Task 11: Testing in React for a Profile Website

Objective:

Learn to write unit and integration tests for a React application using Jest and React Testing Library. This task focuses on ensuring that individual components work as expected and simulating user interactions in the context of a profile website.

Pre-requisites:

- Basic understanding of HTML, CSS, and JavaScript
- Familiarity with a code editor like Visual Studio Code
- Basic knowledge of React

Concepts Covered:

- Unit Testing with Jest
- Integration Testing with React Testing Library

Concepts:

1. **Unit Testing with Jest:**

   Write unit tests for the Header component using Jest.

   ```
   // src/components/Header.js
   import React from 'react';

   const Header = ({ title }) => {
       return (
           <header>
               <h1>{title}</h1>
           </header>
       );
   };

   export default Header;
   ```

   Hints:

   - Use Jest to write basic unit tests for the Header component.
   - Example hint:

     ```
     // src/components/Header.test.js
     import React from 'react';
     import { render } from '@testing-library/react';
     import Header from './Header';

     test('renders the header title', () => {
         const { getByText } = render(<Header title="Profile Website" />);
         expect(getByText('Profile Website')).toBeInTheDocument();
     });
     ```

2. **Integration Testing with React Testing Library:**
   Write integration tests for a user interacting with the ProfileList component.

```
// src/components/ProfileList.js
import React, { useState } from 'react';

const profiles = [
    { id: 1, name: 'John Doe', occupation: 'Software Engineer' },
    { id: 2, name: 'Jane Smith', occupation: 'Graphic Designer' },
    { id: 3, name: 'Sam Johnson', occupation: 'Product Manager' }
];

const ProfileList = () => {
    const [selectedProfile, setSelectedProfile] = useState(null);

    const handleClick = profile => {
        setSelectedProfile(profile);
    };

    return (
        <div>
            <h2>Profile List</h2>
            <ul>
                {profiles.map(profile => (
                    <li key={profile.id} onClick={() => handleClick(profile)}>
                        {profile.name} - {profile.occupation}
                    </li>
                ))}
            </ul>
            {selectedProfile && (
                <div>
                    <h3>Selected Profile</h3>
                    <p>Name: {selectedProfile.name}</p>
                    <p>Occupation: {selectedProfile.occupation}</p>
                </div>
            )}
        </div>
    );
};

export default ProfileList;
```

   Hints:

   - Use React Testing Library to simulate user interactions and test the integration of multiple components.
   - Example hint:

```
// src/components/ProfileList.test.js
import React from 'react';
import { render, fireEvent } from '@testing-library/react';
import ProfileList from './ProfileList';

test('selects and displays a profile on click', () => {
    const { getByText } = render(<ProfileList />);
    fireEvent.click(getByText('John Doe - Software Engineer'));
    expect(getByText('Name: John Doe')).toBeInTheDocument();
});
```

Setup:

1. **Install Visual Studio Code (VS Code):**

   Download and install VS Code from Visual Studio Code.

2. **React Developer Tools:**

   Install the React Developer Tools browser extension for Chrome or Firefox.

3. **Git:**

   Install Git for version control. You can download it from Git.

4. **Node.js and npm:**

   Ensure Node.js and npm are installed on your machine. You can download and install them from Node.js.

5. **Setting Up the React Project:**
   - Open your terminal.
   - Navigate to the directory where you want to create your project.
   - Run the following command to create a new React project:

   ```
   npx create-react-app profile-website
   ```

Tasks:

Part 1: Unit Testing with Jest (30 minutes)

1. **Write unit tests for the Header component using Jest:**
   - Example hint:

   ```
   // src/components/Header.test.js
   import { render } from '@testing-library/react';
   import Header from './Header';

   test('renders the header title', () => {
       const { getByText } = render(<Header title="Profile Website" />);
       expect(getByText('Profile Website')).toBeInTheDocument();
   });
   ```

- Goal: Get familiar with writing basic unit tests in React to ensure individual components work as expected.

Part 2: Integration Testing with React Testing Library (30 minutes)

1. **Write integration tests for a user interacting with the ProfileList component:**
   - Example hint:

     ```
     // src/components/ProfileList.test.js
     import { render, fireEvent } from '@testing-library/react';
     import ProfileList from './ProfileList';

     test('selects and displays a profile on click', () => {
         const { getByText } = render(<ProfileList />);
         fireEvent.click(getByText('John Doe - Software Engineer'));
         expect(getByText('Name: John Doe')).toBeInTheDocument();
     });
     ```

   - Goal: Understand how to write integration tests that simulate user interactions and test the integration of multiple components.

Instructions:

1. Write the required code in the appropriate files in the `src/components` directory.
2. Open the terminal and navigate to the project directory.
3. Run the React project using:

   ```
   npm start
   ```

4. Open the project in a web browser to ensure the code displays correctly.
5. Use the browser's developer tools and React Developer Tools to debug and inspect the elements.
6. Run the tests using:

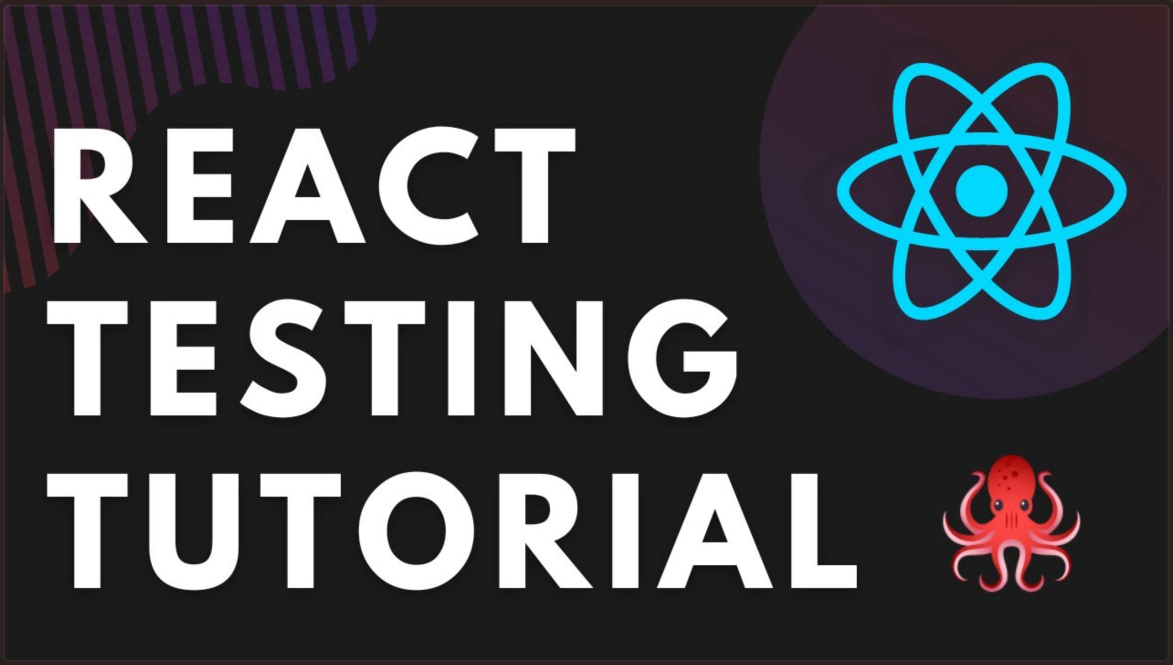
   ```
   npm test
   ```

Resources:

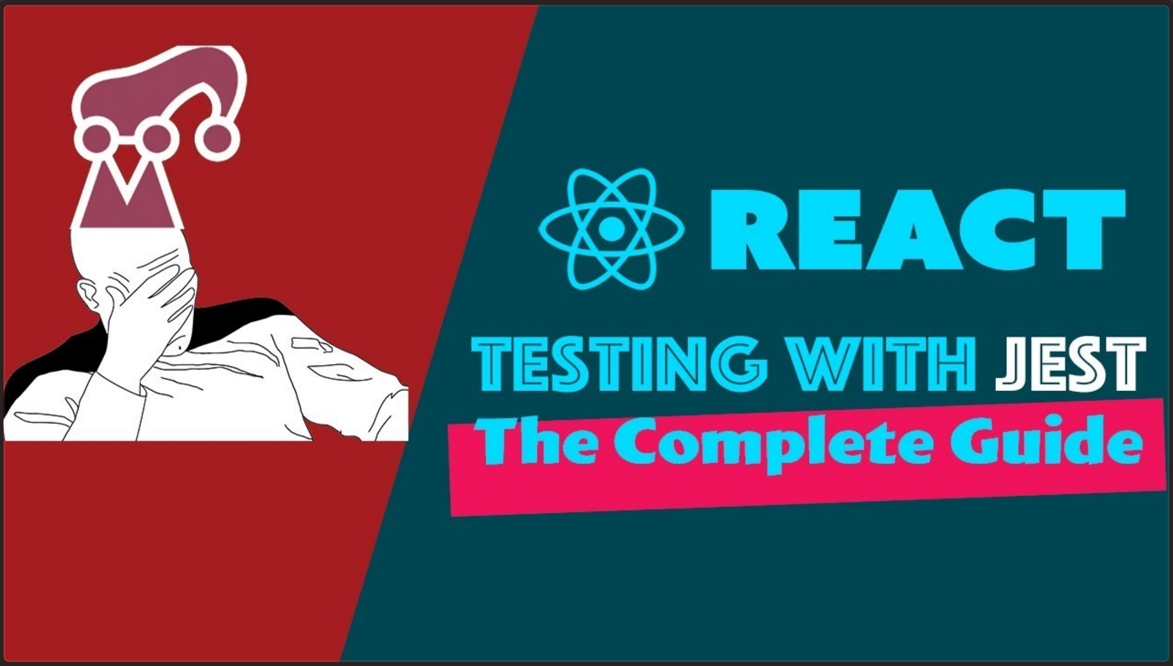
- Jest
  Getting Started · Jest
  Install Jest using your favorite package manager:
  https://jestjs.io/docs/en/getting-started

- React Testing Library | Testing Library
  React Testing Library builds on top of DOM Testing Library by adding
  https://testing-library.com/docs/react-testing-library/intro
- Getting Started – React
  A JavaScript library for building user interfaces
  https://reactjs.org/docs/getting-started.html
- Create React App
  Set up a modern web app by running one command.
  https://create-react-app.dev/
- Documentation for Visual Studio Code
  Find out how to set-up and get the most from Visual Studio Code. Optimized for building and d...
  https://code.visualstudio.com/docs

Videos:

- 

  PedroTech 21:28 min 110,054 Views 2,851 Likes
- [React Testing Library Tutorial]

- 

techsith 23:49min 390,793 Views 4,492 Likes

GitHub Instructions:

1. **Open in Visual Studio Code:**

   After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, VSCode will open the repository directly. If prompted, select "Open" or "Allow" to open the repository in VSCode.

2. **Open the Terminal in VSCode:**

   In VSCode, open a terminal by selecting Terminal > New Terminal from the top menu.

3. **Complete the Task:**

   In VSCode, write your solution in the appropriate files in the `src/components` directory.

4. **Run and Test Your Code:**

   Open your terminal, navigate to your project directory, and run:

   ```
   npm start
   ```

5. **Commit Your Changes:**

   In the VSCode terminal, add your changes to git:

   ```
   git add src/components/*
   ```

   Commit your changes with a meaningful message:

```
git commit -m "Completed task 32"
```

6. **Push Your Changes to Your Repository:**
   Push your changes to your forked repository:

   ```
   git push origin main
   ```

7. **Create a Pull Request:**
   Go to your repository on GitHub.
   Click on the "Pull Requests" tab.
   Click the "New Pull Request" button.
   Ensure the base repository is the original template repository and the base branch is `main` .
   Ensure the head repository is your forked repository and the compare branch is `main` .
   Click "Create Pull Request".
   Add a title and description for your pull request and submit it.

Summary of Commands:

```
# Open in Visual Studio Code

# Open terminal in VSCode

# Complete the task by editing src/components/*

# Navigate to the project directory
cd profile-website

# Run your code
npm start

# Add, commit, and push your changes
git add src/components/*
git commit -m "Completed task 11"
git push origin main

# Create a pull request on GitHub
```