**Progress Portfolio: Flutter Journey - Chapter 1**

In this section, I will document my progress in learning Flutter, starting with an introduction to Flutter and the initial setup process.

**Introduction to Flutter**

Flutter is an open-source UI software development toolkit created by Google. It allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase. I am starting my journey with Flutter, exploring its basic concepts and features.



**Getting Started with Flutter**

For the initial setup, I installed Flutter SDK, set up Android Studio, and created my first Flutter project. I learned how to run a Flutter app on both Android and iOS simulators/emulators.

**First Flutter App**

My first Flutter app was a simple "Hello, World!" application. I explored basic widgets,including Text and Center, and tested them on different screen sizes.



**Progress Portfolio: Flutter Journey - Chapter 2**

This section covers Chapter 2, where I learned how to create my first Flutter app: "Hello, World!". I explored the basics of widgets and the Flutter development environment.

**Creating "Hello, World!" in Flutter**

In this chapter, I learned how to set up a simple Flutter app that displays "Hello, World!" on the screen. This app helped me understand the core Flutter structure, including how to use the MaterialApp widget and Scaffold widget for creating basic layouts.



**Code Example**

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(title: Text('Hello World')),

body: Center(child: Text('Hello, World!')),

),

);

}

}

*December 2024*

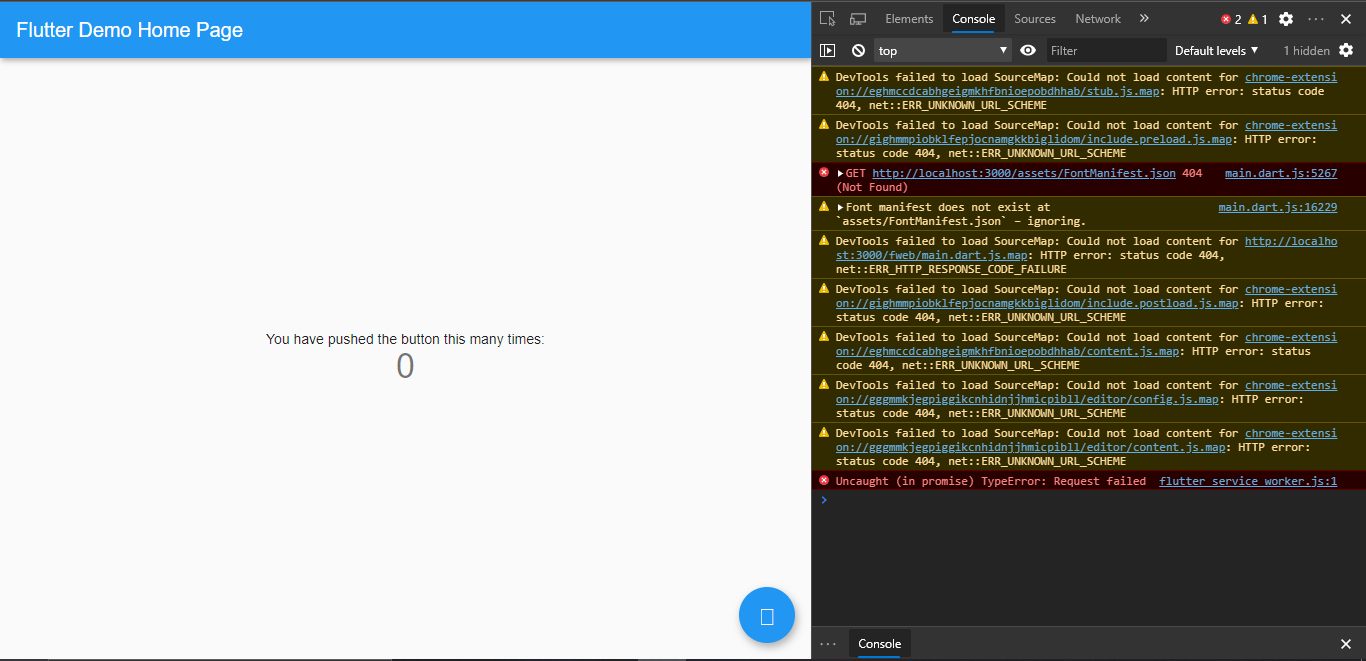
**Progress Portfolio: Flutter Journey - Chapter 3**

This section covers Chapter 3, where I dove into Dart basics, learning key concepts such as variables, data types, control flow, and functions. Dart is the foundation of Flutter, and understanding its syntax is crucial for building apps.

**Learning Dart Basics**

In this chapter, I explored the core concepts of Dart, which included:

* Variables and Data Types
* Control Flow Statements (if-else, loops)
* Functions
* Object-Oriented Programming (OOP) basics



**Code Example: Dart Basics**

void main() {

// Variables

String name = 'Ma. Roylyn';

int age = 25;

// Control Flow

if (age >= 18) {

print('$name is an adult.');

} else {

print('$name is a minor.');

}

// Function Example

greet(name) {

print('Hello, $name!');

}

greet(name);

}

*December 2024*

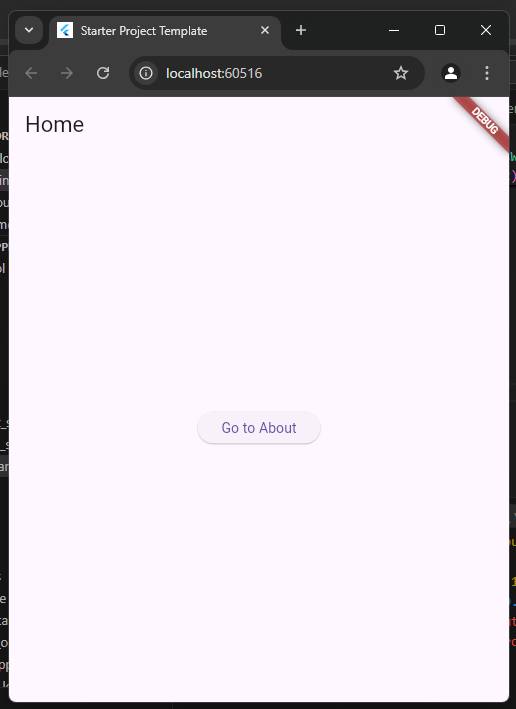
**Progress Portfolio: Flutter Journey - Chapter 4**

In Chapter 4, I learned how to create a starter project template in Flutter. This template serves as a foundation for new Flutter apps, ensuring consistency and saving development time for future projects.

**Creating a Starter Project Template**

Key features of the starter template:

* Defined folder structure: `lib/`, `assets/`, `test/`
* Pre-configured routes for navigation
* A basic home screen layout
* Reusable widgets



**Code Example: Starter Template Structure**

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Starter Project Template',

theme: ThemeData(primarySwatch: Colors.pink),

home: HomeScreen(),

routes: {

'/home': (context) => HomeScreen(),

'/about': (context) => AboutScreen(),

},

);

}

}

class HomeScreen extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Home')),

body: Center(child: Text('Welcome to your starter template!')),

);

}

}

class AboutScreen extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('About')),

body: Center(child: Text('About this project template.')),

);

}

}

*December 2024*

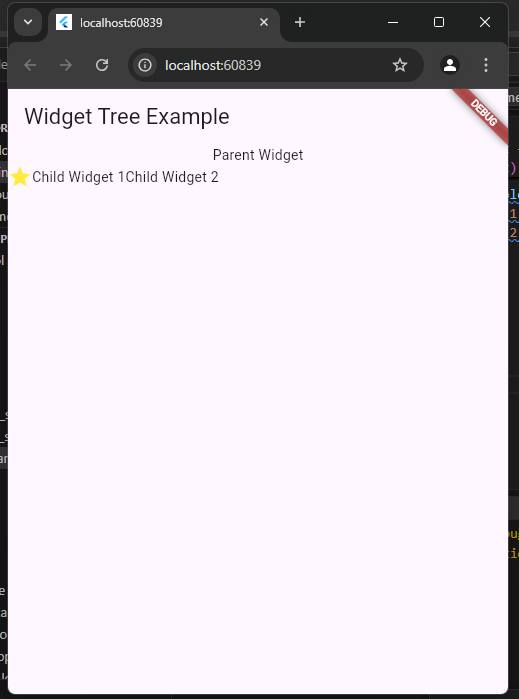
**Progress Portfolio: Flutter Journey - Chapter 5**

In Chapter 5, I explored the Widget Tree in Flutter, which forms the core structure of every Flutter app. Widgets are the building blocks that define the UI and behavior of an app.

**Understanding the Widget Tree**

The Widget Tree represents the hierarchy of widgets in a Flutter application. Key concepts include:

* Parent-Child relationships: Widgets can contain other widgets as children.
* Stateless and Stateful Widgets: How they interact with the tree.
* Tree depth: Complex layouts lead to deeper trees.
* Performance: Keeping the tree optimized for better app responsiveness.



**Code Example: Simple Widget Tree**

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(title: Text('Widget Tree Example')),

body: Column(

children: [

Text('Parent Widget'),

Row(

children: [

Icon(Icons.star, color: Colors.yellow),

Text('Child Widget 1'),

Text('Child Widget 2'),

],

),

],

),

),

);

}

}

*December 2024*

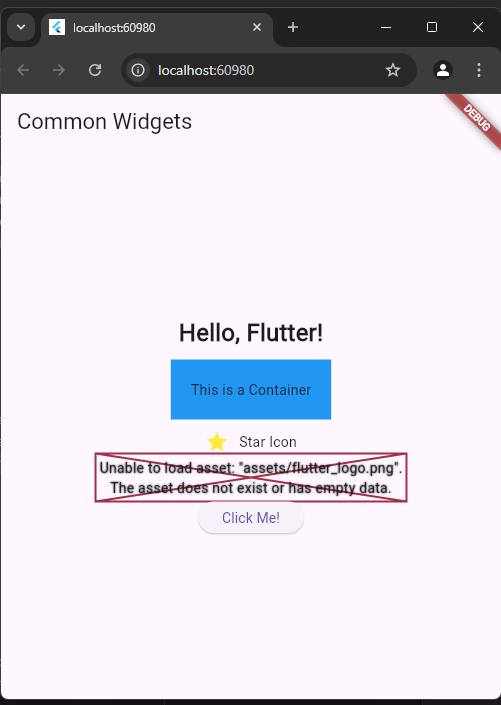
**Progress Portfolio: Flutter Journey - Chapter 6**

Chapter 6 dives into the most commonly used widgets in Flutter, which are essential for creating interactive and visually appealing user interfaces. Widgets such as `Text`, `Container`, `Row`, `Column`, `Image`, `Button`, and more were explored in depth.

**Using Common Widgets**

Here are some examples of frequently used widgets and their functionality:

* **Text:** Displays a string of text with customizable styles.
* **Container:** A versatile widget for layout, styling, and positioning.
* **Row and Column:** Organize widgets horizontally or vertically.
* **Image:** Displays images from assets, URLs, or memory.
* **Button Widgets:** For user interactions, such as `ElevatedButton`, `TextButton`, and `IconButton`.



**Code Example: Common Widgets in Action**

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(title: Text('Common Widgets')),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

'Hello, Flutter!',

style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),

),

Container(

margin: EdgeInsets.all(10),

padding: EdgeInsets.all(20),

color: Colors.blue,

child: Text('This is a Container'),

),

Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Icon(Icons.star, color: Colors.yellow),

SizedBox(width: 10),

Text('Star Icon'),

],

),

Image.asset('assets/flutter\_logo.png', height: 100),

ElevatedButton(

onPressed: () {

print('Button pressed!');

},

child: Text('Click Me!'),

),

],

),

),

),

);

}

}

*December 2024*

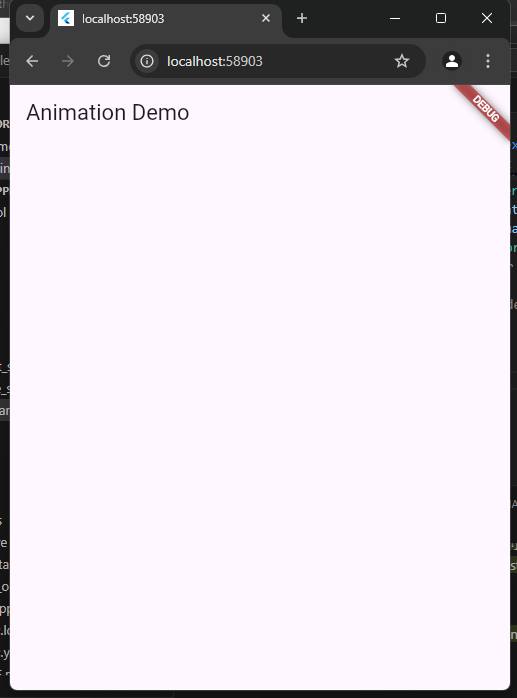
**Progress Portfolio: Flutter Journey - Chapter 7**

Chapter 7 introduces animations in Flutter, exploring how they enhance user interaction and visual appeal. Flutter provides various tools and widgets for creating dynamic animations, such as `AnimatedContainer`, `FadeTransition`, and custom animations with the `AnimationController`.

**Adding Animation to an App**

Animations can make apps more engaging and provide visual feedback to users. Some examples include:

* **AnimatedContainer:** Smoothly transitions changes in properties like color, size, and shape.
* **FadeTransition:** Creates a fading effect for widgets.
* **Custom Animations:** Use `AnimationController` for advanced animations.



**Code Example: Animation in Action**

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: AnimationDemo(),

);

}

}

class AnimationDemo extends StatefulWidget {

@override

\_AnimationDemoState createState() => \_AnimationDemoState();

}

class \_AnimationDemoState extends State with SingleTickerProviderStateMixin {

late AnimationController \_controller;

late Animation \_animation;

@override

void initState() {

super.initState();

\_controller = AnimationController(

duration: Duration(seconds: 2),

vsync: this,

)..repeat(reverse: true);

\_animation = Tween(begin: 0, end: 300).animate(\_controller);

}

@override

void dispose() {

\_controller.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Animation Demo')),

body: Center(

child: AnimatedBuilder(

animation: \_animation,

builder: (context, child) {

return Container(

width: \_animation.value,

height: \_animation.value,

color: Colors.blue,

);

},

),

),

);

}

}

*December 2024*

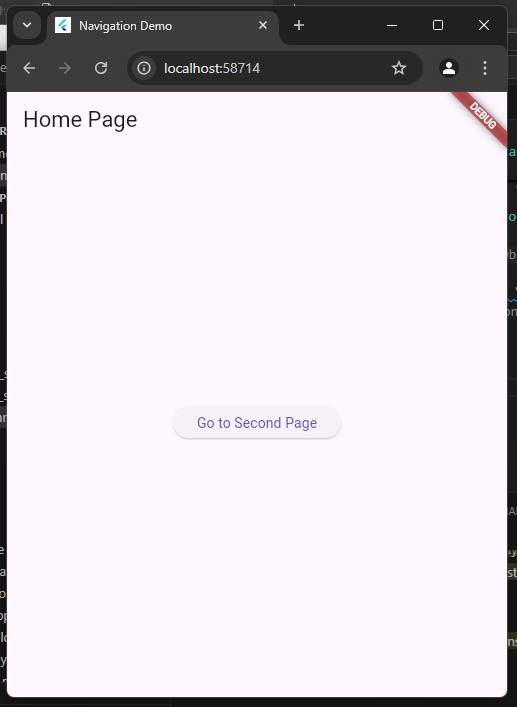
**Progress Portfolio: Flutter Journey - Chapter 8**

Chapter 8 focuses on building an app's navigation system, a critical component for creating multi-screen applications. Flutter provides a straightforward and flexible navigation mechanism using the `Navigator` widget and route management techniques.

**Creating an App's Navigation**

Effective navigation ensures a seamless user experience. Key concepts include:

* **Navigator:** Manages a stack of routes and handles transitions between screens.
* **MaterialPageRoute:** A common route type for material design apps.
* **Named Routes:** Simplify navigation in larger apps by using predefined route names.



**Code Example: Navigation Between Screens**

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Navigation Demo',

initialRoute: '/',

routes: {

'/': (context) => HomePage(),

'/second': (context) => SecondPage(),

},

);

}

}

class HomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Home Page')),

body: Center(

child: ElevatedButton(

onPressed: () {

Navigator.pushNamed(context, '/second');

},

child: Text('Go to Second Page'),

),

),

);

}

}

class SecondPage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Second Page')),

body: Center(

child: ElevatedButton(

onPressed: () {

Navigator.pop(context);

},

child: Text('Go Back'),

),

),

);

}

}