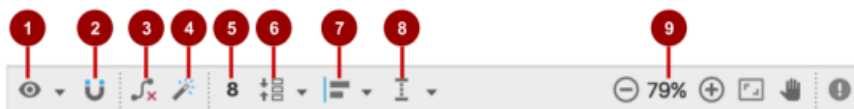
It is possible to **preview an app's layout** with a horizontal orientation, **without having to run the app** on emulator or device.

1. Click Orientation in Editor button
2. Choose **Switch to Landscape** or **Switch to Portrait**

Preview layout with different devices:

1. Click Device in Editor button
2. Choose device

ConstraintLayout toolbar in layout editor



1. **Show:** select **Show Constraints** and **Show Margins** to show them in the preview, or to stop showing them.
2. **Autoconnect:** enable or disable Autoconnect. With Autoconnect enabled, you can drag any element (such as a Button) to any part of a layout to generate constraints against the parent layout.
3. **Pack: clear All Constraints:** Clear all constraints in the entire layout
4. **Infer Constraints:** create the constraint by inference
5. **Default Margins:** set the default margins

6. **Pack:** pack or expand the selected elements
7. **Align:** align the selected elements
8. **Guidelines:** add vertical or horizontal guidelines
9. **Zoom controls:** zoom in or out

ConstraintLayout handles

A constraint is a connection or alignment to:

- Another UI element
- The parent layout
- An invisible guideline

Each constraint appears as a line extending from a circular handle.

Constraint handle

To create a constraint:

- **click** a constraint handle
- **drag the circle** to another constraint handle or to a parent boundary

The constraint is represented by a zigzag line

Resizing handle

- Drag the square resizing handles to resize the element
- **Warning:** doing so **hard-codes** the width and the height dimensions, it is better to avoid for most elements because **hard-coded dimensions don't adapt to different screen densities**.

Align elements by baseline

Align the UI elements that contains texts with another UI element that contains text.

A baseline constraint lets you constrain the elements so that the text baselines match.

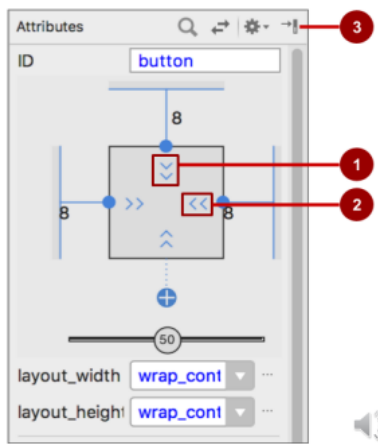
1. Click the baseline constraint button
2. Drag from baseline to other element's baseline

Attributes pane

The **Attributes** pane offers access to all of the XML attributes you can assign to a UI element.

- Click the Attributes tab
- Attributes pane includes:
 - Margin controls for positioning
 - Attributes such as *layout_width*




Attributes pane view inspector

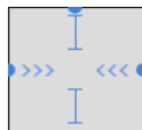


1. Vertical view size control specifies *layout_height*
2. Horizontal view size control specifies *layout_width*
3. Attributes pane close button

layout_width and layout_height

They change with size controls

-  **match_constraint**: Expands element to fill its parent
-  **wrap_content**: Shrinks element to enclose content
-  Fixed number of dp (density-independent pixels)



Set attributes

To view and edit all attributes for element:

- Click **Attributes** tab
- Select element in design, blueprint or component tree
- Change most-used attributes
- Click “View more attributes”

Create layout variant for landscape

To create a variant of the layout strictly for the horizontal orientation, leaving the vertical orientation layout alone:

- Click **Orientation**
- Choose **Create Landscape Variation**
- Layout variant created: *activity_main.xml* (automatically created from Android Studio)
- Edit the layout variant as needed

You can change this layout, which is specifically for horizontal orientation, without changing the original portrait (vertical) orientation.

Event Handling

Events

Something that happens:

- **In the UI:** click, tap, drag
- **In the device:** DetectedActivity such as walking, driving, tilting

Event Handlers

Methods that do something in response to an event.

- An **Event Handler** is a method triggered by a specific event and that does something in response to such event.
- An **Event Listener** is an interface in the View class that contains a single callback method. This method will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.

Click Handler in Java

```
final Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String msg = "Hello Toast!";
        Toast toast = Toast.makeText(this, msg, duration);
        toast.show();
    }
});
```

Updating a View

To update a View the code must **first instantiate an object from the View**. The code can then update the object, which updates the screen.

To refer to the View in the code, **use the *findViewById()*** method of the View class, which looks for a View based on the resource id.