

React JS

Lab Guide

EXERCISES on REACT JS, REACT ROUTER & REDUX

1. Create a React element that represents a login form. It should have as children the following elements

- a. A form with id 'login-form', and action set to '/login'
- b. An input element to take username
- c. An input element to take password
- d. A submit button

You may add additional elements as children if you wish to. You may also enclose the inputs within a div etc. for styling purpose (these are optional).

Also, render the React element inside the root div.

2. Redo the above exercise by creating the element using JSX.

3. Define 2 elements for 2 forms - one for login and the other for sign up.

- the login form is like in the previous exercise
- the signup form asks user to choose username, password and has an additional input that confirms password.

You need to render both these in the root element - Try each of the following 3 ways to do it.

- a. Render the 2 forms as children of a container (like a div, or a section)
- b. Add the 2 forms to an array and render the array of React elements
- c. Enclose the 2 forms in a React.Fragment and render

4. Define function components called LoginForm and SignupForm. They render UI as in the previous exercise. Render them on the page using elements of the component types.

5. Sign up at <https://openweathermap.org/> (Sign up), and view the API key at https://home.openweathermap.org/api_keys. You will need to use the current weather API - <https://openweathermap.org/current> (this is free with usage limits).

Define a function component called WeatherReport that shows details of Wind, Cloudiness, Pressure, Humidity, Sunrise, Sunset and Geographical coordinates for a given city. Get some details for your city from <https://openweathermap.org/city>. These details are to be passed as props. You can also find details of weather by going to this URL - substitute city name with your city's name. Also substitute your api key.

...

api.openweathermap.org/data/2.5/weather?q={city name}&appid={your api key}

...

The weather data also contains a description of the weather and the name of the icon representing the weather condition - use this to display an image of the weather condition within the component. You can find details of how to get the URL of the icon (i.e. image) from the incoming data, here

<https://openweathermap.org/weather-conditions>

Also define a function component called SunriseSunset that shows Sunrise and Sunset timings. Use this within WeatherReport to show the details of sunrise and sunset. The props need to be gathered within WeatherReport using rest operator, and passed to an element of SunriseSunset as props (using props spread operator).

__NOTE__:

- a. The API key may take a few hours to generate. You will not require the API till you start working with the backend to fetch weather data and use it (will be done in forthcoming exercises). For a guide on using the API check <https://openweathermap.org/guide>
- b. If you are able to get the API key, you can try making an Ajax call using Axios to fetch the current weather data for your city. Once the call is successful, you can render the WeatherReport element within the root (passing it the data you obtained using the API call as props).

6. Redo the above exercise using a class components.

7. Create a class component that shows numbers in the Fibonacci sequence. Initially 0, 1 are shown. With the passing of every second a new number should be added to the list of numbers shown. For example, after a second, "0, 1, 1" is displayed, then after one more second, "0, 1, 1, 2" etc. What will you maintain in the state of this component?

8. Add styles to the WeatherReport component by showing the details inside a box with background color dictated by weather condition codes (group codes) - <https://openweathermap.org/weather-conditions>.

For example,

- If the weather is clear, show the details in lightblue background
- If the weather is rainy/drizzle - use dark blue

Similarly choose some colors for the other weather conditions.

9. Create an CitiesWeatherReport component that let's one view weather details for a group of cities. It takes as input (props) the weather details for a group of cities. It should display the weather details for each city. When doing so, use the WeatherReport component created in a previous exercise. Also make sure to give a unique key prop value for each WeatherReport element rendered.

...

```
const tasks = [
  {
    "name": "New York",
    "weather": [
      {
        "id": 800,
        "main": "Clear",
        "description": "clear sky",
        "icon": "01n"
      }
    ]
  }
]
```

```
],
"main": {
  "temp": 281.52,
  "feels_like": 278.99,
  "temp_min": 280.15,
  "temp_max": 283.71,
  "pressure": 1016,
  "humidity": 93
},
...
},
{
  "name": "New York",
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01n"
    }
  ],
  "main": {
    "temp": 281.52,
    "feels_like": 278.99,
    "temp_min": 280.15,
    "temp_max": 283.71,
    "pressure": 1016,
    "humidity": 93
  },
  ...
},
...
]
...
```

__NOTE__: You can get weather details for multiple cities in one API call - check

<https://openweathermap.org/current#severalid>. Extra credits if you make an Ajax call to fetch the weather data and then pass them as props to the CitiesWeatherReport component when rendering it.

10. Create a Calculator component. It has 2 input boxes that take user inputs (operand), and a dropdown that has 4 basic arithmetic operations - +, -, *, /. There is a button that user clicks after entering the operand values and selecting the operation. The complete expression, and the result should be displayed in the component. Use refs to get hold of input elements values.

11. Implement a WeatherTracker component that displays the weather for a list of cities. It render a CityInput component that accepts a city name and adds it to a list, as child. It also displays a CitiesWeatherReport element that shows weather reports for cities in the cities list that is maintained.

First decide what needs to be maintained in state. Which components need to work with the list of cities? Where will this state be maintained - WeatherTracker (parent), CityInput, or CitiesWeatherReport? Where will you define the method to add a city to the list of cities? How will CityInput be able to access this method?

__NOTE__:

- a. The componentDidMount() method of CitiesWeatherReport needs to make an Ajax call to fetch weather details for every city in the list, every time it is rendered.
- b. Fetching data for a city may be redundant, as previous Ajax calls may have fetched the data for existing cities. Try creating a data caching utility which maintains the data fetched for a particular city. Once an Ajax call to fetch data for a city is made frshly, the cached data is updated. If another call goes out for the same city within 15 mintes (say) on the same date, then the cached data may instead be used. This will take some amount of involved design and implementation. The data cache may be part of the state maintained by a component. This is completely optional.

12. We shall add filter functionality in the WeatherTracker component. Add a CityFilter component with a search input. User can type within this input and all cities that have the search term as a substring of the city name are displayed. For example, if "New" is typed, cities like "New York", "New Orleans" are displayed (assuming they exist in the list of added cities).

13. Add a delete button against each city list item in CitiesWeatherReport component. It should remove the city list item when clicked.

14. Add propTypes and defaultProps for all components in the Weather tracker application. Set appropriate data types for props, and also default values for the props where it makes sense.

15. Add loading, and error state UIs for the Weather tracker application apart from the loaded state UI which already exists. You can display a loading message for CitiesWeatherReport as it fetches the data.

16. Use Create React App to create the Weather tracker application. Modularize and organize the code you have created in earlier exercises using a standalone HTML file. Create folders for components, services (that has methods to fetch data from OpenWeatherMap API, and data caching layer in case you have implemented that), models (like CurrentWeather - that models the response from the API), actions etc. Your app should work like before.

17. We shall now break down the application into 3 screens.

* On top of each screen is a Navigation Menu - it has 2 links - "All Cities", and "All Reports".

Clicking "All Cities" takes user to the landing page (<http://localhost:3000/>). Clicking "All Reports" shows the page displaying weather reports for all cities (<http://localhost:3000/all>)

* <http://localhost:3000/>. It shows a list of cities that have been added in the application. The CityInput component accepts the name of a new city. A new component called CitiesList is displayed below it, that shows the names of cities added to the application.

* **http://localhost:3000/:city**, where :city is a placeholder for a city's name. Thus an actual route could be **http://localhost:3000/new+york** or **http://localhost:3000/sunnyvale** etc. The component displays the weather details for the specified city only.

* **http://localhost:3000/all** - The CitiesWeatherReport component that displays weather report of all added cities is displayed (only limited weather information is displayed) Clicking any one city within this list, displays the detailed weather report for that city (**http://localhost:3000/:city**)

18. Introduce Redux into the Weather tracker application.

- * Store the cities list and weather data fetched for cities in the Redux store.
- * Define a reducer to take care of making state changes for fetch of weather data for a particular city, and weather data for a list of cities.
- * Define the weather data for cities in a "normalized" form for easy data access.
- * When making a call to fetch data for a list of list of cities, check whether each city's weather data is already available in the store. If so, exclude that city automatically - a good place to do this is in the Saga middleware functions.

www.digdeeper.in