Bioinformatics

Koumakis Lefteris,

ChatziminaMaria

Lecture 5

Click to edit Master title style

- There are two ways to execute a “Future” and use the value it returns (If it returns anything). The most common way is to wait for the “Future” to return data. For this to work the function calling the code must be marked with “async”.
  - Example:

```
FlatButton(
  child: Text('Run Future'),
  onPressed: () async {
    var value = await myTypedFuture();
  },
)
```

- The other way to handle a “future” is to use the .then() function which is **a callback that's called when future completes successfully(with a value)**.
  - Example:

```
void runMyFuture() {
  myTypedFuture().then((value) {
    // Run the code here using the value
  });
}
```

Click to edit Master title style

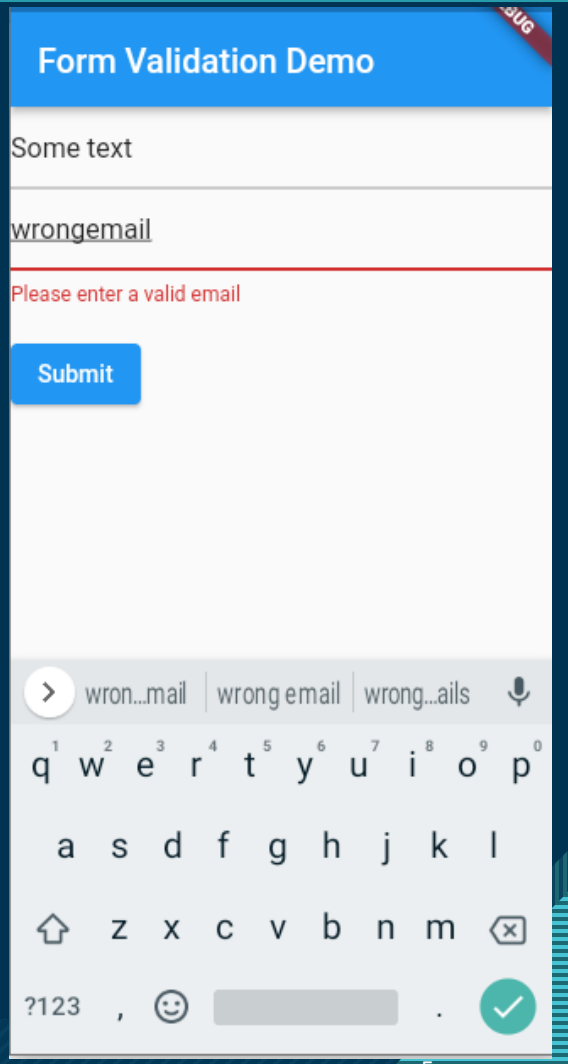
- GlobalKey<FormState>: We create a global key that uniquely characterizes the form and allows us to validate it.
- TextFormField
  - validator: Function that checks the value of the field.
  - controller: It is used so that we can manage the values outside the form
  - The initial value of the field can be given when the controller is created.
  - initialValue: we declare initial value of the field
    - Cannot be used at the same time as the controller.
  - onTap: can be used when we want to add some action when the field is clicked (like showing a datepicker)
  - keyboardType: keyboard type displayed

```dart
final _formKey = GlobalKey<FormState>();

TextEditingController textController = TextEditingController(text:"Some text");
TextEditingController emailController = TextEditingController(); //για να μπορου

@override
Widget build(BuildContext context) {
  // Build a Form widget using the _formKey created above.
  return Form(
    key: _formKey,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        TextFormField(
          controller: textController, //χρησιμοποιουμε τον controller ωστε να μπ
          //initialValue: "Some text", το initial value δεν μπορει να χρησιμοποι
          // The validator receives the text that the user has entered.
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Please enter some text';
            }
            return null;
          },
        ), // TextFormField

        TextFormField(
          controller: emailController,
          // The validator receives the text that the user has entered.
          validator: (value) {
            if (value == null || !value.contains("@")) { //ελεγχουμε αν υπαρχει
              return 'Please enter a valid email';
            }
            return null;
          },
```

Edit to edit Master title style

- When validate() is called it runs the functions for each field validator. If there are no errors it returns true. Otherwise it prints the error messages and returns false.

```
Padding(
  padding: const EdgeInsets.symmetric(vertical: 16.0),
  child: ElevatedButton(
    onPressed: () {
      // Validate returns true if the form is valid, or fals
      if (_formKey.currentState!.validate()) {
        print(emailController.text);  //απλα τυπωνουμε τις τ
        print(textController.text);
        // Αν η φορμα ειναι valid τυπωσε ενα snackbar (στο κ
        // συνηθως εδω σε πραγματικες εφαρμογες σωνονται οι
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Processing Data')),
        );
      }
    },
    child: const Text('Submit'),
  ), // ElevatedButton
```



Exercise: Form Validation (20 minutes)

Click to edit Master title style

- Task 1: Add a Username Field
- Add a new TextFormField for the username. The field must not be empty.
- Task 2: Add a Password Field
- Add a password field. The password must contain at least 6 characters.
- Task 3: Add a Confirm Password Field
- Add a confirm password field. Its value must match the password.
- Task 4: Improve Email Validation
- Check if the email has a valid format, e.g. it contains @ and a dot.

Exercise - Form Validation

TASK 1

```
if (value == null || value.isEmpty) {
  return 'Please enter username';
}
```

TASK 2

```
if (value!.length < 6) {
  return 'Password too short';
}
```

TASK 4

```
if (!RegExp(r'\S+@\S+\.\S+').hasMatch(value!)) {
  return 'Enter valid email';
}
```

TASK 3

```
final passwordController = TextEditingController();

TextFormField(
  controller: passwordController,
  obscureText: true,
  decoration: InputDecoration(labelText: 'Password'),
)

TextFormField(
  obscureText: true,
  decoration: InputDecoration(labelText: 'Confirm Password'),
  validator: (value) {
    if (value != passwordController.text) {
      return 'Passwords do not match';
    }
    return null;
  },
)
```

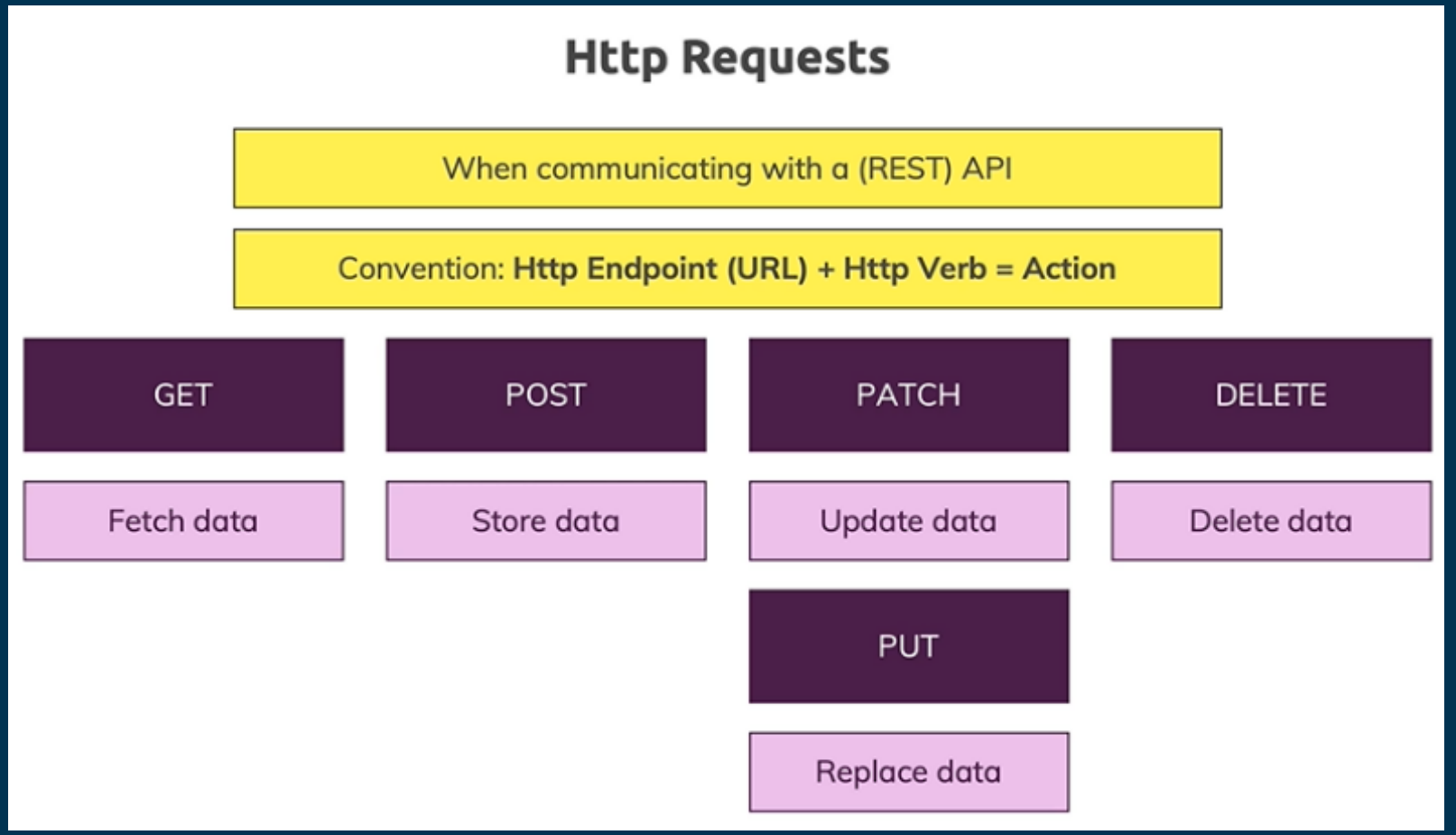
Click to edit Master title style

- Warning: Use key in widget constructors.dart
- A Key is an identifier for Widgets, Elements
- When the widget tree is redrawn flutter needs to know:
  - Which widgets are completely new and which already have state, and this is where the key helps
- It is not mandatory but is considered good practice

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

// ignore: must_be_immutable
class WelcomePage extends StatelessWidget {
  String name,surname;
  WelcomePage({Key? key,required this.name,required this.surname}) : super(key: key);

 @override
 Widget build(BuildContext context) {
   return Scaffold(
   body: Center(
     child: Center(
       child: Column(
         crossAxisAlignment: CrossAxisAlignment.start,
         children: [
           Text('Name: $name'),
           Text('Surname: $surname'),
         ],
       ), // Column
     ), // Center
   ), // Center
   ); // Scaffold
 }
}
```

```dart
Navigator.of(context).push(MaterialPageRoute(
                          builder: (context) => WelcomePage(
                                                  name: _name,
                                                  surname: _surname,
                                                ) // WelcomePage
                        ) // MaterialPageRoute
                        );
```

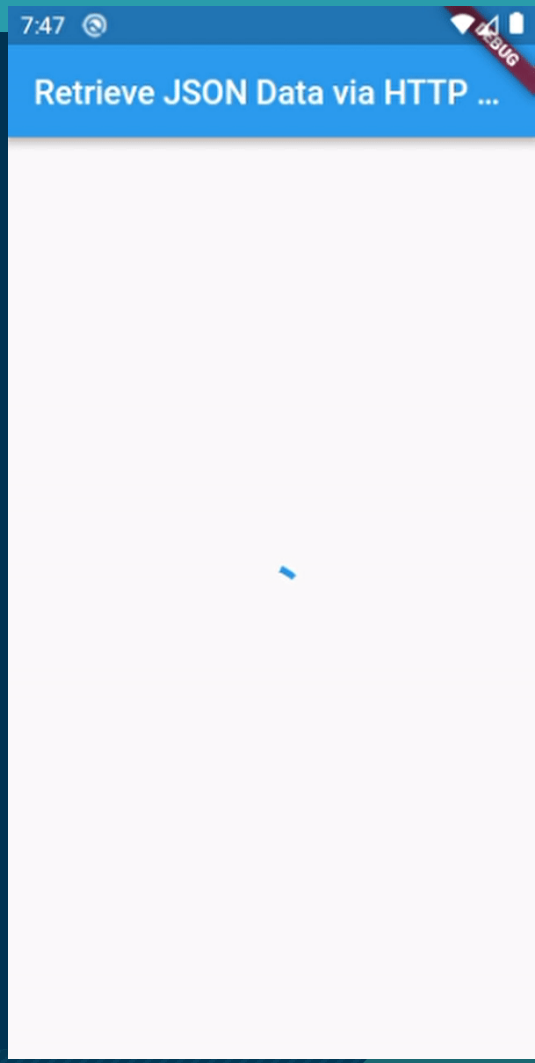
Http requests



HTTP Request
Click to edit Master title style

- http.get returns a future object
- We use the .then((value){}) method on a future when we want to execute code after data has been assigned to the future.
- We use async/await to handle asynchronous operations The await keyword pauses execution until the Future returns a value
- In this example, we show a loader while data is being fetched
- The loader remains visible until the isLoading variable changes from true to false

```dart
class MyGetHttpDataState extends State<MyGetHttpData> {
  // API endpoint
  final String url = "https://jsonplaceholder.typicode.com/users";
  // List to store the fetched data
  List data = [];
  // Loading indicator flag
  var isLoading = true;
  @override
  void initState() {
    super.initState();
    // Call the method to fetch data when the app starts
    getJSONData();
  }
  // Function to fetch JSON data from the API
  Future<void> getJSONData() async {
    try {
      final response = await http.get(Uri.parse(url));
      // Print status and response body for debugging
      if (response.statusCode == 200) {
        // Convert JSON string into Dart object (List/Map)
        final decodedData = json.decode(response.body);
        // Update the state with the fetched data
        setState(() {
          data = decodedData; // Here the API returns a list directly
          isLoading = false;  // Stop showing the loader
        });
      } else {
        // Handle request failure
        setState(() {
          isLoading = false;
        });
      }
    } catch (error) {
      // Handle any errors (e.g., network issues)
      setState(() {
        isLoading = false;
      });
    }
  }
```



GestureDetector vs InkWell

- Both provide many common functions like onTap, onLongPress etc. The main difference is that GestureDetector provides more controls like dragging etc. while it does not include the ripple effect, which InkWell does.
- You can use either of the two according to your needs,
  - if you want the ripple effect, choose InkWell,
  - if you need more controls choose GestureDetector
  - or you can combine both.

Staggered Grid View title style

- Package: https://pub.dev/packages/flutter_staggered_grid_view
  - Installation steps on tab“Installing”
- Insert in pubspec.yaml :
  - flutter_staggered_grid_view:
- If it is not installed automatically when the file is saved, we run:
  - flutter pub get
- We import it into the file we want to use:
  - import 'package:flutter_staggered_grid_view/flutter_staggered_grid_view.dart';

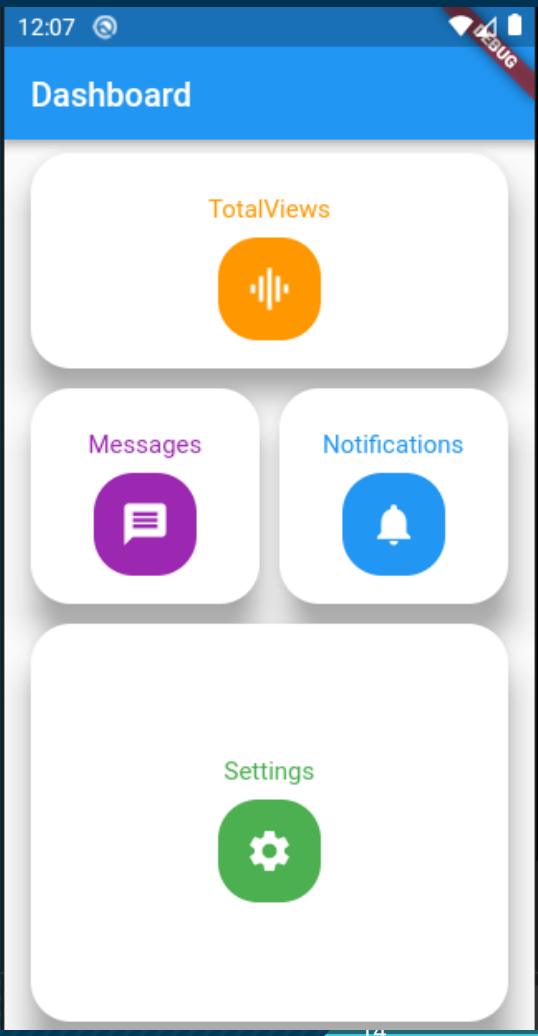
StaggeredGridView title style

- StaggeredGridView.count the constructor
- crossAxisCount:
  - The number of columns we want the grid view to cover
- crossAxisSpacing:
  - The space between boxes/tiles on the horizontal axis
- mainAxisSpacing:
  - The space between the boxes/tiles on the vertical axis

StaggeredGridView title style

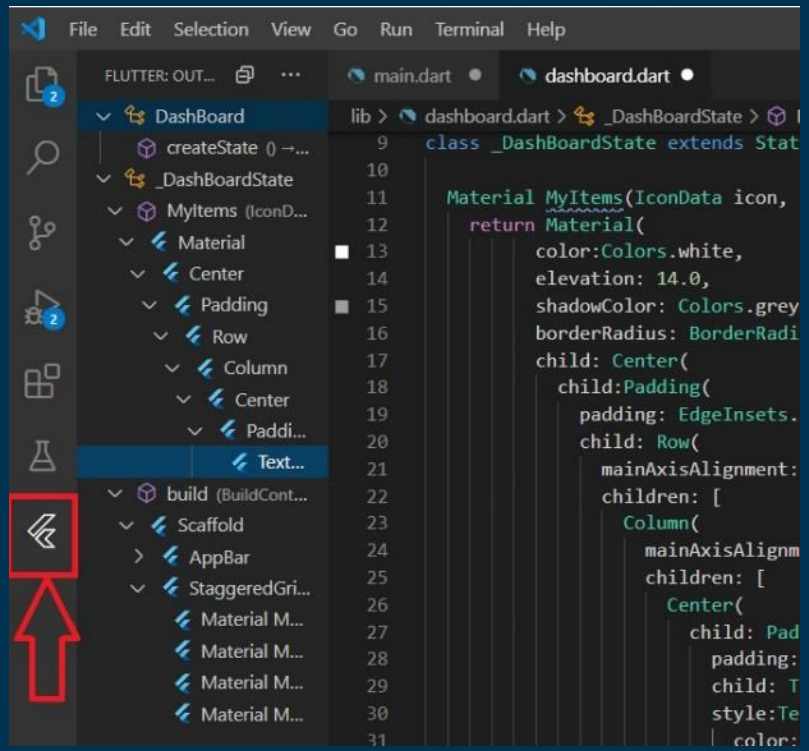
- The “TotalViews” covers both columns that we have declared in crossAxisCount and has a smaller height than “Settings” because in mainAxisCellCount we have the value 1
- “messages” and “notifications” have 1 crossAxisCellCount so each one of them covers one column and are shown next to each other

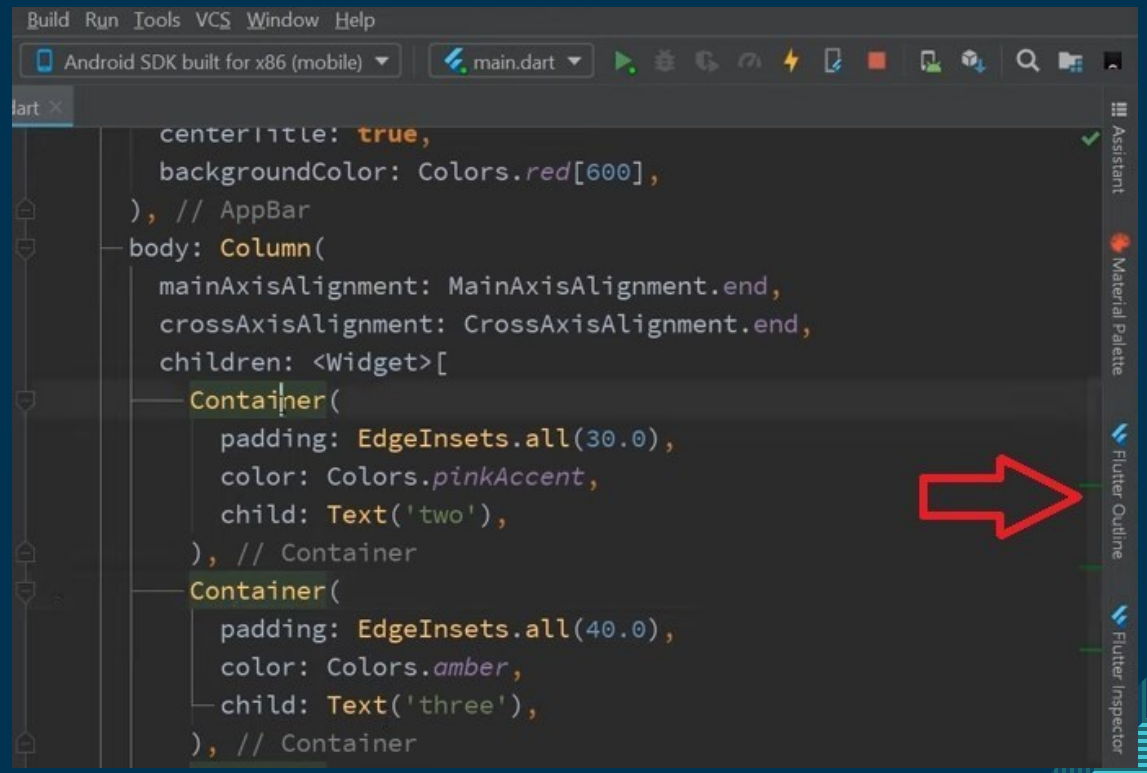
```
body: SingleChildScrollView(
  child: StaggeredGrid.count(
    crossAxisCount:
        2, //ο αριθμος των στηλων που θελουμε να καλυπτει το grid
    crossAxisSpacing:
        12.0, //η αποσταση μεταξυ των κουτιων στον οριζοντιο αξονα
    mainAxisSpacing:
        12.0, //η αποσταση μεταξυ των κουτιων στον καθετο αξονα
    children: [
      StaggeredGridTile.count(
        crossAxisCellCount: 2,//Ο αριθμός των κελιών που παίρνει αυτό το πλακί
        mainAxisCellCount: 1,//Ο αριθμός των κελιών που παίρνει αυτό το πλακίδ
        child: myItems(Icons.graphic_eq, "TotalViews", Colors.orange),
      ), // StaggeredGridTile.count
      StaggeredGridTile.count(
        crossAxisCellCount: 1,
        mainAxisCellCount: 1,
        child: myItems(Icons.message, "Messages", Colors.purple),
      ), // StaggeredGridTile.count
      StaggeredGridTile.count(
        crossAxisCellCount: 1,
        mainAxisCellCount: 1,
        child:
            myItems(Icons.notifications, "Notifications", Colors.blue),
      ), // StaggeredGridTile.count
      StaggeredGridTile.count(
        crossAxisCellCount: 2,
        mainAxisCellCount: 2,
        child: myItems(Icons.settings, "Settings", Colors.green),
      ), // StaggeredGridTile.count
    ],
  ), // StaggeredGrid.count
)); // SingleChildScrollView // Scaffold
```



Flutter outline Master title style

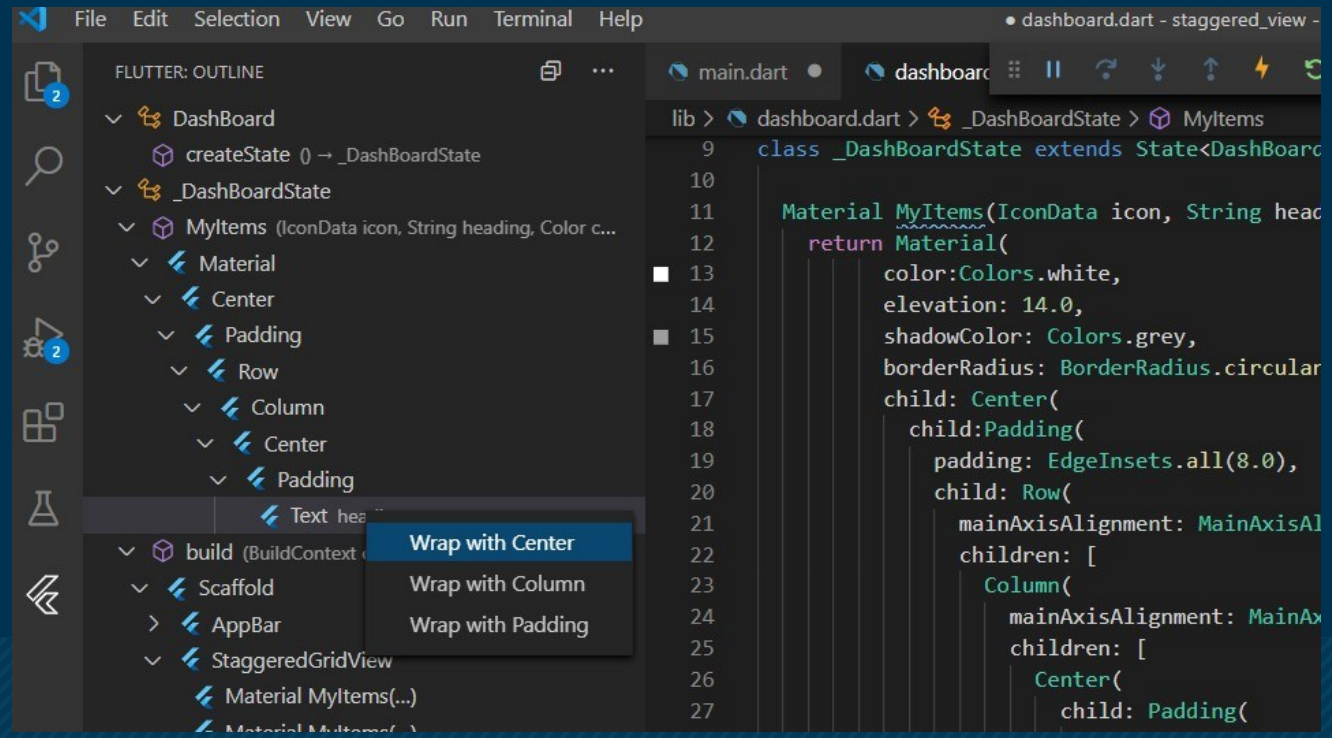
- Command Pallete → write outline view → choose outline view





Flutter outline: Master title style

- We select the widget we want (it is highlighted in the code)
- In the widget tree of the outline view, we right-click on the widget we want and we can wrap it with other widgets such as padding, center, etc.



Questions

- How does sending HTTP requests to a (REST) API work?
  - The Flutter application sends requests with specific Http Verbs and endpoints (paths) to the web server (= the API).
- What is "future" in Dart?
  - An object that waits for an asynchronous operation to complete and then (possibly) uses the data.
- What is "asynchronous" (or "async") code?
  - Code that is not executed immediately but is executed at some point in the future. The rest of the code continues execution without waiting for the async code to complete.

Click to edit Master title style

- Prefer const with constant constructors.dart (prefer_const_constructors)
- Solution:
  - We use const in as many constructors as we can
  - Insert at the beginning of the file: // ignore_for_file: prefer_const_constructors
  - Insert in file analysis_options.yaml under rules
    - prefer_const_constructors : false

```
rules:
  prefer_const_constructors : false
  # avoid_print: false  # Uncomment to disable the `avoid_print` rule
  # prefer_single_quotes: true  # Uncomment to enable the `prefer_single_quotes` rule
```

Variable late Master title style

- When the compiler considers that a variable should not be null we use late
  - The compiler with late "trusts" us to initialize the non-nullable variable after initializing it

```
late List<GDPData> _chartData;
late TooltipBehavior _tooltipBehavior;
```

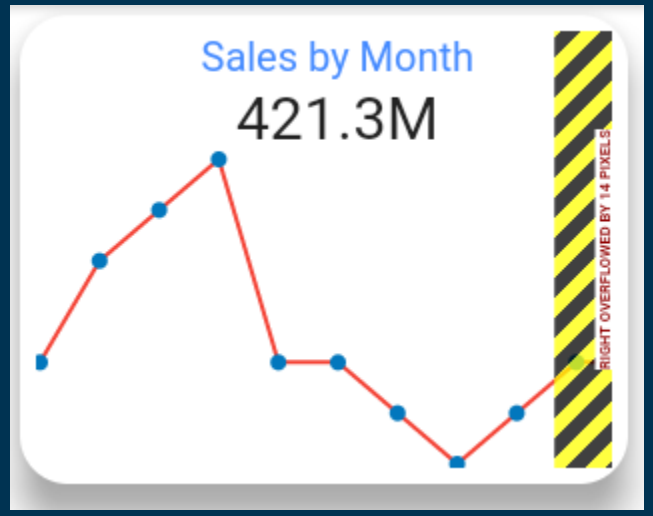
- Lazily initializing a variable
  - The variable may not be needed and its initialization may be "expensive".
  - In the example below if result is never used the "expensive" function _getResult is never called

```
// This is the program's only call to _getResult().
late String result = _getResult(); // Lazily initialized.
```

Click more Give Master Editor style

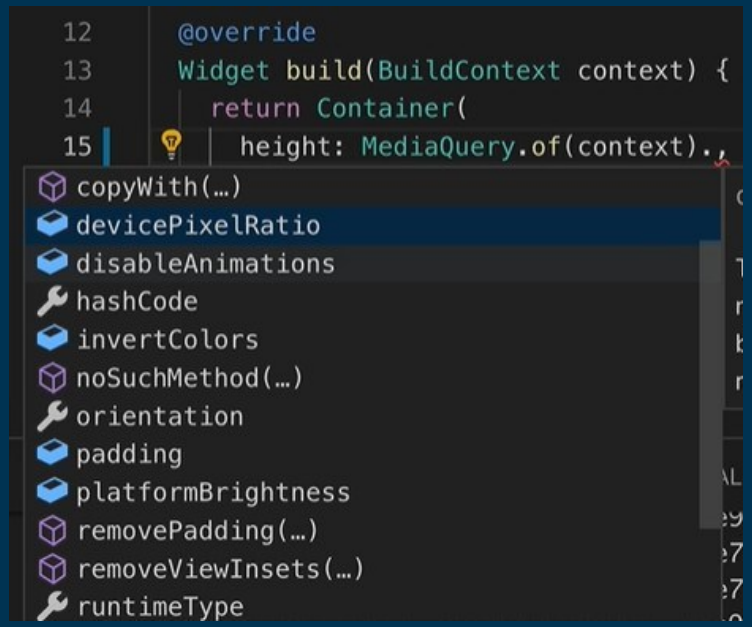
- In the example, the column tries cover more space than the space that can be allocated inside the row, causing an overflow error. The row in the example is the Parent of the Column. Solution:

```
child: Row(
  children: [
    Flexible(
      child: Column(
        // code
              ),
            ),
          ]
        )
```



Get Dynamic Max Device Size

- With the MediaQuery class we have several options:



- The size option gives us the height and width.
  - We can use the percentage of these values by multiplying the height/width with the percentage (60% -> 0.6).

```
@override
Widget build(BuildContext context) {
  return Container(
    height: MediaQuery.of(context).size.height * 0.6,
```

SQFlite

- Supported by Android and iOS
- Used to keep data saved locally
- **SQFlite** plugin : https://pub.dev/packages/sqflite
- CRUD operations
  - create, read, update, and delete (CRUD) the four basic functions

Steps

- Setup **SQFlite**
  - We import the following packages into the pubspec.yaml file under flutter sdk with this exact alignment/spaces.

    sqflite: appropriate version

    path: appropriate version

    path_provider: appropriate version
- Open database
  - Create database
- Create tables
- Perform CRUD
- Flutter Packages versions:
  - any : update to the latest version even if it is not compatible
  - κενό: update to the latest version but takes compatibility into account

```
environment:
  sdk: ">=2.7.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  sqflite: ^2.0.0+4
  path: ^1.8.0
  path_provider: ^2.0.7
```

```
  sqflite:
  path: any
  path_provider:
```

Create database

- Creating a database returns a future.
- getDatabasesPath() returns the path stored on the device depending on whether it is android/iOS
- path.join: joins the storage path with the name we give to our database e.g. 'mydb.db'
- openDatabase(): if the database exists in the given path it opens it otherwise it creates it
- In onCreate we give the command to create the table
- Follows SQL rules

```
static Future<sql.Database> database() async{
    //δημιουργια βασης αν δεν υπαρχει
final dbPath = await sql.getDatabasesPath(); //to path αναλογα android/iOS
return sql.openDatabase(path.join(dbPath,'mydb.db'),version:1,
onCreate:_createDB,
); //αν υπαρχει το αρχειο ανοιγει την βαση διαφορετικα την δημιουργει
```

```
static Future _createDB(db,version){
    return db.execute('''
    CREATE TABLE notes(
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      title TEXT NOT NULL,
      text TEXT NOT NULL,
      extra TEXT)'''); //επιστρεφει ενα future
}
```

Insert in database

- db.insert(): data entry in the corresponding table
- confilictAlgorithm: choose what happens if the id (PRIMARY KEY) already exists, e.g. to be replaced or no action

```
static Future<void> insert( String table, Map<String,Object> data) async{
 print(data); //τυπωνω τα δεδομένα για να δω οτι ειναι σωστά
 final db = await DBHelper.database();
 db.insert(
   table,
   data,
   conflictAlgorithm:
   sql.ConflictAlgorithm.ignore);
//με το await εδω περιμενουμε να τελειωσουν ολες οι διαδικασιες στην συναρτηση insert


}
```

Return data

- dq.query: When only the table name is given (as here) it returns what the table contains
  - Named parameter where: executes the WHERE in the SELECT statement
- It returns a future and more specifically a map list that contains String and dynamic

```
static Future<List<Map<String,dynamic>>> getData(table) async {
 final db = await DBHelper.database();
    return db.query(table); //επιστρεφει μια λιστα απο maps
 }
```

```
static Future<List<Map<String,dynamic>>> getData(table) async {
 final db = await DBHelper.database();
    return db.query(table,where: "title LIKE '%titleofnote%'");
 }
```