Day 19 Revisit

- React Forms.
- Inter-component Communications (Parent to child, Child to Parent)
- Events Handling in React.
- preventDefault() – Do avoid the default behaviour of form when submitting.
- onChange, onClick
- Handling event object.
- Event.target.name = name of the react element.
- Event.targer.value = the value entered by the user for that element.
- Hooks (useState, useEffect)
- Referred React official document, created Simple Products SPA using react.
- Atoms, molecules, Pages, organism
- React DevTools plugin in chrome
- HoC (Components passed as an argument to functions to create more effective components)
- Higher Order Components

https://dev.to/aliraza1603/atomic-design-methodology-with-react-and-typescript-1ac7

https://paulonteri.com/thoughts/atomic-design-react

Day 20 Agenda

- Redux for state management
  - Introduction to Redux
  - Basic building blocks
    - Store
    - Actions, action types and action creators
    - Dispatch mechanism
  - Vanilla Redux for better understanding
- Redux with ReactJS
  - The application flow
  - Choosing the right model for store
  - Implementing Redux for a basic application
- React Routers
- Testing React Application
- API handling in React (fetch, AJAX, Axios)

https://rapidapi.com/blog/how-to-use-an-api-with-react/

React Routing

Browser Router, Routes, Route, Link

Configuring Route and Link

Route Component should have path and element attribute.

Path specifies the part of URL

Element helps to render the corresponding component.

Routes is the containers of all the Route.

Link helps to append the URL (similar to anchor tag in html)

Making API Calls

1) Simple way – using fetch in-built method (JS in-built method)
2) Using AJAX (XMLHttpRequest) [open(), send(), readyState, response]
3) Using third party library (axios) – Provides support to Promise, simple error handling.

Axios will perform JSON handling and Parsing

Error Handling in React Forms

1) Can be done using JS directly
2) Using 3rd Party libraries such as classnames etc.,

https://react-hook-form.com/get-started#Applyvalidation

https://www.geeksforgeeks.org/axios-in-react-a-guide-for-beginners/

Redux – Is a Predictable state container for JS apps

```
npx create-next-app --example with-redux my-app
```

https://blog.logrocket.com/understanding-redux-tutorial-examples/

Redux = Mainly used to manage the state of react application from a central place (Store)

Actions = Plain JS object which contains type (property) & payload (object_

```
{
  type: "INCREMENT",
  payload: {
    incrementBy: 5,
  }
```

```
}
```

Actions are created with action creator (which is a function that returns action)

Action Creator Example :

```
const getIncrementAction = (numberToIncrement) => {
  return {
    type: "INCREMENT",
    payload: {
      incrementBy: numberToIncrement,
    }
  }
}
```

Actions are executed with dispatch() method

Dispatcher = It's a method to execute actions – which sends the action to store

Reducer = Pure function which takes current state as i/p, process and give new_state(changed_state) as o/p

Reducers are based on the `reduce` function in JavaScript, where a single value is calculated from multiple values after a callback function has been carried out.

Here is an example of how reducers work in Redux:

```
const CounterReducer = (state = initialState, action) => {
    switch (action.type) {
      // This reducer handles any action with type "LOGIN"
      case "INCREMENT":
          return state + action.incrementBy ? action.incrementBy : 1
       default:
          return state;
      }
};
```

```
npm install @reduxjs/toolkit
```

```
npm install react-redux
```

```
npm install redux-thunk
```

```
npm install --save-dev redux-devtools-extension
```

```
https://redux.js.org/tutorials/quick-start
```

Pure Function - *A pure function is a function that will always return the same value if given the same parameters, i.e., the function depends on only the parameters and no external data.*

https://react-redux.js.org/introduction/getting-started