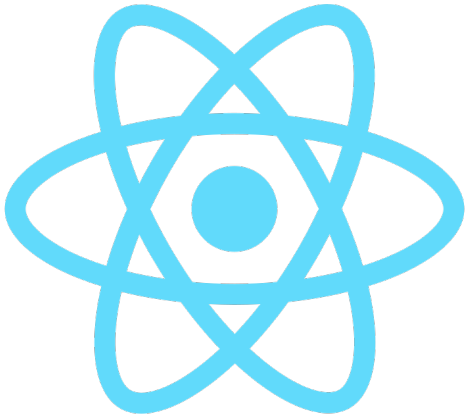


React Native

Un introduction, il y a plein d'autres choses à voir !

Retour à nos moutons : Applications mobiles multi-plateformes

- Les fonctionnalités sont généralement similaires sur les différentes plateformes. Lorsque les applications ciblent plusieurs plateformes et les même fonctionnalités, il est alors intéressant de **partager la même base de code**.

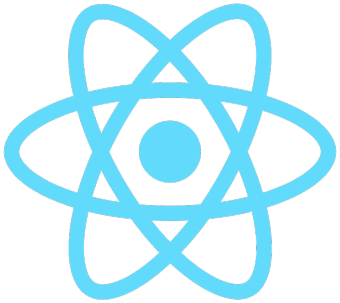
- Maintenir une seule base de code pour des applications destinées à différentes plateformes (e.g., iOS, Android, Windows).
  - **Plus facile à maintenir**, un seul langage pour l'équipe (les équipes).
  - Réduction des **coûts**.
  - Bonne **cohérence graphique et logique** entre les différentes versions.
    - Tout en gardant interfaces utilisateurs avec un **rendu et des interactions proche du natif.**

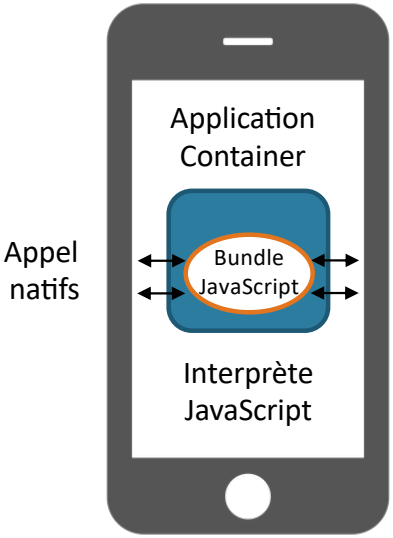
Pourquoi avec des technologies du web ?

- De plus en plus **populaires**.
- Il est parfois compliqué de **construire des interfaces qui s'adaptent aux plateformes et écrans**. Les technos Web offrent:
  - des solutions pour faire du responsive design.
  - des abstractions (indépendantes de la plateforme) des éléments qui composent un UI.
- Souvent plus **rapide** à tester. Important pour des entreprises comme Facebook qui mettent à jours leurs applications plusieurs fois par jour.

Multi-plateformes

- Globalement il y a trois approches :
  1. Compilation pour la plateforme cible.
  2. Web dans une « webview ».
  3. Web dans un container capable d'interagir avec la plateforme sous-jacente





React Native

- Publié par Facebook en 2015.
- Open source.

- Comme son nom l'indique, se base sur React ! Nous avons donc les bases !
  - Même concept de composants, avec propriétés et état !

- **Utilise des composants mobile natifs.**
  - Lorsque vous définissez une vue avec React Native, c'est bien un élément `android.view.view` ou `UIView` qui est créé.

- Permet de tester son application très rapidement.

Initialiser le projet …

- Créer un projet rapidement : `expo-cli`

```
drwxr-xr-x   12 nferry  staff     384 30 jan 17:46 .
drwxr-xr-x    5 nferry  staff     160 30 jan 17:45 ..
drwxr-xr-x    3 nferry  staff      96 30 jan 17:45 .expo-shared
drwxr-xr-x   10 nferry  staff     320 30 jan 17:46 .git
-rw-r--r--    1 nferry  staff     118 26 oct  1985 .gitignore
-rw-r--r--    1 nferry  staff     481 26 oct  1985 App.js
-rw-r--r--    1 nferry  staff     638 30 jan 17:45 app.json
drwxr-xr-x    6 nferry  staff     192 30 jan 17:45 assets
-rw-r--r--    1 nferry  staff     107 26 oct  1985 babel.config.js
drwxr-xr-x  571 nferry  staff   18272 30 jan 17:46 node_modules
-rw-r--r--    1 nferry  staff  362406 30 jan 17:46 package-lock.json
-rw-r--r--    1 nferry  staff     549 30 jan 17:45 package.json
```

Rien de nouveau
Par rapport à React !

UNIVERSITÉ CÔTE D'AZUR

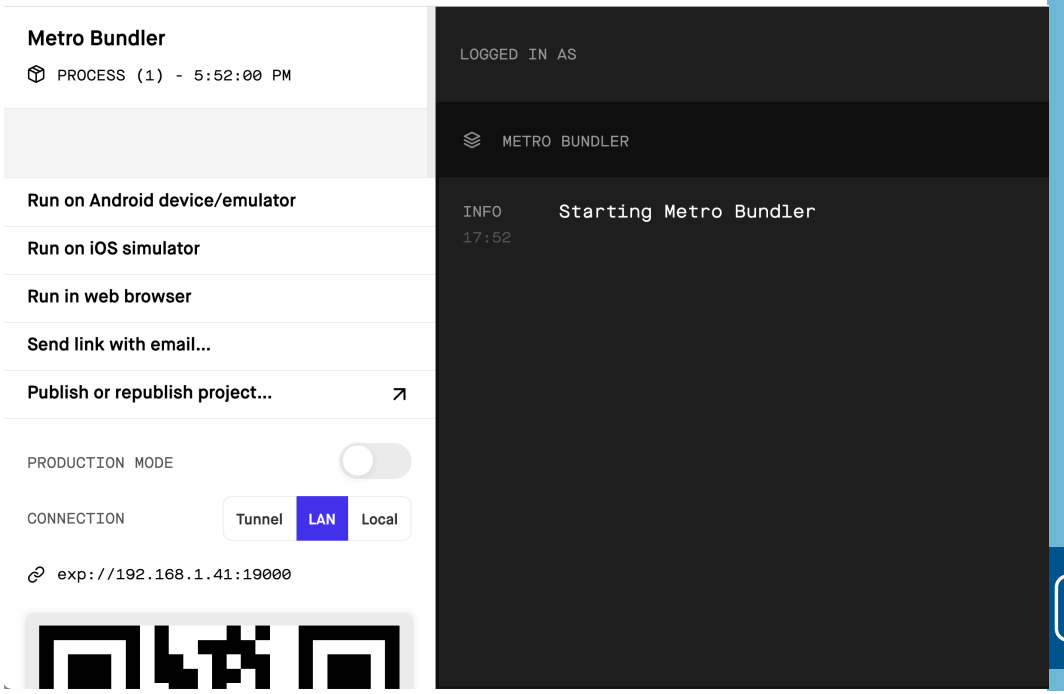
iut Nice Côte d'Azur

… et tester le projet

- **Expo** : une application « hôte » pour tester rapidement notre code et afficher le rendu de nos applications.
  - Télécharger l'appli dans l'app store.
  - Live reloading => pas possible avec du natif il faut re-compiler !

- Lancer le projet : `expo init app`
  - Plusieurs templates possible

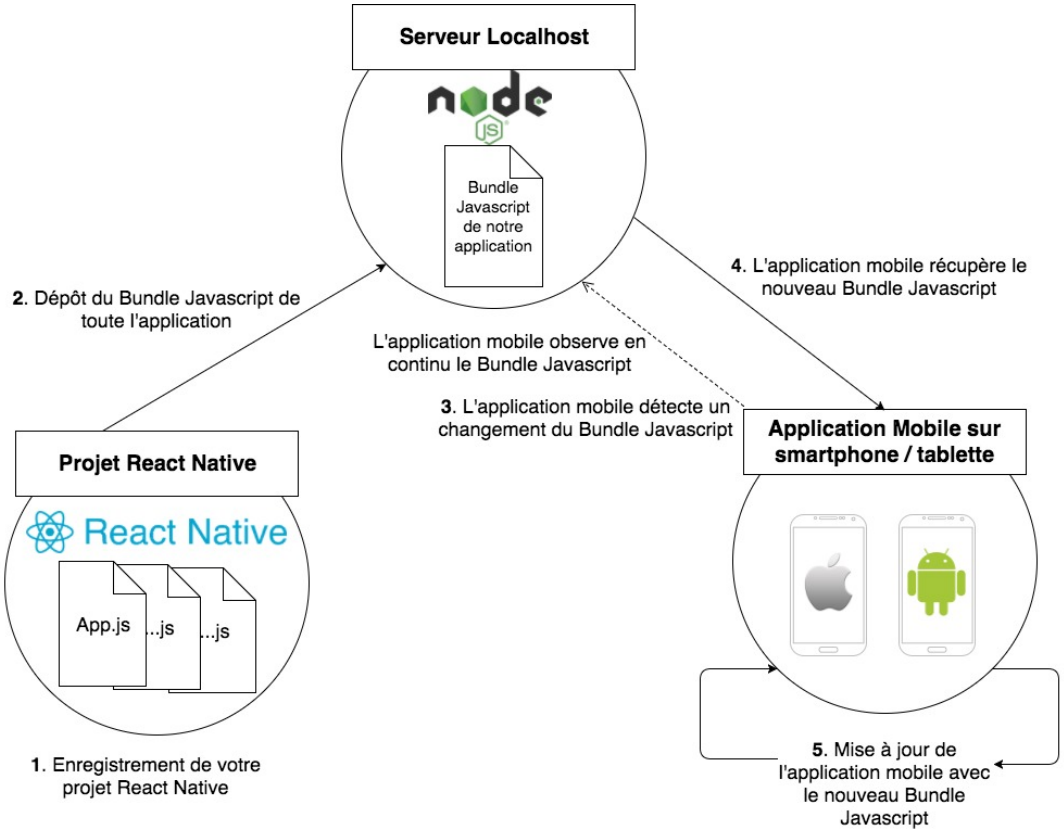
- `npm start` ouvre la page suivante



UNIVERSITÉ CÔTE D'AZUR

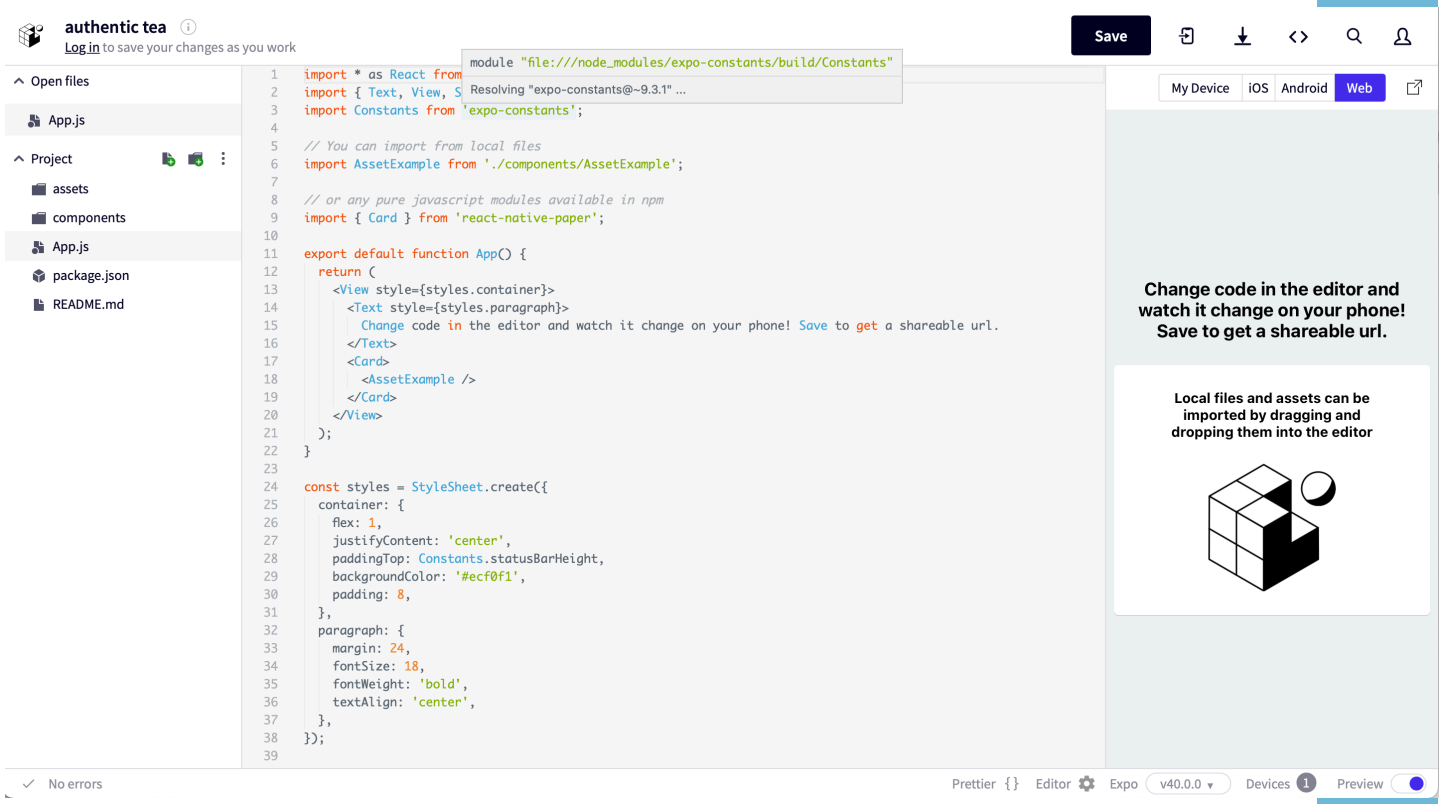
iut Nice Côte d'Azur

Live reloading



Snack

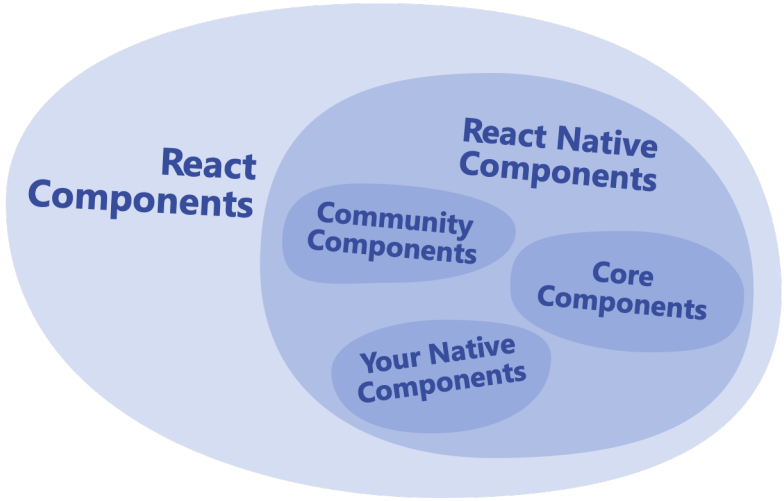
- snack.expo.io.
- Un éditeur en ligne.



UNIVERSITÉ CÔTE D'AZUR

iut Nice Côte d'Azur

Composants



- Trois types de composants :

1. **Composants React**.
2. **Composants React Native** : les composants qui font appel a la plateforme sous-jacente.
3. **Composants customs** : typiquement des composants qui regroupent et customise des composants react-native. La communauté met a disposition de nombreux composants customs.

Basic core Components

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

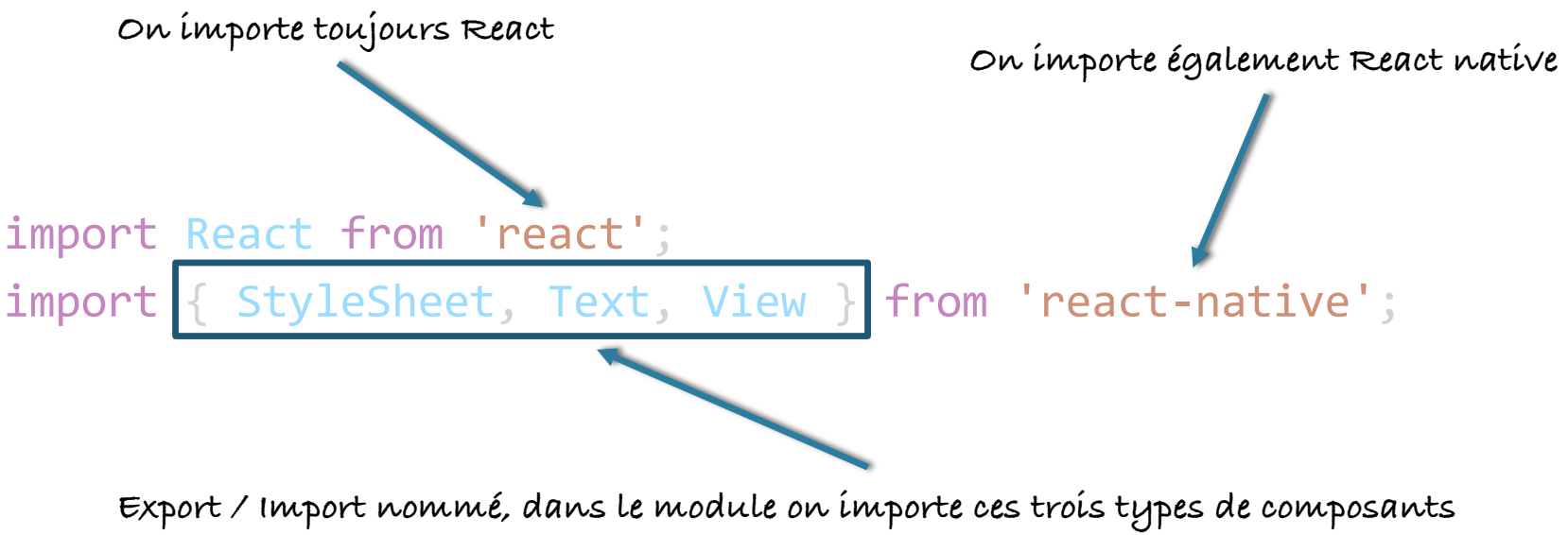
https://reactnative.dev/docs/intro-react-native-components

Core Component

- Liste complète ici : https://reactnative.dev/docs/components-and-apis
  - Basic Components
  - User Interface (`Button`, `Switch`)
  - List Views (`FlatList`, `SectionList`)
  - Android-specific (`PermissionAndroid`, `ToastAndroid`, `DrawerLayoutAndroid`, ...)
  - iOS-specific (`ActionSheetIoS`)
  - Others (`Alert`, `Modal`, `StatusBar`, `PixelRatio`, `Animated`, ...)

Un affichage différent sur iphone et Android ! Normal les composants sont natifs

Importer des composants



Listes

```
<View style={styles.container}>
<FlatList
  data={[
      {key: 'Devin'},
      {key: 'Dan'},
      {key: 'Dominic'},
      {key: 'Jackson'},
      {key: 'James'},
      {key: 'Joel'},
      {key: 'John'},
  ]}
  renderItem={({item}) => <Text style={styles.item}>{item.key}</Text>}
  keyExtractor={(item, index) => index.toString()}
/>
</View>
```

The **FlatList** component requires two props:

1. **data** is the source of information for the list.
2. **renderItem** takes one item from the source and returns a formatted component to render.

Style des composants

- Tous les composants react-native possède un style.

```
<View style={{ marginTop: 20 }}>
```

- On évite d'embarquer les styles directement dans le JSX !

```
const styles = StyleSheet.create({
   container: {
      flex: 1,
      backgroundColor: '#fff',
      alignItems: 'center',
      justifyContent: 'center',
   },
});

<View style={styles.container}>
<View style={styles.container, styles.container2}>
```

StyleSheet permet d'améliorer les performances

Il est possible d'appliquer plusieurs styles

Positionnement

- Tous les composants React-Native sont des boites flexibles. Une `View` est un container avec un layout flexbox.

- Par défaut tous les composants s'adaptent à leur contenu (flex 0).
  - On peut le changer !

Les views sont les unes sur les autres

```
<View style={{ flex: 1, flexDirection: 'column', backgroundColor: 'yellow' }}>
   <View style={{ flex: 1, backgroundColor: 'red' }}></View>
   <View style={{ flex: 2, backgroundColor: 'green' }}></View>
   <View style={{ flex: 1, backgroundColor: 'blue' }}></View>
</View>
```

2 fois plus grande que les autres!

Navigation entre vues

- Une librairie spécifique pour cela.
- Avec sa propre documentation: https://reactnavigation.org/docs/getting-started/

- Trois types de navigation :
  - **StackNavigation** : approche classique, la vue est poussée
  - **TabNavigator** : barre d'onglets/
  - **DrawerNavigator** : Menu apparaissant sur le coté de nos vues.

```
import {createAppContainer} from 'react-navigation'
import {createStackNavigator} from 'react-navigation-stack'
```

StackNavigation

```
<NavigationContainer>
  <Stack.Navigator>
    <Stack.Screen
      name="Home"
      component={Portfolio}
      options={{ title: 'Portfolio' }}
      />
    <Stack.Screen name="add" component={AddBook} />
  </Stack.Navigator>
</NavigationContainer>
```

```
this.props.navigation.navigate('Home')
```

Les composants créé depuis stack.screen sont instanciés avec une propriété navigation

Code spécifique à une plateforme.

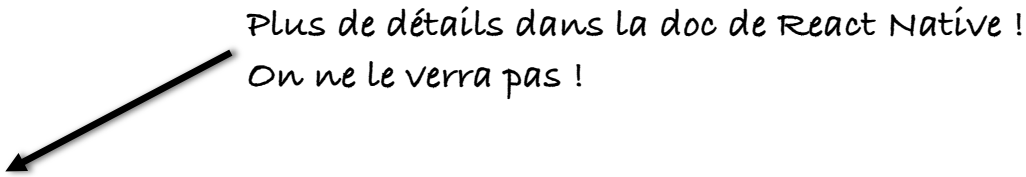
- Il y a un module pour détecter la plateforme (y compris la version) sur laquelle l'application s'exécute.

```
import { Platform} from 'react-native';
```

- Permet d'adapter le contenu, les styles en fonction de l'OS

```
const styles = StyleSheet.create({
   height: Platform.OS === 'ios' ? 200 : 100
});
```

Publier une applications



1. Créer une projet Android
   (cf. support React Native pour cela).

2. Créer le bundle javascript de votre application

3. Publier l'application.

Une grosse communauté

- Plein d'exemples d'applications : https://github.com/ReactNativeNews/React-Native-Apps
- Des composants : https://reactnativeelements.com
- Des composants en paper UI : https://material-ui.com