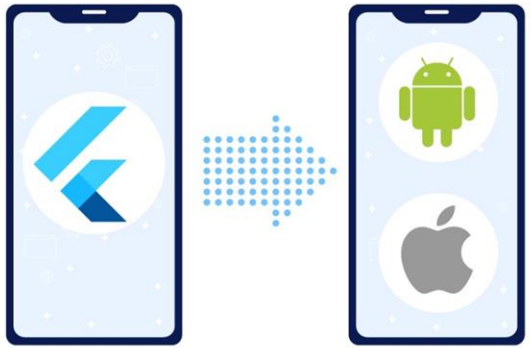
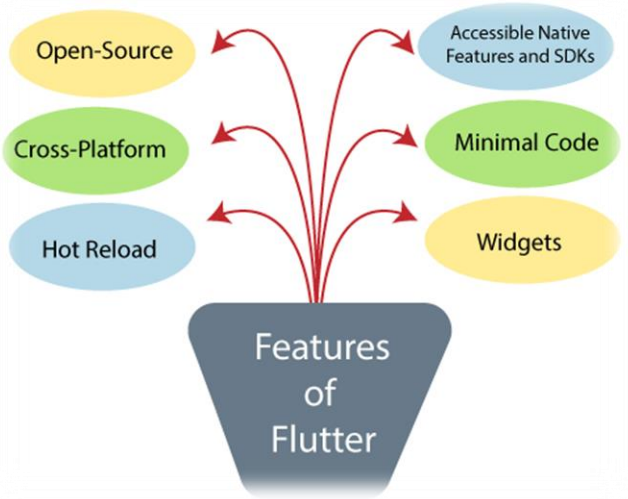
Flutter

By Rabbil Hasan

Flutter Features





Flutter Installation

Install

Get started > Install

Select the operating system on which you are installing Flutter:

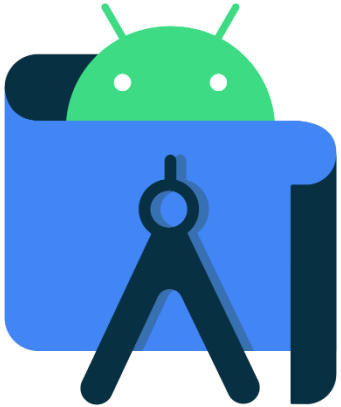
 Windows

 macOS

 Linux

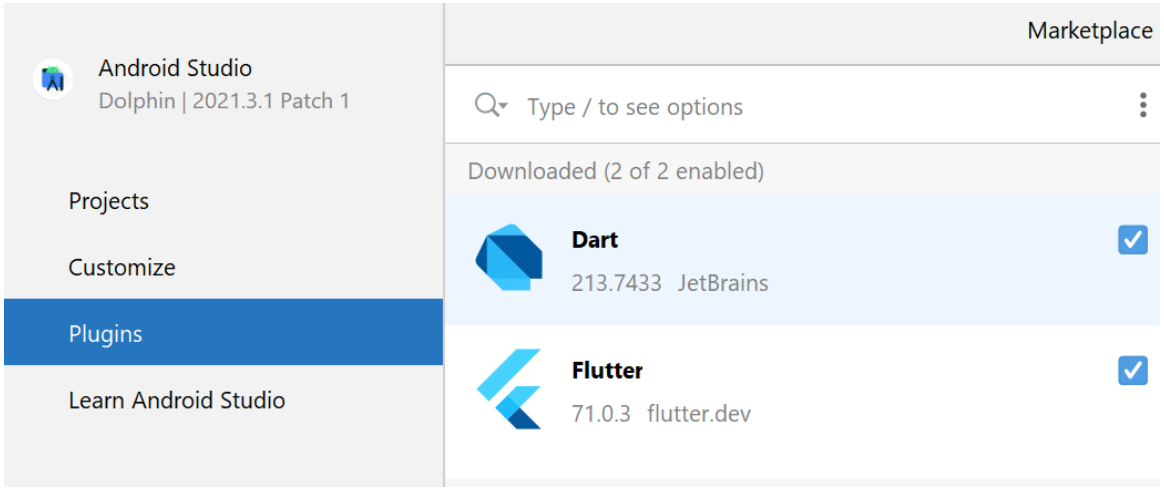
 Chrome OS

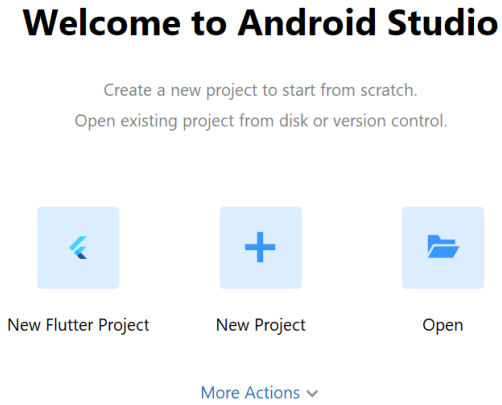
Install Android Studio



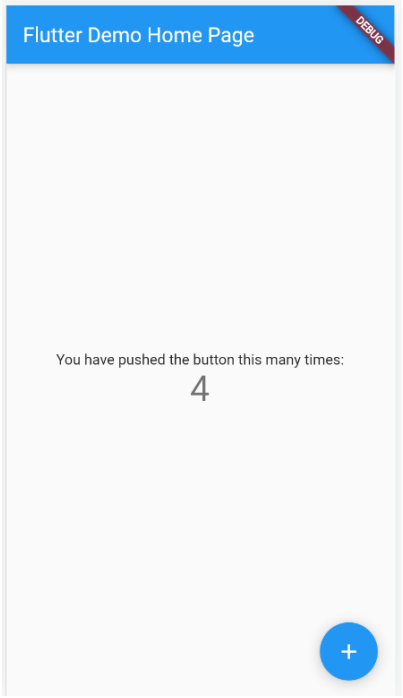
Get the official Integrated Development Environment (IDE) for Android app development.

Install Flutter Extensions





Create & Run Your First Flutter App



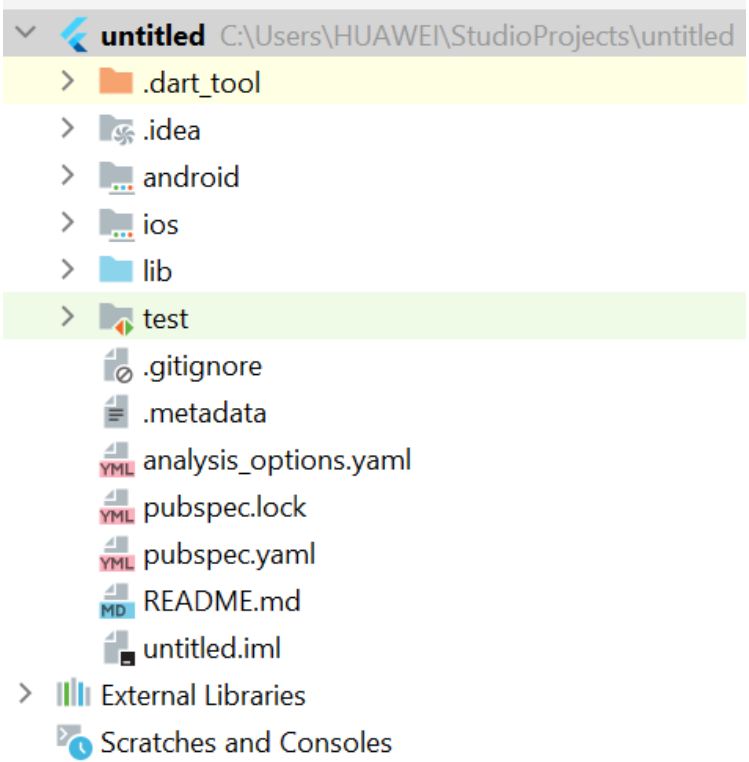
Flutter Project Structure

.idea:

- This folder is at the very top of the project structure.
- Holds the configuration for Android Studio.
- It doesn't matter because we are not going to work with.
- So that the content of this folder can be ignored.

.android:

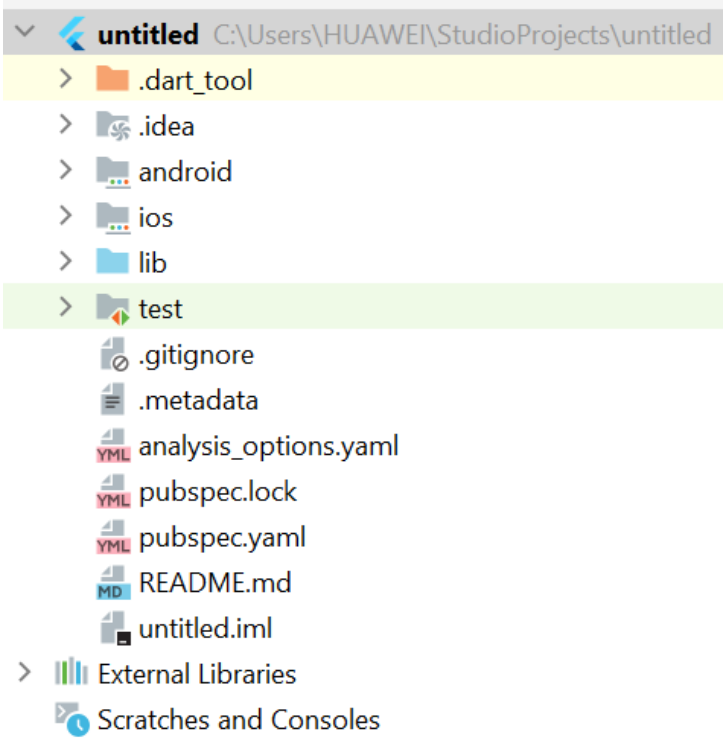
- This folder holds a complete Android project
- Used when you build the Flutter application for Android
- When the Flutter code is compiled into the native code
- It will get injected into this Android project
- That the result is a native Android application



Flutter Project Structure

.ios:

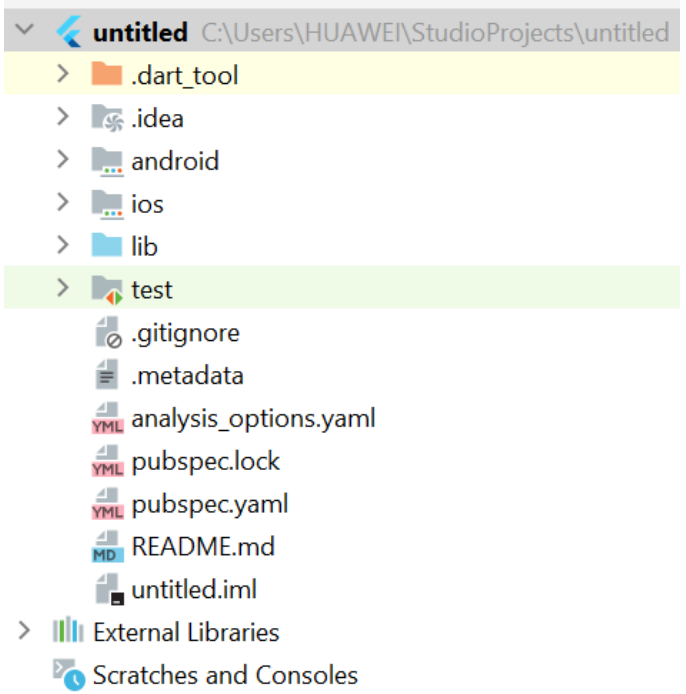
- This folder holds a complete Mac project
- used when you build the Flutter application for iOS
- When the Flutter code is compiled into the native code
- It will get injected into this iOS project
- that the result is a native iOS application
- Building a Flutter application for iOS is only possible when you are working on macOS



Flutter Project Structure

lib:

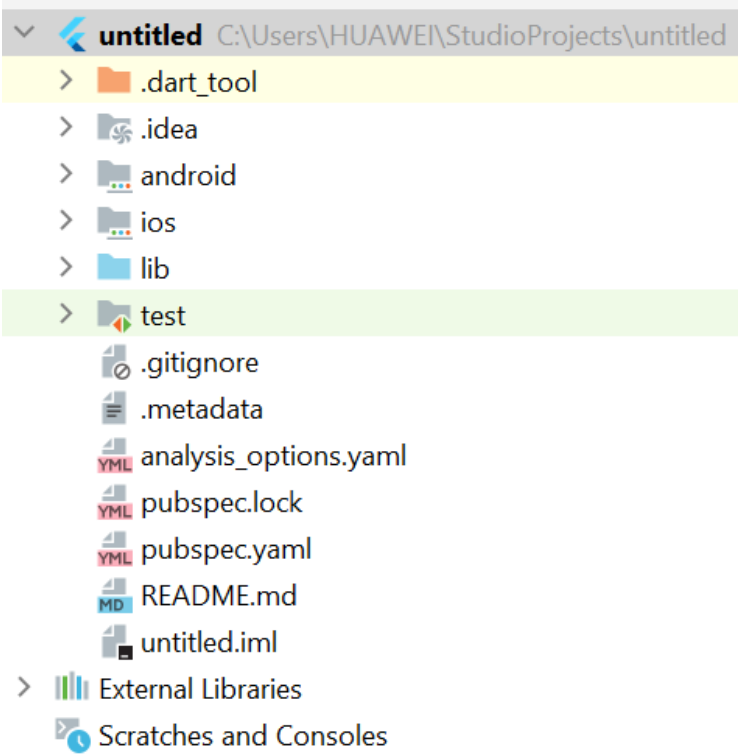
- It is an essential folder, which stands for the library
- It is a folder where we will do our 99 percent of project work
- Inside the lib folder, we will find the Dart files which contain the code of our Flutter application
- By default, this folder contains the file main.dart, which is the entry file of the Flutter application.



Flutter Project Structure

test:

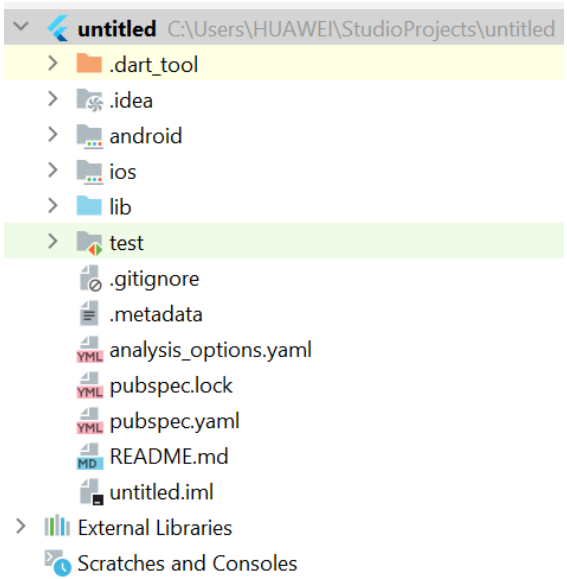
- This folder contains a Dart code,
- Which is written for the Flutter application.
- Perform the automated test when building the app.
- It won't be too important for us here.



Flutter Project Structure

.gitignore:

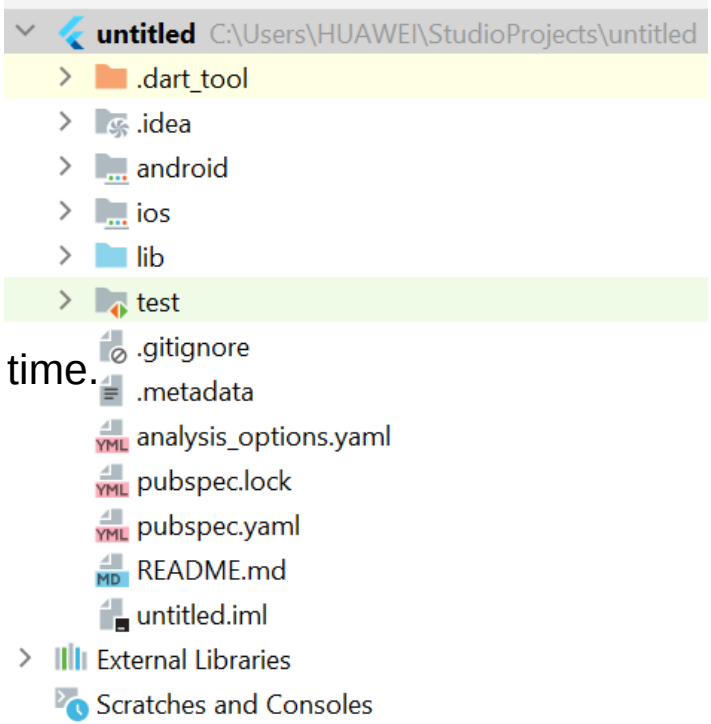
- It is a text file containing a list of files, file extensions, and folders
- that tells Git which files should be ignored in a project.
- Git is a version-control file for tracking changes in source



Flutter Project Structure

.metadata:

- It is an auto-generated file by the flutter tools.
- Used to track the properties of the Flutter project.
- This file performs the internal tasks.
- So you do not need to edit the content manually at any time.

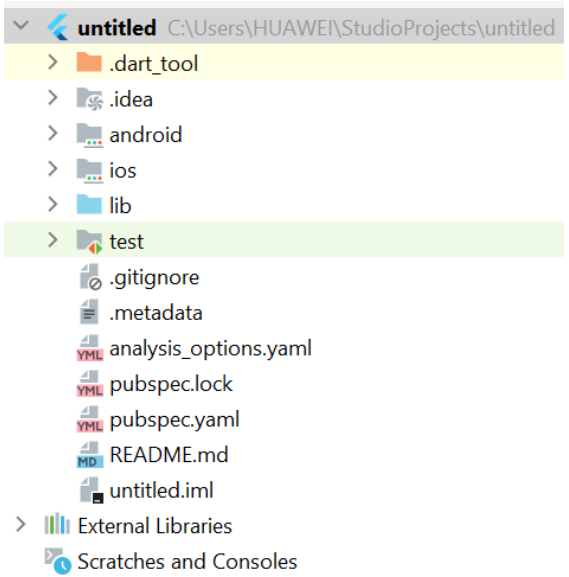


Flutter Project Structure

pubspec.yaml:

It is the project's configuration file that will use a lot during working with the Flutter project. It allows you how your application works. This file contains-

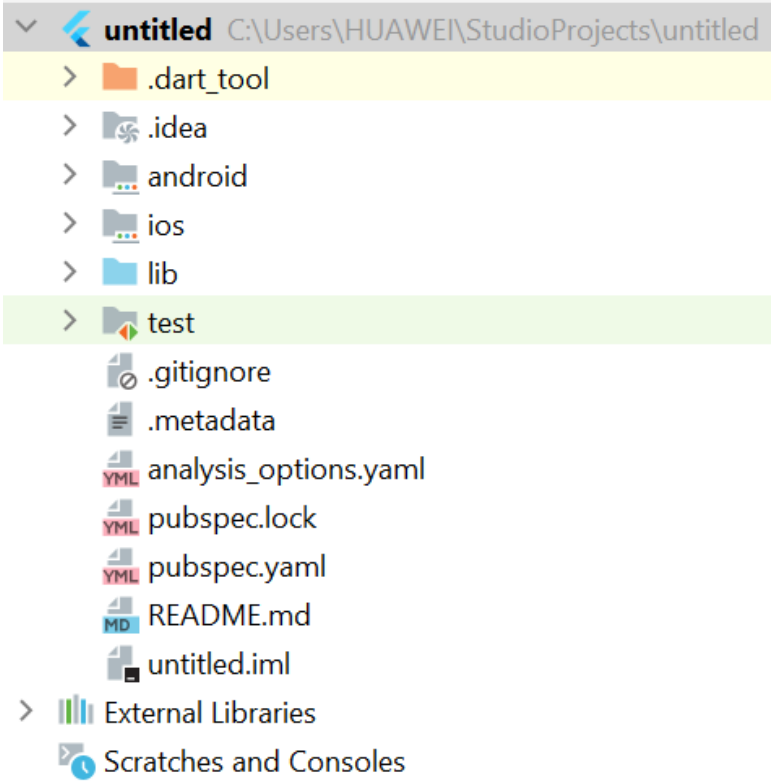
- Project general settings such as name
- Description, and version of the project.
- Project dependencies.
- Project assets (e.g., images).



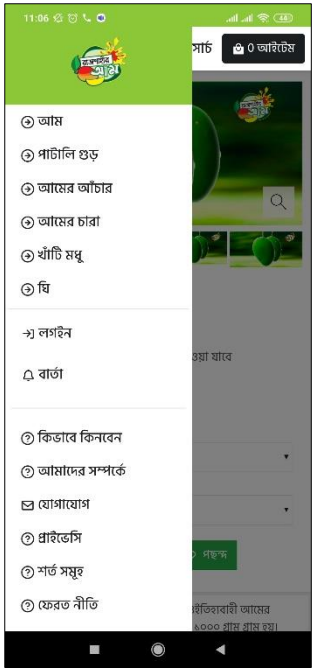
Flutter Project Structure

pubspec.lock:

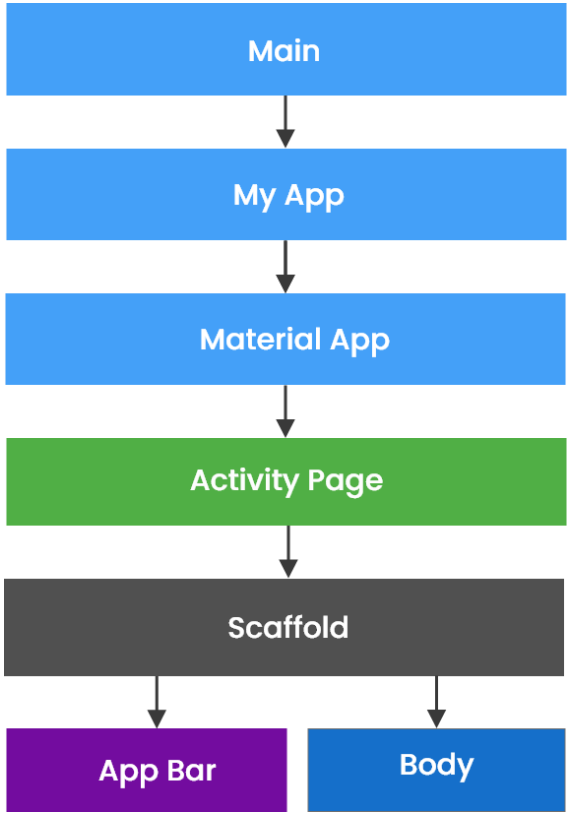
- It is an auto-generated file based on the .yaml file.
- It holds more detail setup about all dependencies.



Activity & widgets concepts



Flutter Main Source Code Flow




```dart
void main(){
  runApp(const MyApp());
}

class MyApp extends StatelessWidget{
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return const MaterialApp(home: HomeActivity());
  }
}

class HomeActivity extends StatelessWidget{
  const HomeActivity({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Hello')),
      body: const Text('Hello World') ,
    );  // Scaffold
  }
}
```

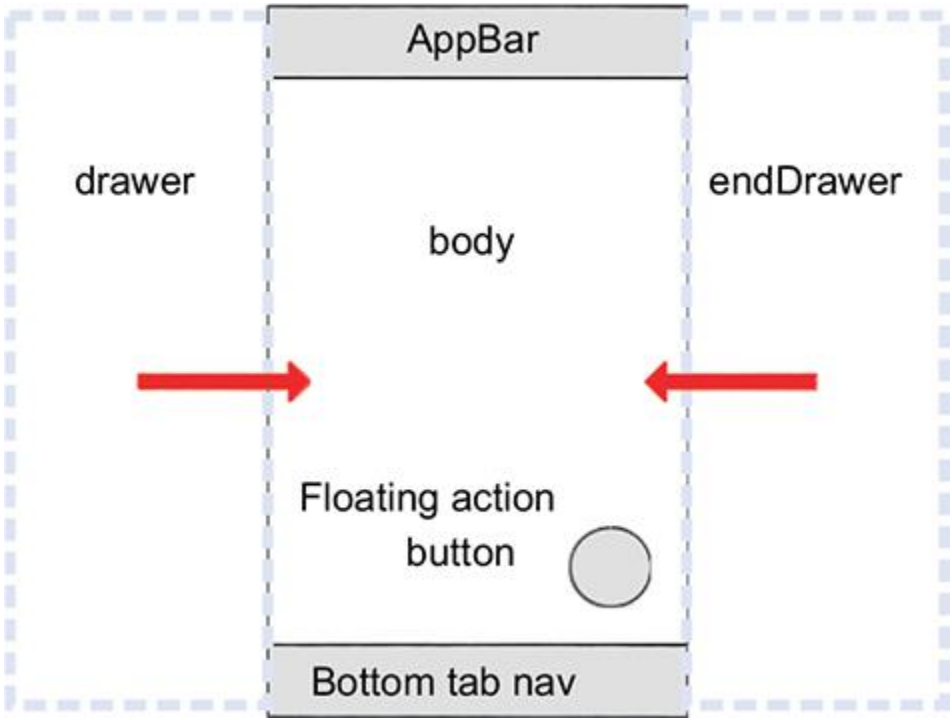
Material App Class

MaterialApp is a predefined class in a flutter. Main or core component of flutter.

- **color:** It controls the primary color used in the application.
- **darkTheme:** It provided theme data for the dark theme for the application.
- **debugShowCheckedModeBanner:** This property takes in a boolean as the object to decide whether to show the debug banner or not.
- **home:** This property takes in widget as the object to show on the default route of the app.
- **title:** The title property takes in a string as the object to decide the one-line description of the app for the device.

Scaffold Class

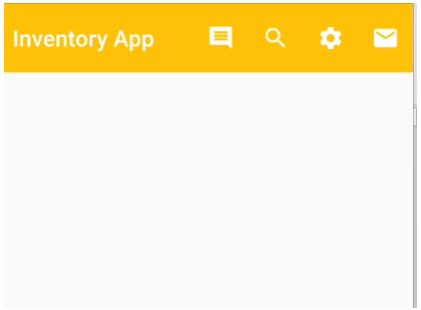
Scaffold will expand or occupy the whole device screen.



Flutter AppBar Widget

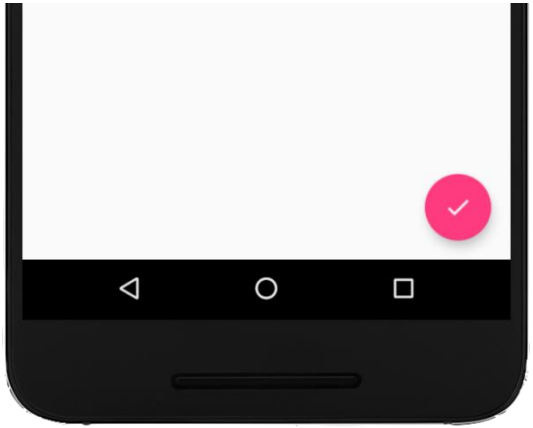
AppBar is usually the topmost component of the app . it contains the toolbar and some other common action buttons.

- **actions:** This property takes in a list of widgets as a parameter to be displayed after the title if the **AppBar** is a row.
- **title:** This property usually takes in the main widget as a parameter to be displayed in the **AppBar**.
- **backgroundColor:** This property is used to add colors to the background of the **Appbar**.
- **elevation:** This property is used to set the z-coordinate at which to place this app bar relative to its parent.
- **shape:** This property is used to give shape to the **Appbar** and manage its shadow.



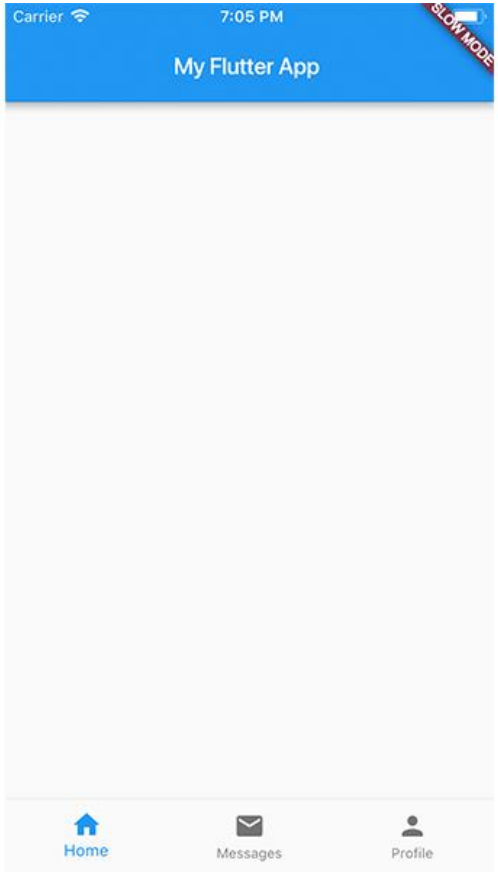
Flutter Floating Action Button

- Floating Action Button is a button that is placed at the right bottom corner by default.
- Floating Action Button is an icon button that floats over the content of the screen at a fixed place.

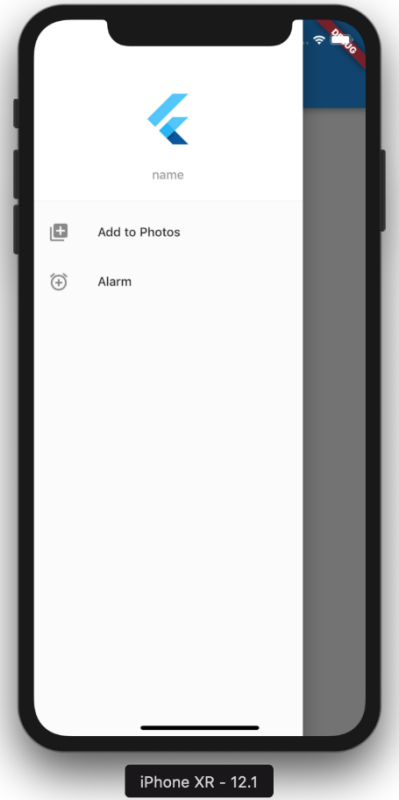


Flutter Bottom Tab

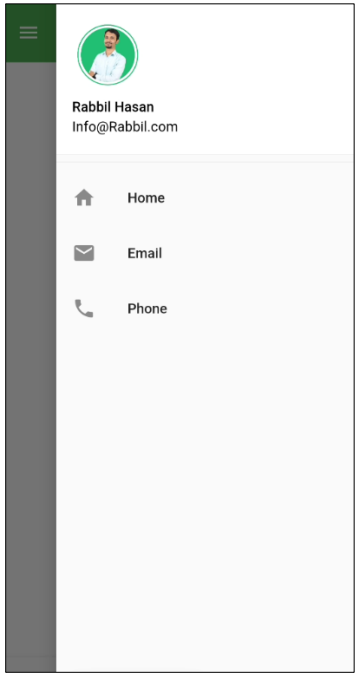
- Menu at the bottom of the Scaffold.
- We have seen this Navigation bar in most of the applications.
- We can add multiple icons or texts or both in the bar as items.



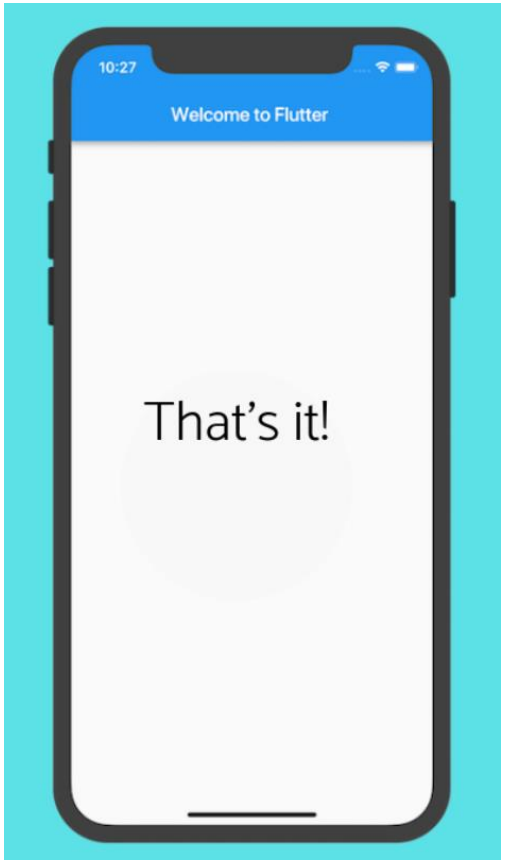
Navigation drawer flutter



Navigation end drawer flutter

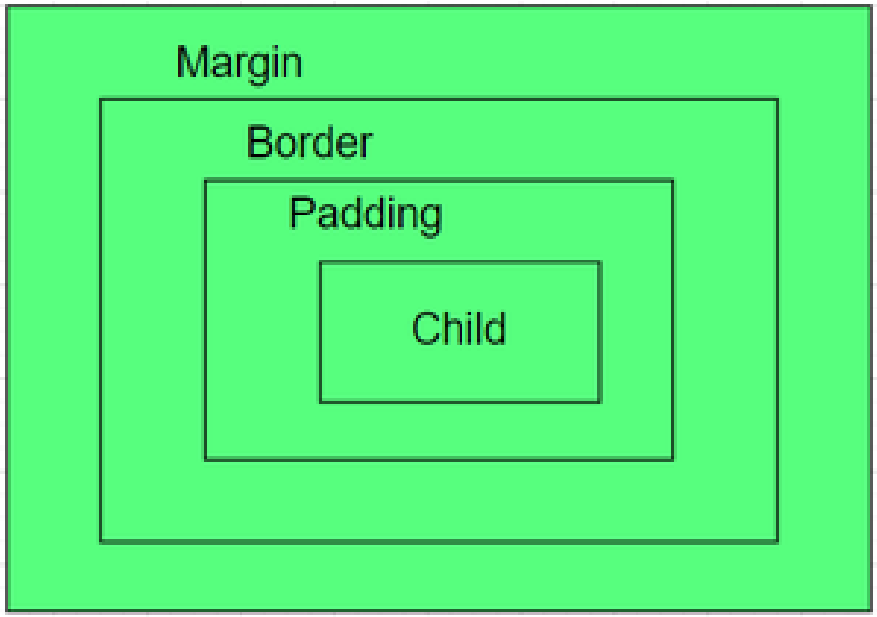


Flutter Body

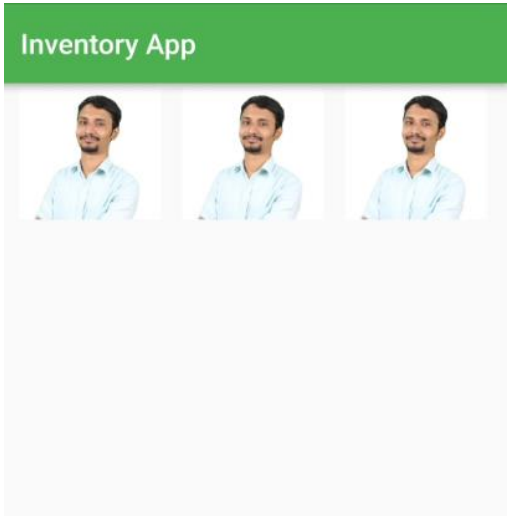


Container in Flutter

Container class in flutter is a convenience widget that combines common painting, positioning, and sizing of widgets.

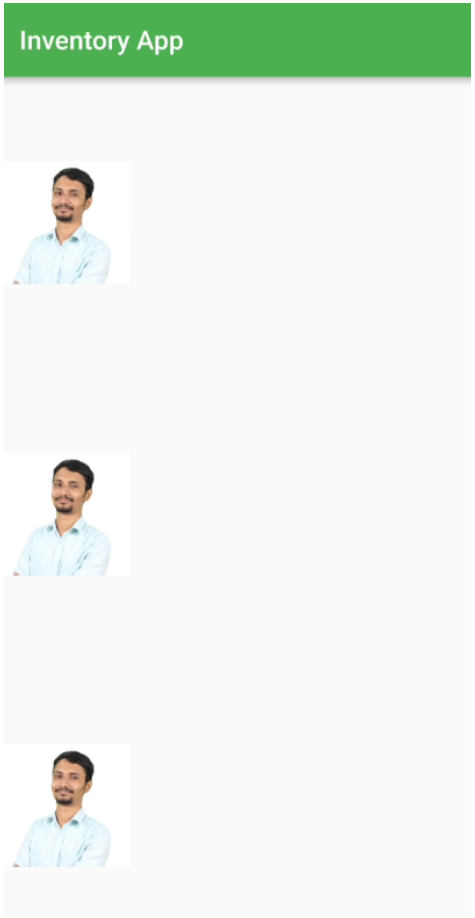


Flutter Row With Container



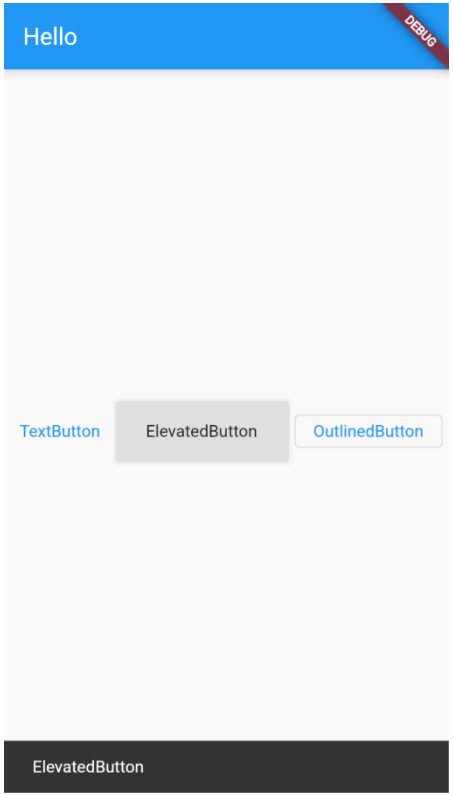

```
body: Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Container(height: 100, width: 100, child: Image.network("https://cdn.rabbil.com/photos/images/2022/11/05/rabbil.jpg")),
    Container(height: 100, width: 100, child: Image.network("https://cdn.rabbil.com/photos/images/2022/11/05/rabbil.jpg")),
    Container(height: 100, width: 100, child: Image.network("https://cdn.rabbil.com/photos/images/2022/11/05/rabbil.jpg"))
  ],
),  // Row
```

Flutter Column With Container



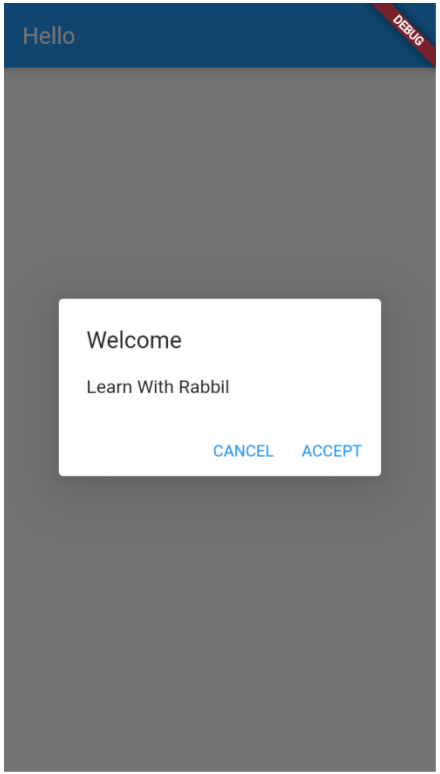

```
body: Column(
  mainAxisAlignment: MainAxisAlignment.spaceAround,
  children: [
    Container(height: 100,width: 100,child: Image.network("https://cdn.rabbil.com/photos/images/2022/11/05/rabbil.jpg")),
    Container(height: 100,width: 100,child: Image.network("https://cdn.rabbil.com/photos/images/2022/11/05/rabbil.jpg")),
    Container(height: 100,width: 100,child: Image.network("https://cdn.rabbil.com/photos/images/2022/11/05/rabbil.jpg"))
  ],
),  // Column
```

Flutter Button

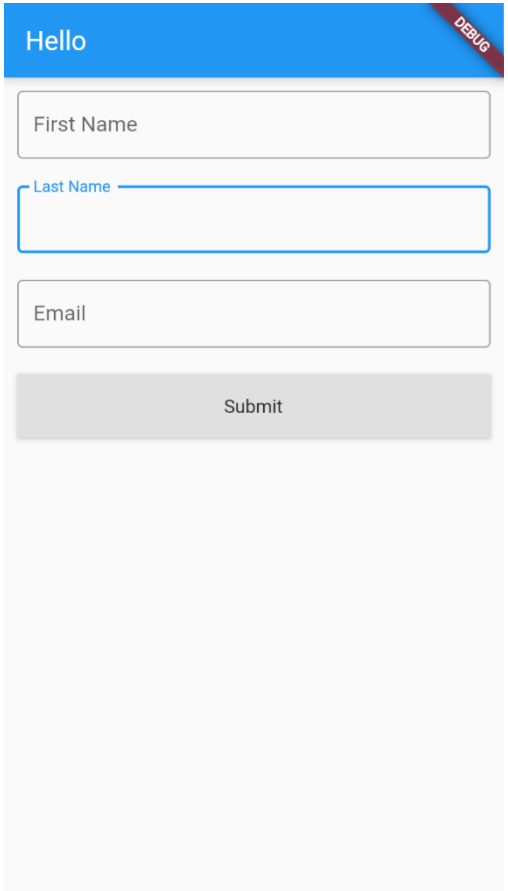


- Elevated Button
- Text Button
- Outline Button

Alert Dialog



Simple Form



List View Builder From Array

Step 01: JSON Array

Step 02: List View Builder

Step 03: Gesture Detector

Step 04: List Item

Step 05: List Item On Tap/On Press

Grid View Builder From Array

Step 01: JSON Array

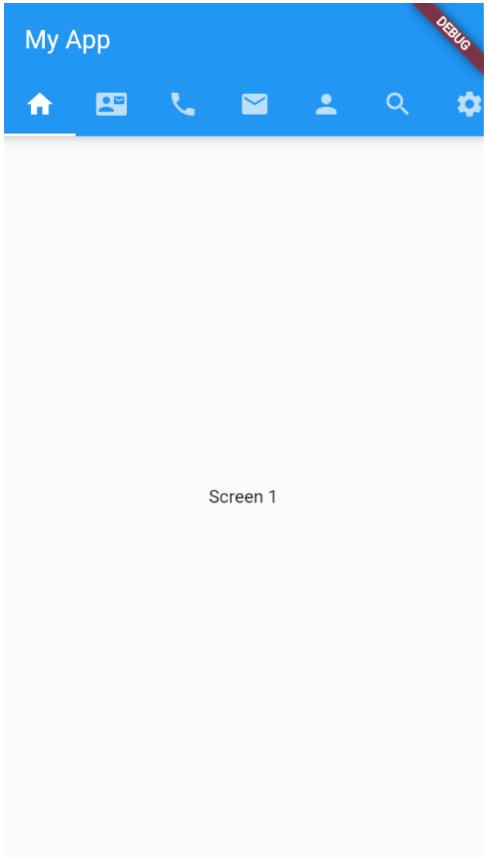
Step 02: Grid View Builder

Step 03: Gesture Detector

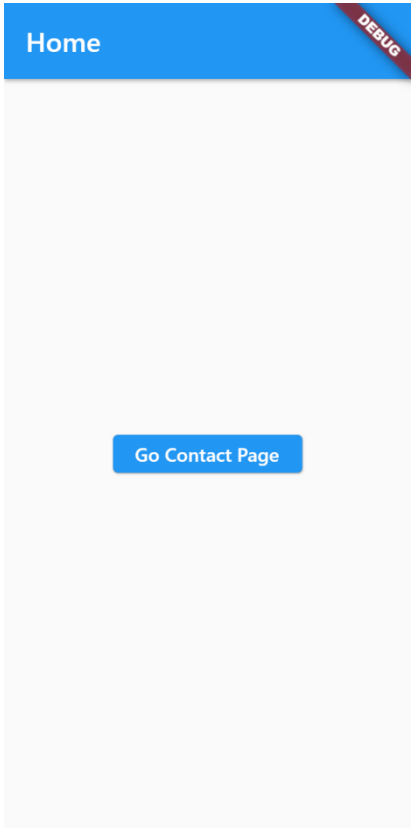
Step 04: Grid Item

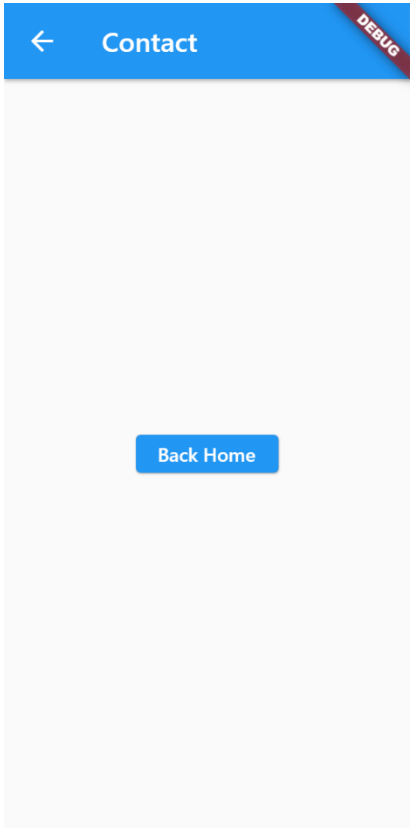
Step 05: Grid Item On Tap/On Press

Tab Bar / Activity/ Fragment

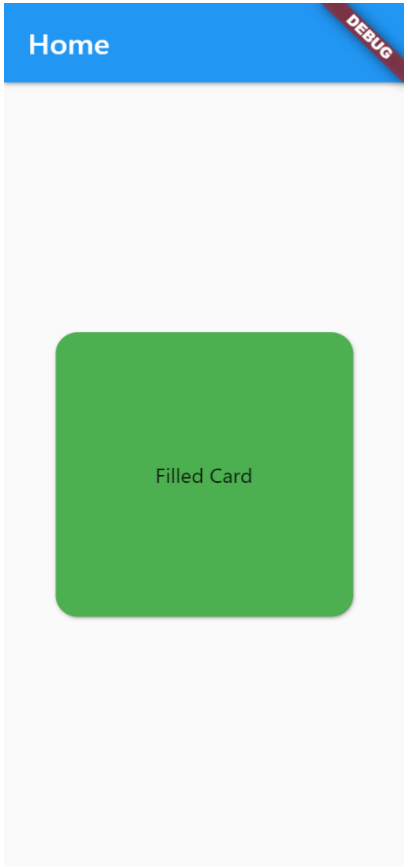


Navigation

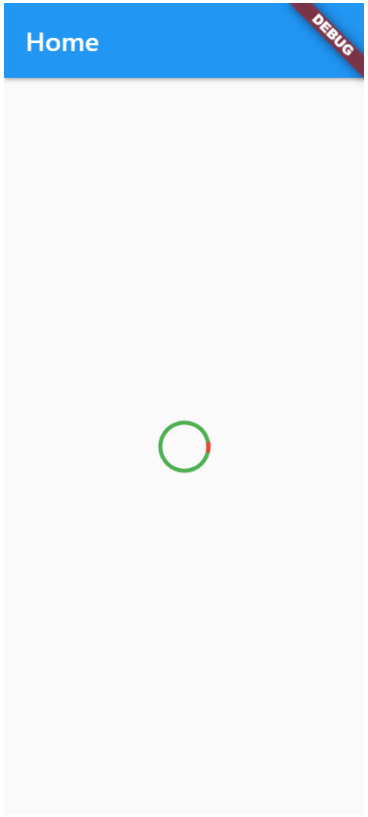




Working With Card Shape

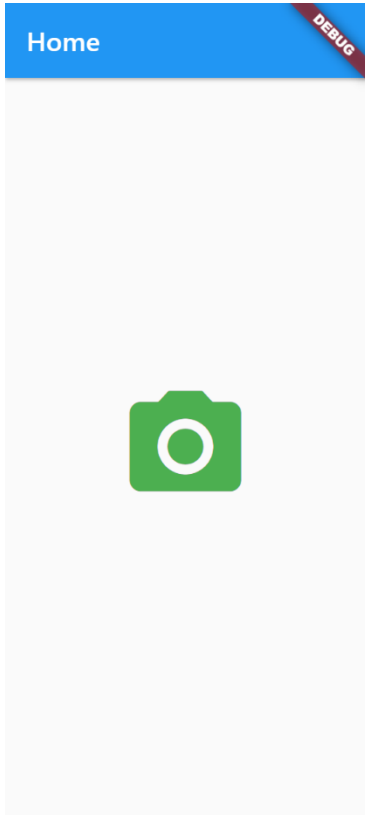


Working With Circular progress



Working With Linear progress

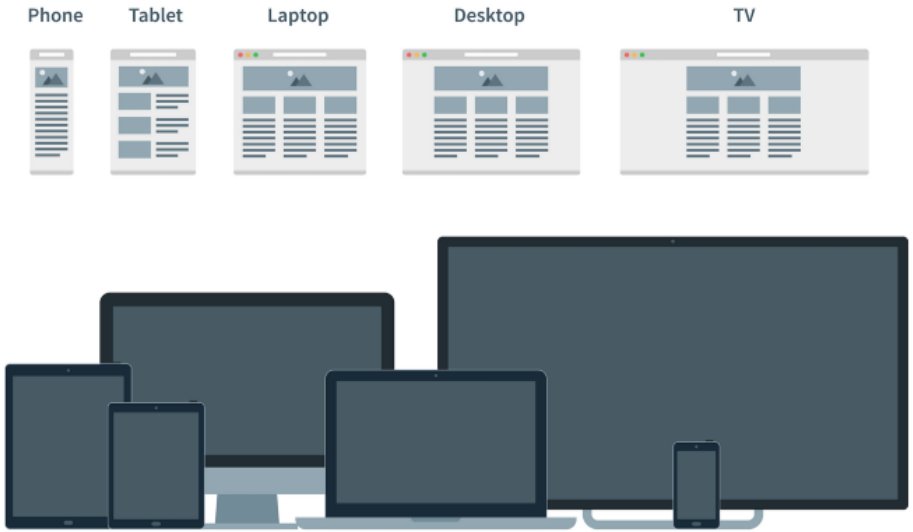
Icons Class In Flutter



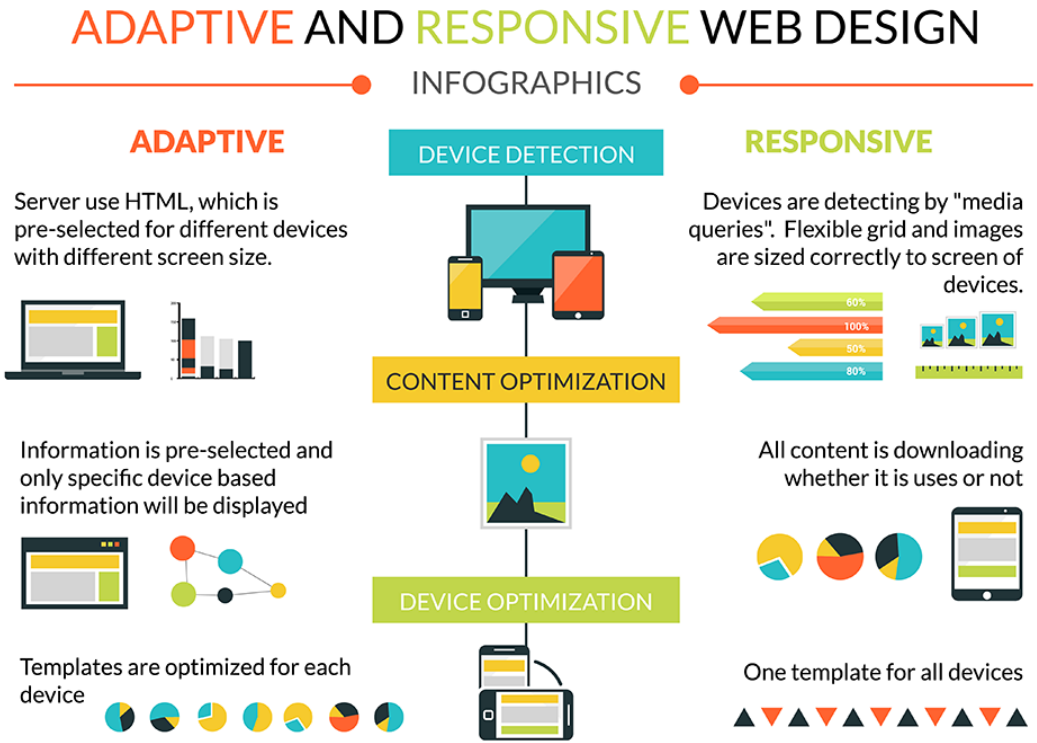
Responsive & Adaptive

Why?

- It is impossible to keep in mind all devices, client may use for the application using.
- Adaptive design and responsive design were created to avoid the classification.
- Actually, AD (adaptive design) and RD (responsive design) solve the same tasks but in different ways.



Responsive vs Adaptive



Responsive & Adaptive For Flutter Application

Why?

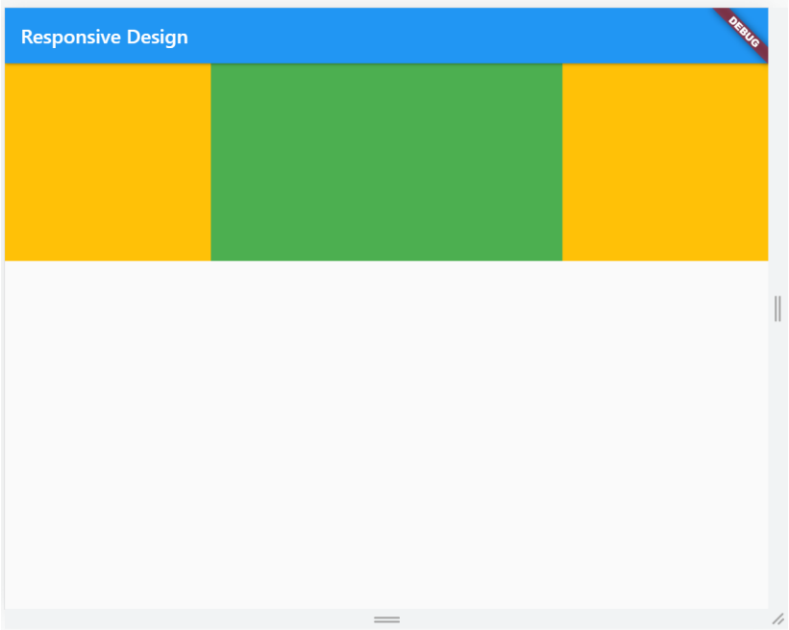
- To take the maximum advantages from Flutter
- To enable your application in different size device with web & desktop too.

Widgets & Packages:

- **Fractionally Sized Box**
- **Media Query**
- **Expanded**
- **Aspect Ratio**
- **Fitted Box**
- **Layout Builder**
- Responsive grid
- **Column Row**
- Responsive Framework
- **Scroll View**

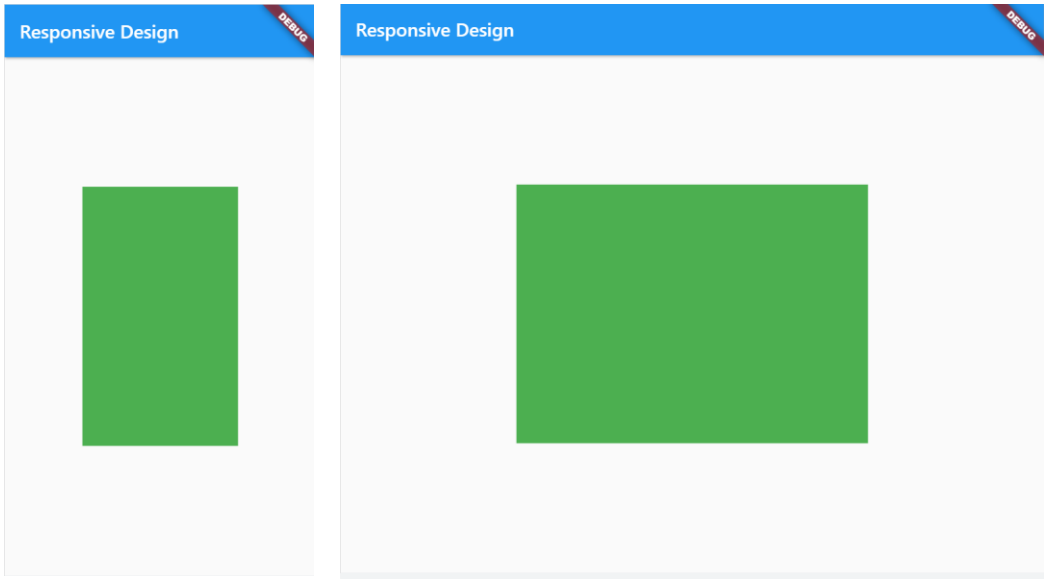
Aspect Ratio

- A widget that attempts to size the child to a specific aspect ratio
- The widget first tries the largest width permitted by the layout constraints.
- The height of the widget is determined by applying the given aspect ratio to the width



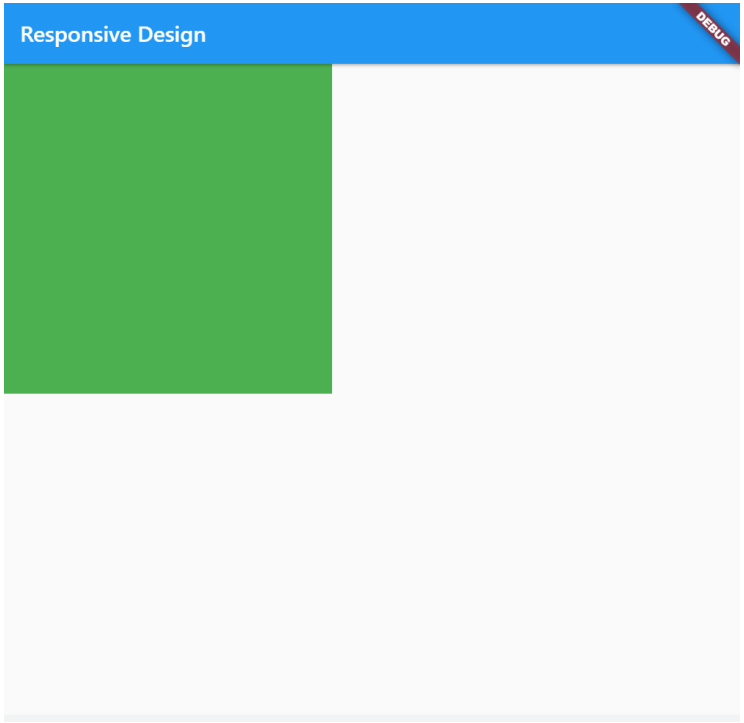
Fractionally Sized Box

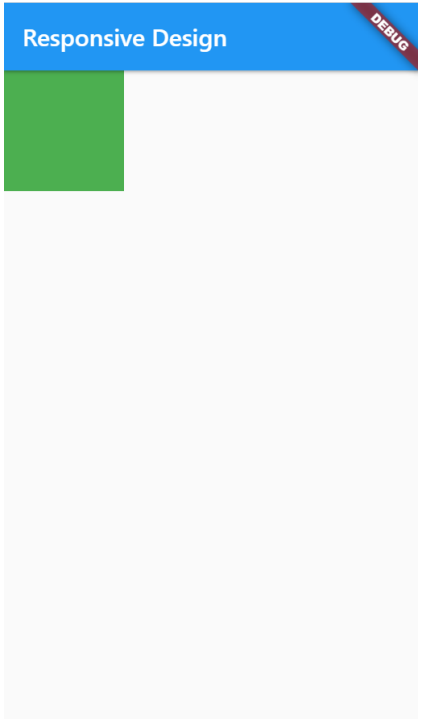
- A widget that sizes its child to a fraction of the total available space.



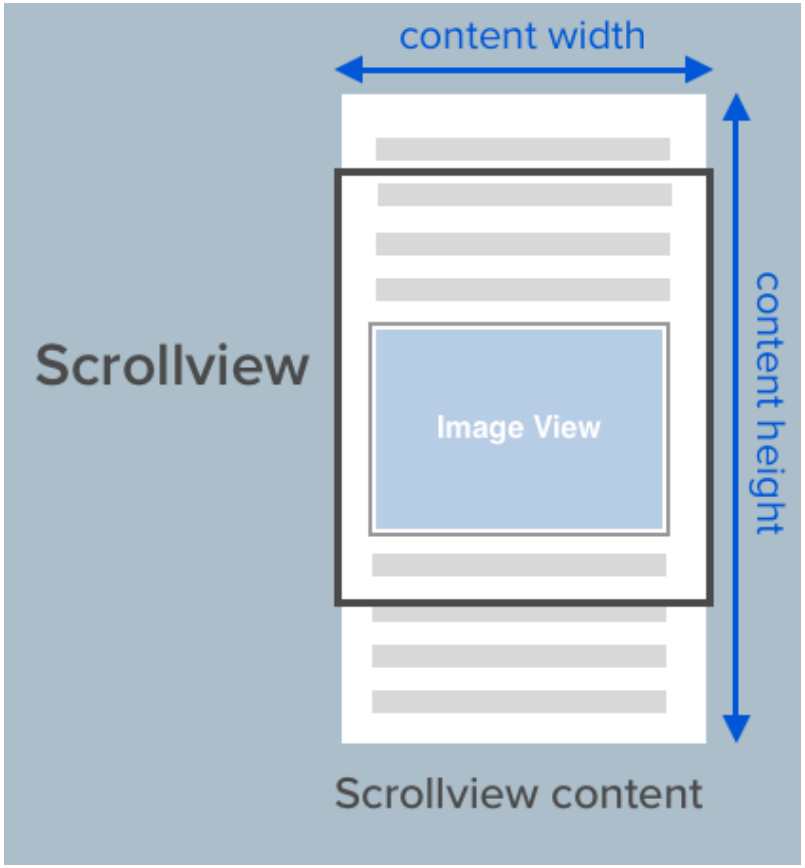
Layout Builder

- Builds a widget tree that can depend on the parent widget's size

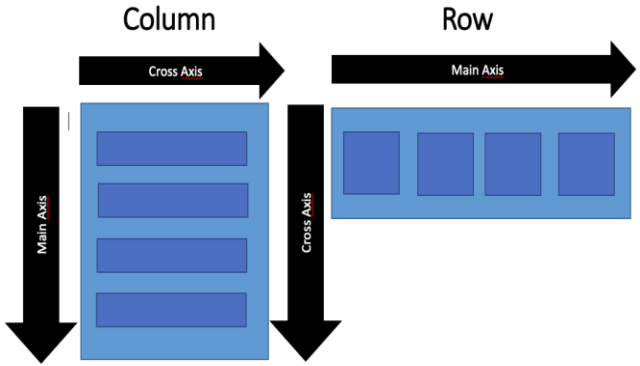


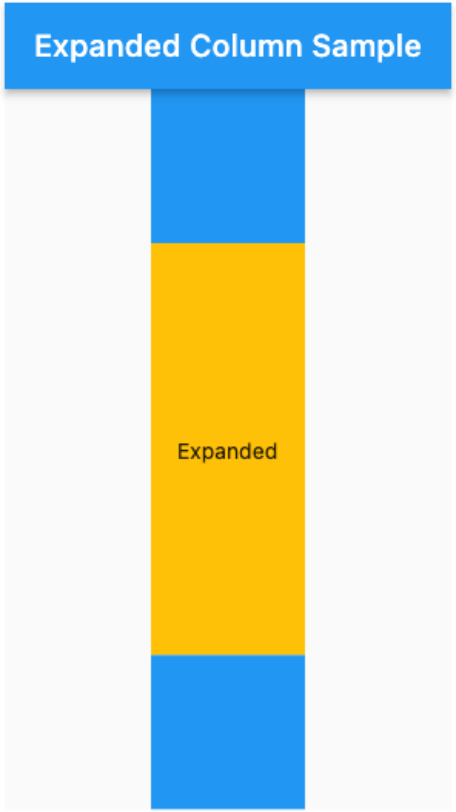


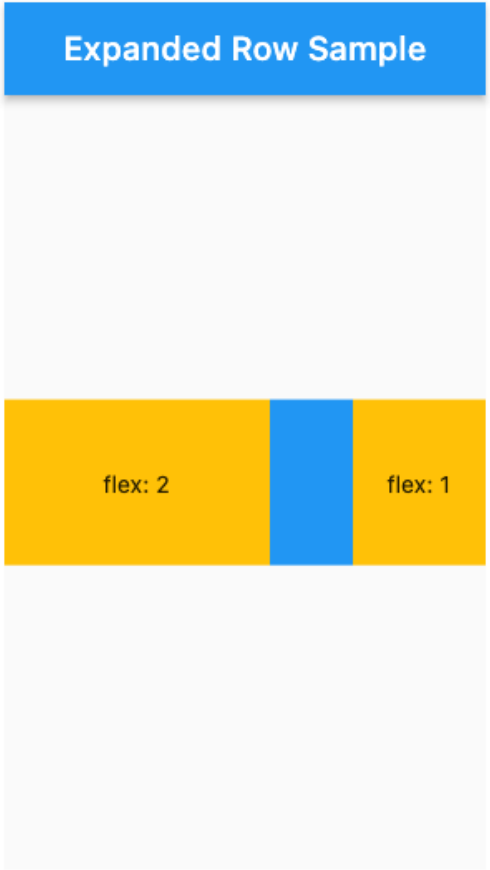
Scroll View


content width
Scrollview
Image View
content height
Scrollview content

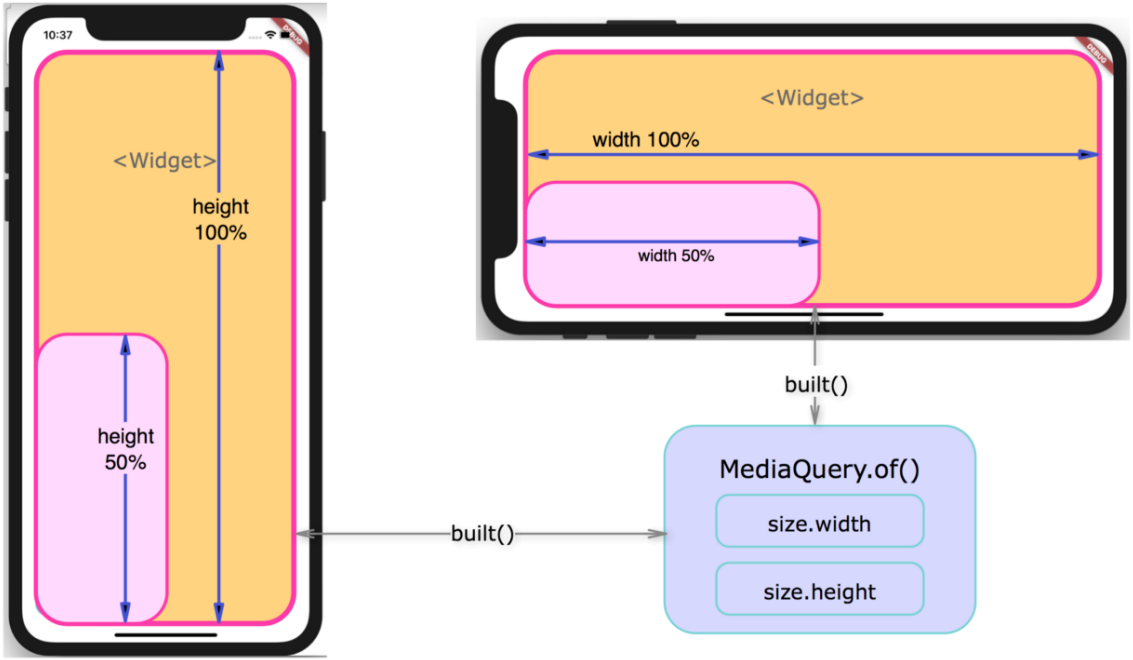
Expanded & Flexible



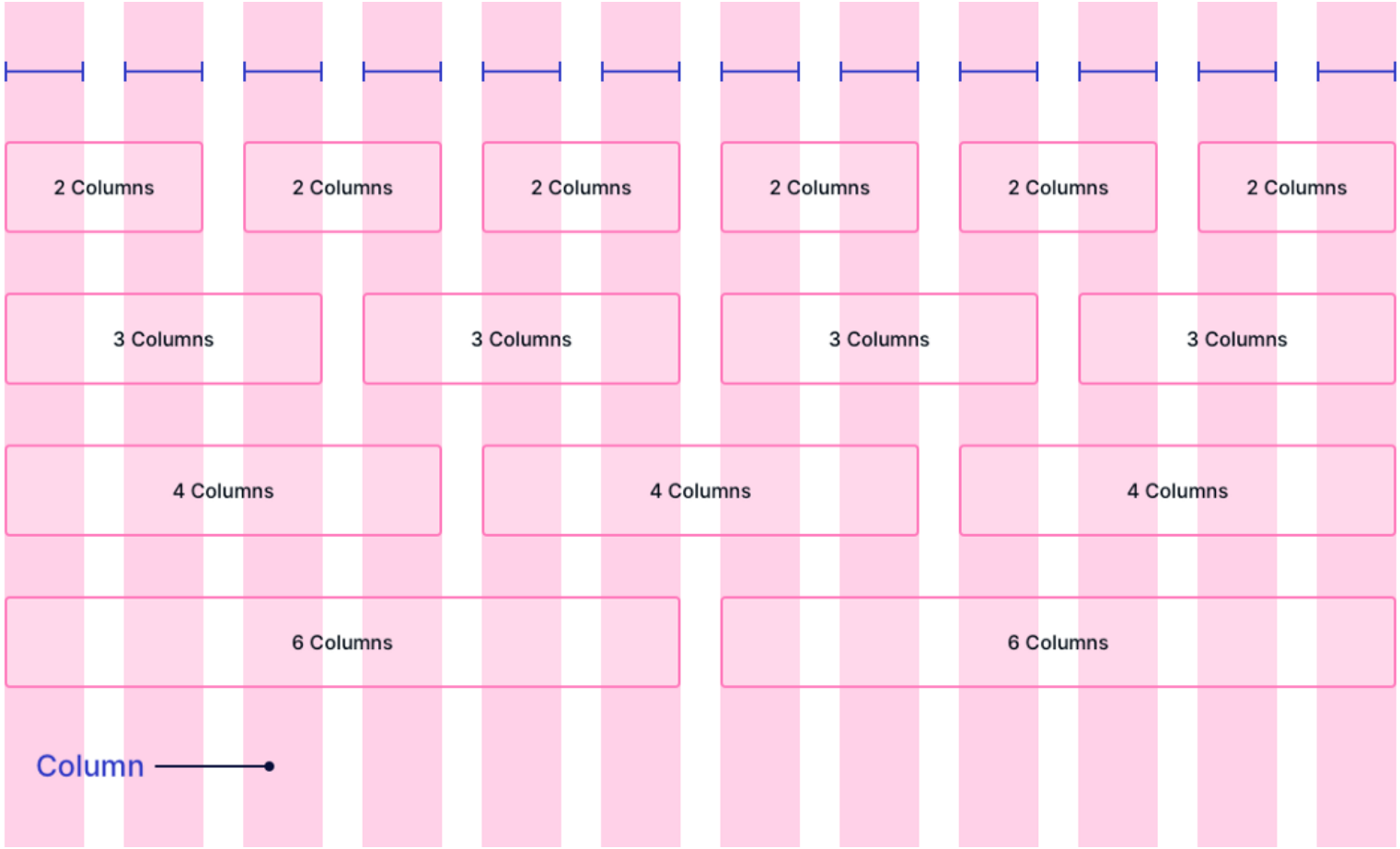




Media Query


10:37
<Widget>
height
100%
height
50%
built()
<Widget>
width 100%
width 50%
built()
MediaQuery.of()
size.width
size.height

Responsive Grid



Stateful Widget

- First Gear
- Pickup Increase
- Second Gear
- Pickup Increase
- Third Gear
- Forth Gear
- Gear Shift Down
- Pickup Decrease

What is state:

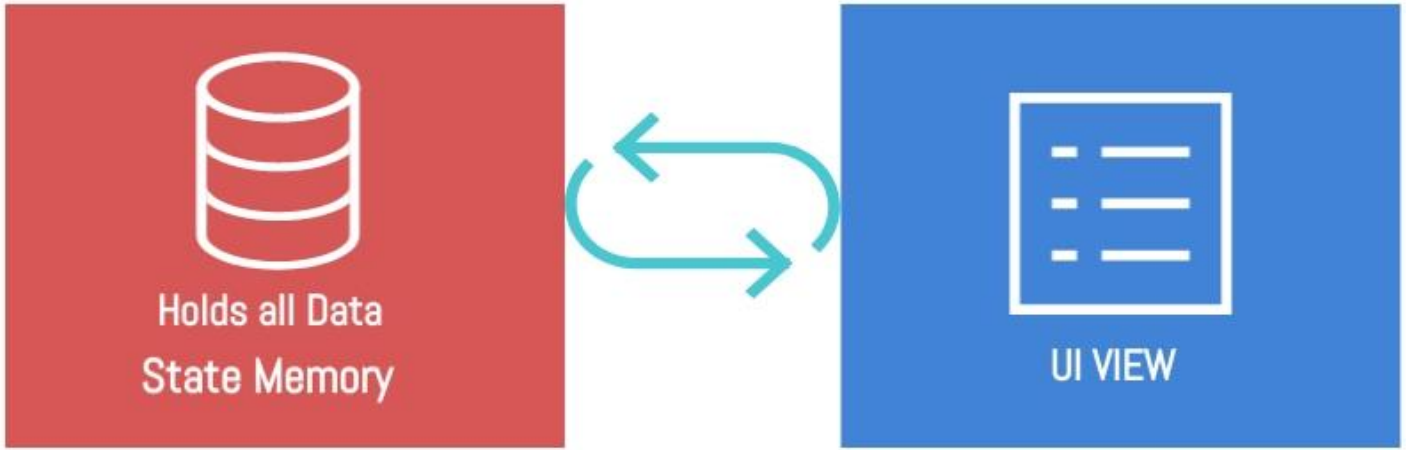
- Anything that exists in the memory of the app while the app is running.
- When state values change, view update automatically

Stateless & Stateful

Stateless & Stateful

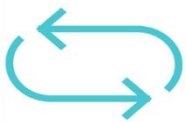
Stateless	Statefull
Not update at runtime	Update at runtime
Update when initiate	Update when initiate
Can't Change UI Dynamically	Can change UI dynamically
No information caching	Can cache information
Not suitable for API calling	Suitable for API call
Not suitable for complex UI	Suitable for complex UI

How Stateful Widget Works



How Stateful Widget Works

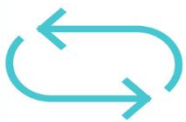
```dart
class MyHomePage extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return MyHomePageUI();
  }
}
```



```dart
class MyHomePageUI extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('App'),) ,
    );  // Scaffold
  }
}
```

Lets Create Our First Stateful Widget

```dart
class MyHomePage extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return MyHomePageUI();
  }
}
```



```dart
class MyHomePageUI extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('App'),) ,
    );  // Scaffold
  }
}
```

State Lifecycle methods

- The Stateful widget is mutable
- It can be change multiple times within its lifetime.
- So we need to understand it's lifecycle

createState()

- This method creates a State object. This object is where all the mutable state for that widget is held.
- This step is not marked as a real step in the lifecycle, but it is important to know what is going on in the background.

initState()

- Automatically called only once, when the state object is created for the first time.
- Use this method to manage HTTP requests and subscribe to streams or any other object that could change the data on this widget.

didChangeDependencies()

- Framework will call this method immediately after the **initState**().
- The build method is always called after this method, so this method is rarely needed

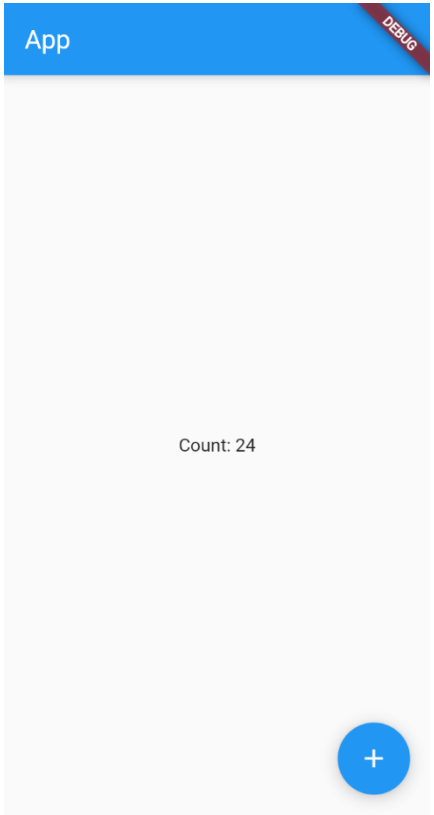
build()

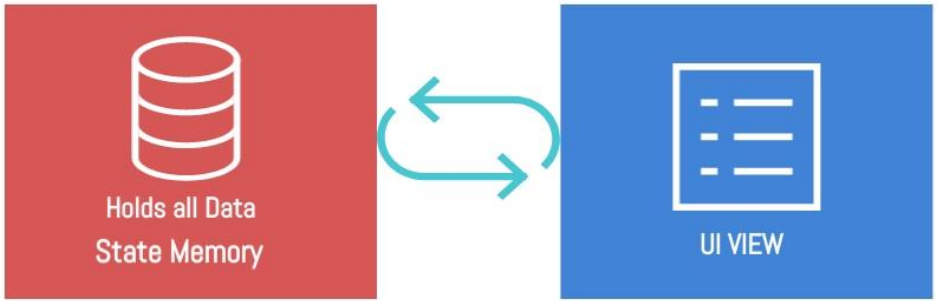
- it will be called many times during the lifecycle, but the first time is after the **didChangeDependencies**() method.
- Whenever the widget that belongs to the state is updated, the framework will always execute this method

setState()

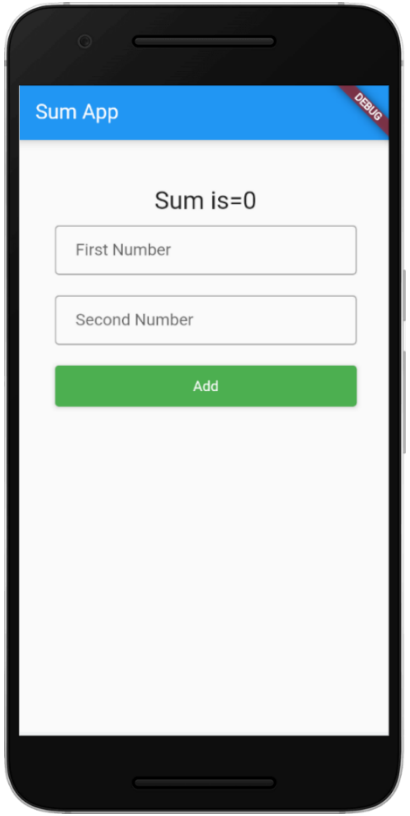
- The setState() method notifies the framework that the internal state of the current object is changed, now it's time to update the view

Lets Create A Counter App

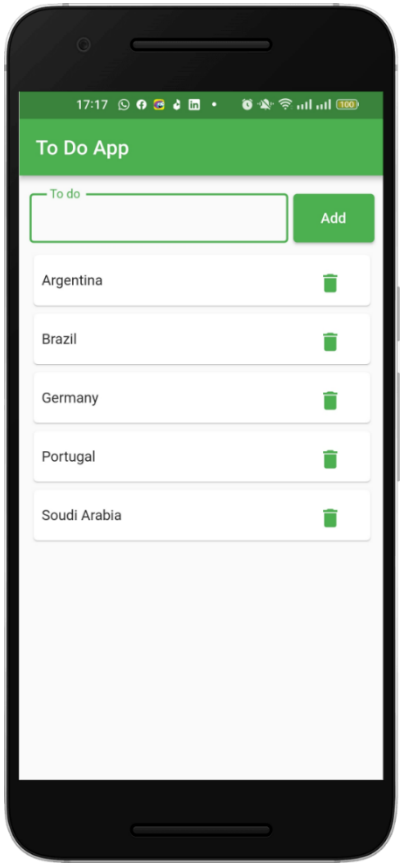


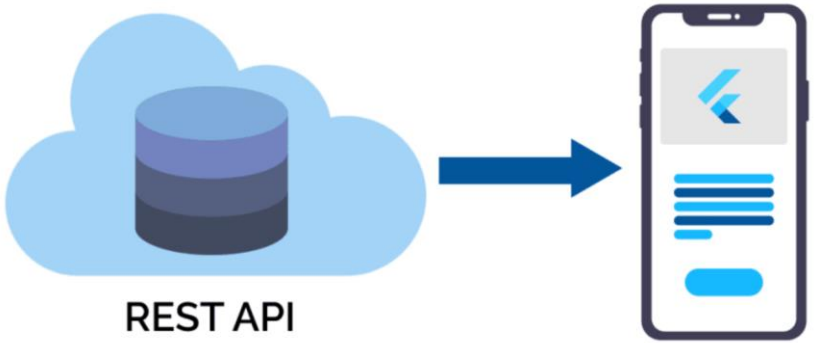


Lets Create Sum Calculator

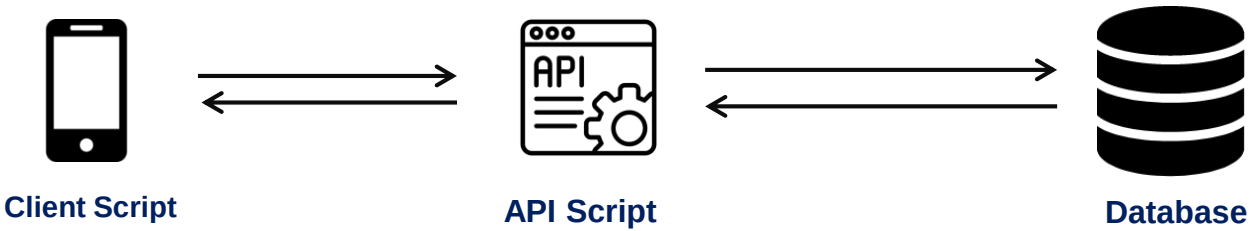


Lets Create To do list




REST API

Representational State Transfer




POSTMAN

Rest API practices

- Best practices of json
- Request response model
- Rest API working flow
- Postman & Documentation

JavaScript Object Notation (JSON)

- JSON is a lightweight data-interchange format that is completely language independent.
- It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data
- The official Internet media type for JSON is application/json.
- It was designed for human-readable data interchange.
- The filename extension is .json.

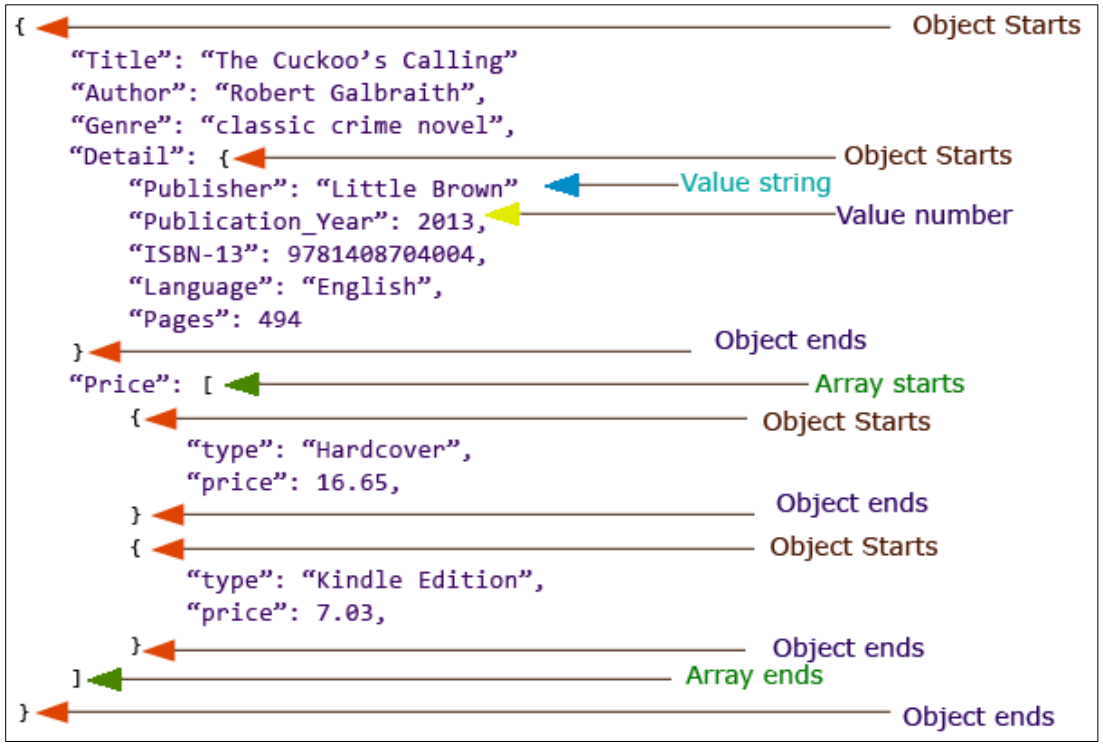
Uses of JSON

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.

Characteristics of JSON

- JSON is easy to read and write.
- It is a lightweight text-based interchange format.
- JSON is language independent.

Understanding JSON Structure



JSON - Data Types

Type	Description
Number	Double- precision floating-point format in JavaScript
String	Double-quoted Unicode with backslash escaping
Boolean	True or False
Array	An ordered sequence of values
Value	It can be a string, a number, true or false, null etc
Object	An unordered collection of key:value pairs
Whitespace	Can be used between any pair of tokens
null	Empty

- Always enclose the **Key : Value** pair within double quotes

```
{'id': '1','name':File} is not right ✗

{"id": 1,"name":"File"} is okay  ✓

{"id": "1","name":"File"} is the best  ✓
```

- Never use Hyphens in your Key fields

```
{"first-name":"Rachel","last-name":"Green"}  is not right. ✗

{"first_name":"Rachel","last_name":"Green"} is okay ✓

{"firstname":"Rachel","lastname":"Green"} is okay ✓

{"firstName":"Rachel","lastName":"Green"} is the best. ✓
```

Bad Special Characters & Solution

Characters	Replace with
Backspace	\b
Form feed	\f
Newline	\n
Carriage return	\r
Tab	\t
Double quote	\"
Backslash	\\

- Bad Special Characters And Solution

```
"<h2>About US <br>\r\n</h2>\r\n<p>It has been exactly 3 years since I wrote my first blog series
\r\nentitled “Flavorful Tuscany”, but starting it was definitely not easy. \r\nBack then, I didn’t know
much about blogging, let alone think that one \r\nday it could become <strong>my full-time job</strong>.
Even though I had\r\n many recipes and food-related stories to tell, it never crossed my mind\r\n that I
could be sharing them with the whole world.</p>\r\n<p>I am now a <strong>full-time blogger</strong> and
the curator of the <a data-cke-saved-href=\"https://ckeditor.com/ckeditor-4/#\"
href=\"https://ckeditor.com/ckeditor-4/#\">Simply delicious newsletter</a>, sharing stories about
traveling and cooking, as well as tips on how to run a successful blog.</p>\r\n<p>If you are tempted by
the idea of creating your own blog, please think about the following:</p>\r\n<ul><li>Your story (what do
you want to tell your audience)</li><li>Your audience (who do you write for)</li><li>Your blog name and
design<br>\r\n\t</li></ul>\r\n<p>After you get your head around these 3 essentials, all you have to do
is grab your keyboard and the rest will follow.</p>"
```

Practices of JSON

- Always create a Root element.

JSON with root element

```
{
"menu": [
  {
    "id": "1",
    "name":"File",
    "value": "F",
    "popup": {
      "menuitem": [
        {"name":"New", "value": "1N", "onclick": "newDoc()"},
        {"name":"Open", "value": "1O", "onclick": "openDoc()"},
        {"name":"Close", "value": "1C", "onclick": "closeDoc()"}
        ]
      }
  },
  {
    "id": "2",
    "name":"Edit",
    "value": "E",
    "popup": {
      "menuitem": [
        {"name":"Undo", "value": "2U", "onclick": "undo()"},
        {"name":"Copy", "value": "2C", "onclick": "copy()"},
        {"name":"Cut", "value": "2T", "onclick": "cut()"}
        ]
      }
  }
  ]
}
```

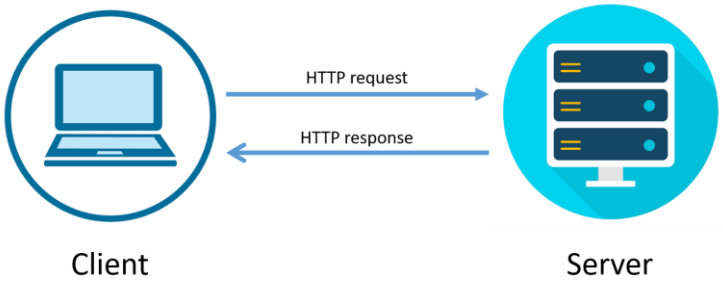
JSON without root element

```
[
  {
    "id": "1",
    "name":"File",
    "value": "F",
    "popup": {
      "menuitem": [
        {"name":"New", "value": "1N", "onclick": "newDoc()"},
        {"name":"Open", "value": "1O", "onclick": "openDoc()"},
        {"name":"Close", "value": "1C", "onclick": "closeDoc()"}
        ]
      }
  },
  {
    "id": "2",
    "name":"Edit",
    "value": "E",
    "popup": {
      "menuitem": [
        {"name":"Undo", "value": "2U", "onclick": "undo()"},
        {"name":"Copy", "value": "2C", "onclick": "copy()"},
        {"name":"Cut", "value": "2T", "onclick": "cut()"}
        ]
      }


  }
]
```

Request Response Model

HTTP/HTTPS Request Response Communication

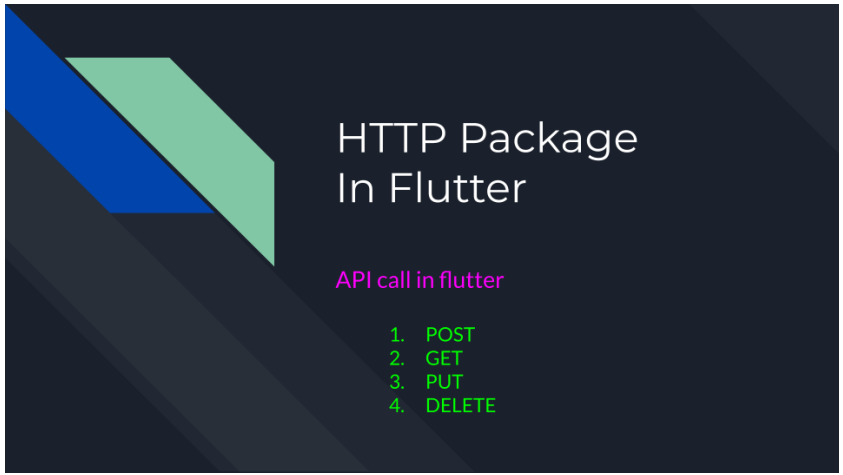


- In request/response communication mode.
- One software module sends a request to a second software module and waits for a response.
- The First software module performs the role of the client.
- The second, the role of the server,
- This is called client/server interaction.

Request Response Model

Application Level Client :

- HTTP Client is an application library used in client side application to generate request and receive response.
- HTTP Client’s libraries varies from platform to platform.



HTTP Request

HTTP Request is the first step to initiate web request/response communication. Every request is a combination of request header, body and request URL.

Http Request Segments

Request Area	Standard Data Type
Body	Simple String, JSON, Download, Redirect, XML
Header	Key Pair Value
URL Parameter	String

Request Response Model

HTTP Request Methods:

Method Name	Responsibilities
GET()	The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.
Head()	Same as GET, but transfers the status line and header section only.
POST()	A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
PUT()	Replaces all current representations of the target resource with the uploaded content.
DELETE()	Removes all current representations of the target resource given by a URI.

Request Response Model

Request Compare GET vs. POST:

Key Points	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	**Can be bookmarked**	Cannot be bookmarked
Cached	Can be	Never
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions

HTTP Response:

Http response is the final step of request-response communication. Every response is a combination of response header, body an d cookies.

Http Response Segments:

Response Area	Standard Data Type
Body	Simple String, JSON, Download, Redirect, XML
Header	Key Pair Value
Cookies	Key Pair Value

Request Response Model

HTTP Response status messages

Code	Meaning	Description
200	OK	The request is OK (this is the standard response for successful HTTP requests)
201	Created	The request has been fulfilled, and a new resource is created
202	Accepted	The request has been accepted for processing, but the processing has not been completed
203	Non-Authoritative Information	The request has been successfully processed, but is returning information that may be from another source
204	No Content	The request has been successfully processed, but is not returning any content
205	Reset Content	The request has been successfully processed, but is not returning any content, and requires that the requester reset the document view

HTTP Response status code

Code	Meaning	Description
206	Partial Content	The server is delivering only part of the resource due to a range header sent by the client
400	Bad Request	The request cannot be fulfilled due to bad syntax
401	Unauthorized	The request was a legal request, but the server is refusing to respond to it.
403	Forbidden	The request was a legal request, but the server is refusing to respond to it
404	Not Found	The requested page could not be found but may be available again in the future
405	Method Not Allowed	A request was made of a page using a request method not supported by that page

HTTP Response status Code

Code	Meaning	Description
408	Request Timeout	Request Timeout
500	Internal Server Error	A generic error message, given when no more specific message is suitable
502	Bad Gateway	The server was acting as a gateway or proxy and received an invalid response from the upstream server
503	Service Unavailable	The server is currently unavailable (overloaded or down)

Response Header:

- Provide proper http response status code.
- Provide proper content type, file type if any.
- Provide cache status if any.
- Authentication token should provide via response header.
- Only string data is allowed for response header.
- Provide content length if any.
- Provide response date and time.
- Follow request-response model described before.

Response Body:

- Avoid providing response status, code, message via response body
- Use JSON best practices for JSON response body.
- For single result, can use String, Boolean directly.
- Provide proper JSON encode-decode before writing JSON Body.
- Follow discussion on JSON described before.

Response Cookies:

- A Restful API may send cookies just like a regular Web Application that serves HTML
- Avoid using response cookies as it is violate stateless principle.
- If required use cookie encryption, decryption and other policies

Request Response Model

When use GET():

- GET is used to request something from server with less amount of data to pass.
- When nothing should change on the server because of your action.
- When request only retrieves data from a web server by specifying parameters
- Get method only carries request url & header not request body.

When use POST():

- POST should be used when the server state changes due to that action.
- When request needs its body, to pass large amount of data.
- When want to upload documents , images , video from client to server

Request Response Model

Request Body:

- Request body should be structured in JSON Array/ Object pattern
- Request body hold multipart/ form-data like images, audio, video etc
- Request body should not hold any auth related information.
- Request body should associated with specific request data model, setter getter can used for this

Request Header:

- Request header should carry all security related information, like token, auth etc.
- Only string **Key:Pair** value is allowed for header .
- Request header should provide user agent information of client application.
- If necessary CSRF/ XSRF should provide via header.
- Request header should associated with middleware controller, where necessary

Bearer Authentication

গাড়ির গ্যারেজ এর টোকেন সিস্টেম

JWT (JSON WEB TOKEN):

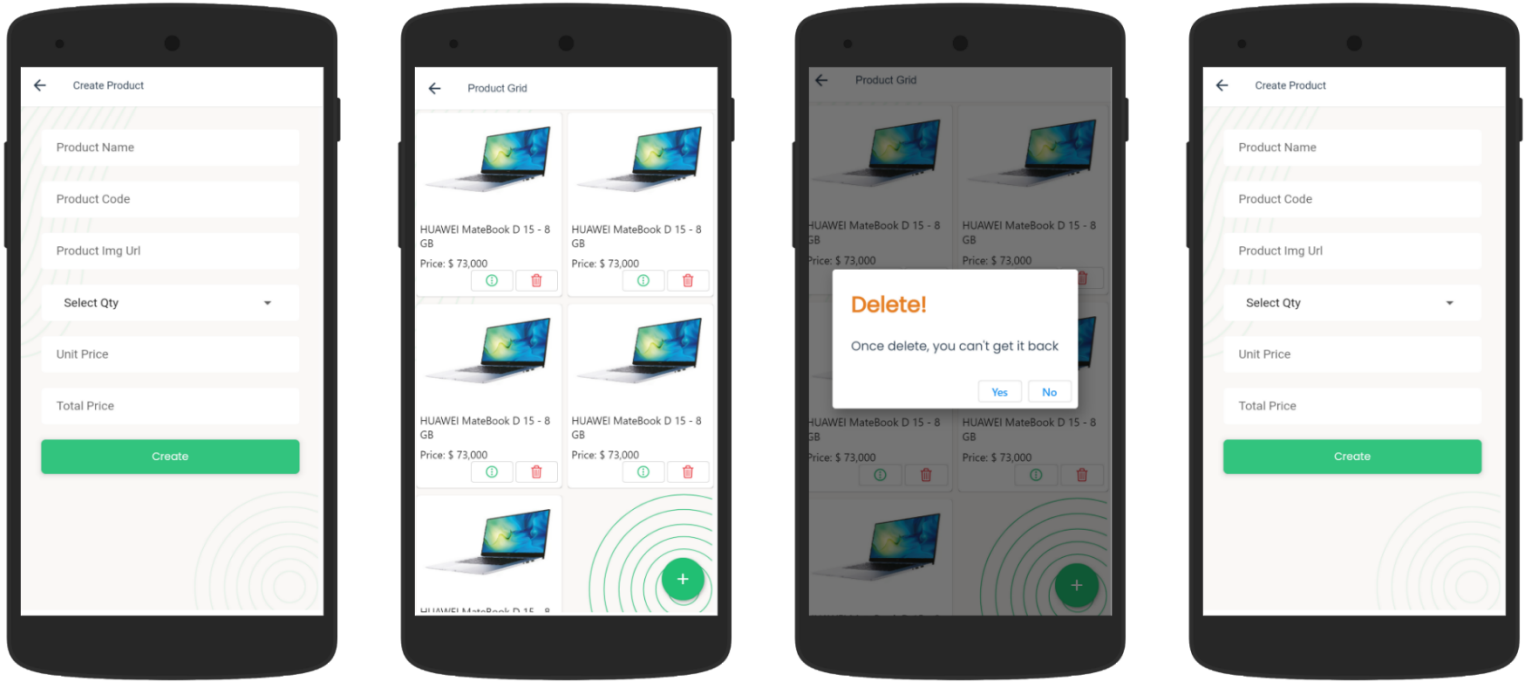
- Compact and self-contained way for securely transmitting information between parties as a JSON object.
- Information can be verified and trusted because it is digitally signed.

USES:

Authorization: Allowing the user to access routes, services, and resources

Information Exchange: Way of securely transmitting information between parties.

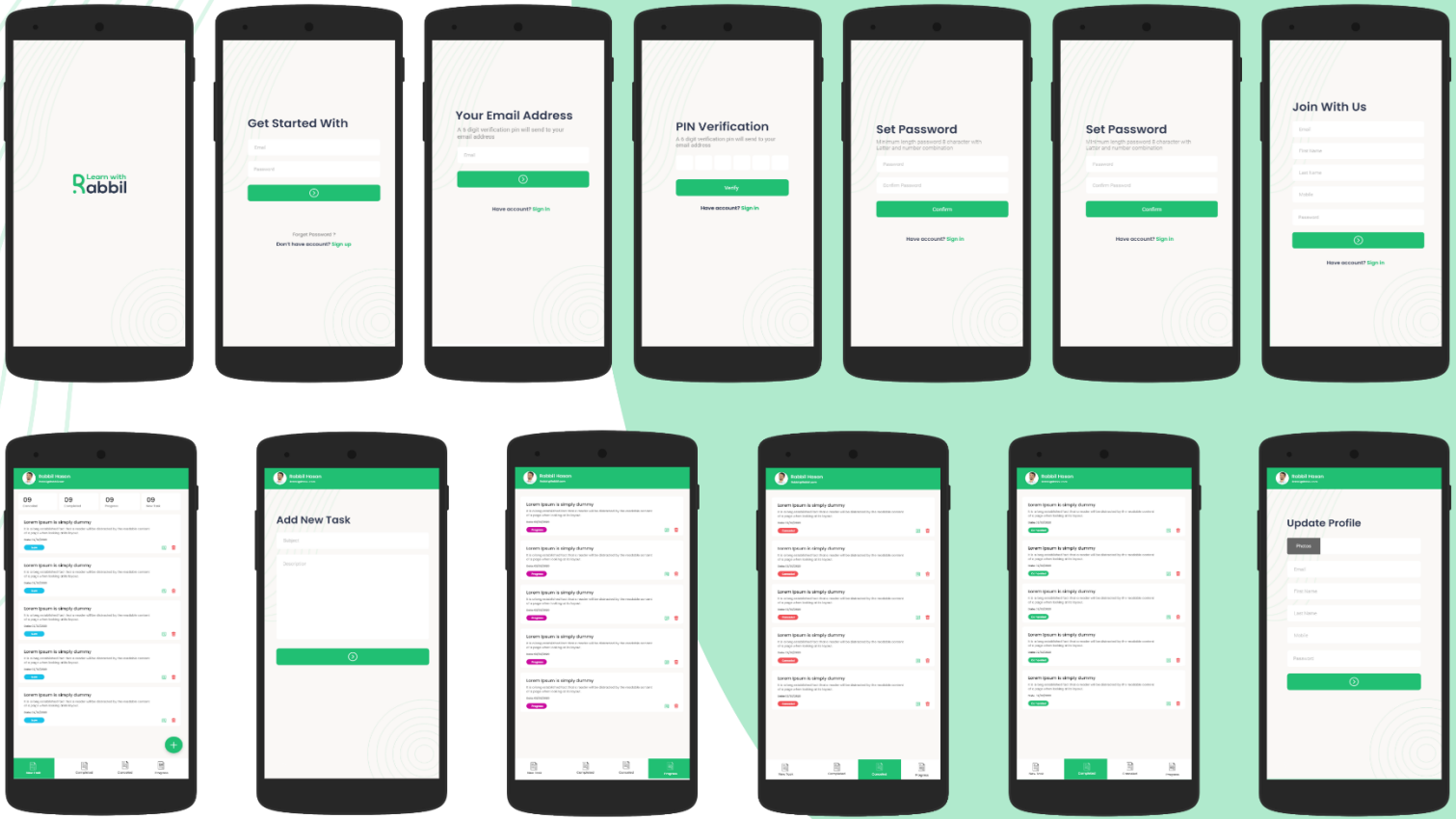
CRUD APP


Create Product
Product Name
Product Code
Product Img Url
Select Qty
Unit Price
Total Price
Create
Product Grid
HUAWEI MateBook D 15 - 8 GB
Price: $ 73,000
Delete!
Once delete, you can't get it back
Yes
No

CRUD APP

- Understanding Rest API Documentation
- Create new flutter project
- Add necessary packages
- Create view according to the Rest API Need
- Call Rest API & make it dynamic

Task Manager Project



Task Manager Project Development Strategy

- Under stand the project flow
- Under stand the Rest API
- Create new flutter project
- Arrange flutter project structure
- Develop features on by one.