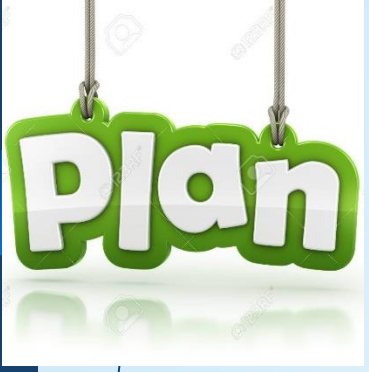
Introduction à React Native

Construire des applications mobiles natives avec JavaScript et React

Réalisée par:
Ben Béchir Chaima



Qu'est-ce que React Native ?

Le framework multiplateforme pour le mobile

- Un framework open-source pour construire des applications mobiles (iOS et Android).
- Utilise JavaScript et la bibliothèque React.
- Ne s'appuie pas sur une "webview" (contrairement à d'autres frameworks hybrides), mais rend des composants UI natifs.
- Permet de partager une grande partie du code entre les deux plateformes.
- Conçu pour le développement d'applications mobiles, pas pour le web.

Les avantages et les concepts clés

Pourquoi choisir React Native ?

- ✓ **Code réutilisable :** Un seul code base pour iOS et Android.
- ✓ **Performances natives :** Utilise les composants natifs, ce qui se traduit par une meilleure fluidité.
- ✓ **Écosystème React :** Exploite la popularité et les outils de React (débogage, etc.).
- ✓ **Hot Reloading :** Permet de voir les modifications en direct sans recompiler l'application.
- ✓ **Composants :** Tout comme dans React.js, l'UI est construite avec des composants.
- ✓ **Le "Bridge" :** La communication entre le code JavaScript et les composants natifs de la plateforme.

Configuration de l'environnement

Préparer votre machine

Option 1: Utilisation d'Expo (Recommandé pour les débutants)

- ❖ **Avantages :** Installation rapide, pas besoin d'Android Studio ni de Xcode, l'outil gère la compilation pour vous.
- ❖ **Étapes : npm install -g expo-cli**, puis **expo init mon-projet**.
- ✓ Exécuter l'application sur un appareil réel en scannant un code QR.

Option 2 : React Native CLI (pour des projets plus complexes)

- ❖ **Avantages :** Plus de contrôle sur le projet, possibilité d'utiliser des modules natifs.
- ❖ **Nécessite :** Installation de Java Development Kit (JDK), Android Studio, Xcode, etc.

La structure d'un projet et le JSX

Naviguer dans votre projet

Fichier principal : App.js ou **index.js**

Composants de base :

- ✓ **<View>:** Le conteneur de base, l'équivalent d'une div en HTML.
- ✓ **<Text>:** Affiche du texte. **Important :** tout le texte doit être à l'intérieur d'une balise **<Text>**.
- ✓ **<Image>:** Affiche des images.
- ✓ **<Button>:** Un composant bouton simple.

Exemple de JSX

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

const App = () => {
  return (
    <View>
      <Text>Bonjour, monde mobile !</Text>
    </View>
  );
};

export default App;
```

Le style et Flexbox

Mettre en forme votre UI

Comment styliser : Utiliser la feuille de style (**StyleSheet**).

Syntaxe : StyleSheet.create({ container: { ... } })

Le modèle de mise en page (Layout) :

- ✓ React Native utilise Flexbox par défaut. Il est crucial de le comprendre.
- ✓ **flexDirection, justifyContent, alignItems, flex.**

Exemple :

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

La navigation

Créer des applications à plusieurs écrans

Bibliothèque recommandée : React Navigation.

Principaux types de navigation :

- ✓ **Stack Navigator :** Navigation par empilement d'écrans (le plus courant).
- ✓ **Tab Navigator :** Navigation par onglets.
- ✓ **Drawer Navigator :** Navigation par tiroir latéral.

Installation : npm install @react-navigation/native et les dépendances nécessaires.

Les composants natifs de la plateforme

Accéder aux fonctionnalités du téléphone

Modules natifs : Pour accéder à la caméra, au GPS, aux contacts, etc.

Exemple : Utiliser la caméra.

- ✓ Il faut installer une bibliothèque comme **react-native-image-picker**.
- ✓ C'est là que le React Native CLI est souvent nécessaire (pour des configurations natives).

Platform API : Un moyen simple de détecter la plateforme (**Platform.OS === 'ios'**) pour du code spécifique.

Les hooks et la gestion de l'état

Rendre l'application interactive

Les hooks useState et **useEffect** fonctionnent exactement de la même manière qu'en React.js.

- ✓ **useState:** Pour gérer l'état local d'un composant (ex: l'état d'un champ de saisie, la visibilité d'un modal).
- ✓ **useEffect:** Pour gérer des effets de bord (appels API, écouteurs d'événements, etc.).

Conclusion et prochaines étapes

Résumé et pistes de développement

Résumé :

- ✓ React Native permet de créer des applications mobiles natives avec JavaScript.
- ✓ Le partage de code et les performances sont ses principaux atouts.
- ✓ Les composants et Flexbox sont au cœur du développement.
- ✓ Expo est idéal pour un démarrage rapide.

Prochaines étapes :

- ✓ Apprendre la gestion d'état globale avec Redux ou le Context API.
- ✓ Créer des requêtes API pour interagir avec un backend.
- ✓ S'intéresser aux tests d'applications mobiles.
- ✓ Publier l'application sur l'App Store et le Play Store.

Merci de votre attention!