Enhancing SPFx Projects: Harnessing Live Data Integration

Don Kirkham, Microsoft MVP, MCT

Gold

Silver

Web

THANK YOU EVENT SPONSORS

We appreciate you supporting

M365 Community Days NYC!

- Tables on 5th
- Please visit them and inquire about their products & services
- To be eligible for prizes make sure to get your bingo card stamped by ALL sponsors
- Raffle at the end of the day and you must be present to win!

M365 COMMUNITY DAYS NYC STORE

Support our New York tech community with every purchase

- https://rebrand.ly/m365nyc-store

Voitanos

voitanos.io | @voitanos

10% OFF

Use promo code ‘m365nyc’

- **Learn SharePoint Framework (SPFx) Development**
- **Learn Microsoft Teams Application Development**
- **Subscribe for free courses!**

M365nyc Community Days

BINGO CARDS

- Info Desk – has the bingo cards, visit them to play
- Bingo Cards = how you win prizes at the end of the event.
- The cards must be stamped by ALL the Sponsors in order to be eligible to win.
- For the grand prizes you must have opted-in when registering.
- Must be here to win at the end of the day.

M365nyc Community Days

MICROSOFT MINGLE

Join us for an hour of networking!

Enjoy a beverage of your choice on us—whether you prefer a beer, tea, coffee, or soda!

Beer Authority – Rooftop
300 W 40h St
[across the street]
http://www.beerauthoritynyc.com

@DonKirkham

Microsoft MVP, M365 Development

DMI Enterprise Architect

https://donkirkham.com

@DonKirkham

/in/DonKirkham

DonKirkham

Picture time!!!

Client-side data interaction: **REST APIs**

REpresentational **S**tate **T**ransfer
Application **P**rogramming **I**nterface

Method

- GET
- POST
- PUT
- PATCH
- DELETE
- OPTIONS
- HEAD

Request URI

Host | Endpoint | Parameters | Query (ODATA)

https://mydomain.sharepoint.com/sites/targetsite/_api/lists
?mode=1&$select=Title

Headers

Authentication, Content Type, Application Type, Language

Body

```
{
   "Title": "New List",
   "Description": "This is my new list for stuff"
}
```

OData Query Operators

```
https://mydomain.sharepoint.com/sites/site/_api
/web/lists/getbytitle('Speaking%20Events')/items
    ?$select=Id,Title,Session,SessionDate
    &$filter=SessionDate gt DateTime'2024-01-01T00:00:00'
    &$orderby=SessionDate
    &$top=10
```

CRUD with SharePoint Data in SPFx

CRUD Operations with SPFx & SharePoint REST API

Reading items

Creating items

Updating items

Deleting items

Consuming REST APIs in SPFx

- **Common requirement in SPFx project is to display or interact with data external to the web part**
  - Data in SharePoint lists & libraries
  - Data accessible via Microsoft Graph REST API
  - Data accessible in external 3rd Party APIs – anonymous & secured

- **SharePoint Framework provides APIs for all situations when you need to work with data sources external to the web part**
  - `HttpClient`: for calling 3rd party REST APIs
  - `SPHttpClient`: for calling the SharePoint REST APIs in current tenant
  - `MSGraphClient`: for calling the Microsoft Graph in the same tenant as the SharePoint Online tenant
  - `AADHttpClient`: for 3rd party REST APIs secured with Azure Active Directory

- **Most scenarios require no extra clients / libraries are required**

Accessing the SharePoint REST API

https://*mydomain*.sharepoint.com/sites/*targetsite*/_api/...

- **Must include an `Authorization` header containing an OAuth bearer token**

- **Other headers used to control how the REST API is used**
  - OData v3 or v4 (default = v4)
  - Amount & type of metadata returned
  - Type of operation to perform (in the case of updates: merge / update)
  - Match versions

SharePoint Framework & REST API

- **SPFx implements calls to SharePoint REST API via the `SPHttpClient`**
- **Available from the existing context:**
  - `this.context.spHttpClient.get()` & `this.context.spHttpClient.post()`
- **Based on the existing `HttpClient` API**
- **Handles the authentication & default config setting:**
  - Authorization HTTP header
  - OData v4
  - Minimal metadata returned

Reading List Items with the REST API & SPFx

```
private _getEvents(): Promise<ISpeakingEvent[]> {

    const _url: string = this.context.pageContext.web.absoluteUrl
        + `/getbytitle('Speaking%20Events')/items`
            + `?$select=Id,Title,Session,SessionDate`
            + `&$filter=SessionDate gt DateTime'2024-01-01T00:00:00'`
            + `&$orderby=SessionDate%20asc`;

    return this.context.spHttpClient.get(endpoint, SPHttpClient.configurations.v1)
        .then (response: HttpClientResponse => {
            return response.json();
        })
        .then(jsonResponse => {
            return jsonResponse.value;
        }) as Promise<ISpeakingEvent[]>;
}
```

Write Operations with SPFx & REST API

- Use the `SPHttpClient.post()` method to write to the SharePoint REST API

- Some operations require additional HTTP headers:
  - `X-HTTP-Method`: specify `MERGE` or `DELETE` in update & delete operations
  - `IF-MATCH`: specify version of the item on the server to be updated / deleted

- Create operation require specific data in the payload body
  - `@odata.type`: specify the type of data being written to the list when creating

Creating List Items with the REST API

- Must specify the type of data as the `@odata.type` property in the payload that is being created
  - Lists can support multiple content types

- Pattern: request the type in a pre-request via the list's `ListItemEntityTypeFullName` property

Get List Entity Type

```
private _getItemEntityType(): Promise<string> {

    const _url: string = this.context.pageContext.web.absoluteUrl
        + `/_api/web/lists/getbytitle('Speaking%20Events')`
        + `?$select=ListItemEntityTypeFullName`;

    return this.context.spHttpClient.get(_url,SPHttpClient.configurations.v1)
        .then(response: SPHttpClientResponse => {
            return response.json();
        })
        .then(jsonResponse => {
            return jsonResponse.ListItemEntityTypeFullName;
        }) as Promise<string>;
}
```

Creating List Items with the REST API & SPFx

```
private _addNewEvent(): Promise<SPHttpClientResponse> {

    const _url: string = this.context.pageContext.web.absoluteUrl
        + `/_api/web/lists/getbytitle('Speaking%20Events')/items`;

    return this._getItemEntityType()
        .then(spEntityType: string => {
            const request: any = {};
            request.body = JSON.stringify({
                '@odata.type': spEntityType,
                Title: 'New Item'
            });

            return this.context.spHttpClient.post(_url,SPHttpClient.configurations.v1,
                request);
        })
}
```

Updating List Items with the REST API

- **Should specify the type of operation to perform**
  - SPHttpClient only contains `post()` method; not `put()` or `patch()`
  - Default behavior is to set properties to supplied values, BUT omitted properties are set to `null`
  - Override behavior using the `MERGE` method in header, using `X-HTTP-Method` header

- **Specify the version of the item to update**
  - When updating items, can specify “only update the item on the server if it is version X”
  - Ensures you aren’t overwriting someone else’s changes unknowingly
  - Enforced with the `IF-MATCH` header & `etag`’s

Updating List Items with REST API & SPFx

```
private _updateEvent(): Promise<SPHttpClientResponse> {
     // first, get the item to update
     return this.context.spHttpClient.get(
          // .. code to get item from SP REST API
     })
    .then((event: ISpeakingEvent) => {
          // update item
          event.Session = 'UPDATED Event';
          // build request object
          const request: any = {};
          request.headers = {
               'X-HTTP-Method': 'MERGE',
               'IF-MATCH': (event as any)['@odata.etag']
          };
          request.body = JSON.stringify(event);

          const _url: string = this.context.pageContext.web.absoluteUrl
                         + `/_api/web/lists/getbytitle('Speaking%20Events')/items(${event.Id})`

          return this.context.spHttpClient.post(endpoint, SPHttpClient.configurations.v1, request);
    });
}
```

Deleting List Items with the REST API

- **Should specify the type of operation to perform**
  - SPHttpClient only contains `post()` method; not `delete()`
  - Override behavior using the `DELETE` method in header, using the `X-HTTP-Method` header

- **Specify the version of the item to delete**
  - When updating items, can specify “only update the item on the server if it is version X”
  - Enforced with the `IF-MATCH` header & `etag`’s
  - Decide: does it matter if the version is different?
  - If not, use `IF-MATCH = '*'`

Deleting List Items with REST API & SPFx

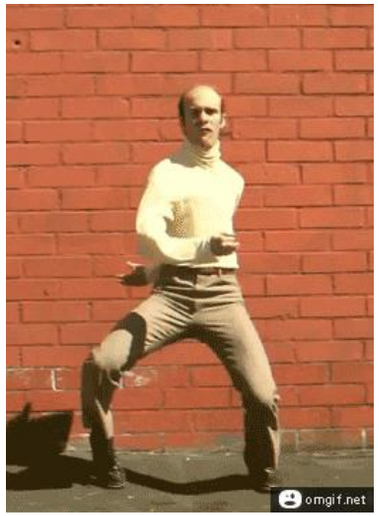
```
private _deleteEvent(): Promise<SPHttpClientResponse> {
     // first, get the item to delete
     return this.context.spHttpClient.get(
          // .. code to get item from SP REST API
     })
    .then((event: ISpeakingEvent) => {
      // delete it
      const request: any = {};
      request.headers = {
        'X-HTTP-Method': 'DELETE',
        'IF-MATCH': '*'
      };

      const _url: string = this.context.pageContext.web.absoluteUrl
                              + `/_api/web/lists/getbytitle('Speaking%20Events')/items(${event.Id})`

      return this.context.spHttpClient.post(_url, SPHttpClient.configurations.v1, request);
    });
}
```

DEMO TIME!

Put on your dancing shoes and let’s have some fun!







Calling Anonymous REST APIs

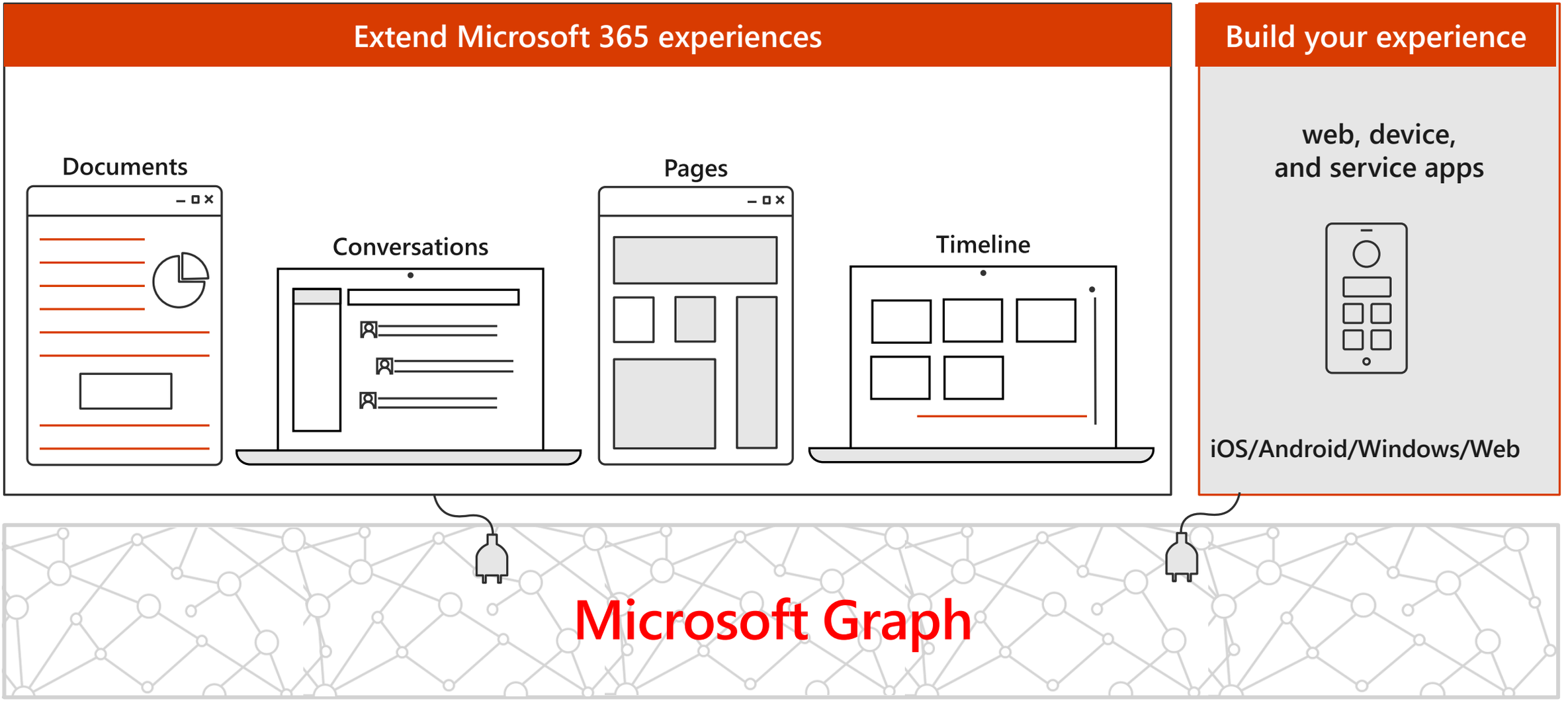
```
private _getCatData: Promise<any> {

    const _url: string = `https://cat-fact.herokuapp.com/facts`,

    return this.context.httpClient.get(_url,HttpClient.configurations.v1)
        .then((response: HttpClientResponse) => {
            return response.json();
        })
        .then(jsonResponse => {
            return jsonResponse;
        }) as Promise<any>;
}
```

Calling the Microsoft Graph

Overview of the Microsoft Graph

SPFx's MSGraphClient

Microsoft 365 Platform



SharePoint Framework Includes a Microsoft Graph Client

- **`MSGraphClient`**: SharePoint Framework's Microsoft Graph Client
- Abstracts the token acquisition from the SharePoint Framework's support for Azure AD
- Wraps the Microsoft Graph JavaScript SDK and initializes it with one line that returns a promise

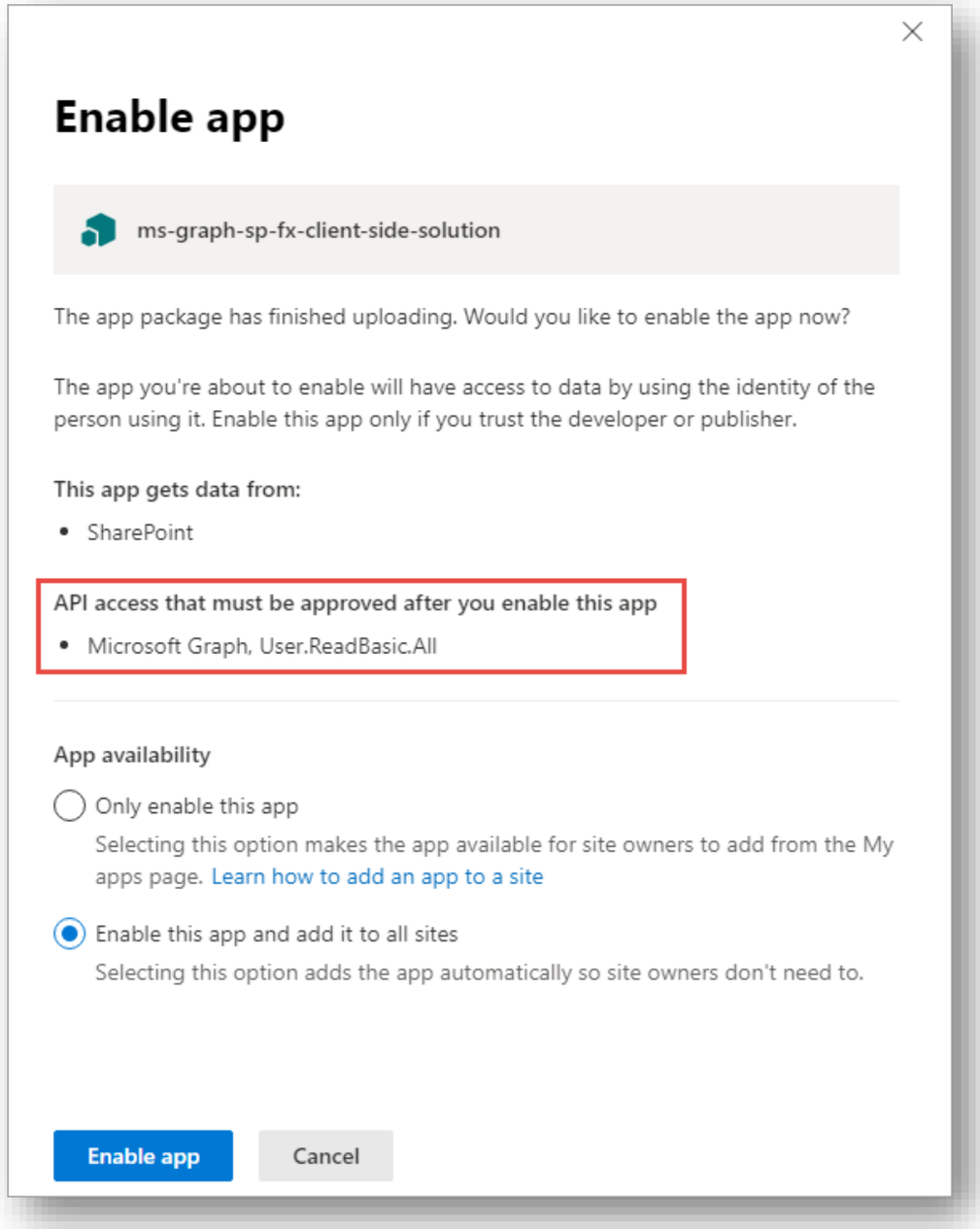
```
import { MSGraphClientV3 } from '@microsoft/sp-http';
// . . .
const graphClient: MSGraphClientV3 = await this.context.msGraphClientFactory.getClient('3');

// use client here
const myEmail: string = await graphClient.api('/me')
        .get((error: GraphError, user: MicrosoftGraph.User) => {
                return user.mail
        });
```

SPFx Solutions Declare Permission Requests

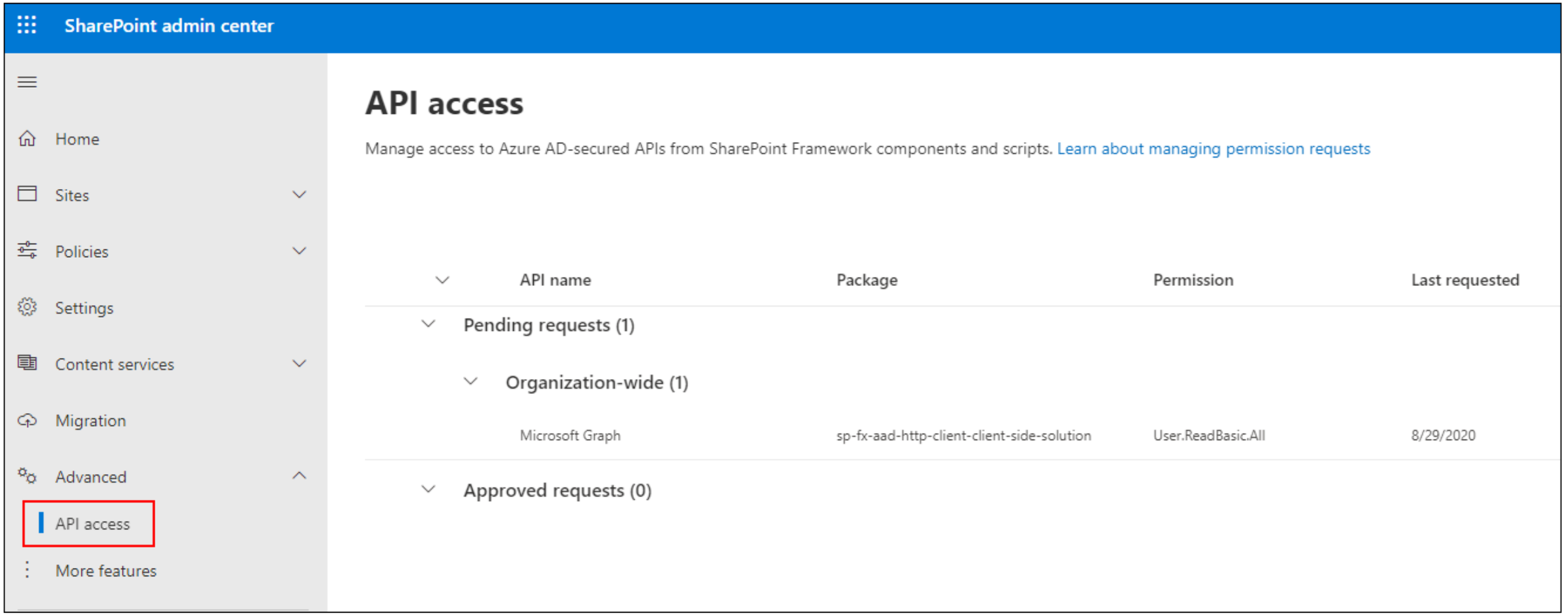
```
// package-solution.json
{
    "solution": {
        ...
        "webApiPermissionRequests": [
            {
                "resource": "Microsoft Graph",
                "scope": "User.ReadBasic.All"
            }
        ],
        ...
    },
    ...
}
```

Add SharePoint Package to SharePoint App Catalog

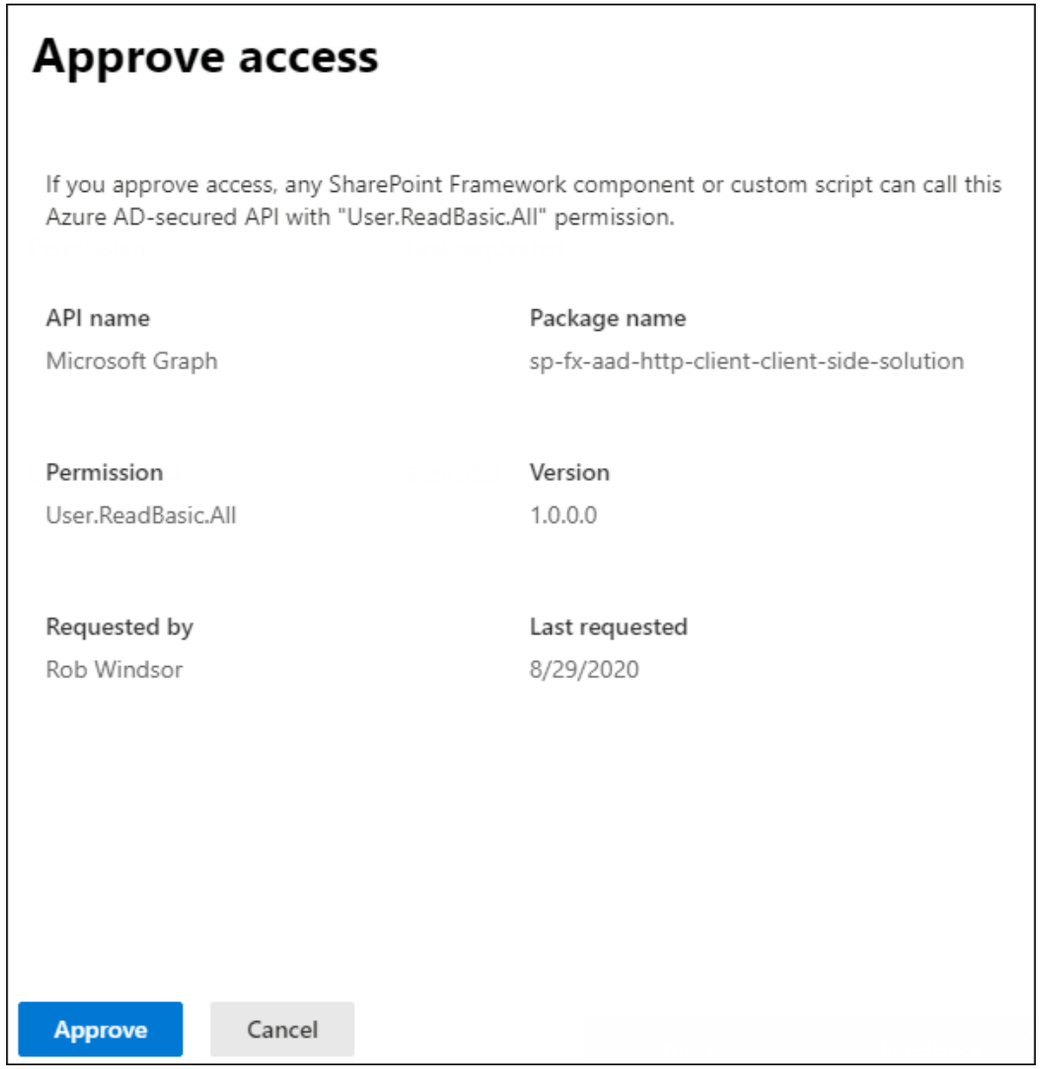
- **Trust dialog**
  - Extra note in dialog notifies of additional step required
  - While application can be installed in SharePoint sites, it does not have the permissions granted that it needs to access Azure AD protected resources



Approve / Reject with SharePoint Online API Management Page



Approve / Reject with SharePoint Online API Management Page

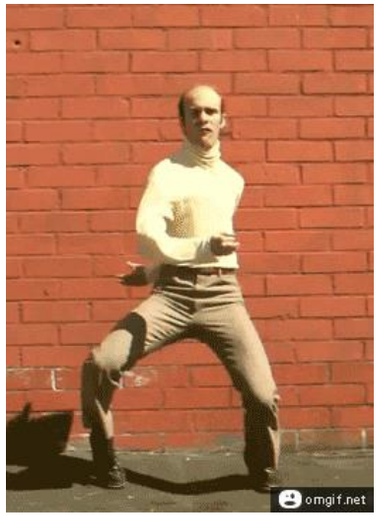


DEMO TIME!

Put on your dancing shoes and let’s have some fun!







PNP Js library for data interaction

Overview of the PnPJs

Getting started

Getting Started with PnPJs

- PnPjs is a collection of fluent libraries for consuming SharePoint, Graph, and Office 365 REST APIs in a type-safe way.
- https://aka.ms/pnpjs
- Ideal for SPFx
- Great documentation
- Easy to set up and use

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

Getting Started with PnPJs

- Initialize by passing the context

```
import { spfi, SPFx } from "@pnp/sp";

//...

protected async onInit(): Promise<void> {
  await super.onInit();
  const sp = spfi().using(SPFx(this.context));
}

//...
```

Getting Started with PnPJs

- Import additional components vs All
  - Keep the bundle smaller
  - SharePoint

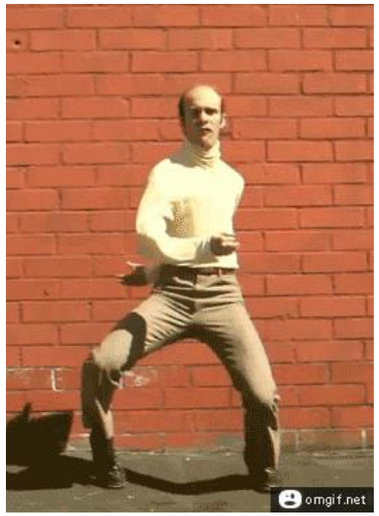
```
import { spfi, SPFx } from "@pnp/sp";
import "@pnp/sp/webs";
import "@pnp/sp/lists";
import "@pnp/sp/items";

// initialize
const sp = spfi (...)

// get all the items from a list
const items: any[] = await sp.web.lists.getByTitle("My List").items();
```

DEMO TIME!

Put on your dancing shoes and let’s have some fun!







Microsoft 365 & Power Platform Community

BLOG COMMUNITY CALLS GUIDANCE SAMPLES & SOLUTIONS SDKS TOOLS TEAM

MICROSOFT 365 & POWER PLATFORM COMMUNITY

Learn from others how to build apps on Microsoft 365 & Power Platform. Don't reinvent the wheel. Focus on what truly matters for your organization.

Sharing is Caring!

aka.ms/m365/community

Thanks!

Do you have any questions?

https://donkirkham.com
@DonKirkham
/in/DonKirkham
DonKirkham