Agenda

- useState Hook in react
- Function-based components, state and state manipulation
- Handling State using hooks (useState() - for the project purpose we will only make use of useState hook. Other hooks will be covered in further modules in detail)
- Handling Different events
- Conditional Rendering
- Forms in react

Before beginning with the project let's get a brief idea about what hooks are in react:

What are react hooks in short?

Before React version 16.8, developers could handle state and other React features only using class components. But with version 16.8, React introduce a new pattern called `Hooks` . With React Hooks, we can use state, and other React features, in a functional component. We will study more abo hooks in upcoming modules.

The useState hook

`useState` is a Hook that allows you to have state variables in functional components. You pass the initial state to this function and it returns a variab with the current state value (not necessarily the initial state) and another function to update this value.

Whereas the state in a class is always an object, with Hooks, the state can be any type. Each piece of state holds a single value, which can be an objec an array, a boolean, or any other type you can imagine.

So when should you use the `useState` Hook? It's especially useful for local component state, but larger projects might require additional sta management solutions.

`useState` is a named export from `react` . To use it, you can write:

```
React.useState
```

OR

```
import React, { useState } from 'react';
```

The `useState` Hook allows you to declare only one state variable (of any type) at a time, like this:

```
import React, { useState } from 'react';

const Message= () => {
   const messageState = useState('');
   const listState = useState([]);
}
```

`useState` takes the initial value of the state variable as an argument. You can pass it directly, as shown in the previous example

But `useState` doesn't return just a variable as the previous examples imply.

It returns an array, where the first element is the state variable and the second element is a function to update the value of the variable:

```
const Message= () => {
   const messageState = useState( '' );
   const message = messageState[0]; // Contains ''
   const setMessage = messageState[1]; // It's a function
}
```

Usually, you'll use array destructuring to simplify the code shown above:

```
const Message= () => {
   const [message, setMessage]= useState( '' );
}
```

The second element returned by `useState` is a function that takes a new value to update the state variable.

Here's an example that uses a text box to update the state variable on every change:

```
const Message = () => {
  const [message, setMessage] = useState( '' );

  return (
    <div>
      <input
         type="text"
         value={message}
         placeholder="Enter a message"
         onChange={e => setMessage(e.target.value)}
       />
      <p>
        <strong>{message}</strong>
      </p>
    </div>
  );
}
```

However, this update function doesn't update the value right away.

Rather, it enqueues the update operation. Then, after re-rendering the component, the argument of `useState` will be ignored and this function w
return the most recent value.If you use the previous value to update state, you must pass a function that receives the previous value and returns the ne
value:

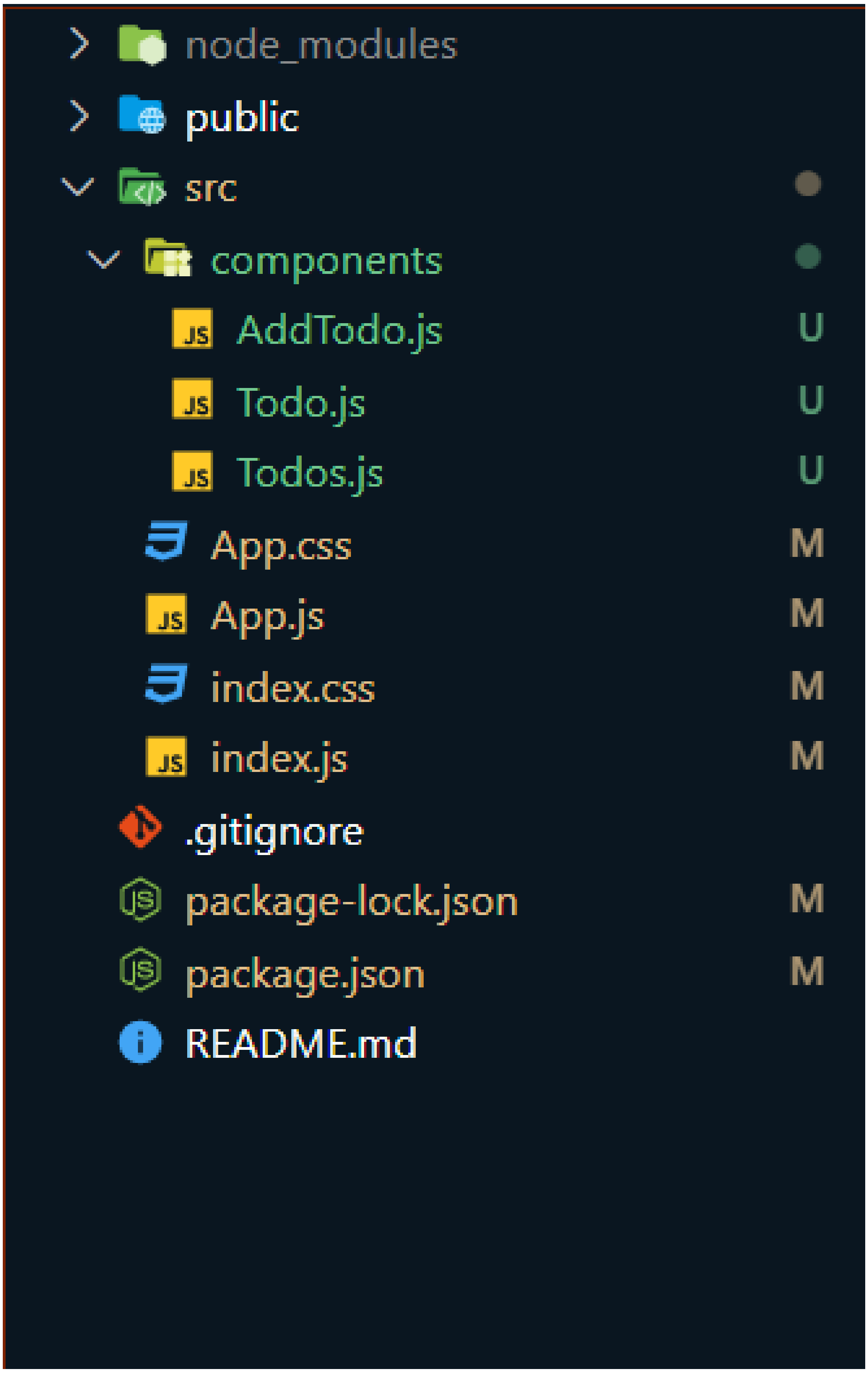
```
const Message = () => {
  const [message, setMessage] = useState( '' );

  return (
    <div>
      <input
        type="text"
        value={message}
        placeholder="Enter some letters"
        onChange={e => {
          const val = e.target.value;
          setMessage(prev => prev + val)
        } }
      />
      <p>
        <strong>{message}</strong>
      </p>
    </div>
  );
};
```

We will study useState in detail in upcoming modules.

Let's Begin

We will use the same code we wrote for React Project - Class-based and re-write all the code according to functional components from scratch.

We will use the same project we created for the last class. The project structure will look like this:



App.js

```
import "./App.css";
import Todos from "./components/Todos";

const App = () => {
    return (
        <div className='container'>
            <h1 className='text-center'>ToDo App in ReactJS</h1>
            <Todos />
        </div>
    );
};

export default App;
```

Todos.js

In Todos component there are 2 state properties **editTodo and todos**

In Class Component *Todos*, we have added some methods which are defined as arrow notation as we are going to access these methods from oth
Components like *Todo* and *AddTodo*.

getTime() is getting used to adding an ID to ToDo.

handleDone, handleDelete and **addNewTodo, editTodo** are method whose references are getting passed as Component attributes, which will t
called from their respective Components.

We have also included our modal to edit todo in this component.

```
import { useState } from "react";
import AddTodo from "./AddTodo";
import Todo from "./Todo";

const Todos = () => {
    const [editTodo, setEditTodo] = useState({});
    const [todos, setTodos] = useState(
        localStorage.getItem("todos")
            ? JSON.parse(localStorage.getItem("todos"))
            : [],
    );

    //Local helper method to get date
    function getTime() {
        let d = new Date();
        var n = d.getTime();
        return n;
    }

    //method called from Todo component
    const handleDelete = todo => {
        const todosArr = todos?.filter(t => {
            return t.id !== todo.id;
        });
        setTodos(todosArr);
        localStorage.setItem("todos", JSON.stringify(todosArr));
    };

    const handleDone = todo => {
        const todosArr = [...todos];
        todosArr?.map(t => {
            if (t.id === todo.id) {
                t.isDone = !t.isDone;
            }
            return t;
        });
        setTodos(todosArr);
        localStorage.setItem("todos", JSON.stringify(todosArr));
    };

    //method called from AddTodo component
    const addNewTodo = value => {
        if (value) {
            const todosArr = [...todos];
            todosArr?.push({
                id: getTime(),
                value: value,
                isDone: false,
            });
            setTodos(todosArr);
            localStorage.setItem("todos", JSON.stringify(todosArr));
        } else {
            alert("Please add a value");
        }
    };

    const editTodoFun = todo => {
        const todosArr = [...todos];
        todosArr?.map(t => {
            if (t.id === todo.id) {
                t.value = todo.value;
            }
            return t;
        });
        setEditTodo({});
        setTodos(todosArr);
        localStorage.setItem("todos", JSON.stringify(todosArr));
    };

    const setEditValue = todo => {
        setEditTodo(todo);
    };

    return (
        <div>
            {todos?.length <= 0 && (
                <div className='alert alert-info text-center' role='alert'>
                    <b>No Todos Added</b>
                </div>
            )}
            <table className='table'>
                <tbody>
                    {todos.map((todo, index) => (
                        <tr key={todo.id}>
                            <Todo
                                index={index + 1}
                                todo={todo}
                                fooDelete={handleDelete}
                                fooDoneDone={handleDone}
                                fooEdit={setEditValue}
                            />
                        </tr>
                    ))}
                    <tr>
                        <td colSpan='4' className='text-center'>
                            <AddTodo fooAddTodo={addNewTodo} />
                        </td>
                    </tr>
                </tbody>
            </table>
            <div className='modal fade' id='exampleModal'>
                <div className='modal-dialog'>
                    <div className='modal-content'>
                        <div className='modal-header'>
                            <h5 className='modal-title' id='exampleModalLabel'>
                                Update Todo Value
                            </h5>
                            <button
                                type='button'
                                className='btn-close'
                                data-bs-dismiss='modal'
                                aria-label='Close'></button>
                        </div>
                        <div className='modal-body'>
                            <form
                                onSubmit={e => {
                                    e.preventDefault();
                                    editTodoFun(editTodo);
                                }}>
                                <div className='mb-3'>
                                    <label htmlFor='recipient-name' className='col-form-label'>
                                        Value:
                                    </label>
                                    {editTodo?.value && (
                                        <input
                                            type='text'
                                            className='form-control'
                                            value={editTodo.value}
                                            onChange={e =>
                                                setEditTodo({
                                                    ...editTodo,
                                                    value: e.target.value,
                                                })
                                            }
                                        />
                                    )}
                                </div>
                                <div className='modal-footer'>
                                    <button
                                        type='button'
                                        className='btn btn-secondary'
                                        data-bs-dismiss='modal'>
                                        Close
                                    </button>
                                    <button
                                        type='submit'
                                        className='btn btn-primary'
                                        data-bs-dismiss='modal'>
                                        Update
                                    </button>
                                </div>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    );
};

export default Todos;
```

Todo.js

Todo Component will represent a single Todo in the list and have methods **fooDoneDone** (*check/ uncheck event handler*) and **fooDelete** (delete butt
event handler)

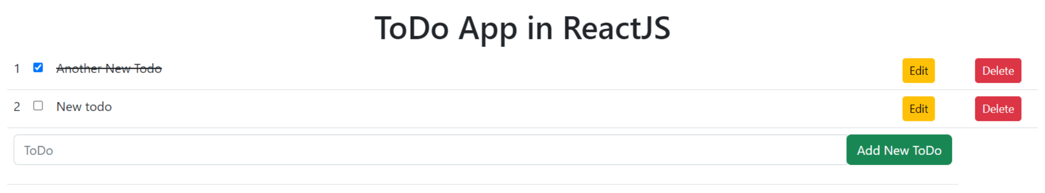
```
import React from "react";

const Todo = props => {

    function renderTodo() {
        if (props.todo.isDone) return <s>{props.todo.value}</s>;
        else return props.todo.value;
    }

    return (
        <React.Fragment>
            <td style={{ width: 10 }} className='text-center'>
                {props.index}
            </td>
            <td style={{ width: 15 }} className='text-center'>
                <input
                    type='checkbox'
                    defaultChecked={props.todo.isDone}
                    onChange={() => props.fooDoneDone(props.todo)}
                />
            </td>
            <td>{renderTodo()}</td>

            <td style={{ width: 100 }} className='text-center'>
                <button
                    data-bs-toggle='modal'
                    data-bs-target='#exampleModal'
                    type='button'
                    className='btn btn-warning btn-sm'
                    onClick={() => props.fooEdit(props.todo)}>
                    Edit
                </button>
            </td>
            <td style={{ width: 100 }} className='text-center'>
                <button
                    onClick={() => props.fooDelete(props.todo)}
                    className='btn btn-danger btn-sm'>
                    Delete
                </button>
            </td>
        </React.Fragment>
    );
};

export default Todo;
```

AddTodo.js

In AddTodo class we have **handleChange** method to set the todo value and we have the state variable **value.**

```
import { useState } from "react";

const AddTodo = props => {
    const [value, setValue] = useState(props.addTodoValue);

    const handleChange = e => {
        setValue(e.target.value);
    };

    const clearInput = () => {
        setValue("");
    };

    const addTodo = e => {
        e.preventDefault();
        props.fooAddTodo(value);
        clearInput();
    };

    return (
        <form onSubmit={addTodo}>
            <div className='input-group mb-3'>
                <input
                    type='text'
                    className='form-control'
                    id='todoValue'
                    placeholder='ToDo'
                    value={value}
                    onChange={handleChange}
                />
                <div className='input-group-append'>
                    <button className='btn btn-success' type='submit' id='button-addon2'>
                        Add New ToDo
                    </button>
                </div>
            </div>
        </form>
    );
};

export default AddTodo;
```

Final View:



ToDo App in ReactJS

1 ☑ ~~Another New Todo~~ Edit Delete

2 ☐ New todo Edit Delete

ToDo Add New ToDo