OneNote JavaScript API programming overview

Article • 07/22/2024

OneNote introduces a JavaScript API for OneNote add-ins on the web. You can create task pane add-ins, content add-ins, and add-in commands that interact with OneNote objects and connect to web services or other web-based resources.

Note

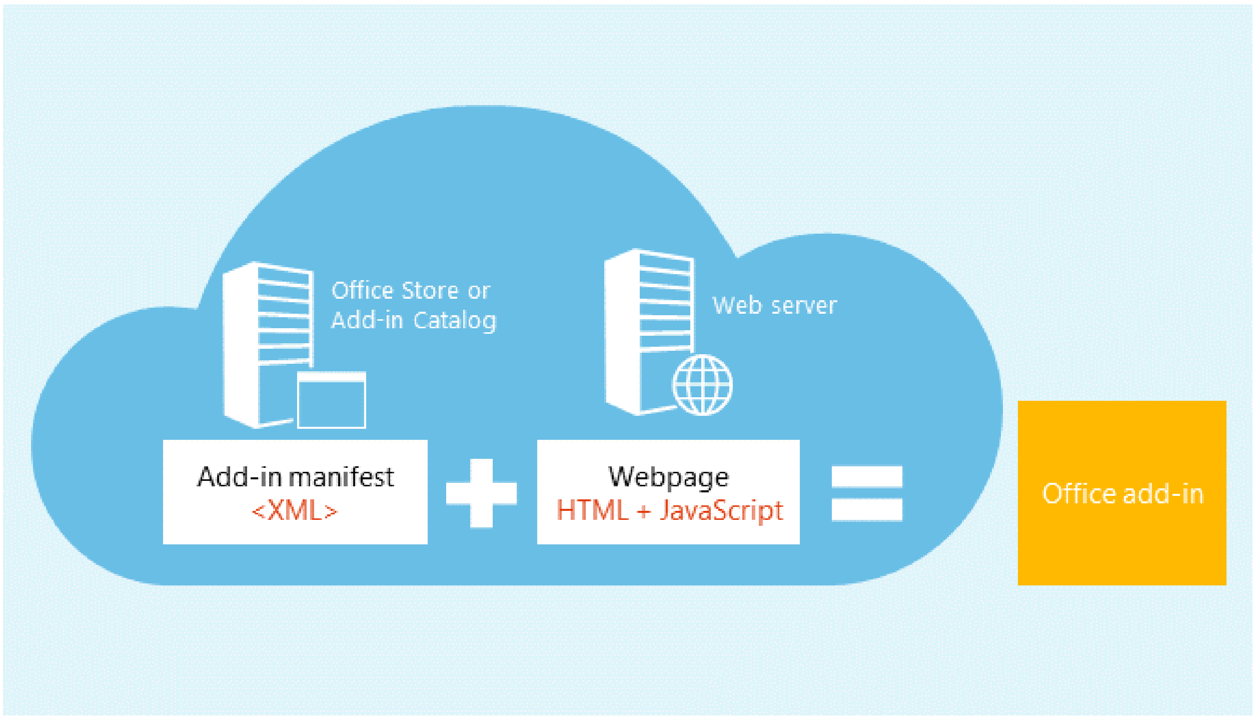
If you plan to publish your add-in to AppSource and make it available within the Office experience, make sure that you conform to the Commercial marketplace certification policies. For example, to pass validation, your add-in must work across all platforms that support the methods that you define (for more information, see section 1120.3 and the Office Add-in application and availability page).

Components of an Office Add-in

Add-ins consist of two basic components:

- A **web application** consisting of a webpage and any required JavaScript, CSS, or other files. These files are hosted on a web server or web hosting service, such as Microsoft Azure. In OneNote on the web, the web application displays in a webview control or iframe.
- A **manifest** that specifies the URL of the add-in's webpage and any access requirements, settings, and capabilities for the add-in. This file is stored on the client. OneNote add-ins use the add-in only manifest format.

Office Add-in = Manifest + Webpage



Using the JavaScript API

Add-ins use the runtime context of the Office application to access the JavaScript API. The API has two layers:

- A **application-specific API** for OneNote-specific operations, accessed through the `Application` object.
- A **Common API** that's shared across Office applications, accessed through the `Document` object.

Accessing the application-specific API through the *Application* object

Use the `Application` object to access OneNote objects such as **Notebook**, **Section**, and **Page**. With application-specific APIs, you run batch operations on proxy objects. The basic flow goes something like this:

1. Get the application instance from the context.

2. Create a proxy that represents the OneNote object you want to work with. You interact synchronously with proxy objects by reading and writing their properties and calling their methods.

3. Call `load` on the proxy to fill it with the property values specified in the parameter. This call is added to the queue of commands.

> ⓘ **Note**
>
> Method calls to the API (such as `context.application.getActiveSection().pages;`) are also added to the queue.

4. Call `context.sync` to run all queued commands in the order that they were queued. This synchronizes the state between your running script and the real objects, and by retrieving properties of loaded OneNote objects for use in your script. You can use the returned promise object for chaining additional actions.

For example:

JavaScript

```javascript
async function getPagesInSection() {
    await OneNote.run(async (context) => {

        // Get the pages in the current section.
        const pages = context.application.getActiveSection().pages;

        // Queue a command to load the id and title for each page.
        pages.load('id,title');

        // Run the queued commands, and return a promise to indicate task
completion.
        await context.sync();

        // Read the id and title of each page.
        $.each(pages.items, function(index, page) {
            let pageId = page.id;
            let pageTitle = page.title;
            console.log(pageTitle + ': ' + pageId);
        });
    });
}
```

See Using the application-specific API model to learn more about the `load`/`sync` pattern and other common practices in the OneNote JavaScript APIs.

You can find supported OneNote objects and operations in the API reference.

OneNote JavaScript API requirement sets

Requirement sets are named groups of API members. Office Add-ins use requirement sets specified in the manifest or use a runtime check to determine whether an Office application supports APIs that an add-in needs. For detailed information about OneNote JavaScript API requirement sets, see OneNote JavaScript API requirement sets.

Accessing the Common API through the *Document* object

Use the `Document` object to access the Common API, such as the getSelectedDataAsync and setSelectedDataAsync methods.

For example:

```javascript
function getSelectionFromPage() {
    Office.context.document.getSelectedDataAsync(
        Office.CoercionType.Text,
        { valueFormat: "unformatted" },
        function (asyncResult) {
            const error = asyncResult.error;
            if (asyncResult.status === Office.AsyncResultStatus.Failed) {
                console.log(error.message);
            }
            else $('#input').val(asyncResult.value);
        });
}
```

OneNote add-ins support only the following Common APIs.

Expand table

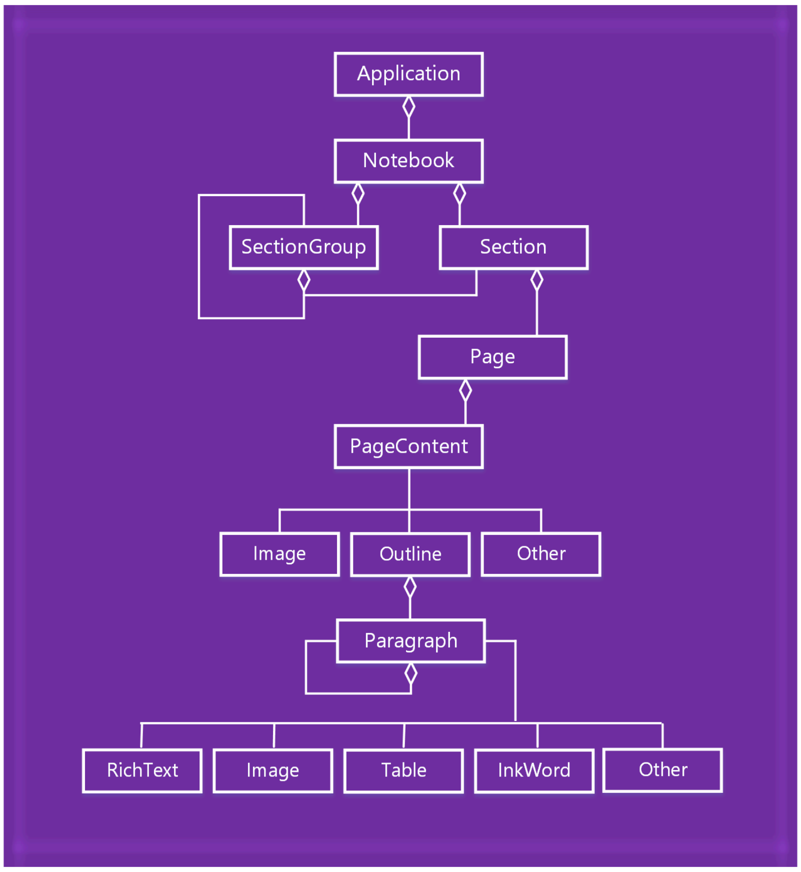
API	Notes
Office.context.document.getSelectedDataAsync	`Office.CoercionType.Text` and `Office.CoercionType.Matrix` only
Office.context.document.setSelectedDataAsync	`Office.CoercionType.Text`, `Office.CoercionType.Image`, and `Office.CoercionType.Html` only
const mySetting = Office.context.document.settings.get(name);	Settings are supported by content add-ins only
Office.context.document.settings.set(name, value);	Settings are supported by content add-ins only
Office.EventType.DocumentSelectionChanged	*None*

In general, you use the Common API to do something that isn't supported in the application-specific API. To learn more about using the Common API, see Common

JavaScript API object model.

OneNote object model diagram

The following diagram represents what's currently available in the OneNote JavaScript API.



See also

- Developing Office Add-ins
- Build your first OneNote add-in
- OneNote JavaScript API reference
- Sample: Rubric Grader

Build your first OneNote task pane add-in

Article • 09/17/2024

In this article, you'll walk through the process of building a OneNote task pane add-in.

Prerequisites

- Node.js (the latest LTS version). Visit the Node.js site to download and install the right version for your operating system.

- The latest version of Yeoman and the Yeoman generator for Office Add-ins. To install these tools globally, run the following command via the command prompt.

  command line

  ```
  npm install -g yo generator-office
  ```

  Note

  Even if you've previously installed the Yeoman generator, we recommend you update your package to the latest version from npm.

- Office connected to a Microsoft 365 subscription (including Office on the web).

  Note

  If you don't already have Office, you might qualify for a Microsoft 365 E5 developer subscription through the Microsoft 365 Developer Program; for details, see the FAQ. Alternatively, you can sign up for a 1-month free trial or purchase a Microsoft 365 plan.

Create the add-in project

Run the following command to create an add-in project using the Yeoman generator. A folder that contains the project will be added to the current directory.

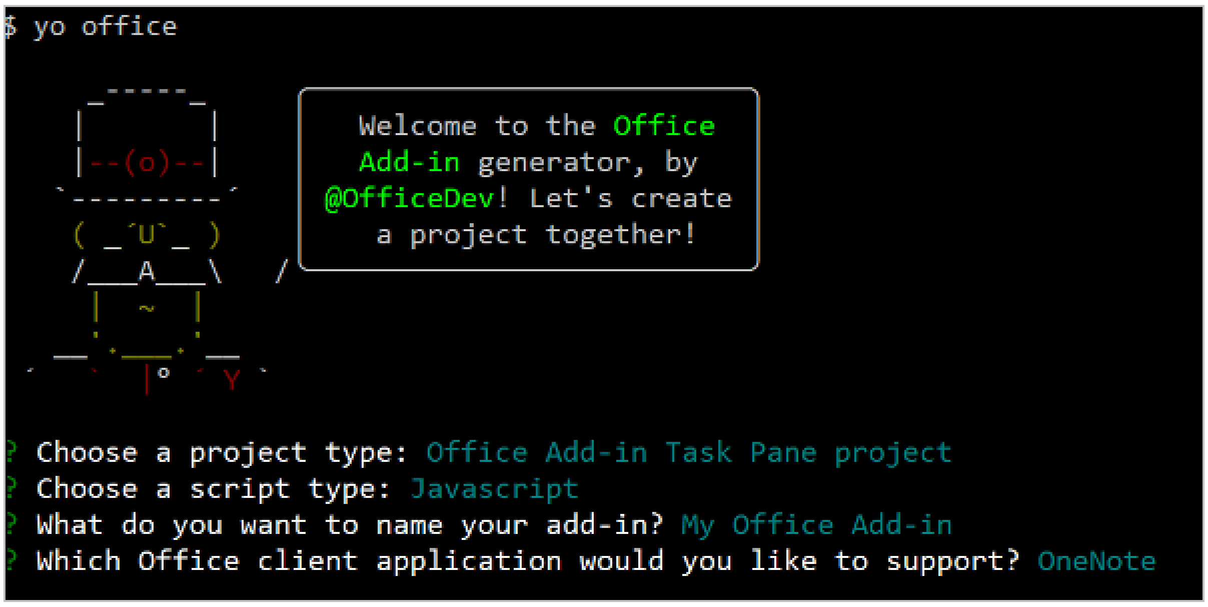
command line

```
yo office
```

> **Note**
>
> When you run the `yo office` command, you may receive prompts about the data collection policies of Yeoman and the Office Add-in CLI tools. Use the information that's provided to respond to the prompts as you see fit.

When prompted, provide the following information to create your add-in project.

- **Choose a project type:** `Office Add-in Task Pane project`
- **Choose a script type:** `Javascript`
- **What do you want to name your add-in?** `My Office Add-in`
- **Which Office client application would you like to support?** `OneNote`



```
$ yo office

                        Welcome to the Office
                        Add-in generator, by
                        @OfficeDev! Let's create
                        a project together!

? Choose a project type: Office Add-in Task Pane project
? Choose a script type: Javascript
? What do you want to name your add-in? My Office Add-in
? Which Office client application would you like to support? OneNote
```

After you complete the wizard, the generator creates the project and installs supporting Node components.

Explore the project

The add-in project that you've created with the Yeoman generator contains sample code for a very basic task pane add-in.

- The **./manifest.xml** file in the root directory of the project defines the settings and capabilities of the add-in.
- The **./src/taskpane/taskpane.html** file contains the HTML markup for the task pane.

- The **./src/taskpane/taskpane.css** file contains the CSS that's applied to content in the task pane.
- The **./src/taskpane/taskpane.js** file contains the Office JavaScript API code that facilitates interaction between the task pane and the Office client application.

Update the code

In your code editor, open the file **./src/taskpane/taskpane.js** and add the following code within the `run` function. This code uses the OneNote JavaScript API to set the page title and add an outline to the body of the page.

JavaScript

```javascript
try {
    await OneNote.run(async (context) => {

        // Get the current page.
        const page = context.application.getActivePage();

        // Queue a command to set the page title.
        page.title = "Hello World";

        // Queue a command to add an outline to the page.
        const html = "<p><ol><li>Item #1</li><li>Item #2</li></ol></p>";
        page.addOutline(40, 90, html);

        // Run the queued commands.
        await context.sync();
    });
} catch (error) {
    console.log("Error: " + error);
}
```

Try it out

1. Navigate to the root folder of the project.

   command line

   ```
   cd "My Office Add-in"
   ```

2. Start the local web server. Run the following command in the root directory of your project.

   command line

```
npm run dev-server
```

> ⓘ **Note**
>
> Office Add-ins should use HTTPS, not HTTP, even while you're developing. If you're prompted to install a certificate after you run one of the following commands, accept the prompt to install the certificate that the Yeoman generator provides. You may also have to run your command prompt or terminal as an administrator for the changes to be made.

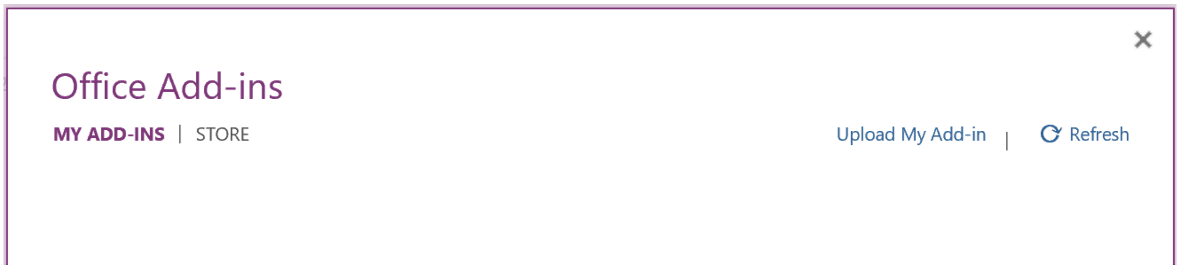
3. In OneNote on the web, open a notebook and create a new page.

4. Choose **Insert** > **Office Add-ins** to open the Office Add-ins dialog.

   - If you're signed in with your consumer account, select the **MY ADD-INS** tab, and then choose **Upload My Add-in**.

   - If you're signed in with your work or education account, select the **MY ORGANIZATION** tab, and then select **Upload My Add-in**.

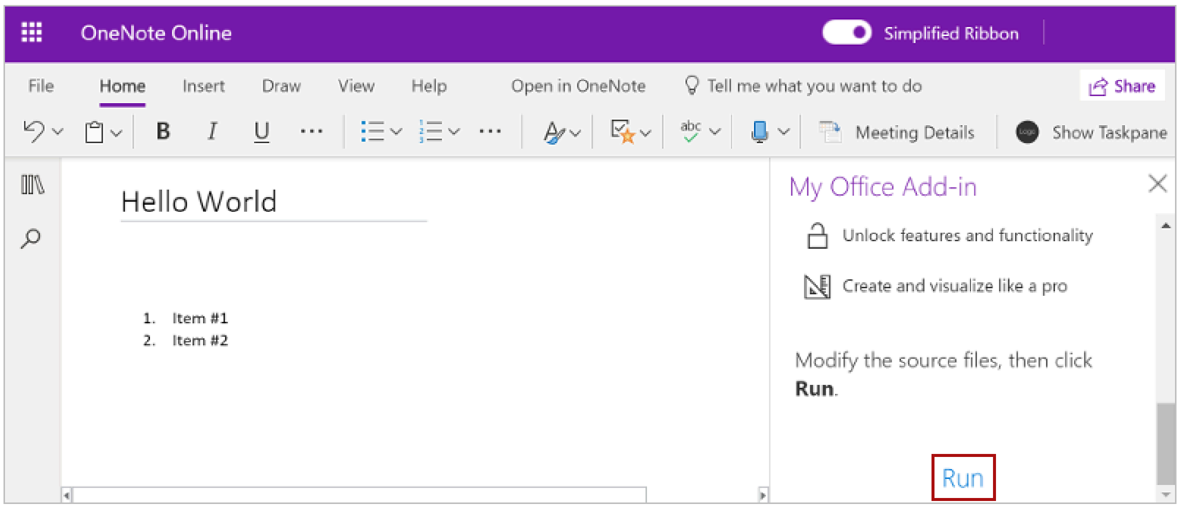
   The following image shows the **MY ADD-INS** tab for consumer notebooks.



5. In the Upload Add-in dialog, browse to **manifest.xml** in your project folder, and then choose **Upload**.

6. From the **Home** tab, choose the **Show Taskpane** button on the ribbon. The add-in task pane opens in an iFrame next to the OneNote page.

7. At the bottom of the task pane, choose the **Run** link to set the page title and add an outline to the body of the page.



8. When you want to stop the local web server and uninstall the add-in, follow these instructions:

   - To stop the server, run the following command.

     command line

     ```
     npm stop
     ```

   - To uninstall the sideloaded add-in, see Remove a sideloaded add-in.

Next steps

Congratulations, you've successfully created a OneNote task pane add-in! Next, learn more about the core concepts of building OneNote add-ins.

OneNote JavaScript API programming overview

Troubleshooting

- Ensure your environment is ready for Office development by following the instructions in Set up your development environment.

- Some of the sample code uses ES6 JavaScript. This isn't compatible with older versions of Office that use the Trident (Internet Explorer 11) browser engine. For information on how to support those platforms in your add-in, see Support older Microsoft webviews and Office versions. If you don't already have a Microsoft 365 subscription to use for development, you might qualify for a Microsoft 365 E5 developer subscription through the Microsoft 365 Developer Program ; for details, see the FAQ. Alternatively, you can sign up for a 1-month free trial or purchase a Microsoft 365 plan.

- The automatic `npm install` step Yo Office performs may fail. If you see errors when trying to run `npm start`, navigate to the newly created project folder in a command prompt and manually run `npm install`. For more information about Yo Office, see Create Office Add-in projects using the Yeoman Generator.

See also

- Office Add-ins platform overview
- Develop Office Add-ins
- OneNote JavaScript API programming overview
- OneNote JavaScript API reference
- Rubric Grader sample
- Using Visual Studio Code to publish

OneNote JavaScript API overview

06/18/2025

A OneNote add-in interacts with objects in OneNote on the web by using the Office JavaScript API Library, which includes two JavaScript object models:

- **OneNote JavaScript API**: These are the application-specific APIs for OneNote. Introduced with Office 2016, the OneNote JavaScript API provides strongly-typed objects that you can use to access objects in OneNote on the web.
- **Common APIs**: Introduced with Office 2013, the Common API can be used to access features such as UI, dialogs, and client settings that are common across multiple types of Office applications.

This section of the documentation focuses on the OneNote JavaScript API, which you'll use to develop most of the functionality in add-ins that target OneNote on the web. For information about the Common API, see Common JavaScript API object model.

Learn programming concepts

See the following articles for information about important programming concepts related to OneNote extensibility.

- OneNote JavaScript API programming overview
- Work with OneNote page content

Learn about API capabilities

For hands-on experience using the OneNote JavaScript API to interact with content in OneNote on the web, complete the OneNote add-in quick start.

For detailed information about the OneNote JavaScript API object model, see the OneNote JavaScript API reference documentation.

See also

- OneNote add-ins documentation
- OneNote add-ins overview
- OneNote JavaScript API reference
- Office client application and platform availability for Office Add-ins

onenote package

Classes

Expand table

OneNote.Application	Represents the top-level object that contains all globally addressable OneNote objects such as notebooks, the active notebook, and the active section.
OneNote.FloatingInk	Represents a group of ink strokes.
OneNote.Image	Represents an Image. An Image can be a direct child of a PageContent object or a Paragraph object.
OneNote.InkAnalysis	Represents ink analysis data for a given set of ink strokes.
OneNote.InkAnalysisLine	Represents ink analysis data for an identified text line formed by ink strokes.
OneNote.InkAnalysisLineCollection	Represents a collection of InkAnalysisLine objects.
OneNote.InkAnalysisParagraph	Represents ink analysis data for an identified paragraph formed by ink strokes.
OneNote.InkAnalysisParagraphCollection	Represents a collection of InkAnalysisParagraph objects.
OneNote.InkAnalysisWord	Represents ink analysis data for an identified word formed by ink strokes.
OneNote.InkAnalysisWordCollection	Represents a collection of InkAnalysisWord objects.
OneNote.InkStroke	Represents a single stroke of ink.
OneNote.InkStrokeCollection	Represents a collection of InkStroke objects.
OneNote.InkWord	A container for the ink in a word in a paragraph.
OneNote.InkWordCollection	Represents a collection of InkWord objects.
OneNote.Notebook	Represents a OneNote notebook. Notebooks contain section groups and sections.
OneNote.NotebookCollection	Represents a collection of notebooks.

OneNote.NoteTag	A container for the NoteTag in a paragraph.
OneNote.Outline	Represents a container for Paragraph objects.
OneNote.Page	Represents a OneNote page.
OneNote.PageCollection	Represents a collection of pages.
OneNote.PageContent	Represents a region on a page that contains top-level content types such as Outline or Image. A PageContent object can be assigned an XY position.
OneNote.PageContent Collection	Represents the contents of a page, as a collection of PageContent objects.
OneNote.Paragraph	A container for the visible content on a page. A Paragraph can contain any one ParagraphType type of content.
OneNote.Paragraph Collection	Represents a collection of Paragraph objects.
OneNote.Point	Represents a single point of ink stroke
OneNote.Point Collection	Represents a collection of Point objects.
OneNote.RequestContext	
OneNote.RichText	Represents a RichText object in a Paragraph.
OneNote.Section	Represents a OneNote section. Sections can contain pages.
OneNote.Section Collection	Represents a collection of sections.
OneNote.SectionGroup	Represents a OneNote section group. Section groups can contain sections and other section groups.
OneNote.SectionGroup Collection	Represents a collection of section groups.
OneNote.Table	Represents a table in a OneNote page.
OneNote.TableCell	Represents a cell in a OneNote table.
OneNote.TableCell Collection	Contains a collection of TableCell objects.
OneNote.TableRow	Represents a row in a table.
OneNote.TableRow Collection	Contains a collection of TableRow objects.

Interfaces

Expand table

OneNote.ImageOcrData	Represents data obtained by OCR (optical character recognition) of an image.
OneNote.InkStrokePointer	Weak reference to an ink stroke object and its content parent.
OneNote.Interfaces.ApplicationData	An interface describing the data returned by calling `application.toJSON()`.
OneNote.Interfaces.ApplicationLoadOptions	Represents the top-level object that contains all globally addressable OneNote objects such as notebooks, the active notebook, and the active section.
OneNote.Interfaces.ApplicationUpdateData	An interface for updating data on the `Application` object, for use in `application.set({ ... })`.
OneNote.Interfaces.CollectionLoadOptions	Provides ways to load properties of only a subset of members of a collection.
OneNote.Interfaces.FloatingInkData	An interface describing the data returned by calling `floatingInk.toJSON()`.
OneNote.Interfaces.FloatingInkLoadOptions	Represents a group of ink strokes.
OneNote.Interfaces.ImageData	An interface describing the data returned by calling `image.toJSON()`.
OneNote.Interfaces.ImageLoadOptions	Represents an Image. An Image can be a direct child of a PageContent object or a Paragraph object.
OneNote.Interfaces.ImageUpdateData	An interface for updating data on the `Image` object, for use in `image.set({ ... })`.
OneNote.Interfaces.InkAnalysisData	An interface describing the data returned by calling `inkAnalysis.toJSON()`.
OneNote.Interfaces.InkAnalysisLineCollectionData	An interface describing the data returned by calling `inkAnalysisLineCollection.toJSON()`.
OneNote.Interfaces.InkAnalysisLineCollectionLoadOptions	Represents a collection of InkAnalysisLine objects.
OneNote.Interfaces.InkAnalysisLineCollectionUpdateData	An interface for updating data on the `InkAnalysisLineCollection` object, for use in `inkAnalysisLineCollection.set({ ... })`.

OneNote.Interfaces.InkAnalysisLineData	An interface describing the data returned by calling `inkAnalysisLine.toJSON()`.
OneNote.Interfaces.InkAnalysisLineLoadOptions	Represents ink analysis data for an identified text line formed by ink strokes.
OneNote.Interfaces.InkAnalysisLineUpdateData	An interface for updating data on the `InkAnalysisLine` object, for use in `inkAnalysisLine.set({ ... })`.
OneNote.Interfaces.InkAnalysisLoadOptions	Represents ink analysis data for a given set of ink strokes.
OneNote.Interfaces.InkAnalysisParagraphCollectionData	An interface describing the data returned by calling `inkAnalysisParagraphCollection.toJSON()`.
OneNote.Interfaces.InkAnalysisParagraphCollectionLoadOptions	Represents a collection of InkAnalysisParagraph objects.
OneNote.Interfaces.InkAnalysisParagraphCollectionUpdateData	An interface for updating data on the `InkAnalysisParagraphCollection` object, for use in `inkAnalysisParagraphCollection.set({ ... })`.
OneNote.Interfaces.InkAnalysisParagraphData	An interface describing the data returned by calling `inkAnalysisParagraph.toJSON()`.
OneNote.Interfaces.InkAnalysisParagraphLoadOptions	Represents ink analysis data for an identified paragraph formed by ink strokes.
OneNote.Interfaces.InkAnalysisParagraphUpdateData	An interface for updating data on the `InkAnalysisParagraph` object, for use in `inkAnalysisParagraph.set({ ... })`.
OneNote.Interfaces.InkAnalysisUpdateData	An interface for updating data on the `InkAnalysis` object, for use in `inkAnalysis.set({ ... })`.
OneNote.Interfaces.InkAnalysisWordCollectionData	An interface describing the data returned by calling `inkAnalysisWordCollection.toJSON()`.
OneNote.Interfaces.InkAnalysisWordCollectionLoadOptions	Represents a collection of InkAnalysisWord objects.
OneNote.Interfaces.InkAnalysisWordCollectionUpdateData	An interface for updating data on the `InkAnalysisWordCollection` object, for use in `inkAnalysisWordCollection.set({ ... })`.

OneNote.Interfaces.InkAnalysisWordData	An interface describing the data returned by calling `inkAnalysisWord.toJSON()`.
OneNote.Interfaces.InkAnalysisWordLoadOptions	Represents ink analysis data for an identified word formed by ink strokes.
OneNote.Interfaces.InkAnalysisWordUpdateData	An interface for updating data on the `InkAnalysisWord` object, for use in `inkAnalysisWord.set({ ... })`.
OneNote.Interfaces.InkStrokeCollectionData	An interface describing the data returned by calling `inkStrokeCollection.toJSON()`.
OneNote.Interfaces.InkStrokeCollectionLoadOptions	Represents a collection of InkStroke objects.
OneNote.Interfaces.InkStrokeCollectionUpdateData	An interface for updating data on the `InkStrokeCollection` object, for use in `inkStrokeCollection.set({ ... })`.
OneNote.Interfaces.InkStrokeData	An interface describing the data returned by calling `inkStroke.toJSON()`.
OneNote.Interfaces.InkStrokeLoadOptions	Represents a single stroke of ink.
OneNote.Interfaces.InkWordCollectionData	An interface describing the data returned by calling `inkWordCollection.toJSON()`.
OneNote.Interfaces.InkWordCollectionLoadOptions	Represents a collection of InkWord objects.
OneNote.Interfaces.InkWordCollectionUpdateData	An interface for updating data on the `InkWordCollection` object, for use in `inkWordCollection.set({ ... })`.
OneNote.Interfaces.InkWordData	An interface describing the data returned by calling `inkWord.toJSON()`.
OneNote.Interfaces.InkWordLoadOptions	A container for the ink in a word in a paragraph.
OneNote.Interfaces.NotebookCollectionData	An interface describing the data returned by calling `notebookCollection.toJSON()`.
OneNote.Interfaces.NotebookCollectionLoadOptions	Represents a collection of notebooks.
OneNote.Interfaces.NotebookCollectionUpdate	An interface for updating data on the `NotebookCollection` object, for use in `notebookCollection.set({ ... })`.

Data	
OneNote.Interfaces.NotebookData	An interface describing the data returned by calling `notebook.toJSON()`.
OneNote.Interfaces.NotebookLoadOptions	Represents a OneNote notebook. Notebooks contain section groups and sections.
OneNote.Interfaces.NoteTagData	An interface describing the data returned by calling `noteTag.toJSON()`.
OneNote.Interfaces.NoteTagLoadOptions	A container for the NoteTag in a paragraph.
OneNote.Interfaces.OutlineData	An interface describing the data returned by calling `outline.toJSON()`.
OneNote.Interfaces.OutlineLoadOptions	Represents a container for Paragraph objects.
OneNote.Interfaces.PageCollectionData	An interface describing the data returned by calling `pageCollection.toJSON()`.
OneNote.Interfaces.PageCollectionLoadOptions	Represents a collection of pages.
OneNote.Interfaces.PageCollectionUpdateData	An interface for updating data on the `PageCollection` object, for use in `pageCollection.set({ ... })`.
OneNote.Interfaces.PageContentCollectionData	An interface describing the data returned by calling `pageContentCollection.toJSON()`.
OneNote.Interfaces.PageContentCollectionLoadOptions	Represents the contents of a page, as a collection of PageContent objects.
OneNote.Interfaces.PageContentCollectionUpdateData	An interface for updating data on the `PageContentCollection` object, for use in `pageContentCollection.set({ ... })`.
OneNote.Interfaces.PageContentData	An interface describing the data returned by calling `pageContent.toJSON()`.
OneNote.Interfaces.PageContentLoadOptions	Represents a region on a page that contains top-level content types such as Outline or Image. A PageContent object can be assigned an XY position.
OneNote.Interfaces.PageContentUpdateData	An interface for updating data on the `PageContent` object, for use in `pageContent.set({ ... })`.
OneNote.Interfaces.PageData	An interface describing the data returned by calling `page.toJSON()`.

OneNote.Interfaces.PageLoadOptions	Represents a OneNote page.
OneNote.Interfaces.PageUpdateData	An interface for updating data on the `Page` object, for use in `page.set({ ... })`.
OneNote.Interfaces.ParagraphCollectionData	An interface describing the data returned by calling `paragraphCollection.toJSON()`.
OneNote.Interfaces.ParagraphCollectionLoadOptions	Represents a collection of Paragraph objects.
OneNote.Interfaces.ParagraphCollectionUpdateData	An interface for updating data on the ParagraphCollection object, for use in `paragraphCollection.set({ ... })`.
OneNote.Interfaces.ParagraphData	An interface describing the data returned by calling `paragraph.toJSON()`.
OneNote.Interfaces.ParagraphLoadOptions	A container for the visible content on a page. A Paragraph can contain any one ParagraphType type of content.
OneNote.Interfaces.ParagraphUpdateData	An interface for updating data on the `Paragraph` object, for use in `paragraph.set({ ... })`.
OneNote.Interfaces.PointCollectionData	An interface describing the data returned by calling `pointCollection.toJSON()`.
OneNote.Interfaces.PointCollectionLoadOptions	Represents a collection of Point objects.
OneNote.Interfaces.PointCollectionUpdateData	An interface for updating data on the `PointCollection` object, for use in `pointCollection.set({ ... })`.
OneNote.Interfaces.PointData	An interface describing the data returned by calling `point.toJSON()`.
OneNote.Interfaces.PointLoadOptions	Represents a single point of ink stroke
OneNote.Interfaces.RichTextData	An interface describing the data returned by calling `richText.toJSON()`.
OneNote.Interfaces.RichTextLoadOptions	Represents a RichText object in a Paragraph.
OneNote.Interfaces.SectionCollectionData	An interface describing the data returned by calling `sectionCollection.toJSON()`.
OneNote.Interfaces.Section	Represents a collection of sections.

CollectionLoadOptions	
OneNote.Interfaces.SectionCollectionUpdateData	An interface for updating data on the `SectionCollection` object, for use in `sectionCollection.set({ ... })`.
OneNote.Interfaces.SectionData	An interface describing the data returned by calling `section.toJSON()`.
OneNote.Interfaces.SectionGroupCollectionData	An interface describing the data returned by calling `sectionGroupCollection.toJSON()`.
OneNote.Interfaces.SectionGroupCollectionLoadOptions	Represents a collection of section groups.
OneNote.Interfaces.SectionGroupCollectionUpdateData	An interface for updating data on the `SectionGroupCollection` object, for use in `sectionGroupCollection.set({ ... })`.
OneNote.Interfaces.SectionGroupData	An interface describing the data returned by calling `sectionGroup.toJSON()`.
OneNote.Interfaces.SectionGroupLoadOptions	Represents a OneNote section group. Section groups can contain sections and other section groups.
OneNote.Interfaces.SectionLoadOptions	Represents a OneNote section. Sections can contain pages.
OneNote.Interfaces.TableCellCollectionData	An interface describing the data returned by calling `tableCellCollection.toJSON()`.
OneNote.Interfaces.TableCellCollectionLoadOptions	Contains a collection of TableCell objects.
OneNote.Interfaces.TableCellCollectionUpdateData	An interface for updating data on the `TableCellCollection` object, for use in `tableCellCollection.set({ ... })`.
OneNote.Interfaces.TableCellData	An interface describing the data returned by calling `tableCell.toJSON()`.
OneNote.Interfaces.TableCellLoadOptions	Represents a cell in a OneNote table.
OneNote.Interfaces.TableCellUpdateData	An interface for updating data on the `TableCell` object, for use in `tableCell.set({ ... })`.
OneNote.Interfaces.TableData	An interface describing the data returned by calling `table.toJSON()`.
OneNote.Interfaces.TableLoadOptions	Represents a table in a OneNote page.

OneNote.Interfaces.TableRowCollectionData	An interface describing the data returned by calling `tableRowCollection.toJSON()`.
OneNote.Interfaces.TableRowCollectionLoadOptions	Contains a collection of TableRow objects.
OneNote.Interfaces.TableRowCollectionUpdateData	An interface for updating data on the `TableRowCollection` object, for use in `tableRowCollection.set({ ... })`.
OneNote.Interfaces.TableRowData	An interface describing the data returned by calling `tableRow.toJSON()`.
OneNote.Interfaces.TableRowLoadOptions	Represents a row in a table.
OneNote.Interfaces.TableUpdateData	An interface for updating data on the `Table` object, for use in `table.set({ ... })`.
OneNote.ParagraphInfo	List information for paragraph.

Enums

Expand table

OneNote.ErrorCodes
OneNote.EventType
OneNote.InsertLocation
OneNote.ListType
OneNote.NoteTagStatus
OneNote.NoteTagType
OneNote.NumberType
OneNote.PageContentType
OneNote.ParagraphStyle
OneNote.ParagraphType

Functions

Expand table

OneNote.run(batch)	Executes a batch script that performs actions on the OneNote object model, using a new request context. When the promise is resolved, any tracked objects that were automatically allocated during execution will be released.
OneNote.run(object, batch)	Executes a batch script that performs actions on the OneNote object model, using the request context of a previously-created API object.
OneNote.run(objects, batch)	Executes a batch script that performs actions on the OneNote object model, using the request context of previously-created API objects.

Function Details

OneNote.run(batch)

Executes a batch script that performs actions on the OneNote object model, using a new request context. When the promise is resolved, any tracked objects that were automatically allocated during execution will be released.

TypeScript

```typescript
export function run<T>(batch: (context: OneNote.RequestContext) => Promise<T>):
Promise<T>;
```

Parameters

`batch` (context: OneNote.RequestContext) => Promise<T>

A function that takes in an OneNote.RequestContext and returns a promise (typically, just the result of "context.sync()"). The context parameter facilitates requests to the OneNote application. Since the Office add-in and the OneNote application run in two different processes, the request context is required to get access to the OneNote object model from the add-in.

Returns

Promise<T>

OneNote.run(object, batch)

Executes a batch script that performs actions on the OneNote object model, using the request context of a previously-created API object.

TypeScript

```
export function run<T>(object: OfficeExtension.ClientObject, batch: (context:
OneNote.RequestContext) => Promise<T>): Promise<T>;
```

Parameters

`object` OfficeExtension.ClientObject

A previously-created API object. The batch will use the same request context as the passed-in object, which means that any changes applied to the object will be picked up by "context.sync()".

`batch` (context: OneNote.RequestContext) => Promise<T>

A function that takes in an OneNote.RequestContext and returns a promise (typically, just the result of "context.sync()"). When the promise is resolved, any tracked objects that were automatically allocated during execution will be released.

Returns

Promise<T>

OneNote.run(objects, batch)

Executes a batch script that performs actions on the OneNote object model, using the request context of previously-created API objects.

TypeScript

```
export function run<T>(objects: OfficeExtension.ClientObject[], batch:
(context: OneNote.RequestContext) => Promise<T>): Promise<T>;
```

Parameters

`objects` OfficeExtension.ClientObject[]

`batch` (context: OneNote.RequestContext) => Promise<T>

A function that takes in an OneNote.RequestContext and returns a promise (typically, just the result of "context.sync()"). When the promise is resolved, any tracked objects that were automatically allocated during execution will be released.

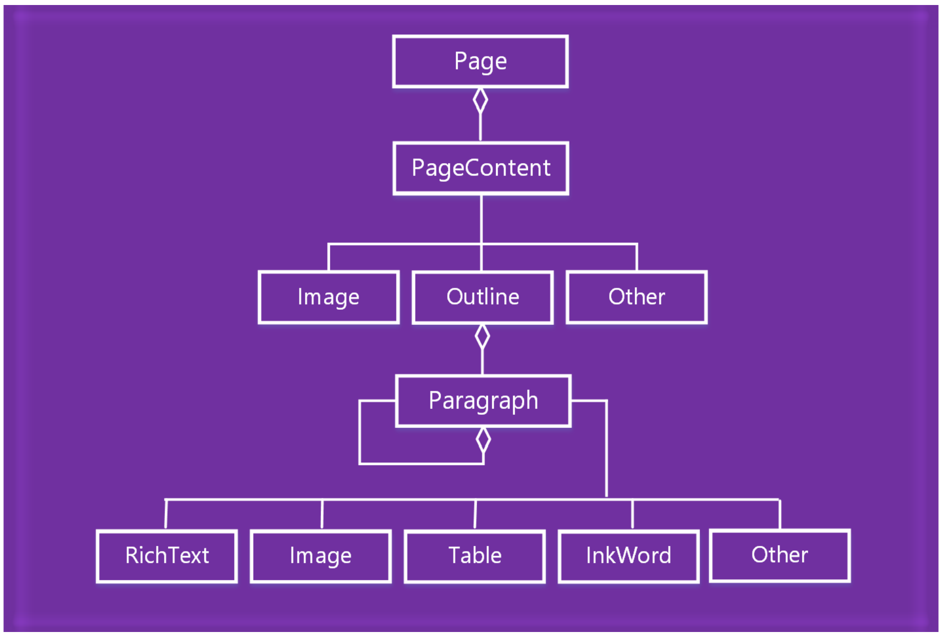
Returns

Promise<T>

Work with OneNote page content

Article • 03/22/2022

In the OneNote add-ins JavaScript API, page content is represented by the following object model.



- A Page object contains a collection of PageContent objects.
- A PageContent object contains a content type of Outline, Image, or Other.
- An Outline object contains a collection of Paragraph objects.
- A Paragraph object contains a content type of RichText, Image, Table, or Other.

To create an empty OneNote page, use one of the following methods.

- Section.addPage
- Page.insertPageAsSibling

Then use methods in the following objects to work with the page content, such as `Page.addOutline` and `Outline.appendHtml`.

- Page
- Outline
- Paragraph

The content and structure of a OneNote page are represented by HTML. Only a subset of HTML is supported for creating or updating page content, as described below.

Supported HTML

The OneNote add-in JavaScript API supports the following HTML for creating and updating page content.

- `<html>`, `<body>`, `<div>`, `<span>`, `<br/>`
- `<p>`
- `<img>`
- `<a>`
- `<ul>`, `<ol>`, `<li>`
- `<table>`, `<tr>`, `<td>`
- `<h1>` ... `<h6>`
- `<b>`, `<em>`, `<strong>`, `<i>`, `<u>`, `<del>`, `<sup>`, `<sub>`, `<cite>`

Note

Importing HTML into OneNote consolidates whitespace. The resulting content is pasted into one outline.

OneNote does its best to translate HTML into page content while ensuring security for users. HTML and CSS standards do not exactly match OneNote's content model, so there will be differences in appearances, particularly with CSS stylings. We recommend using the JavaScript objects if specific formatting is needed.

Accessing page contents

You are only able to access *Page Content* via `Page#load` for the currently active page. To change the active page, invoke `navigateToPage($page)`.

Metadata such as title can still be queried for any page.

See also

- OneNote JavaScript API programming overview
- OneNote JavaScript API reference
- Rubric Grader sample
- Office Add-ins platform overview

Outlook add-ins documentation

With Outlook add-ins, you can use familiar web technologies such as HTML, CSS, and JavaScript to build a solution that can run in Outlook across multiple platforms, including on the web, Windows, Mac, and iOS. Learn how to build, test, debug, and publish Outlook add-ins.

About Outlook add-ins

OVERVIEW

What are Outlook add-ins?

QUICKSTART

Build your first Outlook add-in

Explore Office JavaScript API using Script Lab

HOW-TO GUIDE

Use the Outlook JavaScript API to interact with objects

Navigate and use the Mailbox requirement sets

Build an event-based add-in

Use Microsoft Graph to extend your add-in's capabilities

Test and debug an Outlook add-in

Deploy and publish an Outlook add-in

Key Office Add-ins concepts

OVERVIEW

Office Add-ins platform overview

CONCEPT

Core concepts for Office Add-ins

Design Office Add-ins

Develop Office Add-ins

Resources

REFERENCE

Ask questions

Request features

Report issues

Office Add-ins additional resources