BCS 371 Mobile Application Development I

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

© 2024 Arthur Hoskey. All rights reserved.

Today's Lecture

- Back Stack Overview
- Compose - NavHost

© 2024 Arthur Hoskey. All rights reserved.

<u>Back Stack Overview</u>

- The app keeps track of its navigation state using a back stack.
- The back stack basically contains a stack of places where the app has navigated from.
- Essentially, it keeps track of how it got to the current screen.
- The back stack operates in a last in first out manner (LIFO).

Back Stack Overview

© 2024 Arthur Hoskey. All rights reserved.

Back Stack Flow

- Every time the app navigates to a new screen it pushes a new entry on to the back stack.
- When the app navigates back to the previous screen it pops an entry off the back stack.

Back Stack Flow

© 2024 Arthur Hoskey. All rights reserved.

Back Stack Flow Example

- Assume there is a food delivery app with the following screens:

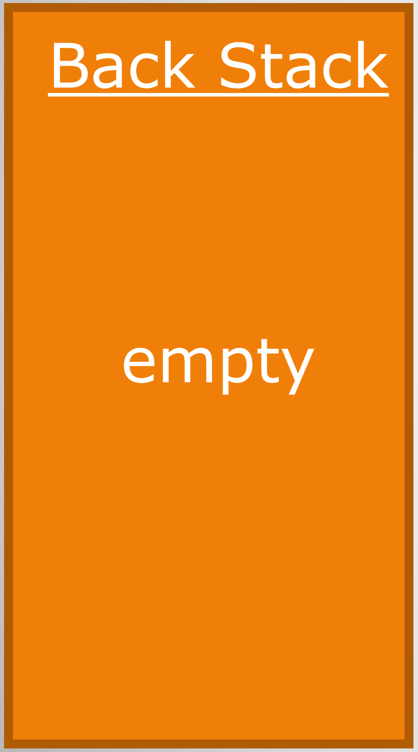
- **Main screen** – Shows a list of different restaurants.
- **Restaurant screen** – Shows a list of food you can order from a restaurant.
- **Item screen** – Allows user to view a food item and add it to their order cart.
- **Cart screen** – View items in cart and place order.

Back Stack Flow Example

© 2024 Arthur Hoskey. All rights reserved.

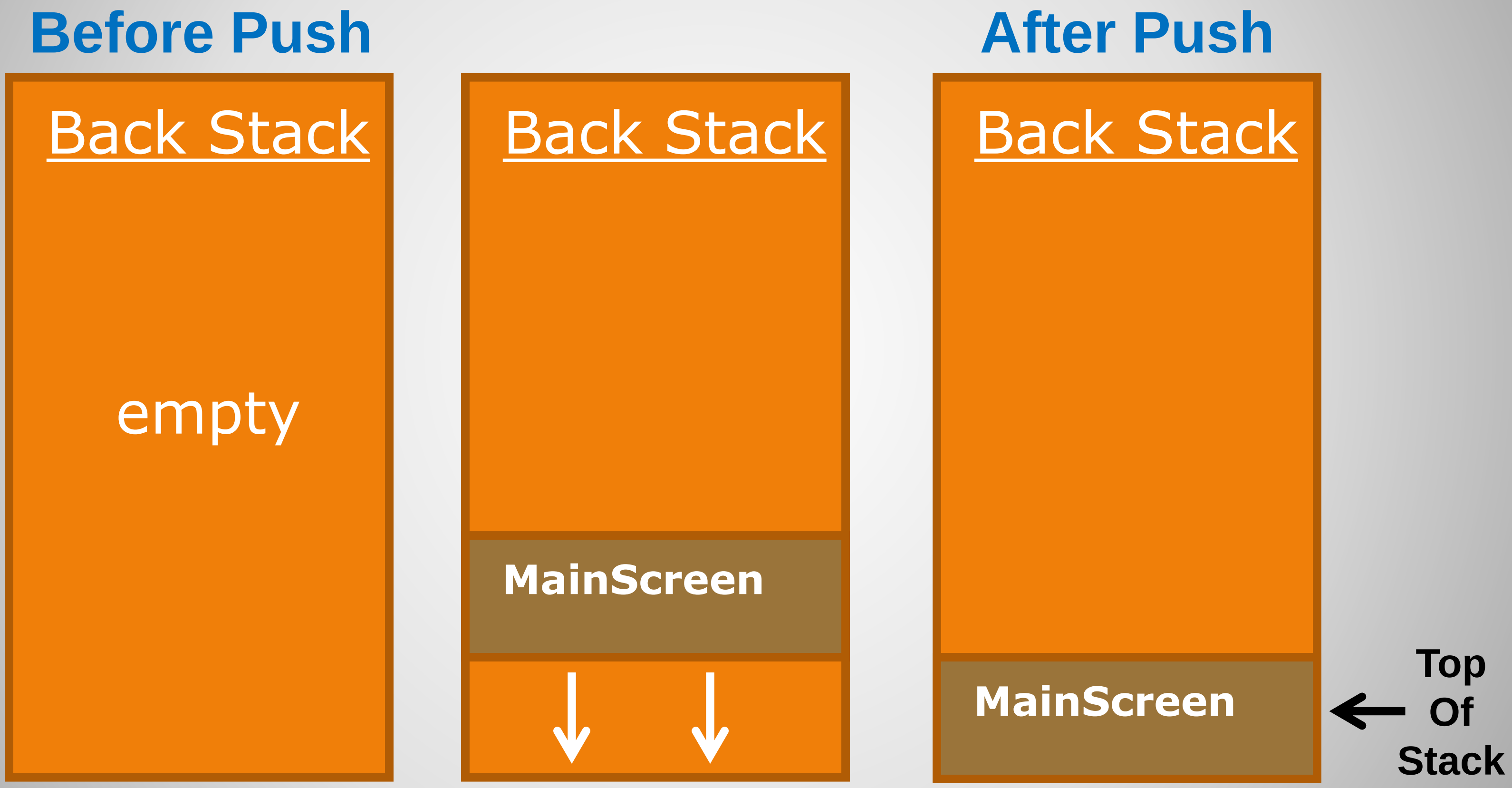
Back Stack Flow Example

Back stack is empty before the user starts the app.



© 2023 Arthur Hoskey. All rights reserved.

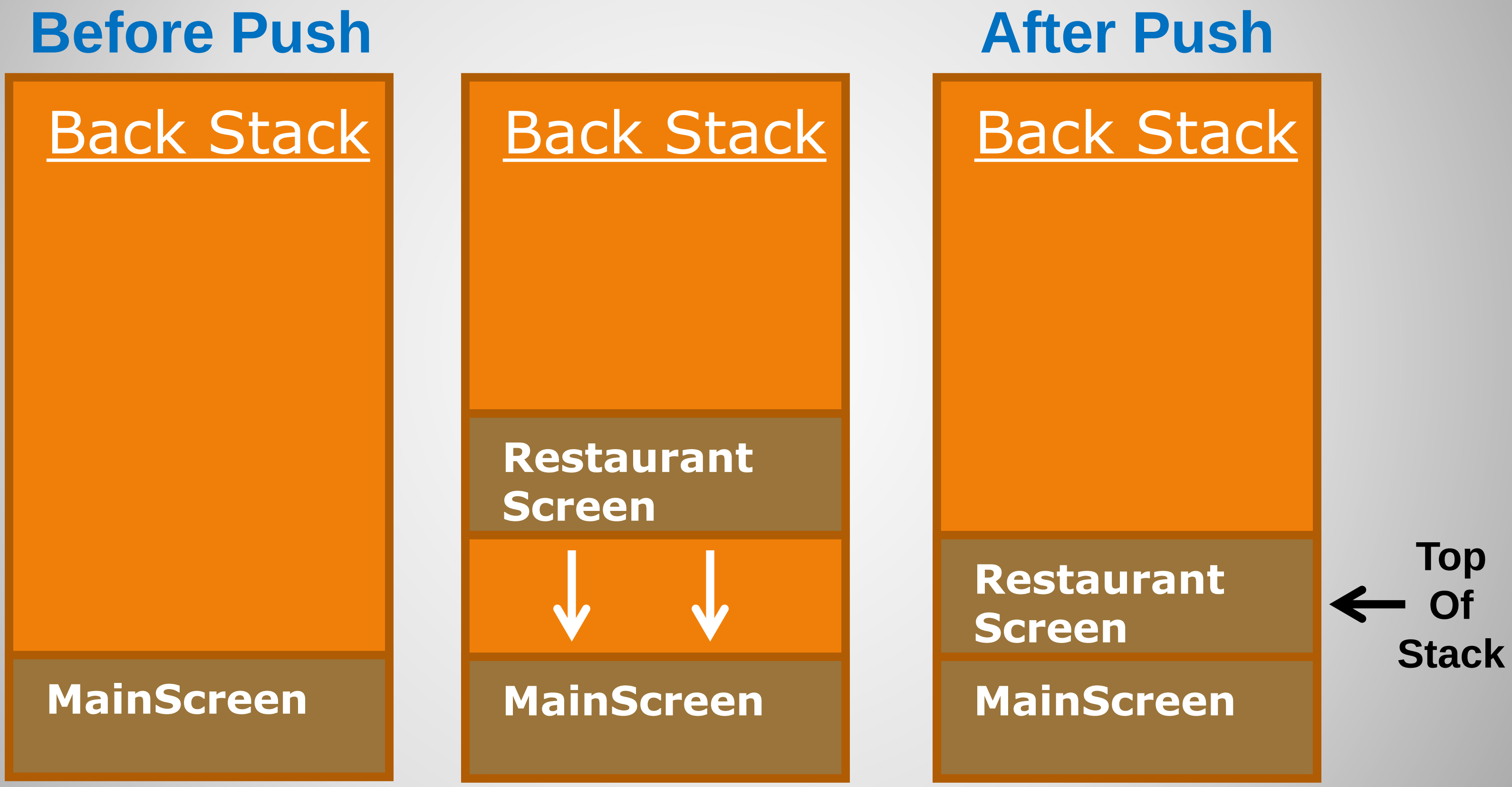
User starts app and is at the MainScreen.
A back stack entry for MainScreen is pushed on to the back stack.
The current screen is always at the top of the back stack.



Back Stack Flow Example

© 2023 Arthur Hoskey. All rights reserved.

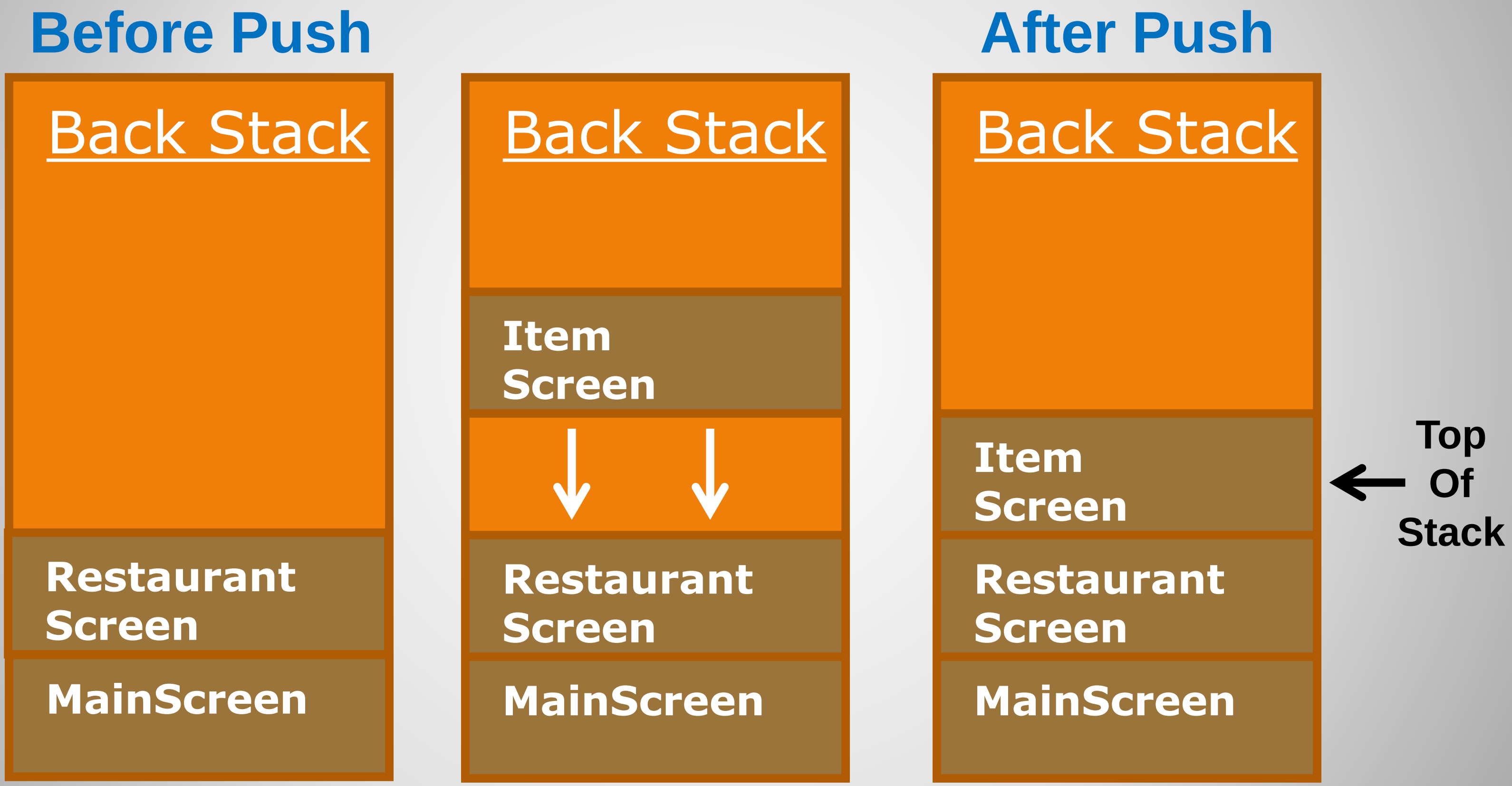
User navigates to RestaurantScreen.
Push back stack entry for RestaurantScreen on to the back stack.



Back Stack Flow Example

© 2023 Arthur Hoskey. All rights reserved.

User navigates to ItemScreen.
Push back stack entry for ItemScreen on to the back stack.

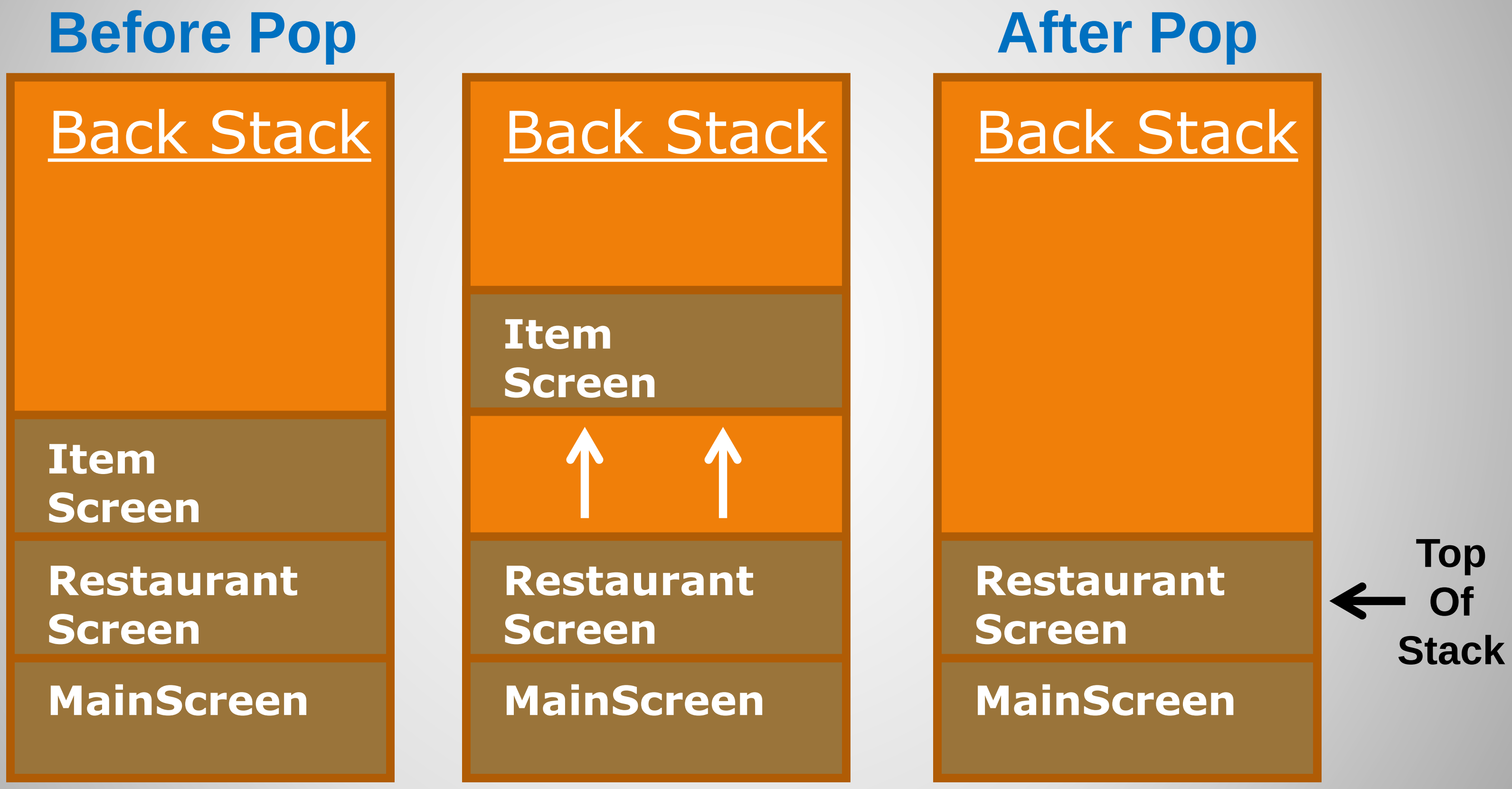


Back Stack Flow Example

© 2023 Arthur Hoskey. All rights reserved.

User adds the item to the cart (by pressing an "Add to Cart" button) and is automatically sent back to the restaurant screen.

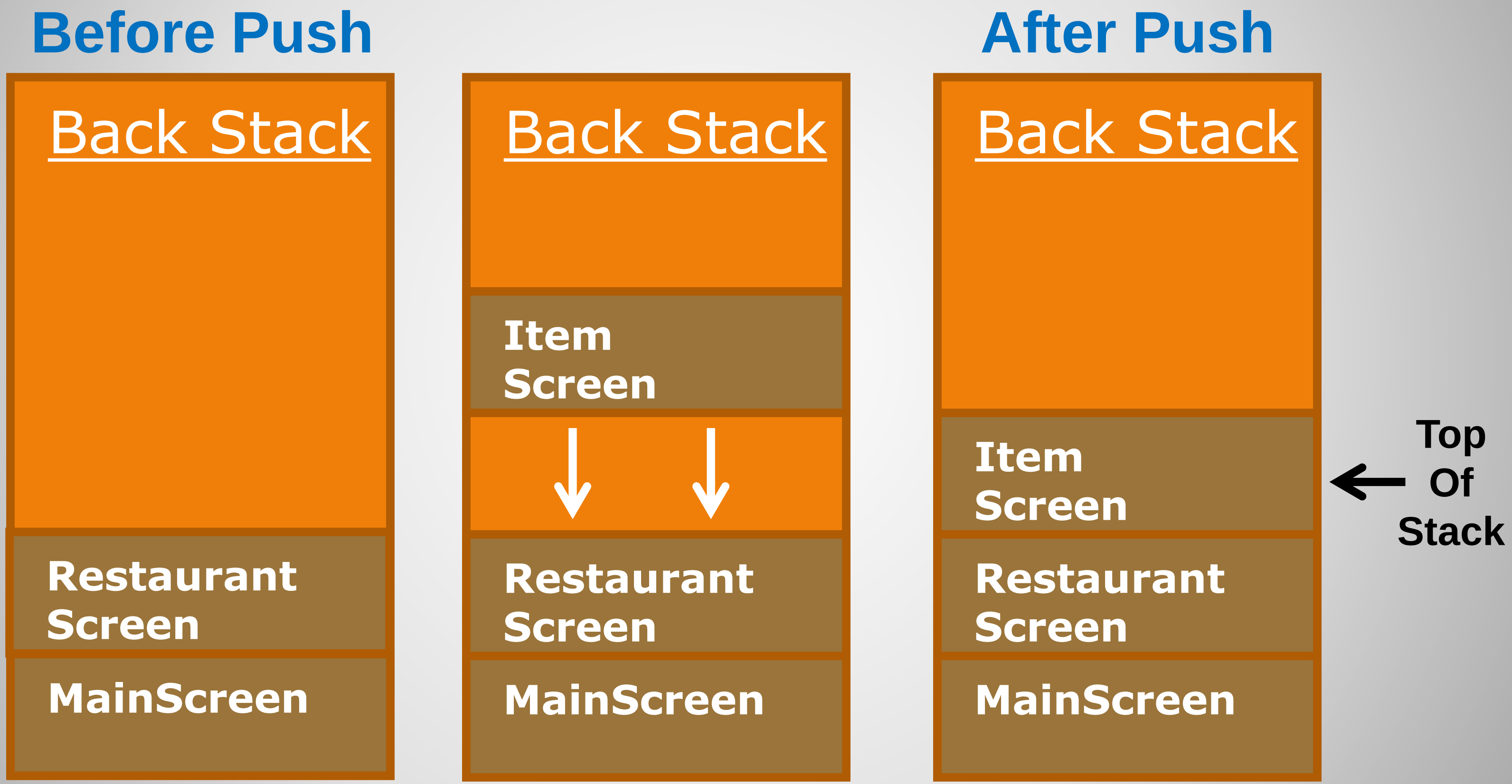
Pop the top entry off the back stack to return to the previous screen.



Back Stack Flow Example

© 2023 Arthur Hoskey. All rights reserved.

User navigates to ItemScreen for a different item.
Push back stack entry for ItemScreen on to the back stack.



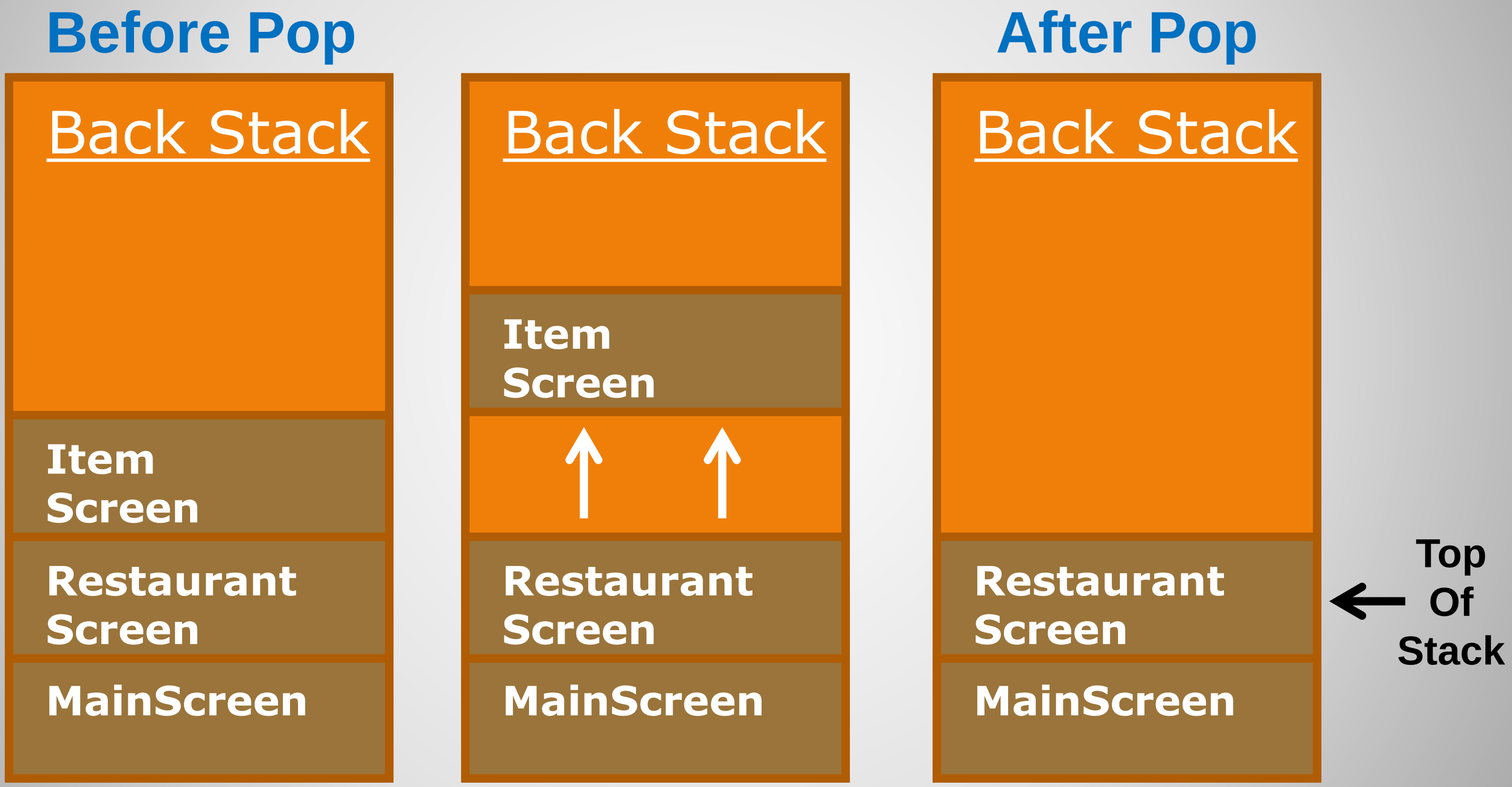
Back Stack Flow Example

© 2023 Arthur Hoskey. All rights reserved.

Back Stack Flow Example

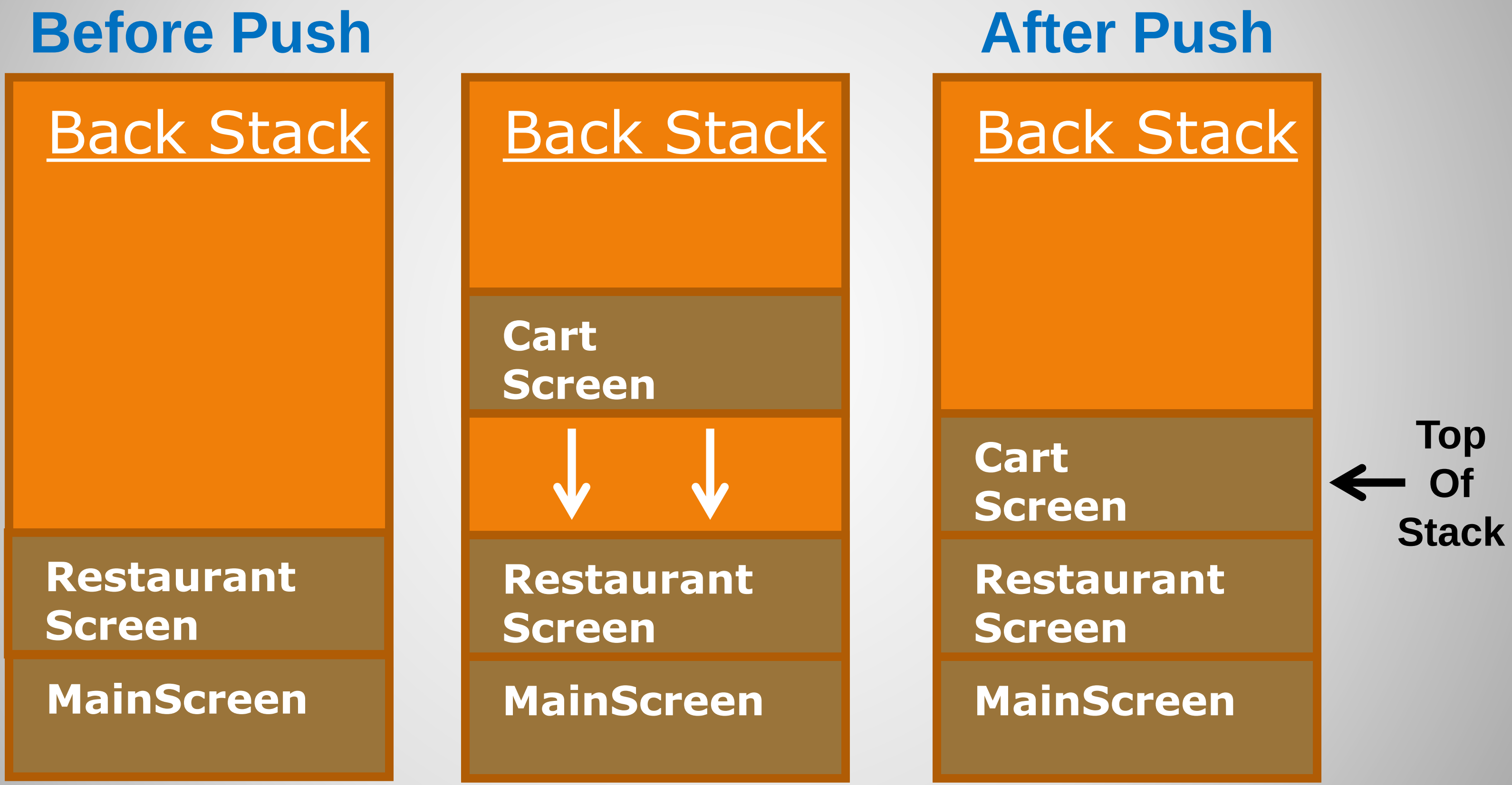
User decides not to add that item and presses the device's back button.

Pushing the device's back button will cause the app to pop the top entry off the back stack (app returns to RestaurantScreen).



© 2023 Arthur Hoskey. All rights reserved.

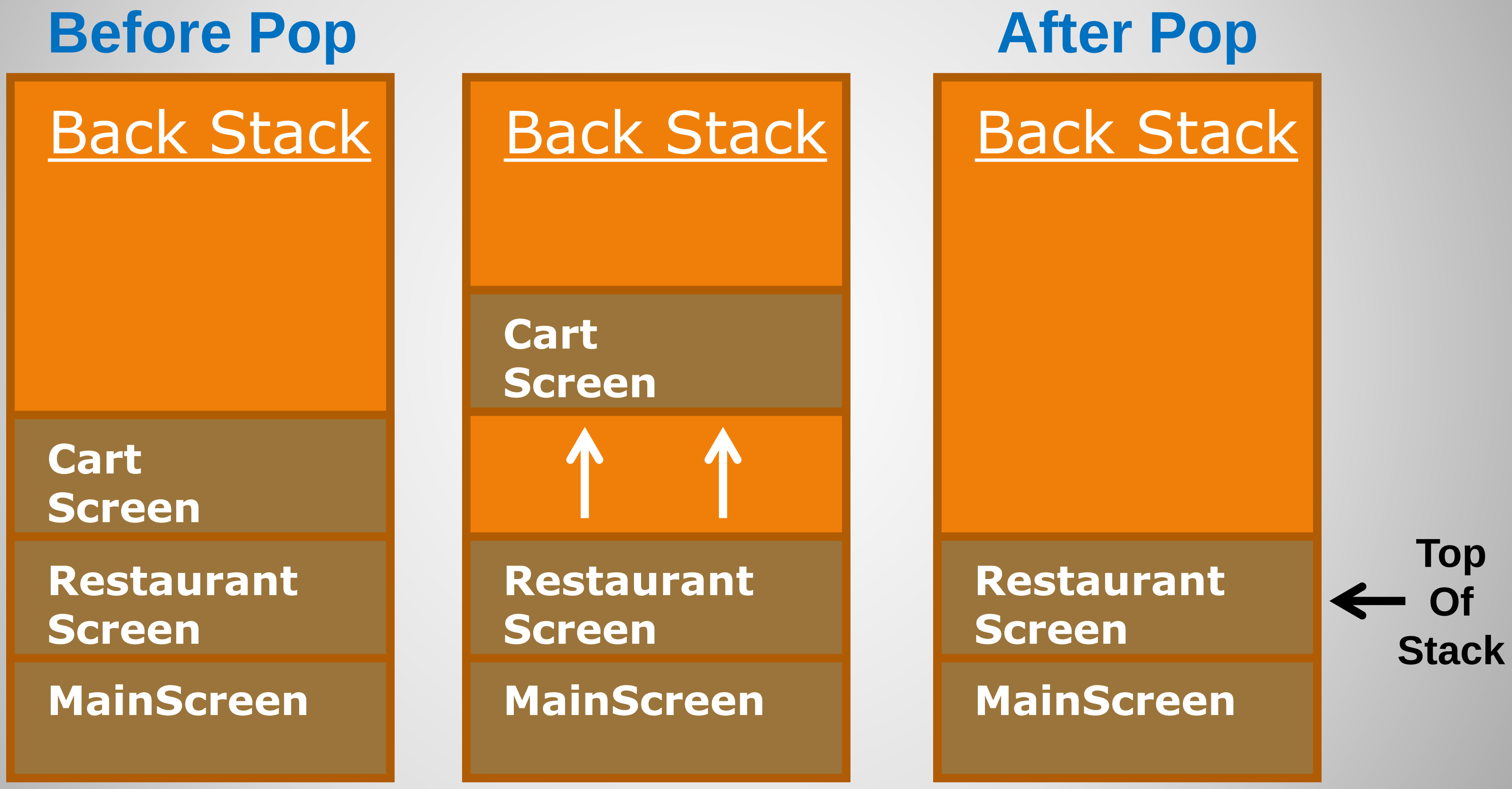
User navigates to CartScreen to place order.
Push back stack entry for CartScreen on to the back stack.



Back Stack Flow Example

© 2023 Arthur Hoskey. All rights reserved.

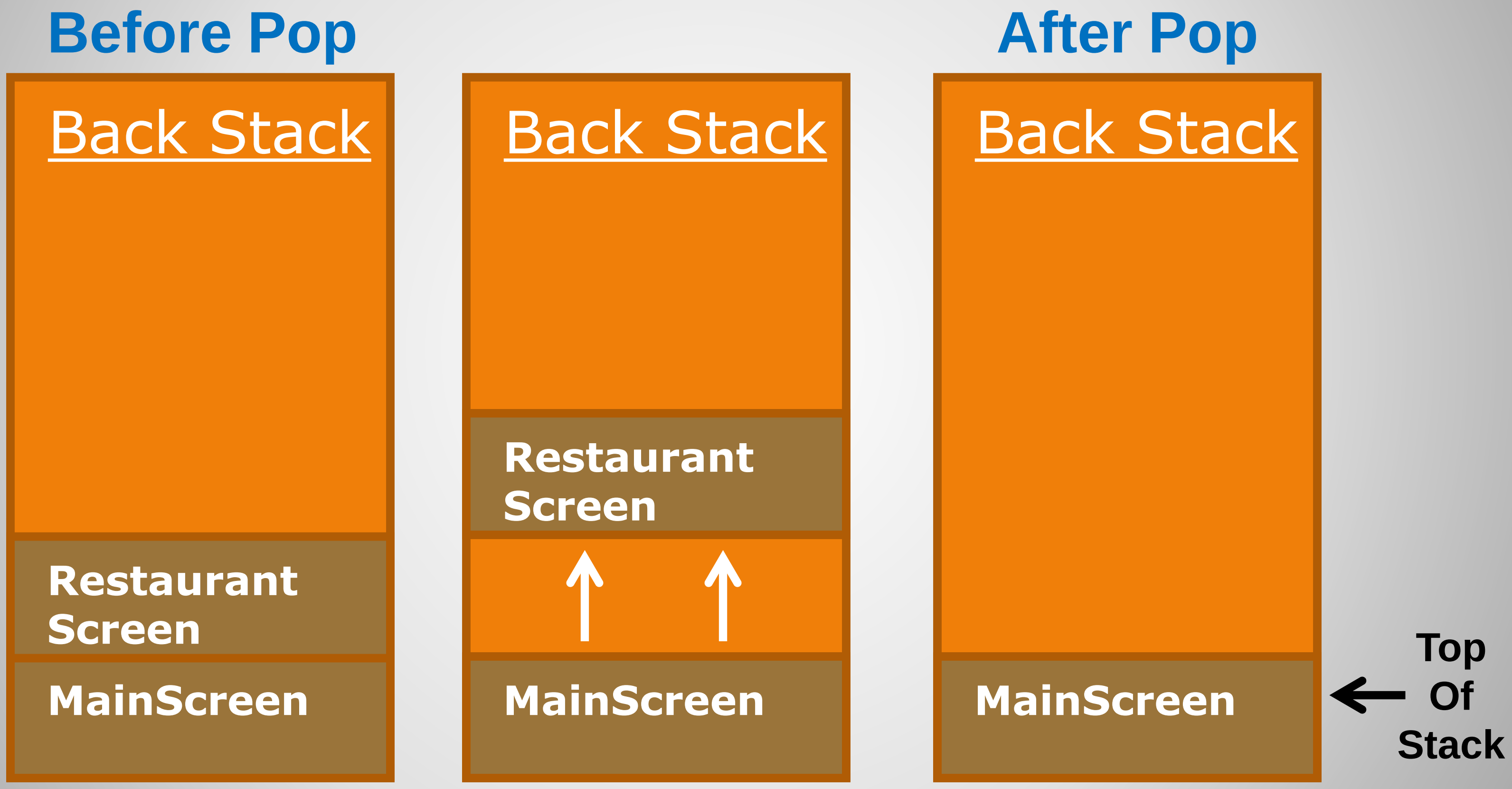
User presses the "Place order button" and is automatically sent back to the RestaurantScreen.



Back Stack Flow Example

© 2023 Arthur Hoskey. All rights reserved.

User presses the device's back button.

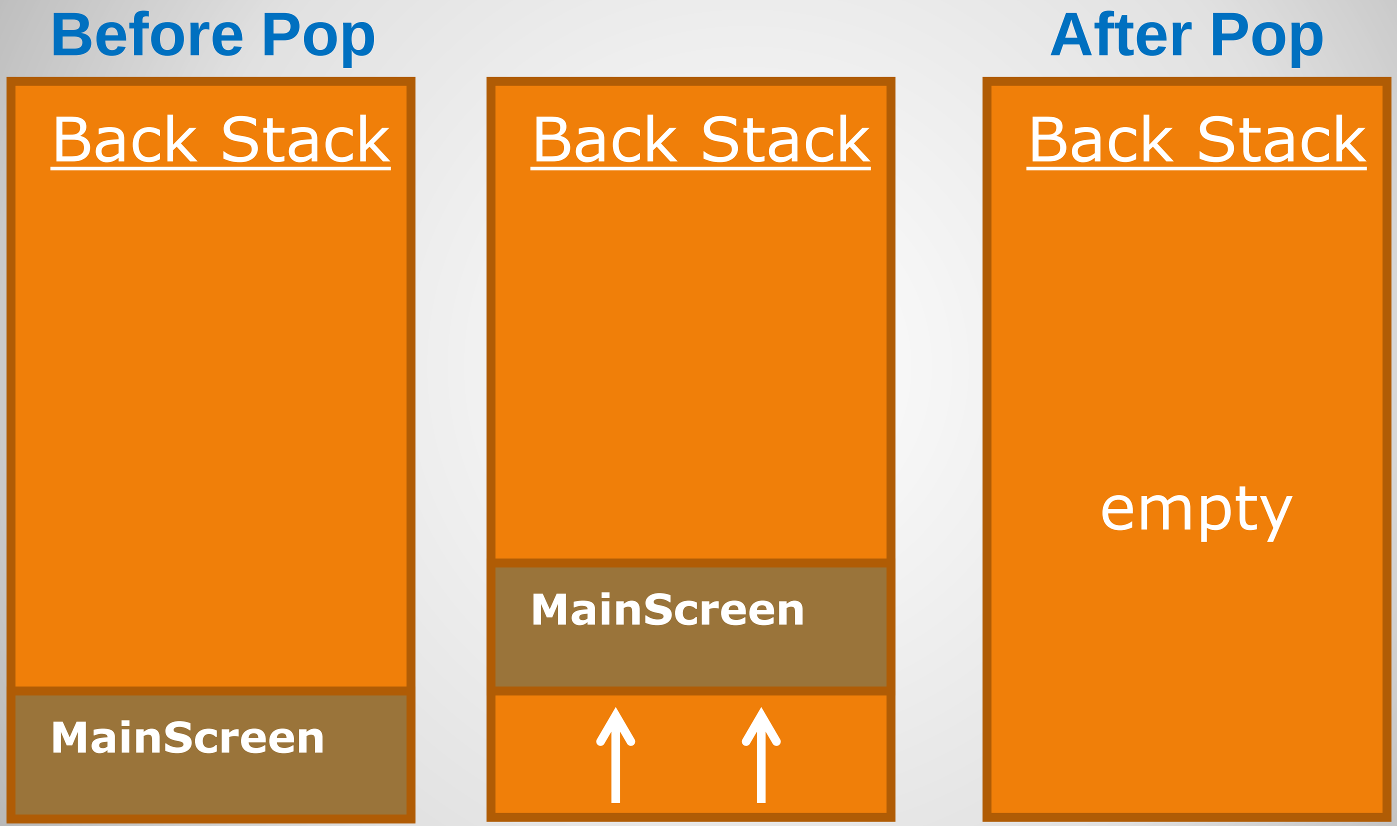


Back Stack Flow Example

© 2023 Arthur Hoskey. All rights reserved.

Back Stack Flow Example

User presses the device's back button.



© 2023 Arthur Hoskey. All rights reserved.

- Now on to navigation and NavHost…

Navigation and NavHost

© 2024 Arthur Hoskey. All rights reserved.

Navigation

Navigation

- Most apps will have multiple screens in the UI.
- Jetpack Compose Navigation allows you to work with those screens.
- You will need to add the following Gradle dependency to the Gradle (app) file (make sure to Sync the Gradle file):

```
implementation("androidx.navigation:navigation-compose:2.8.5")
```

Check the following link for the latest dependency version:
https://developer.android.com/develop/ui/compose/navigation

© 2024 Arthur Hoskey. All rights reserved.

Navigation Components

- **NavHost** – Container for navigation. It links the NavController to the NavGraph.
- **NavController** – Used to navigate between destinations. It maintains the back stack of screens.
- **NavGraph** – Specifies the destinations in the app.

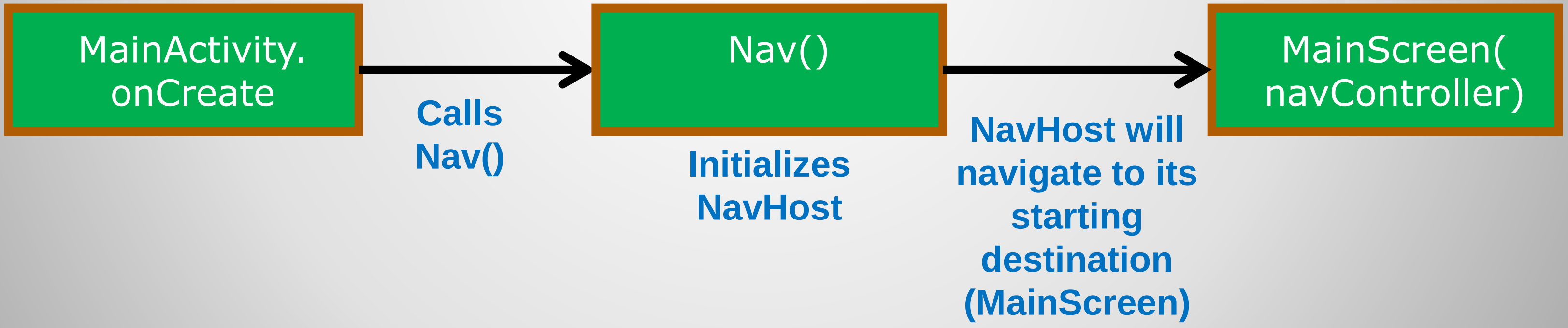
Navigation Components

© 2024 Arthur Hoskey. All rights reserved.

Setup NavHost - Overview

1. Create screen composable functions (one for each screen in app). If you are using a NavigationBar then each screen function takes no parameters otherwise you should pass a NavController as a parameter to the screen functions (the following slides assume no NavigationBar).
2. Create Nav() composable function to setup the NavHost.
3. Update MainActivity.onCreate to call Nav().

The app will call functions in the following sequence:


MainActivity. onCreate
Calls Nav()
Nav()
Initializes NavHost
NavHost will navigate to its starting destination (MainScreen)
MainScreen(navController)

Setup NavHost - Overview

© 2024 Arthur Hoskey. All rights reserved.

1a. Screen Function - mainScreen

- Create a Kotlin file named MainScreen.kt.
- In this example there is one button that is used to navigate to the other screen.

Pass in the NavHostController

```
@Composable
fun MainScreen(navController: NavHostController, modifier: Modifier) {
    Column {
        Text(text = "Main Screen")
        Button(onClick={
            navController.navigate("OtherScreen")
        })
        {
            Text("Go to other screen")
        }
    }
}
```

Call navigate when the button is pressed. Navigate must get a route as a parameter. The route is "OtherScreen" in this example. Each call to the navController's navigate() function pushes the given destination to the top of the stack.

This route name must have already been defined inside the NavHost.

1. Screen Function - mainScreen

© 2024 Arthur Hoskey. All rights reserved.

1b. Screen Function - otherScreen

- Create a Kotlin file named OtherScreen.kt
- There is one button to navigate back to the main screen.

Pass in the NavHostController

```
@Composable
fun OtherScreen(navController: NavHostController, modifier: Modifier)
{
    Column {
        Text(text = "Other Screen")
        Button(onClick={
            navController.popBackStack()
        })
        {
            Text("Go back to main screen")
        }
    }
}
```

Call popBackStack to navigate back to the previous screen.

If we had defined other screens, we could have navigated to one of those instead using the navigate function.

1. Screen Function - Other Screen

© 2024 Arthur Hoskey. All rights reserved.

2. Function – Nav() (Setup NavHost)

- Create a Kotlin file named Nav.kt.
- The route parameter (for composable) identifies the screen. The navController's navigate method takes a route as a parameter.
- Create a Kotlin file named Nav.kt and add the following:

```
@Composable
fun Nav(modifier: Modifier) {
    val navController = rememberNavController()

    NavHost(navController=navController, startDestination = "MainScreen", modifier) {
        composable(route="MainScreen") {
            MainScreen(navController, modifier)
        }
        composable(route="OtherScreen") {
            OtherScreen(navController, modifier)
        }
    }
}
```

Create NavController

Set the starting screen

NavGraph

The composable function adds a destination to the NavHost's NavGraph. (assumes MainScreen and OtherScreen have been defined). Each composable has a route which identifies it as a destination.

The MainScreen(navController) and OtherScreen(navController) functions can be coded to NOT take the NavController as a parameter

2. Function – nav (Setup NavHost)

© 2024 Arthur Hoskey. All rights reserved.

3. Update MainActivity.onCreate

- Should call the Nav() composable function.
- For example:

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            TestNavHostWithScaffoldTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    Nav(Modifier.padding(innerPadding))
                }
            }
        }
    }
}
```

3. Update MainActivity.onCreate

© 2024 Arthur Hoskey. All rights reserved.

Navigate to Different Screens

- Use the NavController's navigate method to navigate to a different screen.
- For example:

```
navController.navigate("OtherScreen")
```

navigate() takes a route as a parameter. The route is "OtherScreen" in this example. Each call to the navController's navigate() function pushes the given destination to the top of the stack.

This route name must have already been defined inside the NavHost's NavGraph.

Assumes that navController has been declared and that it has the type NavHostController

"OtherScreen" is a route that has been defined in the NavHost's NavGraph

```
navController.popBackStack()
```

Use popBackStack() to navigate to the previous screen

Navigate to Different Screens

© 2024 Arthur Hoskey. All rights reserved.

- Now on to passing arguments between screens…

Pass Arguments Between Screens

© 2024 Arthur Hoskey. All rights reserved.

Pass Arguments Between Screens

- One way to pass data is to add parameters to the route for the screen.
- Do the following:

1. Update NavHost entry for the destination screen.
   A. Add a place holder argument for the data to the route. String is the default type for these arguments. To pass a type other than string you can use navArgument to specify it.
   B. Extract parameter from the route and pass that parameter to the screen function call.
2. Update the screen function to take an additional parameter for the data being passed.
3. When calling navigate pass the data as part of the route.

The example on the following slides passes data from MainScreen to DataScreen.

Pass Arguments Between Screens

© 2024 Arthur Hoskey. All rights reserved.

1. Add Placeholder Argument to Route

1. Update NavHost Entry

A. Add parameters to the route for the screen.

B. Extract parameter from route.

```
NavHost(navController=navController, startDestination = "MainScreen", modifier)
{

    // Other screens here

    composable(route="DataScreen/{data}") {

        val param = it.arguments?.getString("data")
        if (param != null) {
            DataScreen(navController, param, modifier)
        }
    }
}
```

Add placeholder to the route (data is the name of the placeholder)

Extract parameter (use placeholder named data in this example. The only argument passed to the lambda is a NavBackStackEntry (this goes in "it"). Extract the data parameter from the NavBackStackEntry using "it".

Call screen function passing in the parameter

© 2024 Arthur Hoskey. All rights reserved.

2. Update Screen Function

- Add a parameter to take the data.

Add parameter to the screen function header

```
@Composable
fun DataScreen(navController: NavHostController, data: String, modifier: Modifier)
{

    // Use data parameter in the composable function here

}
```

2. Update Screen Function

© 2024 Arthur Hoskey. All rights reserved.

3. Pass Data with Route

- Pass data as part of the route in the call to navigate.
- When the button is clicked it navigates to DataScreen.

```
@Composable
fun MainScreen(navController: NavHostController, modifier: Modifier) {
    var dataToPass = "abc"


    Button(onClick={
        navController.navigate("DataScreen/$dataToPass")
    })
    {
        Text("Go to data screen")
    }
}
```

The value in the dataToPass variable will be appended to the end of the route

$ means to treat what follows as a variable name

3. Pass Data with Route

© 2024 Arthur Hoskey. All rights reserved.

Pass Int Data (use navArgument)

Screen method takes an Int parameter

```
@Composable
fun DataScreenTakesInt(navController: NavHostController, dataInt: Int, modifier: Modifier) {
    // Screen code goes here
}
```

Use a navArgument to specify an int type for the parameter (in NavHost)

```
composable(
    route="DataScreenTakesInt/{data}",
    arguments = listOf(navArgument("data") { type = NavType.IntType })
) {
    val param = it.arguments?.getInt("data")
    if (param != null) {
        DataScreenTakesInt(navController, param, modifier)
    }
}
```

Get parameter data as an Int

Int variable pass as argument to navigate (in screen you are navigating from)

```
navController.navigate("DataScreenTakesInt/$dataToPassInt")
```

Pass Int Data (use navArgument)

© 2024 Arthur Hoskey. All rights reserved.

- Now on to passing data back to the previous screen…

Pass Data Back to the Previous Screen

© 2024 Arthur Hoskey. All rights reserved.

Pass Data Back to Previous Screen

- Assume you have a data entry screen, and you need to pass that data back to the previous screen.
- You can do this using the back stack. Specifically, you put the data to pass back into the previous screen's back stack entry.
- That back stack entry can be accessed by the previous screen. The previous screen goes into its back stack entry and retrieves the data.
- Do the following:

1. In EnterDataScreen. Add key/value pairs of data to return to the previous back stack entry. Use savedStateHandle to save the data in the back stack entry.
2. In MainScreen. Retrieve the key/value pairs from MainScreen's back stack entry (will be on top of the back stack at this point). Use savedStateHandle to retrieve the data from the back stack entry.

Pass Data Back to the Previous Screen

© 2024 Arthur Hoskey. All rights reserved.

<u>Pass Data Back to Previous Screen Example</u>

- The sequence of events proceeds as follows:

1. MainScreen navigates to EnterDataScreen.
2. User enters data when in EnterDataScreen.
3. User presses button to finish data entry and go back to previous screen. It should save the data in the previous screen's back stack entry (that is MainScreen's back stack entry). It should pop the back stack after saving the data to the back stack entry.
4. MainScreen retrieves the data from its back stack entry.

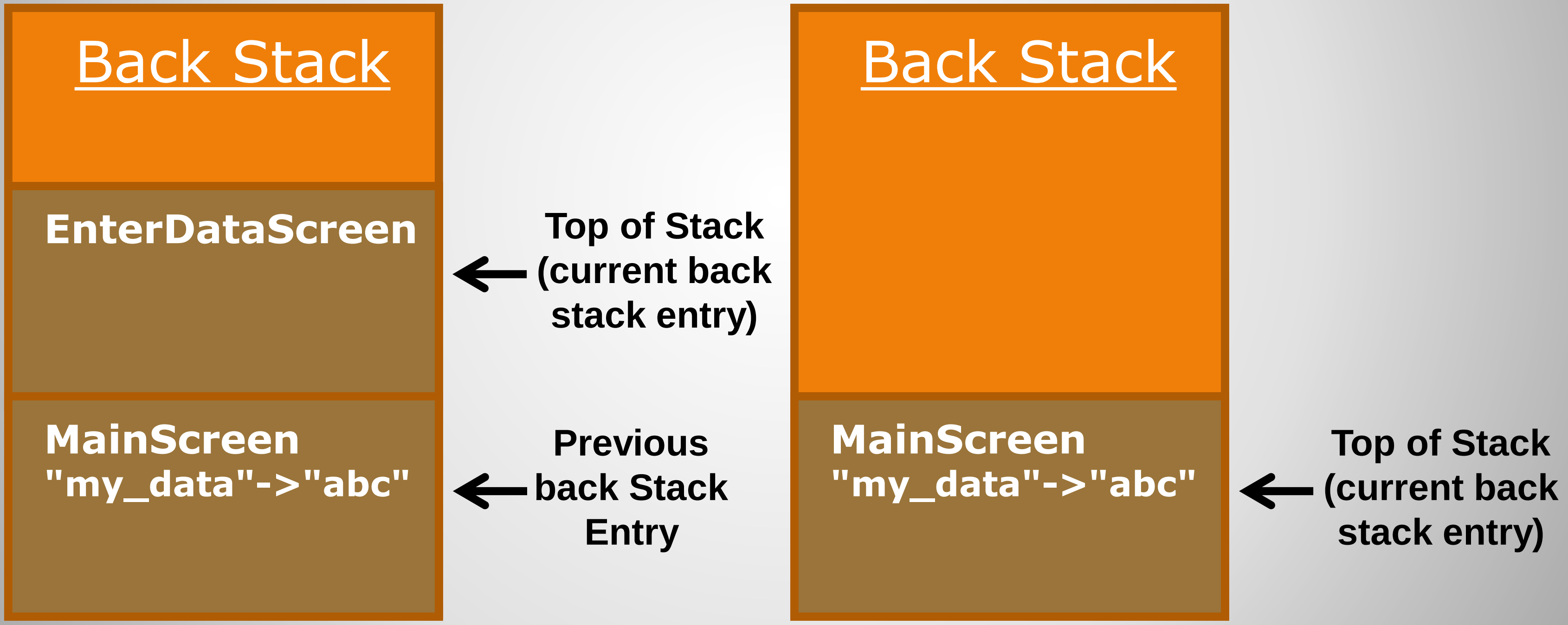
Pass Data Back to the Previous Screen Example

© 2024 Arthur Hoskey. All rights reserved.

Before going back to MainScreen, EnterDataScreen saves the data to return in the previous back stack entry (MainScreen's back stack entry).

The user presses the "Done button" on EnterDataScreen and stores the key/value pair in the previous back stack entry (assume user typed "abc").

MainScreen then accesses the current back stack entry to get the key/value pair.


Back Stack
EnterDataScreen
MainScreen
"my_data"->"abc"
Top of Stack (current back stack entry)
Previous back Stack Entry
Back Stack
MainScreen
"my_data"->"abc"
Top of Stack (current back stack entry)

Passing Data and Back Stack

© 2023 Arthur Hoskey. All rights reserved.

Pass Data Back to Previous Screen Example - EnterDataScreen

- Use the navController to access the back stack.

```
@Composable
fun EnterDataScreen(navController: NavHostController, modifier: Modifier) {
    var data by rememberSaveable { mutableStateOf("") }

    // Other code to fill the data variable with a value to be send back should be added

    Button(onClick={
        navController.previousBackStackEntry?.savedStateHandle?.set("my_data", data)
        navController.popBackStack()
    })
    {
        Text("Done")
    }
}
```

Save data as a key/value pair into the previous screen's back stack entry

Pop the back stack to navigate back to the previous screen

Pass Data Back to the Previous Screen Example - enterDataScreen

© 2024 Arthur Hoskey. All rights reserved.

Pass Data Back to Previous Screen Example - MainScreen

- Use the navController to access the back stack.

Get the value from the current back stack entry (MainScreen's back stack entry)

```
@Composable
fun MainScreen(navController: NavHostController, modifier: Modifier) {
    var data=navController.currentBackStackEntry?.savedStateHandle?.get<String>("my_data")

    // Code to use data in the UI goes here

}
```

The data variable will contain the data sent back from EnterDataScreen.

Pass Data Back to the Previous Screen Example - mainScreen

© 2024 Arthur Hoskey. All rights reserved.

- End of Slides

End of Slides

© 2024 Arthur Hoskey. All rights reserved.