Redux Introductiom

Working with Redux

Managing State in react Application

Problems we may face:

React Application - Tree of Web Components – Root –Branches

Redux maintains in a central place Global State in a container- on a Subscription – Observable Pattern

Redux observes changes to The Data -> based on the event , Observable process drives this method

**There is a Subscription to the Event for the change to the state.**

In redux the central location or global location is called a **store this is the Containier.**

Store is a JavaScript Object contains the concept called **Reducer**- What reducer have is ? The reducer can have the State

WE can have more than 1 **reducer** in my **Store.**

**Reducer**  is a JS function – Has the “**Current State**” – We need to do a Process on the Current State [Manipulating the data]. & returns the “**New State**” Reducer Function (Current State & Action)

Manipulating the Data – **Action on the State [Not mandatory & not predefined]**

Subscription takes 2 event – 1 success & 1-failure.

React is coming up with ContextApi for the State

**Action** is the Java Script Object – represents the change that needs to happen on the State.

Dispatch: - A Function that sends an Action to the state to be “Processed”

Subscribe: To Listen for a change on the state & do something when it does.

Steps:

1. Create a package.Json

D:\ReactTraining\reduxproject>npm init -y

Wrote to D:\ReactTraining\reduxproject\package.json:

{

"name": "reduxproject",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC"

}

1. Initialize the Redux Libraries **npm i redux -S**

Reduxdemo.js

console.log("From Redux file Demo Message");

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](https://developers.google.com/v8/). Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, [npm](https://www.npmjs.com/), is the largest ecosystem of open source libraries in the world.

NODEMON is the tool that is used to Monitor the Changes

Install – Node Mon

**Npm I nodemon –g**

**To monitor Javascript**

**C:\Users\admin\AppData\Roaming\npm**

**Nodemon <<Js Filename>>**

Create Store is available in Redux Library

Const createStore= redux.createStrore

Step-1 - Define Reducer – Store shd contain Reducer

Use the getstate method to get the state

const createstore = redux.createStore; // Function

let mystore = createstore(mathReducer);

console.log("Initial State of the Strore object --> "+mystore.getState());

const redux = require("redux")

console.log("From Redux file Demo - Added Require");

const createstore = redux.createStore; // Function

console.log("From Redux file Demo - crete Store - step");

console.log("step-1 define Reducer");

//Define Reducer

let mathReducer = (state=0) =>{

//Initialize State Inside the function - Takes State & Action object

//returns teh State

return state

}

//Create a Storre object & Pass teh reducer as store holds the Reducer

let mystore = createstore(mathReducer);

console.log("Initial State of the Strore object --> "+mystore.getState());

//getState Method on the Store

Step-3

//Subscribe to Observe the state changes

mystore.subscribe( () => {

console.log("statechanged "+mystore.getState());

})

Dispatcher shd have a mandatory Property called “type” and does an action

Mystore.dispatch

REDUCER -Action is done on Reducer as it returns an updated State

So reucer now takes 1 more argument caklled action

FROM

let mathReducer = (state=0) =>{

TO:

let mathReducer = (state=0,action) =>{

Eg- reduxdemo.js

const redux = require("redux")

console.log("From Redux file Demo - Added Require");

const createstore = redux.createStore; // Function

console.log("From Redux file Demo - crete Store - step");

console.log("step-1 define Reducer");

//Define Reducer

let mathReducer = (state=0,action) =>{

//Initialize State Inside the function - Takes State & Action object

//returns teh State

//Updates the State by the Ction called Increment

if(action.type=="INCREMENT"){

return state+1;

}

else{

return state

}

}

//Create a Storre object & Pass teh reducer as store holds the Reducer

let mystore = createstore(mathReducer);

console.log("Initial State of the Strore object --> "+mystore.getState());

//getState Method on the Store

//STep-2

//Subscribe to Observe the state changes

mystore.subscribe( () => {

console.log("statechanged "+mystore.getState());

})

mystore.dispatch({ type: "INCREMENT"})

console.log("State Validated");

const redux = require("redux")

console.log("From Redux file Demo - Added Require");

const createstore = redux.createStore; // Function

console.log("From Redux file Demo - crete Store - step");

console.log("step-1 define Reducer");

//Define Reducer

let mathReducer = (state=0,action) =>{

//Initialize State Inside the function - Takes State & Action object

//returns teh State

//Updates the State by the Ction called Increment

if(action.type=="INCREMENT"){

return state+1;

}

else if(action.type =="ADD"){

return state+action.payload;

}

else{

return state

}

}

//Create a Storre object & Pass teh reducer as store holds the Reducer

let mystore = createstore(mathReducer);

console.log("Initial State of the Strore object --> "+mystore.getState());

//getState Method on the Store

//STep-2

//Subscribe to Observe the state changes

mystore.subscribe( () => {

console.log("statechanged "+mystore.getState());

})

mystore.dispatch({ type: "INCREMENT"})

mystore.dispatch({ type: "INCREMENT"})

//payload is a property

//that we have defeined - our ouwn data "payload " to teh action

mystore.dispatch({ type: "ADD",payload:13})

console.log("State Validated");

mystore.dispatch({ type: "INCREMENT"})

mystore.dispatch({ type: "ADD",payload:30})

**ACTION CREATORS**

To Simplify the Dispatch

When we are working on Dispatch , we are working an object which is an Action Object

mystore.dispatch({ type: "ADD",payload:30})

The Dispatch can be defined as an function and return an Action Object

dispatch({ type: "ADD",payload:30})

function increment(){

return {type : “INCREMNT”}

}

Function add(num){

Retun {

{ Type:ADD payload:value}

}

}

Mystore.dispatch(increment())

Mystore.dispatch(add(20))

//Define Action Creators

function increment(){

return {type : "INCREMENT"}

}

function add(num){

return {type: "ADD",payload:num }

}

//spatch the Action

mystore.dispatch(increment())

mystore.dispatch(add(20))

Dispatcher 🡪 goes to Reducers (1 or many) which has the add action 🡪 has one store that updates the State for Us.

# JS Quick Review

Spread Operator – Manages the State in the Array

Myarray =[10,20]

Code Snippet Below

let myarray = [10,20]

//add 30 to array

arr2=[...myarray,30]

console.log(arr2)

OBJECT TYPE OF STATE:

//Passing Object to teh State

const redux = require("redux")

//Store

const createstore = redux.createStore; // Function

let mathReducer = (state = { count: 0 }, action) => {

switch (action.type) {

case "INCREMENT":

return { count: state.count + 1 };

break;

case "ADD":

return { count: (state.count + action.payload) };

break;

default:

return state;

}

}

let store = createstore(mathReducer)

store.subscribe(() => {

console.log("state Changed", store.getState());

})

function increment() {

return { type: "INCREMENT" }

}

function add(num) {

return { type: "ADD", payload: num }

}

//spatch the Action

store.dispatch(increment())

store.dispatch(add(500))

SHOPPING CART RESUMED ----Day 3 Session 1 Continued

1. Presenting List of Products in a Tabular Format

**Table – Classname =table from BootStrap – Look & feel for Table**

**Create tH eproductList.js & CartList.js withtehtable format**

**cartItems.js**

import React, { Component } from "react";

export default class CartItems extends Component {

render() {

return (<div >

<h3>CartItems Listed Here</h3>

<table className="table table-bordered">

<thead>

<tr>

<th>Name</th>

<th>Price</th>

<th>Quantity</th>

<th>Amount</th>

</tr>

</thead>

<tbody>

<tr>

<th colspan="3">Total Amount</th>

</tr>

</tbody>

</table>

</div>)

}

}

import React, { Component } from "react";

export default class ProductList extends Component {

render() {

return (<div>

<h3>List of Products</h3>

<table className="table table-bordered">

<thead>

<tr>

<th>Name</th>

<th>Price</th>

<th>Action</th>

</tr>

</thead>

<tbody>

</tbody>

</table>

</div>)

}

}