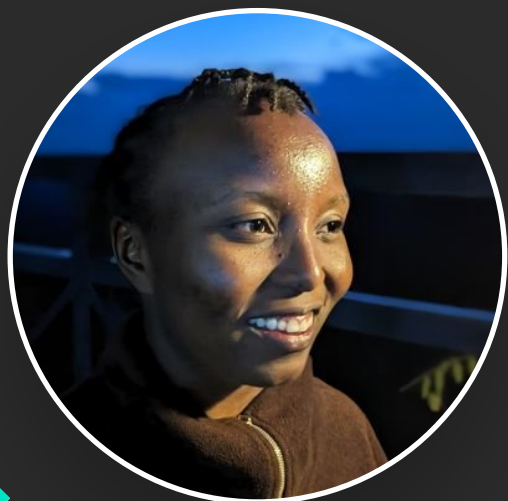




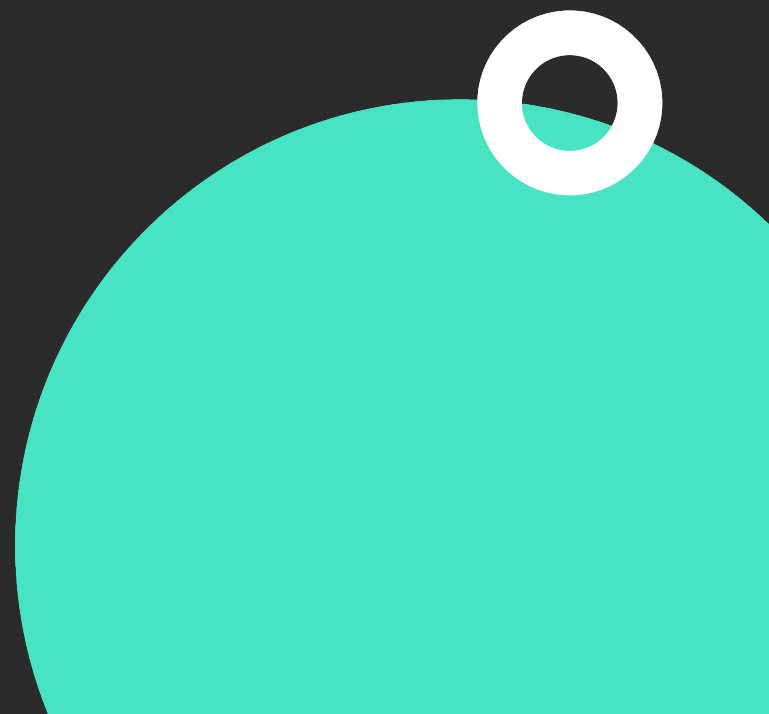
Coil Library: Guide to Handling Images in Android Jetpack Compose



Beatrice Kinya

 @B__Kinya

 Beatrice Kinya



What is Coil?

- Coil is an image loading library, backed by Kotlin Coroutines
- It handles loading images from the internet
- It can also cache images so that you do not have to download images several times





Time to Write Code

Up Next



Subcomposition

- Subcomposition changes the flow of compose layout.
- Instead of composing its children in composition phase, it composes in layout phase.
- This makes subcomposition less performant compared to regular composition.
- Hence, use subcomposition only when you have to and in the parts of the UI where performance is not very critical
- Check out this article to learn more about subcomposition: <https://kinya.hashnode.dev/custom-compose-layouts-clf5ua9jw01mms1nv5k6d84zp>



AsyncImagePainter

- **AsyncImage** and **SubcomposeAsyncImage** uses **AsyncImagePainter** under the hood.
- **AsyncImagePainter** is a **Painter** that executes ImageRequest asynchronously and renders result.
- If you want to use the Painter but cannot use AsyncImage, for instance in advanced animation, you'd use **AsyncImagePainter**.





Time to Write More Code 🧑💻 🧑💻

Up Next



AsyncImagePainter Cont'

- An image request needs size to determine the output image dimensions, i.e width and height.
- Compose resolves size during layout phase. Check Jetpack Compose docs to learn more about compose phases: <https://developer.android.com/jetpack/compose/layouts/basics>.
- Therefore, **AsyncImagePainter.State** will be loading in the initial composition even if the image is already in memory.
- To ensure **AsyncImagePainter.State** is up-to-date in the first composition, set the image size on the image request or use **SubcomposeAsyncImage**.



AsyncImagePainter Cont'

- **AsyncImagePainter** will not finish loading if **AsyncImagePainter.onDraw** is not called
- This can occur if a composable has unbounded width and height constraints like **LazyColumn** or **LazyRow**.
- To use **AsyncImagePainter** with either of lazy layouts you can either:
 - Set bounded width or height respectively or
 - Set image size on the image request.



Set Bounded Width or Height on the Image

```
@Composable
fun ImagesList() {
    LazyColumn() {
        items(3) {
            val painter =
                rememberAsyncImagePainter(model = "<image_url>")
            Image(
                modifier = Modifier.width(300.dp).height(200.dp),
                painter = painter, contentDescription = "Top breed")
        }
    }
}
```



Set Image Size on ImageRequest

```
@Composable
fun ImagesList() {
    LazyColumn() {
        items(3) {
            val imageRequest = ImageRequest.Builder(LocalContext.current)
                .data("<Image Url>")
                .size(Size.ORIGINAL)
                .build()
            val painter =
                rememberAsyncImagePainter(model = imageRequest)
            Image(
                painter = painter, contentDescription = "Top breed")
        }
    }
}
```



Summary

- Using **AsyncImage** to load images
- **SubcomposeAsyncImage**
- **AsyncImagePainter** an alternative to **AsyncImage** for loading images
- Transitions

