Trainer guide
Examples and exercises (React)
React-9 – Inverse data flow

Contents

React-9a	Exercise: Inverse data flow

React-9a | Exercise: Inverse data flow

Link to environment

This exercise gives learners practice on passing functions as props, where state is lifted up to a common ancestor and children components use functions passed as props in order to update state in the parent.

src/App.jsx

- This exercise relates to an aspect of React that can be confusing at first, so don't worry if you don't fully understand it right away.
- Here we have another simple counter app.
    - At the top of the page the counter value is displayed, via the <CounterDisplay> component.
    - Under that is a button to increment the counter, which is the <IncrementButton> component.
    - And finally there is a button that resets the counter to 0, which is the <ResetButton> component.
    - Currently, the buttons are not working as intended.
- In the App component, we have created a state value called "count".
- This state is placed within the parent App component so that it can be accessed by all three child components. For example, <CounterDisplay> receives the count value via a prop and displays that value.

src/components/IncrementButton.jsx

- Now think from the perspective of the <IncrementButton> component. When the button is clicked, it needs to somehow communicate that to the App component. But <App> is the parent of <IncrementButton>.
- The problem we need to solve is: how can a child component talk to its parent component?

Exercise

- Your task is to make the app function as intended.
- In App.jsx, notice that some functions have been included (`handleIncrement` and `handleReset`). If only there was a way to give the relevant components access to these functions...

Solve the exercise live before moving on.

Solution

- Pass the `handleIncrement` function to <IncrementButton> via a prop called `increment`. Add an `onClick` handler to the button that executes `increment`.
- Pass the `handleReset` function to <ResetButton> via a prop called `reset`. Add an `onClick` handler to the button that calls `reset`.

Solution reference