Topic : The Todo App Using Node Persist Storage

By : Abhyanshu Pandey

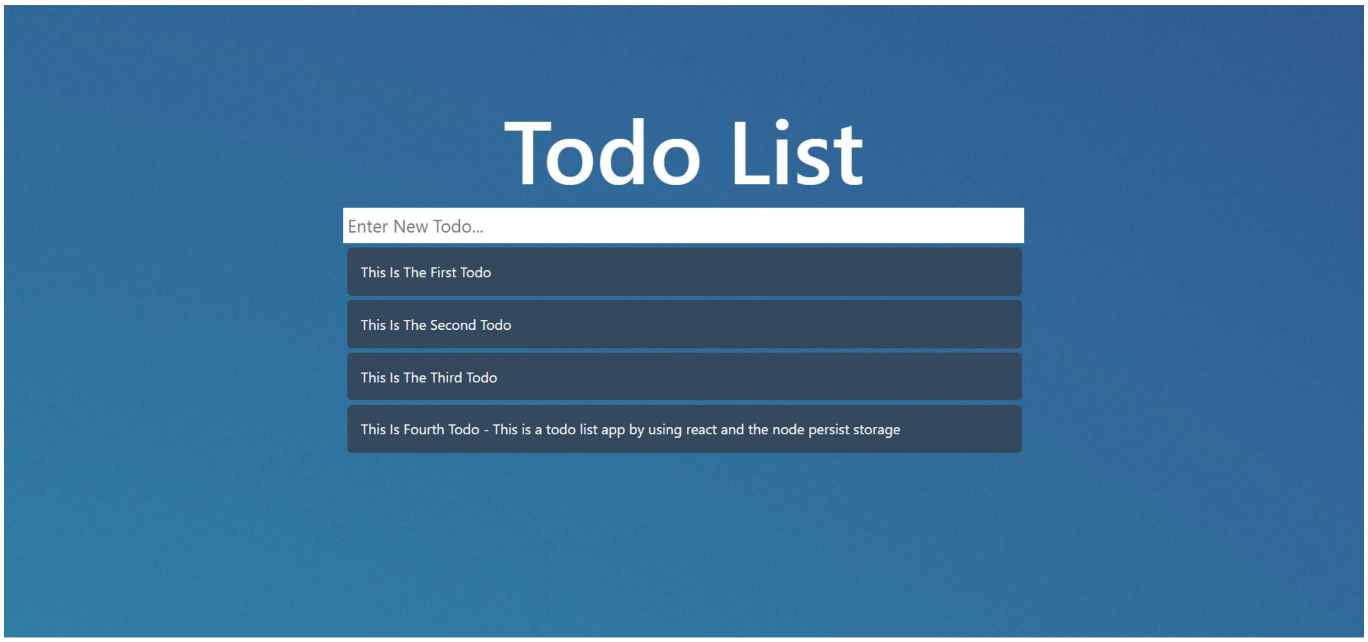
Github link : https://github.com/AbhyanshuPandeyji/Todo-list-with-nodejs.git,

Netlify Link : https://todo-using-node-persist-by-abhyanshu.netlify.app/,

Welcome to this todo app that is been created by using react and node js .

You can refer to instructions file to know how you can run this app , however this app isn’t live yet so there is no link for that .

The Working Image of the app



This Is The First Todo

This Is The Second Todo

This Is The Third Todo

This Is Fourth Todo - This is a todo list app by using react and the node persist storage

The Backend :

1-The server file:

- This will be our main server connection to frontend file
- We will install some packages mentioned in the instructions for the website
- It will take the route.js for routing
- First we will import express for setting up the connection with backend to frontend
- Then cors to select this server as different port numbers
- Then the storage this will come from the node persist storage to save our data
- Importing routes from the route file
- Then importing body parser for the parsing the json data to object
- Initializing storage to save data into node-storage
- Using app as express method to call or to work with all the server side works
- In it using cors,bodyparser json , and urlecoded in app use
- Our main route will be '/'
- Our port number will be 8000

2- The Routes File :

- This file will contain our routes for the connection to the data
- We will import express here and the logical functions we create in controller folder to work with the data
- The route will be taken from the express.Router(); an express function that helps us to create cleaner looking routing system , instead of fetch requests
- There will be one post and one get request both on the '/todos' location since its an single page app

3- The Controller file :

- This will have our todo controller file which will have our logic behind the requests
- We will initialize the storage in it this is the actual place that we need to initialize the storage
- First function will be add todo which will take todo and add it in the storage with getItem and setItem function of the js
- And we will send the todos back in the frontend
- The next function will be get all todos which well take all the todos in the storage and send back to the frontend

The Frontend :

Installation :

- First we will initialize the npm module by npm init for starter
- Then we will do all the regular steps to create a initialize file
- Above steps are not required if you making it individually
- Then we will create a react app with the help of create-react-app

‘ npx create-react-app client ’

- This will create our react app now we can work in it for working with the front end of our app

1- <u>The App.js file :</u>

- We will be working in the src folder usually
- The packages will be installed as recommended in the instruction but for bit more , the packages will be installed in the client side of the app , and it will be different than the server side.
- This will be a simple app.js file which will hold all our components that are required
- It will be a function containing the div within it there will be our three component

  “ <Header/> , <TodoForm/> , <Todos/> “

<u>2- The Components folder :</u>

<u>A)- The Header file :</u>

- This will contain h1 tag with the heading todo list

<u>B)- The Todo Form :</u>

- We will import some packages here
- The react from react
- Usedispatch from react-redux – for sending the function for state management to action folder
- In main function there will be form submission which will take the todo we write in the input box of the return function and also after submission it will turn text in the input box to none
- This submission will be taken and changed by the use state function of react the text will be the initial state and the setText will be new state
- The input box will be have a on change which will set the new value in the box to the text we input it will change continously while we are changing the text inside the box
- At end when the form will be submitted , the submit form function will be called and it will send the dispatch function to change the state from the api call to the backend

<u>C)- The Todos file :</u>

- For this we will import some packages
- The usedispatch to send the api call function for state change
- Use selector to select the specific type of data state
- In the main todos function we will initialize the dispatch
- The use selector will be initialized and take the state by

“ state.todo “

And give that data into the todo variable

- Then the use effect will run and it will take the data of all the todo from the data base
- Then it will clear the localstorage everytime the data been taken so the same data wont be represented multiple times

- In return function it will map on the basis of the todos present in the data base
- Then the map function it will render the todo component from todo.js file on the basis of unique key of todo id and the todo.

<u>D)- The Todo file :</u>

- It will take the todo from the todos.js file as prop for the required data to render
- Todo file will just render the todo based on the data present in the numbered location of that todo from the id

3- The redux folder :

<u>A)- The Store.js file :</u>

- This will take createStore , combineReducer and applyMiddleware
- The create store to – have all the data and utilize it by state , this will also be in the index.js main file within the provider file to wrap the app component in it to utilize the stored states in the app.
- Combine reducer to combine all the type of reducer that will be take and change the state in the app.
- The middle ware it will be thunk to perform the bridge between the store and the state at app
- Then we will also import the composite dev tools to combine all the different tools to our utilization in one place , like middle ware , initial state etc . In our case just middle ware
- All at the end will be in the create store and the store will be exported for the app to use.

<u>B)- The Actions Folder :</u>

<u>1)- The index.js file :</u>

- This file will handle our actions / the api calls
- For this we will install and import the axios package to work with , axios help us to create api calls to backend instead of the fetch requests.
- And we also import the reducers from the reducers folder to work with and send the state back to the reducer to change it
- This will have a api url which will be of the backend to work with , we can also use the proxy for that for now we are using the In action api url -
  “ http://localhost:8000 ”
- First function will be add todo which will be the post request ,this request will data and save it
- Then it will send the data using dispatch the data into the reducer by using its contant variable name with the data as the payload
- If the error happened then it will show the error message
- The second function will be the get all todos , this will be a get request , this will request the

todo from the backend

- Then it will dispatch the todo into the reducer and send the data as a response

2)- The Type File :

- This file will hold our todos constants name so we don’t do mistakes in the name writing when we are mentioning the functions / action type at the reducer file.

C)- The Reducer Folder :

- This will have a reducer file – todoReducer.js
- This reducer file will handle our state change , usually with the switch case .
- It will import the type as action type so we don’t have to write every action type indivdually
- Then the todoreducer funtion will have the state which is initially will be an empty array and second argument as action
- In side this above function will be the switch case with argument as action type
- There will be two action type
- First one will be add new todo which will take the new todo and change the state of the app current no of todos and add one in it
- The second one will be the get all todo to show all the todo in the state sotred in the data base to the local storage it will return the payload as data to the frontend.

APP Thought Note Done:-

Things Learned During Making this project :

1. How to work with the local storage
2. How to interact with the node persisit storage from the react as the frontend and how to add and fetch data from it
3. How to use the single page application with constantly clearing the local storage.

THANK YOU NOTE

- I have been thankful to the teachers that taught me how to do the routing well and how to interact with the different parts of the languages in the stack
- Great full of the community that helped me in many ways for being a better developer.

INSUFICCIENCY IN THE PROJECT

- There is no data deleting for the todo list as the clearing the storage I am still struggling with the clear storage , cannot be able to delete the data of the node persist storage
- Can use the session storage for the deleting the data on the reload of the page , just currently unfamiliar with the difference between local , session and nodepersist storage and how and where to work with them , and how to use them effectively in the app and clear after use.
- Such as the local storage store the data in our local computer browser , but session storage store the data in just an instance if the page reloads the data is gone. But don’t know how to initiate the session storage for data storage. Tried to do it with the window.onload function to clear storage to clear the storage but didn’t work.
- Also the todos are not arranged in the descending format from newest to oldest
- Sorry for Any thing that’s been left out or inconvenience you face in the app.