React Native

React Native est une version de react un petit peu différente. Basé sur les mêmes technologies et concepts, elle permet de créer des applications mobile installable depuis les stores.

Voici le site officiel de React Native :

https://reactnative.dev/

Installation

Afin de commencer à utiliser React Native il faut d'abord créer un nouveau projet :

> Remplacer le nom du projet par le nom de votre choix. Cette commande créé un nouveau dossier contenant l'application react native

```
# Vous pouvez choisir un projet "blank" avec ou sans typescript
npx create-expo-app <nomDuProjet> --template
```

Une fois la commande terminé, il faut se rendre dans le dossier le l'application et lancer la commande :

```
# Démarre l'application react avec Expo
npm start
```

Cette lance le serveur expo. Il vous faut installer l'application expo sur votre téléphone (Google Store, Apple Store).

Une fois installé (et votre compte créé pour les utilisateurs Apple) il faut scanner le QR code présent dans le terminal de la commande `npm start`. Un fois scanné, l'application devrais s'afficher.

Le fichier App.tsx

Le point d'entré d'une application react-native, c'est le fichier `App.tsx`. Ce fichier ne se situe pas dans le dossier `src/Component`. C'est ici que l'on placera le code principal de notre application mobile.

Utiliser les composants React Native

React native est conçu pour s'éxécuter sur des téléphones. En interne, il n'utilise pas HTML ! On utilise plutôt des composants de react native :

> Vous retrouverez tout les composants react native et leurs documentation juste ici : les composants react native

L'utilisation

Exemple : Le composant text permet d'affiche du text, c'est l'équivalent du balise `<p>` :

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

export default function App() {
  return (
    <View>
      <Text>Coucou les amis</Text>
    </View>
  )
}
```

Les équivalents :

- `Text` : celui correspond aux balise textes (p, h1, h2, h...)
- `View` : Correspond à la balise `div`
- `Button` : Correspond à la balise `button`
- `TextInput` : Correspond à la balise `<input/>`
- `Image` : Correspond à la balise `img`
- `Switch` : Correspond à la balise `checkbox`

> Attention à bien lire la documentation, parfois il éxiste des props (des atributs différents).

Le composant `ScrollView`

Différement d'une page web, une application mobile doit être très performante (pour la fluidité). Lorsque l'on as un composant `View` assez grand (son contenu sor de l'écran, on peut scrolle). Afin d'optimiser l'affichage, vous pouvez utiliser un `ScrollView`. Il s'utilise de la même manière que le view, cependant tout ce qui dépasse de l'écran n'est pas affiché mais conservé en mémoire optimisant ainsi les performances.

> En résumé : il faut utiliser le scroll lorsqu'une liste est assez longue, quelle peut dépasser l'écran. Example : Sur une application de vente de vétement en ligne, la liste des produits devrait utiliser une ScrollView.

Le composant `FlatList`

Différement d'une page web, lorsqu'une liste doit se mettre à jour très souvent. Exemple : vous développez une application pour des paris sportif. Dans cette application une liste des derniers paris doit s'afficher (entre 40 et 100 résultat) et en plus doit se mettre à jour constamment ! Ici, la liste est mise) jour en temps reel.

Dans ce cas, il est fortement recommandé d'utiliser la `FlatList`

Les evénements

Différement du web, on ne clique pas sur application mobile, on "press" (toucher). Du coup, il nous faut pas utiliser `onClick` mais plutôt `onPress`. Cependant seulement certains élément peuvent être "touché" (pressed). C'est le cas du `Button` et c'est tout...

Nous pouvons cependant rendre n'importe quelle "touchable" (pressable) en utilisant le composant `TouchableOpacity` :

Exemple rendre un text "touchable" :

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

export default function App() {
  return (
    <View>
      <TouchableOpacity onPress={() => console.log('touché!')}>
        <Text>Coucou les amis</Text>
      </TouchableOpacity>
    </View>
  )
}
```

Styled components

Nous pouvons installer styled components :

```
npm i styled-components
npm i -D @types/styled-components
```