Trainer guide
Examples and exercises (React)
React-4 – Rendering lists

Contents

Trainer demo | Rendering lists

Link to environment

In this demo, you will show learners how to transform an array of data into an array of components.

src/App.jsx

- Here we have a list of to-do items.
- At this stage, we're hard-coding the to-do items as components.
- However, this has some problems. For example:
    - The code is repetitive.
    - In a real-world app, we'd probably receive the to-do items as data from a server. Given that we can't predict what the data will be, our to-do list component should ideally be able to handle any array of to-do items that gets thrown at it.

src/todos.json

- Here we have the data in JSON format. It's an array of objects, with each object representing a single to-do item.

src/App.jsx

- *Import todos.json into App.jsx*
    - `import todos from './todos.json'`
- A handy feature of React is that, if it receives an array of components, it will render all of those components in sequence.
- So, if we could somehow turn this array of data into an array of components, we'd be most of the way there.
- We can use the JavaScript array `map` function to achieve this.
    - *Write in the body of the component:*
      `const todoComponents = todos.map((todo) => <TodoItem title={todo.title} />)`
- Now, we can replace all of the hard-coded TodoItem components and instead simply write `{todoComponents}`.
- *Replace the contents of the <ul> with `{todoComponents}`.*
- *Refresh the page and open the console to show the warning: "Warning: Each child in a list should have a unique "key" prop."*
    - This means that each component in the list needs a "key" prop that uniquely identifies it.
    - The reason for this is that, if the contents of the list change, keys help React identify which items have changed. This is important for performance and for preventing unexpected behaviour.

    - Best practice is to use a unique and constant ID associated with each specific data item.
- We can use the “id” property of each to-do item as a key.
    - *Add a key prop to TodoItem with a value of **todo.id**.*
    - Refresh the page to show that the error no longer appears.
- Finally, instead of creating a separate variable called todoComponents to store the array, we could simply call the map() method directly within the JSX.
    - *Move the **`todos.map(…)`** expression inside the <ul> element.*

```
import React from 'react'
import TodoItem from './components/TodoItem'
import todos from './todos.json'
import './style.css'

export default function App() {
  return (
    <main>
      <h1>To-do List</h1>
      <ul>
        {todos.map((todo) => <TodoItem title={todo.title} key={todo.id} />)}
      </ul>
    </main>
  )
}
```

React-4a | Exercise: Render a list from data

Link to environment

This exercise gives learners practice on rendering lists from data using Array.map().

src/itemsData.json

- All of the data for the items in the fruit market is contained within `itemsData.json`.

src/App.jsx

- Inside the div with class name “items-grid”, we want all of the items to be rendered.
- But this time, we’ll use the Array map() function to transform the array of data into an array of components.

Exercise

- Import the fruit market items data from itemsData.json
- Render a list of <ItemCard> components from this data.

Solve the exercise live before moving on.

Solution reference