# STORING DATA PERMANENTLY - CHEATSHEET

## Adding Dependencies: Storing Data

### Dependencies:

```
val nav_version = "2.7.5"
val compose_version = "1.0.6-alpha08"
val room_version = "2.6.0"
implementation("androidx.room:room-
runtime:$room_version")
implementation("androidx.room:room-
ktx:$room_version")
kapt("androidx.room:room-
compiler:$room_version")
```

### Storage Options:

**SQLite:**
Structured data in a private database.

**Room:**
Abstraction over SQLite.

**Shared Preferences:**
Key-value pairs.

**DataStore:**
Asynchronous key-value storage.

## Hexadecimal and Colors

### Hex Color Codes:

```
<color name="my_red">#FF0000</color>
```

### Usage in Kotlin:

```
val myRed = Color(0xFFFF0000.toInt())
```

### Accessing Colors:

```
Text(
text = "Sample Text",
color = colorResource(id =
R.color.my_custom_color)
)
```

## Card Layout in Jetpack Compose

### Example:

```
Card(
modifier = Modifier
.fillMaxWidth()
.padding(16.dp)
.clickable { /* Handle card click */ },
elevation = 4.dp,
shape = RoundedCornerShape(8.dp)
) {
Column(modifier =
Modifier.padding(16.dp)) {
Text("Card Title", style =
MaterialTheme.typograph
y.h6)
Text("Card Content", style =
MaterialTheme.typogra
phy.body2)
}
}
```

## Kotlin-KAPT Plugin

### Setup:

```
plugins {
id("kotlin-kapt")
}
```

### Uses:

- Processes annotations.
- Generates code at compile-time.
- Validates code before compilation.

## Modifier.heightIn and Elevation

### Example:

```
Modifier.heightIn(min = 100.dp, max =
200.dp).elevation(3.
dp)
```

## Navigation Icon in Jetpack Compose

### Example:

```
TopAppBar(
title = { Text("Title") },
navigationIcon = {
IconButton(onClick = {
navController.navigateUp()
}) {
Icon(Icons.Filled.ArrowBack,
contentDescriptio
n = "Go back")
}
}
)
```

## Keyboard Options in Jetpack Compose

### Example:

```
TextField(
value = textValue,
onValueChange = { textValue = it },
label = { Text("Enter Text") },
keyboardOptions = KeyboardOptions(
keyboardType = KeyboardType.Text,
imeAction = ImeAction.Done
),
keyboardActions = KeyboardActions(
onDone = { /* Define action */ }
)
)
```

## Scaffold in Jetpack Compose

### Usage:

```
Scaffold(
topBar = {
TopAppBar(
title = { Text("Scaffold Example") },
actions = {
IconButton(onClick = { /* doSomething */
}) {
Icon(Icons.Filled.Favorite, contentDes
cription = "Localized description")
}
}
)
},
floatingActionButtonPosition =
FabPosition.End,
floatingActionButton = {
FloatingActionButton(onClick = { /*
handle click
*/ }) {
Icon(Icons.Filled.Add,
contentDescription = "L
ocalized description")
}
},
drawerContent = {
DrawerHeader()
DrawerBody()
}
) { innerPadding →
BodyContent(Modifier.padding(innerPaddin
g).padding(8.d
p))
}
@Composable
fun BodyContent(modifier: Modifier =
Modifier) {
// Your screen content
}
```

## FloatingActionButton in Jetpack Compose

### Example:

```
FloatingActionButton(
onClick = { /* Do something */ },
backgroundColor =
MaterialTheme.colors.secondary,
contentColor =
contentColorFor(backgroundColor = Mater
ialTheme.colors.secondary),
elevation =
FloatingActionButtonDefaults.elevation(6
.d
p)
) {
Icon(Icons.Default.Add,
contentDescription = "Add Ite
m")
}
```

## Using Dummy Data

### Example:

```
data class User(val name: String, val
email: String)
val dummyUsers = listOf(
User("John Doe", "john@example.com"),
User("Jane Smith", "jane@example.com")
)
LazyColumn {
items(dummyUsers) { user →
Text("Name: ${user.name}, Email:
${user.email}")
}
}
```