


React Native, développer des applications mobiles natives

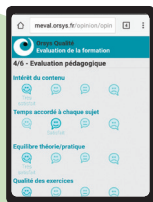
07 - 09 mars 2022



 **SÉMINAIRES - COURS DE SYNTHÈSE - STAGES PRATIQUES - CERTIFICATIONS** 
PARCOURS CERTIFIANTS - FORMATIONS À DISTANCE
E-LEARNING - COACHING 

APRÈS VOTRE FORMATION...

POUR CONTINUER LES ÉCHANGES



L'évaluation de la formation : elle doit être remplie le dernier jour de la formation avant 18h. Elle est accessible en ligne depuis votre poste, tablette ou smartphone à l'adresse **<http://eval.orsys.fr>**. Le mot de passe nécessaire sera fourni par votre formateur.



MyOrsys : espace web réservé aux participants qui ont suivi l'une de nos formations. Vous y trouverez vos supports de cours au format PDF. Les formateurs ont également un accès afin de pouvoir compléter, si besoin, avec des documents annexes : articles, études de cas, corrections d'exercices, etc.



Le blog : "Les carnets d'ORSYS" est alimenté plusieurs fois par semaine et rassemble tous nos articles d'actualité couvrant l'ensemble des domaines enseignés. Un travail réalisé en collaboration avec nos intervenants, spécialistes de leur domaine et parfaitement au fait de l'actualité.



Les conférences d'actualité et les webinars gratuits : pour vous tenir informés et échanger sur des thèmes d'actualité, nous vous proposons tout au long de l'année de venir rencontrer nos experts et d'échanger avec eux.



Les vidéos : avis d'expert, présentation des cours, description de méthodes pédagogiques... directement depuis notre site ou sur notre chaîne YouTube, nos vidéos seront pour vous une source d'informations précieuse !

Suivez notre activité sur les réseaux sociaux :



Notre vocation, votre réussite

LES DOMAINES ABORDÉS

TECHNOLOGIES NUMERIQUES

- Management des SI
- Gestion de projets, MOA
- Développement logiciel
- Big Data, BI, NoSQL, SGBD
- Technologies Web
- Réseaux et sécurité
- Systèmes d'exploitation
- Virtualisation, Cloud et DevOps
- Messagerie, travail collaboratif
- Bureautique
- Graphisme, multimédia, PAO, CAO

COMPETENCES METIERS

- Gestion, comptabilité et finance
- Ressources humaines
- Formation
- Office Manager, assistant(e), secrétaire
- Commercial et relation client
- Marketing digital
- Marketing et communication d'entreprise
- Amélioration continue, Lean, QSE
- Gestion de production, performance opérationnelle
- Achats, services généraux, logistique
- Droit et contrats
- Banque et assurance
- Secteur public
- Santé et action sociale

MANAGEMENT ET DEVELOPPEMENT PERSONNEL

- Management d'entreprise
- Management
- Management avancé et Leadership
- Développement personnel
- Office Manager, assistant(e), secrétaire
- Perfectionnement à l'Anglais
- Gestion de projets

QUELQUES CHIFFRES

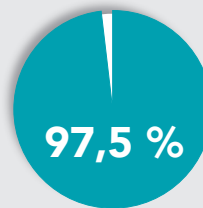


10 000
sessions par an.

Près de **70 000**



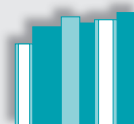
personnes formées en 2018.



de taux de
satisfaction
exprimé par
nos clients.



43
ans
d'expérience.



plus de
2 000
cours différents.

ORSYS

la méthode

DES SPÉCIALISTES RECONNUS



- Les animateurs des cours ORSYS sont des spécialistes reconnus dans leur domaine,



- Hommes et femmes de terrain, responsables de projets d'avant-garde dans l'industrie ou la recherche, consultants ou coaches,
- Concepteurs de leur cours, et notamment de leur documentation,
- Indépendants des constructeurs ou éditeurs de logiciels, de toute « école ».

UNE ORIENTATION OPÉRATIONNELLE



- Les cours ORSYS ont pour objectif de former d'une manière concrète et directement applicable,
- Centrés sur les méthodes et techniques de pointe et porteuses d'avenir,



- Les travaux de groupe, les exercices sont issus des projets réalisés par les animateurs,
- Les cas réels permettent d'aborder les difficultés techniques et aussi organisationnelles.

ORSYS

une offre structurée

LES FORMATIONS



LES SÉMINAIRES

- Maîtriser les concepts et les techniques,
- Faire le point sur l'Etat de l'Art,
- Choisir les solutions d'avenir.



LES COURS PRATIQUES

- Maîtriser les outils, les systèmes, les méthodes,
- Apprendre à les mettre en œuvre,
- Acquérir une spécialisation.



LES COURS DE SYNTHÈSE

Permettent de faire un point complet sur un thème précis : ils présentent les informations les plus récentes dans une optique opérationnelle, et sont accompagnés de démonstrations et d'études de cas.



LES MULTIMODALES

ORSYS vous propose différents types de formations qui associent le présentiel à des solutions innovantes mixant plusieurs modalités d'apprentissage : séquences e-learning, études de cas pratiques, ateliers collaboratifs, serious games, quiz, tests de validation des acquis...

UNE ORGANISATION PAR FILIÈRES



Pour faciliter vos choix de formation, nous avons privilégié une approche par filière. Organisée selon des thématiques technologiques, cette approche correspond à un enchaînement précis de cours, qui permet de passer en quelques semaines de l'initiation à une technologie à sa pleine maîtrise opérationnelle.

FORMATIONS ORSYS



SÉMINAIRES - COURS DE SYNTHÈSE - STAGES PRATIQUES - CERTIFICATIONS



PARCOURS CERTIFIANTS - FORMATIONS À DISTANCE



E-LEARNING - COACHING



Ce support pédagogique vous est remis dans le cadre d'une formation organisée par ORSYS. Il est la propriété exclusive de son créateur et des personnes bénéficiant d'un droit d'usage. Sans autorisation explicite du propriétaire, il est interdit de diffuser ce support pédagogique, de le modifier, de l'utiliser dans un contexte professionnel ou à des fins commerciales. Il est strictement réservé à votre usage privé.

React Native, développer des applications mobiles natives

Jean-Louis Guénégo @ 2022



1

Objectifs pédagogiques

- Mettre en place un environnement de développement React Native
- Concevoir l'architecture logicielle d'une application mobile multiplateforme
- Construire une interface utilisateur fluide et performante
- Utiliser les principaux composants et les API natives proposées par React Native

2

Plan du cours

- **Introduction**
 - Le développement natif, l'hétérogénéité et la fragmentation.
 - Les principaux outils cross-platform.
 - Positionnement de React Native par rapport aux solutions existantes.
- **React Native**
 - Rappels sur ES6/ES2015. Notion de transpileur.
 - React et le superset de JavaScript JSX.
 - React Native : principes clés, fonctionnement général.
 - Installation et configuration de React Native.
 - Outils de développement et de debug.
- **Architecture d'application**
 - Configurer un composant : state & props.
 - React Native et MVC.
 - Le pattern Flux, une alternative au MVC.
 - L'arrivée de Redux : le store, le reducer, les actions.
- **Construire son interface**
 - Les composants de base (View, Text et Image) et leurs cycles de vie.
 - Les événements Touch, la ListView et la ScrollView.
 - Organiser le layout de l'application. Mise en page avec Flexbox.
 - Styler les composants.
 - Les différentes solutions de navigation entre les pages.
 - Composants avancés.
 - Utiliser des composants de la communauté.
 - Ajouter des animations et des transitions.
- **Les formulaires et la gestion des données**
 - Les principaux composants de formulaires.
 - La validation de la saisie et la gestion des erreurs.
 - Redux-form et alternatives.
 - Récupération des données : xmlhttprequest et fetch.
 - Le stockage local.
 - La gestion offline.
- **Interagir avec le terminal**
 - Les principales API natives de React Native.
 - Utiliser les plugins Cordova/PhoneGap.
 - Développer un module natif.
- **Usages avancés**
 - Best Practices et erreurs fréquentes.
 - Tests unitaires et fonctionnels.
 - Publier l'application.
 - Mises à jour Over The Air.
 - Frameworks et outils complémentaires.

Travaux pratiques

- Monter une application complète de gestion de stock avec React Native :
 - Layout
 - SPA
 - Liste des articles (liste de carte)
 - Ajout d'un article (formulaire)
 - Effacement d'article
- Déploiement de l'application

- **Introduction**

- React Native
- Architecture d'application
- Construire son interface
- Les formulaires et la gestion des données
- Interagir avec le terminal
- Usages avancés

Le développement natif, l'hétérogénéité et la fragmentation.

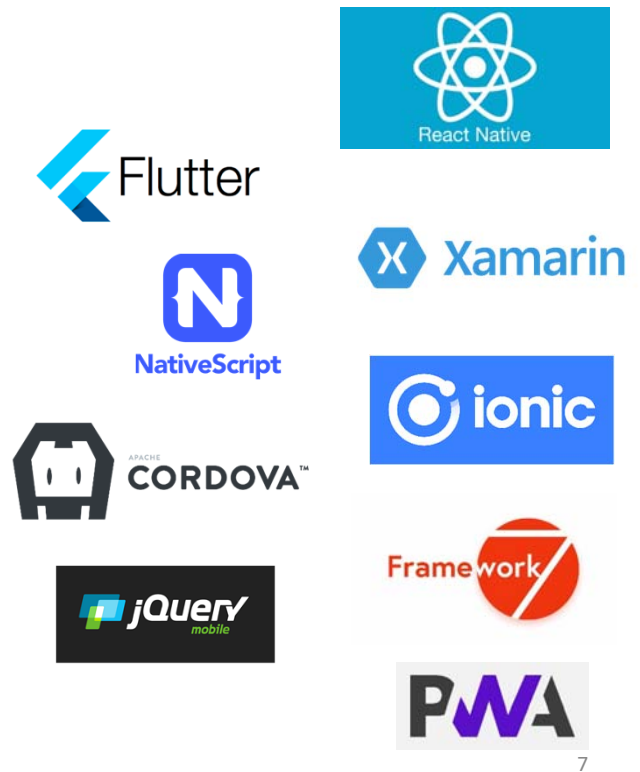
- Monde du mobile :
 - iOS (15%) : Objective C, Swift avec XCode
 - Android (85%) : Java, Kotlin avec Android Studio



- Besoin de simplifier le développement
 - Approche de React Native
 - Basé sur React (issu du web) et Javascript
 - Produit une application native (APK pour Android) ou (IPA pour iOS)

Les principaux outils cross-platform.

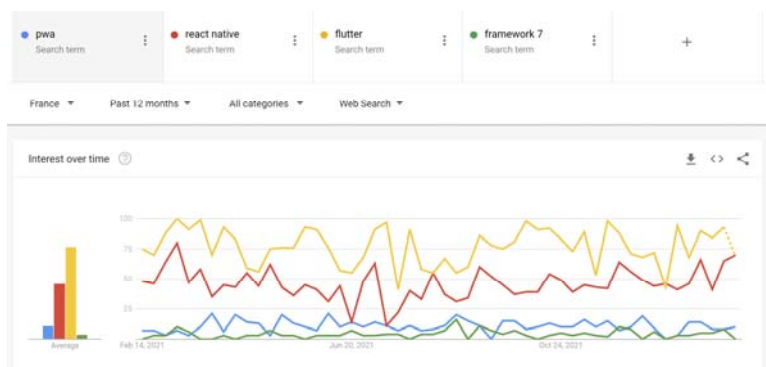
- React Native
- Flutter
- Xamarin
- NativeScript
- Ionic
- Apache Cordova
- Framework 7
- jQuery Mobile
- PWA



Source : <https://www.ideamotive.co/blog/best-react-native-alternatives-for-mobile-development>

Positionnement de React Native par rapport aux solutions existantes.

- Grosso modo, la compétition se fait entre **Flutter** et **React Native**.
 - Flutter : 2018 – Google
 - React Native : 2015 - Facebook
- Voir le Google Trends :

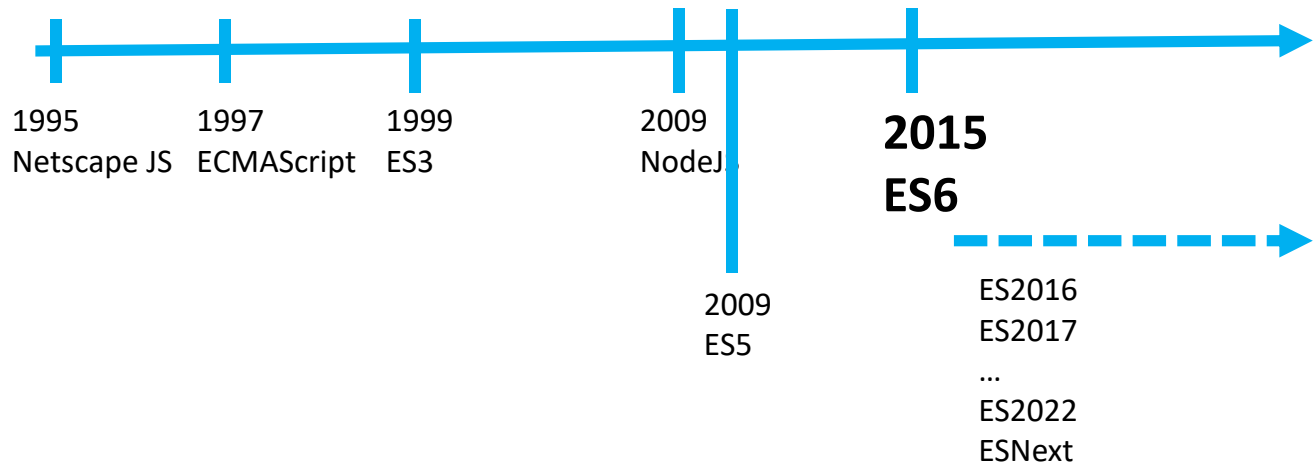


8

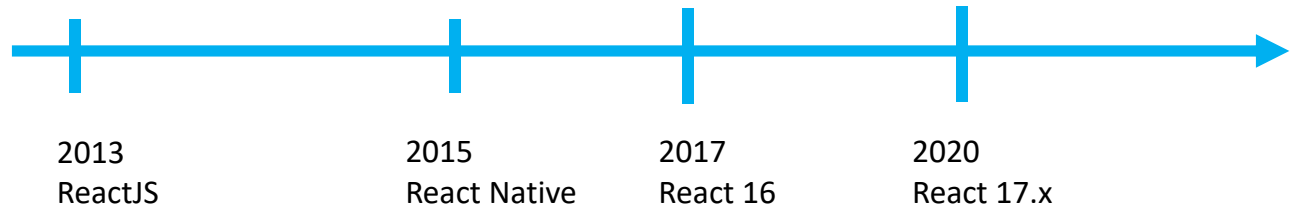
Source : <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021>

- Introduction
- **React Native**
- Architecture d'application
- Construire son interface
- Les formulaires et la gestion des données
- Interagir avec le terminal
- Usages avancés

Rappels sur ES6.



React



11

Apport ES6

- module (import/export)
- let/const (~~var~~)
- arrow function
- Notation de classe et héritage
- Promise native + notation async/await en ES8



12

Notion de transpileur

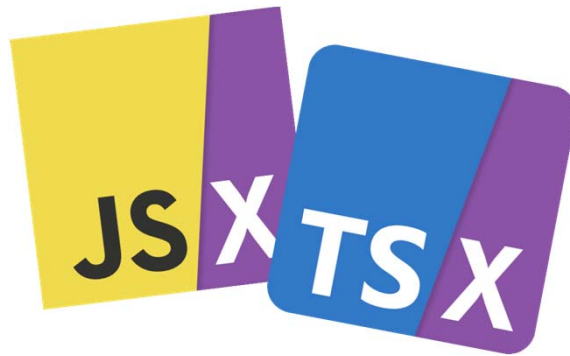
- Typescript -> Javascript
- CoffeeScript -> Javascript
- JSX -> Javascript
- TSX -> Javascript



13

React et le superset de JavaScript JSX.

- Ecrire des bouts de code HTML dans Javascript directement.
- Un composant react est une fonction qui retourne de l'HTML (un JSX).



14

Premier exercice

- Création d'un compte sur expo.dev
- Créer un snack « Hello World »



15

React Native : principes clés, fonctionnement général.

- React s'écrit en **javascript**. Plus particulièrement en **JSX** qui facilite l'écriture de l'HTML.
 - React se décline sur plateforme web avec la librairie 'react-dom'.
 - React se décline sur mobile avec la librairie 'react-native'
- La librairie 'react-native' donne accès au « **core components** » : View, Text, Image, TextInput, ScrollView, etc.
- Ecrire une application React Native revient simplement à écrire des composants utilisant les core components.
 - Si on veut faire plus compliqué, on peut toujours écrire des modules natif en Java (Android) ou Swift (iOS) et les exposer en JS pour react native.

16

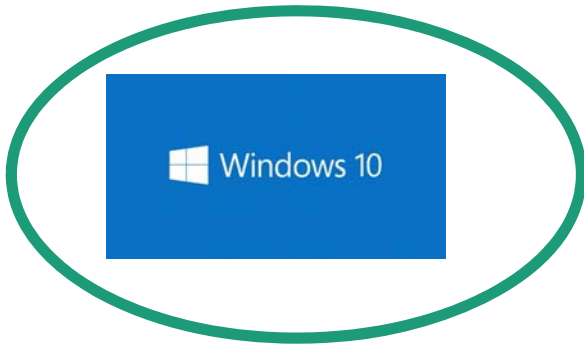
Liste des principaux core components

- View
- Text
- Image
- TextInput
- ScrollView
- FlatList
- SectionList
- Liste exhaustive :
<https://reactnative.dev/docs/components-and-apis>

17

Prérequis d'installation

- React Native fonctionne sur MacOS, Linux et Windows.
- Cette formation se fait sur Windows
 - C'est le système le plus familier des développeurs en France.
 - Cela dit le développement mobile se fait souvent sur MacOS à cause des 15% de part de marché de l'iOS.
iOS => Xcode => MacOS



18

Installation d'outils sur Windows

- NodeJS ≥ 12
- Git (et optionnellement TortoiseGit)
- Visual Studio Code (VSCode)

- OpenJDK 11
- Android Studio
- ~~Xcode~~ (non c'est sur Mac Seulement)

19

Désinstaller

- pour partir d'une machine « propre »

20

Désinstaller IntelliJ IDEA

- Add or Remove Program
 - « intellij »

21

Désinstaller Java

- Il peut y avoir des JRE, des JDK
 - Oracle
 - OpenJDK
- Add or Remove Program
 - « java »
 - « jdk »
- Désinstaller Groovy, Gradle, Maven si existant

22

Android Studio

- Le désinstaller complètement
 - Add or Remove program
 - Le désinstaller en cochant tout
 - Effacer les dossiers si ils existent de Gradle et Maven (repos utilisé par Android)
 - %USERPROFILE%/.gradle
 - %USERPROFILE%/.m2
 - Effacer :
 - %APPDATA%/JetBrains
 - C:/Program Files/Android
 - %LOCALAPPDATA%/Android (et tout ce qui s'apparentrait à Android)
 - %USERPROFILE%\AndroidStudioProjects
 - %USERPROFILE%/AppData/Local/Google/AndroidStudio...

23

Source : <https://stackoverflow.com/questions/39953495/how-to-completely-uninstall-android-studio-from-windowsv10>

Nettoyage du PATH et de toute autre variable d'environnement

- PATH : retirer les répertoires d'Android et java, et groovy, etc.
- ANDROID (et dérivés)
- JAVA
- GROOVY

24

Nettoyage de %USERPROFILE%

- C'est le répertoire du compte utilisateur sous Windows.
- Il contient parfois des répertoires de configuration

25

Machine propre !

- Installation de :
 - Git (et TortoiseGit)
 - NodeJS (avec nvm-windows)
 - Visual Studio Code (dernière version)
 - Java : OpenJDK 11
 - Android Studio (dernière version)

Git

- Google « git »
- <https://git-scm.com/>
- tester si git est déjà installé : `git --version`
- Installer la dernière version par dessus l'existante

27

NodeJS

- L'installer de préférence avec nvm-windows
 - Note : Les pragmatiques peuvent installer la dernière LTS directement depuis google « nodejs ».
- <https://github.com/coreybutler/nvm-windows/releases>
- Installer la dernière LTS (aller sur le site de NodeJS pour vérifier)

Download for Windows (x64)

16.13.2 LTS

Recommended For Most Users

17.3.1 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

28

Visual Studio Code

- Le désinstaller si besoin (Windows add or remove programs)
- Googler « vscode » pour le télécharger en dernière version
- Bien veiller à le mettre en anglais (ctrl+shift+P language)

29

Android Studio

- Googler « Android studio » et télécharger
- Il faut ensuite :
 - télécharger un Android SDK
 - télécharger un Simulateur de téléphone portable (AVD = Android Virtual Device)
- Noter que :
 - Android Studio contient un JDK (openjdk 11)
 - C:\Program Files\Android\Android Studio\jre

30

Variable d'environnement

- ANDROID_HOME qui pointe sur le Android SDK.
 - ANDROID_HOME=%USERPROFILE%\AppData\Local\Android\Sdk
- PATH qui doit contenir le répertoire du SDK
« platform-tools »
 - PATH=%PATH%;% ANDROID_HOME %\platform-tools

31

Platform Tools c'est quoi ?

- Contient :
 - **adb.exe** (Android Debug Bridge) : permet de parler à un Device (ie un smartphone)
 - <https://developer.android.com/studio/command-line/adb>

32

Fin des installations des prérequis

- Noter que on pourrait développer des applications native Android
- Maintenant on passe à React Native



33

Installation et configuration de React Native.

- 2 méthodes :
 - expo cli
 - pas besoin de Xcode (iOS) ou Android Studio si on utilise que l'export web.
 - react native cli
 - on a besoin de Xcode (iOS) ou Android Studio mais possibilité d'utiliser des librairies de composants écrit en langage natif.

34

Source : <https://reactnative.dev/docs/environment-setup>

Expo Cli

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

Créer un projet avec :

```
expo init my-project
```

35

React native CLI

```
npx react-native init MyProject
```

```
> cd MyProject
```

```
> npm start
```

36

Utiliser le JDK de Android Studio ? Mauvaise idée

- Si vous avez l'erreur suivante dans npx react-native :
 - **ERROR: JAVA_HOME is not set and no 'java' command could be found in your PATH.**
- Certains binaires d'Android SDK ont vraiment besoin d'un JDK 11 (ex: avdmanager)
- Installer un JDK 11 (exemple: openjdk 11)
- `JAVA_HOME=C:\openjdk\jdk-11.0.2`
- `PATH=%PATH%;%JAVA_HOME%\bin`

37

Facebook Metro

[illegible]

Metro est l'outil que React Native utilise pour :

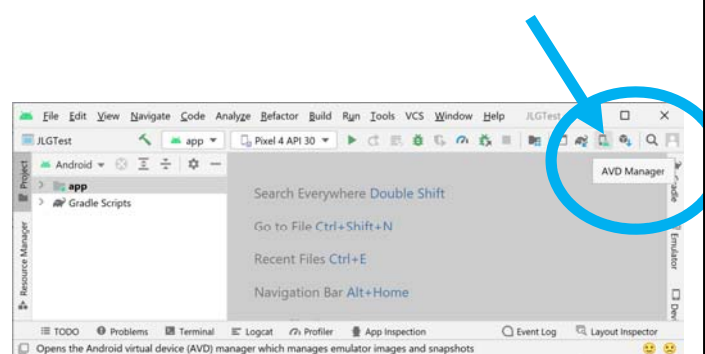
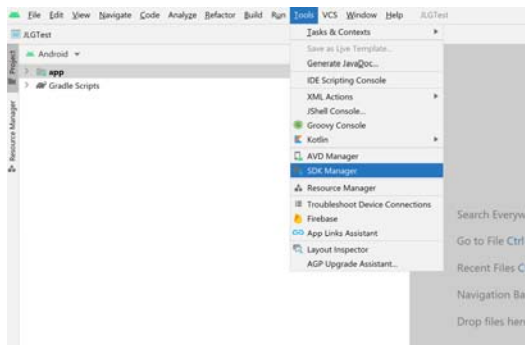
- « bundler » le projet
- l'envoyer sur le device simulateur
- fabriquer l'APK ou l'IPA

38

En savoir plus sur Metro : <https://facebook.github.io/metro/>

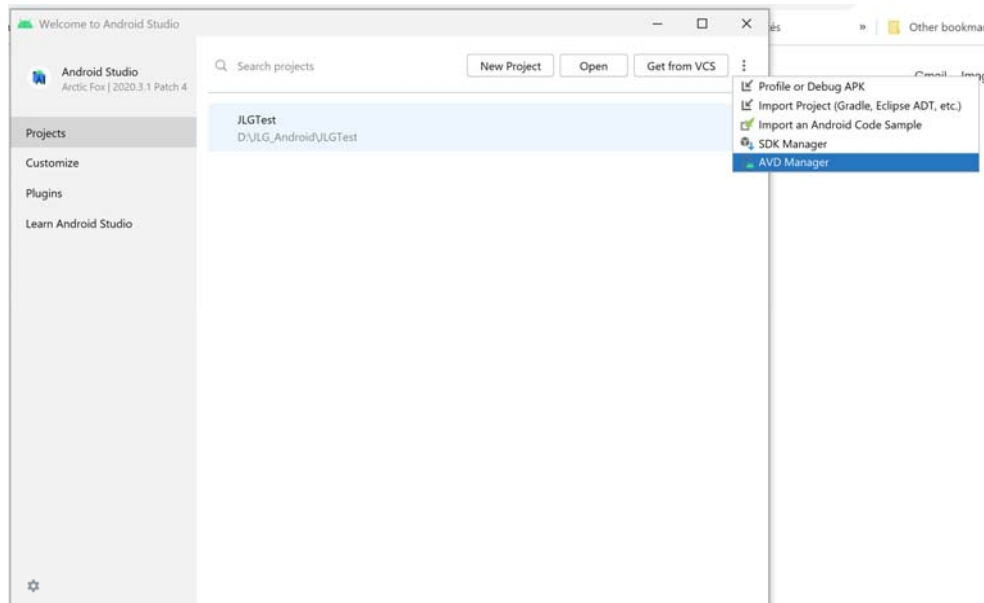
Si le device ne s'allume pas...

- il faut le forcer.
- Ouvrir le AVD Manager (depuis Android Studio), puis démarrer le device



39

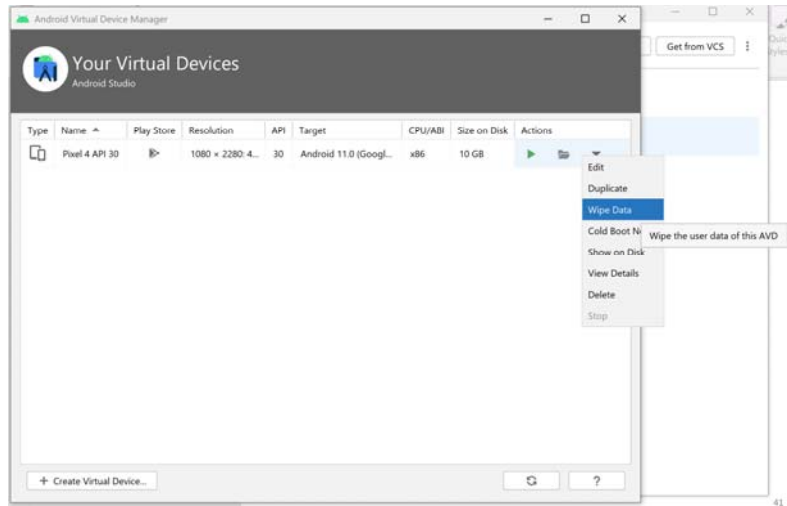
Android Studio > AVD Manager



40

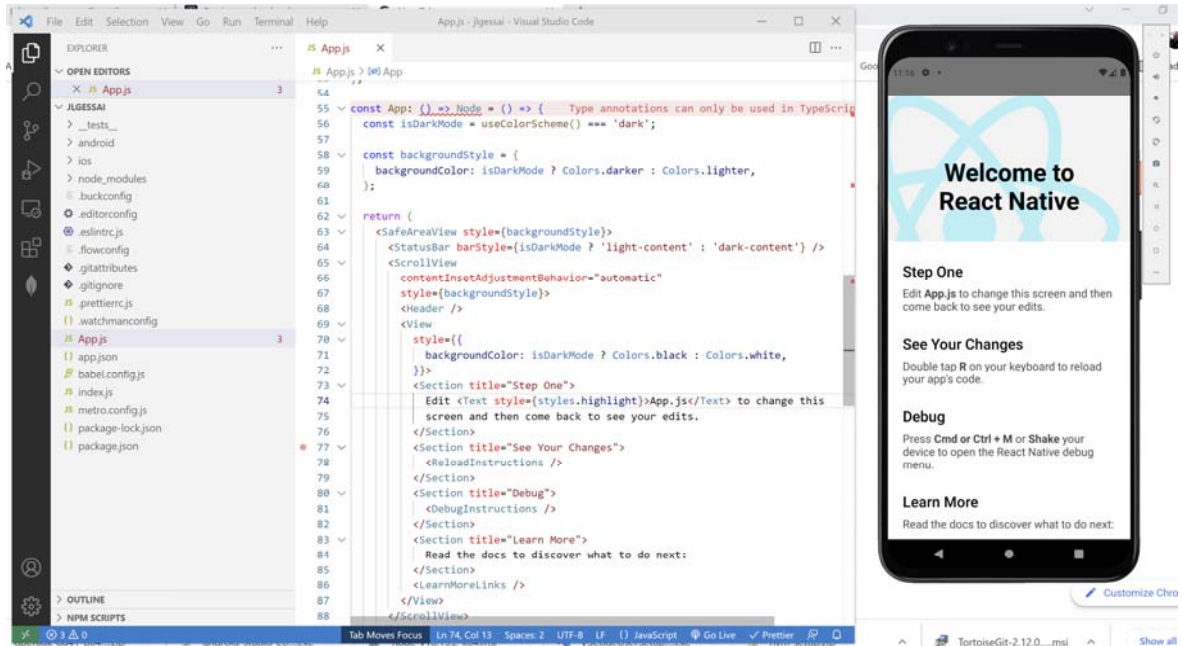
Reset d'un téléphone « pourri »

- Ne pas hésiter à wiper les data



41

Projet visualisé dans le simulateur



42

Reload le projet

- Si ce n'est pas automatique, alors faire dans le device : CTRL+M et ensuite « reload »
- Ou alors dans Metro, taper « r »

43

Effacer un projet expo ou react native

- npm i rimraf -g
 - Permet un effacement rapide sous Windows 10
 - rimraf <répertoire>
- Parfois il faut killer le process **adb** sur windows
 - Task Manager > process > adb.exe

44

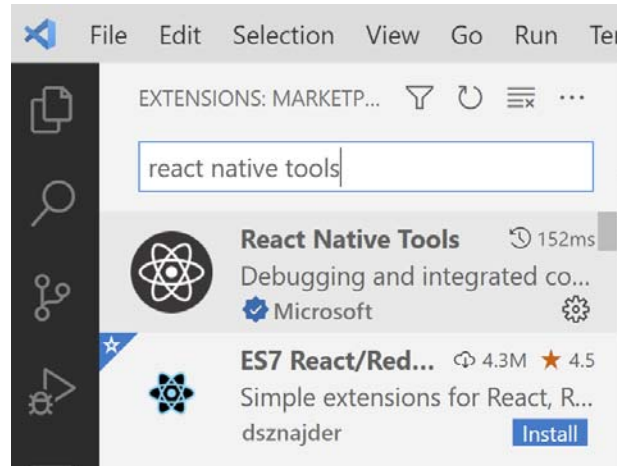
Outils de développement et de debug.

- IDE de développement : VSCode
- Le débogage se fait par test visuel, simulé sur un device ou via webpack (web)
- VSCode : installer l'extension Microsoft react-native-tools
 - Breakpoint
- console.log
- Error Boundaries

45

Breakpoint

- Installer react-native-tools



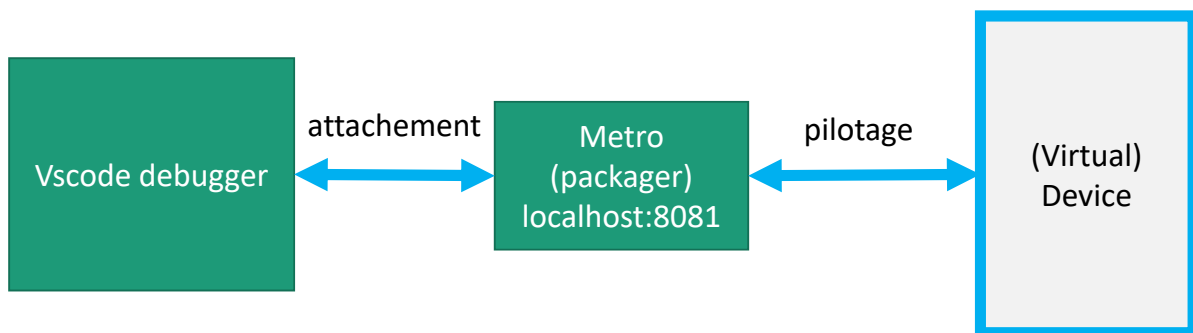
Créer une app avec create-react-native

```
npx react-native init MyProject
```

47

Configuration du debugger en mode « attach »

- Principe : le debugger s'attache à un processus qui tourne déjà.
 - Le processus doit être compatible pour se laisser attacher par le debugger

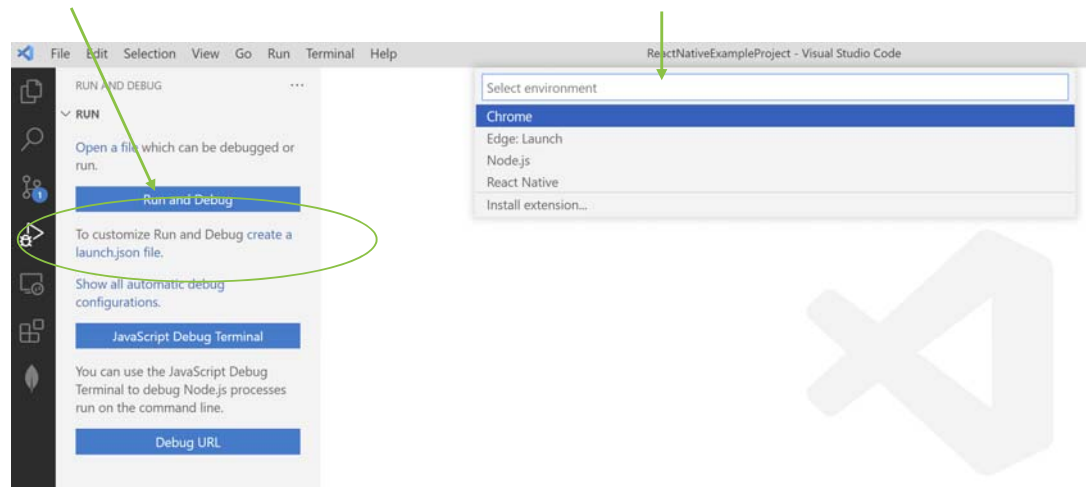


48

Créer le fichier launch.json

1. créer le fichier launch.json

2. choisir une conf au hasard.



Ajouter une configuration react-native

enlever le texte en bleu

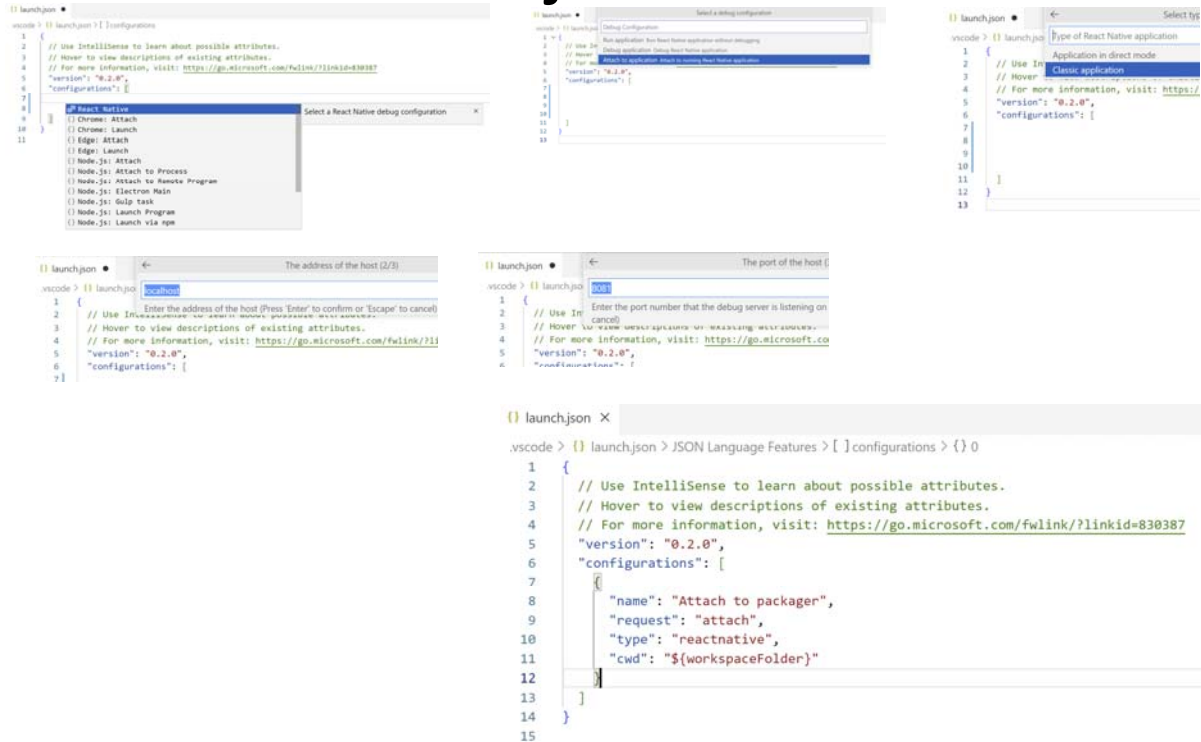
```
D:\_Projets\react-native\example\ios\code\launch.json
vscode > {} launch.json > JSON Language Features > {} configurations > {} 0
1 {
2   // Use IntelliSense to learn about possible attributes.
3   // Hover to view descriptions of existing attributes.
4   // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5   "version": "0.2.0",
6   "configurations": [
7     {
8       "type": "pwa-chrome",
9       "request": "launch",
10      "name": "Launch Chrome against localhost",
11      "url": "http://localhost:8080",
12      "webRoot": "${workspaceFolder}"
13    }
14  ]
15 }
```

Ajouter une nouvelle config

Add Configuration...

50

Fichier launch.json final



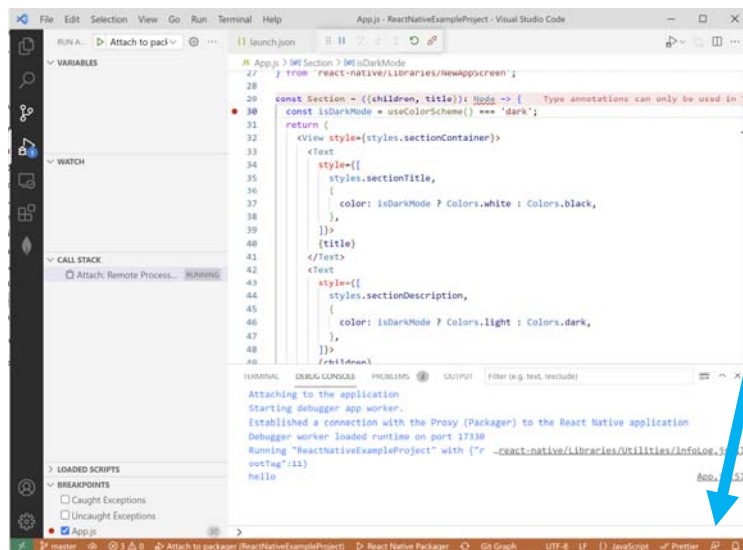
Démarrer le debugger

- Poser un breakpoint quelque part dans App.js
- Lancer le process « attach to packager »
- Metro est lancé directement dans VSCODE
- Lancer seulement après « npm run android » depuis un cmd.
 - Il va seulement allumer l'émulateur car metro est déjà lancé.
- **Vérifier que le mode « debug à distance » n'est pas activé.**
 - **L'enlever peut être dur. Dans ce cas, nettoyer le cache d'android (tous les répertoires qui ne sont pas dans git, mais sont dans le répertoire « android » : .gradle et app/build.**
 - **Refaire un npm run android**
- Activer le mode « debug à distance ».
- Constater que VSCode s'attache au process.
- Constater que VSCode ouvre App.js pour pointer sur le breakpoint.
- Détacher le process, et essayer de recommencer.

52

Barre de statut orange

- Quand le debugger de VSCode est attaché correctement, la barre de status est orange.



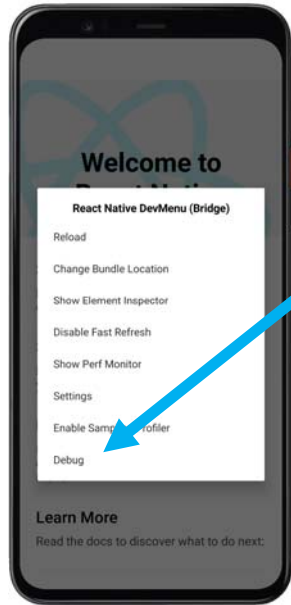
53

Si le debugger de chrome s'allume...

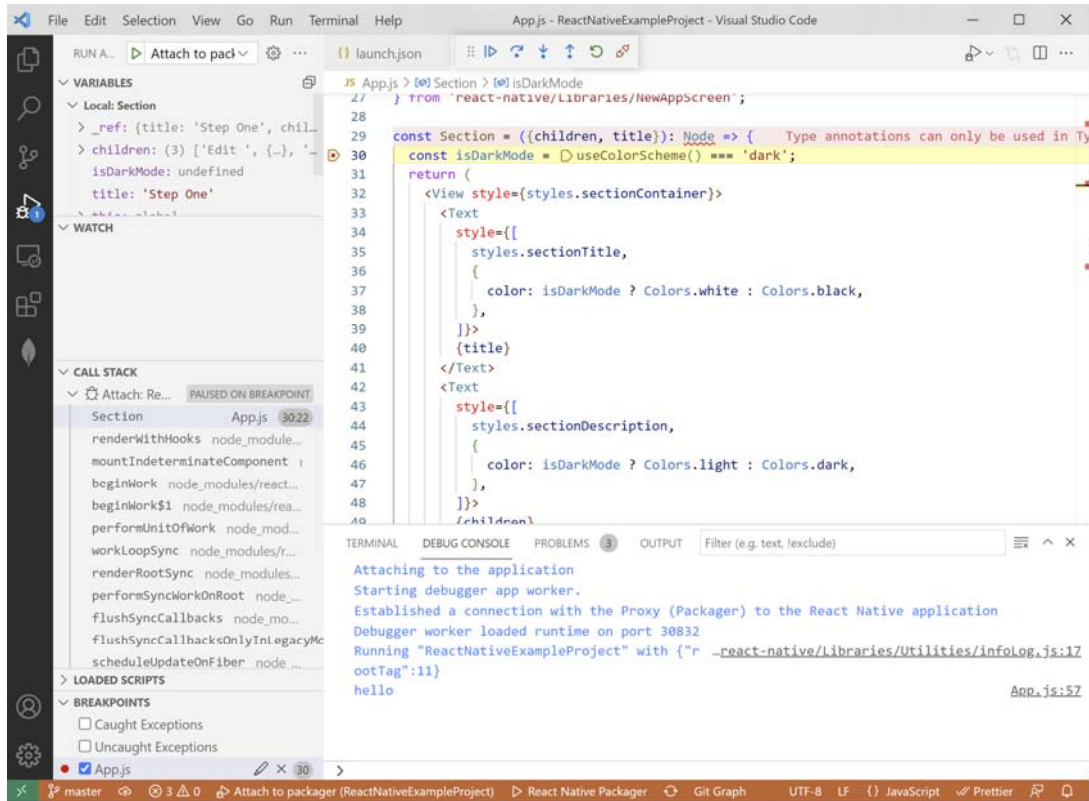
- Alors il est difficile de stopper ce debugger.
- Eteindre tout (emulator, metro et vscode)
- Il faut refaire la build android :
 - effacer le répertoire android/app/build
- npm run android
- Allumer vscode et relancer le debug « attach to packager »
- Lancer depuis l'émulateur le « debug ».

54

Menu depuis l'émulateur



Le libellé doit toujours être « debug »



- Introduction
- React Native
- **Architecture d'application**
- Construire son interface
- Les formulaires et la gestion des données
- Interagir avec le terminal
- Usages avancés

Projet en Typescript

- Créer un projet react-native avec le format Typescript.
- `npx react-native init MyApp --template react-native-template-typescript`

58

Source :

<https://reactnative.dev/docs/typescript>

<https://reactnative.dev/docs/environment-setup>

Configurer un composant : state & props.

- Un composant React est une fonction renvoyant un JSX
- la fonction peut prendre des paramètres. Ces paramètres doivent être immutables et sont appelés les **props**.
- la fonction peut utiliser un hook appelé **useState** pour spécifier un état interne.

59

euh... c'est quoi immutable ?

- Une variable est immutable si quand on change sa valeur alors on change l'adresse mémoire de la valeur.
- Pour une variable contenant un type simple (number, string, boolean) c'est toujours le cas. En revanche, pour un type objet il faut créer un nouvel objet et faire en sorte que la variable pointe sur le nouvel objet.

a n'est pas immutable

```
JS test.js ×
JS test.js > ...
1 let a = {
2   | truc: 123,
3   | };
4
5 a.truc = 456;
6 |
```

b est immutable

```
JS test.js ×
JS test.js > ...
1 let b = {
2   | truc: 123,
3   | };
4
5 b = {
6   | truc: 456,
7   | };
8 |
```

60

Props

TS Header.tsx X

src > layout > TS Header.tsx > ...

```
1 import React from 'react';
2 import {Text, View} from 'react-native';
3
4 const backgroundStyle = {
5   backgroundColor: 'white',
6   height: 50,
7 };
8
9 const AppHeader = (props: {name: string; color: string}) => {
10   return (
11     <View style={backgroundStyle}>
12       <Text style={{color: props.color}}>{props.name}</Text>
13     </View>
14   );
15 };
16
17 export default AppHeader;
18 |
```

TS App.tsx X

TS App.tsx > [W] App

```
19 import AppBody from './src/layout/Body';
20 import AppFooter from './src/layout/Footer';
21 import AppHeader from './src/layout/Header';
22
23 const App = () => {
24   const isDarkMode = useColorScheme() === 'dark';
25
26   const backgroundStyle = {
27     backgroundColor: isDarkMode ? Colors.darker : Colors.lighter,
28   };
29
30   return (
31     <SafeAreaView style={backgroundStyle}>
32       <StatusBar barStyle={isDarkMode ? 'light-content' : 'dark-content'} />
33       <ScrollView
34         contentInsetAdjustmentBehavior="automatic"
35         style={backgroundStyle}>
36         <AppHeader name="Gestion Stock" color="red" />
37         <AppBody></AppBody>
38         <AppFooter></AppFooter>
39       </ScrollView>
40     </SafeAreaView>
41   );
42 };
43
44 export default App;
45
```

61

State

```
TS Body.tsx  X
src > layout > TS Body.tsx > [e] AppBody
1  import React from 'react';
2  import {Text, View} from 'react-native';
3  import AppCounter from '../misc/Counter';
4
5  const AppBody = () => {
6    return (
7      <View>
8        <Text>Body...</Text>
9        <AppCounter />
10     </View>
11   );
12 };
13
14 export default AppBody;
15
```

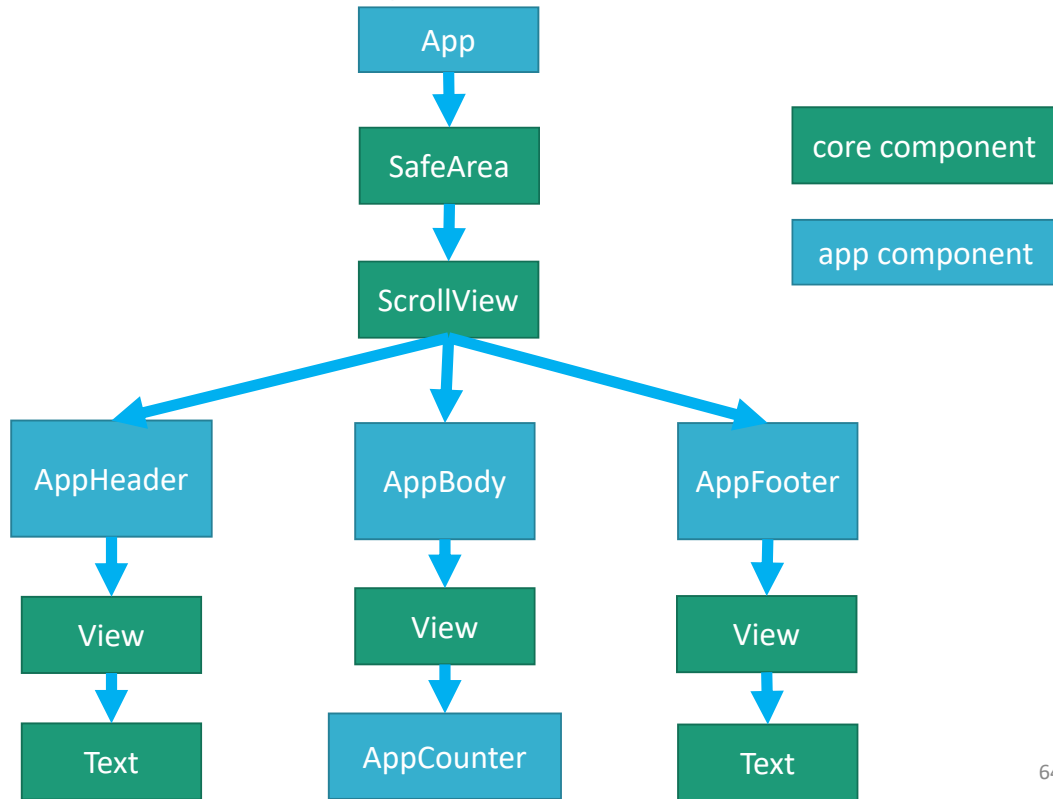
```
TS Counter.tsx  X
src > misc > TS Counter.tsx > [e] AppCounter
1  import React, {useState} from 'react';
2  import {Button, Text, View} from 'react-native';
3
4  const AppCounter = () => {
5    const [counter, setCounter] = useState(10);
6    const increment = () => {
7      setCounter(counter + 1);
8    };
9    return (
10     <View>
11       <Button
12         onPress={increment}
13         title="Increment"
14         color="#841584"
15         accessibilityLabel="Increment the counter"
16       />
17       <Text>Counter: {counter}</Text>
18     </View>
19   );
20 };
21
22 export default AppCounter;
23
```

62

React Native et MVC

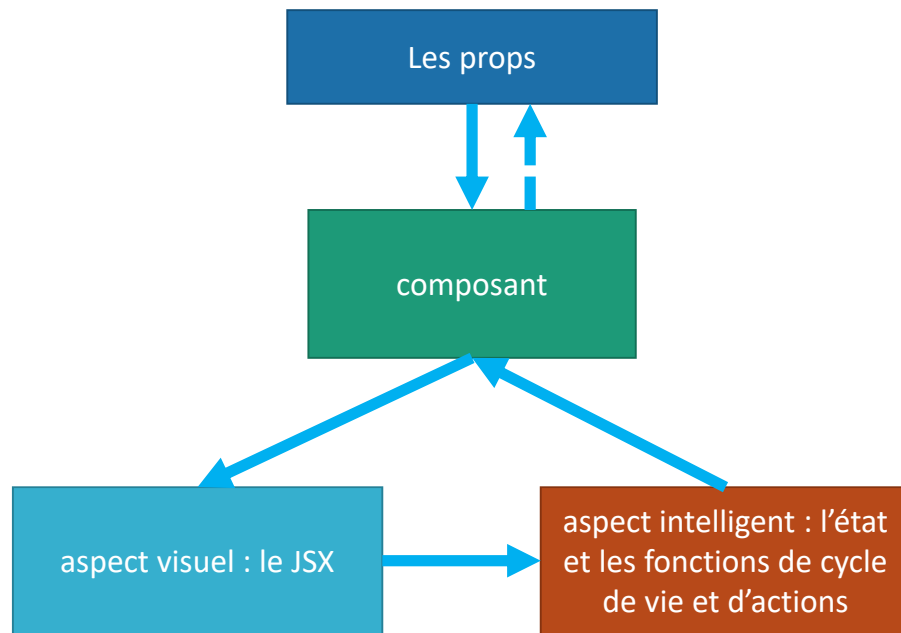
- React Native fonctionne comme React :
 - architecture à composant (arbre de composant)
 - un composant a une intelligence, un aspect visuel, et de la communication avec ses parents et ses enfants.

Arbre de composants



64

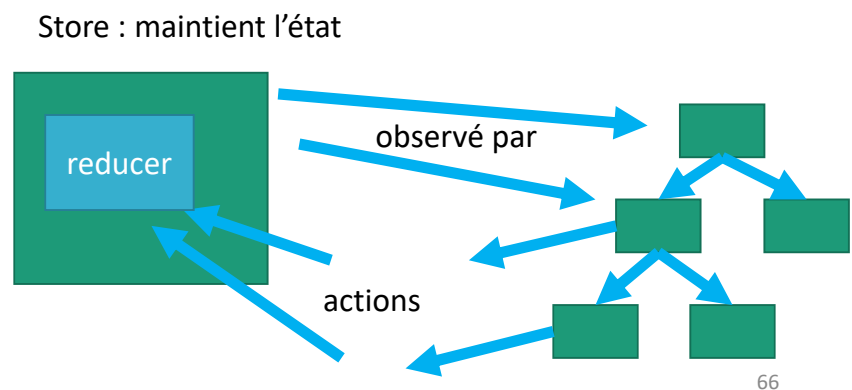
Composant



65

Le pattern Flux, une alternative au MVC.

- Principe : les flux sont tous monodirectionnels
- L'arbre de composants peut générer des actions, qui vont modifier un état, puis cet état peut être observé par l'arbre de composants.



L'arrivée de Redux : le store, le reducer, les actions.

- <https://redux.js.org/>
- Petite librairie de 2kb
- Implémentation du pattern Flux.
- Complètement indépendant de React, React-native, même si souvent associée dans la littérature

67

Mise en œuvre pratique de Redux

1. Définition d'un état initial
 2. Définition d'un reducer global
 3. Définition de la store
 4. Placement des fonctions d'actions
 5. Observation de la store
- Exemple : une application Nodejs qui écoute des commandes claviers qui vont agir sur un compteur.

68

Définition de l'état initial

```
4   const initialState = {  
5       counter: {  
6           value: 10,  
7       },  
8   };
```

69

Définition du reducer global

```
10 function reducer(state = initialState, action) {
11   switch (action.type) {
12     case "counter/increment":
13       console.log("action.nbr: ", action.nbr);
14       return { counter: { value: state.counter.value + action.nbr } };
15     case "counter/decrement":
16       return { counter: { value: state.counter.value - action.nbr } };
17     default:
18       return state;
19   }
20 }
```

IMPORTANT : le reducer doit toujours envoyer un tout nouveau objet reflétant le nouvel état. Car l'état est censé être **immutable**. Un objet est immutable si une fois créé, on ne peut plus le modifier dans la moindre de ses parties. (CRUD)

Note: c'est une bonne idée que de nommer le type de l'action avec une thématique et un verbe.

70

Définition de la store

```
const store = createStore(reducer);
```

71

Placement des fonctions d'actions

```
24 function runAction(type, nbr) {  
25   | store.dispatch({ type, nbr });  
26 }  
  
33 // listen order  
34 loop: while (true) {  
35   console.log("give an order (+/-/q): ");  
36   const order = await getOrder();  
37   switch (order) {  
38     case "+":  
39       runAction("counter/increment", 1);  
40       break;  
41     case "-":  
42       runAction("counter/decrement", 2);  
43       break;  
44     case "q":  
45       console.log("Bye");  
46       break loop;  
47     default:  
48       console.log("order not understood.");  
49   }  
50 }
```

IMPORTANT : une action DOIT être un objet **sérialisable** : en gros il faut pouvoir la transformer en chaîne de caractères sans perdre d'information.

72

Observation de la store

```
29 | // listen the store.  
30 | console.log(store.getState());  
31 | store.subscribe(() => console.log(store.getState()));
```

Critique : c'est un peu redondant que au moindre changement d'état, on repasse dans cette fonction. Si on est intéressé que par une toute petite partie de l'état (on appelle cela un **tranche d'état**), alors on se retape tout l'état quand même...

73

Optimisation de Redux : @reduxjs/toolkit

- Utilisation de la librairie **immer** pour faciliter la construction d'objet immuable
 - <https://github.com/immerjs/immer>
- Utilisation de la librairie **reselect** pour faciliter la gestion de l'abonnement aux tranches d'état
 - <https://github.com/reduxjs/reselect>



74

createStore -> configureStore

```
33  const store = configureStore({
34    reducer: {
35      counter: counterSlice.reducer,
36      list: listSlice.reducer,
37    },
38  });
```

L'état est un ensemble de tranche d'états (appelés des « **slices** ») qui peuvent être indépendant les uns des autres.

75

createSlice

- Regroupe l'état initial, le reducer et nomme la slice.
- Grâce à la librairie immer, plus besoin de se préoccuper du côté immutable du state.

```
6  const counterSlice = createSlice({
7    name: "counter",
8    initialState: {
9      value: 10,
10   },
11   reducers: {
12     increment: (state, action) => {
13       state.value += action.payload;
14     },
15     decrement: (state, action) => {
16       state.value -= action.payload;
17     },
18   },
19 });
20
21 const listSlice = createSlice({
22   name: "list",
23   initialState: {
24     array: [],
25   },
26   reducers: {
27     push: (state, action) => {
28       state.array.push(action.payload);
29     },
30   },
31 });
```

76

Les actions

```
54 // listen order
55 loop: while (true) {
56   console.log("give an order (+/-/p/q): ");
57   const order = await getOrder();
58   switch (order) {
59     case "+":
60       store.dispatch(counterSlice.actions.increment(1));
61       break;
62     case "-":
63       store.dispatch(counterSlice.actions.decrement(2));
64       break;
65     case "p":
66       store.dispatch(listSlice.actions.push("truc"));
67       break;
68     case "q":
69       console.log("Bye");
70       break loop;
71     default:
72       console.log("order not understood.");
73   }
74 }
```

77

L'observation des slices : selecteur

- Les sélecteurs sont des fonctions de requêtage d'état.
- La librairie **reselect**, intégrée à @reduxjs/toolkit apporte quelques facilités.
 - <https://github.com/reduxjs/reselect>

```
40  const selectCounter = (state) => state.counter.value;  
41  const selectListItems = (state) => state.list.array;
```

78

Redux dans react native

```
package.json (Working Tree) X
package.json > {} dependencies
1 {
2   "name": "gestionstock",
3   "version": "0.0.1",
4   "private": true,
5   "scripts": {
6     "android": "react-native run-android",
7     "ios": "react-native run-ios",
8     "start": "react-native start",
9     "test": "jest",
10    "lint": "eslint . --ext .js,.jsx,.ts,.tsx",
11  },
12  "dependencies": {
13    "@reduxjs/toolkit": "^1.7.1",
14    "react": "17.0.2",
15-   "react-native": "0.66.4"
16+   "react-native": "0.66.4",
17+   "react-redux": "^7.2.6"
18  },
19  "devDependencies": {
20    "@babel/core": "^7.12.9",
21    "@babel/runtime": "^7.12.5",
22    "@react-native-community/eslint-config": "^2.0.0",
23    "@types/jest": "^26.0.23",
24+   "@types/react-redux": "^7.1.22",
25    "@types/react-test-renderer": "^17.0.1",
26    "@typescript-eslint/eslint-plugin": "^5.7.0",
27    "@typescript-eslint/parser": "^5.7.0",
28    "babel-jest": "^26.6.3",
29    "eslint": "^7.14.0",
30    "jest": "^26.6.3",
31    "metro-react-native-babel-preset": "^0.66.2",

```

79

App.tsx

```
TS App.tsx (Working Tree) X
TS App.tsx > ...
14  ScrollView,
15  StatusBar,
16  useColorScheme,
17 } from 'react-native';
18 import {Colors} from 'react-native';
19 import AppBody from './src/layout/Body';
20 import AppFooter from './src/layout/Footer';
21 import AppHeader from './src/layout/Header';
22
23 const App = () => {
24   const isDarkMode = useColorScheme() === 'dark';
25
26   const backgroundColor = {
27     backgroundColor: isDarkMode ? Colors.darker : Colors.lighter,
28   };
29
30   return (
31     <SafeAreaView style={backgroundColor}>
32       <StatusBar barStyle={isDarkMode ? 'light-content' : 'dark-content'} />
33       <ScrollView
34         contentInsetAdjustmentBehavior="automatic"
35         style={backgroundColor}>
36         <AppHeader name="Gestion Stock" color="red" />
37         <AppBody />
38         <AppFooter />
39       </ScrollView>
40     </SafeAreaView>
41   );
42 };
43
44 export default App;
45
14  ScrollView,
15  StatusBar,
16  useColorScheme,
17 } from 'react-native';
18 import {Colors} from 'react-native/Libraries/NewAppScreen';
19 import {Provider} from 'react-redux';
20 import AppBody from './src/layout/Body';
21 import AppFooter from './src/layout/Footer';
22 import AppHeader from './src/layout/Header';
23 import {store} from './src/redux/store';
24
25 const App = () => {
26   const isDarkMode = useColorScheme() === 'dark';
27
28   const backgroundColor = {
29     backgroundColor: isDarkMode ? Colors.darker : Colors.lighter,
30   };
31
32   return (
33     <Provider store={store}>
34       <SafeAreaView style={backgroundColor}>
35         <StatusBar barStyle={isDarkMode ? 'light-content' : 'dark-content'} />
36         <ScrollView
37           contentInsetAdjustmentBehavior="automatic"
38           style={backgroundColor}>
39           <AppHeader name="Gestion Stock" color="red" />
40           <AppBody />
41           <AppFooter />
42         </ScrollView>
43       </SafeAreaView>
44     </Provider>
45   );
46 };
47
48 export default App;
49
```

80

La Store

```
TS store.ts (Untracked) X
src > redux > TS store.ts > ...
1-
1+ import {configureStore} from '@reduxjs/toolkit';
2+ import {counterSlice} from './slices/counter.slice';
3+
4+ export const store = configureStore({
5+   reducer: {
6+     counter: counterSlice.reducer,
7+   },
8+ });
9+
10+ // for Typescript
11+ export type RootState = ReturnType<typeof store.getState>;
12+ export type AppDispatch = typeof store.dispatch;
13+
```

81

La slice pour l'exemple counter

TS counter.slice.ts (Untracked) X

src > redux > slices > TS counter.slice.ts > ...

1-

le selecteur

les actions

```
1+ import {createSlice} from '@reduxjs/toolkit';
2+ import {RootState} from '../store';
3+
4+ export const counterSlice = createSlice({
5+   name: 'counter',
6+   initialState: {
7+     value: 10,
8+   },
9+   reducers: {
10+     increment: (state, action) => {
11+       state.value += action.payload;
12+     },
13+     decrement: (state, action) => {
14+       state.value -= action.payload;
15+     },
16+   },
17+ });
18+
19+ export const selectCounter = (state: RootState) => state.counter.value;
20+
21+ export const {increment, decrement} = counterSlice.actions;
22+
```

82

Quelques utilities (hooks)

```
TS hooks.ts (Untracked) X
src > redux > TS hooks.ts > ...
1-
1+ import {TypedUseSelectorHook, useDispatch, useSelector} from 'react-redux';
2+ import {AppDispatch, RootState} from './store';
3+
4+ // Use throughout your app instead of plain `useDispatch` and `useSelector`
5+ export const useAppDispatch = () => useDispatch<AppDispatch>();
6+ export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector;
7+
```

83

Et le nouveau composant counter

```
TS Counter.tsx (Working Tree) X
src > misc > TS Counter.tsx > ...

1 import React, {useState} from 'react';
2 import {Button, Text, View} from 'react-native';

3
4 const AppCounter = () => {
5   const [counter, setCounter] = useState(0);
6   const increment = () => {
7     setCounter(counter + 1);
8   };
9   return (
10     <View>
11       <Button
12         onPress={increment}
13         title="Increment"
14         color="#841584"
15         accessibilityLabel="Increment the counter"
16       />
17       <Text>Counter: {counter}</Text>
18     </View>
19   );
20 };
21
22 export default AppCounter;
23

1 import React, {useState} from 'react';
2 import {Button, Text, View} from 'react-native';
3+ import {useAppDispatch, useAppSelector} from '../redux/hooks';
4+ import {
5+   decrement,
6+   increment,
7+   selectCounter,
8+ } from '../redux/slices/counter.slice';
9
10 const AppCounter = () => {
11+   const counter = useAppSelector(selectCounter);
12+   const dispatch = useAppDispatch();
13+
14   return (
15     <View>
16       <Button
17         onPress={() => dispatch(increment(1))}
18         title="Increment"
19         color="#841584"
20         accessibilityLabel="Increment the counter"
21       />
22       <Button
23         onPress={() => dispatch(decrement(2))}
24         title="Decrement by 2"
25         color="green"
26         accessibilityLabel="Decrement the counter by 2"
27       />
28       <Text>Counter: {counter}</Text>
29     </View>
30   );
31 };
32
33 export default AppCounter;
34
```

84

- Introduction
- React Native
- Architecture d'application
- **Construire son interface**
- Les formulaires et la gestion des données
- Interagir avec le terminal
- Usages avancés

Les composants React Native

- 3 catégories :
 - composant natif (fourni par iOS ou Android)
 - composant de base (fourni par React Native)
 - composant que le développeur construit à partir des composants natifs et de base

86

Les composants de base (View, Text et Image) et leurs cycles de vie.

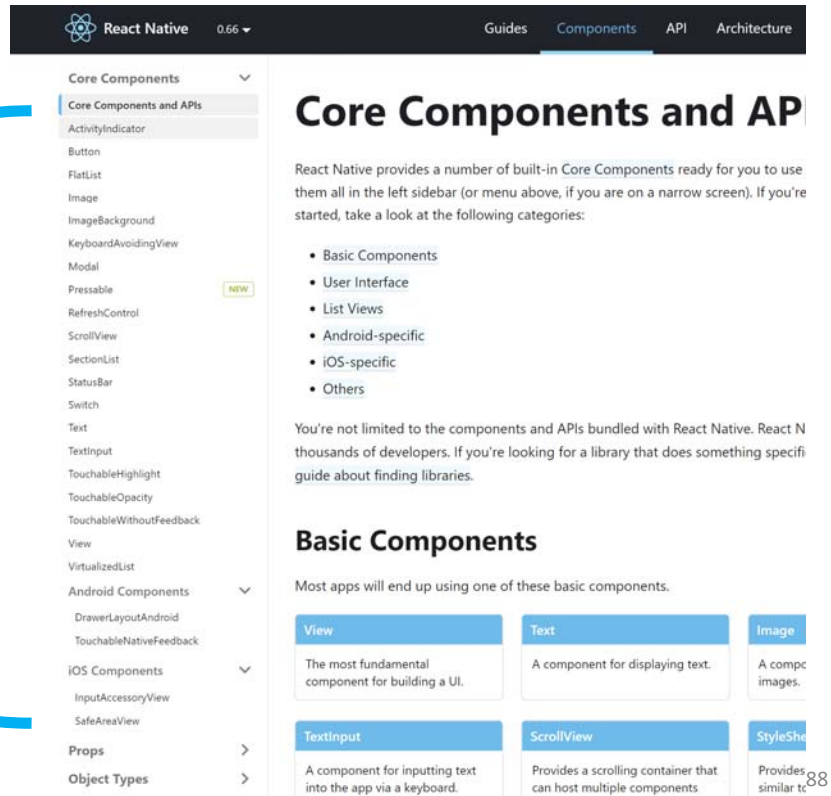
- React Native est muni de composant de base
- Pour chaque plateforme (android, iOS, web), ces composant vont être traduit en composant natif.

REACT NATIVE UI COMPONENT	ANDROID VIEW	IOS VIEW	WEB ANALOG	DESCRIPTION
<code><View></code>	<code><ViewGroup></code>	<code><UIView></code>	A non-scrolling <code><div></code>	A container that supports layout with flexbox, style, some touch handling, and accessibility controls
<code><Text></code>	<code><TextView></code>	<code><UITextView></code>	<code><p></code>	Displays, styles, and nests strings of text and even handles touch events
<code><Image></code>	<code><ImageView></code>	<code><UIImageView></code>	<code></code>	Displays different types of images
<code><ScrollView></code>	<code><ScrollView></code>	<code><UIScrollView></code>	<code><div></code>	A generic scrolling container that can contain multiple components and views
<code><TextInput></code>	<code><EditText></code>	<code><UITextField></code>	<code><input type="text"></code>	Allows the user to enter text

87

Liste des composants fournis d'office

Liste exhaustive ici



The screenshot shows the 'Components' section of the React Native documentation. On the left, a sidebar lists various components under the 'Core Components and APIs' category, which is highlighted with a blue bracket and the text 'Liste exhaustive ici'. The main content area is titled 'Core Components and APIs' and provides an overview of the built-in components. It lists categories such as Basic Components, User Interface, List Views, Android-specific, iOS-specific, and Others. Below this, the 'Basic Components' section is introduced, stating that most apps will use one of these basic components. A grid of six cards follows, each representing a basic component: View, Text, Image, TextInput, ScrollView, and StyleSheet. Each card includes a brief description of the component's function.

React Native 0.66 ▾ Guides Components API Architecture

Core Components and APIs

React Native provides a number of built-in Core Components ready for you to use them all in the left sidebar (or menu above, if you are on a narrow screen). If you're started, take a look at the following categories:

- Basic Components
- User Interface
- List Views
- Android-specific
- iOS-specific
- Others

You're not limited to the components and APIs bundled with React Native. React Native has thousands of developers. If you're looking for a library that does something specific, see [this guide about finding libraries](#).

Basic Components

Most apps will end up using one of these basic components.

View The most fundamental component for building a UI.	Text A component for displaying text.	Image A component for displaying images.
TextInput A component for inputting text into the app via a keyboard.	ScrollView Provides a scrolling container that can host multiple components.	StyleSheet Provides a way to style components.

Cycle de vie d'un composant

- `useEffect`
 - `componentDidMount`
 - `componentDidUpdate`
 - `componentWillUnmount`
- Error Boundaries
 - ne marche que pour les composants écrits en classe.
 - Penser à envelopper l'application avec un `trycatch`

89

Mise en place du ErrorBoundary

```
TS App.tsx (Working Tree) X
TS App.tsx > [0] App
18 import {Colors} from 'react-native';
19 import {Provider} from 'react-redux';
20 import AppBody from './src/layout/Body';
21 import AppFooter from './src/layout/Footer';
22 import AppHeader from './src/layout/Header';
23 import {store} from './src/redux/store';
24
25 const App = () => {
26   const isDarkMode = useColorScheme() === 'dark';
27   const backgroundColor = {
28     backgroundColor: isDarkMode ? Colors.darker : Colors.lighter,
29   };
30   return (
31     <Provider store={store}>
32       <SafeAreaView style={backgroundColor}>
33         <StatusBar barStyle={isDarkMode ? 'light-content' : 'dark-content'} />
34         <ErrorBoundary>
35           <ScrollView>
36             <AppHeader name="Gestion Stock" color="red" />
37             <AppBody />
38             <AppFooter />
39           </ScrollView>
40         </ErrorBoundary>
41       </SafeAreaView>
42     </Provider>
43   );
44 }
45 export default App;
```

90

Ajout d'un rendering error

```
TS Counter.tsx (Working Tree) X
src > misc > TS Counter.tsx > [0] AppCounter

1 import React, {useState} from 'react';
2 import {Button, Text, View} from 'react-native';
3 import {useAppDispatch, useAppSelector} from '../redux/hooks';
4 import {
5   decrement,
6   increment,
7   selectCounter,
8 } from '../redux/slices/counter.slice';
9
10 const AppCounter = () => {
11   const counter = useAppSelector(selectCounter);
12   const dispatch = useAppDispatch();
13
14   return (
15     <View>
16       <Button
17         onPress={() => dispatch(increment(1))}
18         title="Increment"
19         color="#841584"
20         accessibilityLabel="Increment the counter"
21       />
22       <Button
23         onPress={() => dispatch(decrement(2))}
24         title="Decrement by 2"
25         color="green"
26         accessibilityLabel="Decrement the counter by 2"
27       />
28       <Text>Counter: {counter}</Text>
29     </View>
30   );
31 };
32
33 export default AppCounter;

1 import React, {useState} from 'react';
2 import {Button, Text, View} from 'react-native';
3 import {useAppDispatch, useAppSelector} from '../redux/hooks';
4 import {
5   decrement,
6   increment,
7   selectCounter,
8 } from '../redux/slices/counter.slice';
9
10 const AppCounter = () => {
11   const counter = useAppSelector(selectCounter);
12   const dispatch = useAppDispatch();
13
14   if (counter > 15) {
15     throw new Error('CounterComponent: counter cannot be more than 15');
16   }
17
18   return (
19     <View>
20       <Button
21         onPress={() => dispatch(increment(1))}
22         title="Increment"
23         color="#841584"
24         accessibilityLabel="Increment the counter"
25       />
26       <Button
27         onPress={() => dispatch(decrement(2))}
28         title="Decrement by 2"
29         color="green"
30         accessibilityLabel="Decrement the counter by 2"
31       />
32       <Text>Counter: {counter}</Text>
33     </View>
34   );
35 };
36
37 export default AppCounter;
```

91

The ErrorBoundary component

```
TS ErrorBoundary.tsx (Untracked) X
src > misc > TS ErrorBoundary.tsx > ...
1-
1+ import React from 'react';
2+ import {ScrollView, Text, View} from 'react-native';
3+
4+ export class ErrorBoundary extends React.Component<{}, {hasError: boolean}> {
5+   constructor(props = {}) {
6+     super(props);
7+     this.state = {hasError: false};
8+   }
9+
10+   static getDerivedStateFromError(error: any) {
11+     // Update state so the next render will show the fallback UI.
12+     return {hasError: true};
13+   }
14+
15+   componentDidCatch(error: any, errorInfo: any) {
16+     console.log('error: ', error);
17+     console.log('errorInfo: ', errorInfo);
18+   }
19+
20+   render() {
21+     if (this.state.hasError) {
22+       return (
23+         <View>
24+           <Text>et zut... error...</Text>
25+         </View>
26+       );
27+     }
28+
29+     return this.props.children;
30+   }
31+ }
32+
```

92

Composant View

- C'est un composant qui permet d'encapsuler des enfants.
- Il utilise la layout flexbox (ressemble beaucoup au CSS flexbox)
- Une vue est Touchable
 - L'encadrer dans un composant TouchableHighlight
- Une vue est accessible
 - Tester avec Talkback

93

Faire une vue touchable

```
TS Counter.tsx (925cb547^ - 925cb547) X
1 import React, {useState} from 'react';
2 import {Button, Text, View} from 'react-native';
3 import {useAppDispatch, useAppSelector} from '../redux/hooks';
4 import {
5   decrement,
6   increment,
7   selectCounter,
8 } from '../redux/slices/counter.slice';
9
10 const AppCounter = () => {
11   const counter = useAppSelector(selectCounter);
12   const dispatch = useAppDispatch();
13
14   if (counter > 15) {
15     throw new Error('CounterComponent: counter cannot be more than 15');
16   }
17
18   return (
19     <View>
20       <Button
21         onPress={() => dispatch(increment(1))}
22         title="Increment"
23         color="#841584"
24         accessibilityLabel="Increment the counter"
25       />
26       <Button
27         onPress={() => dispatch(decrement(2))}
28         title="Decrement by 2"
29         color="green"
30         accessibilityLabel="Decrement by 2"
31       />
32     <Text>Counter: {counter}</Text>
33   );
34 }
35
```

```
1 import React, {useState} from 'react';
2 import {Button, Text, TouchableHighlight, View} from 'react-native';
3 import {useAppDispatch, useAppSelector} from '../redux/hooks';
4 import {
5   decrement,
6   increment,
7   selectCounter,
8 } from '../redux/slices/counter.slice';
9
10 const AppCounter = () => {
11   const counter = useAppSelector(selectCounter);
12   const dispatch = useAppDispatch();
13
14   if (counter > 15) {
15     throw new Error('CounterComponent: counter cannot be more than 15');
16   }
17
18   return (
19     <View>
20       <Button
21         onPress={() => dispatch(increment(1))}
22         title="Increment"
23         color="#841584"
24         accessibilityLabel="Increment the counter"
25       />
26       <TouchableHighlight
27         onPress={() => dispatch(decrement(2))}
28         onLongPress={() => dispatch(decrement(3))}
29       >
30         <View>
31           <Text>Tap here to decrement of 2...</Text>
32           <Text>Yes I am not a button but you can press me</Text>
33           <Text>
34             And if your press me for a long time, I will decrement 3...
35           </Text>
36         </View>
37       </TouchableHighlight>
38     <Text>Counter: {counter}</Text>
39   );
40 }
41
```

94

Faire une vue accessible

```
TS Counter.tsx (Working Tree) X
src > misc > TS Counter.tsx > [0] AppCounter
12 const dispatch = useAppDispatch();
13
14 if (counter > 15) {
15   throw new Error('CounterComponent: counter cannot be more than 15');
16 }
17
18 return (
19   <View>
20     <Button
21       onPress={() => dispatch(increment(1))}
22       title="Increment"
23       color="#841584"
24       accessibilityLabel="Increment the counter"
25     />
26     <TouchableHighlight
27       onPress={() => dispatch(decrement(2))}
28       onLongPress={() => dispatch(decrement(3))}
29     >
30       <View>
31         <Text>Tap here to decrement of 2 by short tap, and 3 by long tap</Text>
32         <Text>Yes I am not a button but you can press me</Text>
33         <Text>And if your press me for a long time, I will decrement 3...</Text>
34       </View>
35     </TouchableHighlight>
36   </View>
37   <Text>Counter: {counter}</Text>
38 </View>
39 );
40
41 export default AppCounter;
42
43
44
45
46
```

Note d'expérience: un test d'accessibilité devrait être fait avec Talkback, de préférence sur un vrai téléphone et non pas un émulateur.

95

Talkback

- Télécharger Google Accessible Suite
 - Besoin de Google Play Store
- Menu Settings > Accessibility > Talkback
 - Turn on talkback

Composant Text

- Utilisé pour afficher du texte en dur.
 - les labels, les paragraphes, etc.
- Un Text peut avoir un parent Text
 - Cela permet par exemple de spécialiser le style.
- Un Text est touchable
 - On peut faire des press courts, des press longs, etc.
- Un Text est accessible

97

Exemple de Text

```

27     onPress={() => dispatch(decrement(2))}
28     onLongPress={() => dispatch(decrement(3))}
29   </View>
30
31   accessible={true}
32   accessibilityLabel="decrement of 2 by short tap, and 3 by long tap"
33   <Text>Tap here to decrement of 2</Text>
34   <Text>Yes I am not a button but you can press me</Text>
35
36   And if your press me for a long time, I will decrement 3...
37
38   </Text>
39   </View>
40   </TouchableHighlight>
41   <Text>Counter: {counter}</Text>
42
43 </View>
44
45 );
46
47
48
49
50
51
52
53
54
55

```

```

27     onPress={() => dispatch(decrement(2))}
28     onLongPress={() => dispatch(decrement(3))}
29   </View>
30   style={{backgroundColor: 'gray'}}
31   accessible={true}
32   accessibilityLabel="decrement of 2 by short tap, and 3 by long tap"
33   <Text style={{fontWeight: 'bold', fontSize: 20}}>
34     Tap here to decrement of 2...{\n}
35   </Text>
36   <Text style={{color: 'blue'}}>
37     Tap here to decrement of 2...{\n}
38   </Text>
39   <Text style={{color: 'white'}}>
40     Yes I am not a button but you can press me{\n}
41   </Text>
42   <Text style={{color: 'red'}}>
43     And if your press me for a long time, I will decrement 3...
44   </Text>
45   </Text>
46   </View>
47   </TouchableHighlight>
48   <Text>
49     accessible={true}
50     accessibilityLabel={'The counter value is ' + counter}
51     style={{fontSize: 40}}
52     Counter: {counter}
53   </Text>
54   </View>
55
56 );
57
58
59
60
61
62
63
64
65
66

```

Le composant Image

- Marche pour les PNG mais pas les SVG
- Gère l'accessibilité

```
TS Header.tsx (Working Tree) X
src > layout > TS Header.tsx > ...
1 import React from 'react';
2- import {Text, View} from 'react-native';
3
4 const backgroundStyle = {
5   backgroundColor: 'white',
6- height: 50,
7 };
8
9 const AppHeader = (props: {name: string; color: string}) => {
10   return (
11     <View style={backgroundStyle}>
12
13       <Text style={{color: props.color}}>{props.name}</Text>
14     </View>
15   );
16 };
17 export default AppHeader;
18
1 import React from 'react';
2+ import {Image, Text, View} from 'react-native';
3
4 const backgroundStyle = {
5   backgroundColor: 'white',
6+ height: 80,
7+ padding: 10,
8 };
9
10 const AppHeader = (props: {name: string; color: string}) => {
11   return (
12     <View style={backgroundStyle}>
13+     <Image
14+       style={{height: 40, resizeMode: 'contain'}}
15+       source={require('../assets/logo.png')}
16+       accessible={false}
17+     />
18     <Text style={{color: props.color}}>{props.name}</Text>
19   </View>
20 );
21 };
22
23 export default AppHeader;
24
```

99

Le composant Button

```
20  <Button
21    onPress={() => {
22      dispatch(increment(1));
23      Alert.alert('you have incremented the counter');
24    }}
25    title="Increment"
26    color="#841584"
27    accessibilityLabel="Increment the counter"
28  />
```

100

Les composant « Touchables »

- TouchableHighlight
- TouchableNativeFeedback
- TouchableOpacity
- TouchableWithoutFeedback
- Attributs :
 - onPress
 - onLongPress

101

Les événements Touch

- Le Gesture Responder System
 - Plus compliqué que les composants « Touchables »

102

ListView

- FlatList
- SectionList

FlatList

```
TS ArticleList.tsx (Untracked) X
src > misc > TS ArticleList.tsx > ...
1-
1+ import React from 'react';
2+ import {FlatList, Text, View} from 'react-native';
3+
4+ const AppArticleList = () => {
5+   return (
6+     <View>
7+       <FlatList
8+         data={[
9+           {key: 'Tournevis'},
10+          {key: 'Marteau'},
11+          {key: 'Pince'},
12+          {key: 'Clé à molette'},
13+          {key: 'Tondeuse à gazon'},
14+        ]}
15+         renderItem={({item}) => <Text>{item.key}</Text>}
16+       />
17+     </View>
18+   );
19+ };
20+
21+ export default AppArticleList;
22+
```

104

SectionList

```
TS ArticleList.tsx X
src > misc > TS ArticleList.tsx > [?] AppArticleList > data
1 import React from 'react';
2 import {FlatList, SectionList, Text, View} from 'react-native';
3
4 const AppArticleList = () => {
5   return (
6     <View>
7       <SectionList
8         sections=[
9           {title: 'Petit matériel', data: ['Marteau', 'Pince', 'Règle']},
10          {
11            title: 'Gros matériel',
12            data: ['Tondeuse à gazon', 'Karcher'],
13          },
14        ],
15        renderItem={({item}) => <Text>{item}</Text>}
16        renderSectionHeader={({section}) => (
17          <Text style={{fontWeight: 'bold'}}>{section.title}</Text>
18        )}
19        keyExtractor={(item, index) => index + ''}
20      />
21    </View>
22  );
23 };
24
25 export default AppArticleList;
26
```

105

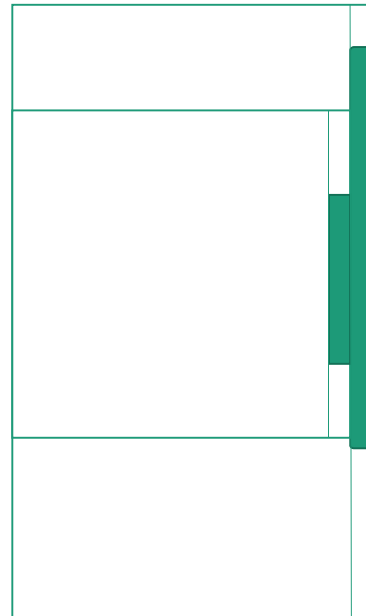
ScrollView

```
TS App.tsx x
TS App.tsx > [0] App
19 import {Provider} from 'react-redux';
20 import AppBody from './src/layout/Body';
21 import AppFooter from './src/layout/Footer';
22 import AppHeader from './src/layout/Header';
23 import {ErrorBoundary} from './src/misc/ErrorBoundary';
24 import {store} from './src/redux/store';
25
26 const App = () => {
27   const isDarkMode = useColorScheme() === 'dark';
28
29   const backgroundStyle = {
30     backgroundColor: isDarkMode ? Colors.darker : Colors.lighter,
31   };
32
33   return (
34     <Provider store={store}>
35       <SafeAreaView style={backgroundStyle}>
36         <StatusBar barStyle={isDarkMode ? 'light-content' : 'dark-content'} />
37         <ErrorBoundary>
38           <AppHeader name="Gestion Stock" color="red" />
39           <ScrollView
40             contentInsetAdjustmentBehavior="automatic"
41             style={backgroundStyle}>
42             <AppBody></AppBody>
43             <AppFooter></AppFooter>
44           </ScrollView>
45         </ErrorBoundary>
46       </SafeAreaView>
47     </Provider>
48   );
49 };
50
51 export default App;
52
```

106

ScrollView et Liste

- Attention un scrollview ne doit pas contenir de FlatList ou de SectionList
 - Ils sont incompatibles car pour un scroll donné, on ne peut pas savoir si c'est l'enfant ou le parent qui doit scroller



107

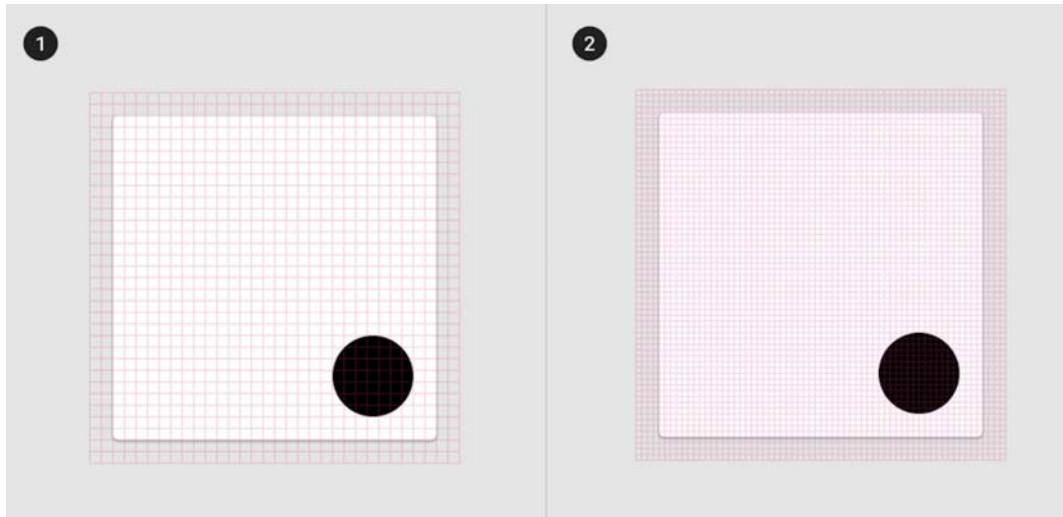
Organiser le layout de l'application

- Layout : comment l'application va agencer les différents composants ?
- React Native utilise le CSS, comme pour le web.
- Tous les composants utilisent une propriété « style », qui est un objet CSS.

108

density independant pixel

le dip est à peu près la même sur tous les téléphones (en principe)



109

Width et Height

- width et height s'expriment en « **density independant pixel** » (dip ou dp). Ce sont des javascript **numbers**.
- width et height peuvent aussi s'exprimer en pourcentage du parent (ou de l'écran si pas de parent)
- Seul les composants natifs de React Native ont un width et un height

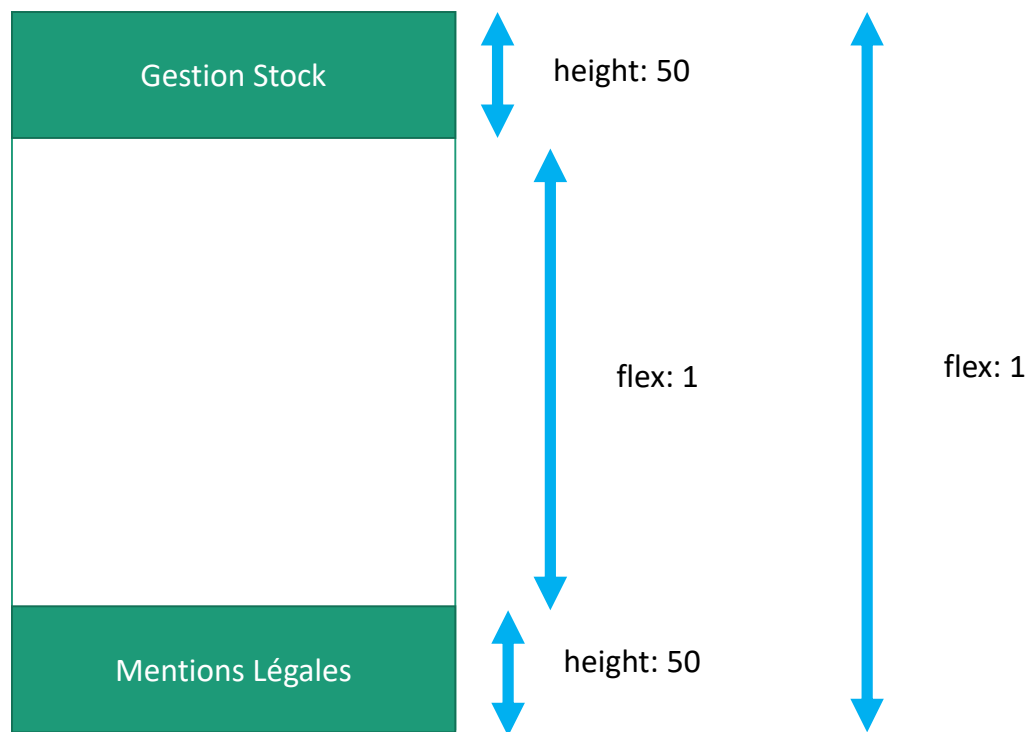
110

Mise en page avec Flexbox

- Tous les « rectangles » adoptent les propriétés flex comme en CSS avec quelques petites différences :
 - flexDirection = 'column'
 - flex-grow = flex
 - Noter que d'une certaine manière c'était déjà comme ça.

111

Exemple de flex



112

Styler les composants.

- Pour regrouper tout en gardant la vérification du typage dans l'IDE, il est recommandé d'utiliser la formule avec `StyleSheet.create()`

```
49  ∨ const style = StyleSheet.create({  
50  ∨    safeAreaView: {  
51    |    height: '100%',  
52    |    },  
53  ∨  });
```

`StyleSheet.create` est une fonction « identité » (elle renvoie son propre argument). Elle ne sert qu'à faire un contrôle de typage.

113

<https://stackoverflow.com/questions/38958888/react-native-what-is-the-benefit-of-using-stylesheet-vs-a-plain-object>

L'attribut style

- Tous les core composants de React Native comprennent le CSS via l'attribut **style**.
- Style peut accepter un objet CSS mais aussi un array d'objets. Les derniers écrasent les premiers.

```
34  ✓ | | | style={ [  
35  | | |   style.safeAreaView,  
36  | | |   {backgroundColor: isDarkMode ? Colors.darker : Colors.lighter},  
37  | | | ]}>
```

114

Utiliser des composants de la communauté.

- Installation : **npm i <librairie>**
- Parfois si la librairie contient du code natif (lié à Android ou iOS) il faut aussi déployer des informations de cette librairie dans les répertoires des projets natifs (iOS et Android)
 - **react-native link <librairie>**
- Toujours se reporter à la documentation de la librairie.

115

Exemple : mettre des icônes

- Librairie : <https://reactnativeelements.com/>
- npm install react-native-elements
- Attention cette librairie a des peerDependencies. Il faut les installer.
- Linker les librairies avec du code natif :
 - npx react-native link react-native-vector-icons
 - react-native link react-native-safe-area-context

```
82 |   "peerDependencies": {  
83 |     "react-native-safe-area-context": "^3.1.9",  
84 |     "react-native-vector-icons": ">7.0.0"  
85 |   },
```

116

Counter : mettre des icônes

Utiliser la balise <Icon>

```
37 <TouchableHighlight
38   onPress={() => dispatch(decrement(2))}
39   onLongPress={() => dispatch(decrement(3))}>
40   <View
41     style={styles.button}
42     accessible={true}
43     accessibilityLabel="decrement of 2 by short tap, and 3 by long tap">
44     <Icon
45       name="arrow-downward"
46       style={styles.icon}
47       color={styles.icon.color}
48       tvParallaxProperties={undefined}></Icon>
49     <Text style={styles.text}>Decrement by 2 (or 3)</Text>
50   </View>
51 </TouchableHighlight>
```

117

Une autre méthode de bouton icone

```
9 } from 'react-native';  
10+ import Icon from 'react-native-vector-icons/MaterialIcons';  
11 import {useAppDispatch, useAppSelector} from '../redux/hooks';
```

```
41 accessibilityLabel="decrement of 2 by short tap  
42+ <Icon.Button  
43+   onPress={() => dispatch(decrement(2))}  
44+   onLongPress={() => dispatch(decrement(3))}  
45   name="arrow-downward"  
46+   backgroundColor="blue">  
47+   Decrement by 2 (or 3)  
48+ </Icon.Button>
```

118

Exemple d'utilisation de font custom

- faire une fonte avec le site <https://fontello.com/>
- Mettre le fichier fontello.ttf dans le répertoire d'android
- Copier toute la fonte dans src/asset
- Créer le fichier Icon.tsx
- L'utiliser

```
android / app / src / main / assets / fonts  
fontello.ttf
```

```
1+ import {createIconSetFromFontello} from 'react-native-vector-icons';  
2+ import fontelloConfig from '../assets/custom-font/config.json';  
3+ export const Icon = createIconSetFromFontello(fontelloConfig);  
4+
```

119

Source : <https://github.com/oblador/react-native-vector-icons#custom-fonts>

Les différentes solutions de navigation entre les pages.

- react navigation : 22k sur Github
 - promu par l'équipe de react native
 - esprit d'application mobile (la librairie prend le pas sur le design)
- react router (for native) : 45k sur Github
 - marche de la même manière que le react router traditionnel

120

React Navigation

Installation de librairies :

```
npm install @react-navigation/native @react-navigation/native-stack  
npm install react-native-screens react-native-safe-area-context
```

Attention si vous avez un JDK version 8, ça va pas le faire. De même si vous avez un JDK 17. Non il faut vraiment un **JDK version 11** (comme c'est marqué dans la doc de react native)

```
10  
11 import (NavigationContainer) from '@react-navigation/native';  
12 import React from 'react';  
13 import {  
14   SafeAreaView,  
15   StatusBar,  
16   StyleSheet,  
17   useColorScheme,  
18   View,  
19 } from 'react-native';  
20 import {Colors} from 'react-native/Libraries/Colors';  
21 import {Provider} from 'react-redux';  
22 import AppBody from './src/layout/Body';  
23 import AppFooter from './src/layout/Footer';  
24 import AppHeader from './src/layout/Header';  
25 import {ErrorBoundary} from './src/misc/ErrorBoundary';  
26 import {store} from './src/redux/store';  
27  
28 const App = () => {  
29   const isDarkMode = useColorScheme() === 'dark';  
30  
31   return (  
32     <Provider store={store}>  
33       <SafeAreaView  
34         style={isDarkMode ? Colors.darker : Colors.lighter}>  
35         <AppHeader />  
36         <AppBody />  
37         <AppFooter />  
38       </SafeAreaView>  
39     </Provider>  
40   );  
41 }  
42  
43 <NavigationContainer>  
44   <App />  
45 </NavigationContainer>
```

121

Source : <https://reactnative.dev/docs/navigation>

Installation de react navigation

- Ne pas oublier de modifier le fichier MainActivity.java
- Voir détails à :
<https://reactnavigation.org/docs/getting-started>
- Tips : le faire avec Android Studio.
 - Formatter le fichier avec CTRL+ALT+L

122

Composant avancés

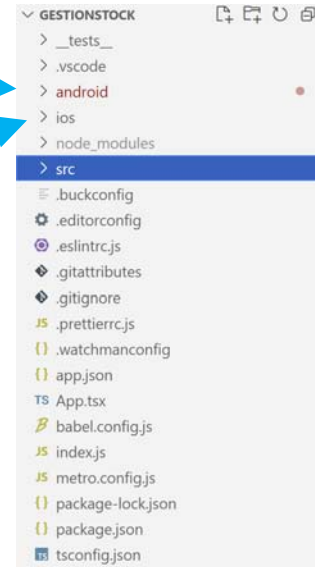
répertoire android

répertoire ios

`npx react-native link <dependencies>`

ou

installation manuel des spécificités de chaque projet.



123

Ajouter des animations et des transitions.

- Réalisé avec l'API « Animated ».
- Principe : on fait bouger dans le temps des paramètres props de composant (style, etc.)
- Possibilité d'utiliser le driver natif (pour les performances)

124

Exemple

```
TS Secrettsx X
src > misc > TS Secrettsx > [0] AppSecret
1 import React, {useRef, useState} from 'react';
2 import {Animated, Button, StyleSheet, Text, View} from 'react-native';
3
4 const AppSecret = () => {
5   const [show, setShow] = useState(false);
6   const fadeAnim = useRef(new Animated.Value(0)).current;
7   return (
8     <View>
9       <Button
10         onPress={() => {
11           const newShow = !show;
12           setShow(newShow);
13           Animated.timing(fadeAnim, {
14             toValue: newShow ? 1 : 0,
15             duration: 2000,
16             useNativeDriver: false,
17           }).start();
18         }}
19         title={show ? 'Hide Secret' : 'Show Secret'}
20         color="#841584"
21       />
22     <Animated.Text
23       style={[
24         styles.text,
25         {
26           opacity: fadeAnim,
27         }
28       ]>
29       This is my secret
30     </Animated.Text>
31   </View>
32 );
33 };
34
35 export default AppSecret;
36
37 const styles = StyleSheet.create({
38   text: {
39     fontWeight: 'bold',
40     fontSize: 20,
41   },
42 });
43
```

125

- Introduction
- React Native
- Architecture d'application
- Construire son interface
- **Les formulaires et la gestion des données**
- Interagir avec le terminal
- Usages avancés

126

Les principaux composants de formulaires.

- React Native a un seul « core component » :
TextInput
- Pour faire davantage comme le web (form), il existe des librairies :
 - **React Hook Form**
 - Formik
 - etc.

127

TextInput

128

Récupération des données : xmlhttprequest et fetch.

- Dans l'absolu préférer fetch à xmlhttprequest.
- Utilisation identique au fetch du web.
- Note Android : localhost est 10.0.2.2

129

Exemple fetch

```
TS Clocktsx  X
src > misc > TS Clocktsx > ...
1  import React, {useEffect, useState} from 'react';
2  import {Button, Text, View} from 'react-native';
3
4  const AppClock = () => {
5    const [date, setDate] = useState('...');
6    const refresh = () => {
7      (async () => {
8        try {
9          const response = await fetch('http://10.0.2.2:3000/api/date');
10         const json = await response.json();
11         setDate(json.date);
12       } catch (err) {
13         console.log('err: ', err);
14       }
15     })();
16   };
17
18   useEffect(() => {
19     refresh();
20   }, []);
21
22   return (
23     <View>
24       <Text>What time is it? It is : {date}</Text>
25       <Button onPress={refresh} title="Refresh" />
26     </View>
27   );
28 };
29
30 export default AppClock;
31
```

130

Le stockage local

- Il existait une API interne AsyncStorage à React Native mais elle est deprecated.
- Utiliser une API externe. La plus connue est
 - `@react-native-async-storage/async-storage`

131

Exemple

```
TS PersistentCounter.tsx X
src > misc > TS PersistentCounter.tsx > [9] AppPersistentCounter
1 import AsyncStorage from '@react-native-async-storage/async-storage';
2 import React, {useEffect, useState} from 'react';
3 import {Button, StyleSheet, Text, View} from 'react-native';
4
5 const AppPersistentCounter = () => {
6   const [counter, setCounter] = useState(20);
7   const set = (newCounter: number) => {
8     setCounter(newCounter);
9     (async () => {
10       try {
11         await AsyncStorage.setItem('@counter', newCounter + '');
12       } catch (err) {
13         console.log('err: ', err);
14       }
15     })();
16   };
17
18   useEffect(() => {
19     (async () => {
20       try {
21         const str = await AsyncStorage.getItem('@counter');
22         if (str === null) {
23           return;
24         }
25         const c = +str;
26         setCounter(c);
27       } catch (err) {
28         console.log('err: ', err);
29       }
30     })();
31   }, []);
32
33   return (
34     <View>
35       <Button
36         onPress={() => set(counter + 1)}
37         title="Increment"

```

132

La gestion offline

- Identifier quand l'application est offline et l'indiquer à l'utilisateur.
- Synchroniser l'état sur le back-end et l'état sur le front-end

133

Exemple de Netinfo

```
TS NetworkStatus.tsx X
src > misc > TS NetworkStatus.tsx > ...
1 import NetInfo, {NetInfoState} from '@react-native-community/netinfo';
2 import React, {useEffect, useState} from 'react';
3 import {Text, View} from 'react-native';
4
5 const AppNetworkStatus = () => {
6   const [state, setState] = useState<NetInfoState | undefined>(undefined);
7   useEffect(() => {
8     const unsubscribe = NetInfo.addEventListener(state => {
9       console.log('Connection type', state.type);
10      console.log('Is connected?', state.isConnected);
11      setState(state);
12    });
13    return function cleanup() {
14      unsubscribe();
15    };
16  }, []);
17  return (
18    <View>
19      {state === undefined ? (
20        <Text>network status state not defined</Text>
21      ) : (
22        <>
23          <Text>Network Status: {state.type}</Text>
24          <Text>
25            Network Is Connected: {state.isConnected ? 'true' : 'false'}
26          </Text>
27        </>
28      )}
29    </View>
30  );
31 };
32
33 export default AppNetworkStatus;
34
```

134

- Introduction
- React Native
- Architecture d'application
- Construire son interface
- Les formulaires et la gestion des données
- **Interagir avec le terminal**
- Usages avancés

135

Les principales API natives de React Native

L'ensemble est documenté sur le site de react native : <https://reactnative.dev/docs/alert>

APIs

AccessibilityInfo

Alert
Animated
Animated.Value
Animated.ValueXY
Appearance
AppRegistry
AppState
DevSettings
Dimensions
Easing
InteractionManager
Keyboard
LayoutAnimation
Linking
PanResponder
PixelRatio
Platform
PlatformColor
RootTag
Share
StyleSheet
SysTrace

Transforms

Vibration

Hooks

useColorScheme
useWindowDimensions

Android APIs

BackHandler
PermissionsAndroid
ToastAndroid

iOS APIs

ActionSheetIOS
DynamicColorIOS
Settings

Développer un module natif

- Module natif = module écrit en code natif Android ou iOS.
 - Conçu pour être appelé depuis javascript.
 - Cela donne la possibilité d'écrire en natif quelque chose puis de l'appeler depuis React Native.
- <https://reactnative.dev/docs/native-modules-android>

137

- Introduction
- React Native
- Architecture d'application
- Construire son interface
- Les formulaires et la gestion des données
- Interagir avec le terminal
- **Usages avancés**

138

Best Practices et erreurs fréquentes

- Maquetter avant de développer
 - Wireframe
- Code Typing:
 - Utiliser Typescript
 - Styles : le séparer du composant et utiliser StyleSheet.create
- Tableau : bien utiliser l'attribut key.
- Décomposer pour réutiliser et modulariser
 - Ecrire du code testable
- Couleurs : utiliser que des palettes
 - primary, secondary, etc.
 - penser avec un système de design (Material, Eva, etc.)
- react-native link
 - Ne pas oublier de linker une librairie quand elle possède des éléments natif
- Utiliser des fontes quand c'est possible (fontello)
- Attention aux ressources statique : ne pas oublier d'utiliser require
- Bien vérifier le JDK utilisé (openJDK11)
- Wiper les data du téléphone
- Débugger les console.log

139

Tests unitaires et fonctionnels

- Analyse statique :
 - linter : eslint
 - type checking (Typescript)
- Ecrire du code testable :
 - Modulariser le code
 - Séparer la logique du business de l'affichage
 - Gestion d'état par une librairie séparée (Redux, MobX, etc.)
 - Un composant ne doit pas gérer un état mais rendre visible un état. Tout au plus il envoie une action qui va modifier un état.
- Test unitaire
 - But est de tester une pièce de manière individuelle. Parfois il faut mocker.
 - Tester avec un framework de test : Jest
 - Given, When, Then
 - AAA (Arrange, Act, Assert)
- Test d'intégration
 - But est de tester le bon assemblage des pièces.
 - Appium, Detox

140

Publier l'application

- Android : fabriquer un fichier APK en release
 - Générer un fichier **keystore** (java keytool)

```
keytool -genkey -v -keystore production.keystore -alias prod -keyalg RSA -keysize 2048 -validity 10000
```

- Lier le fichier keystore dans le **android/app/build.gradle**
- Ajouter les ressources dans les assets

```
npx react-native bundle --platform android --dev false --entry-file index.js --bundle-output  
android/app/src/main/assets/index.android.bundle --assets-dest android/app/src/main/res/
```

- Dans le répertoire **android**, Lancer la commande **gradlew assembleRelease**
- Publier l'application sur le Play Store (ou sur une autre store) selon les modalités indiquées par le constructeur.
(25\$ une fois pour toute pour créer un compte Google Play Store)

141

Créer un compte sur la Play Store : <https://play.google.com/console/signup>

Blog pour créer un APK : <https://instamobile.io/android-development/generate-react-native-release-build-android/>

Publier l'application

- iOS : <https://reactnative.dev/docs/publishing-to-app-store>

142

Mises à jour Over The Air.

- OTA : Over The Air.
- Technologie Microsoft Open Source permettant de mettre à jour la partie javascript d'une application React Native.
- Avantage : bypass the Play Store ou le Apple Store.
 - donc mise à jour plus rapide
 - MAIS : ne doit concerner que les bugs (limitation juridique)
- Inconvénient : limité à javascript et non aux parties natives d'une application.

143

Voir : <https://pagepro.co/blog/react-native-over-the-air-updates/>

Frameworks et outils complémentaires.

- **NativeBase** : des tonnes de composants. C'est le bootstrap de React Native.
- **Teaset** : encore d'autres composants
- **Material Kit React Native** : Google Material Design dans React Native.
- **React Native Elements** : Des composants réalisés par des développeurs. La plateforme web est incluse.
- **AirBnB lottie** : pour les animations Adobe After Effect
- **React Native Vector Icons** : pour les fontes d'icônes.
- **Ignite CLI** : Objectif de remplacer react-native ou expo pour gérer le code d'un projet.
- **React Native MapView** : afficher des google maps.

144

Pour aller plus loin : <https://www.codeinwp.com/blog/react-native-component-libraries/>

Fin !



jlguenego@gmail.com

145