	Course Name: Advanced Web Technology	**EXPERIMENT NO. 9**	
	Course Code: 20CP314P **Faculty: Komal Singh**	**Branch: CSE**	**Semester: VI**
Submitted by: 21BCP362 **Roll no: 21BCP362**			

Objective: Working with react hooks.

Experiment 9:

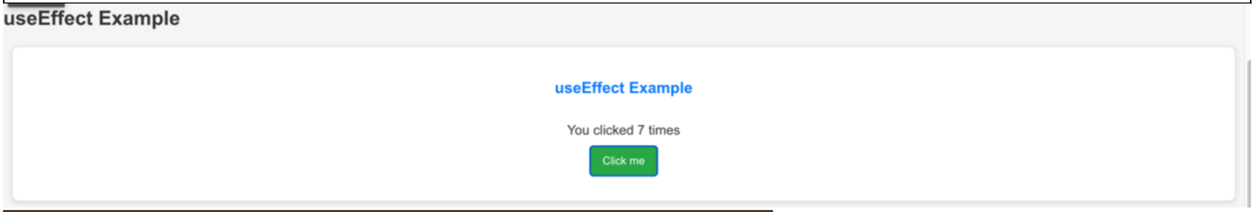
1) Write react programs displaying the usage of each of the below given react hooks:
   a. useReducer
   b. useMemo
   c. useCallback
   d. useContext
   e. useEffect

2) Design a custom hook that helps you declare a default value of a component.

Note: Please include snapshots of all commands, terminal sessions, localhost outputs, and log files in your documentation with necessary steps.

UseEffect

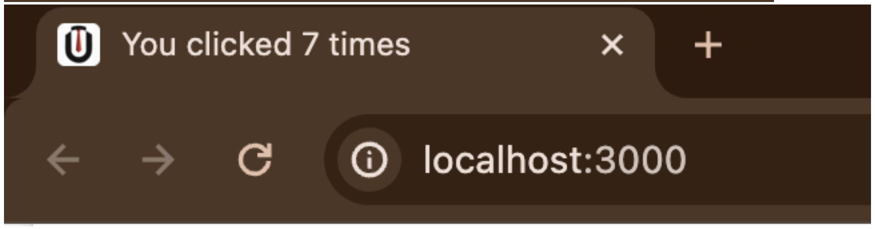
```
// src/components/UseEffectExample.js
import React, { useState, useEffect } from 'react';
import '../ComponentStyles.css';

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

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  });
  return (
    <div className="hook-example use-effect-notification">
      <h3>useEffect Example</h3>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  );
}

export default UseEffectExample;
```


useEffect Example

useEffect Example

You clicked 7 times

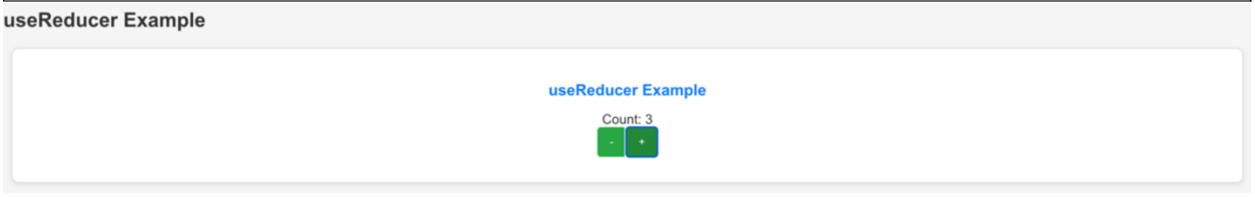
You clicked 7 times



useReducer

Handles a simple counter with increment and decrement actions.

```
// src/components/UseReducerExample.js
import React, { useReducer } from 'react';
import '../ComponentStyles.css';  // Ensure the path is correct based on your project structure
function reducer(state, action) {
  switch (action.type) {
    case 'increment':
      return { count: state.count + 1 };
    case 'decrement':
      return { count: state.count - 1 };
    default:
      throw new Error();
  }
}
function UseReducerExample() {
  const [state, dispatch] = useReducer(reducer, { count: 0 });
  return (
    <div className="hook-example">
      <h3>useReducer Example</h3>
      Count: {state.count}
    <div className="use-reducer-buttons">
     <button onClick={() => dispatch({ type: 'decrement' })}>-</button>
     <button onClick={() => dispatch({ type: 'increment' })}>+</button>
    </div>
   </div>
  );
}
export default UseReducerExample;
```


useReducer Example

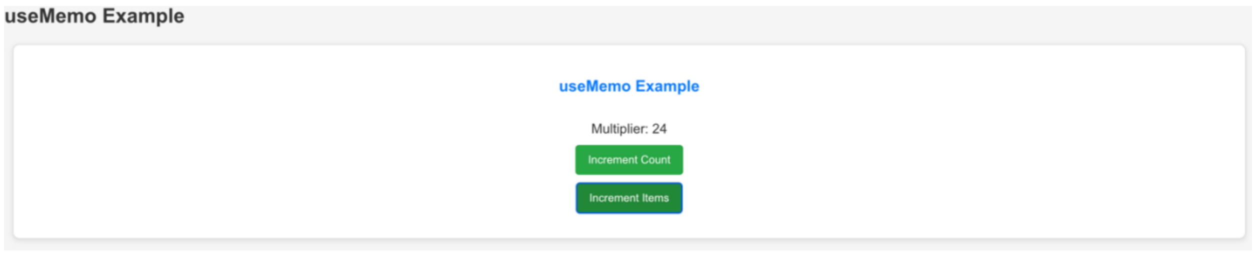
useReducer Example

Count: 3

useMemo

Calculates a "multiplier" based on count and items state, but only recalculates when count or items change.

```
// src/components/UseMemoExample.js
import React, { useState, useMemo } from 'react';
import '../ComponentStyles.css';

function UseMemoExample() {
  const [count, setCount] = useState(0);
  const [items, setItems] = useState(10);

  const multiplier = useMemo(() => {
    console.log('Calculating multiplier...');
    return count * items;
  }, [count, items]);

  return (
    <div className="hook-example">
      <h3>useMemo Example</h3>
      <p>Multiplier: {multiplier}</p>
      <button onClick={() => setCount(count + 1)}>Increment Count</button>
      <button onClick={() => setItems(items + 1)}>Increment Items</button>
    </div>
  );
}

export default UseMemoExample;
```

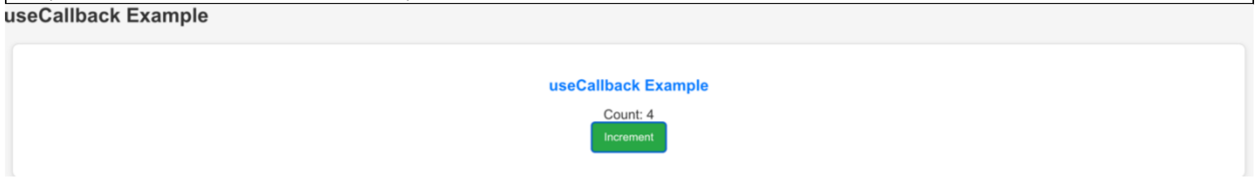

useMemo Example

useMemo Example

Multiplier: 24

useCallback

```
// src/components/UseCallbackExample.js
import React, { useState, useCallback } from 'react';
import '../ComponentStyles.css';

function UseCallbackExample() {
  const [count, setCount] = useState(0);
  const increment = useCallback(() => {
    setCount(c => c + 1);
  }, [setCount]);
  return (
    <div className="hook-example">
      <h3>useCallback Example</h3>
      Count: {count}
      <button onClick={increment} className="use-callback-button">Increment</button>
    </div>
  );
}

export default UseCallbackExample;
```


useCallback Example

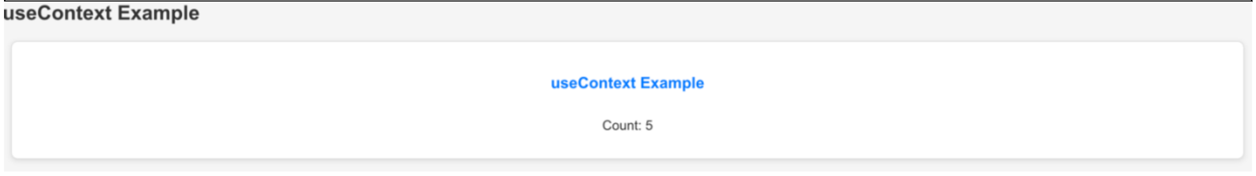
useCallback Example

Count: 4

useContext

```
// src/components/UseContextExample.js
import React, { useContext, createContext } from 'react';
import '../ComponentStyles.css';

const CountContext = createContext();

function Counter() {
  const count = useContext(CountContext);
  return <p>Count: {count}</p>;
}

function UseContextExample() {
  return (
    <div className="hook-example use-context-display">
      <h3>useContext Example</h3>
      <CountContext.Provider value={5}>
        <Counter />
      </CountContext.Provider>
    </div>
  );
}

export default UseContextExample;
```

useContext Example


useContext Example

Count: 5

useDefault

```
// src/hooks/useDefault.js
import { useState } from 'react';

function useDefault(initialValue) {
  const [value, setValue] = useState(initialValue);

  return [value, setValue];
}

export default useDefault;
```

CSS

```
/* src/ComponentStyles.css */

/* General Styles for All Hook Examples */
.hook-example {
  margin-bottom: 40px;
  padding: 20px;
  background-color: #ffffff;
  border: 1px solid #ddd;
  border-radius: 8px;
  box-shadow: 0 2px 6px rgba(0,0,0,0.1);
}

.hook-example h3 {
  color: #007bff;
```

```
}

.hook-example p {
  color: #666;
  font-size: 16px;
}

.hook-example button {
  padding: 10px 15px;
  margin-top: 10px;
  background-color: #28a745;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.hook-example button:hover {
  background-color: #218838;
}

/* Specific Styles for useReducer Example */
.use-reducer-buttons {
  display: flex;
  justify-content: space-between;
  margin-top: 20px;
}

/* Specific Styles for useMemo Example */
.use-memo-output {
  font-style: italic;
}

/* Specific Styles for useCallback Example */
.use-callback-button {
  background-color: #17a2b8;
}

.use-callback-button:hover {
  background-color: #138496;
}

/* Specific Styles for useContext Example */
.use-context-display {
  background-color: #f8f9fa;
  border-left: 5px solid #ffc107;
  padding: 10px;
  margin-top: 10px;
```

```
}

/* Specific Styles for useEffect Example */
.use-effect-notification {
  background-color: #f4f4f4;
  border: 1px solid #ccc;
  padding: 15px;
  font-size: 14px;
}

/* Styles for UncontrolledForm */
.uncontrolled-form-container {
  background-color: #e9ecef;
  border: 1px solid #ced4da;
}
.uncontrolled-reviews-display {
  background-color: #f8f9fa;
  border-top: 3px solid #dee2e6;
  padding: 10px;
}
```