FLUTTER TUTORIALS

BY: **TECMAN**

TECMAN Lesson 2B

WRITING YOUR FIRST FLUTTER APP PART 2

In this lesson, you'll extend a basic Flutter app to include interactivity. You'll also create a second page (called a *route*) that the user can navigate to. Finally, you'll modify the app's theme (color). This lesson extends part 1,

TECMAN LIMITED 2019

What you'll learn in part 2

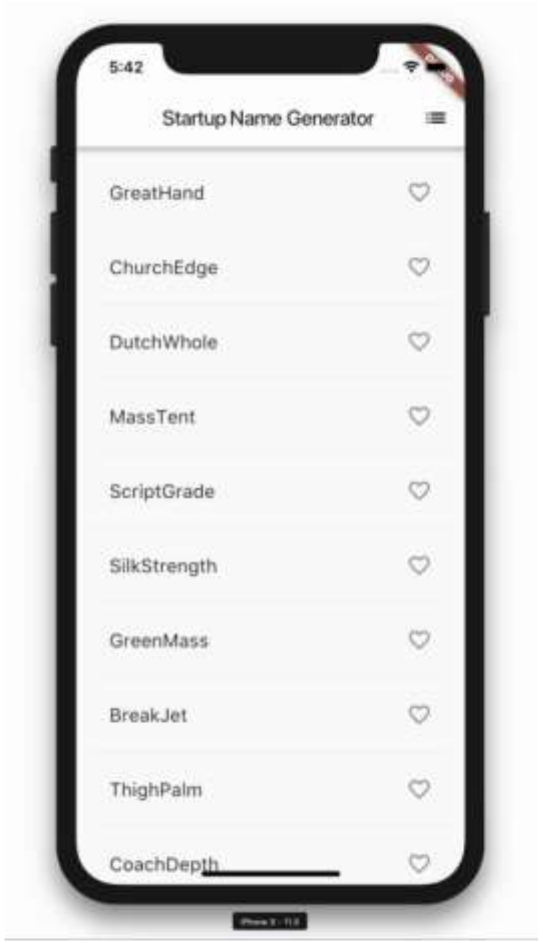
- How to write a Flutter app that looks natural on both iOS and Android.
- Using hot reload for a quicker development cycle.
- How to add interactivity to a stateful widget.
- How to create and navigate to a second screen.
- How to change the look of an app using themes.

TECMAN LIMITED 2019

What you'll build in part 2

You'll start with a simple mobile app that generates an endless list of proposed names for a startup company. By the end of this lesson, the user can select and unselect names, saving the best ones. Tapping the list icon in the upper right of the app bar navigates to a new page (called a *route*) that lists only the favorited names.

What you'll build in part 2



GreatHand
ChurchEdge
DutchWhole
MassTent
ScriptGrade
SilkStrength
GreenMass
BreakJet
ThighPalm
CoachDepth

TECMAN LIMITED 2019

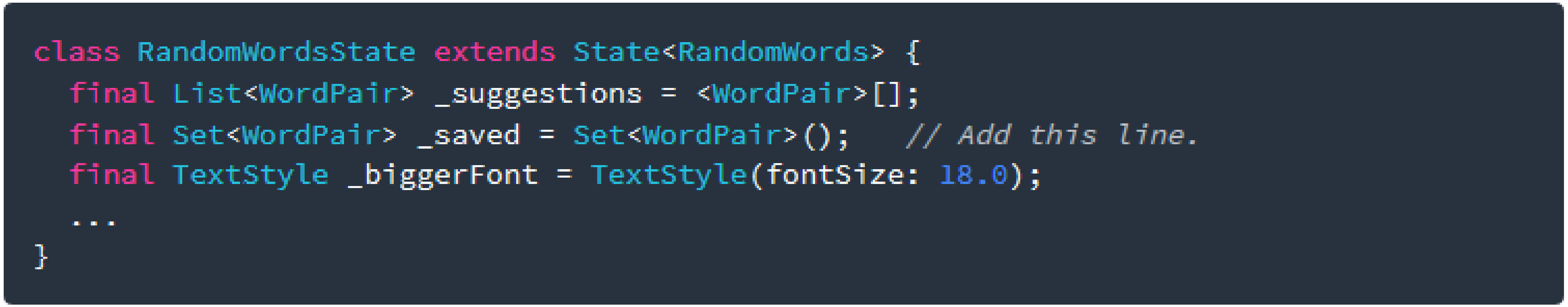
Add icons to the list

In this step, you'll add heart icons to each row. In the next step, you'll make them tappable and save the favorites..

Add a _saved Set to RandomWordsState. This Set stores the word pairings that the user favorited. Set is preferred to List because a properly implemented Set does not allow duplicate entries.

TECMAN LIMITED 2019

Add icons to the list



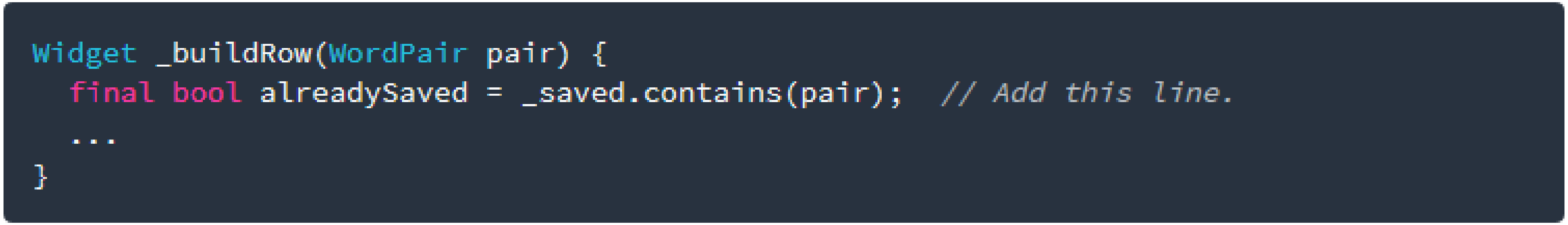
```
class RandomWordsState extends State<RandomWords> {
  final List<WordPair> _suggestions = <WordPair>[];
  final Set<WordPair> _saved = Set<WordPair>();   // Add this line.
  final TextStyle _biggerFont = TextStyle(fontSize: 18.0);
  ...
}
```

TECMAN LIMITED 2019

Add icons to the list

In the _buildRow function, add an alreadySaved check to ensure that a word pairing has not already been added to favorites.

Add icons to the list



```
Widget _buildRow(WordPair pair) {
  final bool alreadySaved = _saved.contains(pair);  // Add this line.
  ...
}
```

TECMAN LIMITED 2019

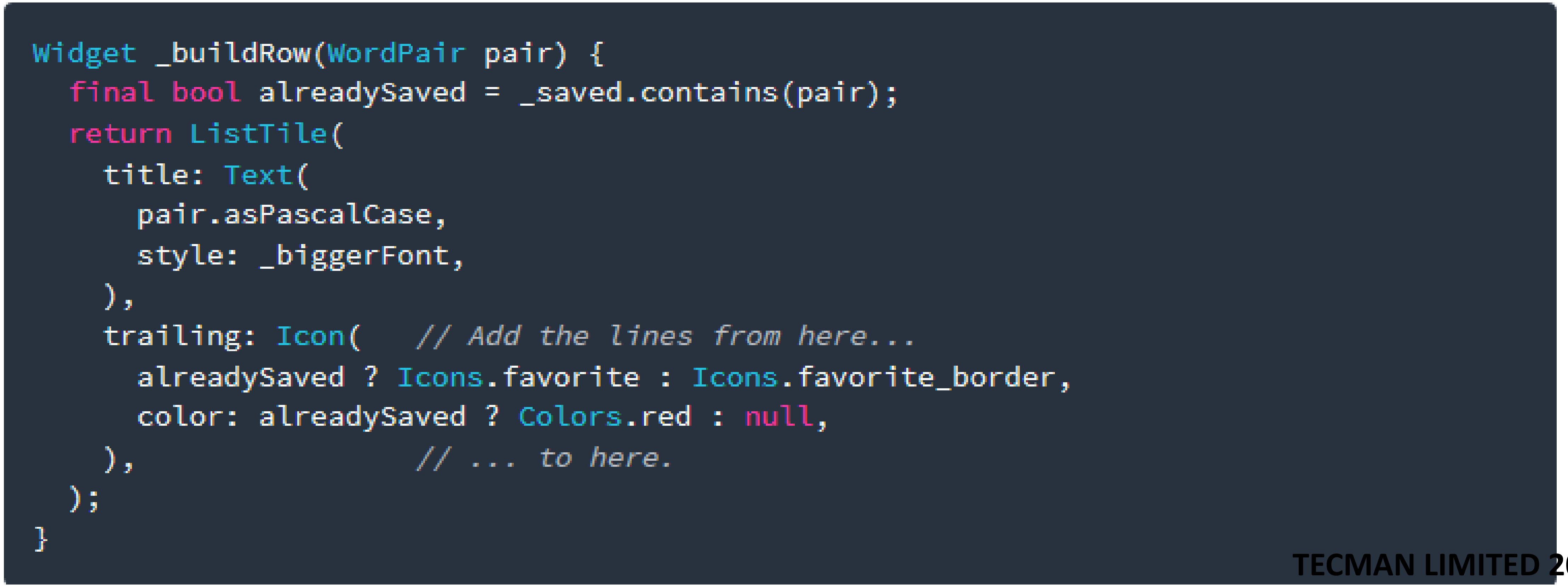
Add icons to the list

In _buildRow() you'll also add heart-shaped icons to the ListTile objects to enable favoriting. In the next step, you'll add the ability to interact with the heart icons.

TECMAN LIMITED 2019

Add icons to the list

Add the icons, as shown below:

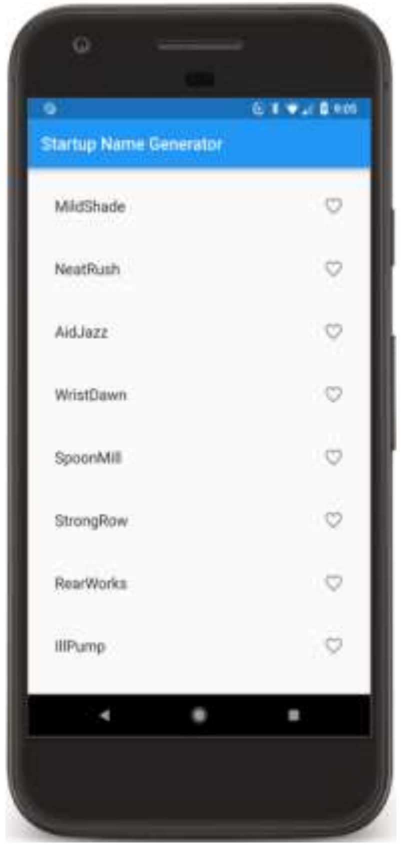


```
Widget _buildRow(WordPair pair) {
  final bool alreadySaved = _saved.contains(pair);
  return ListTile(
    title: Text(
      pair.asPascalCase,
      style: _biggerFont,
    ),
    trailing: Icon(   // Add the lines from here...
      alreadySaved ? Icons.favorite : Icons.favorite_border,
      color: alreadySaved ? Colors.red : null,
    ),                // ... to here.
  );
}
```

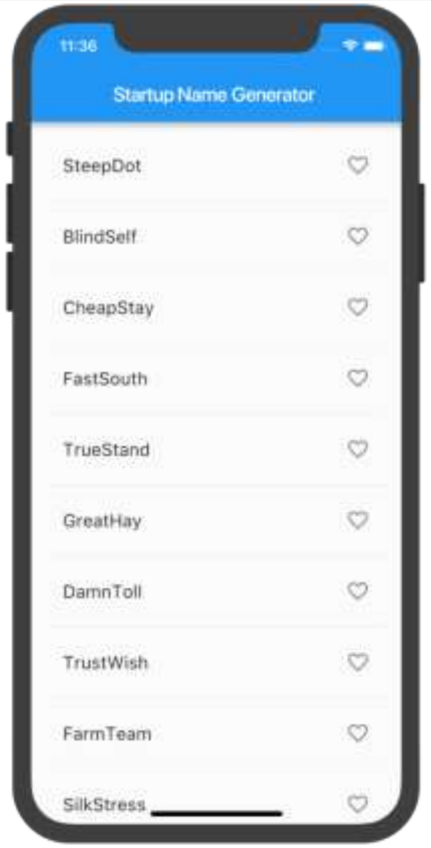
Add icons to the list

Hot reload the app. You should now see open hearts on each row, but they are not yet interactive.

Add icons to the list



ANDROID



iOS

TECMAN LIMITED 2019

Add interactivity

In this step, you'll make the heart icons tappable. When the user taps an entry in the list, toggling its "favorited" state, that word pairing is added or removed from a set of saved favorites.

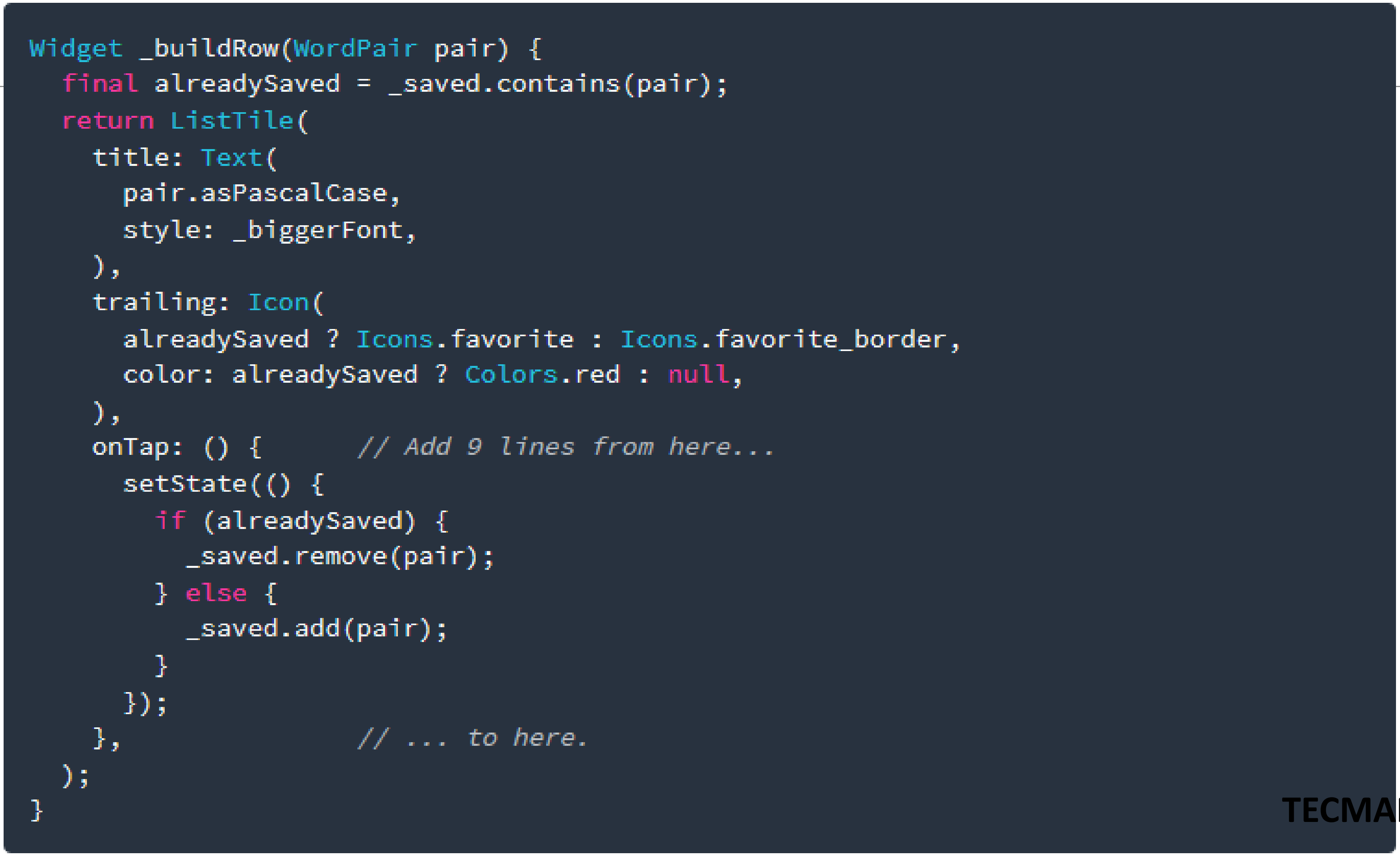
To do this, you'll modify the _buildRow function. If a word entry has already been added to favorites, tapping it again removes it from favorites. When a tile has been tapped, the function calls setState() to notify the framework that state has changed.

Add interactivity

Add onTap, as shown below:

TECMAN LIMITED 2019

Add interactivity

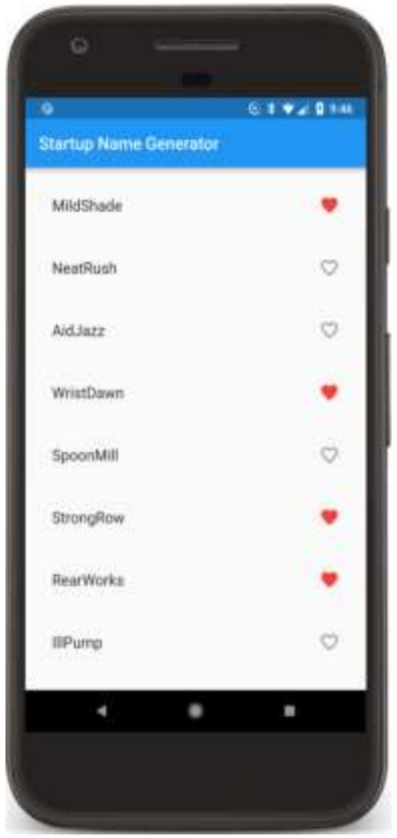


```
Widget _buildRow(WordPair pair) {
  final alreadySaved = _saved.contains(pair);
  return ListTile(
    title: Text(
      pair.asPascalCase,
      style: _biggerFont,
    ),
    trailing: Icon(
      alreadySaved ? Icons.favorite : Icons.favorite_border,
      color: alreadySaved ? Colors.red : null,
    ),
    onTap: () {      // Add 9 lines from here...
      setState(() {
        if (alreadySaved) {
          _saved.remove(pair);
        } else {
          _saved.add(pair);
        }
      });
    },               // ... to here.
  );
}
```

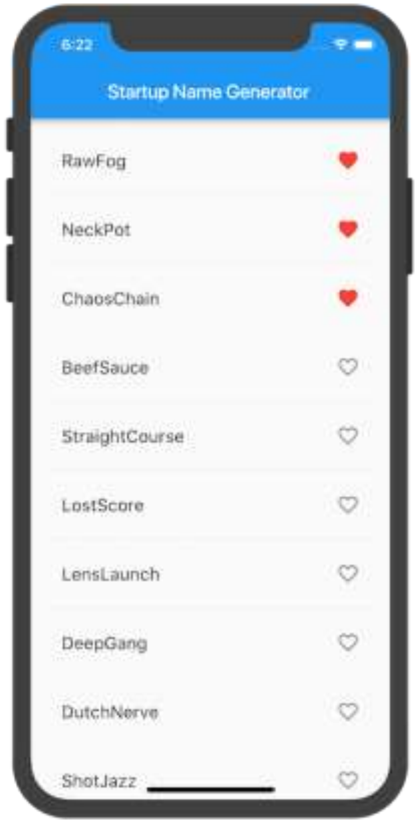
Add interactivity

Hot reload the app. You should be able to tap any tile to favorite, or unfavorite, the entry. Tapping a tile generates an implicit ink splash animation emanating from the tap point.

Add interactivity



ANDROID



iOS

TECMAN LIMITED 2019

Navigate to a new screen

In this step, you'll add a new page (called a route in Flutter) that displays the favorites. You'll learn how to navigate between the home route and the new route.

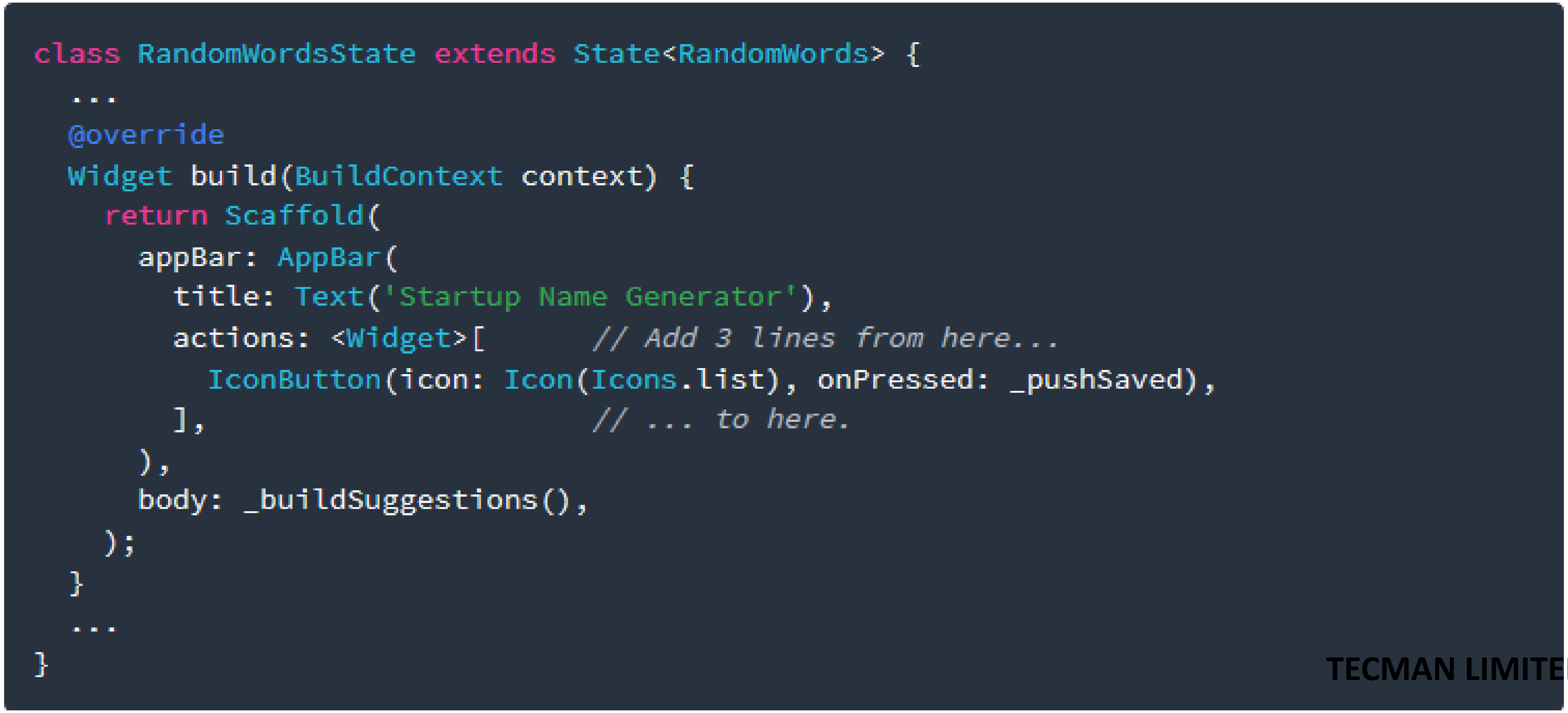
In Flutter, the Navigator manages a stack containing the app's routes. Pushing a route onto the Navigator's stack, updates the display to that route. Popping a route from the Navigator's stack, returns the display to the previous route.

Navigate to a new screen

Next, you'll add a list icon to the AppBar in the build method for RandomWordsState. When the user clicks the list icon, a new route that contains the saved favorites is pushed to the Navigator, displaying the icon.

Add the icon and its corresponding action to the build method:

Navigate to a new screen



```dart
class RandomWordsState extends State<RandomWords> {
  ...
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Startup Name Generator'),
        actions: <Widget>[      // Add 3 lines from here...
          IconButton(icon: Icon(Icons.list), onPressed: _pushSaved),
        ],                      // ... to here.
      ),
      body: _buildSuggestions(),
    );
  }
  ...
}
```

Navigate to a new screen

Add a _pushSaved() function to the RandomWordsState class.

```
void _pushSaved() {
}
```

TECMAN LIMITED 2019

Navigate to a new screen

Hot reload the app. The list icon ( ) appears in the app bar. Tapping it does nothing yet, because the _pushSaved function is empty.

Next, you'll build a route and push it to the Navigator's stack. This action changes the screen to display the new route. The content for the new page is built in MaterialPageRoute's builder property, in an anonymous function.

Navigate to a new screen

Call Navigator.push, as shown below, which pushes the route to the Navigator's stack. The IDE will complain about invalid code, but you will fix that in the next section.

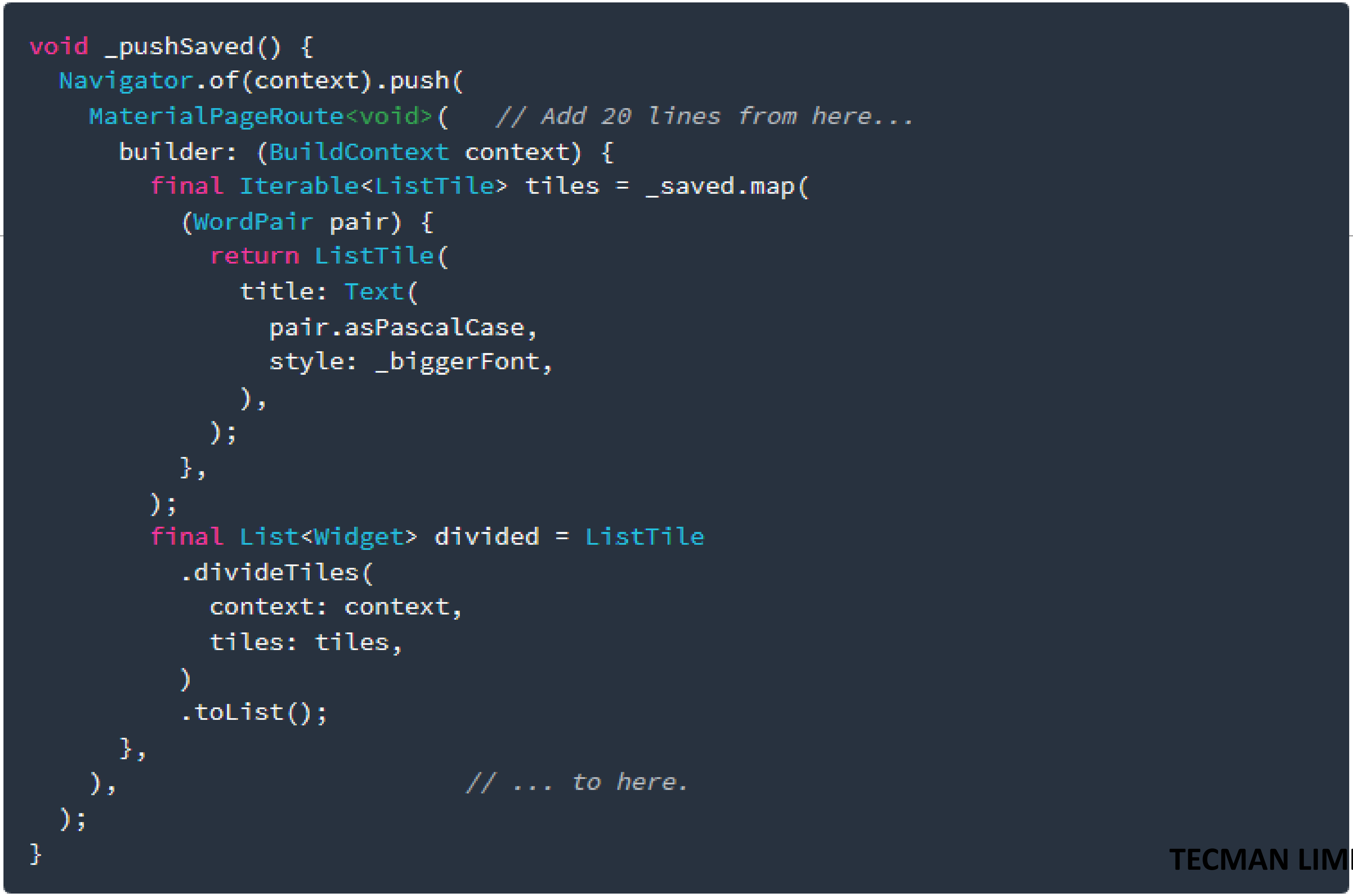
Navigate to a new screen

```
void _pushSaved() {
  Navigator.of(context).push(
  );
}
```

Navigate to a new screen

Next, you'll add the MaterialPageRoute and its builder. For now, add the code that generates the ListTile rows. The divideTiles() method of ListTile adds horizontal spacing between each ListTile. The divided variable holds the final rows, converted to a list by the convenience function, toList().

Add the code, as shown below:



```
void _pushSaved() {
  Navigator.of(context).push(
    MaterialPageRoute<void>(   // Add 20 lines from here...
      builder: (BuildContext context) {
        final Iterable<ListTile> tiles = _saved.map(
          (WordPair pair) {
            return ListTile(
              title: Text(
                pair.asPascalCase,
                style: _biggerFont,
              ),
            );
          },
        );
        final List<Widget> divided = ListTile
          .divideTiles(
            context: context,
            tiles: tiles,
          )
          .toList();
      },
    ),                         // ... to here.
  );
}
```

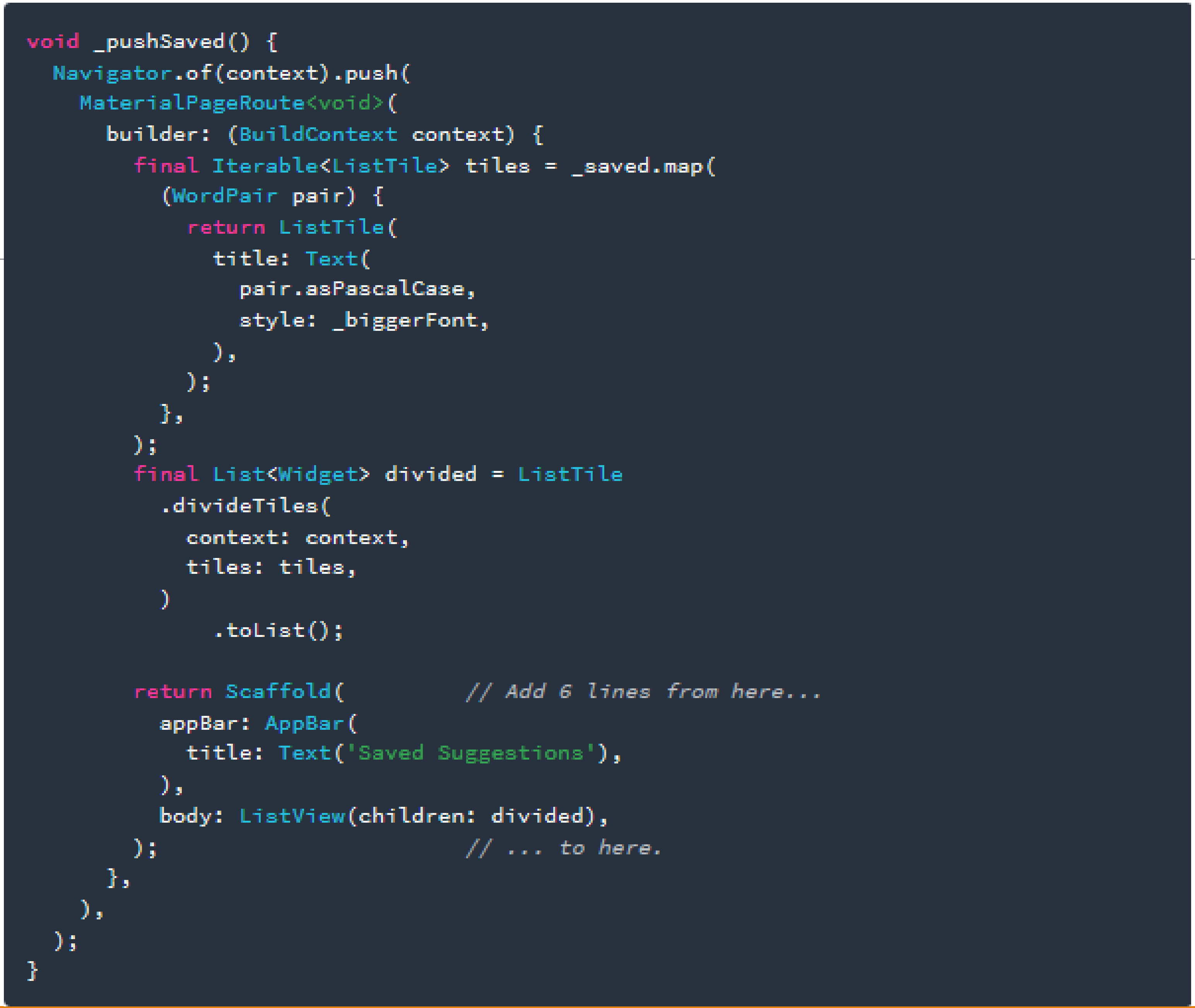
TECMAN LIMITED 2019

Navigate to a new screen

The builder property returns a Scaffold, containing the app bar for the new route, named "Saved Suggestions." The body of the new route consists of a ListView containing the ListTiles rows; each row is separated by a divider.

Add horizontal dividers, as shown below:

TECMAN LIMITED 2019



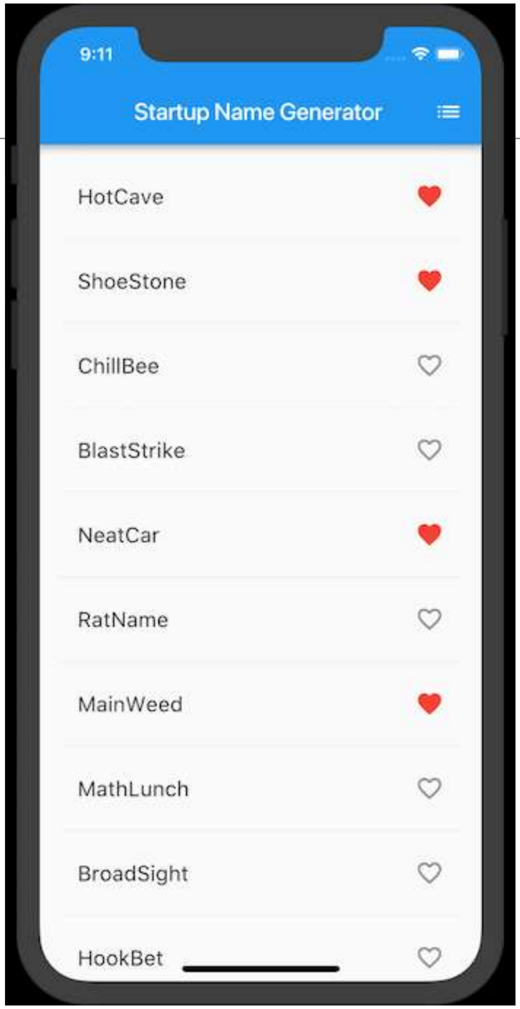
```
void _pushSaved() {
  Navigator.of(context).push(
    MaterialPageRoute<void>(
      builder: (BuildContext context) {
        final Iterable<ListTile> tiles = _saved.map(
          (WordPair pair) {
            return ListTile(
              title: Text(
                pair.asPascalCase,
                style: _biggerFont,
              ),
            );
          },
        );
        final List<Widget> divided = ListTile
          .divideTiles(
            context: context,
            tiles: tiles,
          )
              .toList();

        return Scaffold(         // Add 6 lines from here...
          appBar: AppBar(
            title: Text('Saved Suggestions'),
          ),
          body: ListView(children: divided),
        );                       // ... to here.
      },
    ),
  );
}
```

TECMAN LIMITED 2019

Navigate to a new screen

Hot reload the app. Favorite some of the selections and tap the list icon in the app bar. The new route appears containing the favorites. Note that the Navigator adds a "Back" button to the app bar. You did not have to explicitly implement Navigator.pop. Tap the back button to return to the home route.

Navigate to a new screen

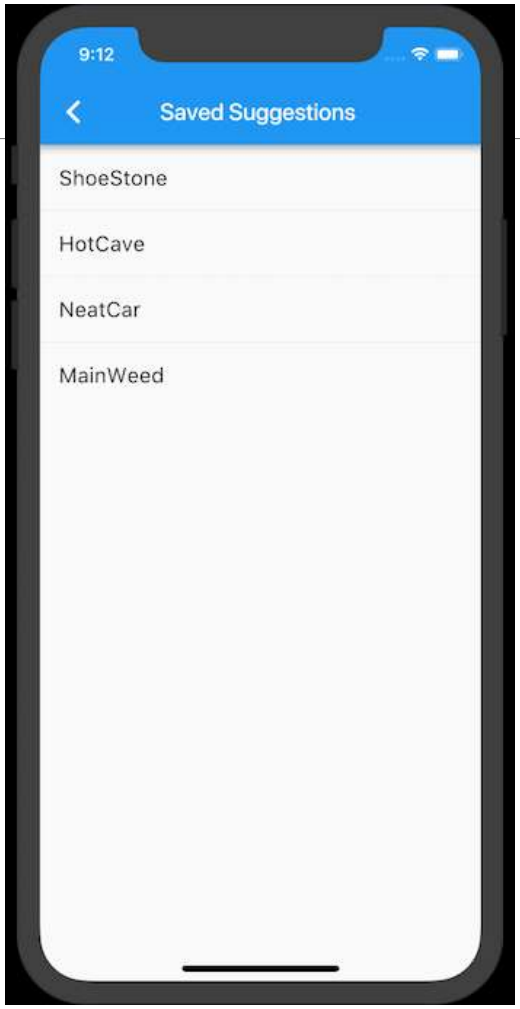


9:11

Startup Name Generator

HotCave ♥
ShoeStone ♥
ChillBee ♡
BlastStrike ♡
NeatCar ♥
RatName ♡
MainWeed ♥
MathLunch ♡
BroadSight ♡
HookBet ♡

Main route



9:12

Saved Suggestions

ShoeStone
HotCave
NeatCar
MainWeed

Saved suggestions route

TECMAN LIMITED 2019

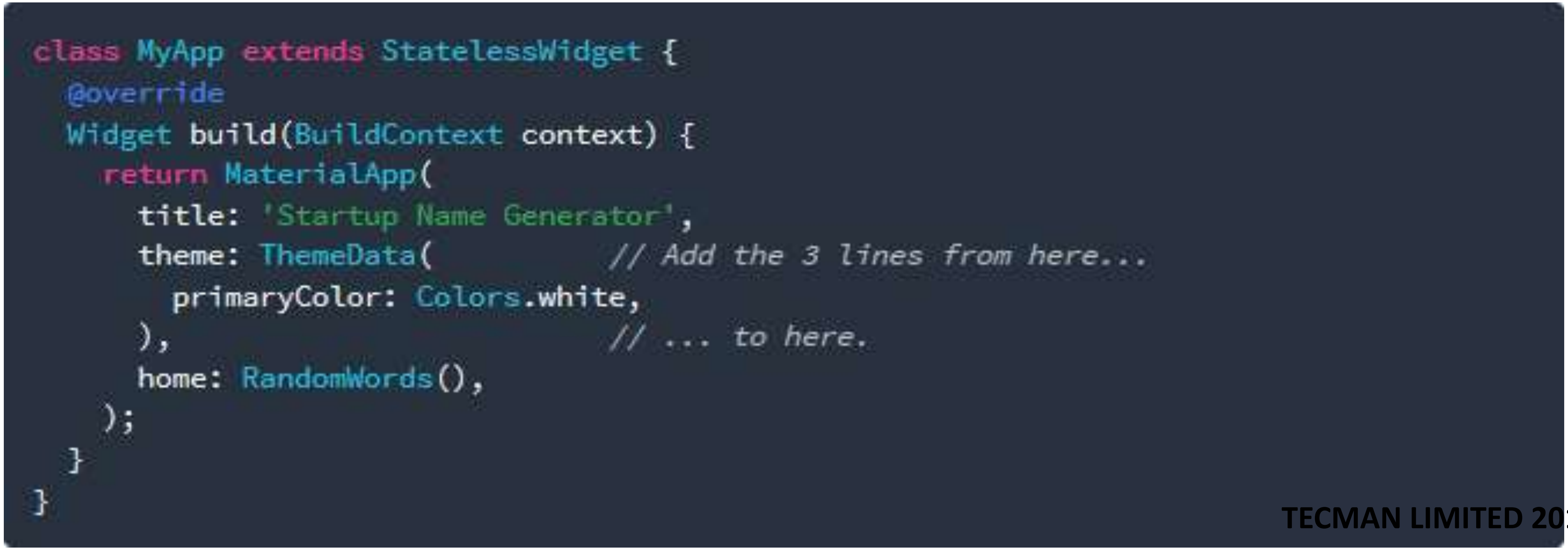
Change the UI using Themes

In this step, you'll modify the app's theme. The theme controls the look and feel of your app. You can either use the default theme, which is dependent on the physical device or emulator, or customize the theme to reflect your branding.

You can easily change an app's theme by configuring the ThemeData class. This app currently uses the default theme, but you'll change the app's primary color to white.

Change the UI using Themes

Change the color in the MyApp class:



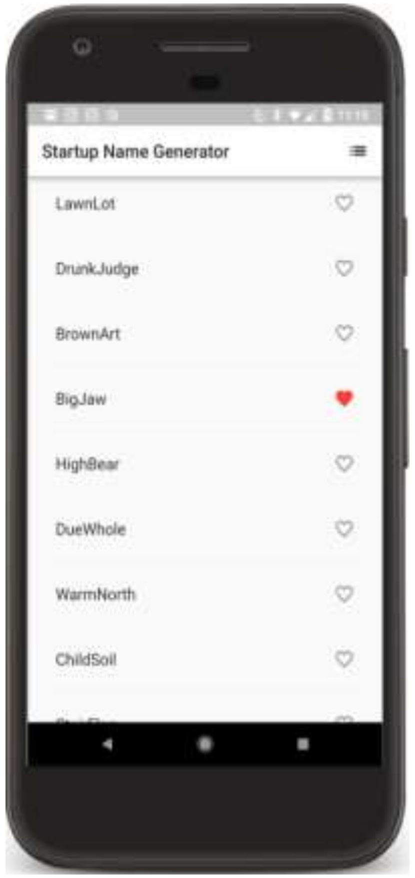
```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Startup Name Generator',
      theme: ThemeData(          // Add the 3 lines from here...
        primaryColor: Colors.white,
      ),                         // ... to here.
      home: RandomWords(),
    );
  }
}
```

Change the UI using Themes

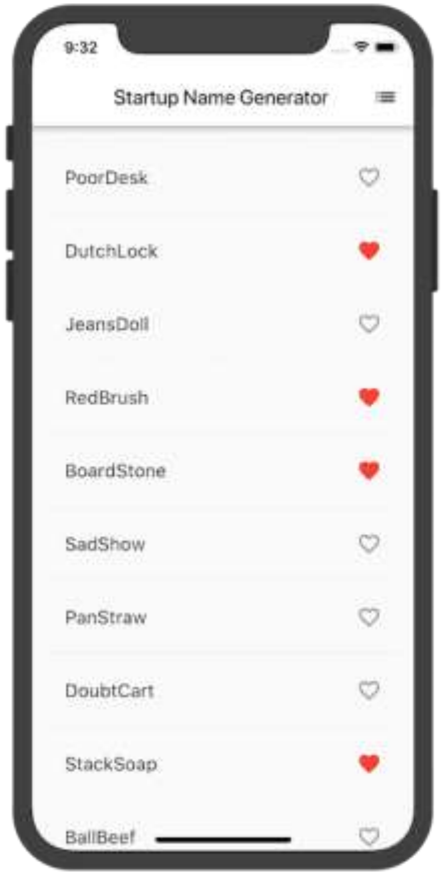
Hot reload the app. The entire background is now white, even the app bar.

As an exercise for the reader, use ThemeData to change other aspects of the UI. The Colors class in the Material library provides many color constants you can play with, and hot reload makes experimenting with the UI quick and easy.

Change the UI using Themes



Android



iOS

TECMAN LIMITED 2019

Well done!

You've written an interactive Flutter app that runs on both iOS and Android. In this lesson, you've:

- Written Dart code.
- Used hot reload for a faster development cycle.
- Implemented a stateful widget, adding interactivity to your app.
- Created a route and added logic for moving between the home route and the new route.
- Learned about changing the look of your app's UI using themes.

TECMAN LIMITED 2019

THANK YOU

TECMAN LIMITED 2019