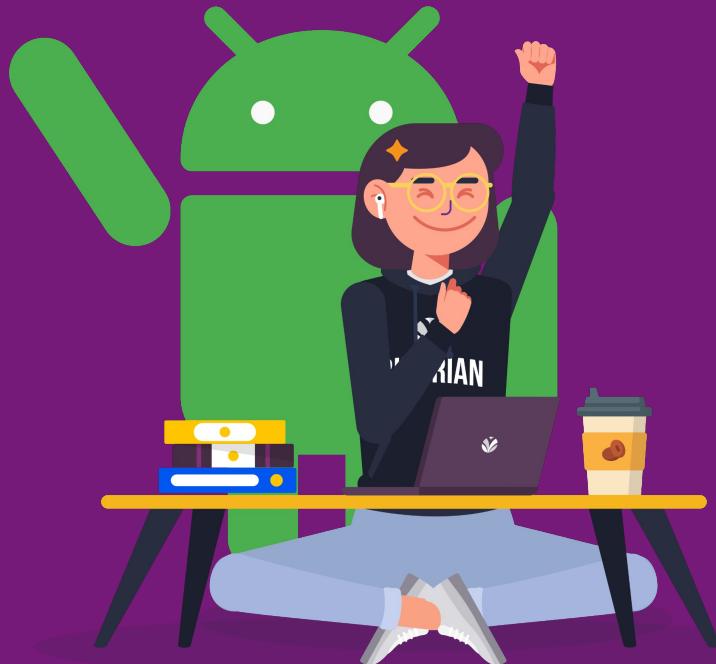


Designing UI in Android

Silver Chapter 2 - Topic 4

Selamat datang di **Chapter 2 Topic 4**
online course Android Developer dari
Binar Academy!

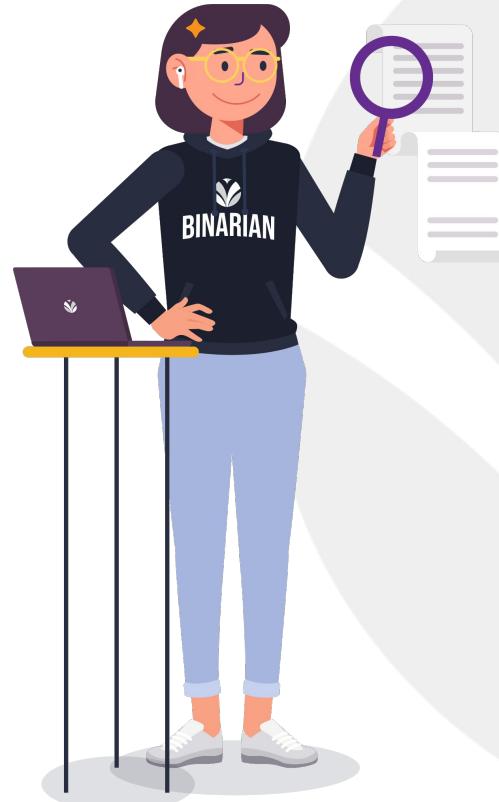


Hai teman-teman 

Masih di Chapter 2 niih...

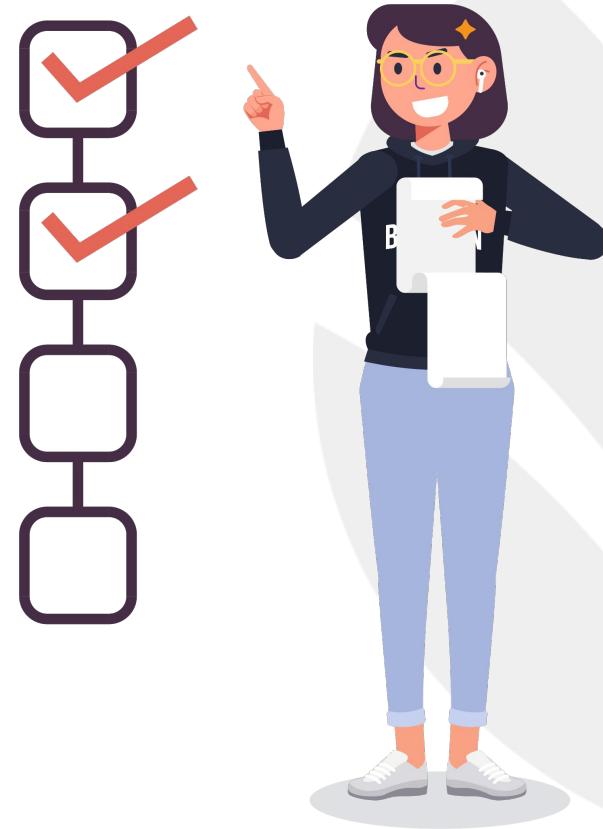
Di topik sebelumnya, kita sudah bahas tentang aturan penamaan atau code convention di Android Studio.

Pada **Topik 4** ini, kita akan belajar **cara menerapkan Android UI layouting**. Yuk, lanjut!



Detailnya, kita bakal bahas hal-hal berikut ini :

- Konsep Layout Android
- Layouting dengan XML
- View & ViewGroup
- Menerapkan Android UI Layouting



Topic ini difokuskan pada acceptance criteria berikut ini:

Memasukkan project ke dalam GIT

Menggunakan View dan ViewGroup untuk membuat layout

Menggunakan file gambar (.svg, .png, .jpg) untuk membuat layout

Membuat layout semirip mungkin dengan contoh dengan menerapkan styling yang berbeda atau semenarik mungkin

Menerapkan android styling dengan memodifikasi styling dari template Android Studio



LAYOUT

Layout pada Android merupakan suatu tampilan tata letak di Android yang berguna untuk mengatur tata letak setiap komponen tampilan.

XML

Menulis / Memuat

View

- TextView
- EditText
- Button
- CheckBox
- RadioButton
- ImageView
- ImageButton
- ProgressBar
- Spinner

ViewGroup

- LinearLayout
- RelativeLayout
- ConstraintLayout
- TableLayout
- FrameLayout
- WebView
- ListView
- GridView

Yupp, kita bakal ngomongin layout beserta cara pembuatan layout khusus dengan XML.

Pertama, apa sih layout itu?



Pasti kamu pernah pakai aplikasi dong?

Sekarang banyak banget aplikasi yang tersedia di PlayStore dengan ragam pelayanan yang ditawarkan.

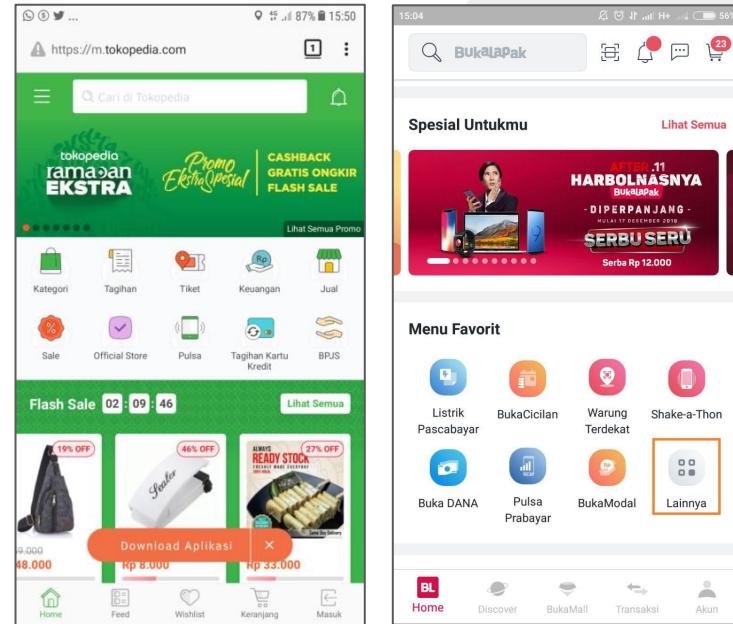
Masing-masing aplikasi yang ada di PlayStore menawarkan fitur, warna dan tampilan yang pastinya berbeda antara satu sama lain~



Jangankan yang berbeda, aplikasi sejenis aja tampilannya bisa variatif

Coba lihat deh dua tampilan aplikasi di samping. Walaupun sama-sama bergerak di e-commerce, tapi tampilannya berbeda banget kan?

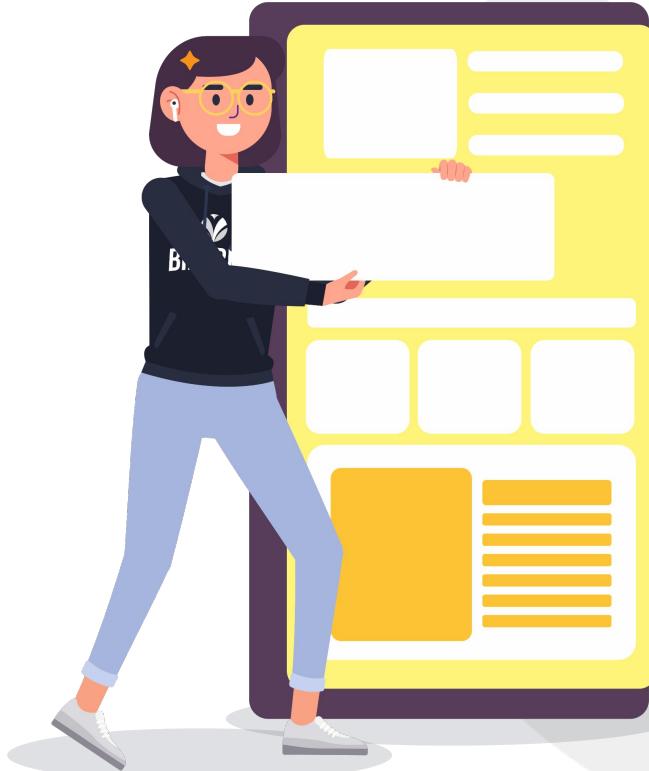
Nah, kita akan memperdalam nih dasar-dasar membuat tampilan alias layout kayak gambar itu.



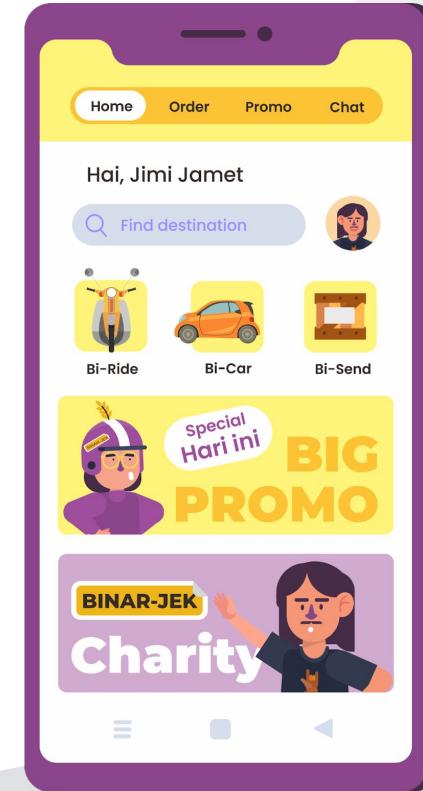
Jadi, apa itu layout?

Layout merupakan **tata letak dari suatu elemen desain**.

Layout mencakup pengaturan penempatan teks, gambar atau komponen lainnya pada aplikasi sehingga tampilan yang dibuat terlihat rapi dan nyaman untuk pengguna.

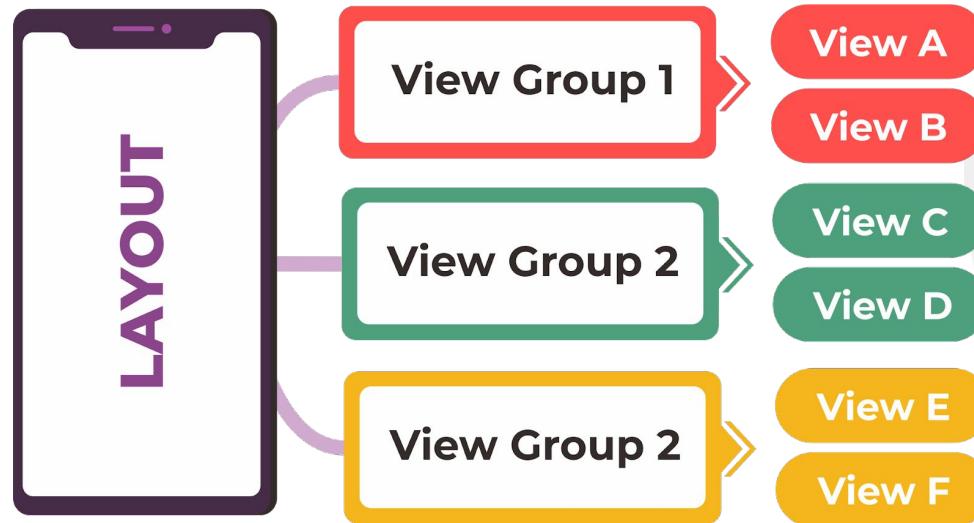


Dasar dari semua komponen tampilan di Android adalah **object view yang dibuat dari view class** dan menempati area yang terdapat di layar. Ia bertanggung jawab untuk menggambar dan menangani event yang terjadi.



Layout meliputi **View** dan **ViewGroup**.

View mewakili **komponen yang dapat kita lihat** seperti teks, tombol, dan gambar. Sedangkan ViewGroup adalah yang **mengatur semua komponen itu tertata rapi**.

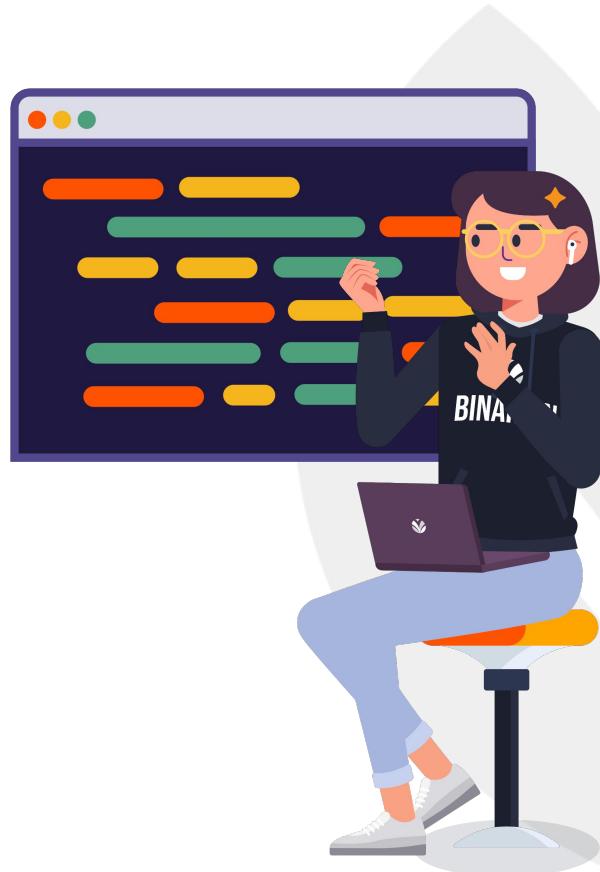


Gampangnya, **ViewGroup** adalah parent dan **views** di dalamnya adalah child-nya.

Baik View maupun ViewGroup, keduanya saling membutuhkan satu sama lain, sehingga kita bisa membuat tampilan yang enak dipandang.

Kita juga bisa membuat layout khusus dengan membuat kelas yang diturunkan dari ViewGroup.

Nah, di sinilah coding skill kita yang sudah dipersiapkan sebelumnya akan bersinar. Semua yang kita lakukan untuk membuat layout khusus nggak akan jauh jauh dari **XML**.



<XML/>

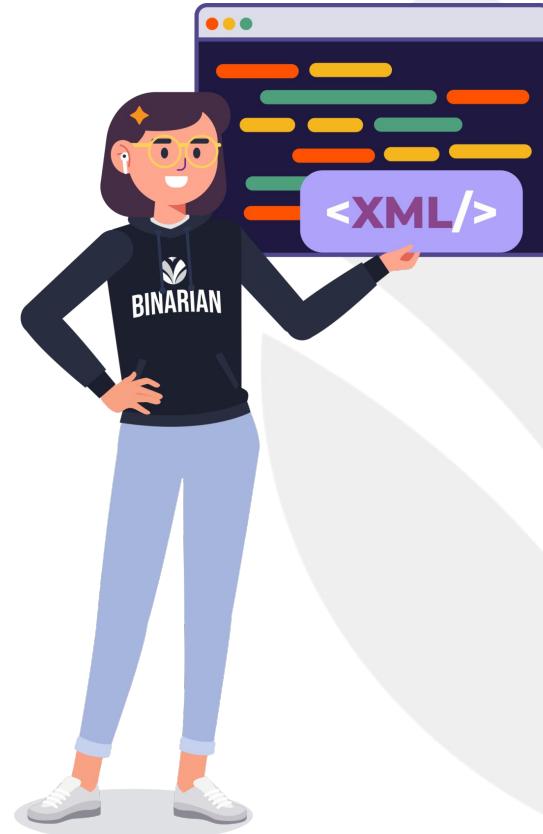
Untuk membuat layout khusus dengan Android Studio kita harus kenalan dulu sama XML.



Jadi, XML itu apa sih?

Android XML adalah format pemrograman dalam bentuk teks yang bisa memberi keleluasaan pada kita dalam **mendesain UI layout dan elemen layar lain yang dikandungnya dengan cepat**.

Kenapa cepat? Karena caranya sama dengan membuat halaman web dalam HTML, yaitu menggunakan serangkaian elemen yang bersarang.



Setiap file layout harus berisi **satu elemen root**, yang harus berupa object **View** atau **ViewGroup**.

Setelah kita mendefinisikan elemen root, kita dapat menambahkan layout object atau widget tambahan sebagai elemen turunan untuk secara bertahap membangun hirarki View yang menentukan layout kita.

View

Text View

Hai Jimi Jamet

Image View



Button View

Home

Promo

Kalau disimpulkan layout bisa dibuat dengan dua cara:

1. Mendeklarasikan UI dalam bentuk Code XML

Android menyediakan XML yang langsung sesuai dengan kelas view dan subclassnya, seperti yang digunakan widget dan layout.

Jika kita menggunakan versi Android Studio terbaru, maka layout XML akan otomatis menggunakan **ConstraintLayout**.

Kita juga dapat menggunakan Android Studio layout editor untuk membangun layout XML dengan **drag-and-drop**.



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Fragment baru" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

2. Instantiate layout elements saat runtime

Aplikasi kita dapat membuat object View dan ViewGroup (dan memanipulasi properties-nya) secara terprogram.



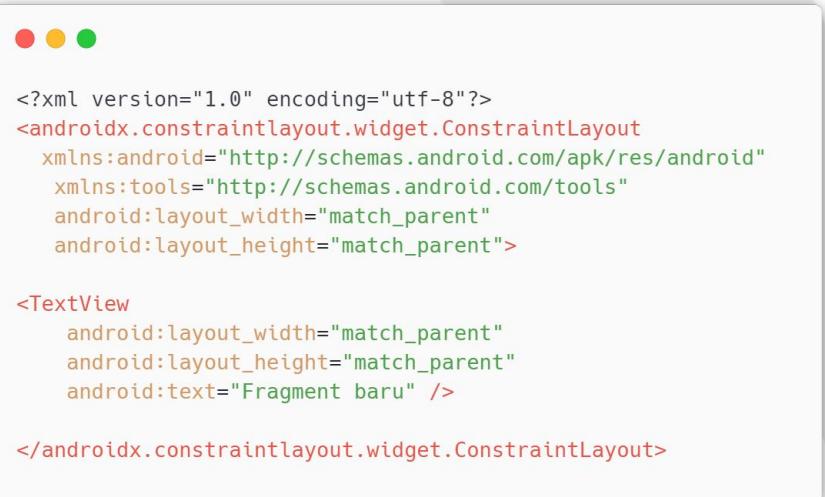
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Fragment baru" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Mendeklarasikan UI dalam bentuk XML membuat kita dapat memisahkan tampilan aplikasi dengan kode untuk mengontrol logic.

Opsi ini juga mempermudah kita dalam menyediakan layout yang berbeda untuk ukuran dan orientasi layar yang berbeda.



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Fragment baru" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

A screenshot of an Android application window. At the top, there are three colored dots (red, yellow, green) for navigation. The main content area contains a single red button with the text "Fragment baru" centered on it. The window has a standard title bar and a decorative white semi-circle at the bottom right corner.

Framework Android memberi kita fleksibilitas untuk menggunakan salah satu atau kedua method ini untuk membangun UI aplikasi.

Sebagai contoh, kita dapat mendeklarasikan layout default aplikasi kita dalam XML, kemudian memodifikasi layout tersebut saat runtime.



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Fragment baru" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Contoh di samping merupakan layout XML yang menggunakan **LinearLayout** vertikal untuk menata TextView dan Button.

Setelah kita mendeklarasikan layout dalam XML, simpan file dengan ekstensi .xml, di folder res/layout/ pada project Android kita, jadi itu akan dikompilasi dengan benar.

Informasi lebih lanjut tentang syntax untuk file layout XML tersedia di dokumen Layout Resources.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

Hello, I am a TextView

HELLO, I AM A BUTTON

Memuat Ulang (Loading) XML

Saat compiling aplikasi, setiap file layout XML dikompilasi ke resource view. Kita harus memuat resource layout dari kode aplikasi dalam method **onCreate()** pada activity.

Method **onCreate()** dalam activity akan dipanggil oleh Android Framework **saat activity kita diluncurkan.**



```
fun onCreate(savedInstanceState: Bundle) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main_layout)
}
```

Cara memuat ulang XML, lakukan dengan memanggil **setContentView()**, lalu berikan referensi ke resource layout kita dalam bentuk: **R.layout.layout_file_name**.

Sebagai contoh, jika layout XML kita disimpan sebagai **main_layout.xml**, kita akan memuatnya untuk activity kita seperti di samping:



```
fun onCreate(savedInstanceState: Bundle) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.main_layout)  
}
```

<View/>

Kamu masih inget nggak? Di bagian slide introduction, XML bercabang menjadi dua, yaitu View dan ViewGroup.

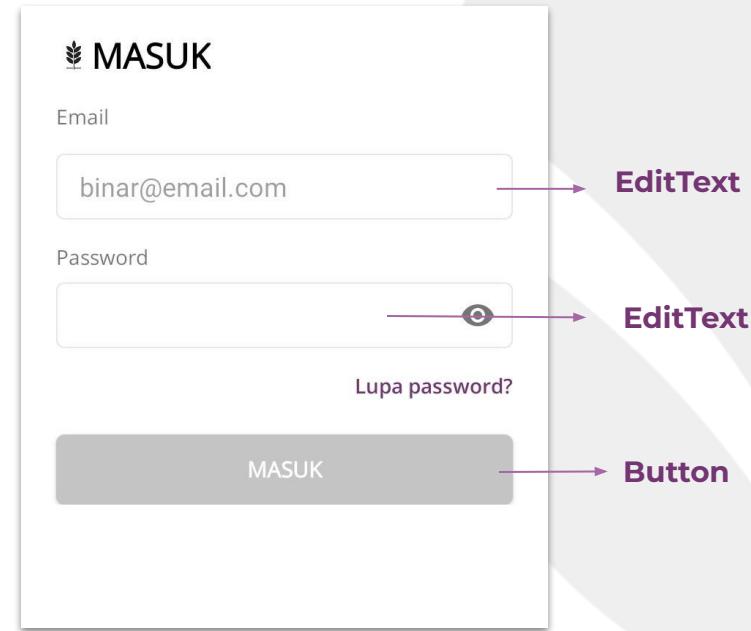
Kita mulai masuk ke situ nih. Kita mulai bahas dari **View** yuk!

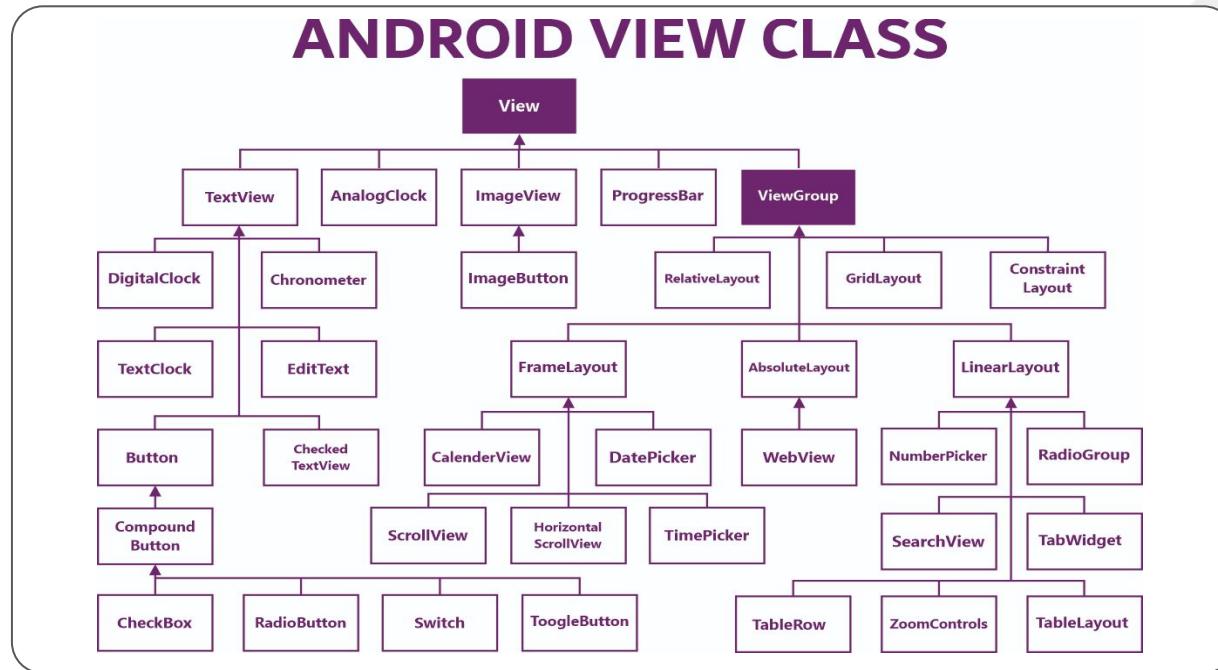


Apa itu Android View?

Android View adalah **kelas dasar untuk semua komponen UI di Android**.

Misalnya, class **EditText** digunakan untuk menerima input dari user di aplikasi Android yang merupakan subclass dari **View**.

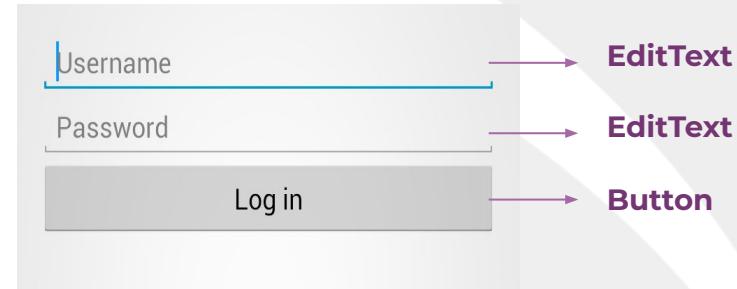




Gambar di atas menggambarkan hierarki pewarisan antar tampilan di Android. Walaupun terlihat bervariasi, tapi kita cukup mempelajari beberapa View yang sering digunakan dalam programming Android aja.

Berikut ini adalah beberapa **subclass View** umum yang akan digunakan dalam aplikasi Android:

1. TextView
2. EditText
3. Button
4. CheckBox
5. RadioButton
6. ImageButton
7. ProgressBar
8. Spinner



Yuk kita bahas satu-satu~

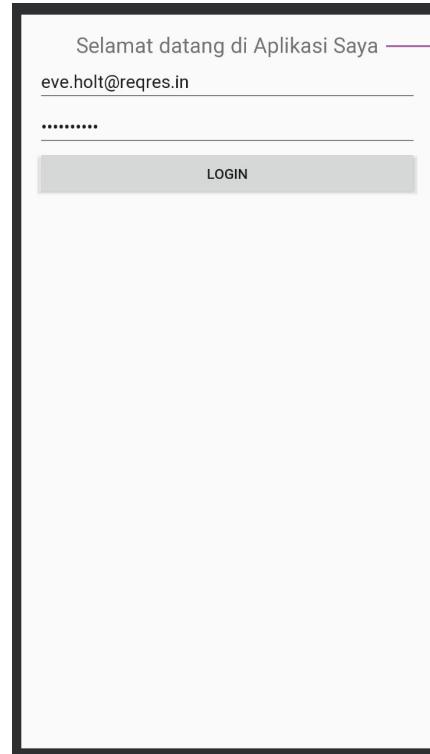
View - TextView

TextView adalah komponen yang digunakan untuk **mengatur dan menampilkan teks ke layar**.

Itu akan bertindak seperti komponen label. Walaupun begitu, user tidak bisa mengubah atau mengedit text yang ada di TextView.

Pada Android, kita dapat membuat komponen TextView dengan dua cara:

- menulisnya dalam file layout XML, atau
- membuatnya dalam file secara terprogram.



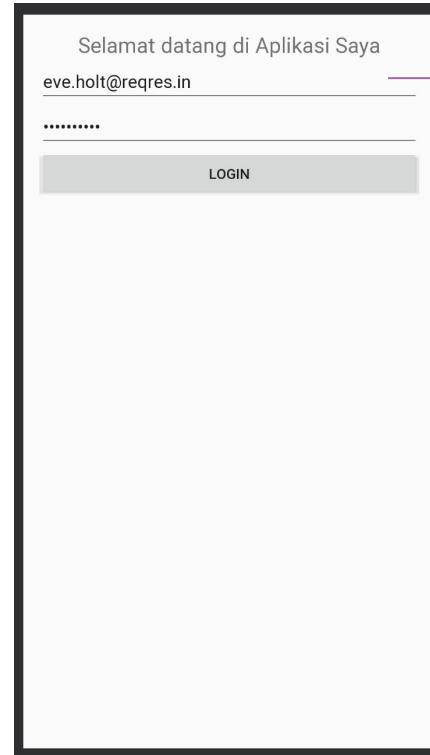
TextView

View - EditText

EditText adalah komponen yang digunakan untuk **menginput berupa teks**.

EditText bertindak sebagai penampung teks yang kita ketik. EditText bisa menangani berbagai macam input, seperti:

- input untuk email,
- password,
- nomor telepon,
- nomor pin, dll.



EditText

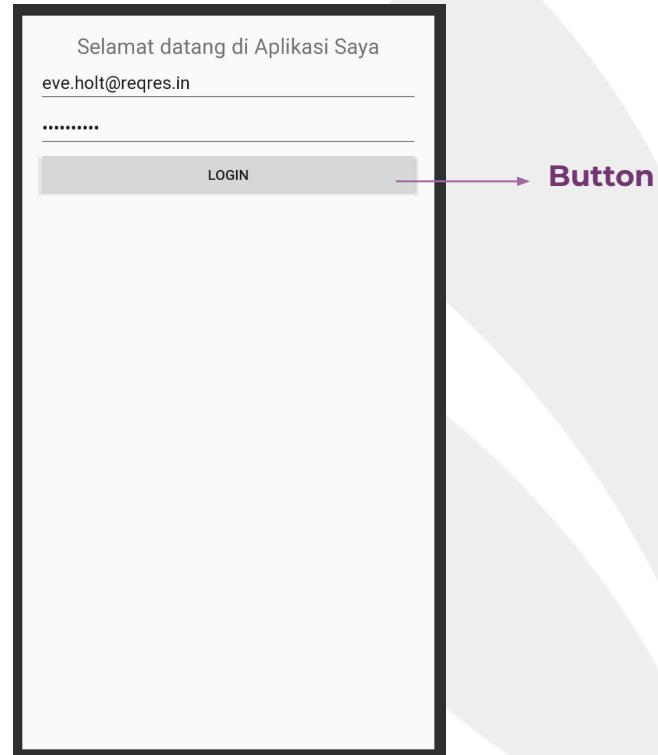
View - Button

Button adalah **komponen yang digunakan untuk melakukan action setiap kali user mengetuk atau menahannya.**

Secara umum, button akan berisi teks atau ikon atau keduanya dan melakukan suatu action ketika user menyentuhnya.

Kita dapat menggunakan berbagai jenis button yang tersedia untuk digunakan, yaitu:

- **ImageButton**,
- **ToggleButton**, dan
- **RadioButton**.

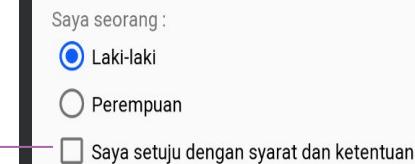


View - CheckBox

Kita bisa menggunakan CheckBox untuk **menandai sebuah atau beberapa pilihan**.

Biasanya kita menemui penggunaan CheckBox ketika awal registrasi yang menandai kalau kita setuju dengan syarat dan ketentuan penggunaan aplikasi.

CheckBox



A snippet of a registration form. It includes a question "Saya seorang:", two radio buttons for "Laki-laki" (selected) and "Perempuan", and a checkbox for "Saya setuju dengan syarat dan ketentuan". A large grey button labeled "DAFTAR" is at the bottom.

Saya seorang :

Laki-laki
 Perempuan
 Saya setuju dengan syarat dan ketentuan

DAFTAR

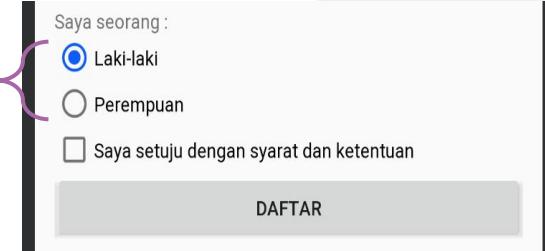
View - RadioButton

Kita dapat menggunakan komponen **RadioButton** di aplikasi Android seperti **CheckBox**, namun user hanya bisa memilih satu opsi saja.

Untuk menggabungkan beberapa **RadioButton** menjadi satu grup, kita bisa menggunakan **RadioGroup**. Itu akan memastikan bahwa user hanya dapat memilih satu opsi dari grup beberapa opsi.

Contohnya seperti pertanyaan jenis kelamin di dalam survei.

RadioButton



Secara default, RadioButton akan muncul dalam keadaan OFF (tidak dipilih). Kita dapat mengubah status tersebut di RadioButton dengan menggunakan attribute **Android:checked**.

Dalam hal ini, jika kita ingin mengubah status RadioButton ke ON (dipilih), maka kita perlu mengatur **Android:checked="true"** di file layout XML.

RadioButton

- Saya seorang :
- Laki-laki
 - Perempuan

Saya setuju dengan syarat dan ketentuan

DAFTAR

View - ImageButton

Secara default, **ImageButton** terlihat sama dengan button lain, ia akan melakukan action ketika user mengetuk atau menyentuhnya. Tetapi, kita dapat menambahkan gambar khusus sebagai ganti teks.

Ini seperti **banner ads** yang kamu temui di website-website pada umumnya.

Kita dapat menambahkan gambar ke button dengan menggunakan attribute **<ImageButton>** dan menambahkan attribute Android:src dalam file layout XML atau dengan menggunakan method **setImageResource()**.



View - ProgressBar

ProgressBar adalah **komponen yang digunakan untuk menunjukkan progress dari action tertentu**. Misalnya, proses download atau upload file.

Secara default, **ProgressBar** akan ditampilkan sebagai roda yang berputar.

Jika kita ingin menampilkannya seperti horizontal bar, maka kita perlu mengubah properti style menjadi horizontal seperti:

`style="?Android:attr/progressBarStyleHorizontal".`



[GIF Image](#)



[GIF Image](#)

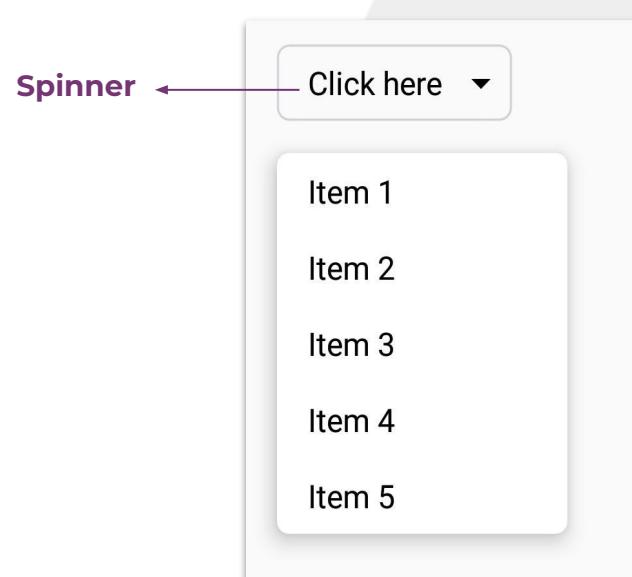
ProgressBar

ProgressBar
Style
Horizontal

View - Spinner

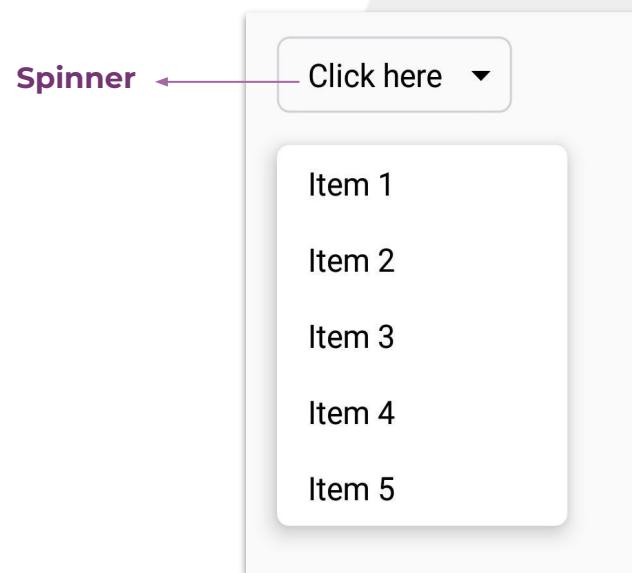
Spinner adalah komponen yang memungkinkan user memilih satu pilihan dari daftar pilihan. Spinner di Android akan sama dengan dropdown menu di web pada umumnya.

Secara default, spinner akan menunjukkan opsi yang dipilih saat ini dan dengan menggunakan **Adapter**.



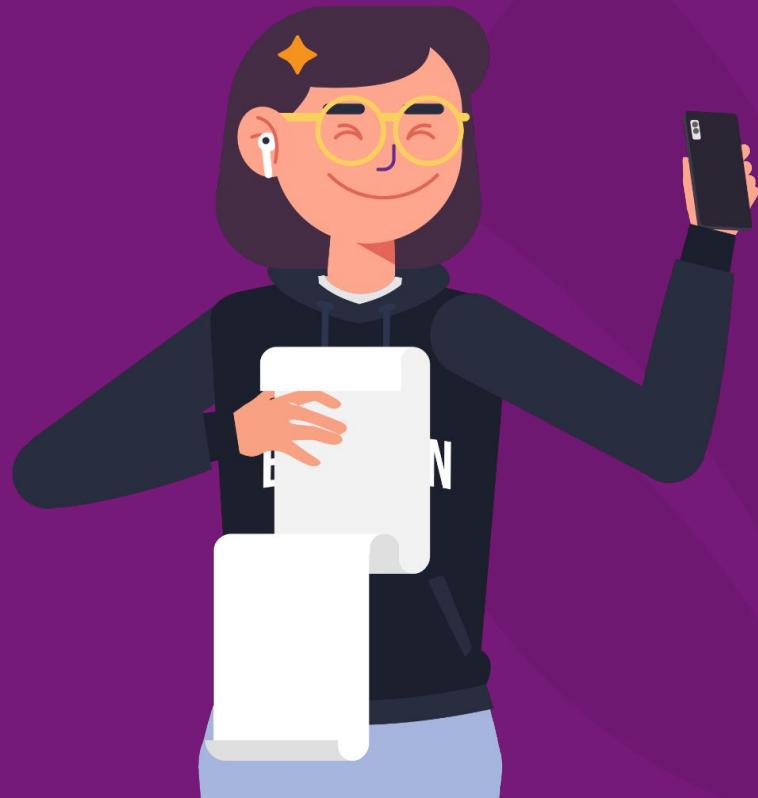
Nah, kita dapat mengisi komponen spinner pada daftar pilihan dengan mendefinisikan **ArrayAdapter** di file activity kita.

Adapter memuat data dari source seperti array atau database dan mengubah setiap item menjadi tampilan hasil dan yang ditempatkan ke dalam daftar.



Sipp markisip banget deh! Kita udah mempelajari beberapa View Android.

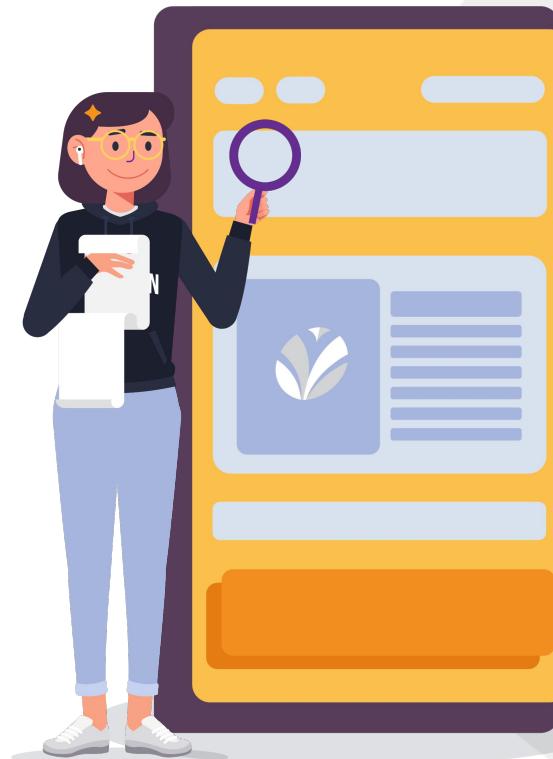
Sekarang kita pelajari “tempat” dari View bertengger, yaitu **ViewGroup**!



Apa sih, **ViewGroup** itu?

Gampangnya, **ViewGroup** adalah subclass dari view.

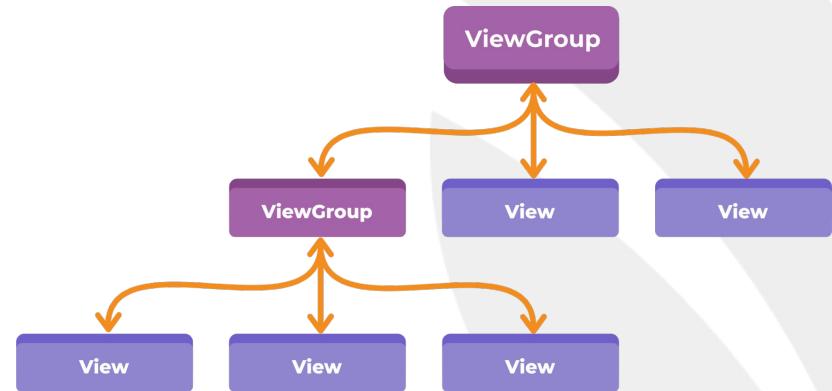
ViewGroup akan menyediakan wadah yang tidak terlihat untuk menampung View dan ViewGroup lainnya.



Dalam hierarki komponen View dan ViewGroup dapat juga digambarkan kayak gambar di samping.

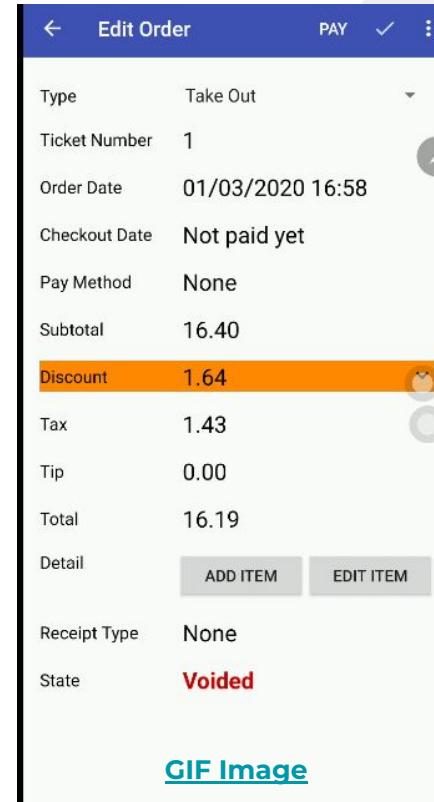
Jika kita artikan, berarti dalam sebuah ViewGroup nantinya dapat menampung dua buah komponen View, dimana satu komponen ViewGroup terdiri dari 3 buah komponen View.

Lalu, di dalam ViewGroup tersebut, bisa terdapat View ataupun ViewGroup lagi. Hal ini disebut dengan **Nested ViewGroup**.



Coba deh kamu lihat gambar di samping. Nah, Nested ViewGroup berjalan kayak gitu.

Kalau kamu klik data pada “Discount”, akan muncul data-data lainnya. Grouping semacam inilah yang disebut dengan **Nested ViewGroup**.



<ViewGroup/>

Kita udah belajar **ViewGroup** secara umum.

Kali ini kita kerucutin pembelajaran kita
menuju ke komponen-komponen **ViewGroup**
yang ada pada Android!



Kita recall dulu yuk, ViewGroup itu apa sih?

ViewGroup **mengatur semua komponen itu tertata rapi**, sehingga kita bisa membuat tampilan yang enak dipandang.

Contohnya, kayak gambar aplikasi di samping ini~



Apa aja tuh komponen-komponen ViewGroup?

Ada banyak komponen dalam ViewGroup yang bisa kita gunakan untuk mendesain. Selanjutnya, kita akan coba bahas beberapa komponennya satu per satu.

- Linear Layout
- Relative Layout
- Table Layout
- Frame Layout
- Constraint Layout
- WebView
- ListView
- GridView
- Scroll

Contoh ViewGroup



LinearLayout

RelativeLayout

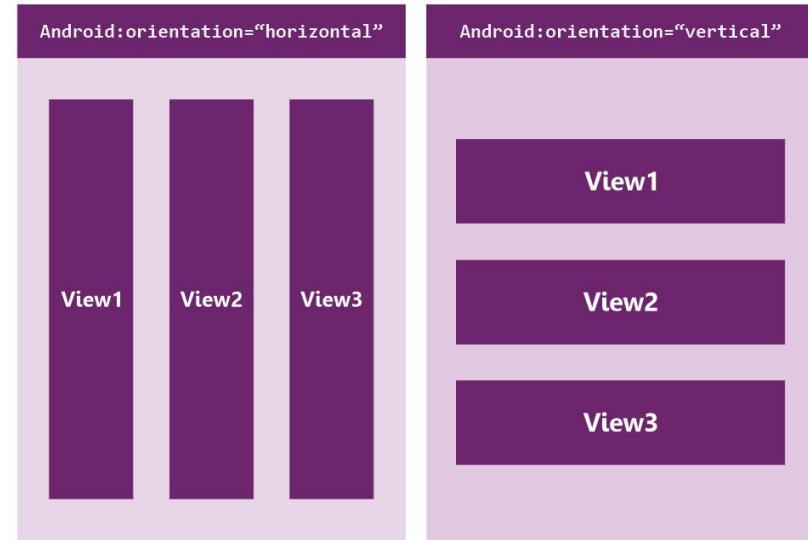
GridLayout

LinearLayout

LinearLayout adalah *layout* yang akan menempatkan komponen-komponen di dalamnya secara berkelompok.

Nah, kita dapat memilih apakah layout akan dimuat secara horizontal maupun vertikal.

Kita dapat menentukan arah LinearLayout menggunakan *attribute* **Android:orientation**.



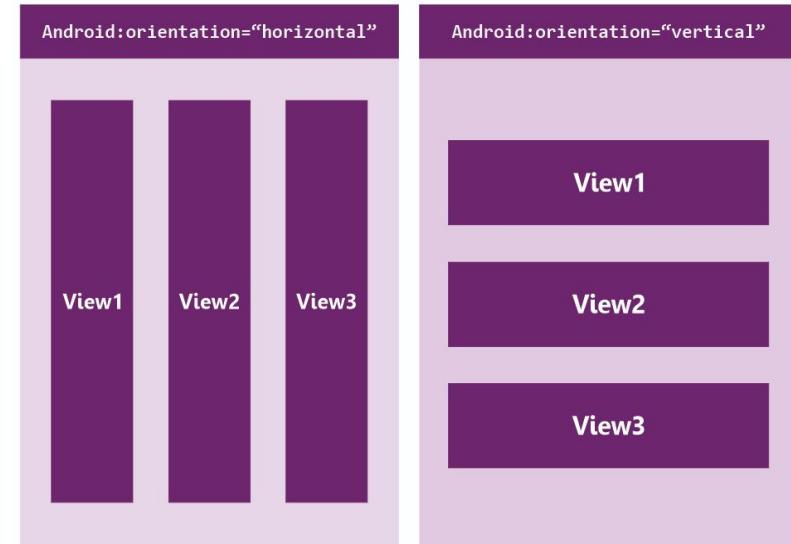
Jika kita ingin menampilkan arahnya menjadi vertikal, maka kita harus mendeklarasikan attribute-nya menjadi:

Android:orientation="vertical".

Kalau mau menampilkan arahnya jadi horizontal, tinggal deklarasikan atributnya jadi:

Android:orientation="horizontal"

Pada LinearLayout juga terdapat **attribute weight** yang berfungsi untuk membagi masing-masing porsi ukuran View, menyesuaikan space yang tersedia dalam layout.

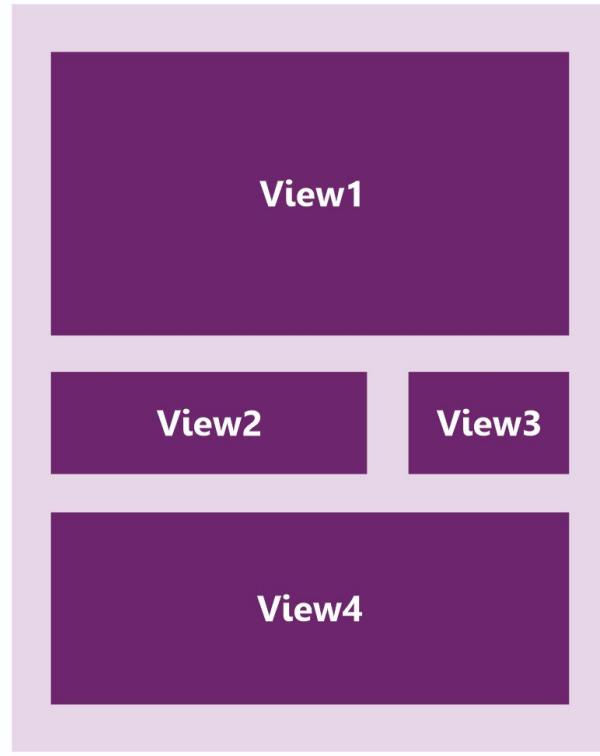


RelativeLayout

[RelativeLayout](#) digunakan untuk **menentukan posisi setiap komponen secara relatif terhadap komponen yang lain.**

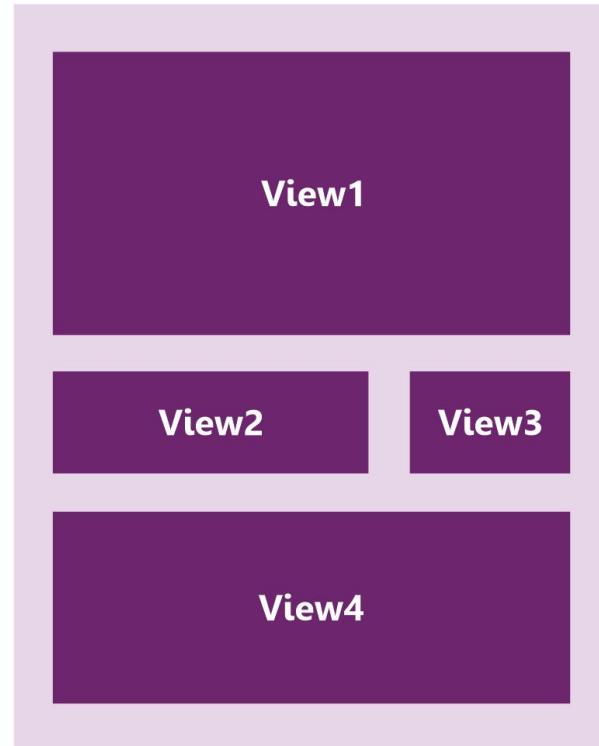
Posisi setiap View dapat ditentukan sebagai relatif terhadap komponen lain (seperti di sebelah kirinya atau di bawahnya tampilan lain).

Atau, ia juga bisa berada di posisi yang relatif terhadap RelativeLayout itu (seperti disejajarkan dengan bagian bawah, kiri, atau tengah).



Dengan menggunakan RelativeLayout, kita dapat **menghilangkan Nested ViewGroup seperti pada LinearLayout.**

Selain itu, kita juga bisa menjaga hierarki layout tetap flat sehingga dapat meningkatkan kinerja aplikasi.

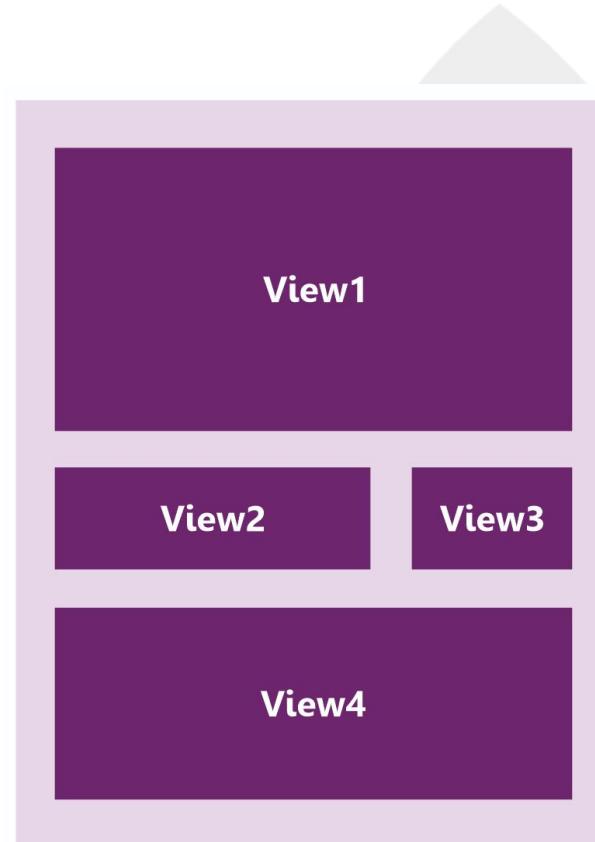


RelativeLayout Positioning Views

Tambahan untuk RelativeLayout, sangat memungkinkan child View menentukan posisi mereka relatif terhadap parent View atau satu sama lain (ditentukan oleh ID).

Dengan begitu, kita bisa menyelaraskan tuh dua elemen dengan batas kanan, atau membuat satu di bawah yang lain, berpusat di layar, berpusat di kiri, dan sebagainya.

Secara default, semua child View digambar di kiri atas layout, jadi kita harus menentukan posisi setiap View menggunakan berbagai properti layout yang tersedia dari RelativeLayout.



Beberapa layout properties yang ada pada views dalam RelativeLayout:

- **Android:layout_alignParentTop**
Jika "true", jadikan tepi atas view nempel dengan tepi atas parent.
- **Android:layout_centerVertical**
Jika "true", pusatkan view secara vertikal di dalam parent.
- **Android:layout_below**
Tepi atas View ini nempel dibawah View yang ditentukan dengan ID View.
- **Android:layout_toRightOf**
Tepi kiri View ini nempel di sebelah kanan View yang ditentukan dengan ID resource.

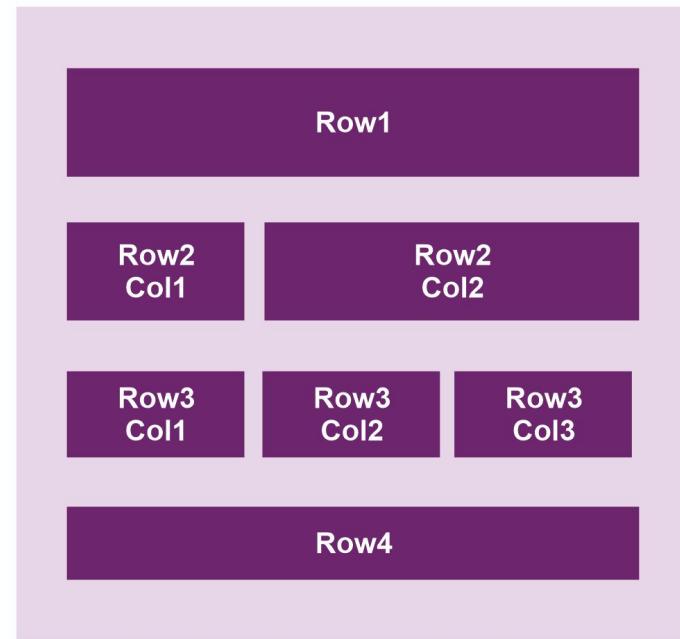
Ini hanya beberapa contoh. Semua attribute layout didokumentasikan di `RelativeLayout.LayoutParams`.

TableLayout

[TableLayout](#) merupakan layout yang digunakan untuk menampilkan komponen dalam baris dan kolom, seperti membuat tabel. Nah, kita bisa menggunakan komponen **<TableRow>** untuk membuat baris di tabel.

Setiap baris punya nol atau lebih cell dan setiap cell dapat menampung satu object View.

TableLayout tidak akan menampilkan garis pembatas ketika membuat baris, kolom, atau cell-nya.



FrameLayout

FrameLayout akan membuat komponen yang ada di dalamnya **menjadi menumpuk atau saling menutupi satu sama lain**. Komponen yang pertama pada layout ini akan menjadi alas bagi komponen-komponen di atasnya.

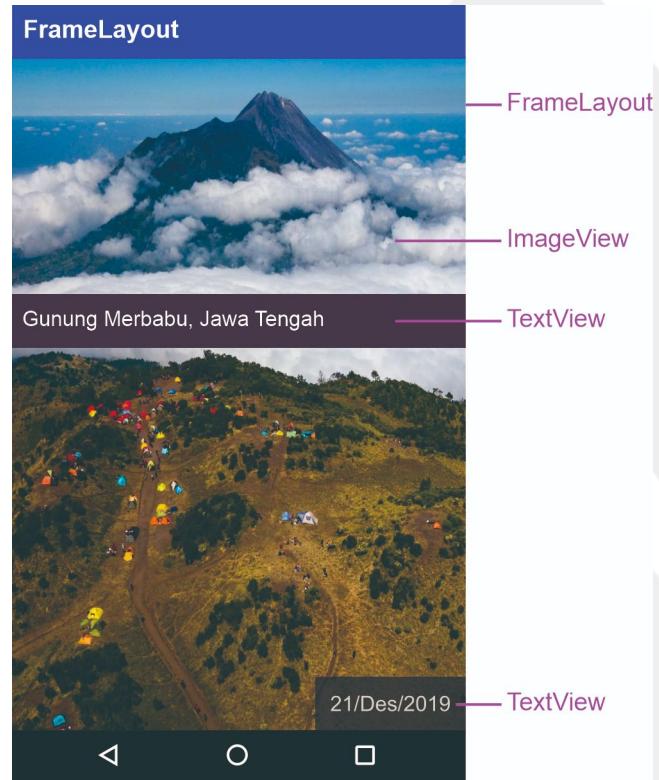
FrameLayout memiliki kemampuan untuk menjadi kontainer untuk fragment-fragment di dalam sebuah activity.



Jika kita ingin menerapkannya langsung pada Android, maka contohnya seperti pada gambar.

Pada contoh gambar, kita menggunakan FrameLayout sebagai ViewGroup teratas. Di dalam FrameLayout ini terdapat ImageView dan beberapa TextView.

Posisi ImageView dan TextView dalam FrameLayout dapat kita atur dengan menggunakan atribut **Android:layout_gravity**.

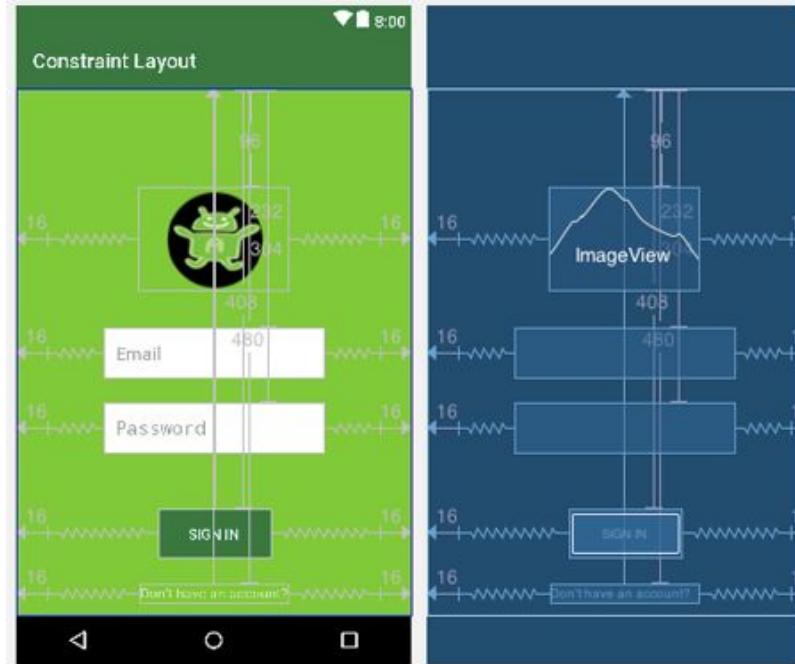


ConstraintLayout

ConstraintLayout memungkinkan kita **membuat layout yang kompleks dengan hierarki flat (tanpa Nested ViewGroup)**.

Mirip dengan dengan RelativeLayout bukan? Di mana semua View ditata sesuai dengan keterkaitannya dengan View, ViewGroup, ataupun pada parent-nya.

Tapi funfact-nya, ConstraintLayout lebih fleksibel daripada RelativeLayout.

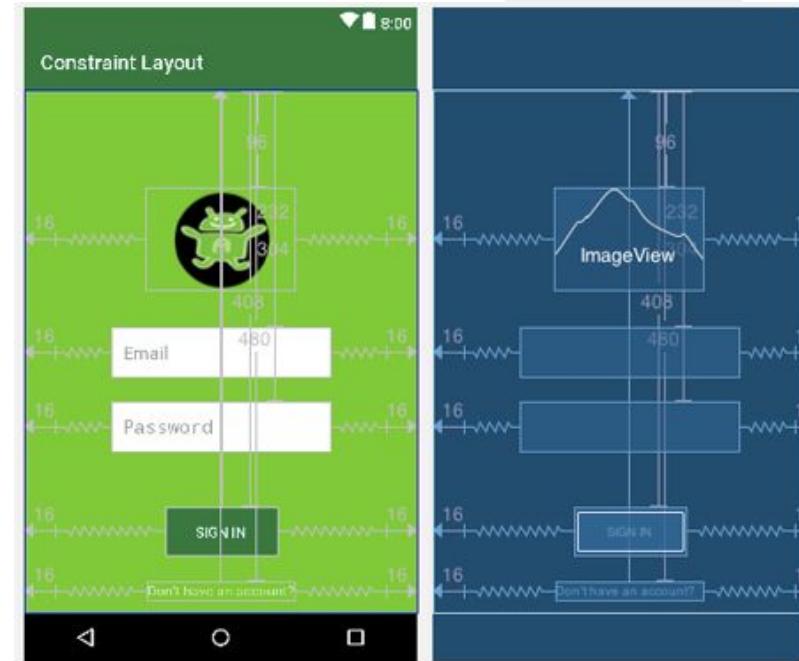


ConstraintLayout mempermudah kita untuk mendesain layout dengan cara **drag-and-drop** yang sudah terintegrasi dengan Android Studio.

ConstraintLayout tersedia dengan API library yang compatible di Android 2.3 (API level 9) dan seterusnya.

Jika kita membuat sebuah project baru dengan menggunakan Android Studio versi 3.4 ke atas, maka ConstraintLayout akan dijadikan default pada layout main. Lain halnya ketika kita sudah mengganti default tersebut.

Saat ini Android developer **direkomendasikan** untuk menggunakan ConstraintLayout.



Jika kamu lebih nyaman mendesain **ConstraintLayout** dengan tampilan Code XML, berikut ini adalah atribut-atribut yang bisa kamu manfaatkan:

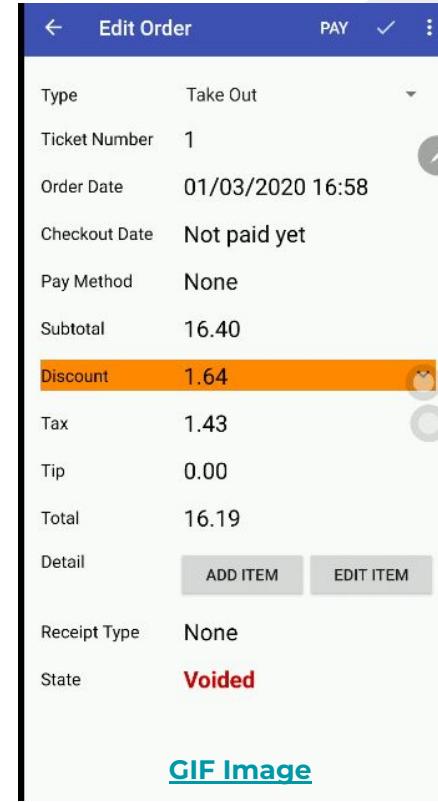
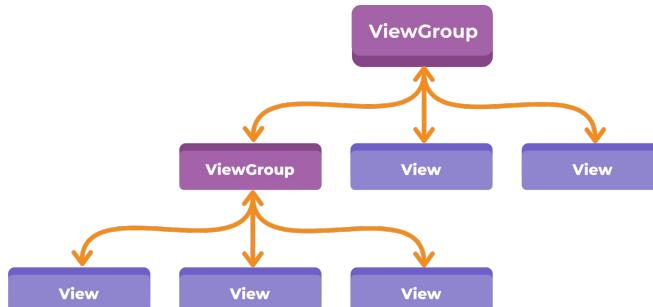
| Atribut | Fungsi |
|---------------------------------|---|
| layout_constraintTop_toTopOf | Menyejajarkan bagian atas View yang diinginkan ke atas yang lain. |
| layout_constraintTop_toBottomOf | Menyejajarkan bagian atas View yang diinginkan ke bagian bawah yang lain. |
| layout_constraintBottom_toTopOf | Menyejajarkan bagian bawah tampilan yang diinginkan ke atas yang lain. |
| layout_constraintBottom | Menyejajarkan bagian bawah tampilan yang diinginkan ke bawah yang lain. (Cont.) |

| Atribut | Fungsi |
|-----------------------------------|--|
| layout_constraintRight_toTopOf | Menyajarkan sebelah kanan tampilan yang diinginkan ke atas yang lain. |
| layout_constraintRight_toBottomOf | Menyajarkan sebelah kanan tampilan yang diinginkan ke bawah yang lain. |
| layout_constraintRight_toLeftOf | Menyajarkan kanan tampilan yang diinginkan ke kiri yang lain. |
| layout_constraintRight_toRightOf | Menyajarkan kanan tampilan yang diinginkan ke kanan yang lain. |

Sebelum melangkah ke slide berikutnya, kamu masing ingat kan dengan **Nested ViewGroup**?

Nested ViewGroup adalah View atau ViewGroup yang terdapat di dalam ViewGroup, kayak gambar di bawah.

Contohnya kayak gambar di samping. Kalau kamu klik data pada “Discount”, akan muncul data-data lainnya. Grouping semacam inilah yang disebut dengan **Nested ViewGroup**.



Nah, berkaitan dengan Nested ViewGroup tadi, ternyata ada yang lebih kompleks darinya, yaitu **deeply nested layouts**. Cek gambar di samping~

Deeply nested layouts ini gak disarankan untuk dipakai karena:

- Membutuhkan lebih banyak komputasi
- Views akan dihitung beberapa kali
- Bisa menyebabkan UI lamban dan tidak responsif

Jadi, **Gunakan ConstraintLayout** untuk menghindari beberapa isu ini!





Nah, beda lagi nih kalo kamu lebih nyaman desain **ConstraintLayout** menggunakan mode tampilan Design.

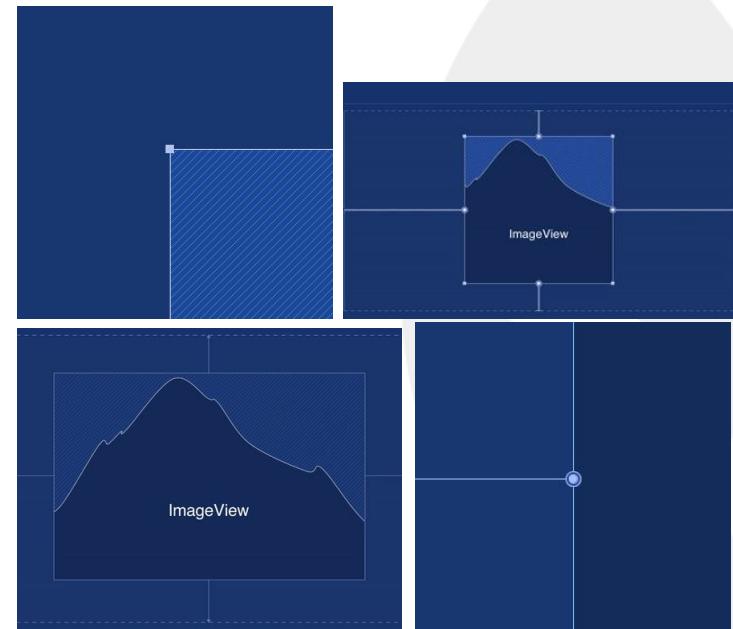
Kita perlu tahu komponen-komponen **ConstraintLayout** supaya membantu kita design dengan mode Design!

Apa aja tuh komponen-komponen ConstraintLayout?

Ada 4 komponen dalam ConstraintLayout yang bisa kita gunakan untuk mendesain, kayak gambar di samping:

1. **Resize Handle**
2. **Resize Anchor**
3. **Side Constraint Handle**
4. **Horizontal Bias**

Yuk kita bahas satu-persatu~

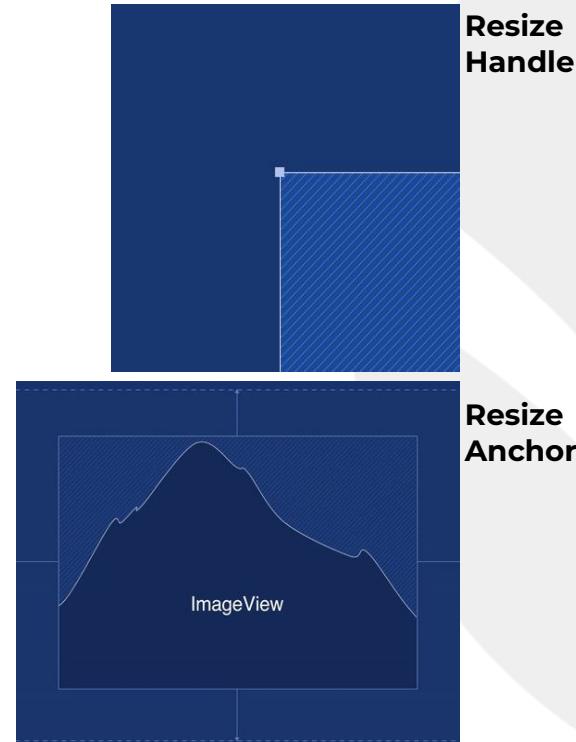


1. Resize Handle

Kita bisa mengubah ukuran View yang dipilih dan menetapkan Constraint untuk ukuran baru.

2. Resize Anchor

Mengubah ukuran View menggunakan resize anchor dalam layout editor akan secara otomatis menghitung ulang Constraint yang telah ditetapkan sebelumnya pada View tersebut.



3. Side Constraint Handle

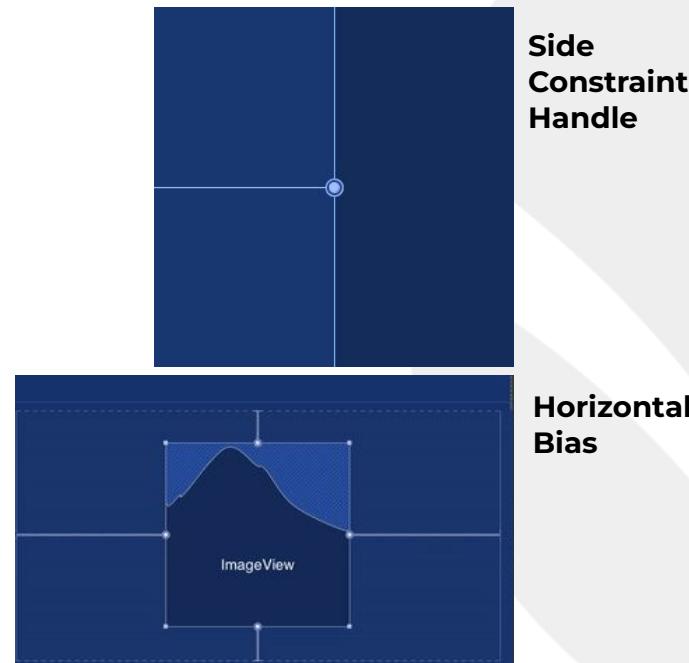
Digunakan untuk menentukan lokasi View dalam layout. Misalnya, anchor ini dapat digunakan untuk menentukan bahwa View yang dipilih selalu ditampilkan di sebelah kiri View lain dengan ukuran margin yang ditentukan.

4. Horizontal Bias

Ini memungkinkan kita untuk memposisikan View di sepanjang sumbu horizontal menggunakan nilai bias, posisi view akan relatif tergantung ukuran lebar layar.

Kita dapat mengatur horizontal bias menggunakan slider di layout editor atau menggunakan attribute di XML kita seperti:

```
app:layout_constraintHorizontal_bias="0,5"
```





Gimana menurutmu bahasan tentang komponen **ConstraintLayout**-nya?

Mudah-mudahan belajar kita sangat ramah bintang lima yaa. Setelah ini kita menuju ke pengaturan ukuran **ConstraintLayout** pada mode Design. Marii~

Constraint Widget di Layout Editor

Di Layout Editor, kamu dapat melihat Constraint Widget pada Attributes Window di sebelah kanan layar.

Kamu akan melihat 3 tipe simbol (terlihat di bawah) yang mengindikasikan tipe constraint mana yang telah diletakkan pada bagian start, end, top, atau bottom dari view yang dipilih.



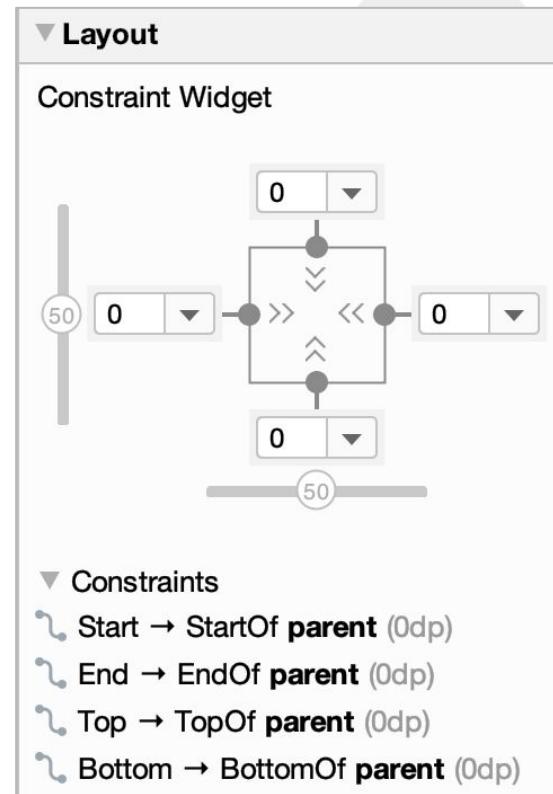
Fixed



Wrap content



Match constraints

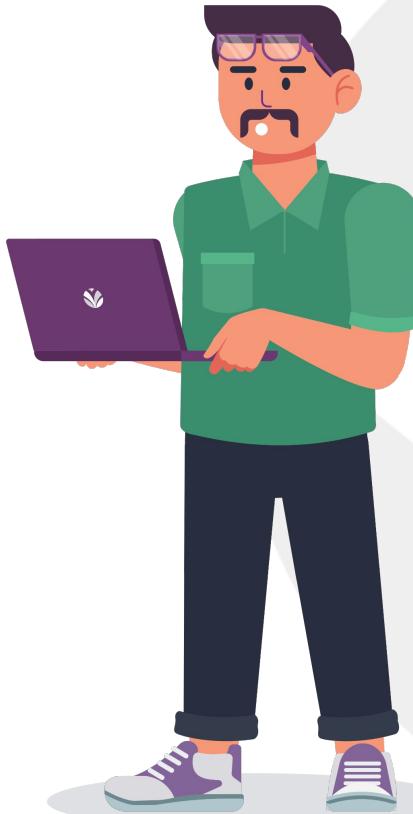


Gimana tuh pengaturan ukuran pada ConstraintLayout?

Pengaturan ukuran ConstraintLayout atau kita sebut **View Sizing**, dilakukan dengan beberapa cara:

- AnySize
- Warp Content
- Auto-connect
- Manual Connect

Yuk kita bahas satu-satu~

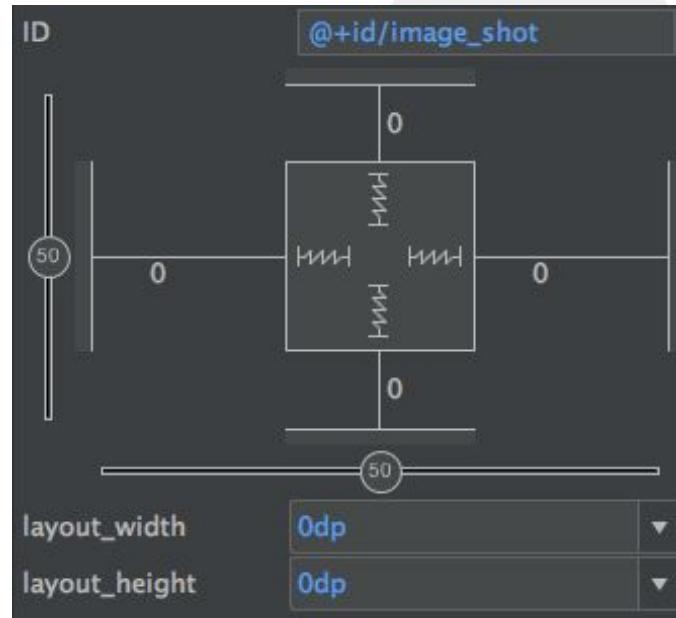


Any Size

Di dalam kotak kita dapat melihat garis berlekuk-lekuk yang menandakan bagaimana View akan mengubah ukuran dalam layout.

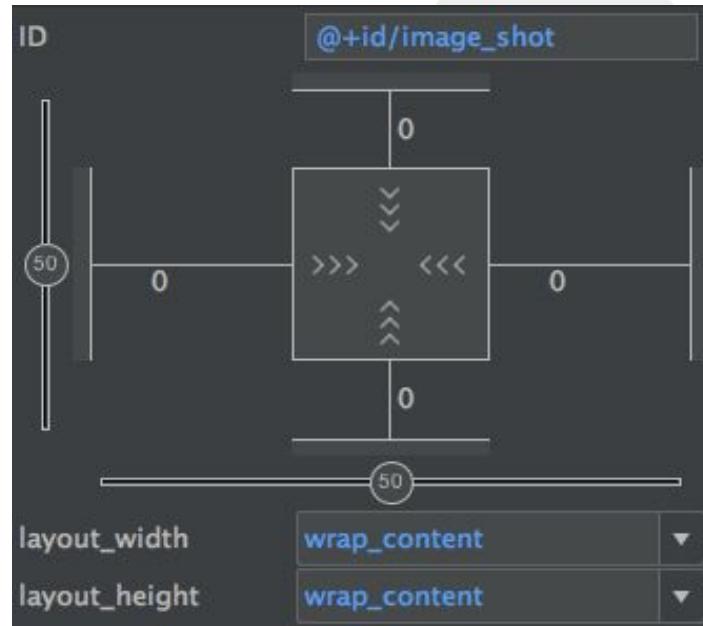
Baris-baris tersebut menyatakan bahwa ukuran View akan mematuhi Constraint, yang berarti akan menjadi **match_parent** dan menggunakan ruang yang tersedia.

Mengubah `layout_width` dan `layout_height` secara manual dengan nilai 0dp sama saja dengan menerapkan Any Size.



Wrap Content

Garis-garis ini sekarang menyatakan bahwa tinggi dan lebar View akan membungkus konten, yang berarti hanya akan sebesar yang diperlukan. Misalnya nih kalau image maka akan sesuai dengan ukuran gambar aslinya.



Auto-connect

Auto-connect memungkinkan kita untuk menempatkan View di dalam layout editor dan membuatnya secara otomatis menghitung dan mengatur Constraint untuk kita.



Now auto-connect is disabled

Manual Connect

Manual connect mirip kayak menonaktifkan Auto-connect. Manual connect dilakukan jika kita ingin secara manual mengatur Constraint dalam layout editor.

Psst! Dengan menggunakan ini, kita bisa lebih leluasa mengatur Constraint dan menghemat waktu kita menghapus Constraint otomatis yang tidak kita butuhkan!



WebView

WebView pada Android merupakan pengembangan dari kelas ViewGroup dan digunakan untuk **menampilkan konten halaman web HTML statis atau konten halaman web dengan URL** di aplikasi Android sebagai bagian dari layout activity.

Pada umumnya, WebView akan bertindak sebagai browser yang disisipkan untuk menampilkan konten halaman web dalam layout activity kita.

Namun fiturnya nggak terlalu lengkap kayak browser normal. Misalnya, nggak adanya address bar, navigation control, dll.

```
<html>
<body>

<!-- Web Page Content -->

</body>
</html>
```

WebView berguna untuk memasukkan dan menampilkan konten halaman web lain atau konten aplikasi di halaman yang diperlukan, seperti end-user agreements, dll.

Kita dapat mengimplementasikan WebView pada Android dengan menambahkan kode kayak gambar di samping pada XML:



```
<?xml version="1.0" encoding="utf-8"?>
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" />
```

ListView

ListView adalah ViewGroup yang berfungsi untuk **mengelompokkan beberapa item dan menampilkannya dalam list yang dapat di scroll secara vertikal.**

Item list secara otomatis dimasukkan ke list menggunakan Adapter yang menyediakan konten dari sumber data seperti array atau database.

Agar ListView dapat menampilkan daftar data, diperlukan sebuah **Adapter**.

Adapter menjembatani antara UI Component dan sumber data yang mengisi data ke UI Component.



Secara umum, ada **berbagai jenis adapter** yang tersedia di Android untuk mengambil data dari data sources yang berbeda, sehingga memungkinkan kita untuk mengisi data ke dalam tampilan adapter.

| Adapter | Deskripsi |
|---------------|---|
| ArrayAdapter | ArrayAdapter menggunakan Array atau List sebagai input. |
| CursorAdapter | Menerima instance kursor sebagai input. |
| SimpleAdapter | Menerima data statis yang ditentukan dalam resources. |
| BaseAdapter | Implementasi umum untuk ketiga jenis adapter dan dapat digunakan untuk ListView, Gridview atau Spinners berdasarkan persyaratan tertentu. |

GridView

GridView adalah ViewGroup yang digunakan untuk menampilkan item dalam bentuk kotak-kotak dua dimensi (baris dan kolom), kotak yang dapat di scroll dan item grid secara otomatis dimasukkan ke GridView layout menggunakan ListAdapter.

ListView dan GridView adalah subclass dari AdapterView dan dapat diisi dengan Adapter yang mengambil data dari source eksternal dan membuat View yang mewakili setiap entri data.

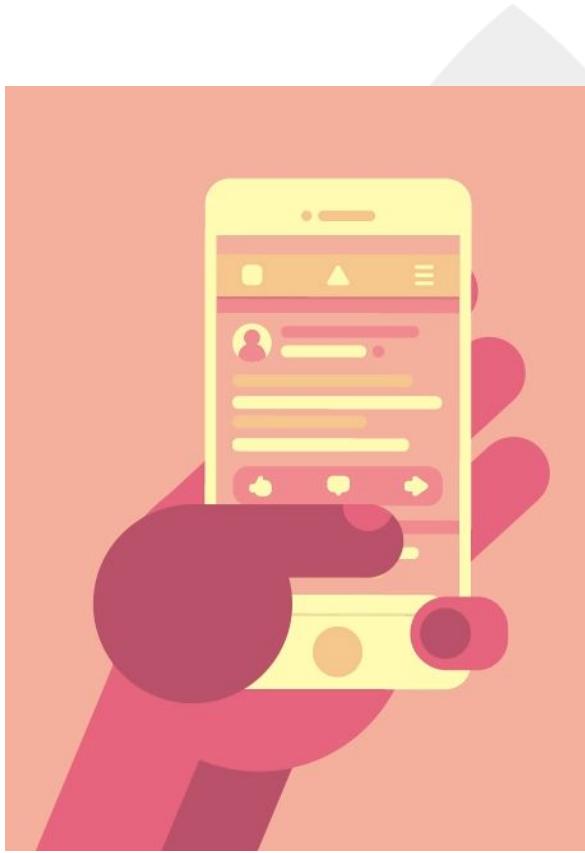


ScrollView

ScrollView adalah sejenis layout yang berguna untuk menambahkan scroll bar vertikal atau horizontal ke konten yang lebih besar dari ukuran sebenarnya seperti LinearLayout, RelativeLayout, FrameLayout, dll.

Secara umum, ScrollView berguna ketika kita memiliki konten yang nggak muat dengan layar aplikasi Android kita.

ScrollView akan memungkinkan pengguna untuk melihat konten lengkapnya dengan cara scrolling.

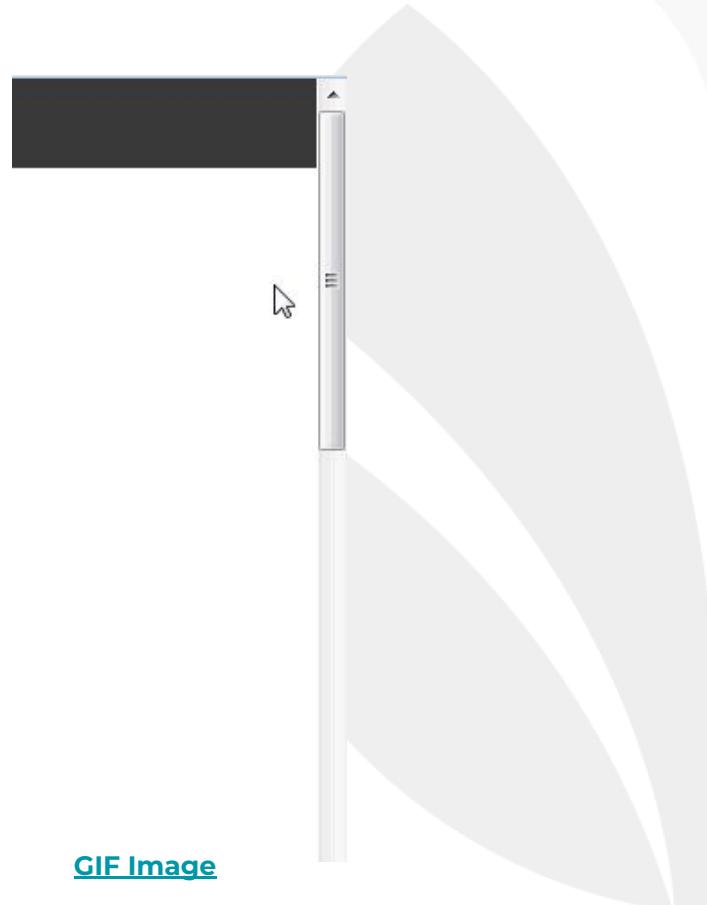


[GIF Image](#)

Android ScrollView **hanya dapat menampung satu ViewGroup.**

Jika kita ingin menambahkan beberapa View dalam ScrollView, maka kita perlu memasukkannya dalam ViewGroup seperti LinearLayout, RelativeLayout, FrameLayout, dll.

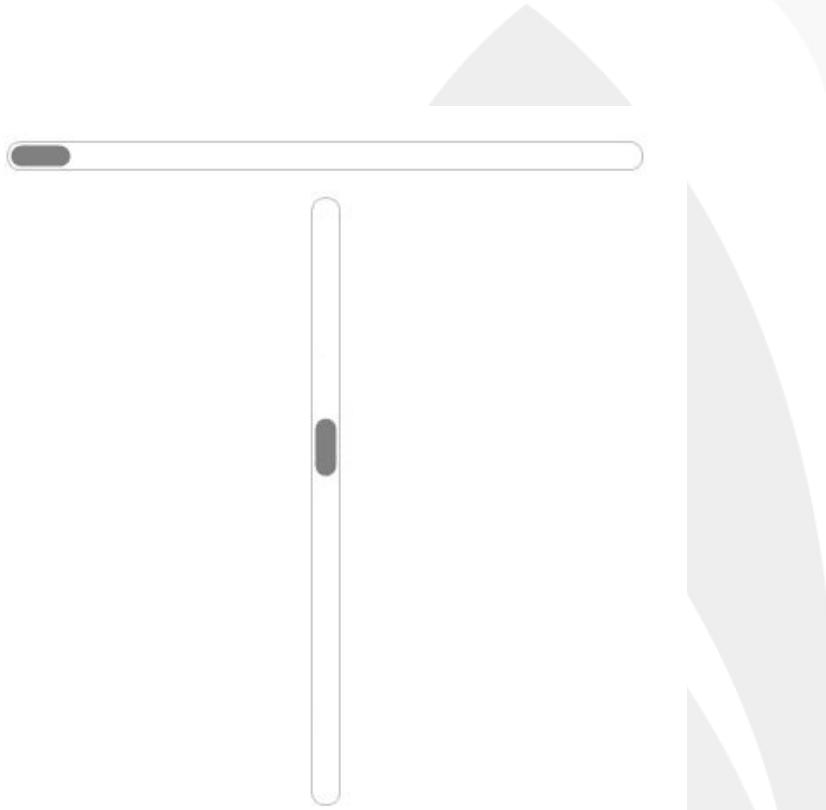
Untuk mengaktifkan scrolling pada aplikasi Android kita, ScrollView adalah pilihan terbaik tetapi kita tidak boleh menggunakan ScrollView bersama dengan ListView atau Gridview karena keduanya akan melakukan scrolling vertikal dengan caranya masing-masing.



[GIF Image](#)

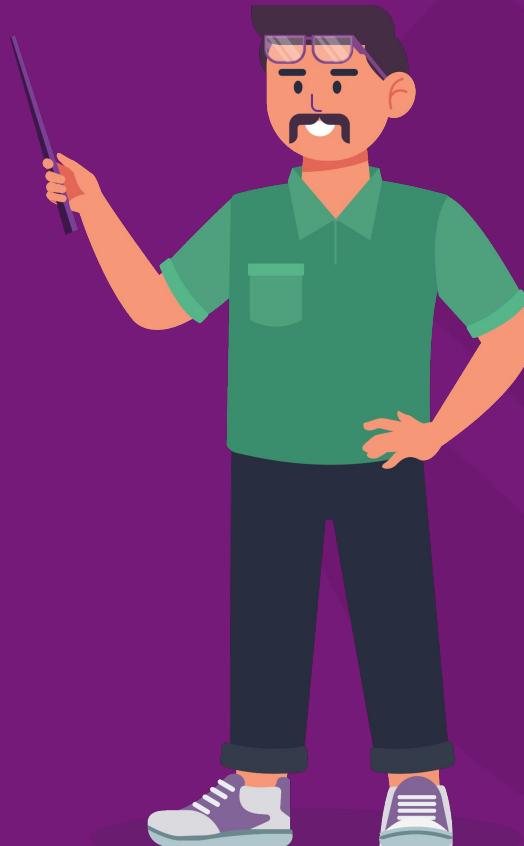
ScrollView hanya mendukung scrolling vertikal saja. Jika kita ingin menerapkan scrolling horizontal, maka kita perlu menggunakan komponen **HorizontalScrollView**.

Android ScrollView memiliki properti bernama **Android:fillViewport**, yang digunakan untuk menentukan apakah ScrollView harus merentangkan kontennya untuk mengisi view port atau tidak.



[GIF Image](#)

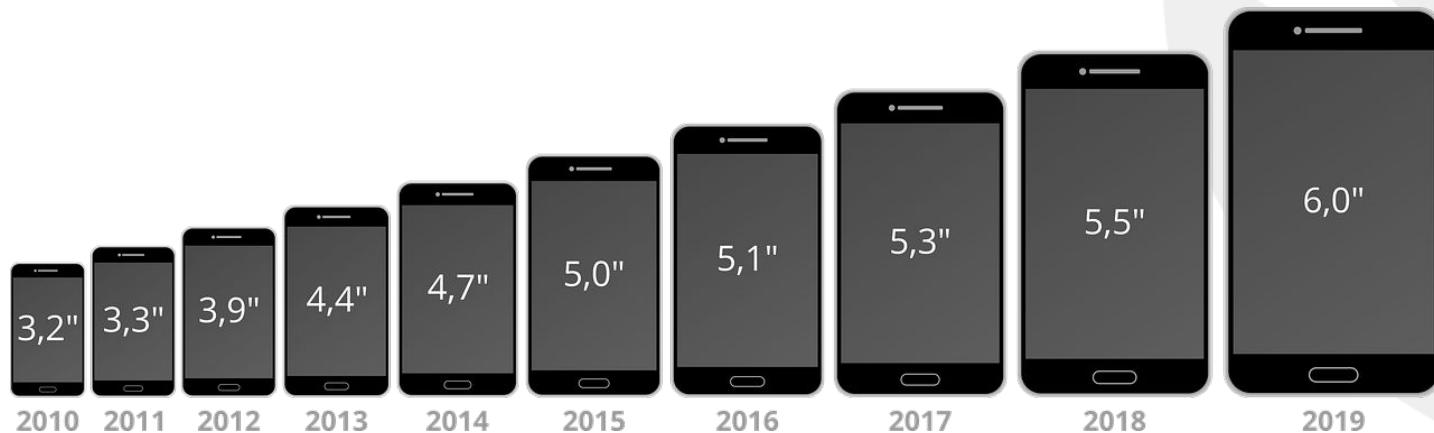
Ngomongin View dan ViewGroup yang udah kita bahas tadi, ada loh satuan dimensi untuk menentukan ukuran keduanya.



Dimensi pada Android

Device Android terkenal dengan keberagaman ukuran perangkat, layar, kepadatan pixelnya, hingga Operating Sistem-nya (OS). Karena hal tersebut, kita perlu mempertimbangkan mengenai penataan layout yang baik dan benar supaya aplikasi dapat berjalan dan tampil secara maksimal.

Kabar baiknya, Android developer sudah memperkirakan hal itu, sehingga Android memiliki satuan unit dimensi sendiri untuk membantu kita menentukan ukuran tinggi dan lebar dari sebuah komponen View atau ViewGroup.



Dimensi pada Android

- **px (pixels)**: jumlah piksel di layar. Tidak sering digunakan karena layar memiliki kerapatan piksel yang berbeda.
- **in (inci)**: ukuran fisik layar. Tidak sering digunakan karena ukuran layar yang berbeda memiliki resolusi yang berbeda. 1 inch = 2.54 centimeter.
- **mm (milimeter)**: ukuran fisik layar, mirip dengan cara kerja inci.

| Dimension | Description | Units / Physical Inch | Density Independent | Same Physical Size on Every Screen |
|-----------|----------------------------|-----------------------|---------------------|------------------------------------|
| px | Pixels | Varies | No | No |
| in | Inches | 1 | Yes | Yes |
| mm | Millimeters | 25.4 | Yes | Yes |
| pt | Points | 72 | Yes | Yes |
| dp | Density independent pixels | ~160 | Yes | No |
| sp | Scale independent pixels | ~160 | Yes | No |

- **pt** (1/72 inci [2,54 cm]): berdasarkan ukuran fisik layar.
- **dp or dip** (Density-independent Pixels): unit abstrak yang didasarkan pada kepadatan fisik layar.

Unit-unit ini relatif terhadap layar 160 dpi, jadi satu dp adalah satu piksel pada layar 160 dpi.

Ini umumnya paling sering digunakan ketika mengatur tata letak atau ukuran apa pun pada aplikasi Android. Rumusnya:

$$\text{dp} = (\text{width in pixels} * 160)$$

screen density

| Dimension | Description | Units / Physical Inch | Density Independent | Same Physical Size on Every Screen |
|-----------|----------------------------|-----------------------|---------------------|------------------------------------|
| px | Pixels | Varies | No | No |
| in | Inches | 1 | Yes | Yes |
| mm | Millimeters | 25.4 | Yes | Yes |
| pt | Points | 72 | Yes | Yes |
| dp | Density independent pixels | ~160 | Yes | No |
| sp | Scale independent pixels | ~160 | Yes | No |

- **sp** (Scale-independent Pixels): ini seperti unit dp, tetapi juga diskalakan oleh preferensi ukuran font pengguna.

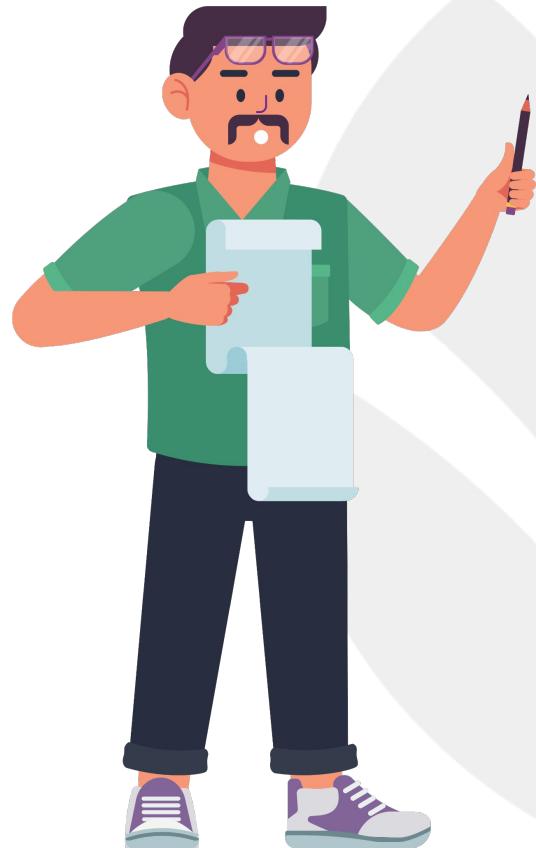
Biasanya digunakan ketika mengatur ukuran font di dalam aplikasi Android.

| Dimension | Description | Units / Physical Inch | Density Independent | Same Physical Size on Every Screen |
|-----------|----------------------------|-----------------------|---------------------|------------------------------------|
| px | Pixels | Varies | No | No |
| in | Inches | 1 | Yes | Yes |
| mm | Millimeters | 25.4 | Yes | Yes |
| pt | Points | 72 | Yes | Yes |
| dp | Density independent pixels | ~160 | Yes | No |
| sp | Scale independent pixels | ~160 | Yes | No |

Jika kita simpulkan, satuan dimensi di Android itu...

Bermanfaat untuk menghasilkan tampilan yang konsisten di perangkat Android.

Lalu kamu tahu nggak sih? Ada 2 jenis satuan dalam dimensi Android lho, yaitu **dip/dp** dan **sp**.



Apa beda keduanya?

- Satuan **sp** digunakan untuk **ukuran teks**. Perbedaannya dengan dp/dip adalah satuan sp Android akan men-scale ukuran teks sesuai dengan setting ukuran teks di device masing-masing (yang biasa dapat diakses melalui menu settings).
- Satuan **dp/dip** digunakan untuk **satuan dari nilai dimensi**. Misalnya, width (**Android:layout_width**) dan height (**Android:layout_height**) dari sebuah komponen View atau ViewGroup.



Contoh Penggunaan Dimensi dalam Android

Misalnya ada tiga buah tablet dengan resolusi tinggi.

- Tablet A memiliki resolusi layar 1920X2160px,
- Tablet B memiliki resolusi 2560X1440px, dan
- Tablet C memiliki resolusi 3840X2160px.

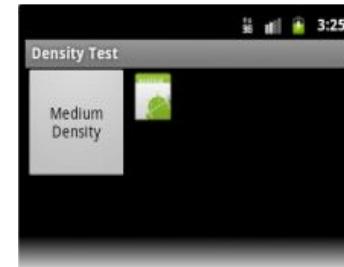
Kita masukkan sebuah button berukuran 500x500px dan terlihat normal pada tablet A.

Tetapi jika diperhatikan pada tablet B maupun C, maka button akan terlihat kecil bahkan terlalu kecil.

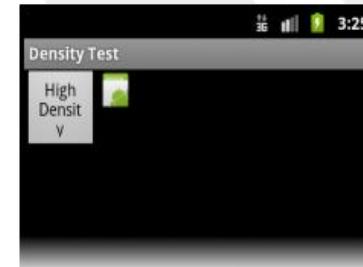
Button 500px X 500px



Tablet A



Tablet B

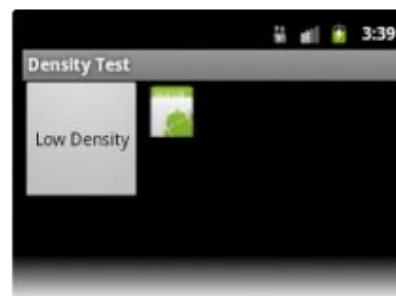


Tablet C

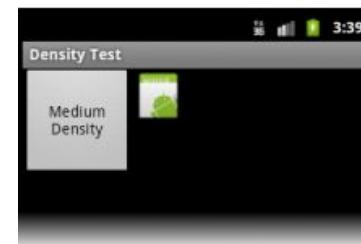
Beda lagi kalau kita menentukan ukuran button-nya menggunakan dp.

Bila kita menggunakan 500x500dp, maka button tersebut akan terlihat lebih baik pada beberapa device dengan ukuran yang berbeda.

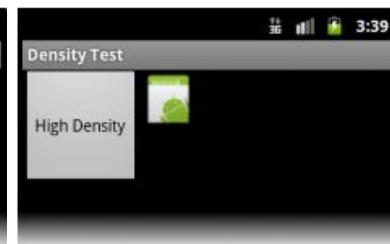
Button 500dp X 500dp



Tablet A



Tablet B



Tablet C

Contoh lainnya:

Gambar ini menjelaskan bahwa ukuran 200dp akan dikonversi pada device dengan density mdpi (device dengan density 160dpi/dots per inch) menjadi 200px dan menjadi 400px pada device dengan density xhdpi (device dengan density 420dpi).



| Density Bucket | Screen Density | Physical Size | Pixel Size |
|----------------|----------------|---------------|---------------------------------|
| ldpi | 120 dpi | 0.5 x 0.5 in | 0.5 in * 120 dpi = 60 x 60 px |
| mdpi | 160 dpi | 0.5 x 0.5 in | 0.5 in * 160 dpi = 80 x 80 px |
| hdpi | 240 dpi | 0.5 x 0.5 in | 0.5 in * 240 dpi = 120 x 120 px |
| xhdpi | 320 dpi | 0.5 x 0.5 in | 0.5 in * 320 dpi = 160 x 160 px |
| xxhdpi | 480 dpi | 0.5 x 0.5 in | 0.5 in * 480 dpi = 240 x 240 px |

Dengan hal tersebut, ukuran tampak sama dan konsisten secara fisik untuk berbagai device dengan ukuran layar yang berbeda.

Catatan: disarankan menggunakan berbagai macam ukuran image dengan beragam density. Hal ini akan mempermudah konversi image jika digunakan pada layar yang berbeda.



| Density Bucket | Screen Density | Physical Size | Pixel Size |
|----------------|----------------|---------------|---------------------------------|
| ldpi | 120 dpi | 0.5 x 0.5 in | 0.5 in * 120 dpi = 60 x 60 px |
| mdpi | 160 dpi | 0.5 x 0.5 in | 0.5 in * 160 dpi = 80 x 80 px |
| hdpi | 240 dpi | 0.5 x 0.5 in | 0.5 in * 240 dpi = 120 x 120 px |
| xhdpi | 320 dpi | 0.5 x 0.5 in | 0.5 in * 320 dpi = 160 x 160 px |
| xxhdpi | 480 dpi | 0.5 x 0.5 in | 0.5 in * 480 dpi = 240 x 240 px |

Sehingga, kesimpulannya...

Disarankan menggunakan berbagai macam ukuran image dengan beragam density karena hal tersebut akan mempermudah konversi image jika digunakan pada layar yang berbeda.



Terakhir nih, kita bahas slicing.

Dalam User Interface (UI) Design, slicing adalah proses memotong-motong komposisi layout dalam interface 2D (comp) menjadi beberapa file gambar (aset digital).

Ini merupakan bagian dari proses pengembangan client side untuk halaman dan/atau situs web.

Meski begitu, proses slicing juga digunakan dalam proses desain UI pada pengembangan software dan game.

Upcoming trip: Seattle
Jun 12-19 • 2 guests



Check In
12:00 PM, Jun 12

Check Out
11:00 AM, Jun 19

Weekly Mix
20 songs, 1hr 52 min





Udah banyak banget teknik layouting yang sudah kita bahas!

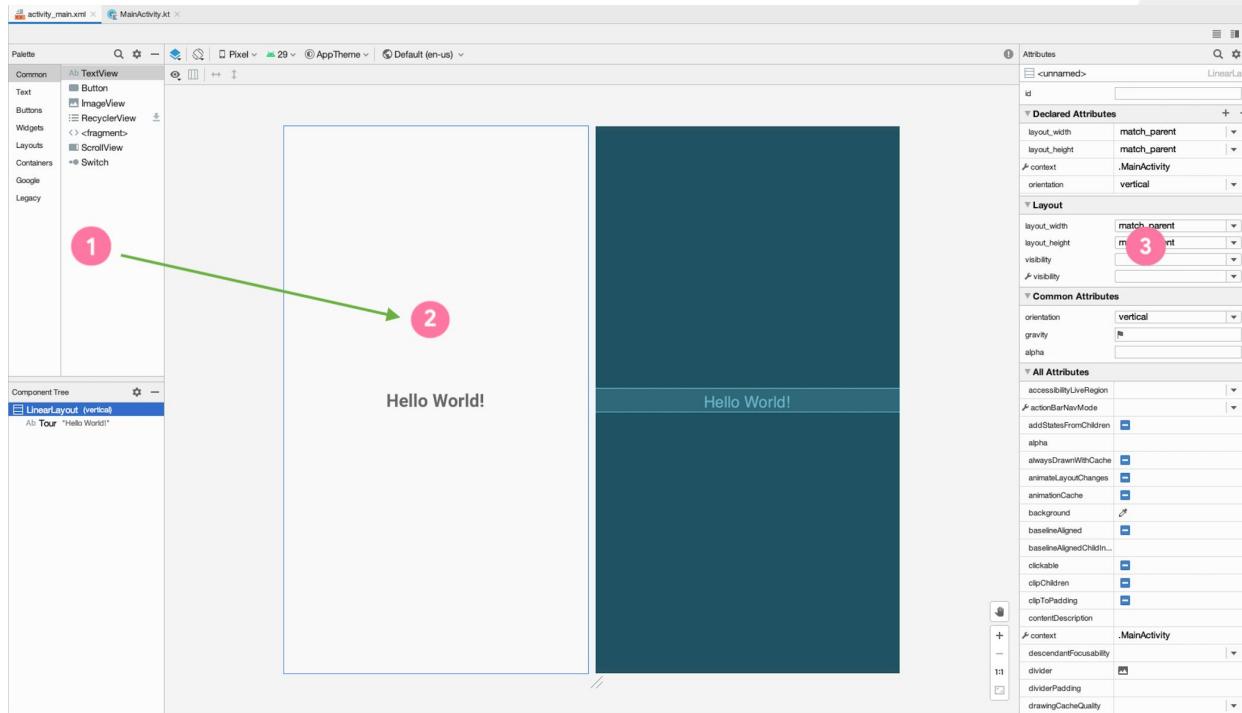
Sekarang, saat nya kita prak to the tek alias Praktek 🔥

Yuk, kita coba buat layout~

Menerapkan Android UI Layouting

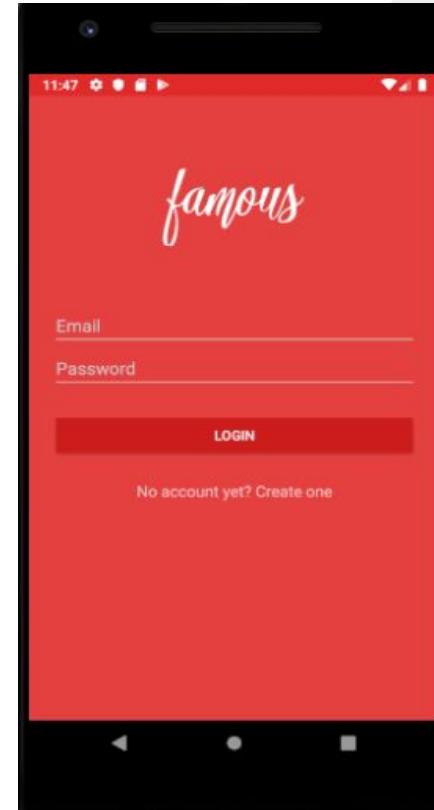
Untuk membuat layout, kamu bisa menggunakan Layout Editor yang ada di Android Studio. Caranya:

- Pilih komponen dari palette (drag n drop) (1) kemudian masukkan ke Design View (2).
- Lihat preview layout di Design View (2).
- Modifikasi atribut view tersebut pada Attributes Window (3).



Nah sebelum praktek membuat layout, kita membutuhkan **desain layout**.

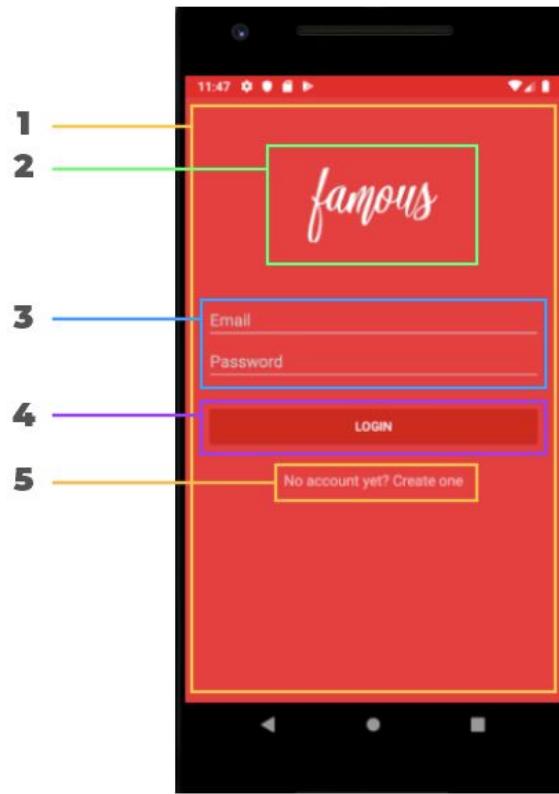
Sebagai contoh, kita akan menggunakan desain layout berikut.



Setelah mendapatkan desainnya, kita harus melakukan slicing ke XML untuk screen dari Androidnya.

Kita akan menggunakan beberapa **View**, antara lain:

1. **LinearLayout** untuk membuat semua View di dalam ViewGroup Horizontal.
2. **ImageView** untuk logo.
3. **EditText** untuk input email dan password.
4. **Button** untuk tombol login.
5. **TextView** untuk menampilkan label yang jika dipencet akan navigasi ke signup.





Setelah kita menentukan setiap komponen **View** dan **ViewGroup** yang akan digunakan dalam desain, kita bisa menuliskannya di **XML Layout**.

Menerapkan Android UI Layouting

```
● ● ●

<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
    </style>

    <style name="AppTheme.Dark" parent="Theme.AppCompat.NoActionBar">
        <item name="colorPrimary">@color/primary_dark</item>
        <item name="colorPrimaryDark">@color/primary</item>
        <item name="colorAccent">@color/accent</item>

        <item name="android:windowBackground">@color/primary</item>

        <item name="colorControlNormal">@color/iron</item>
        <item name="colorControlActivated">@color/white</item>
        <item name="colorControlHighlight">@color/white</item>
        <item name="android:textColorHint">@color/iron</item>

        <item name="colorButtonNormal">@color/primary_darker</item>
        <item name="android:colorButtonNormal ">@color/primary_darker</item>
    </style>

    <style name="AppTheme.Dark.Dialog" parent="Theme.AppCompat.Dialog">
        <item name="colorAccent">@color/white</item>
        <item name="android:textColorPrimary">@color/iron</item>
        <item name="android:background">@color/primary</item>
    </style>

</resources>
```

```
● ● ●

<!-- Password Label -->
<EditText android:id="@+id/input_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:hint="Password"/>

<Button
    android:id="@+id/btn_login"
    android:layout_width=" fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:layout_marginBottom="24dp"
    android:padding="12dp"
    android:text="Login"/>

<TextView android:id="@+id/link_signup"
    android:layout_width=" fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:text="No account yet? Create one"
    android:gravity="center"
    android:textSize="16dip"/>

</LinearLayout>
</ScrollView>
```

Sekarang, kita desain layout XML sedemikian rupa sehingga sangat mirip dengan desain yang kita dapatkan.

Menerapkan Android UI Layouting

Karena komponen View dalam LinearLayout itu cukup banyak, kita bisa menggunakan ScrollView sebagai Root.

Dengan begitu, pengguna bisa melakukan scroll untuk melihat semua komponen meskipun ukuran layar smartphone mereka kecil.

```
<resources>

<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
</style>

<style name="AppTheme.Dark" parent="Theme.AppCompat.NoActionBar">
    <item name="colorPrimary">@color/primary_dark</item>
    <item name="colorPrimaryDark">@color/primary</item>
    <item name="colorAccent">@color/accent</item>

    <item name="android:windowBackground">@color/primary</item>

    <item name="colorControlNormal">@color/iron</item>
    <item name="colorControlActivated">@color/white</item>
    <item name="colorControlHighlight">@color/white</item>
    <item name="android:textColorHint">@color/iron</item>

    <item name="colorButtonNormal">@color/primary_darker</item>
    <item name="android:colorButtonNormal">@color/primary_darker</item>
</style>

<style name="AppTheme.Dark.Dialog" parent="Theme.AppCompat.Dialog">
    <item name="colorAccent">@color/white</item>
    <item name="android:textColorPrimary">@color/iron</item>
    <item name="android:background">@color/primary</item>
</style>

</resources>
```

```
<!-- Password Label -->
<EditText android:id="@+id/input_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:hint="Password"/>

<Button
    android:id="@+id/btn_login"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:layout_marginBottom="24dp"
    android:padding="12dp"
    android:text="Login"/>

<TextView android:id="@+id/link_signup"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:text="No account yet? Create one"
    android:gravity="center"
    android:textSize="16dip"/>

</LinearLayout>
</ScrollView>
```

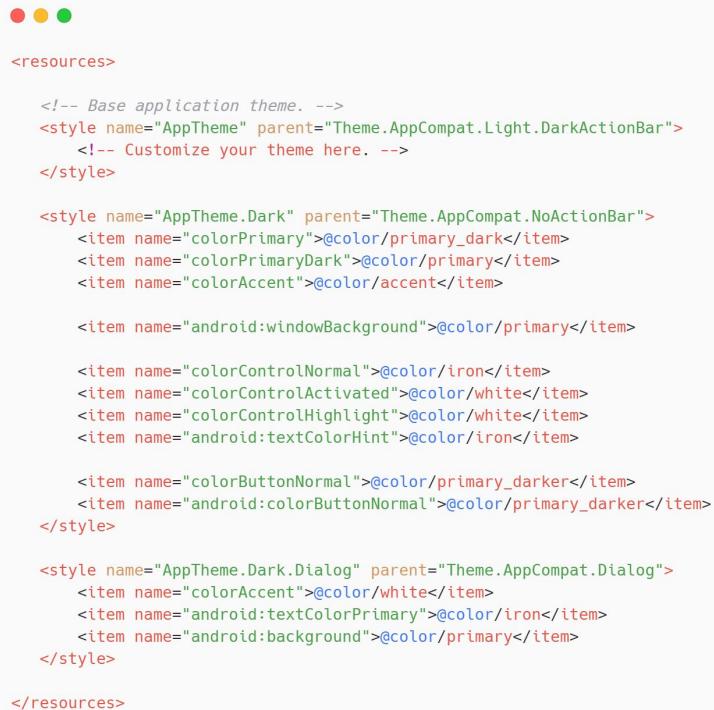


Untuk membuat theme yang berisikan variabel warna dalam app, kita mendefinisikannya pada file **/res/values/styles.xml**

Nah, sekarang kita siapkan style yang akan kita terapkan.

Kita menyiapkan style di styles.xml. Pada styles.xml ini, kita bisa mendefinisikan style tampilan untuk aplikasi kita secara keseluruhan maupun untuk komponen-komponen tertentu.

Contohnya, kita bisa customize tampilan dialog yang diadaptasi dari tema bawaan Android. Coba perhatikan tag style yang terakhir!

A screenshot of an Android application window. At the top, there are three colored window control buttons (red, yellow, green). Below them is the application's title bar. The main content area displays a dark-themed dialog box. Inside the dialog, there is white text on a purple background. The overall theme of the application appears to be a dark mode variation of the standard Android light theme.

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
    </style>

    <style name="AppTheme.Dark" parent="Theme.AppCompat.NoActionBar">
        <item name="colorPrimary">@color/primary_dark</item>
        <item name="colorPrimaryDark">@color/primary</item>
        <item name="colorAccent">@color/accent</item>

        <item name="android:windowBackground">@color/primary</item>

        <item name="colorControlNormal">@color/iron</item>
        <item name="colorControlActivated">@color/white</item>
        <item name="colorControlHighlight">@color/white</item>
        <item name="android:textColorHint">@color/iron</item>

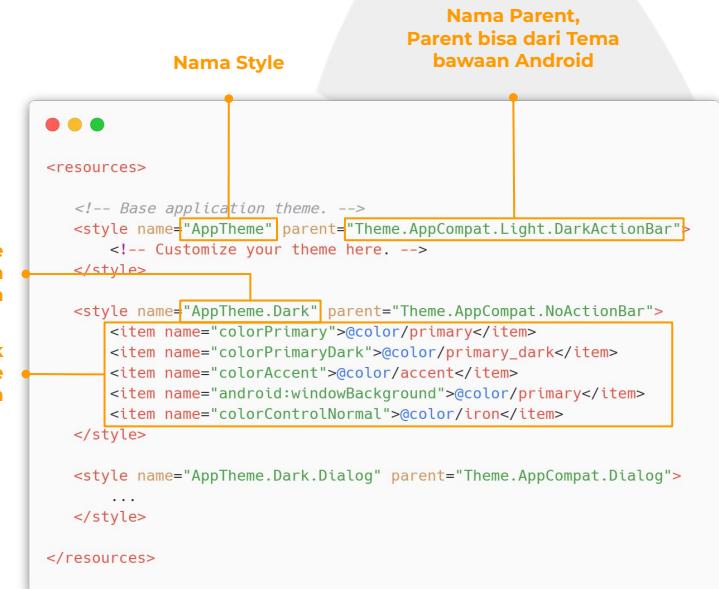
        <item name="colorButtonNormal">@color/primary_darker</item>
        <item name="android:colorButtonNormal">@color/primary_darker</item>
    </style>

    <style name="AppTheme.Dark.Dialog" parent="Theme.AppCompat.Dialog">
        <item name="colorAccent">@color/white</item>
        <item name="android:textColorPrimary">@color/iron</item>
        <item name="android:background">@color/primary</item>
    </style>

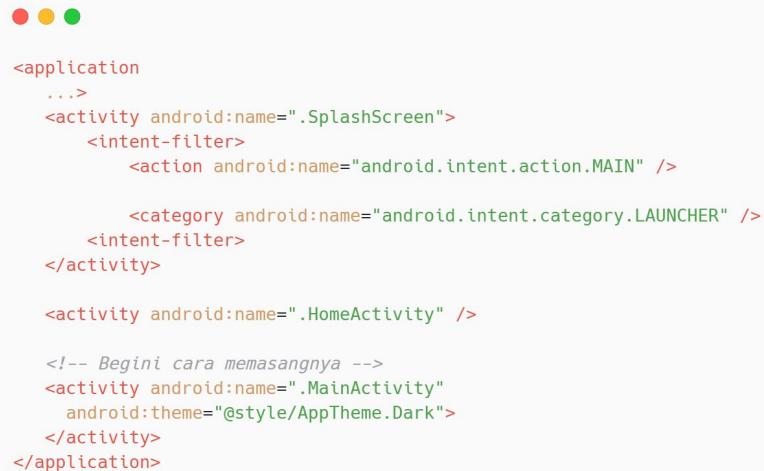
</resources>
```

Supaya makin emezing, kita bedah yuk kode di styles.xml!

- **Atribut name:**
Digunakan untuk penamaan style yang kita buat. Nama setiap style nggak boleh sama yaa!
- **Atribut Parent:**
Digunakan untuk ‘menurunkan’ style dari style yang sudah ada/disiapkan sebelumnya.
- **Tag Item:**
Digunakan untuk mengubah style bawaan dari style parent. Atribut name dalam tag item ini bisa berbeda-beda ya tergantung parent yang digunakan.



Untuk menggunakan theme dari nilai values terhadap sebuah screen (yaitu main activity), kita dapat memanggil dari file **AndroidManifest.xml**



```
<application
    ...
    <activity android:name=".SplashScreen">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <activity android:name=".HomeActivity" />

    <!-- Begini cara memasangnya -->
    <activity android:name=".MainActivity"
        android:theme="@style/AppTheme.Dark">
    </activity>
</application>
```

A screenshot of an AndroidManifest.xml file in a code editor. The code defines an application with three activities: SplashScreen, HomeActivity, and MainActivity. The MainActivity is set as the main launcher activity. The code uses XML tags like <application>, <activity>, <intent-filter>, <action>, <category>, and <!--> to define the application's components and their behavior.

**Biar pemahamanmu makin komplit,
yuk ikut challenge ini~**

Buatlah aplikasi kartu ucapan ulang tahun seperti contoh gambar di samping. Desainlah halamannya semenarik mungkin!

Jika sudah, simpan project kamu ke dalam repository Gitlab. Hasilnya nanti kita bahas di sesi berikutnya yah~





Nah, menurut kamu, bagaimana nantinya penerapan materi Topic 4 ini untuk individual challenge di Chapter 2 ini?

Setelah mempelajari topic ini, kamu bisa mulai mengerjakan challenge di bagian:

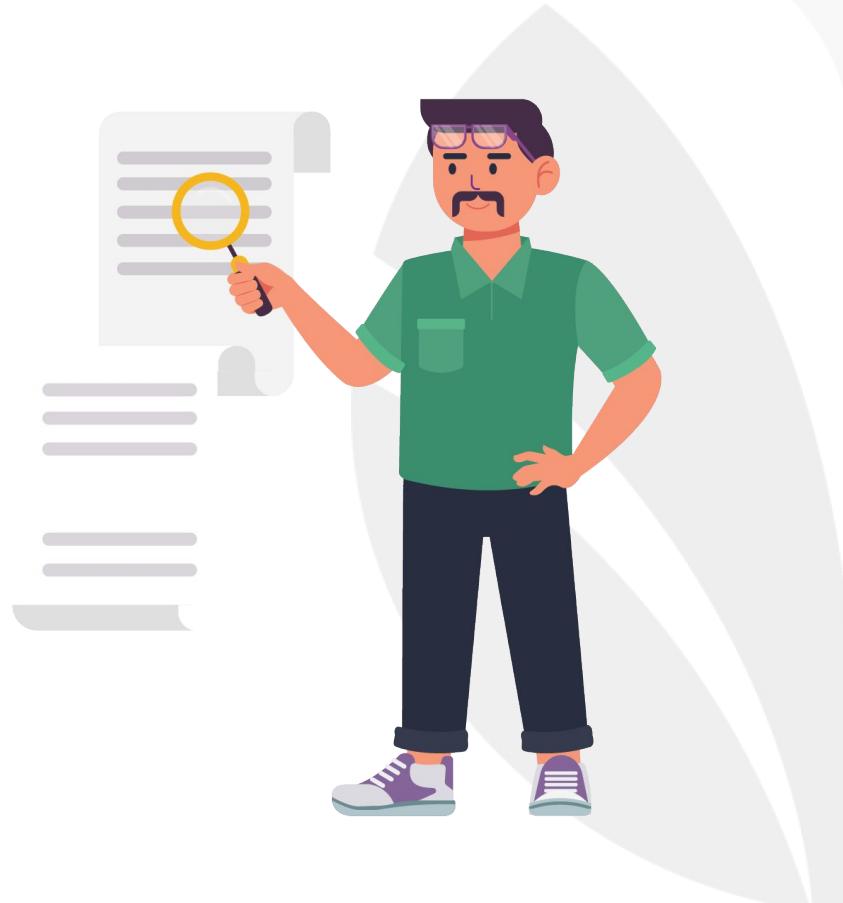
- Menambahkan **TextView**, **EditTextView**, **RadioButton** dan lainnya di layout.
- Mengatur layout menggunakan **ConstraintLayout**.

Jika masih ada pertanyaan, jangan ragu untuk menghubungi facilitator ya!



Referensi dan bacaan lebih lanjut~

1. [Android UI Layouts Tutorial with Examples](#)
2. [Build a Responsive UI with ConstraintLayout | Android Developers](#)
3. [Pixel Density](#)
4. [Tata letak | Developer Android](#)
5. [Relative Layout | Android Developers](#)
6. [Linear Layout | Android Developers](#)
7. [ScrollView | Android Developers](#)
8. [Slices | Android Developers](#)
9. [Device Metrics](#)
10. [Spacing](#)



**Yuhuuu! Kamu udah berhasil
menamatkan Chapter 2 Topic 4 🎉**

**Eitsss, tapi materinya masih berlanjut nih.
Sampai jumpa di Chapter 4 Topic 5 ya,
learners!**

