2022-2023

ISEP
Introduction à
ReactJS

Présenté par Boubacar Demba Mandiang

React, c'est quoi ?

React.js, communément appelé simplement React, est une bibliothèque JavaScript utilisée développée par les ingénieurs de Facebook pour pouvoir construire des interfaces utilisateur. Chaque application web React est composée de composants réutilisables qui constituent des parties de l'interface utilisateur – nous pouvons avoir un composant distinct pour notre barre de navigation, un pour le pied de page, un autre pour le contenu principal, et ainsi de suite. Vous comprendrez mieux cela lorsque nous arriverons à la section où nous devons travailler avec des composants.

1. Avantages et inconvénients de React

Pourquoi l'utiliser ?

Pourquoi utiliser React ?

- **React est rapide.** Les applications créées dans React peuvent gérer des mises à jour complexes tout en restant rapides et réactives.
- **React est modulaire.** Au lieu d'écrire des fichiers de code volumineux et denses, vous pouvez écrire de nombreux fichiers plus petits et réutilisables. La modularité de React peut être une belle solution aux problèmes de **maintenabilité** de JavaScript.
- **React est évolutif.** Les grands programmes qui affichent beaucoup de données changeantes sont ceux où React fonctionne le mieux.
- **React est flexible.** Vous pouvez utiliser React pour des projets intéressants qui n'ont rien à voir avec la création d'une application Web. Les gens sont encore en train de comprendre le potentiel de React. Il y a de la place à explorer.
- **React est populaire.** Bien que cette raison ait certes peu à voir avec la qualité de React, la vérité est que comprendre React vous rendra plus employable.

Inconvénients de React

- Les débutants qui n'ont pas une solide compréhension de JavaScript (en particulier ES6) peuvent avoir du mal à comprendre React.

- React est dépourvu de certaines fonctionnalités courantes comme la gestion d'un état unique et le routage ; vous devrez installer et apprendre à utiliser des bibliothèques externes pour les obtenir.

2. Mise en plus de l'environnement

Les outils à installer

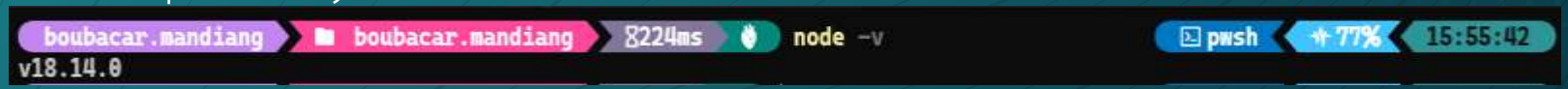
Mise en place de l'environnement

- Il vous faut installer **Node.js** est une plateforme permettant de développer des applications Web qui exploitent les capacités de JavaScript à la fois sur le front-end et le back-end.

- Il vous faut aussi un IDE comme VSCODE ou tout autre ide de votre choix

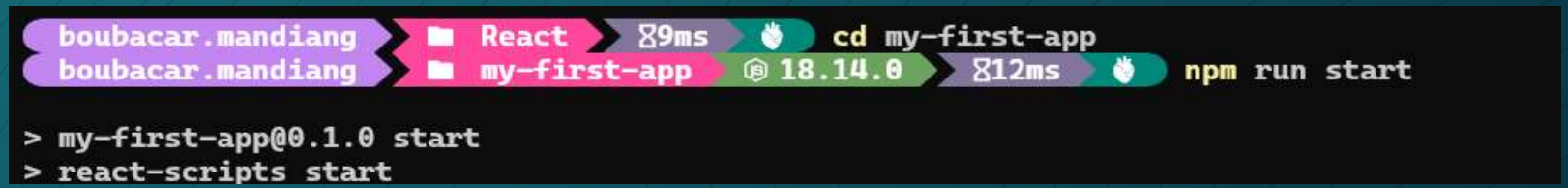
Mise en place de l'environnement

Vérifier que Node.js est bien installé



Création de notre première application React nommé **my-first-app**

Lancement de notre première application



CREATE-REACT-APP est à éviter, Pourquoi ?

- Dès que vous tapez la commande pour initialiser un nouveau projet, vous remarquerez immédiatement la grande quantité de paquets et de configuration que CRA doit subir pour configurer un simple projet React.
- L'initialisation peut aller de 2 à 15 minutes en fonction de votre connexion internet
- Et ce parce que CRA utilise **Webpack** derrière, depuis longtemps il a été le favori indiscutable. Webpack vient avec une fonctionnalité simple mais très pratique : il comprend tous les types de modules JavaScript qui existent (les modules ECMAScript modernes, mais aussi les modules AMD – Asynchronous module definition et CommonJS, des formats qui existaient avant le standard).

  Même si vous utilisez des bibliothèques avec des formats très différents, Webpack va les convertir et packager tout ton code et le code de ces bibliothèques ensemble dans un seul gros fichier JS : le fameux bundle. Le problème est que plus le projet gagne en complexité et en nombre de fichiers, plus le bundle devient long et pénible lors du développement.

Vite

C’est pour ces raisons qu’à la place de create-react-app, on va utiliser **Vite**

L’idée derrière Vite est que, comme les navigateurs modernes supportent les modules ES, on peut maintenant les utiliser directement, au moins pendant le développement, plutôt que de générer un bundle.

Donc lorsque vous chargez une page dans un navigateur quand vous développez avec Vite, vous ne chargez pas un seul gros fichier JS contenant toute l’application mais juste les quelques ESM nécessaires pour cette page.

On peut créer une application React avec vite en utilisant cette commande:

.

3. Les composants sur React

C’est quoi un composant ?

Composant sur React

Un **composant** (ou "*component*" en anglais) est une entité réutilisable dans la bibliothèque JavaScript React. Les composants sont des blocs de code qui encapsulent une logique ou une fonctionnalité spécifique, et peuvent être combinés pour créer des interfaces utilisateur complexes.

Les composants React peuvent être de deux types : les **composants fonctionnels** (ou "*functional components*") et les **composants de classe** (ou "*class components*") et retourne du **JSX**.

En règle générale, les composants React sont conçus pour être réutilisables et modulaires, ce qui permet aux développeurs de créer des interfaces utilisateur plus rapidement et plus facilement. Les composants peuvent être importés et utilisés dans d'autres parties d'un projet React, ce qui permet de gagner du temps et d'éviter de dupliquer du code.

Cycle de vie d'un composant

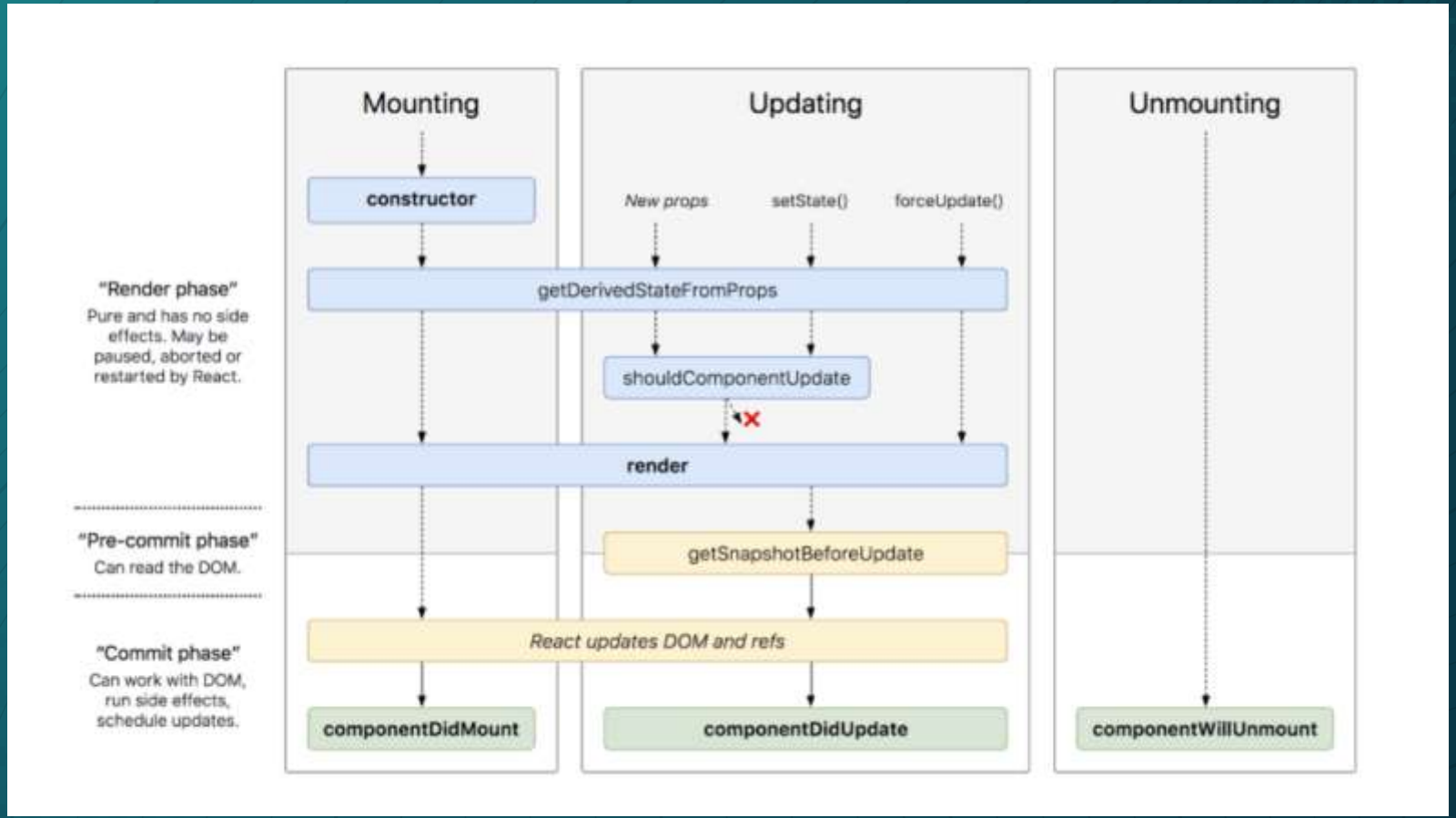
Chaque composant React passe par différentes étapes durant lesquelles il est possible d'intervenir. On appelle ça le cycle de vie d'un composant. Celui-ci se découpe en 3 parties :

- **Mount :** le montage. Il intervient quand une instance du composant est créé dans le DOM.
- **Update :** la mise à jour. Ce cycle de vie est déclenché par un changement d'état du composant.
- **Unmount :** le démontage. Cette méthode est appelée une fois qu'un composant est retiré du DOM.

Tout s'articule autour de ces 3 cycles de vies.

Illustration cycle de vie d'un composant

Ces fonctions sont valables que pour les composants de classes

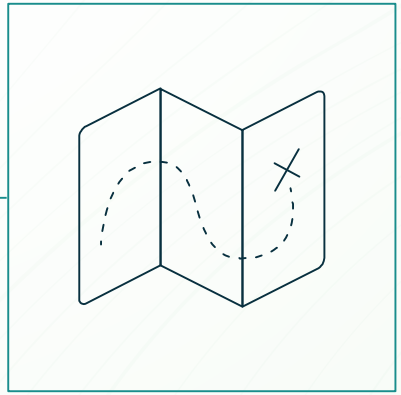


Concepts à connaître pour bien utiliser React

Ces concepts sont très importants et seront développés plus en détails dans le cours avec des exemples pratiques

On peut citer entre autres:

- Pour javascript => les variables, les fonctions, les tableaux, les objets, les promesses, les boucles, les évènements…
- Les composants et leur cycle de vie
- JSX
- La gestion des états (states)
- Les propriétés (props)
- Les hameçons (hooks)
- Le rendu et le rendu conditionnel



Merci!

Avez-vous des questions ?