**Lab 9 – Create a Web Part with all Properties**

1. To create a new web part project

* Create a new project directory in your favourite location.
* Open command prompt
  + Run as administrator

***md spfxPropertyPane***

1. Go to the project directory

*cd* ***spfxPropertyPane***

1. Create a new ***spfxPropertyPane*** by running the Yeoman SharePoint Generator.

yo @microsoft/sharepoint

**When prompted**:

* Accept the default **spfx-property-pane** as your solution name, and then select Enter.

**The next set of prompts ask for specific information about your web part:**

* Enter **OobPropertyPane** as your web part name, and then select Enter.
* Accept the default OobPropertyPane description as your web part description, and then select Enter.
* Accept the default No JavaScript web framework as the framework you would like to use, and then select Enter

Text

Description automatically generated

Open the solution using visual studio code using **code .**

This webpart we are going to add all out of box properties.

* Import all control module
* Replace below code

import {

  IPropertyPaneConfiguration,

  PropertyPaneTextField

} from '@microsoft/sp-property-pane';

To

import {

  IPropertyPaneConfiguration,

  PropertyPaneLabel,

  PropertyPaneTextField,

  PropertyPaneLink,

  PropertyPaneDropdown,

  PropertyPaneCheckbox,

  PropertyPaneChoiceGroup,

  PropertyPaneButton,

  PropertyPaneButtonType,

  PropertyPaneHorizontalRule,

  PropertyPaneSlider,

  PropertyPaneToggle,

  IPropertyPaneDropdownOption

} from '@microsoft/sp-property-pane';

Then replace the properties

export interface IOobPropertyPaneWebPartProps {

  description: string;

}

To

export interface IOobPropertyPaneWebPartProps {

  description: string;

  lbl\_label: string;

  txt\_name: string;

  txt\_multiline: string;

  url: string;

  dropdown1: string;

  checkbox1: string;

  slider1: string;

  toggle1: string;

  btn\_normal: string;

}

* Replace the render method with below code display the values entered in the properties.

public render(): void {

    this.domElement.innerHTML = `

      <div class="${ styles.oobPropertyPane}">

    <div class="${ styles.container}">

      <div class="${ styles.row}">

        <div class="${ styles.column}">

          <p>Name : ${escape(this.properties.txt\_name)}</p>

          <p>Desc : ${escape(this.properties.txt\_multiline)}</p>

          <p>Drop Down Value : ${escape(this.properties.dropdown1)}</p>

          <p>Check box 1 : ${escape(this.properties.checkbox1)}</p>

          <p>slider  : ${escape(this.properties.slider1)}</p>

          <p>toggle : ${escape(this.properties.toggle1)}</p>

          </div>

          </div>

          </div>

          </div>`;

  }

Then replace the getPropertyPaneConfiguration method

protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {

    let getOptions: Array<IPropertyPaneDropdownOption> = this.loadoptions();

    return {

      pages: [

        {

          header: {

            description: "Page 1"

          },

          displayGroupsAsAccordion: true,

          groups: [

            {

              groupName: "Group 1",

              isCollapsed: false,

              groupFields: [

                PropertyPaneLabel('lbl\_label', {

                  text: "Label - text",

                  required: true

                }),

                PropertyPaneTextField('txt\_name', {

                  label: "Enter Name",

                  description: "Please enter Name",

                  disabled: false,

                  maxLength: 25,

                  multiline: false,

                  placeholder: "Enter the name",

                  resizable: false,

                  value: "",

                  onGetErrorMessage: this.validatetextcontrol.bind(this)

                }),

                PropertyPaneTextField('txt\_multiline', {

                  label: "Please Enter Description",

                  multiline: true,

                  description: "Enter Description"

                })

              ]

            },

            {

              groupName: "Group 2",

              isCollapsed: false,

              groupFields: [

                PropertyPaneLink('url', {

                  text: 'My Blog Page',

                  href: "http://jenkinsblogs.com",

                  target: "\_blank"

                }),

                PropertyPaneHorizontalRule(),

                PropertyPaneDropdown('dropdown1', {

                  label: "Select the option",

                  options: getOptions,

                  selectedKey: "k3"

                }),

                PropertyPaneCheckbox('checkbox1', {

                  text: "Yes/No",

                  checked: false,

                })

              ]

            }

          ]

        },

        {

          header: {

            description: "Page 2"

          },

          groups: [

            {

              groupName: "Group 1 - P2",

              groupFields: [

                PropertyPaneSlider('slider1', {

                  label: "set value",

                  min: 1,

                  max: 100,

                  value: 25,

                  showValue: true

                }),

                PropertyPaneToggle('toggle1', {

                  label: "Select Yes or No",

                  checked: false,

                  key: "toggle1",

                  offText: "No",

                  onText: "Yes"

                })

              ]

            },

            {

              groupName: "Group 2 - P2",

              groupFields: [

                PropertyPaneButton('btn\_normal', {

                  text: "Normal Button",

                  buttonType: PropertyPaneButtonType.Normal,

                  onClick: this.ButtonClick\_Event,

                  description: "Normal Button"

                }),

                PropertyPaneButton('btn\_primary', {

                  text: "Primary Button",

                  buttonType: PropertyPaneButtonType.Primary,

                  onClick: this.ButtonClick\_Event1,

                  description: "Primary Button"

                }),

                PropertyPaneButton('btn\_compound', {

                  text: "Compound Button",

                  buttonType: PropertyPaneButtonType.Compound,

                  onClick: this.ButtonClick\_Event,

                  description: "Compound Button Desc"

                }),

                PropertyPaneButton('btn\_command', {

                  text: "command Button",

                  buttonType: PropertyPaneButtonType.Command,

                  onClick: this.ButtonClick\_Event,

                  description: "command Button Desc"

                }),

                PropertyPaneButton('icon\_button', {

                  text: "Add Button",

                  buttonType: PropertyPaneButtonType.Icon,

                  onClick: this.ButtonClick\_Event,

                  icon: 'PeopleAdd'

                }),

                PropertyPaneButton('icon\_button', {

                  text: "Add Button",

                  buttonType: PropertyPaneButtonType.Hero,

                  onClick: this.ButtonClick\_Event,

                  icon: 'Add'

                })

              ]

            }

          ]

        }

      ]

    };

  }

Add below method above this getPropertyPaneConfiguration method

private validatetextcontrol(value: string): string {

    if (value === null || value.trim().length === 0) {

      return " Please enter the value";

    }

    else if (value.length > 15) {

      return " Length should be less than 15";

    }

  }

  private ButtonClick\_Event(){

    alert("Button Click Succeeded");

  }

  private ButtonClick\_Event1() {

    alert("Button1 Click Succeeded");

  }

  private loadoptions(): Array<IPropertyPaneDropdownOption> {

    let options: Array<IPropertyPaneDropdownOption> = new Array<IPropertyPaneDropdownOption>();

    options.push({ key: "k1", text: "text 1" });

    options.push({ key: "k2", text: "text 2" });

    options.push({ key: "k3", text: "text 3" });

    return options;

  }