Βιοϊατρική Πληροφορική

Λευτέρης Κουμάκης,
Μαρία Χατζημηνά

Εργαστήριο 4

Κλάση Map

- Η Map είναι μια κλάση της dart
- Χρησιμοποιεί key-value δομή
- Τη δηλώνουμε με τις { }
- Παράδειγμα:

```
var questions = [
  {
    'questionText': 'What\'s your favorite color?',
    'answers': ['black', 'white', 'purple'],
  },
{
    'questionText': 'What\'s your favorite animal?',
    'answers': ['cat', 'dog', 'mouse'],
  },
];
```

Ερωτήσεις

- Ποια είναι η διαφορά μεταξύ μιας λίστας/list ([]) και ενός χάρτη/map ({}) στο Dart / Flutter;
  - Οι λίστες σας δίνουν μια ταξινομημένη λίστα μεμονωμένων τιμών, που προσδιορίζονται από ένα ευρετήριο (index).
  - Το map χρησιμοποιεί ζεύγη τιμών-κλειδιών (key-value) όπου παίρνουμε τις τιμές βάση του κλειδιού τους.
  - Παράδειγμα:
    - print(questions[0]['questionText']) τυπώνει “What’s your favorite color”

```
var questions = [
  {
    'questionText': 'What\'s your favorite color?',
    'answers': ['black', 'white', 'purple'],
  },
{
    'questionText': 'What\'s your favorite animal?',
    'answers': ['cat', 'dog', 'mouse'],
  },
];
```

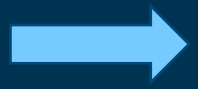
Final vs Const

- “Final“:
  - σημαίνει ότι γίνεται ανάθεση στην μεταβλητή μια φορά:
  - μια final μεταβλητή **πρέπει** να αρχικοποιηθεί. Μόλις εκχωρηθεί μια τιμή, η τιμή μιας final μεταβλητής δεν μπορεί να αλλάξει.
  - Γνωρίζουμε πριν τρέξει το πρόγραμμα ότι η τιμή αυτής της μεταβλητής δεν θα αλλάζει αλλά ακόμα δεν γνωρίζουμε την ακριβή τιμή. Η τιμή εκχωρείται όταν τρέξει το πρόγραμμα
    - final birthday = getBirthDateFromDB()
- “Const”:
  - Η τιμή πρέπει να είναι γνωστή κατά το χρόνο μεταγλώττισης,
    - const birthday = "2008/12/25"
  - Δεν είναι δυνατή η αλλαγή μετά την αρχικοποίηση.
- “Static”:
  - Η static χρησιμοποιείται για μια μεταβλητή επιπέδου κλάσης που είναι η ίδια για κάθε instance της κλάσης, αυτό σημαίνει ότι εάν τα δεδομένα είναι στατικά, μπορεί να προσπελαστεί χωρίς να δημιουργηθεί ένα αντικείμενο.
    - Παράδειγμα η κλάση Colors οπού χρησιμοποιούμε τα χρώματα χωρίς να δημιουργήσουμε αντικείμενο:
    - static const MaterialColor deepPurple = MaterialColor(…..
    - color: Colors.deepPurple

If statements

- Αν είναι true το condition που ελέγχουμε εκτελείται ο κώδικας μετά το “?” ενώ αν είναι false εκτελείται ο κώδικας μετά το “:”

```
void main() {
  var condition = true;
  condition ? print ("Condition is true") : print("Condition is false");

  var value = (1>2)?'δεν ειναι το 1 μεγαλυτερο απο το 2':'Το 2 ειναι μεγαλυτερο απο το 1';
  print(value);
}
```



```
Console

  Condition is true
  Το 2 ειναι μεγαλυτερο απο το 1
```

- Παράδειγμα στην εφαρμογή:

```
body: index < myList.length
?Text("To index einai mikrotero tou myList.length")
:Text("To index einai megalutero h iso tou myList.length")
```

Drawer Widget

- Video και Πληροφορίες: https://api.flutter.dev/flutter/material/Drawer-class.html
- Παράδειγμα :

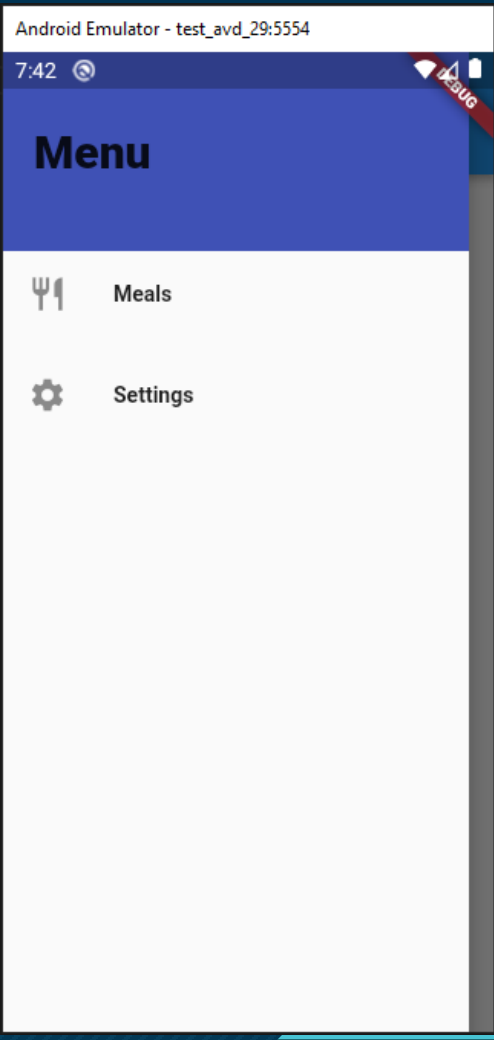
```
return Scaffold(
  appBar: AppBar(title: Text('Drawer')),
  body: Center(child: Text('My Page!')),
  drawer: Drawer(
    //Βαζουμε ενα Listview στο drawer. Γιατι θελουμε να ειμαστε σιγουροι
    // οτι ο χρηστης θα μπορει να κανει scroll σε ολα τα αντικειμενα
    // αν δεν χωρανε στην οθονη
    child: ListView(
      padding: EdgeInsets.zero,
      children: <Widget>[
        DrawerHeader(
          child: Text('Drawer Header'),
          decoration: BoxDecoration(
            color: Colors.blue,
          ), // BoxDecoration
        ), // DrawerHeader
        ListTile(
          title: Text('Item 1'),
          onTap: () {
            // κλεινουμε το drawer
            Navigator.pop(context);
          },
        ), // ListTile
        ListTile(
          title: Text('Item 2'),
          onTap: () {
            // κλεινουμε το drawer
            Navigator.pop(context);
          },
        ), // ListTile
      ], // <Widget>[]
    ), // ListView
  ), // Drawer
); // Scaffold
```

Custom Drawer

- ListTile builder

```dart
class MainDrawer extends StatelessWidget {

  //builder για το ListTile
  Widget buildListTile(String title, IconData icon, VoidCallback? pageHandler) {
    return ListTile(
      leading: Icon(
        icon,
        size: 26,
      ), // Icon
      title: Text(
        title,
        style: const TextStyle(
          fontWeight: FontWeight.bold,
        ), // TextStyle
      ), // Text
      onTap: pageHandler,
    ); // ListTile
  }
```

```dart
@override
Widget build(BuildContext context) {
  return Drawer(
    child: Column(
      children: [
        Container(
          height: 130,
          width: double.infinity,
          padding: EdgeInsets.all(20),
          alignment: Alignment.centerLeft,
          color: Colors.indigo, //χρωμα του header του drawer
          // decoration:BoxDecoration(color:Colors.blue), διαφορετικος τροπος να περασουμε χρωμα
          child: Text(
            "Menu",
            style: TextStyle(
              fontWeight: FontWeight.w900,
              fontSize: 30,
            ), // TextStyle
          ), // Text
        ), // Container
        buildListTile("Meals", Icons.restaurant, () {
          Navigator.of(context)
              .pushNamed('/meals'); //η συναρτηση που περναμε στον pageHandler
        }),
        SizedBox(
          height: 10,
        ), //για να αφησουμε ενα κενο // SizedBox
        buildListTile("Settings", Icons.settings, () {
          Navigator.pushNamed(context,
              '/settings'); //η συναρτηση που περναμε στον pageHandler
        }),
      ],
    ), // Column
  ); // Drawer
}
```



Ερωτήσεις

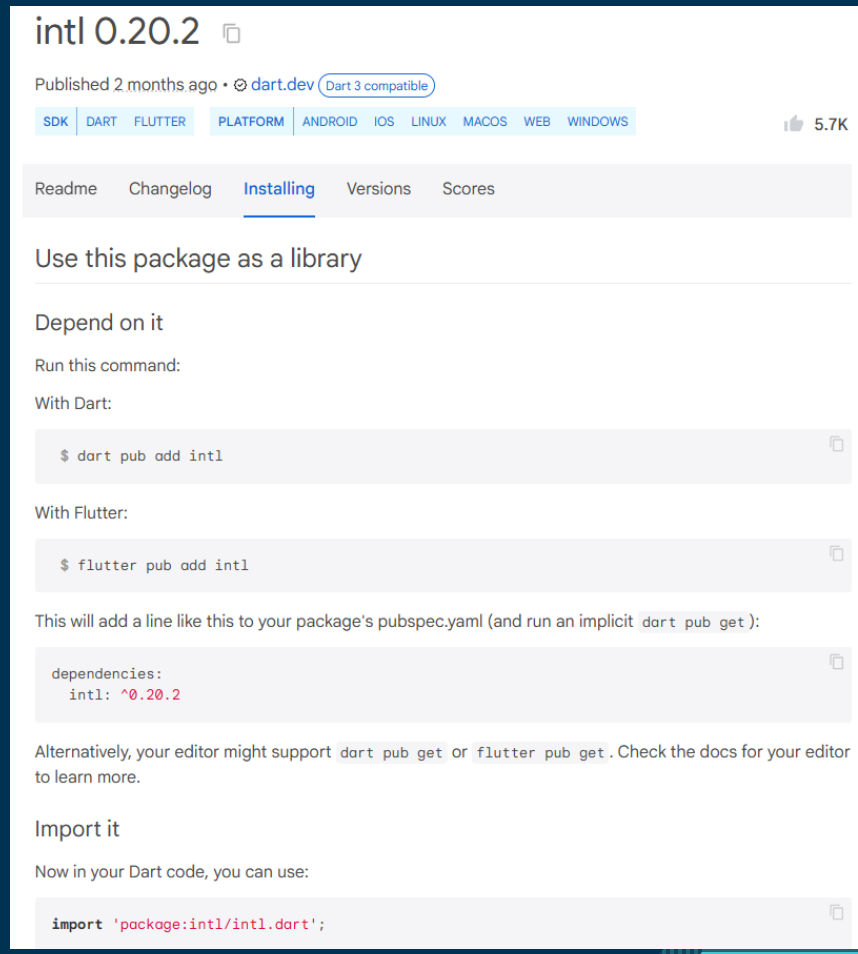
- Τι είναι η “Screen" σε μια εφαρμογή Flutter;
  - Ένα widget που ελέγχει ολόκληρη την οθόνη (ή τουλάχιστον αποτελεί το κύριο περιεχόμενο της οθόνης).
- Τι ισχύει για τα “Screens" και τα "κανονικά Widgets";
  - Και τα δύο είναι κανονικά widget στο τέλος, η μόνη διαφορά είναι ο τρόπος χρήσης των widget και ποιος ο ρόλος που παίζουν.
- Ποια είναι η διαφορά μεταξύ push() και pushNamed();
  - Το push() πλοηγεί σε μια νέα οθόνη δημιουργώντας την "on the fly", το pushNamed () μπορεί να φορτώσει μόνο οθόνες που έχουν καταχωρηθεί εκ των προτέρων.
- Τι ακριβώς είναι μια “route”;
  - Μια διαδρομή που είναι καταχωρημένη στον πίνακα διαδρομών (routes) - λαμβάνει ένα "όνομα" (key) με το οποίο μπορεί να φορτωθεί.
- Τι είναι το "Stack of Pages" (ή το "Stack of Screens");
  - Οι νέες σελίδες συνήθως προωθούνται πάνω από το "Stack of Pages/ Screens". Η κορυφαία (δηλαδή τελευταία) σελίδα / οθόνη είναι η ορατή οθόνη. Η αφαίρεση της τελευταίας οθόνης (Popping) επιστρέφει σε παλαιότερη οθόνη.

Ερωτήσεις

- Ποια είναι η διαφορά μεταξύ της χρήσης Tabs (ανεξάρτητα από ποια tabs) και της χρήσης push() / pushNamed();
  - Τα Tabs αντικαθιστούν την τρέχουσα οθόνη (ή μέρος αυτής) με μια νέα, τα push() / pushNamed() προσθέτουν μια νέα οθόνη στο stack.
- Ποιο widget είναι σημαντικό τόσο για τα Tabs όσο και για τα Drawers;
  - Το widget Scaffold - εισάγονται και τα δύο εκεί.

Installing External Packages

- Παράδειγμα: Format Date
  - Intl package → https://pub.dev/packages/intl
  - Pub.dev: Ιστοσελίδα με πολλά packages που μπορείτε να χρησιμοποιείτε στα projects του Flutter
  - Πατάμε “Installing” στη σελίδα και έχει οδηγίες
  - Κάνουμε copy ότι γραφεί κάτω από το dependencies και το προσθέτουμε στο pubspec.yaml κάτω ακριβώς από την λέξη flutter στο
    - dependencies:
      flutter
      intl: ^0.20.2
  - Σώζουμε το αρχείο και το flutter θα το εγκαταστήσει αυτόματα
  - Το κάνουμε import βάση των οδηγιών στην σελίδα “Installing”
  - Αν το package δεν εγκατασταθεί αυτόματα ανοίγουμε ένα terminal και τρέχουμε
    - flutter packages get



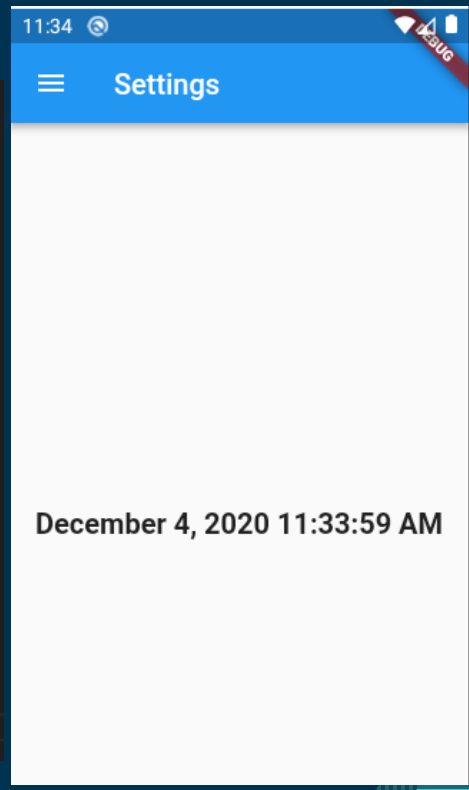
Date Format

- pubspec.yaml

```yaml
version: 1.0.0+1

environment:
  sdk: ">=2.7.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  intl: ^0.20.2

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.0

dev_dependencies:
  flutter_test:
    sdk: flutter
```

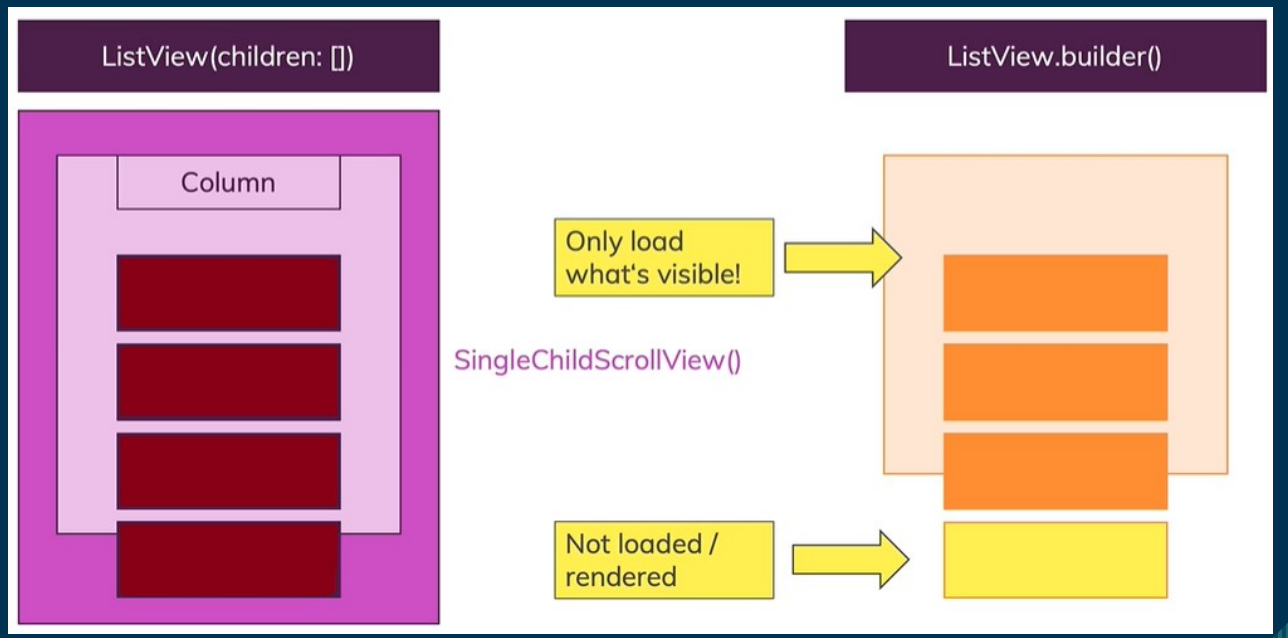
- settings.dart

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


class Settings extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title:Text("Settings")
      ), // AppBar
      drawer: MainDrawer(),
      body: Center(
        child: Text(
            DateFormat().format(DateTime.now()),
            style:TextStyle(fontSize: 20,fontWeight: FontWeight.bold),
        ), // Text
      ), // Center

    ); // Scaffold
  }
}
```



ListView Class

- ListView:
  - Infinity height
  - Αν είναι μέσα σε ένα container που ορίζουμε το height δεν θα έχουμε error
  - Δυο τρόποι να χρησιμοποιηθεί:
    - Με παράμετρο children
      - Renders όλα τα widgets (εμφανή και μη-εμφανή)
    - Με τον εξτρά constructor .builder()
      - Όταν δεν γνωρίζουμε ποσά αντικείμενα θα έχουμε στη λίστα ή είναι πολύ μεγάλος ο αριθμός των αντικειμένων
      - Renders μόνο τα εμφανή αντικείμενα



Future Class

- Url: https://api.flutter.dev/flutter/dart-async/Future-class.html
- Επιτρέπει την δημιουργία αντικειμένων που θα δοθεί τιμή στο μέλλον
- Τη χρησιμοποιούμε όταν θέλουμε να κάνουμε HttpRequests
- Το πρόγραμμα δεν θα περιμένει να ολοκληρωθεί η ανάθεση τιμής στο future αλλά θα εκτελεστεί κανονικά
- FutureBuilder: Παίρνει μια παράμετρο builder η οποία περιέχει την τωρινή κατάσταση του future

Futures

- Υπάρχουν δύο τρόποι για να εκτελέσετε ένα “Future" και να χρησιμοποιήσετε την τιμή που επιστρέφει (Εάν επιστρέψει οτιδήποτε). Ο πιο συνηθισμένος τρόπος είναι να περιμένουμε το “Future" να επιστρέψει δεδομένα. Για να λειτουργήσει αυτό η συνάρτηση που καλεί τον κώδικα πρέπει να επισημανθεί με “async".
  - Παράδειγμα:

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

- Ο άλλος τρόπος για να χειριστείτε ένα “future” είναι να χρησιμοποιήσετε τη συνάρτηση .then() . Παίρνει μια συνάρτηση που θα κληθεί με τον τύπο τιμής του future.
  - Παράδειγμα:

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

Synchronous and Asynchronous programming

- Σε ένα σύγχρονο μοντέλο προγραμματισμού, τα πράγματα συμβαίνουν ένα κάθε φορά. Όταν καλείτε μια συνάρτηση που εκτελεί μια μακροχρόνια ενέργεια, επιστρέφει μόνο όταν η ενέργεια έχει ολοκληρωθεί και μπορεί να επιστρέψει το αποτέλεσμα. Το πρόγραμμά σταματά για το χρόνο που εκτελείται η συνάρτηση και όταν ολοκληρωθεί η συνάρτηση “προχωράει” στην επόμενη ενέργεια/task.
- Ένα ασύγχρονο μοντέλο προγραμματισμού επιτρέπει πολλαπλά πράγματα να συμβούν ταυτόχρονα. Όταν ξεκινάει μια ενέργεια/task, το πρόγραμμά σας συνεχίζει να εκτελείται. Όταν ολοκληρωθεί η ενέργεια, το πρόγραμμα ενημερώνεται και αποκτά πρόσβαση στο αποτέλεσμα.

Read Local JSON file

- Χρησιμοποιούμε το DefaultAssetBundle για να διαβάσουμε το json αρχείο
- Χρησιμοποιούμε το ListView.builder() γιατί δεν γνωρίζουμε από πριν τον αριθμό των αντικειμένων
- Ο builder στην FutureBuilder κρατάει ένα snapshot της τωρινής κατάστασης (state) της future

```
return Scaffold(
    appBar: AppBar(
      title: Text("Load local JSON file"),
    ), // AppBar
    body: Container(
      child: Center(
        // Χρησιμοποιουμε το FutureBuilder και το Future για να κανουμε load το Json
        child: FutureBuilder(
            future: DefaultAssetBundle // το future μπορει να παρει στο μελλον τιμη
                .of(context)
                .loadString('data_repo/starwars_Data.json'),
            builder: (context, snapshot) { //κραταει ενα snapshot της τωρινης καταστασης του future
              // Decode the JSON
              var newData = json.decode(snapshot.data.toString());

              return ListView.builder(
                // Χρησιμοποιουμε τον constructor builder γιατι δεν ξερουμε το πληθος των αντικειμενων
                itemBuilder: (BuildContext context, int index) {
                  return Card(
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.stretch,
                      children: <Widget>[
                        Text("Name: " + newData[index]['name']),
                        Text("Height: " + newData[index]['height']),
                        Text("Mass: " + newData[index]['mass']),
                        Text(
                            "Hair Color: " + newData[index]['hair_color']), // Text
                        Text(
                            "Skin Color: " + newData[index]['skin_color']), // Text
                        Text(
                            "Eye Color: " + newData[index]['eye_color']), // Text
                        Text(
                            "Birth Year: " + newData[index]['birth_year']), // Text
                        Text("Gender: " + newData[index]['gender'])
                      ], // <Widget>[]
                    ), // Column
                  ); // Card
                },
                itemCount: newData == null ? 0 : newData.length, //ελεγχος αν η τιμη ειναι null αλλιως δινουμε
                //το πληθος των αντικειμενων της λιστας που θελουμε να γινουν build στο itemCount
              ); // ListView.builder
            }), // FutureBuilder
      ), // Center
    )); // Container // Scaffold
```