### Implicit Intents and Services

### Implicit Intents

- Declare an Action to occur rather than a class to start
- Actions can be handled by any app that registers for that type of intent
- Intent Filters register Apps with the system

# Default Actions Available with Implicit Intents

- Open a map
- Make a phone call
- View a web page
- Send an Email
- Create a Calendar Event
- Sent a Text Message

# Code

```
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType(HTTP.PLAIN_TEXT_TYPE); // "text/plain" MIME type
// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
```

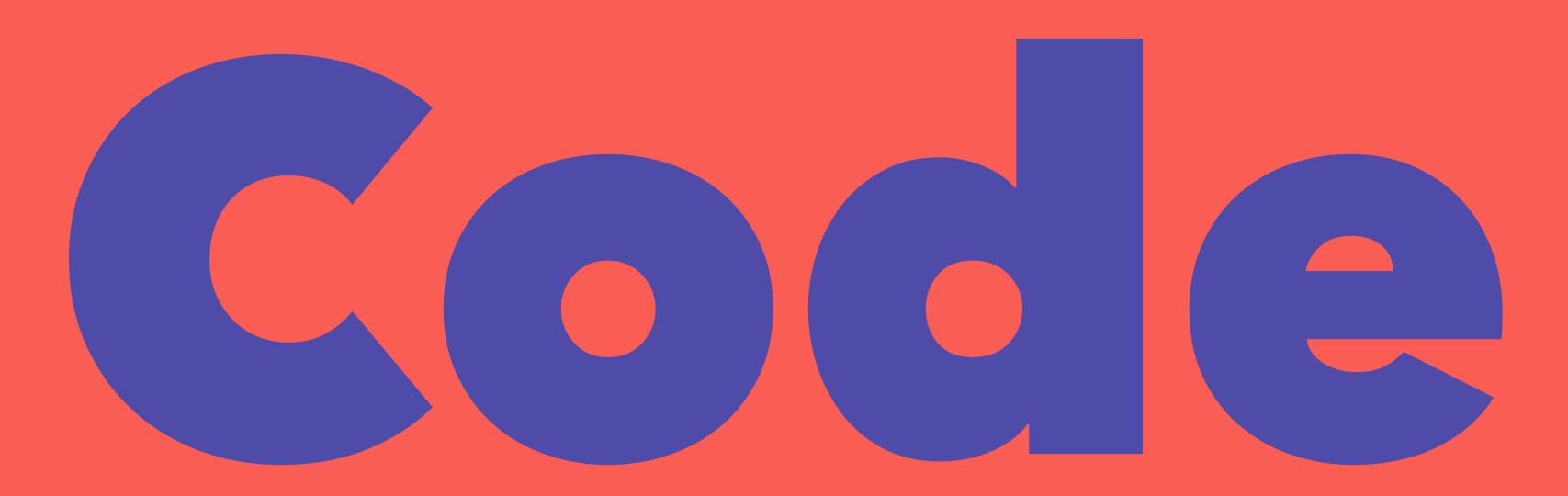
## Doing Stuff in the Background

# Services

- Services allow long running operations to be performed in the background
- A Services continues running indefinitely even if the component that started it is destroyed
- Intents are paired with services to allow services to interface with components like Activities

### Configuring a Service

- Service must be declared in the manifest
- Most Often you'll subclass Intent Service and Override it's onHandleIntent Method
- Repeating Alarms and the AlarmManager can be used to schedule Service Execution



#### IntentService Subclass

```
Constructor
```

```
public PollService() {
    super(TAG);
}
```

#### SetAlarm

```
public static void setServiceAlarm(Context context, boolean isOn) {
    Intent i = new Intent(context, PollService.class);
    PendingIntent pi = PendingIntent.getService(context, 0, i, 0);

    AlarmManager alarmManager = (AlarmManager)context.getSystemService(Context.ALARM_SERVICE);

    if (isOn){
        alarmManager.setRepeating(AlarmManager.RTC, System.currentTimeMillis(), POLL_INTERVAL, pi);
    } else {
        alarmManager.cancel(pi);
        pi.cancel();
    }
}
```

#### isAlarmon?