DOCUMENT TECHNIQUE REACT NATIVE

Table des matières

[**CHAPITRE 1 : SET UP POUR L’ENVIRONEMMENT WINDOWS** 3](#_Toc85816851)

[**Configuration pour Windows :** 3](#_Toc85816852)

[**CHAPITRE 2 : BONJOUR LE MONDE** 5](#_Toc85816853)

[**Bonjour Tout le monde!** 5](#_Toc85816854)

[**Chapitre 3 : Coiffant (stylesheet)** 7](#_Toc85816855)

[Introduction : 7](#_Toc85816856)

[Syntaxe : 7](#_Toc85816857)

[Remarques : 7](#_Toc85816858)

# **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».

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!**

****

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

****

# **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 :

Composant avec état

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

On utiliser un constructor pour cree un état / state car on utilise ici une Classe comme un componant

