Module 2 Building Static Lists with `ListView` and `ListTile`

In this revised first part of Module 2, we will master the art of displaying lists of information. We'll start by building a custom, visually rich card for our e-commerce app. Then, we'll introduce the highly efficient `ListTile` widget and use it to refactor our productivity app's UI. This will clearly demonstrate how to choose the right widget for your specific layout needs.

1. Learning Objectives

- Master the use of `ListView` to create scrollable lists from hard-coded data.
- Build a complex, custom card layout using `Card`, `Column`, `Row`, and `Image`.
- **Introduce, understand, and use the `ListTile` widget as a powerful layout shortcut.**
- **Implement the core properties of `ListTile`: `leading`, `title`, `subtitle`, and `trailing`.**
- Refactor an existing custom widget to use `ListTile` for a cleaner, more standardized look.
- Appreciate the trade-offs between building a custom layout and using a pre-packaged widget like `ListTile`.

2. Core Concepts

... The foundational concepts "A. Fundamental Theory: The Widget Tree & Composition" and "B. Exploration of Options: The Two Widget Personalities" remain the same as before. They are essential prerequisites. ...

C. The Problem of Multiple Items: `ListView`

When you have more content than can fit on a screen, you need a scrollable list. The premier widget for this is `ListView`. Its key feature is **lazy loading** (with the `ListView.builder` constructor), meaning it only renders the items currently visible, making it incredibly performant for long lists. For this part, we will use the simpler default `ListView` constructor, providing it with a hard-coded list of child widgets to master the basic scrolling layout.

D. The Problem of Repetitive Rows: `ListTile`

The Problem: Look at a typical settings screen, a contact list, or a simple to-do list. You'll see a repeating pattern: an icon on the left, some text in the middle (maybe a main line and a smaller line), and another icon or a switch on the right. You *could* build this every time with a `Row`, `Padding`, `Column`, `Expanded`, etc., but that's a lot of repetitive code.

The Solution: `ListTile` (The "Pre-Fabricated" Row)
`ListTile` is a highly optimized, opinionated "convenience widget" designed to solve this exact problem. It provides a standard, Material Design-compliant row layout with dedicated slots for common elements.

Analogy: If `Row` and `Column` are like raw lumber, nails, and screws, `ListTile` is like a pre-assembled cabinet drawer. You don't have to worry about aligning the handle or making sure the sides are square; you just drop it into place and put your stuff in it.

The primary slots of a `ListTile` are:

- `leading`: A widget to display *before* the title. Typically an `Icon` or a `CircleAvatar`.
- `title`: The main content of the tile. Typically a `Text` widget.
- `subtitle`: A smaller widget displayed below the title. Also typically a `Text` widget.
- `trailing`: A widget to display *after* the title, at the end of the row. Typically an `Icon`, `IconButton`, or `Checkbox`.
- `onTap`: A function to be called when the user taps anywhere on the tile.

3. Practical Application: ShopSphere

Goal: Display a scrollable list of several visually distinct product cards. For this application, a `ListTile` is **not** suitable because an e-commerce product card needs a large image and a custom button layout that doesn't fit the `ListTile` structure. This teaches the student to recognize when a custom layout is necessary.

Step-by-Step Implementation:

1. Create a reusable `widgets/product_card.dart`. This widget will be `Stateless` and will accept the product's name, price, and image URL as constructor arguments. This allows us to make each card unique.
2. In `home_page.dart`, create a `ListView`.
3. Inside the `ListView`'s `children`, we will manually create 6-7 `ProductCard` instances, passing different hard-coded data (name, price, image URL) to each one to simulate a real product list.

4. Practical Application: TaskFlow

Goal: Refactor our `TaskItem` widget to use `ListTile`. This is a perfect use case for `ListTile`, resulting in cleaner code and a more standard, readable UI for our task list.

Step-by-Step Implementation:

1. Open `task_item.dart`. It will remain a `StatefulWidget` to manage the `_isChecked` state.
2. We will completely replace the `Row` inside the `Card` with a `ListTile` widget.
3. We will map our components to the `ListTile` slots:
    - `leading:` will hold the `Checkbox`.
    - `title:` will hold the task's `Text` name.
    - `subtitle:` will hold a new hard-coded `Text` widget, like a due date, to demonstrate this property.
    - `trailing:` will hold an `IconButton` with a delete icon.
4. In `home_page.dart`, we will use a `ListView` and populate it with several `TaskItems`, each with a different hard-coded title, to create our scrollable to-do list.

5. Full Code for New/Updated Files

`shop_sphere/lib/widgets/product_card.dart` (New and Enhanced)

```
import 'package:flutter/material.dart';
class ProductCard extends StatelessWidget {
  final String imageUrl;
  final String name;
  final String price;

  const ProductCard({
    super.key,
    required this.imageUrl,
    required this.name,
    required this.price,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: 4.0,
      margin: const EdgeInsets.symmetric(horizontal: 16.0, vertical: 8.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          ClipRRect(
            borderRadius: const BorderRadius.only(
              topLeft: Radius.circular(12.0),
```

```
              topRight: Radius.circular(12.0),
            ),
            child: Image.network(
              imageUrl, // Use the passed-in URL
              height: 200,
              width: double.infinity,
              fit: BoxFit.cover,
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  name, // Use the passed-in name
                  style: const TextStyle(fontSize: 20, fontWeight:
FontWeight.bold),
                  maxLines: 1,
                  overflow: TextOverflow.ellipsis,
                ),
                const SizedBox(height: 8.0),
                Row(
                  mainAxisAlignment: MainAxisAlignment.spaceBetween,
                  children: [
                    Text(
                      price, // Use the passed-in price
                      style: TextStyle(
                        fontSize: 18,
                        fontWeight: FontWeight.w600,
                        color: Theme.of(context).colorScheme.primary,
                      ),
                    ),
                    ElevatedButton(
                      onPressed: () {},
                      child: const Text('Add to Cart'),
                    ),
                  ],
                ),
```

```
              ],
            ),
          ),
        ],
      ),
    );
  }
}
import 'package:flutter/material.dart';
import 'package:shop_sphere/widgets/product_card.dart';

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('ShopSphere'),
      ),
      body: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          const Padding(
            padding: EdgeInsets.fromLTRB(16.0, 16.0, 16.0, 8.0),
            child: Text(
              'Featured Products',
              style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
            ),
          ),
          Expanded(
            child: ListView(
              children: const [
                // We now pass unique, hard-coded data to each card.
                ProductCard(
                  name: 'Modern Accent Chair',
                  price: '\$299.99',
                  imageUrl: 'https://picsum.photos/seed/chair/600/400',
                ),
```

```
                ProductCard(
                  name: 'Minimalist Desk Lamp',
                  price: '\$79.50',
                  imageUrl: 'https://picsum.photos/seed/lamp/600/400',
                ),
                ProductCard(
                  name: 'Hand-Woven Area Rug',
                  price: '\$450.00',
                  imageUrl: 'https://picsum.photos/seed/rug/600/400',
                ),
                ProductCard(
                  name: 'Smart Coffee Maker',
                  price: '\$125.99',
                  imageUrl: 'https://picsum.photos/seed/coffee/600/400',
                ),
                ProductCard(
                  name: 'Noise-Cancelling Headphones',
                  price: '\$349.00',
                  imageUrl: 'https://picsum.photos/seed/headphones/600/400',
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

Project 2: Task flow

`task_flow/lib/task_item.dart` (Refactored to use `ListTile`)

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

class TaskItem extends StatefulWidget {
  final String title;
  const TaskItem({super.key, required this.title});

  @override
  State<TaskItem> createState() => _TaskItemState();
```

```
}

class _TaskItemState extends State<TaskItem> {
  bool _isChecked = false;

  @override
  Widget build(BuildContext context) {
    // We wrap the ListTile in a Card for better visual separation.
    return Card(
      margin: const EdgeInsets.symmetric(horizontal: 16.0, vertical: 4.0),
      // ListTile is now the direct child of the Card.
      child: ListTile(
        // 1. LEADING: The widget at the start of the tile.
        leading: Checkbox(
          value: _isChecked,
          onChanged: (bool? value) {
            setState(() {
              _isChecked = value ?? false;
            });
          },
        ),
        // 2. TITLE: The primary content.
        title: Text(
          widget.title,
          style: TextStyle(
            decoration:
                _isChecked ? TextDecoration.lineThrough :
TextDecoration.none,
            color: _isChecked ? Colors.grey : null, // Use default color when
not checked
          ),
        ),
        // 3. SUBTITLE: Additional content below the title.
        subtitle: const Text('Due: Tomorrow'),
        // 4. TRAILING: The widget at the end of the tile.
        trailing: IconButton(
          icon: const Icon(Icons.delete_outline, color: Colors.redAccent),
          onPressed: () {
            // Logic to delete the task will go here later.
```

```
            print('Delete button pressed for ${widget.title}');
          },
        ),
        // 5. ONTAP: Makes the entire tile tappable.
        onTap: () {
          // A great UX improvement: tap anywhere on the row to toggle.
          setState(() {
            _isChecked = !_isChecked;
          });
        },
      ),
    );
  }
}
```

`task_flow/lib/home_page.dart` (Updated to use `ListView`)

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('My Tasks'),
      ),
      // We use ListView to ensure our list is scrollable as it grows.
      body: ListView(
        children: const [
          SizedBox(height: 8), // A bit of space at the top.
          TaskItem(title: 'Finish Module 2 of Flutter Course'),
          TaskItem(title: 'Submit the weekly report'),
          TaskItem(title: 'Call the client back'),
          TaskItem(title: 'Grocery shopping for the week'),
          TaskItem(title: 'Plan the weekend trip'),
          TaskItem(title: 'Read one chapter of "Atomic Habits"'),
          TaskItem(title: 'Fix bug #734 on the main project'),
```

```
          ],
        ),
      );
    }
}
```

6. Assignment/Challenge

1. **For `ShopSphere`:** Add a `const Divider()` widget between each `ProductCard` in the `ListView` in `home_page.dart`. This is a simple widget that draws a horizontal line, a common way to visually separate list items.
2. **For `TaskFlow`:** Make the `subtitle` in `TaskItem` dynamic. Modify the `TaskItem` constructor to accept a `required String dueDate`. Then, in `home_page.dart`, pass a different hard-coded due date string to each `TaskItem` instance.
3. **Bonus - `ListTile` Density:** In the `TaskItem`'s `ListTile`, add the property `visualDensity: VisualDensity.compact`. Hot reload and observe how it tightens up the vertical padding of the tile. This is useful for creating more information-dense lists.