Starter files: `axiosquiz`, solutions: `styledquiz`

What are Styled Components?

Styled components are a popular CSS-in-JS library for React. They offer a way to write CSS directly within your JavaScript components, using tagged template literals. This approach provides several benefits:

- **Scoping:** Styles are scoped to the component they're defined in, preventing conflicts with other styles in your application.
- **Dynamic Styling:** You can easily style components based on their props or a global theme.
- **No Class Name Hassles:** Styled components generate unique class names automatically, eliminating the need to worry about naming conflicts.
- **Improved Developer Experience:** The syntax feels natural to JavaScript developers and allows for seamless integration with React's component model.

How Styled Components Work

1. **Create a Styled Component:**

```
const QuestionContainer = styled.div`
  margin-bottom: 20px;
`;
```

Here, we use the `styled` object from the `styled-components` library and call it with the HTML tag name you want to style (in this case, `div`). The backticks (``) allow you to write CSS rules within them.

Use the Styled Component:

```
<QuestionContainer>
  <QuestionText>{questionText}</QuestionText>
  {/* ... rest of the question content ... */}
</QuestionContainer>
```

The `QuestionContainer` acts just like a regular React component. When it renders, it creates an HTML `div` element with the CSS styles you've defined applied to it.

Dynamic Styling:

```
const Button = styled.button`

  background-color: ${props => props.primary ? 'blue' :
'gray'};
`;

<Button primary>Click Me</Button>

Styled components can accept props, allowing you to
customize their styles based on the props passed to them.
Here we use props to conditionally change the background
color of the button.
```

Styled Components in the Quiz App

In the quiz application code, styled components are used to:

- **Style individual components:** `QuestionContainer`, `QuestionText`, `OptionLabel`, etc. Each component is styled with specific CSS rules to give it its intended appearance.
- **Encapsulate styles:** The styles are directly associated with the components, making the code more organized and easier to understand.
- **Maintain consistent look:** By using styled components throughout the app, you can ensure a cohesive visual style.

Example: `QuestionContainer`

```
const QuestionContainer = styled.div`
  margin-bottom: 20px;
`;
```

This styled component does the following:

1. **Creates a styled `div` element:** It takes a regular `div` and adds specific styling to it.
2. **Adds margin:** It adds a bottom margin of 20 pixels to create space between questions.

Key Benefits in this Project:

- **Improved Readability:** The CSS styles are directly within the component, making it clear which styles apply to which elements.
- **Modularity:** Each component has its own styling, making the code more modular and maintainable.
- **Dynamic Styling:** The ability to use props for styling could be helpful if you wanted to, for example, change the appearance of a question based on whether it's been answered correctly or not.

Improvements we will make to the code:

- **Styled Components:** Used `styled-components` to create reusable styled components like `QuestionContainer`, `QuestionText`, `OptionLabel`, and `QuizContainer`. This makes the code cleaner and easier to maintain.
- **Error Handling:** Axios's `.catch()` block is used to handle potential errors during the fetch request.

How to implement:

1. **Make sure your JSON Server is running:** `json-server --watch db.json`
2. **Install dependencies:** `npm install axios styled-components`
3. **Run your React app:** `npm start`

```
import React, { useState, useEffect } from 'react';
import styled from 'styled-components';
import axios from 'axios';

const QuestionContainer = styled.div`
  margin-bottom: 20px;
`;

const QuestionText = styled.p`
  font-weight: bold;
`;

const OptionLabel = styled.label`
  display: block;
  margin-bottom: 5px;
`;

const QuizContainer = styled.div`
  max-width: 600px;
```

```
  margin: 0 auto;
  padding: 20px;
  border: 1px solid #ddd;
  border-radius: 5px;
`;

function Question({ questionText, options, correctAnswer,
selectedOption, onSelectOption }) {
  return (
    <QuestionContainer>
      <QuestionText>{questionText}</QuestionText>
      {options.map(option => (
        <OptionLabel key={option}>
          <input
            type="radio"
            name={questionText}
            value={option}
            checked={selectedOption === option}
            onChange={() => onSelectOption(option)}
          />
          {option}
        </OptionLabel>
      ))}
    </QuestionContainer>
  );
}

function useFetch(url) {
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    axios.get(url)
      .then(response => setData(response.data))
      .catch(err => setError(err))
      .finally(() => setLoading(false));
  }, [url]);

  return { data, loading, error };
```

```
}

function Quiz() {
  const [score, setScore] = useState(0);
  const [currentQuestionIndex, setCurrentQuestionIndex] =
useState(0);
  const [selectedOption, setSelectedOption] =
useState(null);
  const { data: questions, loading, error } =
useFetch('http://localhost:3000/questions');

  const handleNextQuestion = () => {
    if (questions[currentQuestionIndex].correctAnswer ===
selectedOption) {
      setScore(score + 1);
    }
    setCurrentQuestionIndex(currentQuestionIndex + 1);
    setSelectedOption(null);
  };

  if (loading) return <p>Loading questions...</p>;
  if (error) return <p>Error: {error.message}</p>;

  return (
    <QuizContainer>
      {currentQuestionIndex < questions.length ? (
        <>
          <Question

questionText={questions[currentQuestionIndex].questionText}

options={questions[currentQuestionIndex].options}

correctAnswer={questions[currentQuestionIndex].correctAnswe
r}
            selectedOption={selectedOption}
            onSelectOption={setSelectedOption}
          />
          <button onClick={handleNextQuestion}>Next</
button>
```

```
        </>
      ) : (
        <p>You scored {score} out of {questions.length}</p>
      )}
    </QuizContainer>
  );
}

export default Quiz;
```