

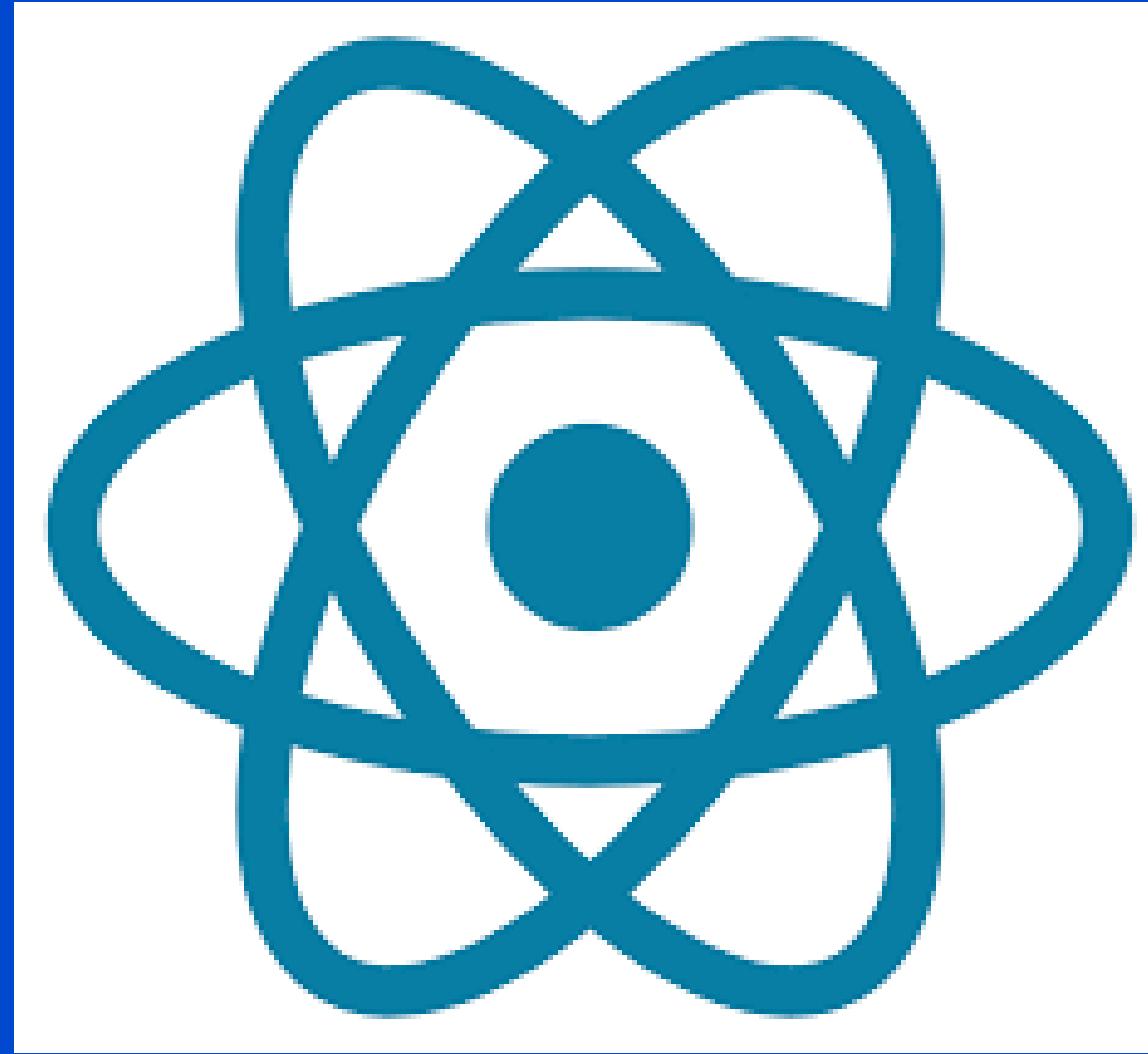
react

**Habib Diallo
Serge Senghor
Amadou Dieng**

Plan

Qu'est-ce que React.js ?

- Qu'est-ce qu'un composant en React ?
- Quelle est la différence entre composant fonctionnel et composant de classe ?
- Qu'est-ce que le Virtual DOM et pourquoi est-il utilisé ?
- Comment fonctionne le cycle de vie d'un composant ?
- Qu'est-ce que le JSX et comment est-il interprété ?
 - Comment passer des props à un composant ?
 - Quelle est la différence entre props et state ?
 - Comment gérer le state dans un composant fonctionnel avec useState ?
- Quelle est la différence entre useState et useEffect ?
- Comment utiliser useContext pour partager un état global ?
- Qu'est-ce que useReducer et dans quel cas l'utiliser ?
- Quelle est la différence entre les hooks natifs et les hooks personnalisés ?



React est une bibliothèque JavaScript créée par Facebook pour construire facilement des interfaces web. Elle utilise des composants réutilisables et un DOM virtuel pour rendre les applications rapides, interactives et faciles à maintenir

Qu'est-ce qu'un composant en React ?

En React, un composant est un élément d'interface utilisateur (UI) indépendant et réutilisable, qui peut être aussi petit qu'un bouton ou aussi grand qu'une page entière

```
function Counter() {  
  const [count, setCount] = useState(0);  
  return <button onClick={() => setCount(count + 1)}>{count}</button>;  
}
```

Quelle est la différence entre composant fonctionnel et composant de classe ?

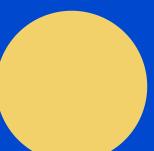
```
function Counter() {  
  const [count, setCount] = useState(0);  
  
  return (  
    <>  
    <p>Compteur : {count}</p>  
    <button onClick={() => setCount(count + 1)}>+1</button>  
    </>  
  );  
}
```

Composant fonctionnel

- Syntaxe : basé sur une fonction JavaScript.
- State et logique : utilise les hooks (ex. useState, useEffect) pour gérer l'état et les effets secondaires.
- Plus simple et moderne : recommandé depuis React 16.8.

```
class Counter extends React.Component {  
  state = { count: 0 };  
  render() {  
    return <button onClick={() => this.setState({ count: this.state.count + 1 })}>{this.state.count}</button>;  
  }  
}
```

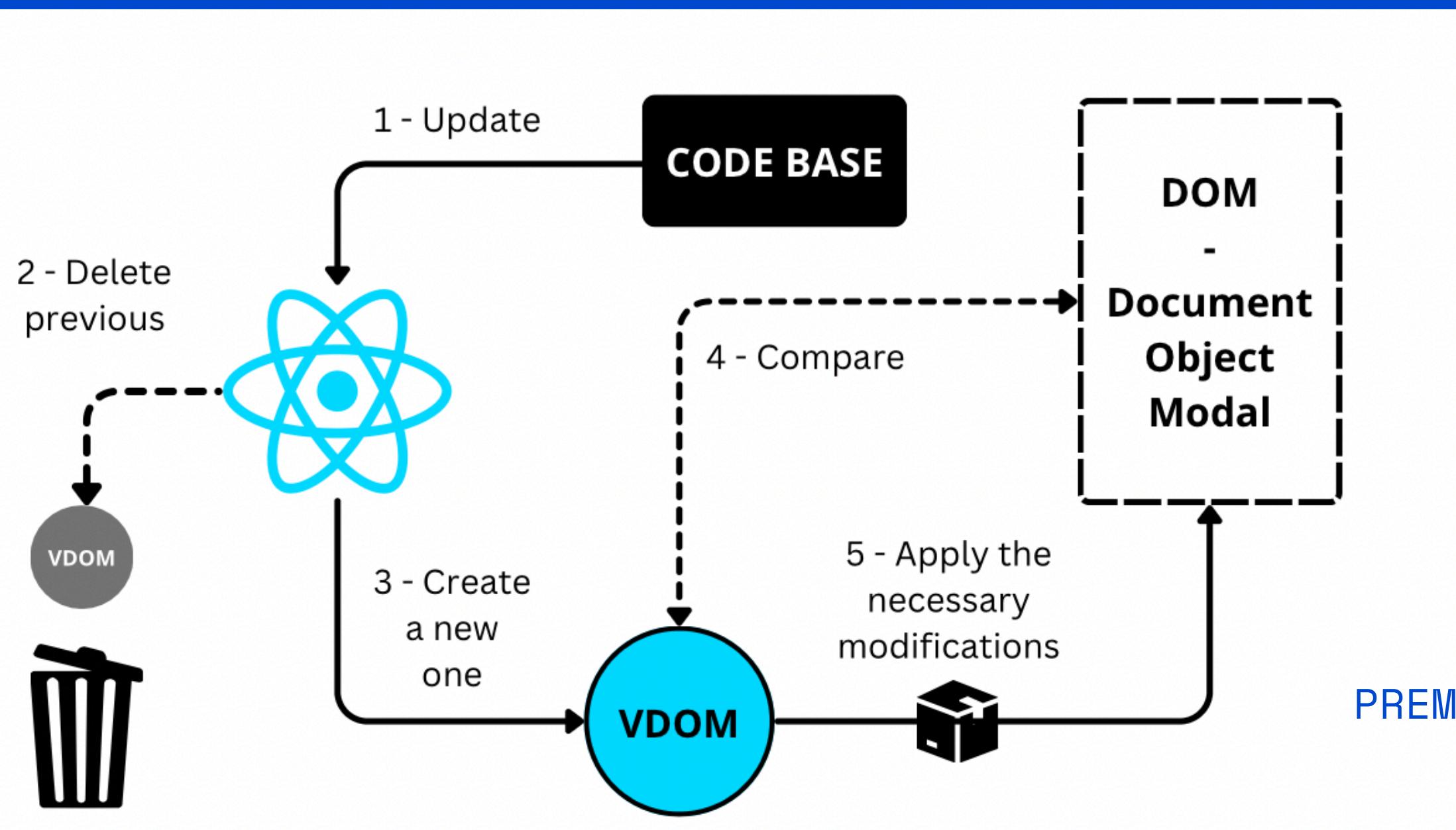
- Syntaxe : basé sur une classe JavaScript qui étend React.Component.
- State et logique : le state est défini dans le constructeur et les fonctions doivent souvent être liées (this).



Qu'est-ce que le Virtual DOM et pourquoi est-il utilisé ?



En React, le DOM virtuel (VDOM) est une représentation légère et en mémoire du DOM réel, utilisée pour optimiser les mises à jour de l'interface utilisateur



- 1) Quand l'application est mise à jour (à chaque appel de `setState()` ou au changement de props), React crée un nouvel arbre du VDOM avec toutes les modifications.
- 2) Il détecte les éléments ayant changé et ceux qui doivent être mis à jour en comparant le nouvel arbre VDOM avec le précédent en utilisant un algorithme de différenciation appelé "Reconciliation".
- 3) À partir de cela, il génère un plan d'action qui détaille les mises à jour à appliquer sur le DOM.
- 4) Enfin, React applique les mises à jour nécessaires au DOM réel en suivant le plan d'action généré avec React DOM.

Comment fonctionne le cycle de vie d'un composant ?

Chaque composant React suit le même cycle de vie :
Un composant est monté lorsqu'il est ajouté à l'écran.

Un composant se met à jour quand il reçoit de nouvelles props ou variables
d'état, généralement à la suite d'une interaction.

Un composant est démonté quand il est retiré de l'écran.

Qu'est-ce que le JSX et comment est-il interprété ?

```
<div className="container">  
  <h1>Mon titre</h1>  
  <p>Mon paragraphe</p>  
</div>
```

```
React.createElement(  
  'div',  
  { className: 'container' },  
  React.createElement('h1', null, 'Mon titre'),  
  React.createElement('p', null, 'Mon paragraphe')  
);
```



Comment passer des props à un composant ?

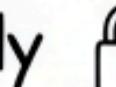
Pour passer des props à un composant dans des frameworks comme React, vous ajoutez des attributs au JSX du composant lors de son utilisation, puis vous les récupérez dans le composant enfant via le paramètre **props**

```
161
162 ✓ function Bonjour(props) {
163     return <h2>Bonjour {props.nom}</h2>;
164 }
165
166 ✓ function App() {
167     return <Bonjour nom="Fatou" />;
168 }
```



Quelle est la différence entre props et state ?

Quelle est la différence entre PROPS et STATE ?

<u>PROPS</u>	?	<u>STATE</u>
<ul style="list-style-type: none">• Read-only • Passées du parent à l'enfant• Immuables • Données externes 		<ul style="list-style-type: none">• Modifiables • Gérée à l'intérieur du composant• Mutables • Données internes 

Quand utiliser quoi ?

- Props pour config, State pour données qui changent.



Comment gérer le state dans un composant fonctionnel avec useState ?

```
function Counter() {  
  const [count, setCount] = useState(0);  
  
  return (  
    <>  
    <p>Compteur : {count}</p>  
    <button onClick={() => setCount(count + 1)}>+1</button>  
    </>  
  );  
}
```

Qu'est-ce qu'un hook en React ?

Un Hook en React est une fonction spéciale qui permet d'utiliser des fonctionnalités de React, comme l'état local et les cycles de vie, sans avoir à écrire de composants de classe



Comment utiliser useContext pour partager un état global ?

```
import { useState, useEffect, createContext, useContext, useReducer } from 'react';

// 1. Context
const AppContext = createContext();
```

```
function App() {
  const [utilisateur, setUtilisateur] = useState("Serge");
  const [age, setAge] = useState(23);
  const [score, setScore] = useState(0);

  return (
    <AppContext.Provider value={{  
      utilisateur,  
      setUtilisateur,  
      age,  
      setAge,  
      score,  
      setScore  
    }}>
```

Windsurf: Refactor | Explain | Generate JSDoc | X

```
function MonPremierComposant() {
  const { utilisateur, age, score } = useContext(AppContext);

  return (
    <div>
      <h1>Je suis ton premier composant React</h1>
      <p>Je suis {utilisateur} et j'ai {age} ans</p>
      <p>Score: {score} points</p>
    </div>
  );
}
```



Quelle est la différence entre les hooks natifs et les hooks personnalisés ?

La principale différence est que les hooks natifs (comme useState, useEffect) sont des fonctionnalités intégrées de React pour ajouter des fonctionnalités (état, effets, etc.) aux composants, tandis que les hooks personnalisés sont des fonctions JavaScript créées par le développeur, commençant par "use", qui regroupent et réutilisent une logique complexe d'état ou d'effets entre différents composants, mais ne sont pas des fonctionnalités directes de React.



ATTENTION

YOUR

FOR

YOU



THANK