Office Add-in code samples

Article • 09/27/2024

These code samples are written to help you learn how to use various features when developing Office Add-ins.

Getting started

The following samples show how to build the simplest Office Add-in with only a manifest, HTML web page, and a logo. These components are the fundamental parts of an Office Add-in. For additional getting started information, see our quick starts and tutorials.

- Excel "Hello world" add-in
- Outlook "Hello world" add-in
- PowerPoint "Hello world" add-in
- Word "Hello world" add-in

Blazor WebAssembly

If your development background is in building VSTO Add-ins, the following samples show how to build Office Web Add-ins using .NET Blazor WebAssembly. You can keep much of your code in C# and Visual Studio.

- Create a Blazor WebAssembly Excel add-in
- Create a Blazor WebAssembly Outlook add-in
- Create a Blazor WebAssembly Word add-in

Excel

Expand table

Name	Description
Data types explorer (preview)	Builds an Excel add-in that allows you to create and explore data types in your workbooks. Data types enable add-in developers to organize complex data structures as objects, such as formatted number values, web images, and entity values.
Open in Teams	Create a new Excel spreadsheet in Microsoft Teams containing data you define.

Name	Description
Insert an external Excel file and populate it with JSON data	Insert an existing template from an external Excel file into the currently open Excel workbook. Then, populate the template with data from a JSON web service.
Create custom contextual tabs on the ribbon	Create a custom contextual tab on the ribbon in the Office UI. The sample creates a table, and when the user moves the focus inside the table, the custom tab is displayed. When the user moves outside the table, the custom tab is hidden.
Custom function sample using web worker	Use web workers in custom functions to prevent blocking the UI of your Office Add-in.
Use storage techniques to access data from an Office Add-in when offline	Implement localStorage to enable limited functionality for your Office Add-in when a user experiences lost connection.
Custom function batching pattern	Batch multiple calls into a single call to reduce the number of network calls to a remote service.

Outlook

Expand table

Name	Description
Report spam or phishing emails in Outlook	Build an integrated spam-reporting solution that's easily discoverable in the Outlook client ribbon. The solution provides the user with a dialog to report an email. It also saves a copy of the reported email to a file for further processing in your backend system.
Encrypt attachments, process meeting request attendees, and react to appointment date/time changes using Outlook event-based activation	Use event-based activation to encrypt attachments when added by the user. Also use event handling for recipients changed in a meeting request, and changes to the start or end date or time in a meeting request.
Identify and tag external recipients using Outlook event-based activation	Use event-based activation to run an Outlook add-in when the user changes recipients while composing a message. The add-in also uses the `appendOnSendAsync` API to add a disclaimer.
Set your signature using Outlook event-based activation	Use event-based activation to run an Outlook add-in when the user creates a new message or appointment. The add-in can respond to events, even when the task pane isn't open. It also uses the `setSignatureAsync` API.

Name	Description
Verify the color categories of a message or appointment before it's sent using Smart Alerts	Use Outlook Smart Alerts to verify that required color categories are applied to a new message or appointment before it's sent.
Verify the sensitivity label of a message	Use the sensitivity label API in an event-based add-in to verify and apply the **Highly Confidential** sensitivity label to applicable outgoing messages.

Word

Expand table

Name	Description
Get, edit, and set OOXML content in a Word document with a Word add-in	This sample shows how to get, edit, and set OOXML content in a Word document. The sample add-in provides a scratch pad to get Office Open XML for your own content and test your own edited Office Open XML snippets.
Import a Word document template with a Word add-in	Shows how to import templates in a Word document.
Load and write Open XML in your Word add-in	This sample add-in shows you how to add a variety of rich content types to a Word document using the setSelectedDataAsync method with ooxml coercion type. The add-in also gives you the ability to show the Office Open XML markup for each sample content type right on the page.
Manage citations with your Word add-in	Shows how to manage citations in a Word document.

Authentication, authorization, and single sign-on (SSO)

Expand table

Name	Description
Office Add-in with SSO using nested app authentication	Shows how to use MSAL.js nested app authentication (NAA) in an Office Add-in to access Microsoft Graph APIs for the signed-in user. The sample displays the signed-in user's name and email. It also

Name	Description
	inserts the names of files from the user's Microsoft OneDrive account into the document.
Outlook add-in with SSO using nested app authentication	Shows how to use MSAL.js nested app authentication (NAA) in an Outlook Add-in to access Microsoft Graph APIs for the signed-in user. The sample displays the signed-in user's name and email. It also inserts the names of files from the user's Microsoft OneDrive account into a new message body.
Use SSO with event-based activation in an Outlook add-in	Shows how to use SSO to access a user's Microsoft Graph data from an event fired in an Outlook add-in.
Single Sign-on (SSO) Sample Outlook Add-in	Use Office's SSO feature to give the add-in access to Microsoft Graph data.
Get OneDrive data using Microsoft Graph and msal.js in an Office Add-in	Build an Office Add-in, as a single-page application (SPA) with no backend, that connects to Microsoft Graph, and access workbooks stored in OneDrive for Business to update a spreadsheet.
Office Add-in auth to Microsoft Graph	Learn how to build a Microsoft Office Add-in that connects to Microsoft Graph, and access workbooks stored in OneDrive for Business to update a spreadsheet.
Outlook Add-in auth to Microsoft Graph .	Build an Outlook add-in that connects to Microsoft Graph, and access workbooks stored in OneDrive for Business to compose a new email message.
Single Sign-on (SSO) Office Add-in with ASP.NET	Use the `getAccessToken` API in Office.js to give the add-in access to Microsoft Graph data. This sample is built on ASP.NET.
Single Sign-on (SSO) Office Add-in with Node.js	Use the `getAccessToken` API in Office.js to give the add-in access to Microsoft Graph data. This sample is built on Node.js.

Office

Expand table

Name	Description
Save custom settings in your Office Add-in	Save custom settings inside an Office Add-in. The add-in stores data as key-value pairs, using the JavaScript API for Office property bag, browser cookies, web storage (localStorage and sessionStorage), or by storing the data in a hidden div in the document.
Use keyboard shortcuts for Office	Create custom keyboard shortcuts to invoke certain actions for your Office Add-in.

Name	Description
Add-in actions	

Shared runtime

Expand table

Name	Description
Share global data with a shared runtime	Set up a basic project that uses the shared runtime to run code for ribbon buttons, task pane, and custom functions in a single browser runtime.
Manage ribbon and task pane UI, and run code on doc open	Create contextual ribbon buttons that are enabled based on the state of your add-in.

Additional samples

Expand table

Name	Description
Use a shared library to migrate your Visual Studio Tools for Office add-in to an Office web add-in	Provides a strategy for code reuse when migrating from VSTO Add-ins to Office Add-ins.
Integrate an Azure function with your Excel custom function	Integrate Azure functions with custom functions to move to the cloud or integrate additional services.
Dynamic DPI code samples	A collection of samples for handling DPI changes in COM, VSTO, and Office Add-ins.

Next steps

Join the Microsoft 365 Developer Program to get resources and information to help you build solutions for the Microsoft 365 platform, including recommendations tailored to your areas of interest.

You might also qualify for a free developer subscription that's renewable for 90 days and comes configured with sample data; for details, see the FAQ.

Office Add-in development lifecycle

06/13/2025

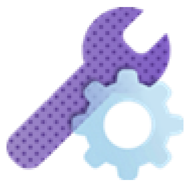
All Office Add-ins are built upon the Office Add-ins platform. They share a common framework through which add-in capabilities are implemented. This means that regardless of whether you're creating an add-in for Excel, Outlook, or another Office application, you can have features such as dialog boxes, add-in commands, task panes, and single sign-on (SSO).

For any add-in you build, you need to understand the following concepts.

- Office application and platform availability
- Office JavaScript API programming patterns
- How to specify an add-in's settings and capabilities in the manifest file
- Troubleshooting your add-in
- Publishing your add-in

For the best foundation for these common features and application-specific implementations, review the following documentation.

Plan
Learn the best practices and system requirements for Office Add-ins.



Develop
Learn the APIs and patterns to develop Office Add-ins.

Test and debug

Learn how to test and debug Office Add-ins.

Publish

Learn how to deploy and publish Office Add-ins.

Reference

View the reference documentation for the Office JavaScript APIs, the add-ins manifest, error code lists, and more.

See also

- Office Dev Center
- Office Add-ins platform overview
- Office client application and platform availability for Office Add-ins

Best practices for developing Office Add-ins

07/30/2025

Great add-ins provide unique, compelling functionality that extend Office apps in visually appealing ways. To build a successful add-in, you'll need to create an engaging first-time user experience, design a polished UI, and optimize performance. Follow the best practices in this article to help your users complete tasks quickly and efficiently.

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).

Provide clear value

Build add-ins that help users complete tasks quickly and efficiently. Focus on scenarios that make sense for Office apps, such as:

- Make core authoring tasks faster and easier with fewer interruptions.
- Enable new scenarios within Office.
- Embed complementary services within Office apps.
- Improve the Office experience to enhance productivity.

Make sure users understand your add-in's value immediately by creating an engaging first-run experience.

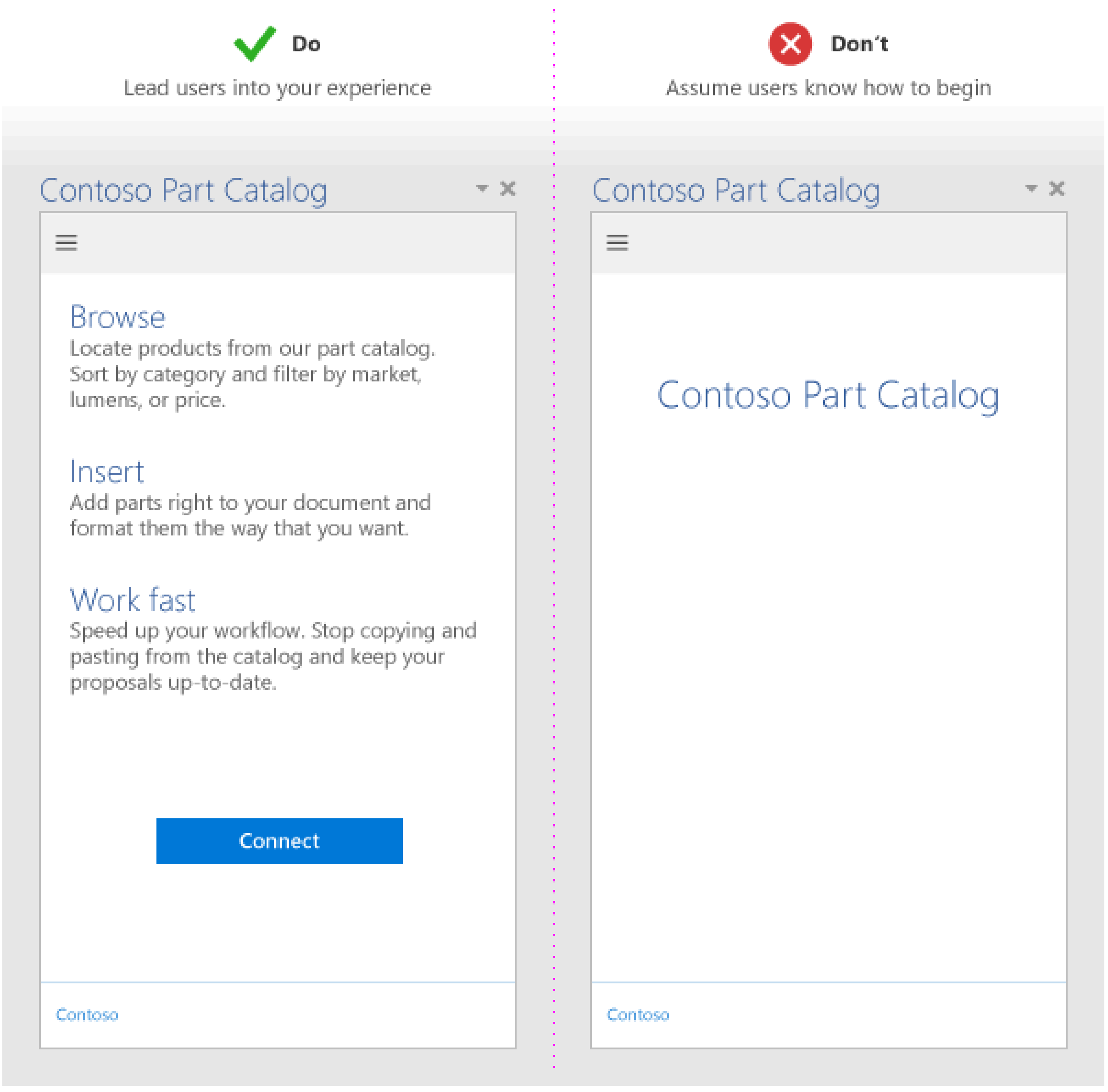
When you're ready to promote your add-in, learn how to create an effective AppSource listing.

- Make your add-in's benefits clear in the title and description. Don't rely only on your brand to communicate what your add-in does.
- Ensure your add-in provides sufficient value to justify users' investment. It shouldn't be just a simple utility or have limited scope.
- If your add-in targets larger organizations and enterprises, several AppSource requirements differ from those of a general commercial marketplace add-in. For more information, see the submission FAQ.

Create an engaging first-run experience

New users are still deciding whether to use or abandon your add-in after downloading it from the store. Here's how to win them over.

- **Make the next steps clear.** Use videos, placemats, paging panels, or other resources to guide users through your add-in.
- **Lead with value, not registration.** Reinforce your add-in's value proposition when it launches rather than immediately asking users to sign in.
- **Provide helpful guidance.** Include teaching UI to guide users and make the experience feel personal.


Do

Lead users into your experience

Contoso Part Catalog

Browse
Locate products from our part catalog. Sort by category and filter by market, lumens, or price.

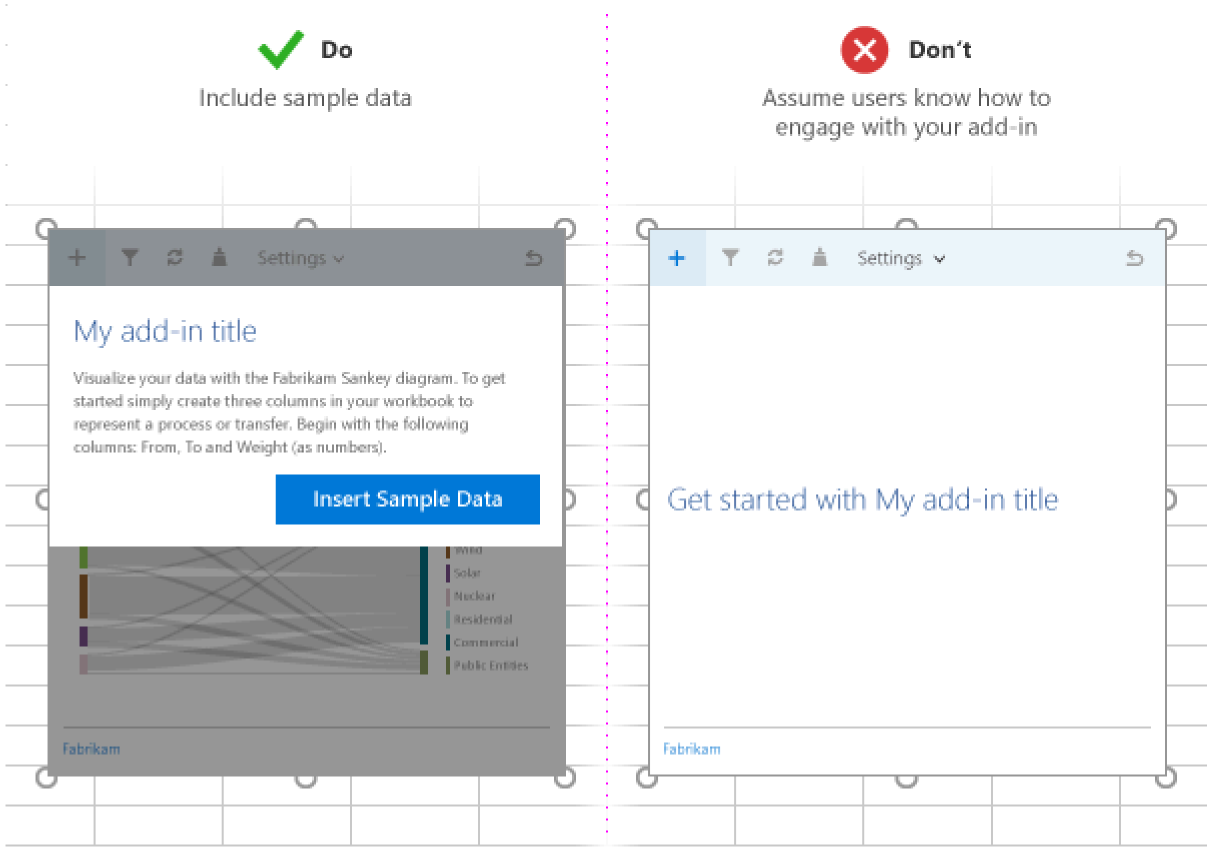
Insert
Add parts right to your document and format them the way that you want.

Work fast
Speed up your workflow. Stop copying and pasting from the catalog and keep your proposals up-to-date.

Don't

Assume users know how to begin

- **Show users what to expect.** If your content add-in binds to data in the user's document, include sample data or a template to show users the expected data format.



Visualize your data with the Fabrikam Sankey diagram. To get started simply create three columns in your workbook to represent a process or transfer. Begin with the following columns: From, To and Weight (as numbers).

- **Offer free trials.** If your add-in requires a subscription, make some functionality available without one.
- **Simplify sign-up.** Prefill information like email and display name, and skip email verifications when possible.
- **Avoid pop-ups.** If you must use them, guide users on how to enable your pop-up window.

For patterns you can apply when developing your first-run experience, see UX design patterns for Office Add-ins.

Use add-in commands

Provide relevant UI entry points for your add-in by using add-in commands. These commands help users discover and access your add-in's functionality directly from the Office ribbon. For details and design best practices, see add-in commands.

Apply UX design principles

Follow these key principles to create add-ins that feel native to Office:

- **Match the Office experience.** Ensure your add-in's look, feel, and functionality complement the Office experience. See Design the UI of Office Add-ins.

- **Prioritize content over chrome.** Avoid unnecessary UI elements that don't add value to the user experience.
- **Keep users in control.** Make sure users understand important decisions and can easily reverse actions your add-in performs.
- **Use branding thoughtfully.** Inspire trust and help orient users, but don't overwhelm or advertise to them.
- **Minimize scrolling.** Optimize for 1366 x 768 resolution.
- **Use licensed images only.** Avoid legal and branding issues that come from unlicensed images.
- **Write clearly.** Use clear and simple language in your add-in.
- **Design for accessibility.** Make your add-in easy for all users to interact with and accommodate assistive technologies like screen readers. See our accessibility guidelines.
- **Support all platforms and input methods.** Design for mouse/keyboard and touch. Ensure your UI responds well to different form factors.

Optimize for touch

Touch support is essential for modern Office add-ins.

- **Detect touch support.** Use the Context.touchEnabled property to detect whether the Office app your add-in runs on is touch-enabled.

  > **Note**
  >
  > This property isn't supported in Outlook.

- **Size controls appropriately.** Make sure all controls work well with touch interaction. For example, buttons need adequate touch targets, and input boxes should be large enough for users to enter text easily.
- **Don't rely on hover or right-click.** These input methods aren't available on touch devices.
- **Support both orientations.** Ensure your add-in works in both portrait and landscape modes. Remember that on touch devices, the soft keyboard might hide part of your add-in.
- **Test on real devices.** Use sideloading to test your add-in on actual touch devices.

Optimize and monitor add-in performance

Performance directly impacts user satisfaction. Follow these guidelines to keep your add-in fast and responsive:

- **Aim for quick loading.** Your add-in should load in 500 ms or less to create the perception of fast UI responses.
- **Respond quickly to interactions.** All user interactions should respond in under one second.
- **Show progress for long operations.** Provide loading indicators for operations that take time.
- **Use a CDN.** Host images, resources, and common libraries on a content delivery network (CDN). Load as much as possible from one place.
- **Follow web optimization best practices.** In production, use only minified versions of libraries. Load only the resources you need and optimize how they're loaded.
- **Provide feedback for longer operations.** When operations take time to execute, give users feedback based on the thresholds in the following table. For more information, see Resource limits and performance optimization for Office Add-ins.

Expand table

Interaction class	Target	Upper bound	Human perception
Instant	<=50 ms	100 ms	No noticeable delay.
Fast	50-100 ms	200 ms	Minimally noticeable delay. No feedback necessary.
Typical	100-300 ms	500 ms	Quick, but too slow to be described as fast. No feedback necessary.
Responsive	300-500 ms	1 second	Not fast, but still feels responsive. No feedback necessary.
Continuous	>500 ms	5 seconds	Medium wait, no longer feels responsive. Might need feedback.
Captive	>500 ms	10 seconds	Long, but not long enough to do something else. Might need feedback.
Extended	>500 ms	>10 seconds	Long enough to do something else while waiting. Might need feedback.

Interaction class	Target	Upper bound	Human perception
Long running	>5 seconds	>1 minute	Users will certainly do something else.

- **Monitor your service.** Use telemetry to monitor service health and user success.
- **Minimize data exchanges.** Reduce data exchanges between your add-in and the Office document. For more information, see Avoid using the context.sync method in loops.

Publish and market your add-in

Ready to share your add-in with the world? Here's how to get started.

- **Create a Partner Center account.** This process can take time, so if you plan to publish to AppSource, start early. See Partner Center account.
- **Create an effective AppSource listing.** Follow these tips:
  - Use succinct, descriptive titles (128 characters or fewer).
  - Write short, compelling descriptions that answer "What problem does this add-in solve?"
  - Convey your add-in's value proposition clearly in the title and description. Don't rely only on your brand.

  Learn more about creating effective AppSource listings.
- **Publish to AppSource.** Follow the AppSource prepublish checklist and submission guide. Make sure to:
  - Test your add-in thoroughly on all supported operating systems, browsers, and devices.
  - Provide detailed testing instructions and resources for certification reviewers.
- **Create a website.** Help users discover your add-in outside of AppSource.
- **Promote your add-in** from your website. See how to promote your add-in.

> ⓘ **Important**
>
> If your add-in targets larger organizations and enterprises, several AppSource requirements differ from those of a general commercial marketplace add-in. For more information, see the submission FAQ.

Support older Microsoft webviews and Office versions (recommended but not required)

See Support older Microsoft webviews and Office versions.

See also

- Office Add-ins platform overview
- Learn about the Microsoft 365 Developer Program