



Build Solutions for Microsoft 365 with a Fluent API Library

Julie Turner

Julie Turner

Partner/CTO

Working with SharePoint since 2007

Microsoft MVP, Office Apps and Services since 2017

Microsoft 365 PnP Team since 2019

Open-source project co-maintainer: PnPJs & hTWOo



<https://julieturner.net/me>

SYMPRAXIS
CONSULTING



Agenda

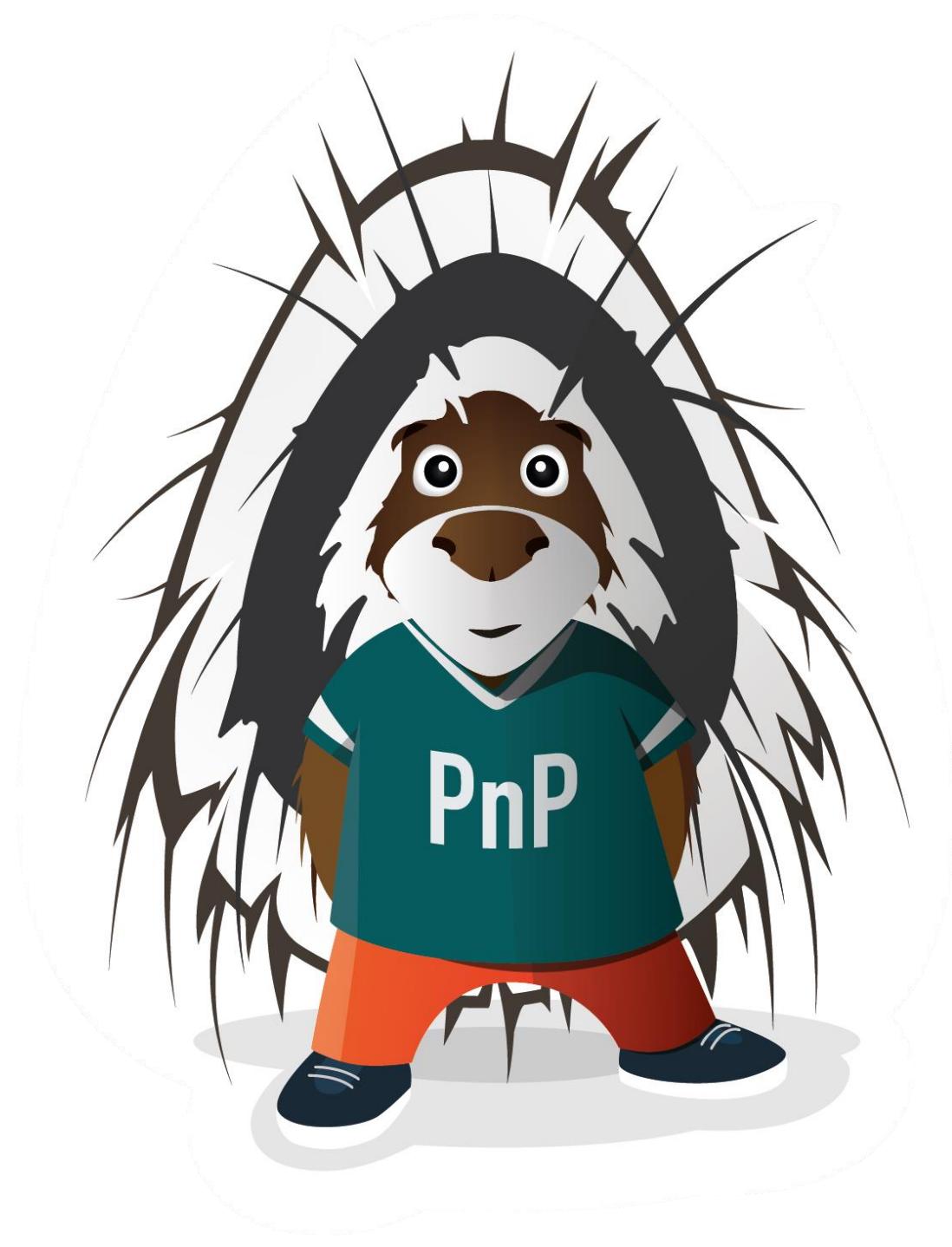
PnP(js) Introduction

Demos

- Installation
- Establish Context
- Get Data
- Caching
- Batching
- Cross Site Calls

What's changed for V4

Q&A



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.

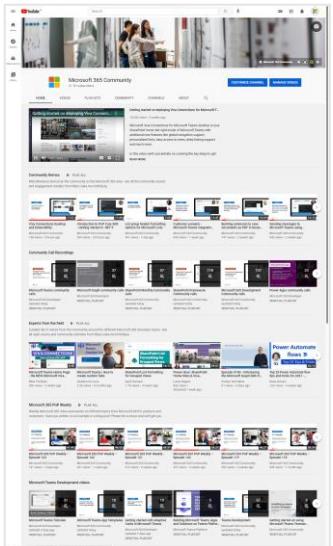
Changing the world one contribution at a time!

SEE INITIATIVES →



<https://aka.ms/community/home>

Microsoft 365 & Power Platform community videos



aka.ms/community/videos

Subscribe today!

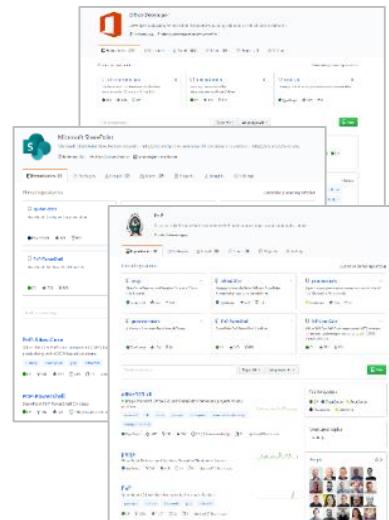
LinkedIn group for discussions



aka.ms/community/Li

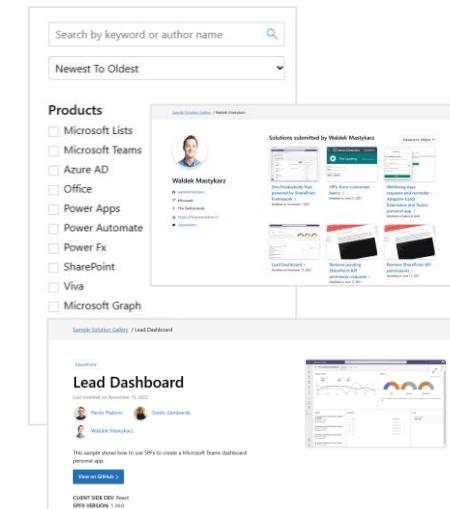
Join today!

Open-source



github.com/pnp
github.com/officedev
github.com/sharepoint
github.com/microsoftgraph

Sample galleries



aka.ms/community/samples

aka.ms/teams-samples

aka.ms/spfx-webparts

aka.ms/spfx-extensions

aka.ms/powerplatform-samples

aka.ms/list-formatting

Open-source initiatives and samples



aka.ms/community/home

What is PnPjs

Co-Maintainers



Beau Cameron
*Practice Lead / Sr.
Principal Architect
DMI*



Patrick Rogers
*Principal Product
Manager
Microsoft*

OPEN SOURCE INITIATIVE
WHOLLY CREATED AND
MAINTAINED BY THE
COMMUNITY



FLUENT LIBRARIES FOR
ACCESSING SHAREPOINT AND
MICROSOFT GRAPH REST APIs
IN A TYPE-SAFE WAY.

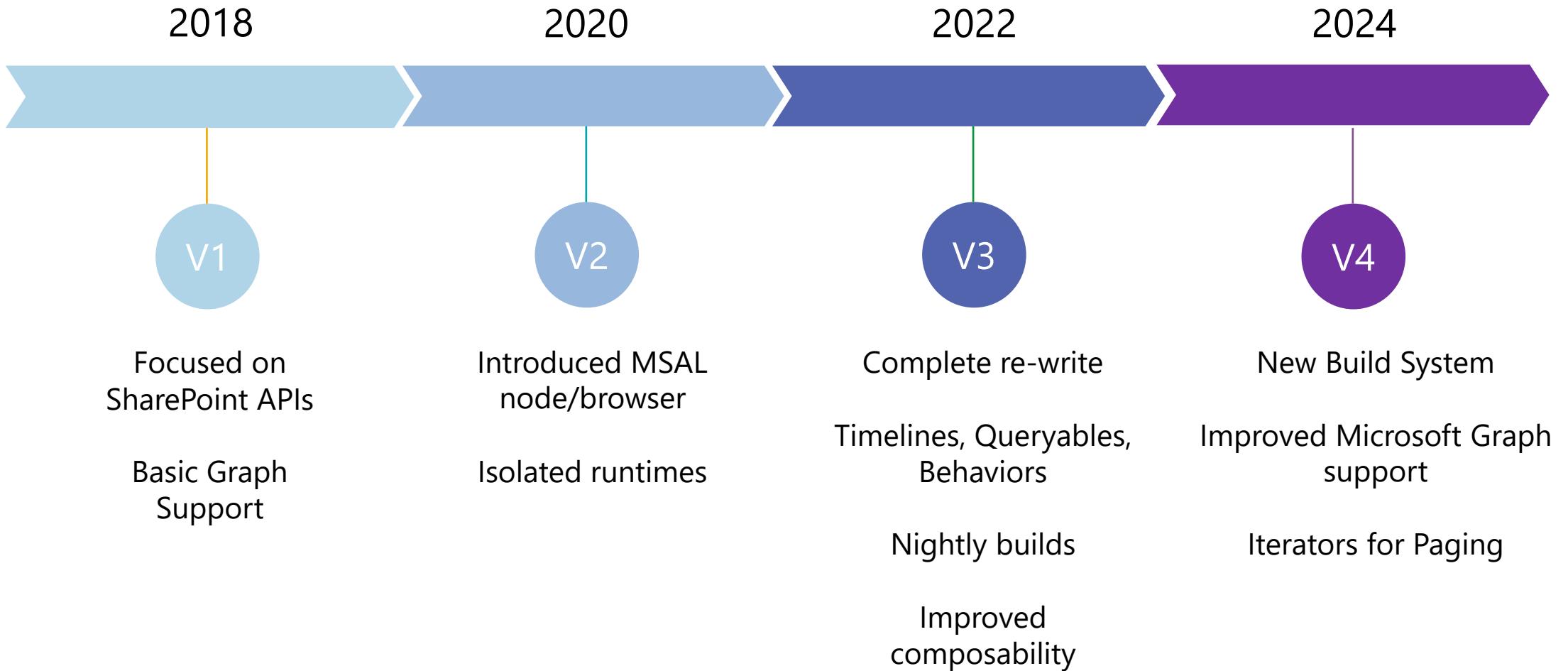


USED IN MORE THAN 37K+
TENANTS, GENERATING 37B+
REQUESTS EACH MONTH



ACCEPTS AND ENCOURAGES
CONTRIBUTIONS AND
CONSTRUCTIVE FEEDBACK
FROM THE COMMUNITY.

A brief history



Why use PnPjs?



Cuts development time

Fluent interface improves development and reduces errors



Reusable, but extensible

Developers can extend the native features with the extensibility model



One library, cross platform

Leverage the library across different scenarios without learning different frameworks

Traditional Approach

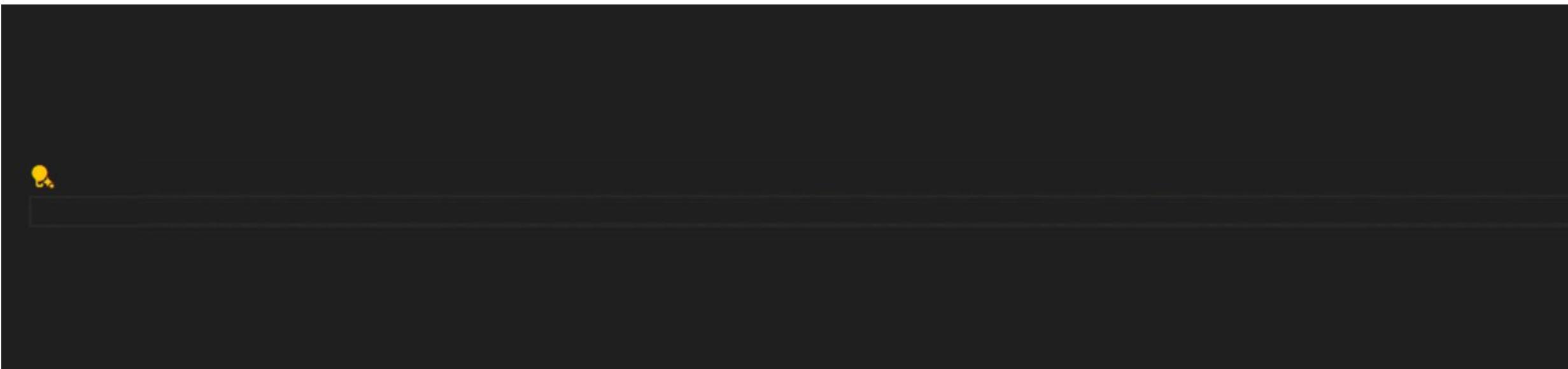
When writing API calls to SharePoint

string concatenation

```
api/web/lists/getbytitle('My List').items?$select=Title,Author/Id$expand=Author&$top=50
```

PnPjs - A fluent interface

Design method that relies on chaining

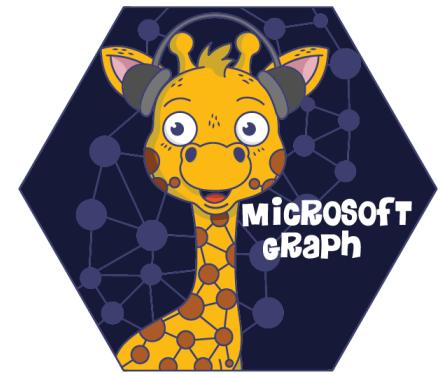


```
_api/web/lists/getbytitle('My List').items?$select=Title,Author/Id&$expand=Author&$top=50
```

What APIs do we support?



+



SharePoint

Attachments
Client-side pages
Column defaults
Comments and likes
Content types
Context info
Favorites
Features
Fields
Files
Folders
Forms
Group Site Manager
Hubsites
List Items

Lists
Navigation
Permissions
Profiles
Recycle Bin
Regional Settings
Related Items
Search
Security
Sharing
SharePoint Admin
Site Designs
Site Groups
Site Scripts
Site Users

Sites
Social
SP.Publishing.SitePageService
SP.Utilities.Utility
Subscriptions
Taxonomy
Tenant Properties
User Custom Actions
Views
Webs

Microsoft Graph

Admin
Analytics
App Catalog
Attachments
Bookings
Calendars
Cloud-Communications
Columns
Compliance
Contacts
Content Types
Conversations
Directory Objects
Files

Groups
Insights
Invitations
List items
Lists
Mail
Members
OneNote
Operations
Permissions
Photos
Places
Planner
Search

Shares
Subscriptions
Taxonomy
Teams
To-do
Users

Extending PnPjs

We expose methods for you to extend Graph & SharePoint APIs

SharePoint

- o spGet
- o spPost
- o spDelete
- o spPatch
- o spPostMerge
- o spPostDelete
- o spPostDeleteEtag

Graph

- o graphGet
- o graphPost
- o graphDelete
- o graphPatch
- o graphPut

```
const queryable = SPQueryable("https://cameroncode.sharepoint.com/sites/contosogroup8/_api/web/lists/getByTitle('SPFx List')/items(1)")  
  .using(AssignFrom(sp.web));  
const itemWithIdOf1 = await spGet(queryable);  
console.log(itemWithIdOf1);
```

Authentication
made *easier*



PnPjs includes built-in Auth Providers



SPFx

- SharePoint
- Microsoft Graph
- Auth Token
- MSAL

Browser

- MSAL

Node

- MSAL Nodejs
- Azure Identity

Getting Started

SharePoint Framework Code Demo

Packages

core	Provides shared functionality across all pnp libraries
queryable	Provides shared query functionality and base classes
sp	Provides a fluent api for working with SharePoint REST
graph	Provides a fluent api for working with Microsoft Graph
msaljsclient	Provides an msal wrapper suitable for use with PnPjs
nodejs	Provides functionality enabling the @pnp libraries within nodejs
azidjsclient	Provides an Azure Identity wrapper suitable for use with PnPjs
sp-admin	Provides a fluent api for working with M365 Tenant admin methods
logging	Light-weight, subscribable logging framework

```
npm install @pnp/sp @pnp/graph ...
```

package.json

{ package.json > ...

```
1  {
2    "name": "pnpjs-demo",
3    "version": "1.0.0",
4    "private": true,
5    "main": "lib/index.js",
6    "engines": {
7      "node": ">=16.13.0 <17.0.0 || >=18.17.1 <19.0.0"
8    },
9    ▷ Debug
10   "scripts": {
11     "build": "gulp bundle",
12     "clean": "gulp clean",
13     "test": "gulp test"
14   },
15   "dependencies": {
16     "react": "17.0.1",
17     "react-dom": "17.0.1",
18     "@microsoft/sp-core-library": "1.18.2",
19     "@microsoft/sp-property-pane": "1.18.2",
20     "@microsoft/sp-webpart-base": "1.18.2",
21     "@microsoft/sp-lodash-subset": "1.18.2",
22     "@microsoft/sp-office-ui-fabric-core": "1.18.2",
23     "@microsoft/sp-adaptive-card-extension-base": "1.18.2",
24     "tslib": "2.3.1",
25     "@pnp/sp": "^4.0.0", ←
26     "@pnp/graph": "^4.0.0",
27     "@pnp/logging": "^4.0.0",
28     "@fluentui/react": "8.106.4"
29   },
30   "devDependencies": {
31     "@types/react": "17.0.45",
```



Home

+ New

Promote

Page details

Analytics

Conversations

Teams

Documents

Notebook

Pages

Welcome to PnP JS Demo!

Update Item Titles

Recent

pnplist

List of documents:

TestLib

Site contents

Recycle bin

Edit

Title	Name	Size (KB)
General	General	0.00
Test Document	Test Document.docx	18.53
Other Doc	Other Doc.docx	20.50
Test Doc 1	Test Doc 1.docx	20.81
Curtisms	Curtisms.docx	34.14
ANDDDD ANOTHER	ANDDDD ANOTHER.docx	18.82
tool-finder-webparts	tool-finder-webparts.sppkg	3184.54
TestTeamForm	TestTeamForm.xlsx	21.41
Document	Document.docx	16.94
Document1	Document1.docx	16.94



Like



Save for later



p

pnpjsteam



Home

+ New

Promote

Page details

Analytics

Conversations

Teams

Documents

Notebook

Pages

Welcome to PnP JS Demo!

Update Item Titles

Recent

pnplist

TestLib

Site contents

Recycle bin

Edit

List of documents:

Title
General-Updated
Test Document.docx-Updated
Other Doc.docx-Updated
Test Doc 1.docx-Updated
Curtisms.docx-Updated
ANDDDD ANOTHER.docx-Updated
tool-finder-webparts.sppkg-Updated
TestTeamForm.xlsx-Updated
Document.docx-Updated
Document1.docx-Updated

Name	Size (KB)
General	0.00
Test Document.docx	18.53
Other Doc.docx	20.50
Test Doc 1.docx	20.81
Curtisms.docx	34.14
ANDDDD ANOTHER.docx	18.82
tool-finder-webparts.sppkg	3184.54
TestTeamForm.xlsx	21.41
Document.docx	16.94
Document1.docx	16.94

Like

1 View

Save for later

Establishing Context

SharePoint Framework Code Demo

...

← →

PnPjs-Demo

PnPjsWebPart.ts

TS PnPjsWebPart.ts X

src > webparts > pnPjs > TS PnPjsWebPart.ts > ...

```
1 import * as React from 'react';
2 import * as ReactDOM from 'react-dom';
3 import { Version } from '@microsoft/sp-core-library';
4 import { BaseClientSideWebPart } from '@microsoft/sp-webpart-base';
5 import pnpjs, { IpnnpjsProps } from './components/PnPjs';
6 import { getSP } from './pnnpjsConfig';
7
8 export interface IpnnpjsWebPartProps {
9   description: string;
10 }
11
12 export default class pnnpjsWebPart extends BaseClientSideWebPart<IpnnpjsWebPartProps> {
13
14   protected async onInit(): Promise<void> {
15     //Initialize our _sp object that we can then use in other packages without having to pass around the context.
16     // Check out pnnpjsConfig.ts for an example of a project setup file.
17     getSP(this.context);
18   }
19
20   public render(): void {
21     const element: React.ReactElement<IpnnpjsProps> = React.createElement(
22       pnpjs, {}
23     );
24   }
}
```

```
1  /* eslint-disable no-var */
2  import { WebPartContext } from "@microsoft/sp-webpart-base";
3  |
4  // import pnp and pnp logging system
5  import { spfi, SPFI, SPFx as spSPFx } from "@pnp/sp";
6  import { graphfi, GraphFI, SPFx as graphSPFx } from "@pnp/graph";
7  import { LogLevel, PnPLogging } from "@pnp/logging";
8  import "@pnp/sp/webs";
9  import "@pnp/sp/lists";
10 import "@pnp/sp/items";
11 import "@pnp/sp/batching";
12
13 var _sp: SPFI = null;
14 var _graph: GraphFI = null;
15
16 export const getSP = (context?: WebPartContext): SPFI => {
17  if (_sp === null) {
18    //You must add the @pnp/logging package to include the PnPLogging behavior it is no longer a peer dependency
19    // The LogLevel set's at what level a message will be written to the console
20    _sp = spfi().using(spSPFx(context)).using(PnPLogging(LogLevel.Warning));
21  }
22  return _sp;
23};
24
25 export const getGraph = (context?: WebPartContext): GraphFI => {
26  if (_graph === null) {
27    //You must add the @pnp/logging package to include the PnPLogging behavior it is no longer a peer dependency
28    // The LogLevel set's at what level a message will be written to the console
```

```
2 import * as ReactDOM from 'react-dom';
3 import { Version } from '@microsoft/sp-core-library';
4 import { BaseClientSideWebPart } from '@microsoft/sp-webpart-base';
5 import pnpjs, { IpnnpjsProps } from './components/PnPjs';
6 import { getSP } from './pnnpjsConfig';
7
8 export interface IpnnpjsWebPartProps {
9   description: string;
10 }
11
12 export default class pnnpjsWebPart extends BaseClientSideWebPart<IpnnpjsWebPartProps> {
13
14   protected async onInit(): Promise<void> {
15     //Initialize our _sp object that we can then use in other packages without having to pass around the context.
16     // Check out pnnpjsConfig.ts for an example of a project setup file.
17     getSP(this.context);
18   }
19
20   public render(): void {
21     const element: React.ReactElement<IpnnpjsProps> = React.createElement(
22       pnpjs, {}
23     );
24
25     ReactDOM.render(element, this.domElement);
26   }
27
28   protected onDispose(): void {
29     ReactDOM.unmountComponentAtNode(this.domElement);
30   }
}
```

Getting Data, Caching, Batching

SharePoint Framework Code Demo

PnPjs.tsx

```
src > webparts > pnPjs > components > PnPjs.tsx > IResponseItem > Title
1 import * as React from "react";
2
3 import { Logger, LogLevel } from "@pnp/logging";
4 import { Caching } from "@pnp/queryable";
5
6 import styles from "./PnPjs.module.scss";
7 import { getSP } from "../pnppjsConfig";
8 import { spfi, SPFI } from "@pnp/sp";
9 import { Label, PrimaryButton } from "@fluentui/react";
10 import { Web } from "@pnp/sp/webs";
11
12 export interface IFile {
13   Id: number;
14   Title: string;
15   Name: string;
16   Size: number;
17 }
18
19 export interface IResponseFile {
20   Length: number;
21 }
22
23 export interface IResponseItem {
24   Id: number;
25   File: IResponseFile;
26   FileLeafRef: string;
27   Title: string;
28 }
29
30 // eslint-disable-next-line @typescript-eslint/no-empty-interface
31 export interface IpnpjsProps {
32 }
33
34 export interface IpnpjsState {
35   items: IFile[];
36 }
37
38 export class pnppjsState implements IpnpjsState {
39   constructor(
40     public items: IFile[] = []
```

File Edit Selection View Go Run Terminal Help < > PnPjs-Demo PnPjs.tsx

EXPLORER ... PnPjs.tsx X

src > webparts > pnpjs > components > PnPjs.tsx > pnpjs

```
43
44  export default class pnpjs extends React.PureComponent<IpnpjsProps, IpnpjsState> {
45    private LOG_SOURCE = "pnpjs";
46    private LIBRARY_NAME = "Documents";
47    private _sp: SPFI;
48
49    constructor(props: IpnpjsProps) {
50      super(props);
51      this.state = new pnpjsState();
52      this._sp = getSP();
53    }
54
55    public componentDidMount(): void {
56      this._readAllFileSize();
57    }
58
59    private _readAllFileSize = async (): Promise<void> => {
60      ...
61    }
62
63    private _updateTitles = async (): Promise<void> => {
64      ...
65    }
66
67    private _getDemoItems = async (): Promise<void> => {
68      ...
69    }
70
71    public render(): React.ReactElement<IpnpjsProps> {
72      ...
73    }
74
75    ...
76
77  }
```

src > webparts > pnpjs > components > PnPjs.tsx > pnpjs

.vscode config dist lib node_modules release src webparts\pnpjs components PnPjs.module.scss PnPjs.ts loc pnpjsConfig.ts pnpjsModels.ts PnPjsWebPart.manifest.json PnPjsWebPart.ts index.ts teams eb15a628-721c-4678-96e0-08c2f4c8... eb15a628-721c-4678-96e0-08c2f4c8... temp .eslintrc.js .gitignore .npmignore .yo-rc.json gulpfile.js LICENSE.md package-lock.json package.json README.md tsconfig.json

OUTLINE TIMELINE BARREL GENERATOR: EXPORT VIEW

The screenshot shows the Visual Studio Code interface with the title bar "PnPjs-Demo". The left sidebar displays the project structure under "PNPJS-DEMO". The main editor area shows the file "PnPjs.tsx" with the following code:

```
src > webparts > pnPjs > components > PnPjs.tsx > pnpjs
44  export default class pnpjs extends React.PureComponent<IpnpjsProps, IpnpjsState> {
54
55    public componentDidMount(): void {
56      this._readAllFileSize();
57    }
58
59    private _readAllFileSize = async (): Promise<void> => {
60      try {
61        const spCache = spfi(this._sp).using(Caching({ store: "session" }));
62
63        const response: IResponseItem[] = await spCache.web.lists
64          .getByTitle(this.LIBRARY_NAME)
65          .items
66          .select("Id", "Title", "FileLeafRef", "File/Length")
67          .expand("File")();
68
69        const items: IFile[] = response.map((item: IResponseItem) => {
70          return {
71            Id: item.Id,
72            Title: item.Title || "Unknown",
73            Size: item.File?.Length || 0,
74            Name: item.FileLeafRef
75          };
76        });
77
78        this.setState({ items });
79      } catch (err) {
80        Logger.write(`${this.LOG_SOURCE} (_readAllFileSize) - ${JSON.stringify(err)}`, LogLevel.Error);
81      }
82    }
83
84    > private _updateTitles = async (): Promise<void> => { ...
85    }
86
87    > private _getDemoItems = async(): Promise<void> => { ...
88    }
89
90    > public render(): React.ReactElement<IpnpjsProps> { ...
91    }
92
93    > public componentDidMount(): void {
94      this._readAllFileSize();
95    }
96
97    > public componentDidUpdate(): void {
98      this._updateTitles();
99    }
100
101   > public componentWillUnmount(): void {
102     this._cache?.dispose();
103   }
104
105   > public componentDidUnmount(): void {
106     this._cache?.dispose();
107   }
108
109   > public render(): React.ReactElement<IpnpjsProps> { ...
110   }
111
112   > public componentDidMount(): void {
113     this._readAllFileSize();
114   }
115
116   > public componentDidUpdate(): void {
117     this._updateTitles();
118   }
119
120   > public componentWillUnmount(): void {
121     this._cache?.dispose();
122   }
123
124   > public componentDidUnmount(): void {
125     this._cache?.dispose();
126   }
127
128   > public render(): React.ReactElement<IpnpjsProps> { ...
129   }
130
131   > public componentDidMount(): void {
132     this._readAllFileSize();
133   }
134
135   > public componentDidUpdate(): void {
136     this._updateTitles();
137   }
138
139   > public componentWillUnmount(): void {
140     this._cache?.dispose();
141   }
142
143   > public componentDidUnmount(): void {
144     this._cache?.dispose();
145   }
146
147   > public render(): React.ReactElement<IpnpjsProps> { ...
148   }
149
150   > public componentDidMount(): void {
151     this._readAllFileSize();
152   }
153
154   > public componentDidUpdate(): void {
155     this._updateTitles();
156   }
157
158   > public componentWillUnmount(): void {
159     this._cache?.dispose();
160   }
161
162   > public componentDidUnmount(): void {
163     this._cache?.dispose();
164   }
165
166   > public render(): React.ReactElement<IpnpjsProps> { ...
167   }
168
169   > public componentDidMount(): void {
170     this._readAllFileSize();
171   }
172
173   > public componentDidUpdate(): void {
174     this._updateTitles();
175   }
176
177   > public componentWillUnmount(): void {
178     this._cache?.dispose();
179   }
180
181   > public componentDidUnmount(): void {
182     this._cache?.dispose();
183   }
184
185   > public render(): React.ReactElement<IpnpjsProps> { ...
186   }
187
188   > public componentDidMount(): void {
189     this._readAllFileSize();
190   }
191
192   > public componentDidUpdate(): void {
193     this._updateTitles();
194   }
195
196   > public componentWillUnmount(): void {
197     this._cache?.dispose();
198   }
199
200   > public componentDidUnmount(): void {
201     this._cache?.dispose();
202   }
203
204   > public render(): React.ReactElement<IpnpjsProps> { ...
205   }
206
207   > public componentDidMount(): void {
208     this._readAllFileSize();
209   }
210
211   > public componentDidUpdate(): void {
212     this._updateTitles();
213   }
214
215   > public componentWillUnmount(): void {
216     this._cache?.dispose();
217   }
218
219   > public componentDidUnmount(): void {
220     this._cache?.dispose();
221   }
222
223   > public render(): React.ReactElement<IpnpjsProps> { ...
224   }
225
226   > public componentDidMount(): void {
227     this._readAllFileSize();
228   }
229
230   > public componentDidUpdate(): void {
231     this._updateTitles();
232   }
233
234   > public componentWillUnmount(): void {
235     this._cache?.dispose();
236   }
237
238   > public componentDidUnmount(): void {
239     this._cache?.dispose();
240   }
241
242   > public render(): React.ReactElement<IpnpjsProps> { ...
243   }
244
245   > public componentDidMount(): void {
246     this._readAllFileSize();
247   }
248
249   > public componentDidUpdate(): void {
250     this._updateTitles();
251   }
252
253   > public componentWillUnmount(): void {
254     this._cache?.dispose();
255   }
256
257   > public componentDidUnmount(): void {
258     this._cache?.dispose();
259   }
260
261   > public render(): React.ReactElement<IpnpjsProps> { ...
262   }
263
264   > public componentDidMount(): void {
265     this._readAllFileSize();
266   }
267
268   > public componentDidUpdate(): void {
269     this._updateTitles();
270   }
271
272   > public componentWillUnmount(): void {
273     this._cache?.dispose();
274   }
275
276   > public componentDidUnmount(): void {
277     this._cache?.dispose();
278   }
279
280   > public render(): React.ReactElement<IpnpjsProps> { ...
281   }
282
283   > public componentDidMount(): void {
284     this._readAllFileSize();
285   }
286
287   > public componentDidUpdate(): void {
288     this._updateTitles();
289   }
290
291   > public componentWillUnmount(): void {
292     this._cache?.dispose();
293   }
294
295   > public componentDidUnmount(): void {
296     this._cache?.dispose();
297   }
298
299   > public render(): React.ReactElement<IpnpjsProps> { ...
300   }
301
302   > public componentDidMount(): void {
303     this._readAllFileSize();
304   }
305
306   > public componentDidUpdate(): void {
307     this._updateTitles();
308   }
309
310   > public componentWillUnmount(): void {
311     this._cache?.dispose();
312   }
313
314   > public componentDidUnmount(): void {
315     this._cache?.dispose();
316   }
317
318   > public render(): React.ReactElement<IpnpjsProps> { ...
319   }
320
321   > public componentDidMount(): void {
322     this._readAllFileSize();
323   }
324
325   > public componentDidUpdate(): void {
326     this._updateTitles();
327   }
328
329   > public componentWillUnmount(): void {
330     this._cache?.dispose();
331   }
332
333   > public componentDidUnmount(): void {
334     this._cache?.dispose();
335   }
336
337   > public render(): React.ReactElement<IpnpjsProps> { ...
338   }
339
340   > public componentDidMount(): void {
341     this._readAllFileSize();
342   }
343
344   > public componentDidUpdate(): void {
345     this._updateTitles();
346   }
347
348   > public componentWillUnmount(): void {
349     this._cache?.dispose();
350   }
351
352   > public componentDidUnmount(): void {
353     this._cache?.dispose();
354   }
355
356   > public render(): React.ReactElement<IpnpjsProps> { ...
357   }
358
359   > public componentDidMount(): void {
360     this._readAllFileSize();
361   }
362
363   > public componentDidUpdate(): void {
364     this._updateTitles();
365   }
366
367   > public componentWillUnmount(): void {
368     this._cache?.dispose();
369   }
370
371   > public componentDidUnmount(): void {
372     this._cache?.dispose();
373   }
374
375   > public render(): React.ReactElement<IpnpjsProps> { ...
376   }
377
378   > public componentDidMount(): void {
379     this._readAllFileSize();
380   }
381
382   > public componentDidUpdate(): void {
383     this._updateTitles();
384   }
385
386   > public componentWillUnmount(): void {
387     this._cache?.dispose();
388   }
389
390   > public componentDidUnmount(): void {
391     this._cache?.dispose();
392   }
393
394   > public render(): React.ReactElement<IpnpjsProps> { ...
395   }
396
397   > public componentDidMount(): void {
398     this._readAllFileSize();
399   }
400
401   > public componentDidUpdate(): void {
402     this._updateTitles();
403   }
404
405   > public componentWillUnmount(): void {
406     this._cache?.dispose();
407   }
408
409   > public componentDidUnmount(): void {
410     this._cache?.dispose();
411   }
412
413   > public render(): React.ReactElement<IpnpjsProps> { ...
414   }
415
416   > public componentDidMount(): void {
417     this._readAllFileSize();
418   }
419
420   > public componentDidUpdate(): void {
421     this._updateTitles();
422   }
423
424   > public componentWillUnmount(): void {
425     this._cache?.dispose();
426   }
427
428   > public componentDidUnmount(): void {
429     this._cache?.dispose();
430   }
431
432   > public render(): React.ReactElement<IpnpjsProps> { ...
433   }
434
435   > public componentDidMount(): void {
436     this._readAllFileSize();
437   }
438
439   > public componentDidUpdate(): void {
440     this._updateTitles();
441   }
442
443   > public componentWillUnmount(): void {
444     this._cache?.dispose();
445   }
446
447   > public componentDidUnmount(): void {
448     this._cache?.dispose();
449   }
450
451   > public render(): React.ReactElement<IpnpjsProps> { ...
452   }
453
454   > public componentDidMount(): void {
455     this._readAllFileSize();
456   }
457
458   > public componentDidUpdate(): void {
459     this._updateTitles();
460   }
461
462   > public componentWillUnmount(): void {
463     this._cache?.dispose();
464   }
465
466   > public componentDidUnmount(): void {
467     this._cache?.dispose();
468   }
469
470   > public render(): React.ReactElement<IpnpjsProps> { ...
471   }
472
473   > public componentDidMount(): void {
474     this._readAllFileSize();
475   }
476
477   > public componentDidUpdate(): void {
478     this._updateTitles();
479   }
480
481   > public componentWillUnmount(): void {
482     this._cache?.dispose();
483   }
484
485   > public componentDidUnmount(): void {
486     this._cache?.dispose();
487   }
488
489   > public render(): React.ReactElement<IpnpjsProps> { ...
490   }
491
492   > public componentDidMount(): void {
493     this._readAllFileSize();
494   }
495
496   > public componentDidUpdate(): void {
497     this._updateTitles();
498   }
499
500   > public componentWillUnmount(): void {
501     this._cache?.dispose();
502   }
503
504   > public componentDidUnmount(): void {
505     this._cache?.dispose();
506   }
507
508   > public render(): React.ReactElement<IpnpjsProps> { ...
509   }
510
511   > public componentDidMount(): void {
512     this._readAllFileSize();
513   }
514
515   > public componentDidUpdate(): void {
516     this._updateTitles();
517   }
518
519   > public componentWillUnmount(): void {
520     this._cache?.dispose();
521   }
522
523   > public componentDidUnmount(): void {
524     this._cache?.dispose();
525   }
526
527   > public render(): React.ReactElement<IpnpjsProps> { ...
528   }
529
530   > public componentDidMount(): void {
531     this._readAllFileSize();
532   }
533
534   > public componentDidUpdate(): void {
535     this._updateTitles();
536   }
537
538   > public componentWillUnmount(): void {
539     this._cache?.dispose();
540   }
541
542   > public componentDidUnmount(): void {
543     this._cache?.dispose();
544   }
545
546   > public render(): React.ReactElement<IpnpjsProps> { ...
547   }
548
549   > public componentDidMount(): void {
550     this._readAllFileSize();
551   }
552
553   > public componentDidUpdate(): void {
554     this._updateTitles();
555   }
556
557   > public componentWillUnmount(): void {
558     this._cache?.dispose();
559   }
560
561   > public componentDidUnmount(): void {
562     this._cache?.dispose();
563   }
564
565   > public render(): React.ReactElement<IpnpjsProps> { ...
566   }
567
568   > public componentDidMount(): void {
569     this._readAllFileSize();
570   }
571
572   > public componentDidUpdate(): void {
573     this._updateTitles();
574   }
575
576   > public componentWillUnmount(): void {
577     this._cache?.dispose();
578   }
579
580   > public componentDidUnmount(): void {
581     this._cache?.dispose();
582   }
583
584   > public render(): React.ReactElement<IpnpjsProps> { ...
585   }
586
587   > public componentDidMount(): void {
588     this._readAllFileSize();
589   }
590
591   > public componentDidUpdate(): void {
592     this._updateTitles();
593   }
594
595   > public componentWillUnmount(): void {
596     this._cache?.dispose();
597   }
598
599   > public componentDidUnmount(): void {
600     this._cache?.dispose();
601   }
602
603   > public render(): React.ReactElement<IpnpjsProps> { ...
604   }
605
606   > public componentDidMount(): void {
607     this._readAllFileSize();
608   }
609
610   > public componentDidUpdate(): void {
611     this._updateTitles();
612   }
613
614   > public componentWillUnmount(): void {
615     this._cache?.dispose();
616   }
617
618   > public componentDidUnmount(): void {
619     this._cache?.dispose();
620   }
621
622   > public render(): React.ReactElement<IpnpjsProps> { ...
623   }
624
625   > public componentDidMount(): void {
626     this._readAllFileSize();
627   }
628
629   > public componentDidUpdate(): void {
630     this._updateTitles();
631   }
632
633   > public componentWillUnmount(): void {
634     this._cache?.dispose();
635   }
636
637   > public componentDidUnmount(): void {
638     this._cache?.dispose();
639   }
640
641   > public render(): React.ReactElement<IpnpjsProps> { ...
642   }
643
644   > public componentDidMount(): void {
645     this._readAllFileSize();
646   }
647
648   > public componentDidUpdate(): void {
649     this._updateTitles();
650   }
651
652   > public componentWillUnmount(): void {
653     this._cache?.dispose();
654   }
655
656   > public componentDidUnmount(): void {
657     this._cache?.dispose();
658   }
659
660   > public render(): React.ReactElement<IpnpjsProps> { ...
661   }
662
663   > public componentDidMount(): void {
664     this._readAllFileSize();
665   }
666
667   > public componentDidUpdate(): void {
668     this._updateTitles();
669   }
670
671   > public componentWillUnmount(): void {
672     this._cache?.dispose();
673   }
674
675   > public componentDidUnmount(): void {
676     this._cache?.dispose();
677   }
678
679   > public render(): React.ReactElement<IpnpjsProps> { ...
680   }
681
682   > public componentDidMount(): void {
683     this._readAllFileSize();
684   }
685
686   > public componentDidUpdate(): void {
687     this._updateTitles();
688   }
689
690   > public componentWillUnmount(): void {
691     this._cache?.dispose();
692   }
693
694   > public componentDidUnmount(): void {
695     this._cache?.dispose();
696   }
697
698   > public render(): React.ReactElement<IpnpjsProps> { ...
699   }
700
701   > public componentDidMount(): void {
702     this._readAllFileSize();
703   }
704
705   > public componentDidUpdate(): void {
706     this._updateTitles();
707   }
708
709   > public componentWillUnmount(): void {
710     this._cache?.dispose();
711   }
712
713   > public componentDidUnmount(): void {
714     this._cache?.dispose();
715   }
716
717   > public render(): React.ReactElement<IpnpjsProps> { ...
718   }
719
720   > public componentDidMount(): void {
721     this._readAllFileSize();
722   }
723
724   > public componentDidUpdate(): void {
725     this._updateTitles();
726   }
727
728   > public componentWillUnmount(): void {
729     this._cache?.dispose();
730   }
731
732   > public componentDidUnmount(): void {
733     this._cache?.dispose();
734   }
735
736   > public render(): React.ReactElement<IpnpjsProps> { ...
737   }
738
739   > public componentDidMount(): void {
740     this._readAllFileSize();
741   }
742
743   > public componentDidUpdate(): void {
744     this._updateTitles();
745   }
746
747   > public componentWillUnmount(): void {
748     this._cache?.dispose();
749   }
750
751   > public componentDidUnmount(): void {
752     this._cache?.dispose();
753   }
754
755   > public render(): React.ReactElement<IpnpjsProps> { ...
756   }
757
758   > public componentDidMount(): void {
759     this._readAllFileSize();
760   }
761
762   > public componentDidUpdate(): void {
763     this._updateTitles();
764   }
765
766   > public componentWillUnmount(): void {
767     this._cache?.dispose();
768   }
769
770   > public componentDidUnmount(): void {
771     this._cache?.dispose();
772   }
773
774   > public render(): React.ReactElement<IpnpjsProps> { ...
775   }
776
777   > public componentDidMount(): void {
778     this._readAllFileSize();
779   }
780
781   > public componentDidUpdate(): void {
782     this._updateTitles();
783   }
784
785   > public componentWillUnmount(): void {
786     this._cache?.dispose();
787   }
788
789   > public componentDidUnmount(): void {
790     this._cache?.dispose();
791   }
792
793   > public render(): React.ReactElement<IpnpjsProps> { ...
794   }
795
796   > public componentDidMount(): void {
797     this._readAllFileSize();
798   }
799
800   > public componentDidUpdate(): void {
801     this._updateTitles();
802   }
803
804   > public componentWillUnmount(): void {
805     this._cache?.dispose();
806   }
807
808   > public componentDidUnmount(): void {
809     this._cache?.dispose();
810   }
811
812   > public render(): React.ReactElement<IpnpjsProps> { ...
813   }
814
815   > public componentDidMount(): void {
816     this._readAllFileSize();
817   }
818
819   > public componentDidUpdate(): void {
820     this._updateTitles();
821   }
822
823   > public componentWillUnmount(): void {
824     this._cache?.dispose();
825   }
826
827   > public componentDidUnmount(): void {
828     this._cache?.dispose();
829   }
830
831   > public render(): React.ReactElement<IpnpjsProps> { ...
832   }
833
834   > public componentDidMount(): void {
835     this._readAllFileSize();
836   }
837
838   > public componentDidUpdate(): void {
839     this._updateTitles();
840   }
841
842   > public componentWillUnmount(): void {
843     this._cache?.dispose();
844   }
845
846   > public componentDidUnmount(): void {
847     this._cache?.dispose();
848   }
849
850   > public render(): React.ReactElement<IpnpjsProps> { ...
851   }
852
853   > public componentDidMount(): void {
854     this._readAllFileSize();
855   }
856
857   > public componentDidUpdate(): void {
858     this._updateTitles();
859   }
860
861   > public componentWillUnmount(): void {
862     this._cache?.dispose();
863   }
864
865   > public componentDidUnmount(): void {
866     this._cache?.dispose();
867   }
868
869   > public render(): React.ReactElement<IpnpjsProps> { ...
870   }
871
872   > public componentDidMount(): void {
873     this._readAllFileSize();
874   }
875
876   > public componentDidUpdate(): void {
877     this._updateTitles();
878   }
879
880   > public componentWillUnmount(): void {
881     this._cache?.dispose();
882   }
883
884   > public componentDidUnmount(): void {
885     this._cache?.dispose();
886   }
887
888   > public render(): React.ReactElement<IpnpjsProps> { ...
889   }
890
891   > public componentDidMount(): void {
892     this._readAllFileSize();
893   }
894
895   > public componentDidUpdate(): void {
896     this._updateTitles();
897   }
898
899   > public componentWillUnmount(): void {
900     this._cache?.dispose();
901   }
902
903   > public componentDidUnmount(): void {
904     this._cache?.dispose();
905   }
906
907   > public render(): React.ReactElement<IpnpjsProps> { ...
908   }
909
910   > public componentDidMount(): void {
911     this._readAllFileSize();
912   }
913
914   > public componentDidUpdate(): void {
915     this._updateTitles();
916   }
917
918   > public componentWillUnmount(): void {
919     this._cache?.dispose();
920   }
921
922   > public componentDidUnmount(): void {
923     this._cache?.dispose();
924   }
925
926   > public render(): React.ReactElement<IpnpjsProps> { ...
927   }
928
929   > public componentDidMount(): void {
930     this._readAllFileSize();
931   }
932
933   > public componentDidUpdate(): void {
934     this._updateTitles();
935   }
936
937   > public componentWillUnmount(): void {
938     this._cache?.dispose();
939   }
940
941   > public componentDidUnmount(): void {
942     this._cache?.dispose();
943   }
944
945   > public render(): React.ReactElement<IpnpjsProps> { ...
946   }
947
948   > public componentDidMount(): void {
949     this._readAllFileSize();
950   }
951
952   > public componentDidUpdate(): void {
953     this._updateTitles();
954   }
955
956   > public componentWillUnmount(): void {
957     this._cache?.dispose();
958   }
959
960   > public componentDidUnmount(): void {
961     this._cache?.dispose();
962   }
963
964   > public render(): React.ReactElement<IpnpjsProps> { ...
965   }
966
967   > public componentDidMount(): void {
968     this._readAllFileSize();
969   }
970
971   > public componentDidUpdate(): void {
972     this._updateTitles();
973   }
974
975   > public componentWillUnmount(): void {
976     this._cache?.dispose();
977   }
978
979   > public componentDidUnmount(): void {
980     this._cache?.dispose();
981   }
982
983   > public render(): React.ReactElement<IpnpjsProps> { ...
984   }
985
986   > public componentDidMount(): void {
987     this._readAllFileSize();
988   }
989
990   > public componentDidUpdate(): void {
991     this._updateTitles();
992   }
993
994   > public componentWillUnmount(): void {
995     this._cache?.dispose();
996   }
997
998   > public componentDidUnmount(): void {
999     this._cache?.dispose();
1000
1001   > public render(): React.ReactElement<IpnpjsProps> { ...
1002   }
```

PnPJS-DEMO

PnPJS.tsx

```
src > webparts > pnPjs > components > PnPjs.tsx > pnpjs
  44  export default class pnpjs extends React.PureComponent<IpnpjsProps, IpnpjsState> {
  45    public render(): React.ReactElement<IpnpjsProps> {
  46      try {
  47        return (
  48          <div data-component={this.LOG_SOURCE} className={styles.pnpjs}>
  49            <Label>Welcome to PnP JS Demo!</Label>
  50            <PrimaryButton onClick={this._updateTitles}>Update Item Titles</PrimaryButton>
  51            <br/><br/>
  52            <PrimaryButton onClick={this._getDemoItems}>Get Items from Demo Site</PrimaryButton>
  53            <Label>List of documents:</Label>
  54            <table width="100%">
  55              <tr>
  56                <td><strong>Title</strong></td>
  57                <td><strong>Name</strong></td>
  58                <td><strong>Size (KB)</strong></td>
  59              </tr>
  60              {this.state.items && this.state.items.map((item, idx) => {
  61                return (
  62                  <tr key={idx}>
  63                    <td>{item.Title}</td>
  64                    <td>{item.Name}</td>
  65                    <td>{(item.Size / 1024).toFixed(2)}</td>
  66                  </tr>
  67                );
  68              })
  69            </table>
  70          </div>
  71        );
  72      } catch (err) {
  73        Logger.write(` ${this.LOG_SOURCE} (render) - ${JSON.stringify(err)}`, LogLevel.Error);
  74      }
  75    }
  76  }
  77}
```



pnpjsteam

Public group Not following 2 members

Home

[+ New](#) Promote Page details Analytics

Published 4/22/2024 Share Edit

Conversations

Teams

Documents

Notebook

Pages

Recent

pnplist

TestLib

Site contents

Recycle bin

Edit

PnPjs Demo



Julie Turner

Welcome to PnP JS Demo!

Update Item Titles

List of documents:

Title	Name	Size (KB)
General	General	0.00
Test Document	Test Document.docx	18.53
Other Doc	Other Doc.docx	20.50
Test Doc 1	Test Doc 1.docx	20.81
Curtisms	Curtisms.docx	34.14
ANDDDD ANOTHER	ANDDDD ANOTHER.docx	18.82
tool-finder-webparts	tool-finder-webparts.sppkg	3184.54
TestTeamForm	TestTeamForm.xlsx	21.41
Document	Document.docx	16.94
Document1	Document1.docx	16.94

Like Save for later

File Edit Selection View Go Run Terminal Help PnPjs-Demo PnPjs.tsx

EXPLORER ... PnPjs.tsx x

src > webparts > pnPjs > components > PnPjs.tsx > pnpjs > _getDemoItems

```
44  export default class pnpjs extends React.PureComponent<IpnpjsProps, IpnpjsState> {
83
84  private _updateTitles = async (): Promise<void> => {
85    try {
86      const [batchedSP, execute] = this._sp.batched();
87
88      const items: IFile[] = JSON.parse(JSON.stringify(this.state.items));
89
90      const res: { Id: number, Title: string }[] = [];
91      for (let i = 0; i < items.length; i++) {
92        batchedSP.web.lists
93          .getByTitle(this.LIBRARY_NAME)
94            .items
95              . getById(items[i].Id)
96                .update({ Title: `${items[i].Name}-Updated` })
97                .then(r => res.push(r));
98
99      await execute();
100
101     // res object only contains eTag of changed item.
102     // Dirty update of UI
103     for (let i = 0; i < res.length; i++) {
104       items[i].Title = `${items[i].Name}-Updated`;
105     }
106     this.setState({ items });
107   } catch (err) {
108     Logger.write(`${this.LOG_SOURCE} (_updateTitles) - ${JSON.stringify(err)}`, LogLevel.Error);
109   }
110 }
111
112
113 > private _getDemoItems = async(): Promise<void> => [
114
115
116
117 ]
```

OUTLINE
TIMELINE
BARREL GENERATOR: EXPORT VIEW

Cross Site Calls

SharePoint Framework Code Demo



p

pnpjsteam

Home

+ New

Promote

Page details

Analytics

Conversations

Teams

Documents

Notebook

Pages

Welcome to PnP JS Demo!

Update Item Titles

Recent

pnplist

TestLib

Site contents

Recycle bin

Edit

Get Items from Demo Site

List of documents:

Title
General-Updated
Test Document.docx-Updated
Other Doc.docx-Updated
Test Doc 1.docx-Updated
Curtisms.docx-Updated
ANDDDD ANOTHER.docx-Updated
tool-finder-webparts.sppkg-Updated
TestTeamForm.xlsx-Updated
Document.docx-Updated
Document1.docx-Updated

Name	Size (KB)
General	0.00
Test Document.docx	18.53
Other Doc.docx	20.50
Test Doc 1.docx	20.81
Curtisms.docx	34.14
ANDDDD ANOTHER.docx	18.82
tool-finder-webparts.sppkg	3184.54
TestTeamForm.xlsx	21.41
Document.docx	16.94
Document1.docx	16.94

Home

+ New

Published 4/22/2024

Conversations

Teams

Documents

Create

Pages

Recent

pnplist

TestLib

Site contents

Recycle bin

Edit

PnPjs Demo



Julie Turner

Welcome to PnP JS Demo!

[Update Item Titles](#)[Get Items from Demo Site](#)

List of documents:

Title
General-Updated
Test Document.docx-Updated
Other Doc.docx-Updated
Test Doc 1.docx-Updated
Curtisms.docx-Updated
ANDDDD ANOTHER.docx-Updated
tool-finder-webparts.sppkg-Updated
TestTeamForm.xlsx-Updated
Document.docx-Updated
Document1.docx-Updated

Name	Size (KB)
General	0.00
Test Document.docx	18.53
Other Doc.docx	20.50
Test Doc 1.docx	20.81
Curtisms.docx	34.14
ANDDDD ANOTHER.docx	18.82
tool-finder-webparts.sppkg	3184.54
TestTeamForm.xlsx	21.41
Document.docx	16.94
Document1.docx	16.94
123456	0.00
WidgetPricingSheet.xlsx-Updated	17.08
AutomatedModel.xlsx-Updated	23.59
Unknown	19.15
Unknown	0.09
Unknown	18.90
Unknown	17.61

```
84     private _updateTitles = async (): Promise<void> => {
```

```
110 }
```

```
111 }
```

```
112
```

```
113     private _getDemoItems = async(): Promise<void> => {
```

```
137         try { ... }
```

```
138     } catch (err) {
```

```
139         Logger.write(` ${this.LOG_SOURCE} (_getDemoItems) - ${JSON.stringify(err)}`, LogLevel.Error);
```

```
140     }
```

```
141 }
```

```
142     public render(): React.ReactElement<IpnPjsProps> {
```

```
143         try {
```

```
144             return (
```

```
145                 <div data-component={this.LOG_SOURCE} className={styles.pnPjs}>
```

```
146                     <Label>Welcome to PnP JS Demo!</Label>
```

```
147                     <PrimaryButton onClick={this._updateTitles}>Update Item Titles</PrimaryButton>
```

```
148                     <br/><br/>
```

```
149                     <PrimaryButton onClick={this._getDemoItems}>Get Items from Demo Site</PrimaryButton>
```

```
150                     <Label>List of documents:</Label>
```

```
151                     <table width="100%">
```

```
152                         <tr>
```

```
153                             <td><strong>Title</strong></td>
```

```
154                             <td><strong>Name</strong></td>
```

```
155                             <td><strong>Size (KB)</strong></td>
```

```
156                         </tr>
```



```
111 }
```

```
112
```

```
113     private _getDemoItems = async(): Promise<void> => {
```

```
137         try { ... }
```

```
138     } catch (err) {
```

```
139         Logger.write(` ${this.LOG_SOURCE} (_getDemoItems) - ${JSON.stringify(err)}`, LogLevel.Error);
```

```
140     }
```

```
141 }
```

```
142     public render(): React.ReactElement<IpnPjsProps> {
```

```
143         try {
```

```
144             return (
```

```
145                 <div data-component={this.LOG_SOURCE} className={styles.pnPjs}>
```

```
146                     <Label>Welcome to PnP JS Demo!</Label>
```

```
147                     <PrimaryButton onClick={this._updateTitles}>Update Item Titles</PrimaryButton>
```

```
148                     <br/><br/>
```

```
149                     <PrimaryButton onClick={this._getDemoItems}>Get Items from Demo Site</PrimaryButton>
```

```
150                     <Label>List of documents:</Label>
```

```
151                     <table width="100%">
```

```
152                         <tr>
```

```
153                             <td><strong>Title</strong></td>
```

```
154                             <td><strong>Name</strong></td>
```

```
155                             <td><strong>Size (KB)</strong></td>
```

```
156                         </tr>
```



```
111 }
```

```
112
```

```
113     private _getDemoItems = async(): Promise<void> => {
```

```
137         try { ... }
```

```
138     } catch (err) {
```

```
139         Logger.write(` ${this.LOG_SOURCE} (_getDemoItems) - ${JSON.stringify(err)}`, LogLevel.Error);
```

```
140     }
```

```
141 }
```

```
142     public render(): React.ReactElement<IpnPjsProps> {
```

```
143         try {
```

```
144             return (
```

```
145                 <div data-component={this.LOG_SOURCE} className={styles.pnPjs}>
```

```
146                     <Label>Welcome to PnP JS Demo!</Label>
```

```
147                     <PrimaryButton onClick={this._updateTitles}>Update Item Titles</PrimaryButton>
```

```
148                     <br/><br/>
```

```
149                     <PrimaryButton onClick={this._getDemoItems}>Get Items from Demo Site</PrimaryButton>
```

```
150                     <Label>List of documents:</Label>
```

```
151                     <table width="100%">
```

```
152                         <tr>
```

```
153                             <td><strong>Title</strong></td>
```

```
154                             <td><strong>Name</strong></td>
```

```
155                             <td><strong>Size (KB)</strong></td>
```

```
156                         </tr>
```



```
44  export default class PnPjs extends React.PureComponent<IPnPjsProps, IPnPjsState> {
45    state = {
46      items: []
47    }
48
49    private _getDemoItems = async(): Promise<void> => {
50      try {
51        const oldItems = this.state.items;
52
53        
54        const webUrl = `${window.location.origin}/sites/Demos`;
55
56        // Optionally
57        // const webSP = spfi(webUrl).using(SPFx({ pageContext: this._pageContext }));
58        // const web = webSP.web;
59
60        
61        const web = Web([this._sp.web, webUrl]);
62
63        let newItems: IFile[] = [];
64
65        if(web){
66          const response: IResponseItem[] = await web.lists
67            .getByTitle(this.LIBRARY_NAME)
68            .items
69            .select("Id", "Title", "FileLeafRef", "File/Length")
70            .expand("File")();
71
72
73          newItems = response.map((item: IResponseItem) => {
74            return {
75              Id: item.Id,
76              Title: item.Title || "Unknown",
77              Size: item.File?.Length || 0,
78              Name: item.FileLeafRef
79            };
80          });
81
82        }
83
84        const items = oldItems.concat(newItems);
85
86        this.setState({ items });
87      } catch (err) {
88        console.error(err);
89      }
90    }
91
92    render() {
93      return (
94        <div>
95          <Table>
96            <thead>
97              <tr>
98                <th>Title</th>
99                <th>Size</th>
100               <th>Name</th>
101             </tr>
102           </thead>
103           <tbody>
104             {this.state.items.map(item => (
105               <tr>
106                 <td>{item.Title}</td>
107                 <td>{item.Size}</td>
108                 <td>{item.Name}</td>
109               </tr>
110             ))}
111           </tbody>
112         </Table>
113       </div>
114     );
115   }
116 }
```

Behaviors

How to create your own

Behaviors

Recipes for extending PnPjs

- Bearer Token
- Browser Fetch
- Browser Fetch with Retry
- Caching
- Inject Headers
- Node Fetch
- Parsers
 - Json
 - Text
 - Blob
 - Buffer
 - Header
- Resolvers

```
const sp = spfi().using(Caching());  
  
const items = await sp.web.lists.getByTitle("Items").items();
```

Behaviors allow *hooking* into lifecycle events

Examples

Pre

- modify the request before it's sent to the server
- inject your own headers

Auth

- modify and include your own bearer token

Send

- provide your own fetch client

Parse

- build your own parser for handling different responses

Data

- hook into the fully parsed response data

Anatomy of a behavior

```
export function MyCustomParser(): TimelinePipe<Query...> {
    return (instance: Queryable) => {
        instance.on.parse.replace(async (url: URL, response: Response, result: any): Promise<[URL, Response, any]> => {
            if(response.ok && typeof result !== "undefined") {
                result = await response.json();
            }
            return [url, response, result];
        });
        return instance;
    };
}
```

Clear - Remove active behaviors

Prepend - Apply before any other behaviors

Replace - Replace all behaviors with this

Creating our own Behavior

```
function InjectHeaders(headers, prepend = false) {
  return (instance) => {
    instance.on.pre.prepend(async (url, init, result) => {
      init.headers = { ...init.headers, ...headers };
      return [url, init, result];
    });
    return instance;
  };
}
const sp = spfi().using(
  SPBrowser({ baseUrl: settings.testing.sp.url }),
  MSAL({ configuration: settings.testing.sp.msal.init, authParams: { scopes: settings.testing.sp.msal.scopes } })
).behavior(InjectHeaders({
  "Accept": "application/json;odata=verbose",
  "Beau-Custom": "myCustomHeader"
}));
const web = await sp.web();
```

NodeJS project

Azure Functions Demo

package.json > ...

```
1  {
2    "name": "azfuncnode",
3    "version": "1.0.0",
4    "description": "",
5    "type": "module",
6    ▷ Debug
7    "scripts": {
8      "build": "tsc",
9      "watch": "tsc -w",
10     "clean": "rimraf dist",
11     "prestart": "npm run clean && npm run build",
12     "start": "func start",
13     "test": "echo \"No tests yet...\""
14   },
15   "dependencies": {
16     "@azure/functions": "4.4.0",
17     "@pnp/azidjsclient": "4.0.0",
18     "@pnp/graph": "4.0.0",
19     "@pnp/nodejs": "4.0.0",
20     "@pnp/sp": "4.0.0",
21     "applicationinsights": "2.9.5"
22   },
23   "devDependencies": {
24     "@types/node": "^18.x",
25     "typescript": "^5.0.0",
26     "rimraf": "^5.0.0"
27   },
28   "main": "dist/src/functions/*.js"
}
```

package.json

```
1  {
2    "compilerOptions": {
3      "module": "ESNext",
4      "target": "ESNext",
5      "outDir": "dist",
6      "rootDir": ".",
7      "sourceMap": true,
8      "strict": false,
9      "moduleResolution": "Node",
10     "allowSyntheticDefaultImports": true,
11     "typeRoots": [
12       "./node_modules/@types"
13     ],
14   }
15 }
```

local.settings.example.json > {} Values

```
1 {  
2   "IsEncrypted": false,  
3   "Values": {  
4     "AzureWebJobsStorage": "UseDevelopmentStorage=true",  
5     "FUNCTIONS_WORKER_RUNTIME": "node",  
6     "AzureWebJobsFeatureFlags": "EnableWorkerIndexing",  
7     "APPLICATIONINSIGHTS_CONNECTION_STRING": "",  
8     "SiteUrl": "https://contoso.sharepoint.com/sites/pnpjs",  
9     "Tenant": "contoso",  
10    "ListGUID": "99999999-9999-9999-9999-999999999999",  
11    // Next 3 setting are only for local Azure Identity debugging  
12    "AZURE_CLIENT_ID": "99999999-9999-9999-9999-999999999999",  
13    "AZURE_TENANT_ID": "99999999-9999-9999-9999-999999999999",  
14    // path to .pem file  
15    "AZURE_CLIENT_CERTIFICATE_PATH": "c:\\cert.pem"  
16  }  
17 }
```

local.settings.json

pnpjsService.ts

```
pnppjsService.ts src\common  
NODE  
blobstorage_  
queuestorage_  
de  
_modules  
common  
models.ts  
pnpjsService.ts  
actions  
httpTrigger.ts  
rite_db_blob_json  
rite_db_blob_extent_json  
rite_db_queue_json  
rite_db_queue_extent_json  
rite_db_table_json  
ignore  
more  
ison  
settings.example.json  
settings.json  
age-lock.json  
age.json  
OME.md  
fig.json  
  
RESOURCES  
WORKSPACE  
HELP AND FEEDBACK  
RIPTS  
UR  
1 import AppInsights from 'applicationinsights';  
2 import { DefaultAzureCredential } from "@azure/identity";  
3 import { MyItem } from './models.js';  
4  
5 import { spfi, SPFI } from "@pnp/sp";  
6 import { AzureIdentity } from "@pnp/azidjsclient";  
7 import { GraphDefault, SPDefault } from "@pnp/nodejs";  
8 import "@pnp/sp/webs/index.js";  
9 import "@pnp/sp/lists/index.js";  
10 import "@pnp/sp/items/index.js";  
11 import { graphfi, GraphFI } from '@pnp/graph';  
12  
13 export interface IPnpjsService {  
14     Init: () => Promise<boolean>;  
15     GetListItem: (id: string) => Promise<MyItem>;  
16 }  
17  
18 export class PnpjsService implements IPnpjsService {  
19     private LOG_SOURCE = "PnpjsService";  
20     private _ready: boolean = false;  
21  
22     private _sp: SPFI = null;  
23     private _graph: GraphFI = null;  
24  
25     public constructor() { }  
26  
27 >     public async Init(): Promise<boolean> { ...  
28 }  
29  
30     public get ready(): boolean {  
31         return this._ready;  
32     }  
33  
34 >     public async GetListItem(id: string): Promise<MyItem> { ...  
35 }  
36  
37     }  
38  
39     export const pnppjs: IPnpjsService = new PnpjsService();  
40
```

```
18  export class PnpjsService implements IPnpjsService {  
22    private _sp: SPFI = null;  
23    private _graph: GraphFI = null;  
24  
25    public constructor() { }  
26  
27    public async Init(): Promise<boolean> {  
28      let retVal = false;  
29      try {  
30  
31        const credential = new DefaultAzureCredential();  
32  
33        this._sp = spfi(process.env.SiteUrl).using(SPDefault({}),  
34          AzureIdentity(credential, [`https://${process.env.Tenant}.sharepoint.com/.default`], null));  
35  
36        this._graph = graphfi().using(GraphDefault({}), AzureIdentity(credential, [`https://graph.microsoft.com/.default`], null));  
37  
38        this._ready = true;  
39        retVal = true;  
40        AppInsights.defaultClient.trackTrace({  
41          message: 'Init success',  
42          properties: {  
43            source: this.LOG_SOURCE,  
44            method: "Init"  
45          },  
46          severity: AppInsights.Contracts.SeverityLevel.Verbose  
47        });  
48      } catch (err) {  
49        AppInsights.defaultClient.trackException({  
50          exception: err,  
51          severity: AppInsights.Contracts.SeverityLevel.Critical  
52        });  
53      }  
54    }  
55  }
```

```
18  export class PnpjsService implements IPnpjsService {
57
58      public get ready(): boolean {
59          return this._ready;
60      }
61
62      public async GetListItem(id: string): Promise<MyItem> {
63          let retVal: MyItem = null;
64          try {
65              const item = await this._sp.web.lists.getById(process.env.ListGUID).items.getById(+id)();
66              retVal = { Id: item.Id, Title: item.Title, Description: item.Description };
67              AppInsights.defaultClient.trackTrace({
68                  message: 'GetListItem success',
69                  properties: {
70                      source: this.LOG_SOURCE,
71                      method: "GetListItem"
72                  },
73                  severity: AppInsights.Contracts.SeverityLevel.Verbose
74              });
75          } catch (err) {
76              AppInsights.defaultClient.trackException({
77                  exception: err,
78                  severity: AppInsights.Contracts.SeverityLevel.Critical,
79                  properties: { source: this.LOG_SOURCE, method: "GetListItem" }
80              });
81          }
82          return retVal;
83      }
84  }
85
86  export const pnpjs: IPnpjsService = new PnpjsService();
87
```

```
1 import { app, HttpRequest, HttpResponseInit, InvocationContext } from "@azure/functions";
2 import AppInsights from 'applicationinsights';
3 import { pnpjs } from "../common/pnpjsService.js";
4
5 export async function httpTrigger(request: HttpRequest, context: InvocationContext): Promise<HttpResponseInit> {
6   const LOG_SOURCE = "httpTrigger";
7   // Initialize Application Insights, setting AutoDependencyCorrelation to true so that all logs for each run are correlated together
8   // in App Insights.
9   AppInsights.setup(process.env.APPLICATIONINSIGHTS_CONNECTION_STRING).setAutoDependencyCorrelation(true);
10
11   // start the client
12   AppInsights.start();
13 >   AppInsights.defaultClient.trackEvent({ ... });
14
15   // Set the default response to a 200 with an empty body.
16   let retVal: HttpResponseInit = { status: 200, body: "" };
17
18 >   try { ...
19 >     } catch (err) { ...
20 >       }
21
22   // Return the appropriate response to the requestor.
23   return retVal;
24
25 > };
26
27 app.http('httpTrigger', {
28   methods: ['GET'],
29   authLevel: 'anonymous',
30   handler: httpTrigger
31 });
32
33 
```

```
20 // ...
21
22 // Set the default response to a 200 with an empty body.
23 let retVal: HttpResponseInit = { status: 200, body: "" };
24
25 try {
26     // If the request is a GET
27     if (request.method == "GET") {
28         const id = request.query.get("id");
29         if (id != null) {
30             const ready = await pnpjs.Init();
31             if (ready) {
32                 const result = await pnpjs.GetListItem(id);
33                 if (result != null) {
34                     retVal = { status: 200, body: JSON.stringify(result) };
35                     AppInsights.defaultClient.trackTrace({ ...
36                         });
37                 } else {
38                     retVal = { status: 400, body: "Item not found." };
39                     AppInsights.defaultClient.trackTrace({ ...
40                         });
41                 }
42             }
43         }
44     }
45     ...
46 }
47
48 } catch (err) {
49 }
50
51
52 // Return the appropriate response to the requestor.
53 return retVal;
54
55 }
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77 }
```

What's changed in V4

Updates

- Bug fixes!
- Async Iterators
- File Labels
- Followed Sites Support
- Get All Sites()
- Mail & Mail Folders Operations
- Site & List Operations
- Drive Items (Files)

Removed/Deprecated

- Paged()
- Drive Item – Move
- Drive Item – Set Content
- SP Taxonomy
- Get-All

New Modules

- admin
- analytics
- app catalog
- compliance
- files
- list-item
- mail
- operations
- permissions
- places
- taxonomy
- to-do

Async Iterators – paging *improved*



Paging in v3

```
let items = await sp.web.lists.getByTitle("SPFx List").items.top(300).getPaged();

if(items.hasNext){
    items = await items.getNext();
    console.log(items.results.length);
}
```

V4 Paging w/ Async Iterators

```
for await (const items of sp.web.lists.getByTitle("SPFx List").items.top(300)) {  
  console.log(items.length);  
  // perform some logic  
  //if logic is met, break out of loop  
  break;  
}
```

V4 Paging w/ Async Iterators

```
for await (const items of sp.web.lists.getByTitle("SPFx List").items.top(1)) {  
  console.log(items.length);  
  if (items[0].Id > 5) {  
    break;  
  }  
}
```

V4 Updates – Removal of 'data'

```
let item = await sp.web.lists.getByTitle("SPFx List").items.add({
  "Title": "Test Item"
});

console.log(item.Title);

let itemQueryable = await sp.web.lists.getByTitle("SPFx List").items.getById(item.data.Id);
```

Questions



Resources

Repository: <https://github.com/pnp/pnpjs>

Documentation: <https://pnp.github.io/pnpjs/>

Today's Sample:
<https://github.com/pnp/pnpjs/tree/version-4/samples/spfx-react-components>

How was the session?

Search for WHOVA
in the App Store or
Google Play



Fill out the Surveys in the
TechCon 365 DC Event App
and be eligible to win **PRIZES!**

