I recently discovered this Html parsing library called Jsoup. It is open source and extremely powerful.

To get a hang of it I have been playing with it.

One of the many things that I did with it, is to make an app that gives you the birthdays of celebrities/famous people.

I don't have the time to be explaining everything I have done in the code.

So I will be just pasting the code here itself.

It is pretty much self-explanatory, and I will try to add comments too.

The app is pretty much a skeleton app, so there is no designing done and looks ugly. So don't judge my skills based on that.

Well, without further ado, here it is:

CODE:

MainActivity:

```
package com.bragitoff.celebbday;

import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

import java.io.IOException;
import java.util.regex.Pattern;

public class MainActivity extends AppCompatActivity {

    EditText celebName;
    TextView bday;
    Button submit;
    String celebNameString;
    String searchUrl;
```

```
    String resourceUrl;
    String bdate;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        celebName=(EditText)findViewById(R.id.celebName);
        bday=(TextView)findViewById(R.id.bday);
        submit=(Button)findViewById(R.id.button);

        submit.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                // Perform action on click
                celebNameString=celebName.getText().toString();
                celebNameString=celebNameString.replaceAll("\\s","\\+");
                //searchUrl=bdayDuckDuckGoSearch(celebNameString);
                searchUrl=bdayWikipediaSearch(celebNameString);
                //DuckDuckGoParser jsoupAsyncTask = new DuckDuckGoParser();
                wikiSearchParser jsoupAsyncTask = new wikiSearchParser();
                jsoupAsyncTask.execute();
                wikiArticleParser jsoupAsyncTask2 = new wikiArticleParser();
                jsoupAsyncTask2.execute();
                bday.setText(bdayWikipediaSearch(celebNameString));
            }
        });

    }
    public String bdayWikipediaSearch(String name){
        String
baseUrl=String.format("https://en.wikipedia.org/w/index.php?search=%s&title=Special:Sea
rch&profile=default&fulltext=1",name);
        return baseUrl;
    }
        private class wikiSearchParser extends AsyncTask<Void, Void, Void> {

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected Void doInBackground(Void... params) {
            try {
                Document doc = Jsoup.connect(searchUrl).get();
                org.jsoup.select.Elements results=doc.select(".mw-search-result-heading
a");
                Element firstResult = results.first();
                resourceUrl="https://en.wikipedia.org"+firstResult.attr("href");
```

```
            } catch (IOException e) {
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onPostExecute(Void result) {
            bday.append("\n\n\n"+resourceUrl);

        }
    }
    private class wikiArticleParser extends AsyncTask<Void, Void, Void> {

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected Void doInBackground(Void... params) {
            try {
                Document doc = Jsoup.connect(resourceUrl).get();
                org.jsoup.select.Elements paras=doc.select(".mw-content-ltr p");
                Element firstPara = paras.first();
                bdate=firstPara.text();
                if(bdate.contains("born")){
                    int i=bdate.indexOf("born");
                    int f=bdate.indexOf(")",i);
                    bdate=bdate.substring(i,f);
                }
                else{
                    bdate="Sorry not found!";
                }

            } catch (IOException e) {
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onPostExecute(Void result) {
            bday.append("\n\n\n"+bdate);

        }
    }
}
```

Layout: `activity_main.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.bragitoff.celebbday.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enter the celeb name:"
        android:layout_alignParentLeft="true"
        android:id="@+id/text"
        />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/text"
        android:id="@+id/celebName"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:id="@+id/button"
        android:layout_below="@+id/celebName"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:id="@+id/bday"
        android:layout_below="@+id/button"
        />

</RelativeLayout>
```

Gradle Dependency:

```
compile 'org.jsoup:jsoup:1.10.2'
```

Permission:

```
<uses-permission android:name="android.permission.INTERNET" />
```

The logic behind the code is pretty naïve, and in many cases may not work.
This was done just as an experiment without any intention of being perfect but more as a practice project.

What the code does is, that it finds the first paragraph of a Wikipedia article on a celebrity name entered by the user.

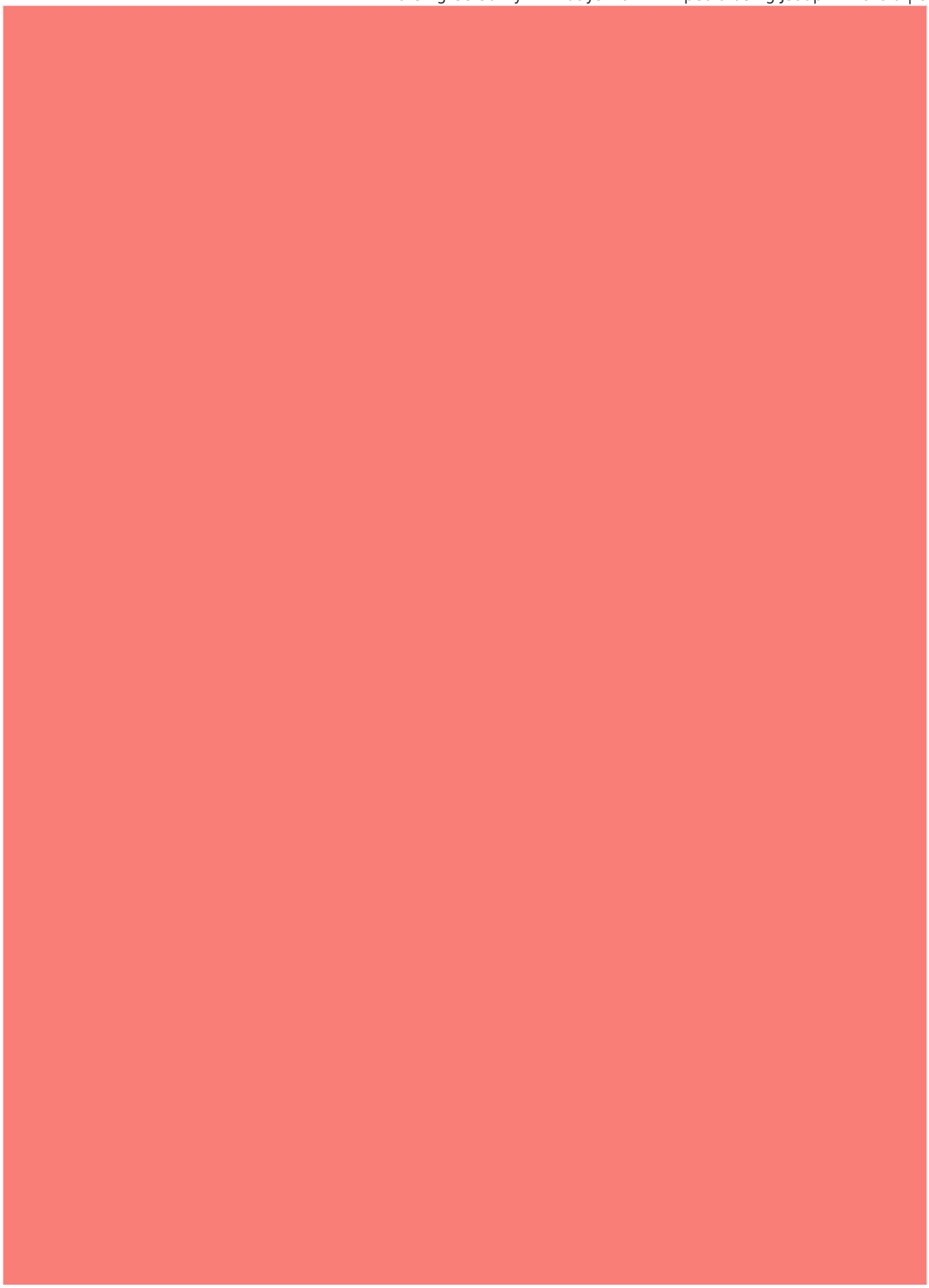
Then it searches the paragraph for the string ‘born’ and using that it prints out the birthday.
This works almost all the time for alive celebrities but will most likely fail for dead celebrities as Wikipedia shows the birthday of dead celebrities in a different manner.
Although one can easily modify the above code to parse the birthday of alive and dead celebrities both by adding a few more conditions and cases by researching a few articles and finding a trend in them.

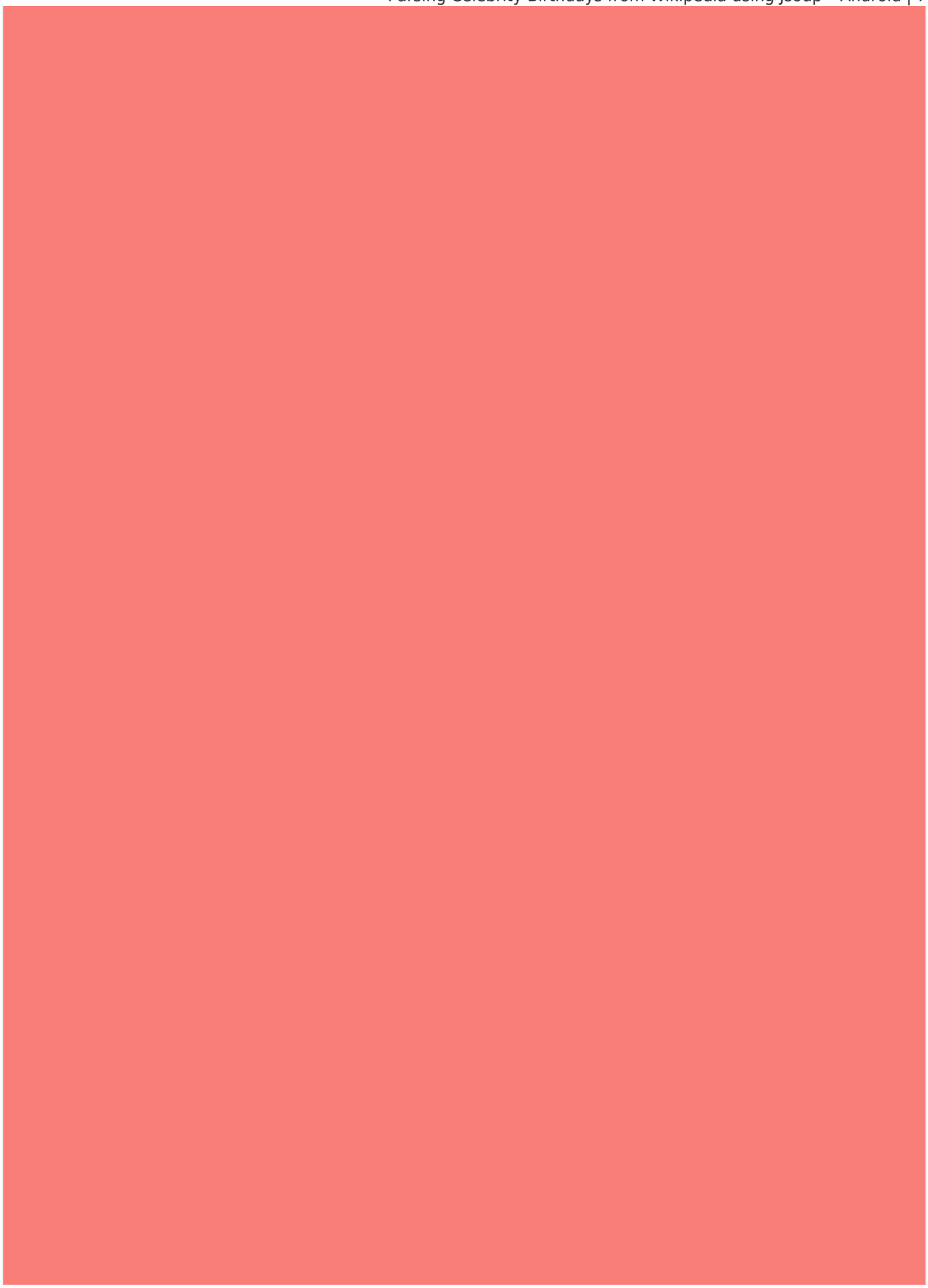
Hope you found it useful!
If you have any comments/questions then drop them in the comments section down below.

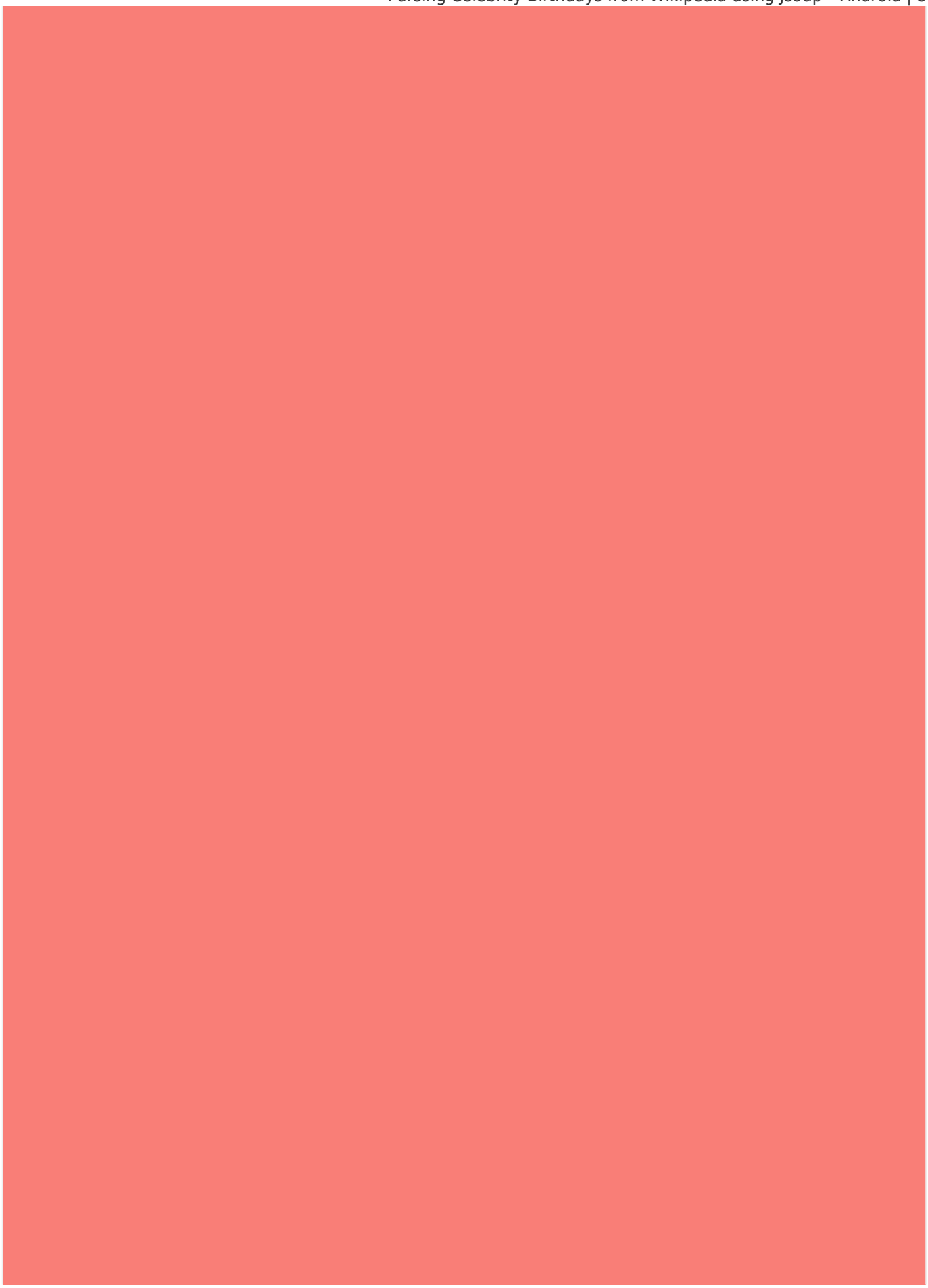
Manas Sharma

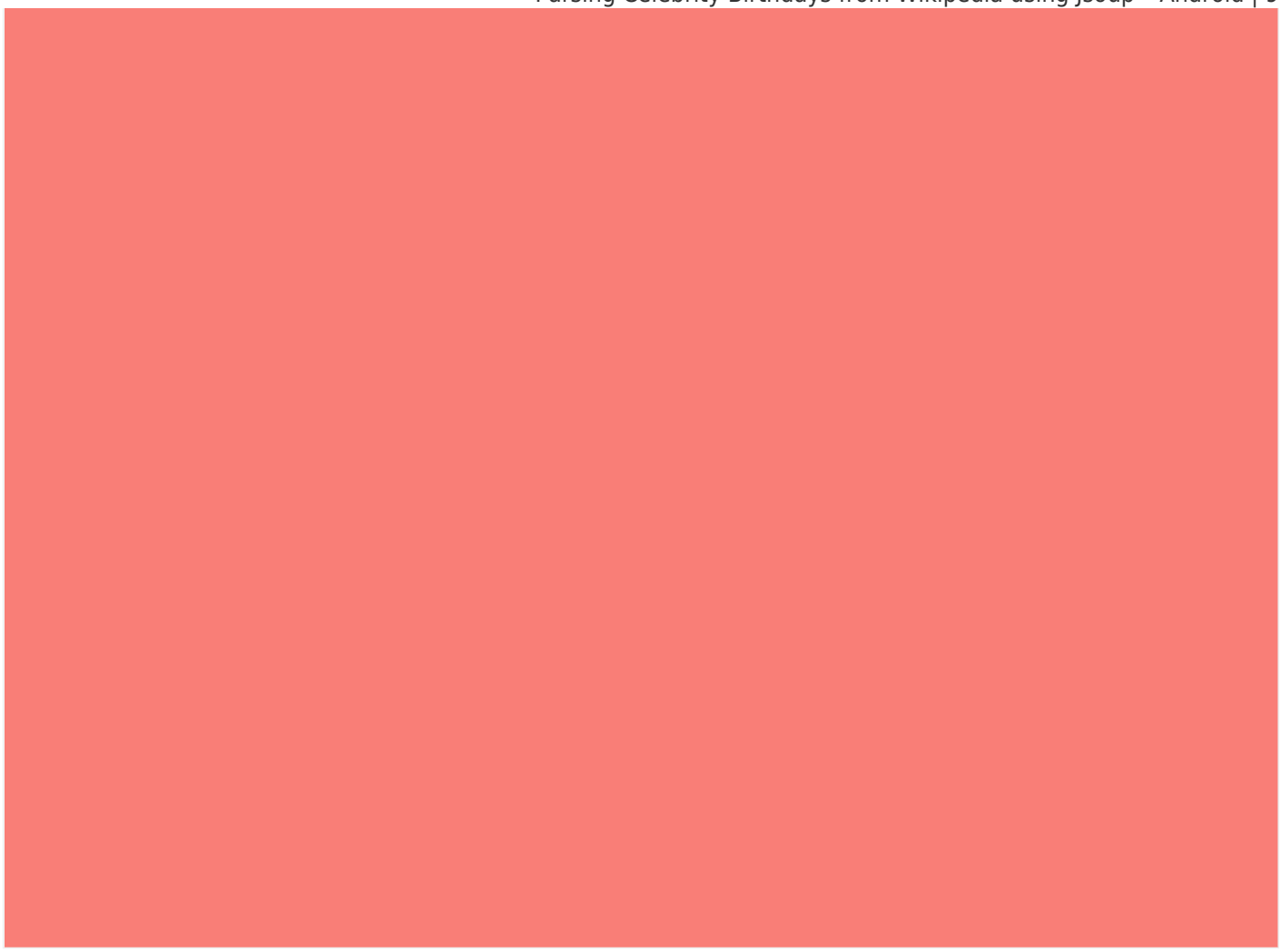
I’m a physicist specializing in computational material science with a PhD in Physics from Friedrich-Schiller University Jena, Germany. I write efficient codes for simulating light-matter interactions at atomic scales. I like to develop Physics, DFT, and Machine Learning related apps and software from time to time. Can code in most of the popular languages. I like to share my knowledge in Physics and applications using this Blog and a YouTube channel.

manas.bragitoff.com/









Share this:

Click to share on Facebook (Opens in new window)

Click to share on Twitter (Opens in new window)

Click to share on WhatsApp (Opens in new window)

Click to share on Pinterest (Opens in new window)

Click to share on Reddit (Opens in new window)

Click to share on LinkedIn (Opens in new window)

Click to email a link to a friend (Opens in new window)

Click to print (Opens in new window)

Click to share on Tumblr (Opens in new window)

Click to share on Pocket (Opens in new window)

Click to share on Telegram (Opens in new window)

[wpedon id="7041" align="center"]