Hello World : Flutter

TP1 : Développement mobile, environnement flutter

Partie 1 : Création du projet de base

1. Création, génération et lancement d'un nouveau projet

Nous allons utiliser comme environnement de développement Android Studio, qui, couplé au SDK flutter permet de lancer l'application au choix sur un simulateur iPhone ou un device physique Android.

Vous allez donc créer un nouveau projet Flutter.

Par défaut Android Studio génère un code source permettant de lancer une application out of the box. Vous le trouverez dans le dossier lib sous le nom main.dart. Ce fichier source sera la base de toute application flutter.

Vous pouvez donc lancer le projet directement sur le simulateur ou un téléphone Android branché.

Prenez ensuite quelques minutes pour appréhender le code fourni ainsi que ses commentaires.

2. Concernant la console

Certains commentaires du code font référence à l'exécution de commandes flutter. Pour pouvoir les tester et les utiliser par la suite lancer un terminal (trouvable dans les applications ou via CMD + Espace).

3. Réalisation du HelloWorld

Nous allons réaliser notre premier Hello World. Pour ce faire, utilisez le code suivant en remplacant le code contenu dans le fichier main.dart.

```
import 'package:flutter/widgets.dart'; // basic set of widgets

// When Dart is running the application, it calls to the main() function
main() => runApp( // The function runApp() starts the Flutter application
  Text( // this is a widget, it renders the given text (think of it like a <span>)
    'Hello, World!!!', // the first argument is a text that needs to be rendered
    textDirection: TextDirection.ltr, // here we set the direction "left to right"
  ),
);
```

Cependant il pose problème si l'application est utilisée sur un iPhone X du fait du notch.

En vous aidant de la documentation sur le widget Center :
https://docs.flutter.io/flutter/widgets/Center-class.html

Rendez l'application utilisable sur un iPhone X.

D'autres widget existent et je vous invite à parcourir la documentation des widgets pour vous familiariser.

Partie 2 : Familiarisation avec les widgets

1. Création de notre propre Widget

Les widgets sont séparés en deux catégories en fonction de leurs états fixe (Stateless) ou dynamique (StateFull).

Exemple de widgets stateless : Text, Icon

Exemple de widgets statefull : Image, CheckBox

Vous trouverez ci-dessous un exemple de widget à implémenter pour utiliser les widgets perso.

```
class MyStatelessWidget extends StatelessWidget {
  // @override annotation is needed for optimization, by using it
  // we say that we don't need the same method from the parent class
  // so the compiler can drop it
  @override
  Widget build(BuildContext context) { // I'll describe [context]
later
    return Text('Hello!');
  }
}
class MyStatelessWidget extends StatelessWidget {
  // @override annotation is needed for optimization, by using it
  // we say that we don't need the same method from the parent class
  // so the compiler can drop it
  @override
  Widget build(BuildContext context) { // I'll describe [context]
later
    return Text('Hello!');
  }
}
```

A présent que vous avez un widget perso il faut pouvoir lui passer un paramètre.

Indices :

- Un attribut sera nécessaire afin de stocker la valeur passée lors de l'instanciation de votre widget.

- Le constructeur d'une classe est le meilleur point d'entrée pour spécifier une valeur d'attribut.

- Pour utiliser une variable ou attribut dans une chaine de caractère il suffit de l'appeler précédée d'un $.

2. Utilisation d'un widget au travers de la détection de gestes.

Nous allons débuter ici un nouveau projet.

Nous allons dans un premier créer notre interface avec un bouton cliquable au centre de celle-ci.

De la même manière que précédemment remplacer le code généré par cette première partie de code :

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

main() => runApp(
  Directionality(
    textDirection: TextDirection.ltr,
    child: Container(
      color: Color(0xFFFFFFFF),
      child: GestureApp(),
    ),
  ),
);
```

Nous allons à présent ajouter une classe du nom de GestureApp qui devra retourner l'arborescence suivante, chaque élément étant le parent du suivant :

- Un Widget Center
- Un Widget GestureDetector qui utilisera l'action ontap pour afficher un message dans la console. (Méthode print();)

- Enfin il faudra représenter le bouton à l'aide d'un Container via le code suivant :

```
Container(
   decoration: BoxDecoration(
     shape: BoxShape.circle,
     color: Color(0xFF17A2B8),
   ),
   width: 80.0,
   height: 80.0,
),
```

Partie 3 : Références

Documentations Flutter : https://flutter.dev/docs

Cookbook : https://flutter.dev/docs/cookbook

Exemples de codes : https://github.com/flutter/samples/blob/master/INDEX.md