

Data Access

© 2018 -2021 ALEXANDER.PAJER@INTEGRATIONS.AT

Contents

Data Access Options

REST and OData Overview

Implementing CRUD Operation using REST

CSOM

PnP/PnPjs

Graph & Webhooks

Client Side Data Access Options

Client Side Data Access Options

Several choices available depending of Environment & Project Type

	JSOM	Raw REST	PnP JS Core	Graph
Characteristics	<ul style="list-style-type: none">• Introduced in SP2010• Mimics CSOM• Works with delegates• Does not support promises• Not being invested in anymore	<ul style="list-style-type: none">• The same across technology stacks• Easy for GET requests• Complex for non-GET requests• Evergreen	<ul style="list-style-type: none">• Fluent API• Works natively with Promises• Open-source community-driven initiative	<ul style="list-style-type: none">• Easy to learn• Covers more M365 Ressources
Future-proof	!	✓	✓	✓
Coverage	Fair	Excellent	Fair	Fair
Ease of use	Hard	Moderate	Easy	Easy

REST and OData Overview

REST

REST is an architectural style, not a specification

The Web is a graph of linked Resources

Resources are identified by URI's

Resources support a fixed set of operations

- In practice, these are defined by HTTP -> CRUDs

Applications follow links to achieve late binding

HTTP Verbs

RESTful (Representational State Transfer) web services use HTTP verbs to map CRUD operations to HTTP methods.

RESTful web services expose either a collection resource (representational of a list) or an element resource (representational of a single item in the list)

HTTP verbs are used as follows;

- Read (GET) > retrieve one or many resources – List or a specific Listitem
- Create (POST) > create a new resource.
- Update (PUT) > update an existing resource.
- Delete (DELETE) > delete an existing resource.

REST Urls in SharePoint

CSOM URLs can go through _api folder

- Simplifies URLs that need to be built
- Removes client.svc file name from URL

You can replace this URL

- http:// https://integrationsonline.sharepoint.com/sites/learning/_vti_bin/client.svc/web/title
- SPSite -> [TLS] -> SPWeb -> Site -> Lists -> [SPFolders] -> SPListItem -> SPField -> SPFieldType

With this URL

- http:// https://integrationsonline.sharepoint.com/sites/learning/_api/lists

Mapping Objects to Resources

Example REST URLs targeting SharePoint sites

`_api/web/lists`

`_api/web/lists/getByTitle('Announcements')`

`_api/web/getAvailableWebTemplates(lcid=1033)`

Returning ATOM XML vs. JSON

Response data format selected with ACCEPT header

- XML can be easier to deal with from managed code
- To get ATOM XML response use "application/atom+xml"
- JSON is easier to deal with when using JavaScript
- To get JSON response use "application/json;odata=verbose"



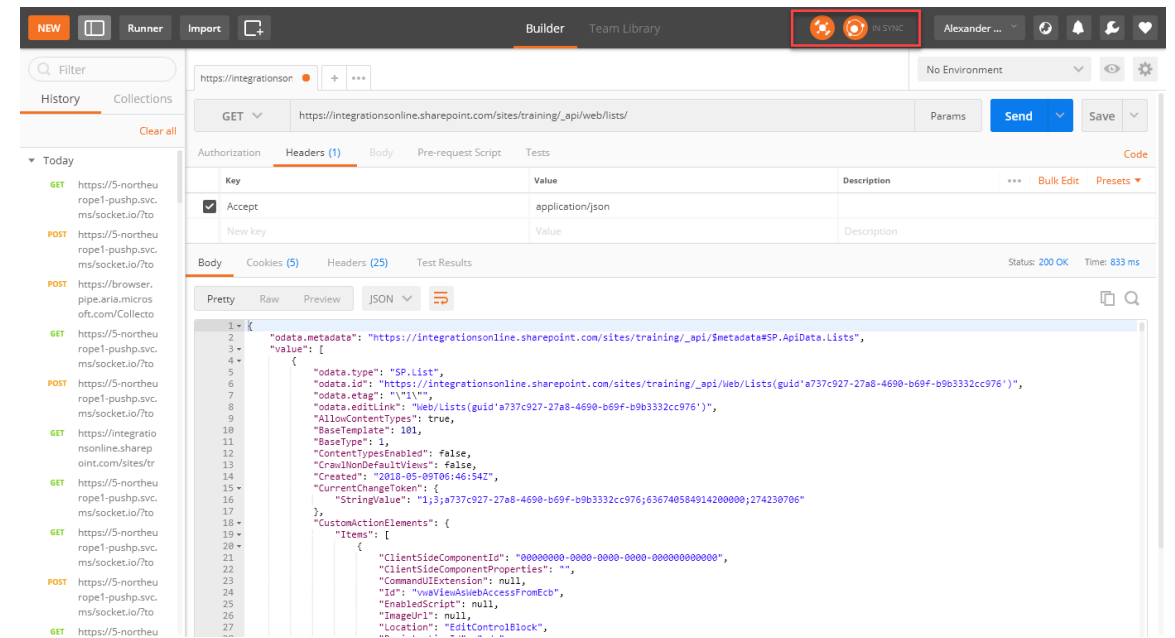
```
Title of the web ▾ {d: {...}} ⓘ  
  ▾ d:  
    Title: "Integrations e. U."  
    ▶ __proto__: Object  
    ▶ __proto__: Object
```

Postman / Postman Interceptor

Postman – Test Client for REST services

Postman Interceptor – Passes Auth from Chrome to Postman

- Works only in Chrome App
- NOT Stand Alone



oData Verbs

\$metadata: Self describing metadata about the list

\$select: Allows to select certain fields only

\$filter=: Returns only entities that match the specified expression.

\$orderby=: Orders results, in ascending or descending order.

oData Verbs

\$top=n: Returns only the first n entities in an entity set

\$skip=n: Skips the first n entities in an entity set.

\$format: Determines whether data should be returned in JSON or the XML-based Atom/AtomPub format.

\$expand: Expands associated entities

Full syntax - <https://msdn.microsoft.com/en-us/library/office/fp142385.aspx>

\$expand

Objects that represent complex types must be expanded in order to access the value

- User fields (Author, Editor, ...)
- Lookup
- Geolocation
- Link

```
/_api/lists/getByTitle('Assets')/Items?$select=LinkTitle,Owner/Name&$expand=Owner
```

```
1 {
2   "d": {
3     "results": [
4       {
5         "_metadata": {
6           "id": "Web/Lists(guid'3812a010-bced-4b5a-b4a9-123ac542f58f')/Items(1)",
7           "uri": "http://sp2013c/_api/Web/Lists(guid'3812a010-bced-4b5a-b4a9-123ac542f58f')/Items(1)",
8           "etag": "\"1\"",
9           "type": "SP.Data.AssetsListItem"
10        },
11        "Owner": {
12          "_metadata": {
13            "id": "90c357f0-2700-4860-88ab-98eadc6da573",
14            "type": "SP.Data.UserInfoItem"
15          },
16          "Name": "i:0#.w|spdom\\administrator"
17        },
18        "LinkTitle": "Chain"
19      },
20    ]
21  }
```

\$expand & user profile

When extending a user field several user profile properties can be retrieved, but not all. The following user profile properties will be returned:

Name

Title

Department

JobTitle

Columns	
A column stores information about each item in the list. The following columns	
Column (click to edit)	Type
Title	Single line of text
Owner	Person or Group
Modified	Date and Time
Created	Date and Time
Created By	Person or Group
Modified By	Person or Group

Managing Lists

Request Digest

- When POST-ing the request, you get a HTTP 403 Forbidden back in response from the Search service with an error message stating *“The security validation for this page is invalid and might be corrupted. Please use your web browser’s Back button to try your operation again.”*
- The reason for getting the 403 Forbidden status is mainly because you are missing an important header in your request which is the X-RequestDigest header.

Creating Lists with JavaScript and jQuery

Create appropriate JavaScript object and convert to JSON format for request body

Create request headers for HTTP Request

Send request to Web server using jQuery \$.ajax function

```
function onCreateList() {  
    var newList = {  
        "__metadata": { "type": "SP.List" },  
        "BaseTemplate": 104,  
        "Description": "A list created by REST",  
        "Title": $("#newListTitle").val()  
    }  
  
    var requestHeaders = {  
        Accept: "application/json;odata=verbose",  
        "X-RequestDigest": $("#__REQUESTDIGEST").val()  
    }  
  
    $.ajax({  
        type: "POST",  
        url: _spPageContextInfo.webAbsoluteUrl + "/_api/web/lists",  
        contentType: "application/json;odata=verbose",  
        data: JSON.stringify(newList),  
        headers: requestHeaders,  
        success: onDataReturned,  
        error: onError  
    });  
}
```

Get form digest value from control on page

Convert request body data to string-based JSON object

Create List

```
var body = JSON.stringify({
  '__metadata': { 'type': 'SP.List' },
  'AllowContentTypes': true,
  'BaseTemplate': 100,
  'ContentTypesEnabled': true,
  'Description': 'My REST List',
  'Title': 'MyRestList'
});
```

```
var digest = $("#__REQUESTDIGEST").val();
var url = _spPageContextInfo.webAbsoluteUrl +
  "/_api/web/lists";

jQuery.ajax({
  url: url,
  type: "POST",
  data: body,
  headers: {
    "accept": "application/json;odata=verbose",
    "content-type": "application/json;odata=verbose",
    "X-RequestDigest": digest
  },
  success: function () { console.log("list created") },
  error: function (msg) { console.log(msg) }
});
```

Update List

"X-RequestDigest"

"X-HTTP-Method":"MERGE",

"IF-MATCH": "*"

```
jQuery.ajax({  
  url: _spPageContextInfo.webAbsoluteUrl + "/_api/web/lists/GetByTitle('MyRestList')",  
  type: "POST",  
  data: JSON.stringify({ '__metadata': { 'type': 'SP.List' }, 'Title': 'My Rest List' }),  
  headers: {  
    "X-HTTP-Method": "MERGE",  
    "accept": "application/json;odata=verbose",  
    "content-type": "application/json;odata=verbose",  
    "X-RequestDigest": $("#__REQUESTDIGEST").val(),  
    "IF-MATCH": "*"   
  },  
  success: function () { console.log("list renamed") },  
  error: function (msg) { console.log(msg) }  
});
```

Delete List

"X-RequestDigest"

"X-HTTP-Method":"DELETE",

"IF-MATCH": "*"

```
$.ajax({  
  url: _spPageContextInfo.webAbsoluteUrl + "/_api/web/Lists/getbytitle('My Rest List')",  
  method: "POST",  
  headers: {  
    "accept": "application/json;odata=verbose",  
    "content-type": "application/json;odata=verbose",  
    "X-RequestDigest": $("#__REQUESTDIGEST").val(),  
    "IF-MATCH": "*",  
    "X-HTTP-Method": "DELETE"  
  },  
  success: function () { console.log("list deleted") },  
  error: function (msg) { console.log(msg) }  
});
```

REST CRUD

REST Query Using JavaScript & jQuery

```
$(onPageLoad);

function onPageLoad() {
    $("#cmdGetSiteInfo").click(onGetSiteInfo);
}

function onGetSiteInfo(){
    var requestUri = _spPageContextInfo.webAbsoluteUrl + "/_api/web/?$select=Title";

    $.ajax({
        type: "GET",
        url: requestUri,
        contentType: "application/json",
        headers : {
            Accept : "application/json;odata="
        },
        success: onDataReturned,
        error: onError
    });
}

function onDataReturned(data){
    var odataResults = data.d;
    var site_title = odataResults.Title;
    $("#results").html("Title: " + site_title);
}

function onError(error) {
    $("#results").text("ERROR : " + JSON.stringify(error));
}
```



```
{
  "d": {
    "__metadata": {
      "id": "http://contoso-fb5f1425f041fd.contosoapps.com/sites/SpAppsCloudHosted/SharePointApp3/_api/Web",
      "uri": "http://contoso-fb5f1425f041fd.contosoapps.com/sites/SpAppsCloudHosted/SharePointApp3/_api/Web",
      "type": "SP.Web"
    },
    "Title": "SharePointApp3"
  }
}
```

Read List

No Request Digest required because of HTTP GET

```
$.ajax({  
  url: _spPageContextInfo.webAbsoluteUrl + "/_api/lists/getByTitle('News')/Items?$select=Title",  
  type: "GET",  
  headers: {  
    "Accept": "application/json;odata=verbose",  
    "Content-Type": "application/json; odata=verbose"  
  },  
  success: function (data) {  
    if (data.d.results) {  
      // TODO: handle the data  
      console.log('handle the data');  
    }  
  },  
  error: function (msg) { console.log(msg) }  
});
```


Read Item

```
$.ajax({
  url: _spPageContextInfo.webAbsoluteUrl +
  "/_api/web/lists/getbytitle('MyRestList')/GetItems(query=@v1)?@v1={\"ViewXml\": \"<View>
  <Query><OrderBy><FieldRef Name='Created' /></OrderBy></Query></View>\"}",
  type: "POST",
  headers: {
    "X-RequestDigest": $("#__REQUESTDIGEST").val(),
    "Accept": "application/json;odata=verbose",
    "Content-Type": "application/json; odata=verbose"
  },
  success: function (data) {
    if (data.d.results) {
      // TODO: handle the data
      console.log('handle the data');
    }
  },
  error: function (msg) { console.log(msg) }
});
```

Http Headers

„Writing“ operations require special HTTP Headers

Create

```
"X-RequestDigest": $("#__REQUESTDIGEST").val()
```

Update

```
"X-RequestDigest": $("#__REQUESTDIGEST").val()  
"If-Match": "*",  
"X-HTTP-Method": "MERGE"
```

Delete

```
"X-RequestDigest": $("#__REQUESTDIGEST").val()  
"IF-MATCH": "*",  
"X-HTTP-Method": "DELETE"
```

Create Item

```
var item = {  
  "__metadata": { "type": getListItemType("MyRestList") },  
  "Title": "CodedItem",  
  "My_x0020_Custom_x0020_Field": "My Custom Value"  
};
```

```
$.ajax({  
  url: _spPageContextInfo.webAbsoluteUrl + "/_api/web/lists/getbytitle('MyRestList')/items",  
  type: "POST",  
  contentType: "application/json;odata=verbose",  
  data: JSON.stringify(item),  
  headers: {  
    "Accept": "application/json;odata=verbose",  
    "X-RequestDigest": $("#__REQUESTDIGEST").val()  
  },  
  success: function () { console.log("item added to list") },  
  error: function (msg) { console.log(msg) }  
});
```

Update Item

```
function updateListItem() {  
    var itemProperties = { 'Title': 'The REST News' };  
    updateItem('News', 1, itemProperties, function () {  
        console.log('item has been updated');  
    }, function() {  
        console.log('error');  
    });  
}
```

```
function updateItem(listTitle, listItemId, itemProperties, success, failure) {  
    var listItemUri = _spPageContextInfo.webAbsoluteUrl + "/_api/web/lists/getbytitle('" + listTitle + "')/items(" + listItemId  
        + ")";  
    var itemPayload = {  
        '__metadata': { 'type': "SP.Data." + listTitle.charAt(0).toUpperCase() + listTitle.slice(1) + "ListItem" }  
    };  
    for (var prop in itemProperties) {  
        itemPayload[prop] = itemProperties[prop];  
    }  
    updateJson(listItemUri, itemPayload, success, failure);  
}
```

Update Item - Cont

```
function updateJson(endpointUri, payload, success, error) {  
    $.ajax({  
        url: endpointUri,  
        type: "POST",  
        data: JSON.stringify(payload),  
        contentType: "application/json;odata=verbose",  
        headers: {  
            "Accept": "application/json;odata=verbose",  
            "X-RequestDigest": $("#__REQUESTDIGEST").val(),  
            "X-HTTP-Method": "MERGE",  
            "If-Match": "*"   
        },  
        success: success,  
        error: error  
    });  
}
```

Delete Item

```
var id = 1;
var url = _spPageContextInfo.webAbsoluteUrl + "/_api/web/lists/GetByTitle('News')/items(" + id + ")";

$.ajax({
  url: url,
  type: "POST",
  headers: {
    "ACCEPT": "application/json;odata=verbose",
    "content-type": "application/json;odata=verbose",
    "X-RequestDigest": $("#__REQUESTDIGEST").val(),
    "IF-MATCH": "*",
    "X-HTTP-Method": "DELETE"
  },
  success: function (data) {
    console.log("Deleted Successfully.");
  },
  error: function (error) {
    console.log(JSON.stringify(error));
  }
});
```

CSOM & JSOM

What is CSOM / JSOM

JSOM is an unofficial but commonly used term for the JavaScript implementation of the SharePoint Client Object Model

Stands for JavaScript Object Model

Implemented in sp.js

SP namespace documented @ [https://docs.microsoft.com/en-us/previous-versions/office/sharepoint-visio/jj193034\(v=office.15\)](https://docs.microsoft.com/en-us/previous-versions/office/sharepoint-visio/jj193034(v=office.15))

Many additional files for

- Base SharePoint
- Service Applications

Out-Of-Box References

SP.js – Client side object model

Core.js - Include dropdown menu items, page layout manipulation, expand/collapse behavior on list views, etc.

Menu.js – Core menu object

Callout.js - Callouts

Sharing.js – Sharing & Permissions

Init.js – contains helper objects like SP.ScriptUtility

Deployment Locations

Several options for deployments of JavaScript based solutions

On Premise

- Site Assets or any other DocumentLibrary
- /_layouts/15/1033/Subfolder
- /_layouts/15/ApplicationPageFolder

Office 365

- Site Assets or any other DocumentLibrary

ctx & g_listData in Global Namespace

Available in List-Related Pages

- g_listData -> Modern UI

ListDataListViewonly

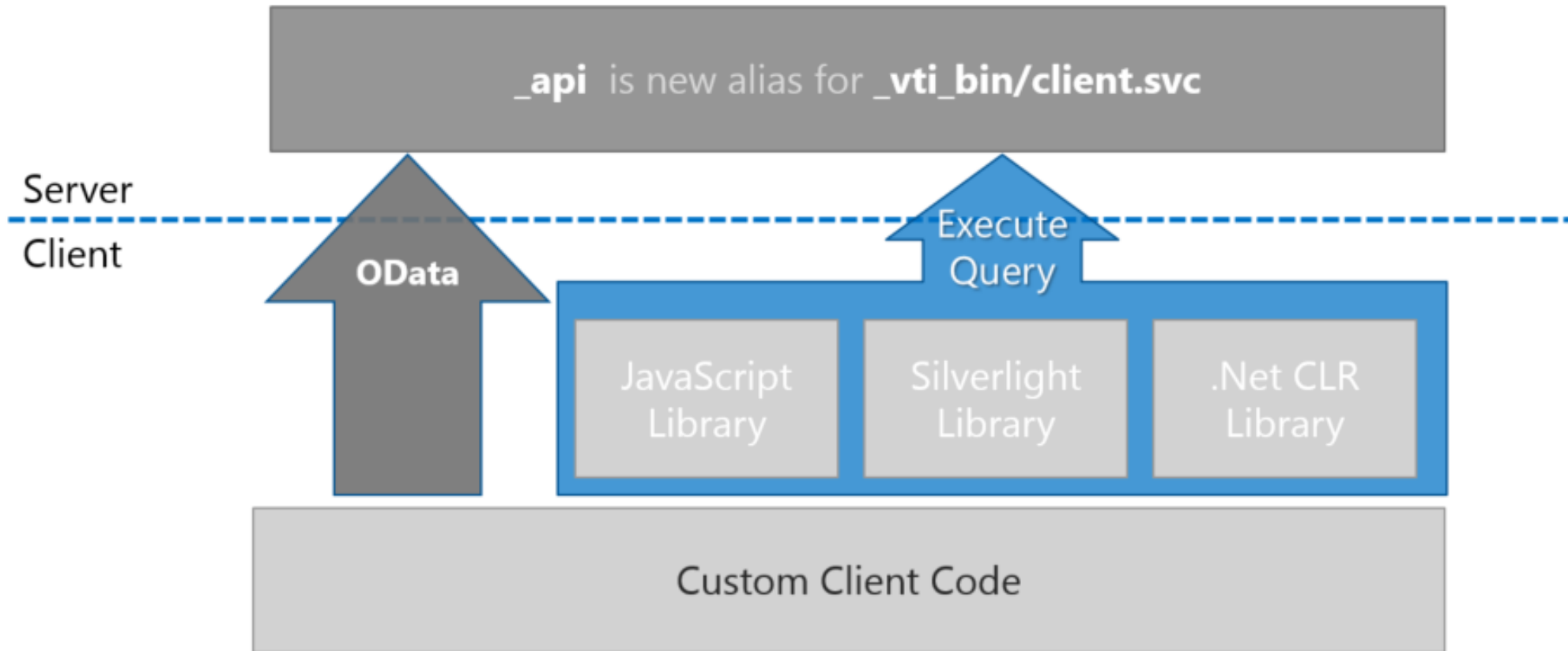
- Row collection property: Column values
- Row is 0-based, based on display, not item ID
- ContentTypeId
- FSObjType(0=ListItem, 1=Folder)

```
▼ Watch
▼ ctx: ContextInfo
  AllowCreateFolder: true
  AllowGridMode: true
  ▶ BasePermissions: Object
  BaseViewID: 1
  CascadeDeleteWarningMessage: null
  ContentTypesEnabled: false
  ControlMode: 4
  CurrentCultureName: "en-US"
  CurrentLanguage: 1033
  CurrentSelectedItems: null
  CurrentUICultureName: "en-US"
  CurrentUserId: 1
  CurrentUserIsSiteAdmin: true
  EnableMinorVersions: false
  ExternalDataList: false
  HasRelatedCascadeLists: 0
  HttpPath: "http://sp2013b/vti bin/owssvr.dll?CS=65001"
  HttpRoot: "http://sp2013b"
  IsAppWeb: false
  IsClientRendering: true
  LastSelectableRowIndex: null
  ▶ ListData: Object
  ListDataJSONItemsKey: "Row"
  ▶ ListSchema: Object
  ListTemplateType: 101
  ListTitle: "Documents"
  ModerationStatus: 0
  NavigateForFormsPages: true
  OfficialFileName: ""
```

Registering & Loading JavaScript

Context, Batching, Loading

SharePoint Remote API



ClientContext

Represents the context for objects and operations

Reference the required libraries

Avoid ctx as a variable

Limited to current site collection without CDL

```
<script
  type="text/javascript"
  src="//ajax.aspnetcdn.com/ajax/4.0/1/MicrosoftAjax.js">
</script>
<script type="text/javascript" src="_layouts/15/sp.runtime.js"></script>
<script type="text/javascript" src="_layouts/15/sp.js"></script>
<script type="text/javascript">
  // Continue your program flow here.
</script>
```

SP.ClientContext object

Initializes a new instance of the ClientContext object for the specified SharePoint site

Avoid calling it ctx because of ctx in ListViews

Overloaded constructor

Cannot span multiple Site Collections

Copy between two ClientContext objects possible

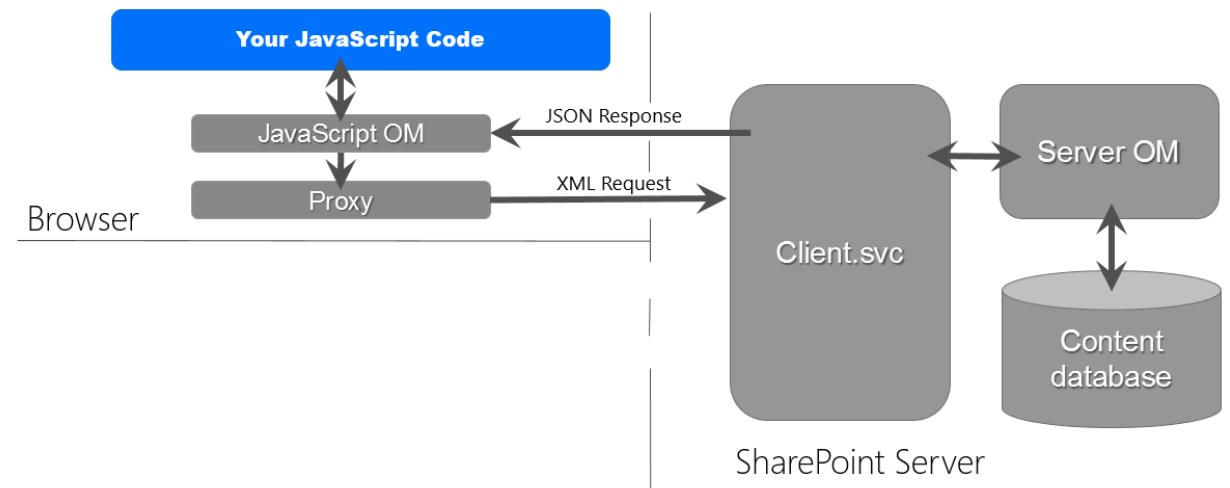
```
var cctx;  
cctx = new SP.ClientContext(); //Connect to the default context  
cctx = new SP.ClientContext("http://sp2013c/8085"); //Connect to a specific Url  
cctx = new SP.ClientContext("/sales"); //Connect to a subweb
```


Batching

Commands are sent to the server in batches

```
var cctx = new SP.ClientContext();
this.web = cctx.get_web();
cctx.load(web, 'Title', 'Created');
cctx.load(web.get_lists());
cctx.executeQueryAsync(function () {
    console.log("Successfully loaded" + web.get_title());
}, onErr);
```

Programming using CSOM



Loading

Objects and their properties must be loaded explicitly

Three Options:

- Load All: `ctxCurrent.load(web)`
- Load Explicit: `ctxCurrent.load(owner, 'CreatedBy', 'Title')`
- Load Collection: `ctxCurrent.load(<objectCollection>, 'Include(<Field1>, <Field2>, <Field3>'));`

Exception Handling -> Server Side Transaction

Exception handling is done using SP.ExceptionHandlingScope

```
function sampleExceptionHandler() {  
    var currCtx = new SP.ClientContext();  
    var scope=new SP.ExceptionHandlingScope(currCtx);  
    var startScope=scope.startScope();  
    var tb=scope.startTry();  
    //attempt something that causes an error  
    var lists=currCtx.get_web().get_lists();  
    var badList=lists.getByTitle("badListName");  
    tb.dispose();  
    var cb=scope.startCatch();  
    /*server-side steps to fix the error  
    cb.dispose();  
    var fb=scope.startFinally();  
    /**server-side actions that will always occur  
    fb.dispose();  
    startScope.dispose();  
    currCtx.executeQueryAsync(onSucceed, onFail);  
}
```

Site & Site Collections

Object Hierarchy

- SDK Reference available at:
- <https://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.aspx>
- <https://msdn.microsoft.com/EN-US/library/office/jj246996.aspx>

SPFarm

SPWebApplication

SPSite

SPWeb

SPList

SPFolder

SPListItem

SP.Site object

SP.Site does not allow creating of Site Collections

SpoOperation class - Represents an operation on a site collection in an Office 365 tenant

Site Collection operations is not supported by JSOM in an on premise installation

As a fallback you can use Admin.aspx

SP.Web object

Represents a Microsoft SharePoint Foundation Web site.

Main entry point for client side access

Has most of the properties and methods of the server side equivalent

```
var clientContext = new SP.ClientContext();  
this.web = clientContext.get_web();  
clientContext.load(web, 'Title', 'Created');
```

Create a Web

SP.WebCreationInformation() holds the parameter to create a Web

```
var clientContext = new SP.ClientContext.get_current();
var web = clientContext.get_web();
var webCreationInfo = new SP.WebCreationInformation();
webCreationInfo.set_title('My JSOM Web Site');
webCreationInfo.set_description('Description of new Web site...');
webCreationInfo.set_language(1033);
webCreationInfo.set_url('MyJSOMWebSite');
webCreationInfo.set_useSamePermissionsAsParentSite(true);
webCreationInfo.set_webTemplate('STS#0');
web.get_webs().add(webCreationInfo);
web.update();

clientContext.executeQueryAsync(function () { console.log("JSOM Web created"); }, onQueryFailed);
```


Update / Delete Web

Update

```
var clientContext = new SP.ClientContext.get_current();
var web = clientContext.get_web();
web.set_title('Updated Web Site');
web.set_description('This is an updated Web site. ');
web.update();
clientContext.load(web, 'Title', 'Description');
clientContext.executeQueryAsync(function () {
    console.log('Title: ' + web.get_title() + ' Description: ' + web.get_description());
}, onQueryFailed);
```

Delete

```
var web = site.openWeb("/MyJSOMWebSite");
web.deleteObject();
clientContext.load(site);
clientContext.load(web);
clientContext.executeQueryAsync(function () {
```

Use Props Bag

A dictionary of key – value to store custom values for your app

Write to property bag

```
var clientContext = new SP.ClientContext.get_current();
var web = clientContext.get_web();
this.properties = web.get_allProperties();
this.properties.set_item("myCustomProperty", "myCustomValue");
clientContext.load(web);
web.update();
clientContext.executeQueryAsync(Function.createDelegate(this, getWebProperty), onQueryFailed);
```

Get value from property bag

```
var val = web.properties.get_item("myCustomProperty");
```

Manage Lists

List Basics

Lists can be accessed using an instance of the List object

Items in a list are represented by ListItem object

To assign a value to a column use

- `ListItem[„Fieldname“] = Value`
- `ListItem.Update()`

Create List

Create list

```
var clientContext = new SP.ClientContext(siteUrl);
var web = clientContext.get_web();
var listCreationInfo = new SP.ListCreationInformation();
listCreationInfo.set_title(listName);
listCreationInfo.set_templateType(SP.ListTemplateType.announcements);
var list = web.get_lists().add(listCreationInfo);
clientContext.load(list);
clientContext.executeQueryAsync(function () { console.log("Create list done"); }, onQueryFailed);
```

Find list using a specific template

```
listItemEnumerator = lists.getEnumerator();
var discussionBoardLists = [];
while (listItemEnumerator.moveNext()) {
    oListItem = listItemEnumerator.get_current();
    if (oListItem.get_baseTemplate() == 108) {
        var listname = oListItem.get_title();
        discussionBoardLists.push(listname);
    }
}
```

Update / Delete List

Update

```
var list = clientContext.get_web().get_lists().getByTitle("News");  
list.set_description("The very cool Announcments list");  
list.update();  
clientContext.load(list);  
clientContext.executeQueryAsync(function () { console.log("Update list done"); }, onQueryFailed);
```

Delete

```
var list = web.get_lists().getByTitle("News");  
list.deleteObject();  
clientContext.executeQueryAsync(function () { console.log("Delete list done"); }, onQueryFailed);
```

Manipulate List

Add field to list

```
var list = clientContext.get_web().get_lists().getByTitle(listName);
this.fld = list.get_fields().addFieldAsXml('<Field DisplayName=\'MyField\' Type=\'Number\' />', true,
SP.AddFieldOptions.defaultValue);
var fieldNumber = clientContext.castTo(fld, SP.FieldNumber);
fieldNumber.set_maximumValue(100);
fieldNumber.set_minimumValue(35);
fieldNumber.update();
clientContext.load(fld);
clientContext.executeQueryAsync(function () { console.log("Add field to list done"); }, onQueryFailed);
```

Set choice values

```
fields = web.get_fields();
var fieldExpiryDate = context.castTo(fields.getInternalNameOrTitle("ProductionType"),
SP.FieldChoice);
var choices = Array("Phase 1 Trial", "Phase 2 Trial", "Phase 3 Trial", "Production");
fieldExpiryDate.set_choices(choices);
context.ExecuteQueryAsync(onUpdateFieldSuccess, onUpdateFieldFail);
```

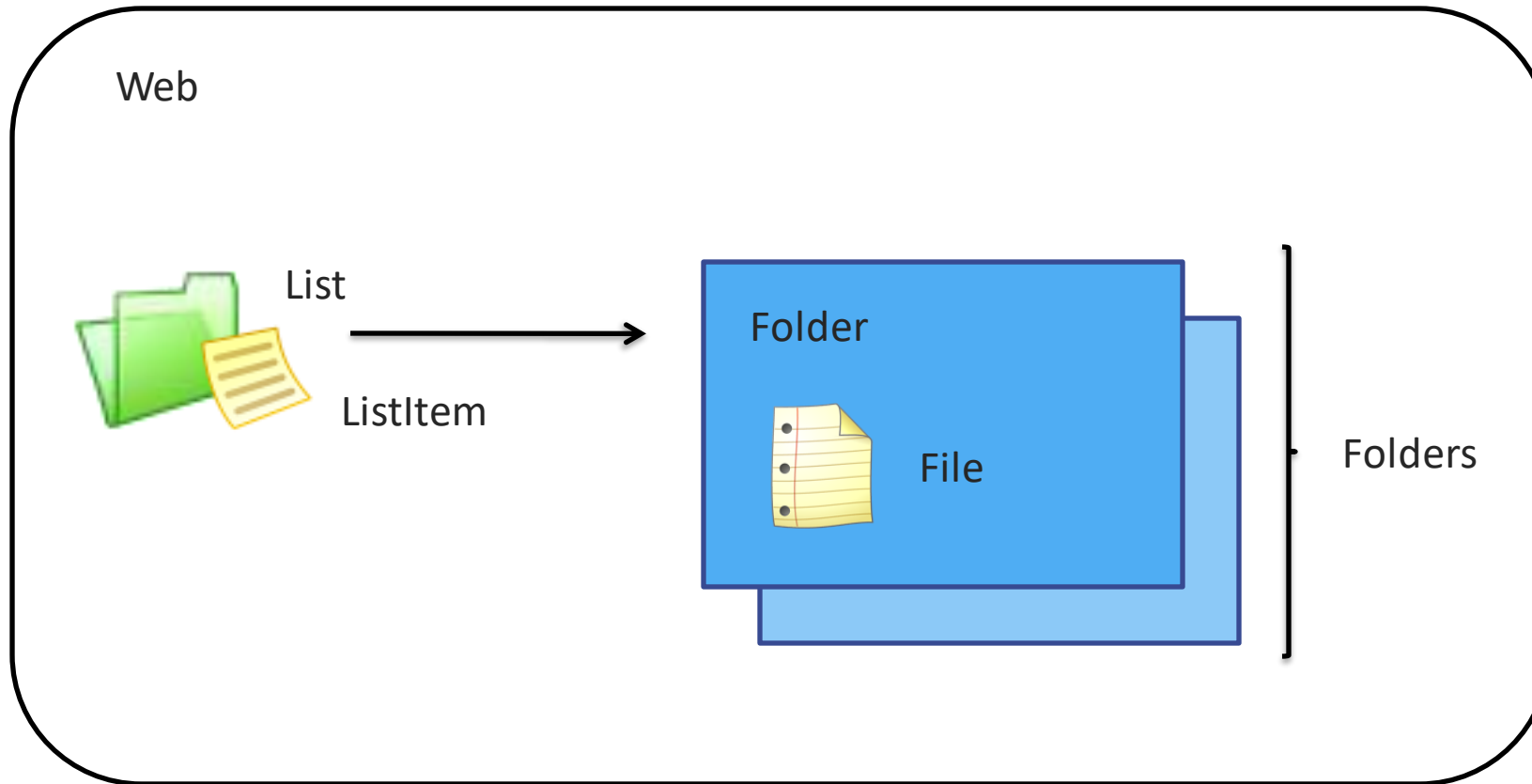
Document Library Basics

Document Libraries are instances of the SPList Class

Items in a Document Library are instances of the SPListItem Class

Each Library has an associated SPFolder in which the files (SPFile) of the list are stored

Document Storage in SharePoint



Working with files

Create

```
list = web.get_lists().getByTitle("Shared Documents");
fci = new SP.FileCreationInformation();
fci.set_url("my new file.txt");
fci.set_content(new SP.Base64EncodedByteArray());
fileContent = "The content of my new file";
for (var i = 0; i < fileContent.length; i++) {

    fci.get_content().append(fileContent.charCodeAt(i));
}
var newFile = list.get_rootFolder().get_files().add(fci);
clientContext.load(this.newFile);
```

ListItem CRUD

Simple Read

List item is represented by SP.ListItem

"Title" is represented displayName

Access to properties is done using "get_PROPERTY()"

```
var clientContext = new SP.ClientContext();
var list = clientContext.get_web().get_lists().getByTitle('News');
var li = list.getItemById(itemId);
clientContext.executeQueryAsync(function() {
    console.log(li.get_displayName());
});
```

CAML Query

Used to search for one or more item

<ViewFields> corresponds to "Select xx from"

<Where> corresponds to condition

```
var camlQuery = new SP.CamlQuery();
camlQuery.set_viewXml('<Query><Where><Eq><FieldRef Name="ID" /><Value Type="Counter">'
    + itemId + '1</Value></Eq></Where></Query><View><RowLimit>100</RowLimit></View>');
var collListItem = list.getItems(camlQuery);
clientContext.load(collListItem, 'Include(Id, DisplayName, HasUniqueRoleAssignments)');
clientContext.executeQueryAsync(
    function() {
        var liInfo = '';
        var listItemEnumerator = collListItem.GetEnumerator();
        while (listItemEnumerator.MoveNext()) {
            var li = listItemEnumerator.get_current();
            ...
        }
    }
```

Simple Create

XXCreationInformation classes are used to set the param sent to server

```
var clientContext = new SP.ClientContext();
var list = clientContext.get_web().get_lists().getByTitle('News');
var itemCreateInfo = new SP.ListItemCreationInformation();
this.li = list.addItem(itemCreateInfo);
li.set_item('Title', 'My New Item!');
li.set_item('Body', 'Hello World!');
li.update();

clientContext.load(li);
clientContext.executeQueryAsync(function () {
    itemId = li.get_id();
    alert('Item created: ' + itemId);
},function(){..})
```

Complex Field Values

Complex Field Values have to be set using objects

- FieldGeoLocationValue
- FieldLookupValue
- FieldUrlValue
- FieldUserValue

```
var urlValue = new SP.FieldUrlValue();  
urlValue.set_url("http://www.example.com");  
urlValue.set_description("test link");  
myItem.set_item("TestURL", urlValue);
```

Lookups

Call fld.get_lookupValue() to get the Value

Use SP.FieldLookupValue() and pass unique ID of the lookup item to write

```
var ctx = SP.ClientContext.get_current();
var web = ctx.get_web();
var lists = web.get_lists();
var listNews = lists.getByTitle("News");
var firstNews = listNews.getItemById(1);
ctx.load(firstNews);
ctx.executeQueryAsync(function() {
    var lookupField = firstNews.get_item("Writer");
    var lookupTitle = lookupField.get_lookupValue();
    console.log("title of lookedup field is " + lookupTitle);
}, onQueryFailed);
```


Managed Metadata

Implemented in sp.taxonomy.js

Documentation @ <https://msdn.microsoft.com/en-us/library/office/jj857114.aspx>

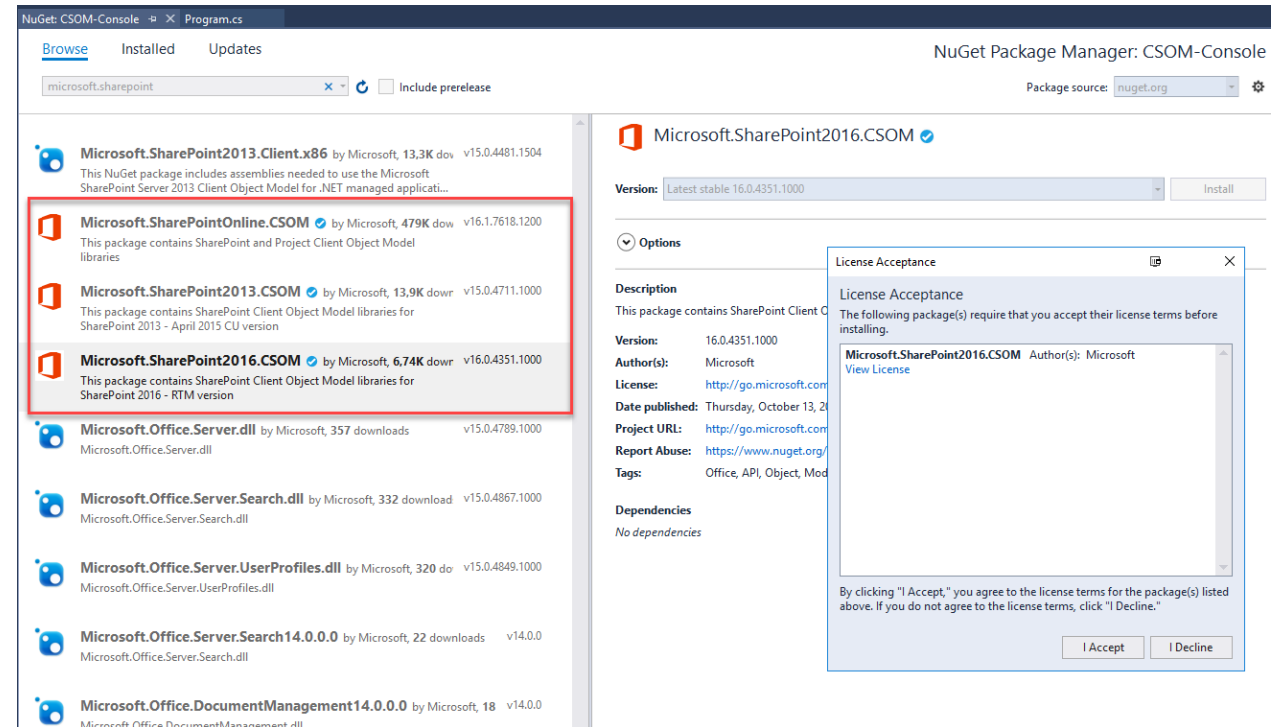
```
var ctx = SP.ClientContext.get_current();
var web = ctx.get_web();
var lists = web.get_lists();
var listNews = lists.getByTitle("News");
var firstNews = listNews.getItemById(1);
ctx.load(firstNews);
ctx.executeQueryAsync(function () {
    var mmField = firstNews.get_item("Topic");
    console.log("title of Managed Metadata field is " + mmField.Label);
}, onQueryFailed);
```

Install NuGet Package

Choose Project Type of your Choice – i.e. Console Application

Available for:

- SP 2013
- SP 2016
- SP Online

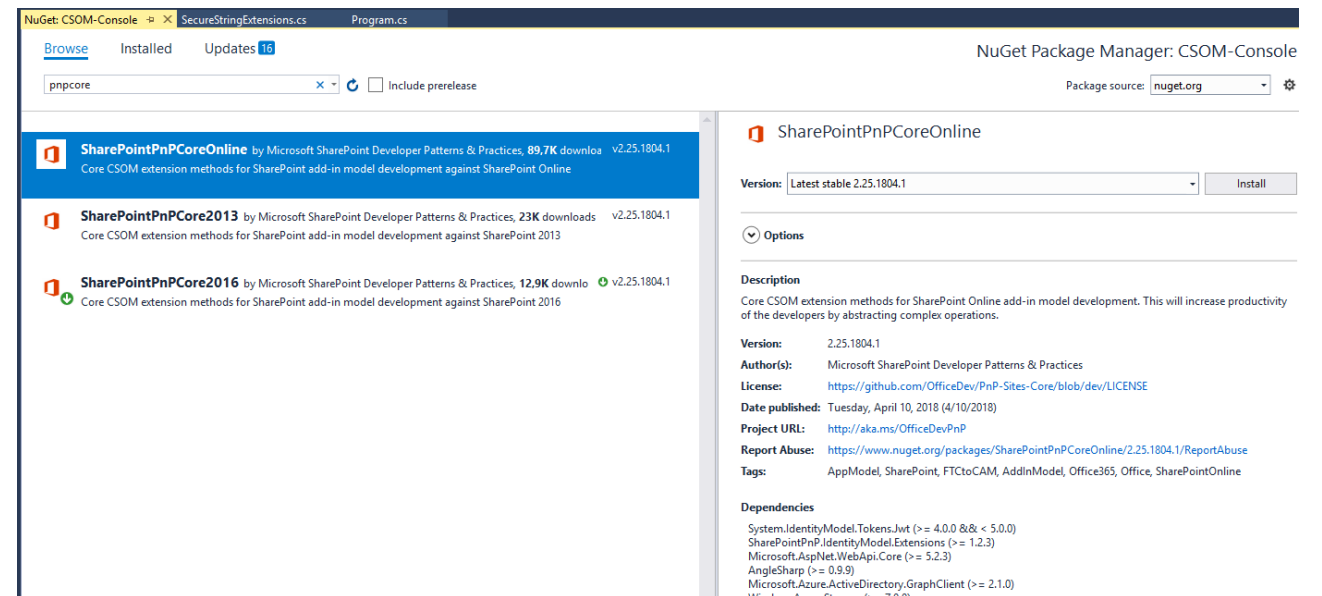


PnP Sites Core

A set of Extensions to traditional CSOM

Consists of:

- App Model Extensions / General Extensions
- Enums
- Utilities
- Authentication Manager



What is PnP JavaScript

PnP/PnPjs

JavaScript API for consuming SharePoint and Office 365 REST APIs in a type-safe way

Node.js & TypeScript based

- Most Samples are SPFx Samples

Provide a “helper” library similar to PnP CSOM

- Lightweight
- Minimal dependencies
- Easy to use
- Supports Promises
- Documented @ <https://github.com/pnp/pnpjs>

PnP/PnPjs Packages

@pnp/common - Provides shared functionality across all pnp libraries

@pnp/config-store - Provides a way to manage configuration within your application

@pnp/graph- Provides functionality to query the Microsoft Graph

@pnp/logging - Light-weight, subscribable logging framework

@pnp/nodejs - Provides functionality enabling the @pnp libraries within nodejs

@pnp/odata - Provides shared odata functionality and base classes

@pnp/pnpjs - Rollup library of core functionality, mimics sp-pnp-js

@pnp/sp - Provides a fluent api for working with SharePoint REST

@pnp/sp-addinhelpers - Provides functionality for working within SharePoint add-ins

Current State

- Robust REST based library providing a fluent API
- Monthly releases driven by community feedback
- Constantly improving docs and guides
- Sample Add-In, SPFx samples, script editor wp
- Used in 419 tenants to make 8.6 million requests*

History

- Started in early 2017 as sp-pnp-js
- Renamed to PnP/PnPjs
 - Moved to a new repository
 - Broken down into several libs:
 - @pnp/common
 - @pnp/pnpjs
 - @pnp/sp
 - ...

Transition Guide @ <https://pnp.github.io/pnpjs/transition-guide.html>

Benefits

- Handles request digest and header settings
- Ensures the correct JSON format and type values
- Easily batch requests inline
- Fluent API guides building your requests
- Easy install with *npm install sp-pnp-js --save*
- Works with classic SharePoint, SPFx, and Add-Ins
- Extensible to support your business scenarios

Installing PnP/PnPjs

Technically installation is based on Node.js & npm

- `npm install @pnp/logging @pnp/common @pnp/odata @pnp/sp @pnp/graph --save`

Can be used in:

- SharePoint Framework
- Node.js
- Classical Solutions & Add-Ins

Needs polyfills to support older browsers

- es6-promises - <https://github.com/stefanpenner/es6-promise>
- fetch - <https://github.com/github/fetch>

Using PnP/PnPjs

Samples

PnP/PnPjs Samples

- SPFx Module (later)

Microsoft Graph

What is Graph

An alternate gateway to Microsoft 365, compared to direct API Endpoints

Single resource that proxies multiple Microsoft services

Allows for easy traversal of objects and relationships

Simplifies token acquisition and management

Eliminates the need to traditional discovery (using “me” and “myorganization”)

Documentation @ <https://graph.microsoft.io/en-us/docs/overview/overview>

Single API

Single API for:

Accessing data

- /me, /users, /groups, /messages, /drive,

Traversing data

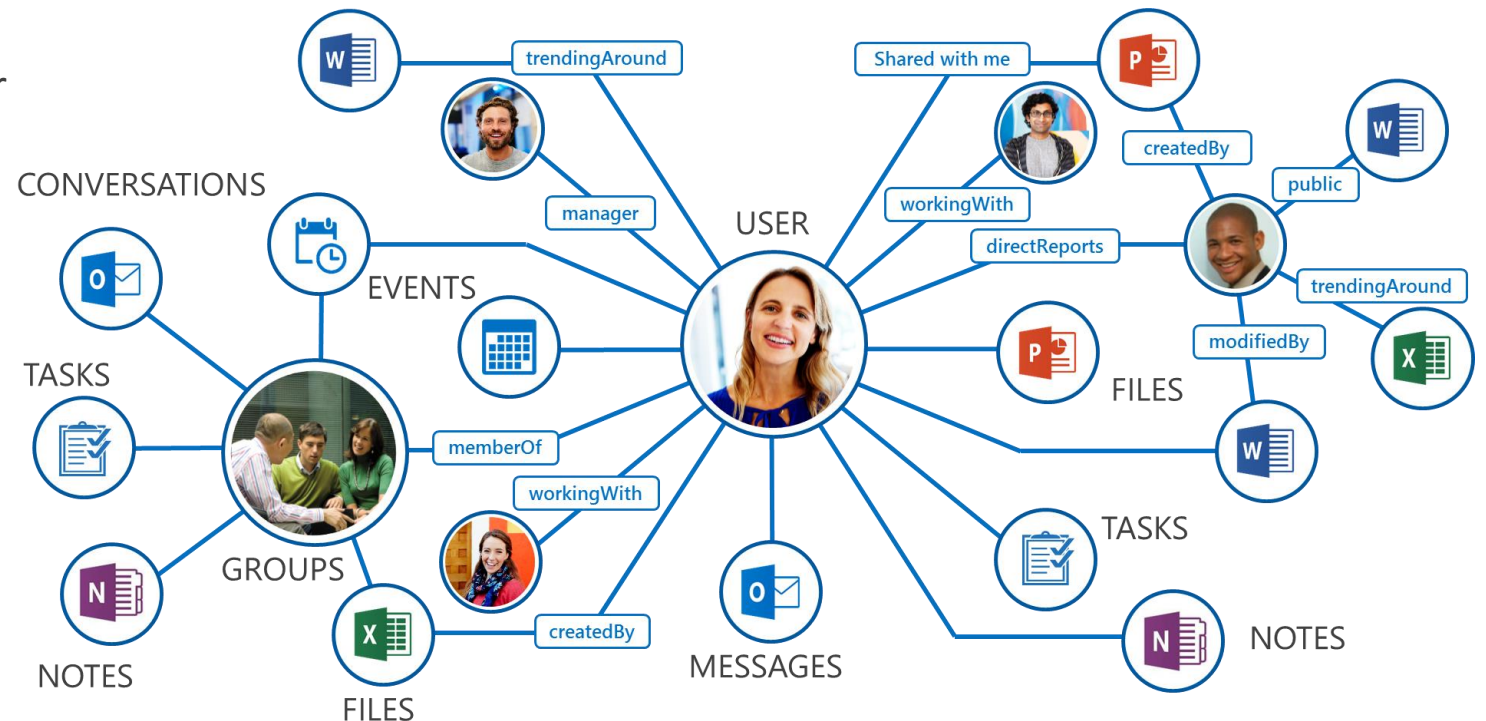
- `/drive/items/<id>/lastmodifiedByUser`

Accessing insights

- `/insights/trending`

Work/School and Personal

<https://graph.microsoft.com/>



Navigating the Microsoft Graph Resources

<https://graph.microsoft.com>

1. Get a tenant-level **entity set**:
 /users
2. Select a member from the entity set:
 /users/{id}
3. Get an entity **property**:
 /users/{id}/department
4. Traverse to related **entity type** via **navigation properties**:
 /users/{id}/files

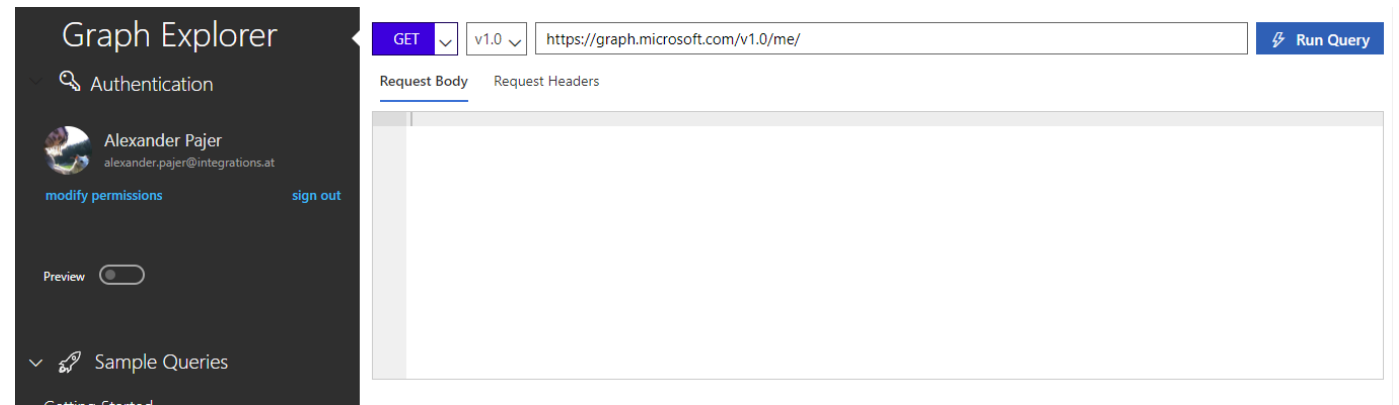
/{version}	/{tenant}	/{entity-set}	/{id}	/{property}
------------	-----------	---------------	-------	-------------

Microsoft Graph Explorer

Can be used to test your Graph Queries

Postman with manual authentication also possible

Graph Typings available @: <https://github.com/microsoftgraph/msgraph-typescript-typings>



Authentication Options

Azure AD only

- Separate auth flow supports Azure AD accounts only

Azure AD and Microsoft Accounts (Preview)

- Converged auth flow supports Azure AD accounts and Microsoft accounts (LiveID - hotmail.com, etc.)

One Drive API

One Drive API 2.0

OneDrive uses OAuth 2.0 for authentication

Documentation @ <https://dev.onedrive.com/>

Available for

- OneDrive,
- OneDrive for Business,
- SharePoint, and
- SharePoint Server 2016

Service	URL Root
OneDrive	<code>https://api.onedrive.com/v1.0</code>
OneDrive for Business	<code>https://{tenant}-my.sharepoint.com/_api/v2.0</code>
SharePoint Online	<code>https://{tenant}.sharepoint.com/{site-relative-path}/_api/v2.0</code>

JavaScript File Picker


The OneDrive file picker SDK enables your web app to quickly integrate OneDrive for opening and saving files without a lot of code

Need to register your application and receive an app ID from the Microsoft Application Registration Portal

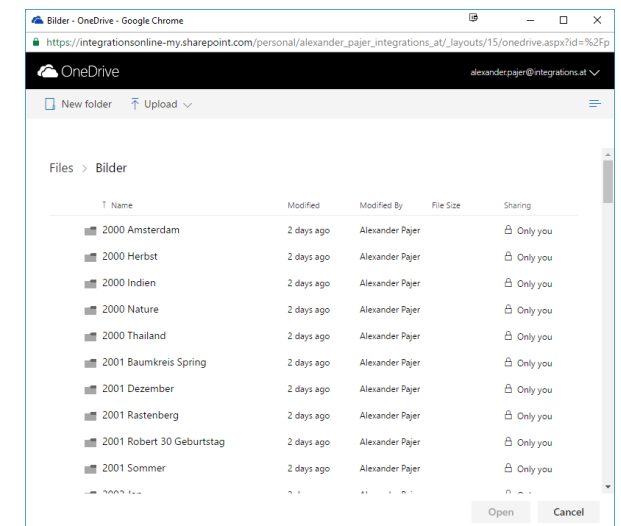
```
<script type="text/javascript">
  function launchOneDrivePicker(){
    var odOptions = { /* ... specify the desired options ... */ };
    OneDrive.open(odOptions);
  }
</script>
<button onClick="launchOneDrivePicker">Open from OneDrive</button>
```

Options ☐ Enable multiselect

Link Type ☒ Download link
☐ Sharing link
☐ Query only

Button 

Result Pick a file to OneDrive to see the return result



Sharing & Permissions

One Drive API offeres methods for Sharing & Permissions

Send invitation

```
POST /drive/items/{item-id}/action.invite
Content-Type: application/json

{
  "requireSignIn": false,
  "sendInvitation": false,
  "roles": ["write"],
  "recipients": [
    { "email": "johndoe@contoso.com" },
    { "email": "accounts_team@contoso.com" }
  ],
  "message": "Here's the document I was talking about yesterday."
}
```

List permissions

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "value": [
    {
      "id": "1",
      "roles": ["write"],
      "link": {
        "webUrl": "https://onedrive.live.com/redir?resid=5D33DD65C6932946!70859&authkey=!AL7N1QAfSWcjNU8&ithint=folder%2cgif",
        "type": "edit"
      }
    },
    {
      "id": "2",
      "roles": ["write"],
      "grantedTo": {
        "user": {
          "id": "5D33DD65C6932946",
          "displayName": "John Doe"
        }
      },
      "inheritedFrom": {
        "driveId": "1234567890ABD",
        "id": "1234567890ABC!123",
        "path": "/drive/root:/Documents"
      }
    }
  ],
}
```

Access One Drive using Graph

Microsoft Graph exposes two resource types for working with files:

Drive - Represents a logical container of files, like a document library or a user's OneDrive.

DriveItem - Represents an item within a drive, like a document, photo, video, or folder.

Path	Resource
<code>/me/drive</code>	User's OneDrive
<code>/me/drives</code>	Enumerate OneDrive resources available to the user.
<code>/drives/{drive-id}</code>	Access a specific Drive by the drive's ID.
<code>/drives/{drive-id}/root/children</code>	Enumerate the DriveItem resources in the root of a specific Drive .
<code>/me/drive/items/{item-id}</code>	Access a DriveItem in the user's OneDrive by its unique ID.
<code>/me/drive/special/{special-id}</code>	Access a special (named) folder in the user's OneDrive by its known name.
<code>/users/{user-id}/drive</code>	Access another user's OneDrive by using the user's unique ID.
<code>/groups/{group-id}/drive</code>	Access the default document library for a group by the group's unique ID.
<code>/shares/{share-id}</code>	Access a DriveItem by its sharedId or sharing URL.

Permissions

The permissions collection includes potentially sensitive information and may not be available for every caller.

- For the owner of the item, all permissions will be returned. This includes co-owners.
- For a non-owner caller, only the permissions that apply to the caller are returned.
- Permission properties that contain secrets (e.g. shareId and webUrl) are only returned for callers that are able to create the Permission.

```
GET /me/drive/items/{item-id}/permissions  
GET /me/drive/root/{path}:/permissions  
GET /drives/{drive-id}/items/{item-id}/permissions
```


Webhooks

WebHooks

The Microsoft Graph REST API uses webhooks to deliver notifications to clients

Using the Microsoft Graph REST API, an app can subscribe to changes on the following resources:

- Messages
- Events
- Contacts
- Group conversations
- Drive root items

Subscription request example

```
POST https://graph.microsoft.com/v1.0/subscriptions
Content-Type: application/json
{
  "changeType": "created,updated",
  "notificationUrl": "https://webhook.azurewebsites.net/notificationClient",
  "resource": "/me/mailfolders('inbox')/messages",
  "expirationDateTime": "2016-03-20T11:00:00.0000000Z",
  "clientState": "SecretClientState"
}
```

Webhook-enabled list item event types

Supported asynchronous list item events in SharePoint

- ItemAdded
- ItemUpdated
- ItemDeleted
- ItemCheckedOut
- ItemCheckedIn
- ItemUncheckedOut
- ItemAttachmentAdded
- ItemAttachmentDeleted
- ItemFileMoved
- ItemVersionDeleted
- ItemFileConverted

Synchronous events are not supported in Webhooks

Required Permissions To Register Webhooks

Microsoft Azure Active Directory (AD) applications

- Set the following Azure AD application permissions

Application	Permission
Office 365 SharePoint Online	Read and write items and lists in all site collections.

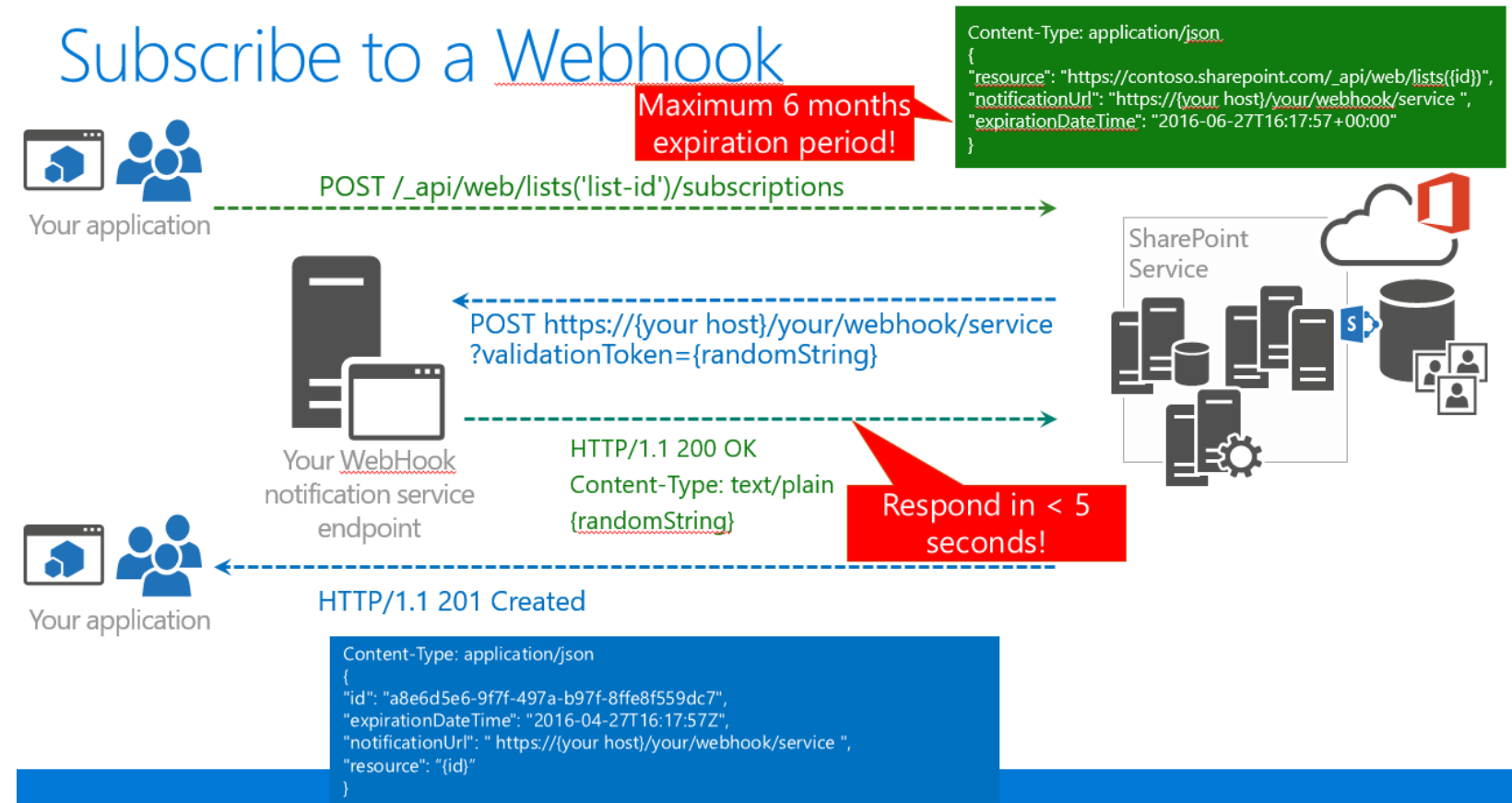
SharePoint add-in

- Set the following SharePoint Add-in permissions (or higher)

Scope	Permission Rights
List	Manage

Subscribe to a Webhook

In a Subscription the Webservice receiving the Post is registered



Process Notifications

Notifications can be processed using

- Stand Alone Service
- Azure Functions

Changes can be received using

- Change Token
- GetChanges()

