

# Cheat Sheet: Advanced Flutter

## Calling APIs in Flutter

Type	Description	Code Example
Making Your First API Call	Learn how to perform a basic HTTP GET request	<pre>Future&lt;void&gt; fetchData() async {   var response = await   http.get(Uri.parse('https://api.example.com/data'));   print(response.body); }</pre>
Handling API Errors	Implement error handling for API requests	<pre>try {   var response = await   http.get(Uri.parse('https://api.example.com/data'));   print(response.body); } catch (e) {   print('Caught error: \$e'); }</pre>
Parsing JSON Data	Decode JSON data received from an API into Dart objects	<pre>var data = jsonDecode(response.body); print('User name: \${data['username']}');</pre>
Asynchronous Programming	Use async and await for asynchronous API calls	<pre>Future&lt;void&gt; getUserData() async {   var response = await   http.get(Uri.parse('https://api.example.com/users/1'));   var data = jsonDecode(response.body);   print('User name: \${data['name']}'); }</pre>
Post Data to API	Send data to an API using the POST method	<pre>Future&lt;void&gt; postData() async {   var response = await   http.post(Uri.parse('https://api.example.com/add'), body: {'name': 'John Doe'});   print('Response status: \${response.statusCode}'); }</pre>

Fetch Data with Headers	Make HTTP requests with additional headers	<pre>Future&lt;void&gt; fetchWithHeaders() async {   var response = await http.get(     Uri.parse('https://api.example.com/data'),     headers: {'Authorization': 'Bearer your_token_here'}   );   print('Data with headers: \${response.body}'); }</pre>
-------------------------	--	---

Introduction to mobile native features in Flutter

Type	Description	Code Example
Accessing the Camera	Utilize the native camera functionality via the Flutter plugin	<pre>Future&lt;void&gt; takePicture() async {   final cameras = await availableCameras();   final firstCamera = cameras.first;   final CameraController controller = CameraController(firstCamera, ResolutionPreset.high);   await controller.initialize();   final image = await controller.takePicture();   controller.dispose(); }</pre>
Using GPS	Fetch the current location using the geolocation plugin	<pre>Position position = await Geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.high); print('Current location: \${position.latitude}, \${position.longitude}');</pre>
Local Storage	Store data locally using the shared_preferences plugin	<pre>SharedPreferences prefs = await SharedPreferences.getInstance(); await prefs.setString('username', 'exampleUser');</pre>
Accessing Device Sensors	Interact with native device sensors like the accelerometer	<pre>StreamSubscription&lt;dynamic&gt; subscription = AccelerometerEvents.listen((AccelerometerEvent event) {   print(event); });</pre>
Managing Notifications	Schedule and manage notifications locally on the device	<pre>FlutterLocalNotificationsPlugin notificationsPlugin = FlutterLocalNotificationsPlugin(); var androidDetails = AndroidNotificationDetails('channelId', 'channelName', 'channelDescription'); var generalNotificationDetails = NotificationDetails(android: androidDetails); await notificationsPlugin.show(0, 'Test Title', 'Test Body', generalNotificationDetails);</pre>

--	--	--

Managing state in Flutter

Type	Description	Code Example
Manage local widget state changes in a StatefulWidget	Provides a way to manage and update the local state of a widget	<pre>setState() {   _counter++; };</pre>
InheritedWidget	Provides a way to pass data down the widget tree and allows descendants to rebuild when the data changes	<pre>MyInheritedWidget(   data: counter,   child: ChildWidget() );</pre>
GlobalKey and FormState	Utilize GlobalKey to access the state from anywhere in the app, commonly used with forms	<pre>final _formKey = GlobalKey&lt;FormState&gt;(); Form(   key: _formKey,   child: Column(     children: [       TextFormField(validator: (value) =&gt; value.isEmpty ? 'Cannot be empty' : null),       ElevatedButton(         onPressed: () {           if (_formKey.currentState.validate()) {             // Process data           }         },         child: Text('Submit')       )     ],   ) );</pre>
Provider for state management	A wrapper around InheritedWidget to make state management easier and more efficient	<pre>ChangeNotifierProvider(   create: (context) =&gt; DataModel(),   child: Consumer&lt;DataModel&gt;(     builder: (context, dataModel, child) {       return Text('\${dataModel.value}');     },   ), );</pre>

Flutter Persistence with Local Storage

Type	Description	Code Example
SharedPreferences	Ideal for simple data such as user preferences and settings	<pre>final prefs = await SharedPreferences.getInstance(); prefs.setInt('counter', 10); int counter = prefs.getInt('counter') ?? 0;</pre>
SQLite	Ideal for structured data and complex queries. It works like a traditional database	
Files	Read/write files for documents or media	
Third-Party Packages	<b>Hive:</b> A fast, NoSQL database for Flutter. <b>LocalStorage:</b> Similar to web local storage, ideal for storing JSON data.	
Initialize Local Storage	Set up the chosen storage method at the start of your app.	<pre>final prefs = await SharedPreferences.getInstance();</pre>
CRUD Operations	Implement Create, Read, Update, and Delete functions.	<pre>// Saving data prefs.setInt('counter', 10); // Retrieving data int counter = prefs.getInt('counter') ?? 0; // Deleting data prefs.remove('counter');</pre>
Serialization/Deserialization	Encode and decode data using JSON for complex structures	<pre>import 'dart:convert'; Map userMap = jsonDecode(jsonString); var user = User.fromJson(userMap);</pre>

<b>Integrating with State Management</b>	Use state management tools (e.g., Provider, Riverpod) to handle local storage updates dynamically in the UI	<pre>class UserProvider with ChangeNotifier {   User _user;   User get user =&gt; _user;   void loadUser() {     String userJson = storage.getItem('user');     _user = User.fromJson(jsonDecode(userJson));     notifyListeners();   } }</pre>
--	---	---

Plugins in Flutter

Type	Description	Code Example
<b>Using the url_launcher Plugin</b>	Open a URL in the mobile browser from a Flutter app	<pre>if (await canLaunch(url)) {   await launch(url); } else {   throw 'Could not launch \$url'; }</pre>
<b>Managing Plugin Compatibility</b>	Ensure that plugins are compatible with the version of Flutter you are using	<pre>// Typically handled in your pubspec.yaml by specifying compatible versions dependencies:   flutter:     sdk: flutter   url_launcher: ^6.0.3</pre>
<b>Camera Plugin</b>	Access the device camera to capture photos and videos	<pre>final cameras = await availableCameras(); final firstCamera = cameras.first; final cameraController = CameraController(firstCamera, ResolutionPreset.medium); await cameraController.initialize();</pre>
<b>Audio Players Plugin</b>	Play audio files and streams in Flutter with the Audio Players plugin	<pre>final player = AudioPlayer(); await player.play(UrlSource('https://example.com/audio.mp3'));</pre>
<b>Flutter Maps Plugin</b>	Integrate maps into your Flutter applications using the maps plugin	<pre>GoogleMap(   onMapCreated: _onMapCreated,   initialCameraPosition: CameraPosition(     target: LatLng(0.0, 0.0),     zoom: 10.0,</pre>

),  
);



**Skills** Network