Сучасні технології мобільного програмування

Використання Flutter для кросплатформної розробки мобільних застосунків

Слайди до лекцій

Що таке Flutter?

- Open-source SDK для розробки застосунків від Google.
- Дозволяє створювати Android, iOS, Web та Desktop застосунки з єдиного кодової бази.
- Мова програмування: Dart.

Переваги Flutter

- Швидкий розробницький цикл (‘Hot Reload’).
- Висока перформансність.
- Безшовна крос-платформність.
- Широкий вибір готових віджетів.

Ключові компоненти Flutter

- **Widgets:** Основа взаємодії Flutter.
- **Dart:** Основна мова Flutter.
- **Flutter Engine:** Обробляє графіку та анімацію.
- **Foundation Library:** Базові компоненти для застосунків.

Встановлення Flutter SDK

- Завантажити Flutter SDK з офіційного сайту.
- Розпакувати в будь-яке місце.
- Додати Flutter до PATH.
- Запустити команду flutter doctor.

Налаштування Android Studio та XCode

- **Android Studio:**
  - Завантажити та встановити Android Studio.
  - Додатково встановити Flutter та Dart Plugins (включення у File > Settings > Plugins).
  - Перевірити Android SDK та емулятори.
- **XCode:**
  - Завантажтить та встановити XCode з App Store.
  - Додати XCode Command Line Tools (запуск команди xcode-select --install).
  - Перевірити iOS Simulator.

Приклади з віджетами та лейаутами

- Container та Row:

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('Flutter Layout')),
        body: Column(
          ...
```

Приклади з віджетами та лейаутами

```
...
          children: [
            Container(
              color: Colors.blue,
              height: 100,
              width: double.infinity,
              child: const Center(child: Text('Container
Example', style: TextStyle(color: Colors.white))),
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceAround,
              children: const [
                Icon(Icons.star, color: Colors.red),
                Icon(Icons.star, color: Colors.green),
                Icon(Icons.star, color: Colors.blue),
              ],
            ),
          ],
        ),
      ),      );  } }
```

Приклади з віджетами та лейаутами

- ListView:

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('ListView Example')),
        body: ListView(
...
```

Приклади з віджетами та лейаутами

```
...
          children: const [
            ListTile(title: Text('Item 1')),
            ListTile(title: Text('Item 2')),
            ListTile(title: Text('Item 3')),
          ],
        ),
      ),
    );
  }
}
```

Приклади із основними лейаутами

- GridView:

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('GridView Example')),
        body: GridView.count(
          crossAxisCount: 2,
          children: List.generate(4, (index) {
```

...

Приклади із основними лейаутами

```
...
              return Card(
                color: Colors.amber,
                child: Center(child: Text('Item $index')),
              );
            }),
          ),
        ),
      );
    }
}
```

Створення власних віджетів

- Створення нового класу, який успадковує StatelessWidget або StatefulWidget.
- Використання методу build для побудови структури віджету.
- Приклад:

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
 ...
```

Створення власних віджетів

```
...
@override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('Custom Widget
Example')),
        body: const Center(child: CustomWidget()),
      ),
    );
  }
}
 ...
```

Створення власних віджетів

```
...
class CustomWidget extends StatelessWidget {
  const CustomWidget({super.key});

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.all(16),
      color: Colors.blue,
      child: const Text(
        'This is a custom widget!',
        style: TextStyle(color: Colors.white),
      ),
    );
  }
}
```

Інтеграція з API

- Використання бібліотек, як-от http для роботи з мережевими запитами.
- Проста реалізація GET-запиту:

```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
...
```

Інтеграція з API

```
...
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('API Integration')),
        body: const ApiExample(),
      ),
    );
  }
}

class ApiExample extends StatefulWidget {
  const ApiExample({super.key});

  @override
  _ApiExampleState createState() => _ApiExampleState();
}
...
```

Інтеграція з API

```
 ...
class _ApiExampleState extends State<ApiExample> {
  String data = 'Loading...';

  @override
  void initState() {
    super.initState();
    fetchData();
  }

  Future<void> fetchData() async {
    final response =
        await
http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts/
1'));
    if (response.statusCode == 200) {
      setState(() {
        data = jsonDecode(response.body)['title'];
      });
...
```

Інтеграція з API

```
...
    } else {
      setState(() {
        data = 'Error loading data';
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Center(child: Text(data));
  }
}
```

Навігація між екранами

- Використання Navigator для переходу між екранами.
- Приклад реалізації:

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
…
```

Навігація між екранами

…

```
      routes: {
        '/': (context) => const FirstScreen(),
        '/second': (context) => const SecondScreen(),
      },
    );
  }
}

class FirstScreen extends StatelessWidget {
  const FirstScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('First Screen')),
      body: Center(
        child: ElevatedButton(
```

…

Навігація між екранами

```
...
          onPressed: () {
            Navigator.pushNamed(context, '/second');
          },
          child: const Text('Go to Second Screen'),
        ),
      ),
    );
  }
}
 ...
```

Навігація між екранами

```
...
class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Screen')),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text('Back to First Screen'),
        ),
      ),
    );
  }
}
```

Управління станом

- Основні підходи:
  - setState: Локальне управління станом.
  - Provider: Рекомендується для складних проектів.
  - Інші: Riverpod, Bloc, Redux.
- Приклад використання setState:

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
 ...
```

Управління станом

```
...
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: CounterScreen(),
    );
  }
}

class CounterScreen extends StatefulWidget {
  @override
  _CounterScreenState createState() => _CounterScreenState();
}
...
```

Управління станом

```
...
class _CounterScreenState extends State<CounterScreen> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Counter Example')),
      body: Center(
        child: Text('Counter: $_counter', style: const
TextStyle(fontSize: 24)),
      ),
...
```

Управління станом

```
• • •
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

Перспективи розвитку Flutter

- **Активна підтримка Google:** Регулярні оновлення та вдосконалення.
- **Зростаюча популярність:** Велика спільнота розробників і багатий вибір плагінів.
- **Нові можливості:**
  - Flutter для web-додатків і десктопів.
  - Інтеграція з Fuchsia OS (майбутня операційна система Google).
- **Попит на ринку:**
  - Зростання вакансій для Flutter-розробників.
  - Використання великими компаніями (Alibaba, BMW, Google Ads).
- **Гнучкість та масштабованість:**
  - Інструмент для створення застосунків будь-якої складності.
  - Підтримка різних платформ з єдиним кодом.