BUILD X : ANDROID

Lecture One: Layouts & Views

Recap

Recap – What are we doing?

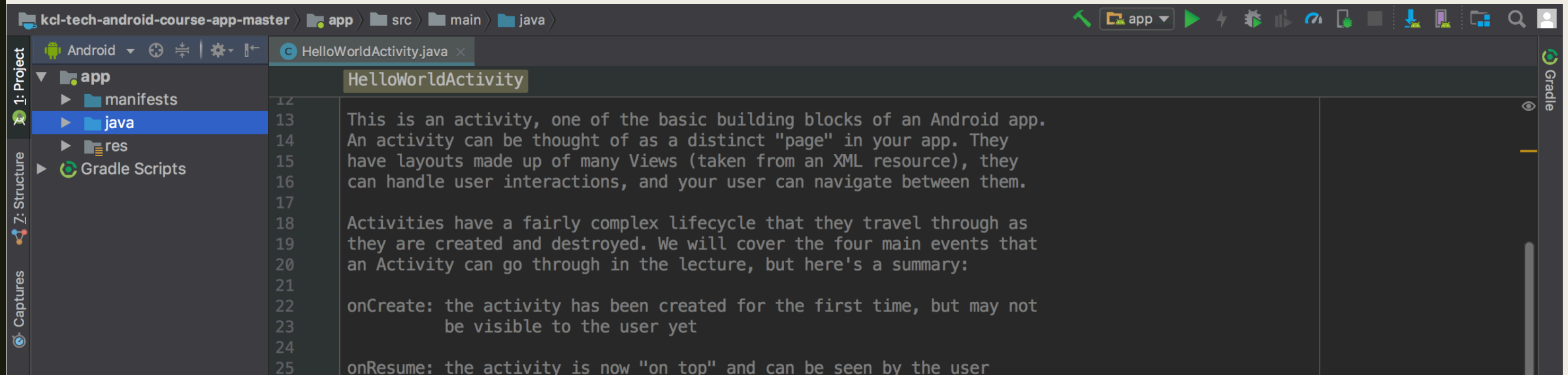
We're going to build a basic todo list app, **KCL Tech Todo**.

If you want a preview, it is **available on Google Play.**

- Search for "**KCL Tech Todo"** on Google or on the Play Store.

Recap – Environment Setup

- You should have **Android Studio** installed on your machine.
- You should have the blank project downloaded (https://tinyurl.com/android-blank-project)
- You should be able to run the blank project on your phone or on your emulator



Recap – Components of an Android App

- **Activities** – a page in your app. This is a Java class.
  - These have a complex “create, resume, pause, stop” lifecycle.
- **Resources** - files to store values that your app relies on. These are XML files.
  - Strings, dimensions, colours, layouts, styles, etc.
- **Layouts** – a specific type of resource. **This is an XML file**.
  - These describe the structure and appearance of your app.
- **AndroidManifest.xml** – a spec-sheet for your app
  - Describe what your app needs, what it contains, and what it can do.

Layouts

A layout is an **XML resource file** that tells the device about the **structure and appearance** of sections of your app.

> *"Put this here, now put that there, and put this next to that, etc..."*

Q: How do we know **which layout** to use for **which activity**?

Hint: One line of code...

O Layouts! Wherefore art thou?

All layouts live in one folder:

/app/res/layout

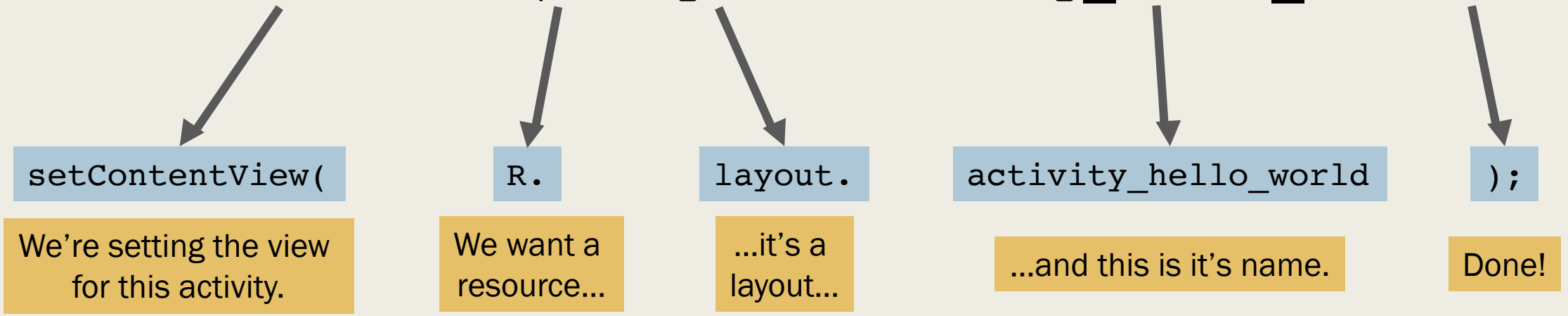
We tell Android which layout to use for a certain activity by adding one line:

```
setContentView(R.layout.activity_hello_world);
```

What's going on here?

O Layouts! Wherefore art thou?

```
setContentView(R.layout.activity_hello_world);
```

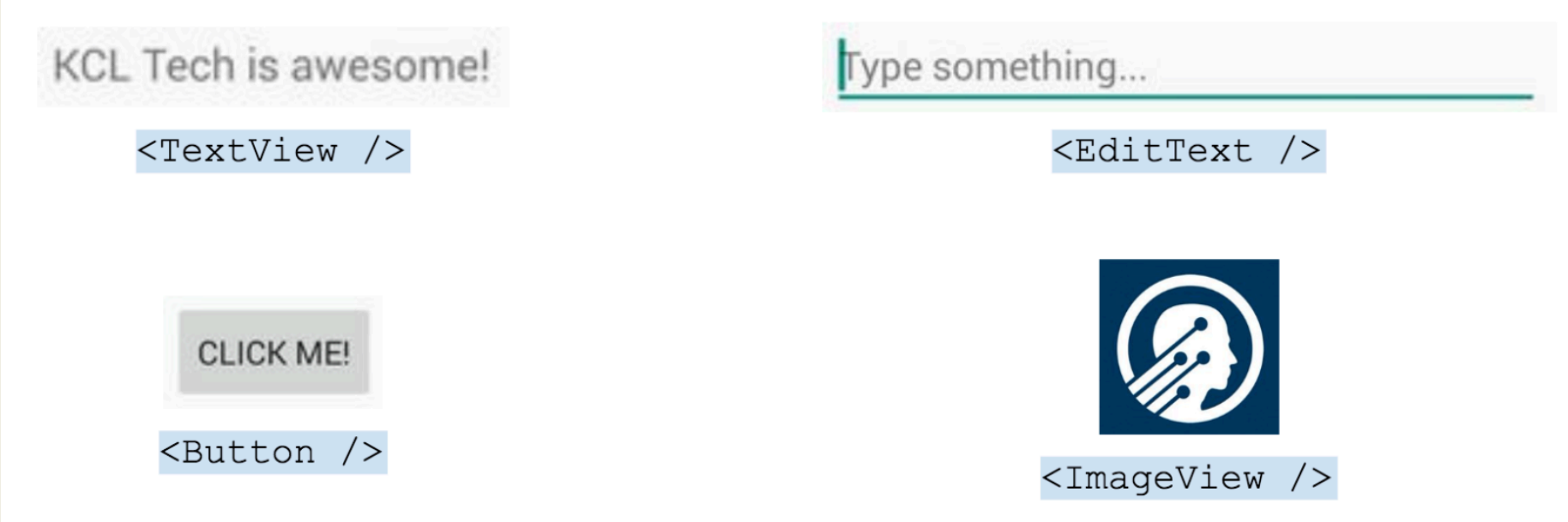



CALM DOWN
I GOT THIS

Parts of a Layout: Views

A layout will usually **contain many views.**

Views are added to a layout in their **XML files.**



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

  <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="KCL Tech is awesome!" />

  <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Type something..." />

  <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Click me!" />

</LinearLayout>
```

What is this?

Text View

Edit Text View

Button View

View Attributes

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

These two **attributes** are **required on every view.**

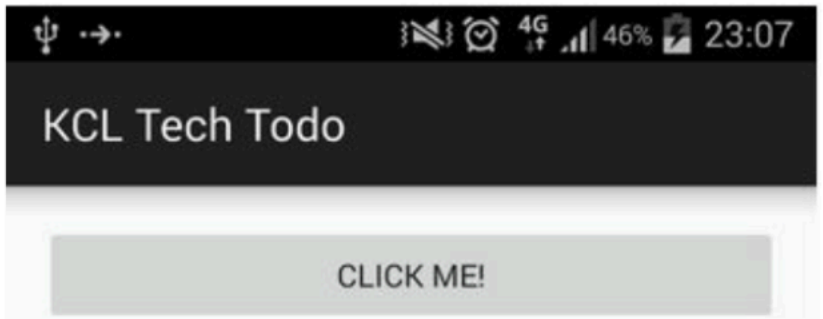
- There are others, like **android:text**, **android:padding**, **android:visibility**, etc.
- These attributes **define everything** about your view.

These define the height and width of the view. You can use a number of pixels here, or another measurement that Android supports (more later).

But what *are* "**match_parent**" and "**wrap_content**"?

View Attributes

`match_parent`



`wrap_content`

Both can be used for width or height, and you can use them in any combination.

- **match_parent**: make this view as wide/tall as the view surrounding it.
- **wrap_content**: make this view just wide/tall enough to show what it contains, but no more.

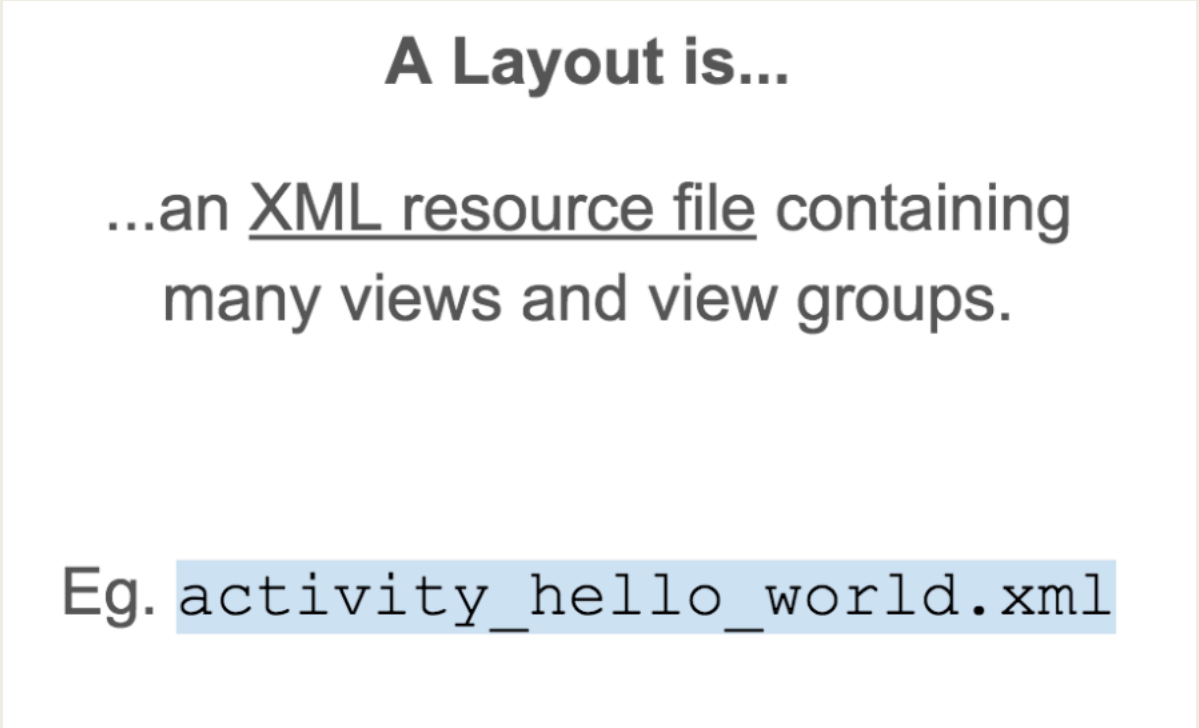
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

  <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="KCL Tech is awesome!" />

  <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Type something..." />

  <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click me!" />

</LinearLayout>
```

What is this?

Also a view!

//Side Note: Stupid Names

LinearLayout is an example of a View Group, which we'll get to in a bit.
But just to give some clarity:



LinearLayout is a special type of **View**, which **is part of a Layout resource file**.

Now You Try!

You have 10 minutes.

Update your “Hello world” layout to include a **TextView**, **EditText** and **Button**.

You’ll have to use the tag names mentioned above, the two attributes mentioned, as well as the **`android:text`** attribute for **TextViews** and **Buttons**, and the **`android:hint`** attribute if you want a prompt for your **EditText**.

Parts of a Layout: View Groups

Some special views can contain other views – these are called **View Groups**.

- The views inside a view group are called it's **children**.
- The view group containing views is the **parent** of those views.

- You cannot see a **view group**, only it's children views inside it.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

  <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="KCL Tech is awesome!" />

  <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Type something..." />

  <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click me!" />

</LinearLayout>
```

//Side Note: Stupid Names

A Layout is...

...an XML resource file containing many views and view groups.

Eg. `activity_hello_world.xml`

A View Group is...

...a special type of view that can contain and organise other views.

Eg. `<LinearLayout />`, `<ScrollView />`

<ScrollView />

The simplest View Group that exists!

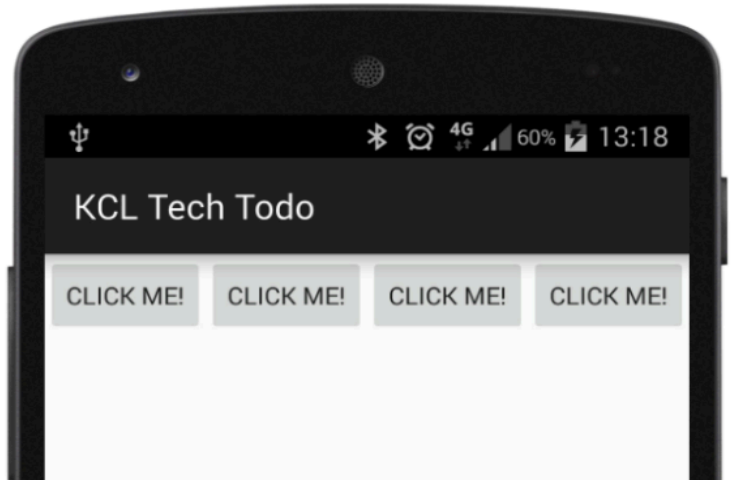
- Has **only** one child (often another View Group)
- **Adds a scrollbar** if it gets too tall/wide

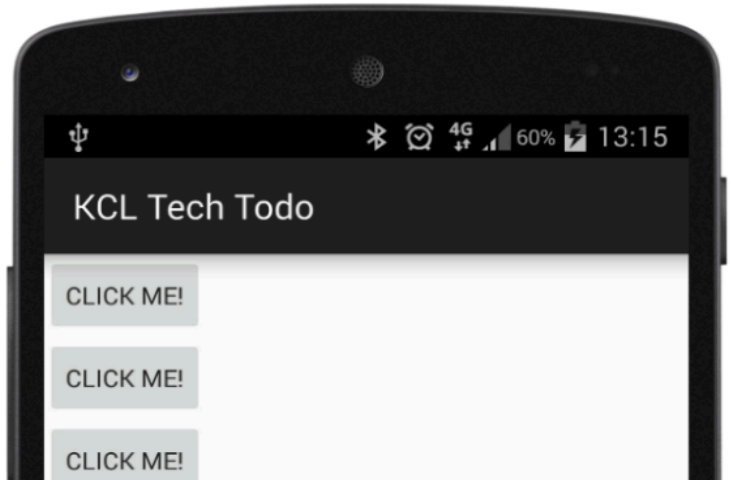
- **That's it.**

<LinearLayout />

Does one thing: **stack views**.

- Accepts many children.
- Stacks children **horizontally or vertically**.
  - Specified by **android:orientation** attribute.





<RelativeLayout />

A step up in complexity.
Let's you specify structures like:

> "put this view to the *left* of that view,
> put this view *underneath* this view;
> make this view line up with the right edge of another, etc..."

- Accepts many children
- Positions children relative to each other, or the parent.

Parts of a Layout: Other Resources

Resources we're going to use live in this folder:

/app/res/values

Specifically:

- Strings – /app/res/values/**strings.xml**
- Dimensions - /app/res/values/**dimens.xml**
- Styles - /app/res/values/**styles.xml**

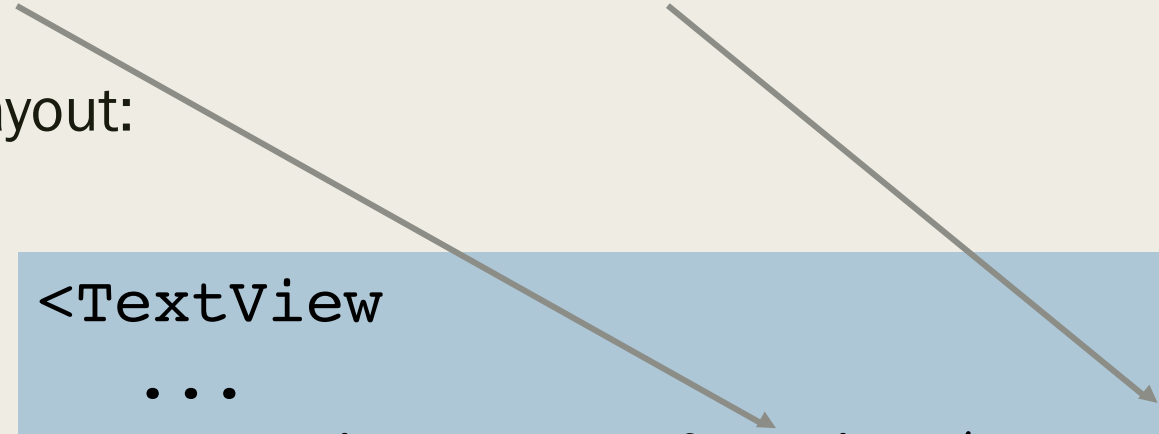
Using String Resources

In the **strings.xml** file:

```
<string name="hello_world">Hello World!</string>
```

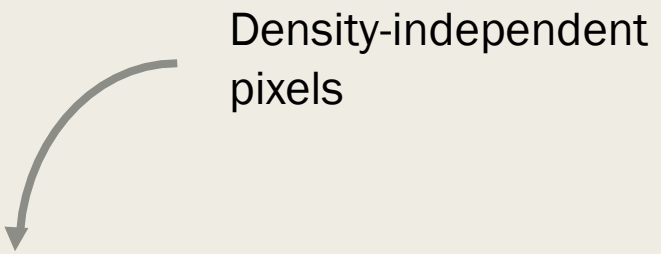
In the layout:

```
<TextView
   ...
   android:text="@string/hello_world" />
```



Using Dimension Resources

In the **dimens.xml** file:




```
<dimen name="hello_world_height">24dp</dimen>
```

In the layout:

```
<TextView
   ...
   android:layout_height="@dimen/hello_world_height" />
```

Using Style Resources

In the **styles.xml**

```
<style name="hello_world_text"
   <item name="android:textStyle">bold</item>
   <item name="android:textColor">#00ff00</item>
   <item name="android:textSize">18sp</item>
</style>
```

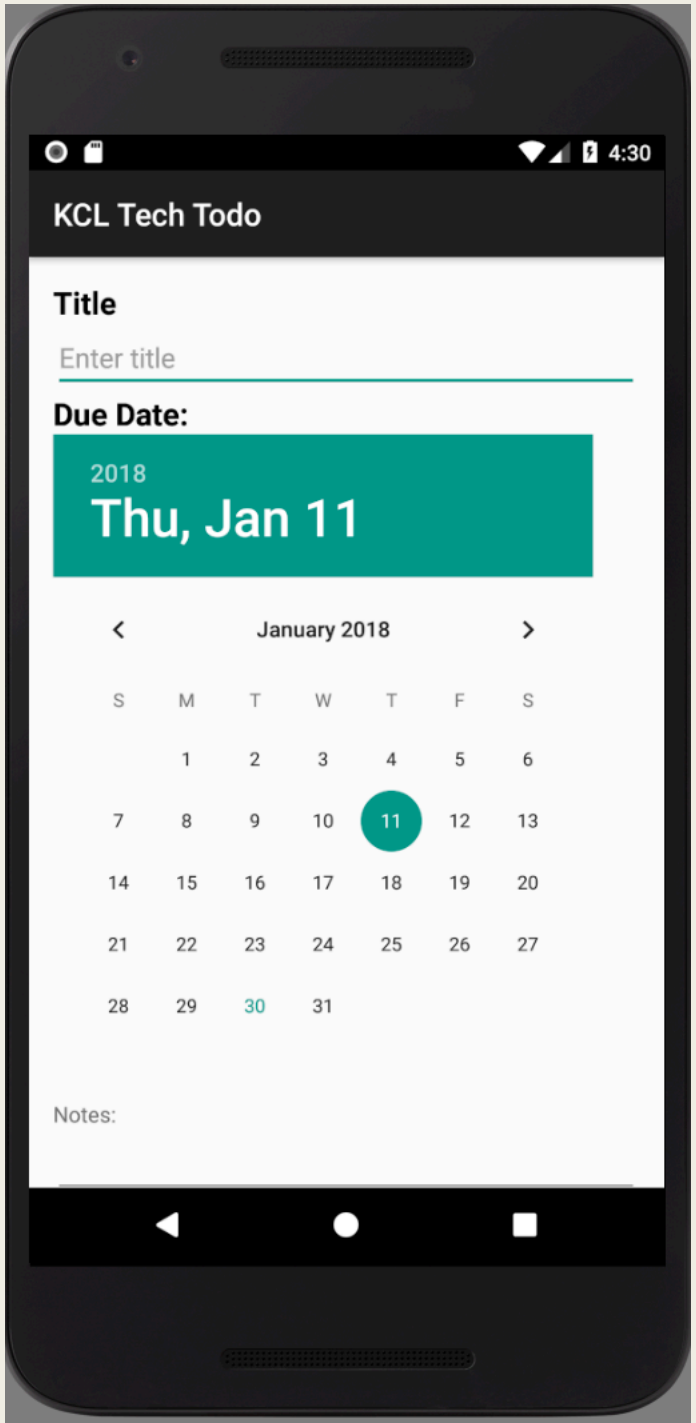
Using Style Resources

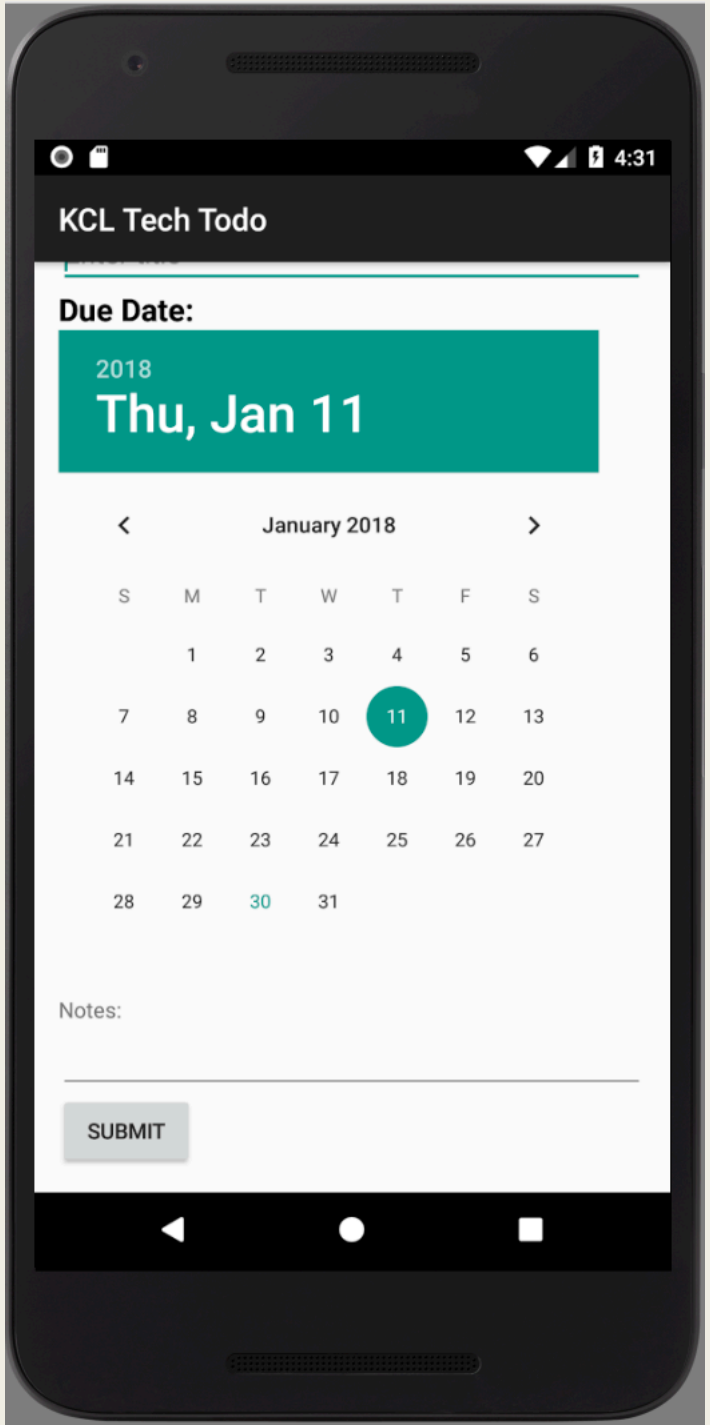
In the layout:

```
<TextView
   ...
   style="@style/hello_world_text" />
```

Now You Try!

Try to build this layout.
You have 40 minutes






MY WORK HERE
IS DONE
memegenerator.net