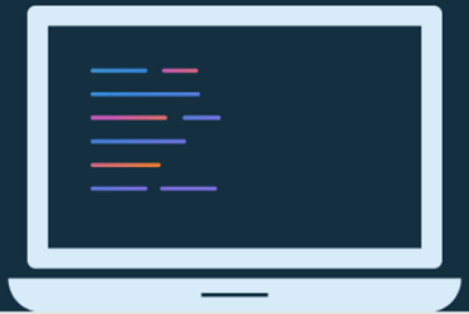
Bài học 13: Thiết kế giao diện người dùng của ứng dụng



Giới thiệu về bài học này

Bài học 13: Thiết kế giao diện người dùng của ứng dụng

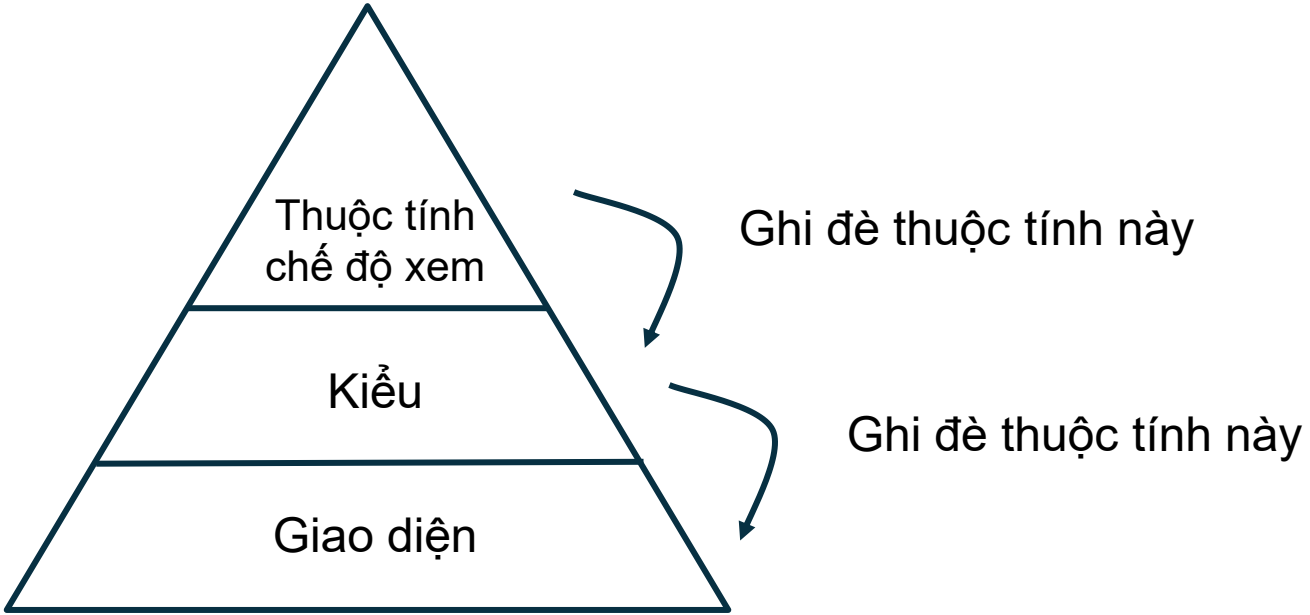
- Định kiểu cho Android
- Kiểu chữ
- Material Design
- Thành phần Material
- Bản địa hóa
- Ứng dụng mẫu
- Tóm tắt

Định kiểu cho Android

Hệ thống định kiểu của Android

- Dùng để chỉ định thiết kế hình ảnh của ứng dụng
- Giúp bạn duy trì giao diện nhất quán trên toàn ứng dụng của mình
- Phân cấp (bạn có thể kế thừa từ những kiểu mẹ và ghi đè các thuộc tính cụ thể)

Mức độ ưu tiên của từng phương thức định kiểu



Giao diện

- Là tập hợp các tài nguyên được đặt tên, hữu ích trên toàn ứng dụng
- Tài nguyên được đặt tên còn gọi là thuộc tính giao diện
- Ví dụ:
  - Dùng một giao diện để xác định màu chính và màu phụ trong ứng dụng
  - Dùng một giao diện để đặt phông chữ mặc định cho mọi văn bản trong một hoạt động

Khai báo giao diện

Trong tệp `res/values/themes.xml`:

```
<style name="Theme.MyApp" parent="Theme.MaterialComponents.Light">
    <item name="colorPrimary">@color/orange_500</item>
    <item name="colorPrimaryVariant">@color/orange_700</item>
    <item name="colorSecondary">@color/pink_200</item>
    <item name="colorSecondaryVariant">@color/pink_700</item>
    ...
</style>
```

Áp dụng giao diện

Trong tệp `AndroidManifest.xml`:

```
<manifest ... >
    <application ... >
        <activity android:theme="@style/Theme.MyApp" ... >
        </activity>
    </application>
</manifest>
```

Trong tệp bố cục:

```
<ConstraintLayout …
    android:theme="@style/Theme.MyApp">
```

Tham chiếu đến thuộc tính giao diện trong một bố cục

Trong tệp bố cục:

```
<LinearLayout …
    android:background="?attr/colorSurface">
```

Dùng cú pháp `?attr/themeAttributeName`.

Ví dụ: `?attr/colorPrimary`
`?attr/colorPrimaryVariant`

Kiểu

- Kiểu là tập hợp các thuộc tính chế độ xem, dành riêng cho một loại chế độ xem
- Dùng kiểu để tạo tập hợp thông tin định kiểu có thể tái sử dụng, chẳng hạn màu hoặc kích thước phông chữ
- Phù hợp để khai báo một số thiết kế phổ biến được dùng trên toàn ứng dụng của bạn

Khai báo kiểu

Trong tệp `res/values/styles.xml`:

```
<style name="DescriptionStyle">
    <item name="android:textColor">#00FF00</item>
    <item name="android:textSize">16sp</item>
    ...
</style>
```

Áp dụng kiểu

Trên một chế độ xem trong tệp bố cục:

```
<TextView
    style="@style/DescriptionStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/description_text" />
```

Tham chiếu đến thuộc tính giao diện trong một kiểu

Trong tệp `res/values/styles.xml`:

```
<style name="DescriptionStyle">
    <item name="android:textColor">?attr/colorOnSurface</item>
    <item name="android:textSize">16sp</item>
    ...
</style>
```

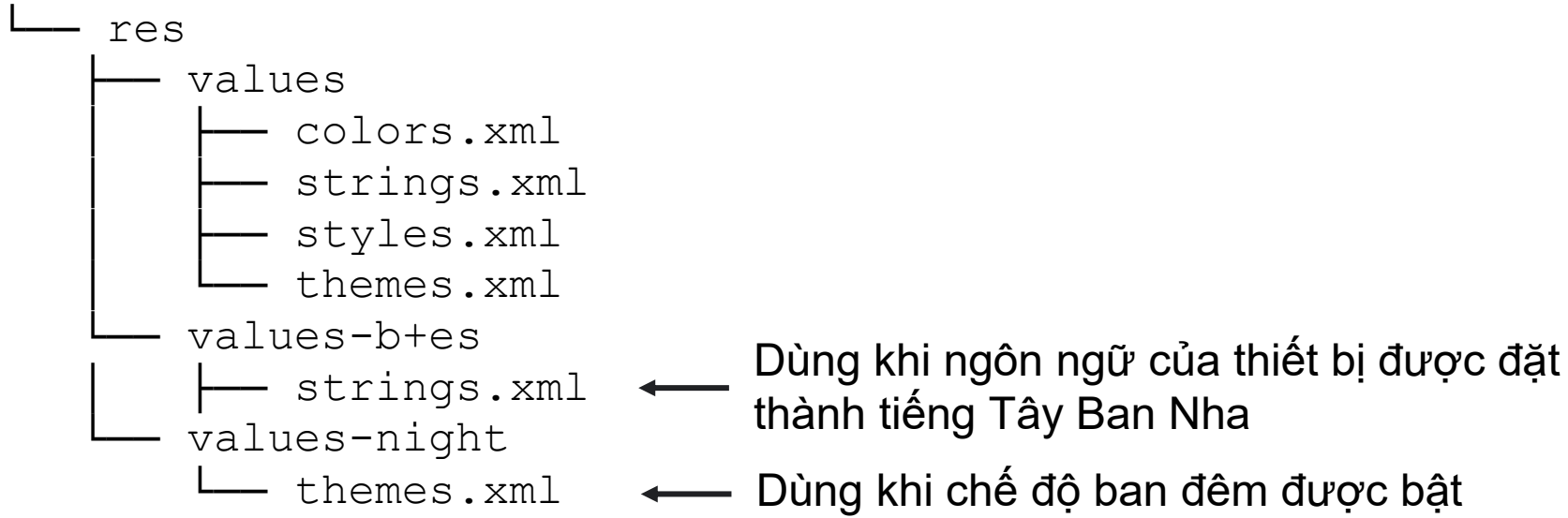
Thuộc tính chế độ xem

- Dùng các thuộc tính chế độ xem để đặt thuộc tính rõ ràng cho mỗi chế độ xem
- Bạn có thể dùng mọi thuộc tính có thể đặt được thông qua kiểu hoặc giao diện
- Dùng cho các thiết kế tùy chỉnh hoặc thiết kế một lần, chẳng hạn như lề, khoảng đệm hoặc các hạn chế

Thư mục tài nguyên

```
└── res
    ├── drawable
    ├── drawable-*
    ├── layout
    ├── menu
    ├── mipmap-*
    ├── navigation
    ├── values
    │   ├── colors.xml
    │   ├── dimens.xml
    │   ├── strings.xml
    │   ├── styles.xml
    │   └── themes.xml
    └── values-*
```

Cung cấp tài nguyên thay thế



Tài nguyên màu

Là một cách để đặt tên và chuẩn hóa màu trên toàn ứng dụng của bạn

Trong tệp `res/values/colors.xml`:

```
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    ...
</resources>
```

Được chỉ định là màu thập lục phân ở dạng `#AARRGGBB`

Tài nguyên thứ nguyên

Là một cách để đặt tên và chuẩn hóa các giá trị thứ nguyên trong bố cục của bạn

- Khai báo các giá trị thứ nguyên của bạn trong tệp `res/values/dimens.xml`:

```
<resources>
    <dimen name="top_margin">16dp</dimen>
</resources>
```

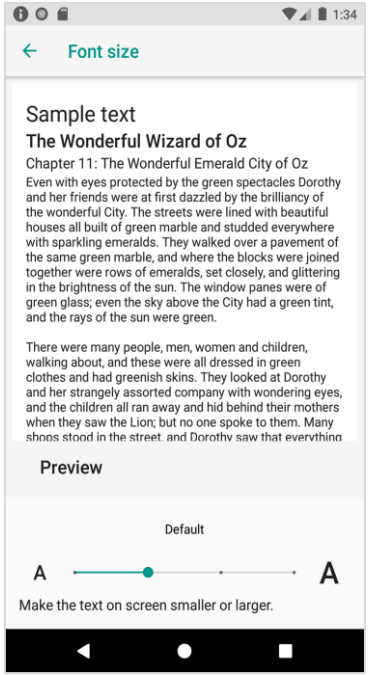
- Tham chiếu đến các giá trị đó bằng `@dimen/<name>` trong bố cục hoặc `R.dimen.<name>` trong mã:

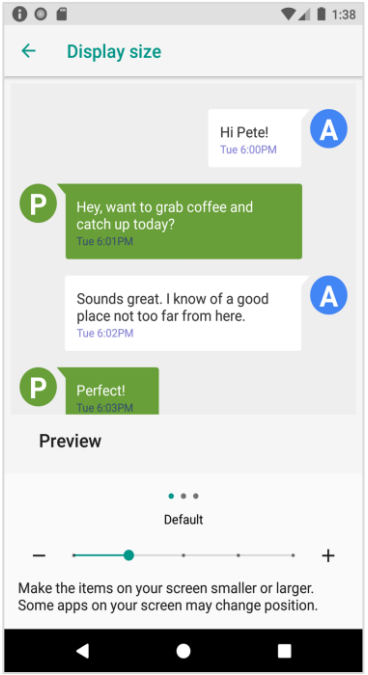
```
<TextView …
    android:layout_marginTop="@dimen/top_margin" />
```

Kiểu chữ

Pixel không phụ thuộc vào tỷ lệ (sp)

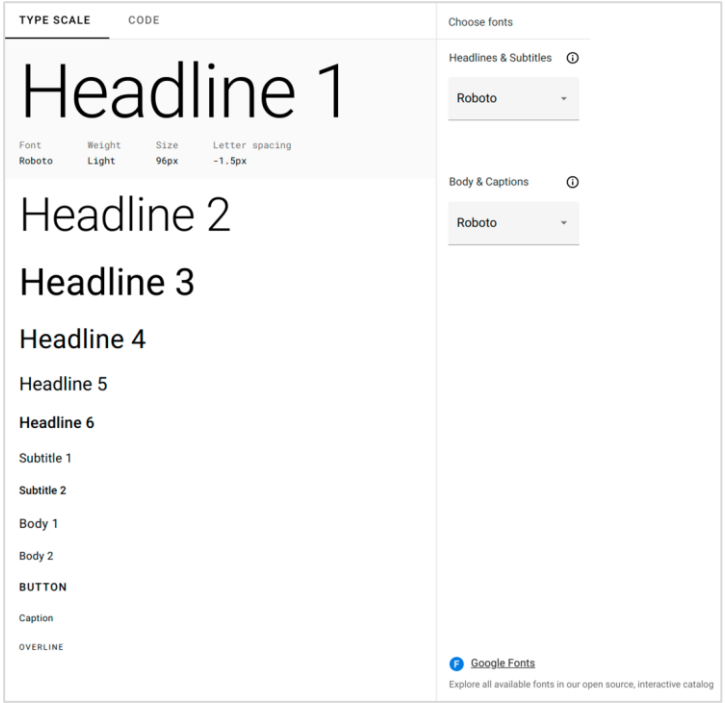
- Cấu trúc văn bản tương đương với pixel không phụ thuộc vào mật độ (dp)
- Chỉ định kích thước văn bản bằng sp (có tính đến lựa chọn ưu tiên của người dùng)
- Người dùng có thể điều chỉnh Kích thước phông chữ và Kích thước hiển thị trong ứng dụng Cài đặt (sau phần Màn hình)





Thang loại

- Là một nhóm kiểu được thiết kế để cùng hoạt động theo cách gắn kết cho ứng dụng và nội dung của bạn
- Chứa các danh mục văn bản có thể tái sử dụng với mục đích chủ định cho từng danh mục (ví dụ: tiêu đề, tiêu đề phụ, chú thích)



TextAppearance

Kiểu `TextAppearance` thường thay đổi một hoặc nhiều thuộc tính sau:

- kiểu chữ (`android:fontFamily`)
- độ đậm (`android:textStyle`)
- kích thước văn bản (`android:textSize`)
- viết hoa (`android:textAllCaps`)
- khoảng cách chữ cái (`android:letterSpacing`)

Các ví dụ sử dụng TextAppearance

```
<TextView
    ...
    android:textAppearance="@style/TextAppearance.MaterialComponents.Headline1"
    android:text="@string/title" />


<TextView
    ...
    android:textAppearance="@style/TextAppearance.MaterialComponents.Body1"
    android:text="@string/body_text" />
```

Tùy chỉnh TextAppearance theo cách của bạn

```
<style name="TextAppearance.MyApp.Headline1"
        parent="TextAppearance.MaterialComponents.Headline1">
    ...
    <item name="android:textStyle">normal</item>
    <item name="android:textAllCaps">false</item>
    <item name="android:textSize">64sp</item>
    <item name="android:letterSpacing">0</item>
    ...
</style>
```

Dùng TextAppearance tùy chỉnh trong một giao diện

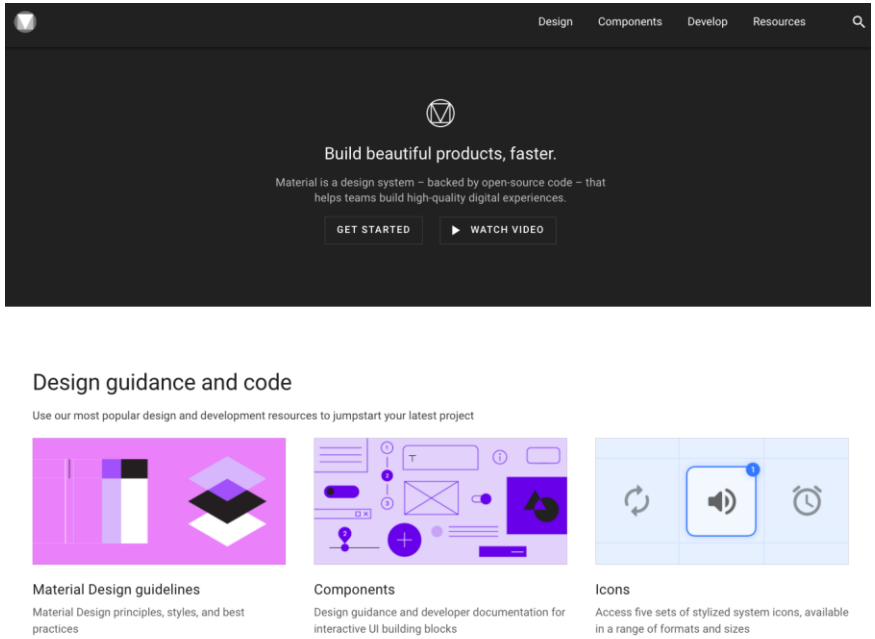
```
<style name="Theme.MyApp" parent="Theme.MaterialComponents.Light">
    ...
    <item name="textAppearanceHeadline1">@style/TextAppearance.MyApp.Headline1</item>
    ...
</style>
```

Material Design

Giới thiệu về Material

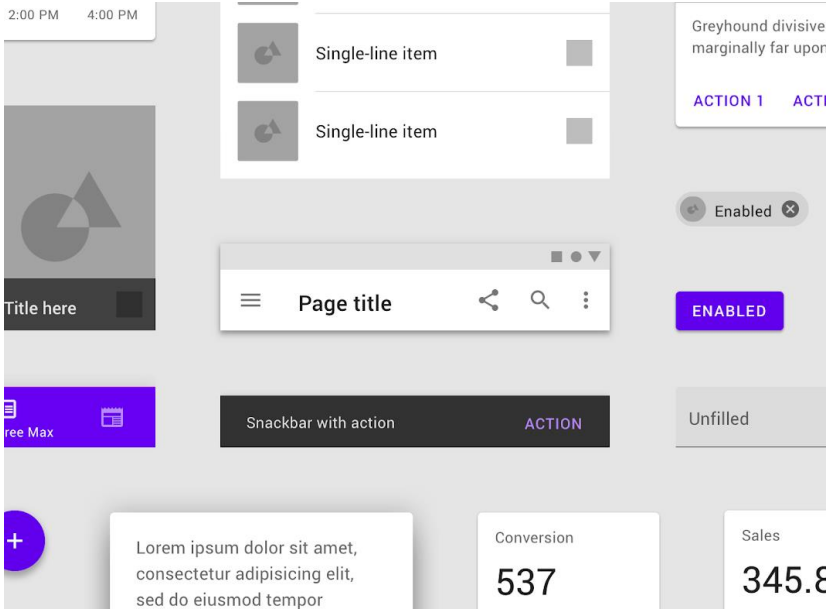
Là hệ thống các nguyên tắc, thành phần và công cụ mà bạn có thể tùy chỉnh theo ý mình để thiết kế giao diện người dùng

[Trang chủ Material Design]

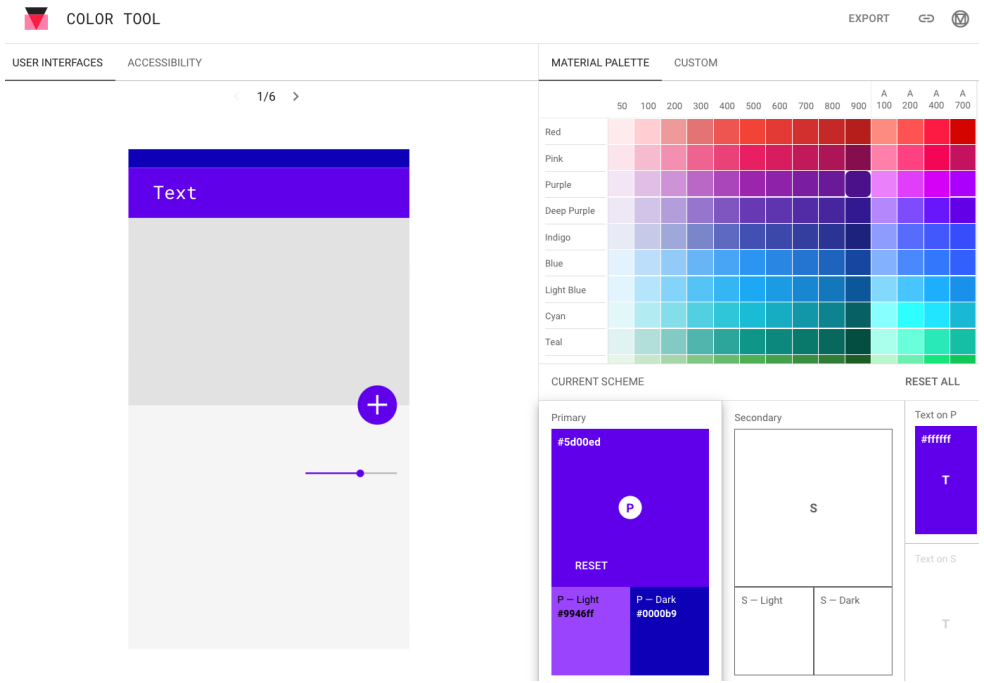


Thành phần Material

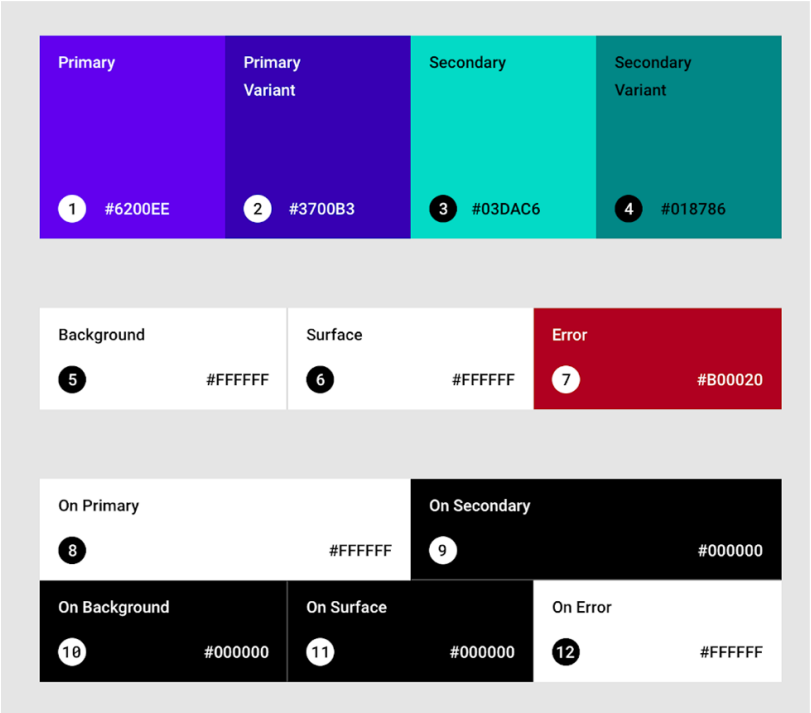
Là các khối xây dựng tương tác để tạo giao diện người dùng



Công cụ chọn màu Material



Giao diện màu Material cơ sở



Thư viện Thành phần Material cho Android

```
implementation 'com.google.android.material:material:<version>'
```

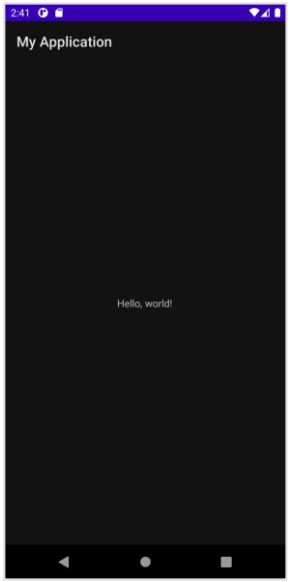
Giao diện Material

- `Theme.MaterialComponents`
- `Theme.MaterialComponents.NoActionBar`
- `Theme.MaterialComponents.Light`
- `Theme.MaterialComponents.Light.NoActionBar`
- `Theme.MaterialComponents.Light.DarkActionBar`
- `Theme.MaterialComponents.DayNight`
- `Theme.MaterialComponents.DayNight.NoActionBar`
- `Theme.MaterialComponents.DayNight.DarkActionBar`

Ví dụ về giao diện Material

`Theme.MaterialComponents.DayNight.DarkActionBar`

Chế độ sáng

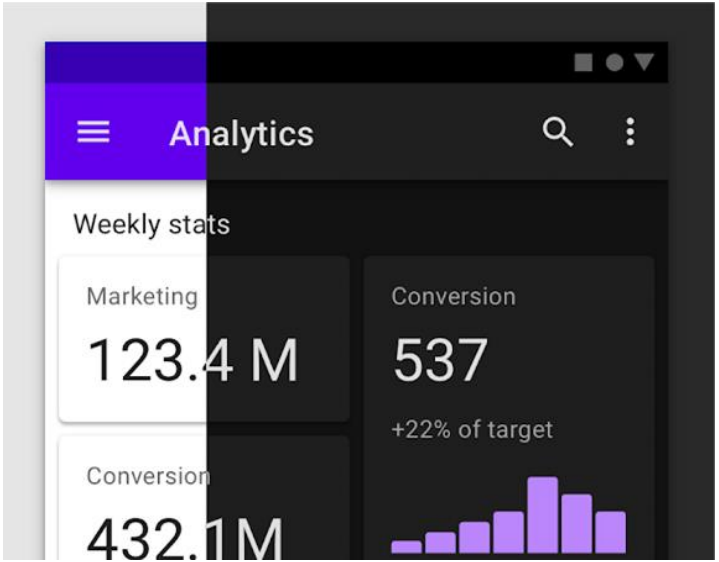


Chế độ tối

Giao diện tối

Là giao diện người dùng ở chế độ ánh sáng yếu hiển thị hầu hết các giao diện tối

- Thay thế các giao diện phủ màu sáng và văn bản tối bằng các giao diện phủ màu tối và văn bản sáng
- Giúp mọi người dễ dàng sử dụng thiết bị trong môi trường ánh sáng yếu
- Giúp người dùng có thị lực kém và những người nhạy cảm với ánh sáng mạnh dễ nhìn hơn
- Có thể giảm đáng kể mức sử dụng năng lượng (tùy vào thiết bị)



Hỗ trợ giao diện tối

Trong tệp `values/themes.xml`:

```
<style name="AppTheme" parent="Theme.MaterialComponents.DayNight">
    <item name="colorPrimary">@color/orange_500</item>
    ...
```

Trong tệp `values-night/themes.xml`:

```
<style name="AppTheme" parent="Theme.MaterialComponents.DayNight">
    <item name="colorPrimary">@color/orange_200</item>
    ...
```

Thành phần Material

Thành phần Material

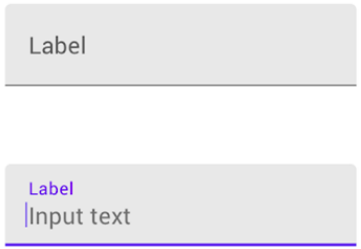
Thư viện thành phần được cung cấp cho Android và các nguyên tắc thiết kế

- Trường văn bản
- Nút
- Trình đơn
- Thẻ
- Khối
- Thanh ứng dụng (trên cùng và dưới cùng)
- Nút hành động nổi (FAB)
- Ngăn điều hướng
- Thanh điều hướng dưới cùng
- Thanh thông báo nhanh

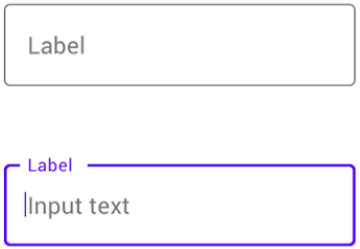
...và nhiều thành phần khác!

Trường văn bản

- Bao gồm `TextInputLayout` với chế độ xem con `TextInputEditText`
- Hiển thị nhãn nổi hoặc gợi ý văn bản trước khi người dùng nhập văn bản
- 2 loại:



Trường văn bản được tô màu nền



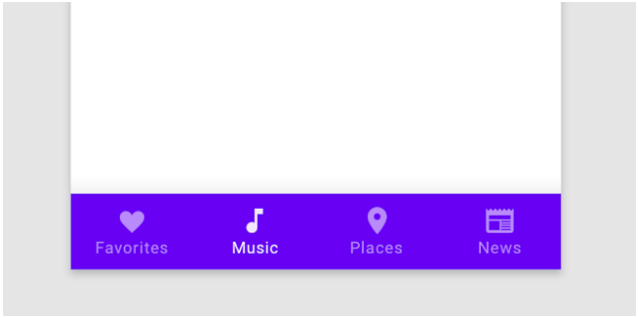
Trường văn bản có đường viền

Ví dụ về trường văn bản

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textField"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/label"
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</com.google.android.material.textfield.TextInputLayout>
```

Thanh điều hướng dưới cùng

- Cho phép di chuyển giữa các đích ở cấp cao nhất trong ứng dụng của bạn
- Mẫu thiết kế thay thế cho ngăn điều hướng
- Giới hạn ở tối đa 5 vị trí



Ví dụ về thanh điều hướng dưới cùng

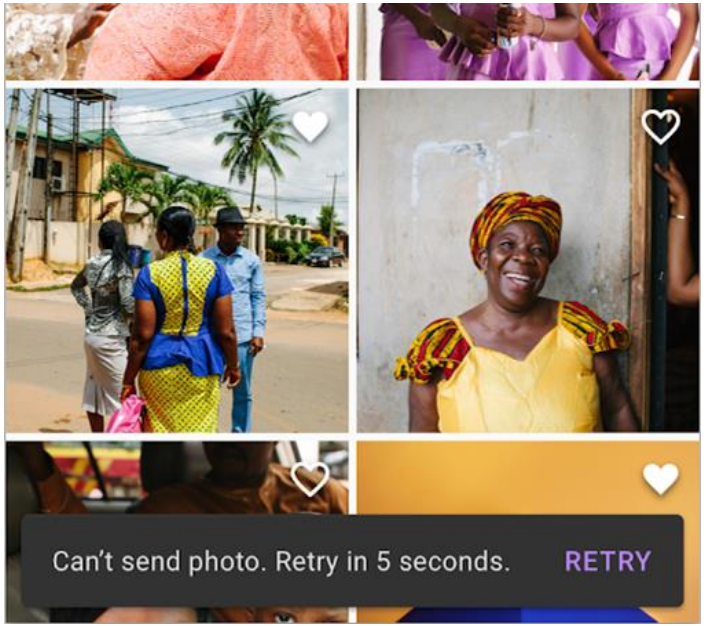
```
<LinearLayout …>

    ...

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_navigation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:menu="@menu/bottom_navigation_menu" />

</LinearLayout>
```

Trình xử lý thanh điều hướng dưới cùng

```
bottomNav.setOnNavigationItemSelectedListener { item ->
    when(item.itemId) {
        R.id.item1 -> {
            // Respond to navigation item 1 click
            true
        }
        R.id.item2 -> {
            true
        }
        else -> false
    }
}
```

Thanh thông báo nhanh

- Hiển thị các thông báo ngắn trong ứng dụng
- Các thông báo có một khoảng thời gian (`SHORT`, `LONG` hoặc `INDEFINITE`)
- Có thể chứa một thao tác không bắt buộc
- Hoạt động hiệu quả nhất trong `CoordinatorLayout`
- Hiển thị ở cuối vùng chứa bao quanh



Ví dụ về Thanh thông báo nhanh

Hiển thị một thông báo mẫu:

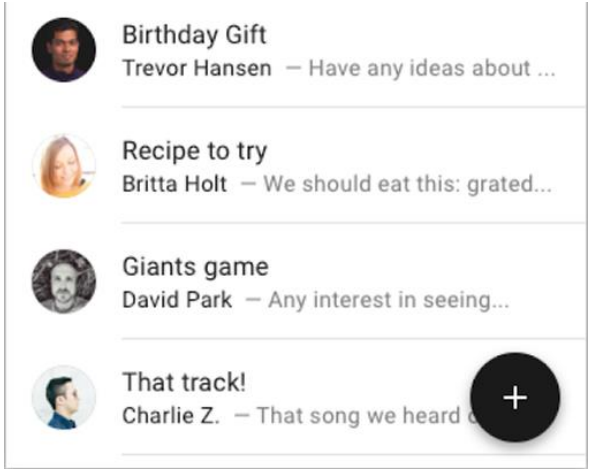
```
Snackbar.make(view, R.string.text_label, Snackbar.LENGTH_SHORT)
    .show()
```

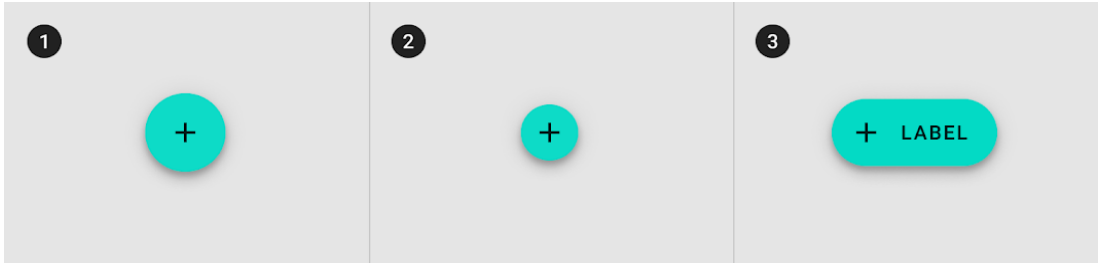
Thêm một thao tác vào Thanh thông báo nhanh:

```
Snackbar.make(view, R.string.text_label, Snackbar.LENGTH_LONG)
    .setAction(R.string.action_text) {
        // Responds to click on the action
    }
    .show()
```

Nút hành động nổi (FAB)

- Thực hiện thao tác phổ biến nhất của màn hình (ví dụ: tạo email mới)
- Có nhiều kích thước (thông thường, thu nhỏ và mở rộng)





CoordinatorLayout

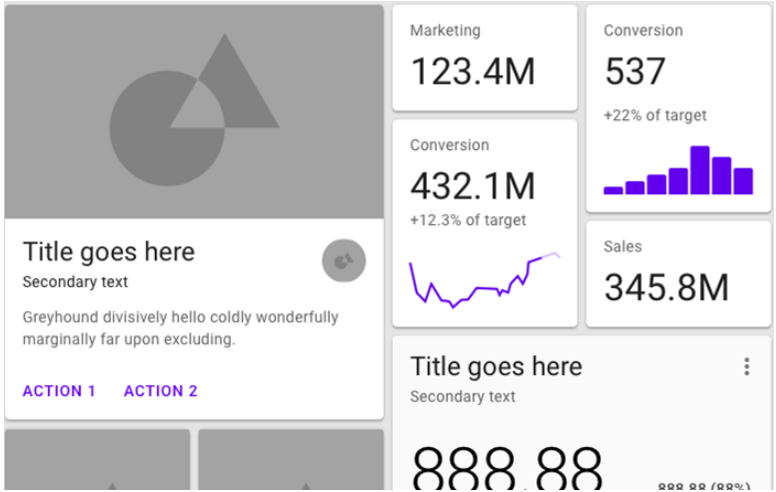
- Đóng vai trò là vùng chứa ở cấp cao nhất trong một ứng dụng
- Quản lý hoạt động tương tác của các chế độ xem con, chẳng hạn như cử chỉ
- Khuyên dùng với các chế độ xem như Thanh thông báo nhanh hoặc FAB

Ví dụ về FAB

```
<androidx.coordinatorlayout.widget.CoordinatorLayout ...>
    ....
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/floating_action_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        android:contentDescription="@string/fab_content_desc"
        app:fabSize="normal" <!-- or mini or auto -->
        app:srcCompat="@drawable/ic_plus"/>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Thẻ

- Là một thẻ lưu giữ nội dung và các thao tác cho một mục.
- Các thẻ thường được sắp xếp theo danh sách, lưới hoặc trang tổng quan.
- Dùng `MaterialCardView`.



Ví dụ về MaterialCardView

```
<com.google.android.material.card.MaterialCardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <ImageView .../>
        <TextView .../>
    </LinearLayout>

</com.google.android.material.card.MaterialCardView>
```

Bản địa hóa

Bản địa hóa ứng dụng

- Tách biệt các chương trình thành phần được bản địa hóa của ứng dụng (ví dụ: văn bản, tệp âm thanh, đơn vị tiền tệ và số) với chức năng Kotlin chính của ứng dụng nếu có thể.

  Ví dụ: Trích xuất các chuỗi mà người dùng thấy được ra tệp `strings.xml`.

- Khi người dùng chạy ứng dụng của bạn, hệ thống Android sẽ chọn những tài nguyên cần tải dựa trên ngôn ngữ của thiết bị.
- Nếu không tìm thấy tài nguyên dành riêng cho ngôn ngữ, thì Android sẽ dùng các tài nguyên mặc định mà bạn đã xác định làm giải pháp dự phòng.

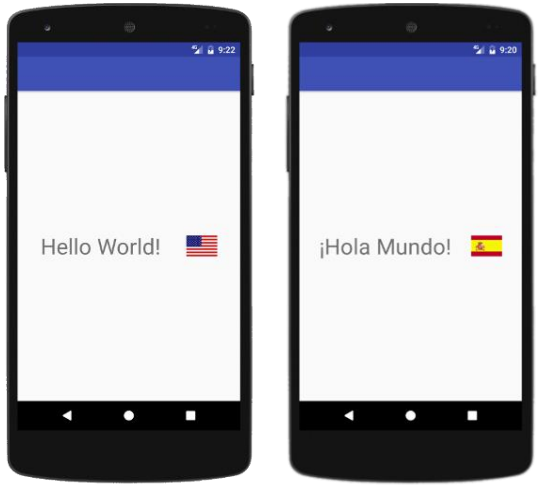
Hỗ trợ nhiều ngôn ngữ và văn hóa

- Chọn những ngôn ngữ cần hỗ trợ.
- Tạo các thư mục dành riêng cho ngôn ngữ trong thư mục `res`:

  `<resource type>-b+<language code>`
  `[+<country code>]`

  Ví dụ: `layout-b+en+US`
  `values-b+es`

- Cung cấp tài nguyên dành riêng cho ngôn ngữ (chẳng hạn như chuỗi và tài nguyên có thể vẽ) trong các thư mục đó.



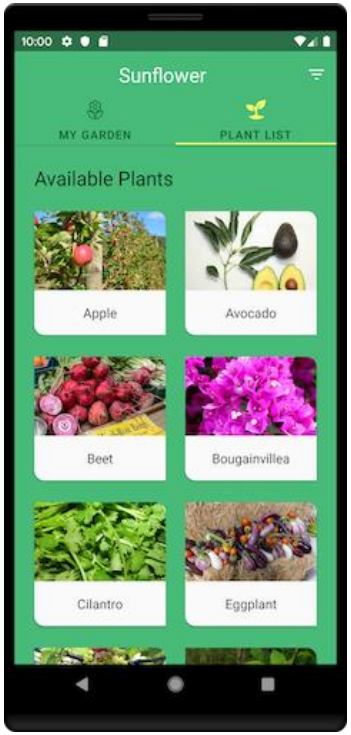
Hỗ trợ các ngôn ngữ sử dụng chữ viết từ phải sang trái (RTL)

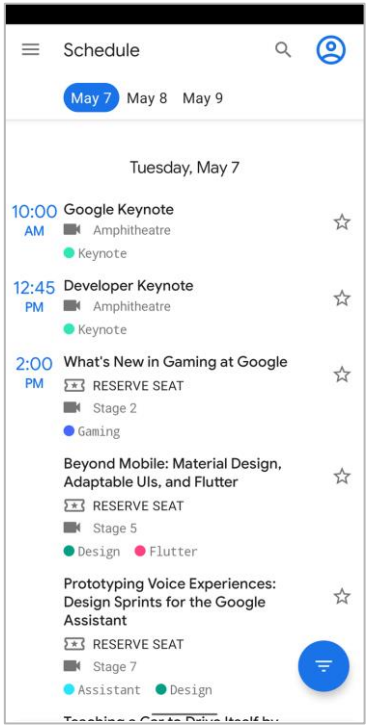
- Người dùng có thể chọn một ngôn ngữ sử dụng chữ viết từ phải sang trái (RTL).
- Thêm `android:supportsRtl="true"` vào thẻ ứng dụng trong tệp kê khai.
- Chuyển đổi `trái` và `phải` thành `đầu` và `cuối` tương ứng trong các tệp bố cục của bạn (thay đổi `android:paddingLeft` thành `android:paddingStart`).
- Bản địa hóa các chuỗi và định dạng văn bản trong các thông báo.
- Dùng bộ hạn định tài nguyên `-ldrtl` để cung cấp tài nguyên thay thế nếu muốn.

Ứng dụng mẫu

Xem các ứng dụng khác

Ứng dụng Sunflower





Ứng dụng Google I/O

Tóm tắt

Tóm tắt

Trong Bài học 13, bạn đã tìm hiểu cách:

- Tùy chỉnh giao diện hình ảnh của ứng dụng bằng cách dùng kiểu và giao diện
- Chọn trong số các thang loại định sẵn cho văn bản trong ứng dụng của bạn (hoặc tạo giao diện văn bản của riêng bạn)
- Chọn màu giao diện cho ứng dụng của bạn bằng công cụ chọn màu Material
- Dùng thư viện Thành phần Material để đẩy nhanh quá trình phát triển giao diện người dùng
- Bản địa hóa ứng dụng của bạn để hỗ trợ nhiều ngôn ngữ và văn hóa

Tìm hiểu thêm

- Material Design
- Thành phần Material
- Công cụ để chọn màu
- Giao diện tối
- Bản địa hóa ứng dụng
- Bài đăng trên blog: Giao diện và kiểu, Các thuộc tính giao diện phổ biến, Ưu tiên các thuộc tính giao diện, Lớp phủ giao diện
- Mã mẫu: Ứng dụng Sunflower, Ứng dụng Google I/O, Kho lưu trữ Android GitHub

Lộ trình

Thực hành những gì bạn đã học được bằng cách hoàn thành lộ trình này:

Bài học 13: Thiết kế giao diện người dùng của ứng dụng