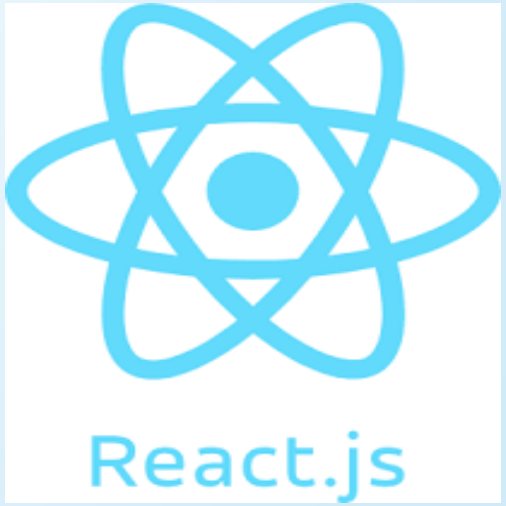
Introduction à React.js

Construire des interfaces utilisateur modernes

Réalisée par:
Ben Béchir Chaima



Qu'est-ce que React.js ?

Une bibliothèque JavaScript pour l'UI

- Une bibliothèque JavaScript open-source pour construire des interfaces utilisateur (UI).
- Créée et maintenue par Facebook (désormais Meta).
- Axée sur les composants : tout dans React est un composant.
- Utilise une approche déclarative : vous décrivez ce que vous voulez, et React s'occupe de mettre à jour le DOM.
- S'appuie sur le concept du "Virtual DOM" pour des performances optimisées.

Les concepts clés

Les fondations de React

- **Composants :** Les blocs de construction de votre UI. Ils peuvent être des fonctions (composants fonctionnels) ou des classes.
- **JSX :** Une extension de syntaxe pour JavaScript qui permet d'écrire du code qui ressemble à du HTML directement dans le code JavaScript.
- **Props (Propriétés) :** Le moyen de passer des données d'un composant parent à un composant enfant. Les props sont en lecture seule.
- **State (État) :** Un objet qui contient des données qui peuvent changer au fil du temps et qui influencent le rendu du composant. Quand l'état change, le composant se met à jour.
- **Virtual DOM :** Une copie en mémoire du DOM réel. React compare l'état actuel et le nouvel état, puis n'effectue que les changements nécessaires sur le DOM réel, ce qui est plus rapide.

Créer votre premier composant

Hello World en React

Code JSX

```jsx
import React from 'react';
import ReactDOM from 'react-dom';

function Bienvenue(props) {
  return <h1>Bonjour, {props.nom}</h1>;
}

const element = <Bienvenue nom="Ali" />;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

❖ **Explications :**

- ✓ On importe **React** et **ReactDOM**.
- ✓ On définit un composant fonctionnel **Bienvenue** qui accepte des **props**.
- ✓ On utilise le JSX pour le rendu de l'élément.
- ✓ **ReactDOM.render** lie le composant à un élément HTML existant.

Les Hooks (Hooks)

Les fonctions qui donnent du pouvoir aux composants

Introduits avec React 16.8 pour permettre aux composants fonctionnels d'avoir un état et des effets de cycle de vie, sans avoir à utiliser les classes.
Rendent le code plus lisible et plus simple à tester.

❖ **Hooks les plus courants :**

✓ **useState :** Pour ajouter un état local à un composant fonctionnel.

Exemple : **const [count, setCount] = useState(0);**

✓ **useEffect:** Pour gérer les effets de bord (appels API, gestion du DOM, etc.) dans un composant fonctionnel.

Exemple : **useEffect(() => { document.title = 'Mon titre'; });**

Gérer le State (useState)

Rendre votre UI dynamique

Code d'exemple

```jsx
import React, { useState } from 'react';

function Compteur() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Vous avez cliqué {count} fois</p>
      <button onClick={() => setCount(count + 1)}>
        Cliquez-moi
      </button>
    </div>
  );
}
```

❖ Explications :

- ✓ **useState(0)** initialise le compteur à 0.
- ✓ **setCount(count + 1)** met à jour la valeur de **count**.
- ✓ La modification de l'état entraîne le rendu du composant.

Les listes et les clés (Keys)

Afficher des collections de données

- **Comment faire :** Utiliser la méthode **map()** sur un tableau pour générer une liste d'éléments JSX.

- **L'importance des keys:**
  - ✓ Chaque élément d'une liste doit avoir un attribut **key** unique.
  - ✓ Cela aide React à identifier les éléments qui ont changé, ont été ajoutés ou supprimés.
  - ✓ **Key** doit être une chaîne de caractères ou un nombre unique (ex: un ID d'une base de données).

Gérer les événements

Interagir avec l'utilisateur

Syntaxe : Utiliser des attributs comme **onClick**, **onChange**, etc.

Exemple :

```
function handleClick() {
  alert('Bouton cliqué !');
}

<button onClick={handleClick}>Cliquez-moi</button>
```

❖ **Points importants :** Le nom de la fonction est passé comme référence, et non pas appelée directement (**handleClick** vs **handleClick()**).

Introduction au React Router

La navigation dans les applications SPA

- **Concept :** React est souvent utilisé pour créer des Single Page Applications (SPA). Pour gérer la navigation entre différentes vues, on utilise une bibliothèque comme **React Router**.

- **Points clés :**
  - ✓ **BrowserRouter :** Le routeur principal.
  - ✓ **Route :** Définit une route pour un composant.
  - ✓ **Link :** Pour créer des liens sans recharger la page.

Conclusion et prochaines étapes

Résumé et pistes de développement

Résumé :

- ✓ React est une bibliothèque pour construire des interfaces utilisateur basées sur des composants.
- ✓ Le JSX, les **props**, le state et les **hooks** sont des concepts fondamentaux.
- ✓ Le Virtual DOM assure des mises à jour performantes.

Prochaines étapes :

- ✓ Gérer l'état global avec des bibliothèques comme Redux ou le Context API.
- ✓ Apprendre à faire des requêtes API avec **fetch** ou **axios**.
- ✓ Découvrir le rendu côté serveur avec Next.js.
- ✓ S'intéresser aux tests de composants.

Merci de votre attention!