Introduction to Android Development

Android is one of the most popular operating systems with over 88% of phones running android.

Android Versions

CODE NAME	VERSION NUMBER	API LEVEL	RELEASE NAME
No Code Name	1.0	1	September 23,2008
No Code Name	1.1	2	February 9,2009
Cupcake	1.5	3	April 27,2009
Donut	1.6	4	September 15,2009
Eclair	2.0 – 2.1	5 – 7	October 26,2009

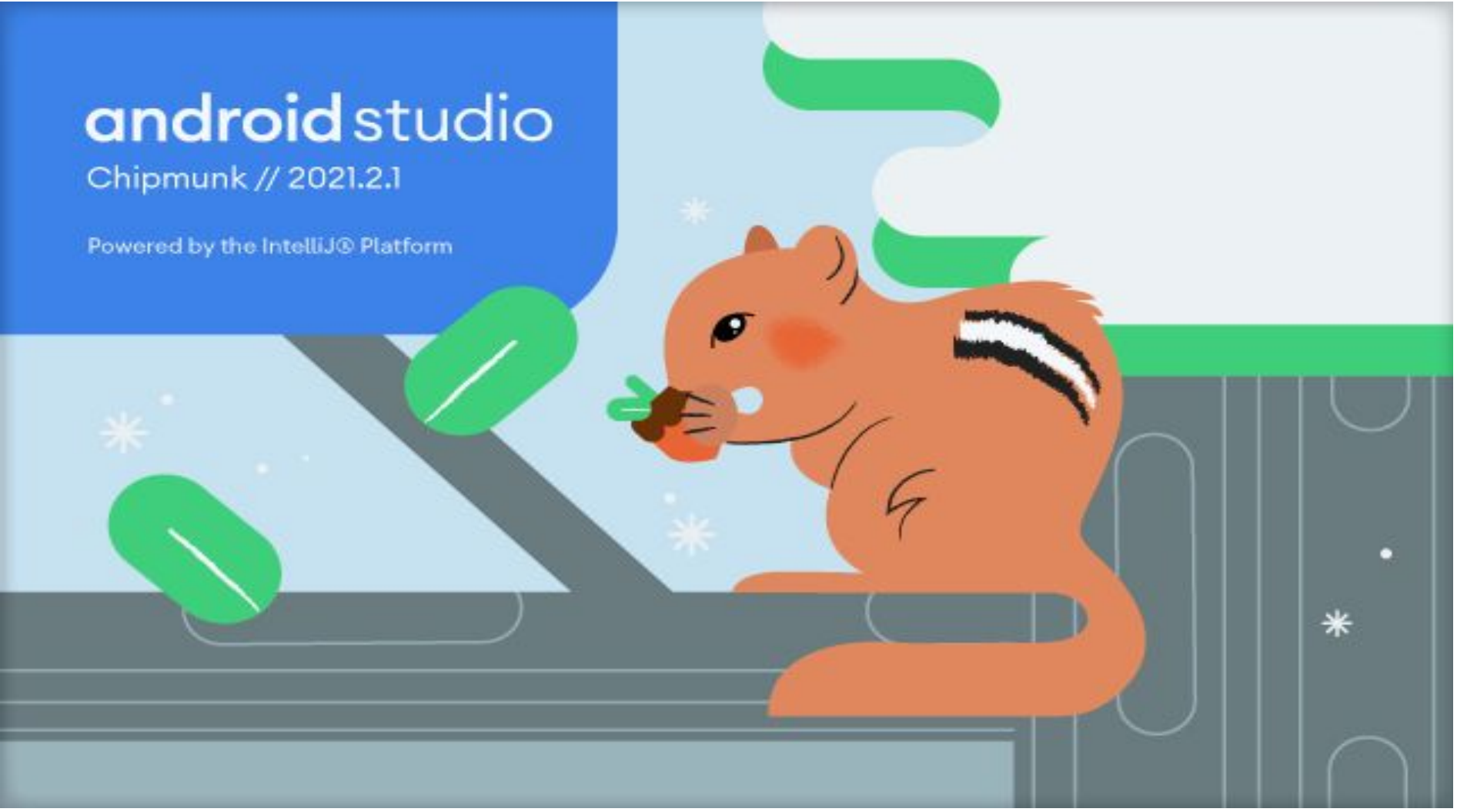
Froyo	2.2 – 2.2.3	8	May 20,2010
Gingerbread	2.3 – 2.3.7	9 – 10	December 6,2010
Honeycomb	3.0 – 3.2.6	11 – 13	February 22,2011
Ice Cream Sandwich	4.0 – 4.0.4	14 – 15	October 18,2011
Jelly Bean	4.1 – 4.3.1	16 – 18	July 9,2012

KiKat	4.4 – 4.4.4	19 – 20	October 31,2013
Lollipop	5.0 – 5.1.1	21 – 22	November 12,2014
Marshmallow	6.0 – 6.0.1	23	October 5,2015
Nougat	7.0	24	August 22,2016
Nougat	7.1.0 – 7.1.2	25	October 4,2016
Oreo	8.0	26	August 21,2017

Oreo	8.1	27	December 5,2017
Pie	9.0	28	August 6,2018
Android 10	10.0	29	September 3,2019
Android 11	11	30	September 8,2020
Android 12	12	31	October 4, 2021

Android studio

Android Studio is official IDE for Google's Android operating system , built on IntelliJ IDEA . Android Studio Available for download onWindows , macOS and Linux based operating systems.


android studio
Chipmunk // 2021.2.1
Powered by the IntelliJ® Platform

XML & Java code

Android uses xml to declare layouts and java to provide logic.

When you create a android project 2 files get generated MainActivity(java) and activity_main(xml) , the xml file is used to create the views which you will be setting in the java file in the setContentView .

The android build system created R.java file which contains your xml ids and other xml declaration . the java file can access the views in the xml by referring to R.id,R.string etc . basically its like a address of the xml view which you can refer from java .

The Layout Editor

The layouts editor is used to quickly build layouts by dragging UI elements which is easier to write XML by hand.

We can set up different attributes easily using the design mode in the layout editor.

To create your new Android project, follow these steps:

- Install the latest version of Android Studio.
- In the Welcome to Android Studio window, click Create New Project.
- If you have a project already opened, select File > New > New Project.


Welcome to Android Studio
Android Studio
Version 4.1.1
Create New Project
Open an Existing Project
Get from Version Control
Profile or Debug APK
Import Project (Gradle, Eclipse ADT, etc.)
Import an Android Code Sample
Configure
Get Help

- If you have a project already opened, select File > New > New Project.

- In the Select a Project Template window, select Empty Activity and click Next.
- In the Configure your project window, complete the following:
- Enter "My First App" in the Name field.
- Enter "com.example.myfirstapp" in the Package name field.
- If you'd like to place the project in a different folder, change its Save location.
- Select either Java or Kotlin from the Language drop-down menu.
- Select the lowest version of Android you want your app to support in the Minimum SDK field.

- If your app will require legacy library support, mark the Use legacy android.support libraries checkbox.
- Leave the other options as they are.
- Click Finish.

What is an APK?

An APK is a collection of different files (like code, audio, video, etc.) compiles and bundled into a single file.

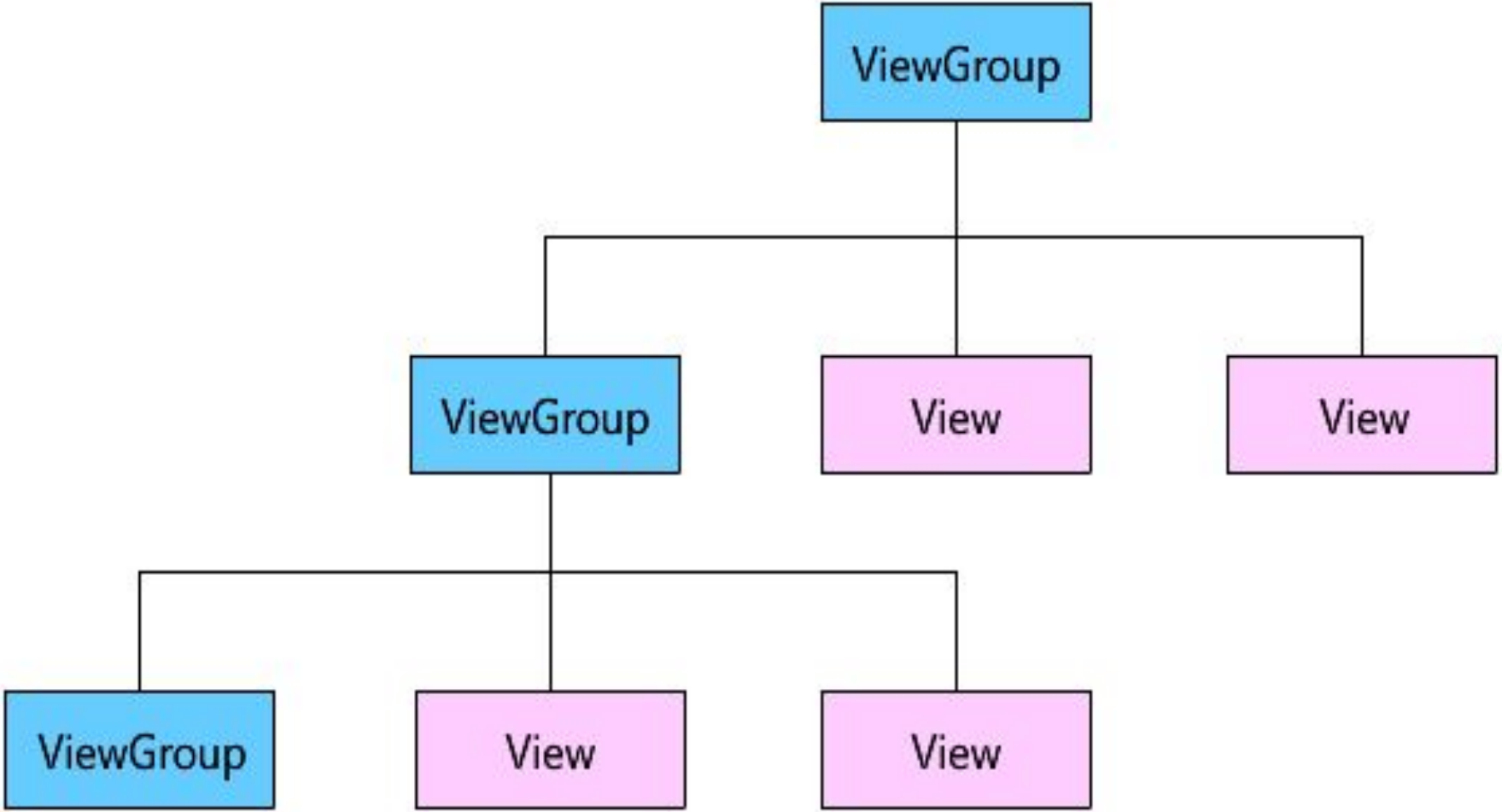
What is an AVD?

AVD stands for Android virtual device AVC is an emulator configuration that simulates a physical Android device.

Android UI layouts

View is the base class for widgets (like buttons, text fields etc.)
View = Basic building blocks
View group holds view and view group. (just like a box can hold objects and more boxes)

A layout defines the structure for a user interface in your app, such as in an activity. All elements in the layout are built using a hierarchy of View and ViewGroup objects. A View usually draws something the user can see and interact with. Whereas a ViewGroup is an invisible container that defines the layout structure for View and other ViewGroup objects, as shown in figure 1.


ViewGroup
ViewGroup
View
View
ViewGroup
View
View

XML vs Java in Android

XML is the skeleton code that describes the Uld layout java drives this XML

Linear Layout

Linear layout is a simple layout used in android for layout designing. In the Linear layout all the elements are displayed in linear fashion means all the childs/elements of a linear layout are displayed according to its orientation. The value for orientation property can be either horizontal or vertical.

1.Vertical:

In this all the child are arranged vertically in a line one after the other. In below code snippets we have specified orientation “vertical” so the childs/views of this layout are displayed vertically.

```
<LinearLayout
xmlns:android=http://schemas.android.com/apk/res/android
   Android:layout_width=”fill_parent”
   Android:layout_height=”wrap_content”
   Android:orientation=”vertical”>
   <Button
      Android:layout_width=”wrap_content”
      Android:layout_height=”wrap_content”
      Android:text=”Button1”
      Android:id=”@+id/button”
      Android:background=”#358a32” />

   <Button
      Android:layout_width=”wrap_content”
      Android:layout_height=”wrap_content”
      Android:text=”Button2”
      Android:id=”@+id/button2”
      Android:background=”#0058b6” />

</LinearLayout>
```

2. Horizontal:

In this all the child are arranged horizontally in a line one after the other. In below code snippets we have specified orientation “horizontal” so the childs/views of this layout are displayed horizontally.

```
<LinearLayout
xmlns:android=http://schemas.android.com/apk/res/android
   Android:layout_width=”fill_parent”
   Android:layout_height=”wrap_content”
   Android:orientation=”horizontal”>

   <Button
      Android:layout_width=”wrap_content”
      Android:layout_height=”wrap_content”
      Android:text=”Button1”
      Android:id=”@+id/button”
      Android:background=”#358a32” />

   <Button
      Android:layout_width=”wrap_content”
      Android:layout_height=”wrap_content”
      Android:text=”Button2”
      Android:id=”@+id/button2”
      Android:background=”#0058b6” />

</LinearLayout>
```

Buttons

A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it.

XML

```
<Button
    Android:id="@+id/button_id"
    Android:layout_height="wrap_content"
    Android:layout_width="wrap_content"
    Android:text="@string/self_destruct" />
```

Java

```
Public class MyActivity extends Activity {
```

```
    Protected void onCreate(Bundle savedInstanceState)
{
        Super.onCreate(savedInstanceState);

        setContentView(R.layout.content_layout_id);

        final Button button = findViewById(R.id.button_id);
        button.setOnClickListener(new
View.OnClickListener() {
            public void onClick(View v) {
                // Code here executes on main thread after
user presses button
            }
        });
    }
}
```

TextView

A user interface element that displays text to the user

XML

```
<LinearLayout

Xmlns:android=http://schemas.android.com/apk/res/android
    Android:layout_width=”match_parent”
    Android:layout_height=”match_parent”>
  <TextView
     Android:id=”@+id/text_view_id”
     Android:layout_height=”wrap_content”
     Android:layout_width=”wrap_content”
     Android:text=”@string/hello” />
 </LinearLayout>
```

JAVA

```
Public class MainActivity extends Activity {

   Protected void onCreate(Bundle savedInstanceState) {
      Super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      final TextView helloTextView = (TextView)
findViewById(R.id.text_view_id);
      helloTextView.setText(R.string.user_greeting);
```

```
    }
}
```

EditText

A user interface element for entering and modifying text.

EditText code in XML:

```
<EditText
Android:id="@+id/simpleEditText"
android:layout_height="wrap_content"
android:layout_width="match_parent"/>
```

JAVA File

```
EditText simpleEditText = (EditText) findViewById(R.id.simpleEditText);
String editTextValue = simpleEditText.getText().toString();
```

Example

XML

```
<LinearLayout
xmlns:android=http://schemas.android.com/apk/res/android
   Android:layout_width=”match_parent”
   Android:layout_height=”match_parent”
   Android:gravity=”center”
      Android:orientation=”vertical”>

   <TextView
      Android:text=”@string/hello_world”
      Android:layout_width=”wrap_content”
      Android:layout_height=”wrap_content”
            Android:textSize=”25sp”
            Android:id=”@+id/tv1”/>
      <EditText
            Android:layout_width=”wrap_content”
            Android:layout_height=”wrap_content”
            Android:id=”@+id/et1”
            Android:hint=”Enter your name”/>
```

```
<Button
    Android:layout_width=”wrap_content”
    Android:layout_height=”wrap_content”
    Android:id=”@+id/bt1”
    Android:text=”Click me”
    Android:textSize=”25sp”
    Android:textColor=”#098154”
    Android:backgroundtint=”#ffffff”
    Android:onClick=”call”
    />

</LinearLayout>
```

Java File

```
package com.mycompany.myapp;
import android.app.*;
import android.os.*;
import android.widget.*;
import android.view.*;
```

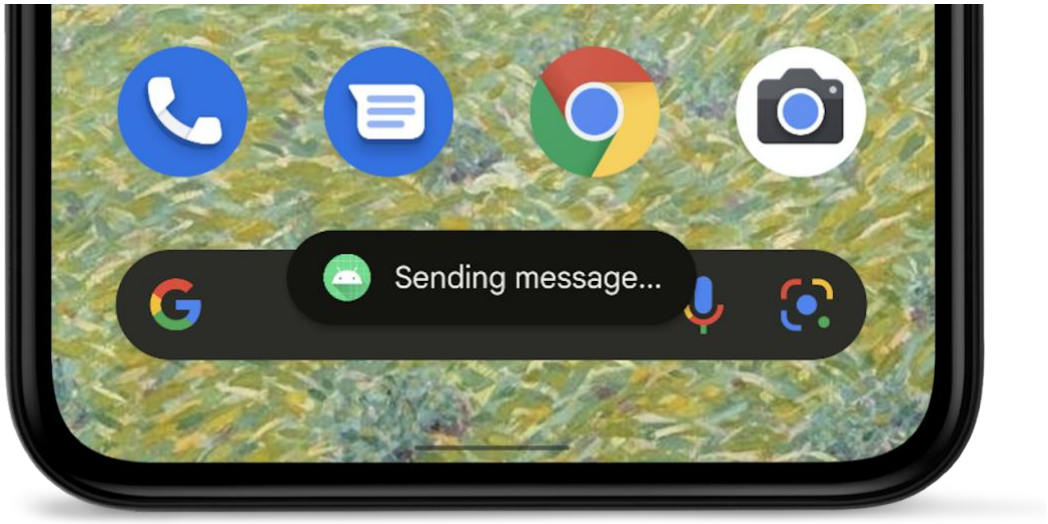
```
public class MainActivity extends Activity
{
    TextView t1;
    EditText e1;
  @Override
  protected void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

        e1=(EditText)findViewById(R.id.et1);
        t1=(TextView) findViewById(R.id.tv1);
  }
    public void call(View view){
        String name=et1.getText().toString();
        t1.setText("Hello " +name);
    }
}
```

TechBuzz

Toasts

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.

For example, clicking Send on an email triggers a "Sending message…" toast, as shown in the following screen capture:



Instantiate a Toast object

Use the makeText() method, which takes the following parameters:

1. The application Context.
2. The text that should appear to the user.
3. The duration that the toast should remain on the screen.

The makeText() method returns a properly initialized Toast object.

Show the toast

To display the toast, call the show() method, as demonstrated in the following example:

```
Toast.makeText(context, text, duration).show();
```

Example

```
Context context = getApplicationContext();

String  text = "Hello toast!";

Int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);

Toast.show();
```

Example

Toast toast=Toast.makeText(getApplicationContext(),"Hello World",Toast.LENGTH_SHORT).show();

WebView

WebView is a view that display web pages inside your application. You can also specify HTML string and can show it inside your application using WebView. WebView makes turns your application to a web application.

In order to add WebView to your application, you have to add <WebView> element to your xml layout file. Its syntax is as follows –

```
<WebView
xmlns:android=http://schemas.android.com/apk/res/android
  Android:id=”@+id/webview”
  Android:layout_width=”fill_parent”
  Android:layout_height=”fill_parent”      />
```

In order to use it, you have to get a reference of this view in Java file. To get a reference, create an object of the class WebView. Its syntax is –

WebView browser = (WebView) findViewById(R.id.webview);

In order to load a web url into the WebView, you need to call a method loadUrl(String url) of the WebView class, specifying the required url. Its syntax is:

Browser.loadUrl(https://www.studysection.com/Users/login,);

canGoBack()

This method specifies the WebView has a back history item.