Flutter

A (Very) Brief Overview

Flutter: a new dev system

- Flutter is a new development platform for mobile and Web application development

  https://flutter.dev/

- Created in 2017 by Google; it is undergoing active development.
- Growing popularity among the non-native development systems – about to surpass React Native (developed by Facebook).
- Flutter apps are developed in Dart, an OO programming language.

Flutter: a new dev system

- An app created with Flutter can be deployed on an iOS or Android mobile device; it can be deployed as a Web application, as well.
- Flutter development can be done using Xcode (on Mac OS), Android Studio (on Mac OS, Windows, Chrome OS), Visual Studio Code on Windows, or even without an IDE.
- An app developed on a non-Mac OS system and targeted for iOS must be moved to Xcode on Mac OS for testing on an iOS Simulator and for final packaging.

Flutter: a new dev system

- Flutter installation docs:

  `https://flutter.dev/docs/get-started/install`

- Can be installed on MS Windows, Mac OS X, Linux, and ChromeOS.
- Installation is relatively easy, but some issues may have to be resolved.
- `flutter doctor` is very helpful in resolving issues.
- AndroidStudio needs a Flutter plugin to allow Flutter development.

Flutter: widgets

- Unlike in native Android development, layouts in Flutter are mixed-in with the app’s code (logic).
- It is a bit more like JavaFX/Swing.
- Flutter UI screen includes a layout formed with several widgets as its children.
- Special Flutter classes provide the basis for creating stateless (StatelessWidget) and stateful (StatefulWidget) widgets.

Flutter: widgets

- A **StatelessWidget** does not have a mutable state; it stays intact after it has been created.
- A **StatefulWidget** does have a state, which is represented by a (subclass of) State class; what constitutes the state is up toe the app’s developer.
- A StatefulWidget’s rendering (appearance) can change due to its state changes.

Flutter: widgets

- Flutter widgets include (among ***many***):
  - Icon
  - Image
  - Text
  - TextField
  - Buttons (various types)
  - CheckBox, RadioBox, Slider, etc.
  - Date/Times pickers
  - Many others…

Flutter: layouts

- Flutter layouts include (among ***many***):
  - Row (similar to Android's horizontal LinearLayout)
  - Column
  - Flow – somewhat like JavaFX's FlowPane
  - GridView
  - ListView
  - Stack – overlapping views (similar to FrameLayout)
  - Table – for tabular data (similar to TableLayout)

Flutter: programming in Dart

- Dart is an OO programming language

  https://dart.dev/overview

- Dart classes usually form a hierarchy (like in Java).
- Only single inheritance is allowed.
- Interfaces and abstract classes are available.
- Packages are there, as well.

Flutter: programming in Dart

- Dart classes have **constructors**, used to create objects.
- Classes have **instance variables** and **methods**, which may be **static**, as in Java.
- Methods (and constructors) may have
  - **positional** (required) parameters (like in Java):

    ```
    method( int a, String name, String id )
    ```

  - and a call looks like a usual call in Java*:*

    ```
    method( 11, ’Jim’,  ‘123456789’ );
    ```

Flutter: programming in Dart

- Positional parameters may be followed by:
  - **named parameters**, listed within `{…}`, which may be *required* (keyword) or *optional* (question mark)

```
method( int a, {required int id, String? name})
```

  - an example call:

```
method( 11, name: 'Jim', id: 12345 );
```

or

```
method( 11, id: 12345 );
```

Flutter: programming in Dart

- Positional parameters may be followed by:
  - **optional positional parameters** (but not named ***and*** optional)

```
method( int a, [String name, int id])
```

  - an example call:

```
method( 11, ’Jim’, 12345 );
```

or

```
method( 11, ’Jim’ );
```

Flutter: a StatelessWidget

```
class PushCounter extends StatelessWidget {
 @override
 Widget build( BuildContext context ) {
  return MaterialApp(
   title: 'Push Counter',
   theme: ThemeData(
    primarySwatch: Colors.blue,
   ),
   home: Scaffold(
     appBar: AppBar(
      title: const Text('Push Counter'),
     ),
     body: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
       PushCounterWidget(),
      ],
     )
   ),
  );
 }
}
```

A method declaration

A method call

Passing named parameters: title

Passing named parameters: theme

Scaffold is a basic app’s structure

It has an appBar and a body

A constructor call

Column’s children -- an array

Flutter: a StatefulWidget

```
class PushCounterWidget extends StatefulWidget {
  const PushCounterWidget({Key? key}) : super(key: key);

  @override
  _PushCounterWidgetState createState() => _PushCounterWidgetState();
}
```

Constructor

A Key is an identifier for widgets

The arrow symbol

A function that executes the expression to its right and returns its value

Underscore in front makes it `private`

Flutter: a State

```
class _PushCounterWidgetState extends State<PushCounterWidget> {
  int _counter = 0;
  @override
  Widget build(BuildContext context) {

    return Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        ElevatedButton(
            onPressed: () {
              print('Clicked!');
              setState(() {
                _counter++;
              });
            }, child: Text('Push Me!')),
        Container(width: 10),
        Text('Pushes: ' + _counter.toString())
      ],
    );
  }
}
```

A private variable

An alignment spec inside the Row

setState() notifies the framework of a state change; leads to a redraw (calling build)

Container provides A gap between children

Flutter: input from the user

```
class TripCost extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Trip Cost',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
          appBar: AppBar(
            title: const Text('Trip Cost'),
          ),
          body: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              TripCostWidget(),
            ],   )    ),    );   }. }
```

Flutter: input from the user

```
class TripCostWidget extends StatefulWidget {
  const TripCostWidget({Key? key}) : super(key: key);
  @override
  _TripCostWidgetState createState() => _TripCostWidgetState();
}
```

Flutter: input from the user

```
class _TripCostWidgetState extends State<TripCostWidget> {

  final _distanceController = TextEditingController();
  final _gasCostController = TextEditingController();
  final _mpgController = TextEditingController();
  String _cost = "0.0";

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        SizedBox(
          width: 250,
          child: TextField(
              textAlign: TextAlign.right,
              decoration: const InputDecoration(
                  hintText: 'Trip distance (miles)'),
              controller: _distanceController,
              keyboardType: TextInputType.number
          ),
        ),
        Container(height: 20),
…
```

Controllers for TextFields

A TextField with a controller, a hint, accepting only numbers

A spacer (a Container)

Flutter: input from the user

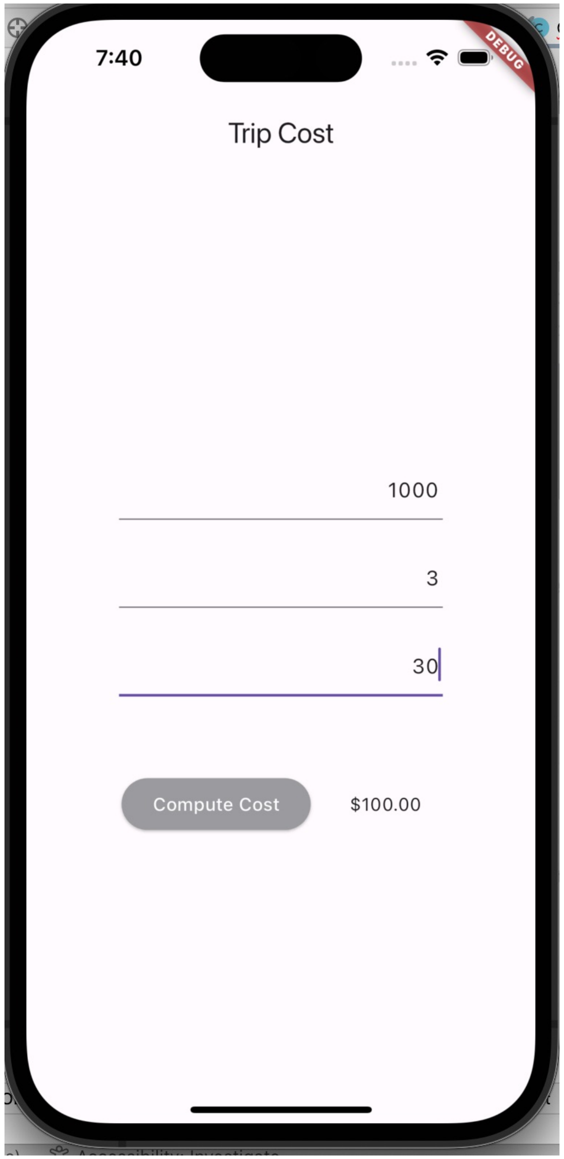
```
…
Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          ElevatedButton(
              onPressed: () {
                setState(() {
                  _cost = _computeCost( _distanceController.text,
                                                            _gasCostController.text,
                                                            _mpgController.text );
                });
              }, child: Text( 'Compute Cost' )
          ),
          Container(width: 30),
          Container(
              width: 70,
              child: Text( '\$' + _cost.toString() )
          )
        ],
      )
    ],
  );
}
```

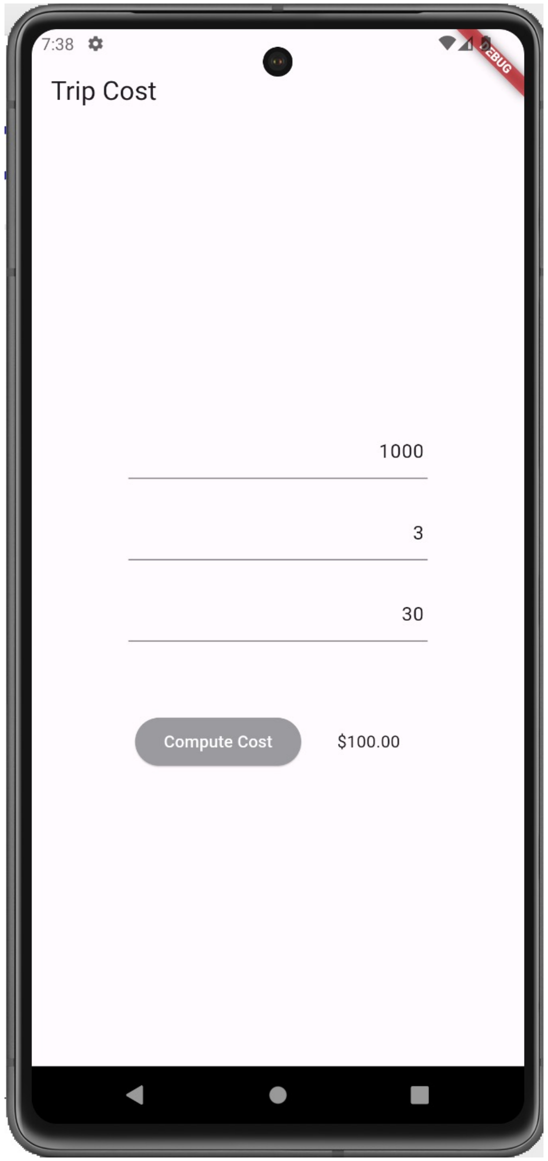
Retrieving a TextField's Value (text)

Trip's cost, redisplayed by setState() after its computation

iOS and Android Emulators

iOS





Android