Build X: Android

Lecture Three:
Events, Intents and the Activity Lifecycle

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

Feedback

Feedback

- **Speed**: you said the speed is about right.
  - That's great, but if you ever need us to slow down (or speed up!), just let us know.
- **Breaks**: you said you want a break, so we'll give you one!
  - We'll take a 10 minute break in the middle of the lecture.
- **Slides**: you said you want us to upload the slides, so we will!
  - Slides and notes will go online before each lecture.
- **Other Feedback**: we really want to hear what you think, so don't be shy!

Recap

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: Layouts, Views and View Groups

- **Layout**: a specific type of application resource
  - These define the structure and appearance of parts of your app
- **View**: an individual component of a layout
  - A button, an image, a text input, etc.
- **View Group**: a special type of view that can contain others
  - You can’t see the view group, but you can use it to organise the views inside it
- **Other Resources**: used to supplement a layout, amongst other things
  - Strings, styles, dimension, etc.

View Events/Actions

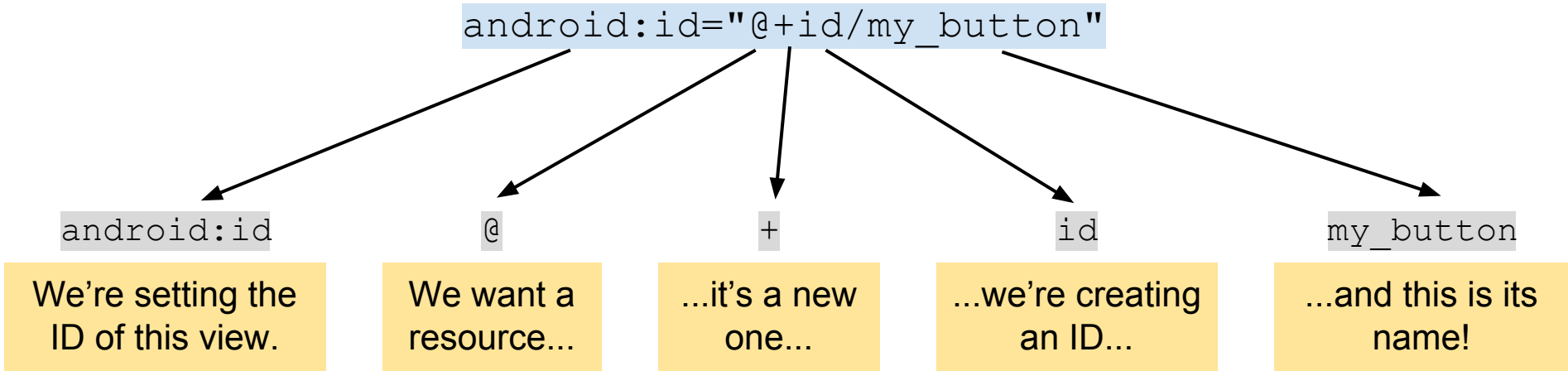
Events: Giving a View an ID

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/click_me_label"
    style="@style/click_me_button" />
```

Events: Giving a View an ID

```
<Button
    android:id="@+id/my_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/click_me_label"
    style="@style/click_me_button" />
```

What’s in a name?




HELLO!
My name is
my_button

Events: Giving a View an ID

```
<Button
    android:id="@+id/my_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/click_me_label"
    style="@style/click_me_button" />
```

Events: Getting a Reference to the View

I'm making a new variable called `myButton`, and it's going to hold a `Button` object.

This is the name of view to look for.

```
Button myButton =
    (Button) findViewById(R.id.my_button);
```

By the way, the view you find will be a button.

Search through the layout and find a view for me.

Events: Listen for an Event

```
Button myButton =
     (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(
     new OnClickListener() {
          public void onClick(View v) {
               // do something!
          }
     }
);
```

Events: Listen for an Event

```
Button myButton =
    (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(
    new OnClickListener() {
        public void onClick(View v) {
            // do something!
        }
    }
);
```

Events: Listen for an Event

```
Button myButton =
    (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(
    new OnClickListener() {
        public void onClick(View v) {
            // do something!
        }
    }
);
```

Events: Listen for an Event

```
Button myButton =
    (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(
    new OnClickListener() {
        public void onClick(View v) {
            // do something!
        }
    }
);
```

Events: Listen for an Event

```
Button myButton =
    (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(
    new OnClickListener() {
        public void onClick(View v) {
            // do something!
        }
    }
);
```

Events: Listen for an Event

```
Button myButton =
    (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(
    new OnClickListener() {
        public void onClick(View v) {
            // do something!
        }
    }
);
```

Events: Listen for an Event

```
Button myButton =
     (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(
     new OnClickListener() {
          public void onClick(View v) {
               // do something!
          }
     }
);
```

Coding time...

Events: Your Turn

- Give your button an ID, then set an `onClickListener` for it in the activity.

- Check the handout for some click actions you could try!
  - Didn't get a handout? Go to http://android.kcl.tech and scroll to **Lecture Three**

Intents

Intents

- An **intent** tells Android “I intend to do something”
  - I intend to go to another activity
  - I intend to share a photo
  - I intend to send a message

    ...
  - I intend to do some **custom action** defined by my application

Intents: Going to a Different Activity

I'm making a new variable called `intent`, and it's going to hold a `Intent` object.

```
Intent intent =
    new Intent(this, MyActivity.class);
```

Make a new intent...

...with a context of the current activity...

...and a target of an activity in the class called `MyActivity`.

Intents: Going to a Different Activity

```
Intent intent =
    new Intent(this, MyActivity.class);
startActivity(intent);
```

Intents: Going to a Different Activity

```
Intent intent =
    new Intent(this, MyActivity.class);
startActivity(intent);
```

Coding time...

Intents: Your Turn

- Create a new activity, and call it whatever you want.
  - We'll show you how!

- Use the `onClick` method from the last section to start an intent that takes the user to your new activity.

Intents: Sending Extras

Intents: Sending Extras

```
Intent intent =
    new Intent(this, MyActivity.class);
EditText msgInput =
    (EditText) findViewById(R.id.msg_input);
String msg = msgInput.getText().toString();
intent.putExtra(EXTRA_MESSAGE, msg);
startActivity(intent);
```

Intents: Sending Extras

```
Intent intent =
    new Intent(this, MyActivity.class);
EditText msgInput =
    (EditText) findViewById(R.id.msg_input);
String msg = msgInput.getText().toString();
intent.putExtra(EXTRA_MESSAGE, msg);
startActivity(intent);
```

Intents: Sending Extras

```
Intent intent =
    new Intent(this, MyActivity.class);
EditText msgInput =
    (EditText) findViewById(R.id.msg_input);
String msg = msgInput.getText().toString();
intent.putExtra(EXTRA_MESSAGE, msg);
startActivity(intent);
```

Intents: Sending Extras

```
Intent intent =
    new Intent(this, MyActivity.class);
EditText msgInput =
    (EditText) findViewById(R.id.msg_input);
String msg = msgInput.getText().toString();
intent.putExtra(EXTRA_MESSAGE, msg);
startActivity(intent);
```

Intents: Sending Extras

```
Intent intent =
    new Intent(this, MyActivity.class);
EditText msgInput =
    (EditText) findViewById(R.id.msg_input);
String msg = msgInput.getText().toString();
intent.putExtra(EXTRA_MESSAGE, msg);
startActivity(intent);
```

Intents: Sending Extras

```
Intent intent =
    new Intent(this, MyActivity.class);
EditText msgInput =
    (EditText) findViewById(R.id.msg_input);
String msg = msgInput.getText().toString();
intent.putExtra(EXTRA_MESSAGE, msg);
startActivity(intent);
```

Intents: Sending Extras

```
Intent intent =
    new Intent(this, MyActivity.class);
EditText msgInput =
    (EditText) findViewById(R.id.msg_input);
String msg = msgInput.getText().toString();
intent.putExtra(EXTRA_MESSAGE, msg);
startActivity(intent);
```

Intents: Reading Extras

Intents: Reading Extras

- **Every activity** is launched with an intent, which you can access.
- To get the intent, call `Intent intent = getIntent()`.
- To get the extras, call `Bundle extras = intent.getExtras();`.
- A good place to call these methods is in the `onCreate()` method.
  - We'll be looking at this next!

Intents: Reading Extras

```
public void onCreate (...) {
    Intent intent = getIntent();
    Bundle extras = intent.getExtras();
    String msg = extras.getStringExtra(EXTRA_MESSAGE);

    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(msg);
    setContentView(textView);
}
```

Intents: Reading Extras

```
public void onCreate (...) {
    Intent intent = getIntent();
    Bundle extras = intent.getExtras();
    String msg = extras.getStringExtra(EXTRA_MESSAGE);

    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(msg);
    setContentView(textView);
}
```

Intents: Reading Extras

```
public void onCreate (...) {
    Intent intent = getIntent();
    Bundle extras = intent.getExtras();
    String msg = extras.getStringExtra(EXTRA_MESSAGE);

    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(msg);
    setContentView(textView);
}
```

Intents: Reading Extras

```
public void onCreate (...) {
    Intent intent = getIntent();
    Bundle extras = intent.getExtras();
    String msg = extras.getStringExtra(EXTRA_MESSAGE);

    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(msg);
    setContentView(textView);
}
```

Intents: Reading Extras

```
public void onCreate (...) {
    Intent intent = getIntent();
    Bundle extras = intent.getExtras();
    String msg = extras.getStringExtra(EXTRA_MESSAGE);

    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(msg);
    setContentView(textView);
}
```

Intents: Reading Extras

```
public void onCreate (...) {
    Intent intent = getIntent();
    Bundle extras = intent.getExtras();
    String msg = extras.getStringExtra(EXTRA_MESSAGE);

    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(msg);
    setContentView(textView);
}
```

Intents: Reading Extras

```
public void onCreate (...) {
    Intent intent = getIntent();
    Bundle extras = intent.getExtras();
    String msg = extras.getStringExtra(EXTRA_MESSAGE);

    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(msg);
    setContentView(textView);
}
```

Intents: Reading Extras

```
public void onCreate (...) {
    Intent intent = getIntent();
    Bundle extras = intent.getExtras();
    String msg = extras.getStringExtra(EXTRA_MESSAGE);

    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(msg);
    setContentView(textView);
}
```

Intents: Reading Extras

```
public void onCreate (...) {
    Intent intent = getIntent();
    Bundle extras = intent.getExtras();
    String msg = extras.getStringExtra(EXTRA_MESSAGE);

    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(msg);
    setContentView(textView);
}
```

Intents: Reading Extras

```
public void onCreate (...) {
    Intent intent = getIntent();
    Bundle extras = intent.getExtras();
    String msg = extras.getStringExtra(EXTRA_MESSAGE);

    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(msg);
    setContentView(textView);
}
```


30STC


NO.
SENORGIF.COM

Coding time...

Intents: Your Turn (Again)

- **Add a string extra** to the intent you created earlier.

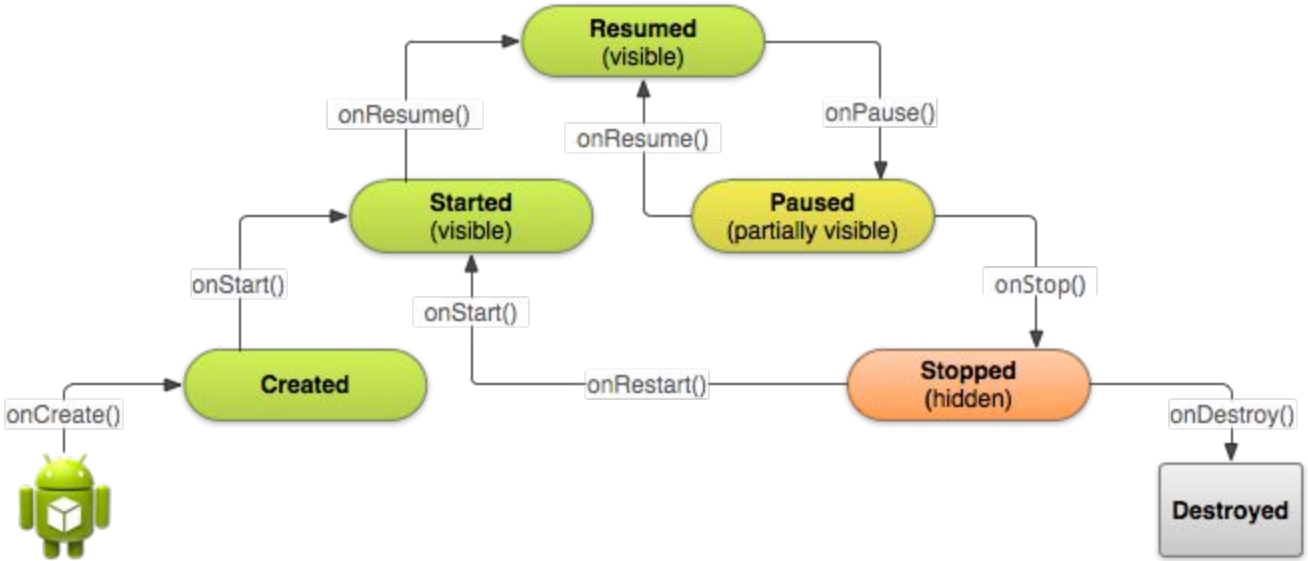
- **Read that string extra** in the new activity, and display it on the screen.

Break! (10 mins)

Activity Lifecycle

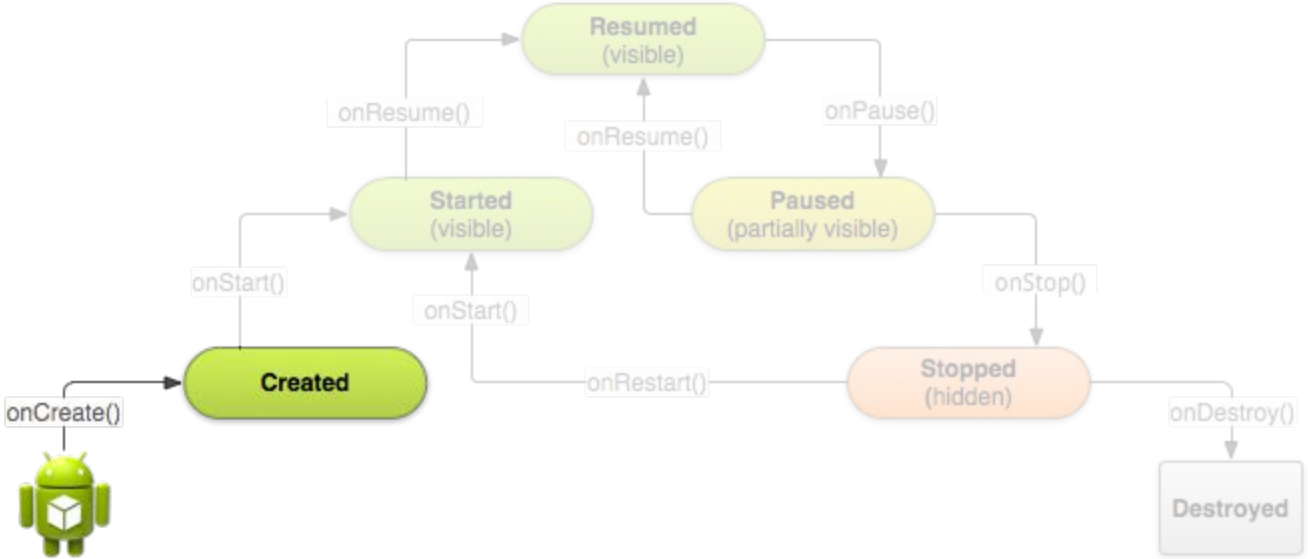
Activity Lifecycle: States

- During its lifetime, an activity will move between several **states**.



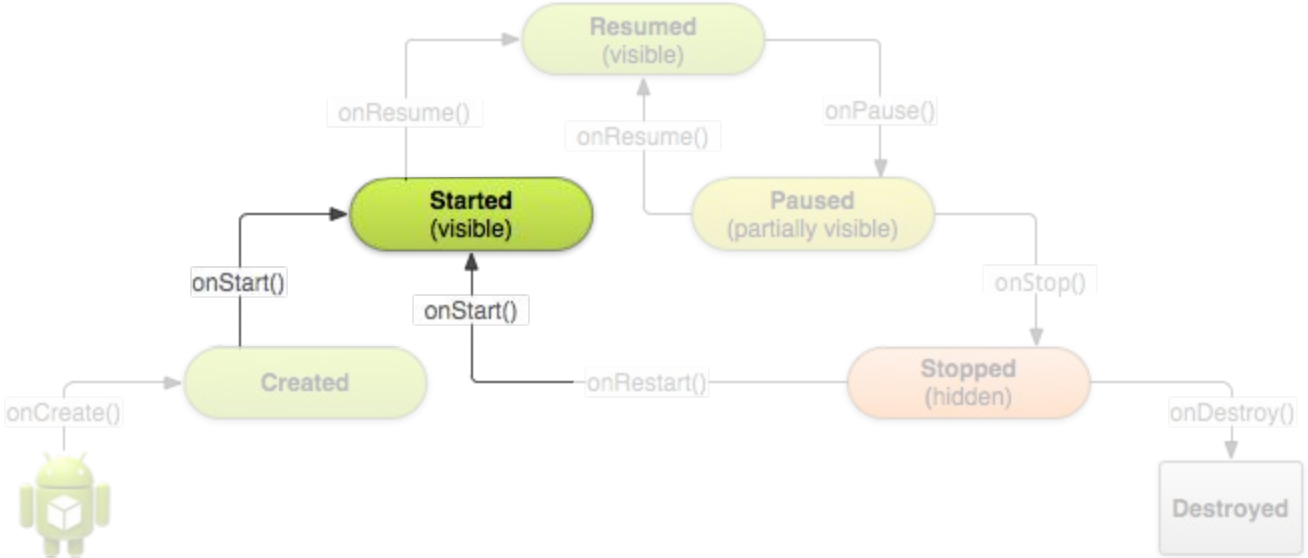
Activity Lifecycle: `onCreate()`

- Called when the activity is first created. This is where you can create views, setup data sources, etc. Always followed by `onStart()`.



Activity Lifecycle: `onStart()`

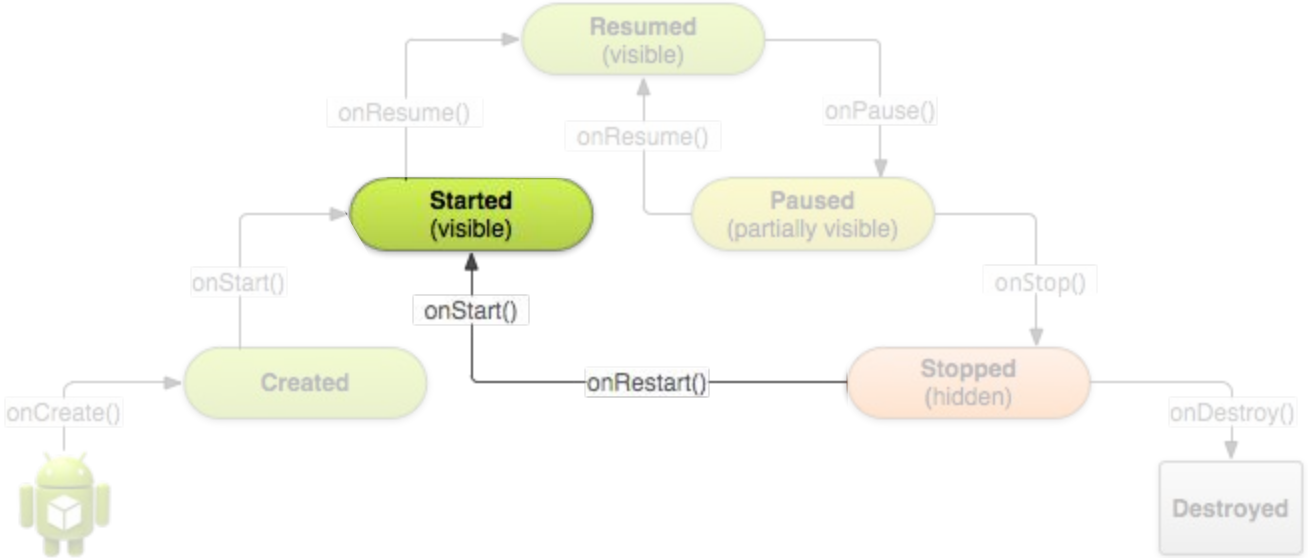
- The activity is becoming visible to the user. Always followed by `onResume()`.



Activity Lifecycle: `onRestart()`

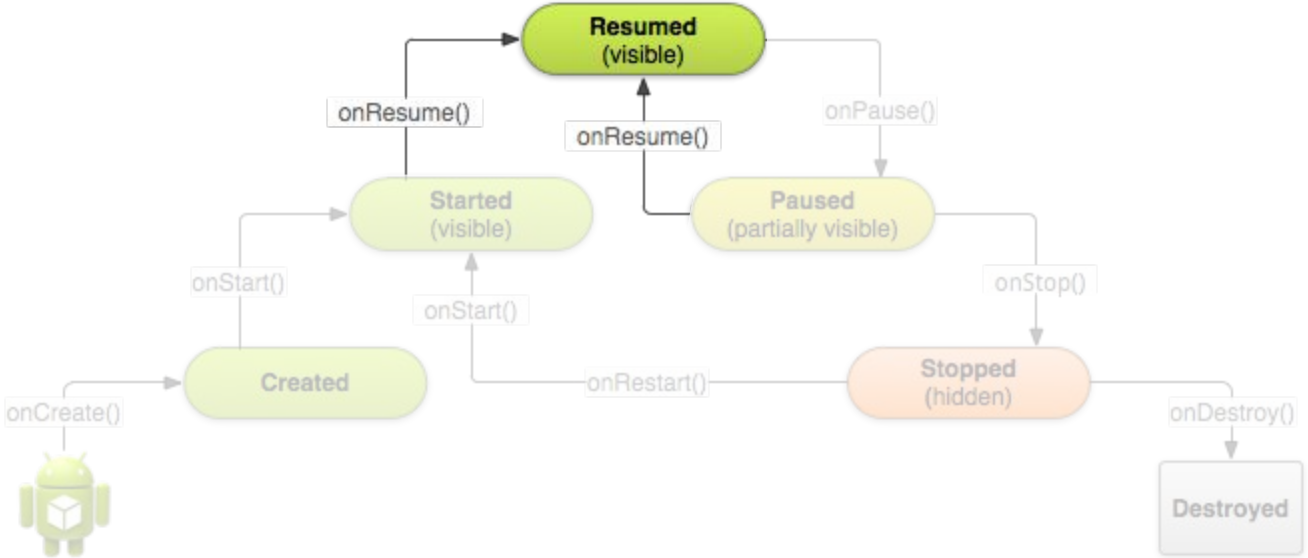
- Called after the activity has been stopped, prior to it being started again.

  Always followed by `onStart()`.



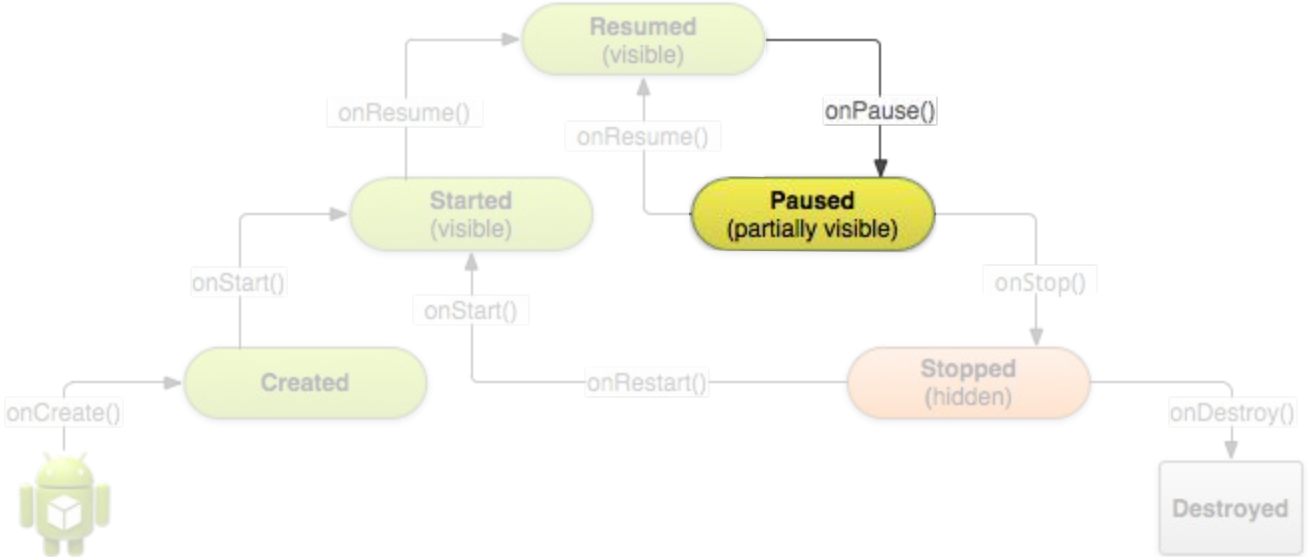
Activity Lifecycle: `onResume()`

- Called when the activity will **start interacting** with the user. At this point, your activity is at the stop of the stack and receiving user input.



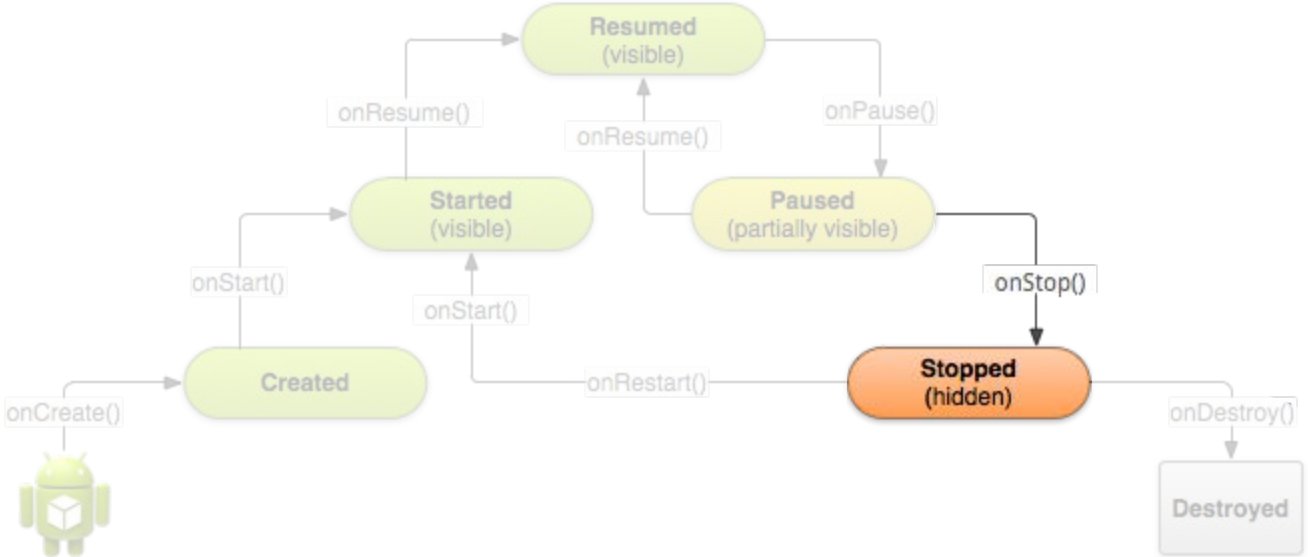
Activity Lifecycle: `onPause()`

- Called when the activity is going into the background, but is not being killed yet. Example: when a dialog pops up, or another activity starts.
- This is a counterpart to `onResume()`.



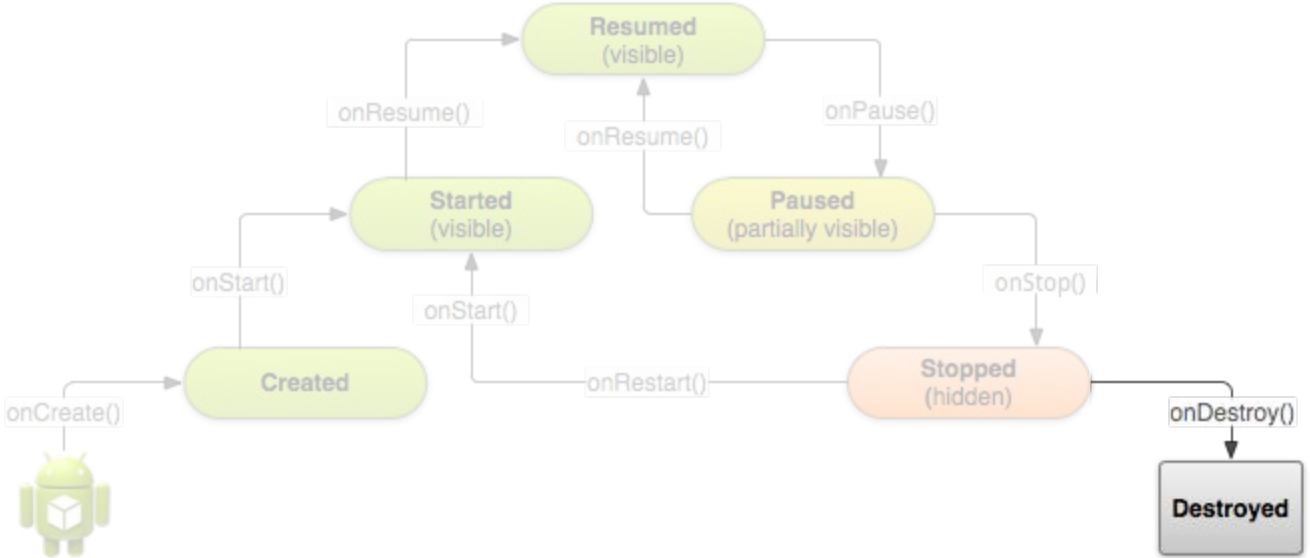
Activity Lifecycle: `onStop()`

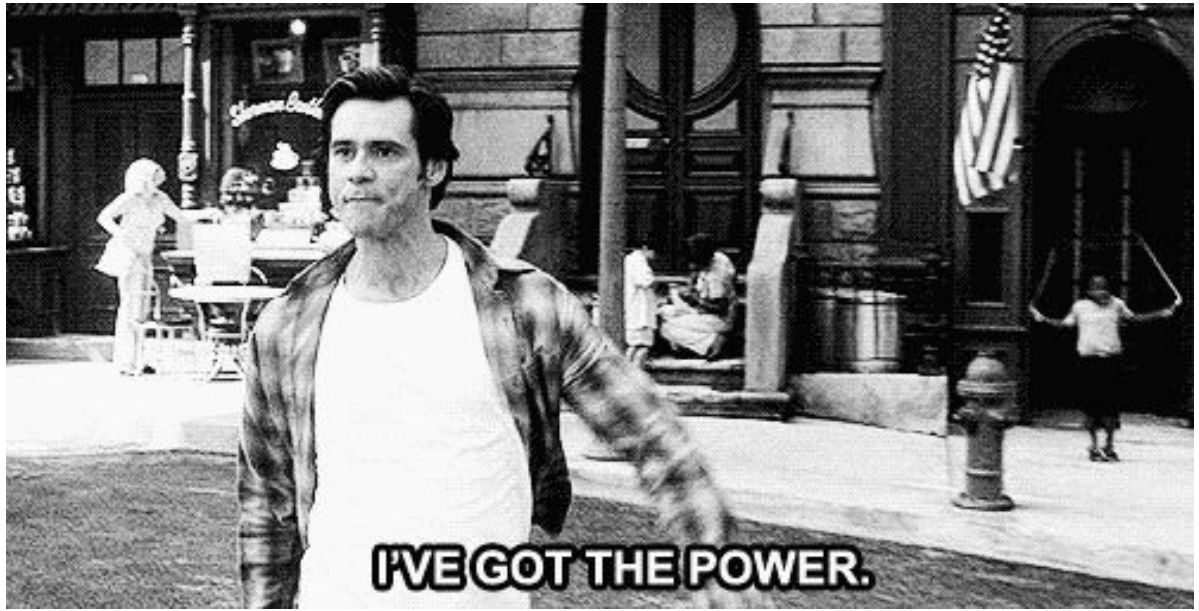
- Called when the activity is completely hidden from the user. Followed by `onRestart()` or `onDestroy()`, or sometimes nothing.



Activity Lifecycle: `onDestroy()`

- Called when your activity is completely destroyed, either by the user fully exiting it, or the system killing it to conserve resources.




I'VE GOT THE POWER.

Coding time...

Intents: Activity Lifecycle

- **Create an activity** (or use the same one), and set the view to be a single `TextView` as shown in the intent slides.

- Add a line to this `TextView` each time one of the lifecycle methods is called
    - Ask questions if you need help, or use the Slack channel.

- Study how the methods are called, and in which order.

Done!

Almost.

Unitu{hack} | LONDON

NOV 14TH 2015

Now we’re done!

What Now?

- **Next week is reading week**
  - Don’t turn up; we won’t be here!
- **Finish off the lifecycle activity**
  - Remember: the Slack channel is there to help you.
- Mark and Maria will be posting some exercises in the **Slack channel**
  - Give it a go and get involved!
- **After the break**: `ListViews`, databases and putting it all together