BUILD X : ANDROID

Lecture Two: Events, Intents and the Activity Lifecycle

Handout:
https://tinyurl.com/android-lecture2-handout

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 – 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, input field, text field, image, etc.
- View Group: a special type of View **that can contain other views**
  - You can't see the view group, but you can see the views inside of it
  - LinearLayout, ScrollView, RelativeLayout, etc.
- Other Resources: used to supplement a layout, amongst other things
  - Strings, styles, dimensions, etc.

View Events/Actions

Handout:
https://tinyurl.com/android-lecture2-handout

Events

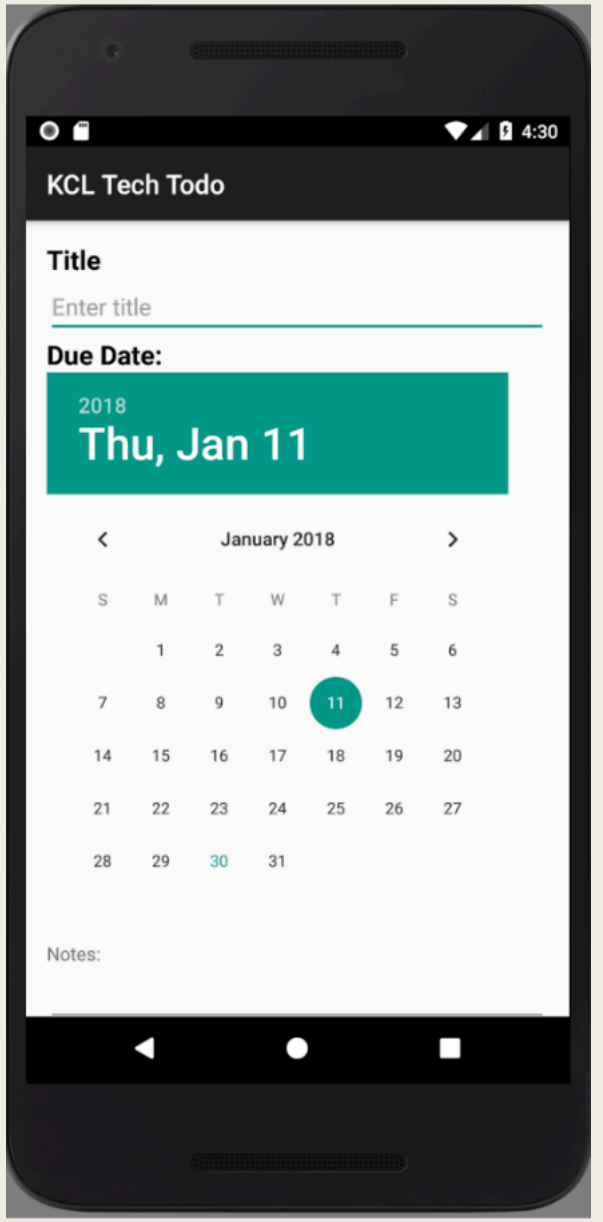
We have Views…
but they don't really do anything.

Most Views exist for users to interact with.

We need to listen for actions on a View.

But first we need a way to refer to them in our Java code…

How?



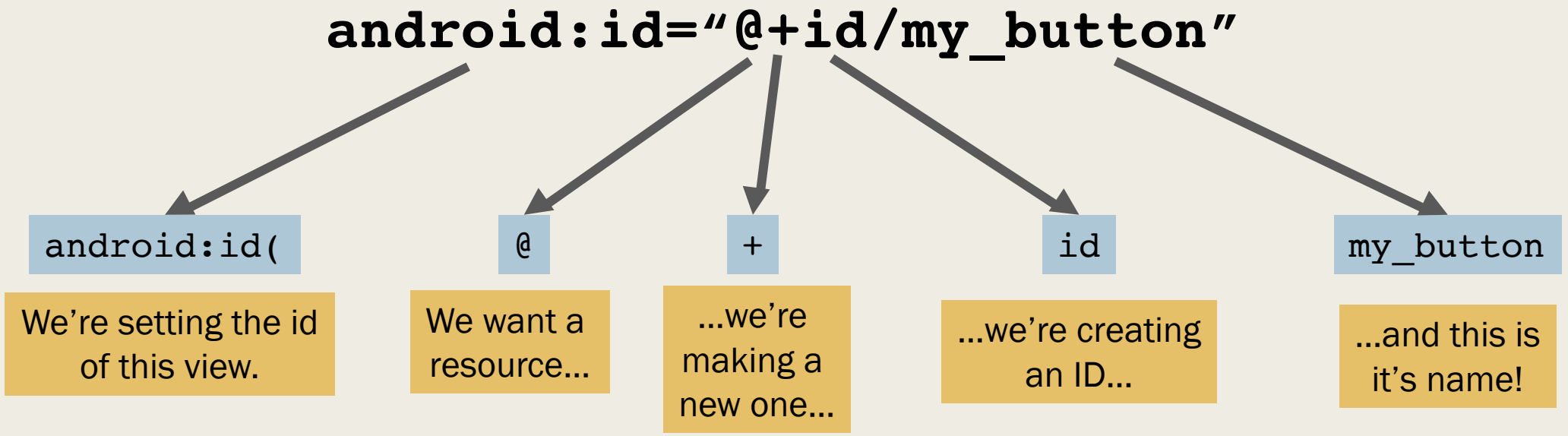
Events – Giving a View an ID

```
<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 a View

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

...this is the name of the 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 – Getting a Reference to a View

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

Now how do we do the stuff?

Events - Listen for an Event

An event is essentially an action on a View.

For a button, the onClick event occurs when the button is clicked.

We need to **listen** for that event, and define a response when it happens.

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

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

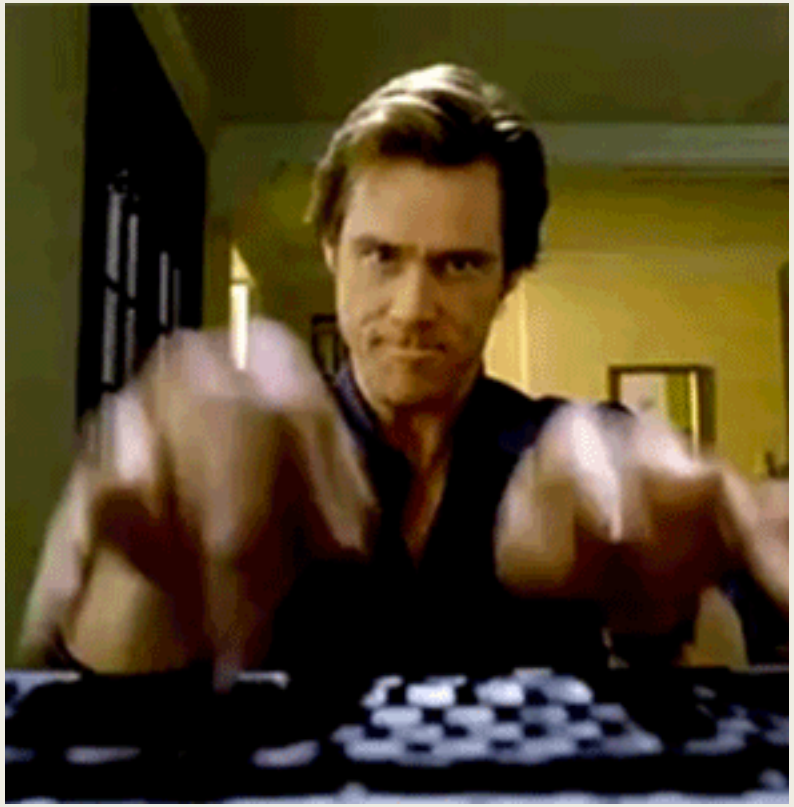
```
Button myButton =
    (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(
    new OnClickListener() {
        public void onClick(View v) {
            // do something!
        }
    }
);
```

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

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

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

```
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 to make it do **something**.

- Check the handout for some click actions you can try!
  - *Don't have one? Get one from here:*

https://tinyurl.com/android-lecture2-handout

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 share 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(CurrentActivity.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(CurrentActivity.this,
                    MyActivity.class);

startActivity(intent);
```

Coding time...

Intents – Your Turn

- Create a new activity, and call it what ever you like.
- 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

What if you wanted to send some information with the new Intent? Send extras.

An **extra** is a simple key/value pair, where they **key** is always a String and the **value** is any primitive type (int, string, boolean, etc.).

Adding an **extra** to an **Intent** is pretty straight forward.

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

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

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

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

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

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

```
Intent intent =
    new Intent(CurrentActivity.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 what that is next!

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Coding time...

Intents – Your Turn (Again)

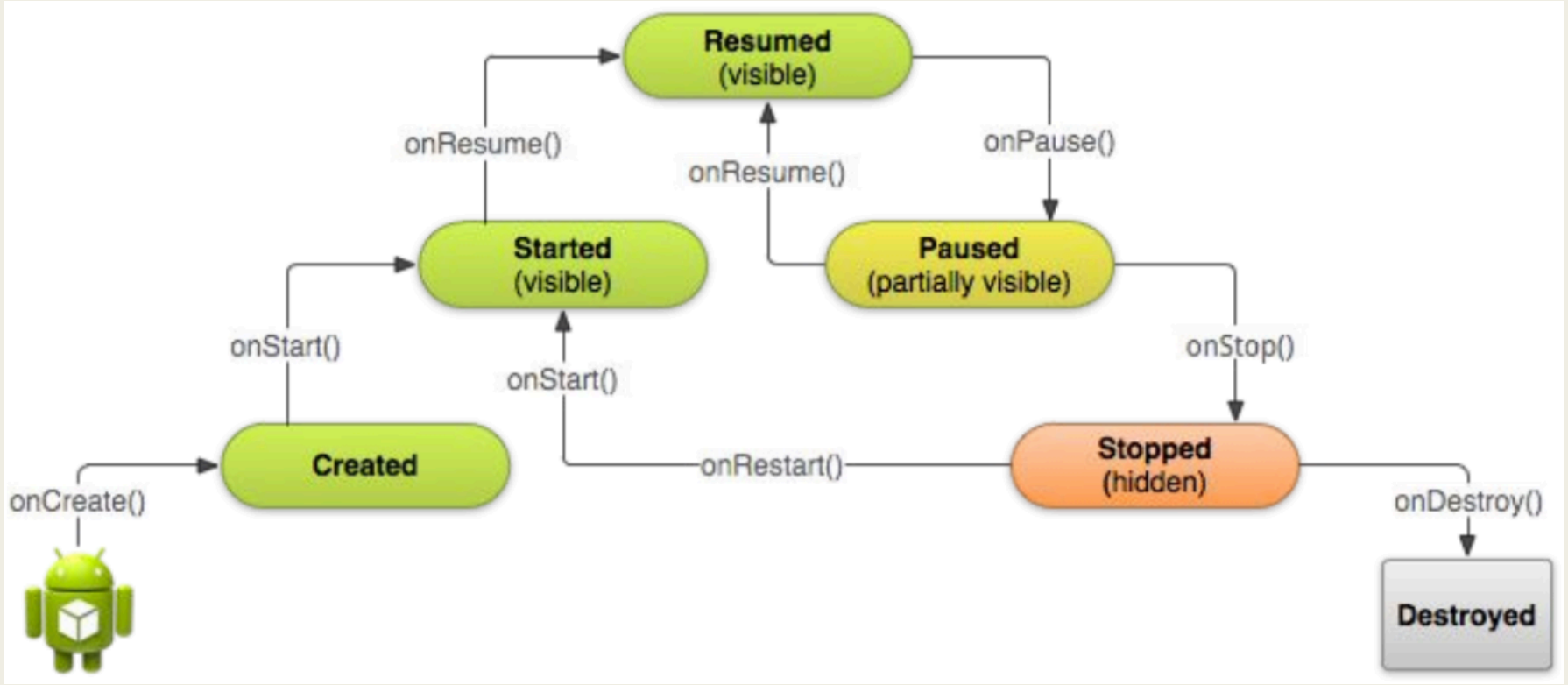
- **Add a string extra** to the intent you created earlier
- **Read that string extra** in the new activity, and display it on the screen.


Feels good man

Activity Lifecycle

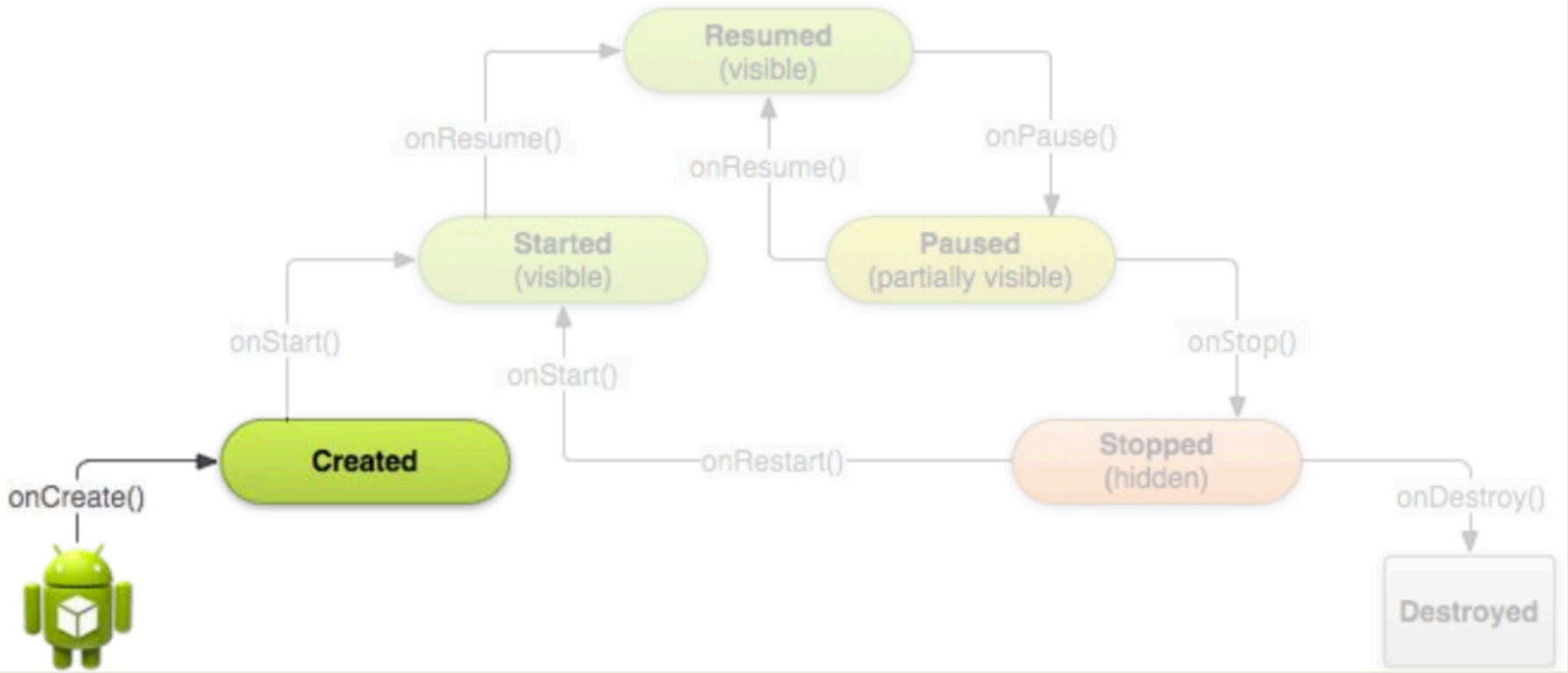
Activity Lifecycle - States

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



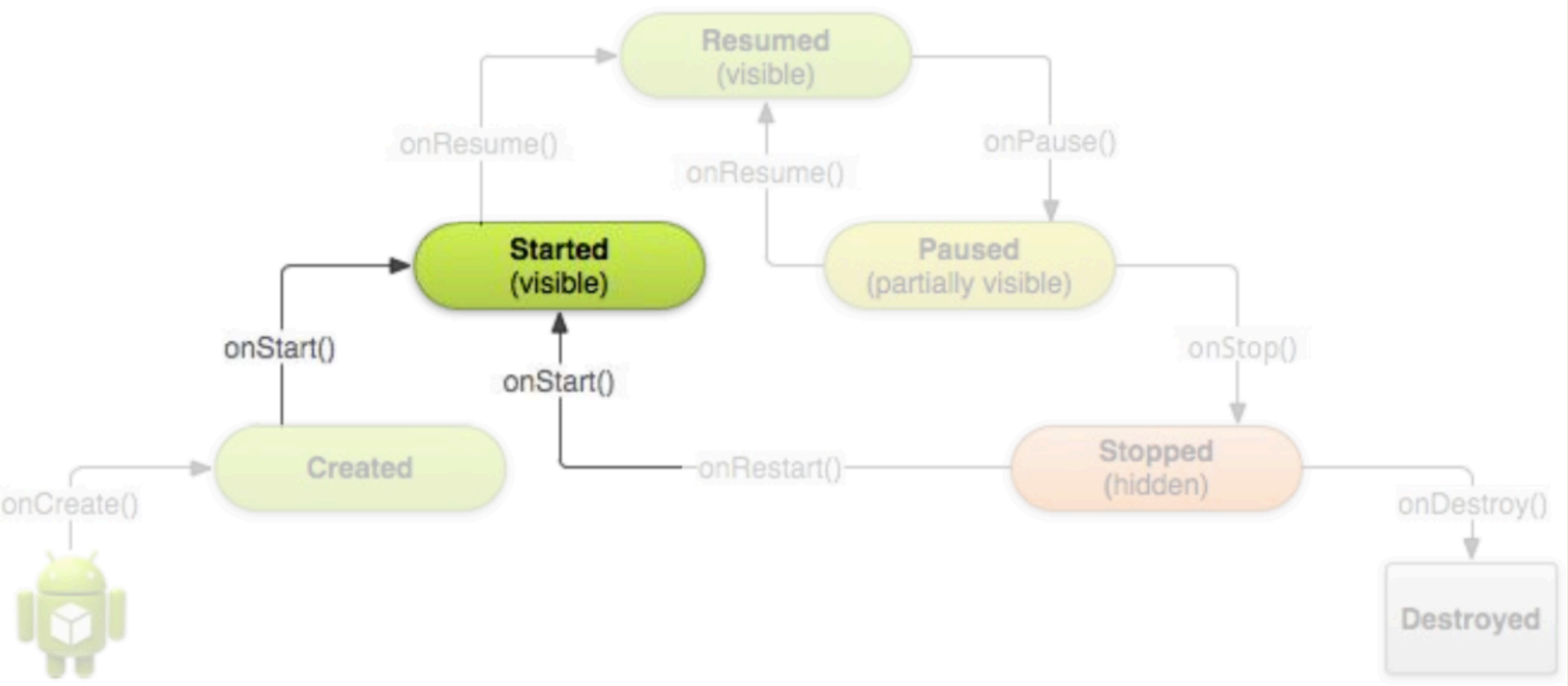
Activity Lifecycle – **`onCreate()`**

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



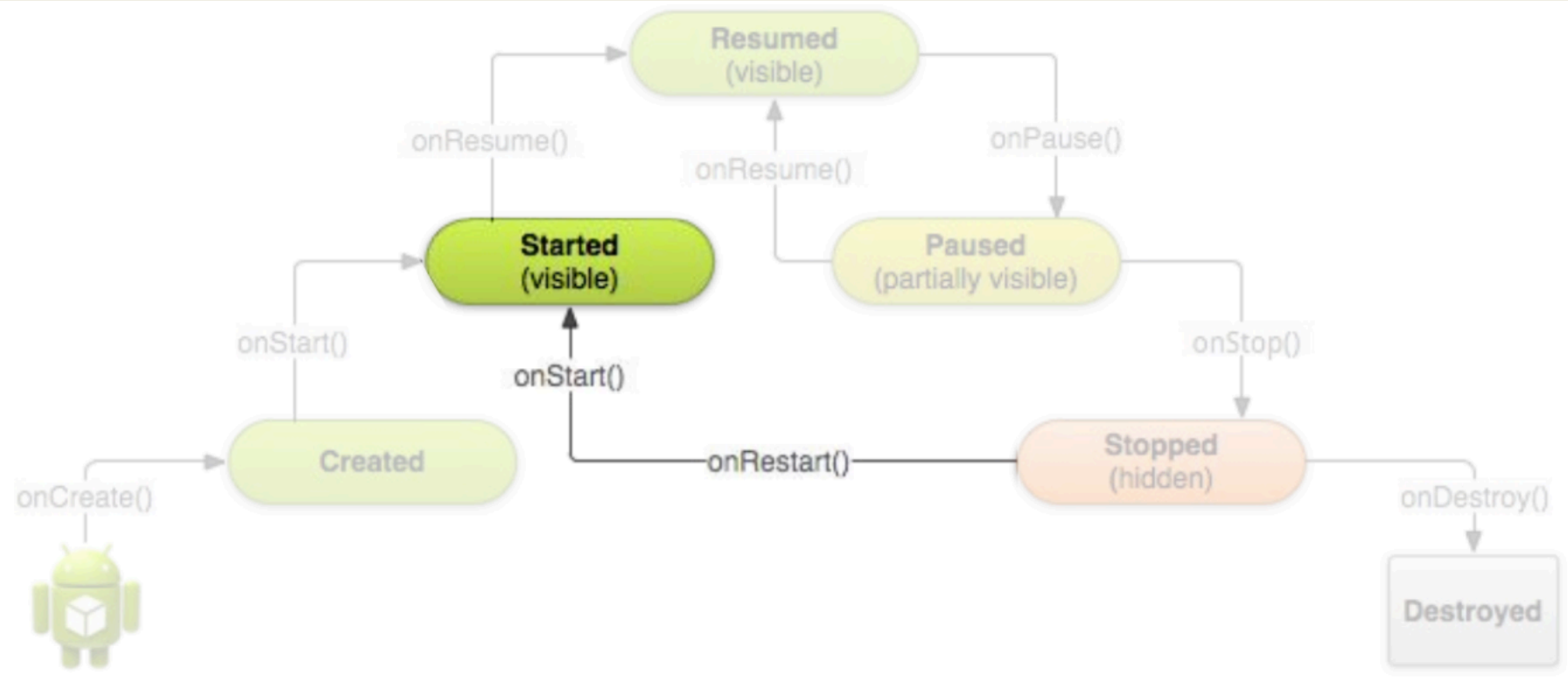
Activity Lifecycle – **`onStart()`**

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



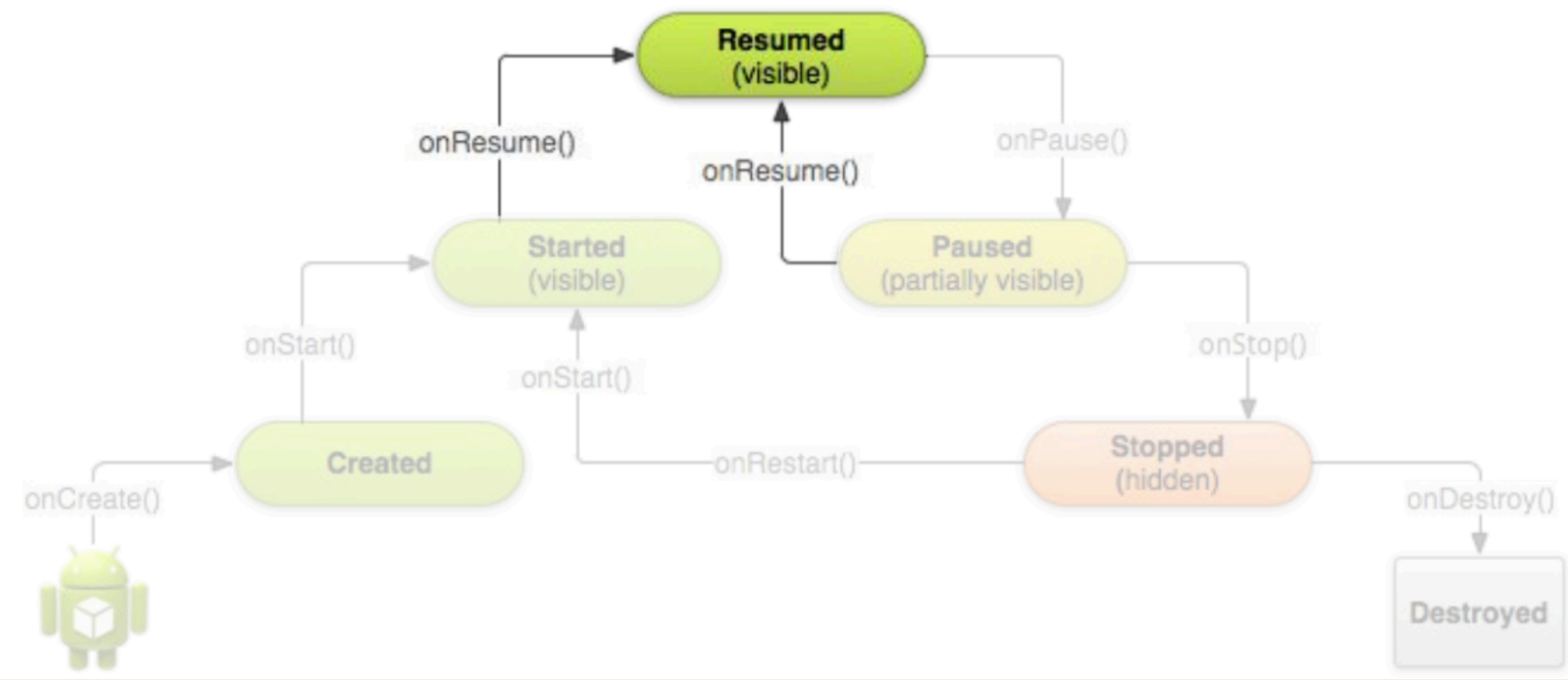
Activity Lifecycle – `onRestart()`

Called after the activity has been stopped, **prior to it being started again**. Always followed by **`onStart()`**.



Activity Lifecycle – **`onResume()`**

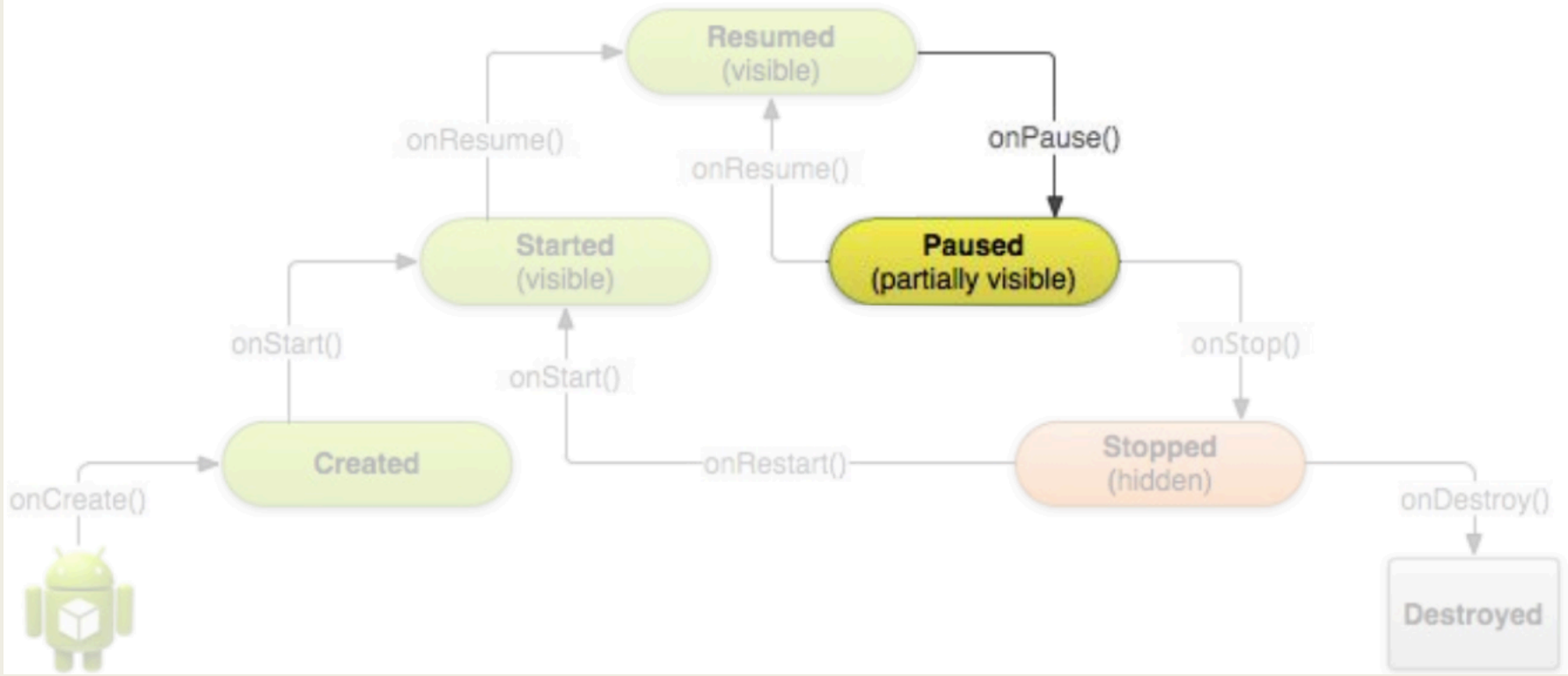
Called when the activity will **start interacting** with the user. At this point, your activity is at the top of the stack and receiving user input.



Activity Lifecycle – **`onPause()`**

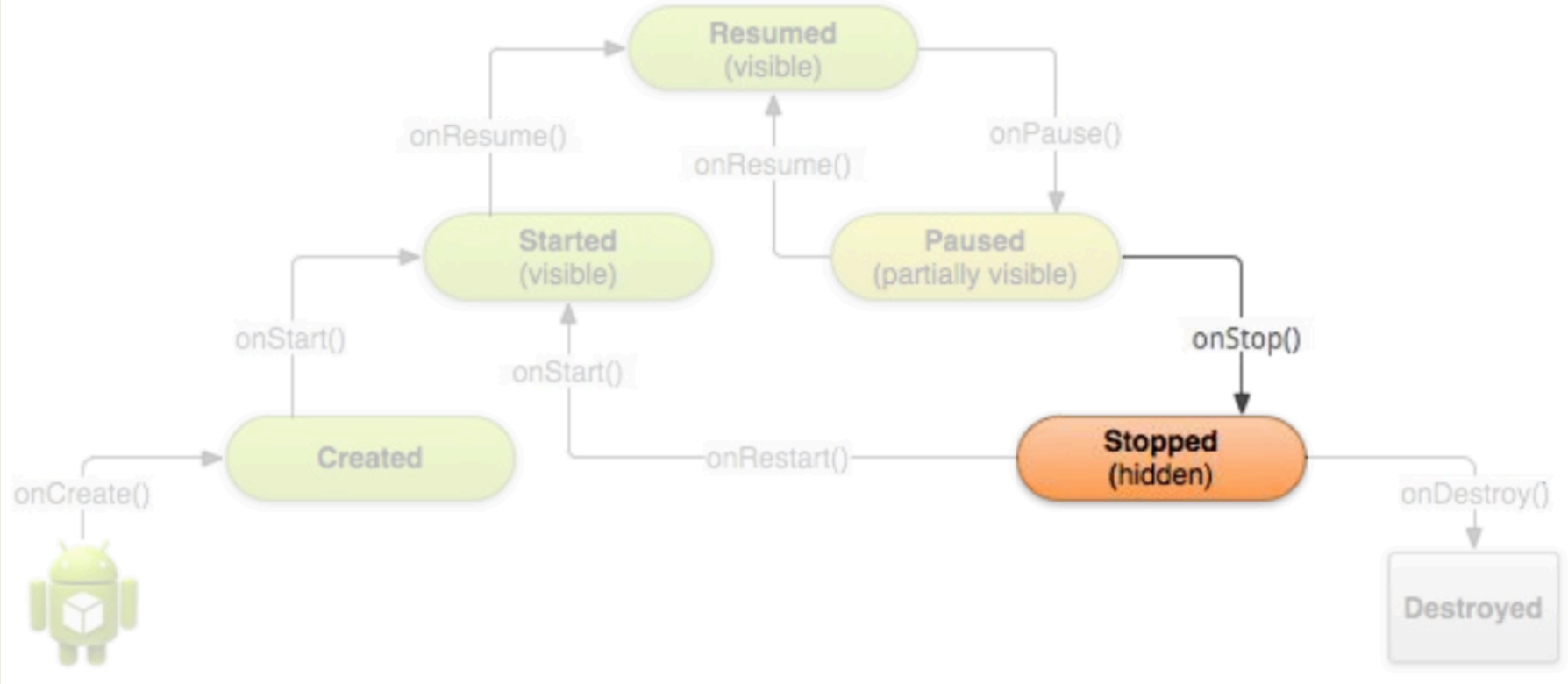
Called when an activity is going into the background, but is not being killed yet (e.g. When a dialog pops up, another activity starts, etc.)

This is a counterpart to **`onResume()`**.



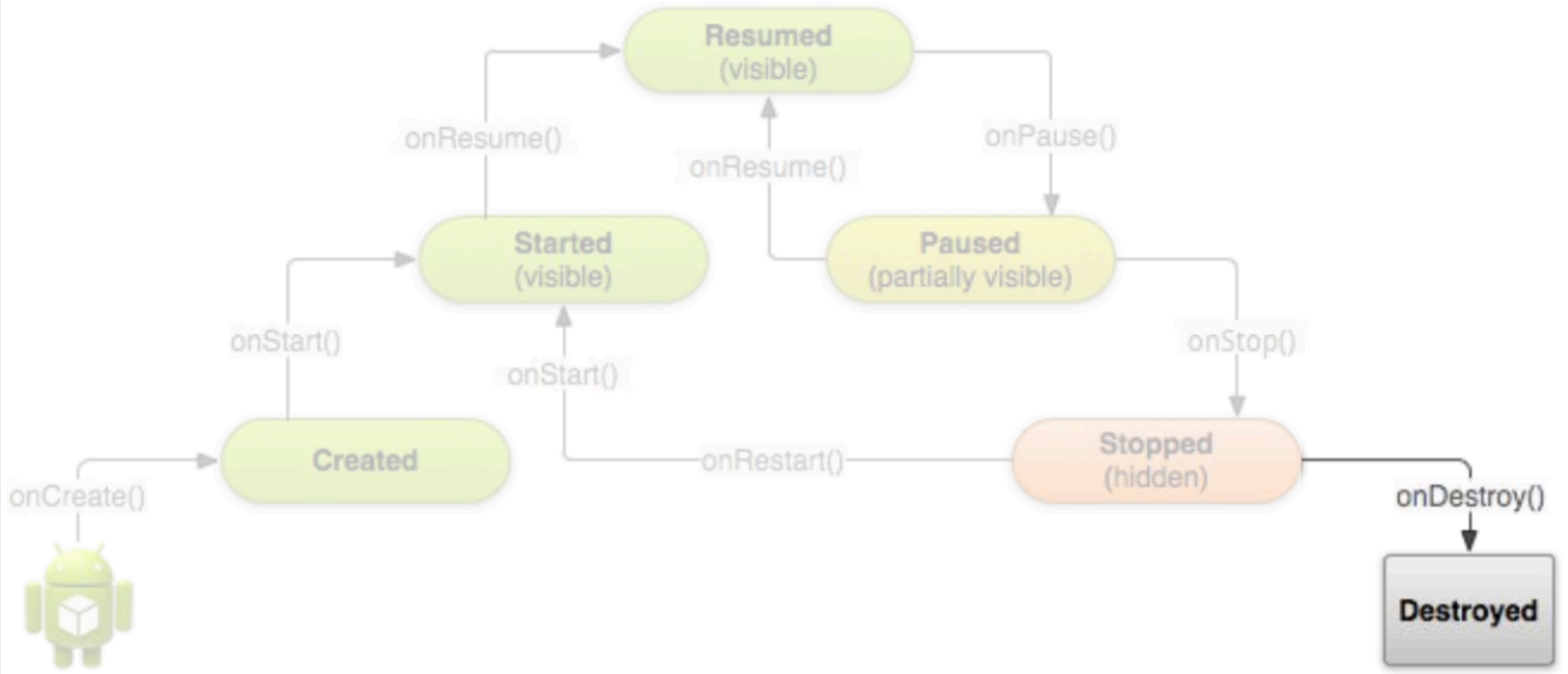
Activity Lifecycle – **`onStop()`**

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.



Activity Lifecycle – Your Turn!

- **Create an activity** (or use the same one) and set the view to be a single **TextView** as shown in the **Intent** section.
- Add a line to this **TextView** each time one of the lifecycle methods is called
  - Ask if you need help!
- Study how the methods are called, and in which order.


Guess what...?
We're DONE!
quickmeme.com