# 1.2: Layout, Tampilan, dan Sumber Daya

**Materi:**

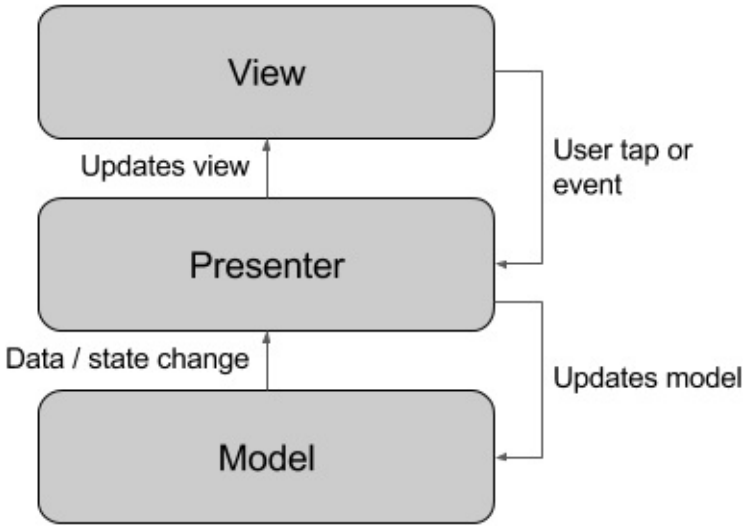
- Pola model-view-presenter
- Tampilan
- File sumber daya
- Merespons klik tampilan
- Praktik terkait
- Ketahui selengkapnya

Bab ini menjelaskan layout antarmuka pengguna di layar dan sumber daya lain yang Anda buat untuk aplikasi, dan kode yang akan digunakan untuk merespons ketukan pengguna pada elemen antarmuka pengguna.

## Pola model-view-presenter

Menautkan aktivitas ke sumber daya layout adalah contoh dari bagian pola arsitektur *model-view-presenter* (MVP). Pola MVP adalah cara yang sudah umum digunakan untuk mengelompokkan fungsi aplikasi:

- **Tampilan.** Tampilan adalah elemen antarmuka pengguna yang menampilkan data dan respons terhadap tindakan pengguna. Setiap elemen layar adalah tampilan. Sistem Android menyediakan berbagai jenis tampilan.
- **Presenter.** Presenter menghubungkan tampilan aplikasi ke model. Presenter menyediakan tampilan dengan data sebagaimana ditetapkan oleh model, dan juga menyediakan masukan pengguna dari tampilan kepada model.
- **Model.** Model menetapkan struktur data aplikasi dan kode untuk mengakses dan memanipulasi data. Beberapa aplikasi yang Anda buat dalam pelajaran bisa digunakan bersama model untuk mengakses data. Aplikasi Hello Toast tidak menggunakan model data, namun Anda bisa memikirkan logikanya — menampilkan pesan, dan meningkatkan penghitung ketukan — sebagai model.




## Tampilan

UI terdiri dari hierarki objek yang disebut *tampilan*, setiap elemen layar adalah *tampilan*. Kelas View menyatakan blok pembangunan dasar untuk semua komponen UI, dan kelas dasar untuk kelas yang menyediakan komponen UI interaktif seperti tombol, kotak centang, dan bidang entri teks.

Tampilan memiliki lokasi, yang dinyatakan sebagai pasangan koordinat kiri dan atas, dan dua dimensi, yang dinyatakan sebagai lebar dan tinggi. Unit untuk lokasi dan dimensi adalah piksel yang tidak tergantung perangkat (dp).

Sistem Android menyediakan ratusan tampilan yang telah didefinisikan sebelumnya, termasuk yang menampilkan:

- Teks (TextView)
- Bidang untuk memasukkan dan mengedit teks (EditText)
- Pengguna tombol bisa mengetuk (Button) dan komponen interaktif lainnya
- Teks yang bisa digulir (ScrollView) dan item yang bisa digulir (RecyclerView)
- Gambar (ImageView)

Anda bisa mendefinisikan tampilan untuk muncul di layar dan merespons ketukan pengguna. Tampilan juga bisa didefinisikan untuk menerima masukan teks, atau tidak terlihat hingga diperlukan.

Anda bisa menetapkan tampilan di file sumber daya layout XML. Sumber daya layout ditulis dalam XML dan dicantumkan dalam folder **layout** di folder **res** dalam Project: Tampilan Android.

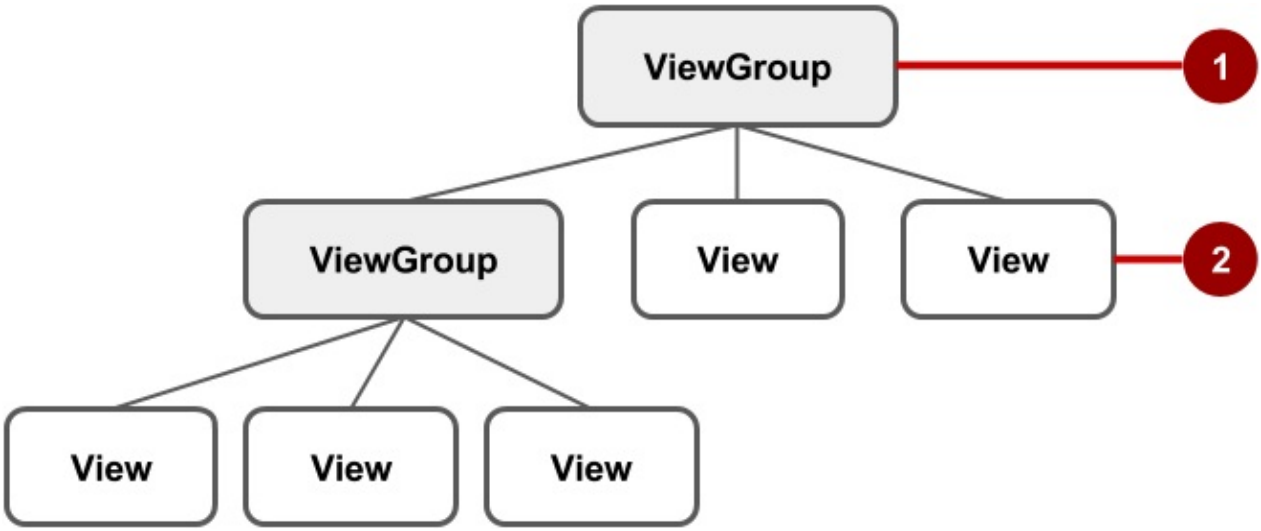
## Grup tampilan

Tampilan bisa dikelompokkan bersama di dalam *grup tampilan* (ViewGroup), yang berfungsi sebagai kontainer tampilan. Hubungannya adalah induk-anak, dalam hal ini *induk* adalah grup tampilan, dan *anak* adalah tampilan atau grup tampilan dalam grup. Berikut ini adalah grup tampilan yang umum:

- ScrollView: Grup yang berisi satu tampilan anak lainnya dan memungkinkan pengguliran tampilan anak.
- RecyclerView: Grup yang berisi daftar tampilan atau grup tampilan lainnya dan memungkinkan penggulirannya dengan menambahkan dan membuang tampilan secara dinamis dari layar.

## Grup tampilan layout

Tampilan untuk layar dikelola dalam hierarki. Di *akar* hierarki ini adalah ViewGroup yang berisi layout keseluruhan layar. Layar anak grup tampilan bisa menjadi tampilan atau grup tampilan lain seperti yang ditampilkan dalam gambar berikut.



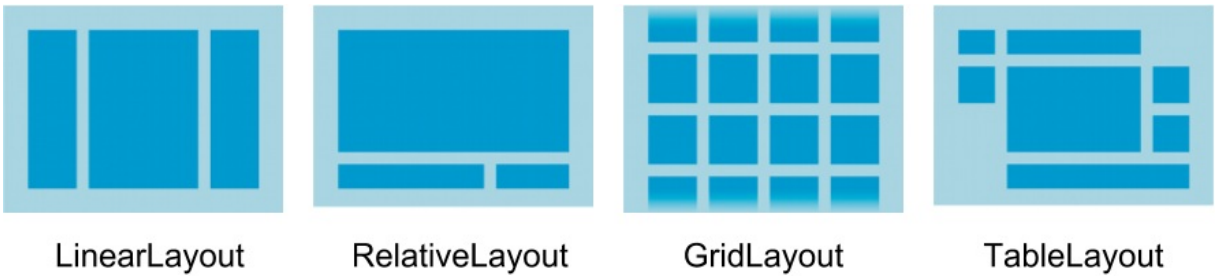

Dalam gambar di atas:

1. Grup tampilan *akar*.
2. Rangkaian tampilan anak dan grup tampilan pertama yang induknya adalah akar.

Beberapa grup tampilan ditandai sebagai *layout* karena grup tampilan tersebut mengelola tampilan anak dalam cara khusus dan umumnya digunakan sebagai grup tampilan akar. Beberapa contoh layout adalah:

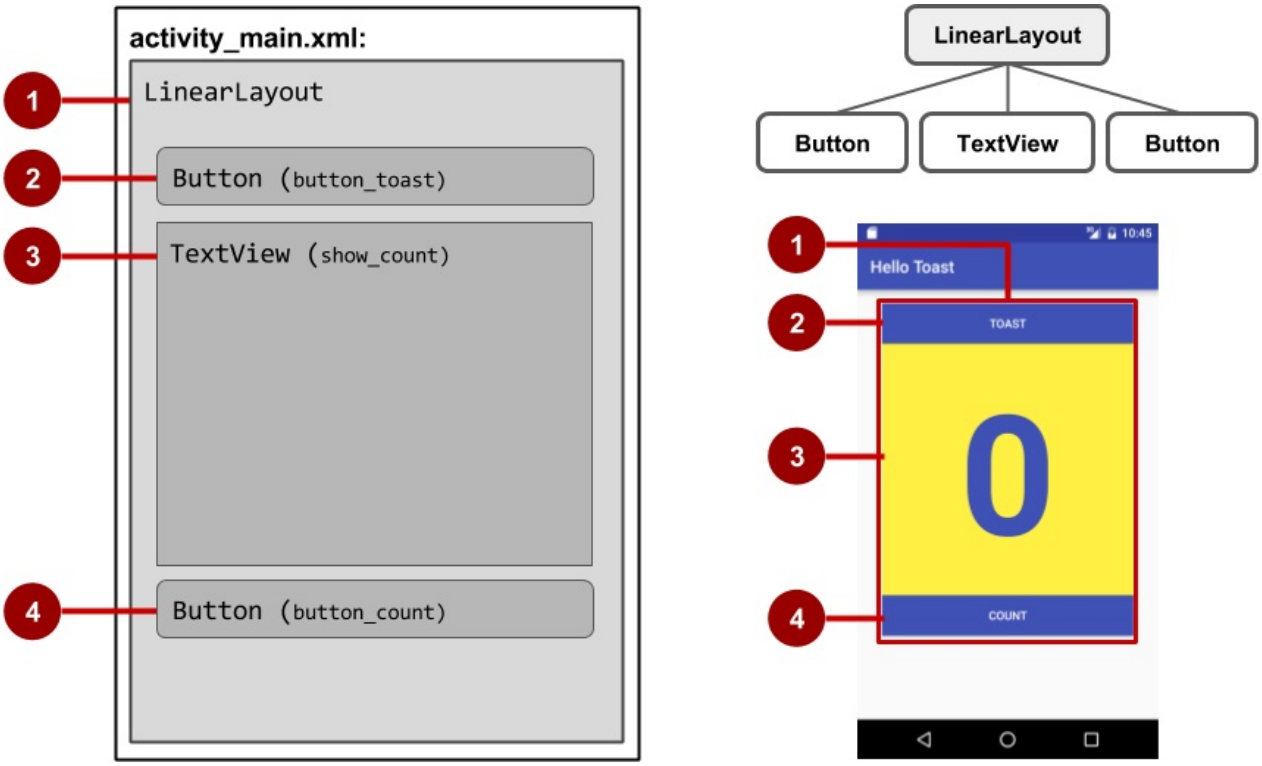
- LinearLayout: Grup tampilan anak yang diposisikan dan disejajarkan secara horizontal atau secara vertikal.
- RelativeLayout: Grup tampilan anak yang setiap tampilannya diposisikan dan disejajarkan relatif terhadap tampilan dalam grup tampilan. Dengan kata lain, posisi tampilan anak bisa dijelaskan dalam hubungan satu sama lain atau dengan grup tampilan induk.
- ConstraintLayout: Grup tampilan anak yang menggunakan titik jangkar, tepi, dan panduan untuk mengontrol cara memosisikan tampilan relatif terhadap elemen lain di layout. ConstraintLayout didesain untuk mempermudah saat menyeret dan melepaskan tampilan di editor layout.

- TableLayout: Grup tampilan anak yang disusun ke dalam baris dan kolom.
- AbsoluteLayout: Grup yang memungkinkan Anda menetapkan lokasi pasti (koordinat x/y) tampilan anaknya. Layout mutlak bersifat kurang fleksibel dan lebih sulit dikelola daripada tipe layout lainnya tanpa pemosisian mutlak.
- FrameLayout: Grup tampilan anak bertumpuk. FrameLayout didesain untuk memblokir area di layar guna menampilkan satu tampilan. Tampilan anak digambar bertumpuk, dengan anak yang baru saja ditambahkan di atas. Ukuran FrameLayout adalah ukuran tampilan anak terbesarnya.
- GridLayout: Grup yang menempatkan layar anaknya dalam kotak persegi panjang yang bisa digulir.




**Tip**: Ketahui selengkapnya tentang tipe layout yang berbeda di Objek Layout Umum.

Contoh sederhana dari layout dengan tampilan anak adalah aplikasi Hello Toast di salah satu pelajaran awal. Tampilan aplikasi Hello Toast muncul dalam gambar di bawah ini sebagai diagram file layout (activity_main.xml), bersama diagram hierarki (kanan atas) dan tangkapan layar layout yang benar-benar selesai (kanan bawah).




Dalam gambar di atas:

1. Layout akar LinearLayout, yang berisi semua tampilan anak, disetel ke orientasi vertikal.
2. Button ( `button_toast` ) tampilan anak. Sebagai tampilan anak pertama, muncul di bagian atas di layout linear.
3. TextView ( `show_count` ) tampilan anak. Sebagai tampilan anak kedua, muncul di bawah tampilan anak pertama di layout linear.
4. Button ( `button_count` ) tampilan anak. Sebagai tampilan anak ketiga, muncul di bawah tampilan anak kedua di layout linear.

Hierarki tampilan bisa tumbuh menjadi kompleks untuk aplikasi yang menampilkan banyak tampilan di layar. Penting untuk memahami hierarki tampilan, karena akan memengaruhi apakah tampilan terlihat dan apakah digambar secara efisien.

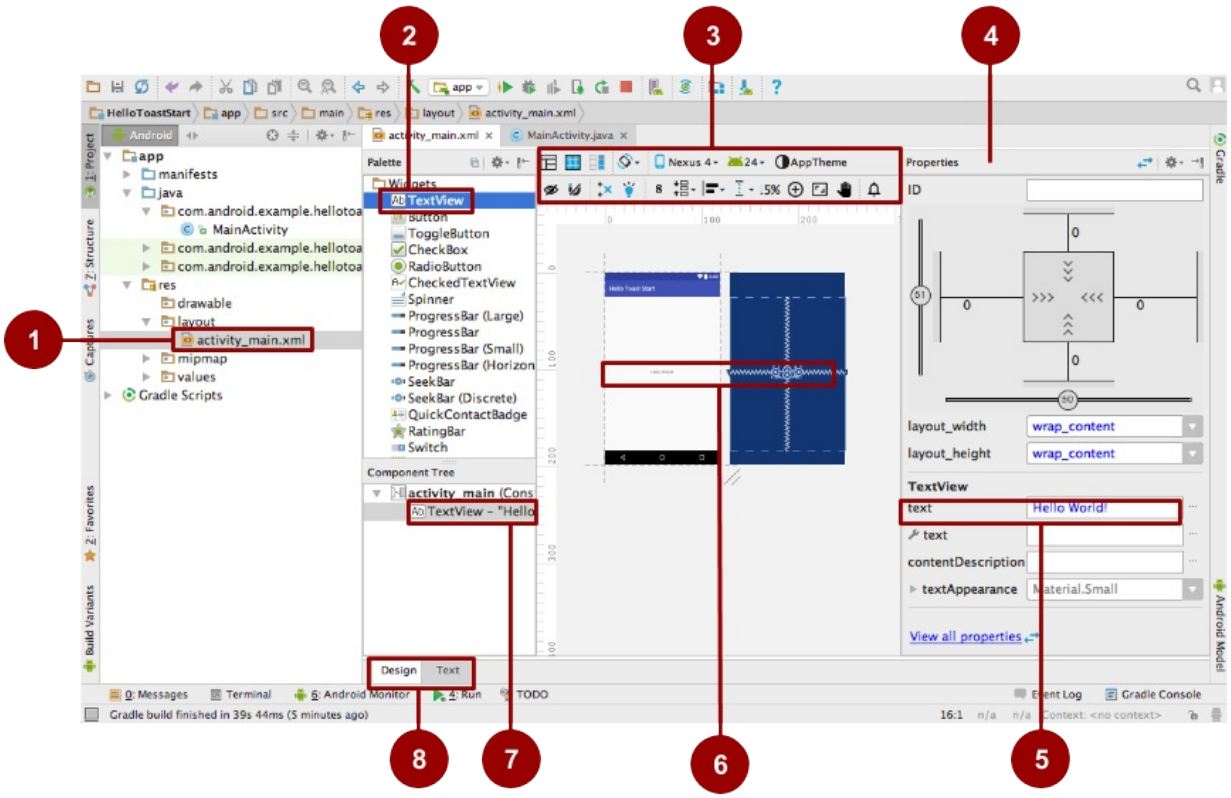
**Tip**: Anda bisa menjelajahi hierarki tampilan aplikasi menggunakan Hierarchy Viewer. Hierarchy Viewer menampilkan tampilan pohon hierarki dan memungkinkan Anda menganalisis kinerja tampilan di perangkat Android. Masalah kinerja dibahas di bab berikutnya.

Anda mendefinisikan tampilan di editor layout, atau dengan memasukkan kode XML. Editor layout menunjukkan representasi visual kode XML.

## Menggunakan editor layout

Gunakan editor layout untuk mengedit file layout. Anda bisa menyeret dan melepas objek tampilan ke dalam panel grafik, serta menyusun, mengubah ukuran, dan menetapkan propertinya. Anda akan segera melihat efek perubahan yang dilakukan.

Untuk menggunakan editor layout, buka file layout XML. Editor layout muncul bersama tab **Design** di bagian bawah yang disorot. (Jika tab **Text** disorot dan Anda melihat kode XML, klik tab **Design**.) Untuk template Empty Activity, layout seperti yang ditampilkan dalam gambar di bawah ini.




Dalam gambar di atas:

1. File **XML layout**. File layout XML, biasanya diberi nama file **activiy_main.xml**. Klik dua kali untuk membuka editor layout.
2. **Palet elemen UI** (tampilan). Panel Palette menyediakan daftar elemen UI dan layout. Tambahkan elemen atau layout ke UI dengan menyeretnya ke panel desain.
3. **Bilah alat desain**. Bilah alat panel desain menyediakan tombol untuk mengonfigurasi penampilan layout dalam panel desain dan untuk mengedit properti layout. Lihat gambar di bawah ini untuk detail.

   **Tip**: Arahkan kursor ke atas setiap ikon untuk menampilkan keterangan alat yang merangkum fungsinya.

4. **Panel Properties**. Panel Properties menyediakan kontrol properti untuk tampilan yang dipilih.
5. **Kontrol properti**. Kontrol properti sesuai dengan atribut XML. Yang ditampilkan dalam gambar adalah properti `Text` dari TextView yang dipilih, yang disetel ke `Hello World!` .
6. **Panel desain**. Seret tampilan dari panel Palette ke panel desain untuk memosisikannya di layout.
7. **Component Tree**. Panel Component Tree menampilkan hierarki tampilan. Klik tampilan atau grup tampilan dalam panel ini untuk memilihnya. Gambar menampilkan TextView yang dipilih.

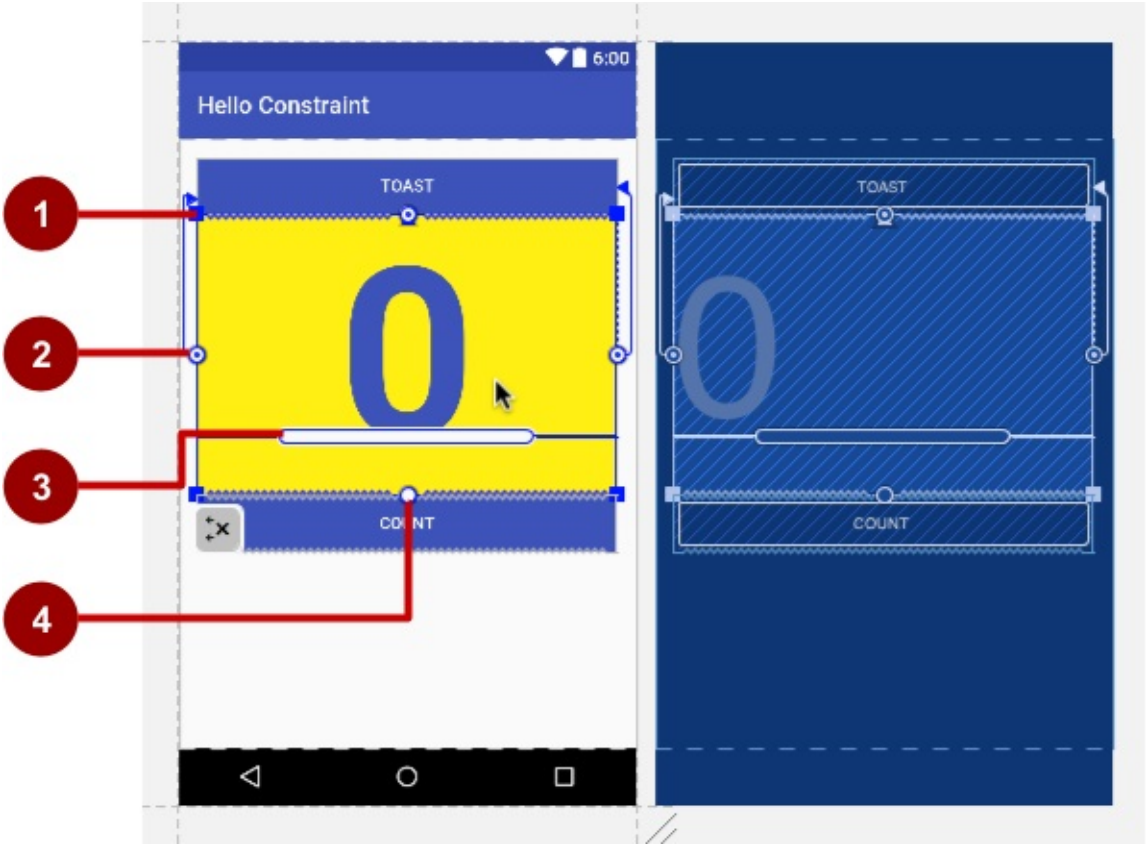
8. Tab **Design** dan **Text**. Klik **Design** untuk melihat editor layout, atau **Text** untuk melihat kode XML.

Bilah alat (bantu) desain editor layout menawarkan deretan tombol yang memungkinkan Anda mengonfigurasi penampilan layout:


Dalam gambar di atas:

1. **Design**, **Blueprint**, dan **Both**: Klik ikon **Design** (ikon pertama) untuk menampilkan pratinjau warna layout. Klik ikon **Blueprint** (ikon tengah) untuk hanya menampilkan ringkasan setiap tampilan. Anda bisa melihat *kedua* tampilan bersebelahan dengan mengeklik ikon ketiga.
2. **Orientasi layar**: Klik untuk memutar perangkat antara lanskap dan potret.
3. **Tipe dan ukuran perangkat**: Pilih tipe perangkat (ponsel/tablet, Android TV, atau Android Wear) dan konfigurasi layar (ukuran dan kepadatan).
4. **Versi API**: Pilih versi Android yang digunakan untuk pratinjau layout.
5. **Tema aplikasi**: Pilih tema UI yang akan diterapkan pada pratinjau.
6. **Bahasa**: Pilih bahasa untuk menampilkan string UI Anda. Daftar ini hanya menampilkan bahasa yang tersedia dalam sumber daya string.
7. **Varian Layout**: Alihkan ke salah satu layout alternatif untuk file ini, atau buat yang baru.

Editor layout menawarkan lebih banyak fitur di tab **Design** bila Anda menggunakan ConstraintLayout, termasuk tuas untuk mendefinisikan pembatas. *Pembatas* adalah koneksi atau penyejajaran ke tampilan lainnya, ke layout induk, atau ke panduan yang tidak terlihat. Setiap pembatas muncul sebagai garis yang membentang dari tuas melingkar. Setiap tampilan memiliki tuas pembatas melingkar di bagian tengah setiap sisi. Setelah memilih tampilan di panel Component Tree atau mengekliknya di layout, tampilan juga menampilkan tuas pengubah ukuran pada setiap sudut.




Dalam gambar di atas:

1. **Tuas pengubah ukuran**.

2. **Garis pembatas dan tuas**. Dalam gambar, pembatas menyejajarkan sisi kiri tampilan ke sisi kiri tombol.
3. **Tuas patokan**. Tuas patokan menyejajarkan patokan teks tampilan ke patokan tampilan lainnya.
4. **Tuas pembatas** tanpa garis pembatas.

## Menggunakan XML

Terkadang lebih cepat dan lebih mudah mengedit kode XML secara langsung, terutama saat menyalin dan menempelkan kode untuk tampilan serupa.

Untuk menampilkan dan mengedit kode XML, buka file layout XML. Editor layout muncul bersama tab **Design** di bagian bawah yang disorot. Klik tab **Text** untuk melihat kode XML. Yang berikut ini menampilkan cuplikan kode XML untuk LinearLayout dengan Button dan TextView:

```
<LinearLayout 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"
    ... >

    <Button
        android:id="@+id/button_toast"
        android:layout_width="@dimen/my_view_width"
        android:layout_height="wrap_content"
        ... />

 <TextView
        android:id="@+id/show_count"
        android:layout_width="@dimen/my_view_width"
        android:layout_height="@dimen/counter_height"
        ... />
    ...
</LinearLayout>
```

## Atribut XML (properti tampilan)

Tampilan memiliki *properti* yang mendefinisikan tempat munculnya tampilan di layar, ukurannya, dan bagaimana kaitan tampilan dengan tampilan lainnya, dan cara tampilan merespons masukan pengguna. Saat mendefinisikan tampilan di XML, properti dirujuk sebagai *atribut*.

Misalnya, dalam keterangan XML TextView berikut, `android:id`, `android:layout_width`, `android:layout_height`, `android:background`, merupakan atribut XML yang diterjemahkan secara otomatis menjadi properti TextView:

```
<TextView
       android:id="@+id/show_count"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:background="@color/myBackgroundColor"
       android:textStyle="bold"
       android:text="@string/count_initial_value"
/>
```

Atribut biasanya berbentuk seperti ini:

```
android:attribute_name="value"
```

*attribute_name* adalah nama atribut. *value* adalah string dengan nilai untuk atribut. Misalnya:

```
android:textStyle="bold"
```

Jika *value* adalah sumber daya, seperti warna, simbol `@` menetapkan jenis sumber dayanya. Misalnya:

```
android:background="@color/myBackgroundColor"
```

Atribut latar belakang disetel ke sumber daya warna yang diidentifikasi sebagai `myBackgroundColor`, yang dideklarasikan sebagai `#FFF043`. Sumber daya warna dijelaskan dalam "Atribut terkait gaya" dalam bab ini.

Setiap tampilan dan grup tampilan mendukung berbagai atribut XML-nya sendiri. Beberapa atribut telah ditetapkan untuk tampilan (misalnya, TextView mendukung atribut `textSize`), namun atribut-atribut ini juga diwarisi oleh tampilan apa pun yang mungkin memperluas kelas TextView. Sebagian bersifat umum untuk semua tampilan, karena diwarisi dari kelas View akar (seperti atribut `android:id`). Untuk keterangan atribut khusus, lihat bagian ringkasan dokumentasi kelas View.

## Mengidentifikasi tampilan

Untuk mengidentifikasi tampilan secara unik dan merujuknya dari kode, Anda harus memberikan ID. Atribut `android:id` memungkinkan Anda menetapkan `id` yang unik— yakni identifier sumber daya untuk tampilan.

Misalnya:

```
android:id="@+id/button_count"
```

Bagian `"@+id/button_count"` dari atribut di atas akan membuat `id` baru yang disebut `button_count` untuk tampilan. Anda menggunakan simbol plus ( `+` ) untuk menunjukkan bahwa Anda sedang membuat baru `id`.

## Merujuk tampilan

Untuk merujuk ke identifier sumber daya yang ada, hilangkan simbol plus ( `+` ). Misalnya, untuk merujuk tampilan berdasarkan `id` -nya dalam atribut *lain*, misalnya `android:layout_toLeftOf` (yang dijelaskan di bagian berikutnya) untuk mengontrol posisi tampilan, Anda perlu menggunakan:

```
android:layout_toLeftOf="@id/show_count"
```

Dalam atribut di atas, `"@id/show_count"` merujuk ke tampilan dengan identifier sumber daya `show_count`. Atribut memosisikan tampilan agar "ke kiri dari" tampilan `show_count`.

## Memosisikan tampilan

Beberapa atribut pemosisian yang terkait layout diperlukan untuk tampilan, dan secara otomatis muncul bila Anda menambahkan tampilan ke layout XML, siap untuk Anda tambahkan nilainya.

## Pemosisian LinearLayout

Misalnya, LinearLayout diperlukan untuk menyetel atribut ini:

- android:layout_width
- android:layout_height
- android:orientation

Atribut `android:layout_width` dan `android:layout_height` bisa menggunakan salah satu dari tiga nilai ini:

- `match_parent` akan meluaskan tampilan untuk mengisi induknya dengan lebar dan tinggi. Bila LinearLayout adalah tampilan akar, ia akan meluaskan ke ukuran layar perangkat. Untuk tampilan dalam grup tampilan akar, ia akan meluaskan ke ukuran grup tampilan induk.
- `wrap_content` akan menciutkan dimensi tampilan yang cukup besar untuk menampung isinya. (Jika tidak ada isinya, tampilan menjadi tidak terlihat.)
- Gunakan `dp` (piksel yang tidak tergantung perangkat) berjumlah tetap untuk menetapkan ukuran tetap, yang disesuaikan untuk ukuran layar perangkat. Misalnya, `16dp` berarti 16 piksel yang tidak tergantung perangkat. Piksel

yang tidak tergantung perangkat dan dimensi lain dijelaskan di "Dimensi" dalam bab ini.

Atribut `android:orientation` bisa berupa:

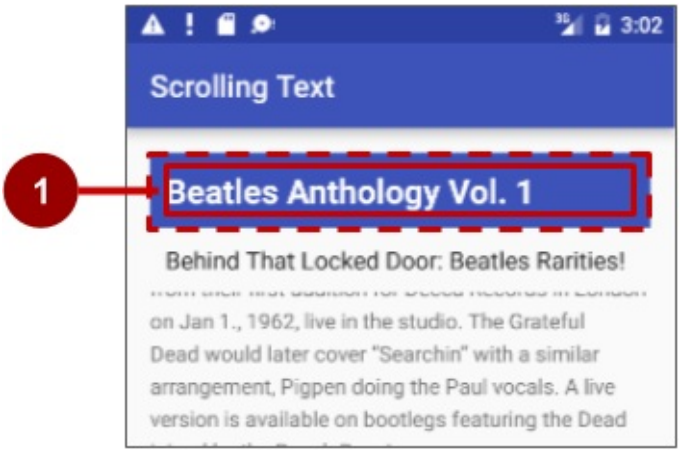
- `horizontal:` Tampilan disusun dari kiri ke kanan.
- `vertical:` Tampilan disusun dari atas ke bawah.

Atribut terkait layout lainnya antara lain:

- `Android:layout_gravity` : Atribut ini digunakan bersama tampilan untuk mengontrol tempat menyusun tampilan dalam grup tampilan induknya. Misalnya, atribut berikut memusatkan tampilan secara horizontal di layar:

  ```
  android:layout_gravity="center_horizontal"
  ```

- Pengisi adalah ruang, yang diukur dalam piksel yang tidak tergantung perangkat, antara tepi tampilan dan isi tampilan, seperti yang ditampilkan dalam gambar di bawah ini.




Dalam gambar di atas:

1. Pengisi adalah ruang antara tepi tampilan (garis putus-putus) dan isi tampilan (garis utuh). Pengisi tidak sama dengan margin, yang merupakan ruang dari tepi tampilan ke induknya.

Ukuran tampilan menyertakan pengisinya. Berikut ini adalah atribut pengisi yang umum digunakan:

- `Android:padding` : Menyetel pengisi keempat tepi.
- `android:paddingTop` : Menyetel pengisi tepi atas.
- `android:paddingBottom` : Menyetel pengisi tepi bawah.
- `android:paddingLeft` : Menyetel pengisi tepi kiri.
- `android:paddingRight` : Menyetel pengisi tepi kanan.
- `android:paddingStart` : Menyetel pengisi awal tampilan; digunakan di tempat di atas, terutama bersama tampilan yang panjang dan sempit.
- `android:paddingEnd` : Menyetel pengisi, dalam piksel, tepi ujung; yang digunakan bersama `android:paddingStart` .

**Tip**: Untuk melihat semua atribut XML untuk LinearLayout, lihat bagian Rangkuman referensi LinearLayout di Panduan Developer. Layout akar lainnya, seperti RelativeLayout dan AbsoluteLayout, mencantumkan atribut XML-nya di bagian Rangkuman.

## Pemosisian RelativeLayout

Grup tampilan berguna lainnya untuk layout adalah RelativeLayout, yang bisa Anda gunakan untuk memosisikan tampilan anak yang berhubungan satu sama lain atau dengan induk. Atribut yang bisa Anda gunakan bersama RelativeLayout antara lain berikut ini:

- android:layout_toLeftOf: Memosisikan tepi kanan tampilan ini ke kiri tampilan lainnya (yang diidentifikasi melalui `ID` -nya).
- android:layout_toRightOf: Memosisikan tepi kiri tampilan ini ke kanan tampilan lainnya (yang diidentifikasi melalui `ID` -

nya).
- android:layout_centerHorizontal: Memusatkan tampilan ini secara horizontal dalam induknya.
- android:layout_centerVertical: Memusatkan tampilan ini secara vertikal dalam induknya.
- android:layout_alignParentTop: Memosisikan tepi atas tampilan ini agar cocok dengan tepi atas induknya.
- android:layout_alignParentBottom: Memosisikan tepi bawah tampilan ini agar cocok dengan tepi bawah induknya.

Untuk daftar lengkap atribut tampilan dalam RelativeLayout, lihat RelativeLayout.LayoutParams.

### Atribut terkait gaya

Anda menetapkan atribut gaya untuk menyesuaikan penampilan tampilan. Tampilan yang *tidak* memiliki atribut gaya, misalnya `android:textColor`, `android:textSize`, dan `android:background`, menggunakan gaya yang didefinisikan di tema aplikasi.

Berikut ini adalah atribut terkait gaya yang digunakan dalam contoh layout XML di bagian sebelumnya:

- `Android:background`: Menetapkan warna atau sumber daya dapat digambar untuk digunakan sebagai latar belakang.
- `android:text`: Menetapkan teks untuk ditampilkan di tampilan.
- `android:textColor`: Menetapkan warna teks.
- `android:textSize`: Menetapkan ukuran teks.
- `android:textStyle`: Menetapkan gaya teks, misalnya `bold`.

## File sumber daya

File sumber daya adalah cara memisahkan nilai statis dari kode sehingga Anda tidak harus mengubah kode itu sendiri untuk mengubah nilai. Anda bisa menyimpan semua string, layout, dimensi, warna, gaya, dan teks menu secara terpisah di file sumber daya.

File sumber daya disimpan dalam folder yang berada di folder **res**, termasuk:

- **drawable**: Untuk gambar dan ikon
- **layout**: Untuk file sumber daya layout
- **menu**: Untuk item menu
- **mipmap**: Untuk kumpulan ikon aplikasi yang sudah dihitung dan dioptimalkan yang digunakan oleh Peluncur
- **values**: Untuk warna, dimensi, string, dan gaya (atribut tema)

Sintaks untuk merujuk sumber daya di layout XML adalah seperti berikut:

```
@package_name:resource_type/resource_name
```

- *package_name* adalah nama paket tempat sumber daya berada. Ini tidak diperlukan saat merujuk sumber daya dari paket yang sama — yakni, yang disimpan di folder **res** proyek Anda.
- *resource_type* adalah `R` subkelas untuk tipe sumber daya. Lihat Tipe Sumber Daya untuk informasi selengkapnya tentang setiap tipe sumber daya dan cara merujuknya.
- *resource_name* adalah nama file sumber daya tanpa ekstensi, atau nilai atribut `android:name` di elemen XML.

Misalnya, pernyataan layout XML berikut menyetel atribut `android:text` ke sumber daya `string`:

```
android:text="@string/button_label_toast"
```

- *resource_type* adalah `string`.
- resource_name adalah `button_label_toast.`
- *package_name* tidak diperlukan karena sumber daya disimpan di proyek (dalam file strings.xml).

Contoh lainnya: pernyataan layout XML ini menyetel atribut `android:background` ke sumber daya `color`, dan karena sumber daya didefinisikan di proyek (dalam file colors.xml), *package_name* tidak ditetapkan:

```
android:background="@color/colorPrimary"
```

Dalam contoh berikut, pernyataan layout XML menyetel atribut `android:textColor` ke sumber daya `color` . Akan tetapi, sumber daya tidak didefinisikan dalam proyek, melainkan disediakan oleh Android, sehingga *package_name* `android` juga harus ditetapkan, yang diikuti dengan titik dua:

```
android:textColor="@android:color/white"
```

**Tip**: Untuk informasi selengkapnya tentang mengakses sumber daya dari kode, lihat Mengakses Sumber Daya. Untuk konstanta warna Android, lihat sumber daya R.color standar Android.

## File sumber daya nilai

Menyimpan nilai sebagai string dan warna dalam file sumber daya terpisah mempermudah pengelolaannya, terutama jika Anda menggunakannya lebih dari sekali di layout.

Misalnya, penting sekali menyimpan string dalam file sumber daya terpisah untuk menerjemahkan dan melokalkan aplikasi, sehingga Anda bisa membuat file sumber daya string untuk setiap bahasa tanpa mengubah kode. File sumber daya untuk gambar, warna, dimensi, dan atribut lainnya berguna untuk mengembangkan aplikasi bagi orientasi dan ukuran layar perangkat yang berbeda.

## String

Sumber daya string berada dalam file **strings.xml** dalam folder **values** di dalam folder **res** saat menggunakan Project: Tampilan Android. Anda bisa mengedit file ini secara langsung dengan membukanya:

```
<resources>
    <string name="app_name">Hello Toast</string>
    <string name="button_label_count">Count</string>
    <string name="button_label_toast">Toast</string>
    <string name="count_initial_value">0</string>
</resources>
```

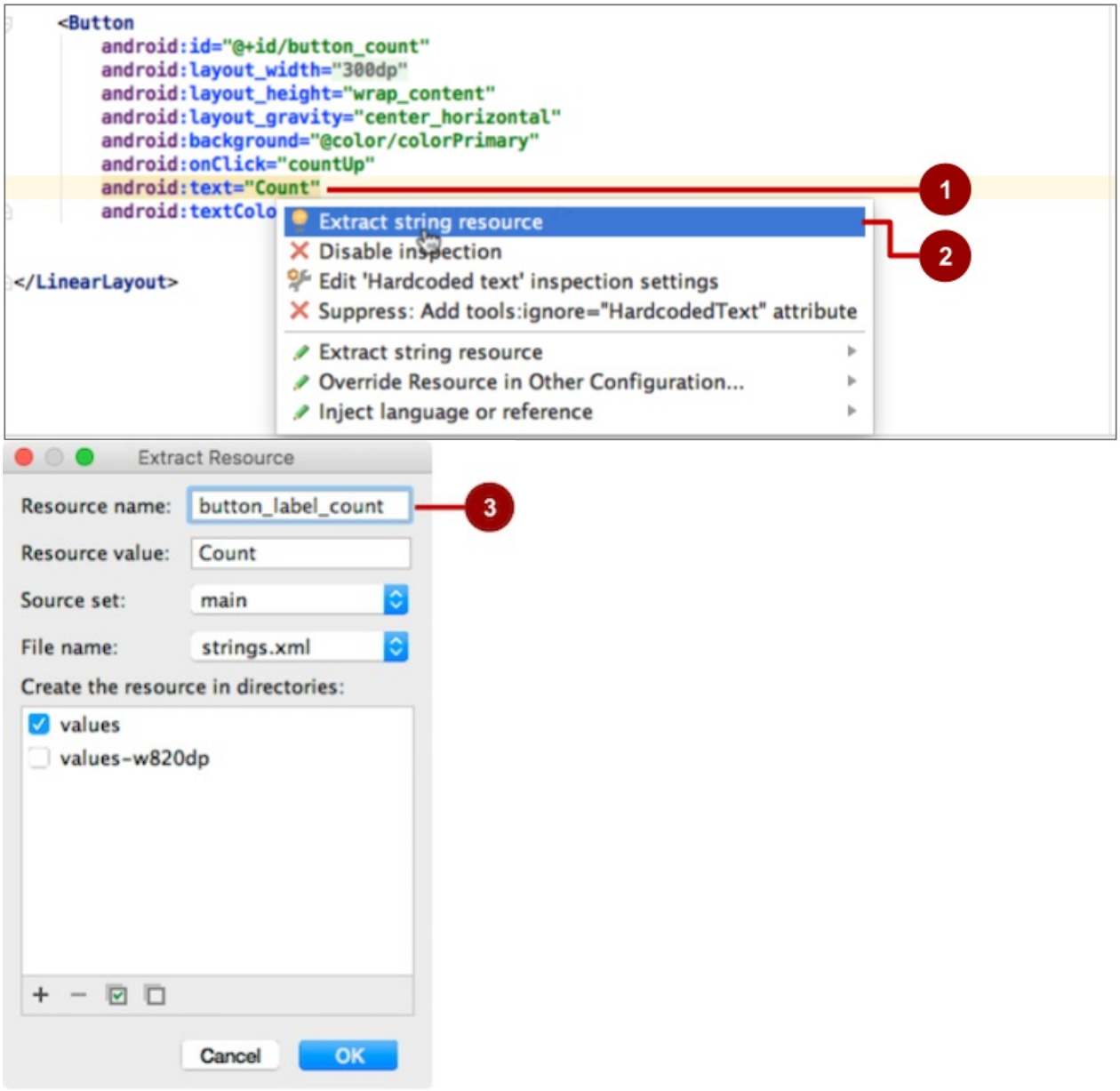
Di sini, `name` (misalnya, `button_label_count` ) adalah nama sumber daya yang Anda gunakan di kode XML, seperti dalam atribut berikut:

```
android:text="@string/button_label_count"
```

Nilai string `name` ini adalah kata ( `Count` ) yang dimasukkan dalam tag `<string></string>` (Anda jangan menggunakan tanda kutip kecuali jika tanda kutip harus menjadi bagian dari nilai string.)

## Mengekstrak string ke sumber daya

Anda juga harus *mengekstrak* string hard-code dalam file layout XML ke sumber daya string. Untuk mengekstrak string hard-code dalam layout XML, ikuti langkah-langkah ini (lihat gambar):




1. Klik string hard-code, dan tekan Alt-Enter di Windows, atau Option-Return pada Mac OS X.
2. Pilih **Extract string resource**.
3. Edit nama Sumber Daya untuk nilai string.

Selanjutnya Anda bisa menggunakan nama sumber daya di kode XML. Gunakan ekspresi `"@string/resource_name"` (termasuk tanda kutip) untuk merujuk sumber daya string:

```
android:text="@string/button_label_count"
```

## Warna

Sumber daya warna berada di file **colors.xml** dalam folder **values** di dalam folder **res** saat menggunakan Project: Tampilan Android. Anda bisa mengedit file ini secara langsung:

```
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="myBackgroundColor">#FFF043</color>
</resources>
```

Di sini, `name` (misalnya, `colorPrimary`) adalah nama sumber daya yang Anda gunakan dalam kode XML:

```
android:textColor="@color/colorPrimary"
```

Nilai warna `name` ini adalah nilai warna heksadesimal ( `#3F51B5` ) yang dimasukkan dalam tag `<color></color>` . Nilai heksadesimal menetapkan nilai merah, hijau, dan biru (RGB). Nilai selalu diawali dengan karakter pound ( `#` ), yang diikuti dengan informasi Alpha-Red-Green-Blue. Misalnya, nilai heksadesimal hitam adalah #000000, sedangkan nilai heksadesimal untuk varian biru langit adalah #559fe3. Nilai warna dasar dicantumkan dalam dokumentasi kelas Color.

Warna `colorPrimary` adalah salah satu warna dasar yang telah didefinisikan sebelumnya dan digunakan untuk bilah aplikasi. Di aplikasi produksi, Anda bisa, misalnya, menyesuaikannya agar sesuai dengan merek. Menggunakan warna dasar untuk elemen UI lainnya akan membuat UI seragam.

**Tip**: Untuk spesifikasi desain material warna Android, lihat Gaya dan Menggunakan Tema Material. Untuk nilai heksadesimal warna umum, lihat Kode Warna Heksadesimal. Untuk konstanta warna Android, lihat sumber daya R.color standar Android.

Anda bisa melihat blok kecil pilihan warna di margin kiri di sebelah deklarasi sumber daya warna di **colors.xml**, juga di margin kiri di sebelah atribut yang menggunakan nama sumber daya di file XML layout.


**Tip**: Untuk melihat warna di munculan, aktifkan fitur dokumentasi Munculan otomatis. Pilih **Android Studio > Preferences > Editor > General > Code Completion**, dan centang opsi "Autopopup documentation in (ms)". Selanjutnya Anda bisa mengarahkan kursor ke atas nama sumber daya warna untuk melihat warnanya.

## Dimensi

Dimensi harus dipisahkan dari kode untuk mempermudah pengelolaannya, terutama jika Anda perlu menyesuaikan layout untuk resolusi perangkat yang berbeda. Selain itu juga memudahkan pengukuran yang konsisten untuk tampilan, dan untuk mengubah ukuran beberapa tampilan dengan mengubah satu sumber daya dimensi.

Sumber daya dimensi berada di file **dimens.xml** dalam folder **values** di dalam folder **res** saat menggunakan Project: Tampilan Android. **dimens.xml** yang ditampilkan di tampilan bisa menjadi folder yang memiliki lebih dari satu file **dimens.xml** untuk resolusi perangkat yang berbeda. Misalnya, aplikasi yang dibuat dari template Empty Activity menyediakan file **dimens.xml** kedua untuk 820 dp.

Anda bisa mengedit file ini secara langsung dengan membukanya:

```
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
    <dimen name="my_view_width">300dp</dimen>
    <dimen name="count_text_size">200sp</dimen>
    <dimen name="counter_height">300dp</dimen>
</resources>
```

Di sini, `name` (misalnya, `activity_horizontal_margin`) adalah nama sumber daya yang Anda gunakan di kode XML:

```
android:paddingLeft="@dimen/activity_horizontal_margin"
```

Nilai `name` ini adalah pengukuran (`16dp`) yang dimasukkan dalam tag `<dimen></dimen>`.

Anda bisa mengekstrak dimensi dengan cara yang sama seperti string:

1. Klik dimensi hard-code, dan tekan Alt-Enter di Windows, atau tekan Option-Return di Mac OS X.
2. Pilih **Extract dimension resource**.
3. Edit nama Sumber Daya untuk nilai dimensi.

Piksel yang tidak tergantung perangkat (`dp`) tidak tergantung resolusi layar. Misalnya, `10px` (10 piksel tetap) akan terlihat sedikit lebih kecil pada layar yang beresolusi lebih tinggi, namun Android akan menskalakan 1 `0dp` (10 piksel yang tidak tergantung perangkat) untuk melihat langsung di layar resolusi berbeda. Ukuran teks juga bisa disetel untuk terlihat langsung di layar resolusi berbeda dengan menggunakan ukuran *piksel yang diskalakan* (`sp`).

**Tip**: Untuk informasi selengkapnya tentang unit `dp` dan `sp`, lihat Mendukung Kepadatan Berbeda.

### Gaya

Gaya adalah sumber daya yang menetapkan atribut umum seperti tinggi, pengisi, warna font, ukuran font, dan warna latar belakang. Gaya ditujukan untuk atribut yang memodifikasi rupa tampilan.

Gaya didefinisikan di file **styles.xml** dalam folder **values** di dalam folder **res** saat menggunakan Project: Tampilan Android. Anda bisa mengedit file ini secara langsung. Gaya dibahas dalam bab berikutnya, bersama Spesifikasi Desain Material.

### File sumber daya lainnya

Android Studio mendefinisikan sumber daya lainnya yang dibahas dalam bab lain:

- **Gambar dan ikon**. Folder **drawable** menyediakan sumber daya ikon dan gambar. Jika aplikasi Anda tidak memiliki folder drawable, Anda bisa membuatnya di dalam folder res secara manual. Untuk informasi selengkapnya tentang sumber daya dapat digambar, lihat Sumber Daya Dapat Digambar di bagian Sumber Daya Aplikasi dari Panduan Developer Android.
- **Ikon yang dioptimalkan**. Folder **mipmap** umumnya berisi kumpulan ikon aplikasi yang sudah dihitung dan dioptimalkan, yang digunakan oleh Peluncur. Luaskan folder untuk melihat apakah versi ikon disimpan sebagai sumber daya untuk kepadatan layar yang berbeda.
- **Menu**. Anda bisa menggunakan file sumber daya XML untuk mendefinisikan item menu dan menyimpannya di proyek dalam folder **menu**. Menu dijelaskan dalam bab berikutnya.

## Merespons klik tampilan

*Kejadian klik* terjadi bila pengguna mengetuk atau mengeklik tampilan yang bisa diklik seperti Button, ImageButton, ImageView (mengetuk atau mengeklik gambar), atau FloatingActionButton.

Pola model-view-presenter berguna untuk memahami cara merespons klik tampilan. Bila kejadian terjadi bersama tampilan, kode *presenter* akan melakukan aksi yang memengaruhi kode *model*. Agar pola ini berfungsi, Anda harus:

- Menulis metode Java yang melakukan aksi khusus, yang ditentukan oleh logika kode *model* — yakni tindakan yang bergantung pada hal yang Anda inginkan untuk dilakukan aplikasi bila kejadian ini terjadi. Ini biasanya disebut sebagai *penangan kejadian*.
- Mengaitkan metode penangan kejadian ini ke *tampilan*, sehingga metode berjalan bila kejadian terjadi.

### Atribut onClick

Android Studio menyediakan pintasan untuk mempersiapkan tampilan yang bisa diklik, dan untuk mengaitkan penangan kejadian dengan tampilan: gunakan atribut `android:onClick` bersama elemen tampilan yang bisa diklik di layout XML.

Misalnya, ekspresi XML berikut di file layout untuk serangkaian Button. `showToast()` sebagai penangan kejadian:

```
android:onClick="showToast"
```

Bila tombol `b` diketuk, atribut `android:onClick` -nya akan memanggil metode `showToast()` .

Tulis penangan kejadian, misalnya `showToast()` yang direferensikan dalam kode XML di atas, untuk memanggil metode lain yang mengimplementasikan logika *model* aplikasi:

```
public void showToast(View view) {
        // Do something in response to the button click.
}
```

Agar dapat digunakan bersama atribut `android:onClick` , metode `showToast()` harus berupa `public` , mengembalikan `void` , dan memerlukan parameter `view` agar dapat mengetahui tampilan mana yang memanggil metode.

Android Studio menyediakan pintasan untuk membuat *stub* penangan kejadian (placeholder untuk metode yang bisa Anda isi nanti) di kode Java untuk aktivitas yang terkait dengan layout XML. Ikuti langkah-langkah ini:

1. Di dalam file layout XML (misalnya activity_main.xml), klik nama metode dalam pernyataan atribut `android:onClick` .
2. Tekan Alt-Enter di Windows atau Option-Return di Mac OS X, dan pilih **Create onClick event handler**.
3. Pilih aktivitas yang terkait dengan file layout (misalnya **MainActivity**) dan klik **OK**. Ini akan membuat stub metode placeholder di MainActivity.java.

## Memperbarui tampilan

Untuk memperbarui materi tampilan, misalnya mengganti teks di TextView, terlebih dahulu kode Anda harus membuat instance objek dari tampilan. Selanjutnya kode Anda bisa memperbarui objek, dengan demikian memperbarui tampilan.

Untuk merujuk tampilan dalam kode Anda, gunakan metode findViewById() kelas View, yang akan mencari tampilan berdasarkan `id` sumber daya. Misalnya, pernyataan berikut menyetel `mShowCount` menjadi TextView dengan `show_count` ID sumber daya:

```
mShowCount = (TextView) findViewById(R.id.show_count);
```

Dari poin ini, kode Anda bisa menggunakan `mShowCount` untuk menyatakan TextView, sehingga bila Anda memperbarui `mShowCount` , tampilan akan diperbarui.

Misalnya, ketika tombol berikut dengan atribut `android:onClick` diketuk, onClick akan memanggil metode `countUp()` :

```
android:onClick="countUp"
```

Anda bisa mengimplementasikan `countUp()` untuk menaikkan hitungan, mengubah hitungan ke string, dan menyetel string sebagai teks untuk objek `mShowCount` :

```
public void countUp(View view) {
        mCount++;
        if (mShowCount != null)
            mShowCount.setText(Integer.toString(mCount));
}
```

Karena Anda sudah mengaitkan `mShowCount` dengan TextView untuk menampilkan hitungan, metode `mShowCount.setText()` memperbarui tampilan teks di layar.