

Build X: Android

Lecture Two:
Layouts, Views and View Groups

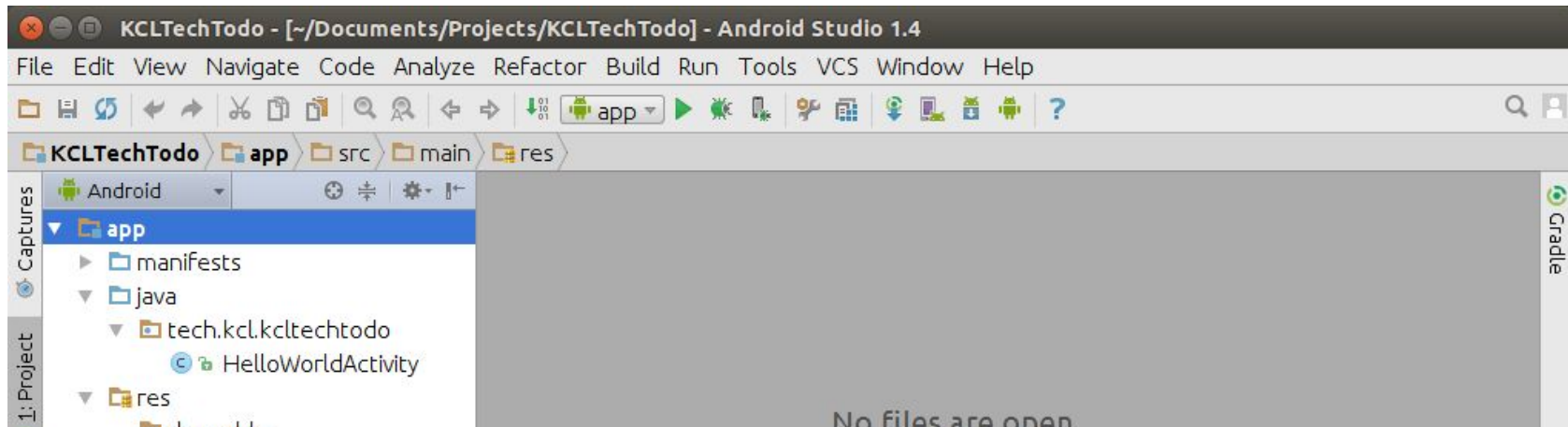
Notes: <http://android.kcl.tech/>



Recap

Recap: Environment Setup

- You should have **Android Studio** installed on your machine.
- You should have the **blank project** downloaded (*from <http://android.kcl.tech>*)
 - Don't worry if you can't open it yet, we're going to do that together



Recap: KCL Tech Todo

- We're going to build a basic todo list app, **KCL Tech Todo**.
- If you want a preview, it is **available on Google Play**.
 - Go to <http://tiny.cc/kcl-tech-todo>
 - Scan the QR code



Recap: App Components

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

Layouts

Layouts

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

“Put this here, now put that there, and put this next to that, etc.”

- **Q:** How do we tell Android **which layout** to use for **which activity**?
- **A:** One line of code, but not just yet...

// Side Note: XML

```
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder to buy pasta</heading>  
  <message>Don't forget, buy pasta!</message>  
</note>
```


// Side Note: XML

```
<person>  
  <name>Josh</name>  
  <age>22</age>  
  <gender>Male</gender>  
  <favouriteFood>Pasta</favouriteFood>  
</person>
```

// Side Note: XML

```
<resources>  
  <string name="app_name">KCL Tech Todo</string>  
  <string name="hello_world">Hello world!</string>  
</resources>
```

Layouts

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

“Put this here, now put that there, and put this next to that, etc.”


- **Q:** How do we tell Android **which layout** to use for **which activity**?
- **A:** One line of code, but not just yet...

O Layouts! Wherefore art thou?

- All layouts live in one folder:

`/app/res/layout`

Short for
“resources”



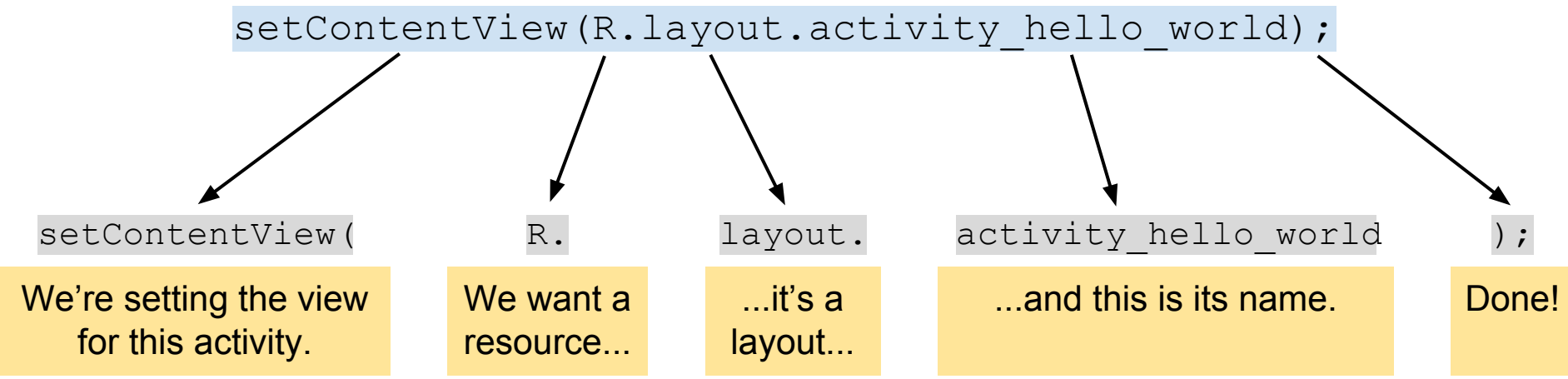
- We tell Android which layout to use with one line:

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

- **What's going on here?**

O Layouts! Wherefore art thou?

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



```
graph TD; A[setContentView(R.layout.activity_hello_world);] --> B[setContentView(]; A --> C[R.]; A --> D[layout.]; A --> E[activity_hello_world]; A --> F[)]; B --> B1[We're setting the view for this activity.]; C --> C1[We want a resource...]; D --> D1[...it's a layout...]; E --> E1[...and this is its name.]; F --> F1[Done!];
```

setContentView(

We're setting the view
for this activity.

R.

We want a
resource...

layout.

...it's a
layout...

activity_hello_world

...and this is its name.

);

Done!





Parts of a Layout: Views

Views


- A layout will usually **contain many views**.
- Views are added to a layout in their **XML files**.



```
<TextView />
```



```
<Button />
```



```
<EditText />
```



```
<ImageView />
```

```
<?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>
```

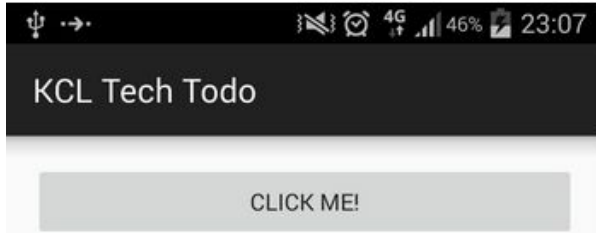
View Attributes

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

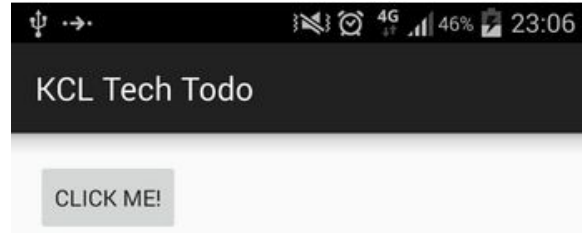
- Those two **attributes** are **required on every view**.
 - There are others, like `android:text`, `android:padding`, `android:visibility`, etc.
- What are `match_parent` and `wrap_content`?

View Attributes

`match_parent`



`wrap_content`



Now You Try!

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

```
<?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>
```

Parts of a Layout: View Groups

View Groups

- Some special views can contain other views - these are **view groups**.
- The views inside a view group are called its **children**.
- The view group containing views is the **parent** of those views.
- You can't see a view group; you can see the 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: More Stupid Names

A Layout is...

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

Eg. `activity_hello_world.xml`

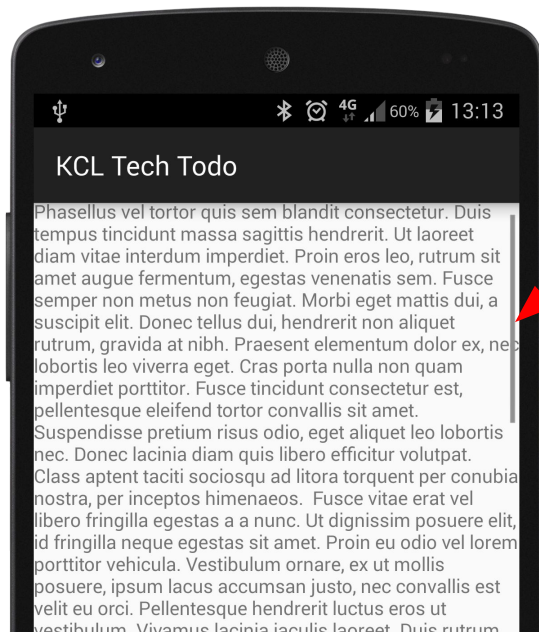
A View Group is...

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

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

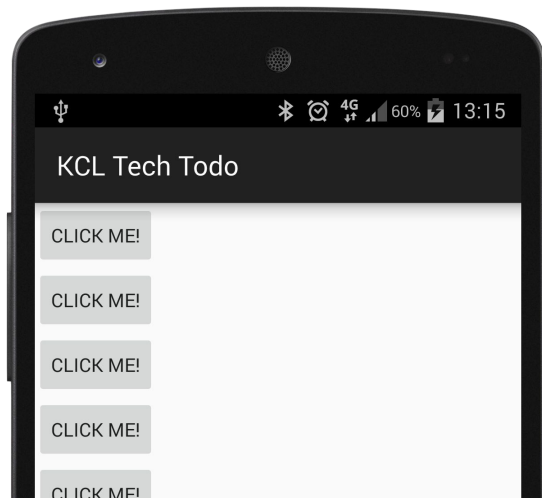
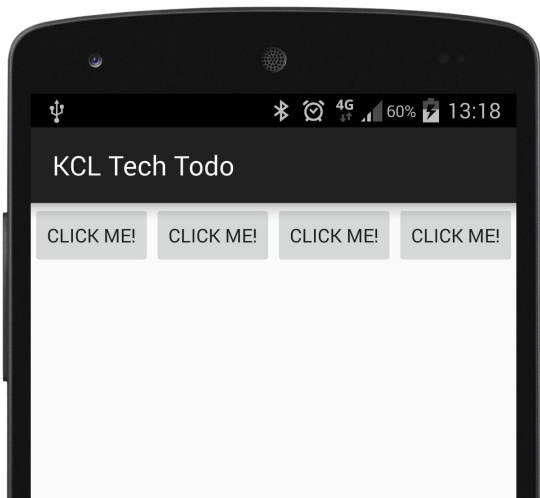
```
<ScrollView />
```

- Accepts **only one child**.
- Adds a **scrollbar**.
- Seriously, that's it.



```
<LinearLayout />
```

- Accepts **many children**.
- Stacks children horizontally or vertically.



```
<RelativeLayout />
```

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







Parts of a Layout: Other Resources

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`

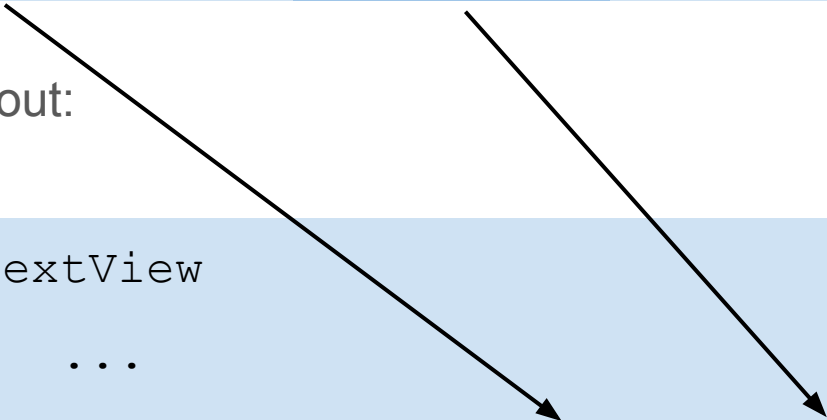
Using String Resources

- In the `strings.xml` file:

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

- In the layout:

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

Two arrows originate from the `hello_world` attribute in the `strings.xml` block above. One arrow points to the `android:text` attribute in the `TextView` layout block below, and the other points to the `hello_world` resource name within the same attribute.

Using Dimension Resources


- In the `dimens.xml` file:

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

Density-independent
pixels

- In the layout:

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



Using Style Resources

- In the `styles.xml` file:

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

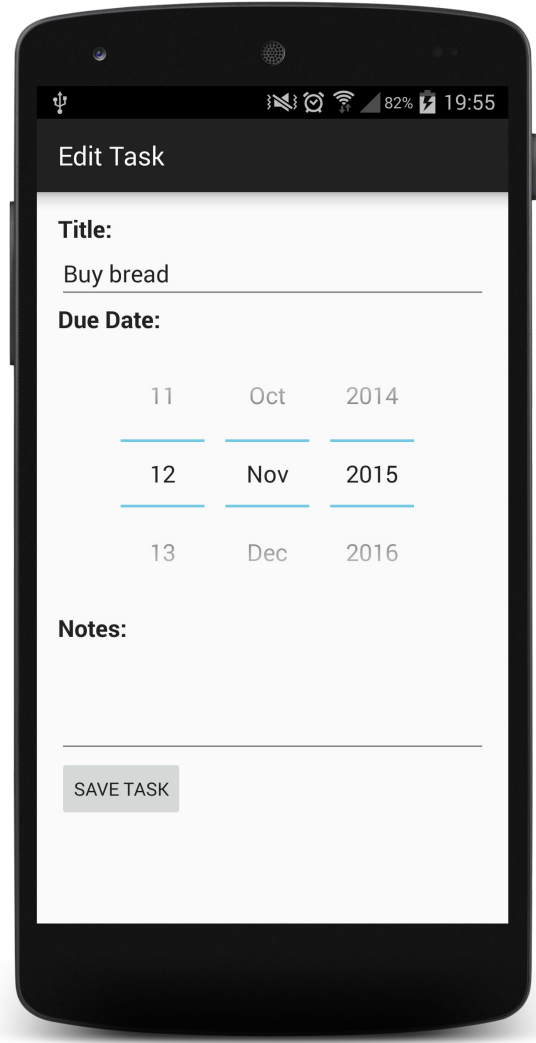
Using Style Resources

- In the layout:

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

Now You Try!

Try to build this layout.

A black smartphone displaying a task management app. The screen shows a status bar at the top with various icons and the time 19:55. Below the status bar is a dark header with the text "Edit Task". The main content area is white and contains a form with the following elements: a "Title:" label followed by a text input field containing "Buy bread"; a "Due Date:" label followed by a date picker showing "11 Oct 2014", "12 Nov 2015", and "13 Dec 2016" with horizontal lines between the rows; a "Notes:" label followed by a large text input area; and a "SAVE TASK" button at the bottom.

Edit Task

Title:
Buy bread

Due Date:

11	Oct	2014
12	Nov	2015
13	Dec	2016

Notes:

SAVE TASK

Okay, done!



Almost.

Next Steps

- Join the **Slack** channel!
 - `kcltechhq.slack.com` >> `#android-programming`
- Fill in our **survey**!
 - `android.kcl.tech` >> **Lecture Two**
- Try the `RelativeLayout` exercise.
 - `android.kcl.tech` >> **Lecture Two**
- Try to use an `ImageView`.