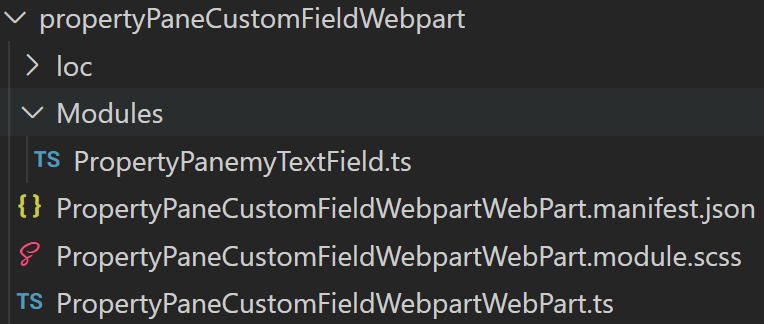
Create Custom Property Pane using Typescript for SPfx

1. Create a WebPart using yo @microsoft/sharepoint
2. Example Webpart Name : PropertyPaneCustomFieldWebpartWebPart
3. Add a folder “Modules” under webpart folder
4. Add a file names “PropertyPanemyTextField.ts”



1. To create custom control, import
   1. IPropertyPaneField
   2. IPropertyPaneCustomFieldProps
   3. PropertyPaneFieldType
2. Add below code in the file

import { IPropertyPaneField, IPropertyPaneCustomFieldProps, PropertyPaneFieldType } from '@microsoft/sp-property-pane';

1. Create an interface to set the properties

export interface PropertyPanemyTextFieldProps {

properties:any;

label:string;

description?:string;

color?:string;

backgroundcolor?:string;

defaultvalue?:string;

}

1. ? means optional property
2. Create a class and implements IpropertyPaneField
3. Declare public variable from IPropertyPaneCustomFieldProps
   1. Type
   2. targetProperty
   3. Properties
   4. Config
   5. CurrentValue – to manintain the control value
4. Create a constructor to initialize the methods onRender and OnDispose

this.targetProperty = targetProperty;

this.properties = {

key: "MyCustomControl",

context: context,

onRender : this.render.bind(this),

onDispose : this.dispose.bind(this)

};

this.config = config;

1. Call the render method
2. Build the render method to bring your custom control
3. addEvents method to add textbox event to get the new value
4. applychanges method to apply the changes and return back to webpart

export class PropertyPanemyTextField implements IPropertyPaneField<IPropertyPaneCustomFieldProps>{

  public type: any = PropertyPaneFieldType.Custom;

  public targetProperty:string;

  public properties: IPropertyPaneCustomFieldProps;

  private config : PropertyPanemyTextFieldProps;

  private currentValue:string = "";

  constructor(targetProperty:string,

    config : PropertyPanemyTextFieldProps,

    context?:any)

    {

    this.targetProperty = targetProperty;

    this.properties = {

      key: "MyCustomControl",

      context: context,

      onRender : this.render.bind(this),

      onDispose : this.dispose.bind(this)

    };

    this.config = config;

  }

  private render(element:HTMLElement,

    context:any,

    changeCallback:(targetProperty:string, newValue:any)=>void)

  {

    this.currentValue = this.config.properties[this.targetProperty];

    let html =

    `<div style="background-color:${this.config.backgroundcolor}">

      <div class="ms-TextField">

      <div class="ms-TextField-wrapper" style="padding:5px">

        <label class="ms-Label" style="color:${this.config.color || "gray"};">${this.config.label}</label>

        <div class="ms-TextField-fieldGroup" style="padding:5px">

          <input class="ms-TextField-field" style="width:100%" type="text" value="${this.currentValue || this.config.defaultvalue}">

        </div>

      <div>

      <span class="description">${this.config.description||""}</span>

      </div>

    </div>`;

    element.innerHTML = html;

    this.addEvents(element, changeCallback);

  }

  private addEvents(element:HTMLElement, callback:(targetProperty:string, newValue:any)=>void ){

    let inputTextElement : HTMLInputElement = element.getElementsByTagName('input')[0];

    inputTextElement.onchange=()=>{

      this.applyChanges(element, inputTextElement, callback);

    };

}

private applyChanges(element: HTMLElement, inputTextElement : HTMLInputElement, callback:(targetProperty:string, newValue:any)=>void ){

  let newValue = inputTextElement.value;

  callback(this.targetProperty, newValue);

}

  private dispose(element:HTMLElement){

   element.innerHTML="";

  }

}