Cours MERN - Semaine 6

Introduction à React et Vite

Composants, JSX et Props

Abdelweheb GUEDDES & Mohamed Ben Jazia / Ecole Polytechnique Sousse

27 octobre 2025

Table des matières

1 Objectifs Pédagogiques

À la fin de cette séance, vous serez capable de :
- **Comprendre** la différence entre une application multi-pages (MPA) et une Single Page Application (SPA)
- **Expliquer** ce qu'est React et son rôle dans le développement frontend
- **Maîtriser** la syntaxe JSX et ses règles fondamentales
- **Créer** des composants React fonctionnels réutilisables
- **Utiliser** les props pour transmettre des données entre composants
- **Initialiser** un projet React moderne avec Vite
- **Afficher** des listes de données avec la méthode `.map()`
- **Structurer** une application React selon les bonnes pratiques

2 Partie 1 : Concepts Théoriques (1h30)

2.1 Single Page Application (SPA)

2.1.1 Application Multi-Pages Traditionnelle

Dans une application web traditionnelle :
1. L'utilisateur clique sur un lien
2. Le navigateur charge une nouvelle page HTML complète depuis le serveur
3. Tout le contenu (header, footer, etc.) est rechargé
4. L'écran "clignote" pendant le chargement

Problèmes :
- Lent (requête réseau à chaque clic)
- Expérience utilisateur saccadée
- Duplication du code (header/footer rechargés à chaque fois)

2.1.2 Single Page Application (SPA)

React permet de créer des SPAs où :
- Une seule page HTML est chargée au départ
- JavaScript génère et met à jour le contenu dynamiquement
- Pas de rechargement complet de la page
- Navigation fluide et rapide

Analogie : Imaginez un livre avec une seule page magique qui change son contenu au lieu de tourner physiquement les pages. C'est le principe d'une SPA !

2.2 React : La Bibliothèque UI

2.2.1 Qu'est-ce que React ?

React est une **bibliothèque JavaScript** développée par Facebook (Meta) pour construire des interfaces utilisateur.

Caractéristiques :
- **Déclaratif :** On décrit QUOI afficher, pas COMMENT

— **Basé sur les composants :** L'interface est divisée en morceaux réutilisables
— **Learn Once, Write Anywhere :** Peut être utilisé pour web, mobile (React Native), desktop

2.2.2 La Philosophie des Composants

Un composant est une fonction JavaScript qui retourne du JSX (ce qui doit s'afficher).
Principe : Diviser une interface complexe en petits morceaux indépendants.

```
function Greeting() {
    return <h1>Bonjour le monde !</h1>;
}
```

Listing 1 – Un composant simple

Analogie : Si votre interface est une maison, les composants sont les briques. Chaque brique a un rôle précis et peut être réutilisée.

2.3 Le JSX : JavaScript + XML

2.3.1 Qu'est-ce que le JSX ?

JSX (JavaScript XML) est une extension syntaxique qui ressemble à du HTML mais qui est en fait du JavaScript.

Important : Le JSX n'est PAS du HTML ! C'est une syntaxe spéciale qui sera transformée en JavaScript par Vite.

```
// Ce que vous écrivez (JSX)
const element = <h1>Hello</h1>;

// Ce que Vite/Babel transforme (JavaScript)
const element = React.createElement('h1', null, 'Hello');
```

Listing 2 – Transformation JSX

2.3.2 Règles Fondamentales du JSX

1. Un seul élément racine

```
//  ERREUR - Plusieurs racines
function Wrong() {
    return (
        <h1>Titre</h1>
        <p>Paragraphe</p>
    );
}

//  CORRECT - Enveloppé dans une div
function Correct() {
    return (
        <div>
```

```
            <h1>Titre</h1>
            <p>Paragraphe</p>
        </div>
    );
}

//  CORRECT - Fragment (pas de div supplémentaire)
function BetterWay() {
    return (
        <>
            <h1>Titre</h1>
            <p>Paragraphe</p>
        </>
    );
}
```

2. Attributs en camelCase

```
// HTML traditionnel
<div class="container" onclick="handleClick()">
    <label for="name">Nom</label>
</div>

// JSX (camelCase)
<div className="container" onClick={handleClick}>
    <label htmlFor="name">Nom</label>
</div>
```

Pourquoi ? Parce que `class` et `for` sont des mots réservés en JavaScript !

3. Expressions JavaScript entre accolades

```
function UserCard() {
    const name = "Alice";
    const age = 25;
    const isAdmin = true;

    return (
        <div>
            {/* Variable */}
            <h1>{name}</h1>

            {/* Expression */}
            <p>Age dans 5 ans : {age + 5}</p>

            {/* Condition ternaire */}
            <span>{isAdmin ? "Admin" : "User"}</span>

            {/* Condition avec && */}
            {isAdmin && <button>Panneau Admin</button>}

            {/* Fonction */}
            <p>{name.toUpperCase()}</p>
        </div>
    );
```

```
}
```

Listing 3 – Utilisation des accolades

4. Fermeture obligatoire des balises

```
//  HTML valide mais JSX invalide
<input type="text">
<img src="photo.jpg">

//  JSX - Balises auto-fermantes
<input type="text" />
<img src="photo.jpg" />
```

2.4 Les Props : Communication entre Composants

2.4.1 Concept

Les **props** (properties) permettent de passer des données d'un composant parent à un composant enfant.

Analogie : C'est comme passer des paramètres à une fonction.

```
// Composant Parent
function App() {
    return (
        <div>
            <WelcomeMessage name="Alice" age={25} />
            <WelcomeMessage name="Bob" age={30} />
        </div>
    );
}

// Composant Enfant
function WelcomeMessage(props) {
    return (
        <div>
            <h2>Bienvenue {props.name} !</h2>
            <p>Vous avez {props.age} ans</p>
        </div>
    );
}
```

Listing 4 – Passage et utilisation de props

2.4.2 Déstructuration des Props

Pour un code plus lisible, on peut déstructurer les props :

```
// Avec props
function UserCard(props) {
    return <h1>{props.name}</h1>;
}

// Avec déstructuration (recommandé)
```

```
function UserCard({ name, age, job }) {
    return (
        <div>
            <h1>{name}</h1>
            <p>Age : {age}</p>
            <p>Métier : {job}</p>
        </div>
    );
}
```

2.4.3 Props avec Valeurs par Défaut

```
function Button({ text = "Cliquer", color = "blue" }) {
    return (
        <button style={{ backgroundColor: color }}>
            {text}
        </button>
    );
}

// Utilisation
<Button text="Envoyer" color="green" />
<Button /> {/* Utilisera "Cliquer" et "blue" */}
```

Listing 5 – Valeurs par défaut

Règle importante : Les props sont en **lecture seule**. Un composant enfant ne peut jamais modifier ses props. Les données vont toujours du parent vers l'enfant (flux unidirectionnel).

2.5 Afficher des Listes avec .map()

Pour afficher une liste d'éléments, on utilise la méthode `.map()` de JavaScript :

```
function StudentList() {
    const students = [
        { id: 1, name: "Alice", grade: 18 },
        { id: 2, name: "Bob", grade: 15 },
        { id: 3, name: "Charlie", grade: 16 }
    ];

    return (
        <ul>
            {students.map((student) => (
                <li key={student.id}>
                    {student.name} : {student.grade}/20
                </li>
            ))}
        </ul>
    );
```

```
17 }
```

Listing 6 – Affichage d'une liste

La prop `key` est obligatoire !

Chaque élément d'une liste doit avoir une prop `key` unique. Elle aide React à identifier quel élément a changé, a été ajouté ou supprimé.

Règles :

— Utiliser un identifiant unique (ID de la base de données)
— Ne PAS utiliser l'index du tableau si la liste peut changer d'ordre
— La key doit être stable (pas de `Math.random()`)

2.6 Vite : L'Outil de Build Rapide

2.6.1 Pourquoi Vite ?

Vite est un outil de build moderne qui remplace Create React App (CRA).

Avantages :

— **Démarrage ultra-rapide :** $<$ 1 seconde vs 30+ secondes pour CRA
— **Hot Module Replacement :** Vos modifications apparaissent instantanément
— **Build optimisé :** Production rapide et efficace
— **Simple :** Configuration minimale

Comment Vite est si rapide ?

Vite utilise les **modules ES natifs** du navigateur moderne. Au lieu de bundler tous vos fichiers avant de démarrer, Vite sert les fichiers directement et laisse le navigateur les charger à la demande.

Comparaison :

— Webpack/CRA : Bundle tout → attend → démarre (lent)
— Vite : Démarre → sert à la demande (rapide)

3 Partie 2 : Atelier Pratique (1h30)

3.1 Étape 1 : Créer le Projet avec Vite

```
# Créer le projet
npm create vite@latest mon-blog-react -- --template react

# Se déplacer dans le dossier
cd mon-blog-react

# Installer les dépendances
npm install

# Lancer le serveur de développement
npm run dev
```

Listing 7 – Installation

Ouvrez votre navigateur à `http://localhost:5173`

3.2 Étape 2 : Nettoyer et Comprendre la Structure

Structure générée :

```
mon-blog-react/
|-- node_modules/       # Dependances
|-- public/             # Fichiers statiques
|-- src/
|   |-- App.jsx         # Composant principal
|   |-- main.jsx        # Point d'entree
|   +-- index.css       # Styles
|-- index.html          # Page HTML
|-- package.json        # Configuration npm
+-- vite.config.js      # Configuration Vite
```

Nettoyage :

1. Supprimer le contenu inutile de `src/App.jsx`
2. Créer un dossier `src/components`

3.3 Étape 3 : Créer le Premier Composant

```
function App() {
  return (
    <div>
      <h1>Mon Blog React</h1>
      <p>Bienvenue sur mon premier projet React !</p>
    </div>
  );
}

export default App;
```

Listing 8 – src/App.jsx - Version simple

3.4 Étape 4 : Créer un Composant Article

```jsx
function Article({ title, author, content }) {
  return (
    <article style={{
      border: '1px solid #ddd',
      padding: '20px',
      marginBottom: '20px',
      borderRadius: '8px'
    }}>
      <h2>{title}</h2>
      <p style={{ color: '#666', fontSize: '14px' }}>
        Par {author}
      </p>
      <p>{content}</p>
    </article>
  );
}

export default Article;
```

Listing 9 – src/components/Article.jsx

3.5 Étape 5 : Utiliser le Composant dans App

```jsx
import Article from './components/Article';

function App() {
  return (
    <div style={{ maxWidth: '800px', margin: '0 auto', padding: '20px' }}>
      <h1>Mon Blog React</h1>

      <Article
        title="Introduction à React"
        author="Alice"
        content="React est une bibliothèque JavaScript pour construire des interfaces..."
      />

      <Article
        title="Qu'est-ce que Vite ?"
        author="Bob"
        content="Vite est un outil de build ultra-rapide pour les projets frontend..."
      />
    </div>
  );
}

```

```
export default App;
```

Listing 10 – src/App.jsx - Avec composant

3.6 Étape 6 : Afficher une Liste d'Articles

```
import Article from './components/Article';

function App() {
  // Données d'articles
  const articles = [
    {
      id: 1,
      title: "Introduction à React",
      author: "Alice",
      content: "React est une bibliothèque JavaScript..."
    },
    {
      id: 2,
      title: "Qu'est-ce que Vite ?",
      author: "Bob",
      content: "Vite est un outil de build ultra-rapide..."
    },
    {
      id: 3,
      title: "Les Composants en React",
      author: "Charlie",
      content: "Les composants sont les briques de base..."
    }
  ];

  return (
    <div style={{ maxWidth: '800px', margin: '0 auto', padding: '20px' }}>
      <h1>Mon Blog React</h1>
      <p style={{ color: '#666' }}>
        {articles.length} articles disponibles
      </p>

      {articles.map((article) => (
        <Article
          key={article.id}
          title={article.title}
          author={article.author}
          content={article.content}
        />
      ))}
    </div>
  );
}

```

```jsx
export default App;
```

Listing 11 – src/App.jsx - Avec liste

3.7 Étape 7 : Créer un Composant Header

```jsx
function Header({ title, subtitle }) {
  return (
    <header style={{
      backgroundColor: '#3b82f6',
      color: 'white',
      padding: '30px 20px',
      textAlign: 'center',
      marginBottom: '30px',
      borderRadius: '8px'
    }}>
      <h1 style={{ margin: 0, fontSize: '2.5rem' }}>
        {title}
      </h1>
      {subtitle && (
        <p style={{ margin: '10px 0 0 0', opacity: 0.9 }}>
          {subtitle}
        </p>
      )}
    </header>
  );
}

export default Header;
```

Listing 12 – src/components/Header.jsx

3.8 Étape 8 : Créer un Composant Footer

```jsx
function Footer({ author, year }) {
  return (
    <footer style={{
      textAlign: 'center',
      padding: '20px',
      marginTop: '40px',
      borderTop: '2px solid #ddd',
      color: '#666'
    }}>
      <p> {year} {author} – Tous droits réservés</p>
      <p style={{ fontSize: '14px', marginTop: '10px' }}>
        Créé avec React + Vite
      </p>
    </footer>
  );
}

```

```jsx
export default Footer;
```

Listing 13 – src/components/Footer.jsx

3.9 Étape 9 : Application Finale Complète

```jsx
import Header from './components/Header';
import Article from './components/Article';
import Footer from './components/Footer';

function App() {
  const articles = [
    {
      id: 1,
      title: "Introduction à React",
      author: "Alice",
      content: "React est une bibliothèque JavaScript pour construire des interfaces utilisateur modernes et réactives."
    },
    {
      id: 2,
      title: "Qu'est-ce que Vite ?",
      author: "Bob",
      content: "Vite est un outil de build ultra-rapide qui améliore drastiquement l'expérience de développement."
    },
    {
      id: 3,
      title: "Les Composants en React",
      author: "Charlie",
      content: "Les composants sont les briques de base de React. Ils permettent de diviser l'interface en morceaux réutilisables."
    },
    {
      id: 4,
      title: "Le JSX en Pratique",
      author: "Alice",
      content: "JSX est une syntaxe qui ressemble à HTML mais qui est en fait du JavaScript transformé."
    }
  ];

  const currentYear = new Date().getFullYear();

  return (
    <div style={{
      minHeight: '100vh',
      display: 'flex',
      flexDirection: 'column'
    }}>
```

```
      <div style={{ maxWidth: '800px', margin: '0 auto', padding: '20px', flex: 1 }}>
        <Header
          title="Mon Blog React"
          subtitle="Apprendre React avec des exemples pratiques"
        />

        <div style={{ marginBottom: '20px', padding: '15px', backgroundColor: '#f0f9ff', borderRadius: '8px' }}>
          <strong>{articles.length}</strong> articles disponibles
        </div>

        {articles.map((article) => (
          <Article
            key={article.id}
            title={article.title}
            author={article.author}
            content={article.content}
          />
        ))}
      </div>

      <Footer
        author="Ecole Polytechnique Sousse"
        year={currentYear}
      />
    </div>
  );
}

export default App;
```

Listing 14 – src/App.jsx - Version finale

4 Concepts Clés à Retenir

Les 5 Piliers de cette Séance :

1. **Composants :** Fonctions qui retournent du JSX

```
function MonComposant() {
    return <div>Contenu</div>;
}

```

2. **JSX :** Syntaxe ressemblant à HTML dans JavaScript

```
<h1>{variable}</h1>
<div className="class" />

```

3. **Props :** Passage de données parent → enfant

```
<Enfant name="Alice" age={25} />
```

4. **Listes :** Utiliser .map() avec key

```
{items.map(item => <li key={item.id}>{item.name}</li>)}
```

5. **Structure :** Diviser en composants réutilisables

5 Exercices Pratiques

Exercice 1 : Composant Badge

Créez un composant `Badge` qui affiche un badge coloré.

Props attendues :

- `text` : Le texte à afficher
- `color` : La couleur du badge (par défaut : "blue")

Utilisation :

```
<Badge text="Nouveau" color="green" />
<Badge text="Populaire" color="red" />
```

Style suggéré :

```
{
    display: 'inline-block',
    padding: '5px 10px',
    backgroundColor: color,
    color: 'white',
    borderRadius: '4px',
    fontSize: '12px',
    fontWeight: 'bold'
}
```

Exercice 2 : Liste de Produits

Créez un composant `ProductCard` qui affiche :

- Un nom de produit
- Un prix
- Une catégorie

Puis dans `App.jsx`, affichez une liste de 5 produits avec leurs informations.

Données exemple :

```
const products = [
    { id: 1, name: "Laptop", price: 1200, category: "Electronique" },
    { id: 2, name: "Souris", price: 25, category: "Accessoires" },
    // ... 3 autres produits
];
```

Exercice 3 : Composant CommentCard

Créez un système de commentaires avec :

1. Un composant `CommentCard` avec props :
   - `author` : Nom de l'auteur
   - `text` : Texte du commentaire
   - `likes` : Nombre de likes
2. Affichez une liste de 3-4 commentaires
3. Bonus : Ajoutez un composant `CommentList` qui reçoit un tableau de commentaires en prop

6 Conclusion

Félicitations ! Vous avez appris les fondamentaux de React :

- Créer des composants
- Utiliser le JSX
- Passer des props
- Afficher des listes
- Structurer une application

Prochaine semaine : Nous verrons comment rendre nos composants **interactifs** avec le **State** (useState) pour gérer des données qui changent !

Travail à Rendre

Votre compte rendu doit inclure :

1. **Le code de vos 3 composants : `Header`, `Footer`, `Article`**
2. **Le code de votre `App.jsx` avec la liste d'articles**
3. **Une explication avec VOS MOTS du rôle de la prop `key`**
4. **Une capture d'écran de votre application fonctionnelle**

Échéance : La veille de la prochaine séance à 23h59

Ce travail est obligatoire et noté.