

Develop Office Add-ins

Article • 05/19/2025

Tip

Please review [Office Add-ins platform overview](#) before reading this article.

All Office Add-ins are built upon the Office Add-ins platform. For any add-in you build, you'll need to understand important concepts like application and platform availability, Office JavaScript API programming patterns, how to specify an add-in's settings and capabilities in the manifest file, how to design the UI and user experience, and more. Core development concepts like these are covered here in the **Development lifecycle > Develop** section of the documentation. Review the information here before exploring the application-specific documentation that corresponds to the add-in you're building (for example, [Excel](#)).

Create an Office Add-in

You can create an Office Add-in by using the [Yeoman generator for Office Add-ins](#), Visual Studio, Microsoft 365 Agents Toolkit, or the [Office Add-ins Development Kit](#).

Yeoman generator

The Yeoman generator for Office Add-ins can be used to create a Node.js Office Add-in project that can be managed with Visual Studio Code or any other editor. The generator can create Office Add-ins for any of the following:

- Excel
- OneNote
- Outlook
- PowerPoint
- Project
- Word
- Excel custom functions

Create your project using HTML, CSS and JavaScript (or TypeScript), or using React. If you choose React, you can choose between JavaScript and Typescript as well. For more information about creating add-ins with the generator, see [Yeoman generator for Office Add-ins](#).

Visual Studio

Visual Studio can be used to create Office Add-ins for Excel, Outlook, Word, and PowerPoint. An Office Add-in project gets created as part of a Visual Studio solution and uses HTML, CSS, and JavaScript. For more information about creating add-ins with Visual Studio, see [Develop Office Add-ins with Visual Studio](#).

Agents Toolkit

The Agents Toolkit can be used to create almost any kind of Microsoft 365 extension. For details about creating an add-in, see [Create Office Add-in projects with Microsoft 365 Agents Toolkit](#).

Office Add-ins Development Kit (preview)

The Office Add-ins Development Kit is an extension for Visual Studio Code. It lets you create new add-in projects and load samples directly from the IDE. Download the extension from the [Visual Studio Marketplace](#) or learn more in the article [Create Office Add-in projects using Office Add-ins Development Kit for Visual Studio Code](#).

Understand the two parts of an Office Add-in

An Office Add-in consists of two parts.

- The add-in manifest that defines the settings and capabilities of the add-in.
- The web application that defines the UI and functionality of add-in components such as task panes, content add-ins, and dialog boxes.

The web application uses the Office JavaScript API to interact with content in the Office document where the add-in is running. Your add-in can also do other things that web applications typically do, like call external web services, facilitate user authentication, and more.

Define an add-in's settings and capabilities

An Office Add-in's manifest defines the settings and capabilities of the add-in. You'll configure the manifest to specify things such as:

- Metadata that describes the add-in (for example, ID, version, description, display name, default locale).
- Office applications where the add-in will run.
- Permissions that the add-in requires.

- How the add-in integrates with Office, including any custom UI that the add-in creates (for example, a custom tab or custom ribbon buttons).
- Location of images that the add-in uses for branding and command iconography.
- Dimensions of the add-in (for example, dimensions for content add-ins, requested height for Outlook add-ins).
- Rules that specify when the add-in activates in the context of a message or appointment (for Outlook add-ins only).
- Keyboard shortcuts (for Excel and Word add-ins only).

For detailed information about the manifest, see [Office Add-ins manifest](#).

Interact with content in an Office document

An Office Add-in can use the Office JavaScript APIs to interact with content in the Office document where the add-in is running.

Access the Office JavaScript API library

The Office JavaScript API library can be accessed via the Office JS content delivery network (CDN) at: `https://appsforoffice.microsoft.com/lib/1/hosted/office.js`. To use Office JavaScript APIs within any of your add-in's web pages, you must reference the CDN in a `<script>` tag in the `<head>` tag of the page.

HTML

```
<head>
...
<script src="https://appsforoffice.microsoft.com/lib/1/hosted/office.js"
type="text/javascript"></script>
</head>
```

ⓘ Note

To use preview APIs, reference the preview version of the Office JavaScript API library on the CDN: `https://appsforoffice.microsoft.com/lib/beta/hosted/office.js`.

For more information about accessing the Office JavaScript API library, including how to get IntelliSense, see [Referencing the Office JavaScript API library from its content delivery network \(CDN\)](#).

API models

The Office JavaScript API includes two distinct models:

- **Application-specific** APIs provide strongly-typed objects that can be used to interact with objects that are native to a specific Office application. For example, you can use the Excel JavaScript APIs to access worksheets, ranges, tables, charts, and more. Application-specific APIs are currently available for the following Office applications.
 - [Excel](#)
 - [OneNote](#)
 - [PowerPoint](#)
 - [Word](#)

This API model uses [promises](#) and allows you to specify multiple operations in each request you send to the Office application. Batching operations in this manner can significantly improve add-in performance in Office applications on the web. Application-specific APIs were introduced with Office 2016.

ⓘ **Note**

There's also an application-specific API for [Visio](#), but you can use it only in SharePoint Online pages to interact with Visio diagrams that have been embedded in the page. Office Web Add-ins are not supported in Visio.

See [Using the application-specific API model](#) to learn more about this API model.

- **Common** APIs can be used to access features such as UI, dialogs, and client settings that are common across multiple types of Office applications. This API model uses [callbacks](#), which allow you to specify only one operation in each request sent to the Office application. Common APIs were introduced with Office 2013 and can be used to interact with any supported Office applications. For details about the Common API object model, which includes APIs for interacting with Outlook, PowerPoint, and Project, see [Common JavaScript API object model](#).

ⓘ **Note**

Custom functions without a [shared runtime](#) run in a [JavaScript-only runtime](#) that prioritizes execution of calculations. These functions use a slightly different programming model.

API requirement sets

[Requirement sets](#) are named groups of API members. Requirement sets can be specific to Office applications, such as the `ExcelApi 1.7` requirement set (a set of APIs that can only be used in Excel), or common to multiple applications, such as the `DialogApi 1.1` requirement set (a set of APIs that can be used in any Office application that supports the Dialog API).

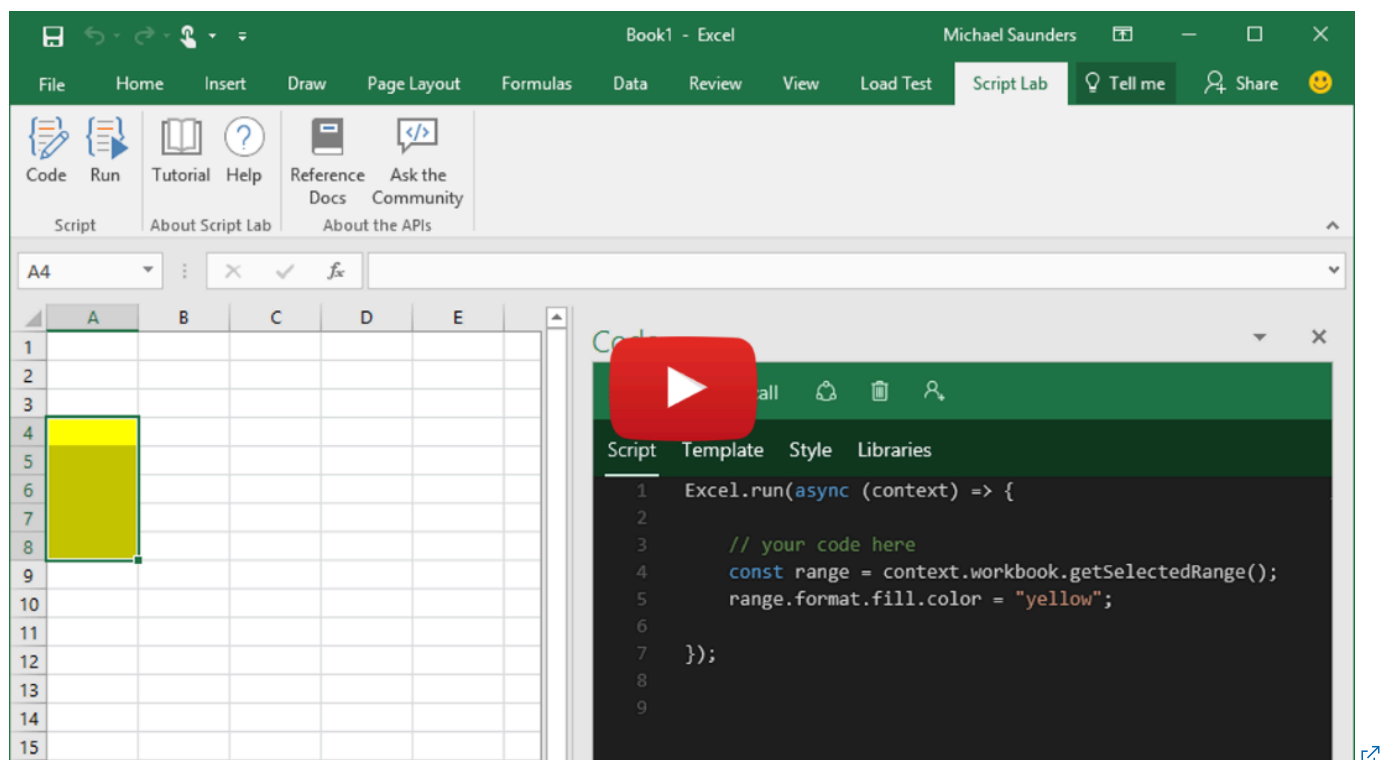
Your add-in can use requirement sets to determine whether the Office application supports the API members that it needs to use. For more information about this, see [Specify Office applications and API requirements](#).

Requirement set support varies by Office application, version, and platform. For detailed information about the platforms, requirement sets, and Common APIs that each Office application supports, see [Office client application and platform availability for Office Add-ins](#).

Explore APIs with Script Lab

Script Lab is an add-in that enables you to explore the Office JavaScript API and run code snippets while you're working in an Office program such as Excel or Word. It's available for free via AppSource and is a useful tool to include in your development toolkit as you prototype and verify the functionality you want in your add-in. In Script Lab, you can access a library of built-in samples to quickly try out APIs or even use a sample as the starting point for your own code.

The following one-minute video shows Script Lab in action.



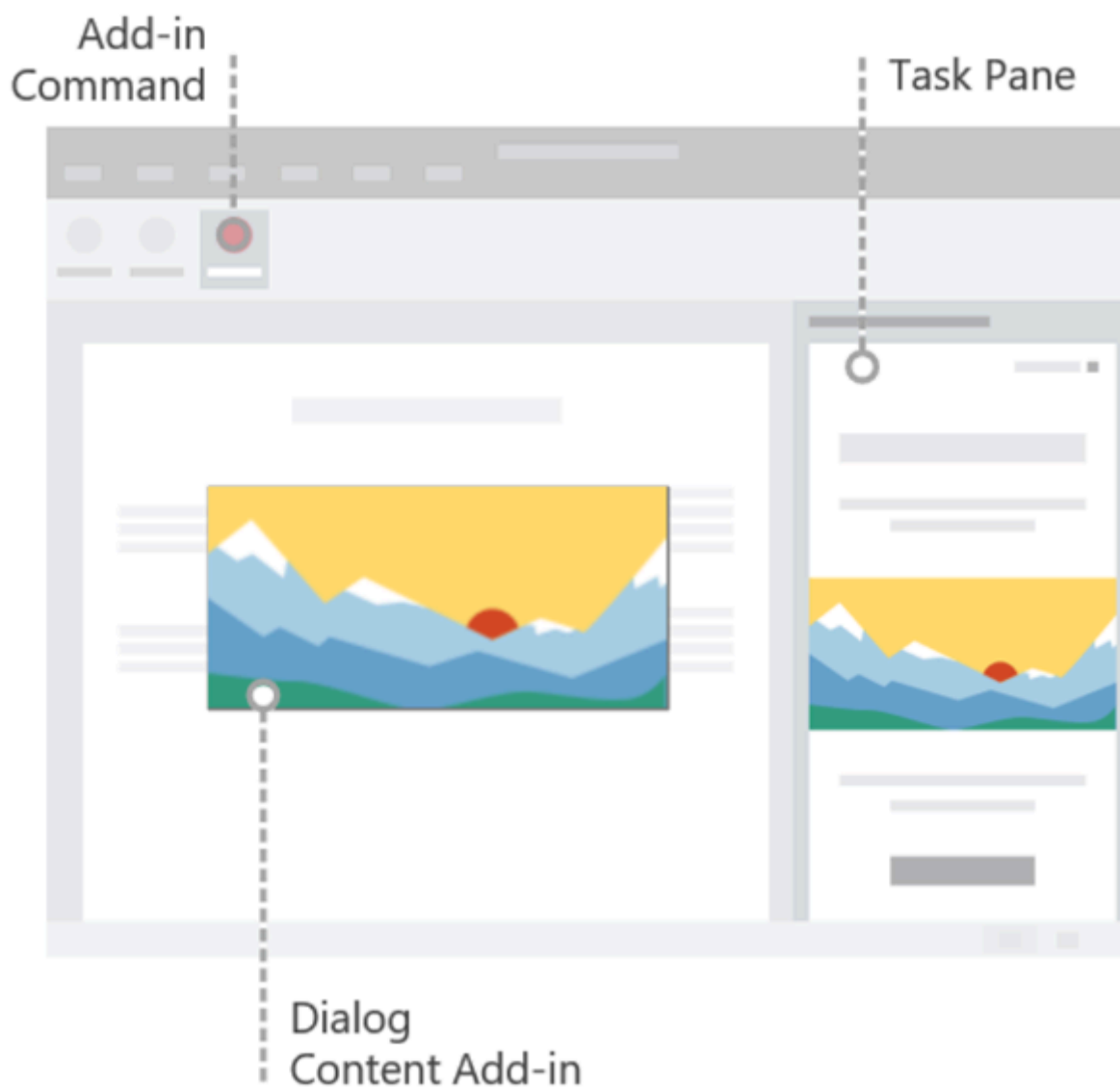
For more information about Script Lab, see [Explore Office JavaScript APIs using Script Lab](#).

Extend the Office UI

An Office Add-in can extend the Office UI by using add-in commands and HTML containers such as task panes, content add-ins, or dialog boxes.

- [Add-in commands](#) can be used to add a custom tab, custom buttons and menus to the default ribbon in Office, or to extend the default context menu that appears when users right-click (or select and hold) text in an Office document or an object in Excel. When users select an add-in command, they initiate the task that the add-in command specifies, such as running JavaScript code, opening a task pane, or launching a dialog box.
- HTML containers like [task panes](#), [content add-ins](#), and [dialog boxes](#) can be used to display custom UI and expose additional functionality within an Office application. The content and functionality of each task pane, content add-in, or dialog box derives from a web page that you specify. Those web pages can use the Office JavaScript API to interact with content in the Office document where the add-in is running, and can also do other things that web pages typically do, like call external web services, facilitate user authentication, and more.

The following image shows an add-in command on the ribbon, a task pane to the right of the document, and a dialog box or content add-in over the document.



For more information about extending the Office UI and designing the add-in's UX, see [Office UI elements for Office Add-ins](#).

Next steps

This article has outlined the different ways to create Office Add-ins, introduced the ways that an add-in can extend the Office UI, described the API sets, and introduced Script Lab as a valuable tool for exploring Office JavaScript APIs and prototyping add-in functionality. Now that you've explored this introductory information, consider continuing your Office Add-ins journey along the following paths.

Create an Office Add-in

You can quickly create a basic add-in for Excel, OneNote, Outlook, PowerPoint, Project, or Word by completing a [5-minute quick start](#). If you've previously completed a quick start and want to create a slightly more complex add-in, you should try the [tutorial](#).


Learn more

Learn more about developing, testing, and publishing Office Add-ins by exploring this documentation.

Tip

For any add-in that you build, you'll use information in the [Development lifecycle](#) section of this documentation, along with information in the application-specific section that corresponds to the type of add-in you're building (for example, [Excel](#)).

See also

- [Office Add-ins platform overview](#)
- [Learn about the Microsoft 365 Developer Program](#) 
- [Design Office Add-ins](#)
- [Test and debug Office Add-ins](#)
- [Publish Office Add-ins](#)