**Alekhya Yanala**

Here are some beginner-level React useEffect practice coding challenges:

1. **Title Update on Component Mount:**
   * Create a component that updates the document's title to "Welcome to My Page" when the component mounts.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Count Component with useEffect:**
   * Create a counter component with a button that increments the count. Use useEffect to log the count value in the console whenever it changes.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Fetching Data from an API:**
   * Build a component that fetches data from a placeholder API (use one of apis that we used earlier) when the component mounts and displays the data in a list.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Update State After Delay:**
   * Create a component with a button that, when clicked, waits for 3 seconds before changing a piece of state (e.g., isLoading) to false.

Before pressing Start:

A blue start button with black text

AI-generated content may be incorrect.

After Pressing Start: it will wait form 3sec

A blue and black text

AI-generated content may be incorrect.

After 3sec:

A blue button with black text

AI-generated content may be incorrect.

1. **Tracking Window Resize:**
   * Create a component that tracks and displays the current width and height of the browser window using useEffect to listen for resize events.

A white background with black text

AI-generated content may be incorrect.

A black text on a white background

AI-generated content may be incorrect.

1. **Conditional Effect Execution:**
   * Create a component with an input field. Use useEffect to log a message when the value of the input changes, but only if the value is not empty.

Before Typing The Input:

A screenshot of a computer

AI-generated content may be incorrect.

After Typing the Input:

A screenshot of a computer

AI-generated content may be incorrect.

1. **Effect Triggered by Props Change:**
   * Create a parent component that passes a prop to a child component. In the child, use useEffect to log a message whenever the prop changes.

A close-up of a text

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

1. **Timer with Cleanup:**
   * Create a timer that counts down from 10 seconds. Use useEffect to start the timer when the component mounts and to clean up any timers when it unmounts.

A screenshot of a computer

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

1. **Fetching Data with Dependencies:**

* Create a component that fetches user data from an API. Use useEffect to fetch data when a button is clicked, and ensure it only fetches data when the button is clicked (not on initial render).

Before Clicking the Button:

A close-up of a white background

AI-generated content may be incorrect.

After Clicking the Button:

A close-up of a text

AI-generated content may be incorrect.

A close-up of a text

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

1. **Effect with Multiple Dependencies:**

* Build a component with two state variables (searchTerm and pageNumber). Use useEffect to fetch results based on both the searchTerm and pageNumber states, and trigger the effect whenever either of them changes.

A screenshot of a computer

AI-generated content may be incorrect.

A close-up of a sign

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Focus Input on Mount:**

* Create a component with an input field that automatically **focuses (css focus)** on the input when the component is mounted using useEffect.

A screenshot of a computer

AI-generated content may be incorrect.

1. **LocalStorage Sync:**

* Create a component that saves user input to localStorage. Use useEffect to update localStorage whenever the input changes, and load the value from localStorage when the component mounts.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Event Listener for Scroll:**

* Build a component that listens for scroll events using useEffect and logs the current scroll position of the window.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Updating Parent State from Child: (I will cover this in tmrw’s class)**

* Create a parent-child component setup where the parent passes a state to the child. The child uses useEffect to trigger a state update in the parent whenever a certain condition is met (e.g., a button click).

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

1. **Changing Styles Dynamically: (I will cover this in tmrw’s class)**

* Create a component that changes the background color of a div when the component mounts using useEffect. Change the color back to the original when the component unmounts.

A blue and black text

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

1. **Polling with useEffect: (I will cover this in tmrw’s class)**

* Build a component that periodically fetches data from an API every 5 seconds using setInterval in useEffect. Make sure to clean up the interval when the component unmounts.

A white background with black text

AI-generated content may be incorrect.

1. **Toggling Dark Mode:**

* Create a component with a toggle switch to enable/disable dark mode. Use useEffect to update the page's background color or class based on the dark mode state.

A white and grey switch

AI-generated content may be incorrect.

A screen shot of a black background

AI-generated content may be incorrect.

1. **Synchronizing Multiple Effects:**

* Create a component with two effects: one for logging a message when a button is clicked and another for logging a message when the component is mounted. Ensure both effects work together without conflicting.

A screenshot of a computer

AI-generated content may be incorrect.

After Clicking the button:

A screenshot of a computer

AI-generated content may be incorrect.

1. **Unmounting Effects in a Modal: (I will cover this in tmrw’s class)**

* Create a modal component that fetches data when it mounts. When the modal is closed, clean up the data fetching request using useEffect.

A blue button with black text

AI-generated content may be incorrect.

On clicking “Open Modal”:

A screenshot of a computer

AI-generated content may be incorrect.

On clicking “Close Modal”:

A blue and white button with black text

AI-generated content may be incorrect.

1. **Timer Start/Stop on Button Click: (I will cover this in tmrw’s class)**

* Create a button that toggles a timer. When clicked, the timer should start and increment every second. Clicking the button again should stop the timer.

A screenshot of a computer screen

AI-generated content may be incorrect.

By Clicking Start Timer:

A blue and black text

AI-generated content may be incorrect.

After Clicking “Stop Timer”:

A screenshot of a computer

AI-generated content may be incorrect.

1. **Effect that Runs Once:**

* Create a component that fetches data from an API on the initial render using useEffect, and ensure that the data is fetched only once (even if the component re-renders).

A close up of a sign

AI-generated content may be incorrect.

1. **Custom Hook for Fetching Data**

* Create a custom hook that handles fetching data from an API. The hook should use useEffect internally to fetch data and return the data, loading state, and error state.

A close-up of a computer code

AI-generated content may be incorrect.