SharePoint Framework Deep Dive - Web Part Property Pane

Rob Windsor
rob@robwindsor.com
@robwindsor

About Me

Rob Windsor

.NET/Microsoft 365 developer, trainer, author
Microsoft MVP | Microsoft 365 Development

Twitter: https://twitter.com/robwindsor
LinkedIn: https://www.linkedin.com/in/rwindsor
Blog: https://robwindsor.hashnode.dev
YouTube: https://www.youtube.com/@RobWindsor
GitHub: https://github.com/rob-windsor

Overview

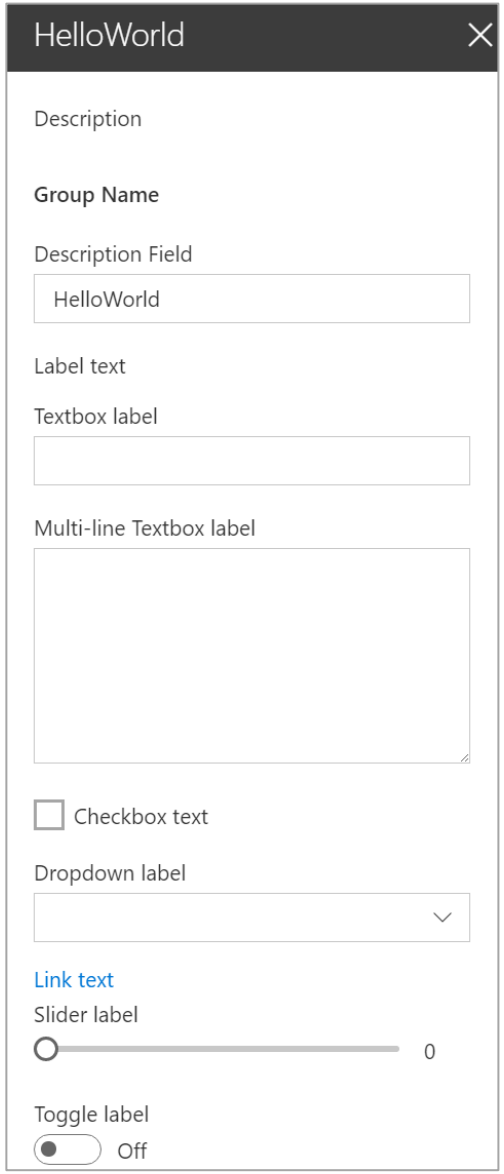
The property pane has three key elements

- **Pages, Headers, Groups**

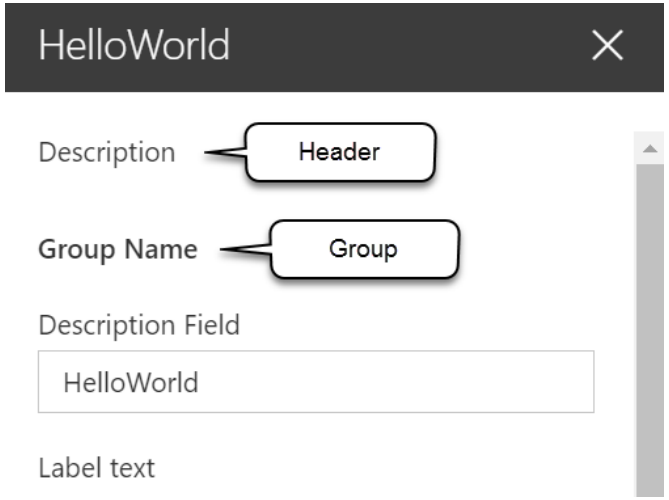
Property panes ***must*** contain a **page** and at least one group, the header is optional

The property pane supports the following field types

- Label
- Textbox
- Multi-line Textbox
- Checkbox
- Dropdown
- Link
- Slider
- Toggle
- Custom



Implementing the header, groups, and fields




```
protected get getPropertyPaneConfiguration(): IPropertyPaneConfiguration {
     return {
       pages: [
         {
           header: {
             description: strings.PropertyPaneDescription
           },
           groups: [
             {
               groupName: strings.BasicGroupName,
               groupFields: [
                 PropertyPaneTextField('description', {
                   label: strings.DescriptionFieldLabel
                 }),
                 PropertyPaneLabel('labelField', {
                   text: 'Label text'
                 })
               ]
             }
           ]
         }
       ]
     };
  }
```

Implementing properties

Define an interface in your web part that includes one or more target properties

Import the corresponding field types in the web part class

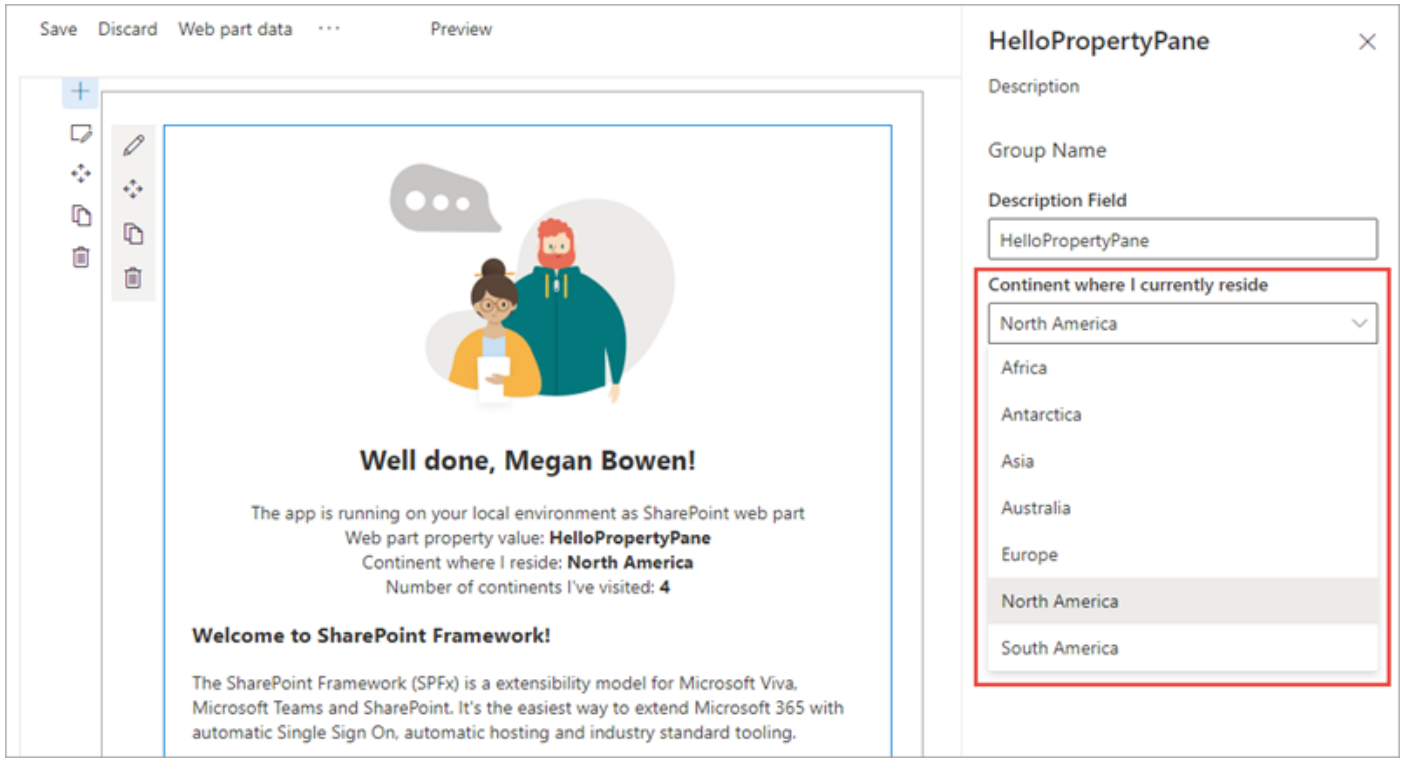
Field types are available as modules in the `@microsoft/sp-property-pane` library

Modify the default `getPropertyPaneConfiguration` method and add the properties to the `groupFields` array

Native property pane fields

Property pane supports the following field types out-of-the-box

- Label
- Textbox
- Multi-line Textbox
- Checkbox
- Dropdown
- Link
- Slider
- Toggle
- Custom



Also possible to create your own custom field types

Adding configuration properties to React web parts

Modify the `propertyPaneSettings` method and add the property to the `groupFields` array

```
protected get propertyPaneSettings(): IPropertyPaneSettings {
     return {
       pages: [
         {
           header: {
             description: strings.PropertyPaneDescription
           },
           groups: [
             {
               groupName: strings.BasicGroupName,
               groupFields: [
                 PropertyPaneTextField('description', {
                   label: strings.DescriptionFieldLabel
                 }),
```

Adding configuration properties to React web parts

Pass the property value to React component when the React component is created

```
public render(): void {
      const element: React.ReactElement<IHelloWorldWebPartProps> = React.createElement(HelloWorldComponent, {
          description: this.properties.description
      });

      ReactDom.render(element, this.domElement);
}
```

Adding configuration properties to React web parts

Use the property value in the React component

```
export default class HelloPropertyPane extends React.Component<IHelloPropertyPaneProps> {
  public render(): React.ReactElement<IHelloPropertyPaneProps> {
    const {
      description,
      isDarkTheme,
      environmentMessage,
      hasTeamsContext,
      userDisplayName
    } = this.props;

    return (
      <section className={`${styles.helloPropertyPane} ${hasTeamsContext ? styles.teams : ''}`}>
        <div className={styles.welcome}>
          <h2>Well done, {escape(userDisplayName)}!</h2>
          <div>{environmentMessage}</div>
          <div>Description property value: <strong>{escape(description)}</strong></div>
        </div>
      </section>
    );
  }
}
```

Property validation

Add method that implements validation of property value(s)

```
private validateDescription(value: string): string {
  let result = "";

  if (value == null || value.trim().length === 0) {
    result = "Please enter a description";
  }

  return result;
}
```

Property validation

Bind the validation method to the `onGetErrorMessage` property of the field

```
protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {
  return {
    pages: [
      {
        header: {
          description: strings.PropertyPaneDescription
        },
        groups: [
          {
            groupName: strings.BasicGroupName,
            groupFields: [
              PropertyPaneTextField('description', {
                label: strings.DescriptionFieldLabel,
                onGetErrorMessage: this.validateDescription.bind(this)
              }),
```

Handling property field changes

The property pane has two interaction modes:

- Reactive
- Non-reactive

Reactive mode: every change = change event is triggered

- Reactive behavior automatically updates the web part user interface with the new property field values

Non-reactive mode: does not update the web part user interface automatically unless the user confirms the changes

- While reactive mode is sufficient for many scenarios, at times you will need non-reactive behavior.

Property pane modes

Default mode = reactive mode

Override the default behavior by adding the `disableReactivePropertyChanges` method to the web part class

```
protected get disableReactivePropertyChanges(): boolean {
  return true;
}
```

Dynamically populate property pane dropdown

Add a method to get that will populate the dropdown in the web part class

```
private async loadLists(): Promise<IPropertyPaneDropdownOption[]> {
  const restUrl = this.context.pageContext.web.absoluteUrl + "/_api/web/lists?$filter=(Hidden eq false)";

  let result: IPropertyPaneDropdownOption[] = [];
  try {
    const response = await this.context.spHttpClient.get(restUrl, SPHttpClient.configurations.v1);
    if (response.ok) {
      const data = await response.json();
      result = data.value.map((list: any) => {
        return { key: list.Id, text: list.Title };
      });
    }
  } catch (ex) {
    console.log(ex);
  }

  return result;
}
```

Dynamically populate property pane dropdown

Add class-level variables to store the options to be shown in the dropdown in the web part class

Add override of the `onPropertyPaneConfigurationStart` method in the web part class

```
private lists: IPropertyPaneDropdownOption[] | undefined = undefined;
private listsDropdownDisabled: boolean = true;

protected onPropertyPaneConfigurationStart(): void {
  if (this.lists) {
    return;
  }

  this.listsDropdownDisabled = true;

  this.loadLists()
    .then((listOptions: IPropertyPaneDropdownOption[]): void => {
      this.lists = listOptions;
      this.listsDropdownDisabled = false;
      this.context.propertyPane.refresh();
      this.render();
    });
}
```

Dynamically populate property pane dropdown

Set dropdown properties to values of class-level variables created earlier

```
protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {
  return {
    pages: [
      {
        header: {
          description: strings.PropertyPaneDescription
        },
        groups: [
          {
            groupName: strings.BasicGroupName,
            groupFields: [
              PropertyPaneDropdown("list", {
                label: "List",
                options: this.lists,
                disabled: this.listsDropdownDisabled
              })              ]
          }
        ]
      }
    ]
  };
}
```

Handling property value changes

Add override of the `onPropertyPaneFieldChanged` method in the web part class

Also override `onAfterPropertyPaneChangesApplied` method in the web part class if it's in non-reactive mode

```
protected onPropertyPaneFieldChanged(propertyPath: string, oldValue: any, newValue: any): void {
  super.onPropertyPaneFieldChanged(propertyPath, oldValue, newValue);

  alert(`Path: ${propertyPath}; Old value: ${oldValue}; New Value: ${newValue}`);
}

protected onAfterPropertyPaneChangesApplied(): void {
  alert("Property pane changes applied");
}
```

Property Pane Controls for SPFx

Open-source community library

Managed by the SharePoint
Patterns & Practices(PnP) team

Includes reusable controls with logic tied to existing SharePoint site

https://pnp.github.io/sp-dev-fx-property-controls/

Popular Controls from the PnP Library

PropertyFieldColorPicker

- Generates a color picker

PropertyFieldDateTimePicker

- Create a datetime picker

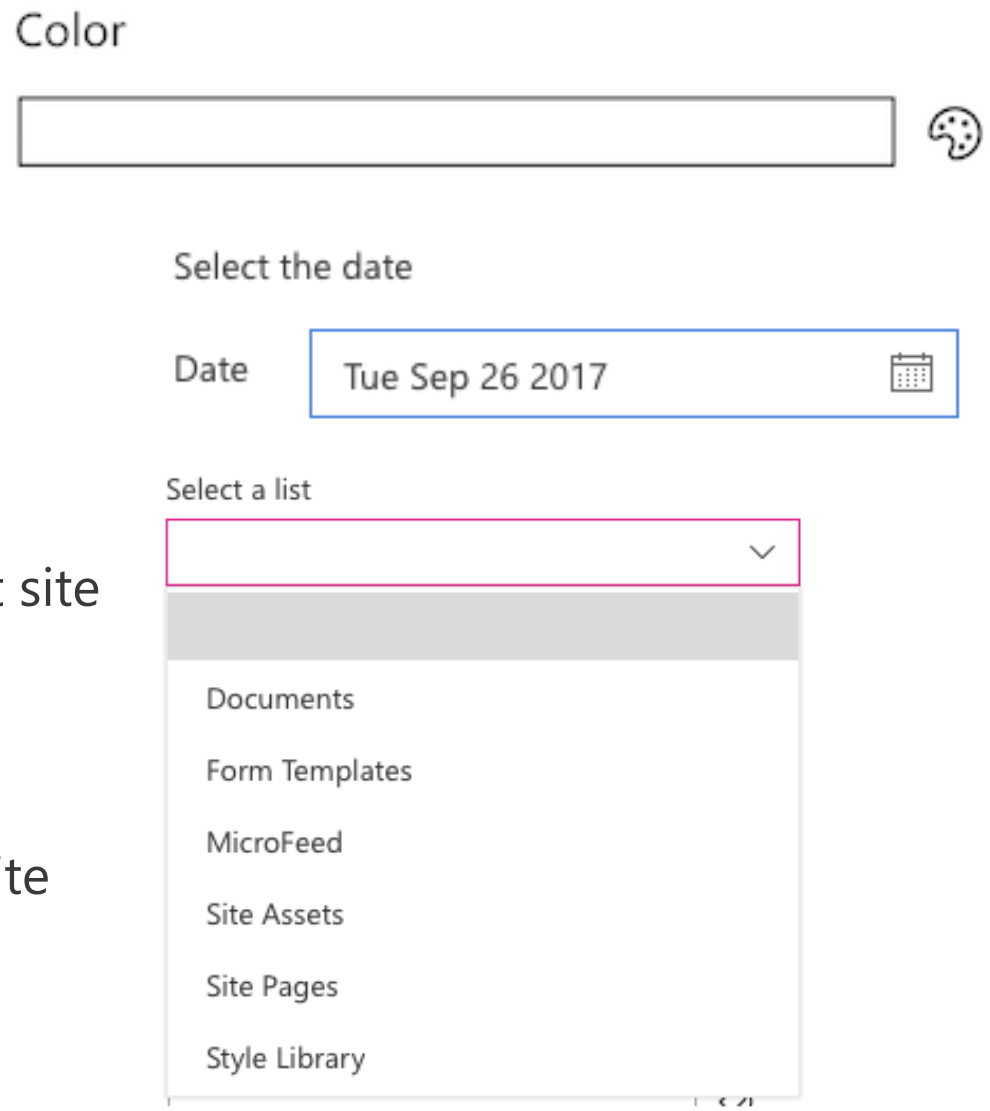
PropertyFieldListPicker

- Displays dropdown of lists from current SharePoint site
- Supports single or multi-select

PropertyFieldPeoplePicker

- Lists users & groups from the current SharePoint site

… and many more!



Adding to your project

```
// install as NPM package
npm install @pnp/spfx-property-controls --save --save-exact

// import reference in web part
import { PropertyFieldListPicker, PropertyFieldListPickerOrderBy } from
  '@pnp/spfx-property-controls/lib/PropertyFieldListPicker’;

// add to property pane configuration
PropertyFieldListPicker('lists', {
  label: 'Select a list',
  selectedList: this.properties.lists,
  includeHidden: false,
  orderBy: PropertyFieldListPickerOrderBy.Title,
  disabled: false,
  onPropertyChange: this.onPropertyPaneFieldChanged.bind(this),
  properties: this.properties,
  context: this.context,
  onGetErrorMessage: null,
  deferredValidationTime: 0,
  key: 'listPickerFieldId'
})
```

Implementing custom property pane fields

Define an interface in your web part that includes one or more target properties

Import `PropertyPaneCustomField` field type in the web part class

`PropertyPaneCustomField` field type is available as a modules in the `@microsoft/sp-property-pane` library

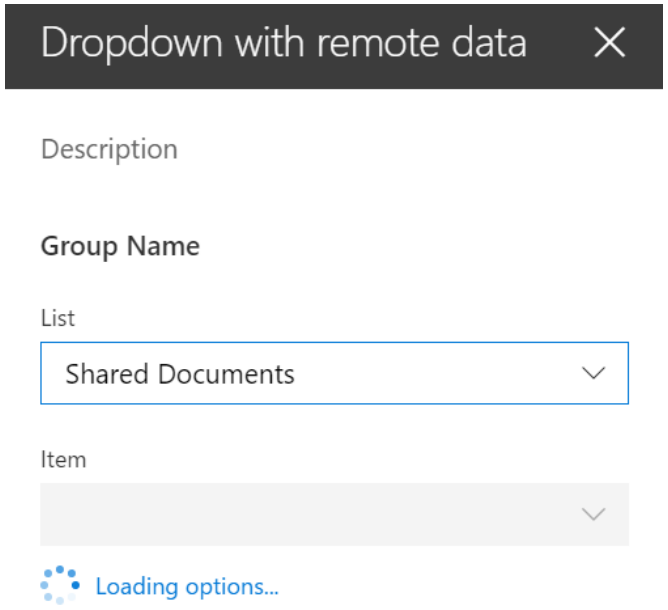
Implementing custom property pane fields

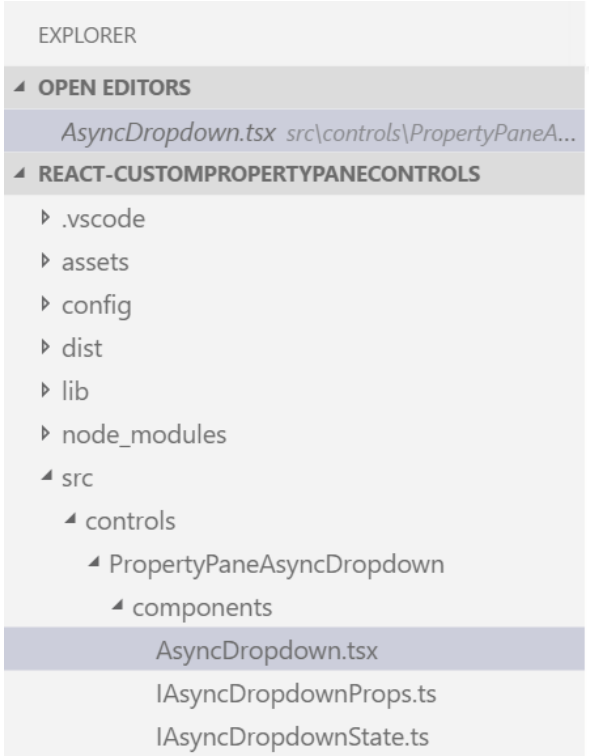
```
// Web Part: create a render method for the custom field
private _customFieldRender(elem: HTMLElement): void {
  elem.innerHTML = '<div><h1>This is a custom field.</h1></div>’;
}

// Add custom field definition in a groupFields array
protected get propertyPaneSettings(): IPropertyPaneSettings {
  return {
    pages: [
      {
        header: { description: strings.PropertyPaneDescription},
        groups: [
          {
            groupName: strings.BasicGroupName,
            groupFields: [
              PropertyPaneCustomField('customField', {
                onRender: this._customFieldRender.bind(this)
              }),
            ]
          } ] } ]
  }
}
```

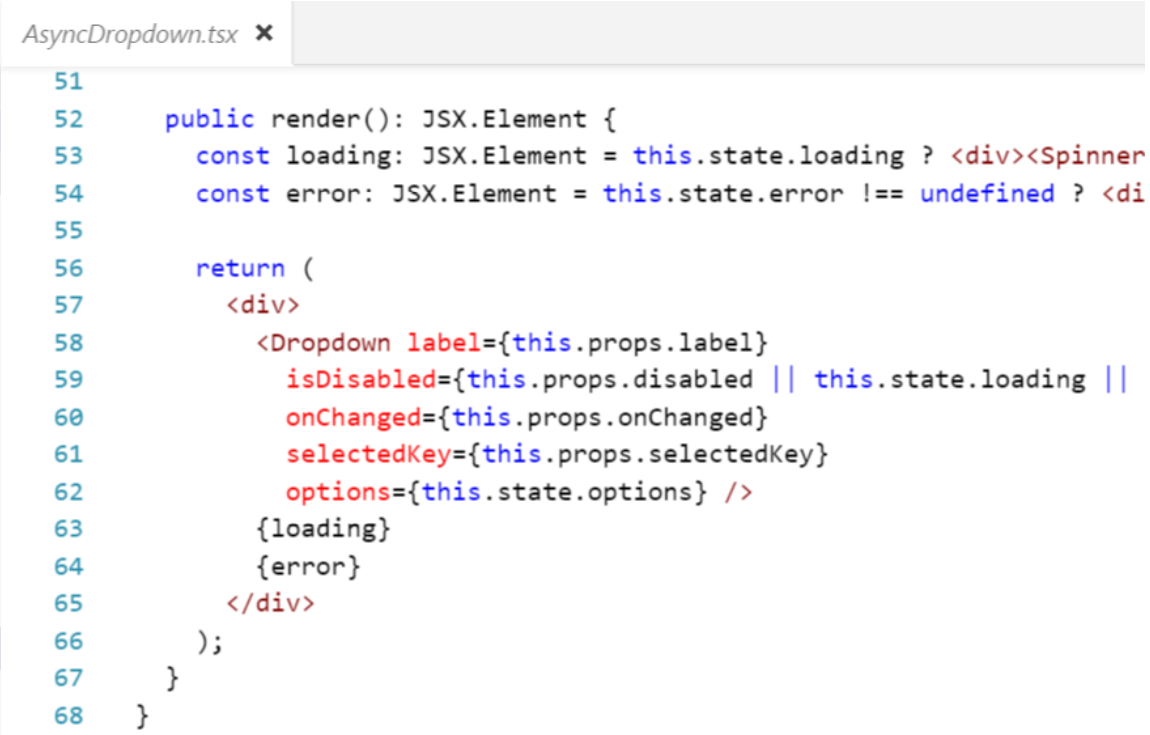
Creating custom property pane controls

When out-of-the-box property pane controls don't meet your needs you can create your own custom controls

Creating custom controls promotes code reusability







https://github.com/SharePoint/sp-dev-fx-webparts/tree/master/samples/react-custompropertypanecontrols

Resources

Overview of the SharePoint Framework
https://learn.microsoft.com/sharepoint/dev/spfx/sharepoint-framework-overview

Make your SharePoint Client-Side Web Part Configurable
https://learn.microsoft.com/sharepoint/dev/spfx/web-parts/basics/integrate-with-property-pane

Build Custom Controls for the Property Pane
https://learn.microsoft.com/sharepoint/dev/spfx/web-parts/guidance/build-custom-property-pane-controls

Reusable Property Panel Controls for SharePoint Framework Solutions
https://sharepoint.github.io/sp-dev-fx-property-controls/