

SPFx Forms and React Formik

Easily create custom SharePoint list forms



Sven Sieverding

Microsoft 365 Consultant

HEC GmbH

sven.sieverding@hec.de



<https://www.linkedin.com/in/sven-sieverding/>



@365knoten

Blog: <https://www.365knoten.de/>

Slides & Code

<https://github.com/365knoten/CollabdaysBE2023ReactFormik>



Thanks to our sponsors!

Platinum



Gold



Silver



SharePint



Community



Organized by

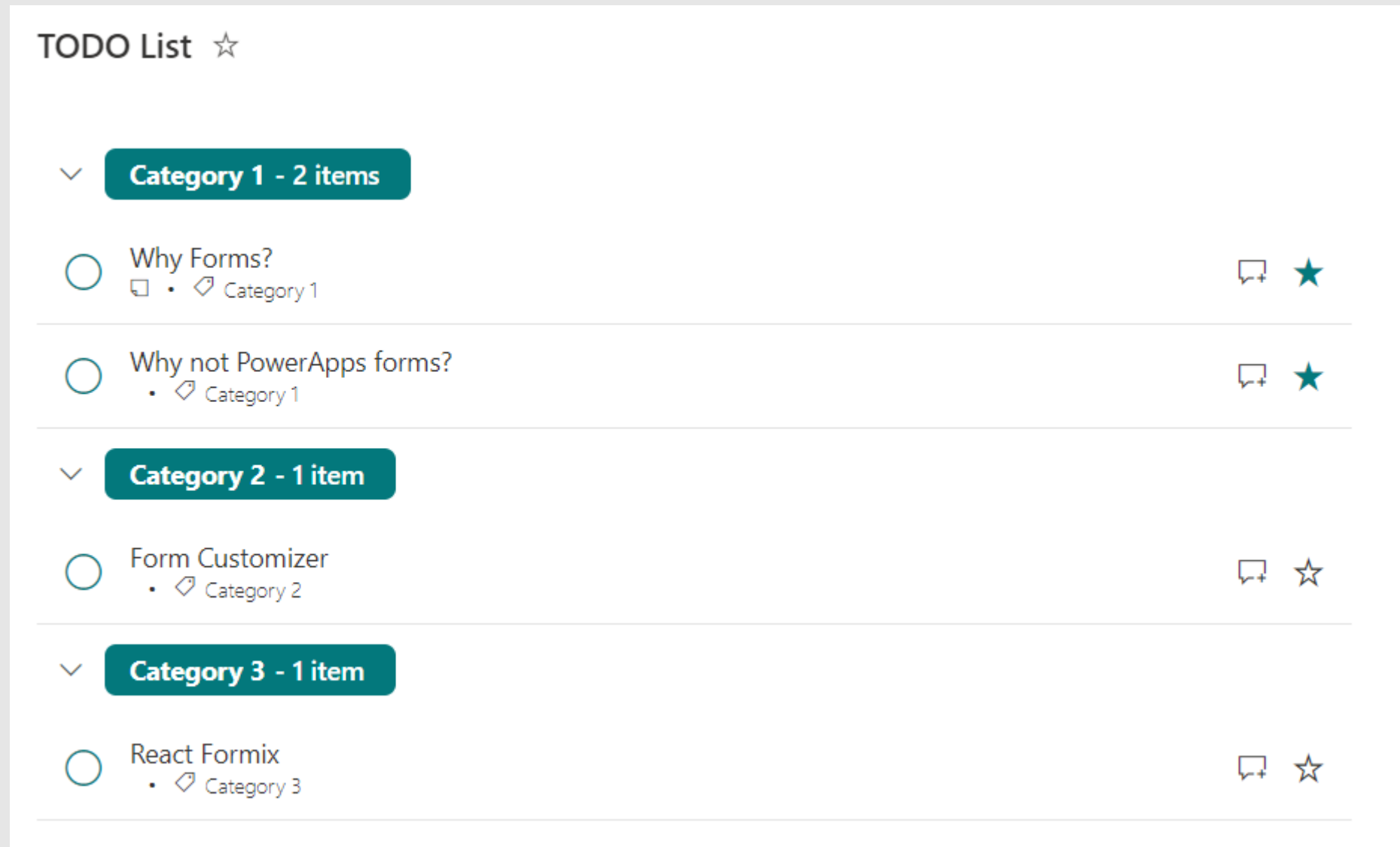


SharePoint Lists

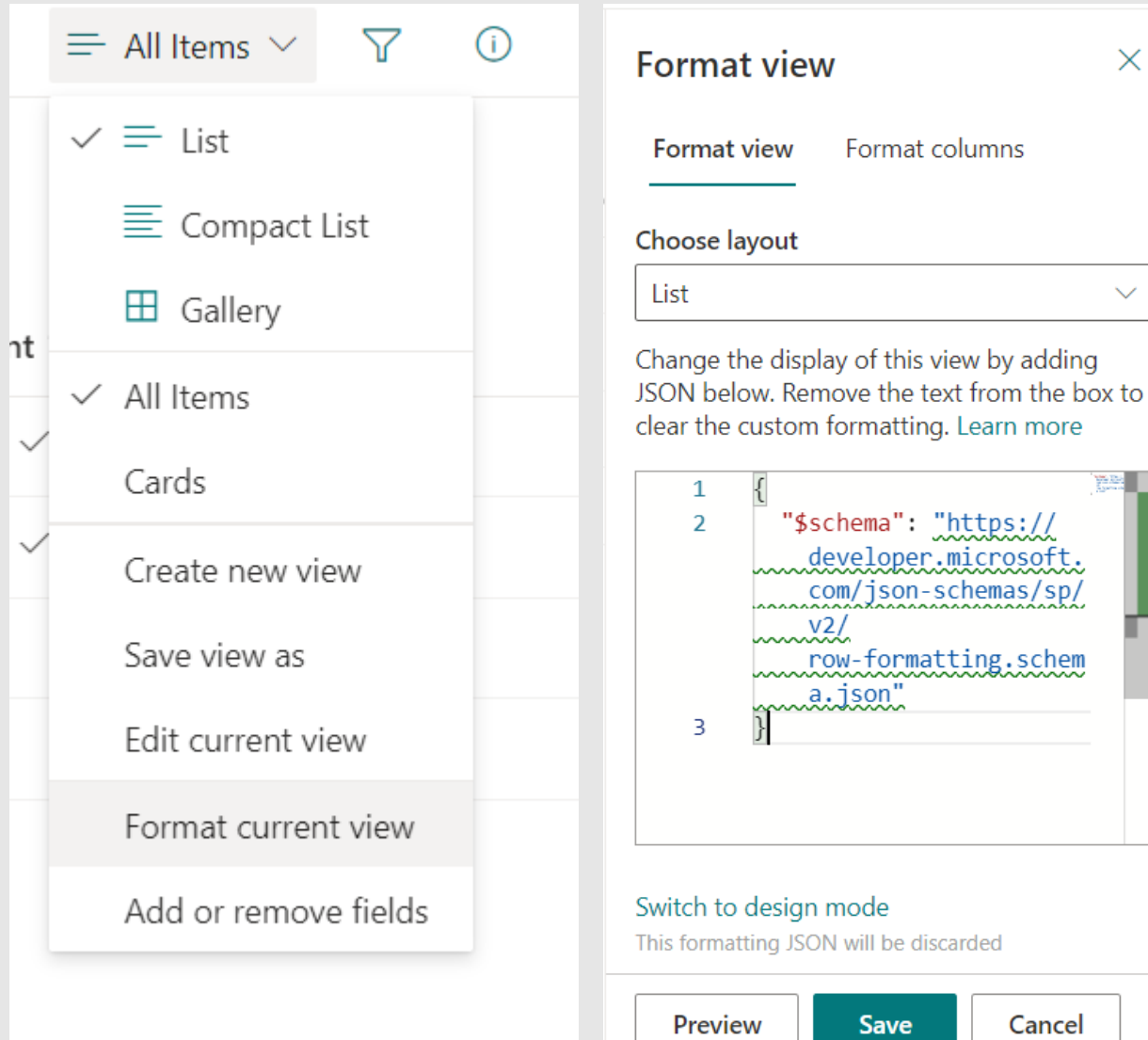
TODO List ☆

Task Name ▾	Assigned To ▾	Due Date ▾	Description ▾	Category ▾	Important ▾	Status ▾
📌 Why Forms?	Sven Sieverding	21.10.2023	Show what is possible	Category 1	✓	
📌 Why not PowerApps forms?	Sven Sieverding	21.10.2023		Category 1	✓	
📌 Form Customizer	Sven Sieverding	21.10.2023		Category 2		✓
📌 React Formix	Sven Sieverding	21.10.2023		Category 3		✓

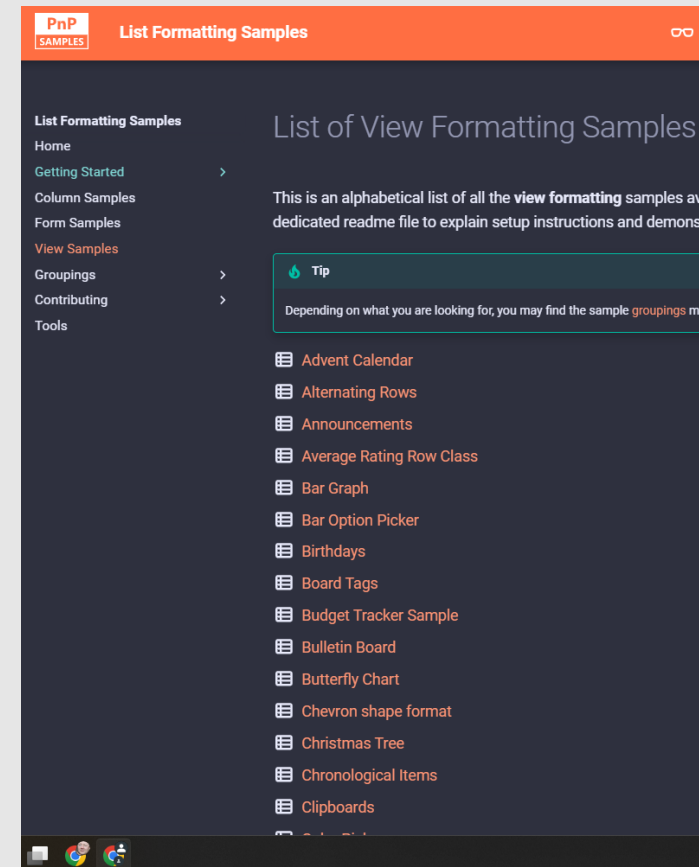
SharePoint Json List Formatting






SharePoint Json List Formatting



• <https://pnp.github.io/List-Formatting/>



SharePoint

 Save  Cancel  Copy link

New item

 Task Name

Why Forms?


 Assigned To



Sven Sieverding



Enter a name or email address

 Due Date

21.10.2023



 Description

Enter value here

 Category

Category 1

 Important



Yes

 Status



Yes

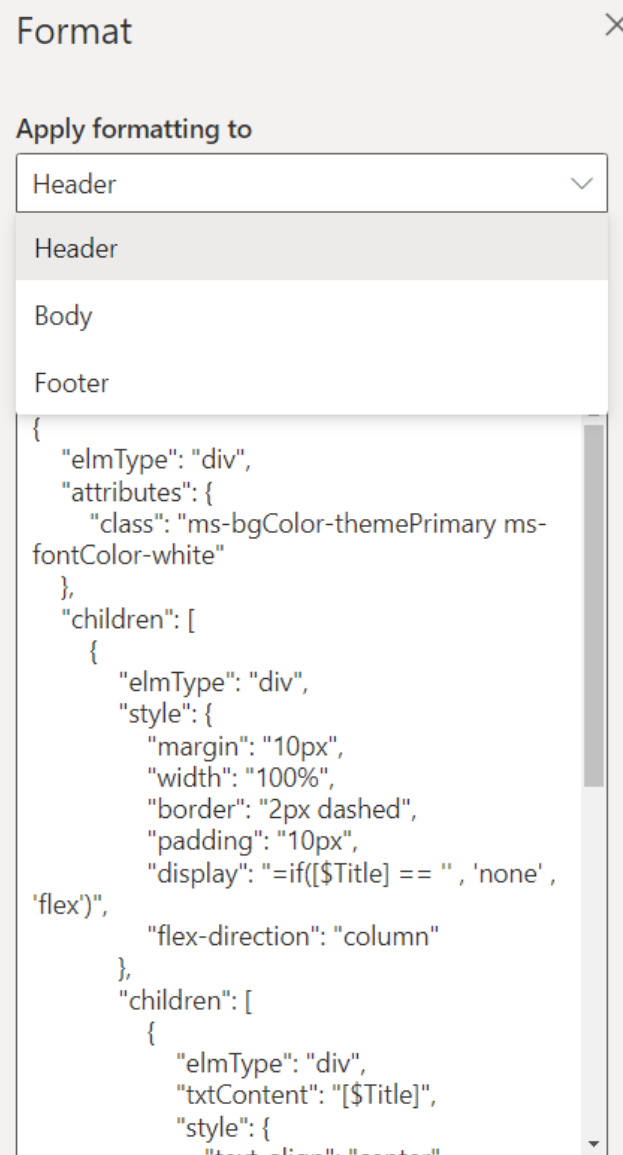
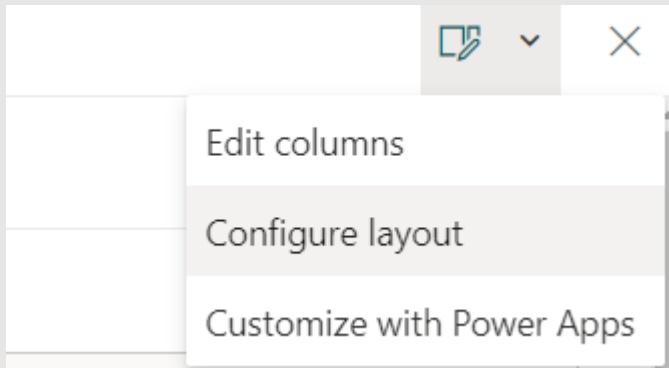
How to customize forms?

SharePoint JSON List Formatting

PowerApps

SPFx Form Customizer

SharePoint JSON List Formatting



Sh

Why Forms?

Task Info

 Task Name

Why Forms?

 Description

Show what is possible

States and Dates

 Important

✓

 Status

Enter value here

 Assigned To

Sven Sieverding

 Due Date

21.10.2023

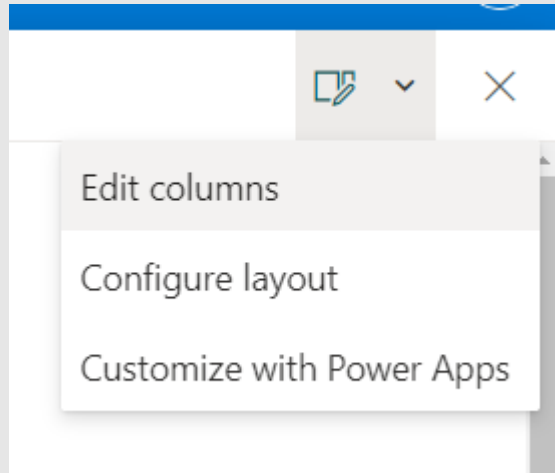
 Date

Sa; Oktober 21

 Attachments

Add or remove attachments

SharePoint JSON List Formatting



Save Cancel

Edit columns in the form

Select a column to show or hide it in the form. To reorder columns, use drag and drop, or find more options next to each column. Required columns and columns with conditional formulas can't be hidden.

Task Info

☒ Description

States and Dates

☒ Category

☒ Important

☒ Status

☒ Task Name

☒ Assigned To

☒ Due Date

Move Up

Move Down

Edit conditional formula

Edit conditional formula for Category field:

To determine whether this field is shown or hidden, specify a conditional formula based on the value of another field. Leave it blank to clear the condition.

[Learn to use conditional formulas in a list form.](#)

Enter your custom formula here:

=if([\$Status]=true,'true','false')

Save

Cancel

SharePoint JSON List Formatting


States and Dates

 Status

 Yes

 Important

✓

 Category

Category 1

 Assigned To

 Due Date

 Date

States and Dates

 Status

 Important

✓

 Assigned To

Sven Sieverding

 Due Date

 Date

How to customize forms?

SharePoint JSON List Formatting

- Can modify form header and footer ✓
- Can set field visibility based on formula ✓
- Can easily be deployed programmatically ✓
- Can not do complex modifications and validations on the form ✗

PowerApps Forms

Edit columns

Configure layout

Customize with Power Apps

← Back to SharePoint

↶ | ∨ | 📁 | ∨ | ...

✎ Editing

🔗

💬

▶

💾

∨

📌

Fill ∨ = *fx* ∨ `RGBA(255, 255, 255, 1)` ∨

☰

📁

+

🗑

📄

📄

(x)

🔧

⚙

👤

Tree view

×

Screens Components

🔍 Search

+ New screen ∨

> 📁 App

📄 SharePointIntegration

∨ 📄 FormScreen1 ...

> 📄 SharePointForm1

SCREEN

Description

Show what is possible

Title

Why Forms?

Status

☐

Important

☒

📄 FormScreen1

— ————— + 80 % ↗

PowerApps Forms

Save Cancel Copy link

Description

Show what is possible

Title

Why Forms?

Status

☐

Important

☒

Category

Category 1

Assigned To

Sven Sieverding

Due Date





10/21/2023

Date



Sa; Oktober 21

How to customize forms?

SharePoint JSON List Formatting





- Can modify form header and footer 
- Can set field visibility based on formula 
- Can easily be deployed programmatically 
- Can not do complex modifications and validations on the form 

PowerApps



- Can do complex modifications and validations on the form 
- It is difficult to move the form programmatically between sites/ stages /tenants 

How to customize forms?




SharePoint JSON List Formatting

- Can modify form header and footer 
- Can set field visibility based on formula 
- Can easily be deployed programmatically 
- Can not do complex modifications and validations on the form 

PowerApps

- Can do complex modifications and validations on the form 
- It is difficult to move the form programmatically between sites/ stages /tenants 

SPFx Form Customizer

- Part of SharePoint Framework since Version 1.15 
- Deployment as an .sppkg Package 
- Can be programmatically deployed 
- Package can include list definition in itself (Advanced scenario) 

Start a Form Customizer project

```
md react-formix-form-customizer  
cd react-formix-form-customizer
```

```
# Create a new form customizer project
```

```
npx -p yo -p @microsoft/generator-sharepoint yo @microsoft/sharepoint
```

```
# We need this library later
```

```
npm install @pnp/sp --save
```



Welcome to the Microsoft
365 SPFx Yeoman
Generator@1.18.0

See <https://aka.ms/spfx-yeoman-info> for more information on how to use this generator.

Let's create a new Microsoft 365 solution.

What is your solution name? **react-formix-form-customizer**

Which type of client-side component to create? **Extension**

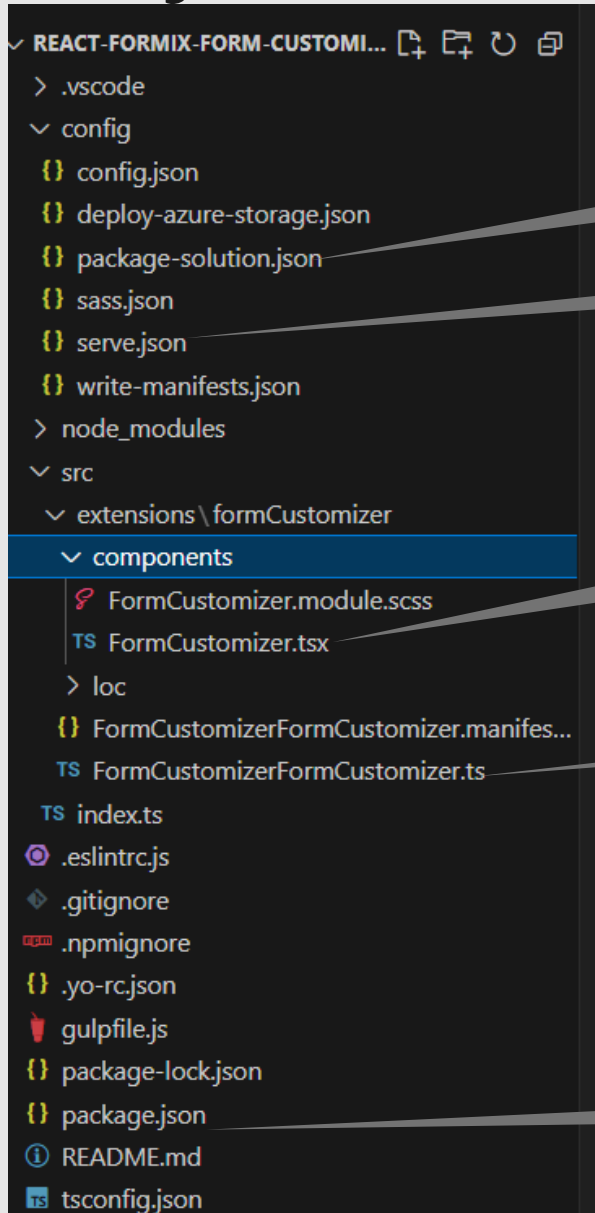
Which type of client-side extension to create? **Form Customizer**

Add new Form Customizer to solution **react-formix-form-customizer**.

What is your Form Customizer name? **Form Customizer**

Which template would you like to use? **React**

Project Structure



Version ID: Update with every deployment

Serve Configuration: Define the page to be opened during debug

The main **React** component of the form

The main customizer class

Startup scripts

serve.json

```
{
  "$schema": "https://developer.microsoft.com/json-schemas/spfx-build/spfx-serve.schema.json",
  "port": 4321,
  "https": true,
  "serveConfigurations": {
    "default": {
      "pageUrl": "https://365knoten.sharepoint.com/sites/MyAppPage/_layouts/15/SPListForm.aspx",
      "formCustomizer": {
        "componentId": "5390b044-a499-48d7-a70d-fd4f26139598",
        "PageType": 8,
        "RootFolder": "/sites/MyAppPage/Lists/TOD0%20List",
        "properties": {
          "sampleText": "Value"
        }
      }
    }
  },
  "formCustomizer": {
    "componentId": "5390b044-a499-48d7-a70d-fd4f26139598",
    "PageType": 8,
    "RootFolder": "/sites/MyAppPage/Lists/TOD0%20List",
    "properties": {
      "sampleText": "Value"
    }
  },
  "pageUrl": "https://365knoten.sharepoint.com/sites/MyAppPage/_layouts/15/SPListForm.aspx",
  "formCustomizer": {
    "componentId": "5390b044-a499-48d7-a70d-fd4f26139598",
    "PageType": 8,
    "RootFolder": "/sites/MyAppPage/Lists/TOD0%20List",
    "ID": 1,
  }
}
```

Site Url

Site Path

ID of the Item to edit in debug mode

package.json

```
"scripts": {  
  "serve:newform": "gulp serve --config=formCustomizer_NewForm",  
  "serve:editform": "gulp serve --config=formCustomizer_EditForm",  
  "serve:viewform": "gulp serve --config=formCustomizer_NewForm",  
  "dist": "gulp clean && gulp build --ship && gulp bundle --ship && gulp package-solution --ship",  
  "eslint:autofix": "eslint --fix --ext .ts,.tsx src/"  
},
```

```
C:\react-formix-form-customizer> npm run serve:editform
```

Load debug scripts?

Allow debug scripts?

WARNING: This page contains unsafe scripts that, if loaded, could potentially harm your computer. Do not proceed unless you trust the developer and understand the risks.

If you are unsure, click Don't load debug scripts.

Load debug scripts

Don't load debug scripts

If you don't know
what to do, click
here.

The generated form:

<You will see nothing, just an empty SharePoint page>

FormCustomizer.tsx

```
const LOG_SOURCE: string = 'FormCustomizer';

export default class FormCustomizer extends React.Component<IFormCustomizerProps, {}> {
  public componentDidMount(): void {
    Log.info(LOG_SOURCE, 'React Element: FormCustomizer mounted');
  }

  public componentWillUnmount(): void {
    Log.info(LOG_SOURCE, 'React Element: FormCustomizer unmounted');
  }

  public render(): React.ReactElement<{}> {
    return <div className={styles.formCustomizer} />;
  }
}
```

FormCustomizerFormCustomizer.ts – render Method

```
public render(): void {  
    // Use this method to perform your custom rendering.  
  
    const formCustomizer: React.ReactElement<{}> =  
        React.createElement(FormCustomizer, {  
            context: this.context,  
            displayMode: this.displayMode,  
            onSave: this._onSave,  
            onClose: this._onClose  
        } as IFormCustomizerProps);  
  
    ReactDOM.render(formCustomizer, this.domElement);  
}
```


FormCustomizerFormCustomizer.ts – onClose & onSave

```
private _onSave = (): void => {  
    // You MUST call this.formSaved() after you save the form.  
    this.formSaved();  
}  
  
private _onClose = (): void => {  
    // You MUST call this.formClosed() after you close the form.  
    this.formClosed();  
}
```

Change FormCustomizerFormCustomizer.ts – add _item variable

```
export default class FormixCustomizerFormCustomizer  
  extends BaseFormCustomizer<IFormixCustomizerFormCustomizerProperties> {  
  
  private _item: any = {};
```

Change FormCustomizerFormCustomizer.ts – change onInit()

```
public onInit(): Promise<void> {  
    if (this.context.itemId !== undefined) {  
        // itemId is set on an Edit and Displayform  
        // if it is set load data from SharePoint  
        return spfi().using(SPFx(this.context))  
            .web  
            .lists  
            .getById(this.context.list.guid.toString())  
            .items  
            .getById(this.context.itemId)()  
            .then((item: any) => {  
                // The following fields need to be removed from the item,  
                // if we want to save the object again  
                delete item["odata.editLink"];  
                delete item["odata.etag"];  
                delete item["odata.id"];  
                delete item["odata.metadata"];  
                delete item["odata.type"];  
                delete item.odata;  
                this._item = item;  
                console.log(item);  
            })  
    }  
  
    return Promise.resolve();  
}
```

Change FormCustomizerFormCustomizer.ts – change _onSave()

```
private _onSave = (item: any): void => {  
    // If we are in Edit Mode: Update the existing item  
    if (this.displayMode === FormDisplayMode.Edit && this.context.itemId !== undefined) {  
        spfi().using(SPFx(this.context))  
            .web.lists.getById(this.context.list.guid.toString())  
            .items.getById(this.context.itemId)  
            .update(item)  
            .then(() => {  
                this.formSaved();  
            })  
    };  
  
    // If we are in New Mode: Create a new Item  
    if (this.displayMode === FormDisplayMode.New) {  
        spfi().using(SPFx(this.context))  
            .web.lists.getById(this.context.list.guid.toString())  
            .items  
            .add(item)  
            .then(() => {  
                this.formSaved();  
            })  
    }  
}
```

Change FormCustomizer.tsx

```
export const FormCustomizer = (props: IFormCustomizerProps) => {
  const [text, setText] = React.useState(props.item.Title)
  return <div>
    <TextField
      name="Title"
      value={text}
      label="Title"
      onChange={(e, value) => {
        setText(value);
      }}
    />
    <PrimaryButton text="Save" onClick={
      () => {
        props.item.Title = text;
        props.onSave(props.item)
      }
    } />
    <DefaultButton text="Cancel" onClick={
      () => {
        props.onClose()
      }
    } />
  </div>
}
```

The result:

Title	
Save	Cancel

Great, but.....

```
export const FormCustomizer = (props: IFormCustomizerProps) => {  
  const [text, setText] = React.useState(props.item.Title)  
  return <div>  
    <TextField  
      name="Title"  
      value={text}  
      label="Title"  
      onChange={(e, value) => {  
        setText(value);  
      }}  
    />  
    <PrimaryButton text="Save" onClick={()  
      => {  
        props.item.Title = text;  
        props.onSave(props.item)  
      }}  
    />  
    <DefaultButton text="Cancel" onClick={()  
      => {  
        props.onClose()  
      }}  
    />  
  </div>  
}
```

This does not scale well, if we add multiple fields

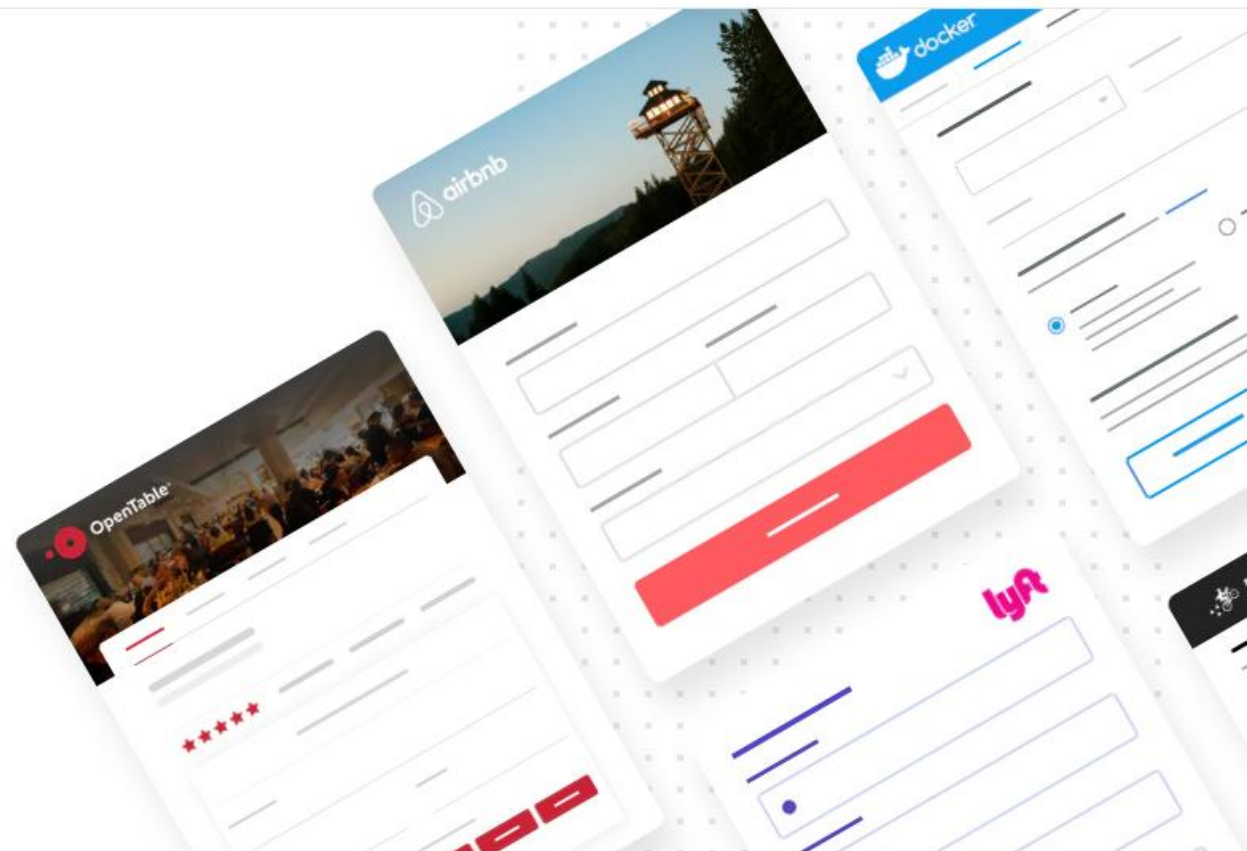
We will repeat code

Setting up validation and form logic is going to be a hassle

The whole form is going to be ugly and complex

Build forms in React, without the tears

Formik is the world's most popular open source form library for React and React Native.

[Get Started](#)[GitHub](#)

Declarative

Formik takes care of the repetitive and annoying stuff—keeping track of values/errors/visited fields, orchestrating validation, and handling submission—so you don't have to. This means you spend less time wiring up state and change handlers and more time focusing on your business logic.

Intuitive

No fancy subscriptions or observables under the hood, just plain React state and props. By staying within the core React framework and away from magic, Formik makes debugging, testing, and reasoning about your forms a breeze. If you know React, and you know a bit about forms, you know Formik!

Adoptable

Since form state is inherently local and ephemeral, Formik does not use external state management libraries like Redux or MobX. This also makes Formik easy to adopt incrementally and keeps bundle size to a minimum.

React Formik

Objectives:

- Getting values in and out of form state
- Validation and error messages
- Handling form submission
- Easy and clean code

```
npm install formik
```

React Formik : Core Idea

We have this.... Our „onSave“ Function

We have this.... Our „item“

```
<Formik
  onSubmit={SUBMITFUNCTION}
  initialValues={INITIALVALUES}
  validate={VALIDATIONFUNCTION}>

  ({ values, handleSubmit, isSubmitting, handleChange }) => (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        name="Task Name"
        onChange={handleChange}
        value={values.Title}
      />
      <button type="submit" disabled={isSubmitting}>
        Submit
      </button>
    </form>
  )
</Formik>
```

React Formik : Validation function (ToDoFormValidator.ts)

```
export default (item: any) => {  
  const errors: any = {};  
  
  if (!item.Title) {  
    errors.Title = "Title is required";  
  }  
  if (!item.Description || item.Description.length <= 20) {  
    errors.Description = "Description must be at least 20 characters";  
  }  
  if (!item.DueDate || item.DueDate < new Date()) {  
    errors.DueDate = "Due Date is required and must be in the future";  
  }  
  
  return errors;  
};
```

React Formik : Extract into reusable components

```
<Formik
  onSubmit={SUBMITFUNCTION}
  initialValues={INITALVALUES}
  validate={VALIDATIONFUNCTION}>SPFormikForm

  ({({ values, handleSubmit, isSubmitting, handleChange }) => (
    <form onSubmit={handleSubmit}>

      <input
        type="text"
        name="Task Name"
        onChange={handleChange}
        value={values.Title}
      />TextField

      <button type="submit" disabled={isSubmitting}>
        SubmitFormButtons
      </button>

    </form>
  )})
</Formik>
```

New File: SPFormikForm.tsx

```
export default (props: ISFormikFormProps) => {  
  return (  
    <Formik  
      onSubmit={async (values: any, { setSubmitting }) => {  
        setSubmitting(true);  
        await props.onSave(values);  
        setSubmitting(false);  
      }}  
      initialValues={props.initialValues}  
      validate={props.validator}  
    >  
      ({ { handleSubmit } }) => (  
        <form onSubmit={handleSubmit}>  
          {props.children}  
          <FormButtons onCancel={props.onCancel} />  
        </form>  
      )  
    </Formik>  
  );  
};
```

New File: TextField.tsx

```
import * as React from "react";
import { TextField as FabricTextField } from "@fluentui/react";
import { ErrorMessage, useField } from "formik";

export const TextField = (props: {fieldname:string,label:string}) => {
  const [field] = useField(props.fieldname);

  return (
    <>
      <FabricTextField
        name={props.fieldname}
        value={field.value}
        label={props.label}
        onChange={(e) => {
          field.onChange(e);
        }}
      />
      <ErrorMessage component="div" name={props.fieldname} />
    </>
  );
};
```

New File: ToggleField.tsx

```
import { Toggle } from "@fluentui/react";
import { ErrorMessage, useField, useFormikContext } from "formik";

export const ToggleField = (props: {fieldname:string,label:string}) => {
  const [field] = useField(props.fieldname);
  const { setFieldValue } = useFormikContext();

  return (
    <>
      <Toggle
        defaultValue={field.value}
        checked={field.value}
        label={props.label}
        onChange={(_, checked) => setFieldValue(field.name, checked)}
      />
      <ErrorMessage component="div" name={props.fieldname} />
    </>
  );
};
```

New File: DateField.tsx

```
import * as React from "react";
import { DatePicker } from "office-ui-fabric-react";
import { useField, useFormikContext } from "formik";
import { IFieldParams } from "../IFieldParams";

export const DateField = (props: IFieldParams) => {
  const [field, meta] = useField(props.fieldname);
  const { setFieldValue } = useFormikContext();

  return (
    <>
      <DatePicker
        value={field.value === null ? null : new Date(field.value)}
        label={props.label}
        onSelectDate={(date) => setFieldValue(field.name, date)}
      />
    </>
  );
};
```


Why all that?

Because now we finally get to shorten everything to this:



Form Customizer: Final Version

```
export const FormCustomizer = (props: IFormCustomizerProps) => {  
  
  return <SPFormikForm  
    initialValues={props.item}  
    onSave={props.onSave}  
    onCancel={props.onClose}  
    validator={ToDoFormValidator}  
  >  
    <ToDoForm />  
  </SPFormikForm>  
}
```

New File: ToDoForm.tsx

```
export default () => {  
  
  return (  
    <Stack styles={stackStyles}>  
      <Pivot>  
        <PivotItem headerText="Task Info">  
          <TextField fieldname="Title" label="Task Name" />  
          <MultilineTextField fieldname="Description" label="Description" />  
        </PivotItem>  
        <PivotItem headerText="States and Dates">  
          <Stack horizontal tokens={containerStackTokens} styles={stackStyles}>  
            <ToggleField fieldname="Status" label="Status" />  
            <ToggleField fieldname="Important" label="Important" />  
          </Stack>  
          <DateField fieldname="DueDate" label="Due Date" />  
        </PivotItem>  
      </Pivot>  
    </Stack>  
  )  
}
```

Our Form: Final Version



Task Info

States and Dates

Task Name

Why not PowerApps forms?

Description

Test

Description must be at least 20 characters

Cancel

Save

Our Form: Final Version

Task Info

States and Dates

Status


Important

☒ ☒

Due Date is required and must be in the future

Due Date

Tue Oct 10 2023

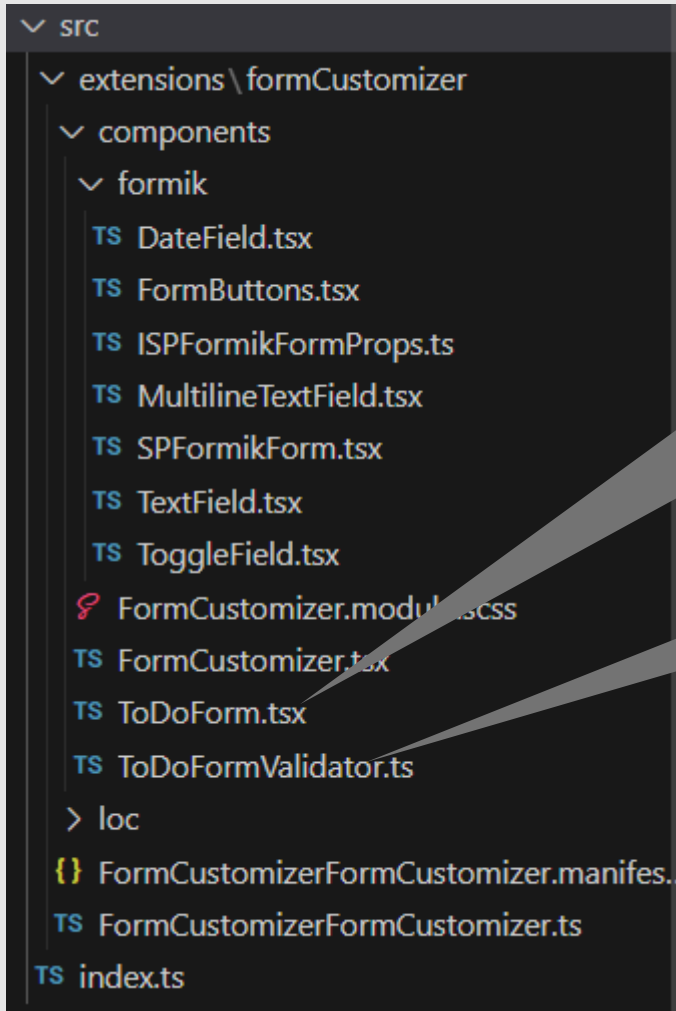


Due Date is required and must be in the future

Cancel

Save

If you want to create your own Formik form



Change the Form to use your fields

Change the Validator to your logic

Just copy & paste the rest

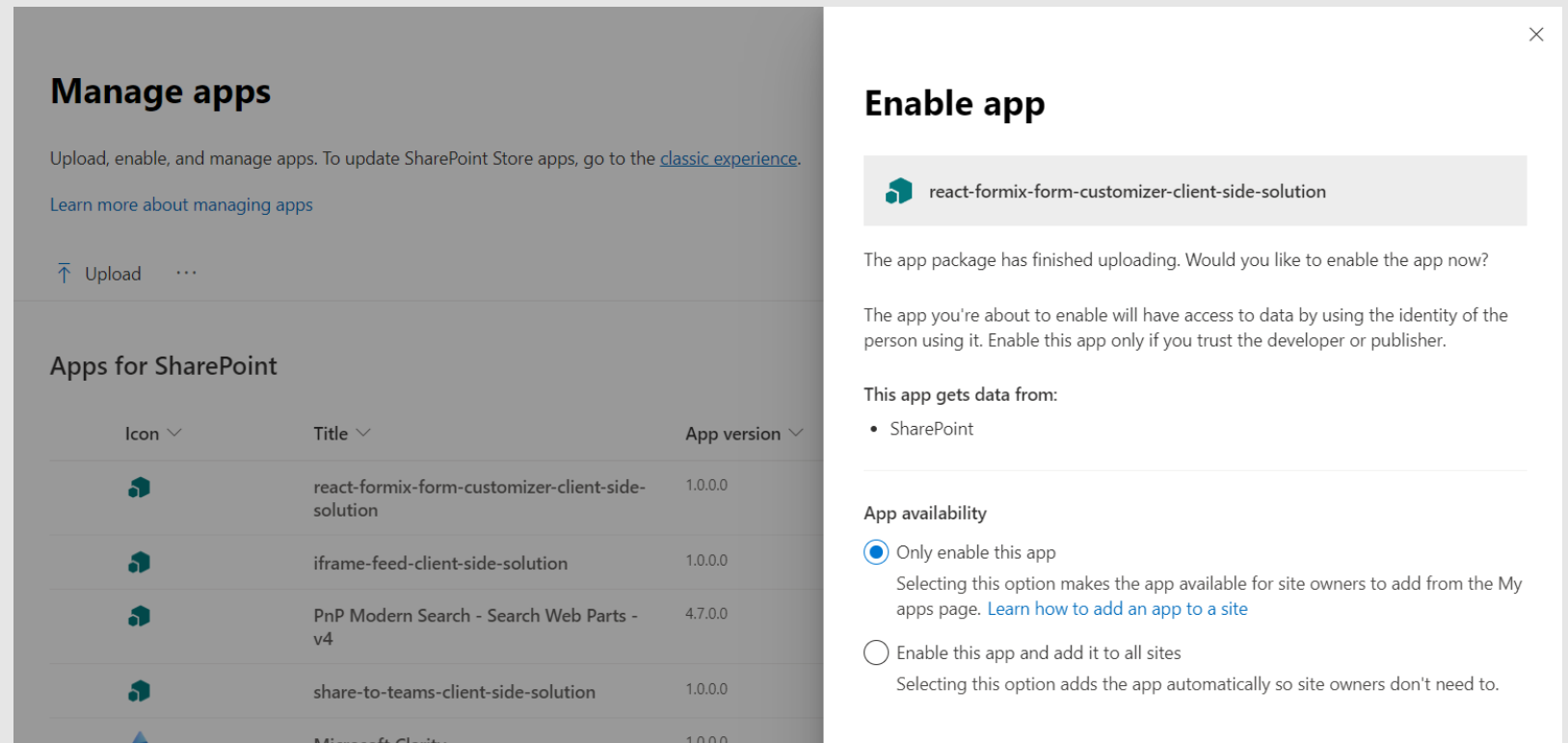
Deployment

```
> npm run dist
```

- Then go to „./sharepoint/solutuion“ and get the „.sppkg“ File

Deployment

- Open „https://<yourtenant>-admin.sharepoint.com/sites/appcatalog/_layouts/15/tenantAppCatalog.aspx“ and upload the file there
- Choose the option to only enable the app.








Manage apps

Upload, enable, and manage apps. To update SharePoint Store apps, go to the [classic experience](#).


[Learn more about managing apps](#)

↑ Upload ...

Apps for SharePoint

Icon	Title	App version
	react-formix-form-customizer-client-side-solution	1.0.0.0
	iframe-feed-client-side-solution	1.0.0.0
	PnP Modern Search - Search Web Parts - v4	4.7.0.0
	share-to-teams-client-side-solution	1.0.0.0
	Microsoft Clarity	1.0.0.0

Enable app

 react-formix-form-customizer-client-side-solution

The app package has finished uploading. Would you like to enable the app now?

The app you're about to enable will have access to data by using the identity of the person using it. Enable this app only if you trust the developer or publisher.

This app gets data from:

- SharePoint

App availability

☒ Only enable this app
Selecting this option makes the app available for site owners to add from the My apps page. [Learn how to add an app to a site](#)

☐ Enable this app and add it to all sites
Selecting this option adds the app automatically so site owners don't need to.

Deployment

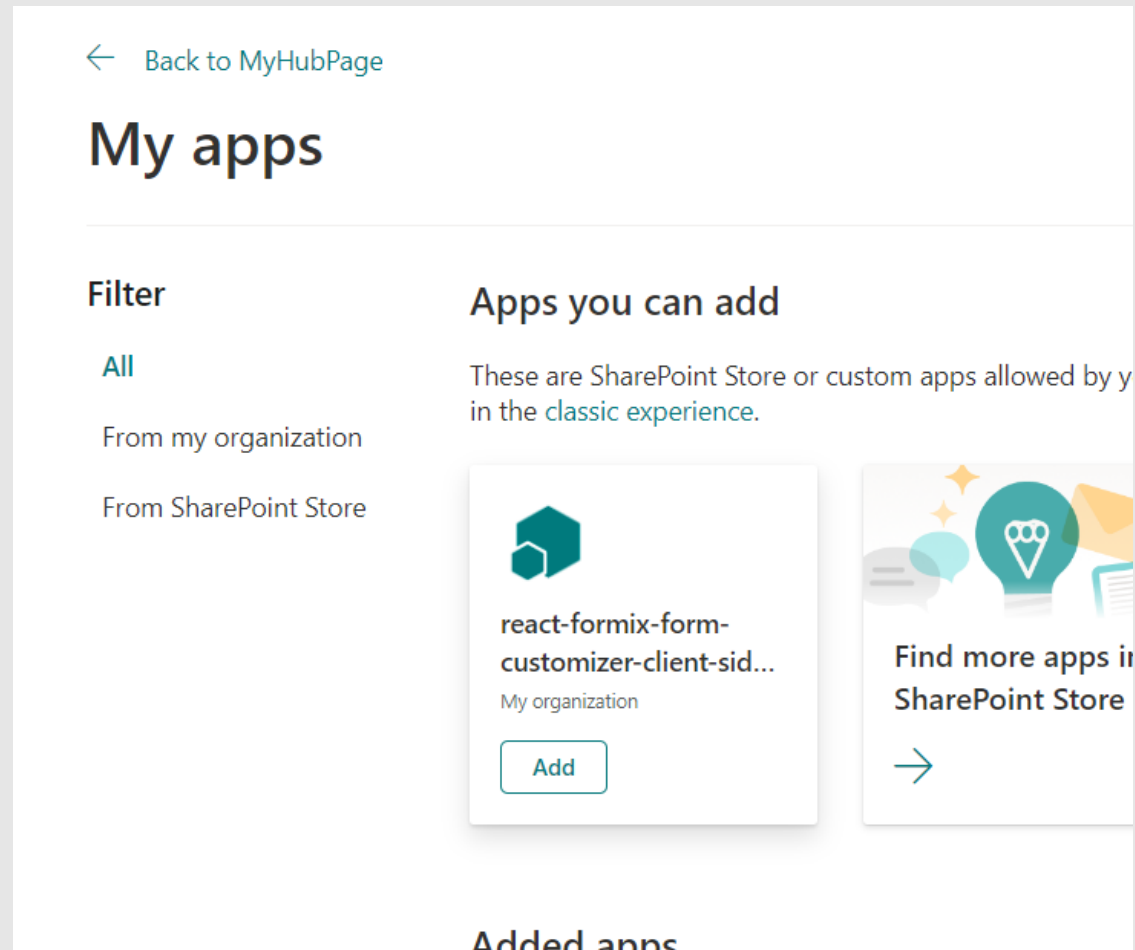
- Now Open PnP Powershell, connect to that site and execute this

```
# Your List name
$listname="TODO List"
# You get the componentID form the id field of the
# FormCustomizerFormCustomizer.manifest.json File
$componentID="5390b044-a499-48d7-a70d-fd4f26139598"
# Item is the default contenttype
$contenttypeName="Item"

Set-PnPContentType -Identity $contenttypeName `
-List $listname `
-NewFormClientIdSideComponentId $componentID `
-EditFormClientIdSideComponentId $componentID `
-DisplayFormClientIdSideComponentId $componentID`
```

Deployment

- Go to your site and open „Site Contents“. Click „New->App“
- Install the app here



Thank you for your attention and happy coding

Slides & Code

<https://github.com/365knoten/CollabdaysBE2023ReactFormik>



Sven Sieverding

Microsoft 365 Consultant

HEC GmbH

sven.sieverding@hec.de



<https://www.linkedin.com/in/sven-sieverding/>



@365knoten

Blog: <https://www.365knoten.de/>