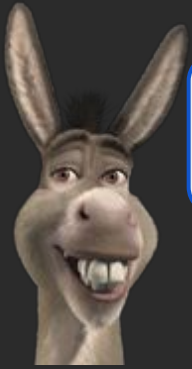
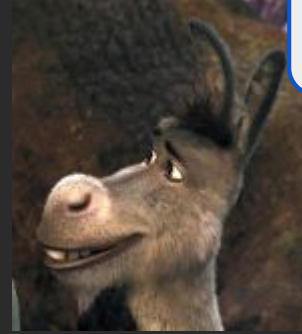




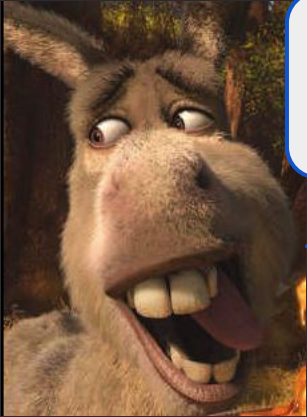
# *UI tests with Compose*



Shrek, Compose  
functions are the  
future!



Shrek, do you know  
something about  
testing?



Shrek, let's try to  
test a Composable  
function..

Shrek, are you  
here?



FINE...  
Let's test a composable  
function.



```
compose_bom = "2023.05.01"  
compose_compiler = "1.4.3"
```

```
compose_bom = { group = "androidx.compose", name = "compose-bom", version.ref = "compose_bom" }  
compose_bom_ui = { module = "androidx.compose.ui:ui" }  
compose_bom_preview = { module = "androidx.compose.ui:ui-tooling-preview" }  
compose_bom_activity = { module = "androidx.activity:activity-compose" }  
compose_bom_ui_tooling = { module = "androidx.compose.ui:ui-tooling" }  
compose_bom_ui_test_manifest = { module = "androidx.compose.ui:ui-test-manifest" }  
ui_compose_test = { group = "androidx.compose.ui", name = "ui-test-junit4", version.ref = "compose_compiler" }
```

- **libs.versions.toml**

According to the spec, we need to add the dependencies...

```
androidTestImplementation(libs.ui.compose.test)  
  
// Compose  
implementation(platform(libs.compose.bom))  
implementation(libs.compose.bom.ui)  
implementation(libs.compose.bom.preview)  
implementation(libs.compose.bom.activity)  
debugImplementation(libs.compose.bom.ui.tooling)  
debugImplementation(libs.compose.bom.ui.test.manifest)  
implementation(libs.bundles.compose.extras)
```

- **build.gradle.kts (app module)**

Shrek, before you  
ask, I found a  
composable function  
to test...



It is an app bar. I want to test the following behaviors:

- There is a back event, the left arrow icon appears;
- There is no back event, the left arrow icon doesn't appear.



AppToolbarPreview - dark mode


← Jumangi

AppToolbarPreview - light mode

← Jumangi



```
class AppToolbarTest {  
    @get:Rule  
    val composeTestRule = createComposeRule()  
  
    @Test  
    fun showAppBarNavigationUpIcon() {...}  
  
    @Test  
    fun hideAppBarNavigationUpIcon() {...}
```



Shrek, it is so beautiful. Is it a dream?

FINE...

Let's do it.

We need to create a class defining the compose test rule.

First test, we need to test if our Composable function is showing the button when there is a valid event...

```
@Test
fun showAppBarNavigationUpIcon() {
    // arrange
    val backEvent = {

    }

    // act
    composeTestRule.setContent {
        AppToolbar(
            title = "any title",
            backEvent = backEvent
        )
    }

    // assert
    composeTestRule
        .onNodeWithContentDescription(label: "Back")
        .assertExists()
}
```



Second test, we need to test if our Composable function hides the button when no event is passed...

```
@Test
fun hideAppBarNavigationUpIcon() {
    // arrange
    val backEvent : (()->Unit)? = null

    // act
    composeTestRule.setContent {
        AppToolbar(
            title = "any title",
            backEvent = backEvent
        )
    }

    // assert
    composeTestRule
        .onNodeWithContentDescription(label: "Back")
        .assertDoesNotExist()
}
```

