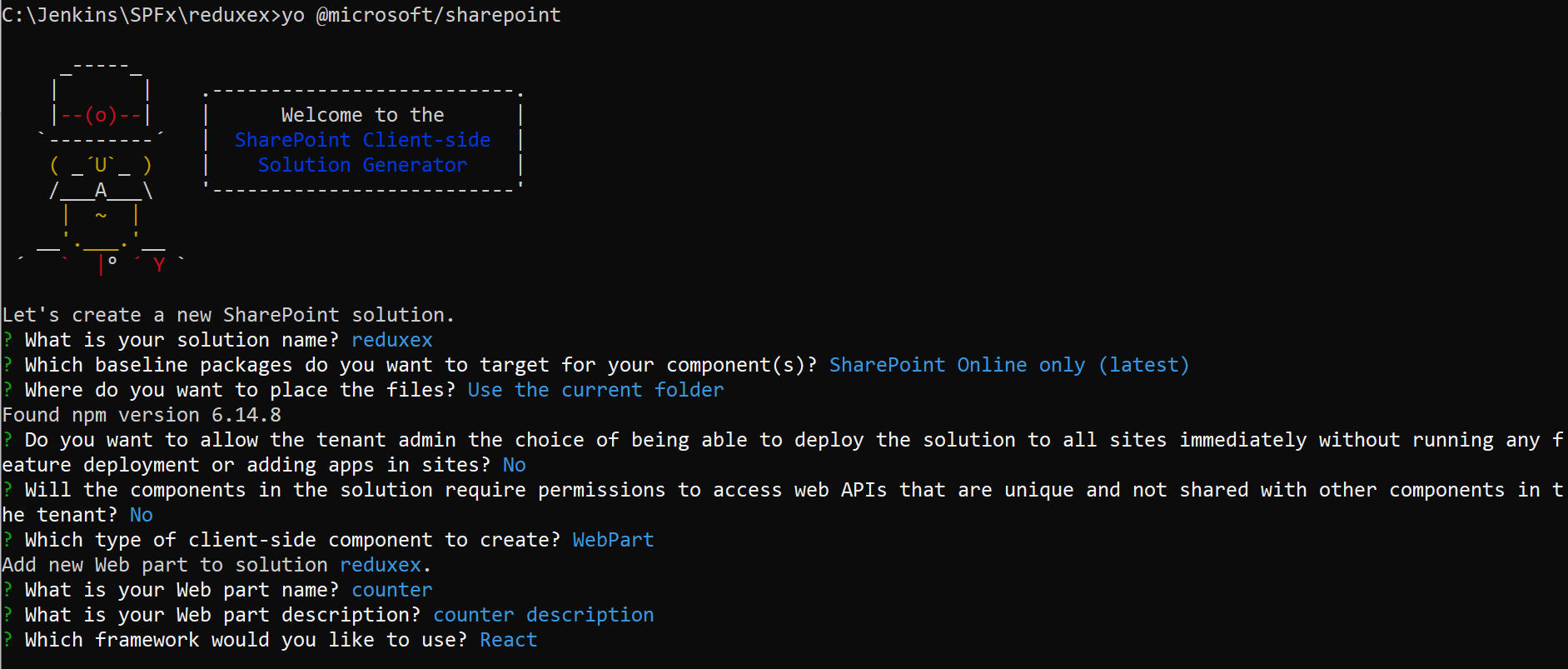
**SPFx – React with Redux**

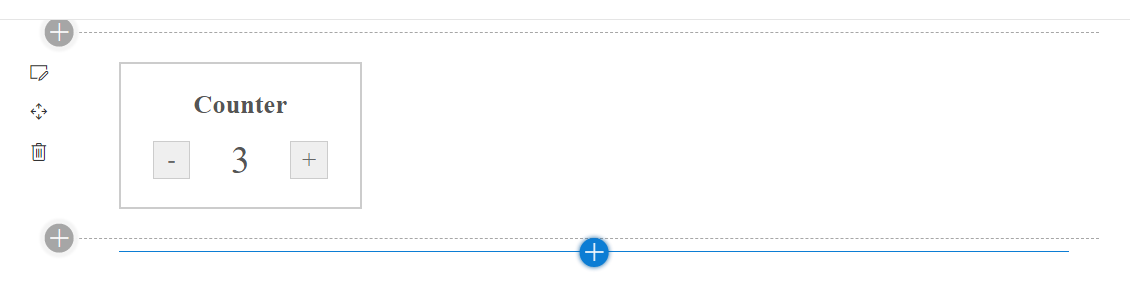
**Create a new SPFx web part project**

* Navigate to your favourite folder
* create new folder
* move to newly created folder
* Open Command prompt and run following comment to create a new web part by running the Yeoman SharePoint Generator
  + - yo @microsoft/sharepoint
* When prompted:
* Enter the webpart name as your solution name, and then select Enter.



Open visual studio code

In this example I am creating counter functionality



First design normal React

Go to Counter.tsx file under components and replace below code

import \* as React from 'react';

import styles from './Counter.module.scss';

import { ICounterProps } from './ICounterProps';

import { ICounterState } from './ICounterState';

import { escape } from '@microsoft/sp-lodash-subset';

import  CounterComp  from './CounterComp';

export default class Counter extends React.Component<ICounterProps, ICounterState, {}> {

  public render(): React.ReactElement<ICounterProps> {

    return (

      <div>

        <CounterComp></CounterComp>

      </div>

    );

  }

}

Create a new file CounterComp.tsx and paste below code

import \* as React from 'react';

import styles from './Counter.module.scss';

import { ICounterProps } from './ICounterProps';

import { ICounterState } from './ICounterState';

export default class CounterComp extends React.Component<ICounterProps,ICounterState, {}> {

  constructor(props:ICounterProps, state:ICounterState)

  {

    super(props);

    this.state={

      count: 0

    };

  }

  private increment = () => {

    this.setState({

        count: this.state.count + 1

    });

  }

  private decrement = () => {

    this.setState({

      count: this.state.count - 1

    });

  }

  public render(): React.ReactElement<ICounterProps> {

    return (

        <div className={styles.counter}>

        <h2>Counter</h2>

        <div>

          <button onClick={this.decrement}>-</button>

          <span>{this.state.count}</span>

          <button onClick={this.increment}>+</button>

        </div>

      </div>

    );

  }

}

Then create a ICounterState.ts file, ADD the below code

export interface ICounterState{

  count:number;

}

Then open ICounterProps.ts and change the description as optional

description?: string;

Then replace the Counter.module.scss file with below css class

@import '~office-ui-fabric-react/dist/sass/References.scss';

.counter {

    border: 2px solid #ccc;

    padding: 20px;

    width: 150px;

    text-align: center;

    font-family: "Source Sans Pro";

    color: #555;

}

.counter h2 {

    margin: 0;

    margin-bottom: 15px;

    font-weight: 600;

}

.counter>div {

    display: flex;

    align-items: center;

    justify-content: center;

}

.counter>div>button+span,

.count {

    margin: 0 20px;

    font-size: 30px;

    flex-basis: 40px;

    flex-grow: 0;

}

.counter button {

    border: 1px solid #ccc;

    color: #666;

    font-size: 20px;

    line-height: 0;

    height: 30px;

    width: 30px;

    transition: 0.2s all;

}

.counter button:active,

.counter button:focus {

    outline: none;

    box-shadow: 0 0 2px rgb(87, 156, 87);

}

.counter button:active {

    background-color: rgb(87, 156, 87);

    border-color: rgb(35, 141, 35);

    color: #fff;

}

Then make the react state with redux store

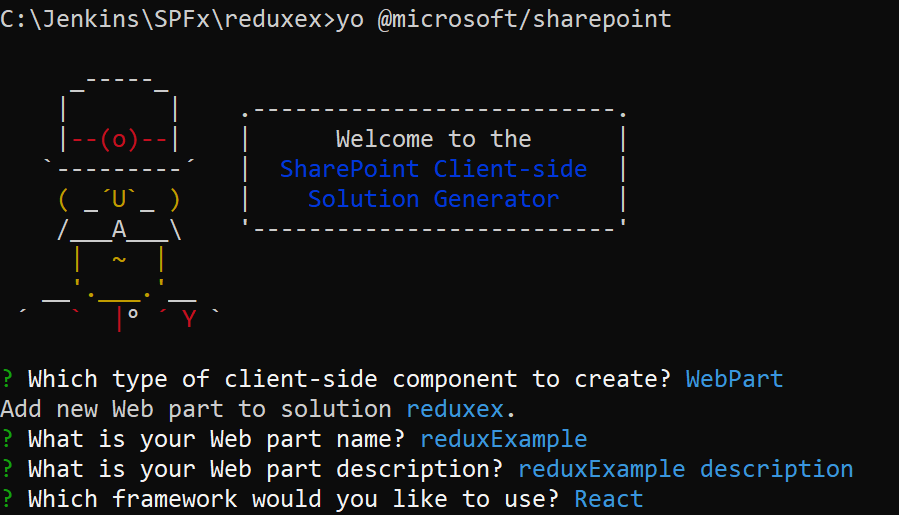
Create the Redux Store

Install the library and required dependencies

* npm install redux
* npm install react-redux
* redux gives you a store, and lets you keep state in it, and get state out, and respond when the state changes. But that is all it does.
* It is react-redux that lets you connect pieces of the state to React components.

Create a new Webpart Redux example in the same solution

* Open Command prompt and run following comment to create a new web part by running the Yeoman SharePoint Generator
  + - yo @microsoft/sharepoint



1. Create action - Create a file Action.ts

export enum actionTypes{

  INCREMENT,

  DECREMENT

}

export interface IAction{

  type: actionTypes;

}

export const increment = ():IAction => {

 return {

  type: actionTypes.INCREMENT

 };

};

export const decrement = ():IAction => {

  return {

    type: actionTypes.DECREMENT

   };

};

1. Create Applicate State - Create a file IApplicationState.ts

export interface IApplicationState{

  count:number;

}

1. Update the properties IReduxExampleProps.ts

import { Store } from 'redux';

import { IApplicationState } from './IApplicationState';

export interface IReduxExampleProps {

  store:Store<IApplicationState>;

}

1. Create a Reducer.ts file to handle the action

import { actionTypes, IAction } from './Action';

import {IApplicationState} from './IApplicationState';

const initialState: IApplicationState = {

  count: 0,

};

export default (state: IApplicationState = initialState, action: IAction) => {

  switch(action.type) {

    case actionTypes.INCREMENT:

      return {

        count: state.count + 1

      };

    case actionTypes.DECREMENT:

      return {

        count: state.count - 1

      };

    default:

      return state;

  }

};

1. Update the react main component file **ReduxExample.tsx**

import \* as React from 'react';

import styles from './ReduxExample.module.scss';

import { IReduxExampleProps } from './IReduxExampleProps';

import { escape } from '@microsoft/sp-lodash-subset';

import { IApplicationState } from './IApplicationState';

import { increment, decrement } from './Action';

import Newcomponent from './Newcomponent';

export default class ReduxExample extends React.Component<IReduxExampleProps, IApplicationState, {}> {

private store = this.props.store;

public render(): React.ReactElement<IReduxExampleProps> {

return (

<div>

<div className={styles.counter}>

<h2>Counter</h2>

<div>

<button onClick={() => { this.store.dispatch(decrement()); }}>-</button>

<span>{this.store.getState().count}</span>

<button onClick={() => { this.store.dispatch(increment()); }}>+</button>

</div>

</div>

<Newcomponent store={this.store}></Newcomponent>

</div>

);

}

}

1. Update the scss file ReduxExample.module.scss

@import '~office-ui-fabric-react/dist/sass/References.scss';

.counter {

    border: 2px solid #ccc;

    padding: 20px;

    width: 150px;

    text-align: center;

    font-family: "Source Sans Pro";

    color: #555;

}

.counter h2 {

    margin: 0;

    margin-bottom: 15px;

    font-weight: 600;

}

.counter>div {

    display: flex;

    align-items: center;

    justify-content: center;

}

.counter>div>button+span,

.count {

    margin: 0 20px;

    font-size: 30px;

    flex-basis: 40px;

    flex-grow: 0;

}

.counter button {

    border: 1px solid #ccc;

    color: #666;

    font-size: 20px;

    line-height: 0;

    height: 30px;

    width: 30px;

    transition: 0.2s all;

}

.counter button:active,

.counter button:focus {

    outline: none;

    box-shadow: 0 0 2px rgb(87, 156, 87);

}

.counter button:active {

    background-color: rgb(87, 156, 87);

    border-color: rgb(35, 141, 35);

    color: #fff;

}

1. Go to Main webpart file - ReduxExampleWebPart.ts

First import below classes

import { IReduxExampleProps } from './components/IReduxExampleProps';

import { IApplicationState } from './components/IApplicationState';

import { createStore, Store } from 'redux';

import reducer from './components/Reducer';

Add below code in webpart class

private store: Store<IApplicationState>;

  protected onInit(): Promise<void> {

    //create the store instance

    this.store = createStore(reducer);

    this.store.subscribe(this.render);

    this.getStore = this.getStore.bind(this);

    return super.onInit();

  }

  private getStore(): Store<IApplicationState>{

    return this.store;

  }

Update the render method

public render(): void {

    const element: React.ReactElement<IReduxExampleProps> = React.createElement(

      ReduxExample,

      {

        store: this.getStore()

      }

    );

    ReactDom.render(element, this.domElement);

  }

Finally create new react component to validate the changes – Newcomponent.tsx

import \* as React from 'react';

import styles from './ReduxExample.module.scss';

import { IReduxExampleProps } from './IReduxExampleProps';

import { escape } from '@microsoft/sp-lodash-subset';

import { IApplicationState } from './IApplicationState';

import { increment, decrement } from './Action';

export default class Newcomponent extends React.Component<IReduxExampleProps, IApplicationState, {}> {

  private store = this.props.store;

  public render(): React.ReactElement<IReduxExampleProps> {

    return (

      <div>

        <div>New Component executed</div>

        <h2>Counter Value : {this.store.getState().count} </h2>

      </div>

    );

  }

}

**How the above code works**

Whenever user access the webpart, it creates a store and access the reducer 🡪 reducer initiate the action.

While user click the action from react component, it gets the store information from the redux store.

