DOCUMENT TECHNIQUE REACT NATIVE

Table des matières

[Table des Figures 3](#_Toc86239413)

[**CHAPITRE 1 : SET UP POUR L’ENVIRONEMMENT WINDOWS** 4](#_Toc86239414)

[**Configuration pour Windows :** 4](#_Toc86239415)

[**CHAPITRE 2 : BONJOUR LE MONDE** 6](#_Toc86239416)

[**Bonjour Tout le monde!** 7](#_Toc86239417)

[**Chapitre 3 : Coiffant (stylesheet)** 8](#_Toc86239418)

[**Introduction :** 8](#_Toc86239419)

[**Syntaxe :** 9](#_Toc86239420)

[**Remarques :** 9](#_Toc86239421)

[**Examples :** 9](#_Toc86239422)

[**Chapitre 6 : Composants** 10](#_Toc86239423)

[Examples : 10](#_Toc86239424)

[Composant de base : 11](#_Toc86239425)

[Composant avec état 11](#_Toc86239426)

[Composant sans état ( composant without state) 12](#_Toc86239427)

[**Chapitre 7 : Disposition** 13](#_Toc86239428)

[**Flexbox** : 13](#_Toc86239429)

[**Chaptire 8 : States ( les états)** 15](#_Toc86239430)

[**Syntaxe :** 15](#_Toc86239431)

[**Examples :** 15](#_Toc86239432)

[**SetState :** 15](#_Toc86239433)

[**Chapitre 10 : Exécuter une application sur l’appareil (version android ) & Expo** 17](#_Toc86239434)

[**Exécuter sur l’appareil :** 17](#_Toc86239435)

[**Chapitre 11 : Images** 19](#_Toc86239436)

[Examples : 19](#_Toc86239437)

[Module d’image 19](#_Toc86239438)

[ Image locale  et Image externe : 19](#_Toc86239439)

[**Source d’image conditionnelle :** 20](#_Toc86239440)

[**Screen :** 21](#_Toc86239441)

[21](#_Toc86239442)

[**Chapitre 12 : Instructions en ligne de commande** 21](#_Toc86239443)

[**Examples :** 21](#_Toc86239444)

[**Vérifier la version** 21](#_Toc86239445)

[**Examples de sortie** 21](#_Toc86239446)

[**Chapitre 13 : Le débogage** 22](#_Toc86239447)

# Table des Figures

[Figure 1:react native logo 4](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239448)

[Figure 2: Hello World 6](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239449)

[Figure 3: Hello World Screen 7](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239450)

[Figure 4: basic component 10](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239451)

[Figure 5: component avec état 10](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239452)

[Figure 6: composant sans état 11](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239453)

[Figure 7: flexdirection : 'row' 12](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239454)

[Figure 8 : flexdirection row screen 13](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239455)

[Figure 9: counter add 14](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239456)

[Figure 10: counter add screen 15](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239457)

[Figure 11: adb devices 16](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239458)

[Figure 12: expo start server 17](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239459)

[Figure 13: expo run on android emulator 17](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239460)

[Figure 14: image locale et externe 18](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239461)

[Figure 15: image locale & externe screen 19](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239462)

[Figure 16: image conditionnelle 19](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239463)

[Figure 17: image conditionnelle screen 20](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239464)

[Figure 18: react native check version 20](file:///C:\Users\ahafdi\Desktop\react%20native\ignite\PizzaApp\documentation_RN\documentation\DOCUMENT%20TECHNIQUE%20REACT%20NATIVE.docx#_Toc86239465)

# **CHAPITRE 1 : SET UP POUR L’ENVIRONEMMENT WINDOWS**

**React Native** vous permet de créer des applications mobiles en utilisant uniquement JavaScript. Il utilise la même conception que React, vous permettant de composer une interface utilisateur mobile riche à partir de composants déclaratifs. Avec React Native, vous ne créez pas une «application Web mobile», une «application HTML5» ou une «application hybride».

Figure :react native logo

Vous construisez une véritable application mobile qui ne se distingue pas d'une application créée avec Objective-C ou Java. React Native utilise les mêmes éléments fondamentaux que les applications iOS et Android classiques. Vous venez de mettre ces blocs de construction ensemble en utilisant JavaScript et React.

Il est open-source et maintenu par Facebook.

• Site Internet (https://reactnative.dev/)

• Documentation (https://reactnative.dev/docs/getting-started)

• GitHub Repository(<https://github.com/facebook/react-native>)

## **Configuration pour Windows :**

**Remarque:** vous ne pouvez pas développer d’applications réactives pour iOS sur Windows, mais uniquement des applications Android réactives.

**Outils / Environnement**

* Windows 10
* outil de ligne de commande (par exemple, ligne de commande Powershell ou Windows)
* Chocolaté ( étapes pour configurer via PowerShell via ce lien https://blog.fbalashov.com/2016/07/react-native-android-apps-on-windows.html#setup-choco)
* Le JDK (version 8)
* Studio Android Une machine Intel avec la technologie de virtualisation activée pour HAXM (facultatif, nécessaire uniquement si vous souhaitez utiliser un émulateur)

1. **Configurez votre machine pour réagir au développement natif :**

Démarrez la ligne de commande en tant qu'administrateur exécutez les commandes suivantes:

***choco install nodejs.install***

***choco install python2***

Redémarrez la ligne de commande en tant qu'administrateur pour pouvoir exécuter npm

***npm install -g react-native-cli***

1. **Définissez vos variables d'environnement :**

Ouvrez la fenêtre Variables d'environnement en naviguant vers: [Clic droit] Menu "Démarrer" -> Système -> Paramètres système avancés -> Variables d'environnement Dans la section inférieure, recherchez la variable système "Path" et ajoutez l'emplacement d'installation de react-native à l'étape 1. Si vous n'avez pas ajouté de variable d'environnement ANDROID\_HOME, vous devrez également le faire ici. Dans la fenêtre "Variables d'environnement", ajoutez une nouvelle variable système nommée "ANDROID\_HOME" et la valeur correspondant au chemin d'accès à votre SDK Android. Redémarrez ensuite la ligne de commande en tant qu'administrateur pour pouvoir y exécuter des commandes réactives.

1. **Créez votre projet**

Créez votre projet En ligne de commande, accédez au dossier dans lequel vous souhaitez placer votre projet et exécutez la commande suivante:

**react-native init ProjectName**

1. **Lancez votre projet**

Démarrez un émulateur depuis Android Studio Accédez au répertoire racine de votre projet en ligne de commande et exécutez-le:

***cd ProjectName***

***react-native run-android***

# **CHAPITRE 2 : BONJOUR LE MONDE**

On crée un projet avec la commande :

***expo init Hello World***

on lance le project avec la commande : (on lance la commande sur le path du projet )

***expo start***

on ouvrit le fichier App.js

Après on écrit *<Text>Bonjour Tout le Monde </Text> :*

On rafraichit le fichier avec la commande « control + S »

Et voila Félicitations! Vous avez écrit avec succès votre premier Hello World!

### **Bonjour Tout le monde!**

****

Figure : Hello World

Vous devriez voir Hello World! écrit à l'écran!

****

Figure : Hello World Screen

# **Chapitre 3 : Coiffant (stylesheet)**

### **Introduction :**

Les styles sont définis dans un objet JSON avec des noms d'attributs de style similaires, comme dans CSS. Un tel objet peut soit être mis en ligne dans le style prop d'un composant, soit être transmis à la fonction StyleSheet.create(StyleObject) et être stocké dans une variable pour un accès en ligne plus court en utilisant un nom de sélecteur similaire à une classe. en CSS.

### **Syntaxe :**

* <Component style={styleFormaStyleSheet }/>
* <Component style={styleObject }/>
* <Component style={style1,style2 }/>

### **Remarques :**

La plupart des styles React Native sont leurs formulaires CSS, mais dans un cas camel. Ainsi, la text-decoration devient textDecoration . Contrairement aux CSS, les styles ne sont pas hérités. Si vous souhaitez que les composants enfants héritent d'un certain style, vous devez le fournir explicitement à l'enfant. Cela signifie que vous ne pouvez pas définir une famille de polices pour une View entière. La seule exception à cette règle est le composant Text : les Text imbriqués héritent de leurs styles parents.

### **Examples :**

Il y a plusieurs manière pour définir un style :

* Style en ligne :

***<Text style={{color : ‘red}} >Red Text </Text>***

* Styling à l’aide d’une feuille de style :

***Const styles = StyleSheet.create {{***

***red : {***

***color : ‘red’***

***},***

***}}***

* Styling à l’aide de plusieurs styles :

const styles = StyleSheet.create({

red: { color: 'red' },

greenUnderline: { color: 'green',

textDecoration: 'underline' },

big: { fontSize: 30 } });

//code of the class

<Text style={{styles.red,styles.big > Big red </Text>

<Text style={{styles.red,styles.greenUnderline > Big red </Text>

# **Chapitre 6 : Composants**

**Examples :**

### Composant de base :



Figure : basic component

Composant avec état

Ces composants auront des états changeants.(componant with states)

On utiliser un constructor pour créer un état / state car on utilise ici une Classe comme un composant



Figure : component avec état

### Composant sans état ( composant without state)

(ce sont les fonctions , funtional componants are stateless , class componants are statefull )

Figure : composant sans état

# **Chapitre 7 : Disposition**

Examples :

### **Flexbox** :

Flexbox est une mode de mise en page permettant la disposition des éléments sur une page de manière à ce que les éléments se comportent de manière prévisible lorsque la mise en page doit

Prendre en charge différents tailles d’écran et différents périphériques d’affichage .par défaut ,

Flexbox représente les enfants (childreen ) sous forme d’une colonne (column), Mais on peut le changer en utilisant « flexDirection : ‘row » dans le styling .

**flexDirection**

Figure : flexdirection : 'row'

**screen :**

****

Figure : flexdirection row screen

Ila ya plusieurs possibilité pour changer vers une colonne inversé ou ligne inversé (flexDirection : ‘row-reverse ‘ or ‘column-reverse ‘)

# **Chaptire 8 : States ( les états)**

### **Syntaxe :**

* Void **setState** (fonction |object nextState , [rappel de fonction])

### **Examples :**

Pour créer des applications qui contient par exemple des composants qui ont besoin d’étre changé instantanément et étre affiché sur l’écran , alors React Native offre les états ou states ,

Pour modifer la vue d’une application , on peut utiliser **setState** ,setState effectue une fusion superficielle entre l’état nouveau et précédent et déclaneche un re-render du composant .

Pour bien expliquer setState et comment ça fonctionne je vais prendre l’exemple d’un compteur avec button

### **SetState :**



Figure : counter add

**Screen :**

****

Figure : counter add screen

# **Chapitre 10 : Exécuter une application sur l’appareil (version android ) & Expo**

### **Exécuter sur l’appareil :**

1. Adb devices ( cette commande pour afficher tous les appareil /emulator connectés à l’ordinateur



Figure : adb devices

1. emulator -avd Pixel\_XL\_API\_30 ( lancer l’emulator /simulateur)
2. react-native run-android ( exécuter le projet sur le simulateur android)
3. pour Expo (j’utilise expo sur cette documentation )
4. expo start ( on lance cette commande sur la destination du projet pour lancer le serveur d’Expo)

Figure : expo start server

1. On clique sur ‘ run on Android device/emulator’ :

Figure : expo run on android emulator

# **Chapitre 11 : Images**

### Examples :

### Module d’image

On peut importer Image à partir du package « react-native »

* Image locale  et Image externe :

Figure : image locale et externe

* **Screen :**

****

Figure : image locale & externe screen

**Source d’image conditionnelle :**

Figure : image conditionnelle

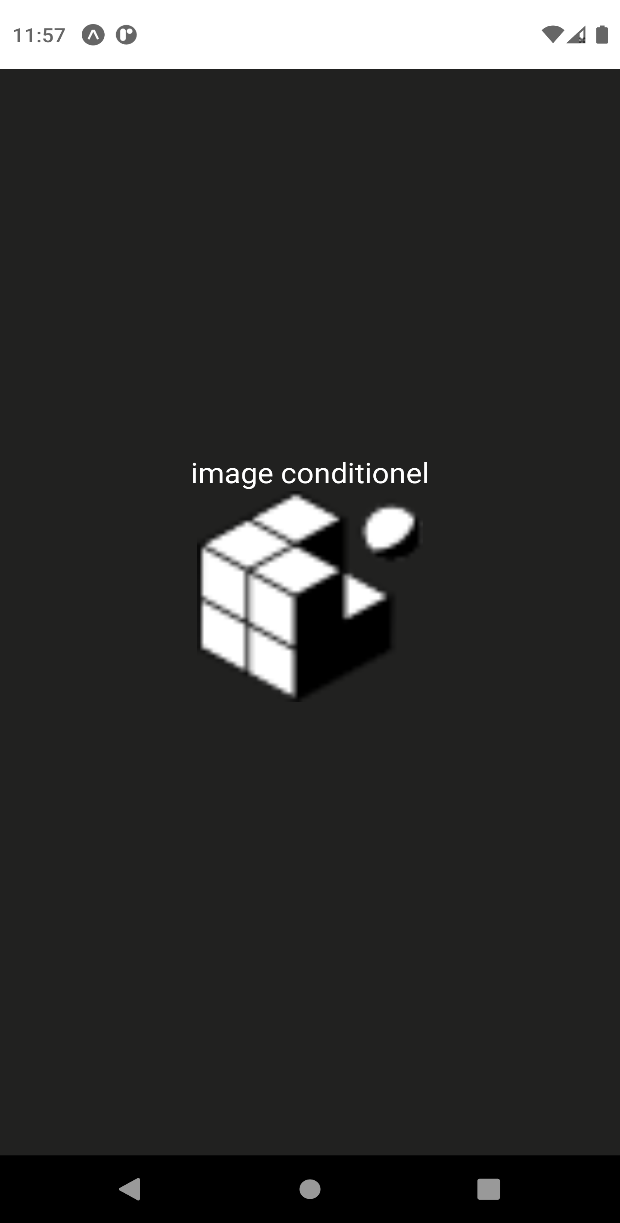
**Screen :**

Figure : image conditionnelle screen

# **Chapitre 12 : Instructions en ligne de commande**

### **Examples :**

### **Vérifier la version**

* **$ react-native -v**

### **Examples de sortie**

Figure : react native check version

Pour Initialiser

* **$ react-native init MyFirstProjectRN**
* **$ expo init MyFirstProjectEXPO**

Pour courir pour Android

* **$ cd MyFirstProjectRN**
* **$ react-native run-android**

# **Chapitre 13 : Le débogage**

Syntaxe :

* Débogeur