4. ReactJS-HOL

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

```
JS 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

```
JS 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)

```
JS 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.

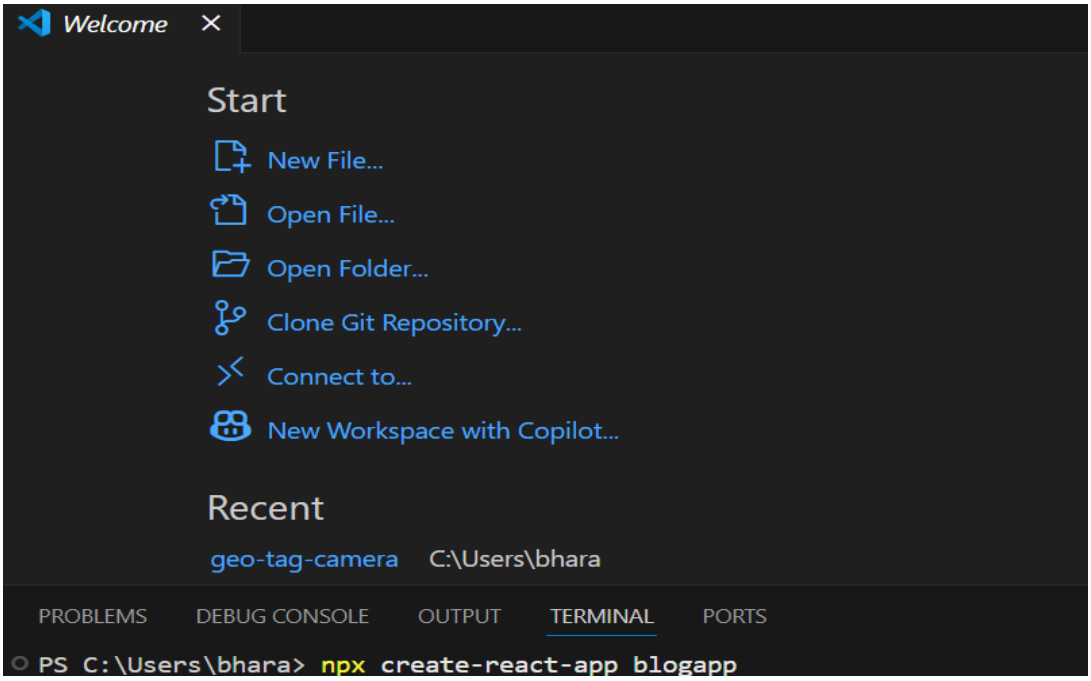
```
JS Posts.js U ×
1   class Posts extends React.Component {
2 >     constructor(props) {…
5       }
6 >     loadPosts() {…
8       }
9 >     componentDidMount() {…
11      }
12 >    render() {…
14      }
15      componentDidCatch(error, info) {
16          //code
17      }
18  }
```

Figure 6: componentDidCatch() hook

10. Add the Posts component to App component.

11. Build and Run the application using *npm start* command.

Create a New React Application



Create the Post.js File

In the src folder, create a new file named Post.js and add the following code:

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

```
JS Post.js U ×   JS Posts.js U   JS App.js M
src > JS Post.js > ...
1   // src/Post.js
2   class Post {
3     constructor(id, title, body) {
4       this.id = id;
5       this.title = title;
6       this.body = body;
7     }
8   }
9
10  export default Post;
```

Create the Posts Component

In the src folder, create another file named Posts.js and create a class-based component:

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

```
class Posts extends Component {
  constructor(props) {
    super(props);
    this.state = {
      posts: [],
      error: null
    };
  }
  loadPosts = async () => {
    try {
      const response = await fetch('https://jsonplaceholder.typicode.com/posts');
      const data = await response.json();
      const posts = data.map(p => new Post(p.id, p.title, p.body));
      this.setState({ posts });
    } catch (error) {
      this.setState({ error });
    }
  }
  componentDidMount() {
    this.loadPosts();
  }

  componentDidCatch(error, info) {
    alert("Something went wrong: " + error.toString());
  }
  render() {
    return (
      <div>
```

```
      <h1>All Posts</h1>
      {this.state.posts.map(post => (
        <div key={post.id}>
          <h3>{post.title}</h3>
          <p>{post.body}</p>
        </div>
      ))}
    </div>
  );
 }
}
export default Posts;
```

```
JS Posts.js > ...
import React, { Component } from 'react';
import Post from './Post';
class Posts extends Component {
  constructor(props) {
    super(props);
    this.state = {
      posts: [],
      error: null
    };
  }
  loadPosts = async () => {
    try {
      const response = await fetch('https://jsonplaceholder.typicode.com/posts');
      const data = await response.json();
      const posts = data.map(p => new Post(p.id, p.title, p.body));
      this.setState({ posts });
    } catch (error) {
      this.setState({ error });
    }
  }

  componentDidMount() {
    this.loadPosts();
  }

  componentDidCatch(error, info) {
    alert("Something went wrong: " + error.toString());
  }

  render() {
    return (
      <div>
        <h1>All Posts</h1>
        {this.state.posts.map(post => (
          <div key={post.id}>
            <h3>{post.title}</h3>
            <p>{post.body}</p>
          </div>
        ))}
      </div>
    );
  }
}

export default Posts;
```

App.js:

```
import React from 'react';

import './App.css';

import Posts from './Posts';

function App() {

  return (

    <div className="App">

      <Posts />

    </div>

  );

}
```

export default App;

Run the Application:

npm start

OUTPUT:

---

voluptatem doloribus consectetur est ut ducimus

ab nemo optio odio delectus tenetur corporis similique nobis repellendus rerum omnis facilis vero blanditiis debitis in nesciunt doloribus dicta dolores magnam minus velit

beatae enim quia vel

enim aspernatur illo distinctio quae praesentium beatae alias amet delectus qui voluptate distinctio odit sint accusantium autem omnis quo molestiae omnis ea eveniet optio

voluptas blanditiis repellendus animi ducimus error sapiente et suscipit

enim adipisci aspernatur nemo numquam omnis facere dolorem dolor ex quis temporibus incidunt ab delectus culpa quo reprehenderit blanditiis asperiores accusantium ut quam in voluptatibus voluptas ipsam dicta

et fugit quas eum in in aperiam quod

id velit blanditiis eum ea voluptatem molestiae sint occaecati est eos perspiciatis incidunt a error provident eaque aut aut qui

consequatur id enim sunt et et

voluptatibus ex esse sint explicabo est aliquid cumque adipisci fuga repellat labore molestiae corrupti ex saepe at asperiores et perferendis natus id esse incidunt pariatur

repudiandae ea animi iusto

officia veritatis tenetur vero qui itaque sint non ratione sed et ut asperiores iusto eos molestiae nostrum veritatis quibusdam et nemo iusto saepe

aliquid eos sed fuga est maxime repellendus