マテリアルデザインのカラーの決め方 その２

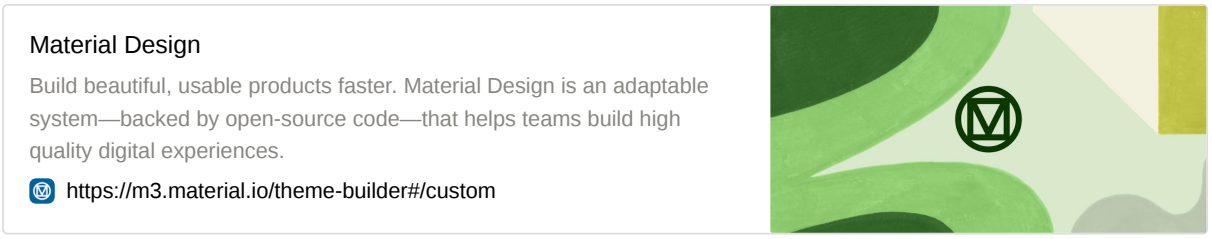
Stack	個人開発
Tag	
Created	@September 22, 2023 9:17 PM
Edited	@September 28, 2023 11:52 PM
Share to Web	☐

Jetpack Compose で使用する `ColorScheme` は以下のプロパティを持っている。これらにプロパティに対して明示的にカラーを設定してみる。

```
class ColorScheme(
    primary: Color,
    onPrimary: Color,
    primaryContainer: Color,
    onPrimaryContainer: Color,
    inversePrimary: Color,
    secondary: Color,
    onSecondary: Color,
    secondaryContainer: Color,
    onSecondaryContainer: Color,
    tertiary: Color,
    onTertiary: Color,
    tertiaryContainer: Color,
    onTertiaryContainer: Color,
    background: Color,
    onBackground: Color,
    surface: Color,
    onSurface: Color,
    surfaceVariant: Color,
    onSurfaceVariant: Color,
    surfaceTint: Color,
```

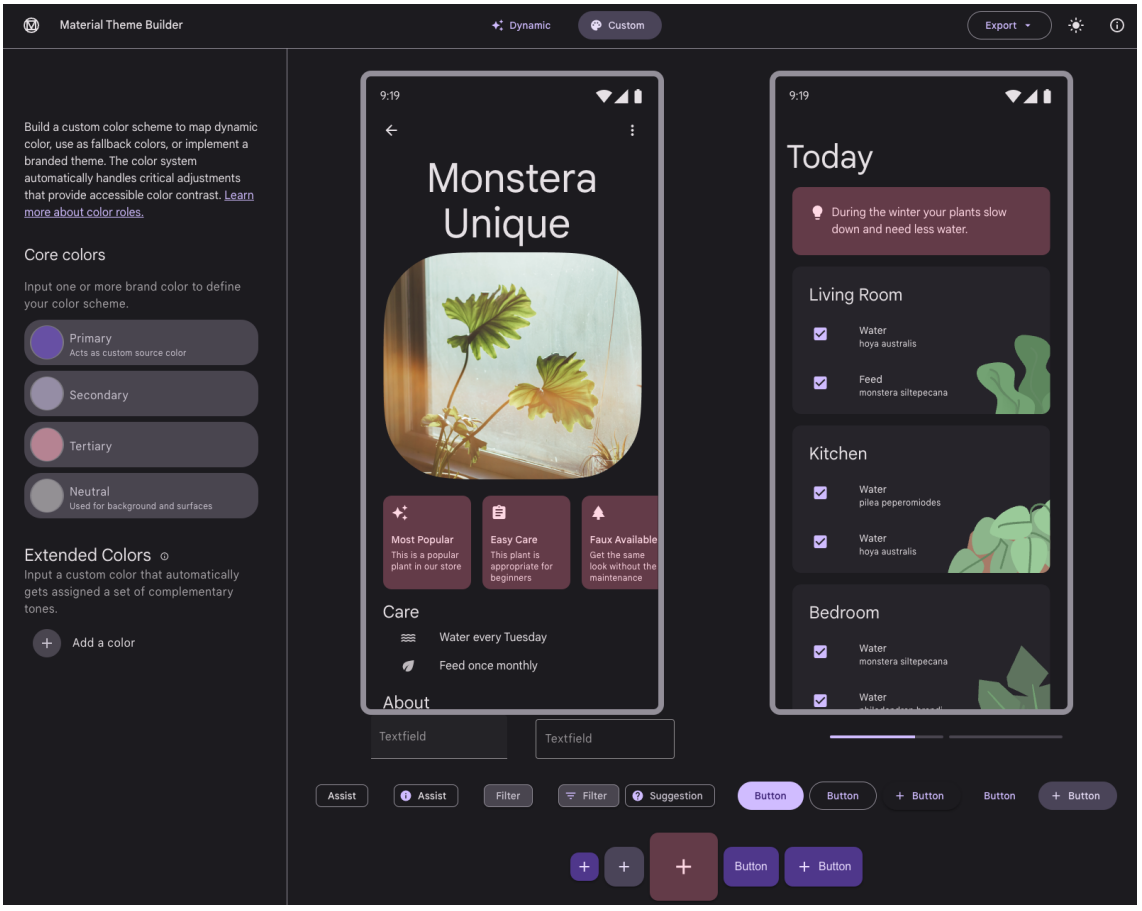
```
    inverseSurface: Color,
    inverseOnSurface: Color,
    error: Color,
    onError: Color,
    errorContainer: Color,
    onErrorContainer: Color,
    outline: Color,
    outlineVariant: Color,
    scrim: Color,
)
```

Material Theme Builder を使ってマテリアルデザインのカラーを決めてみる。



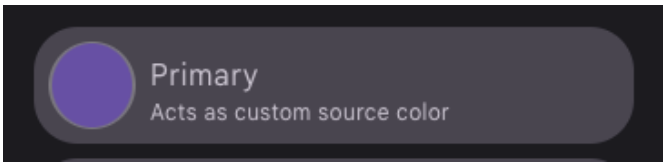
Material Theme Builder を開く

デフォルトでカラーを提案してくれている。

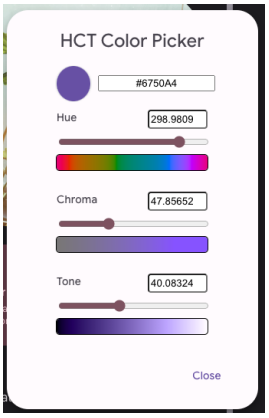


Primary カラーを変更する

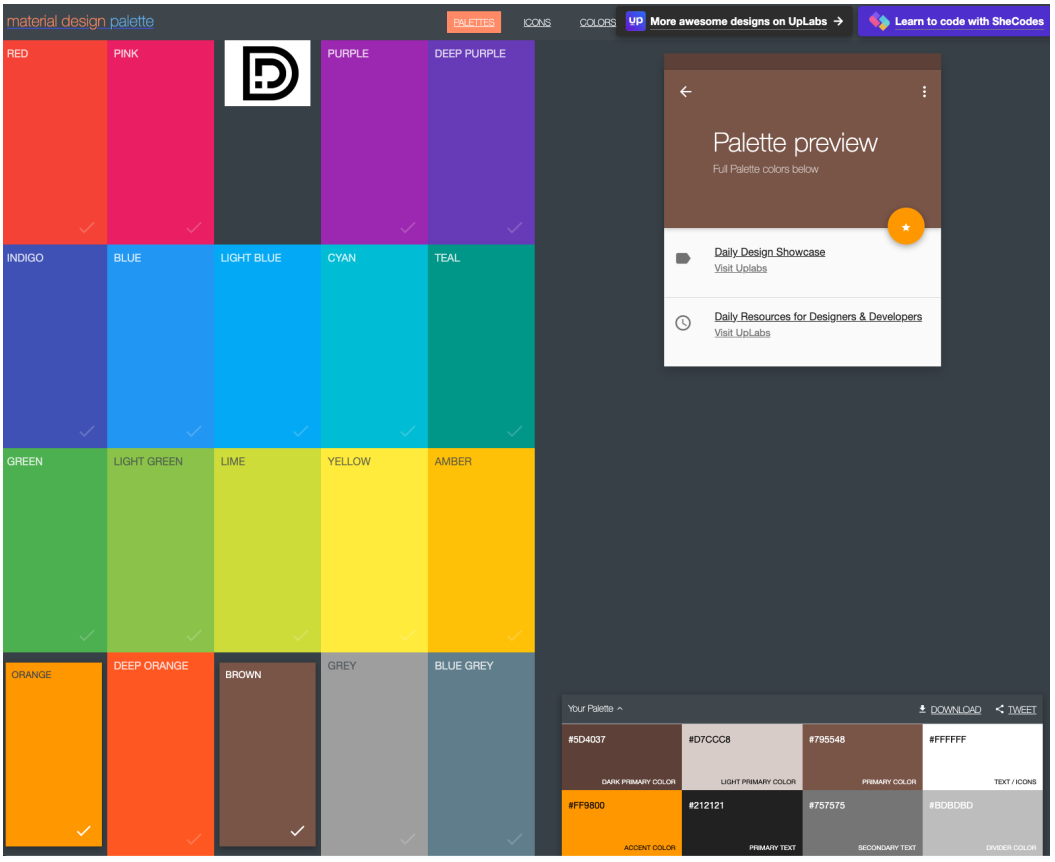
「Primary」の左の○をクリックする 。



すると以下の「HCT Color Picker」表示される。

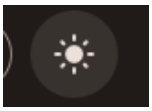


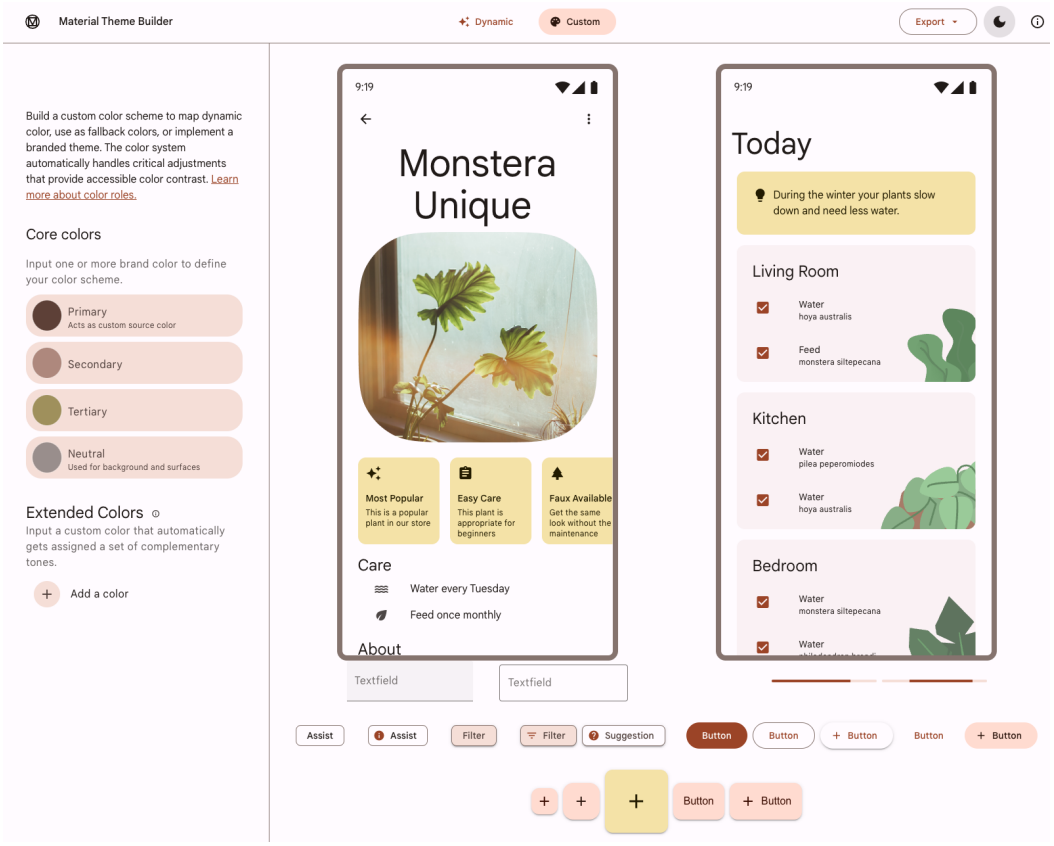
プライマリーカラーに指定するカラーを設定すれば良いのだが、どのカラーがプライマリーカラーとしてふさわしいのかさっぱりわからない。そこで material design palette を使う。material design palette はある程度色を絞ってくれているので選びやすい。例として「`#5D4037`」を使うことにする。



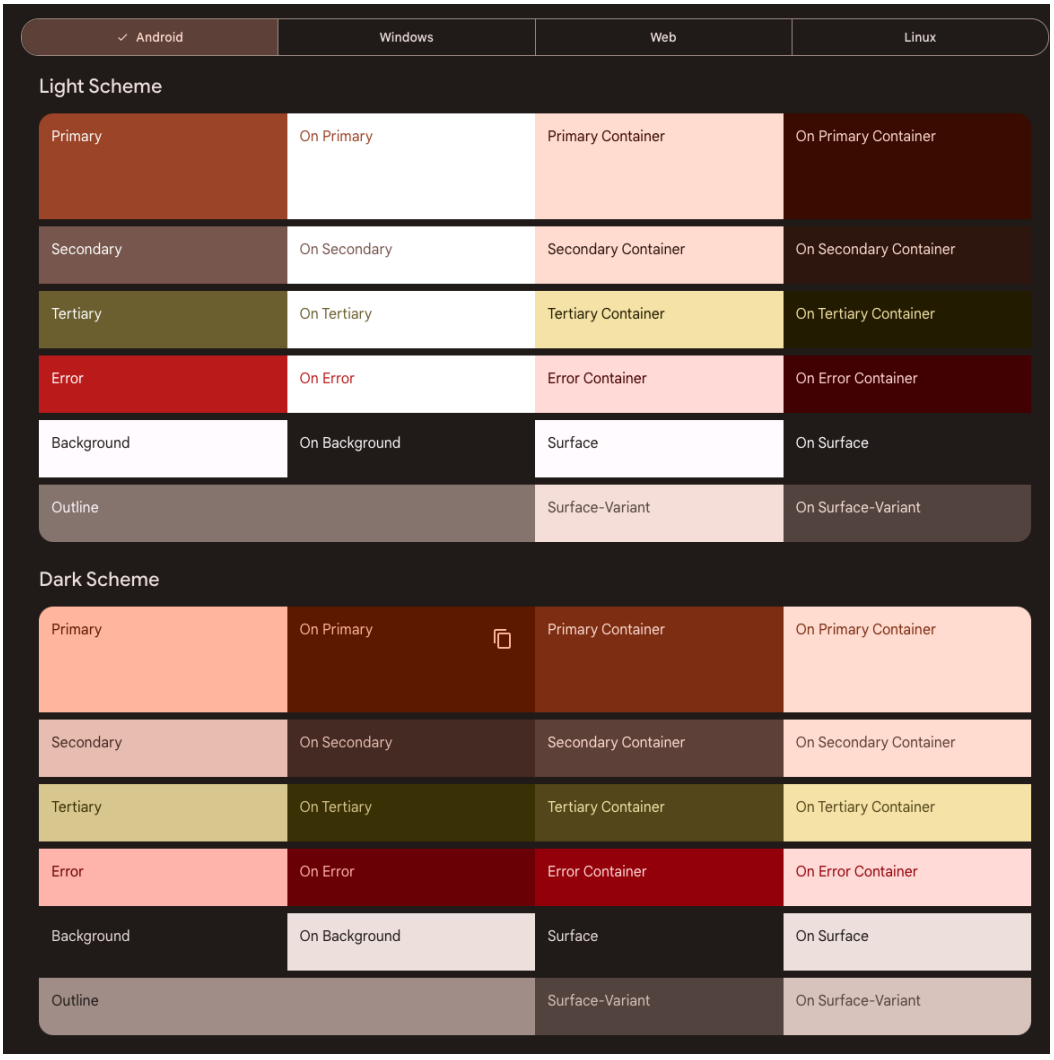
HCT Color Picker に「`#5D4037`」を入れると他のカラーを提案してくれる。

画面右上の以下をクリックすると画面が Light Mode に切り替わる。





画面をスクロールすると Light Scheme と Dark Scheme が表示される。



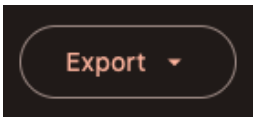
Android Studio で Jetpack Compose のプロジェクトを作った時に自動生成された Theme.kt を見てみる。

```
import androidx.compose.material3.darkColorScheme
import androidx.compose.material3.lightColorScheme

...
val DarkColorScheme = darkColorScheme(
    primary = Purple80,
    secondary = PurpleGrey80,
    tertiary = Pink80
)

val LightColorScheme = lightColorScheme(
    primary = Purple40,
    secondary = TestColor,
    tertiary = Pink40
)
```

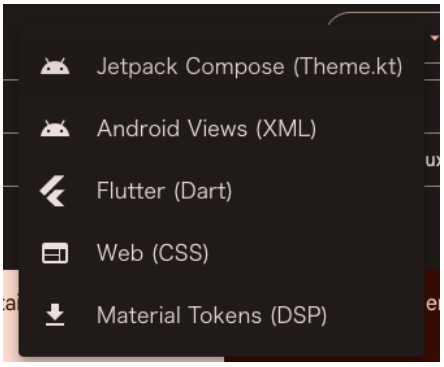
どうやら、提案された Light Scheme と Dark Scheme は `darkColorScheme()` と `lightColorScheme()` のコンストラクタに設定できそうである。

カラーコードを Export

画面右上の Export ボタンをクリックする。



続いて、「Jetpack Compose (Theme.kt)」 を選択する。material-theme.zip がダウンロードされる。



zip には Color.kt と Theme.kt が入っている。

```
import androidx.compose.ui.graphics.Color

val md_theme_light_primary = Color(0xFF9B4427)
val md_theme_light_onPrimary = Color(0xFFFFFFFF)
val md_theme_light_primaryContainer = Color(0xFFFFDBD0)
val md_theme_light_onPrimaryContainer = Color(0xFF3A0B00)
val md_theme_light_secondary = Color(0xFF77574D)
val md_theme_light_onSecondary = Color(0xFFFFFFFF)
val md_theme_light_secondaryContainer = Color(0xFFFFDBD0)
val md_theme_light_onSecondaryContainer = Color(0xFF2C160E)
val md_theme_light_tertiary = Color(0xFF6B5E2F)
val md_theme_light_onTertiary = Color(0xFFFFFFFF)
val md_theme_light_tertiaryContainer = Color(0xFFF4E2A7)
val md_theme_light_onTertiaryContainer = Color(0xFF221B00)
val md_theme_light_error = Color(0xFFBA1A1A)
val md_theme_light_errorContainer = Color(0xFFFFDAD6)
val md_theme_light_onError = Color(0xFFFFFFFF)
val md_theme_light_onErrorContainer = Color(0xFF410002)
val md_theme_light_background = Color(0xFFFFFBFF)
val md_theme_light_onBackground = Color(0xFF201A18)
val md_theme_light_surface = Color(0xFFFFFBFF)
val md_theme_light_onSurface = Color(0xFF201A18)
val md_theme_light_surfaceVariant = Color(0xFFF5DED7)
val md_theme_light_onSurfaceVariant = Color(0xFF53433F)
val md_theme_light_outline = Color(0xFF85736E)
val md_theme_light_inverseOnSurface = Color(0xFFFBEEEA)
val md_theme_light_inverseSurface = Color(0xFF362F2D)
val md_theme_light_inversePrimary = Color(0xFFFFB59E)
val md_theme_light_shadow = Color(0xFF000000)
val md_theme_light_surfaceTint = Color(0xFF9B4427)
```

```
val md_theme_light_outlineVariant = Color(0xFFD8C2BC)
val md_theme_light_scrim = Color(0xFF000000)

val md_theme_dark_primary = Color(0xFFFFB59E)
val md_theme_dark_onPrimary = Color(0xFF5D1800)
val md_theme_dark_primaryContainer = Color(0xFF7C2D12)
val md_theme_dark_onPrimaryContainer = Color(0xFFFFDBD0)
val md_theme_dark_secondary = Color(0xFFE7BDB1)
val md_theme_dark_onSecondary = Color(0xFF442A22)
val md_theme_dark_secondaryContainer = Color(0xFF5D4037)
val md_theme_dark_onSecondaryContainer = Color(0xFFFFDBD0)
val md_theme_dark_tertiary = Color(0xFFD7C68D)
val md_theme_dark_onTertiary = Color(0xFF3A3005)
val md_theme_dark_tertiaryContainer = Color(0xFF52461A)
val md_theme_dark_onTertiaryContainer = Color(0xFFF4E2A7)
val md_theme_dark_error = Color(0xFFFFB4AB)
val md_theme_dark_errorContainer = Color(0xFF93000A)
val md_theme_dark_onError = Color(0xFF690005)
val md_theme_dark_onErrorContainer = Color(0xFFFFDAD6)
val md_theme_dark_background = Color(0xFF201A18)
val md_theme_dark_onBackground = Color(0xFFEDE0DC)
val md_theme_dark_surface = Color(0xFF201A18)
val md_theme_dark_onSurface = Color(0xFFEDE0DC)
val md_theme_dark_surfaceVariant = Color(0xFF53433F)
val md_theme_dark_onSurfaceVariant = Color(0xFFD8C2BC)
val md_theme_dark_outline = Color(0xFFA08D87)
val md_theme_dark_inverseOnSurface = Color(0xFF201A18)
val md_theme_dark_inverseSurface = Color(0xFFEDE0DC)
val md_theme_dark_inversePrimary = Color(0xFF9B4427)
val md_theme_dark_shadow = Color(0xFF000000)
val md_theme_dark_surfaceTint = Color(0xFFFFB59E)
val md_theme_dark_outlineVariant = Color(0xFF53433F)
val md_theme_dark_scrim = Color(0xFF000000)
```

```
private val LightColors = lightColorScheme(
    primary = md_theme_light_primary,
    onPrimary = md_theme_light_onPrimary,
    primaryContainer = md_theme_light_primaryContainer,
    onPrimaryContainer = md_theme_light_onPrimaryContainer,
    secondary = md_theme_light_secondary,
    onSecondary = md_theme_light_onSecondary,
    secondaryContainer = md_theme_light_secondaryContainer,
    onSecondaryContainer = md_theme_light_onSecondaryContainer,
    tertiary = md_theme_light_tertiary,
    onTertiary = md_theme_light_onTertiary,
    tertiaryContainer = md_theme_light_tertiaryContainer,
    onTertiaryContainer = md_theme_light_onTertiaryContainer,
    error = md_theme_light_error,
    errorContainer = md_theme_light_errorContainer,
    onError = md_theme_light_onError,
    onErrorContainer = md_theme_light_onErrorContainer,
    background = md_theme_light_background,
    onBackground = md_theme_light_onBackground,
    surface = md_theme_light_surface,
    onSurface = md_theme_light_onSurface,
    surfaceVariant = md_theme_light_surfaceVariant,
    onSurfaceVariant = md_theme_light_onSurfaceVariant,
    outline = md_theme_light_outline,
    inverseOnSurface = md_theme_light_inverseOnSurface,
```

```
    inverseSurface = md_theme_light_inverseSurface,
    inversePrimary = md_theme_light_inversePrimary,
    surfaceTint = md_theme_light_surfaceTint,
    outlineVariant = md_theme_light_outlineVariant,
    scrim = md_theme_light_scrim,
)


private val DarkColors = darkColorScheme(
    primary = md_theme_dark_primary,
    onPrimary = md_theme_dark_onPrimary,
    primaryContainer = md_theme_dark_primaryContainer,
    onPrimaryContainer = md_theme_dark_onPrimaryContainer,
    secondary = md_theme_dark_secondary,
    onSecondary = md_theme_dark_onSecondary,
    secondaryContainer = md_theme_dark_secondaryContainer,
    onSecondaryContainer = md_theme_dark_onSecondaryContainer,
    tertiary = md_theme_dark_tertiary,
    onTertiary = md_theme_dark_onTertiary,
    tertiaryContainer = md_theme_dark_tertiaryContainer,
    onTertiaryContainer = md_theme_dark_onTertiaryContainer,
    error = md_theme_dark_error,
    errorContainer = md_theme_dark_errorContainer,
    onError = md_theme_dark_onError,
    onErrorContainer = md_theme_dark_onErrorContainer,
    background = md_theme_dark_background,
    onBackground = md_theme_dark_onBackground,
    surface = md_theme_dark_surface,
    onSurface = md_theme_dark_onSurface,
    surfaceVariant = md_theme_dark_surfaceVariant,
    onSurfaceVariant = md_theme_dark_onSurfaceVariant,
    outline = md_theme_dark_outline,
    inverseOnSurface = md_theme_dark_inverseOnSurface,
    inverseSurface = md_theme_dark_inverseSurface,
    inversePrimary = md_theme_dark_inversePrimary,
    surfaceTint = md_theme_dark_surfaceTint,
    outlineVariant = md_theme_dark_outlineVariant,
    scrim = md_theme_dark_scrim,
)
```

上記はそのまま使う。

テーマにカラーパレットを適用する

これをテーマに適用すれば完成。

```
import androidx.compose.material3.MaterialTheme

@Composable
fun ComposePdfViewerTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    // Dynamic color is available on Android 12+
    dynamicColor: Boolean = true,
    content: @Composable () -> Unit
) {
    val colorScheme = when {
        // NOTE: プロジェクト作成時に自動生成された以下のコードにより、 Android S(31) 以上は
        //       システムによってカラーが決められてしまう。開発者がカラーを制御したい場合は無効にする。
```

```
//        dynamicColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
//            val context = LocalContext.current
//            if (darkTheme) dynamicDarkColorScheme(context) else dynamicLightColorScheme(context)
//        }

        darkTheme -> DarkColors
        else -> LightColors
    }
    val colorScheme = LightColors
...
    MaterialTheme(
        colorScheme = colorScheme,
        typography = Typography,
        content = content
    )
}
```

設定したカラーパレットを直接使用する例。

```
import androidx.compose.material3.MaterialTheme

@Composable
fun PdfViewerAppBar(currentScreen: Screen) {
    TopAppBar(
        title = {
            Text(
                text = stringResource(id = currentScreen.appBarTitleResId),
                color = MaterialTheme.colorScheme.onPrimary
            )
        },
        backgroundColor = MaterialTheme.colorScheme.secondary
    )
}
```

【注意】material3 の `MaterialTheme` を使用すること！！