Script de Déroulement du Cours React

> Ce document détaille le déroulé complet du transfert de compétences React, minute par minute.

Vue d'Ensemble

Jour	Thème	Durée	Exercices	Projet
Jour 1	Fondamentaux React	7h	ex01 → ex07	-
Jour 2	État & Formulaires	7h	ex08 → ex10 + module2/ex01-02	Étapes 1-2
Jour 3	React Avancé	7h	module2/ex03 → ex06	Étapes 3-4
Jour 4	Architecture & Synthèse	7h	module2/ex07 → ex10	Étapes 5-6

JOUR 1 : Les Fondamentaux de React

🕘 09:00-09:30 | Accueil et Introduction (30 min)

Contenu

- Tour de table rapide (prénom, expérience JS, attentes)
- Présentation des objectifs de la formation
- Vérification de l'environnement (Node.js, VS Code, extensions)

Actions

```
# Vérifier les prérequis
node -v    # v18+
npm -v     # v9+
```

Slides

📊 **React_Les_Fondamentaux.pdf** - Section "Introduction"

🕘 09:30-10:15 | Qu'est-ce que React ? (45 min)

Contenu Théorique

1. Historique (Facebook 2013 → Meta 2024)
2. Philosophie : UI = f(state)
3. Virtual DOM expliqué simplement
4. Composants : l'analogie LEGO

5. JSX : HTML + JavaScript = ❤️

Slides

📊 **React_Les_Fondamentaux.pdf** - Sections "Présentation" et "Virtual DOM"

Démonstration Live

```
// Créer un projet Vite ensemble
npm create vite@latest demo-react -- --template react
cd demo-react
npm install
npm run dev
```

Montrer :

- Structure du projet Vite
- `main.jsx` et `App.jsx`
- Hot Module Replacement (HMR) en action

🕙 10:15-10:30 | ☕ Pause (15 min)

🕙 10:30-11:30 | JSX en Profondeur (1h)

Contenu Théorique (20 min)

📊 **React_Les_Fondamentaux.pdf** - Section "JSX"

Points clés :

1. Les 5 règles d'or du JSX (voir `cheatsheets/01-jsx-cheatsheet.md`)
2. Expressions JavaScript avec `{}`
3. `className` au lieu de `class`
4. Attributs en camelCase
5. Fragments `<>...</>`

Démonstration Live (10 min)

```jsx
function Demo() {
  const nom = "React";
  const age = new Date().getFullYear() - 2013;

  return (
    <div className="container">
      <h1>Bonjour {nom} !</h1>
      <p>Tu as {age} ans</p>
      <p>2 + 2 = {2 + 2}</p>
```

```
      </div>
    );
  }
```

🎯 Exercice 01 : Carte de Visite (30 min)

Lancer l'exercice :

> "Ouvrez `exercices/module-1-bases/ex01-carte-visite.md`"

Objectif : Créer une carte de visite avec des expressions JSX

Timing :

- 5 min : Lecture de l'énoncé
- 15 min : Réalisation autonome (indices disponibles)
- 10 min : Correction collective + questions

Points de vigilance :

- Calcul dynamique de l'âge
- Un seul élément parent
- `className` et non `class`

🕥 11:30-12:30 | Composants et Structure (1h)

Contenu Théorique (15 min)

📊 **React_Les_Fondamentaux.pdf** - Section "Composants"

Points clés :

1. Un composant = une fonction qui retourne du JSX
2. Nom en PascalCase
3. Un fichier = un composant (convention)
4. Import/Export ES6

🎯 Exercice 02 : Header et Footer (30 min)

Lancer l'exercice :

> "Ouvrez `exercices/module-1-bases/ex02-header-footer.md`"

Objectif : Créer et composer des composants

Timing :

- 5 min : Lecture
- 20 min : Réalisation
- 5 min : Correction rapide

Transition vers les Props (15 min)

Montrer le problème :

```
// Header.jsx - Le texte est "en dur"
function Header() {
  return <header><h1>Mon Site</h1></header>;
}

// Comment le rendre dynamique ?
```

🕐 12:30-14:00 | 🍽 Pause Déjeuner (1h30)

🕐 14:00-15:00 | Props : Passer des Données (1h)

Contenu Théorique (20 min)

📊 **React_Les_Fondamentaux.pdf** - Section "Props"

Référence : `cheatsheets/02-composants-props-cheatsheet.md`

Points clés :

1. Props = paramètres du composant
2. Toujours en lecture seule (immutables)
3. Déstructuration des props
4. Props par défaut
5. `children` : la prop spéciale

Démonstration Live (10 min)

```
// Évolution du Header
function Header({ titre, sousTitre = "Bienvenue" }) {
  return (
    <header>
      <h1>{titre}</h1>
      <p>{sousTitre}</p>
    </header>
  );
}

// Utilisation
<Header titre="TeamHub" sousTitre="Votre intranet" />
<Header titre="Autre Page" />  {/* sousTitre = "Bienvenue" */}
```

🎯 Exercice 05 : Carte Utilisateur (30 min)

Lancer l'exercice :

> "Ouvrez `exercices/module-1-bases/ex05-carte-utilisateur.md`"

Objectif : Passer et déstructurer des props

Timing :

- 5 min : Lecture
- 20 min : Réalisation
- 5 min : Correction

15:00-15:15 | Pause (15 min)

15:15-16:15 | Listes et Rendu (1h)

Contenu Théorique (15 min)

React_Les_Fondamentaux.pdf - Section "Listes"

Référence : `cheatsheets/04-rendu-conditionnel-listes-cheatsheet.md`

Points clés :

1. `.map()` pour itérer
2. `key` : pourquoi c'est obligatoire
3. Jamais l'index comme key (sauf liste statique)

Exercice 03 : Liste de Courses (25 min)

Lancer l'exercice :

> "Ouvrez `exercices/module-1-bases/ex03-liste-courses.md`"

Timing :

- 5 min : Lecture
- 15 min : Réalisation
- 5 min : Correction

Exercice 06 : Galerie d'Images (20 min)

Lancer l'exercice :

> "Ouvrez `exercices/module-1-bases/ex06-galerie-images.md`"

Objectif : Combiner map() avec des images et du CSS

Timing :

- 5 min : Lecture

- 12 min : Réalisation
- 3 min : Correction rapide

🕓 16:15-17:00 | Rendu Conditionnel (45 min)

Contenu Théorique (15 min)

📊 **React_Les_Fondamentaux.pdf** - Section "Rendu Conditionnel"

Les 3 méthodes :

1. Ternaire : `condition ? <A /> : <B />`
2. ET logique : `condition && <A />`
3. If/else avec variable

⚠ Piège du `0` avec `&&`

🎯 Exercice 07 : Rendu Conditionnel (30 min)

Lancer l'exercice :

> "Ouvrez `exercices/module-1-bases/ex07-rendu-conditionnel.md`"

Timing :

- 5 min : Lecture
- 20 min : Réalisation
- 5 min : Correction

🕓 17:00-17:30 | Introduction à useState (30 min)

Contenu Théorique (20 min)

📊 **React_Les_Fondamentaux.pdf** - Section "useState"

Référence : `cheatsheets/03-hooks-cheatsheet.md`

Points clés :

1. Pourquoi `let` ne suffit pas (re-render)
2. Syntaxe : `const [state, setState] = useState(initial)`
3. setState déclenche un re-render
4. setState est asynchrone
5. Forme fonctionnelle : `setState(prev => prev + 1)`

Démonstration Live (10 min)

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

function Compteur() {
```

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

  return (
    <div>
      <p>Compteur : {count}</p>
      <button onClick={() => setCount(count + 1)}>+1</button>
      <button onClick={() => setCount(prev => prev - 1)}>-1</button>
    </div>
  );
}
```

🕘 17:30-18:00 | Mise en Pratique useState (30 min)

🎯 Exercice 04 : Compteur Interactif

Lancer l'exercice :

> "Ouvrez `exercices/module-1-bases/ex04-compteur.md`"

Objectif : Premier composant interactif avec useState

Timing :

- 5 min : Lecture
- 20 min : Réalisation
- 5 min : Correction

🕕 18:00 | Fin du Jour 1

Récapitulatif (5 min)

- JSX et expressions
- Composants et composition
- Props et passage de données
- map() et key pour les listes
- Rendu conditionnel
- useState pour l'interactivité

Devoirs (optionnel)

- Relire les cheatsheets 01 à 04
- Tenter l'exercice 08 en autonomie

JOUR 2 : État Avancé et Formulaires

🕘 09:00-09:15 | Révisions Jour 1 (15 min)

Quiz Rapide

1. Quelle est la différence entre props et state ?
2. Pourquoi utiliser `key` dans une liste ?
3. Que fait `setState` ?

Questions/Réponses

🕘 09:15-10:00 | Classes Dynamiques (45 min)

Contenu Théorique (15 min)

- Template literals pour className
- Bibliothèques (clsx, classnames) - mention seulement
- Pattern toggle

🎯 Exercice 08 : Toggle Thème (30 min)

Lancer l'exercice :

> "Ouvrez `exercices/module-1-bases/ex08-toggle-theme.md`"

Objectif : Gérer des classes CSS dynamiquement

Timing :

- 5 min : Lecture
- 20 min : Réalisation
- 5 min : Correction

🕘 10:00-10:15 | ☕ Pause (15 min)

🕘 10:15-11:30 | Formulaires Contrôlés (1h15)

Contenu Théorique (20 min)

📊 **React_Les_Fondamentaux.pdf** - Section "Formulaires"

Points clés :

1. Input contrôlé : `value` + `onChange`
2. État pour chaque champ (ou objet)
3. Gestion du `submit`
4. Validation basique

Démonstration Live (15 min)

```jsx
function LoginForm() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
    if (!email.includes('@')) {
      setError('Email invalide');
      return;
    }
    console.log('Connexion avec', { email, password });
  };

  return (
    <form onSubmit={handleSubmit}>
      {error && <p className="error">{error}</p>}
      <input
        type="email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        placeholder="Email"
      />
      <input
        type="password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
        placeholder="Mot de passe"
      />
      <button type="submit">Connexion</button>
    </form>
  );
}
```

🎯 Exercice 09 : Formulaire de Contact (40 min)

Lancer l'exercice :

> "Ouvrez `exercices/module-1-bases/ex09-formulaire-contact.md`"

Objectif : Formulaire complet avec validation

Timing :

- 5 min : Lecture
- 30 min : Réalisation
- 5 min : Correction

🕥 11:30-12:30 | Mini-App Synthèse : Todo List (1h)

🎯 Exercice 10 : Application Todo

Lancer l'exercice :

> "Ouvrez `exercices/module-1-bases/ex10-todo-app.md`"

Contexte :

> "C'est l'exercice de synthèse du Module 1. Il combine tout ce qu'on a vu."

Fonctionnalités :

- Ajouter une tâche
- Marquer comme complétée
- Supprimer une tâche
- Filtrer (toutes/actives/complétées)
- Compteur

Timing :

- 10 min : Lecture et questions
- 40 min : Réalisation (accompagnement actif)
- 10 min : Correction et discussion

Conseil :

> "Commencez par la structure statique, puis ajoutez l'interactivité progressivement."

🕐 12:30-14:00 | 🍽 Pause Déjeuner (1h30)

🕐 14:00-14:30 | Introduction à useEffect (30 min)

Contenu Théorique

📊 **React_Avancé_Hooks_Architecture_Performance.pdf** - Section "useEffect"

Référence : `cheatsheets/03-hooks-cheatsheet.md`

Points clés :

1. Effets de bord (side effects)
2. Les 3 cas d'utilisation :
   - `useEffect(() => {...})` : chaque render
   - `useEffect(() => {...}, [])` : au montage uniquement
   - `useEffect(() => {...}, [dep])` : quand `dep` change
3. Fonction de cleanup (return)

Démonstration Live

```
function Timer() {
  const [seconds, setSeconds] = useState(0);
```

```
  useEffect(() => {
    const interval = setInterval(() => {
      setSeconds(prev => prev + 1);
    }, 1000);

    // Cleanup : appelé au démontage
    return () => clearInterval(interval);
  }, []); // [] = une seule fois au montage

  return <p>Temps écoulé : {seconds}s</p>;
}
```

🕘 14:30-15:15 | useEffect en Pratique (45 min)

🎯 Exercice Module 2 - Ex01 : Chronomètre

Lancer l'exercice :

> "Ouvrez `exercices/module-2-avance/ex01-chronometre.md`"

Objectif : Maîtriser useEffect et cleanup

Timing :

- 5 min : Lecture
- 30 min : Réalisation
- 10 min : Correction approfondie (importance du cleanup)

🕘 15:15-15:30 | ☕ Pause (15 min)

🕘 15:30-16:30 | Fetch API avec React (1h)

Contenu Théorique (20 min)

Points clés :

1. Fetch dans useEffect
2. Pattern loading/error/data
3. Async/await dans useEffect (fonction interne)
4. Gestion des erreurs

Démonstration Live (10 min)

```
function UserList() {
  const [users, setUsers] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
```

```
  useEffect(() => {
    const fetchUsers = async () => {
      try {
        const res = await fetch('https://jsonplaceholder.typicode.com/users');
        if (!res.ok) throw new Error('Erreur réseau');
        const data = await res.json();
        setUsers(data);
      } catch (err) {
        setError(err.message);
      } finally {
        setLoading(false);
      }
    };

    fetchUsers();
  }, []);

  if (loading) return <p>Chargement...</p>;
  if (error) return <p>Erreur : {error}</p>;

  return (
    <ul>
      {users.map(user => <li key={user.id}>{user.name}</li>)}
    </ul>
  );
}
```

Exercice Module 2 - Ex02 : Fetch Données

Lancer l'exercice :

> "Ouvrez `exercices/module-2-avance/ex02-fetch-donnees.md`"

Timing :

- 5 min : Lecture
- 20 min : Réalisation
- 5 min : Correction

16:30-18:00 | Projet Fil Rouge - Étapes 1-2 (1h30)

Introduction au Projet TeamHub (10 min)

> "Maintenant qu'on a les bases solides, on va construire une vraie application : TeamHub, un mini intranet d'entreprise."

Présenter :

- L'objectif global
- Les 6 étapes
- Les données fournies (`projet-fil-rouge/data/`)

🎯 Étape 1 : Structure de Base (40 min)

Lancer l'étape :

> "Ouvrez `projet-fil-rouge/etapes/etape-1-structure/enonce.md`"

Objectif : Layout responsive avec Header, Footer, Navigation

Timing :

- 5 min : Lecture
- 30 min : Réalisation
- 5 min : Validation

🎯 Étape 2 : Annuaire des Employés (40 min)

Lancer l'étape :

> "Ouvrez `projet-fil-rouge/etapes/etape-2-annuaire/enonce.md`"

Objectif : Afficher la liste des employés avec leurs infos

Timing :

- 5 min : Lecture
- 30 min : Réalisation
- 5 min : Validation

🕘 18:00 | Fin du Jour 2

Récapitulatif

- Formulaires contrôlés et validation
- Todo App : synthèse Module 1
- useEffect et cycle de vie
- Fetch API avec états loading/error
- Début du projet TeamHub

JOUR 3 : React Avancé

🕘 09:00-09:15 | Révisions Jour 2 (15 min)

Quiz Rapide

1. Quand utiliser `[]` comme dépendances de useEffect ?
2. Pourquoi le cleanup est important ?
3. Comment gérer le loading pendant un fetch ?

🕘 09:15-10:15 | useEffect Avancé (1h)

🎯 Exercice Module 2 - Ex03 : Recherche Temps Réel

Lancer l'exercice :

> "Ouvrez `exercices/module-2-avance/ex03-recherche-temps-reel.md`"

Concepts :

- Dépendances de useEffect
- Debounce (bonus)
- AbortController pour annuler les requêtes

Timing :

- 10 min : Explication du debounce
- 5 min : Lecture
- 35 min : Réalisation
- 10 min : Correction

🕘 10:15-10:30 | ☕ Pause (15 min)

🕘 10:30-11:15 | Pattern Children et Composition (45 min)

Contenu Théorique (15 min)

Points clés :

1. `props.children` pour la composition
2. Render props (mention)
3. Compound components (mention)

🎯 Exercice Module 2 - Ex04 : Modal Réutilisable

Lancer l'exercice :

> "Ouvrez `exercices/module-2-avance/ex04-modal.md`"

Timing :

- 5 min : Lecture
- 20 min : Réalisation
- 5 min : Correction

🕘 11:15-12:30 | useReducer (1h15)

Contenu Théorique (25 min)

React_Avancé_Hooks_Architecture_Performance.pdf - Section "useReducer"

Points clés :

1. Quand préférer useReducer à useState
2. Pattern : `(state, action) => newState`
3. Actions avec type et payload
4. Immutabilité

Démonstration Live (10 min)

```
const initialState = { count: 0 };

function reducer(state, action) {
  switch (action.type) {
    case 'INCREMENT':
      return { count: state.count + 1 };
    case 'DECREMENT':
      return { count: state.count - 1 };
    case 'RESET':
      return initialState;
    case 'SET':
      return { count: action.payload };
    default:
      return state;
  }
}

function Counter() {
  const [state, dispatch] = useReducer(reducer, initialState);

  return (
    <div>
      <p>Count: {state.count}</p>
      <button onClick={() => dispatch({ type: 'INCREMENT' })}>+</button>
      <button onClick={() => dispatch({ type: 'DECREMENT' })}>-</button>
      <button onClick={() => dispatch({ type: 'SET', payload: 10 })}>Set
10</button>
    </div>
  );
}
```

Exercice Module 2 - Ex05 : Panier useReducer

Lancer l'exercice :

> "Ouvrez `exercices/module-2-avance/ex05-panier-reducer.md`"

Timing :

- 5 min : Lecture

- 30 min : Réalisation
- 5 min : Correction

🕘 12:30-14:00 | 🍽 Pause Déjeuner (1h30)

🕑 14:00-15:00 | Custom Hooks (1h)

Contenu Théorique (15 min)

Points clés :

1. Extraire la logique réutilisable
2. Convention : préfixe `use`
3. Composition de hooks
4. Exemples courants : useLocalStorage, useFetch, useDebounce

Démonstration Live (10 min)

```
// hooks/useLocalStorage.js
function useLocalStorage(key, initialValue) {
  const [value, setValue] = useState(() => {
    const stored = localStorage.getItem(key);
    return stored ? JSON.parse(stored) : initialValue;
  });

  useEffect(() => {
    localStorage.setItem(key, JSON.stringify(value));
  }, [key, value]);

  return [value, setValue];
}

// Utilisation
function App() {
  const [theme, setTheme] = useLocalStorage('theme', 'light');
  // ...
}
```

🎯 Exercice Module 2 - Ex06 : useLocalStorage Hook

Lancer l'exercice :

> "Ouvrez `exercices/module-2-avance/ex06-use-local-storage.md`"

Timing :

- 5 min : Lecture
- 25 min : Réalisation

- 5 min : Correction

🕘 15:00-15:15 | ☕ Pause (15 min)

🕘 15:15-16:30 | Projet TeamHub - Étape 3 (1h15)

🎯 Étape 3 : Recherche et Filtres

Lancer l'étape :

> "Ouvrez `projet-fil-rouge/etapes/etape-3-recherche-filtres/enonce.md`"

Objectif : Ajouter recherche temps réel et filtres par département

Fonctionnalités :

- SearchBar avec état contrôlé
- DepartmentFilter (select ou boutons)
- Filtrage combiné

Timing :

- 10 min : Lecture et questions
- 55 min : Réalisation
- 10 min : Revue de code

🕘 16:30-18:00 | Projet TeamHub - Étape 4 (1h30)

🎯 Étape 4 : Système d'Annonces

Lancer l'étape :

> "Ouvrez `projet-fil-rouge/etapes/etape-4-annonces/enonce.md`"

Objectif : Créer un système d'annonces avec fetch simulé

Fonctionnalités :

- AnnouncementList avec loading/error
- AnnouncementCard avec catégories
- AnnouncementForm pour ajouter
- Tri par date, annonces épinglées

Timing :

- 10 min : Lecture
- 70 min : Réalisation
- 10 min : Validation

🕘 18:00 | Fin du Jour 3

Récapitulatif

- useEffect avancé (debounce, abort)
- Composition avec children
- useReducer pour état complexe
- Custom hooks
- TeamHub : recherche, filtres, annonces

JOUR 4 : Architecture et Finalisation

🕘 09:00-09:15 | Révisions Jour 3 (15 min)

Questions/Réponses

Clarifier les points de blocage éventuels

🕘 09:15-10:15 | Optimisation et Performance (1h)

Contenu Théorique (20 min)

📊 **React_Avancé_Hooks_Architecture_Performance.pdf** - Section "Performance"

Points clés :

1. React.memo : mémoiser un composant
2. useCallback : mémoiser une fonction
3. useMemo : mémoiser une valeur calculée
4. Quand optimiser (et quand ne pas)

🎯 Exercice Module 2 - Ex08 : Optimisation Memo

Lancer l'exercice :

> "Ouvrez `exercices/module-2-avance/ex08-optimisation-memo.md`"

Timing :

- 5 min : Lecture
- 25 min : Réalisation
- 10 min : Discussion sur les cas d'usage

🕘 10:15-10:30 | ☕ Pause (15 min)

🕘 10:30-11:15 | Liste Infinie (45 min)

🎯 Exercice Module 2 - Ex07 : Liste Infinie

Lancer l'exercice :

> "Ouvrez `exercices/module-2-avance/ex07-liste-infinie.md`"

Concepts :

- IntersectionObserver
- Pagination côté client
- useRef pour les références DOM

Timing :

- 5 min : Explication IntersectionObserver
- 5 min : Lecture
- 30 min : Réalisation
- 5 min : Correction

🕘 11:15-12:30 | Context API (1h15)

Contenu Théorique (25 min)

📊 **React_Avancé_Hooks_Architecture_Performance.pdf** - Section "Context"

Points clés :

1. Problème du prop drilling
2. createContext, Provider, useContext
3. Pattern : Context + Custom Hook
4. Quand utiliser Context vs Props vs État global

Démonstration Live (15 min)

```
// contexts/ThemeContext.jsx
const ThemeContext = createContext();

export function ThemeProvider({ children }) {
  const [theme, setTheme] = useState('light');

  const toggleTheme = () => {
    setTheme(prev => prev === 'light' ? 'dark' : 'light');
  };

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      {children}
    </ThemeContext.Provider>
  );
}
```

```
export function useTheme() {
  const context = useContext(ThemeContext);
  if (!context) {
    throw new Error('useTheme must be used within ThemeProvider');
  }
  return context;
}
```

🎯 Exercice Module 2 - Ex09 : Mini App Context

Lancer l'exercice :

> "Ouvrez `exercices/module-2-avance/ex09-context-api.md`"

Timing :

- 5 min : Lecture
- 25 min : Réalisation
- 5 min : Correction

🕙 12:30-14:00 | 🍽 Pause Déjeuner (1h30)

🕑 14:00-15:30 | Projet TeamHub - Étape 5 (1h30)

🎯 Étape 5 : Profil Utilisateur avec useReducer

Lancer l'étape :

> "Ouvrez `projet-fil-rouge/etapes/etape-5-profil-reducer/enonce.md`"

Objectif : Page de profil éditable avec useReducer

Actions du reducer :

- SET_USER
- START_EDIT
- CANCEL_EDIT
- UPDATE_FIELD
- SAVE
- SET_ERROR

Timing :

- 10 min : Lecture et conception du reducer
- 70 min : Réalisation
- 10 min : Validation

🕒 15:30-15:45 | ☕ Pause (15 min)

🕘 15:45-17:15 | Projet TeamHub - Étape 6 (1h30)

🎯 Étape 6 : Context API et Custom Hooks

Lancer l'étape :

> "Ouvrez `projet-fil-rouge/etapes/etape-6-context-hooks/enonce.md`"

Objectif : Finaliser avec état global et persistance

Fonctionnalités :

- UserContext : utilisateur courant, rôles
- ThemeContext : thème clair/sombre
- useLocalStorage : persistance
- Permissions : formulaire admin-only

Timing :

- 10 min : Lecture
- 70 min : Réalisation
- 10 min : Validation finale

🕘 17:15-17:45 | Synthèse Finale (30 min)

🎯 Exercice Module 2 - Ex10 : Synthèse Complète (optionnel)

> "Si le temps le permet, vous pouvez tenter l'exercice de synthèse finale."

Alternative : Revue de code du projet TeamHub

Récapitulatif Global

Checklist des compétences acquises (voir `guides/guide-formateur.md`)

🕘 17:45-18:00 | Clôture (15 min)

Pour Aller Plus Loin

- React Router pour le routing
- React Query / TanStack Query pour le data fetching
- Zustand ou Redux pour l'état global complexe
- TypeScript avec React
- Testing avec Vitest et React Testing Library

Ressources

- Documentation officielle : react.dev
- Les cheatsheets de la formation

- Les exercices "Pour aller plus loin"

Feedback

- Tour de table : ce qui a été le plus utile
- Points à améliorer

Certificat / Attestation

Distribution si applicable

Annexes

Checklist Matériel Formateur

Avant la Formation

- ☐ Vérifier que tous les PDFs s'ouvrent correctement
- ☐ Préparer un projet Vite vierge pour les démos
- ☐ Tester tous les exercices
- ☐ Préparer les données de TeamHub
- ☐ Vérifier la connexion internet (pour les APIs)

À Chaque Jour

- ☐ Projeter les slides
- ☐ Avoir les cheatsheets accessibles
- ☐ Ouvrir VS Code avec le bon workspace
- ☐ Lancer le serveur de développement

Timing Récapitulatif

Jour	Exercices	Projet	Théorie
J1	ex01-07	-	~2h30
J2	ex08-10 + m2/ex01-02	Étapes 1-2	~1h30
J3	m2/ex03-06	Étapes 3-4	~1h
J4	m2/ex07-09 (+ex10)	Étapes 5-6	~1h30

Adaptations Possibles

Si le Groupe est Rapide

- Ajouter les sections "Pour aller plus loin"
- Faire l'exercice 10 du module 2
- Ajouter des fonctionnalités à TeamHub

Si le Groupe est Plus Lent

- Réduire les exercices (garder ex01, 03, 05, 09, 10 du module 1)
- Simplifier les étapes du projet
- Donner plus de code de départ

Format Condensé (3 jours)

- Jour 1 : Module 1 complet
- Jour 2 : Module 2 (ex01-06) + Étapes 1-3
- Jour 3 : Module 2 (ex07-09) + Étapes 4-6