cjundang restruct

616cbc2 · 13 hours ago

สรุปเนื้อหาจาก Lecture 6 & Lecture 7 (React Native)

1. การสร้าง ListView ด้วย FlatList

ที่อยู่เนื้อหา:

- Lecture 6 → CRUD บน To-do List App
- Lecture 7 → แสดงบทเรียน EduLesson

หลักการ

- ใช้ **FlatList** สำหรับสร้าง UI ที่แสดงข้อมูลหลายรายการ
- ทำงานแบบ **Virtualized List** คือ แสดงเฉพาะข้อมูลที่อยู่บนหน้าจอเพื่อให้แอปมีประสิทธิภาพ

ตัวอย่างโค้ด — To-do List

```
<FlatList
  data={todos}  // Array เก็บข้อมูลทั้งหมด
  keyExtractor={(item) => item.id.toString()}
  renderItem={({ item }) => (
    <View style={styles.todoItem}>
      <Text>{item.title}</Text>
    </View>
  )}
/>
```

ตัวอย่างโค้ด — EduLesson List

```
<FlatList
  data={lessons}
  keyExtractor={(item, idx) => item.title + ':' + idx}
  renderItem={({ item }) => (
    <TouchableOpacity onPress={() => navigation.navigate('Detail', { unit:
      <Image source={{ uri: item.imageUrl }} style={{ width: 40, height: 40
      <View>
        <Text>{item.title}</Text>
        <Text>{item.subtitle}</Text>
      </View>
    </TouchableOpacity>
  )}
/>
```

---

2. การรับข้อมูลจากผู้ใช้ (User Input)

หลักการ

- ใช้ TextInput สำหรับให้ผู้ใช้กรอกข้อมูล
- กำหนดค่า state เพื่อผูกค่ากับอินพุต

ตัวอย่างโค้ด

```
const [inputText, setInputText] = useState('');

<TextInput
  value={inputText}
  onChangeText={setInputText}
  style={styles.input}
  placeholder="Enter a task"
/>

<TouchableOpacity onPress={addTodo} style={styles.button}>
  <Text style={styles.buttonText}>Add</Text>
</TouchableOpacity>
```

กระบวนการ

1. ผู้ใช้กรอกข้อมูลใน TextInput
2. กดปุ่ม Add
3. เรียกฟังก์ชัน `addTodo()` → เพิ่มข้อมูลใหม่ไปยัง state และบันทึกลง AsyncStorage

3. การบันทึกข้อมูลใน Local Storage ด้วย AsyncStorage

ที่อยู่เนื้อหา: Lecture 6 → CRUD Operations

หลักการ

- ใช้ AsyncStorage สำหรับเก็บข้อมูลแบบ Key-Value
- ต้อง serialize ข้อมูลก่อนบันทึก และ parse กลับมาใช้งาน

ตัวอย่างโค้ด

บันทึกข้อมูล

```
await AsyncStorage.setItem('todos', JSON.stringify(updatedTodos));
```

อ่านข้อมูล

```
const data = await AsyncStorage.getItem('todos');
if (data) setTodos(JSON.parse(data));
```

ลบข้อมูล

```
const filtered = todos.filter(todo => todo.id !== id);
setTodos(filtered);
await AsyncStorage.setItem('todos', JSON.stringify(filtered));
```

จุดเด่น

- ใช้ JSON เพื่อแปลง array ↔ string
- รักษาข้อมูลไว้แม้ผู้ใช้ปิดแอป
- ทำงานแบบ Asynchronous

---

4. Navigator แบบ Stack + การส่งค่าไปยังหน้าถัดไป

ที่อยู่เนื้อหา: Lecture 7 → Navigation Fundamentals

4.1 การสร้าง Stack Navigator

ติดตั้ง dependencies:

```
npm install @react-navigation/native
npm install @react-navigation/native-stack
```

โค้ดตั้งค่า Navigator:

```
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack'

const Stack = createNativeStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="Detail" component={DetailScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

4.2 การส่งค่าระหว่าง Screens

ส่งค่าจาก Home → Detail

```
navigation.navigate('Detail', { unitId: item.id });
```

รับค่าที่หน้า Detail

```
const { unitId } = route.params;
```

4.3 การใช้ Replace / Reset

- `navigation.replace('Login')` → แทนที่หน้าปัจจุบัน
- `navigation.reset({...})` → ล้าง stack แล้วเริ่มใหม่

Lecture 7 ยังสอน

- การเปลี่ยน Header Title แบบ dynamic:

```
<Stack.Screen
  name="Detail"
```

```
  component={DetailScreen}
  options={({ route }) => ({ title: route.params.unitId })}
/>
```

---

5. Web API: การอ่านและแสดงข้อมูล

ในสองไฟล์นี้ **ไม่ได้เจาะลึก Web API** แต่จากเนื้อหา To-do App และ EduLesson App สามารถเพิ่มการเชื่อมต่อ API ได้ง่าย ๆ ด้วย **fetch()** หรือ **axios** เพื่ออ่าน/เขียนข้อมูลจาก Server

ตัวอย่างโค้ดเรียก API

```
import { useEffect, useState } from 'react';

export default function App() {
  const [data, setData] = useState([]);

  useEffect(() => {
    fetch('https://jsonplaceholder.typicode.com/posts')
      .then(res => res.json())
      .then(json => setData(json));
  }, []);

  return (
    <FlatList
      data={data}
      keyExtractor={(item) => item.id.toString()}
      renderItem={({ item }) => <Text>{item.title}</Text>}
    />
  );
}
```

สรุปเนื้อหาที่เน้น

ฟีเจอร์	Lecture 6	Lecture 7
ListView	FlatList สำหรับ To-do	FlatList สำหรับบทเรียน + รูปภาพ + Icon
User Input	ใช้ TextInput เพื่อเพิ่ม To-do	ไม่เน้น
Local Storage	บันทึกข้อมูลด้วย AsyncStorage	ไม่เน้น

ฟีเจอร์	Lecture 6	Lecture 7
Navigator (Stack)	ไม่ครอบคลุม	ครอบคลุมเต็มรูปแบบ
Web API	ไม่ได้ใช้จริง	ไม่ได้ใช้จริง แต่สามารถเสริมการเชื่อมต่อ