

Développement d'applications mobiles

Chapitre 3 - Flutter - concepts de base

Daniel Schreurs

1^{er} octobre 2022

Haute École de la Province de Liège

Table des matières i

1. Introduction

In Flutter, everything is a widget.

2. Material Component

Scaffol

AppBar

floatingActionButton

bottomNavigationBar

3. Cupertino (Style iOS)

4. Détection de la plateforme

5. Médias

Icônes

FontAwesome

6. Images

Web

Statique

7. Polices de caractères

Introduction

Introduction : In Flutter, everything is a widget.

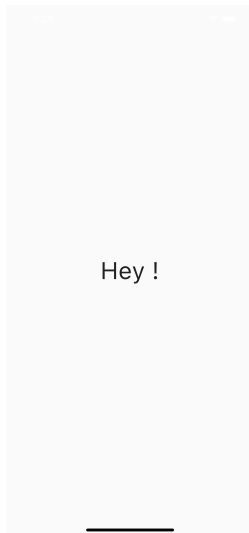
- Flutter est composé de widgets...beaucoup de widgets ;
- Un widget permet de décrire :
 - Une interface ;
 - Une fonctionnalité ;

Material Component

Material Component : Scaffold

- Le **Scaffold** est la structure de base des applications Android;
- Un ensemble pouvant regrouper plusieurs éléments comme :
 - Une *AppBar*;
 - Un *body*;
 - une *BottomNavigationBar*;
 - Et beaucoup d'autres encore...

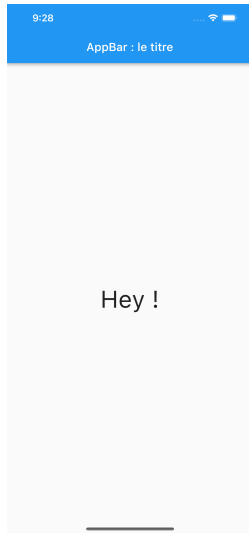
```
1 return Scaffold(  
2     body: Text("Hey"),  
3 );
```



Material Component : AppBar

- L'**AppBar** est un widget;
- Se situer en haut de l'écran;
- Enfant de *Scaffold*.

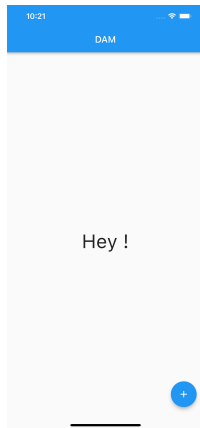
```
1  return Scaffold(  
2      appBar: AppBar(  
3          title: Text(  
4              widget.title  
5          ),  
6      body: Text("Hey"),  
7  );
```



Material Component : FloatingActionButton

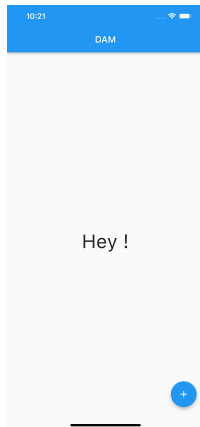
- Le `FloatingActionButton` est un widget;
- Consiste en un bouton flottant;
- Qui peut être placé : `centerDocked`, `bottomNavigationBar`, `centerFloat`, `endDocked`, `bottomNavigationBar`, `bottomNavigationBar`, `endTop`, `miniStartTop`, `startTop`.

```
1 FloatingActionButton:  
  FloatingActionButton(  
2    onPressed: () => {print("  
    Bonjour")},  
3    child: const Icon(Icons.  
      add),  
4  ),
```



Material Component : bottomNavigationBar

- La `bottomNavigationBar` est une barre de navigation ;
- Située en bas de l'écran ;
- Par défaut, sa taille est nulle ;
- Il convient d'y inclure un Container par exemple ;



Material Component : bottomNavigationBar

```
1 floatingActionButtonLocation:
    FloatingActionButtonLocation.centerDocked,
2 bottomNavigationBar: BottomAppBar(
3     shape: CircularNotchedRectangle(),
4     child: Container(
5         height: 50,
6         child: Row(
7             mainAxisAlignment: MainAxisAlignment.spaceAround
8             ,
9             children: [
10                IconButton(
11                    onPressed: pressed,
12                    icon: Icon(Icons.add_location),
13                ),
14                IconButton(
15                    onPressed: pressed,
16                    icon: Icon(Icons.forward),
17            ),],),),),),
```

Cupertino (Style iOS)

- Style présent sur iOS ;
- Nom donné en référence à la ville où se situe le siège d'Apple ;

Nous ne couvrons pas ces widgets.

Détection de la plateforme

- Pour se fondre dans le style de son système d'exploitation hôte, il convient alors de lui faire posséder les deux styles : Material Design et Cupertino ;
- Créer deux Widgets, un pour Android et l'autre pour iOS ;
- Il faut importer `Platform`.

Détection de la plateforme

Ces propriétés retournent `true` si la plateforme est détectée au runtime.

- `Platform.isAndroid` pour détecter si le programme s'exécute sur;
- `Platform.isFuchsia` pour détecter si le programme s'exécute sur Fuchsia;
- `Platform.isIOS` pour détecter si le programme s'exécute sur iOS;
- `Platform.isLinux` pour détecter si le programme s'exécute sur Linux;
- `Platform.isMacOS` pour détecter si le programme s'exécute sur macOS.

Détection de la plateforme

```
1 @override
2 Widget build(BuildContext context) {
3   return Platform.isIOS ? myCupertinoApp() :
4     myCupertinoApp();
5 }
```

myCupertinoApp et myCupertinoApp étant des classes Widget personnelles.

Médias

- Il est donc primordial de savoir intégrer des médias ;
- Les icônes sont l'un des outils les plus employés de nos jours ;
- Elles permettent d'obtenir une compréhension rapide simplement au travers d'un visuel ;
- Des conventions se sont mises en place.

Médias : Icônes

- L'insertion de l'icône peut être réalisée au travers du widget `Icon`;
- Ce dernier permet :
 - De définir la taille;
 - Sa couleur.
- Flutter offre un vaste choix d'icônes;
- Soit en travaillant avec la classe `Icons` qui comporte une collection au style Material Design;
- soit avec la classe `CupertinoIcons` qui propose un style iOS.

```
1 const Icon(  
2     Icons.notifications,  
3     size: 16,  
4     color: kMainTextColor,  
5 ),
```

Il est possible de retrouver l'ensemble de ces icônes
material.io/resources/icons

- [FontAwesome](#) est un site Internet bien connu des développeurs web.
- Il offre une collection très importante d'icônes en tout genre.

Médias : FontAwesome

The screenshot shows the Font Awesome website homepage. At the top is a navigation bar with links: Start, Search, Icons, Docs, Plans, Support, and Blog. Below the navigation bar is a purple promotional banner for 'Start Miss Out!' regarding a new Pro subscription. The main content area features the headline 'Take the hassle out of icons in your website.' and a sub-headline stating 'Font Awesome is the Internet's icon library and toolkit, used by millions of designers, developers, and content creators.' Below this are two buttons: 'Start for Free' and 'Get \$20 Off Pro'. A video player shows a man in a blue apron with the Font Awesome logo working in a cafe. The page is decorated with a grid of various icons. At the bottom, there is a 'LATEST UPDATES' section with the text 'Discover what's new in Font Awesome' and a row of icons representing different features or updates.

Start Miss Out! To celebrate the release of Font Awesome Shop, get \$20 off a new Pro subscription for a limited time!

Take the hassle out of icons in your website.

Font Awesome is the Internet's icon library and toolkit, used by millions of designers, developers, and content creators.


[Start for Free](#) [Get \\$20 Off Pro](#)

Awesome

LATEST UPDATES

Discover what's new in Font Awesome

Médias : FontAwesome



font_awesome_flutter 10.2.1

Published 18 days ago · @ fluttercommunity-dev · Not verified

SDK | FLUTTER | PLATFORM | ANDROID | IOS | LINUX | MACOS | WEB | WINDOWS

Readme · Changelog · Example · Installing · Versions · Scores

font_awesome_flutter

Flutter Community · font_awesome_flutter

Michael Spies (@michaelspies)

See also:

The free **Font Awesome** icon pack available as set of Flutter icons - based on font awesome version 6.2.0.

This icon pack includes only the free icons offered by Font Awesome out-of-the-box. If you have purchased the pro icons and want to enable support for them, please see the instructions below.

Installation

In the `dependencies` section of your `pubspec.yaml`, add the following line:

```
dependencies {
  font_awesome_flutter: ^latest_version
}
```


Usage

```
import 'package:font_awesome_flutter/font_awesome_flutter.dart';

class MyWidget extends StatelessWidget {
  Widget build(BuildContext context) {
    return Icon(
      // Use the Flutter Widget = FontAwesomeIcon class for the IconData
      Icons: FontAwesomeIcons.diamond,
      color: Colors.purple(0.8),
    );
  }
}
```

Icon names

Icon names equal those on the [official website](#), but are written in lower camel case. If more than one icon style is available for an icon, the style name is used as prefix, except for "regular". Due to restrictions in dart, icons starting with numbers have those numbers written out.



2991 140 100+

likes · packages · community

Publisher

@fluttercommunity-dev

Metadata

The Font Awesome icon pack available as Flutter icons. Provides 1500+ additional icons to use in your apps.

Repository (GitHub)

[View/report issues](#)

Documentation

[API reference](#)

License

© MIT (LICENSE)

Dependencies

Flutter

More

[Packages that depend on font_awesome_flutter](#)

Images

- Pour rendre une application agréable en termes d'expérience utilisateur, il est courant d'insérer des images.
- Pour cela, plusieurs constructeurs nommés sont mis à disposition.
- Ils correspondent aux différents types de provenance d'une image, celles incluses au projet ou celles venant directement du web.

Images : Web

```
1 ClipRRect(  
2   borderRadius: kBorderRadiusItem,  
3   child: Image.network(  
4       'https://image.tmdb.org/t/p/w154/$path',  
5       height: 230,  
6       width: 154,  
7   ),  
8 ),
```

```
1 Image.asset(  
2     'assets/img/actor_placeholder.jpg',  
3     height: 190,  
4     width: 154,  
5     fit: BoxFit.fitWidth,  
6 );
```

Attention

Il faut impérativement charger l'image dans l'application dans le fichier *pubspec.yaml*.

Polices de caractères

Polices de caractères

Pour charger une police de caractères, il faut :

- Télécharger les fichiers dans un dossier du projet;
- Déclarer la police de caractères dans le fichier *pubspec.yaml*;
- Utiliser la police de caractères dans un widget spécifique.

```
1 awesome_app/  
2   fonts/  
3     Raleway-Regular.ttf  
4     Raleway-Italic.ttf  
5     RobotoMono-Regular.ttf  
6     RobotoMono-Bold.ttf
```

Définir une police par défaut

```
1 return MaterialApp(  
2   title: 'Custom Fonts',  
3   // Ici  
4   theme: ThemeData(fontFamily: 'Raleway'),  
5   home: const MyHomePage(),  
6 );
```