JETPACK COMPOSE

XML vs. Jetpack Compose

Jetpack Compose:

- **Declarative UI:** Define UI declaratively.
- **Kotlin-Based:** Uses Kotlin for expressive syntax.

XML:

- **Separation of Concerns:** UI in XML, logic in Kotlin/Java.
- **Static Layouts:** Requires more boilerplate for dynamic changes.

Composables

Definition:

Functions annotated with @Composable that define UI elements.

Creating Composables:

```
@Composable
fun CustomText(text: String) {
Text(text, style = TextStyle(fontSize = 20.sp, color =
Color.Red))
}
@Composable
fun MyApp() {
Column {
CustomText("Hello, World!")
CustomText("Welcome to Jetpack Compose!")
}
}rn go(f, seed, [])
}
```

Column:

```
@Composable
fun CustomColumn() {
Column(modifier = Modifier.fillMaxSize(),
verticalArrangement = Arrangement.Center,
horizontalAlignment = Alignment.CenterHorizontall
y) {
Text("Item 1")
Text("Item 2")
Text("Item 3")
}
}
```

Dropdown Menu

```
@Composable
fun DropdownMenuExample() {
var expanded by remember { mutableStateOf(false) }
Box {
IconButton(onClick = { expanded = !expanded }) {
Icon(imageVector = Icons.Default.ArrowDropDow
n, contentDescription = null)
}
DropdownMenu(expanded = expanded, onDismissRequest
= { expanded = false }) {
DropdownMenuItem(onClick = { /* Do something
*/ }) {
Text("Option 1")
}
DropdownMenuItem(onClick = { /* Do something
*/ }) {
Text("Option 2")
}
}
}
}
```

Text Composables

TextField:

```
@Composable
fun MyTextField() {
TextField(value = "", onValueChange = {}, label = { Te
xt("Enter your name") })
}
```

BasicTextField:

```
@Composable
fun MyBasicTextField() {
BasicTextField(value = "", onValueChange = {})
}
```

OutlinedTextField:

```
@Composable
fun MyOutlinedTextField() {
OutlinedTextField(value = "", onValueChange = {}, label = {
Text("Enter your email") })
}
```

Preview Composable

Usage:

```
@Preview
@Composable
fun PreviewMyButton() {
MyButton()
}
```

Button Composable

Definition:

Clickable element for user interaction.

Example:

```
@Composable
fun MyButton() {
Button(onClick = { println("Button was
clicked!") }) {
Text("Click me")
}
}
```

Dropdown Menu

Column:

```
@Composable
fun SimpleScreen() {
Column(
horizontalAlignment = Alignment.CenterHorizontally,
verticalArrangement = Arrangement.Center,
modifier = Modifier.fillMaxSize()
) {
Image(painter = painterResource(id = R.drawabl
e.ic_launcher_foreground), contentDescription = null)
Button(onClick = { /* Do something */ }) {
Text("Click me")
}
}
}
```

tutorials.EU

Context

Definition:

Access resources, launch activities, show toasts.

Example:

```
@Composable
fun ShowToastButton() {
val context = LocalContext.current
Button(onClick = {
Toast.makeText(context, "Button
clicked!", Toast.L
ENGTH_SHORT).show()
}) {
Text("Click me")
}
}
```

Box Composable

Definition:

Container for stacking elements.

Example:

```
@Composable
fun IconButton() {
Box(contentAlignment = Alignment.Center, modifier = Mo
difier.size(100.dp)) {
Icon(imageVector = Icons.Default.Star, contentDesc
ription = null)
Text("Star")
}
}
```

Icon Composable

Usage:

Add icons to UI elements.

Example:

```
@Composable
fun StarIcon() {
Icon(imageVector = Icons.Default.Star, contentDescript
ion = "Star Icon")
}
```

Space vs. Padding

Padding:

```
Text("Hello", modifier = Modifier.padding(16.dp))
}
```

Spacer:

```
Column {
Text("Item 1")
Spacer(modifier = Modifier.height(20.dp))
Text("Item 2")
}
```