ICS 26011: APPLICATIONS DEVELOPMENT AND EMERGING TECHNOLOGIES 3 (MOBILE PROGRAMMING)

MATERIAL DESIGN, STYLES AND THEMES

ALMA V. PEROL
Instructor
avperol@ust.edu.ph

Module Outline

- Material Design
- Principles of Material Design
- Components
- Theming
- Styles and Themes

What is Material Design?

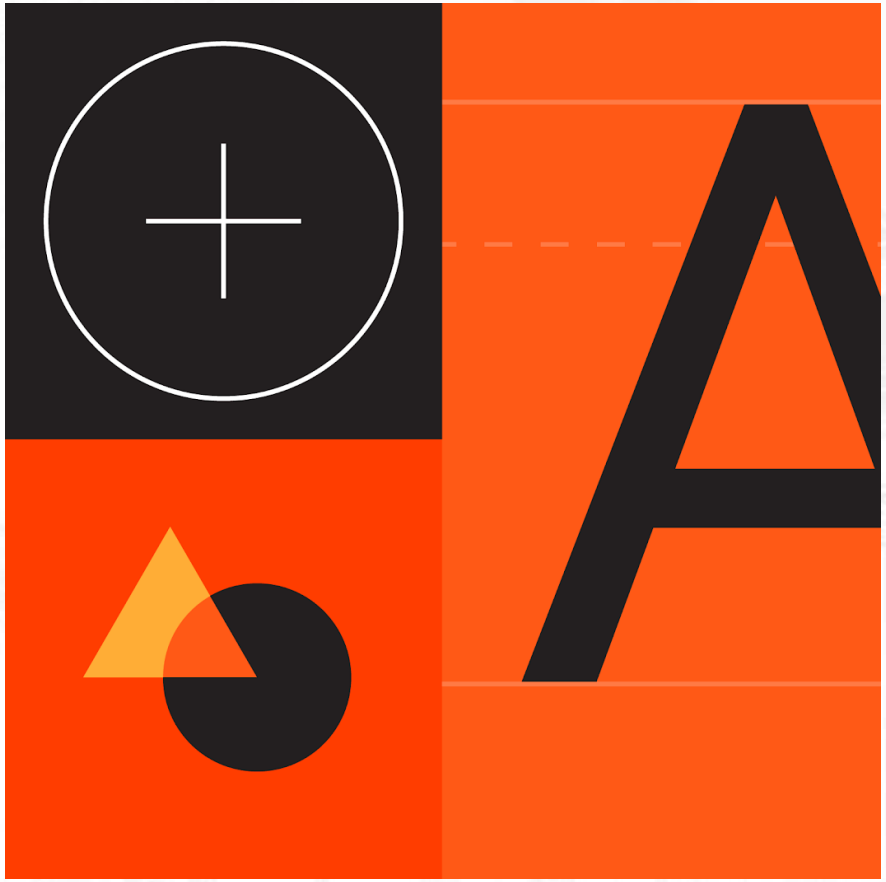
- **Material** is a design system created by Google to help teams build high-quality digital experiences for Android, iOS, Flutter, and the web.

Principles (Material is the Metaphor)

Material Design is inspired by the physical world and its textures, including how they reflect light and cast shadows. Material surfaces reimagine the mediums of paper and ink.

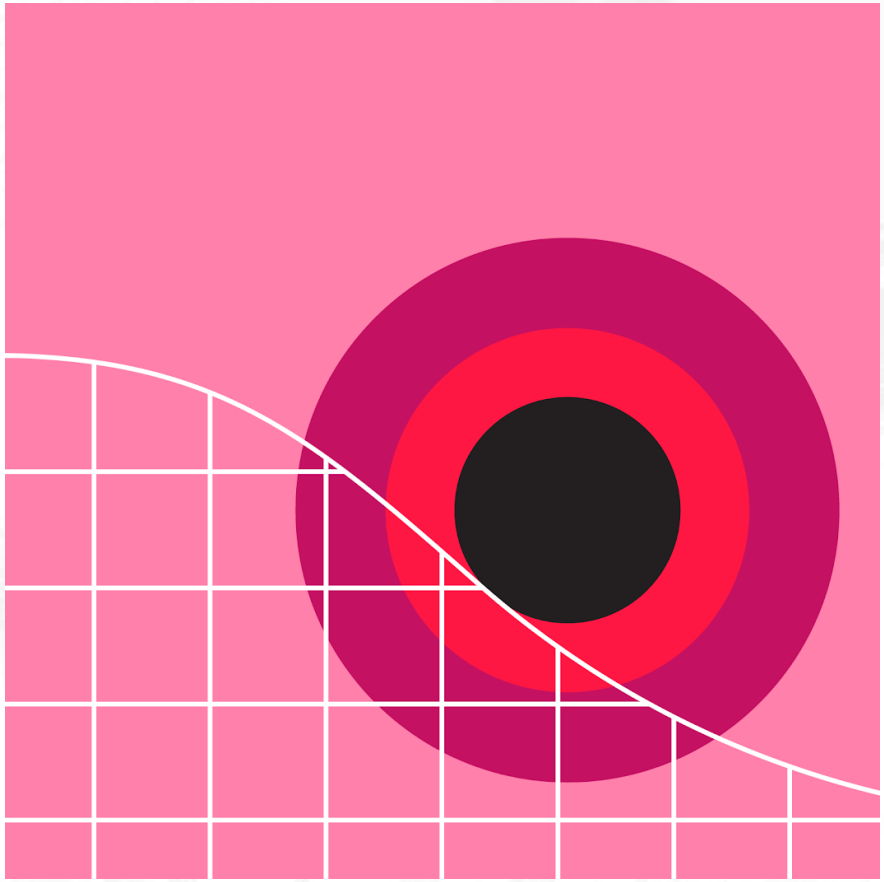
Principles (Bold, Graphic, Intentional)

Material Design is guided by print design methods — typography, grids, space, scale, color, and imagery — to create hierarchy, meaning, and focus that immerse viewers in the experience.



Principles (Motion Provides Meaning)

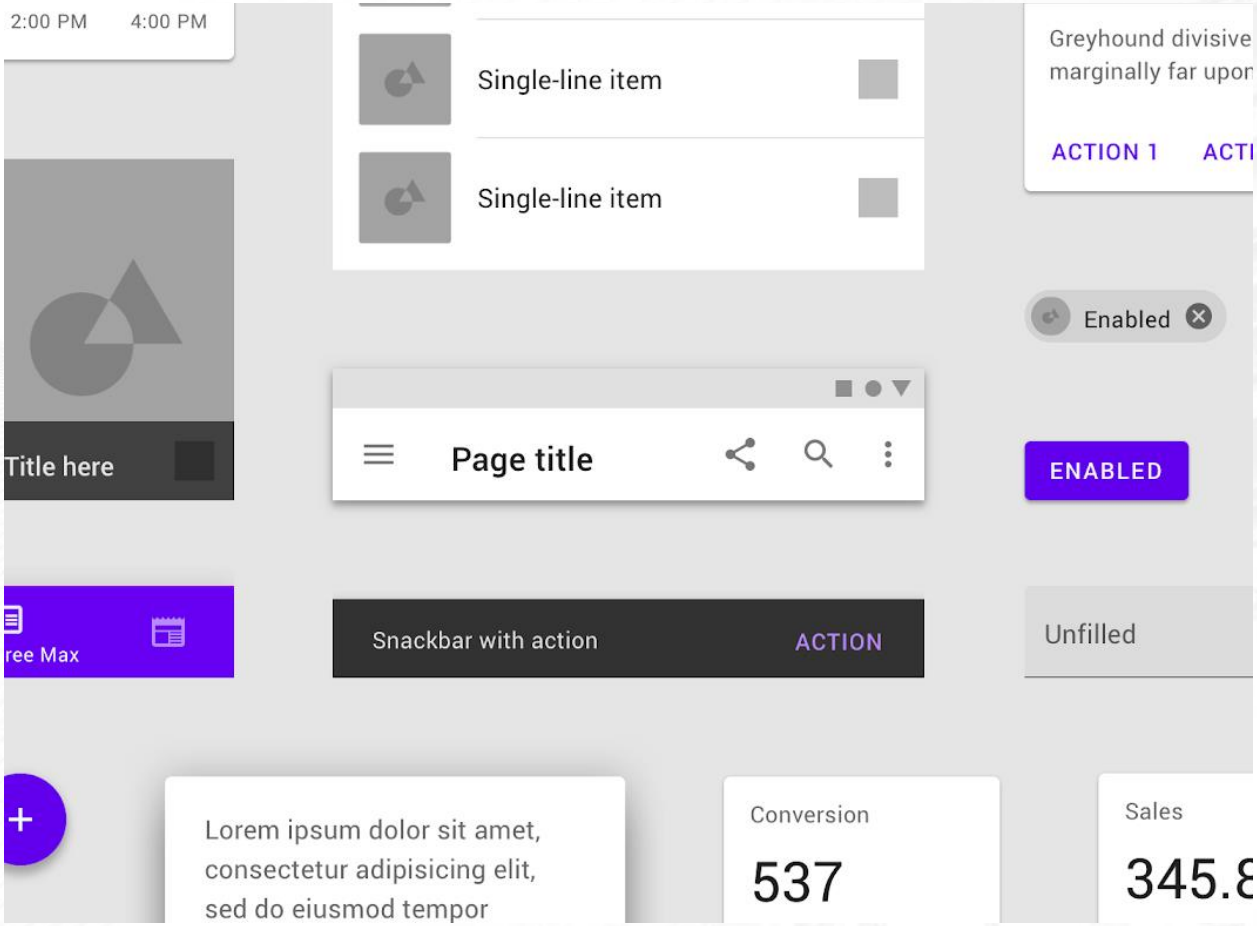
Motion focuses attention and maintains continuity through subtle feedback and coherent transitions. As elements appear on screen, they transform and reorganize the environment with interactions generating new transformations.



Components

- **Material Components** are interactive building blocks for creating a user interface, and include a built-in states system to communicate focus, selection, activation, error, hover, press, drag, and disabled states. Component libraries are available for Android, iOS, Flutter, and the web.

Components


2:00 PM
4:00 PM
Single-line item
Single-line item
Greyhound divisive
marginally far upon
ACTION 1
Enabled
Title here
Page title
ENABLED
ree Max
Snackbar with action
ACTION
Unfilled
Lorem ipsum dolor sit amet,
consectetur adipisicing elit,
sed do eiusmod tempor
Conversion
537
Sales
345.8

Components

Components cover a range of interface needs, including:

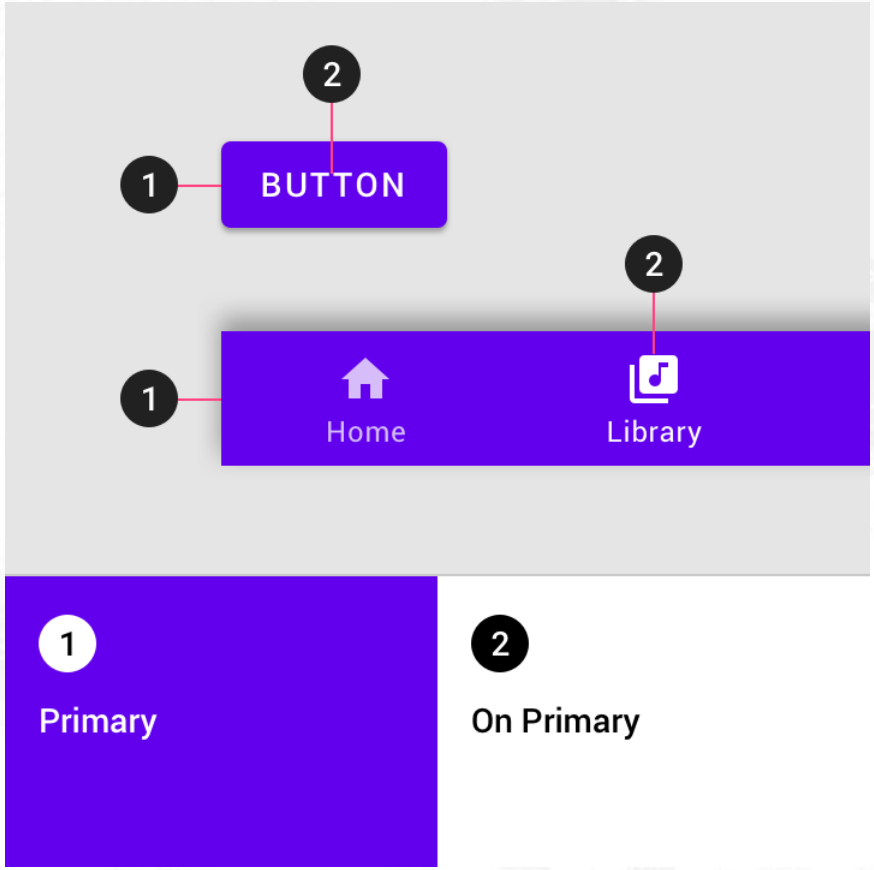
- **Display**: Placing and organizing content using components like cards, lists, and sheets.
- **Navigation**: Allowing users to move through the product using components like navigation drawers and tabs.
- **Actions**: Allowing users to perform tasks using components such as the floating action button.
- **Input**: Allowing users to enter information or make selections using components like text fields, chips, and selection controls.
- **Communication**: Alerting users to key information and messages using components such as snack bars, banners, and dialogs.

Theming

- **Material Theming** makes it easy to customize Material Design to match the look and feel of your brand, with built-in support and guidance for customizing colors, typography styles, and corner shapes.

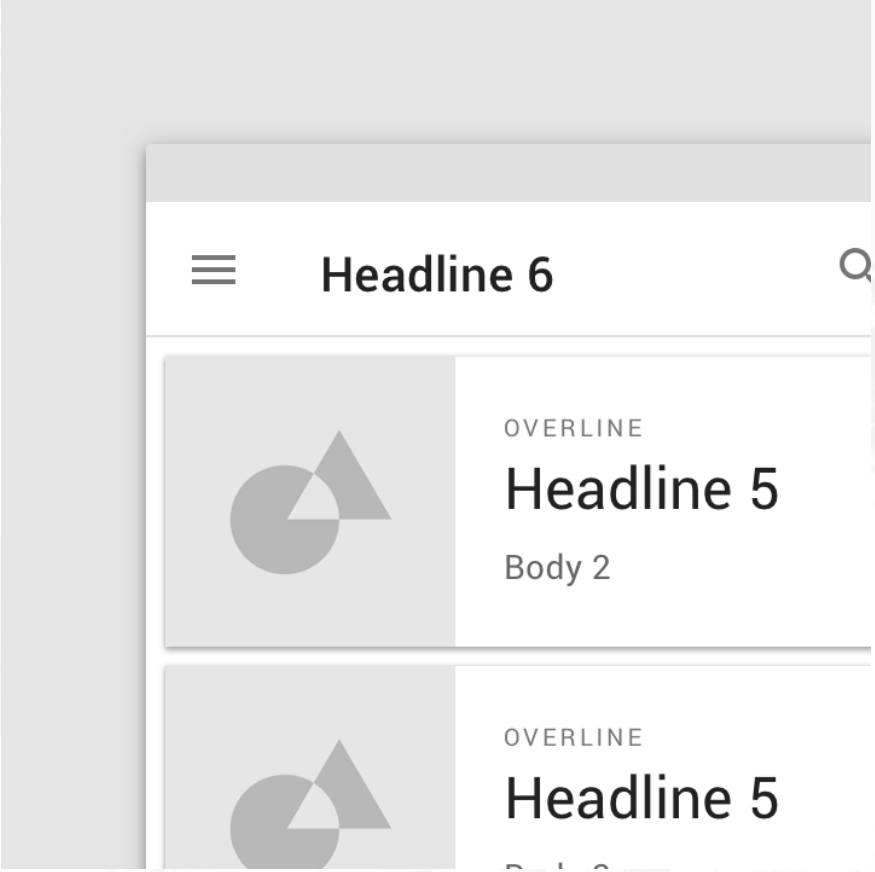
Colors

- Material's color system is an organized approach to applying color to a UI.
- Every color also has a complementary color used for elements placed "on" top of it to promote consistency and accessible contrast.



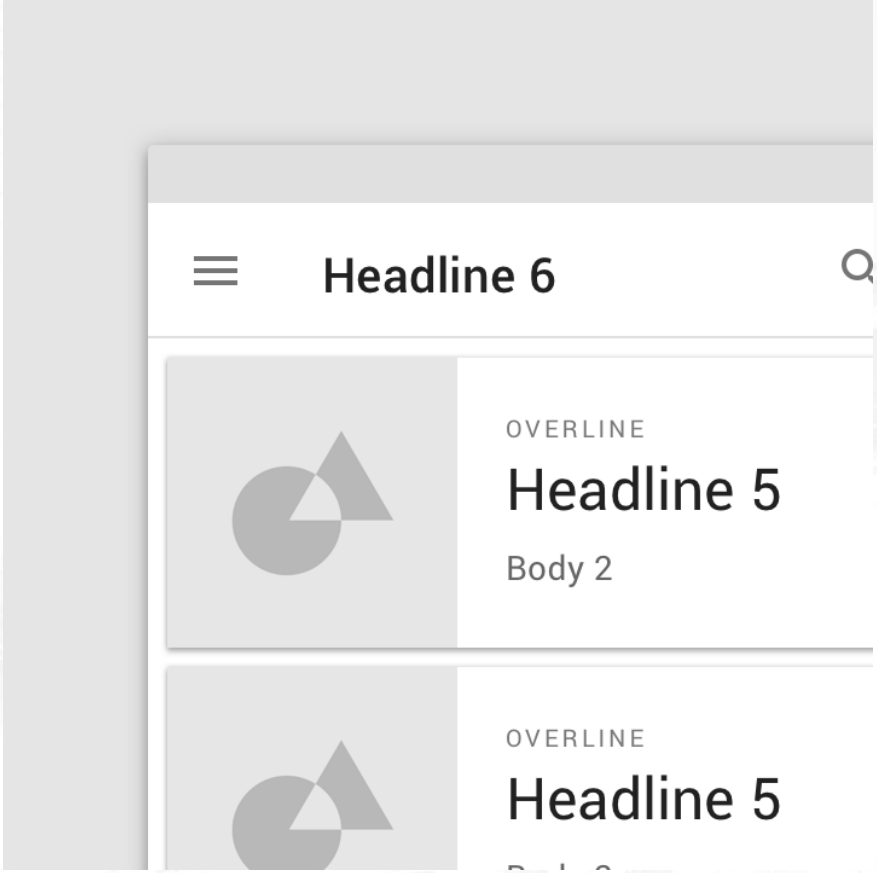
Typography

- The Material Design type scale provides 13 typography styles for everything from headlines to body text and captions. Each style has a clear meaning and intended application within an interface.
- Important attributes, such as the typeface, font weight, and letter case, can be modified to match your brand and design.



Shapes

- Applying shape styles can help direct attention or identify components, communicate their state, and express your brand.
- All Material Components are grouped into shape categories based on their size (small, medium, large).
- You can generate your own shape styles with the shape customization tool.



Styles and Themes

- In android, **Styles and Themes** are used to change the look and feel of Views and appearance of application based on our requirements. By using **Styles and Themes** we can reduce the code duplication and make our app light & responsive.
- Generally, the style is a combination of multiple attributes such as background color, font color, font size, font style, height, width, padding, margin, etc. and it is used to change the look and feel of View or window.

Styles and Themes

- In android, the style is defined in a separate XML resource file and we can use that defined style for the Views in XML that specifies the layout. The Styles in android are similar to CSS styles in web design.

```
<TextView
    android:id="@+id/txtResult"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="#86AD33"
    android:textSize="20dp"
    android:textStyle="bold" />
```

Styles and Themes

- Suppose if we use same or similar *TextView* styles in multiple places of our application, then the duplication of code will increase and in future, if we want to change the style of our application, we need to update the same style in all the places and it's a time-consuming process.
- We can overcome this problem by defining a style for the particular view and use the same style in all the places wherever it is required in our application.

Styles and Themes

- If we use style attribute we can move all style attributes related to particular view to a separate XML resource file and refer that file in XML layout like shown below:

```
<TextView
    android:id="@+id/txtResult"
    style="@style/TextviewStyle"/>
```

Styles and Themes

- If you observe above code snippet, we removed all style attributes from XML layout and moved those attributes to a style definition called TextviewStyle. In following sections we will see the definition of TextviewStyle attribute.

```
<TextView
    android:id="@+id/txtResult"
    style="@style/TextviewStyle"/>
```

Styles and Themes

- In android, **theme** is a **style** that is applied to an entire activity or app, instead of an individual View like as mentioned above. When we applied a **style** as a **theme**, the views in activity or app apply to the all style attributes that supports. For example. If we apply **TextviewStyle** as a **theme** for an activity, then the text of all the views in activity appears in the same style.

Styles and Themes

- To define a set of styles, we need to create a new XML file in **/res/values** directory of our project and the root node of XML file must be a **<resources>**.
- To create a **style** in the XML file, we need to follow the below steps.
  - We need to add **<style>** element in the XML file with a **name** attribute to uniquely identify the style.
  - To define attributes of style, we need to add an **<item>** elements with a name that defines a style attribute and we need to add appropriate value to each **<item>** element.

Styles and Themes

- In android, we can define multiple styles using **<style>** element with a **name** attribute to uniquely identify the style in an XML file.

```
<resources>
    <style name="TextviewStyle">
        <item name="android:textColor">#86AD33</item>
        <item name="android:textStyle">bold</item>
        <item name="android:textSize">20dp</item>
        <item name="android:padding">10dp</item>
    </style>
</resources>
```

Android Apply a Style to View

- Once we are done with the creation of style, we can use it for the views which are defined in the XML layout file with **style** attribute.

```
<TextView
    android:id="@+id/txtResult"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="@style/TextviewStyle"
    android:text="Welcome to Tutlane"/>
```

Android Style Inheritance

- In android, by using **parent** attribute in **<style>** element we can inherit the properties from an existing style and define only the attributes that we want to change or add. We can inherit the styles that we created ourselves or from the styles that are built into the platform.

```
<style name="TextviewStyle" parent="@android:style/Widget.TextView">
    <item name="android:textColor">#86AD33</item>
    <item name="android:textStyle">bold</item>
    <item name="android:textSize">20dp</item>
</style>
```

Android Defining Themes

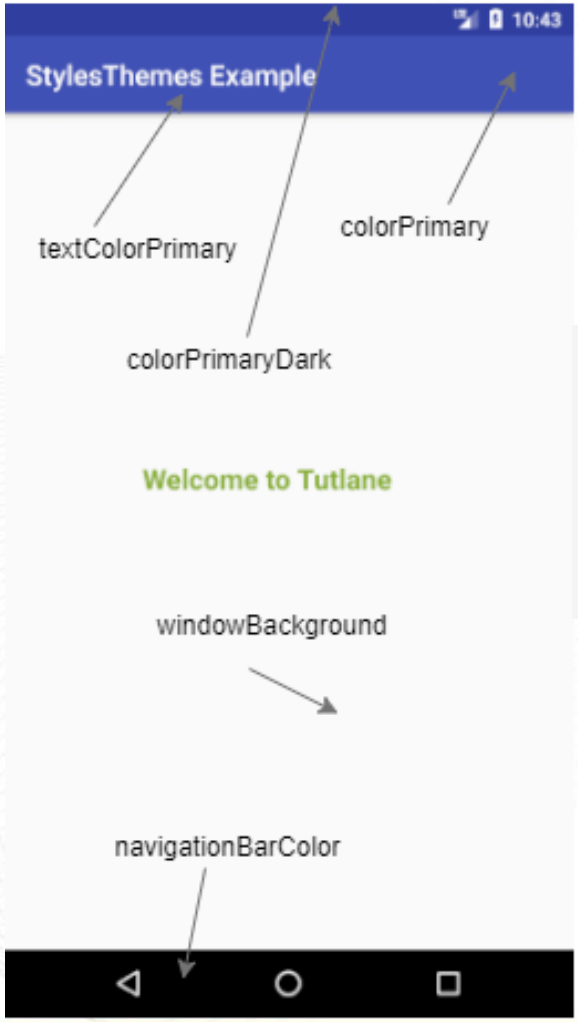
- In android, **theme** is a **style** that is applied to an entire activity or app, instead of an individual View like as mentioned above. When we applied a **style** as a **theme**, the views in activity or app apply to the all style attributes that supports. For example. If we apply **TextviewStyle** as a **theme** for an activity, then the text of all the views in activity appears in the same style.

```
<color name="custom_theme_color">#b0b0ff</color>
<style name="CustomTheme" parent="Theme.AppCompat.Light">
    <item name="android:windowBackground">@color/custom_theme_color</item>
    <item name="android:colorBackground">@color/custom_theme_color</item>
</style>
```

Android Defining Themes

- The above code overrides **windowBackground** and **colorBackground** properties of **Theme.AppCompat.Light** theme.

Android Styling Color Palette




```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

Android Themes Attributes

#	Name	Theme Attribute
1	Primary	`colorPrimary`
2	Primary Variant	`colorPrimaryVariant`
3	Secondary	`colorSecondary`
4	Secondary Variant	`colorSecondaryVariant`
5	Background	`colorBackground`
6	Surface	`colorSurface`
7	Error	`colorError`
8	On Primary	`colorOnPrimary`
9	On Secondary	`colorOnSecondary`
10	On Background	`colorOnBackground`
11	On Surface	`colorOnSurface`
12	On Error	`colorOnError`

Thank You!

Resources and Acknowledgements

- Material Design. (n.d). www.material.io
- Android Developer Fundamentals. https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/
- Tutlane. (n.d.). Android View and ViewGroup. https://www.tutlane.com/tutorial/android/