Microsoft

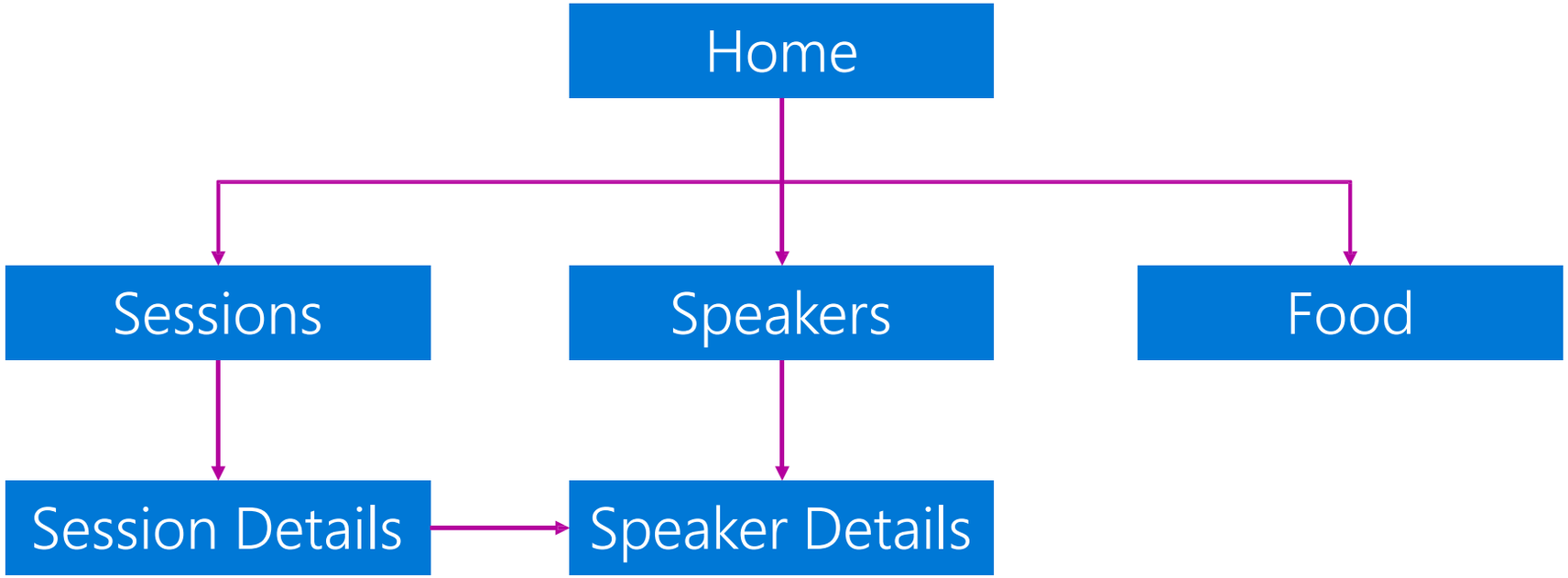
Navigation Patterns in Xamarin.Forms

Objectives

1. Move through a sequence of pages using stack navigation
2. Configure the navigation bar
3. Display transitory content using modal pages
4. Switch among a small group of pages using tab navigation

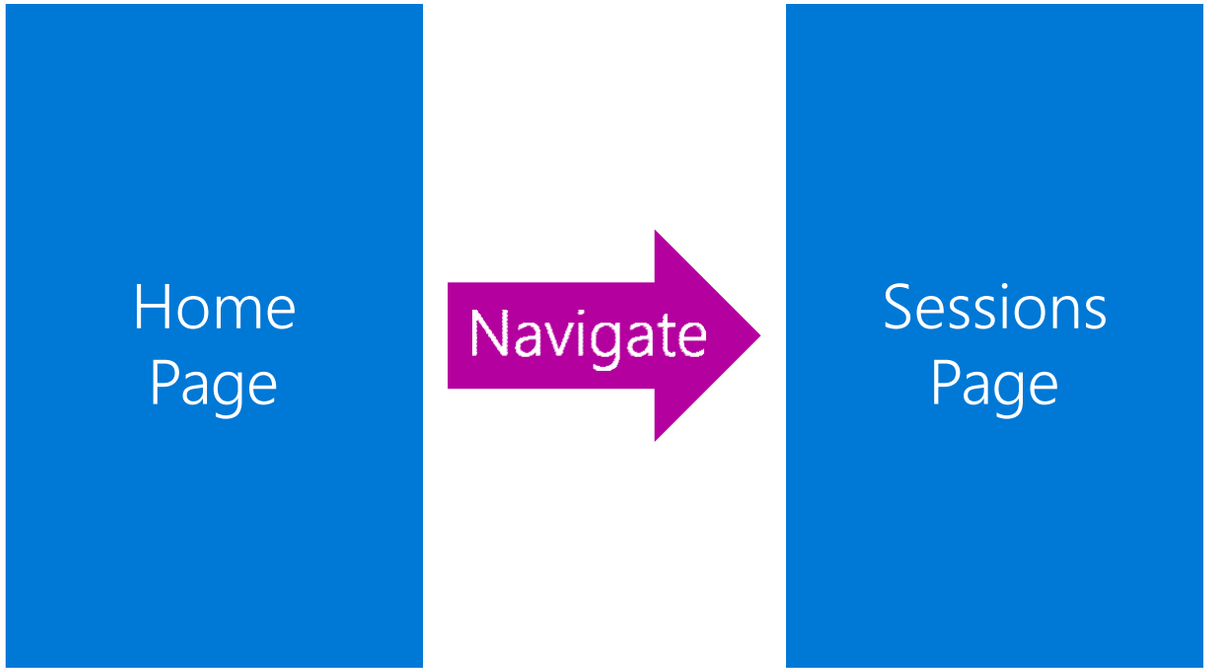
What is navigation?

Navigation is the set of transitions between the pages of your app; the term typically includes both the UI and user actions needed to make the transitions



Navigation granularity

The Xamarin.Forms navigation model uses **Page** as the unit of navigation

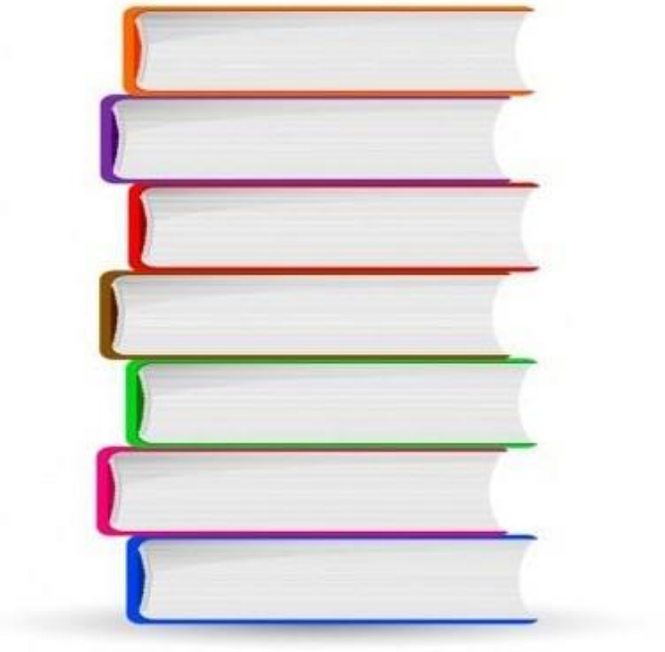


Navigation replaces the entire page of content

Move through a sequence of pages using stack navigation

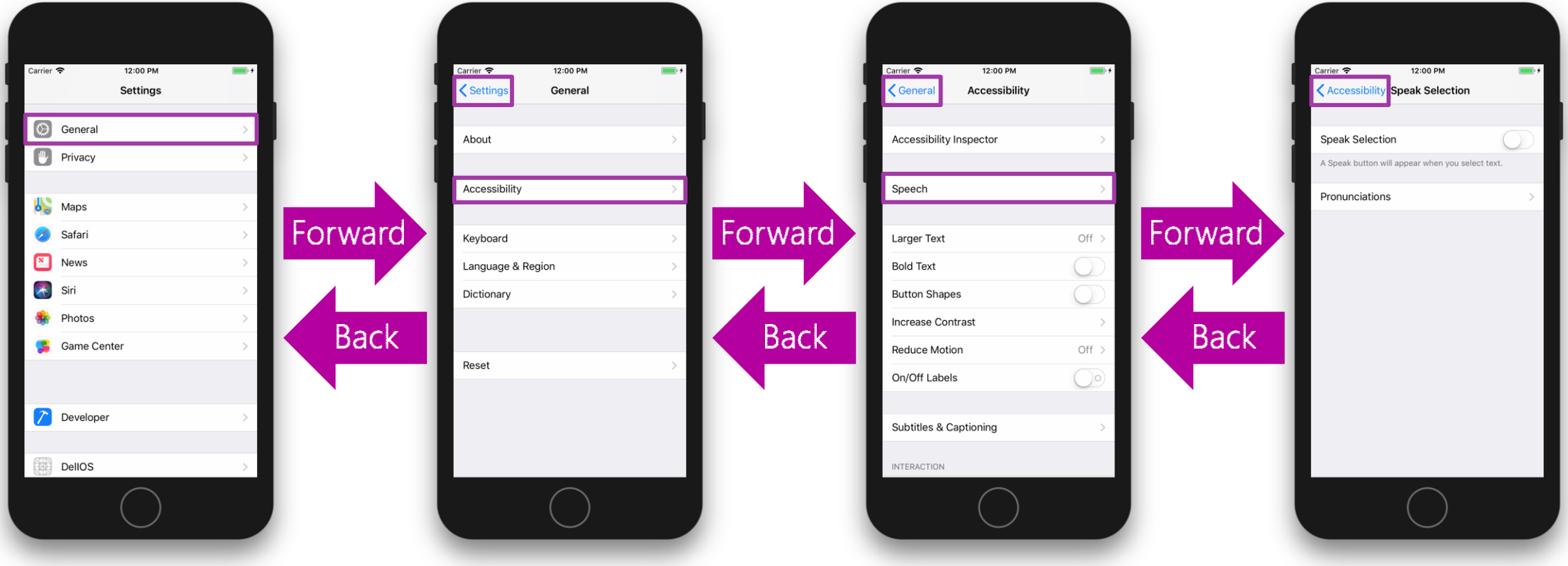
Tasks

1. Create a NavigationPage
2. Navigate forward and backward through a stack of Pages



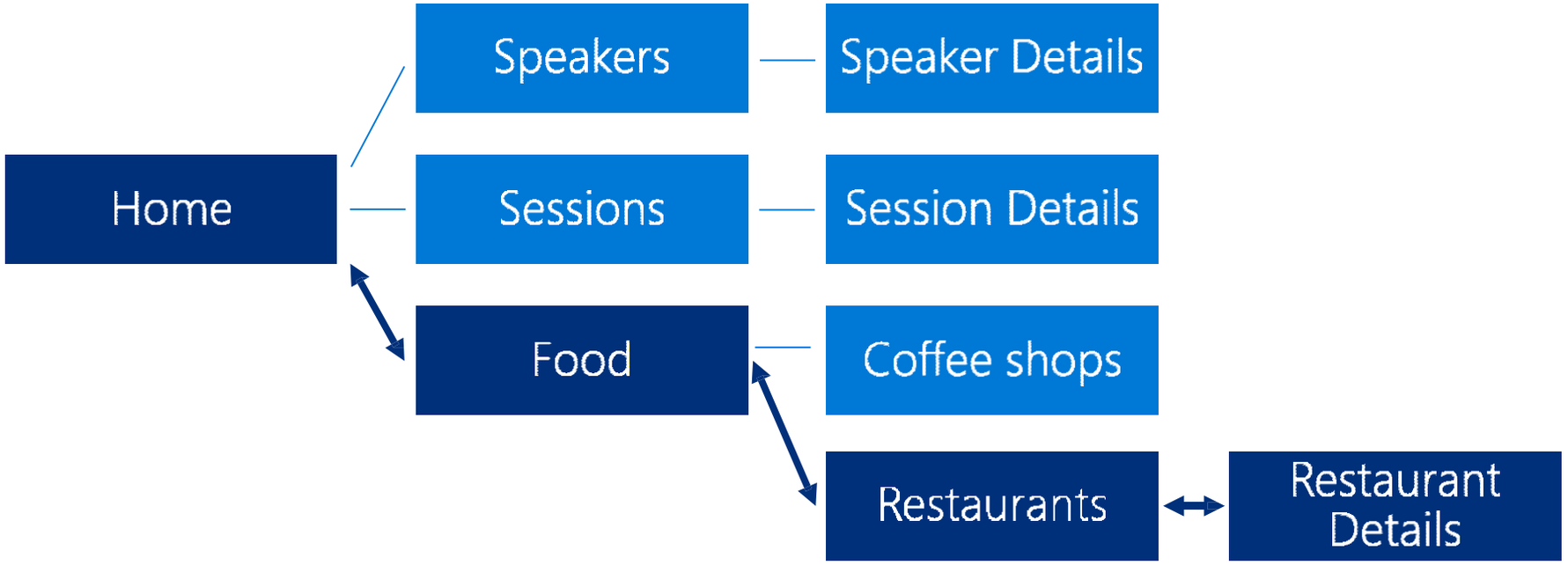
What is stack navigation?

Stack navigation is a navigation paradigm that constrains users to move forward to new pages and back to previous pages



When to use stack navigation?

Stack navigation is appropriate for a linear path through a hierarchy of pages; especially when the pages progress from general to specific

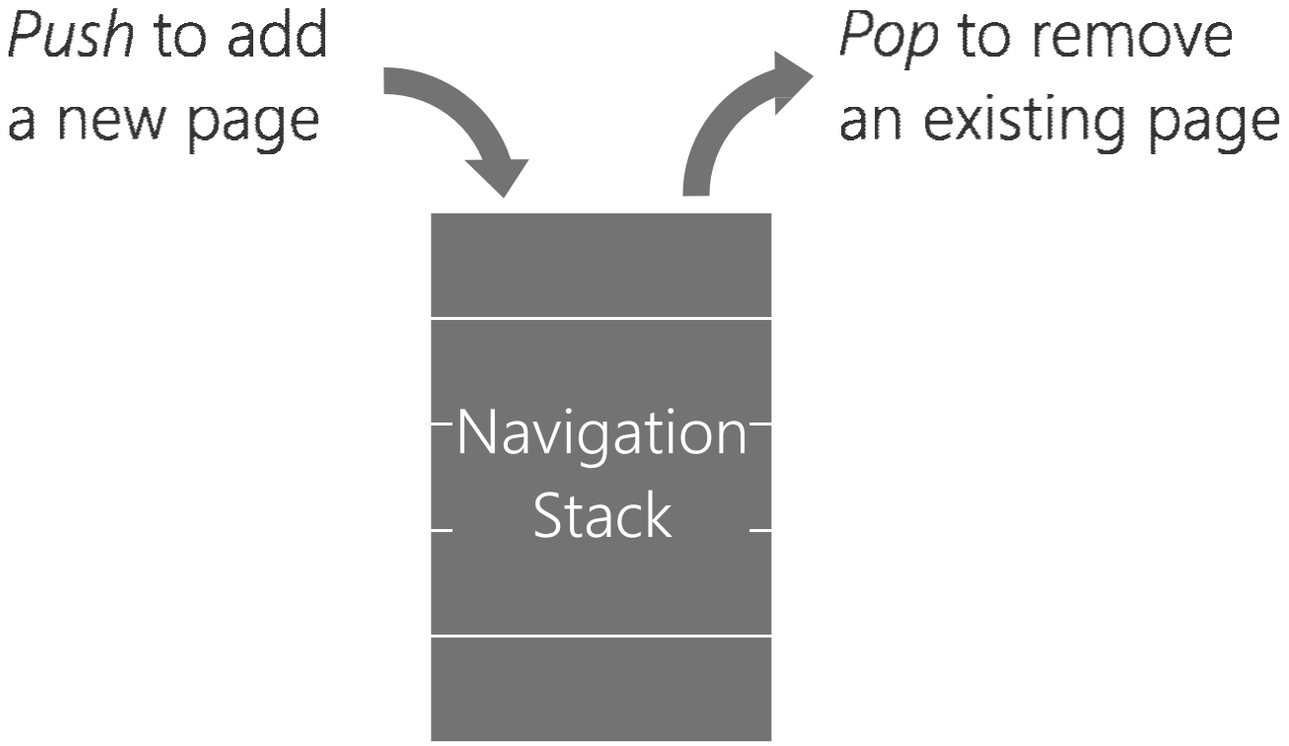


General

Specific

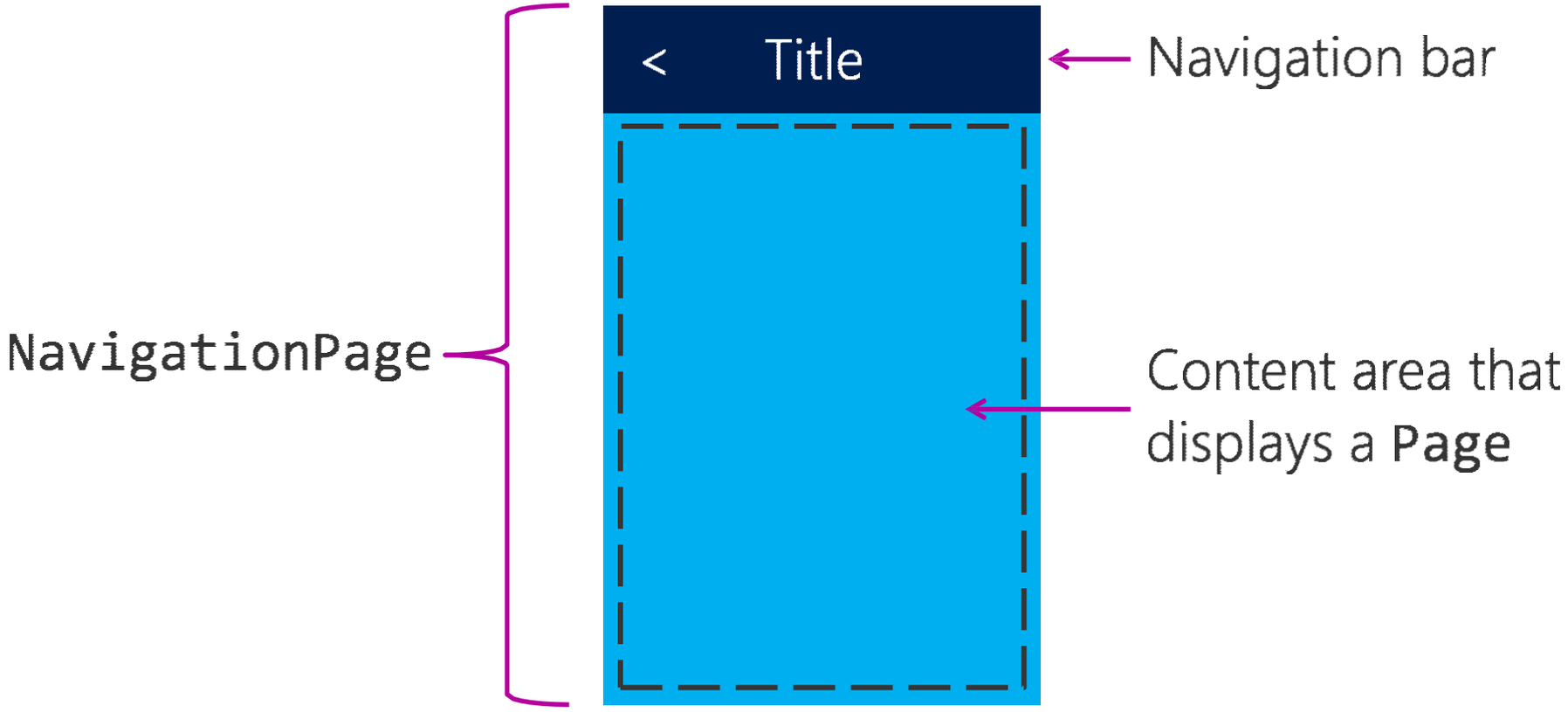
The Navigation Stack

Stack navigation uses a *stack* to store the sequence of active pages



What is NavigationPage?

`NavigationPage` is a Xamarin.Forms page that implements stack navigation and provides a navigation bar



NavigationPage instance

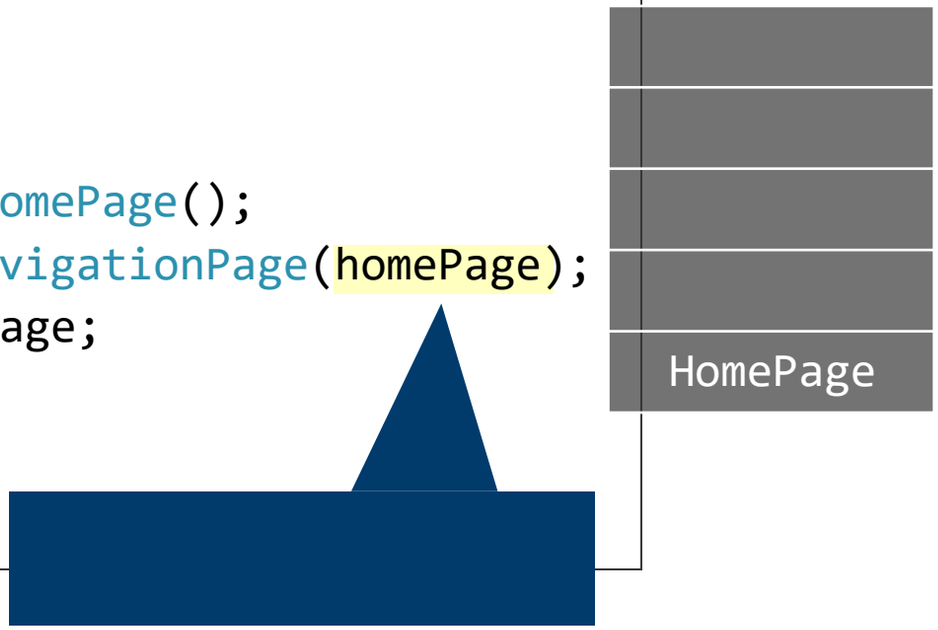
You must have a **`NavigationPage`** instance in your visual tree to use the navigation stack; common practice is to use one as your App's **`MainPage`**

```
public partial class App : Application
{
    public App()
    {
        ...

        var navPage = new NavigationPage();
        this.MainPage = navPage;
        ...
    }
}
```

How to load your initial page

You typically pass your initial page to the **`NavigationPage`** constructor

```
public partial class App : Application
{
    public App()
    {
        ...
        var homePage = new HomePage();
        var navPage = new NavigationPage(homePage);
        this.MainPage = navPage;
        ...
    }
}
```



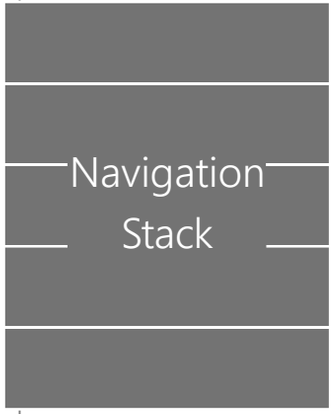
Navigation methods

`NavigationPage` provides methods to push and pop pages

Add a page

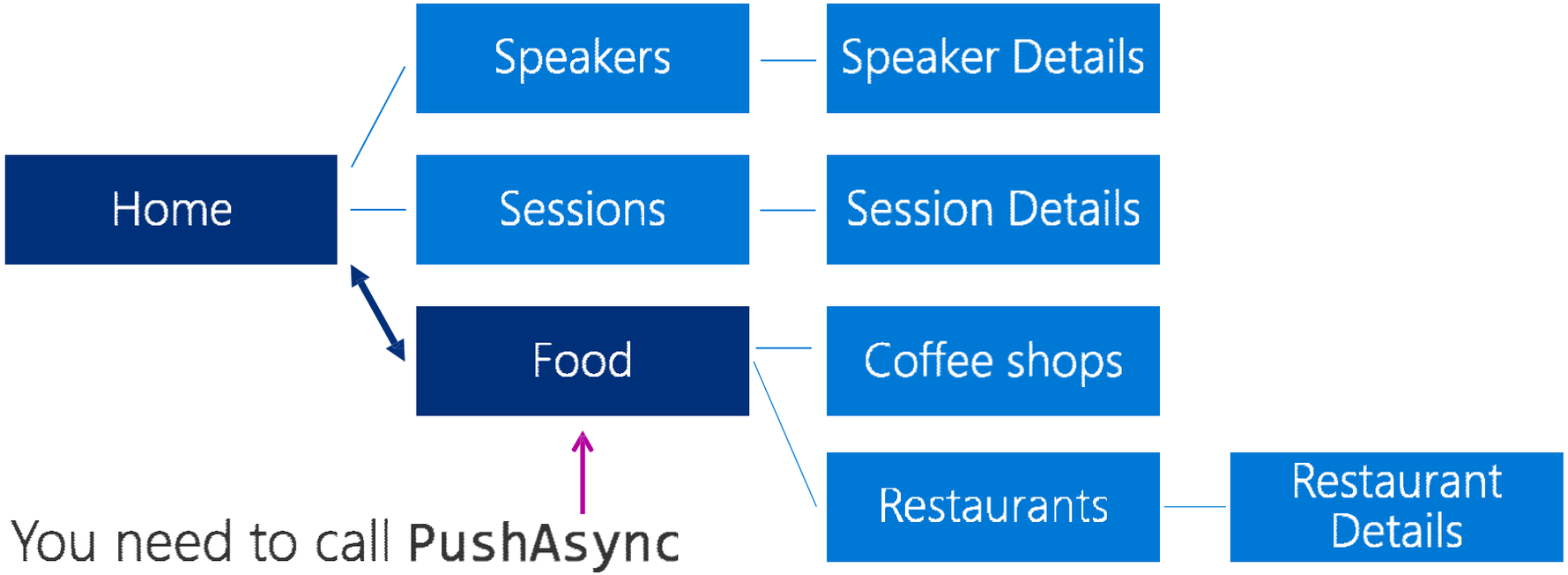
Remove the top (visible) page

```
public class NavigationPage : ...
{   ...
    Task PushAsync(Page page, bool animated);
    Task PushAsync(Page page); // 'animated' is true

    Task<Page> PopAsync(bool animated);
    Task<Page> PopAsync(); // 'animated' is true
}
```



Distributed navigation logic

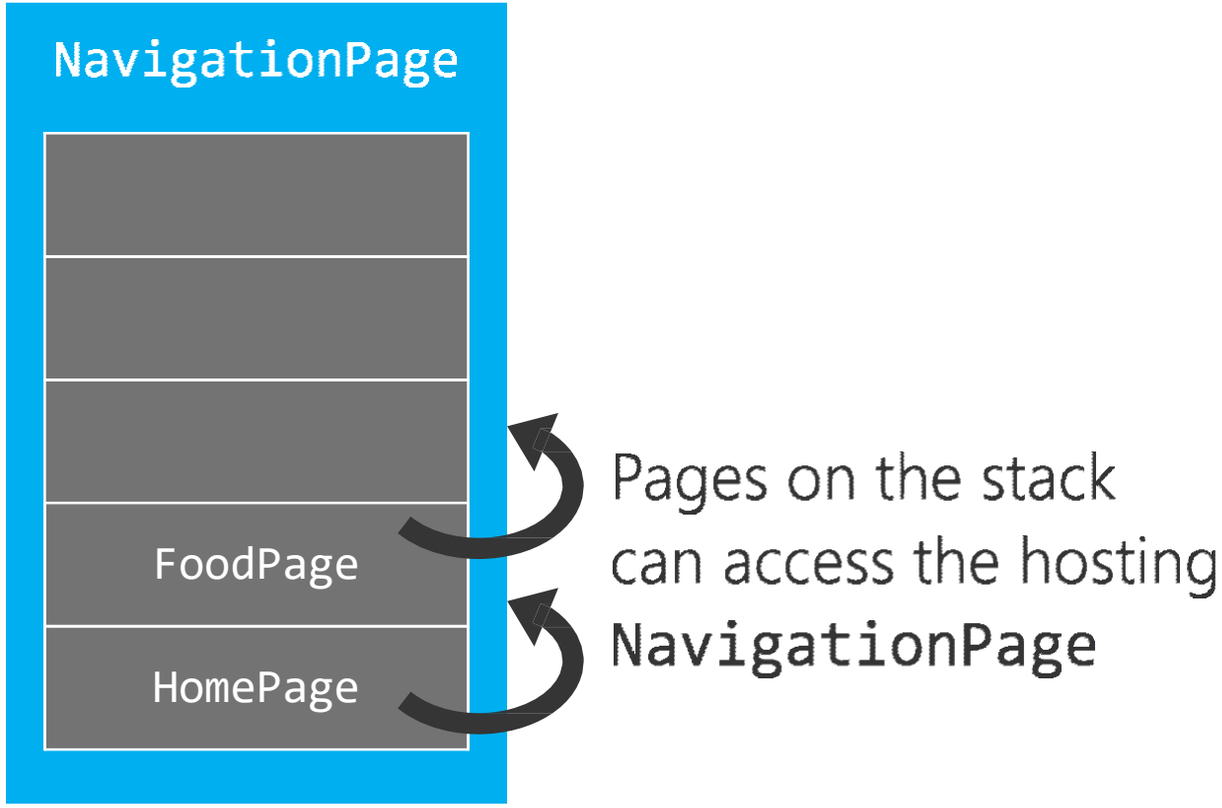
Typically, you initiate navigation push/pop operations from inside your app's pages



You need to call **PushAsync** from here to navigate to the restaurants page

How to access the Navigation Stack

Every page on the Navigation Stack has a **`Navigation`** property that lets you access the hosting **`NavigationPage`**



INavigation interface

A page's **Navigation** property is an **INavigation** reference; it refers to the hosting **NavigationPage** indirectly

```
public partial class FoodPage : ContentPage
{
    async void OnRestaurantsClicked(object sender, EventArgs e)
    {
        INavigation nav = this.Navigation;
        ...
    }
    ...
}
```

The **INavigation** interface defines push/pop methods

The property refers to a proxy object that forwards calls to the **NavigationPage**

How to push a page

To navigate to a new page, call **`PushAsync`** on the current page's **`Navigation`** property

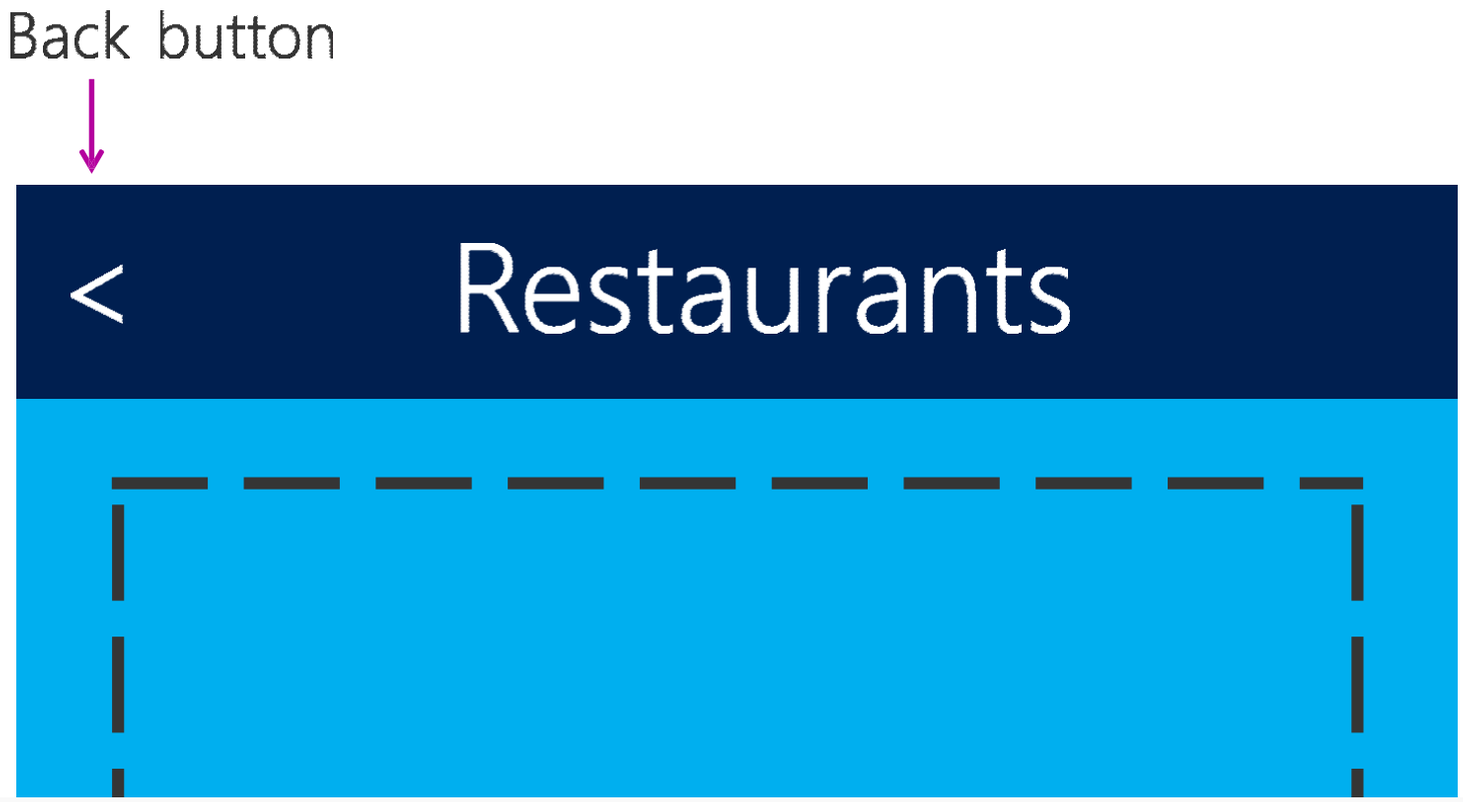
```
public partial class FoodPage : ContentPage
{
    async void OnRestaurantsClicked(object sender, EventArgs e)
    {
        var destination = new RestaurantsPage(...);

        await this.Navigation.PushAsync(destination);
    }
    ...
}
```

Instantiate the target page

Navigate forward

Automatic pop

The navigation bar includes a back button that automatically pops the stack



Built-in device back-buttons also automatically pop the navigation stack.

Programmatic pop

You can programmatically return to the previous page by calling **PopAsync** on the current page's **Navigation** property

```
public partial class RestaurantsPage : ContentPage
{
   async void OnCancelClicked(object sender, EventArgs e)
   {
      ...
      await this.Navigation.PopAsync();
   }
   ...
}
```

Navigate back

Push and pop methods are async

Most navigation operations are asynchronous; the Navigation Stack is not guaranteed to be ready for further use until the **`async`** operation completes

Wait until the first operation completes…

…before starting another operation

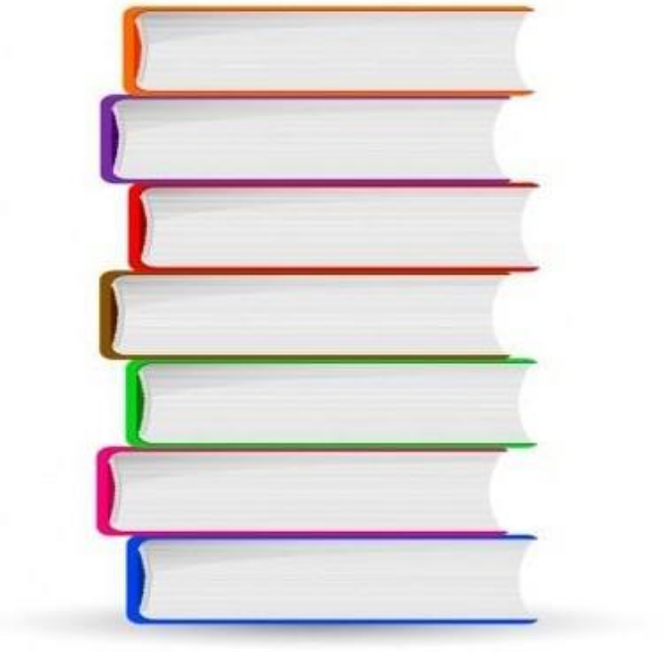
```
async void OnHomeClicked()
{
    await this.Navigation.PopAsync();
    await this.Navigation.PopAsync();
}
```

Exercise

Implement stack navigation

Summary

1. Create a NavigationPage
2. Navigate forward and backward through a stack of Pages



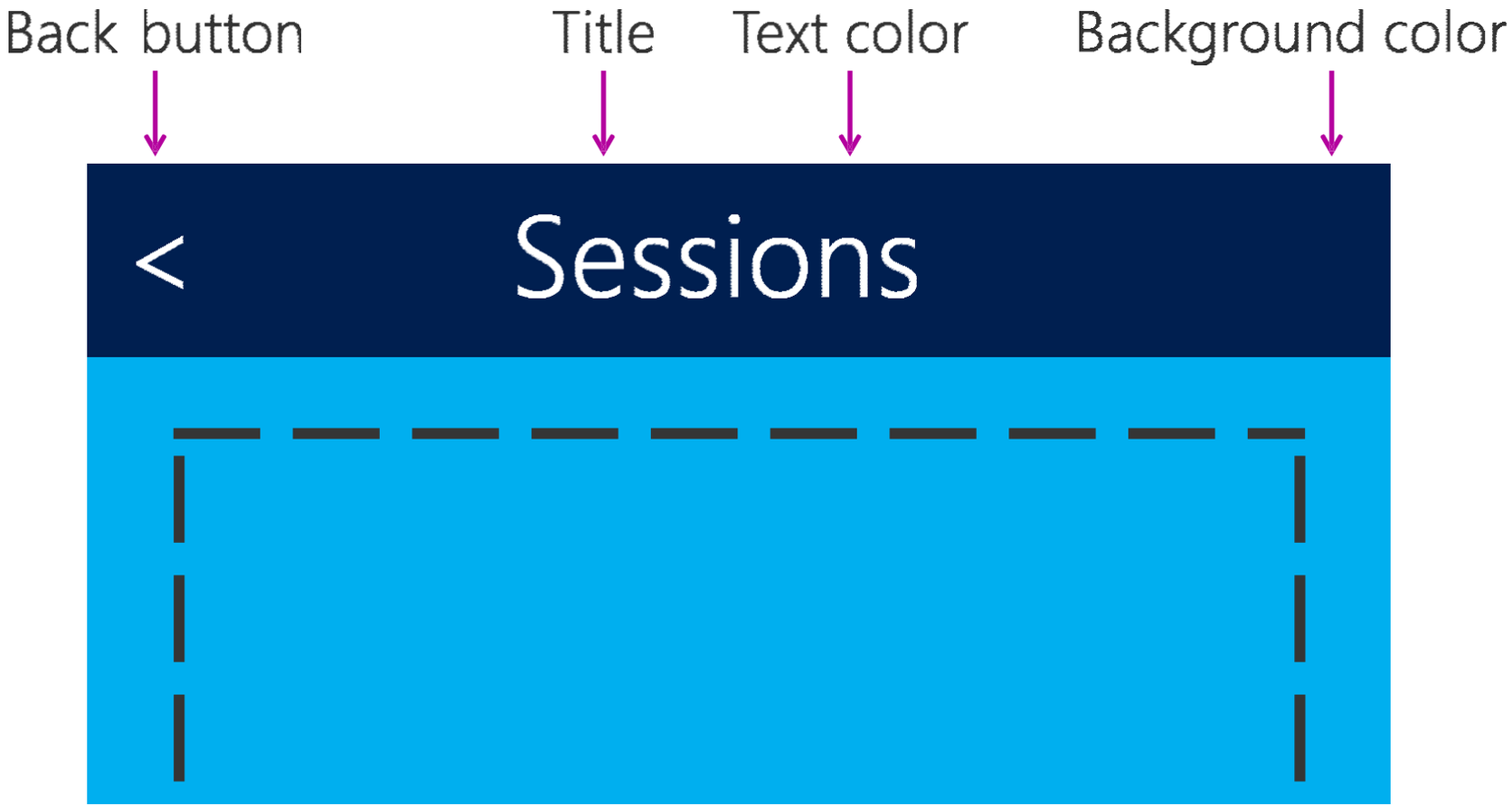
Configure the navigation bar

Tasks

1. Display the page title
2. Set background and text colors
3. Get notified when the back button is clicked
4. Hide the back button
5. Set the back button title for iOS

Navigation bar anatomy

The navigation bar provides UI elements that describe the current page



Navigation bar title

The navigation bar displays the **`Page`**'s **`Title`** property

```
<ContentPage ...
   x:Class="MyApp.SessionsPage"
   Title="Sessions">
   ...
</ContentPage>
```

Can set in XAML

```
public partial class SessionsPage : ContentPage
{  ...
   public SessionsPage()
   {
      InitializeComponent();

      this.Title = "Sessions";
   }
}
```

Can set in code

Navigation bar colors

The navigation bar lets you set the background color and the text color

```
public partial class App : Application
{
   public App()
   {
      InitializeComponent();

      var nav = new NavigationPage(new HomePage());

      nav.BarBackgroundColor = Color.Blue;
      nav.BarTextColor = Color.White;
}
```

Colors are instance properties of the navigation bar

Back button label in iOS

iOS displays the title of the *previous* page next to the back button

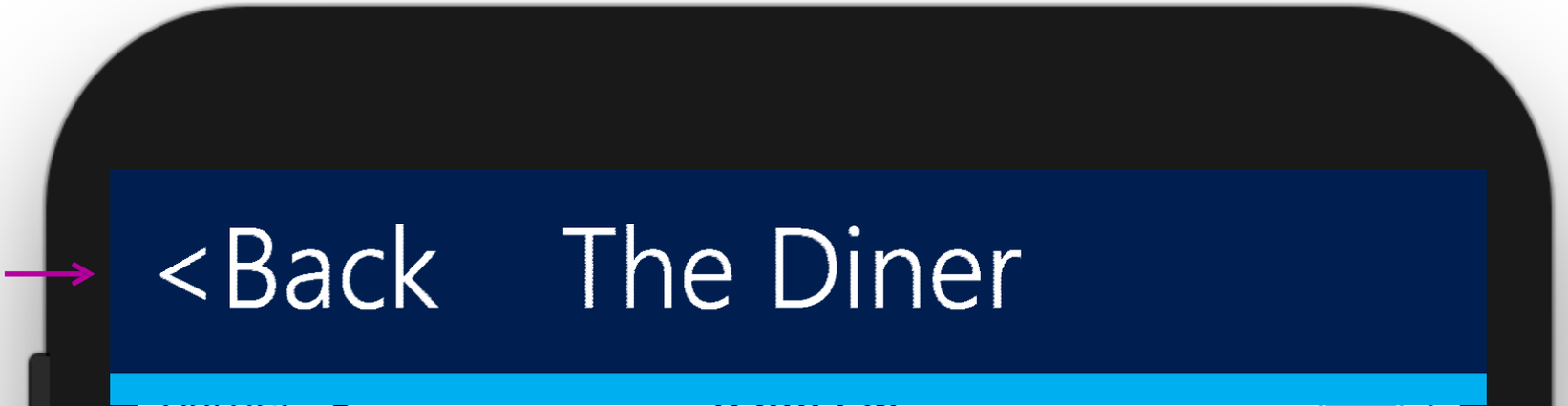
Custom back button label in iOS

`NavigationPage` lets you customize the text on the iOS back button; this is typically used to shorten the label to avoid conflict with the page title

Set a custom label on the target page (i.e. the page that is the target of the back button)

```
public partial class RestaurantsPage : ContentPage
{   ...
    public RestaurantsPage()
    {   ...
        this.Title = "Restaurants";
        NavigationPage.SetBackButtonTitle(this, "Back");
    }
}
```

Label shown in iOS



Back button notification

Xamarin.Forms notifies you when the user taps the navigation bar back button or the device back button

```
public partial class RestaurantsPage : ContentPage
{   ...
    protected override bool OnBackButtonPressed()
    {
        if (...)
            return true;

        if (...)
            return false;
    }
}
```

You will handle the button press (i.e. no automatic navigation performed)

Allow the default navigation (i.e. pop the stack)

Hide the back button

You can hide the navigation bar back button

Button is shown by default, you can hide it when needed

```
<ContentPage ...
   x:Class="MyApp.EditSchedulePage"
   NavigationPage.HasBackButton="False">
   ...
</ContentPage>
```

The back-button is automatically unavailable on the root page of the navigation stack because there is only one page on the stack

Group Exercise

Configure the navigation bar

Summary

1. Display the page title
2. Set background and text colors
3. Get notified when the back button is clicked
4. Hide the back button
5. Set the back button title for iOS

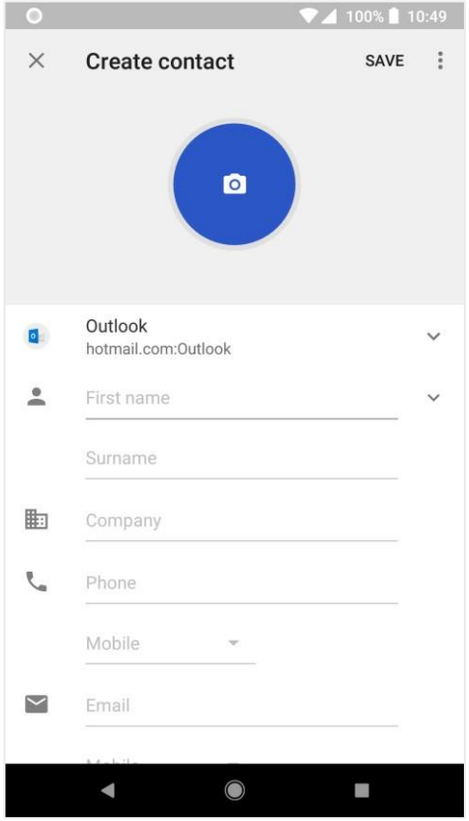
Display transitory content using modal pages

Tasks

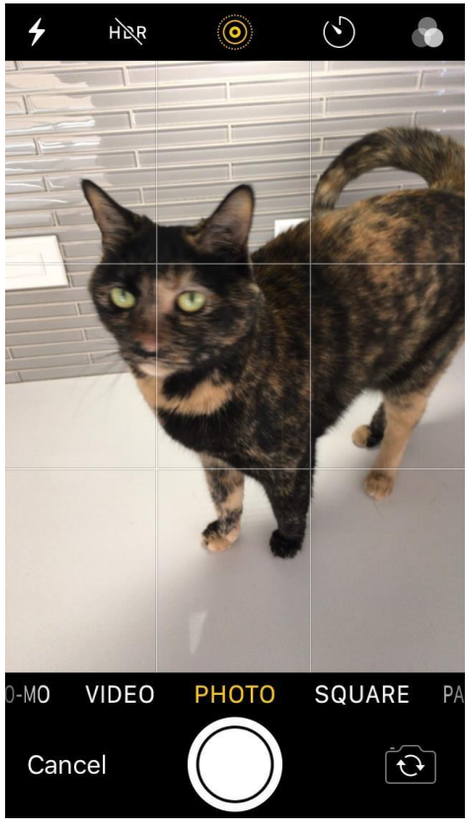
1. Use the Modal Stack to show a modal page
2. Decide when to use a modal page
3. Handle device back button use

Motivation

Some operations are atomic; it would be confusing or error prone to leave them in a half completed state



Add/edit contact

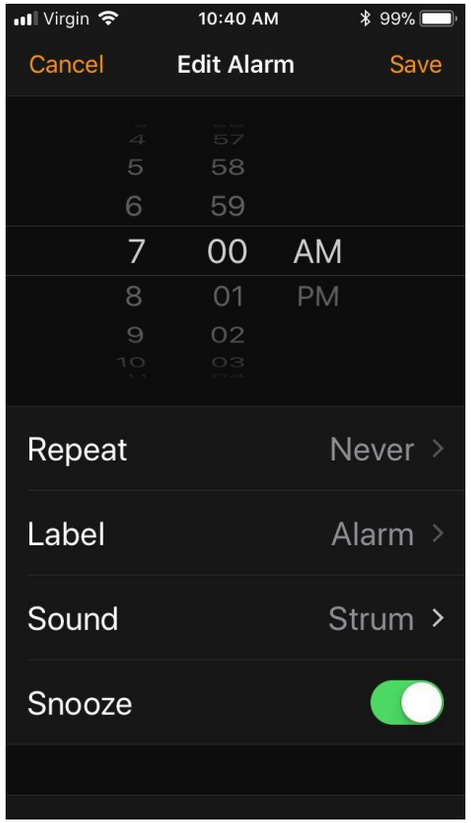


Take photo

What is a modal page?

A *modal page* is a page with restricted navigation options: the user must indicate they are finished before moving to a different page

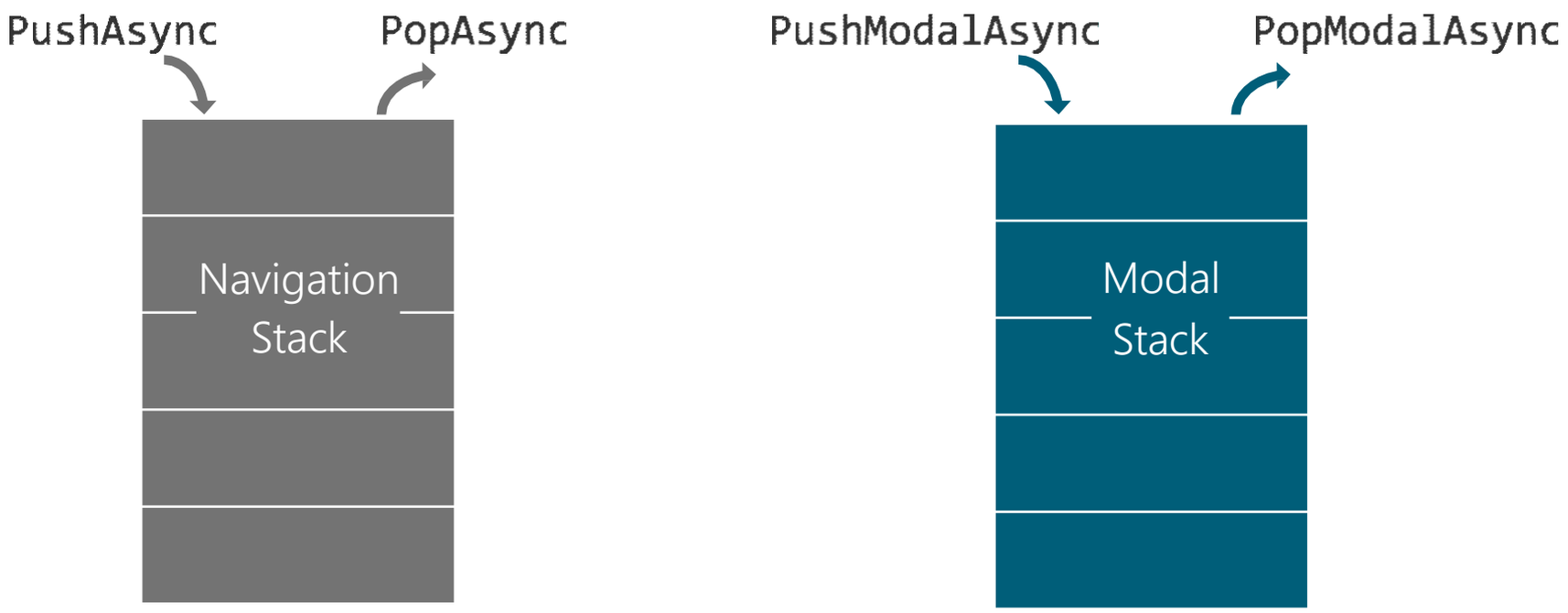
Provide in-page navigation to dismiss the page



Do not show your primary navigation UI

Navigation stack and Modal stack

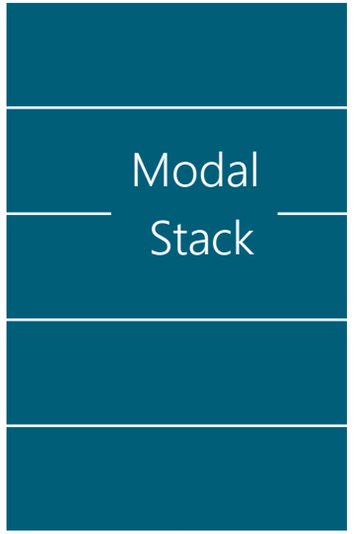
The Xamarin.Forms infrastructure maintains two stacks each with their own push/pop methods



Stack availability

The Modal Stack is always available so it works with navigation strategies like *tab* or *drawer* which may not have a **`NavigationPage`**

Only available on pages hosted in a **`NavigationPage`**



Available on every page

Stack services

The Navigation Stack and the Modal Stack offer different services tailored to their intended use

Navigation Stack	Modal Stack
Push/pop	Push/pop
Read-only stack access	Read-only stack access
Direct stack manipulation (e.g. insert/remove)	-
Only available with `NavigationPage` which includes nav bar, title, and back-button UI	-

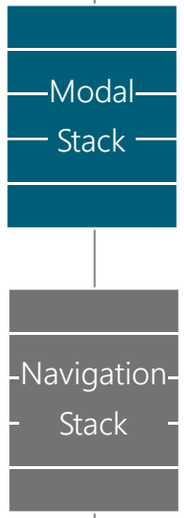
Full navigation solution

Limited to push/pop pages

Stack APIs

Both Stack APIs are available through a **Page**'s **Navigation** property

Page's **Navigation** property is of type **INavigation**




```
public interface INavigation
{   ...

    IReadOnlyList<Page> ModalStack { get; }
    Task        PushModalAsync(Page page);
    Task<Page>  PopModalAsync();

    IReadOnlyList<Page> NavigationStack { get; }
    Task        PushAsync(Page page);
    Task<Page>  PopAsync();

}
```

How to implement modal pages

Use the Modal Stack APIs to navigate to/from modal pages

Go to a modal page

```
public class AlarmsPage : ContentPage
{   ...
    async void OnEditClick(...) { await Navigation.PushModalAsync(new EditPage()); }
}
```

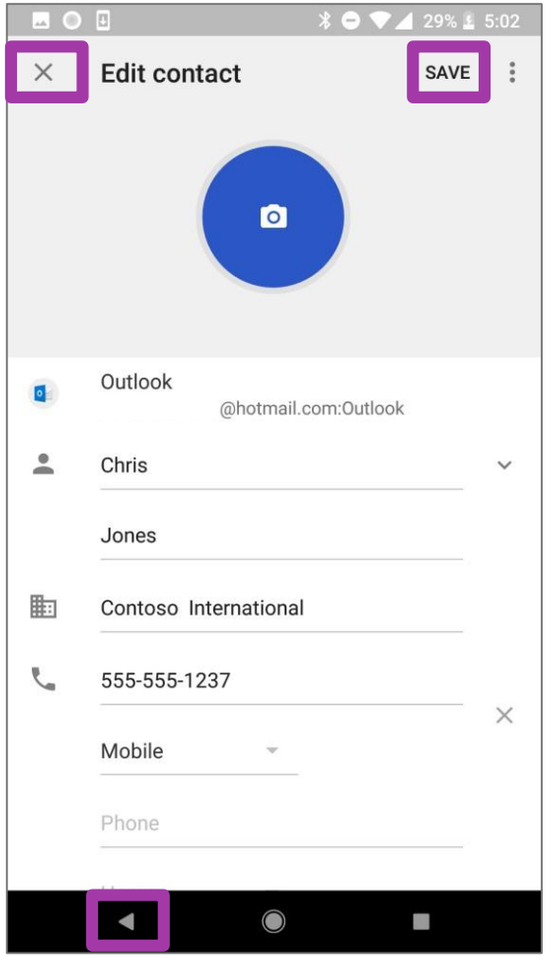
```
public class EditPage : ContentPage
{   ...
    async void OnSaveClick  (...) { ... await Navigation.PopModalAsync(); }
    async void OnCancelClick(...) { ... await Navigation.PopModalAsync(); }
}
```

Return

Device back buttons and modal pages

Device back buttons allow the user to bypass the page's navigation to leave

We want the user to use these...



...but this button lets them leave without choosing save/cancel

When to handle device back button on modal pages

You should customize the back button behavior whenever the default behavior is not immediately obvious

```
public partial class EditContactPage : ContentPage
{   ...
    protected override bool OnBackButtonPressed()
    {
        ...
    }
}
```

Ask user for confirmation before leaving page

Exercise

Display a modal page

Discussion: when to use a modal page

Why are each of these pages reasonable candidates for modal pages?

- Take photo
- Contact list
- Login
- Register new user account
- Home page
- Settings

Summary

1. Use the Modal Stack to show a modal page
2. Decide when to use a modal page
3. Handle device back button use

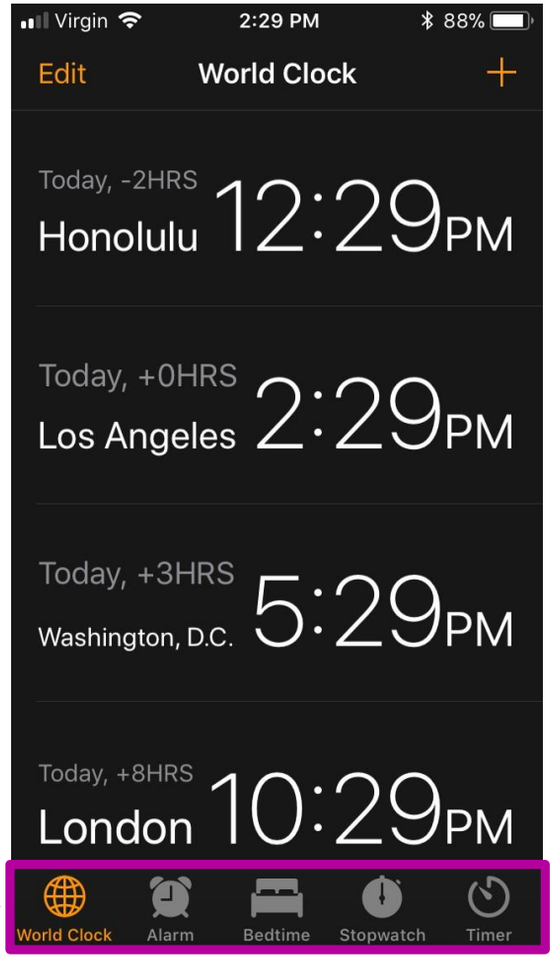
Switch among a small group of pages using tab navigation

Tasks

1. Create a **TabbedPage**
2. Populate a **TabbedPage** with pages
3. Customize tab appearance

What is tab navigation?

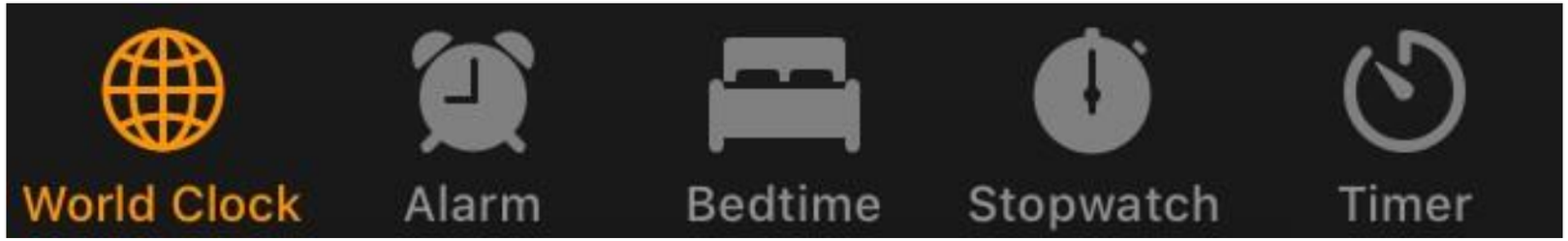
Tab navigation is a navigation paradigm that uses always visible UI elements called *tabs* to change the visible page



User selects a tab to switch to that page

When to use tab navigation?

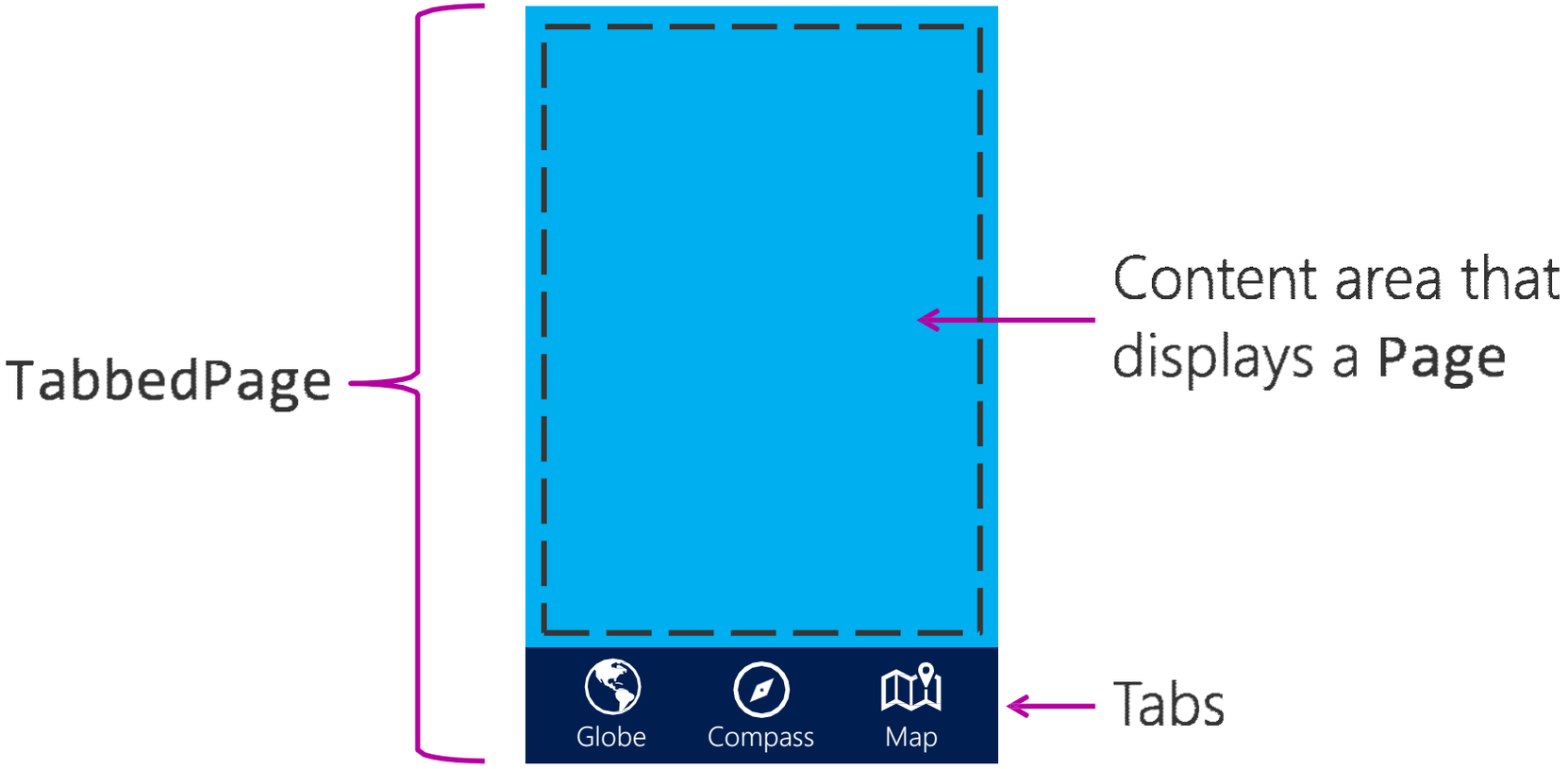
Tab navigation is typically used when the app has 3 5 screens of equal importance



Tabs are visible which helps users discover all app features

What is TabbedPage?

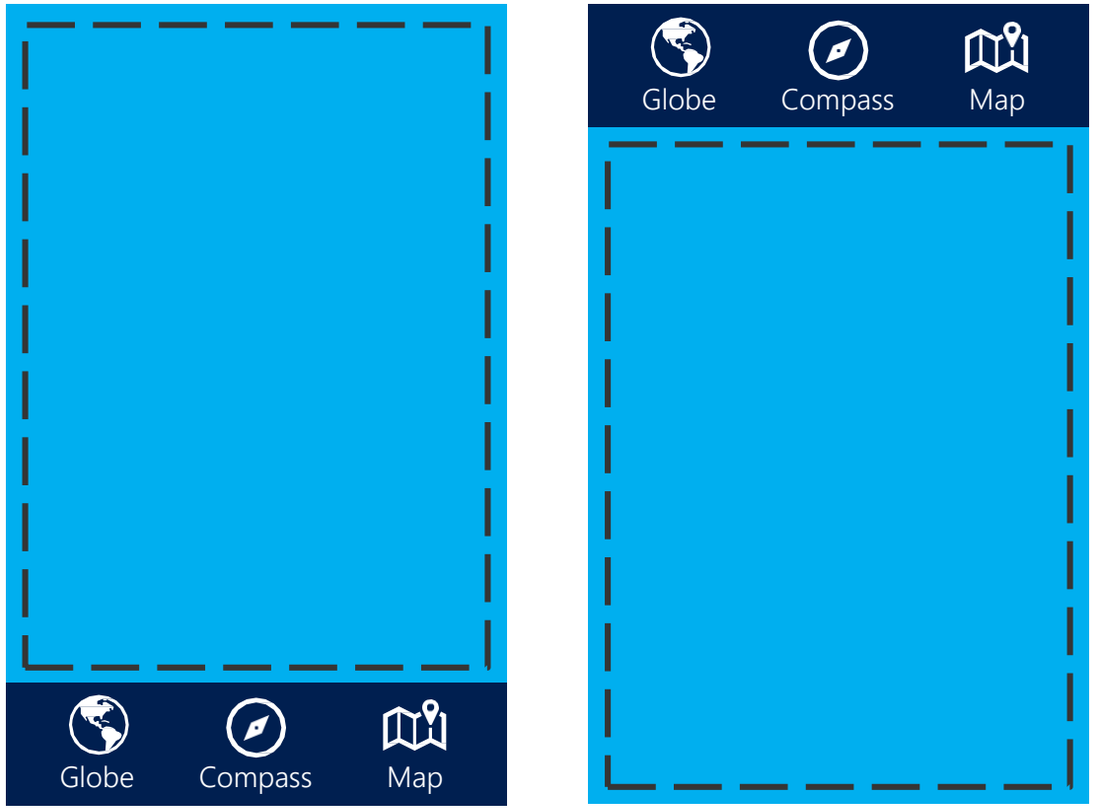
TabbedPage is a Xamarin.Forms page that displays a set of tabs and switches content when the user selects a tab



Tab position

The position of the tab strip in a **TabbedPage** automatically conforms to the style guidelines of each platform

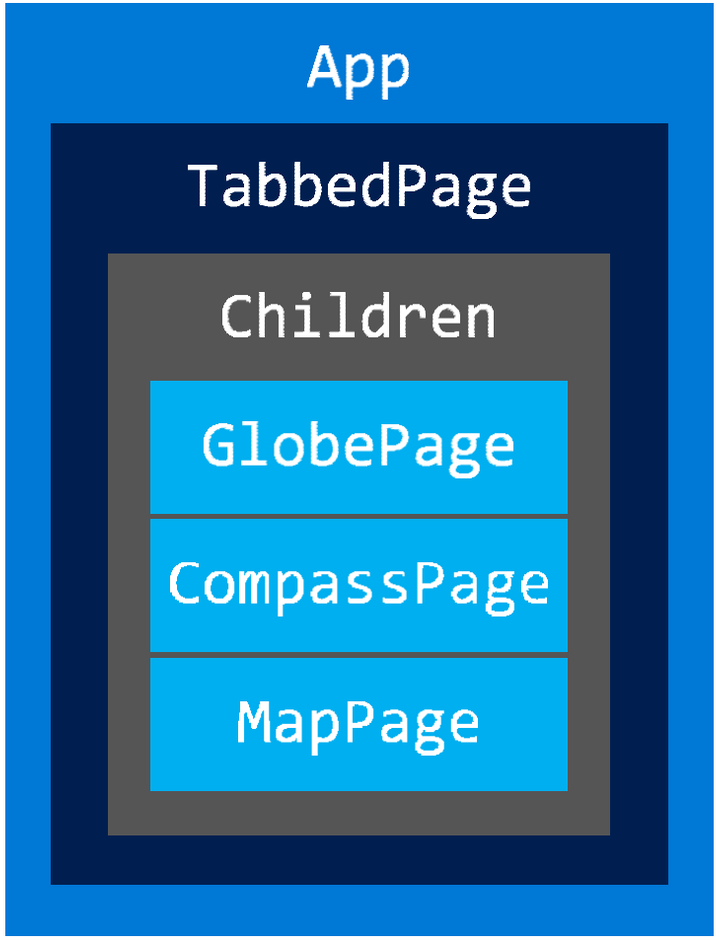
Tabs may appear at the top or the bottom of the UI



How to use TabbedPage

You create a **TabbedPage** instance and populate it with your pages

TabbedPage instance is assigned to the App's **MainPage** property



TabbedPage holds a collection of **Pages**

Create a tab page directly in C#

You can create a **TabbedPage** in C#, populate it with pages and use the **TabbedPage** instance as your app's **MainPage**

```
public partial class App : Application
{
    public App()
    {
        var tabbedPage = new TabbedPage();

        tabbedPage.Children.Add(new GlobePage());
        tabbedPage.Children.Add(new CompassPage());
        tabbedPage.Children.Add(new MapPage());

        MainPage = tabbedPage;
    }
}
```

Create the TabbedPage

Add the pages

Display in your UI

Create a TabbedPage derived class

You can create a **TabbedPage** derived class and use that as your App's **MainPage**; this allows you to add your pages in XAML

HomePage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<TabbedPage ...
   x:Class="WorldTraveler.HomePage">
   <TabbedPage.Children>
       <local:GlobePage   ... />
       <local:CompassPage ... />
       <local:MapPage     ... />
   </TabbedPage.Children>
</TabbedPage>
```

HomePage.xaml.cs

```
public partial class HomePage : TabbedPage
{
   ...
}
```

The "Tabbed Page" item template in Visual Studio will generate this page for you.

TabbedPage tabs

`TabbedPage` auto generates its tabs from the pages in its **`Children`** collection

```
<TabbedPage.Children>
   <local:GlobePage   Title="Globe"   Icon="globe.png" />
   <local:CompassPage Title="Compass" Icon="compass.png" />
   <local:MapPage     Title="Map"     Icon="map.png" />
</TabbedPage.Children>
```

Tabs show each child page's **`Title`** and **`Icon`**

Customize tab colors

TabbedPage lets you customize the tab background and text colors

Set your app's preferred colors

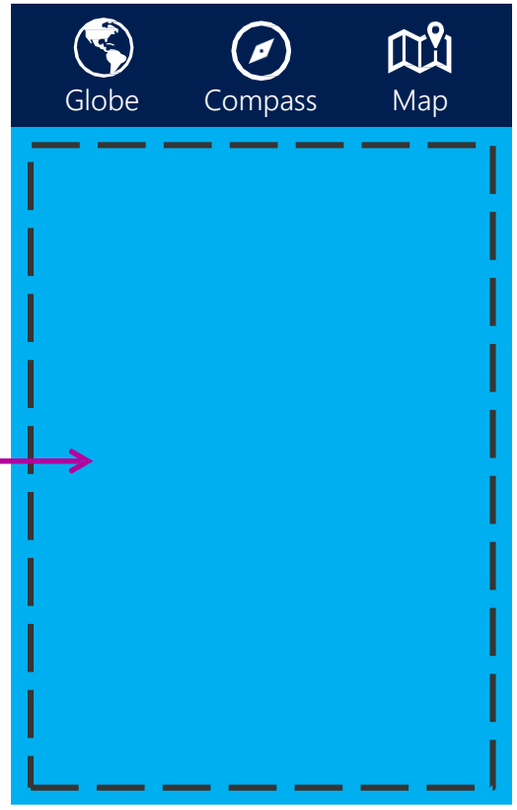
```
<TabbedPage ...
   xmlns:local="clr-namespace:WorldTraveler"
   x:Class="WorldTraveler.MainPage"
   BarBackgroundColor="Blue"
   BarTextColor="White">

   <TabbedPage.Children>
      <local:GlobePage   Title="Globe"   Icon="globe.png" />
      <local:CompassPage Title="Compass" Icon="compass.png" />
      <local:MapPage     Title="Map"     Icon="map.png" />
   </TabbedPage.Children>

</TabbedPage>
```

Tab switching is automatic

TabbedPage automatically swaps the displayed page when the user selects a tab

1. User selects a tab

2. Content updates automatically



Exercise

Implement tab navigation

Summary

1. Create a **TabbedPage**
2. Populate a **TabbedPage** with pages
3. Customize tab appearance

Microsoft

Thank You!