

Atelier 02.04 Conditions

Exercice 1 — If Statement Basique

Objectif :

Comprendre comment utiliser la condition if pour afficher un contenu différent.

Instructions :

Créer le composant « WelcomeMessage » qui affiche :

- “Bienvenue 🙋” si `isLoggedIn = true`
- “Veuillez-vous connecter” sinon.

Correction

```
// Fichier : src/components/WelcomeMessage.jsx
export default function WelcomeMessage() {
  const isLoggedIn = true;
  if (isLoggedIn) {
    return (<h1>Bienvenue 🙋</h1>);
  }
  else {
    return (<h1>Veuillez vous connecter</h1>);
  }
}
```

Exercice 2 — Opérateur logique &&

Objectif :

Afficher un texte seulement si une condition est vraie (sans else).

Instructions :

Afficher “Vous avez un nouveau message 📧” seulement si `hasMessage = true`.

Correction

```
//src/components/NewMessageNotification.jsx
function NewMessageNotification() {
  const hasMessage = true;

  return (
    <div>
      {hasMessage && <p>Vous avez un nouveau message 📧</p>}
    </div>
  );
}

export default NewMessageNotification;
```

Exercice 3 — Opérateur Ternaire

Objectif :

Savoir choisir entre deux éléments dans le JSX avec l’opérateur ternaire.

Instructions :

Afficher :

- “Mode sombre activé 🌙” si `darkMode = true`

- “Mode clair ☀️” sinon.

Correction

```
//src/components/DarkModeStatus.jsx
function DarkModeStatus() {
  const darkMode = false;

  return (
    <h3>{darkMode ? "Mode sombre activé 🌙" : "Mode clair ☀️"}</h3>
  );
}

export default DarkModeStatus;
```

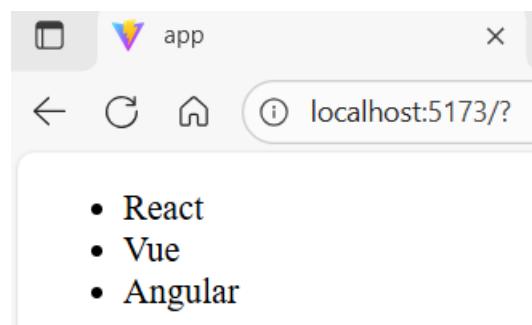
Exercice 4 — Iteration avec map() + key

Objectif :

Afficher une liste d’éléments correctement avec key.

Instructions :

Afficher la liste : ["React", "Vue", "Angular"] dans une .



Correction

```
//src/components/FrameworkList.jsx
function FrameworkList() {
  const frameworks = [
    { id: 1, name: "React" },
    { id: 2, name: "Vue" },
    { id: 3, name: "Angular" }
  ];

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

export default FrameworkList;
```

Exercice 5 — Combiner Condition + map()

Objectif :

N’afficher la liste que si elle n’est pas vide.

Instructions :

Créer un composant SafeList

Si la liste students est vide → afficher “Aucun élément”.

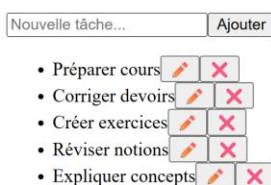
Sinon → afficher la liste.

Correction

```
//src/components/SafeList.jsx
function SafeList() {
  const students = [
    { id: 1, name: "Kamel" },
    { id: 2, name: "Sami" },
    { id: 5, name: "Eve" },
    { id: 6, name: "Lina" }
  ];
  return (
    students.length === 0 ?
      <p>No students available.</p>
      :
      <ul>
        {students.map((student) => (
          <li key={student.id}>{student.name}</li>
        )))
      </ul>
  );
}
export default SafeList;
```

Projet Final — ToDo List**Objectifs :**

- Gérer un tableau d'objets (state)
- Ajouter une tâche
- Supprimer une tâche
- Modifier une tâche
- Styliser avec CSS

 **Liste des Tâches**
**Etape 1 : Affichage**

Étapes à suivre :

1. Créer le fichier du composant
 - Dans le dossier src/components, crée un fichier nommé : TodoApp.jsx
 - Initialise state nommée **tasks** par deux tasks.

```
{ id: 1, title: "Learn React", completed: false },
{ id: 2, title: "Build a Todo App", completed: false },
{ id: 3, title: "Profit!", completed: true }
```

- Afficher les tâches dans une liste

```
import { useState } from "react";
import "./Todo.css";

export default function TodoApp(){
  const [tasks]=useState([
    { id: 1, title: "Learn React", completed: false },
    { id: 2, title: "Build a Todo App", completed: false },
    { id: 3, title: "Profit!", completed: true }
  ]);
  return (
    <div>
      <h1>Todo Application</h1>
      {tasks.length === 0 ? (
        <p>No tasks available.</p>
      ) : (
        <ul>
          {tasks.map((task) => (
            <li key={task.id}>
              {task.title}
            </li>
          )))
        </ul>
      )}
    </div>
  );
}
```

2. Importer et afficher le composant

- Ouvre ton fichier main.jsx et ajoute :
- import TodoApp from "./components/TodoApp";
- Dans le return, insère :
- <TodoApp />

3. Ajouter les deux boutons d'actions

```
import { useState } from "react";
import "./Todo.css";

export default function TodoApp() {
  const [tasks] = useState([
    { id: 1, title: "Learn React", completed: false },
    { id: 2, title: "Build a Todo App", completed: false },
    { id: 3, title: "Profit!", completed: true }
  ]);
  return (
    <div>
      <h1>Todo Application</h1>
      {tasks.length === 0 ? (
        <p>No tasks available.</p>
      ) : (
        <ul>
          {tasks.map((task) => (
            <li key={task.id}>
              {task.title}
              <div className="button-group">
                <button onClick={null}></button>
                <button onClick={null}></button>
              </div>
            </li>
          )))
        </ul>
      )}
    </div>
  );
}
```

```
        <button onClick={null}>X</button>
      </div>
    </li>
  ))}
</ul>
)
</div>
);
}
```

4. Dans le même dossier, crée un fichier : Todo.css

```
/* src/components/Todo.css */
ul {
  list-style: none;
  padding: 0;
}

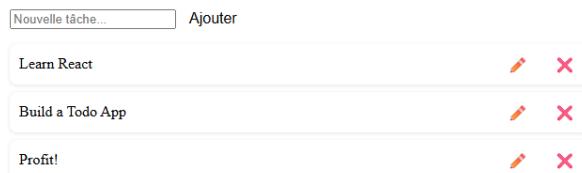
li {
  display: flex;
  justify-content: space-between;
  align-items: center;
  background: #fff;
  margin: 8px 0;
  padding: 10px;
  border-radius: 8px;
  box-shadow: 0 1px 4px rgba(0, 0, 0, 0.1);
  transition: transform 0.2s ease, background 0.2s ease;
}

li:hover {
  background: #e3f2fd;
  transform: scale(1.02);
}
button {
  background: none;
  border: none;
  cursor: pointer;
  font-size: 16px;
  margin-left: 8px;
  transition: color 0.2s ease;
}
button:hover {
  color: #1976d2;
}
.button-group {
  display: flex;
  gap: 8px;
}
input {
  flex: 1;
  padding: 10px 12px;
  border: 1px solid #ccc;
  border-radius: 8px 0 0 8px;
  font-size: 16px;
  outline: none;
  transition: border 0.2s ease, box-shadow 0.2s ease;
}
```

Etape 2 : Ajouter des tâches

Ajouter un champs input de type de texte et un bouton « ajouter »

Todo Application



```
import { useState } from "react";
import "./Todo.css";

export default function TodoApp() {
  const [tasks] = useState([
    { id: 1, title: "Learn React", completed: false },
    { id: 2, title: "Build a Todo App", completed: false },
    { id: 3, title: "Profit!", completed: true }
  ]);
  return (
    <div>
      <h1>Todo Application</h1>
      <input
        type="text"
        placeholder="Nouvelle tâche..."
      />
      <button onClick={null}>Ajouter</button>

      {tasks.length === 0 ? (
        <p>No tasks available.</p>
      ) : (
        <ul>
          {tasks.map((task) => (
            <li key={task.id}>
              {task.title}
              <div className="button-group">
                <button onClick={null}>Ø</button>
                <button onClick={null}>X</button>
              </div>
            </li>
          ))}
        </ul>
      )}
    </div>
  );
}
```

Etape 3 : Ajouter l'action d'ajout d'une nouvelle tâche

- Ajouter state nommée **input**, elle va récupérer la valeur d'input à chaque modification, et à cette state associer la fonction **setInput**

```
const [input, setInput] = useState("");

.....
<input type="text"
placeholder="Nouvelle tâche..."
onChange={(e)=> setInput(e.target.value)}
/>
```

- Associer à la state tasks , la fonction setTasks
- Ajouter la fonction **addTask** que sera appelée lorsque l'utilisateur lance sur OnClick. Cette fonction affiche la valeur d'input dans la console, puis vider input

```
function addTask() {
  console.log("Ajouter la tâche :", input);
```

```

        setInput("");
    }
.....
<input
value={input}
type="text"
placeholder="Nouvelle tâche..."
onChange={(e)=> setInput(e.target.value)}
/>

```

4. Dans la fonction `addTasks()`, ajouter code pour conserver les anciennes tâches dans `tasks`, et ajouter la nouvelle.

```
setTasks([...tasks,{ id: tasks.length + 1, title: input, completed: false }]);
```

code final

```

import { useState } from "react";
import "./Todo.css";

export default function TodoApp() {
  const [tasks, setTasks] = useState([
    { id: 1, title: "Learn React", completed: false },
    { id: 2, title: "Build a Todo App", completed: false },
    { id: 3, title: "Profit!", completed: true }
  ]);

  const [input, setInput] = useState("");

  function addTask() {
    if (input.trim() === "") return;

    console.log("Ajouter la tâche :", input);
    setTasks([...tasks,{ id: tasks.length + 1, title: input, completed: false }]);
    setInput("");
  }

  return (
    <div>
      <h1>Todo Application</h1>
      <input
        value={input}
        type="text"
        placeholder="Nouvelle tâche..."
        onChange={(e)=> setInput(e.target.value)}
      />
      <button onClick={addTask}>+</button>

      {tasks.length === 0 ? (
        <p>No tasks available.</p>
      ) : (
        <ul>
          {tasks.map((task) => (
            <li key={task.id}>
              {task.title}
              <div className="button-group">
                <button onClick={null}>Ø</button>
                <button onClick={null}>X</button>
              </div>
            </li>
          )));
        </ul>
      )}
    </div>
  );
}

```

Etape 3 : Edit

5. Ajouter une function editTask(id) , prend l'id de la tache a modifier, et affiche prompt pour taper la nouvelle valeur de cette tâche.

Dans le bouton de la modification, utilise ce code

```
<button onClick={()=>editTask(task.id)}>✎</button>
```

Ajouter cette fonction et observer la console

```
function editTask(id) {
  console.log("Modifier la tâche avec l'id :", id);
}
```

Modifier editTask et tester la modification d'une tache.

```
function editTask(id) {
  const nouvelValeur = prompt("Entrez la nouvelle valeur de la tâche :");
  if(!nouvelValeur) return;
  if (nouvelValeur.trim() === "") return;
  setTasks(
    tasks.map((task) =>
      task.id === id ? { ...task, title: nouvelValeur } : task
    )
  );
  console.log("Modifier la tâche avec l'id :", id);
}
```

Dans le prompt, on veut afficher l'ancien titre de la tache

Utiliser ce code

```
<button onClick={() => editTask(task.id,task.title)}>✎</button>

function editTask(id, currentTitle) {
  const nouvelValeur = prompt("Entrez la nouvelle valeur de la tâche :",currentTitle);
  if(!nouvelValeur) return;
  if (nouvelValeur.trim() === "") return;
  setTasks(
    tasks.map((task) =>
      task.id === id ? { ...task, title: nouvelValeur } : task
    )
  );
  console.log("Modifier la tâche avec l'id :", id);
}
```

Code final

```
import { useState } from "react";
import "./Todo.css";

export default function TodoApp() {
  const [tasks, setTasks] = useState([
    { id: 1, title: "Learn React", completed: false },
    { id: 2, title: "Build a Todo App", completed: false },
    { id: 3, title: "Profit!", completed: true }
  ]);
}
```

```

const [input, setInput] = useState("");
function addTask() {
    if (input.trim() === "") return;
    console.log("Ajouter la tâche :", input);
    setTasks([...tasks, { id: tasks.length + 1, title: input, completed: false }]);
    setInput("");
}

function editTask(id, currentTitle) {
    const nouvelleValeur = prompt("Entrez la nouvelle valeur de la tâche :", currentTitle);
    if (!nouvelleValeur) return;
    if (nouvelleValeur.trim() === "") return;
    setTasks(
        tasks.map(task =>
            task.id === id ? { ...task, title: nouvelleValeur } : task
        )
    );
    console.log("Modifier la tâche avec l'id :", id);
}

return (
    <div>
        <h1>Todo Application</h1>
        <input
            value={input}
            type="text"
            placeholder="Nouvelle tâche..."
            onChange={(e) => setInput(e.target.value)}
        />
        <button onClick={addTask}>+</button>

        {tasks.length === 0 ? (
            <p>No tasks available.</p>
        ) : (
            <ul>
                {tasks.map((task) => (
                    <li key={task.id}>
                        {task.title}
                        <div className="button-group">
                            <button onClick={() => editTask(task.id, task.title)}>✎</button>
                            <button onClick={() => null}>X</button>
                        </div>
                    </li>
                ))}
            </ul>
        )}
    </div>
)
}

```

Etape 3 : delete

Utilise la fonction filter() dans la fonction deleteTask

filtre crée **un nouveau tableau** contenant **uniquement** les éléments qui respectent une condition.

```

const deleteTask = (id) => {
    setTasks(tasks.filter((task) => task.id !== id));
};

```

Code final

```
import { useState } from "react";
import "./Todo.css";

export default function TodoApp() {
  const [tasks, setTasks] = useState([
    { id: 1, title: "Learn React", completed: false },
    { id: 2, title: "Build a Todo App", completed: false },
    { id: 3, title: "Profit!", completed: true }
  ]);

  const [input, setInput] = useState("");

  function addTask() {

    if (input.trim() === "") return;
    console.log("Ajouter la tâche :", input);
    setTasks([...tasks, { id: tasks.length + 1, title: input, completed: false }]);
    setInput("");
  }

  function editTask(id, currentTitle) {
    const nouvelleValeur = prompt("Entrez la nouvelle valeur de la tâche :", currentTitle);
    if (nouvelleValeur.trim() === "") return;
    setTasks(
      tasks.map((task) =>
        task.id === id ? { ...task, title: nouvelleValeur } : task
      )
    );
    console.log("Modifier la tâche avec l'id :", id);
  }

  const deleteTask = (id) => {
    setTasks(tasks.filter((task) => task.id !== id));
  };

  return (
    <div>
      <h1>Todo Application</h1>
      <input
        value={input}
        type="text"
        placeholder="Nouvelle tâche..."
        onChange={(e) => setInput(e.target.value)}
      />
      <button onClick={addTask}>+</button>

      {tasks.length === 0 ? (
        <p>No tasks available.</p>
      ) : (
        <ul>
          {tasks.map((task) => (
            <li key={task.id}>
              {task.title}
              <div className="button-group">
                <button onClick={() => editTask(task.id, task.title)}>✎</button>
                <button onClick={() => deleteTask(task.id)}>✖</button>
              </div>
            </li>
          ))}
        </ul>
      );
    </div>
  );
}
```