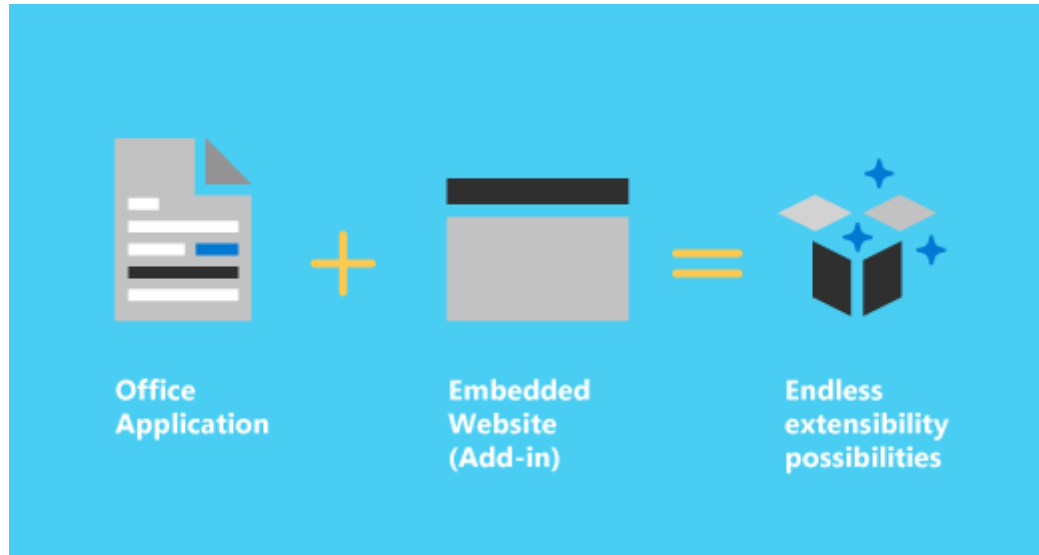


Office Add-ins platform overview

Article • 02/19/2025

You can use the Office Add-ins platform to build solutions that extend Office applications and interact with content in Office documents. With Office Add-ins, you can use familiar web technologies such as HTML, CSS, and JavaScript to extend and interact with Outlook, Excel, Word, PowerPoint, OneNote, and Project. Your solution can run in Office across multiple platforms, including Windows, Mac, iPad, and in a browser.



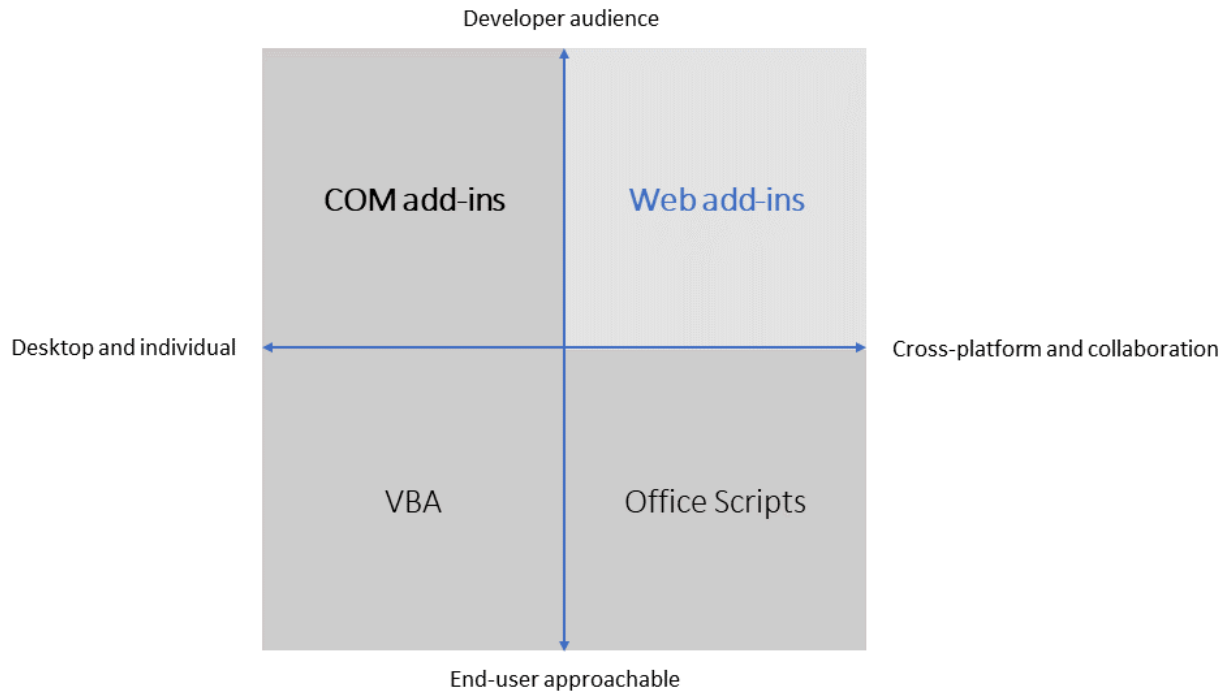
Office Add-ins can do almost anything a webpage can do inside a browser. Use the Office Add-ins platform to:

- **Add new functionality to Office clients** - Bring external data into Office, automate Office documents, expose functionality from Microsoft and others in Office clients, and more. For example, use Microsoft Graph API to connect to data that drives productivity.
- **Create new rich, interactive objects that can be embedded in Office documents** - Embed maps, charts, and interactive visualizations that users can add to their own Excel spreadsheets and PowerPoint presentations.

How are Office Add-ins different from COM and VSTO add-ins?

COM and VSTO add-ins are earlier Office integration solutions that run only in Office on Windows. Unlike COM and VSTO add-ins, Office Add-ins are web add-ins: the application (for example, Excel), reads the add-in manifest and connects the add-in's custom ribbon buttons and menu commands in the UI. When needed, it loads the add-

in's JavaScript and HTML code, which runs in the context of a browser or webview control in a sandbox.



Office Add-ins provide the following advantages over add-ins built using VBA, COM, or VSTO.

- Cross-platform support: Office Add-ins run in Office on the web, Windows, Mac, and iPad.
- Centralized deployment and distribution: Admins can deploy Office Add-ins centrally across an organization.
- Easy access via AppSource: You can make your solution available to a broad audience by submitting it to AppSource.
- Based on standard web technology: You can use any library you like to build Office Add-ins.

Why use Office Add-ins?

Cross platform
(Web, Windows,
Mac, iPad)



Centralized
deployment
and distribution




Easy access
via AppSource



Built on
standard web
technologies



Important

COM and VSTO add-ins aren't supported in the [new Outlook on Windows](#) . These add-ins are still supported in the classic Outlook on Windows desktop client. To learn more, see [Develop Outlook add-ins for new Outlook on Windows](#).

Components of an Office Add-in

An Office Add-in includes two basic components: a manifest file and your own web application. The manifest defines various settings, including how your add-in integrates with Office clients. Your web application needs to be hosted on a web server, or web hosting service, such as Microsoft Azure.

Manifest

The manifest specifies settings and capabilities of the add-in, such as:

- The add-in's display name, description, ID, version, and default locale.
- How the add-in integrates with Office.
- The permission level and data access requirements for the add-in.

Web app

The most basic Office Add-in consists of a static HTML page that is displayed inside an Office application, but that doesn't interact with either the Office document or any other Internet resource. However, to create an experience that interacts with Office documents

or allows the user to interact with online resources from an Office client application, you can use any technologies, both client and server side, that your hosting provider supports (such as ASP.NET, PHP, or Node.js). To interact with Office clients and documents, you use the Office.js JavaScript APIs.



Extending and interacting with Office clients

Office Add-ins can do the following within an Office client application.

- Extend functionality (any Office application)
- Create new objects (Excel or PowerPoint)

Extend Office functionality

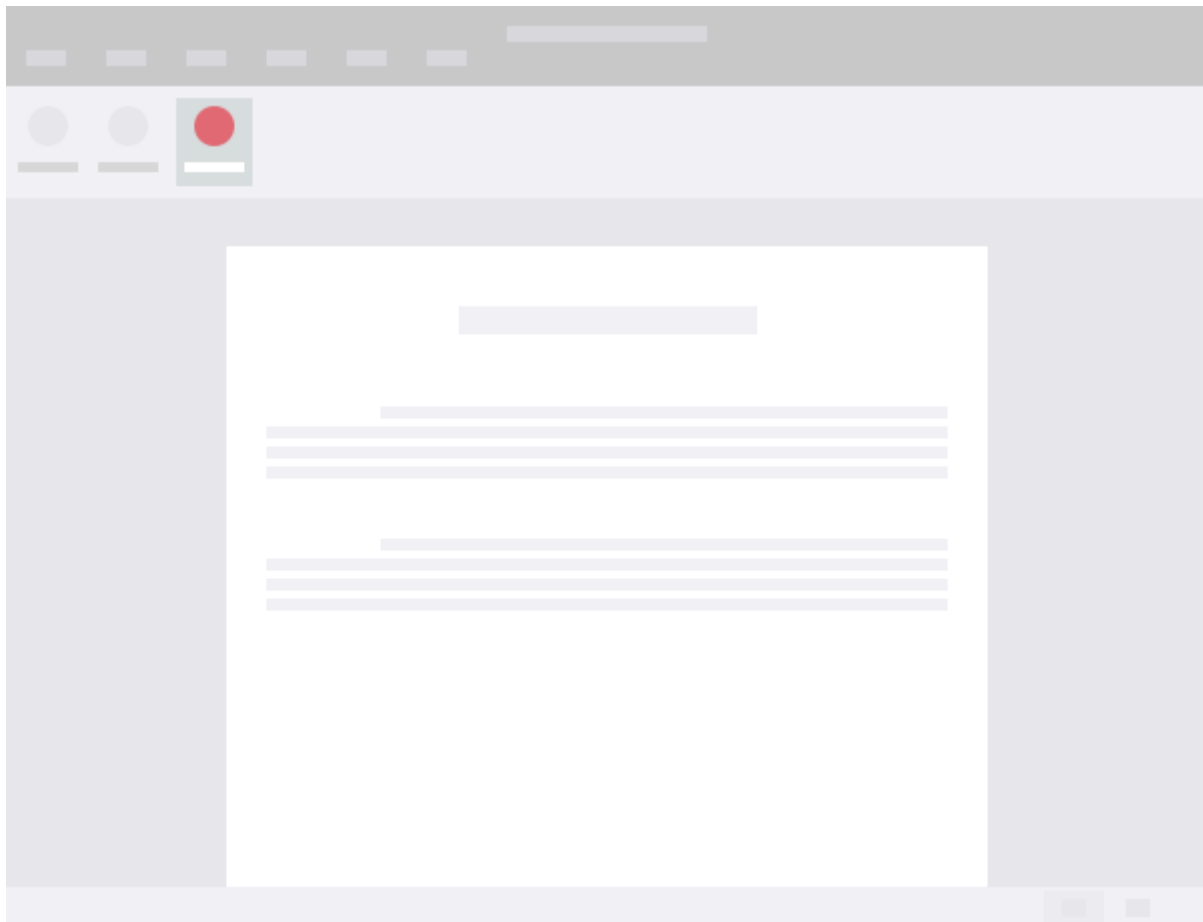
You can add new functionality to Office applications via the following:

- Custom ribbon buttons and menu commands (collectively called "add-in commands").
- Insertable task panes.

Custom UI and task panes are specified in the add-in manifest.

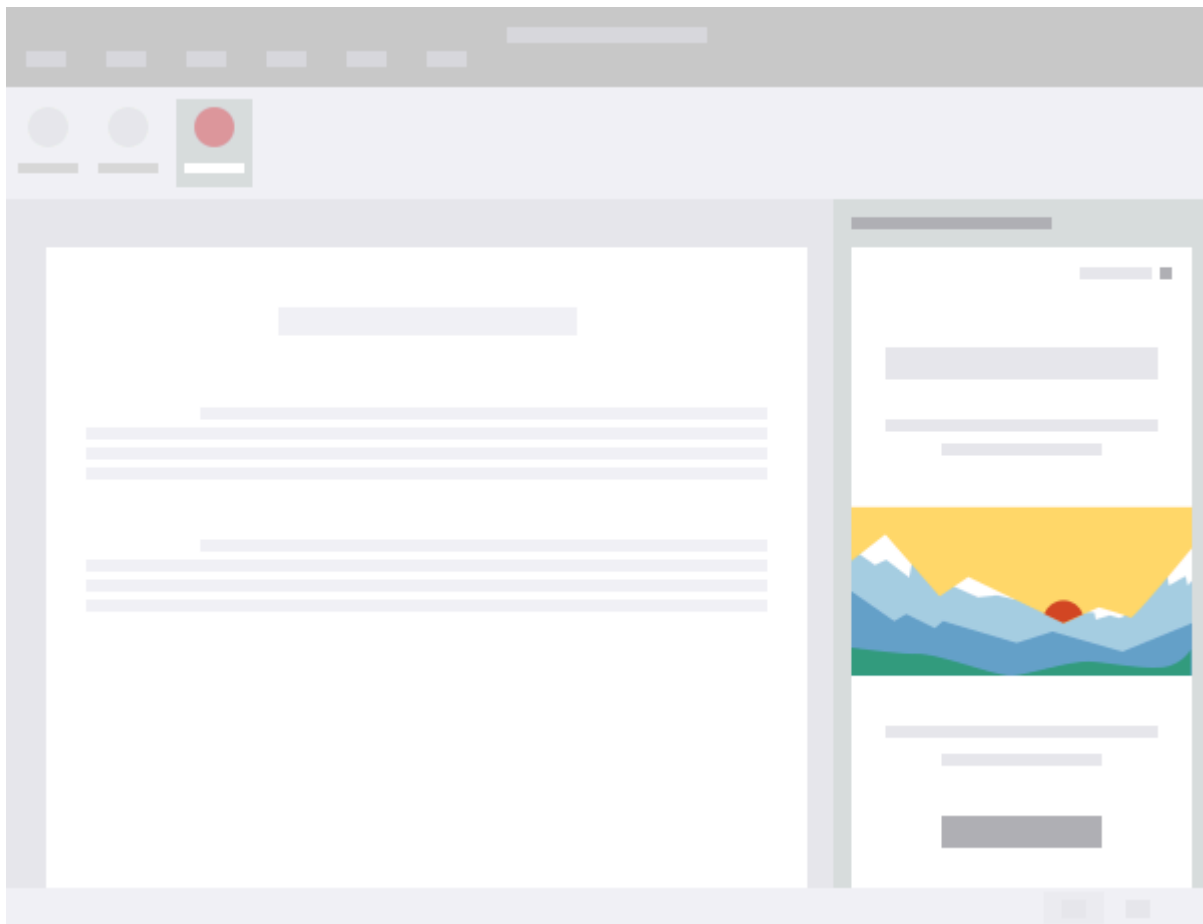
Custom buttons and menu commands

You can add custom ribbon buttons and menu items to the ribbon in Office on the web and on Windows. This makes it easy for users to access your add-in directly from their Office application. Custom buttons and menu items can launch different actions such as showing a task pane with custom HTML or executing a JavaScript function.



Task panes

You can use task panes in addition to add-in commands to enable users to interact with your solution. Clients that don't support add-in commands (Office on iPad) run your add-in as a task pane. In Excel, Word, and PowerPoint, users launch task pane add-ins via the **Home > Add-ins** button. In Outlook, users launch task pane add-ins via the add-in button or via the **All Apps** button on the ribbon.



Extend Outlook functionality

Users can run Outlook add-ins when they view, reply, or create emails, meeting requests, meeting responses, meeting cancellations, or appointments. Outlook add-ins can do the following:

- Extend the Office app ribbon.
- Display contextually next to an Outlook item when you're viewing or composing it.
- Perform a task when a specific event occurs, such as when a user creates a new message.

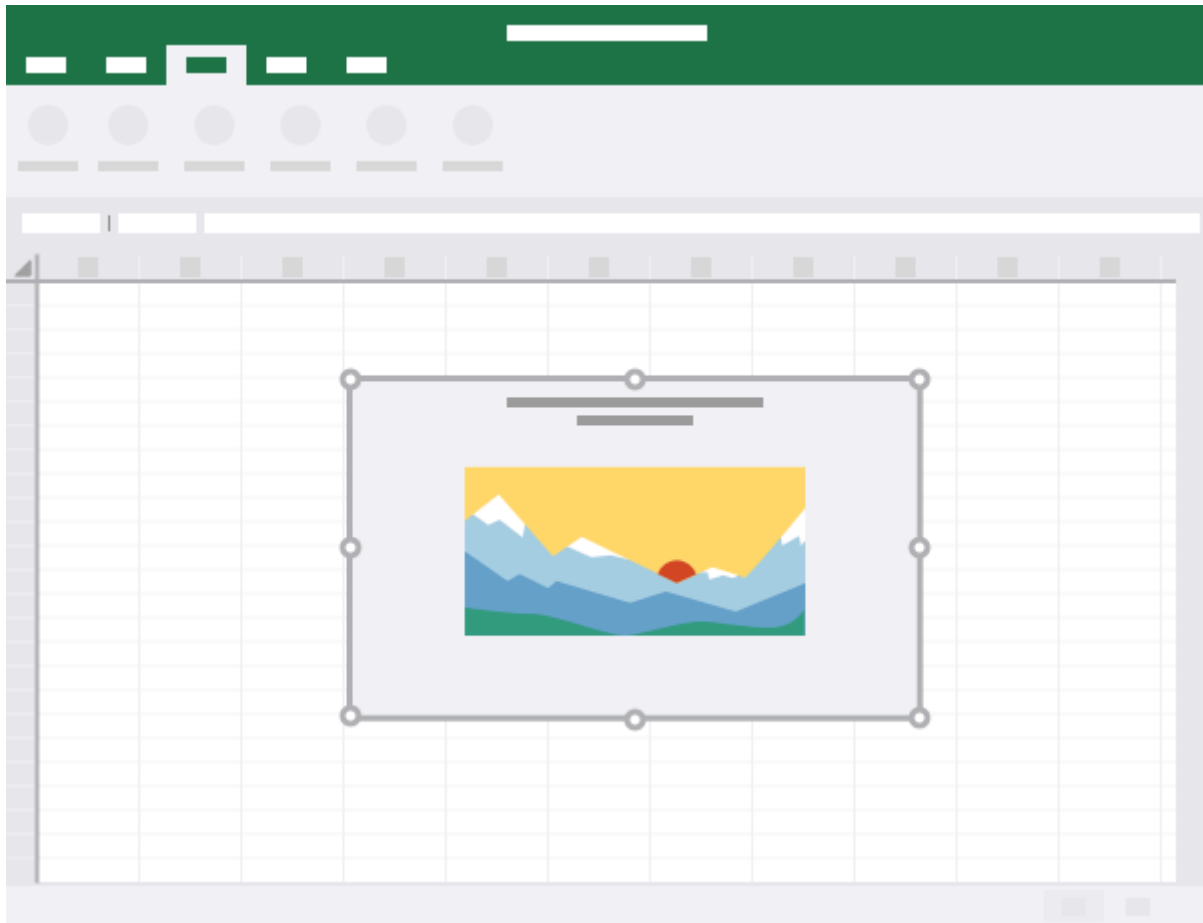
📌 Note

Add-ins that interact with the user's calendar, meetings, or appointments are available only if the user opens the calendar in Outlook, not Teams.

For an overview of Outlook add-ins, see [Outlook add-ins overview](#).

Create new objects in Office documents

You can embed web-based objects called content add-ins within Excel and PowerPoint documents. With content add-ins, you can integrate rich, web-based data visualizations, media (such as a YouTube video player or a picture gallery), and other external content.



Office JavaScript APIs

The Office JavaScript APIs contain objects and members for building add-ins and interacting with Office content and web services. There's a common object model that's shared by Excel, Outlook, Word, PowerPoint, OneNote, and Project. There are also more extensive application-specific object models for Excel, OneNote, PowerPoint, and Word. These APIs provide access to well-known objects such as paragraphs and workbooks, which makes it easier to create an add-in for a specific application.

Code samples

Learn how to build the simplest Office Add-in with only a manifest, HTML web page, and a logo. The following samples will help you get started in the Office application you're interested in.

- [Excel "Hello world" add-in](#) [↗](#)
- [Outlook "Hello world" add-in](#) [↗](#)

- [PowerPoint "Hello world" add-in](#) ↗
- [Word "Hello world" add-in](#) ↗

Next steps

For a more detailed introduction to developing Office Add-ins, see [Develop Office Add-ins](#).

See also

- [Core concepts for Office Add-ins](#)
- [Develop Office Add-ins](#)
- [Design Office Add-ins](#)
- [Test and debug Office Add-ins](#)
- [Publish Office Add-ins](#)
- [Learn about the Microsoft 365 Developer Program](#) ↗

Beginner's guide

06/27/2025

Want to get started building your own cross-platform Office extensions? The following steps show you what to read first, what tools to install, and recommended tutorials to complete.

ⓘ Note


If you're experienced in creating VSTO add-ins for Office, we recommend that you immediately turn to [VSTO add-in developer's guide](#), which is a superset of the information in this article.

Step 0: Prerequisites

- Office Add-ins are essentially web applications embedded in Office. So, you should first have a basic understanding of web applications and how they are hosted on the web. There's an enormous amount of information about this on the Internet, in books, and in online courses. A good way to start if you have no prior knowledge of web applications at all is to search for "What is a web app?" on Bing.
- The primary programming language you'll use in creating Office Add-ins is JavaScript or TypeScript. You can think of TypeScript as a strongly-typed version of JavaScript. If you are not familiar with either of these languages, but you have experience with VBA, VB.Net, C#, you'll probably find TypeScript easier to learn. Again, there's a wealth of information about these languages on the Internet, in books, and in online courses.

Step 1: Begin with fundamentals

We know you're eager to start coding, but there are some things about Office Add-ins that you should read before you open your IDE or code editor.

- [Office Add-ins platform overview](#): Find out what Office Web Add-ins are and how they differ from older ways of extending Office, such as VSTO add-ins.
- [Develop Office Add-ins](#): Get an overview of Office Add-in development and lifecycle including tooling, creating an add-in UI, and using the JavaScript APIs to interact with the Office document.
- ["Hello world" samples](#) : Learn how to build the simplest Office Add-in with only a manifest, HTML web page, and a logo. These samples will help you understand the fundamental parts of an Office Add-in.

There are a lot of links in those articles, but if you're a beginner with Office Add-ins, we recommend that you come back here when you've read them and continue with the next section.

Step 2: Explore and try out existing samples

You've got the big picture now, so dive in by installing our [Script Lab add-in](#) to try out code samples in the various Office applications. The samples available in Script Lab show how to use many of the Office JavaScript APIs.

Step 3: Install tools and create your first add-in

Next, create an add-in using one of our quick starts. For the purpose of learning the platform, we recommend the [Excel quick start](#).

Step 4: Code

You can't learn to drive by reading the owner's manual, so start coding with this [Excel tutorial](#). You'll be using the Office JavaScript library and some JSON or XML in the add-in's manifest. There's no need to memorize anything, because you'll be getting more background about both in a later steps.

Step 5: Understand the JavaScript library

For an overview of the Office JavaScript library, see [Develop Office Add-ins](#).

Then return to Script Lab and use it like a playground: make your own code changes to the local copy of any samples you try and see how the results are affected.

Step 6: Understand the manifest

Get an understanding of the purposes of the add-in manifest and an introduction to its XML markup or JSON in [Office Add-ins manifest](#).

Step 7: Create a Partner Center account

If you plan to [publish your add-in to AppSource](#), create a [Partner Center account](#). This could take some time. Get this process going as soon as possible to avoid release delays.

Next Steps

Congratulations on finishing the beginner's learning path for Office Add-ins! Here are some suggestions for further exploration of our documentation:

- Tutorials or quick starts for other Office applications:
 - [OneNote quick start](#)
 - [Outlook tutorial](#)
 - [PowerPoint tutorial](#)
 - [Project quick start](#)
 - [Word tutorial](#)
- Scenarios and other code samples:
 - [Excel: Create a spreadsheet from your web page and embed your add-in](#)
 - [Outlook: Report spam or phishing emails](#) ↗
 - [Word: Import a document template](#)
 - [Word: Manage citations](#)
 - [Office Add-in code samples](#)
- Other important subjects:
 - [Develop Office Add-ins](#)
 - [Best practices for developing Office Add-ins](#)
 - [Design Office Add-ins](#)
 - [Test and debug Office Add-ins](#)
 - [Deploy and publish Office Add-ins](#)
 - [Resources](#)
 - [Learn about the Microsoft 365 Developer Program](#) ↗

Set up your development environment

Article • 05/19/2025

This guide helps you set up tools so you can create Office Add-ins by following our quick starts or tutorials. If you already have these installed, you're ready to begin a quick start, such as this [Excel React quick start](#).


Get Microsoft 365

You need a Microsoft 365 account. You might qualify for a Microsoft 365 E5 developer subscription, which includes all Office apps, 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](#).

Install the environment

There are three kinds of development environments to choose from. The scaffolding of Office Add-in projects that is created in the three environments is different, so if multiple people will be working on an add-in project, they must all use the same environment.

- **Node.js environment:** Recommended. In this environment, your tools are installed and run at a command line. The server-side of the web application part of the add-in is written in JavaScript or TypeScript and is hosted in a Node.js runtime. There are many helpful add-in development tools in this environment, such as an Office linter and a bundler/task-runner called webpack. The project creation and scaffolding tool is a command line tool called the Office Yeoman Generator (also called "Yo Office"), though you can still use the Visual Studio Code extensions mentioned in the next option.
- **Visual Studio Code:** Choose this environment if you use Visual Studio Code and would prefer to create projects from extensions rather than command line tools. The project creation and scaffolding tools are Microsoft 365 Agents Toolkit or Office Add-ins Development Kit extensions.
- **Visual Studio environment:** Choose this environment only if your development computer is Windows, and you want to develop the server-side of the add-in with a .NET based language and framework, such as ASP.NET. The add-in project templates in Visual Studio aren't updated as frequently as those in the Node.js environment. More information later on the **Visual Studio environment** tab.

 **Note**

Visual Studio for Mac doesn't include the project scaffolding templates for Office Add-ins, so if your development computer is a Mac, you should work with the Node.js environment.

Select the tab for the environment you choose.

Node.js environment

The main tools to be installed are:

- Node.js
- npm
- A code editor of your choice
- Office Yeoman Generator (Yo Office)
- The Office JavaScript linter

This guide assumes that you know how to use a command-line tool.

Install Node.js and npm

Node.js is a JavaScript runtime you use to develop modern Office Add-ins.

Install Node.js by [downloading the latest recommended version from their website](#) .

Follow the installation instructions for your operating system.

npm is an open source software registry from which to download the packages used in developing Office Add-ins. It's usually installed automatically when you install Node.js. To check if you already have npm installed and see the installed version, run the following in the command line.

command line

```
npm -v
```

If, for any reason, you want to install it manually, run the following in the command line.

command line

```
npm install npm -g
```



Tip

You may wish to use a Node version manager to allow you to switch between multiple versions of Node.js and npm, but this isn't strictly necessary. For details on how to do this, [see npm's instructions](#).

Install a code editor

You can use any code editor or IDE that supports client-side development to build your web part, such as:

- [Visual Studio Code](#) (recommended)
- [Atom](#)
- [Webstorm](#)

Install the Yeoman generator — Yo Office

The project creation and scaffolding tool is [Yeoman generator for Office Add-ins](#), affectionately known as **Yo Office**. You need to install the latest version of [Yeoman](#) and Yo Office. To install these tools globally, run the following command via the command prompt.

command line

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

Install and use the Office JavaScript linter

Microsoft provides a JavaScript linter to help you catch common errors when using the Office JavaScript library. If you create an add-in project with either the [Yeoman generator for Office Add-ins](#) or [Agents Toolkit](#), then the linter is installed and configured for you. Skip to [Run the linter](#).

If you created your project manually, install and configure the linter with the following steps.

1. In the root of the project, run the following two commands (after you've [installed Node.js and npm](#)).

command line

```
npm install office-addin-lint --save-dev  
npm install eslint-plugin-office-addins --save-dev
```

2. In the root of the project, create a text file named **eslint.config.js** (or **.mjs**), if there isn't one already there. Be sure to inherit the recommended configuration for **eslint-plugin-office-addins**. The **plugins** object should include a mapping to the **eslint-plugin-office-addins** plugin object. The following is a simple example that includes settings for TypeScript. Your **eslint.config.js** file may have additional properties and configurations.

JavaScript

```
const officeAddins = require("eslint-plugin-office-addins");
const tsParser = require("@typescript-eslint/parser");
const tsESLint = require("typescript-eslint");

export default [
  ...tsESLint.configs.recommended,
  ...officeAddins.configs.recommended,
  {
    plugins: {
      "office-addins": officeAddins,
    },
    languageOptions: {
      parser: tsParser,
    },
  },
];
```

3. In the root of the project, open the **package.json** file and be sure that the **scripts** array has the following member.

JSON

```
"lint": "office-addin-lint check",
```

Run the linter

Run the linter with the following command either in the terminal of an editor, such as Visual Studio Code, or in a command prompt. Problems found by the linter appear in the terminal or prompt, and also appear directly in the code when you're using an editor that supports linter messages, such as Visual Studio Code.

command line

```
npm run lint
```

Install Script Lab

Script Lab is a tool for quickly prototyping code that calls the Office JavaScript Library APIs. Script Lab is itself an Office Add-in and can be installed from AppSource at [Script Lab](#). There's a version for Excel, PowerPoint, and Word, and a separate version for Outlook. For information about how to use Script Lab, see [Explore Office JavaScript API using Script Lab](#).

Next steps

Try creating your own add-in or use [Script Lab](#) to try built-in samples.

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 a [tutorial](#).

Explore the APIs with Script Lab

Explore the library of built-in samples in [Script Lab](#) to get a sense for the capabilities of the Office JavaScript APIs.

See also

- [Core concepts for Office Add-ins](#)
- [Developing Office Add-ins](#)
- [Design Office Add-ins](#)
- [Test and debug Office Add-ins](#)
- [Publish Office Add-ins](#)
- [Learn about the Microsoft 365 Developer Program](#)

Build an Excel task pane add-in

05/07/2025

In this article, you'll walk through the process of building an Excel task pane add-in. You'll use either the Office Add-ins Development Kit or the Yeoman generator to create your Office Add-in. Select the tab for the one you'd like to use and then follow the instructions to create your add-in and test it locally. If you'd like to create the add-in project within Visual Studio Code, we recommend the Office Add-ins Development Kit.

Office Add-ins Development Kit

Prerequisites

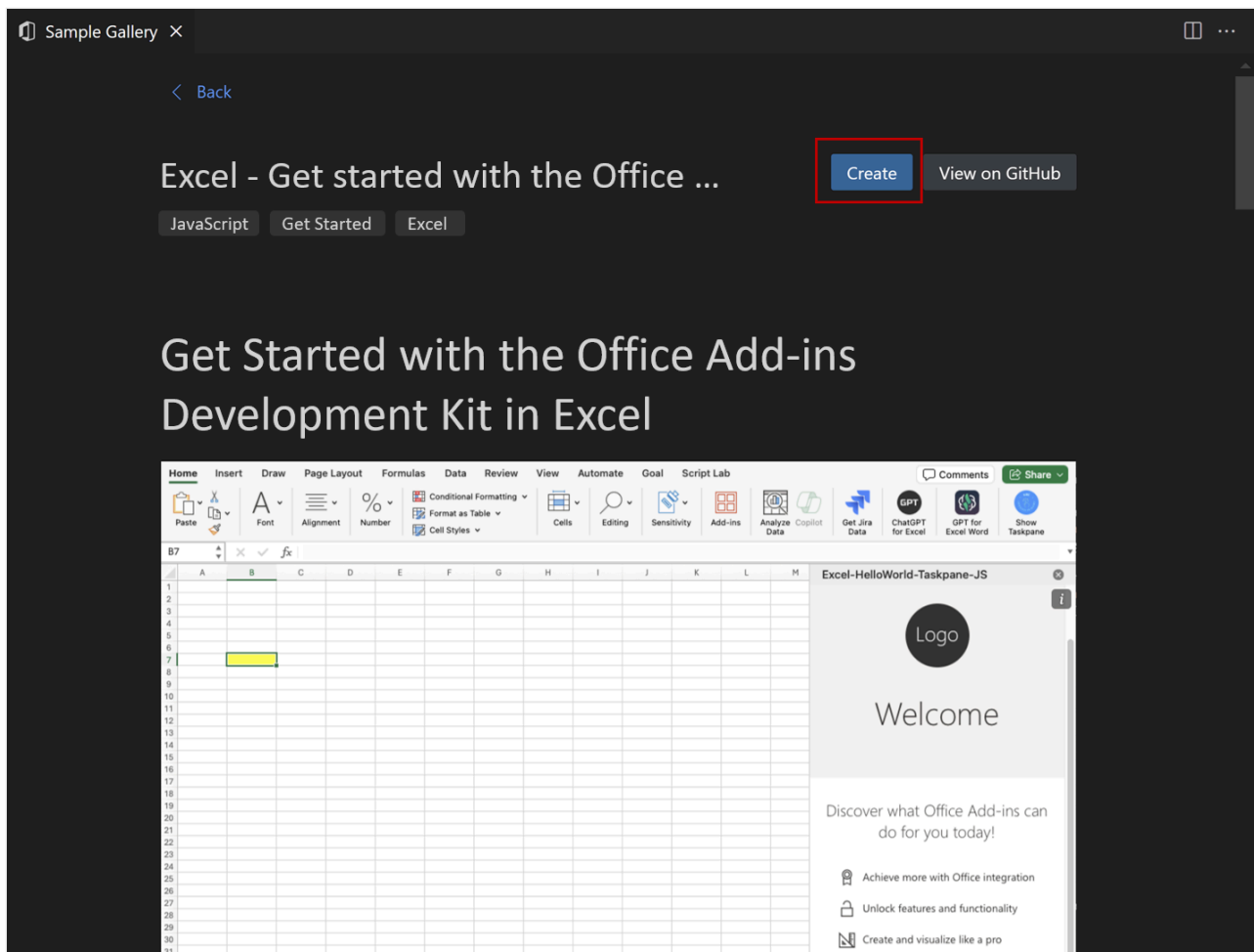
- Download and install [Visual Studio Code](#).
- Node.js (the latest LTS version). Visit the [Node.js site](#) to download and install the right version for your operating system. To verify if you've already installed these tools, run the commands `node -v` and `npm -v` in your terminal.
- Office connected to a Microsoft 365 subscription. You might qualify for a Microsoft 365 E5 developer subscription through the [Microsoft 365 Developer Program](#), see [FAQ](#) for details. Alternatively, you can [sign up for a 1-month free trial](#) or [purchase a Microsoft 365 plan](#).

Create the add-in project

Click the following button to create an add-in project using the Office Add-ins Development Kit for Visual Studio Code. You'll be prompted to install the extension if don't already have it. A page that contains the project description will open in Visual Studio Code.

Create an add-in in Visual Studio Code

In the prompted page, select **Create** to create the add-in project. In the **Workspace folder** dialog that opens, select the folder where you want to create the project.



The Office Add-ins Development Kit will create the project. It will then open the project in a *second* Visual Studio Code window. Close the original Visual Studio Code window.

! Note

If you use VSCode Insiders, or you have problems opening the project page in VSCode, install the extension manually by following [these steps](#), and find the sample in the sample gallery.

Explore the project

The add-in project that you've created with the Office Add-ins Development Kit contains sample code for a basic task pane add-in. If you'd like to explore the components of your add-in project, open the project in your code editor and review the files listed below. When you're ready to try out your add-in, proceed to the next section.

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

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

Try it out

1. Open the extension by selecting the Office Add-ins Development Kit icon in the **Activity Bar**.
2. Select **Preview Your Office Add-in (F5)**
3. In the Quick Pick menu, select the option **{Office Application} Desktop (Edge Chromium)**, where '{Office Application}' is the appropriate application, such as "Excel" or "Word". This will launch the add-in and debug the code.

The development kit checks that the prerequisites are met before debugging starts. Check the terminal for detailed information if there are issues with your environment. After this process, the Office desktop application launches and sideloads the add-in. Please note that the first time you run a project, it may take a few minutes to install the dependencies. You'll need to install the certificate when prompted.

Stop testing your Office Add-in

Once you are finished testing and debugging the add-in, *always* close the add-in by following these steps. (Closing the Office application or web server window doesn't reliably deregister the add-in.)

1. Open the extension by selecting the Office Add-ins Development Kit icon in the **Activity Bar**.
2. Select **Stop Previewing Your Office Add-in**. This closes the web server and removes the add-in from the registry and cache.
3. Close the Office application window.

Troubleshooting

If you have problems running the add-in, take these steps.

- Close any open instances of Office.
- Close the previous web server started for the add-in with the **Stop Previewing Your Office Add-in** Office Add-ins Development Kit extension option.

The article [Troubleshoot development errors with Office Add-ins](#) contains solutions to common problems. If you're still having issues, [create a GitHub issue](#) and we'll help you.

For information on running the add-in on Office on the web, see [Sideload Office Add-ins to Office on the web](#).

For information on debugging on older versions of Office, see [Debug add-ins using developer tools in Microsoft Edge Legacy](#).

Next steps

Congratulations, you've successfully created an Excel task pane add-in! Next, learn more about the capabilities of an Excel add-in and build a more complex add-in by following along with the [Excel add-in tutorial](#).

Code samples

- [Excel "Hello world" add-in](#): Learn how to build a simple Office Add-in with only a manifest, HTML web page, and a logo.

See also

- [Office Add-ins platform overview](#)
- [Develop Office Add-ins](#)
- [Excel JavaScript object model in Office Add-ins](#)
- [Excel add-in code samples](#)
- [Excel JavaScript API reference](#)
- [Using Visual Studio Code to publish](#)