

Introduction à React

Qu'est-ce que React

- React, parfois appelé un framework JavaScript côté client, est une bibliothèque JavaScript créée par Facebook.
- React est une bibliothèque JavaScript pour construire des interfaces utilisateur.
- React est utilisé pour créer des “single-page applications”.
- React nous permet de créer des composants d'interface utilisateur réutilisables.

Comment fonctionne React ?

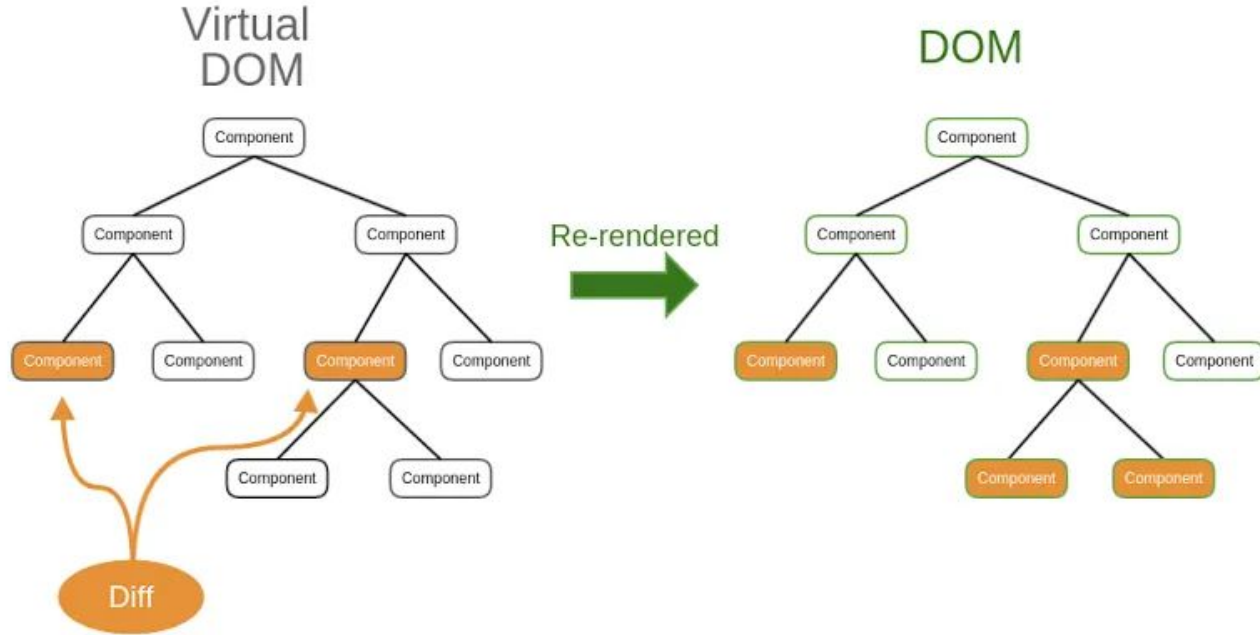
React crée un DOM VIRTUEL en mémoire.

Au lieu de manipuler directement le DOM du navigateur, React crée un DOM virtuel en mémoire, où il effectue toutes les manipulations nécessaires, avant d'apporter les modifications dans le DOM du navigateur.

React ne modifie que ce qui doit l'être !

React détecte les modifications apportées et ne modifie que ce qui doit l'être.

Comment fonctionne React ?



Render HTML avec React

La fonction `createRoot`

La fonction `createRoot()` prend un argument, un élément HTML.

Le but de cette fonction est de définir l'élément HTML où un composant React doit être affiché.

La méthode `render`

Ensuite, la méthode `render()` est appelée pour définir le composant React qui doit être rendu.

Il y a un autre dossier à la racine de votre projet React, nommé "public". Dans ce dossier, se trouve un fichier `index.html`.

Vous remarquerez un seul `<div>` dans le corps de ce fichier. C'est là que notre application React sera rendue.

Render HTML avec React - exemple



```
const container = document.getElementById('root');  
const root = ReactDOM.createRoot(container);  
root.render(<p>Hello</p>);
```

```
// inside index.html  
<body>  
  <div id="root"></div>  
</body>
```

React JSX

Qu'est-ce que le JSX ?

- JSX signifie JavaScript XML.
- Le JSX nous permet d'écrire du HTML dans React.
- Le JSX facilite l'écriture et l'ajout de HTML dans React.

Coder en JSX

Le JSX nous permet d'écrire des éléments HTML en JavaScript et de les placer dans le DOM sans avoir besoin de méthodes `createElement()` et/ou `appendChild()`.

Le JSX convertit les balises HTML en éléments React.

React JSX

Qu'est-ce que le JSX ?

- JSX signifie JavaScript XML.
- Le JSX nous permet d'écrire du HTML dans React.
- Le JSX facilite l'écriture et l'ajout de HTML dans React.

Coder en JSX

Le JSX nous permet d'écrire des éléments HTML en JavaScript et de les placer dans le DOM sans avoir besoin de méthodes `createElement()` et/ou `appendChild()`.

Le JSX convertit les balises HTML en éléments React.

ce n'est pas obligé d'utiliser le JSX, mais il facilite l'écriture d'applications React

React JSX - exemple



// Without JSX:

```
const myElement = React.createElement('h1', {}, 'I do not use JSX!');
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(myElement);
```

// JSX:

```
const myElement = <h1>This is JSX!</h1>;
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(myElement);
```

React JSX

Comme vous pouvez le voir dans le deuxième exemple, le JSX nous permet d'écrire du HTML directement dans le code JavaScript.

Le JSX est une extension du langage JavaScript basée sur ES6, et il est traduit en JavaScript standard lors de l'exécution à l'aide d'un compilateur tel que Babel.

Les composants React

Les composants sont des bouts de code indépendants et réutilisables. Ils remplissent le même rôle que les fonctions JavaScript, mais fonctionnent de manière isolée et renvoient du HTML.

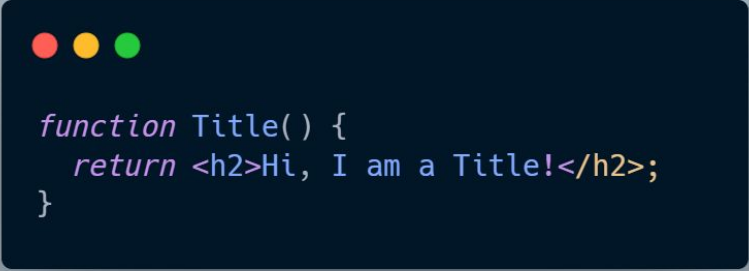
Les composants se déclinent en deux types : les composants de class (Class components) et les composants fonctionnels (Function components). Dans ce tutoriel, nous nous concentrerons sur les composants fonctionnels.

Dans d'anciens projets React, vous pourriez trouver principalement l'utilisation de composants de class. Il est désormais recommandé d'utiliser des composants fonctionnels en combinaison avec les Hooks, qui ont été ajoutés dans React 16.8.

Les composants React

Les composant Fonctionnel (Function components)

Un composant fonctionnel est une fonction qui renvoie du HTML




```
function Title() {  
  return <h2>Hi, I am a Title!</h2>;  
}
```

Les composants React - Props

- Les Props sont des arguments transmis aux composants React.
- Les Props sont transmises aux composants via des attributs HTML.
- "Props" signifie "propriétés".
- Les Props de React sont similaires aux arguments de fonction en JavaScript et aux attributs en HTML.
- Pour envoyer des props à un composant, utilisez la même syntaxe que pour les attributs HTML

NB: Les Propriétés de React sont en lecture seule ! Vous obtiendrez une erreur si vous essayez de changer leur valeur.

Les composants React - Props - example



```
const myElement = <Car brand="Renault" />;  
  
// Car Component  
function Car(props) {  
  return <h2>I am a { props.brand }!</h2>;  
}  
  
// result: I am a Renault
```

Les événements React


- Tout comme les événements DOM HTML, React peut effectuer des actions en fonction des événements utilisateur.
- React dispose des mêmes événements que le HTML : click, change, mouseover, etc.
- Les événements React sont écrits en syntaxe camelCase :

`onClick` au lieu de `onclick`.

- Les gestionnaires d'événements React sont écrits entre accolades :

`onClick={shoot}` au lieu de `onClick="shoot()"`.

Les événements React - example



```
// HTML
<button onclick="start()">Hi!</button>

// React
function Car() {
  const start = () => {
    alert("car started!");
  }

  return (
    <button onClick={start}>Start the car!
  </button>
  )
}
```


Les listes React

- In React, you will render lists with some type of loop.
- The JavaScript `map()` array method is generally the preferred method.
- Lorsque vous exécutez ce code dans votre `create-react-app`, il fonctionnera mais vous recevrez un avertissement indiquant qu'aucune "key" n'a été fournie pour les éléments de la liste.
- Les clés (keys) permettent à React de suivre les éléments. Ainsi, si un élément est mis à jour ou supprimé, seul cet élément sera re-rendu au lieu de toute la liste.
- Les clés doivent être uniques entre chaque élément frère (sibling). Cependant, elles peuvent être dupliquées à l'échelle globale.
- Généralement, la clé (`key`) devrait être un identifiant unique attribué à chaque élément. En dernier recours, vous pouvez utiliser l'indice du tableau comme clé.

Les listes React - example

```
function Car(props) {  
  return <li>I am a { props.brand }</li>;  
}  
  
function Garage() {  
  const cars = [  
    {id: 1, brand: 'Ford'},  
    {id: 2, brand: 'BMW'},  
    {id: 3, brand: 'Audi'}  
  ];  
  return (  
    <>  
      <h1>Who lives in my garage?</h1>  
      <ul>  
        {cars.map((car) => <Car key={car.id} brand={car.brand}>  
/>)} </ul>  
      </>  
    );  
  }  
}
```

Les Hooks React

- Les Hooks nous permettent de nous "accrocher" aux fonctionnalités de React telles que l'état (state) et les méthodes du cycle de vie.
- Les Hooks ne peuvent être appelés que à l'intérieur des composants fonctionnels React.
- Les Hooks ne peuvent être appelés qu'au niveau supérieur d'un composant.
- Les Hooks ne peuvent pas être conditionnels.
- Certains hooks intégrés:
 - useState
 - useEffect
 - useContext
 - useRef
 - useReducer
 - useCallback
 - useMemo
- Si vous avez de la logique qui doit être réutilisée dans plusieurs composants, vous pouvez construire vos propres Hooks personnalisés.

Les Hooks React - exemple

- Le Hook `useState` de React nous permet de suivre l'état (state) dans un composant fonctionnel.
- L'état (state) fait généralement référence aux données ou aux propriétés qui doivent être suivies dans une application.

Les Hooks React - example



```
import { useState } from "react";
import ReactDOM from "react-dom/client";

function FavoriteColor() {
  const [color, setColor] = useState("red");

  return (
    <>
      <h1>My favorite color is {color}!</h1>
      <button
        type="button"
        onClick={() => setColor("blue")}
      >Blue</button>
    </>
  )
}
```

MERCI