**Session\_1 Getting Started - First App**

Lesson 3 What is Ionic?

Lesson 7 Create first app

Lesson 10 Course structure

**Session\_2 Mastering the Basics - Lesson\_37\_Finished\_Section\_code\Section Code (Finished**

Lesson\_15 Starters

Lesson\_16 Structure of an Ionic project

Lesson\_17 How Ionic Works?

Lesson 18 Pages vs Components

Lesson 20 Initialize navigation in Ionic

Lesson\_21 Generatic Pages in Ionic 3 - bugfixes

Lesson\_21 Generate Pages Adding navigation

Lesson\_25 Passing Data Between pages

Lesson 26 Navigate back

Lesson 29 Quick way of navigation

Lesson 34 Theming your application

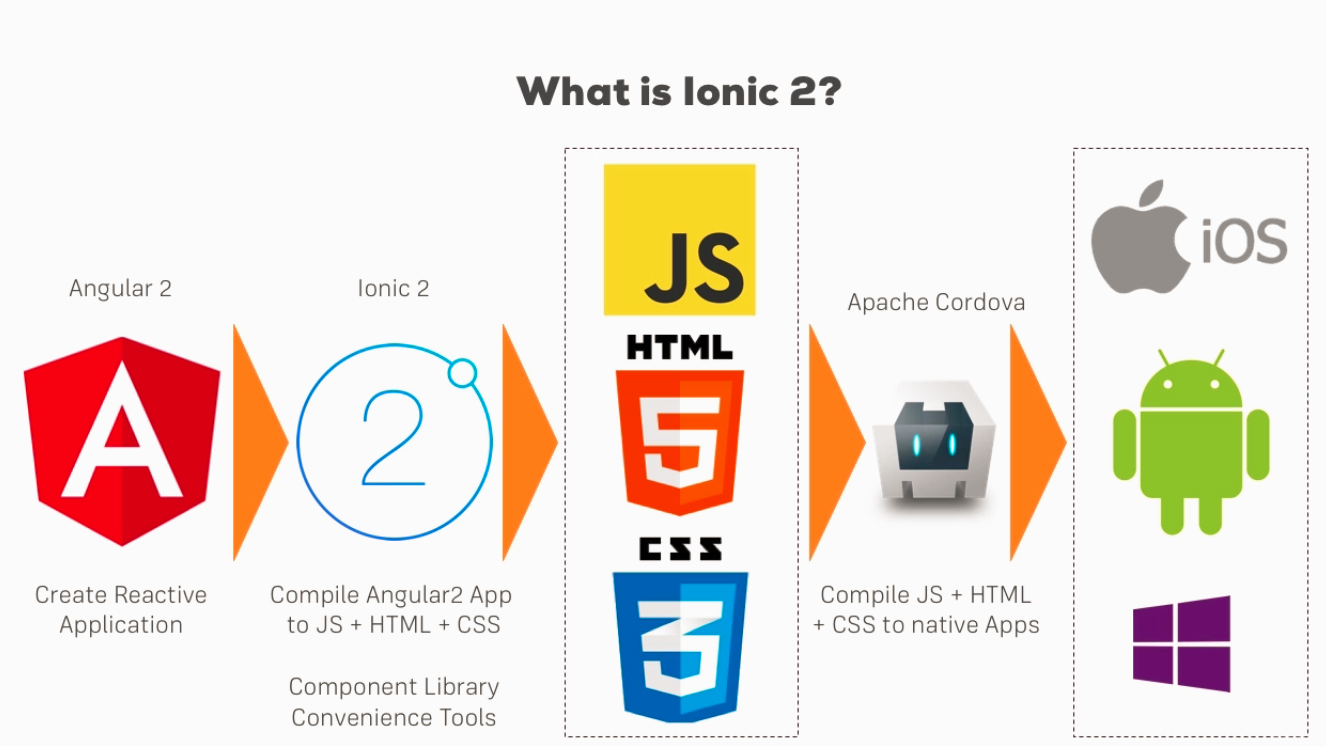
Lesson 35 CSS Utilities for your application

Lesson 30 Configuring Page Transition

**Session\_3\_Navigation\_Pages&Components - Section-Code-Finished**

**Session\_1**

**What is Ionic?**



**Lesson 7 Create first app**

**C:\Users\Lenovo\Desktop\Isti\Programozás-\Kurzusok\Udemy\Ionic\Session\_1\First App\firstapp**

1. Install Ionic + Cordova

**npm install -g ionic cordova**

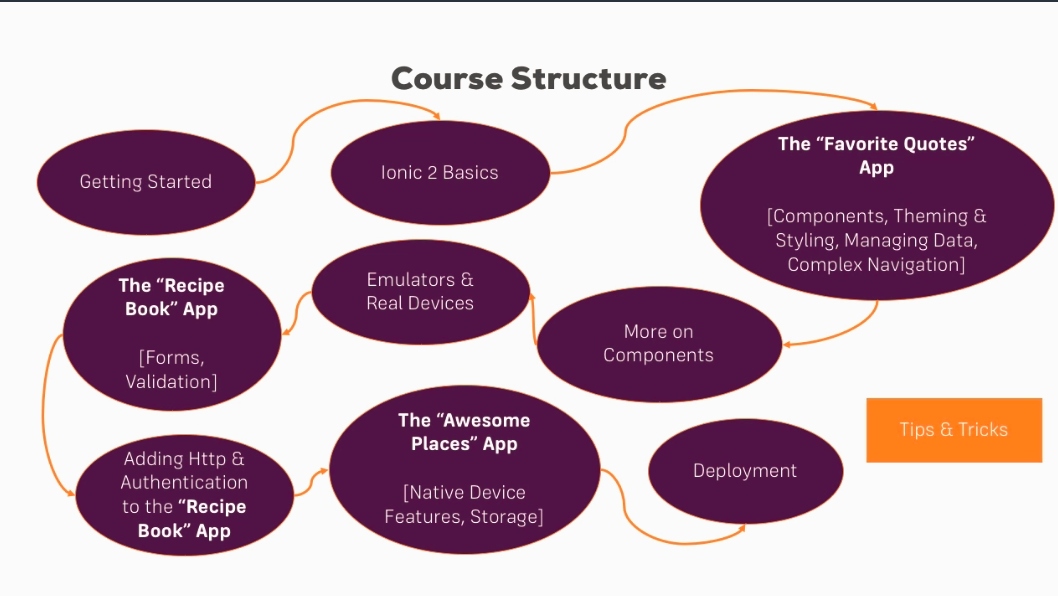
1. Navigate to the folder & run

**ionic start firstapp --type=ionic-angular**

1. Once it finished run

**Ionic serve**

**Lesson 10 Course Structure**



**Session\_2 Mastering the Basics**

**Lesson\_15 Starters**

There are quite a few prebuilt starter code packages.

<https://ionicframework.com/docs/cli/starters.html>

**Lesson\_16 Structure of an Ionic project**

* **Hooks** - For cordova mobile process event hooks
* **Platforms** – In which platforms would you like to test it
* **Plugins -**  contains the installed specific cordova plugins to interact with the mobile elements

Src

**App** – where the app starts

**Assets -**  contains all of the the static files

**Pages –** Contains the different route pages

**Theme –** for styling

**Declaration.d.ts** – to tell angular global variables

**Lesson\_17 How Ionic Works**?

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

<meta charset="UTF-8">

<title>Ionic App</title>

<meta name="viewport" content="viewport-fit=cover, width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">

<meta name="format-detection" content="telephone=no">

<meta name="msapplication-tap-highlight" content="no">

<link rel="icon" type="image/x-icon" href="assets/icon/favicon.ico">

<link rel="manifest" href="manifest.json">

<meta name="theme-color" content="#4e8ef7">

<!-- add to homescreen for ios -->

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<!-- cordova.js required for cordova apps (remove if not needed) -->

<script src="cordova.js"></script>

<!-- un-comment this code to enable service worker

<script>

if ('serviceWorker' in navigator) {

navigator.serviceWorker.register('service-worker.js')

.then(() => console.log('service worker installed'))

.catch(err => console.error('Error', err));

}

</script>-->

<link href="build/main.css" rel="stylesheet">

</head>

<body>

<!-- Ionic's root component and where the app will load -->

<ion-app></ion-app>

<!-- The polyfills js is generated during the build process -->

<script src="build/polyfills.js"></script>

<!-- The vendor js is generated during the build process

It contains all of the dependencies in node\_modules -->

<script src="build/vendor.js"></script>

<!-- The main bundle js is generated during the build process -->

<script src="build/main.js"></script>

</body>

</html>

This is where we start,

1. **<ion-app></ion-app>**
2. **Main.js**, **polyfill.js**, **cordova.js** fill add functionality, conversion to browser and mobile
3. **App.module.ts**
4. **App.component.ts**

import { NgModule, ErrorHandler } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';

import { MyApp } from './app.component';

import { HomePage } from '../pages/home/home';

import { TabsPage } from '../pages/tabs/tabs';

import { StatusBar } from '@ionic-native/status-bar';

import { SplashScreen } from '@ionic-native/splash-screen';

@NgModule({

declarations: [

MyApp,

HomePage,

],

imports: [

BrowserModule,

IonicModule.forRoot(MyApp)

],

bootstrap: [IonicApp],

entryComponents: [

MyApp,

HomePage,

],

providers: [

StatusBar,

SplashScreen,

{provide: ErrorHandler, useClass: IonicErrorHandler}

]

})

export class AppModule {}

* **IonicModule.forRoot(MyApp)** 🡪 This will be the starting point for our ionic app
* **bootstrap: [IonicApp],** 🡪 This will be the bootstrapped global ionic object which will give us functionality
* **entryComponents: []** -> You have to declare here the created pages

export class MyApp {

rootPage:any = TabsPage;

constructor(platform: Platform, statusBar: StatusBar, splashScreen: SplashScreen) {

platform.ready().then(() => {

// Okay, so the platform is ready and our plugins are available.

// Here you can do any higher level native things you might need.

statusBar.styleDefault();

splashScreen.hide();

});

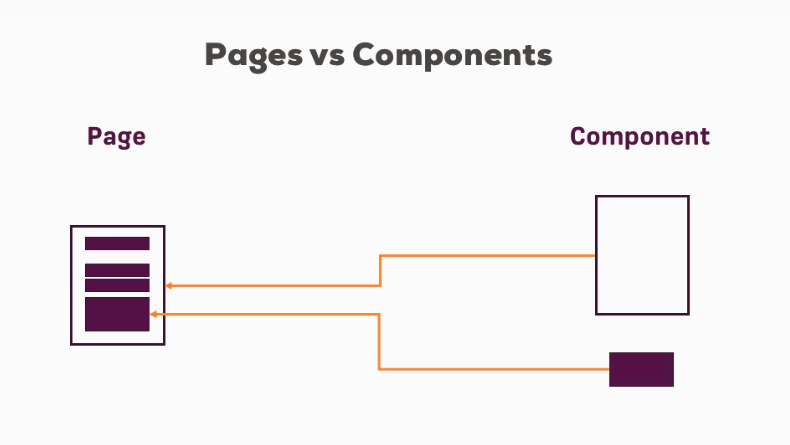
}

}

So this is like lifecycle hook to execute some code when the app starts

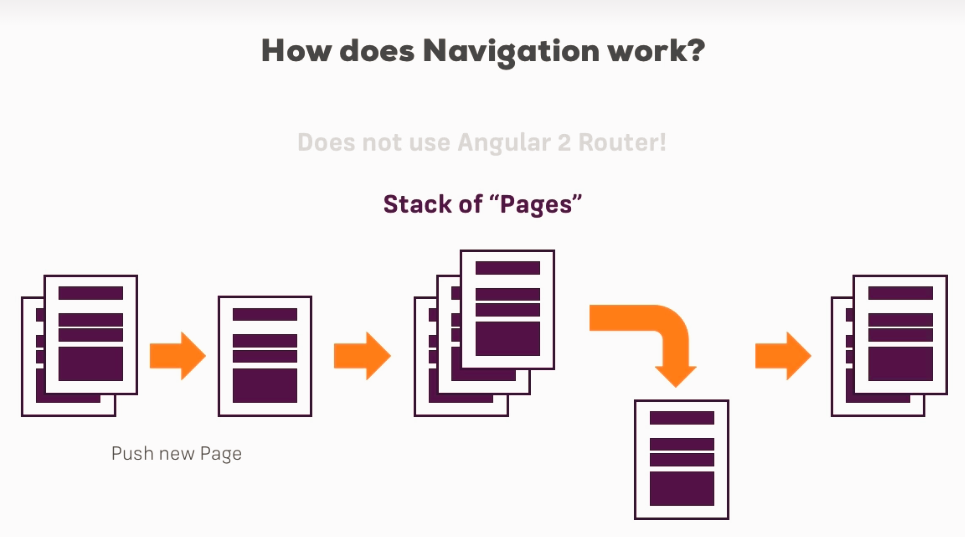
**Lesson 18 Pages vs Components**

**Pages**  take the whole screen, while the component might take the whole screen or just a part of it.



**Lesson 19 Navigation**

Instead of routing, ionic use a stack of pages to navigate.



**Lesson 20 Initialize navigation in Ionic**

<ion-nav [root]="rootPage"></ion-nav>

It will initialize the stack of pages, and in **app.module.ts**  will contain all of the available pages which can be pushed or removed from the current stack.

We tell which component is the **rootPage**  in the **app.component.ts**

export class MyApp {

rootPage = HomePage;

constructor(platform: Platform) {

platform.ready().then(() => {

// Okay, so the platform is ready and our plugins are available.

// Here you can do any higher level native things you might need.

StatusBar.styleDefault();

Splashscreen.hide();

});

}

}

**Lesson\_21 Generatic Pages in Ionic 3 - bugfixes**

If your project uses Ionic 3 (check the package.json to see the version number), ionic generate page XY  will give you a slightly different output than shown in the videos:

a) You get an additional .module.ts  file: This file might contain a little bug, make sure it uses **IonicPageModule.forChild(...)** and also **import { IonicPageModule } from 'ionic-angular'**   and NOT IonicModule.forChild(...) !

b) Your page will be named **YourName**  and not YourNamePage . Refer to it as **YourName**  in your code then (and not YourNamePage ).

Besides that, it works the same. Make sure to add it to your **declarations[]** and **entryComponents[]** array in **AppModule** and you should be good to go.

**Also in order to use the CLI in updated Ionic 3 projects use this command**

npm install --save-dev @ionic/app-scripts@latest

**Lesson\_21 Generate Pages Adding navigation**

**C:\Users\Lenovo\Desktop\Isti\Programozás-\Kurzusok\Udemy\Ionic\Session\_2 Mastering the Basics\Lesson\_37\_Finished\_Section\_code\Section Code (Finished)\src\pages\**

1. type the following code in the firstApp folder

**Ionic generate page users**

There are a few default helper to define the html for the ionic app

* Ion-header
* Ion-Navbar
* Ion-content
* Padding will give us a nice padding
* Ion footer

**Adding Navigation**

1. Create a button in the **home.html** to call a navigation function on **home.ts**  which will use **NavController** built In object

**home.html**

Button created to call **onGoToUsers()**

<ion-content padding>

<!--<button ion-button [navPush]="usrPage">Users</button>-->

<button ion-button (click)="onGoToUsers()">Users</button>

</ion-content>

**home.ts**

Using **NavController** we add the next page to the **pageStack**

We can import the **UsersPage** and make it as a variable

import { Component } from '@angular/core';

import { UsersPage } from "../users/users";

import { NavController } from "ionic-angular";

@Component({

selector: 'page-home',

templateUrl: 'home.html'

})

export class HomePage {

usrPage = UsersPage;

constructor (private navCtrl: NavController) {}

onGoToUsers() {

**this.navCtrl.push(this.usrPage)**

.catch((error) => console.log('Access denied, Argument was ' + error));

}

}

**Lesson\_25 Passing Data Between pages**

**/users**

**1** create a button in the **users.html**

2 to call a navigator function in the **users.ts** to navigate to the **user** page and

3 pass a value as **navParams**.

4 **users** ts will listen to the navParams and display it in it’s html

**users.html**

<ion-content padding>

<p>The Users</p>

<button ion-button (click)="**onLoadUser**('Max')">User 'Max'</button>

<hr>

<button ion-button (click)="onLoadUser('Anna')">User 'Anna'</button>

<hr>

<button ion-button navPop>Go Back</button>

</ion-content>

**Users.ts**

onLoadUser(name: string) {

this.navCtrl.push(UserPage, **{userName: name});** //This is how we pass data to the next page

}

**User.ts**

So here we are listening to **NavParams**

import { Component, OnInit } from "@angular/core";

import { **NavParams**, NavController } from "ionic-angular";

@Component({

selector: 'page-user',

templateUrl: 'user.html'

})

export class UserPage implements OnInit {

name: string;

constructor(private navParams: NavParams, //With this we can listen to the recieved parameters

private navCtrl: NavController) {

}

ngOnInit() {

this.name = **this.navParams.get('userName');** //We get the recieved data from the navParams

}

onGoBack() {

// this.navCtrl.pop();

this.navCtrl.popToRoot();

}

}

**User.html**

Displaying the received name

<ion-content padding>

<p>Hi, I'm {{ name }}</p>

</ion-content>

**Lesson 26 Navigate back**

1. Create a button on the **user.html** to call a function
2. Create a function which uses **NavController** to pop(), or popToRoot() functions In **user.ts**

**User.html**

<ion-content padding>

<p>Hi, I'm {{ name }}</p>

<button ion-button (click)="**onGoBack**()">Confirm</button>

</ion-content>

**User.ts**

import { Component, OnInit } from "@angular/core";

import { NavParams, NavController } from "ionic-angular";

@Component({

selector: 'page-user',

templateUrl: 'user.html'

})

export class UserPage implements OnInit {

name: string;

constructor(private navParams: NavParams, //With this we can listen to the recieved parameters

private navCtrl: **NavController**) {

}

ngOnInit() {

this.name = this.navParams.get('userName'); //We get the recieved data from the navParams

}

onGoBack() {

// this.navCtrl.**pop();** //This will go back 1 layer

this.navCtrl.**popToRoot();** //This will go back to the rootLayer

}

}

**Lesson 29 Quick way of navigation**

**Important navPush,** can’t handle the **ionViewCanEnter** lifecycle hook if it is returning false

Navigate away with **navPush / navParams**

**Home.ts**

We can store the reference for an other pages as a property

import { Component } from '@angular/core';

import { UsersPage } from "../users/users";

import { NavController } from "ionic-angular";

@Component({

selector: 'page-home',

templateUrl: 'home.html'

})

export class HomePage {

usrPage = UsersPage;

//We can store the reference for an other pages as a property

}

**Home.html**

We can use [navPush] / [navParams]=”your-data” to navigate to different pages + passing some data

<button ion-button [**navPush**]="usrPage">Users</button>

Navigating back with **navPop**

**Users.html**

<button ion-button **navPop**>Go Back</button>

**Lesson 34 Theming your application**

<https://ionicframework.com/docs/theming/theming-your-app/>

theme/**variables.scss**

**Lesson 35 CSS Utilities for your application**

<https://ionicframework.com/docs/theming/css-utilities/>

**Lesson 30 Configuring Page Transition**

Besides the page you want to go to and data you want to pass along, you can pass a third argument to push() (and a first argument to pop()): [Navigation Options](https://ionicframework.com/docs/v2/api/navigation/NavController/#navoptions)

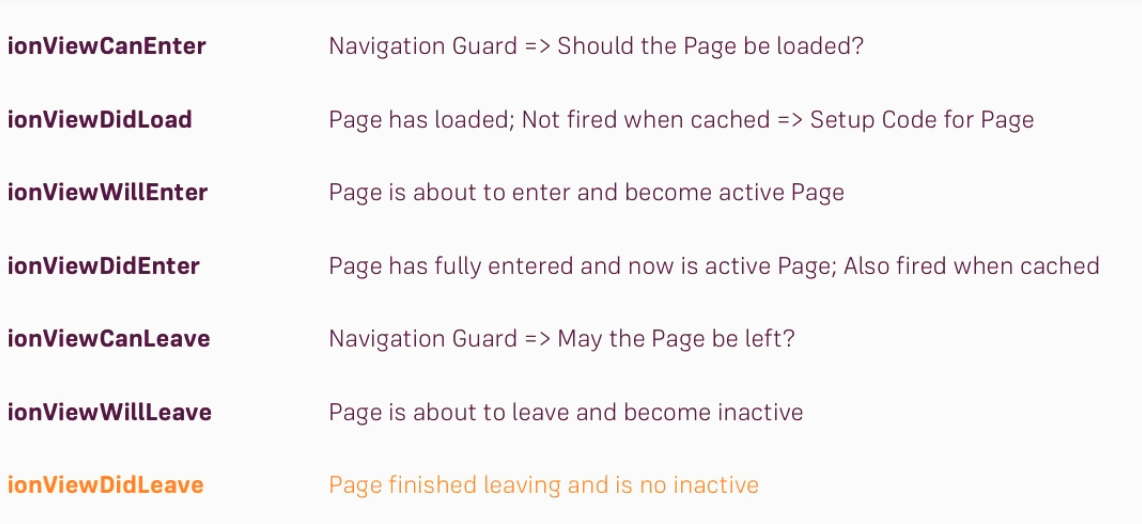
These options allow you to configure the page transition. You do set your own configuration by passing a JS object where you may set the following properties:

* **animate** (boolean): Whether or not the transition should animate.
* **animation** (string): What kind of animation should be used.
* **direction** (string): The conceptual direction the user is navigating. For example, is the user navigating forward, or back?
* **duration** (number): The length in milliseconds the animation should take.
* **easing** (string): The easing for the animation.

Example:

1. this.navCtrl.push(NewPage, {}, {
2. direction: 'back', // default for push is 'forward'
3. duration: 2000, // 2 seconds
4. easing: 'ease-out'
5. });

**Lesson 31 Understanding the lifecycle of a page**



**Lesson 31 Lifecycles in action**

**Users.ts**

export class UsersPage {

constructor (private navCtrl: NavController) {}

//Navigation Guard

ionViewCanEnter(): boolean | Promise<void> {

console.log('ionViewCanEnter');

const rnd = Math.random();

return rnd > 0.1;

}

//Everyhing is loaded which is required for the page

ionViewDidLoad() {

console.log('ionViewDidLoad');

}

//We are about to enter the page

ionViewWillEnter() {

console.log('ionViewWillEnter');

}

//We entered the page

ionViewDidEnter() {

console.log('ionViewDidEnter');

}

//We are about the leave the page

ionViewCanLeave(): boolean | Promise<void> {

console.log('ionViewCanLeave');

const promise = new Promise((resolve, reject) => {

setTimeout(() => {

resolve();

}, 1000);

});

return promise;

}

//We will leave the page right now!

ionViewWillLeave() {

console.log('ionViewWillLeave');

}

//We left the page

ionViewDidLeave() {

console.log('ionViewDidLeave');

}

//we will destroy the page

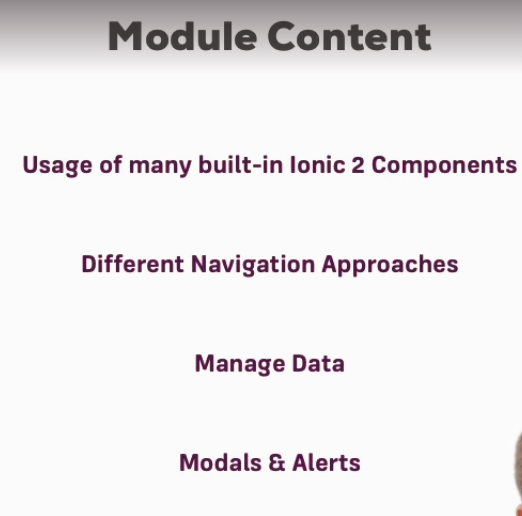
ionViewWillUnload() {

console.log('ionViewWillUnload');

}

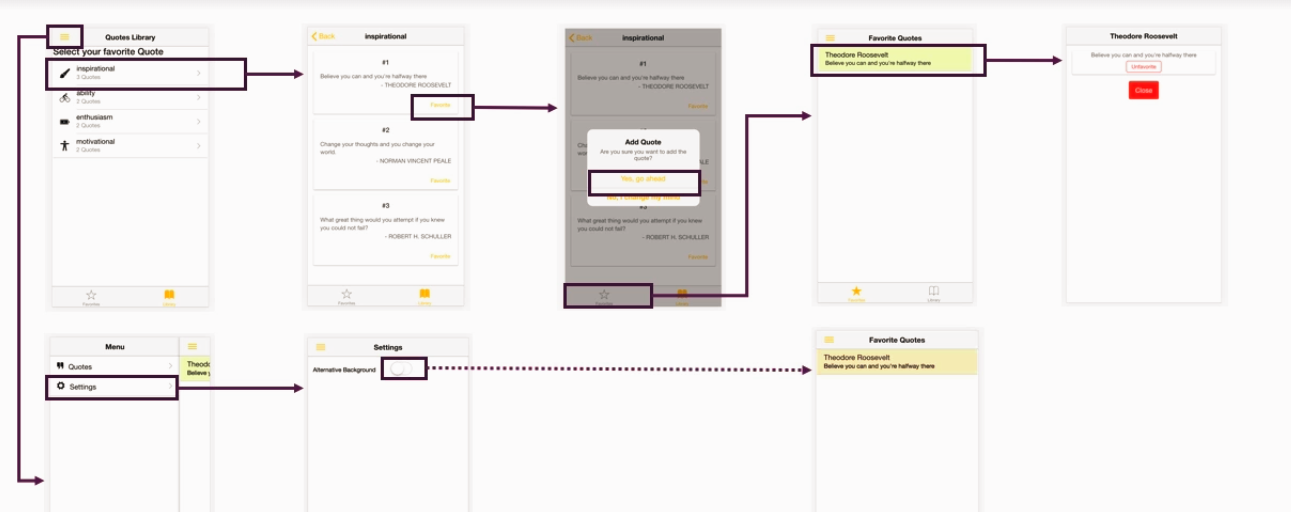
}

**Session\_3\_Navigation\_Pages&Components**



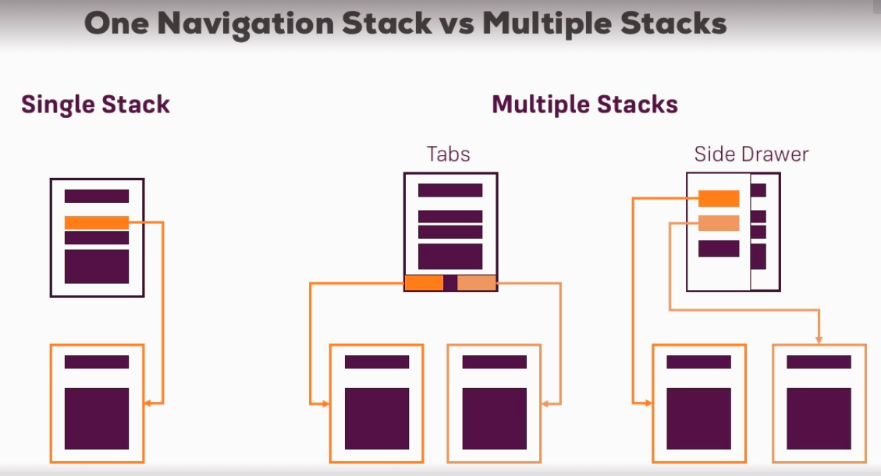
**Lesson 40 Structuring your app**

It is really important to have a clear view how the app will be interacted, what are the different pages, options.



**Lesson 42 One navigation stack vs Multiple stack**

**Multiple stack** is when there are more pages in the same level



**Lesson 43 Implementing Tabs Navigation**

We are creating a tab component which will help us with the navigation

It is exported as **TabsPage**  since it is merging the two imported pages, (favouritesPage, libraryPage).

**Pages/tabs/tab.ts**

import { Component } from '@angular/core';

import { FavoritesPage } from "../favorites/favorites";

import { LibraryPage } from "../library/library";

@Component({

selector: 'page-tabs',

template: `

<ion-tabs>

<ion-tab [root]="favoritesPage" tabTitle="Favorites" tabIcon="star"></ion-tab>

<ion-tab [root]="libraryPage" tabTitle="Library" tabIcon="book"></ion-tab>

</ion-tabs>

`

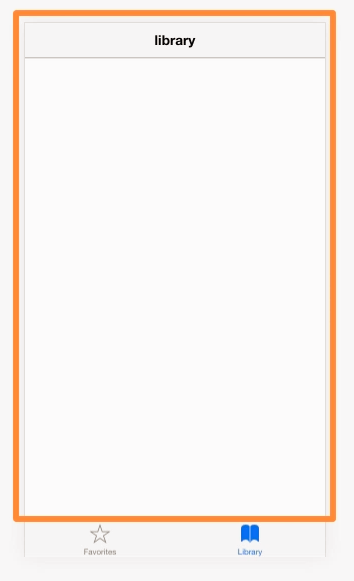
})

export class **TabsPage** {

favoritesPage = FavoritesPage;

libraryPage = LibraryPage;

}

 It has to be set as **rootPage** in the **app.component.ts**

**rootPage = TabsPage**

And to be included in the **app.module.ts**

**Declarations:[ TabsPage]**