

React.js - Développeur Web

◆ Chapitre 1 : Introduction à React

◆ Présentation de React : Pourquoi l'utiliser ?

★ Qu'est-ce que React ?

React est une **bibliothèque JavaScript** développée par **Facebook (Meta)** et utilisée pour la création d'interfaces utilisateur interactives et réactives. Il est principalement utilisé pour construire des **applications web monopages (SPA - Single Page Applications)** et des interfaces dynamiques.

◆ Présentation de React : Pourquoi l'utiliser ?

★ Qu'est-ce que React ?

React est une **bibliothèque JavaScript** développée par **Facebook (Meta)** et utilisée pour la création d'interfaces utilisateur interactives et réactives. Il est principalement utilisé pour construire des **applications web monopages (SPA - Single Page Applications)** et des interfaces dynamiques.

✂ Pourquoi utiliser React ?

✓ Composants réutilisables

- React est basé sur une architecture **modulaire** où l'interface est découpée en **composants** indépendants et réutilisables.
- Facilite la maintenance et l'évolutivité des projets.

✓ Performance optimisée avec le Virtual DOM

- React utilise un **Virtual DOM** qui permet de mettre à jour l'interface de manière **efficace et rapide**, sans recharger la page entière.
- Seules les parties de l'UI qui ont changé sont mises à jour, améliorant ainsi les performances.

✓ Unidirectional Data Flow (Flux de données unidirectionnel)

- Contrairement à d'autres frameworks, React adopte un **flux de données unidirectionnel**, ce qui facilite la gestion de l'état et réduit les bugs.

✓ Grande communauté et écosystème riche

- Une communauté active avec des milliers de **bibliothèques et outils** disponibles (Redux, React Router, Tailwind, Material UI, etc.).

✓ Facilité d'apprentissage

- Syntaxe **JSX** intuitive qui mélange JavaScript et HTML.
- Approche déclarative qui simplifie le développement des interfaces utilisateur.

✓ Support mobile avec React Native

- React permet aussi de développer des **applications mobiles natives** avec **React Native**, en réutilisant une grande partie du code écrit pour le web.

◆ Installation et configuration de l'environnement (Node.js, npm, Vite/Create React App)

Avant de commencer à coder avec **React**, il faut configurer un environnement de développement adapté. Voici les étapes essentielles pour l'installation et la configuration.

1. Installation de Node.js et npm

Pourquoi ?

React utilise **Node.js** pour exécuter des outils comme **npm** (**N**ode **P**ackage **M**anager) et gérer les dépendances.

◆ Vérifier si Node.js est installé

Ouvre un terminal et tape la commande suivante :

```
node -v
```

Si Node.js est installé, la version s'affiche. Si ce n'est pas le cas, installe-le en suivant les étapes ci-dessous.

◆ Télécharger et installer Node.js

- Va sur <https://nodejs.org/>
- Télécharge la version **LTS (Long-Term Support)**
- Installe Node.js (npm est inclus avec Node.js)

◆ Vérifier npm

Après l'installation, vérifie que **npm** est bien installé avec :

```
npm -v
```

2. Créer un projet React avec Vite (Recommandé)

Pourquoi Vite ?

Vite est plus rapide que Create React App (CRA) car il optimise le développement avec un serveur de build ultra-rapide.

◆ **Installer Vite et créer un projet React**

Dans le terminal, exécute :

```
npm create vite@latest nom-du-projet --template react
```

Remplace **nom-du-projet** par le nom de ton projet.

◆ **Aller dans le dossier du projet**

```
cd nom-du-projet
```

◆ **Installer les dépendances**

```
npm install
```

◆ **Démarrer le projet**

```
npm run dev
```

Cela lancera un serveur local (par défaut sur `http://localhost:5173`).

3. Alternative : Créer un projet React avec Create React App (CRA)

⚠ **Create React App est plus lent** et moins optimisé que Vite, mais reste une option classique.

◆ **Créer un projet avec CRA**

```
npx create-react-app nom-du-projet
```

(Npx est inclus avec npm, il permet d'exécuter des paquets sans les installer globalement.)

◆ **Aller dans le dossier du projet et lancer le serveur**

```
cd nom-du-projet  
npm start
```

L'application s'ouvre sur `http://localhost:3000`.

◆ Concepts clés : Composants, JSX, Virtual DOM

Avant de commencer à coder en React, il est essentiel de comprendre ses concepts fondamentaux.

1. Composants (Components)

◆ Définition

- Un **composant** est un **bloc réutilisable** qui représente une partie de l'interface utilisateur (bouton, formulaire, carte, etc.).
- En React, **tout est basé sur des composants**.

◆ Types de composants

☞ Composants fonctionnels (recommandés)

- Ce sont de simples **fonctions JavaScript** qui retournent du JSX.
- Plus faciles à lire, tester et optimiser.

Exemple :

```
jsx

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

☞ Composants de classe (ancienne méthode)

- Définis avec une **classe ES6** et un `render()`.
- Utilisés avant l'introduction des **Hooks** (React 16.8).

Exemple :

```
jsx

class Bonjour extends React.Component {
  render() {
    return <h1>Salut, {this.props.nom} !</h1>;
  }
}
```

★ Bonnes pratiques

- ✓ Un composant **doit être indépendant et réutilisable**
- ✓ Toujours nommer les composants en **PascalCase** (`MonComposant.js`)

2. JSX (JavaScript XML)

◆ Définition

JSX est une **extension syntaxique** qui permet d'écrire du HTML directement dans du JavaScript.

◆ Pourquoi JSX ?

- ✓ Plus lisible et intuitif
- ✓ Permet de combiner logique et UI dans un seul fichier
- ✓ Sécurisé et optimisé après compilation

◆ Exemple de JSX

```
jsx  
  
const titre = <h1>Bienvenue</h1>;
```

◆ JSX vs JavaScript

Sans JSX, on écrirait ceci :

```
jsx  
  
const titre = React.createElement('h1', {}, 'Bienvenue');
```

Avec JSX, c'est **plus propre et lisible** ✓

✦ Bonnes pratiques JSX

- ✓ Un composant **doit retourner un seul élément parent**

✗ Mauvais :

```
jsx  
  
function App() {  
  return (  
    <h1>Salut</h1>  
    <p>Bienvenue</p>  
  );  
}
```

✓ Bon :

```
jsx  
  
function App() {  
  return (  
    <>  
      <h1>Salut</h1>  
      <p>Bienvenue</p>  
    </>  
  );  
}
```

```
);  
}
```

🔑 **Astuce** : Utiliser `<></>` (**Fragments**) pour éviter des `<div>` inutiles.

3. Virtual DOM 📄

🔑 Qu'est-ce que le DOM ?

Le **DOM (Document Object Model)** est la structure HTML interprétée par le navigateur.

🔑 Problème avec le DOM classique

- Modifier directement le DOM est **lent** ⚠️
- Chaque mise à jour **rafraîchit toute la page**, ce qui **ralentit les performances**

🔑 Solution : Virtual DOM

- **React crée une copie virtuelle du DOM en mémoire**
- Lorsqu'un changement est détecté, **React met à jour uniquement les parties modifiées**, au lieu de recharger toute la page
- Cela **améliore considérablement les performances**

🔑 Comment ça marche ?

1. React garde un **Virtual DOM** en mémoire
2. Lorsqu'un état change, React compare l'ancien et le nouveau Virtual DOM
3. Il met à jour **seulement les parties modifiées** du **vrai DOM**
4. 🔑 **Exemple illustré**

Action utilisateur	Virtual DOM met à jour	DOM réel est modifié
L'utilisateur clique sur un bouton	Virtual DOM met à jour le bouton	React met à jour uniquement ce bouton dans le DOM

★ Avantages du Virtual DOM

- ✓ **Optimisation des performances**
- ✓ **Moins de manipulations du DOM réel**
- ✓ **Expérience utilisateur fluide**

Conclusion

- ✓ **Les composants** rendent le code modulaire et réutilisable.
- ✓ **JSX** permet d'écrire du HTML directement dans JavaScript, rendant le code plus lisible.
- ✓ **Le Virtual DOM** améliore les performances en mettant à jour uniquement les parties nécessaires.

◆ Premier projet React : structure d'un projet

1. Structure d'un projet React

Après avoir créé un projet avec **Vite** (ou **Create React App**), voici la structure typique :

mon-projet-react/

```
| — node_modules/      # Dépendances installées
| — public/             # Fichiers publics (favicon, index.html...)
| — src/                # Code source de l'application
|   | — App.jsx         # Composant principal
|   | — main.jsx        # Point d'entrée de l'application
|   | — components/     # Dossier pour les composants React
|   | — assets/         # Images, styles et ressources
| — .gitignore          # Fichiers à ignorer par Git
| — package.json        # Liste des dépendances et scripts
| — vite.config.js      # Configuration de Vite
| — README.md           # Documentation du projet
```

Fichiers importants :

- `App.jsx` : Composant principal
- `main.jsx` : Monte l'application dans le DOM
- `package.json` : Contient les dépendances et scripts

2. Création d'une première application React

Nous allons créer une **application simple** qui affiche un message de bienvenue et un compteur interactif.

Étape 1 : Créer un projet React avec Vite

Dans ton terminal, exécute :

```
npm create vite@latest mon-premier-react --template react
```

```
cd mon-premier-react
```

```
npm install
```

```
npm run dev
```

Ouvre <http://localhost:5173> dans ton navigateur

```
D:\CFITECH\React\cours react 2025\exercices>npm create vite@latest test --template react
```

```
> npx
> create-vite test react

? Select a framework: » - Use arrow-keys. Return to submit.
  Vanilla
  Vue
>  React✓
  Preact
  Lit
  Svelte
  Solid
  Qwik
  Angular
  Others
```

```
> npx
> create-vite test react

✓ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
  TypeScript
  TypeScript + SWC
>  JavaScript✓
  JavaScript + SWC
  React Router v7 ↗
```



```
> npx
> create-vite test react

✓ Select a framework: » React
✓ Select a variant: » JavaScript

Scaffolding project in D:\CFITECH\React\cours react 2025\exercices\test...

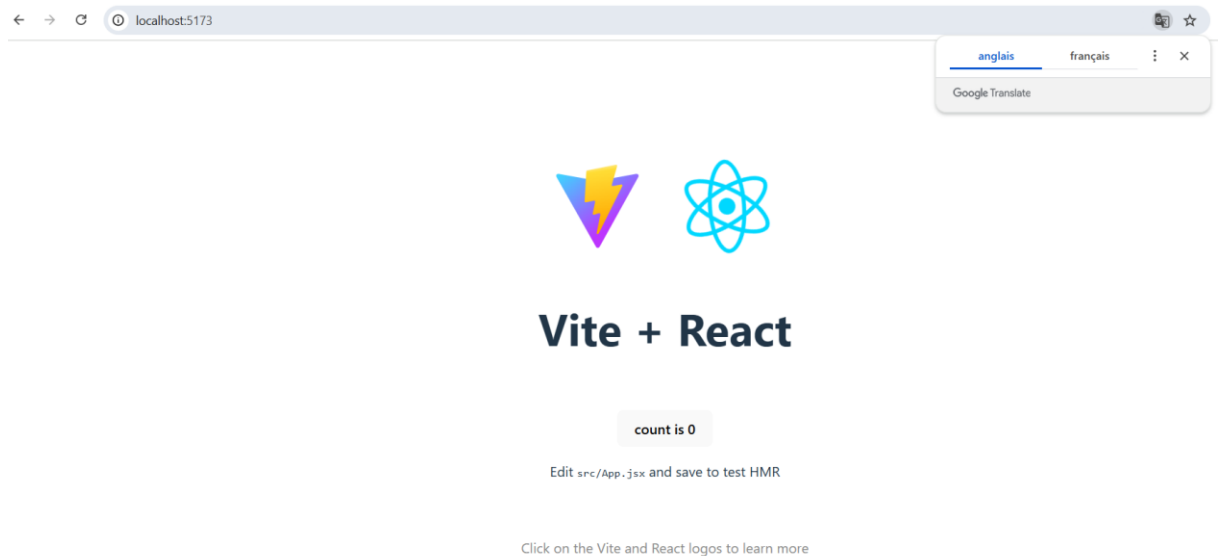
Done. Now run:

cd test ✓
npm install ✓
npm run dev ✓
```

```
C:\Windows\system32\cmd.e: X + v

VITE v6.0.11 ready in 270 ms

→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```



Étape 2 : Modifier `App.jsx` pour afficher un message de bienvenue

Dans `src/App.jsx`, remplace le code par ceci :

`jsx`

```
import { useState } from 'react';

import './App.css';

function App() {

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

  return (

    <div className="container">

      <h1>Bienvenue sur mon premier projet React !</h1>

      <p>Ceci est une application React simple.</p>

      <h2>Compteur : {count}</h2>

      <button onClick={() => setCount(count + 1)}>➕ Augmenter</button>

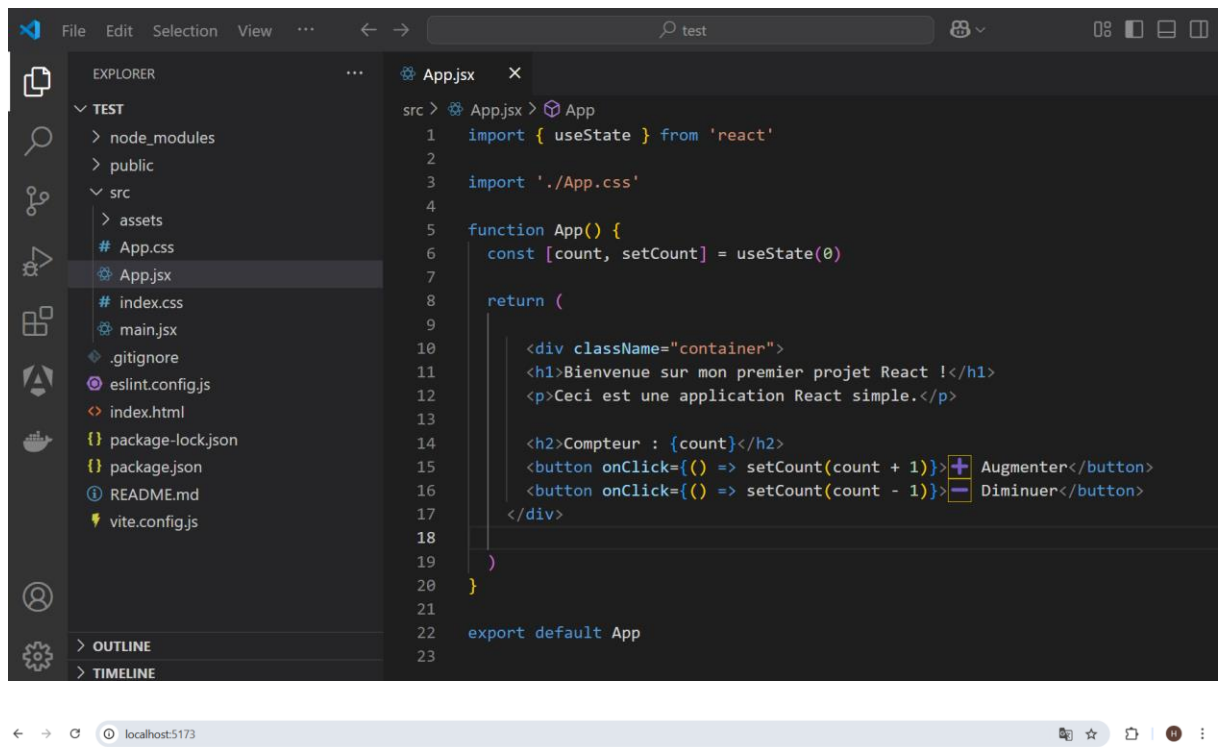
      <button onClick={() => setCount(count - 1)}>➖ Diminuer</button>

    </div>

  );

}

export default App;
```



Bienvenue sur mon premier projet React !

Ceci est une application React simple.

Compteur : -4

+ Augmenter - Diminuer

Étape 3 : Ajouter un peu de style

Dans `src/App.css`, remplace le contenu par :

```
.container {  
  
  text-align: center;  
  
  font-family: Arial, sans-serif;  
  
  margin-top: 50px;  
  
}
```

```
h1 {  
  color: #2c3e50;  
}
```

```
button {  
  margin: 10px;  
  padding: 10px 15px;  
  font-size: 16px;  
  cursor: pointer;  
  border: none;  
  border-radius: 5px;  
}
```

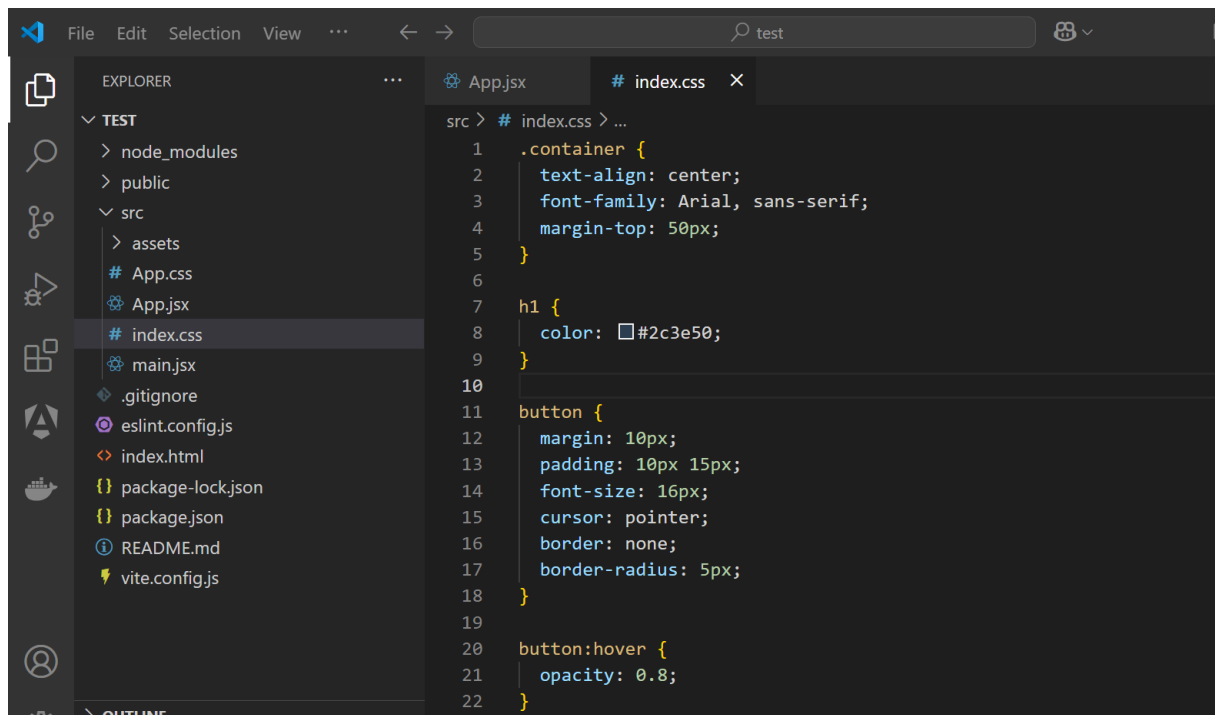
```
button:hover {  
  opacity: 0.8;  
}
```

```
button:first-of-type {  
  background-color: #27ae60;  
  color: white;  
}
```


```
button:last-of-type {  
  background-color: #c0392b;
```

```
color: white;

}
```



```
src > # index.css > ...
1  .container {
2      text-align: center;
3      font-family: Arial, sans-serif;
4      margin-top: 50px;
5  }
6
7  h1 {
8      color: #2c3e50;
9  }
10
11 button {
12     margin: 10px;
13     padding: 10px 15px;
14     font-size: 16px;
15     cursor: pointer;
16     border: none;
17     border-radius: 5px;
18 }
19
20 button:hover {
21     opacity: 0.8;
22 }
```



```
23
24 button:first-of-type {
25     background-color: #27ae60;
26     color: white;
27 }
28
29 button:last-of-type {
30     background-color: #c0392b;
31     color: white;
32 }
33
```

Étape 4 : Lancer l'application

Dans le terminal, tape :

```
npm run dev
```

Bienvenue sur mon premier projet React !

Ceci est une application React simple.

Compteur : 0



```
D:\CFITECH\React\cours react 2025\exercices\test>npm run dev
```

Résumé

- ✓ On a créé un projet React avec Vite
- ✓ On a compris la structure d'un projet
- ✓ On a ajouté un premier composant avec un état (`useState`)
- ✓ On a appliqué du CSS pour améliorer l'interface