



# React Native Init

3J pour maîtriser le développement  
mobile



# Le Formateur

Guilian GANSTER

Développeur web & mobile

Freelance depuis 4 ans  
dans les technologies du  
web & mobile.

Développeur fullstack  
React / React native  
Firebase / Strapi



# Les 4 règles du développeur

**Dans la doc,  
tu chercheras**

Une plus grande  
autonomie, tu obtiendra

**L'indentation,  
tu respecteras**

Plus lisible, ton code  
sera

**Des questions,  
Tu poseras**

Plus de connaissance,  
Tu acquierera

**Des exercices,  
Tu pratiqueras**

Plus grande ta maîtrise  
deviendra

• • • • •

React Native ?

REACT NATIVE

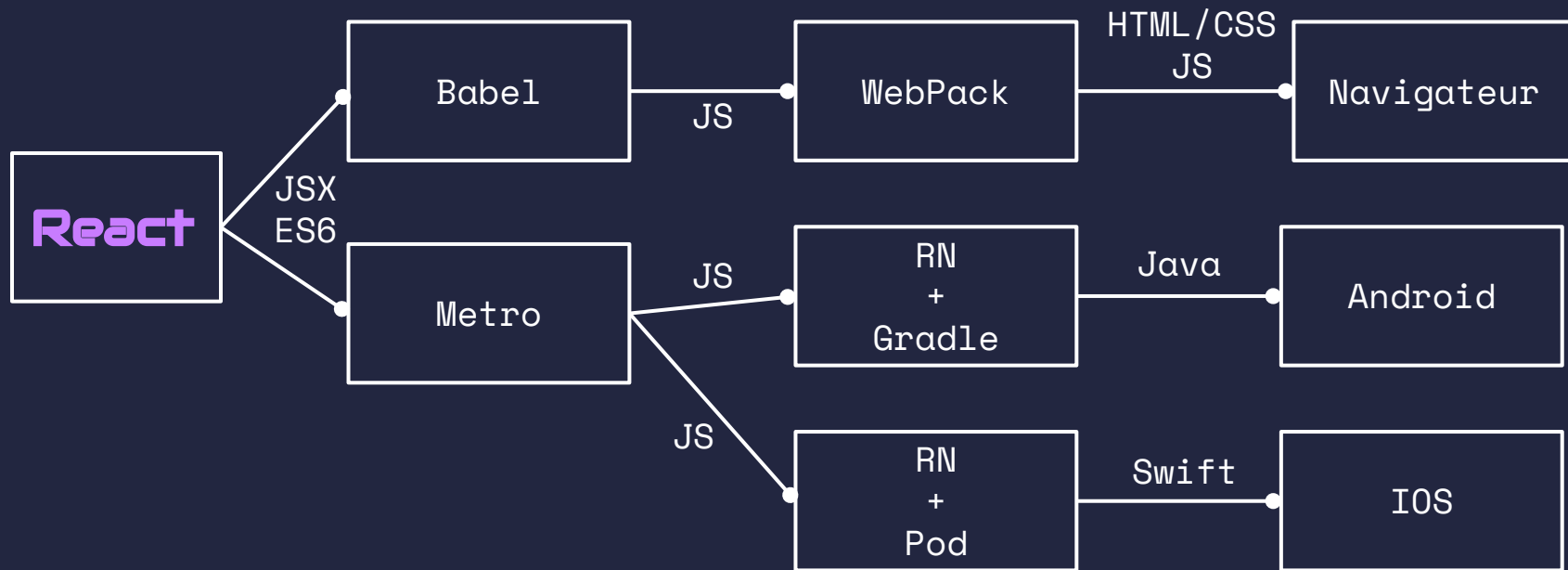




Created by		facebook	
Popularity			
Learning Curve			
Backward Compatibility			
Ecosystem			
Documentation			



# Les enjeux du cross plateforme



# Vos nouveaux outils de développement



**VSCode**

<https://code.visualstudio.com/>



**Nodejs**

<https://nodejs.org/en/>



**Android studio**

<https://developer.android.com/studio>

# Rappels ES6

Pour vous exercer:

<https://www.programiz.com/javascript/online-compiler/>

```
● ● ● structuration

//array structuration
let arrayStructuration = [1, 2]
console.log(arrayStructuration); //[1, 2]

//array copy
let arrayCopy = [...arrayStructuration, 3, 4]
console.log(arrayCopy); //[1, 2, 3, 4]

//array destructuration
let [a, b] = arrayCopy;
console.log(a); //1
console.log(b); //2
```

```
● ● ● structuration

//object structuration
let obj = {
  message: "hello world"
}
console.log(obj); //{message: "hello world"}

//object add key
obj.type = "success";
console.log(obj); //{message: "hello world", type: "success"}

//object copy
let copy = {...obj}

//object destructuration
const {message} = obj;
console.log(message); //"hello world"
```





# Rappels ES6

```
var vs let

function varTest() {
  var x = 1;
  if (true) {
    var x = 2; // c'est la même variable !
    console.log(x); // 2
  }
  console.log(x); // 2
}

function letTest() {
  let x = 1;
  if (true) {
    let x = 2; // c'est une variable différente
    console.log(x); // 2
  }
  console.log(x); // 1
}

varTest();
letTest();
```

Pour vous exercer:  
<https://www.programiz.com/javascript/online-compiler/>



# Rappels ES6

Cheat sheet: <https://htmlcheatsheet.com/js/>

## fonctions fléchées

```
//from this
function foo() {
  console.log("bar");
}

//to this
const foo = () => {
  console.log("bar");
}

//or this
const foo = () => console.log("bar");
```

## array functions

```
//from this
const array = [1, 2, 3];

for (let i = 0; i < array.length; i++) {
  console.log(array[i]);
}

//to this
const array = [1, 2, 3];

array.map(i => console.log(i));
```

# Cheat sheet Fonction Fléchées



Uploaded using RayThis Extension

```
const a = (param) => param; // paramètre unique, return implicite

const b = param => param; // paramètre unique (parenthèse non requise), return implicite

const c = (param1, param2) => param1 + param2; // paramètres multiples (parenthèses requises), return implicite
```



Uploaded using RayThis Extension

```
const a = () => {
  return "hello"
} // multi-lignes, return explicite

const b = () => "hello" // une seule ligne, return implicite

const c = () => (
  "hello"
)// multi ligne, return implicite
```



# Rappels ES6

Cheat sheet: <https://htmlcheatsheet.com/js/>

## fonctions fléchées

```
//from this
function foo() {
  console.log("bar");
}

//to this
const foo = () => {
  console.log("bar");
}

//or this
const foo = () => console.log("bar");
```

## array functions

```
//from this
const array = [1, 2, 3];

for (let i = 0; i < array.length; i++) {
  console.log(array[i]);
}

//to this
const array = [1, 2, 3];

array.map(i => console.log(i));
```

# Rappels ES6: asynchrone

```
Uploaded using RayThis Extension

//retourne une promesse qui se résoudra après ms millisecondes
const sleep = (ms) => new Promise(resolve => setTimeout(resolve, ms));

const synchroneFunction = () => {
  console.log("start");
  sleep(3000);
  console.log("end");// executé immédiatement :-(
}

synchroneFunction();

const asynchroneFunction = async () => {
  console.log("start");
  await sleep(3000);
  console.log("end");// executé après 3 secondes :-D
}

asynchroneFunction();
```

```
Uploaded using RayThis Extension

//retourne une promesse qui se résoudra après ms millisecondes
const sleep = (ms) => new Promise(resolve => setTimeout(resolve, ms));

const synchroneFunction = () => {
  console.log("start");
  sleep(3000).then(() => {
    console.log("end");// executé après 3 secondes
  })
}

synchroneFunction();
```

# Expo vs React Native CLI

## Expo

- service tiers
- Une librairie de base fournie
- De nombreux outils pratiques: Simplifie le développement
- Mais limité à l'écosystème Expo

npm eject

## React native CLI

- Développé par la team react native & la communauté
- Développement "bare-bone" (seulement une configuration de base)
- Pas d'aide au développement (uniquement des composants de base)
- Flexibilité totale : Intégration avec n'importe quel code natif



# Votre premier composant

```

● ● ● structuration

import React from "react"; // requis pour les composants
import { StyleSheet, Text, SafeAreaView } from "react-native"; // import de la lib standard

const MyComponent = () => { // un composant commence toujours par une majuscule
  return (
    <SafeAreaView style={styles.container}>
      <Text>Hello world</Text>
    </SafeAreaView>
  )
}

const styles = StyleSheet.create({ // les styles sont définis dans un objet, sous forme css in js

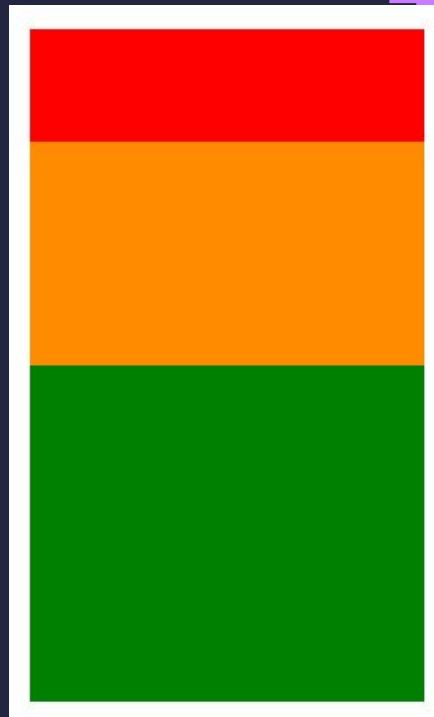
  container: {
    flex: 1, // prend toute la taille à l'écran
    backgroundColor: "grey", // en css in js, les - sont remplacés par des majuscules
  }
});

export default MyComponent;
```

# Mise en Forme du layout

```
const Flex = () => {
  return (
    <View style={[[styles.container, {
      // Try setting `flexDirection` to `"row"`.
      flexDirection: "column"
    }]]}>
      <View style={{ flex: 1, backgroundColor: "red" }} />
      <View style={{ flex: 2, backgroundColor: "darkorange" }} />
      <View style={{ flex: 3, backgroundColor: "green" }} />
    </View>
  );
};

//https://css-tricks.com/snippets/css/a-guide-to-flexbox/
```





# Les événements

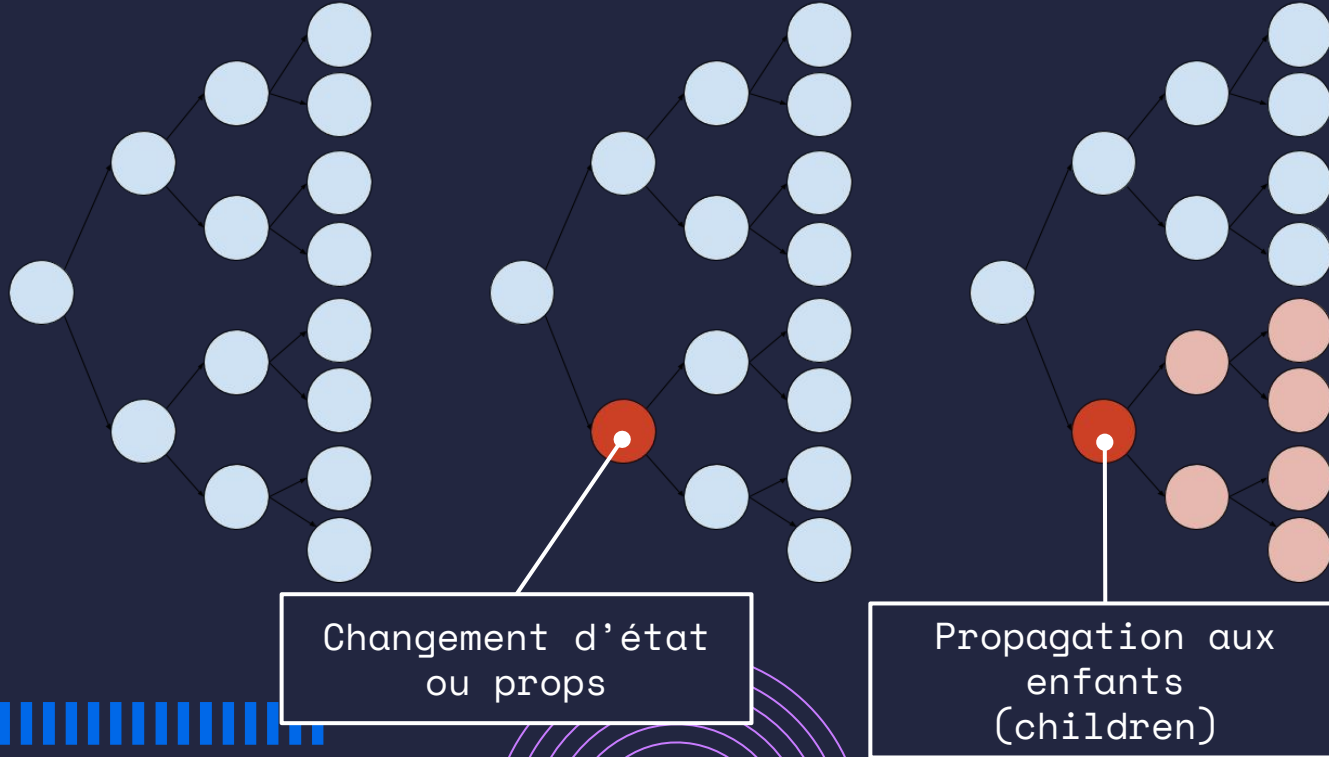
```
● ● ● structuration

const MyComponent = () => { //un composant commence toujours par une majuscule
  return (
    <SafeAreaView style={styles.container}>
      <Button title="PressMe" onPress={() => console.log("hello world")} />
    </SafeAreaView>
  );
};

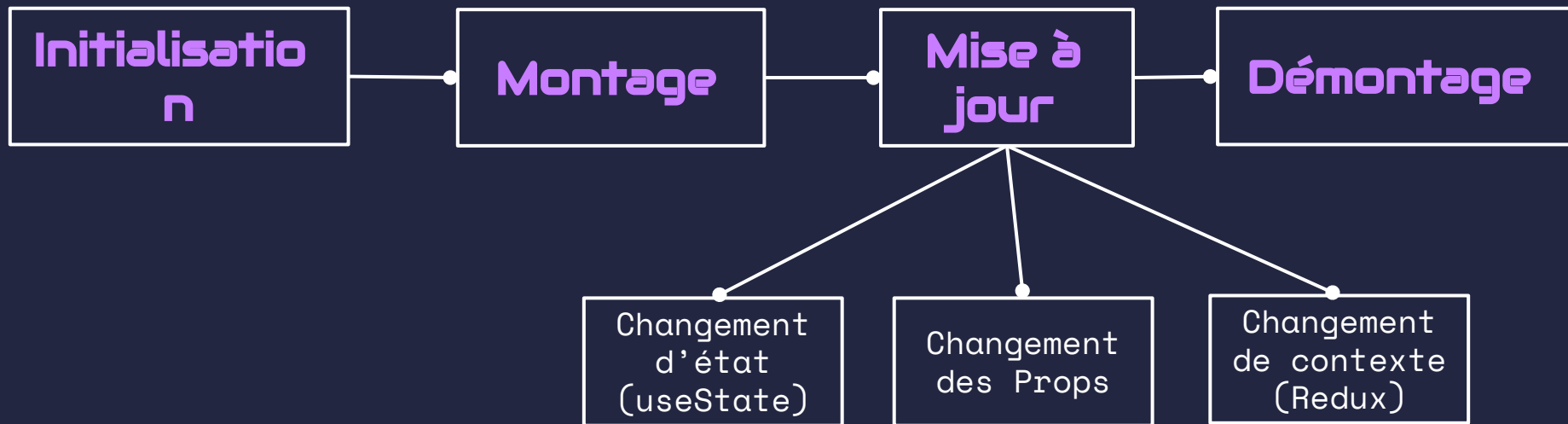
//plus propre avec un handler
const MyComponent = () => {
  //une fonction "privée" (restreinte au composant)
  //commence par un _
  const _handlePress = () => {
    console.log("hello world");
  }

  return (
    <SafeAreaView style={styles.container}>
      <Button title="PressMe" onPress={_handlePress} /> /* équivalent a () => _handlePress */
    </SafeAreaView>
  )
}
```

# Le cycle de vie



# Le cycle de vie



# Votre premier hook: useState

```
useState

import React, {useState} from "react";
import {SafeAreaView, Text, Button} from "react-native";

const App = () => {
  //useState renvoi un array de taille 2,
  //on utilise la destructuration pour récupérer les 2 valeurs: le state et la fonction qui
  //permet de modifier le state
  const [count, setCount] = useState(0); //0 = la valeur par défaut à l'initialisation

  return (
    <SafeAreaView>
      <Text>Count: {count}</Text>{/* On affiche le state avec un echappement du JSX*/}
      <Button title="Increment" onPress={() => setCount(count + 1)} />{/* On modifie le state avec la
fonction renvoyée par useState*/}
    </SafeAreaView>
  )
}

export default App;
```

# Le hook d'effet: useEffect

```
Uploaded using RayThis Extension

import React, {useState, useEffect} from "react";
import {SafeAreaView, Text, Button} from "react-native";

const App = () => {
  const [count, setCount] = useState(0);

  useEffect(() => {
    console.log("exécuté a chaque render");
  });

  useEffect(() => {
    console.log("exécuté a chaque changement de count");
  }, [count]);

  useEffect(() => {
    console.log("exécuté une seule fois, au montage du composant");
  }, []);

  return (
    <SafeAreaView>
      <Text>Count: {count}</Text>
      <Button title="Increment" onPress={() => setCount(count + 1)} />
    </SafeAreaView>
  )
}

export default App;
```

```
Uploaded using RayThis Extension

import React, {useState, useEffect} from "react";
import {SafeAreaView, Text, Button} from "react-native";

const App = () => {
  const [count, setCount] = useState(0);

  useEffect(() => {
    console.log("exécuté une seule fois, au montage du composant");
    return () => { //permet de nettoyer la précédente execution du useEffect
      console.log("exécuté au démontage du composant");
    }
  }, []);

  return (
    <SafeAreaView>
      <Text>Count: {count}</Text>
      <Button title="Increment" onPress={() => setCount(count + 1)} />
    </SafeAreaView>
  )
}

export default App;
```

# Les hooks custom

Uploaded using RayThis Extension

```
const useCount = (defaultValue) => {
  const [count, setCount] = useState(defaultValue);

  const increment = () => setCount(count + 1);
  const decrement = () => setCount(count - 1);

  useEffect(() => {
    console.log("Count changed to: ", count);
  }, [count]);

  return [count, increment, decrement];
}
```

Uploaded using RayThis Extension

```
const App = () => {
  const [count, increment, decrement] = useCount(0);

  return (
    <SafeAreaView>
      <Text>Count: {count}</Text>
      <Button title="Decrement" onPress={decrement} />
      <Button title="Increment" onPress={increment} />
    </SafeAreaView>
  )
}
```

# Design pattern

.....



# Conditional rendering

Utilisation:

- Écran de chargement
- Gestion des autorisations
- Gestion d'erreur
- ...

```
const Article = ({item}) => {  
  if (!item) return <ActivityIndicator />  
  return (  
    <View>  
      { /* ... */ }  
    </View>  
  )  
}
```



# Conditional rendering

Utilisation:

- Écran de chargement
- Gestion des autorisations
- Gestion d'erreur
- ...

Uploaded using RayThis Extension

```
const Article = ({item}) => {  
  return (  
    <View>  
      {item ?  
        <View>  
          { /* ... */ }  
        </View>  
      :  
        <ActivityIndicator />  
    }  
    </View>  
  )  
}
```



# Conditional rendering

Utilisation:

- Écran de chargement
- Gestion des autorisations
- Gestion d'erreur
- ...

```
Uploaded using RayThis Extension

const Article = ({item}) => {
  return (
    <View>
      {item &&
        <View>
          { /* ... */ }
        </View>
      }
    </View>
  )
};
```

# Array Rendering

Utilisation:

- Affichage d'une liste

```
Uploaded using RayThis Extension

const Article = ({item}) => {
  console.log(item.comments);
  /*
  [
    {id: 0, content: "commentaire 1"},
    {id: 1, content: "commentaire 2"},
    {id: 2, content: "commentaire 3"},
  ]
  */

  return (
    <View>
      {item.comments.map((comment) => {
        return (
          <View key={comment.id}>
            <Text>{comment.content}</Text>
          </View>
        )
      })}
    </View>
  )
}
```

# Array Rendering

Utilisation:

- Affichage d'une liste

```
Uploaded using RayThis Extension

const Article = ({item}) => {
  console.log(item.comments);
  /*
  [
    {id: 0, content: "commentaire 1"},
    {id: 1, content: "commentaire 2"},
    {id: 2, content: "commentaire 3"},
  ]
  */

  return (
    <View>
      {item.comments.map((comment) => {
        return (
          <View key={comment.id}>
            <Text>{comment.content}</Text>
          </View>
        )
      })}
    </View>
  )
}
```

# Children prop

Se comporte comme une props

Permet une meilleure lisibilité  
dans certains cas.

```
Uploaded using RayThis Extension

const MyFullScreenView = ({children}) => {
  return (
    <View style={{flex: 1}}>
      {children}
    </View>
  )
}

const App = () => {
  return (
    <MyFullScreenView>
      <Text>Some text</Text>
    </MyFullScreenView>
  )
}
```

# Props de rendu

Partager du code entre des composants

Surcharger une vue

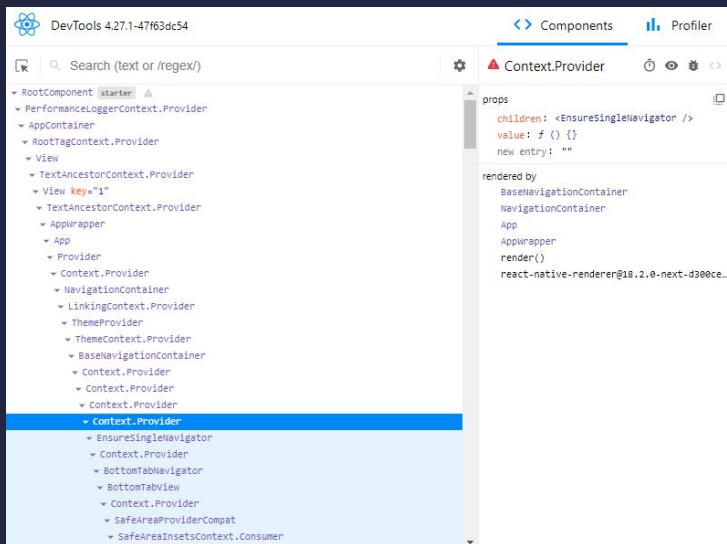


Uploaded using RayThis Extension

```
<DataProvider render={data => (  
  <h1>Bonjour {data.target}</h1>  
)}>/>
```



# Debugging: arborescence et état



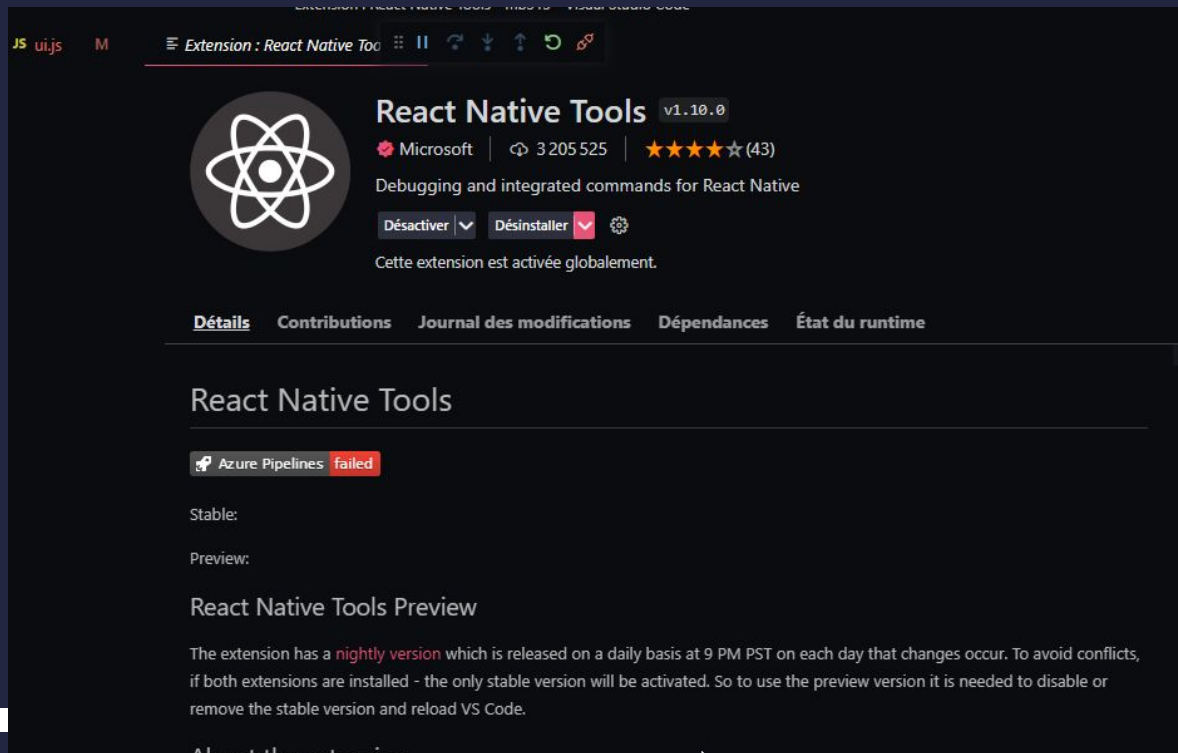
## Installation:

```
$ npm install -g react-devtools
```

```
$ adb reverse tcp:8097 tcp:8097
```




# Debugging: breakpoint



The screenshot shows the Visual Studio Code interface with the 'React Native Tools' extension page. The extension is by Microsoft, has 3,205,525 downloads, and a 4.5-star rating from 43 reviews. It is currently installed and active. The page includes tabs for 'Détails', 'Contributions', 'Journal des modifications', 'Dépendances', and 'État du runtime'. The 'Détails' tab is selected, showing the extension's name, version (v1.10.0), and a description: 'Debugging and integrated commands for React Native'. Below this, there is a section for 'React Native Tools' with a status bar indicating 'Azure Pipelines failed'. The 'Stable' version is listed, and a 'Preview' section is also visible, mentioning a 'nightly version' released daily at 9 PM PST.

JS uijs M Extension : React Native Tools

 **React Native Tools** v1.10.0

Microsoft | 3 205 525 | ★★★★★ (43)


Debugging and integrated commands for React Native

Désactiver | Désinstaller

Cette extension est activée globalement.

[Détails](#) [Contributions](#) [Journal des modifications](#) [Dépendances](#) [État du runtime](#)

## React Native Tools

 Azure Pipelines **failed**

Stable:

Preview:

### React Native Tools Preview

The extension has a **nightly version** which is released on a daily basis at 9 PM PST on each day that changes occur. To avoid conflicts, if both extensions are installed - the only stable version will be activated. So to use the preview version it is needed to disable or remove the stable version and reload VS Code.

About the extension



# Les contextes

Provider

Abonné grâce à  
useContext

Changement d'état du  
contexte

Rendu chez tous les  
composants abonné

# Les contextes

Uploaded using RayThis Extension

```
import {createContext, useState, useContext} from "react";

const Context = createContext();

const Default = {
  darkmode: false
}

const Provider = ({children}) => {
  const [ctx, setCtx] = useState({...Default});

  return (
    <Context.Provider value={[ctx, setCtx]}>
      {children}
    </Context.Provider>
  )
}

const useUI = () => useContext(Context);

export default useUI;
export {Provider};
```