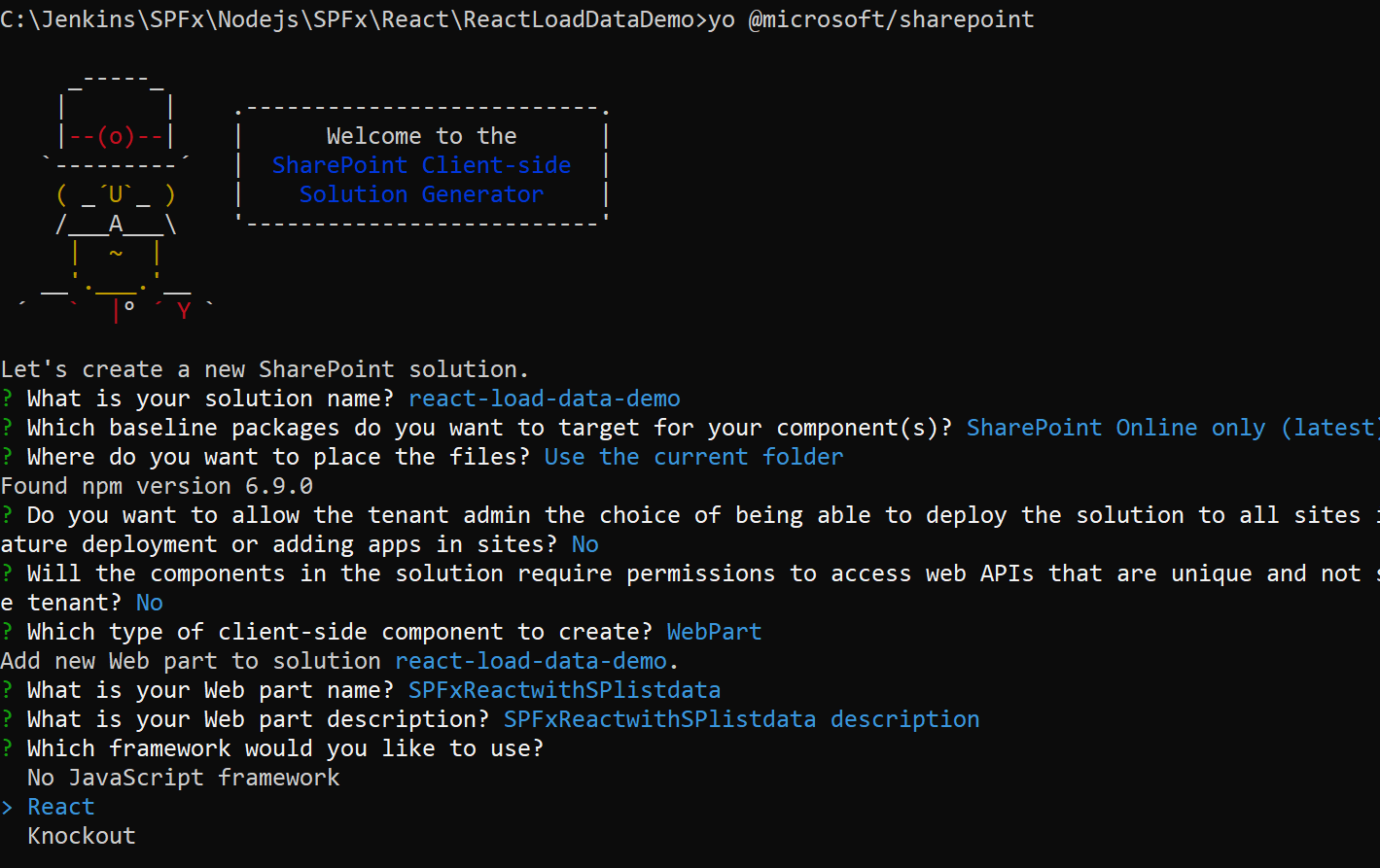
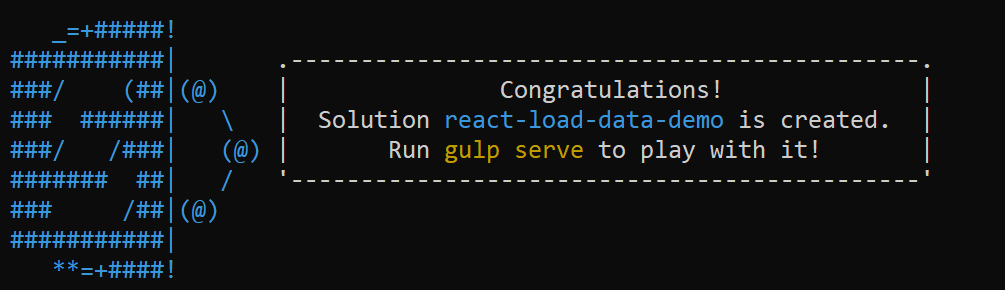
**Lab 14 - Create SPFx webpart using React and load data from SharePoint**

**Example 1 : Create a react webpart and add components**

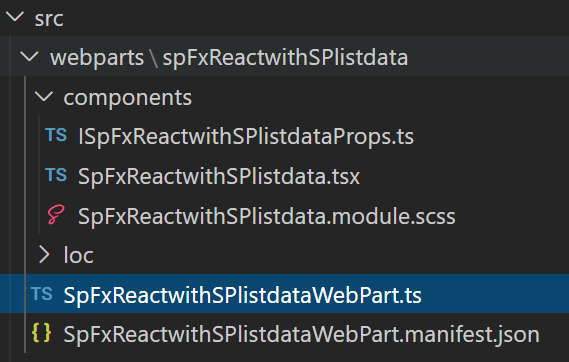
1. Create a folder “**ReactLoadDataDemo**” under your favourite folder
2. Move to the folder “**ReactLoadDataDemo**”
3. Give webpart name as “**SPFxReactwithSPlistdata**”
4. Create a webpart same as previous example, and select the framework as React



1. Enter to create new solution



1. Open the solution in Visual Studio Code
2. Type **Code .**



1. Create new file **IProducts.ts** to create an interface under component folder and paste below code

export interface IProducts{

  id:number;

  Title:string;

}

1. Create another file **ProductList.tsx** to create a component under component folder and paste below code.
   * Imported **react** and **IProducts** interface
   * Create a public interface **IProductListProps**
   * Create a class **ProductList** to render the component

import \* as React from 'react';

import {IProducts} from './IProducts';

export interface IProductListProps{

  products:IProducts[];

}

export class ProductList extends React.Component<IProductListProps,{}>

{

  public render():React.ReactElement<IProductListProps>

  {

    return(<ul>

      {

        this.props.products.map(productItem => (

        <li>{productItem.Title}</li>

      ))

      }

      </ul>

      );

  }

}

1. Go to SpFxReactwithSPlistdata.tsx file.

Add below code to import the component and interfaces

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

import { ProductList, IProductListProps } from './components/ProductList';

Add the private member inside the class to initial the products

private \_products:IProducts[]= [

    {id:1,Title:"Laptop"},

    {id:2,Title:"Mobile"},

    {id:3,Title:"Phone"},

    {id:4,Title:"HeadPhone"},

    {id:5,Title:"Desktop"},

  ];

1. Then update the render method

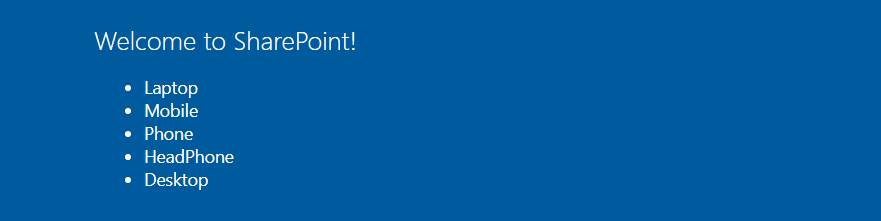
<div className={ styles.column }>

              <span className={ styles.title }>Welcome to SharePoint!</span>

              <ProductList products={this.\_products}/>

            </div>

1. Output



**Example 2 : add button event to the list of Products**

1. Open Component **ProductsList.tsx**
2. Add a button in a render method, inside the <li> tag

<button type="button" id={product.id.toString() } onClick={ ()=>this.OnButtonClick(product)}>click</button>

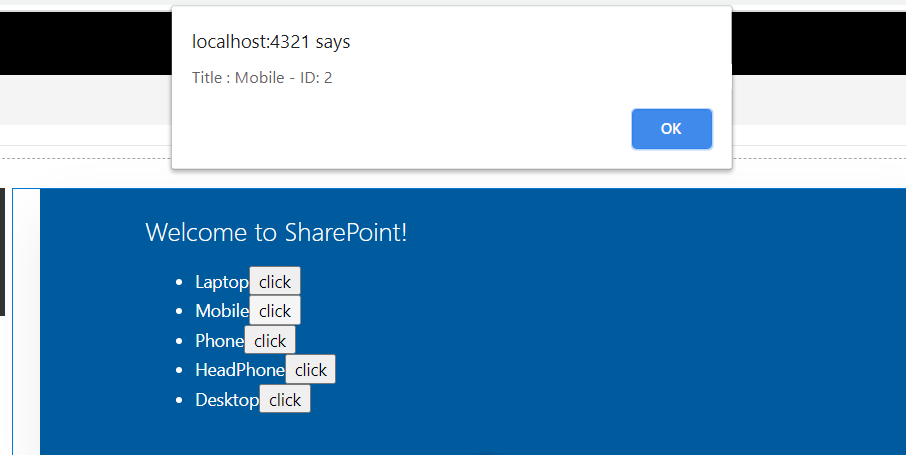
1. Add on private method to execute the button event

private OnButtonClick(product:IProducts):void{

    alert("Title : " + product.Title + " - ID: " + product.id);

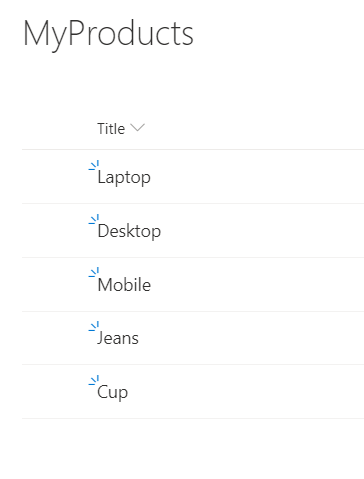
  }

**Output**



**Example 3 : Bind SharePoint Data**

1. Create a SharePoint List “**MyProducts**”
2. Add some products



1. Go back to Visual Studio Code
2. Create a state interface, for that create a file **ISpFxReactwithSPlistdataState.ts** under components folder, then copy the below code

import {IProducts} from './IProducts';

export interface ISpFxReactwithSPlistdataState{

  products:IProducts[];

}

1. Go to SpFxReactwithSPlistdata.tsx file
2. Import **ISpFxReactwithSPlistdataState** interface

import {ISpFxReactwithSPlistdataState} from './ISpFxReactwithSPlistdataState';

1. Add **SpFxReactwithSPlistdataState** React.Component type

export default class SpFxReactwithSPlistdata extends React.Component<ISpFxReactwithSPlistdataProps,ISpFxReactwithSPlistdataState, {}> {

1. Create constructor in **SpFxReactwithSPlistdata.tsx file** and add Header

import { SPHttpClient, SPHttpClientResponse } from '@microsoft/sp-http';

constructor(props:ISpFxReactwithSPlistdataProps){

    super(props);

    this.state= {products:[]};

  }

1. Add a private method to call the SharePoint list

private getProductsFromSpList():Promise<IProducts[]>{

    return new Promise<IProducts[]>((resolve,reject) => {

      const endpoint: string = `${this.props.currentsiteURL}/\_api/lists/getbytitle('MyProducts')/items?$select=Id,Title`;

      this.props.spHttpClient.get(endpoint,SPHttpClient.configurations.v1)

      .then((response:SPHttpClientResponse)=>{

        return response.json();

      })

      .then((jsonResponse:any)=>{

        let splistItemProducts:IProducts[]=[];

        for(let index=0;index<jsonResponse.value.length; index++){

          splistItemProducts.push({

            id: jsonResponse.value[index].Id,

            Title: jsonResponse.value[index].Title,

          });

          resolve(splistItemProducts);

        }

      });

    });

  }

1. To do that, we need SPHttpClient and Websiteurl
2. Open Webpart file **SpFxReactwithSPlistdataWebPart.ts**
3. Add two parameter to pass url and sphttpclient property, for this add the yellow highlighted codes in the render method

public render(): void {

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

      SpFxReactwithSPlistdata,

      {

        description: this.properties.description,

        spHttpClient:this.context.spHttpClient,

        currentsiteURL:this.context.pageContext.web.absoluteUrl

      }

    );

    ReactDom.render(element, this.domElement);

  }

1. Open props interface file **ISpFxReactwithSPlistdataProps**.ts and add header

import {SPHttpClient} from '@microsoft/sp-http';

Add two properties

export interface ISpFxReactwithSPlistdataProps {

  description: string;

  spHttpClient:SPHttpClient;

  currentsiteURL:string;

}

1. Go to **SpFxReactwithSPlistdata**.tsx file
2. Then execute the get product from list method after render, add below code

public componentWillMount():void{

    this.getProductsFromSpList()

    .then((splistproducts:IProducts[]) =>{

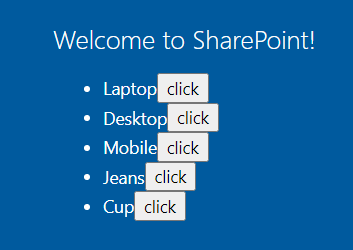
      this.setState({products:splistproducts});

    });

  }

1. Update the ProductList tag in render method

<ProductList products={this.state.products}/>



**To remove the record from the array**

1. Go to ProductList.tsx file and add a type

export type RemoveProductCallback = (products:IProducts)=>void;

1. Update the render method button control

<button type="button" id={product.id.toString()} onClick={ ()=>this.OnButtonClick(product)}>Delete</button>

1. Update Interface

export interface IProductListProps{

  products:IProducts[];

  onRemoveProduct:RemoveProductCallback;

}

1. Update OnButtonClick method

private OnButtonClick(product:IProducts):void{

    //alert("Title : " + product.Title + " - ID: " + product.id);

    this.props.onRemoveProduct(product);

  }

1. Go to **SpFxReactwithSPlistdata**.tsx file
2. Update the <**Productlist** control

<ProductList products={this.state.products} onRemoveProduct={this.\_removeProduct}/>

1. Add \_removeProduct event

private \_removeProduct = (productToRemove:IProducts):void=>{

    const newProducts = this.state.products.filter(product => product != productToRemove);

    this.setState({products:newProducts});

  }

**To remove the record permanent from SP list**

1. Add below code in \_removeProduct method

this.deleteProductFromSPList(productToRemove.id);

1. Create new private method to delete the item from SPList

private deleteProductFromSPList(id:number):Promise<IProducts[]>{

      return new Promise<IProducts[]>((resolve,reject) => {

        const endpoint: string = `${this.props.currentsiteURL}/\_api/lists/getbytitle('MyProducts')/items(${id})`;

        this.props.spHttpClient.post(endpoint,SPHttpClient.configurations.v1,{

          headers: {

            'Accept': 'application/json;odata=nometadata',

            'Content-type': 'application/json;odata=verbose',

            'odata-version': '',

            'IF-MATCH': '\*',

            'X-HTTP-Method': 'DELETE'

          }

        })

        .then((response:SPHttpClientResponse):void =>{

          alert(`Item with ID: ${id} successfully Deleted`);

        },(error: any): void => {

          alert(`${error}`);

        });

      });

  }

1. Modify the OnButtonClick button in ProductsList.tsx

private OnButtonClick(product:IProducts):void{

    //alert("Title : " + product.Title + " - ID: " + product.id);

    if (window.confirm('Are you sure you want to delete the latest item?')) {

      this.props.onRemoveProduct(product);

     }

  }

Test it