
Flutter & Dart Fundamentals

Introduction to Flutter & Your First App (Project: Digital Business Card)

Welcome to the exciting world of Flutter! In Chapter 1, you built a solid foundation in the Dart programming language. Now, it's time to put that knowledge into action and see how Dart powers Flutter to create beautiful, natively compiled applications for mobile, web, and desktop from a single codebase.

This chapter will introduce you to the core concepts of Flutter. We'll explore what makes Flutter special, understand its architecture, and, most importantly, you'll build your very first Flutter application! We'll start by understanding the default "counter" app that Flutter creates, and then we'll move on to a practical project: a **Digital Business Card**. This project will allow you to apply fundamental Flutter concepts to create a simple yet visually appealing UI.

Learning Objectives:

Upon completing this chapter, you will be able to:

- Understand what Flutter is, its core architecture, and its key advantages for app development.
 - Successfully set up a new Flutter project and understand its basic directory structure.
 - Grasp the fundamental "Everything is a Widget" concept in Flutter.
 - Differentiate between `StatelessWidget` and `StatefulWidget` at a high level (deep dive later).
 - Use essential Flutter UI widgets like `MaterialApp`, `Scaffold`, `AppBar`, `Text`, `Center`, `Icon`, and `Image`.
 - Employ basic layout widgets such as `Container`, `Padding`, `Column`, and `Row` to structure your UI.
 - Understand and utilize Flutter's Hot Reload and Hot Restart features for rapid development.
 - Build and run your first complete Flutter application: a Digital Business Card.
 - Add assets like images to your Flutter project and display them.
-

Concepts and Technologies Covered:

1. What is Flutter?

- A UI toolkit by Google for building beautiful, natively compiled applications for mobile (iOS, Android), web, desktop (Windows, macOS, Linux), and embedded devices from a single codebase.
- **Flutter Architecture Overview:**
 - **Flutter Engine:** Written primarily in C++, provides low-level rendering support using Google's Skia graphics library. It also interfaces with platform-specific SDKs (like Android and iOS).
 - **Flutter Framework:** A rich set of libraries written in Dart. It includes foundational classes, rendering layers, animation, gestures, and a vast collection of UI widgets (Material Design and Cupertino).
 - **Widgets:** The fundamental building blocks of a Flutter UI. "Everything is a Widget."

2. Why Flutter? Key Advantages:

- **Fast Development:** Hot Reload allows you to see changes in your code reflected almost instantly without losing app state.
- **Expressive and Flexible UI:** Flutter's layered architecture allows for full customization, resulting in beautiful and highly-branded UIs. You control every pixel on the screen.
- **Excellent Performance:** Flutter compiles to native ARM or Intel machine code, as well as JavaScript, ensuring high performance on all platforms.
- **Cross-Platform from a Single Codebase:** Write once, run everywhere significantly reduces development time and cost.
- **Growing Community and Ecosystem:** A large and active community, rich documentation, and a vast number of third-party packages on pub.dev.

3. Setting Up Your Flutter Environment (Recap/Check):

- Ensure Flutter SDK is installed and `flutter doctor` reports no critical issues.
- IDE (VS Code or Android Studio) is configured with Flutter and Dart plugins.
- An emulator, simulator, or physical device is ready for running apps.

4. Creating Your First Flutter Project:

- Using the command line: `flutter create my_first_app`
- Using your IDE's UI.

5. Understanding the Flutter Project Structure:

- `lib/main.dart`: The entry point of your Flutter application.
- `pubspec.yaml`: Project metadata, dependencies (packages), and asset declarations.
- `android/` and `ios/`: Platform-specific project files. You'll rarely touch these for basic Flutter development.

- `web/`, `windows/`, `linux/`, `macos/`: Platform-specific files for other targets.
- `test/`: Directory for your application tests.

6. "Everything is a Widget": The Core Philosophy

- In Flutter, UIs are built by composing widgets. Widgets describe what their view should look like given their current configuration and state.
- Examples: `Text` is a widget, `Image` is a widget, `Padding` is a widget, even the entire app is a widget.

7. `StatelessWidget` vs. `StatefulWidget` (Introduction):

- **`StatelessWidget`**: A widget whose state cannot change once it's built. Its appearance and behavior are determined by the configuration information provided by its parent. Used for static content.
- **`StatefulWidget`**: A widget that has mutable state. Its appearance can change dynamically during the lifetime of the widget in response to user interactions or data changes. We'll explore this more in later chapters.

8. Core UI Widgets (Material Design):

- `MaterialApp`: A top-level widget that sets up Material Design theming, navigation, and other essential app functionalities.
- `Scaffold`: Provides a basic Material Design visual layout structure (`AppBar`, `body`, `FloatingActionButton`, `Drawer`, etc.).
- `AppBar`: The top app bar, usually containing a title and actions.
- `Text`: Displays a string of text with optional styling.
- `Center`: A widget that centers its child within itself.
- `Icon`: A graphical icon widget, typically from the Material Icons library.
- `Image`: Displays an image. Can load from assets, network, memory, or files.
 - `Image.asset()`: For images included in your app's assets.
 - `Image.network()`: For images loaded from a URL.


9. Basic Layout Widgets:

- `Container`: A versatile widget for styling (color, shape, borders, padding, margin) and positioning its child.
- `Padding`: Adds space around its child widget.
- `Column`: Arranges its children widgets vertically.
- `Row`: Arranges its children widgets horizontally.
- `SizedBox`: A box with a specified size. Often used to create fixed spaces between widgets.

10. Flutter's Development Cycle: Hot Reload & Hot Restart:

- **Hot Reload (Ctrl+S or ⚡ button)**: Injects updated source code files into the running Dart Virtual Machine (VM). The VM updates classes with the new versions of fields and functions. The Flutter framework then automatically rebuilds the widget

tree, allowing you to quickly see the effects of your changes without losing the current app state. This is a game-changer for UI development.

- **Hot Restart (Shift+Ctrl+S or  button):** Resets the app state and rebuilds the app from scratch. Slower than Hot Reload but useful when changes to state management or native code have occurred.

Project Description: Digital Business Card

- **Real-world Business Problem:** Freelancers, consultants, or anyone networking needs a quick and modern way to share their contact information. A physical business card can get lost or outdated. A simple digital version is always accessible on their phone.
- **Solution:** We will develop a "Digital Business Card" application. This app will display:
 - A profile picture.
 - The person's name.
 - Their title or profession.
 - Contact information like phone number and email address, accompanied by relevant icons.
 - (Optional) A short bio or social media link.

This project is perfect for beginners because it allows us to focus on UI construction and basic widget composition without complex logic or state management.

Step-by-Step Project Implementation: Building Your Digital Business Card

Let's get our hands dirty and build this app!

Step 1: Create a New Flutter Project

1. Open your terminal or command prompt.
2. Navigate to the directory where you want to create your project.
3. Run the command:

```
flutter create digital_business_card
```
4. Once created, navigate into the project directory:

```
cd digital_business_card
```
5. Open this project in your preferred IDE (VS Code or Android Studio).

Step 2: Explore and Clean Up `lib/main.dart`

Open the `lib/main.dart` file. You'll see the default counter application code. Let's briefly understand it before we replace it.

```
// lib/main.dart (Default Counter App - simplified)
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp()); // Entry point of the app
}

// MyApp is a StatelessWidget
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp( // Provides Material Design theming
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue, // Sets the primary color theme
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'), // The
first screen
    );
  }
}

// MyHomePage is a StatefulWidget (because the counter needs to change
state)
class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0; // This is the state variable
```

```

void _incrementCounter() {
  setState(() { // This function tells Flutter to rebuild the UI
    _counter++;
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold( // Basic Material Design layout structure
    appBar: AppBar(
      title: Text(widget.title),
    ),
    body: Center( // Centers its child
      child: Column( // Arranges children vertically
        mainAxisAlignment: MainAxisAlignment.center, // Centers
children in the column
        children: <Widget>[
          const Text(
            'You have pushed the button this many times:',
          ),
          Text(
            '$_counter', // Displays the counter value
            style: Theme.of(context).textTheme.headlineMedium,
          ),
        ],
      ),
    floatingActionButton: FloatingActionButton( // The + button
      onPressed: _incrementCounter, // Calls the increment function
      tooltip: 'Increment',
      child: const Icon(Icons.add),
    ),
  );
}
}

```

Use **main()** function: The entry point of every Dart application. `runApp()` inflates the given widget and attaches it to the screen.

- **MyApp**: Usually a `StatelessWidget` that sets up the `MaterialApp`.

- **MaterialApp**: Wraps your application and provides essential Material Design features.
- **MyHomePage**: A `StatefulWidget` because the counter value changes.
- **Scaffold**: Provides the basic structure (app bar, body).
- **setState()**: Crucial for `StatefulWidget`. It tells Flutter that the state has changed and the UI needs to be rebuilt to reflect those changes.

Exercise 1: Run the Default App

- Connect a device or start an emulator/simulator.
- Run the app from your IDE (usually a "Run" button or by pressing F5 in VS Code).
- Interact with the counter app. Try pressing the "+" button.
- Experiment with **Hot Reload**: Change the `primarySwatch` in `ThemeData` from `Colors.blue` to `Colors.green`, save the file (Ctrl+S), and see the app bar color change instantly without the counter resetting.

Now, **delete all the code** in `lib/main.dart`. We'll start fresh!

Step 3: Setting Up the Basic Structure for Our Business Card

Type or paste the following into your `lib/main.dart`:

```
import 'package:flutter/material.dart';

void main() {
  runApp(const DigitalBusinessCardApp());
}

class DigitalBusinessCardApp extends StatelessWidget {
  const DigitalBusinessCardApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false, // Removes the debug banner
      home: BusinessCard(), // Our main screen
    );
  }
}

class BusinessCard extends StatelessWidget {
```

```

const BusinessCard({super.key});

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.teal, // Let's set a background color
    body: Center( // Center everything on the screen
      child: Text(
        'My Digital Business Card',
        style: TextStyle(
          color: Colors.white,
          fontSize: 24.0,
          fontWeight: FontWeight.bold,
        ),
      ),
    ),
  );
}

```

Explanation:

- We've created `DigitalBusinessCardApp` (our root widget) and `BusinessCard` (our main screen widget). Both are `StatelessWidget` because our business card content won't change dynamically for now.
- `MaterialApp`'s `home` property is set to an instance of `BusinessCard`.
- `Scaffold` provides the screen structure. We've given it a `teal` background color.
- The body of the `Scaffold` currently contains a `Center` widget, which in turn contains a `Text` widget.
- `TextStyle` is used within the `Text` widget to customize its appearance (color, size, weight).

Action: Run the app now. You should see a teal screen with the text "My Digital Business Card" centered and styled in white.

Step 4: Adding the Profile Picture

1. **Create an `assets` folder:** In the root of your `digital_business_card` project (at the same level as `lib` and `pubspec.yaml`), create a new folder named `assets`.
2. **Add an image:** Find a profile picture (e.g., `profile.png` or `profile.jpg`) and place it inside the `assets` folder. For this example, let's assume you have `profile.png`.
3. **Declare assets in `pubspec.yaml`:** Open `pubspec.yaml`. Find the `flutter:` section and uncomment/add the `assets:` part:


```

4. flutter:
5.   uses-material-design: true
6.
7.   # To add assets to your application, add an assets section, like
   this:
8.   assets:
9.     - assets/profile.png # Make sure this path matches your image file
10.    # You can also list the whole folder:
      # - assets/

```

Important: YAML is very sensitive to indentation. Ensure `assets:` is indented correctly under `flutter:`, and the image path (`- assets/profile.png`) is indented under `assets:`. Usually, it's two spaces per indent level. After editing `pubspec.yaml`, your IDE might prompt you to run `flutter pub get`, or you might need to save the file and it will run automatically. If not, run `flutter pub get` in your terminal. This makes the assets available to your app.

11. Modify `BusinessCard` widget to display the image:

```

// ... (imports and main function remain the same)

class DigitalBusinessCardApp extends StatelessWidget {
  // ... (same as before)
}

class BusinessCard extends StatelessWidget {
  const BusinessCard({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.teal,
      body: SafeArea( // Ensures content is not obscured by notches or
status bars
        child: Center(
          child: Column( // We'll arrange elements vertically
            mainAxisAlignment: MainAxisAlignment.center, // Center
Column children vertically
            children: <Widget>[
              CircleAvatar(
                radius: 50.0, // Size of the circle

```

```

        backgroundImage: AssetImage('assets/profile.png'), //
Load image
    ),
    Text(
        'Your Name',
        style: TextStyle(
            fontFamily: 'Pacifico', // We'll add this custom font
later
            fontSize: 40.0,
            color: Colors.white,
            fontWeight: FontWeight.bold,
        ),
    ),
    Text(
        'FLUTTER DEVELOPER',
        style: TextStyle(
            fontFamily: 'SourceSansPro', // And this one
            color: Colors.teal.shade100,
            fontSize: 20.0,
            letterSpacing: 2.5,
            fontWeight: FontWeight.bold,
        ),
    ),
],
),
),
),
);
}
}

```

Explanation of Changes:

- **SafeArea:** Wraps the `body` content to avoid system intrusions (like the notch on an iPhone or the status bar).
- **Column:** We've replaced the single `Text` widget inside `Center` with a `Column`. `Column` arranges its children vertically.
- **mainAxisAlignment: MainAxisAlignment.center:** This property of `Column` centers its children along its main axis (vertical).

- **CircleAvatar**: A widget that displays a circular image.
 - **radius**: Defines the size of the circle.
 - **backgroundImage**: Takes an `ImageProvider`. `AssetImage('assets/profile.png')` loads the image from your assets folder.
- **Text widgets for Name and Title**: Added below the `CircleAvatar`.
- **fontFamily**: We've specified custom fonts. We'll add these in the next step. For now, Flutter will fall back to a default font.

Action: Run the app. You should see your profile picture (or a placeholder if the image isn't found), your name, and your title.

Step 5: Adding Custom Fonts (Optional but good for aesthetics)

1. **Download Fonts:** Go to fonts.google.com and download:
 - Pacifico
 - Source Sans Pro (select a few weights like Regular and Bold)
2. **Create a fonts folder:** Inside your project's root directory, create a `fonts` folder.
3. **Add Font Files:** Extract the downloaded `.ttf` font files and place them into the `fonts` folder. For example:
 - `fonts/Pacifico-Regular.ttf`
 - `fonts/SourceSansPro-Regular.ttf`
 - `fonts/SourceSansPro-Bold.ttf`

4. Declare Fonts in `pubspec.yaml`:

```

5. flutter:
6.   uses-material-design: true
7.   assets:
8.     - assets/profile.png
9.
10.  fonts:
11.    - family: Pacifico
12.      fonts:
13.        - asset: fonts/Pacifico-Regular.ttf
14.    - family: SourceSansPro
15.      fonts:
16.        - asset: fonts/SourceSansPro-Regular.ttf
17.        - asset: fonts/SourceSansPro-Bold.ttf
          weight: 700 # For bold

```

Important: Again, indentation is crucial. Ensure `fonts:` aligns with `assets:` and `uses-material-design:`. After saving `pubspec.yaml`, `flutter pub get` should run. You might need to **stop and restart your application completely (Hot Restart or stop and run again)** for font changes to take full effect.

Action: After restarting, your name should appear in the Pacifico font, and your title in Source Sans Pro.

Step 6: Adding Contact Information with Icons and Cards

Let's add the phone number and email using `Card`, `ListTile`, `Icon`, and `Row` widgets for better structure.

```
// ... (imports and main function, DigitalBusinessCardApp class remain the same)
```

```
class BusinessCard extends StatelessWidget {
  const BusinessCard({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.teal,
      body: SafeArea(
        child: Center( // Added Center here to center the Column in the
afeArea
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              CircleAvatar(
                radius: 50.0,
                backgroundImage: AssetImage('assets/profile.png'),
              ),
              Text(
                'Your Name', // Replace with your actual name
                style: TextStyle(
                  fontFamily: 'Pacifico',
                  fontSize: 40.0,
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
```

```

    ),
  ),
  Text(
    'FLUTTER DEVELOPER', // Replace with your title
    style: TextStyle(
      fontFamily: 'SourceSansPro',
      color: Colors.teal.shade100,
      fontSize: 20.0,
      letterSpacing: 2.5,
      fontWeight: FontWeight.bold,
    ),
  ),
  SizedBox( // Creates a separating line
    height: 20.0,
    width: 150.0,
    child: Divider(
      color: Colors.teal.shade100,
    ),
  ),
  Card( // A card for the phone number
    margin: EdgeInsets.symmetric(vertical: 10.0, horizontal:
25.0),

    child: ListTile(
      leading: Icon(
        Icons.phone,
        color: Colors.teal,
      ),
      title: Text(
        '+1 234 567 8900', // Replace with your phone
        style: TextStyle(
          color: Colors.teal.shade900,
          fontFamily: 'SourceSansPro',
          fontSize: 20.0,
        ),
      ),
    ),
  ),
  Card( // A card for the email

```

```

        margin: EdgeInsets.symmetric(vertical: 10.0, horizontal:
25.0),

        child: ListTile(
          leading: Icon(
            Icons.email,
            color: Colors.teal,
          ),
          title: Text(
            'your.email@example.com', // Replace with your email
            style: TextStyle(
              fontFamily: 'SourceSansPro',
              fontSize: 20.0,
              color: Colors.teal.shade900,
            ),
          ),
        ),
      ),
    ],
  ),
),
),
);
}
}

```

Explanation of New Widgets and Concepts:

- **SizedBox with Divider:** We've added a `SizedBox` to constrain the width of a `Divider` widget, creating a horizontal line effect. `Divider` draws a thin horizontal line.
- **Card:** A Material Design card. It's a panel with slightly rounded corners and a shadow, used to group related information.
 - `margin:` Adds space around the `Card`. `EdgeInsets.symmetric` is useful for providing same padding/margin vertically and horizontally.
- **ListTile:** A convenient widget for creating a row with up to 3 lines of text and optional leading/trailing icons. Commonly used within `Cards` or `ListView`s.
 - `leading:` A widget to display before the title (we're using an `Icon`).
 - `title:` The primary content of the `ListTile` (we're using a `Text` widget).
- **Icon:** Displays a Material Design icon.
 - `Icons.phone` and `Icons.email` are predefined constants for common icons.

- `color`: Sets the icon color.
- **`Colors.teal.shade100` and `Colors.teal.shade900`**: Material colors come in shades. `shade100` is a lighter shade, and `shade900` is a darker shade of teal. This helps create visual hierarchy.

Action: Run the app. You should now see your complete Digital Business Card with the profile picture, name, title, a separating line, and cards for phone and email, each with an icon!

Try This (Mini-Exercise):

- Change the `backgroundColor` of the `Scaffold`.
 - Modify the `radius` of the `CircleAvatar`.
 - Experiment with different `Icons` (e.g., `Icons.web` for a website).
 - Change the `fontFamily`, `fontSize`, or `color` of any `Text` widget.
 - Add another `Card` for a website or social media profile.
-

Key Flutter Features and Packages Used:

- **Flutter SDK Core:**
 - `flutter/material.dart`: Provides Material Design widgets.
 - `StatelessWidget`: Base class for widgets that don't manage internal state.
- **Core Widgets:**
 - `MaterialApp`: Root of Material Design apps.
 - `Scaffold`: Basic visual layout structure.
 - `SafeArea`: Avoids system intrusions.
 - `Center`: Centers its child.
 - `Text`: Displays styled text.
 - `CircleAvatar`: Displays circular images.
 - `Image.asset`: Loads images from project assets.
 - `Icon`: Displays Material Design icons.
 - `Card`: Material Design card for grouping info.
 - `ListTile`: Standard list item layout.
 - `Divider`: A thin horizontal line.

- **Layout Widgets:**
 - `Column`: Arranges children vertically.
 - `SizedBox`: Creates a box with a specific size, often used for spacing or constraining child widgets.
 - (Implicitly `Padding` and `Margin` through `Card`'s `margin` property and `ListTile`'s internal padding).
 - **Project Configuration:**
 - `pubspec.yaml`: For declaring assets (images, fonts) and dependencies.
-

Further Exercises for Practice:

1. **Add a "Website" Card:** Create another `Card` similar to the phone and email cards, but for a website (e.g., your portfolio or LinkedIn). Use an appropriate icon like `Icons.web`.
 2. **Change the Layout:** Try to place the `CircleAvatar` and the `Name/Title Text` widgets side-by-side using a `Row` instead of a `Column` for that section. You might need to adjust sizing and alignment.
 3. **Customize TextStyle More:** Explore other `TextStyle` properties like `letterSpacing`, `wordSpacing`, `decoration` (e.g., `TextDecoration.underline`).
 4. **Use a Container for Background:** Instead of setting `backgroundColor` on the `Scaffold`, try wrapping the main `Column` in a `Container` widget and give the `Container` a background color, padding, or even rounded corners using `decoration: BoxDecoration(...)`.
 5. **Abstract Information:** Imagine you want to create many business cards. How could you make the `BusinessCard` widget more reusable by passing in the name, title, phone, email, and image path as parameters to its constructor? (Hint: Add properties to the `BusinessCard` class and initialize them in the constructor). This is a more advanced step that hints at widget composition and reusability.
-

Expected Outcomes:

By the end of this chapter, you will have:

- A fully functional Digital Business Card application running on your emulator/device.
- A practical understanding of how to create a Flutter project, structure its UI using various widgets, and add assets.
- Hands-on experience with `StatelessWidget`, `Column`, `Row` (implicitly via `ListTile`), `Card`, `CircleAvatar`, `Text`, `Icon`, and `Image.asset`.
- Experienced the power and speed of Flutter's Hot Reload.

- A solid foundation to build upon as we explore more complex Flutter concepts and widgets in the upcoming chapters.