React – Hooks, Événements et Rendering Conditionnel

1) Hook useState

Définition : useState est un Hook qui permet d'ajouter un état local dans un composant fonctionnel.
Cas d'usage : compteur, formulaire, affichage conditionnel, suivi d'état (loading, auth…).
Exemple :

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

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

  return (
    <div>
      <h3>Compteur : {count}</h3>
      <button onClick={() => setCount(c => c + 1)}>+1</button>
      <button onClick={() => setCount(0)}>Remettre à zéro</button>
    </div>
  );
}
```

2) Gestion des événements (onClick, onChange…)

Définition : React attache des gestionnaires d'événements avec des props camelCase (ex: onClick).
Cas d'usage : boutons, formulaires, raccourcis clavier, survols.
Exemple :

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

function FormulaireSimple() {
  const [texte, setTexte] = useState('');

  function handleChange(e) {
    setTexte(e.target.value);
  }

  function handleClick() {
    alert('Tu as saisi : ' + texte);
  }

  return (
    <div>
      <input value={texte} onChange={handleChange} placeholder="Tape quelque chose" />
      <button onClick={handleClick}>Afficher</button>
    </div>
  );
}
```

3) Rendering conditionnel

Définition : Afficher/masquer du JSX selon une condition (if, ternaire, &&).
Cas d'usage : login/logout, loaders, messages d'erreur, toggle menus.
Exemple :

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

function ToggleMessage() {
  const [visible, setVisible] = useState(false);

  return (
    <div>
      <button onClick={() => setVisible(v => !v)}>
        {visible ? 'Cacher' : 'Afficher'} le message
      </button>

      {visible ? (
        <p>Voici le message affiché (via ternaire).</p>
      ) : (
        <p>(Rien à voir ici)</p>
      )}

      {visible && <p>Message additionnel (via &&).</p>}
    </div>
  );
}
```