COURS 1 : Introduction à React Native et mise en place du projet

Bienvenue dans cette première séance de formation ! Dans ce module, nous allons explorer les bases de React Native, découvrir l'écosystème Expo, comprendre l'intérêt de ES6, et mettre en place votre premier projet mobile. Vous apprendrez également à afficher un simple texte sur l'écran. Ce cours détaillé inclut de nombreux exemples, explications étape par étape, et liens utiles pour approfondir vos connaissances.

---

1. Introduction à React Native

1.1 Qu'est-ce que React Native ?

React Native est un framework développé par Facebook qui permet de créer des applications mobiles natives en utilisant JavaScript et React.

Principaux avantages :

- **Développement cross-platform :** Un même code peut fonctionner sur iOS et Android.
- **Performance native :** Les composants de base sont rendus en utilisant des composants natifs.
- **Hot Reloading :** Permet de visualiser instantanément les modifications de code sans redémarrage complet de l'application.

1.2 Différences avec React pour le Web

- **DOM vs Composants Natifs :** En React web, vous manipulez le DOM. En React Native, vous utilisez des composants tels que `View`, `Text` et `Image` qui se traduisent en éléments natifs.
- **Style :** Au lieu de CSS, vous utilisez des objets JavaScript via `StyleSheet` pour définir les styles.
- **Navigation :** La navigation se fait souvent avec des bibliothèques comme React Navigation au lieu de React Router.

Exemple comparatif :

React (Web)

```jsx
// Utilisation d'un élément <div> et d'une classe CSS
<div className="container">
  <p>Bienvenue sur mon site web</p>
</div>
```

React Native

```jsx
// Utilisation d'un élément <View> et d'un style via StyleSheet
import { StyleSheet, Text, View } from 'react-native';

<View style={styles.container}>
  <Text>Bienvenue sur mon application mobile</Text>
</View>

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
});
```

---

2. Introduction à Expo

2.1 Qu’est-ce qu’Expo ?

Expo est un ensemble d’outils et de services qui facilite le développement avec React Native.

Pourquoi utiliser Expo ?

- **Configuration simplifiée :** Pas besoin de configurer Xcode ou Android Studio pour démarrer.
- **API prêtes à l’emploi :** Accès à de nombreuses fonctionnalités natives (caméra, notifications, etc.) sans configuration supplémentaire.

- **Outils de développement :** Expo CLI, Expo Go (pour tester sur un appareil mobile) et des services de build.

2.2 Avantages d'Expo

- **Démarrage rapide :** Créez et lancez une application en quelques minutes.
- **Mises à jour OTA (Over The Air) :** Déployez des mises à jour sans passer par les stores.
- **Communauté active :** Une large base d'utilisateurs et une documentation complète.

---

3. Présentation de Javascript ES6 et son intérêt (pour Débutants)

1. Les Variables et l'Affectation

En ES6, on utilise `let` et `const`.

- `let` permet de déclarer une variable modifiable.
- `const` permet de déclarer une constante (non modifiable).

Exemple :

```javascript
let age = 25;
age = 26; // Ok

const pi = 3.14;
pi = 3.1415; // Erreur ! Impossible de modifier une constante.
```

2. Les Objets

Un objet est une collection de propriétés.

Exemple :

```javascript
const user = {
  name: "Alice",
```

```
  age: 30,
};
```

3. Les Tableaux et Fonctions Utiles

Un tableau permet de stocker plusieurs valeurs dans une seule variable.

Exemple :

```javascript
const fruits = ["Pomme", "Banane", "Orange"];
console.log(fruits[0]); // Pomme
```

Fonctions utiles pour manipuler les tableaux

- `.push()` : ajoute un élément à la fin du tableau.
- `.pop()` : supprime le dernier élément.
- `.shift()` : supprime le premier élément.
- `.unshift()` : ajoute un élément au début.
- `.map()` : transforme chaque élément du tableau.
- `.filter()` : filtre les éléments selon une condition.
- `.reduce()` : applique une fonction sur tous les éléments pour obtenir une valeur unique.
- `.splice(index, count)` : enlève `count` éléments à partir de `index`.
- `.concat()` : fusionne deux tableaux.

Exemples :

```javascript
const nombres = [1, 2, 3, 4, 5];
nombres.push(6); // [1, 2, 3, 4, 5, 6]
nombres.pop(); // [1, 2, 3, 4, 5]

const doubles = nombres.map(n => n * 2);
console.log(doubles); // [2, 4, 6, 8, 10]

const pairs = nombres.filter(n => n % 2 === 0);
console.log(pairs); // [2, 4]

const somme = nombres.reduce((acc, n) => acc + n, 0);
```

```
console.log(somme); // 15

// Supprimer un élément précis
const index = nombres.indexOf(3);
if (index !== -1) {
  nombres.splice(index, 1);
}
console.log(nombres); // [1, 2, 4, 5]

// Ajouter un élément sans modifier le tableau original (utile en React)
const newNombres = [...nombres, 7];
console.log(newNombres); // [1, 2, 4, 5, 7]
```

4. Les Fonctions et les Procédures

Une fonction retourne une valeur tandis qu'une procédure exécute une action sans retourner de valeur.

Fonction (avec `return`) :

```javascript
function addition(a, b) {
  return a + b;
}
console.log(addition(5, 3)); // 8
```

Procédure (sans `return`) :

```javascript
function afficherMessage(message) {
  console.log(message);
}
afficherMessage("Bonjour !");
```

5. Les Fonctions Flèches (Arrow Functions)

Les fonctions flèches permettent une syntaxe plus concise.

```javascript
const multiplier = (a, b) => a * b;
console.log(multiplier(4, 5)); // 20
```

6. Les Template Strings

Les backticks `` permettent d'insérer des variables directement dans une chaîne.

```javascript
const nom = "Bob";
console.log(`Bonjour ${nom} !`);
```

7. Les Promises et `async/await`

Les Promises

Une `Promise` permet de gérer des opérations asynchrones.

```javascript
const fetchData = () => {
  return new Promise((resolve, reject) => {
    setTimeout(() => resolve("Données chargées"), 2000);
  });
};
fetchData().then(data => console.log(data));
```

`async/await`

Une alternative plus lisible aux `Promises` .

```javascript
const fetchDataAsync = async () => {
  const data = await fetchData();
  console.log(data);
};
fetchDataAsync();
```

8. La Déstructuration

Permet d'extraire facilement des valeurs d'un objet ou d'un tableau.

```javascript
const user = { name: "Charlie", age: 35 };
const { name, age } = user;
console.log(name, age);
```

9. Le Spread Operator (`...`)

Permet de copier ou fusionner des objets/tableaux.

Avec un tableau

```javascript
const numbers = [1, 2, 3];
const newNumbers = [...numbers, 4, 5];
console.log(newNumbers); // [1, 2, 3, 4, 5]
```

Avec un objet

```javascript
const user = { name: "Alice", age: 30 };
const updatedUser = { ...user, age: 31, city: "Paris" };
console.log(updatedUser); // { name: "Alice", age: 31, city: "Paris" }
```

10. Les Modules (import/export)

Pour organiser le code en fichiers distincts.

Fichier `math.js`

```javascript
export const add = (a, b) => a + b;
```

Fichier `main.js`

```javascript
import { add } from './math.js';
console.log(add(2, 3));
```

11. Les Classes

Syntaxe orientée objet en ES6.

```javascript
class Person {
  constructor(name, age) {
    this.name = name;
```

```
    this.age = age;
  }
  greet() {
    console.log(`Bonjour, je suis ${this.name}`);
  }
}
const alice = new Person("Alice", 25);
alice.greet();
```

Ce sont les concepts fondamentaux d'ES6 que tout développeur React devrait maîtriser !

4. Installation de l'environnement de développement

4.1 Prérequis

- **Node.js et npm :** Assurez-vous d'avoir Node.js installé (npm est inclus).
- **Visual Studio Code :** Un éditeur de code performant tel que VS Code.

4.2 Installation d'Expo CLI

```
npm install -g expo-cli
```

4.3 Vérification de l'installation

```
expo --version
```

---

5. Création du projet The Todo List App avec Expo

5.1 Initialiser le projet

```
expo init TheTodoListApp
```

Sélectionnez le template **blank (TypeScript)**.

5.2 Lancer le projet

```
cd TheTodoListApp
expo start
```

Scannez le QR code affiché avec l'application Expo Go pour tester l'application.

---

6. Premier écran : affichage d'un simple texte

6.1 Modification du fichier `App.tsx`

tsx

```tsx
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text style={styles.welcomeText}>Bienvenue sur The Todo List App !</Text>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#f0f0f0',
    alignItems: 'center',
    justifyContent: 'center',
  },
  welcomeText: {
    fontSize: 20,
    fontWeight: 'bold',
```

```
    color: '#333',
  },
});
```

---

7. TP : Mise en place du projet et affichage d'un message de bienvenue

Objectif du TP

Créer un projet **The Todo List App** en utilisant Expo, et afficher un message de bienvenue personnalisé.

Étapes à suivre

1. **Installation des outils**

   ```
   npm install -g expo-cli
   ```

2. **Création du projet**

   ```
   expo init TheTodoListApp
   ```

3. **Démarrer le projet**

   ```
   cd TheTodoListApp
   expo start
   ```

4. **Modification de** `App.jsx` avec le code de la section précédente.

Rendu du TP

- **Un dépôt Git** contenant votre projet.
- **Un fichier README.md** décrivant le projet.

8. Conclusion et prochaines étapes

Ce que vous avez appris aujourd'hui :

- Les bases de React Native et ses différences avec React Web.
- L'utilisation d'Expo pour simplifier le développement.
- L'intérêt de ES6.
- La mise en place d'un projet React Native et l'affichage d'un premier écran.

Prochaine étape : Structurer et ajouter des interactions dans l'application.

Bonne exploration ! 🚀