Introducing Flutter

john.dumais@comcast.net
https://github.com/GraciesPadre

Flutter as a cross-device development framework

- Quick comparison between flutter & other cross-device frameworks
- Enough Dart to build a flutter app
- Getting started and the basic layout of a flutter app
- The use of Material and Cupertino widget sets
- Hot reload
- Consuming and displaying information from a web service
- Test framework
- Stuff I haven’t tried yet

Quick comparison between flutter & other cross-platform development frameworks

- Javascript embedded web pages
    - Cordova, Phone Gap
        - Manipulating the DOM is expensive, so doesn’t perform all that well.
- Javascript bridge
    - An app needing to talk directly to the rendering engine needs to be compiled to native code and cross the javascript bridge twice
- Frameworks that use native capabilities
    - Xamarin, with or without forms

Quick comparison between flutter & other cross-platform development frameworks

- Flutter
  - Compiles to ARM code
  - Has its own rendering engine

Quick comparison between flutter & other cross-platform development frameworks

Quick comparison between flutter & other cross-platform development frameworks

- Dart
- Code sharing between web & mobile apps
    - Angular Dart & Flutter app for same models & controllers
    - Each needs its own views

Dart

- Syntax similar to java
- Similar to go in some respects
    - Concurrency using the CSP model
        - Where go uses channels & go routines, Dart uses “isolates”
            - Has its own private chunk of memory and runs an event loop
            - No threads

Dart

- Observer pattern is codified
    - Very similar to ReactiveX

Dart

- 2 forms of visibility
  - Controlled through prepending _
    - Public
    - Package private
      - Everything in a package is visible to everything else in the same package

Dart

Visibility

```dart
class PetsDisplayPage extends StatefulWidget {
  PetsDisplayPage({Key key, @required petRetrievalTag}) : _petRetrievalTag = petRetrievalTag, super(key: key);

  final String _petRetrievalTag;

  @override
  _PetsDisplayPageState createState() => _PetsDisplayPageState(_petRetrievalTag);
}
```

Getting started

- https://flutter.dev
- Installs a set of framework tools available through a terminal window
- Builds using Android Studio and Xcode
- Can use Android emulator or iOS simulator or provisioned physical device

Getting started

- Visual studio code has flutter plugins
- “Awesome flutter snippets”
    - Commonly-used constructs
    - A pretty wide array of icon images

Getting started

- flutter create project_name
    - Creates folder structure with skeleton app for both iOS & Android
        - Includes pubspec.yaml
            - Metadata to specify dependencies
                - flutter pub get
- Runs flutter doctor

Getting started

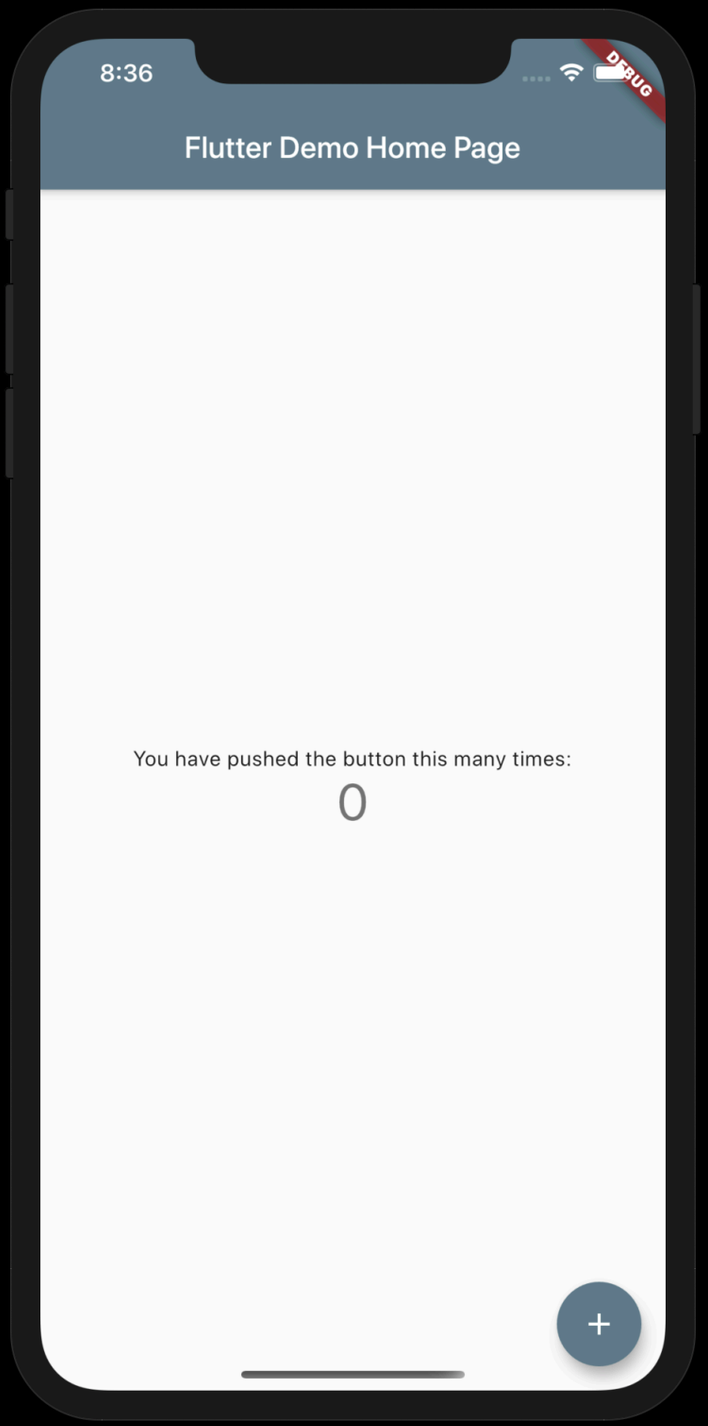
- flutter doctor

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, v1.12.13+hotfix.8, on Mac OS X 10.15.3 19D76,
    locale en-US)
[✓] Android toolchain - develop for Android devices (Android SDK version 29.0.2)
[✓] Xcode - develop for iOS and macOS (Xcode 11.3.1)
[✓] Android Studio (version 3.5)
[!] Connected device
    ! No devices available

! Doctor found issues in 1 category.
```

Getting started

- Creates a “counter” app
    - cd project_name
    - flutter run

Getting started



Anatomy of an app

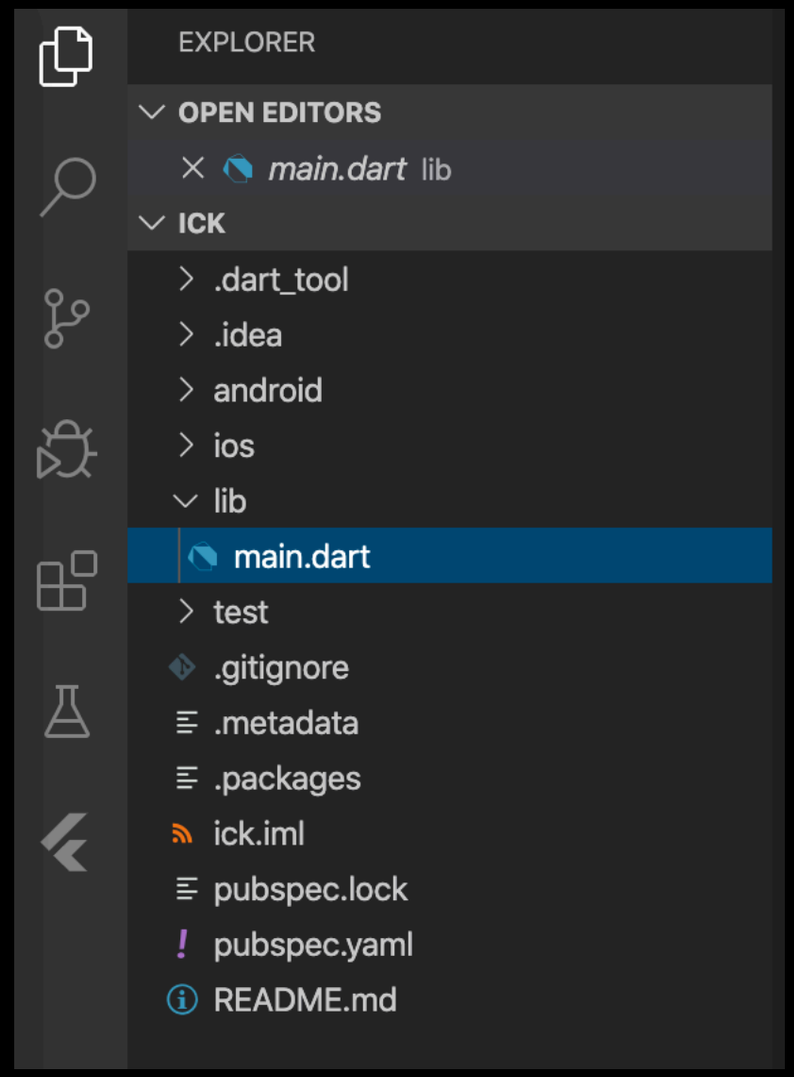
- Pages are composed hierarchically
  - Decorator pattern handles stuff like schemes, padding, etc.
- Most things you will spend time with are Widgets
  - Either StatelessWidget or StatefulWidget
- Native hosting component

Anatomy of an app

```
import UIKit
import Flutter

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
  override func application(
    _ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
  ) -> Bool {
    GeneratedPluginRegistrant.register(with: self)
    return super.application(application, didFinishLaunchingWithOptions: launchOptions)
  }
}
```

Anatomy of an app

- Your app-specific code goes in the lib folder



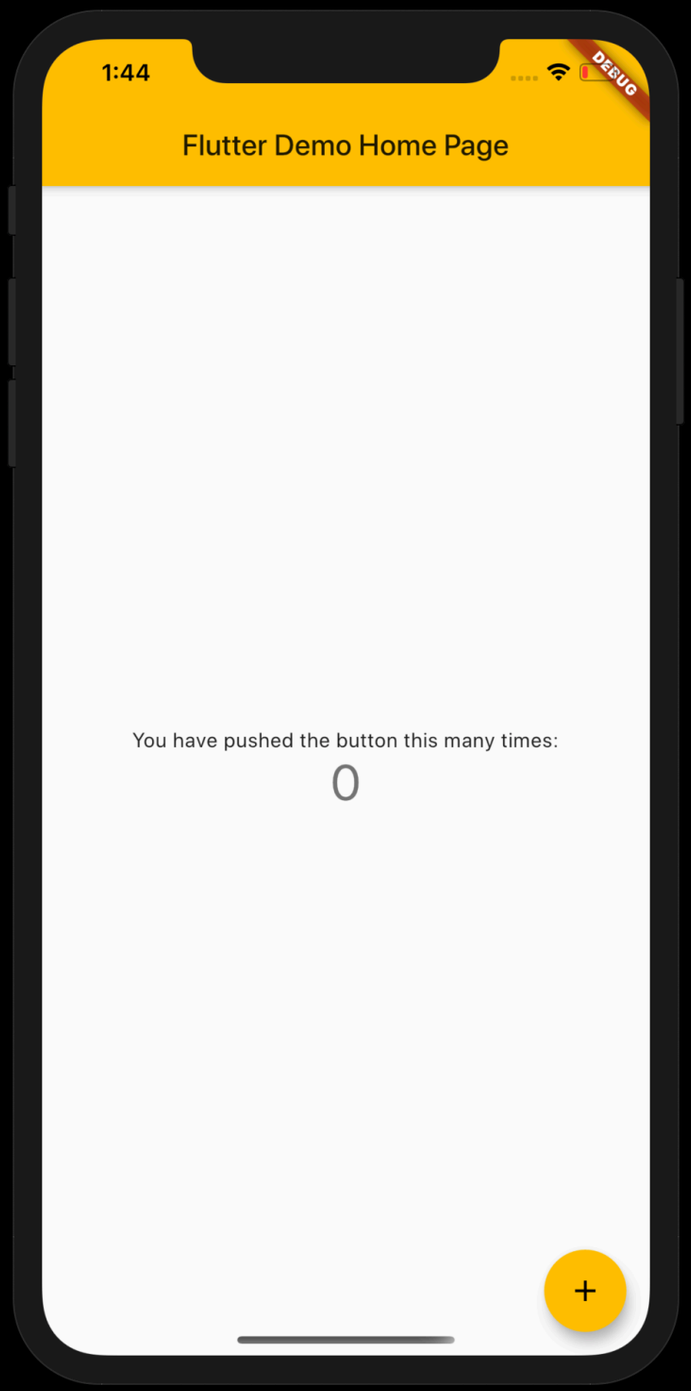
Anatomy of an app

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

```dart
class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;

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

```dart
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.display1,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
}
```

Hot Reload



iPhone 11 Pro Max — 13.3

Different App Themes

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

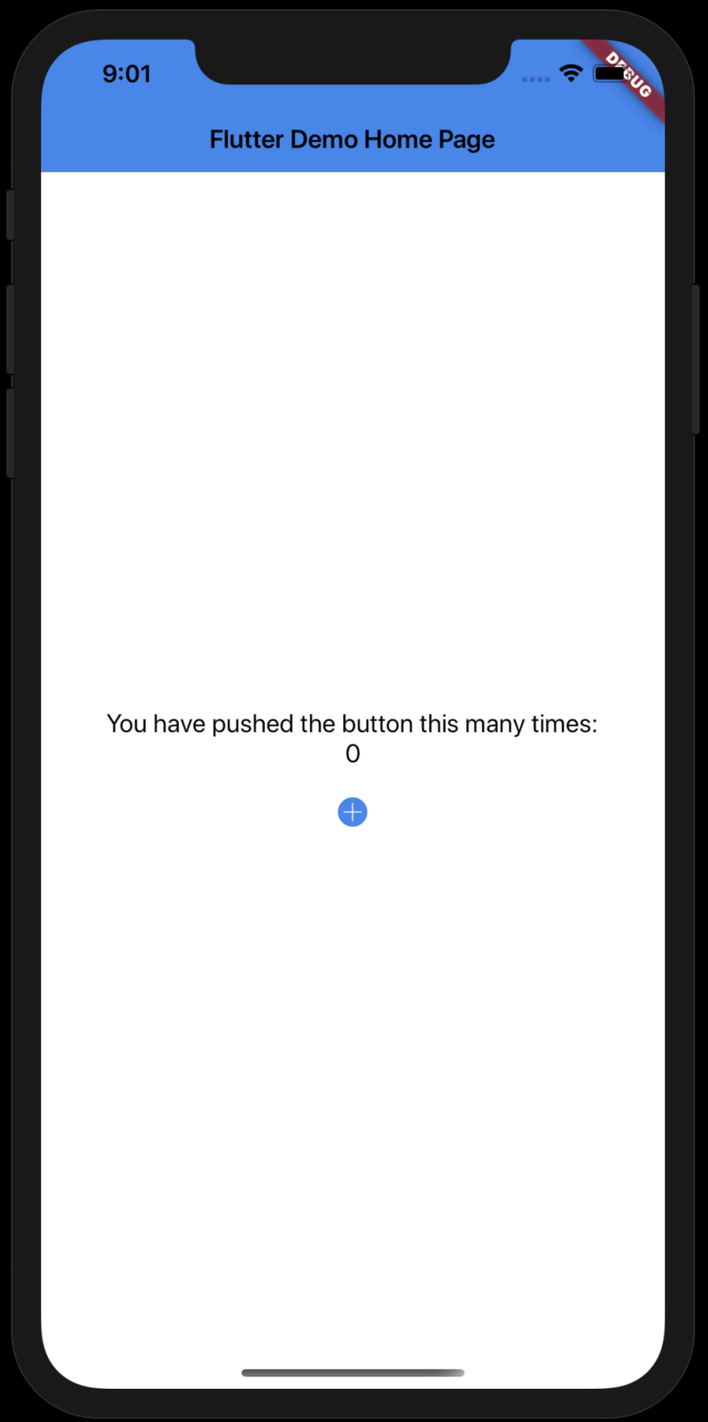
```dart
import 'package:flutter/cupertino.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return CupertinoApp(
      title: 'Flutter Demo',
      theme: CupertinoThemeData(
        primaryColor: Color.fromRGBO(74, 134, 232, 1),
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
```

Different App Themes

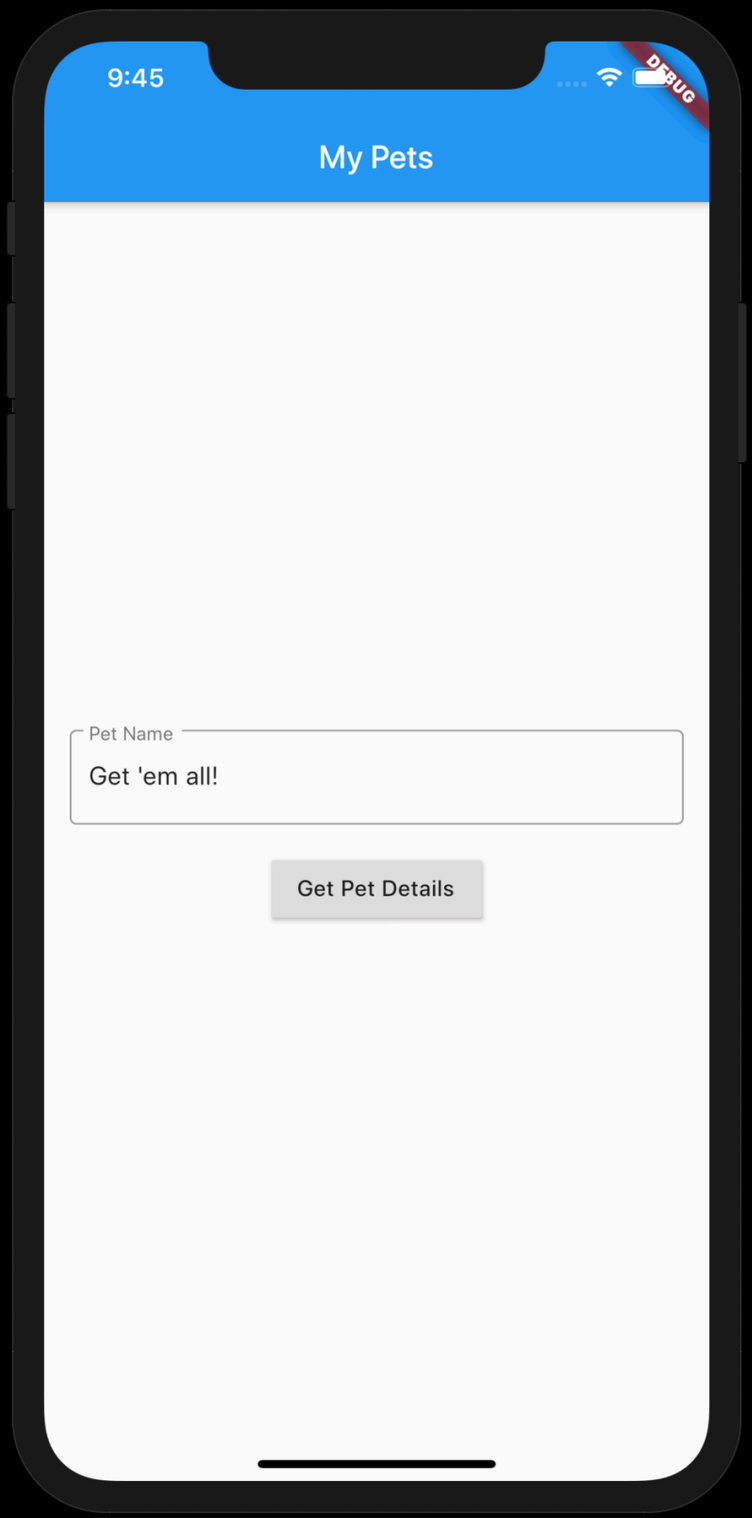
```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Text(
            'You have pushed the button this many times:',
          ),
          Text(
            '$_counter',
            style: Theme.of(context).textTheme.display1,
          ),
        ],
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
      child: Icon(Icons.add),
    ),
  );
}
}
```

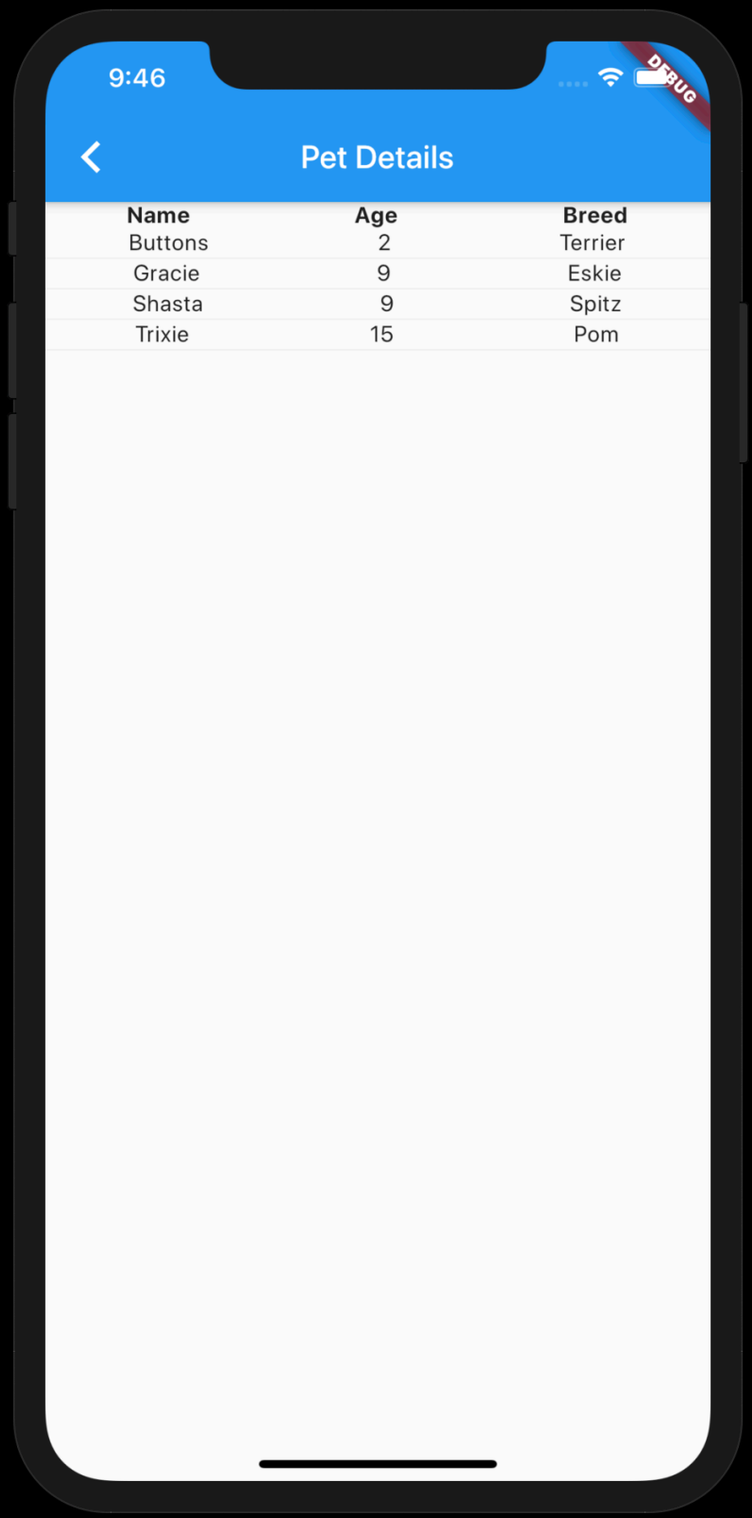
```dart
@override
Widget build(BuildContext context) {
  return CupertinoPageScaffold(
    navigationBar: CupertinoNavigationBar(
      middle: Text(widget.title),
      backgroundColor: CupertinoTheme.of(context).primaryColor,
    ),
    child: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Text(
            'You have pushed the button this many times:',
          ),
          Text(
            '$_counter',
            style: CupertinoTheme.of(context).textTheme.textStyle,
          ),
          CupertinoButton(
            onPressed: _incrementCounter,
            child: Icon(CupertinoIcons.add_circled_solid),
          )
        ],
      ),
    ),
  );
```

Different App Themes



Data from a web service





Data from a web service

- Need an entry in pubspec.yaml
- Model
  - Could use https://flutter.dev/docs/development/data-and-backend/json to generate json parsing code
- Http client

Data from a web service

- Need an entry in pubspec.yaml

```
version: 1.0.0+1

environment:
  sdk: ">=2.1.0 <3.0.0"

dependencies:
  http: "^0.12.0+1"
```

Data from a web service

- Model
- https://github.com/GraciesPadre/petServer


```
{
  "pets_collection": {
    "Buttons": {
      "age": 2,
      "breed": "Terrier"
    },
    "Gracie": {
      "age": 9,
      "breed": "Eskie"
    },
    "Shasta": {
      "age": 9,
      "breed": "Spitz"
    },
    "Trixie": {
      "age": 15,
      "breed": "Pom"
    }
  }
}
```

Data from a web service

- Model

```dart
class PetData {
  PetData(this.name, this.age, this.breed);

  final String name;
  final int age;
  final String breed;

  factory PetData.fromJson(String name, Map<String, dynamic> json) => PetData(name, json['age'], json['breed']);
}

class AllPetsData {
  AllPetsData(this.petsData);

  final List<PetData> petsData;

  factory AllPetsData.fromJson(Map<String, dynamic> json) {
    List<PetData> result = [];

    var petData = json["pets_collection"];

    for (var name in petData.keys) {
      var pet = petData[name];
      result.add(PetData.fromJson(name, pet));
    }

    return AllPetsData(result);
  }
}
```

Data from a web service

- Model
  - Could generate json parser using a code generator
    - https://pub.dev/packages/json_serializable

Data from a web service

- Http client

```dart
import 'package:http/http.dart';
import 'package:pets_client/models/pets_data_model.dart';
import 'dart:convert';

class HttpServices {
  HttpServices(this.url);

  final String url;

  Client client = Client();

  Future<AllPetsData> getPetData() async {
    final response = await client.get(url);

    if (response.statusCode == 200) {
      return AllPetsData.fromJson(json.decode(response.body));
    } else {
      throw Exception("Failed to load pet data.");
    }
  }
}
```

Data from a web service

- Displaying data in a list view
- Future and AsyncSnapshot
- Populate list view using a FutureBuilder

Data from a web service

```
  @override
  Widget build(BuildContext context) {
    var url = "http://localhost:8080/pet" + (_petRetrievalTag.startsWith("Get 'em all") ? "" : "?name=" + _petRetrievalTag);
    Future<AllPetsData> httpFuture = HttpServices(url).getPetData();

    return Scaffold(
      appBar: AppBar(
        title: const Text("Pet Details"),
      ),
      body: Center(
        child: _makeFutureBuilder(httpFuture, context, (ctx, snap) => _makePetListView(ctx, snap)),
      ),
    );
  }
```

Data from a web service

```
FutureBuilder<AllPetsData> _makeFutureBuilder(Future<AllPetsData> httpFuture, BuildContext context, Function listViewMaker) {
  return FutureBuilder(
    future: httpFuture,
    builder: (BuildContext context, AsyncSnapshot snapshot) {
      switch (snapshot.connectionState) {
        case ConnectionState.none:
        case ConnectionState.waiting:
          return Center(child: CircularProgressIndicator());
        default:
          if (snapshot.hasError)
            return Center(child: Text('Error: ${snapshot.error}'));
          else
            return listViewMaker(context, snapshot);
      }
    },
  );
}
```

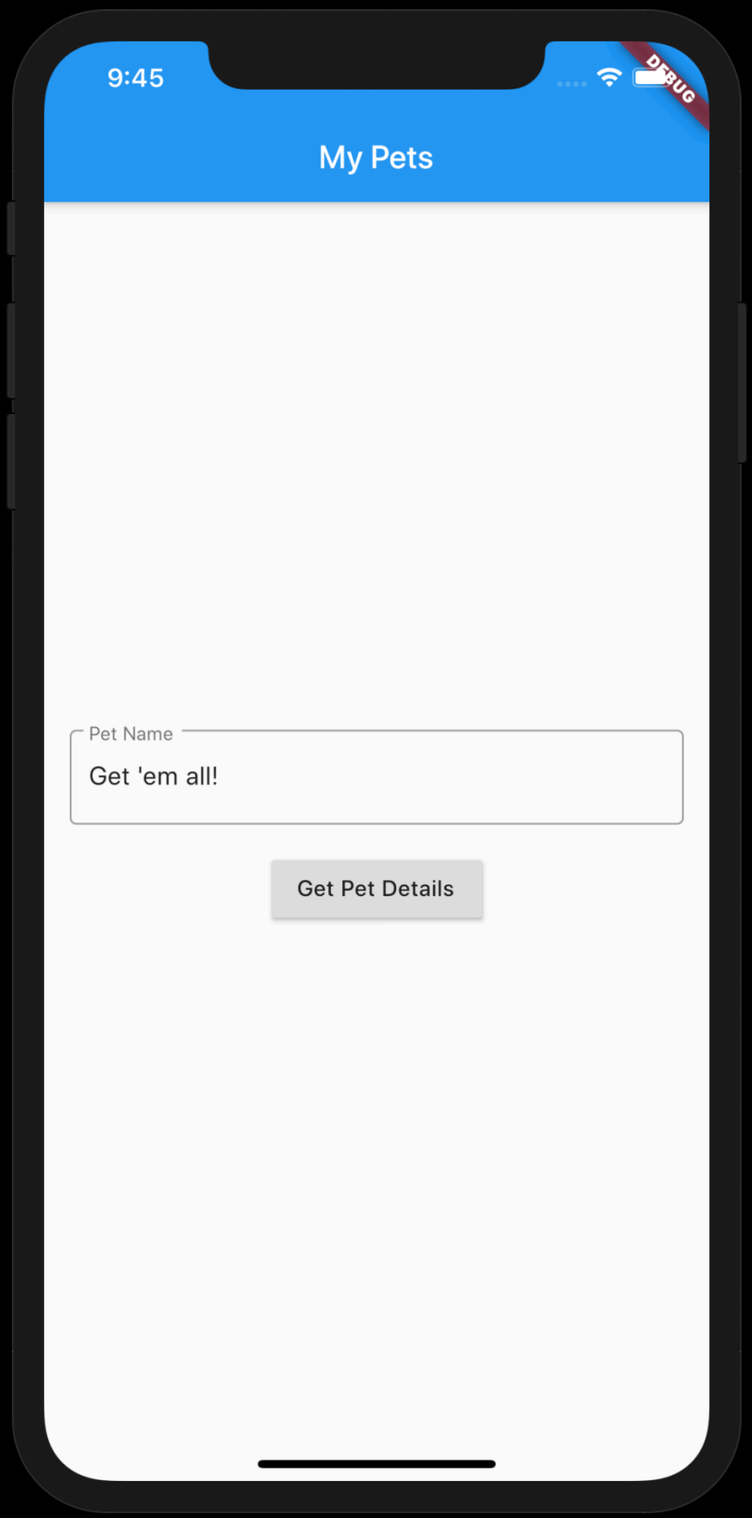
Data from a web service

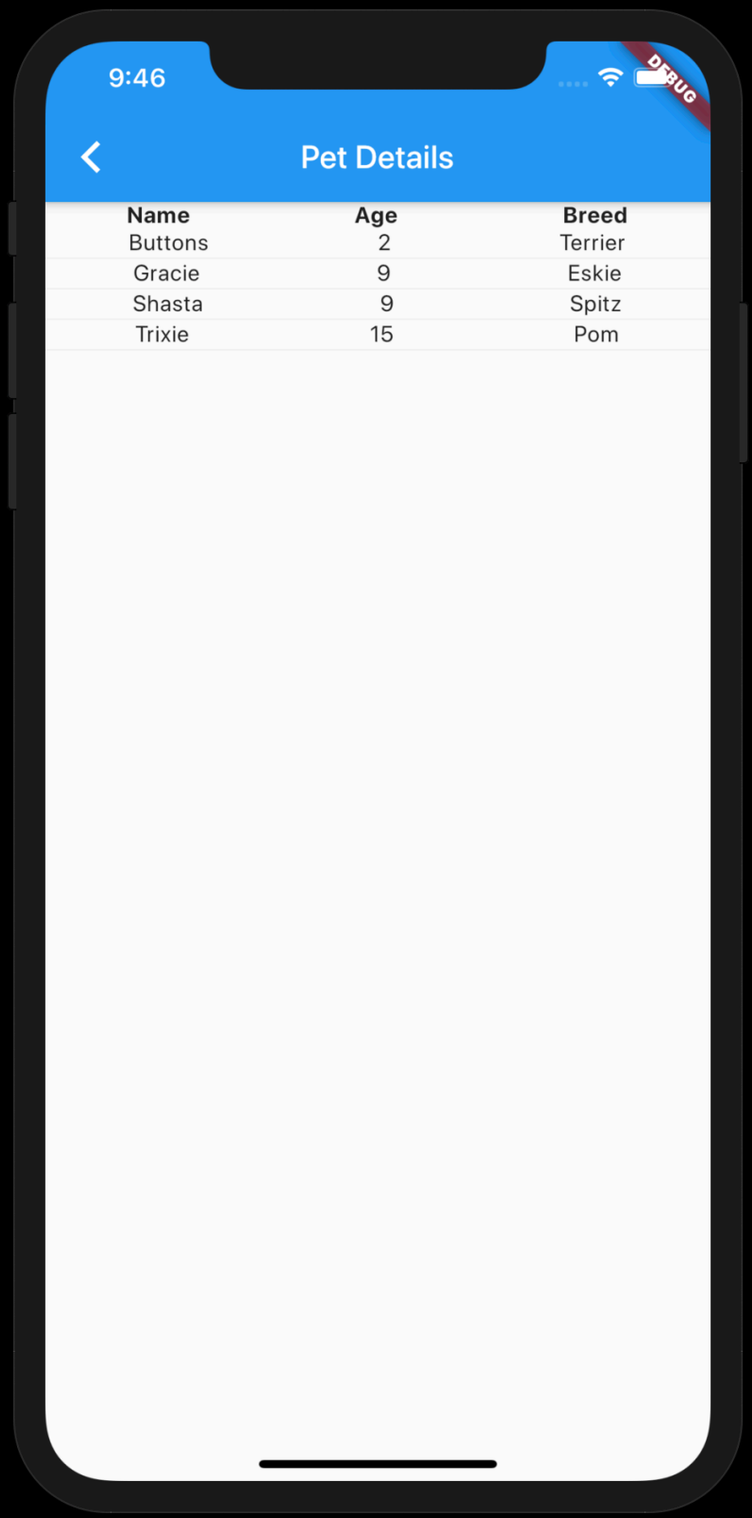
```
Widget _makePetListView(BuildContext context, AsyncSnapshot snapshot) {
   return _makeListView(
     context,
     snapshot,
     (allPetsData, index) => allPetsData.petsData[index].age.toString()
   );
  }
```

Data from a web service

```
Widget _makeListView(BuildContext context, AsyncSnapshot snapshot, Function age) {
  AllPetsData allPetsData = snapshot.data;
  return new ListView.builder(
      itemCount: allPetsData.petsData.length,

      itemBuilder: (BuildContext context, int dataIndex) {
        return Column(
          children: <Widget>[
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceAround,
              children: <Widget>[
                Text(allPetsData.petsData[dataIndex].name),
                Text(age(allPetsData, dataIndex)),
                Text(allPetsData.petsData[dataIndex].breed),
              ],
            ),
            Divider(
              height: 2.0,
            ),
          ],
        );
      },
  );
}
```

Data from a web service





Navigation

```
@override void initState() {
  _showPetDetailsButton = RaisedButton(
    child: const Text("Get Pet Details"),
    onPressed: () {
      if (_textController.text.length > 0) {
        Navigator.push(context, MaterialPageRoute(builder: (context) => PetsDisplayPage(petRetrievalTag: _textController.text,)));
      } else {
        _showPetNamTagEmptyDialog();
      }
    },
  );
```

Navigation

- Using a navigation drawer

  - https://medium.com/flutter-community/flutter-vi-navigation-drawer-flutter-1-0-3a05e09b0db9

Test framework

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

import 'package:counter/main.dart';

void main() {
  testWidgets('Counter increments smoke test', (WidgetTester tester) async {
    // Build our app and trigger a frame.
    await tester.pumpWidget(MyApp());

    // Verify that our counter starts at 0.
    expect(find.text('0'), findsOneWidget);
    expect(find.text('1'), findsNothing);

    // Tap the '+' icon and trigger a frame.
    await tester.tap(find.byIcon(Icons.add));
    await tester.pump();

    // Verify that our counter has incremented.
    expect(find.text('0'), findsNothing);
    expect(find.text('1'), findsOneWidget);
  });
}
```

Stuff I haven’t tried yet

- Native services
    - Location services, push notifications, etc.

Resources

- https://flutter.dev
- https://flutter.dev/docs/codelabs
- https://livebook.manning.com/book/flutter-in-action/about-this-book/
- https://pragprog.com/news/programming-flutter-in-beta
- https://github.com/GraciesPadre/petServer
- https://github.com/GraciesPadre/pets_client