Avec Expo, on va pouvoir apprendre tous ces concepts directement dans un éditeur en ligne : [https://snack.expo.io/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa20tWmhLd2ZLcXZKZTExOXdlaDV0Z3BsREhkZ3xBQ3Jtc0trQWE3Z3F1WXdTVmowbWN3cGtERTNqV2o1Q0lfNVVwWm1tWWJFbi00RTB4d3pNc1luRlFCRjBRdEs2NnNPelVDbzJWQzE5U25yQkx1MFdCV0g4cDJZOG5xcEhqNVAxQk5PZUJvb1YzejdCV3JJelhRUQ&q=https%3A%2F%2Fsnack.expo.io%2F)

Installe React Native sur votre environnement :

<https://www.youtube.com/watch?v=V4foI7A6PzM>

Une fois React Native installé sur votre environnement, vous pouvez récupérer le code de l’application, disponible sur [le dépôt GitHub](https://github.com/Gaellan/react-native-quotes).

git clone <https://github.com/Gaellan/react-native-quotes.git>

Ensuite il faut installer les dépendances, il y a 3 librairies utilisées :

* [react-native-elevated-view](https://github.com/alekhurst/react-native-elevated-view) : est une librairie qui permet d’obtenir l’effet d’élévation typique d’Android aussi bien sur iOS qu’Android.
* [react-native-responsive-dimensions](https://github.com/DaniAkash/react-native-responsive-dimensions) : est une librairie qui permet d’obtenir des des hauteurs, largeurs et tailles de police responsive quelle que soit la taille et la résolution d’écran.
* [react-native-vector-icons](https://github.com/oblador/react-native-vector-icons) : est une librairie qui permet d’utiliser les icônes des principales librairies d’icônes vectorielles (Font Awesome, Material Icons, …)

cd react-native-quotes/ReactNativeQuotesnpm installreact-native link

Une fois les dépendances installées, il ne reste plus qu’à lancer l’application :

react-native run-android

Ou

react-native run-ios

L’organisation des fichiers :

* Le dossier *\_\_tests\_\_* est installé par React Native et contient le test unitaire de l’application.
* Le dossier *android* est installé par React Native et contient le code Java natif pour votre application Android.
* Le dossier *ios* est installé par React Native et contient le code Objective-C natif pour votre application iOS.
* Le dossier *node\_modules* est installé par npm et contient les dépendances Javascript.
* Le dossier *components* est créé par moi, il contient le code des composants réutilisables (ici le bouton).
* Le dossier *containers* est créé par moi, il contient le code des écrans (ici l’écran principal).
* Le fichier *index.js* charge l’application via le fichier *App.js*.
* Le fichier *App.js* charge l’écran principal.

L’écran principal :

Ici le constructor charge les props de base fournies par le Component, puis on initialise l’état de l’écran principal.

Ici, la fonction *getQuote* fait appel à [l ‘API Fetch](https://developer.mozilla.org/fr/docs/Web/API/Fetch_API) pour interroger l’API, elle retourne une promesse qui résout la [Response](https://developer.mozilla.org/fr/docs/Web/API/Response" \t "_blank) de l’API. Dans la Response, nous récupérons la citation et son auteur, et nous en servons pour mettre à jour l’état de l’écran principal.

La méthode *componentWillMount* est appelée à chaque fois que l’écran est chargé ou rechargé, elle appelle *getQuote* pour charger la citation et son auteur.

Dans la première partie (condition if)de la méthode *render*, si la citation ou l’auteur n’est pas initialisé, on va afficher un loader ([ActivityIndicator](https://facebook.github.io/react-native/docs/activityindicator.html" \t "_blank)) et le bouton.

Dans la seconde partie (condition else), on affiche la citation, son auteur et le bouton refresh.

Les styles de l’écran principal :

Les valeurs utilisées pour [les styles](https://facebook.github.io/react-native/docs/style.html) sont très similaires à celles utilisées en CSS, mais les noms sont en camel case. *justifyContent* et *alignItems* permettent d’aligner les éléments verticalement et horizontalement.

C’est ici que l’on utilise react-native-responsive-dimensions, *responsiveWidth(100)* correspond à 100% de la largeur, *responsiveHeight(2)* à 2% de la hauteur, etc etc.

Le bouton refresh :

Dans l’écran principal, nous avions passé une prop “action” au bouton :

action={this.getQuote.bind(this)}

Cette action est la fonction *getQuote* à laquelle nous lions le contexte de l’écran principal (pour lui permettra de manipuler l’état de l’écran principal alors qu’elle est appelée depuis un autre composant). Les composant “[Touchable](https://facebook.github.io/react-native/docs/handling-touches.html)” sont des wrappers qui permettent de rendre des zones cliquables. On attache donc la méthode *getQuote* au clic sur le bouton, en utilisant le *onPress* du TouchableOpacity.