Hans-on 4

Objectives

- Explain the need and Benefits of component life cycle
- Identify various life cycle hook methods
- List the sequence of steps in rendering a component

In this hands-on lab, you will learn how to:

- Implement componentDidMount() hook
- Implementing componentDidCatch() life cycle hook.

Prerequisites

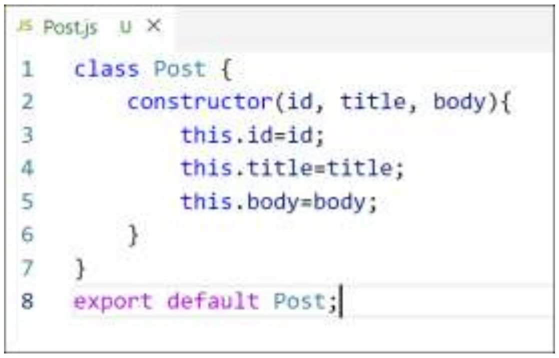
The following is required to complete this hands-on lab:

- Node.js
- NPM
- Visual Studio Code

Notes

Estimated time to complete this lab: **60 minutes.**

1. Create a new react application using *create-react-app* tool with the name as "blogapp"
2. Open the application using VS Code
3. Create a new file named as **Post.js** in **src folder** with following properties

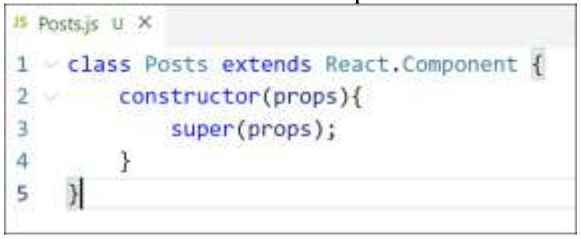


Post.js U ×

```
class Post {
    constructor(id, title, body){
        this.id=id;
        this.title=title;
        this.body=body;
    }
}
export default Post;
```

Figure 1: Post class

4. Create a new class based component named as **Posts** inside **Posts.js** file



Posts.js U ×

```
class Posts extends React.Component {
    constructor(props){
        super(props);
    }
}
```

Figure 2: Posts Component

5. Initialize the component with a list of Post in state of the component using the constructor

6. Create a new method in component with the name as **loadPosts()** which will be responsible for using Fetch API and assign it to the component state created earlier. To get the posts use the url (https://jsonplaceholder.typicode.com/posts)

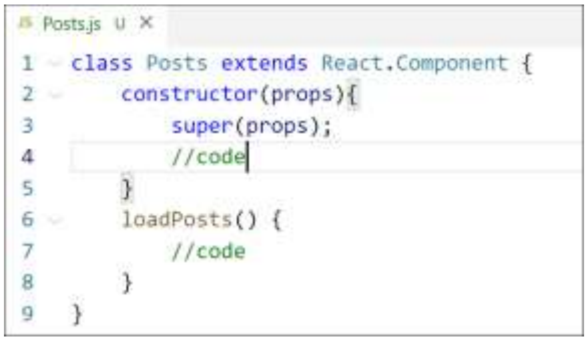

Posts.js U ×
```
class Posts extends React.Component {
    constructor(props){
        super(props);
        //code
    }
    loadPosts() {
        //code
    }
}
```

Figure 3: loadPosts() method

7. Implement the **componentDidMount()** hook to make calls to **loadPosts()** which will fetch the posts

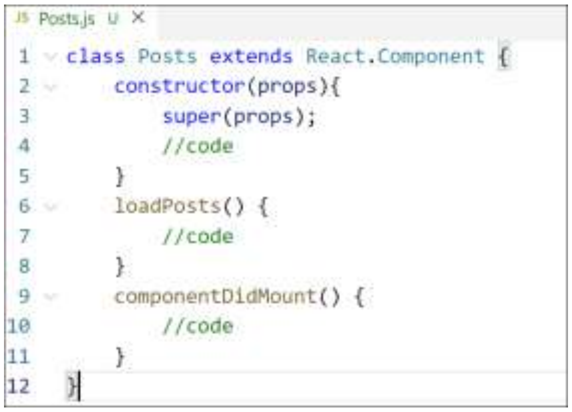

Posts.js U ×
```
class Posts extends React.Component {
    constructor(props){
        super(props);
        //code
    }
    loadPosts() {
        //code
    }
    componentDidMount() {
        //code
    }
}
```

Figure 4: componentDidMount() hook

8. Implement the **render()** which will display the title and post of posts in html page using heading and paragraphs respectively.

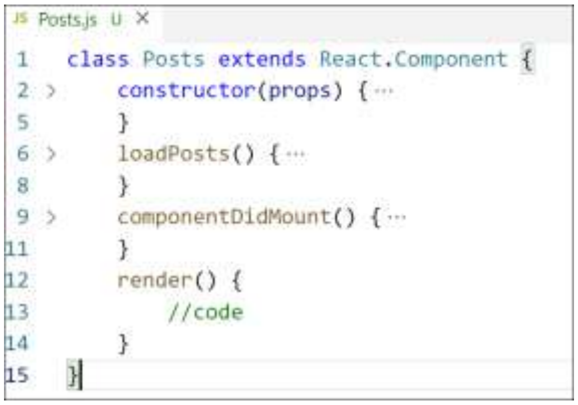
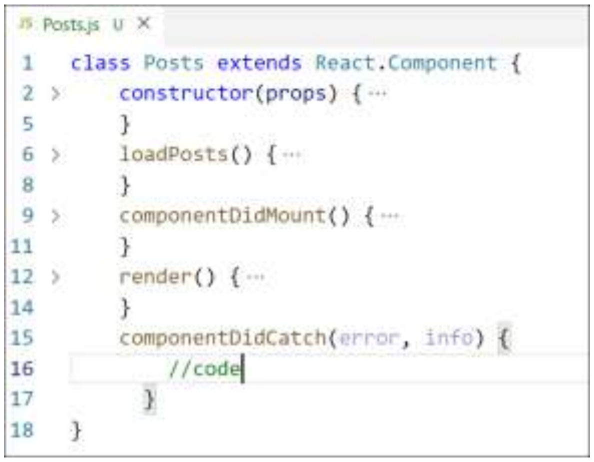

Posts.js U ×
```
class Posts extends React.Component {
    constructor(props) {···
    }
    loadPosts() {···
    }
    componentDidMount() {···
    }
    render() {
        //code
    }
}
```

Figure 5: render() method

9. Define a **componentDidCatch()** method which will be responsible for displaying any error happing in the component as alert messages.

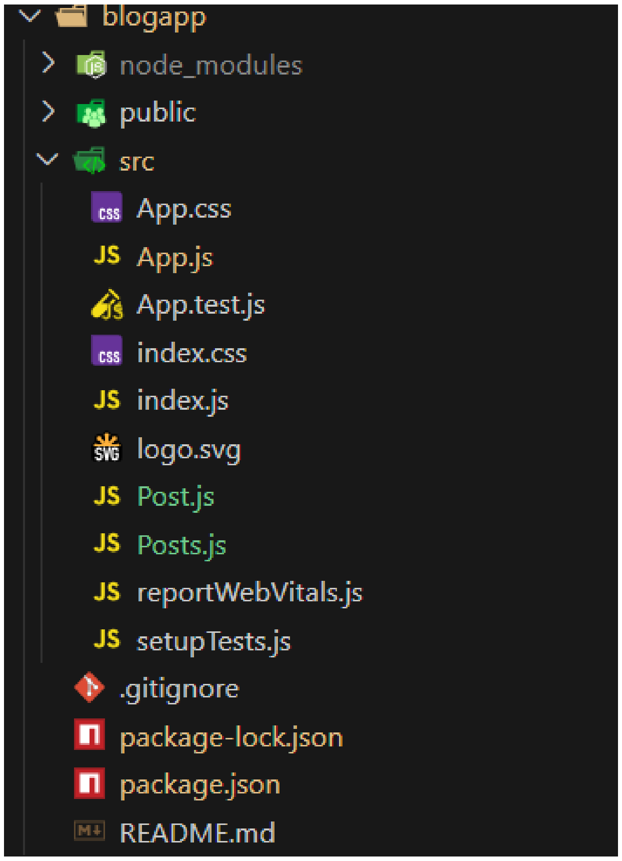


```
class Posts extends React.Component {
    constructor(props) {···
    }
    loadPosts() {···
    }
    componentDidMount() {···
    }
    render() {···
    }
    componentDidCatch(error, info) {
        //code
    }
}
```

Figure 6: componentDidCatch() hook

10. Add the Posts component to App component.
11. Build and Run the application using *npm start* command.

Implementation



```
blogapp
  node_modules
  public
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    Post.js
    Posts.js
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

App.js

```
import logo from './logo.svg';
import './App.css';
import Posts from './Posts';

function App() {
  return (
    <div className="App">
      <Posts/>
    </div>
  );
}

export default App;
```

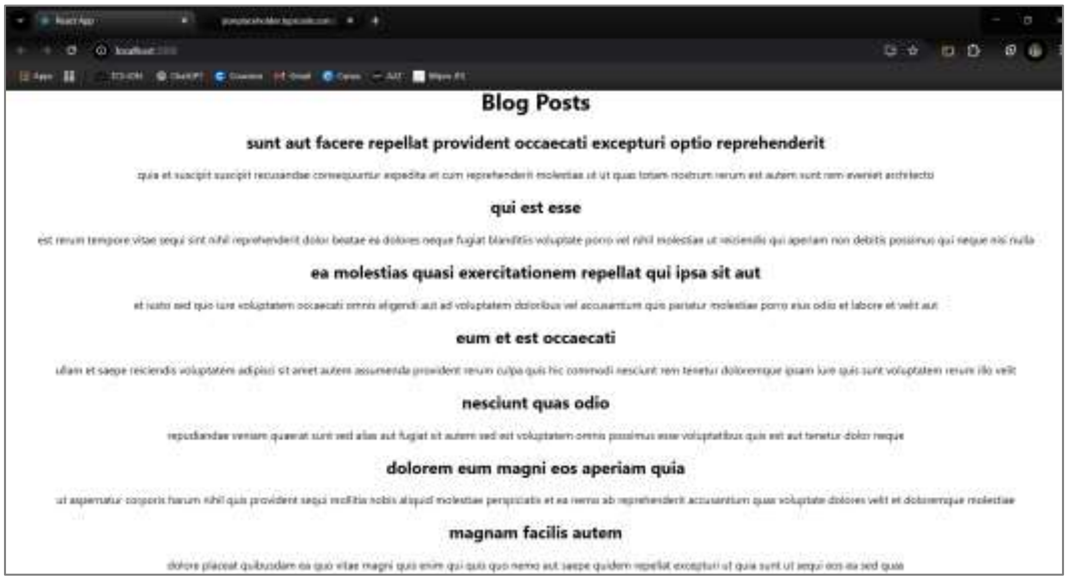
Post.js

```
class Post {
  constructor(id, title, body) {
    this.id = id;
    this.title = title;
    this.body = body;
  }
}

export default Post;
```

Posts.js

```
import React, { Component } from 'react';
import Post from './Post';

class Posts extends Component {
  constructor(props) {
    super(props);
    this.state = {
      posts: []
    };
  }

  loadPosts = () => {
    fetch('https://jsonplaceholder.typicode.com/posts')
      .then(response => response.json())
      .then(data => {
        const postList = data.map(post => new Post(post.id, post.title, post.body));
        this.setState({ posts: postList });
      })
      .catch(error => {
        console.error('Error fetching posts:', error);
        throw error;
    });
  }

  componentDidMount() {
    this.loadPosts();
  }

  componentDidCatch(error, info) {
    alert("An error occurred in Posts component: " + error.message);
  }

  render() {
    return (
      <div>
        <h1>Blog Posts</h1>
        {this.state.posts.map(post => (
          <div key={post.id}>
            <h2>{post.title}</h2>
            <p>{post.body}</p>
          </div>
        ))}
      </div>
    );
  }
}

export default Posts;
```

<u>Output:</u>



Blog Posts

sunt aut facere repellat provident occaecati excepturi optio reprehenderit

quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto

qui est esse

est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla

ea molestias quasi exercitationem repellat qui ipsa sit aut

et iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et velit aut

eum et est occaecati

ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit

nesciunt quas odio

repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque

dolorem eum magni eos aperiam quia

ut aspernatur corporis harum nihil quis provident sequi mollitia nobis aliquid molestiae perspiciatis et ea nemo ab reprehenderit accusantium quas voluptate dolores velit et doloremque molestiae

magnam facilis autem

dolore placeat quibusdam ea quo vitae magni quis enim qui quis quo nemo aut saepe quidem repellat excepturi ut quia sunt ut sequi eos ea sed quas