

Atelier 02.08 Hooks

📁 Structure du dossier

```
src/
└── Hooks/
    Counter.jsx
    FormUser.jsx
    FavoriteColor.jsx
    Timer.jsx
    Users.jsx
    InputFocus.jsx
    PreviousValue.jsx
    ThemeContext.jsx
    ThemeProvider.jsx
    ThemeButton.jsx
    useLocalStorage.js
    TodoList.jsx
```

Exercice 1 : useState — Compteur

Objectif

Comprendre comment gérer un état simple et déclencher une mise à jour.

Instructions

- Créez le fichier : src/Hooks/Counter.jsx
- Importez useState
- Affichez une valeur count
- Ajoutez deux boutons : Incrémente / Décrémenter

Correction — Counter.jsx

```
// src/Hooks/Counter.jsx
import { useState } from "react";

function Counter() {
    // Déclare l'état count avec une valeur initiale 0
    const [count, setCount] = useState(0);

    return (
        <div>
            <h2>Compteur : {count}</h2>
            {/* Incrémente */}
            <button onClick={() => setCount(count + 1)}>+1</button>
            {/* Décrémente */}
            <button onClick={() => setCount(count - 1)}>-1</button>
        </div>
    );
}
```

```

    <button onClick={() => setCount(count - 1)}>-1</button>
  </div>
);
}

export default Counter;

```

Exercice 2 : useState — Formulaire contrôlé**Objectif**

Saisir des données dans des inputs et les stocker dans l'état.

Instructions

1. Créez FormUser.jsx
2. Dans le composant :
 - o Créez un état form avec { nom: "", email: "" }
 - o Utilisez name des inputs pour mettre à jour dynamiquement l'état

Kamel ABBASSI

abbassi.kamel@gmail.com

Nom : Kamel ABBASSI

Email : abbassi.kamel@gmail.com

Correction — FormUser.jsx

```

// src/Hooks/FormUser.jsx
import { useState } from "react";

function FormUser() {
  const [form, setForm] = useState({ nom: "", email: "" });

  const handleChange = (e) => {
    // Mise à jour dynamique selon l'attribut name
    setForm({ ...form, [e.target.name]: e.target.value });
  };

  return (
    <div>
      <input name="nom" placeholder="Nom" onChange={handleChange} />
      <input name="email" placeholder="Email" onChange={handleChange} />

      <p>Nom : {form.nom}</p>
      <p>Email : {form.email}</p>
    </div>
  );
}

export default FormUser;

```

Exercice 3 : useState — Mise à jour d'un objet**Objectif**

Modifier une seule propriété d'un objet sans écraser les autres.

Instructions

1. Créez FavoriteColor.jsx
2. Déclarez un objet car
3. Ajoutez un bouton qui change **uniquement** la couleur

Marque : Toyota

Modèle : Corolla

Couleur : Bleu

Changer couleur

Correction — FavoriteColor.jsx

```
// src/Hooks/FavoriteColor.jsx
import { useState } from "react";

function FavoriteColor() {
  const [car, setCar] = useState({
    brand: "Toyota",
    model: "Corolla",
    color: "Rouge"
  });

  const changeColor = () => {
    setCar(prev => ({ ...prev, color: "Bleu" }));
  };

  return (
    <div>
      <p>Marque : {car.brand}</p>
      <p>Modèle : {car.model}</p>
      <p>Couleur : {car.color}</p>
      <button onClick={changeColor}>Changer couleur</button>
    </div>
  );
}

export default FavoriteColor;
```

Exercice 4 : useEffect — Timer + Nettoyage**Objectif**

Comprendre l'exécution de useEffect et la fonction de nettoyage.

Instructions

1. Créez Timer.jsx
2. Déclarez un état seconds = 0
3. Utilisez setInterval pour l'incrémenter chaque seconde
4. Nettoyez l'intervalle avec return () => clearInterval()

Temps écoulé : 68s

Correction — Timer.jsx

```
// src/Hooks/Timer.jsx
import { useState, useEffect } from "react";

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

  useEffect(() => {
    const interval = setInterval(() => {
      // Forme fonctionnelle = évite les problèmes d'état désynchronisé
      setSeconds(prev => prev + 1);
    }, 1000);

    // Nettoyage : éviter fuite mémoire
    return () => clearInterval(interval);
  }, []); // [] = exécuter seulement au montage

  return <h2>Temps écoulé : {seconds}s</h2>;
}

export default Timer;
```

Exercice 5 : useEffect — Appel API

Instructions

- Créez Users.jsx
- Utilisez fetch dans useEffect

- Leanne Graham
- Ervin Howell
- Clementine Bauch
- Patricia Lebsack
- Chelsey Dietrich
- Mrs. Dennis Schulist
- Kurtis Weissnat
- Nicholas Runolfsdottir V
- Glenna Reichert
- Clementina DuBuque

Correction — Users.jsx

```
// src/Hooks/Users.jsx
import { useState, useEffect } from "react";

function Users() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    fetch("https://jsonplaceholder.typicode.com/users")
      .then(r => r.json())
      .then(setUsers);
  }, []);

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

export default Users;
```

Exercice 6 : useRef — Focus automatique**Correction — InputFocus.jsx**

```
// src/Hooks/InputFocus.jsx
import { useRef } from "react";

function InputFocus() {
  const inputRef = useRef(null);

  return (
    <div>
      <input ref={inputRef} />
      <button onClick={() => inputRef.current.focus()}>Focus</button>
    </div>
  );
}

export default InputFocus;
```

Exercice 7 : useRef + useEffect — Suivi valeur précédente**Correction — PreviousValue.jsx**

```
// src/Hooks/PreviousValue.jsx
import { useState, useEffect, useRef } from "react";

function PreviousValue() {
  const [value, setValue] = useState("");
  const prev = useRef("");

  useEffect(() => {
    prev.current = value;
  }, [value]);

  return (
    <div>
      <input onChange={(e) => setValue(e.target.value)} />
      <p>Actuelle : {value}</p>
      <p>Précédente : {prev.current}</p>
    </div>
  );
}

export default PreviousValue;
```

Exercice 8 : Custom Hook — LocalStorage (Détailé)**Objectif**

Créer un Hook réutilisable pour stocker des données permanentes.

Correction — useLocalStorage.js

```
// src/Hooks/useLocalStorage.js
import { useState } from "react";

export default function useLocalStorage(key, initialValue) {
  const [value, setValue] = useState(() => {
    const saved = localStorage.getItem(key);
    return saved ? JSON.parse(saved) : initialValue;
  });

  const updateValue = (newValue) => {
    setValue(newValue);
    localStorage.setItem(key, JSON.stringify(newValue));
  };
}
```

```

    return [value, updateValue];
}

```

Exemple d'utilisation — TodoList.jsx

```

// src/Hooks/TodoList.jsx
import { useState } from "react";
import useLocalStorage from "./useLocalStorage";

function TodoList() {
  const [todos, setTodos] = useLocalStorage("todos", []);
  const [text, setText] = useState("");

  const addTodo = () => {
    setTodos([...todos, text]);
    setText("");
  };

  return (
    <div>
      <input value={text} onChange={e => setText(e.target.value)} />
      <button onClick={addTodo}>Ajouter</button>

      <ul>
        {todos.map((t, i) => <li key={i}>{t}</li>)}
      </ul>
    </div>
  );
}

export default TodoList;

```

NewsTech

On ajoute un nouvel exercice complet sur :

- useEffect() pour appeler une API
- Afficher des actualités (nouvelles technologies)
- Ajouter du CSS pour rendre l'affichage propre
- Tout se met dans src/Hooks/NewsTech.jsx

Nous allons utiliser l'API gratuite **dev.to** (pas besoin de clé API).

Endpoint :

https://dev.to/api/articles?tag=javascript&per_page=6

Tu peux changer le tag (react, ai, python, etc.)

- Exercice — Récupérer et Afficher des Actualités Tech depuis une API

Objectif

- Apprendre à utiliser useEffect() pour faire un appel API.
- Apprendre à utiliser useState() pour stocker les données récupérées.

- Créer un affichage listé des articles.
- Ajouter du style pour une présentation agréable.

Instructions

1. Dans le dossier src/Hooks/, créer un fichier NewsTech.jsx.
2. Créer un composant fonctionnel NewsTech().
3. Déclarer un état articles sous forme de tableau vide.
4. Utiliser useEffect() pour faire un fetch() vers l'API dev.to.
5. Mettre à jour articles avec les données reçues.
6. Afficher chaque article avec :
 - Titre
 - Auteur
 - Image illustrative
 - Lien "Lire l'article"
7. Créer un fichier CSS pour styliser l'affichage.
8. Importer le composant dans App.jsx et le tester.

Correction

```
// src/Hooks/NewsTech.jsx
import { useEffect, useState } from "react";
import "./NewsTech.css"; // On importe la feuille de style

export default function NewsTech() {
  // articles stockera les données de l'API
  const [articles, setArticles] = useState([]);

  // useEffect : appelé après le premier rendu
  useEffect(() => {
    fetch("https://dev.to/api/articles?tag=react&per_page=6")
      .then(response => response.json())
      .then(data => {
        setArticles(data); // On stocke les articles dans l'état
      })
      .catch(error => console.error("Erreur API :", error));
  }, []); // Tableau vide => l'appel se fait seulement au chargement

  return (
    <div className="news-container">
      <h2>Actualités Tech (via dev.to)</h2>
      <p>Voici les dernières actualités dans le domaine des technologies.</p>

      <div className="news-grid">
        {articles.map((article) => (
          <div className="news-card" key={article.id}>
            {/* Affiche l'image seulement si elle existe */}
            {article.cover_image && (
              <img src={article.cover_image} alt="cover" className="news-image" />
            )}
            <h3>{article.title}</h3>
            <p>◆ Auteur : {article.user.name}</p>
            <p>◆ Publié le : {new Date(article.published_at).toLocaleString()}</p>
            <a href={article.url} target="_blank" rel="noopener noreferrer">
```

```
        Lire l'article
      </a>
    </div>
  )}
</div>
</div>
);
}
```

src/Hooks/NewsTech.css

```
/* src/Hooks/NewsTech.css */

.news-container {
  padding: 20px;
  font-family: Arial, sans-serif;
}

.news-container h2 {
  margin-bottom: 10px;
}

.news-grid {
  margin-top: 20px;
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(260px, 1fr));
  gap: 20px;
}

.news-card {
  border: 1px solid #ddd;
  border-radius: 8px;
  background: white;
  padding: 15px;
  transition: transform 0.2s;
}

.news-card:hover {
  transform: scale(1.05);
  border-color: #007bff;
}

.news-card img {
  width: 100%;
  height: 150px;
  object-fit: cover;
  border-radius: 6px;
}

.news-card h3 {
  font-size: 1.1rem;
  margin: 10px 0;
}

.news-card a {
  display: inline-block;
  margin-top: 8px;
  text-decoration: none;
  color: #007bff;
  font-weight: bold;
}

.news-card a:hover {
  text-decoration: underline;
}
```

main.jsx (pour tester)

Ajoute la ligne :

```
import NewsTech from "./Hooks/NewsTech";
// ...
<NewsTech />
```