03. Layout widgets

Column, Row, Container, and Stack

- Widgets for Layout
- Column
  - column1.dart
  - column2.dart
- Row
  - row1.dart
  - row2_mainAxisAlignment.dart
- Contatiner
  - container1.dart
  - container2.dart
  - container3.dart
- Stack
  - stack1.dart
  - stack2.dart

Widgets for Layout

- The widgets we have discussed are single widgets.
- However, we need widgets that can lay out, parent, and align with other widgets.
- Flutter supports `Column`, `Row`, `Container`, and `Stack` in this case.

Column

- Column hosts multiple widgets aligned in a column with the `children` parameter.
- As a result, the widgets in a column widget are aligned horizontally.

column1.dart

```
return Column(
  children: <Widget>[
      Text("Hello"),
      Text("Flutter"),
  ],
```

- Column has a children property to host multiple widgets in a list.
- Two Text widgets are positioned in a column.

column2.dart

```
return Column(
   mainAxisAlignment: MainAxisAlignment.center,
   children: <Widget>[
       Text("Hello"), Text("Flutter"),
   ],
```

- We use `MainAxisAlignment.center` to center the widgets for alignment.
- There are other options, such as `start` or `end`, among many options.

Row

- Row widget is the same as the Column widget except that widgets are aligned horizontally.

row1.dart

```
return Row(
  children: <Widget>[
      Text("Hello"),
      Text("Flutter"),
  ],
);
```

row2_mainAxisAlignment.dart

- We can use the same options as the Column widget.

```
return Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
      Text("Hello"),
      Text("Flutter"),
  ],
```

Contatiner

- The container contains another widget.
  - It can position the contained widget.
  - It can decorate itself with the decoration property.

container1.dart

```
body:  Container(
  height: 80, width: 260, color: Colors.blueGrey,
  child: const Text(...),
),
```

- The Container widget contains only one widget as implied by the `child` parameter (not children).
- We use `height` and `width` parameters for configuration.

Decoration in Flutter

```
return Container(
  ...
  decoration: BoxDecoration(
    color: Colors.blue,
    border: Border.all(),
  ),
);
```

- For decoration, we use the decoration property and the `BoxDecoration` widget.

```
Text(style: TextStyle(...)
ElevatedButton(style: ...)
Card(color: Colors.yellow[100],...
```

- There are other decoration styles.
- Some support the `style` parameter.
- Some support the decoration property.

container2.dart

```
class BlueBox extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(...);
  }
}
```

- When the container code is large, we need to make a Stateless widget to host it within the `build()`.

```
return Scaffold(
  appBar: AppBar(title: const Text('Container Demo'),),
  body: BlueBox(),
);
```

- We can use the BlueBox StatelessWidget in the scaffold.
- In this way, we can make GUI components that other components can use.

container3.dart

```
class RedBox extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(50.0),
      margin: EdgeInsets.all(30.0),
      decoration: BoxDecoration(
        color: Colors.red,
        border: Border.all(),
      ),
    );
  }
}
```

```
return Container(
  padding: EdgeInsets.all(50.0),
  margin: EdgeInsets.all(30.0),
```

- `EdgeInsets` configures margin and padding in a container.
  - It can be used in other widgets.
- For the decoration for the Container, the `BoxDecoration` widget is used.

```
body: Center( // centering
  child: Column( // alignment
    children: <Widget>[RedBox(), RedBox()],
  ),
),
```

- In the main Widget, we use a list `[]` to store multiple RedBox widgets.
- This is a useful technique to use widgets as components in multiple places.

Stack

- The Stack widget is used to stack widgets on top of each other.
  - **Container**: A "wrapper" for decorating a `single` widget
  - **Stack**: A "layer system" for overlapping `multiple` widgets

stack1.dart

```
Stack(
  children: [Container(...), Text("Hello"),],
```

- When widgets are in a stack, they are stacked on top of other widgets.
- The Text will be displayed and overlapped with the Container.

stack2.dart

```
Stack(
  children: [
    Positioned(top: 0, left: 200,
      child: Container(child: const Text(),),),
    Positioned(top: 40, left: 65,
      child: Text("Hello"),),
  ],
),
```

- When we need to specify a position in a Stack, we use the `Positioned` Widget.