Super Moustapha ! Voici le cours **“Leçon 4 : Listes et gestion des événements”** que tu pourras intégrer dans ta formation React Native. Je te structure le tout avec des explications claires, un exemple de code, puis un exercice pratique.

📜 Leçon 4 : Listes et gestion des événements

✅ Objectifs :

- Afficher des listes dynamiques avec `FlatList` et `ScrollView`.
- Gérer les interactions utilisateur avec `onPress`.
- Ajouter dynamiquement des éléments avec `TextInput`.
- Mettre à jour l’état global avec `useState`.

🔸 1. `FlatList` vs `ScrollView`

- `ScrollView` est utilisé pour des contenus **fixes et courts** (toute la liste est rendue d’un coup).
- `FlatList` est plus performant pour **des listes longues ou dynamiques**, car il rend les éléments au fur et à mesure.

```tsx
import { ScrollView, Text } from 'react-native';

<ScrollView>
  <Text>Élément 1</Text>
  <Text>Élément 2</Text>
</ScrollView>
```

```tsx
import { FlatList, Text } from 'react-native';

<FlatList
  data={[{ id: '1', title: 'Tâche A' }, { id: '2', title: 'Tâche B' }]}
  keyExtractor={(item) => item.id}
```

```
  renderItem={({ item }) => <Text>{item.title}</Text>}
/>
```

---

2. Gérer les événements avec `onPress`

```tsx
import { TouchableOpacity, Text } from 'react-native';

<TouchableOpacity onPress={() => alert("Tâche cliquée !")}>
  <Text>Appuie ici</Text>
</TouchableOpacity>
```

---

3. Ajouter une tâche avec `TextInput` et `useState`

```
import React, { useState } from 'react';
import { View, TextInput, FlatList, TouchableOpacity, Text, StyleSheet } from 'react-native';

export default function TodoListScreen() {
  const [task, setTask] = useState('');
  const [tasks, setTasks] = useState([]);

  const addTask = () => {
    if (task.trim()) {
      setTasks([...tasks, { id: Date.now().toString(), title: task }]);
      setTask('');
    }
  };

  return (
    <View style={styles.container}>
      <TextInput
        placeholder="Ajouter une tâche..."
        value={task}
        onChangeText={setTask}
        style={styles.input}
```

```
        />
        <TouchableOpacity style={styles.button} onPress={addTask}>
          <Text style={styles.buttonText}>Ajouter</Text>
        </TouchableOpacity>

        <FlatList
          data={tasks}
          keyExtractor={(item) => item.id}
          renderItem={({ item }) => (
            <View style={styles.taskItem}>
              <Text>{item.title}</Text>
            </View>
          )}
        />
      </View>
    );
  }

  const styles = StyleSheet.create({
    container: {
      flex: 1,
      padding: 20,
      backgroundColor: '#f5f5f5',
    },
    input: {
      borderWidth: 1,
      borderColor: '#ccc',
      padding: 12,
      borderRadius: 8,
      marginBottom: 10,
      backgroundColor: '#fff',
    },
    button: {
      backgroundColor: '#007bff',
      padding: 12,
      borderRadius: 8,
      alignItems: 'center',
      marginBottom: 20,
    },
    buttonText: {
      color: '#fff',
      fontWeight: 'bold',
    },
```

```
    taskItem: {
      padding: 12,
      marginBottom: 10,
      backgroundColor: '#e0e0e0',
      borderRadius: 8,
    },
  });
```

📝 Exercice : Ajouter une tâche à la liste

Objectif :

- Crée un écran où l'utilisateur peut :
  - Entrer une tâche via un champ `TextInput` .
  - Cliquer sur un bouton "Ajouter".
  - Voir la tâche ajoutée à une liste (`FlatList`).
- Bonus : afficher une alerte quand on appuie sur une tâche.

Si tu veux, je peux te fournir une version Snack prête à tester ou l'ajouter dans ta structure de cours au format Markdown ou PDF.

Souhaites-tu aussi la **Leçon 5 : Suppression et édition d'éléments de liste** pour la suite ?