# Annexe 7C – Thème Material & Styles associés

Cette année, Android Studio utilise par défaut le thème Material3 comme base pour les thèmes ( polices, couleurs ) de nos applications.

Différence entre style et thème : Un **style** est un ensemble de propriétés (taille de texte, couleur, marges...) appliqué à un type spécifique de vue (TextView, Button, etc.). Un **thème** est un style global qui s'applique à toute l’application ou à une Activity, et qui peut définir des styles pour plusieurs composants à la fois via des rôles de couleur et des attributs de thème.

Regardez les deux fichiers themes.xml présents dans le dossier res/values/themes

Le thème de votre app porte habituellement le nom de votre app et a comme parent une des instances du thème Material.

1. Changer le parent à Theme.Material3.Dark et regardez le résultat.
2. On peut voir l’apparence de plusieurs thèmes parent dans le fichier de positionnement xml ( « theme for preview »)

## Les couleurs

Les couleurs énoncées se rapportent à des composantes ou des éléments de ces composantes. Il est utile de les définir dans ce fichier pour maintenir une cohérence à travers les activités d'une app plutôt qu'assigner une couleur à chaque composante.

Des rôles sont des couleurs qui s’assignent à un / plusieurs types de composantes

1. Définissez des nouvelles couleurs dans le fichier colors.xml et utilisez-les dans votre thème en assignant celles-ci dans les rôles colorPrimary et colorSecondary

Listez les changements : Cliquez ou appuyez ici pour entrer du texte.

Utilisez le rôle colorOnPrimary et regardez le résultat

Rôles des couleurs : [Color roles - Material Design 3](https://m3.material.io/styles/color/roles)

## Autres propriétés

On peut également redéfinir des éléments précis en rapport avec des composantes en utilisant la syntaxe suivante :

<item name="strokeWidth">12dp</item>  
<item name="strokeColor">@color/maCouleur</item>

1. Essayez les lignes de codes xml précédentes dans le style principal
2. Les TextViews ne sont pas impactés par le thème Material3. On doit donc créer un style propre pour les TextViews. On pourrait le faire également pour les SeekBars, les Buttons, etc.

Faire un style pour les TextViews

Dans le style principal, ajouter la ligne :

<item name="android:textViewStyle">@style/monStyle</item>

Et dans le fichier themes, créer le style monStyle qui va s’inspirer du style de base des MaterialTextView, qui lui s’inspire des AppCompatTextView :

<style name="monStyle" parent="Widget.MaterialComponents.TextView">  
 <!—ajouter des balises item pour modifier l’apparence des textViews   
  
</style>

## Comment trouver les propriétés qu'on peut modifier ?

Pour un TextView : <https://developer.android.com/reference/android/widget/TextView.html>

Pour un bouton / autres composants : : <https://github.com/material-components/material-components-android/blob/master/docs/components/Button.md>

Exercice : changer la couleur des textes sur les TextViews à l’intérieur du nouveau style

Un site très intéressant pour le thème Material : <https://material.io/blog/android-material-theme-color>

## Autres thèmes

* AppCompat : thème permettant d'être utilisé même sur des anciennes versions d'Android ( compatibilité ) . Il est plus " de base " que Material:
  + il permet de modifier l'apparence des composantes une-à-une (ex: les boutons sont gris par défaut, on peut changer leurs couleurs avec la propriété background )
  + Les effets de clic sur les composantes sont moins présents
  + Les couleurs sont "pleines" plutôt que composées de teintes en dégradé comme avec le thème Material

1. Utiliser le thème AppCompat dans votre fichier theme.xml en mettant l’actuel en commentaire

[Material Theme Builder](https://material-foundation.github.io/material-theme-builder/)

1. Différence entre style et thème :

Un **style** est un ensemble de propriétés (taille de texte, couleur, marges...) appliqué à un type spécifique de vue (TextView, Button, etc.). Un **thème** est un style global qui s'applique à toute l’application ou à une Activity, et qui peut définir des styles pour plusieurs composants à la fois via des rôles de couleur et des attributs de thème.

1. Exploration des thèmes :

Dans **res/values/themes/themes.xml** (mode clair) et **res/values-night/themes/themes.xml** (mode sombre) :

<resources>

<!-- Thème clair par défaut -->

<style name="AppTheme" parent="Theme.Material3.Light.NoActionBar">

<!-- Couleurs primaires et secondaires -->

<item name="colorPrimary">@color/primaryColor</item>

<item name="colorSecondary">@color/secondaryColor</item>

</style>

</resources>

Le thème de l’application (AppTheme) hérite d’un thème Material3.

1. Changer le parent à Theme.Material3.Dark :

Modifier parent :

<style name="AppTheme" parent="Theme.Material3.Dark.NoActionBar">

**Résultat :**

* Fond et surfaces en mode sombre.
* Textes et icônes en blanc pour un bon contraste.

**Liste des changements :**

* colorSurface : blanc → #121212 (sombre)
* colorOnSurface : noir → #FFFFFF
* colorBackground : blanc → #121212
* colorOnBackground : noir → #FFFFFF

1. Définition de nouvelles couleurs :

Dans **res/values/colors.xml** :

<resources>

<color name="primaryColor">#BB86FC</color>

<color name="primaryVariantColor">#3700B3</color>

<color name="secondaryColor">#03DAC6</color>

<color name="onPrimaryColor">#FFFFFF</color>

</resources>

Dans **themes.xml** :

<item name="colorPrimary">@color/primaryColor</item>

<item name="colorPrimaryVariant">@color/primaryVariantColor</item>

<item name="colorSecondary">@color/secondaryColor</item>

<item name="colorOnPrimary">@color/onPrimaryColor</item>

**Liste des changements :**

* colorPrimary : violet clair (#BB86FC) pour AppBar et FloatingActionButton.
* colorPrimaryVariant : violet foncé (#3700B3) pour états survolés.
* colorSecondary : turquoise (#03DAC6) pour accents.
* colorOnPrimary : blanc (#FFFFFF) pour textes/icônes sur surfaces primaires.

1. Propriétés additionnelles :

Ajouter dans le style principal :

<item name="strokeWidth">12dp</item>

<item name="strokeColor">@color/secondaryColor</item>

**Effet :** bordure de 12 dp turquoise sur les composants Material (exemple : Button Outlined).

1. Styles personnalisés pour TextView :

Dans le style principal :

<item name="android:textViewStyle">@style/MonTextViewStyle</item>

Créer **MonTextViewStyle** dans **styles.xml** :

<style name="MonTextViewStyle" parent="Widget.MaterialComponents.TextView">

<item name="android:textColor">@color/primaryColor</item>

<item name="android:textSize">18sp</item>

<item name="android:fontFamily">@font/roboto\_medium</item>

</style>

**Exercice :** changer la couleur des textes sur les TextViews à l’intérieur du style (android:textColor).

1. Recherche des propriétés modifiables :

* **TextView** : https://developer.android.com/reference/android/widget/TextView.html
* **Button** : https://github.com/material-components/material-components-android/blob/master/docs/components/Button.md

1. Autres thèmes :

* **AppCompat** : compatibilité pour anciennes versions, hérite de Theme.AppCompat.Light.NoActionBar.
* Personnalisation au coup par coup (background, ripple…) sans dégradés Material.

1. Outils complémentaires :

* **Material Theme Builder** : https://material.io/resources/theme-editor

© Éric Labonté, Cégep du Vieux Montréal