互联网应用开发技术

Web Application Development

第17课
WEB移动端-FLUTTER DESIGN

Episode Seventeen

Flutter Design

陈昊鹏
chen-hp@sjtu.edu.cn

User Interface

- Widgets
  - Flutter widgets are built using a modern framework that takes inspiration from React.
  - The central idea is that you build your UI out of widgets.
  - Widgets describe what their view should look like given their current configuration and state.
  - When a widget's state changes, the widget rebuilds its description, which the framework diffs against the previous description in order to determine the minimal changes needed in the underlying render tree to transition from one state to the next.

User Interface - Widgets

- Hello World
  - The minimal Flutter app simply calls the runApp() function with a widget:

```
void main() {
  runApp(
    Center(
      child: Text(
        'Hello, world!',
        textDirection: TextDirection.ltr,
      ),
    ),
  );
}
```

User Interface - Widgets

- Basic widgets
  - Flutter comes with a suite of powerful basic widgets, of which the following are commonly used:
  - **Text :** The Text widget lets you create a run of styled text within your application.
  - **Row, Column :** These flex widgets let you create flexible layouts in both the horizontal (Row) and vertical (Column) directions.
  - **Stack :** Instead of being linearly oriented (either horizontally or vertically), a Stack widget lets you place widgets on top of each other in paint order.
  - **Container :** The Container widget lets you create a rectangular visual element.

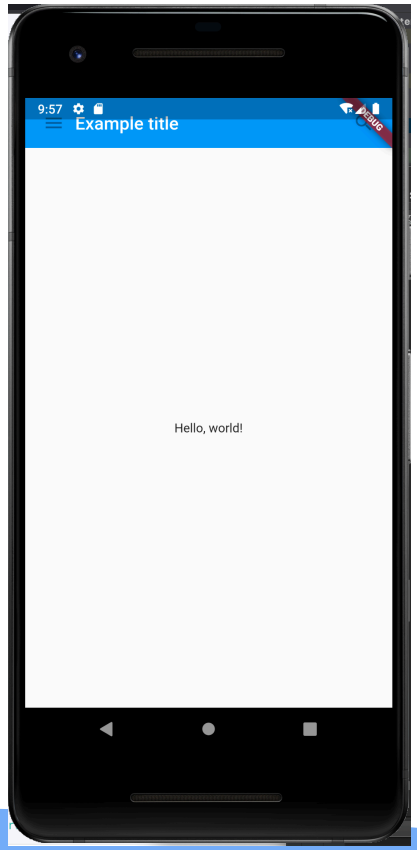
User Interface - Widgets


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

class MyAppBar extends StatelessWidget {
  MyAppBar({this.title});

  // Fields in a Widget subclass are always marked "final".

  final Widget title;

  @override
  Widget build(BuildContext context) {
    return Container(
      height: 56.0, // in logical pixels
      padding: const EdgeInsets.symmetric(horizontal: 8.0),
      decoration: BoxDecoration(color: Colors.blue[500]),
      // Row is a horizontal, linear layout.
      child: Row(
        // <Widget> is the type of items in the list.
        children: <Widget>[
          IconButton(
            icon: Icon(Icons.menu),
            tooltip: 'Navigation menu',
            onPressed: null, // null disables the button
          ),
          // Expanded expands its child to fill the available space.
          Expanded(
            child: title,
          ),
          IconButton(
            icon: Icon(Icons.search),
            tooltip: 'Search',
            onPressed: null,
          ),
        ],
      ),
    );
  }
}
```

User Interface - Widgets


```
class MyScaffold extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Material is a conceptual piece of paper on which the UI appears.
    return Material(
      // Column is a vertical, linear layout.
      child: Column(
        children: <Widget>[
          MyAppBar(
            title: Text(
              'Example title',
              style: Theme.of(context).primaryTextTheme.title,
            ),
          ),
          Expanded(
            child: Center(
              child: Text('Hello, world!'),
            ),
          ),
        ],
      ),
    );
  }
}
```

```
void main() {
  runApp(MaterialApp(
    title: 'My app', // used by the OS task switcher
    home: MyScaffold(),
  ));
}
```

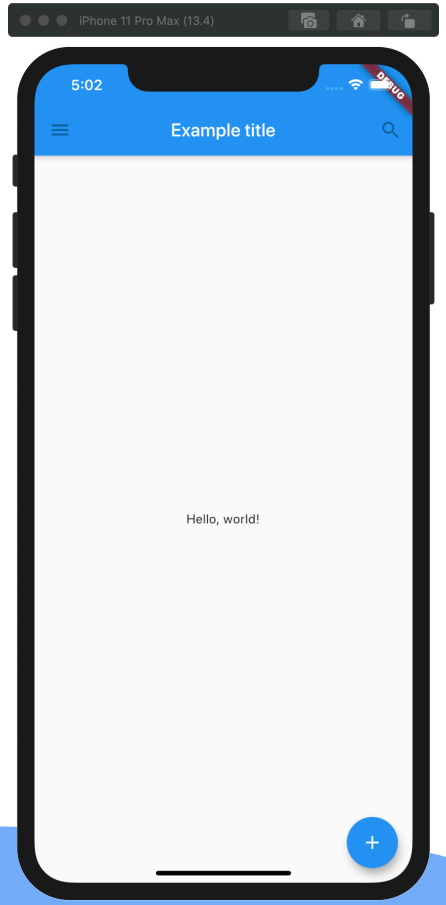


User Interface - Widgets

- Using Material Components
  - Flutter provides a number of widgets that help you build apps that follow Material Design.
  - A Material app starts with the MaterialApp widget, which builds a number of useful widgets at the root of your app, including a Navigator, which manages a stack of widgets identified by strings, also known as "routes".
  - The Navigator lets you transition smoothly between screens of your application.
  - Using the MaterialApp widget is entirely optional but a good practice.

User Interface - Widgets


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

class TutorialHome extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Scaffold is a layout for the major Material Components.
    return Scaffold(
      appBar: AppBar(
        leading: IconButton(
          icon: Icon(Icons.menu),
          tooltip: 'Navigation menu',
          onPressed: null,
        ),
        title: Text('Example title'),
        actions: <Widget>[
          IconButton(
            icon: Icon(Icons.search),
            tooltip: 'Search',
            onPressed: null,
          ),
        ],
      ),
      // body is the majority of the screen.
      body: Center(
        child: Text('Hello, world!'),
      ),
      floatingActionButton: FloatingActionButton(
        tooltip: 'Add', // used by assistive technologies
        child: Icon(Icons.add),
        onPressed: null,
      ),
    );
  }
}
```



User Interface - Widgets

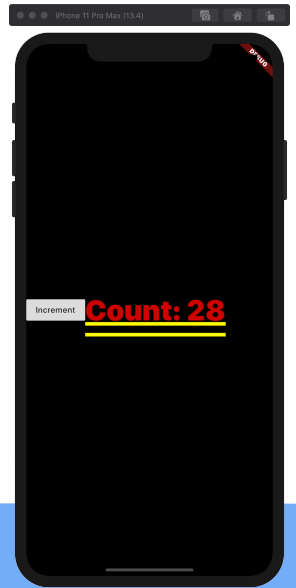
- Handling gestures
  - Most applications include some form of user interaction with the system.

```
class MyButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        print('MyButton was tapped!');
      },
      child: Container(
        height: 36.0,
        padding: const EdgeInsets.all(8.0),
        margin: const EdgeInsets.symmetric(horizontal: 8.0),
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(5.0),
          color: Colors.lightGreen[500],
        ),
        child: Center(
          child: Text('Engage'),
        ),
      ),
    );
  }
}
```

User Interface - Widgets

- Changing widgets in response to input
  - StatefulWidgets are special widgets that know how to generate State objects, which are then used to hold state.

```
class Counter extends StatefulWidget {
  @override
  _CounterState createState() => _CounterState();
}

class _CounterState extends State<Counter> {
  int _counter = 0;
  void _increment() {
    setState(() {
      _counter++;
    });
  }
```




```
  @override
  Widget build(BuildContext context) {
    return Row(
      children: <Widget>[
        RaisedButton(
          onPressed: _increment,
          child: Text('Increment'),
        ),
        Text('Count: $_counter'),
      ],
    );
  }
}
```

User Interface - Widgets

- Changing widgets in response to input
  - In Flutter, change notifications flow “up” the widget hierarchy by way of callbacks, while current state flows “down” to the stateless widgets that do presentation.

```
class CounterDisplay extends StatelessWidget {
 CounterDisplay({this.count});
 final int count;
 @override
 Widget build(BuildContext context) {
  return Text('Count: $count');
 }
}

class CounterIncrementor extends StatelessWidget {
 CounterIncrementor({this.onPressed});
 final VoidCallback onPressed;
 @override
 Widget build(BuildContext context) {
  return RaisedButton(
   onPressed: onPressed,
   child: Text('Increment'),
  );
 }
}
```

```
class Counter extends StatefulWidget {
 @override
 _CounterState createState() => _CounterState();
}

class _CounterState extends State<Counter> {
 int _counter = 0;
 void _increment() {
  setState(() {
   ++_counter;
  });
 }
 @override
 Widget build(BuildContext context) {
  return Row(children: <Widget>[
   CounterIncrementor(onPressed: _increment),
   CounterDisplay(count: _counter),
  ]);
 }
}
```

Shopping List


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

class Product {
  const Product({this.name});
  final String name;
}

typedef void CartChangedCallback(Product product, bool inCart);

class ShoppingListItem extends StatelessWidget {
  ShoppingListItem({this.product, this.inCart, this.onCartChanged})
      : super(key: ObjectKey(product));

  final Product product;
  final bool inCart;
  final CartChangedCallback onCartChanged;

  Color _getColor(BuildContext context) {
    // The theme depends on the BuildContext because different parts of the tree
    // can have different themes.  The BuildContext indicates where the build is
    // taking place and therefore which theme to use.

    return inCart ? Colors.black54 : Theme.of(context).primaryColor;
  }
```

Shopping List


```
  TextStyle _getTextStyle(BuildContext context) {
    if (!inCart) return null;

    return TextStyle(
      color: Colors.black54,
      decoration: TextDecoration.lineThrough,
    );
  }

  @override
  Widget build(BuildContext context) {
    return ListTile(
      onTap: () {
        onCartChanged(product, inCart);
      },
      leading: CircleAvatar(
        backgroundColor: _getColor(context),
        child: Text(product.name[0]),
      ),
      title: Text(product.name, style: _getTextStyle(context)),
    );
  }
}
```

Shopping List

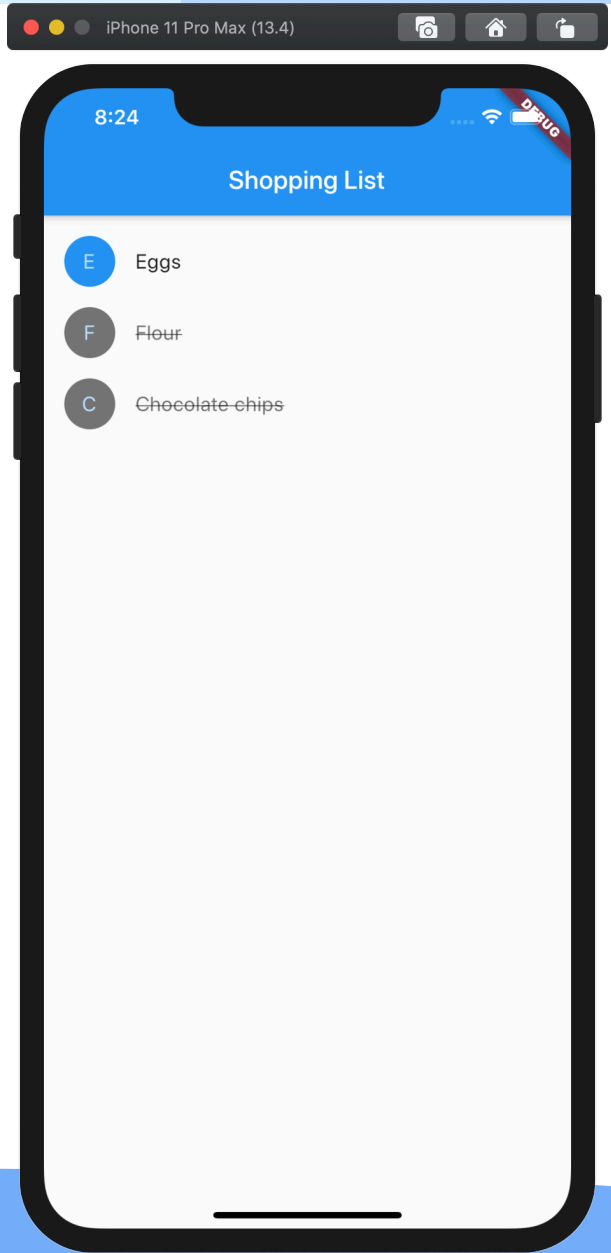

```
class ShoppingList extends StatefulWidget {
  ShoppingList({Key key, this.products}) : super(key: key);

  final List<Product> products;

  // The framework calls createState the first time a widget appears at a given
  // location in the tree. If the parent rebuilds and uses the same type of
  // widget (with the same key), the framework re-uses the State object
  // instead of creating a new State object.

  @override
  _ShoppingListState createState() => _ShoppingListState();
}

class _ShoppingListState extends State<ShoppingList> {
  Set<Product> _shoppingCart = Set<Product>();

  void _handleCartChanged(Product product, bool inCart) {
    setState(() {
      // When a user changes what's in the cart, you need to change
      // _shoppingCart inside a setState call to trigger a rebuild.
      // The framework then calls build, below,
      // which updates the visual appearance of the app.
```

Shopping List


```
    if (!inCart)
      _shoppingCart.add(product);
    else
      _shoppingCart.remove(product);
   });
  }

  @override
  Widget build(BuildContext context) {
   return Scaffold(
    appBar: AppBar(
     title: Text('Shopping List'),
    ),
    body: ListView(
     padding: EdgeInsets.symmetric(vertical: 8.0),
     children: widget.products.map((Product product) {
      return ShoppingListItem(
       product: product,
       inCart: _shoppingCart.contains(product),
       onCartChanged: _handleCartChanged,
      );
     }).toList(),
    ),
   );
  }
}
```

Shopping List


```
void main() {
  runApp(MaterialApp(
    title: 'Shopping App',
    home: ShoppingList(
      products: <Product>[
        Product(name: 'Eggs'),
        Product(name: 'Flour'),
        Product(name: 'Chocolate chips'),
      ],
    ),
  ));
}
```



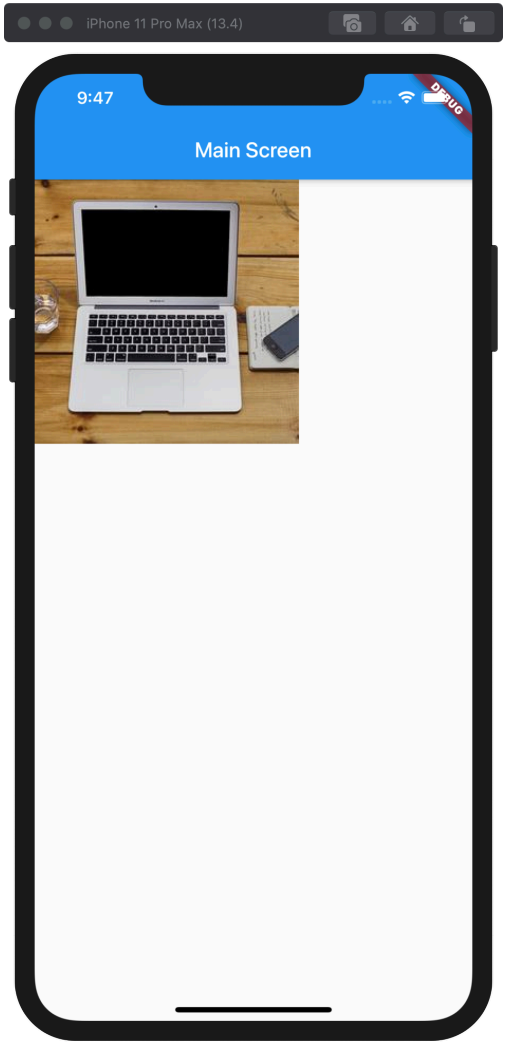
Navigation

- Animate a widget across screens

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

class HeroApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Transition Demo',
      home: MainScreen(),
    );
  }
}

class MainScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Main Screen'),
      ),
      body: GestureDetector(
        child: Hero(
          tag: 'imageHero',
          child: Image.network(
            'https://picsum.photos/250?image=9',
          ),
        ),
        onTap: () {
          Navigator.push(context, MaterialPageRoute(builder: (_) {
            return DetailScreen();
          }));
        },
      ),
    );
  }
}
class DetailScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: GestureDetector(
        child: Center(
          child: Hero(
            tag: 'imageHero',
            child: Image.network(
              'https://picsum.photos/250?image=9', ),
          ),
        ),
        onTap: () {
          Navigator.pop(context); },
      ),
    );
  }
}
```

Navigation

- Animate a widget across screens



Navigation

- Navigate to a new screen and back

```
class FirstRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('First Route'),
      ),
      body: Center(
        child: RaisedButton(
          child: Text('Open route'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => SecondRoute()),
            );
          },
        ),
      ),
    );
  }
}
```

```
class SecondRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Second Route"),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Go back!'),
        ),
      ),
    );
  }
}
```

Navigation

- Navigate to a new screen and back

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

void main() {
  runApp(MaterialApp(
    title: 'Navigation Basics',
    home: FirstRoute(),
  ));
}
```

Navigation

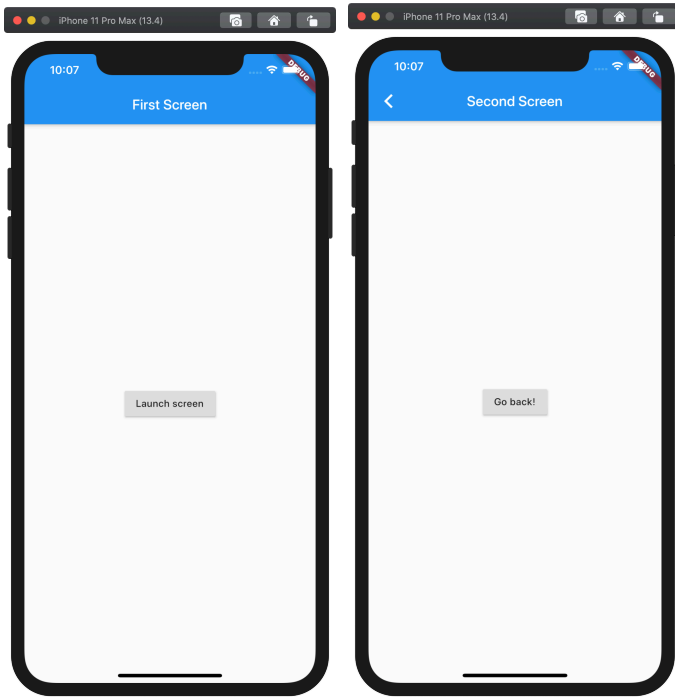
- Navigate with named routes

```
class FirstScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('First Screen'),
      ),
      body: Center(
        child: RaisedButton(
          child: Text('Launch screen'),
          onPressed: () {
            // Navigate to the second screen using a named route.
            Navigator.pushNamed(context, '/second');
          },
        ),
      ),
    );
  }
}
```

```
class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Second Screen"),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: () {
            // Navigate back to the first screen by popping the current
            // route off the stack.
            Navigator.pop(context);
          },
          child: Text('Go back!'),
        ),
      ),
    );
  }
}
```

- Navigate with named routes

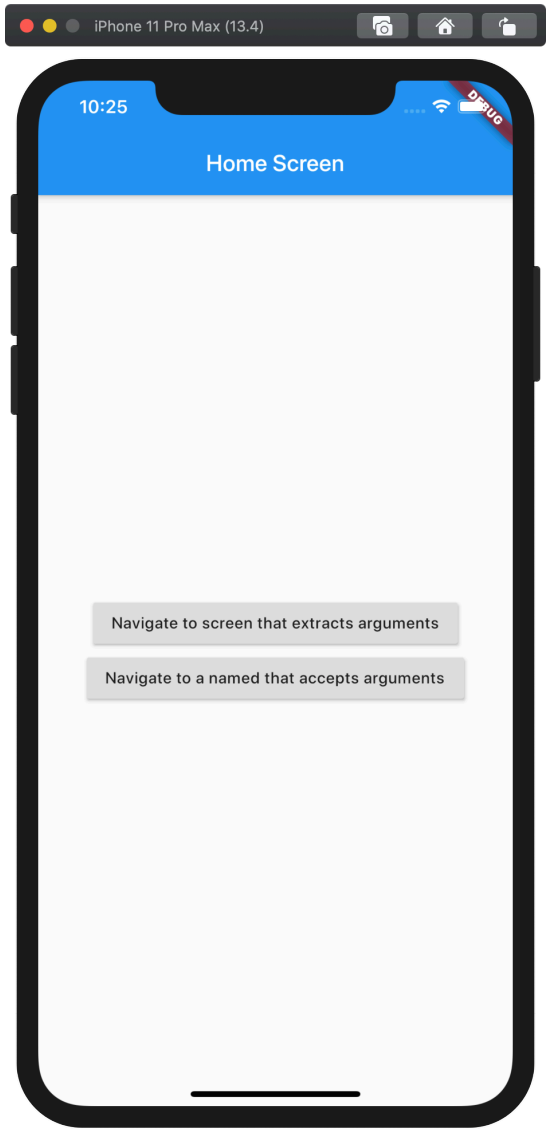
```
import 'package:flutter/material.dart';

void main() {
  runApp(MaterialApp(
    title: 'Named Routes Demo',
    // Start the app with the "/" named route. In this case, the app starts
    // on the FirstScreen widget.
    initialRoute: '/',
    routes: {
      // When navigating to the "/" route, build the FirstScreen widget.
      '/': (context) => FirstScreen(),
      // When navigating to the "/second" route, build the SecondScreen widget.
      '/second': (context) => SecondScreen(),
    },
  ));
}
```



Navigation

- Pass arguments to a named route

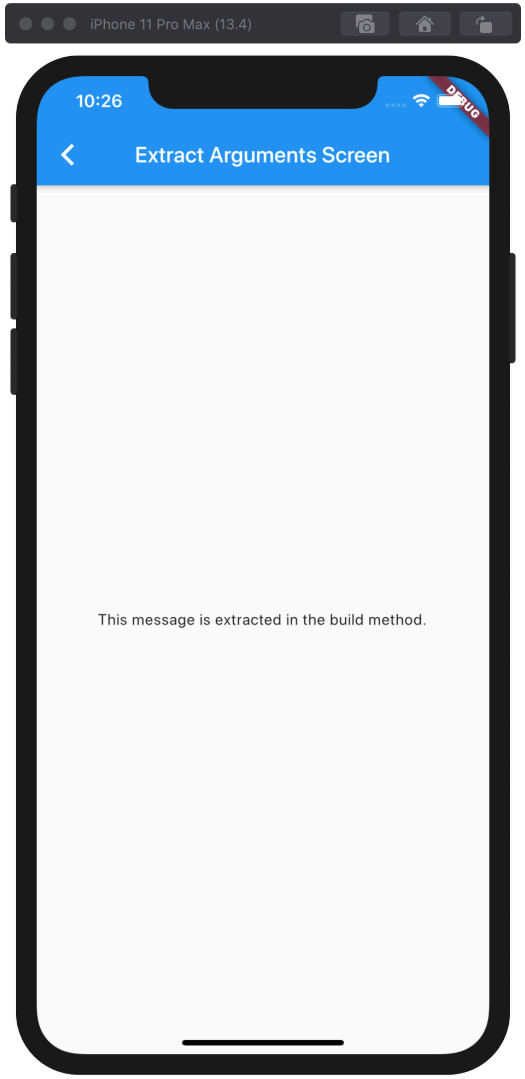
```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      // Provide a function to handle named routes. Use this function to
      // identify the named route being pushed, and create the correct
      // Screen.
      onGenerateRoute: (settings) {
        // If you push the PassArguments route
        if (settings.name == PassArgumentsScreen.routeName) {
          // Cast the arguments to the correct type: ScreenArguments.
          final ScreenArguments args = settings.arguments;

          // Then, extract the required data from the arguments and
          // pass the data to the correct screen.
          return MaterialPageRoute(
            builder: (context) {
              return PassArgumentsScreen(
                title: args.title,
                message: args.message,
              );
            },
          );
        }
```



Navigation

- Pass arguments to a named route

```
      // The code only supports PassArgumentsScreen.routeName right now.
      // Other values need to be implemented if we add them. The assertion
      // here will help remind us of that higher up in the call stack, since
      // this assertion would otherwise fire somewhere in the framework.
      assert(false, 'Need to implement ${settings.name}');
      return null;
    },
    title: 'Navigation with Arguments',
    home: HomeScreen(),
    routes: {
      ExtractArgumentsScreen.routeName: (context) =>
          ExtractArgumentsScreen(),
    });
  }
}
```



Navigation

- Pass arguments to a named route

```
class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Home Screen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            // A button that navigates to a named route that. The named route
            // extracts the arguments by itself.
            RaisedButton(
              child: Text("Navigate to screen that extracts arguments"),
```

- Pass arguments to a named route

```
onPressed: () {
  // When the user taps the button, navigate to a named route
  // and provide the arguments as an optional parameter.
  Navigator.pushNamed(
    context,
    ExtractArgumentsScreen.routeName,
    arguments: ScreenArguments(
      'Extract Arguments Screen',
      'This message is extracted in the build method.',
    ),
  );
},
),
```

Navigation

- Pass arguments to a named route

```
// A button that navigates to a named route. For this route, extract
// the arguments in the onGenerateRoute function and pass them
// to the screen.
RaisedButton(
  child: Text("Navigate to a named that accepts arguments"),
  onPressed: () {
    // When the user taps the button, navigate to a named route
    // and provide the arguments as an optional parameter.
    Navigator.pushNamed(
      context,
      PassArgumentsScreen.routeName,
      arguments: ScreenArguments(
        'Accept Arguments Screen',
        'This message is extracted in the onGenerateRoute function.',
      ),
    );
  },
 ),
 ],
 ),
 ),
 );
 }
}
```

- Pass arguments to a named route

```
// A Widget that extracts the necessary arguments from the ModalRoute.
class ExtractArgumentsScreen extends StatelessWidget {
  static const routeName = '/extractArguments';

  @override
  Widget build(BuildContext context) {
    // Extract the arguments from the current ModalRoute settings and cast
    // them as ScreenArguments.
    final ScreenArguments args = ModalRoute.of(context).settings.arguments;

    return Scaffold(
      appBar: AppBar(
        title: Text(args.title),
      ),
      body: Center(
        child: Text(args.message),
      ),
    );
  }
}
```

- Pass arguments to a named route

```
// A Widget that accepts the necessary arguments via the constructor.
class PassArgumentsScreen extends StatelessWidget {
  static const routeName = '/passArguments';

  final String title;
  final String message;

  // This Widget accepts the arguments as constructor parameters. It does not
  // extract the arguments from the ModalRoute.
  //
  // The arguments are extracted by the onGenerateRoute function provided to the
  // MaterialApp widget.
  const PassArgumentsScreen({
    Key key,
    @required this.title,
    @required this.message,
  }) : super(key: key);
```

Navigation

- Pass arguments to a named route

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(title),
      ),
      body: Center(
        child: Text(message),
      ),
    );
  }
}

// You can pass any object to the arguments parameter. In this example,
// create a class that contains both a customizable title and message.
class ScreenArguments {
  final String title;
  final String message;

  ScreenArguments(this.title, this.message);
}
```

Navigation

- Return data from a screen

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

void main() {
 runApp(MaterialApp(
  title: 'Returning Data',
  home: HomeScreen(),
 ));
}

class HomeScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Text('Returning Data Demo'),
   ),
   body: Center(child: SelectionButton()),
  );
 }
}
```

```
class SelectionButton extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return RaisedButton(
   onPressed: () {
    _navigateAndDisplaySelection(context);
   },
   child: Text('Pick an option, any option!'),
  );
 }

 // A method that launches the SelectionScreen and awaits the result
 // from Navigator.pop.
 _navigateAndDisplaySelection(BuildContext context) async {
  // Navigator.push returns a Future that completes after calling
  // Navigator.pop on the Selection Screen.
  final result = await Navigator.push(
   context,
   MaterialPageRoute(builder: (context) => SelectionScreen()),
  );

  // After the Selection Screen returns a result, hide any previous
  // snackbars and show the new result.
  Scaffold.of(context)
   ..removeCurrentSnackBar()
   ..showSnackBar(SnackBar(content: Text("$result")));
 }
}
```

Navigation

- Return data from a screen

```
class SelectionScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Pick an option'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: RaisedButton(
                onPressed: () {
                  // Close the screen and return "Yep!" as the result.
                  Navigator.pop(context, 'Yep!');
                },
                child: Text('Yep!'),
              ),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: RaisedButton(
                onPressed: () {
                  // Close the screen and return "Nope!" as the result.
                  Navigator.pop(context, 'Nope.');
                },
                child: Text('Nope.'),
              ),
            )
          ],
        ),
      ),
    );
  }
}
```

- Return data from a screen

Navigation

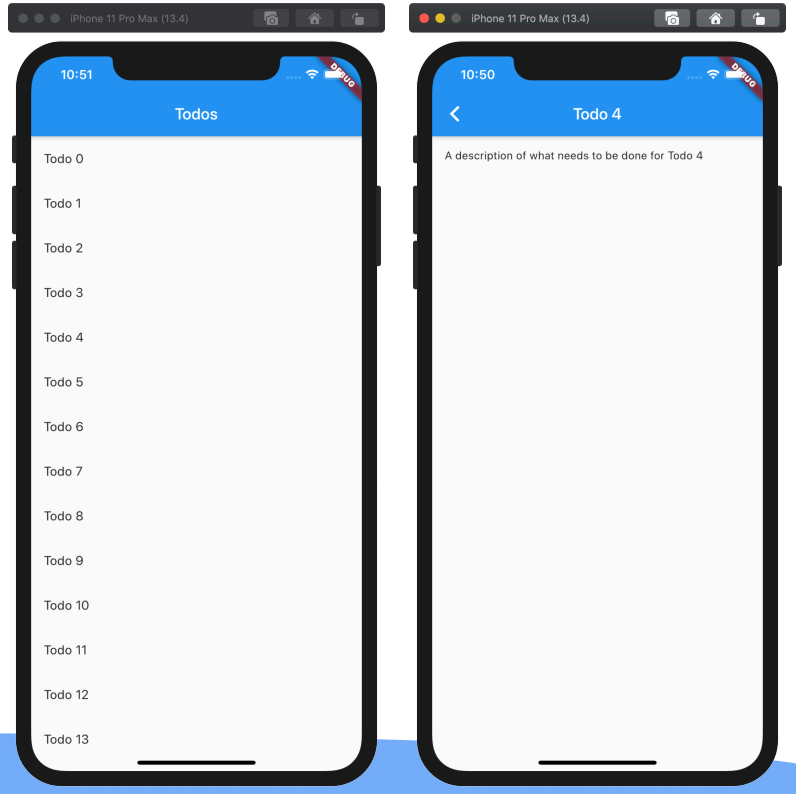
- Send data to a new screen

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

class Todo {
  final String title;
  final String description;

  Todo(this.title, this.description);
}

void main() {
  runApp(MaterialApp(
    title: 'Passing Data',
    home: TodosScreen(
      todos: List.generate(
        20,
          (i) => Todo(
          'Todo $i',
          'A description of what needs to be done for Todo $i',
        ),
      ),
    ),
  ));
}
```

```
class TodosScreen extends StatelessWidget {
  final List<Todo> todos;
  TodosScreen({Key key, @required this.todos}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Todos'),
      ),
      body: ListView.builder(
        itemCount: todos.length,
        itemBuilder: (context, index) {
          return ListTile(
            title: Text(todos[index].title),
            // When a user taps the ListTile, navigate to the DetailScreen.
            // Notice that you're not only creating a DetailScreen, you're
            // also passing the current todo through to it.
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => DetailScreen(todo: todos[index]), ),
              );
            },);},),);
  }
}
```

- Send data to a new screen

```
class DetailScreen extends StatelessWidget {
  // Declare a field that holds the Todo.
  final Todo todo;

  // In the constructor, require a Todo.
  DetailScreen({Key key, @required this.todo}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // Use the Todo to create the UI.
    return Scaffold(
      appBar: AppBar(
        title: Text(todo.title),
      ),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Text(todo.description),
      ),
    );
  }
}
```



Networking

- Fetch data from the internet
  - To install the http package, add it to the dependencies section of the pubspec.yaml file.

```
dependencies:
   http: <latest_version>
```

  - Make a network request

```
Future<http.Response> fetchAlbum() {
  return http.get('https://jsonplaceholder.typicode.com/albums/1’);
}
```

- Create an Album class

```
class Album {
  final int userId;
  final int id;
  final String title;

  Album({this.userId, this.id, this.title});

  factory Album.fromJson(Map<String, dynamic> json) {
    return Album(
      userId: json['userId'],
      id: json['id'],
      title: json['title'],
    );
  }
}
```

- Convert the http.Response to an Album

```
Future<Album> fetchAlbum() async {
  final response = await http.get('https://jsonplaceholder.typicode.com/albums/1');

  if (response.statusCode == 200) {
    // If the server did return a 200 OK response,
    // then parse the JSON.
    return Album.fromJson(json.decode(response.body));
  } else {
    // If the server did not return a 200 OK response,
    // then throw an exception.
    throw Exception('Failed to load album');
  }
}
```

- Fetch the data
  - Call the fetch() method in either the initState() or didChangeDependencies() methods.

```
class _MyAppState extends State<MyApp> {
  Future<Album> futureAlbum;

  @override
  void initState() {
    super.initState();
    futureAlbum = fetchAlbum();
  }
```

- Display the data
  - To display the data on screen, use the FutureBuilder widget.

```
FutureBuilder<Album>(
  future: futureAlbum,
  builder: (context, snapshot) {
    if (snapshot.hasData) {
      return Text(snapshot.data.title);
    } else if (snapshot.hasError) {
      return Text("${snapshot.error}");
    }

    // By default, show a loading spinner.
    return CircularProgressIndicator();
  },
);
```

- Sending data to server

```
Future<http.Response> createAlbum(String title) {
  return http.post(
    'https://jsonplaceholder.typicode.com/albums',
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode(<String, String>{
      'title': title,
    }),
  );
}
```

- Sending data to server

```
Future<http.Response> createAlbum(String title) {
  return http.post(
    'https://jsonplaceholder.typicode.com/albums',
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode(<String, String>{
      'title': title,
    }),
  );
}
```

- Convert the http.Response to an Album

```
Future<Album> createAlbum(String title) async {
  final http.Response response = await http.post(
    'https://jsonplaceholder.typicode.com/albums',
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode(<String, String>{
      'title': title,
    }),
  );
  if (response.statusCode == 201) {
    // If the server did return a 201 CREATED response,
    // then parse the JSON.
    return Album.fromJson(json.decode(response.body));
  } else {
    // If the server did not return a 201 CREATED response,
    // then throw an exception.
    throw Exception('Failed to load album');
  }
}
```

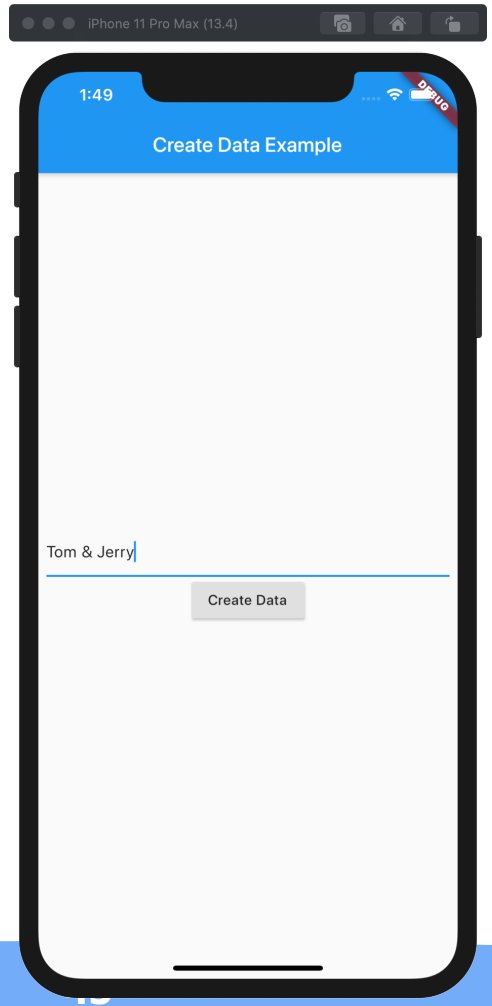
- Get a title from user input

```
Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: TextField(
        controller: _controller,
        decoration: InputDecoration(hintText: 'Enter Title'),
      ),
    ),
    RaisedButton(
      child: Text('Create Data'),
      onPressed: () {
        setState(() {
          _futureAlbum = createAlbum(_controller.text);
        });
      },
    ),
  ],
)
```

Networking

- Display the response on screen

```
FutureBuilder<Album>(
  future: _futureAlbum,
  builder: (context, snapshot) {
    if (snapshot.hasData) {
      return Text(snapshot.data.title);
    } else if (snapshot.hasError) {
      return Text("${snapshot.error}");
    }

    return CircularProgressIndicator();
  },
)
```



References

- Google Flutter
  - http://www.flutter.dev
- Flutter Gallery
  - https://flutter.github.io/gallery/#/
- Flutter Samples
  - https://flutter.github.io/samples/#
- Android Studio: License for package Android SDK Build-Tools 28.0.2 not accepted.
  - https://blog.csdn.net/didah/article/details/87894670
- A tour of the Dart language
  - https://dart.dev/guides/language/language-tour#class-variables-and-methods
- Dart Language Specification v2.2
  - https://dart.dev/guides/language/specifications/DartLangSpec-v2.2.pdf
- DartPad
  - https://dartpad.dev/
- Language samples
  - https://dart.dev/samples
- pub.dev
  - https://pub.dev/

- *Web*开发技术
- *Web Application Development*

Thank You!