

Design the UI of Office Add-ins

Article • 04/04/2023

Office Add-ins extend the Office experience by providing contextual functionality that users can access within Office clients. Add-ins empower users to get more done by enabling them to access external functionality within Office, without costly context switches.

Your add-in UI design must integrate seamlessly with Office to provide an efficient, natural interaction for your users. Take advantage of [add-in commands](#) to provide access to your add-in and apply the best practices that we recommend when you create a custom HTML-based UI.

Office design principles

Office applications follow a general set of interaction guidelines. The applications share content and have elements that look and behave similarly. This commonality is built on a set of design principles. The principles help the Office team create interfaces that support customers' tasks. Understanding and following them will help you support your customers' goals inside of Office.

Follow the Office design principles to create positive add-in experiences.

- **Design explicitly for Office.** The functionality, as well as the look and feel, of an add-in must harmoniously complement the Office experience. Add-ins should feel native. They should fit seamlessly into Word on an iPad or PowerPoint on the web. A well-designed add-in will be an appropriate blend of your experience, the platform, and the Office application. Apply document and UI theming where appropriate. Consider using Fluent UI for the web as your design language and tool set. The Fluent UI for the web has two flavors.
 - **For non-React UIs:** Use **Fabric Core**, an open-source collection of CSS classes and Sass mixins that give you access to colors, animations, fonts, icons, and grids. (It's called "Fabric Core" instead of "Fluent Core" for historical reasons.) To get started, see [Fabric Core in Office Add-ins](#).

ⓘ Note

While Fabric Core is the recommended library to design non-React add-ins, the team is working on **Fluent UI Web Components** to provide a newer solution. Built on [FAST](#) [↗], the Fluent UI Web Components library allows you to

use, customize, and build Web Components to create a more modern and standards-based UI. We invite you to test this library by **completing the quick start** and welcome feedback on your experience through [GitHub](#).

- **For React UIs:** use **Fluent UI React**, a React front-end framework designed to build experiences that fit seamlessly into a broad range of Microsoft products. It provides robust, up-to-date, accessible React-based components which are highly customizable using CSS-in-JS. To get started, see [Fluent UI React in Office Add-ins](#).
- **Favor content over chrome.** Allow customers' page, slide, or spreadsheet to remain the focus of the experience. An add-in is an auxiliary interface. No accessory chrome should interfere with the add-in's content and functionality. Brand your experience wisely. We know it's important to provide users with a unique, recognizable experience but avoid distraction. Strive to keep the focus on content and task completion, not brand attention.
- **Make it enjoyable and keep users in control.** People enjoy using products that are both functional and visually appealing. Craft your experience carefully. Get the details right by considering every interaction and visual detail. Allow users to control their experience. The necessary steps to complete a task must be clear and relevant. Important decisions should be easy to understand. Actions should be easily reversible. An add-in is not a destination – it's an enhancement to Office functionality.
- **Design for all platforms and input methods.** Add-ins are designed to work on all the platforms that Office supports, and your add-in UX should be optimized to work across platforms and form factors. Support mouse/keyboard and touch input devices, and ensure that your custom HTML UI is responsive to adapt to different form factors. For more information, see [Touch](#).

See also

- [Add-in development best practices](#)

Use Fluent UI React in Office Add-ins

Article • 02/12/2025

[Fluent UI React](#) is the official open-source JavaScript front-end framework designed to build experiences that fit seamlessly into a broad range of Microsoft products, including Microsoft 365 applications. It provides robust, up-to-date, accessible React-based components which are highly customizable using CSS-in-JS.

ⓘ Note

This article describes the use of Fluent UI React in the context of Office Add-ins. However, it's also used in a wide range of Microsoft 365 apps and extensions. For more information, see [Fluent UI React](#) and the [Fluent UI Web](#) open source repository.

This article describes how to create an add-in that's built with React and that uses Fluent UI React components.

Create an add-in project

You'll use the [Yeoman generator for Office Add-ins](#) to create an add-in project that uses React.

ⓘ Note

The React-based Add-ins created with the generator use Fluent UI React V9. This version doesn't support the Trident (IE) webview. If your add-in's users have versions of Office that require Trident, use one of the samples in [Office-Add-ins-Fluent-React-version-8](#) instead of this article. For information about which versions of Office use Trident, see [Browsers and webview controls used by Office Add-ins](#).

Install the 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 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

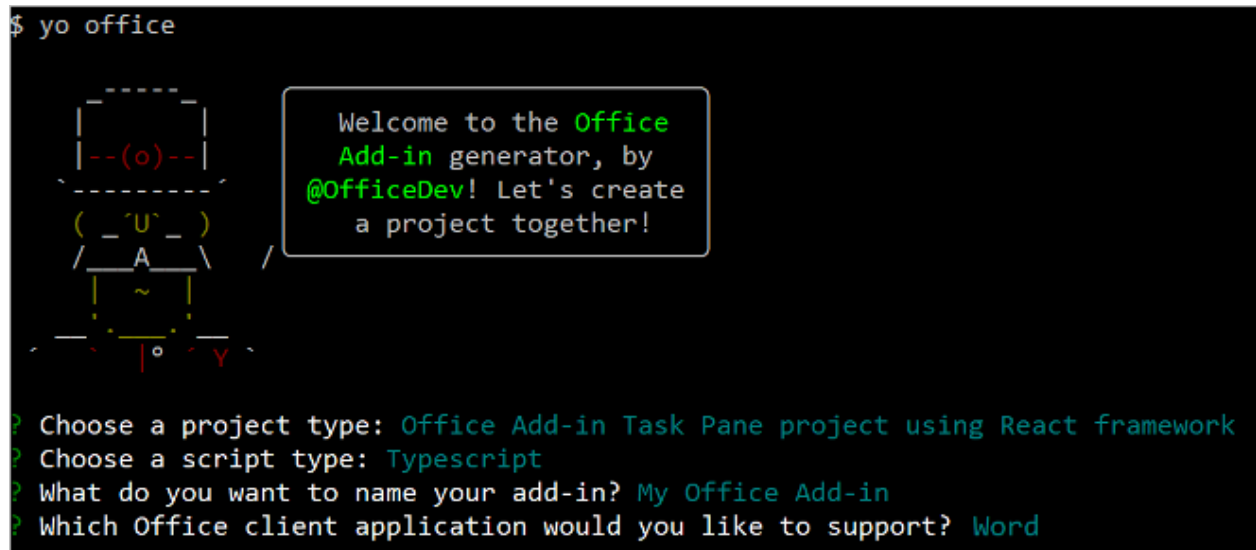
When you run the `yo office` command, you may receive prompts about the data collection policies of Yeoman and the Office Add-in CLI tools. Use the information that's provided to respond to the prompts as you see fit.

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

- **Choose a project type:** Specify `Office Add-in Task Pane project using React framework`.
- **Choose a script type:** Specify either `TypeScript` or `JavaScript`.
- **What do you want to name your add-in?** Specify `My Office Add-in`.

- Which Office client application would you like to support? Specify one of the hosts. (The screenshots in this article use `Word`. Running the project for the first time is easier if you select `Excel`, `PowerPoint`, or `Word`. See [Try it out.](#))

The following is an example.



```
$ yo office

  --(o)--
  |  _U_  |
  |  A   | /
  |  ~   |
  |  _   |
  |  o   | \
  |  Y   |

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 using React framework
? Choose a script type: Typescript
? What do you want to name your add-in? My Office Add-in
? Which Office client application would you like to support? Word
```

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

ⓘ Note

Fluent UI React v9 or later isn't supported with the Trident (IE) or EdgeHTML (Edge Legacy) webview controls. If your version of Office is using either of those, the task pane of the add-in generated by Yo Office simply contains a message to upgrade your version of Office. For more information, see [Browsers and webview controls used by Office Add-ins](#).

Explore the project

The add-in project that you've created with the Yeoman generator 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 following files. The file name extensions depend on which language you choose. TypeScript extensions are in parentheses. When you're ready to try out your add-in, proceed to the next section.

- The `./manifest.xml` or `./manifest.json` file in the root directory of the project defines the settings and capabilities of the add-in. To learn more about the `manifest.xml` file, see [Office Add-ins with the add-in only manifest](#). To learn more

about the **manifest.json** file, see [Office Add-ins with the unified app manifest for Microsoft 365](#).

⚠ Note

The [unified manifest for Microsoft 365](#) can be used in production Outlook add-ins. It's available only as a preview for Excel, PowerPoint, and Word add-ins.

- The `./src/taskpane/taskpane.html` file contains the HTML markup for the task pane and loads the Office JavaScript Library. It also tests whether the webview control supports Fluent UI React v9 and displays a special message if it doesn't.
- The `./src/taskpane/index.jsx (tsx)` file is the React root component. It loads React and Fluent UI React, ensures that the Office JavaScript library has been loaded, and applies the Fluent-defined theme.
- The `./src/taskpane/office-document.js (ts)` file contains the Office JavaScript API code that facilitates interaction between the task pane and the Office client application.
- The `./src/taskpane/components/` folder contains the React component `*.jss (tsx)` files that create the UI.

Try it out

1. Navigate to the root folder of the project.

```
command line
```

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

2. Complete the following steps to start the local web server and sideload your add-in.

⚠ 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.

- If this is your first time developing an Office Add-in on your machine, you may be prompted in the command line to grant Microsoft Edge WebView a loopback exemption ("Allow localhost loopback for Microsoft Edge WebView?"). When prompted, enter **Y** to allow the exemption. Note that you'll need administrator privileges to allow the exemption. Once allowed, you shouldn't be prompted for an exemption when you sideload Office Add-ins in the future (unless you remove the exemption from your machine). To learn more, see ["We can't open this add-in from localhost" when loading an Office Add-in or using Fiddler](#).

```
> office-addin-taskpane-js@0.0.1 start
> office-addin-debugging start manifest.xml

Debugging is being started...
App type: desktop
? Allow localhost loopback for Microsoft Edge WebView? (Y/n) Y
```

💡 Tip

If you're testing your add-in on Mac, run the following command before proceeding. When you run this command, the local web server starts.

command line

```
npm run dev-server
```

- To test your add-in, run the following command in the root directory of your project. This starts the local web server and opens the Office host application with your add-in loaded.

command line

```
npm start
```

⚠ Note

If you're testing your add-in in Outlook, `npm start` sideloads the add-in to both the Outlook desktop and web clients. For more information on how to sideload add-ins in Outlook, see [Sideload Outlook add-ins for testing](#).

- To test your add-in in Excel, Word, or PowerPoint on the web, run the following command in the root directory of your project. When you run this command, the local web server starts. Replace "{url}" with the URL of a Word document on your OneDrive or a SharePoint library to which you have permissions.

ⓘ Note

If you are developing on a Mac, enclose the `{url}` in single quotation marks. Do *not* do this on Windows.

command line

```
npm run start -- web --document {url}
```

The following are examples.

- `npm run start -- web --document https://contoso.sharepoint.com/:t:/g/EZGxP7ksiE5DuxvY638G798BpuhluxCMfF1WZQj3VYhYQ?e=F4QM1R`
- `npm run start -- web --document https://1drv.ms/x/s!jkcH7spkM4EGgcZUgqthk4IK3N0ypVw?e=Z6G1qp`
- `npm run start -- web --document https://contoso-my.sharepoint-df.com/:t:/p/user/EQda453DNTpFn11bFPhOVR0BwlrzetaXvnaRYii2lDr_oQ?e=RSccmNP`

If your add-in doesn't sideload in the document, manually sideload it by following the instructions in [Manually sideload add-ins to Office on the web](#).

ⓘ Note

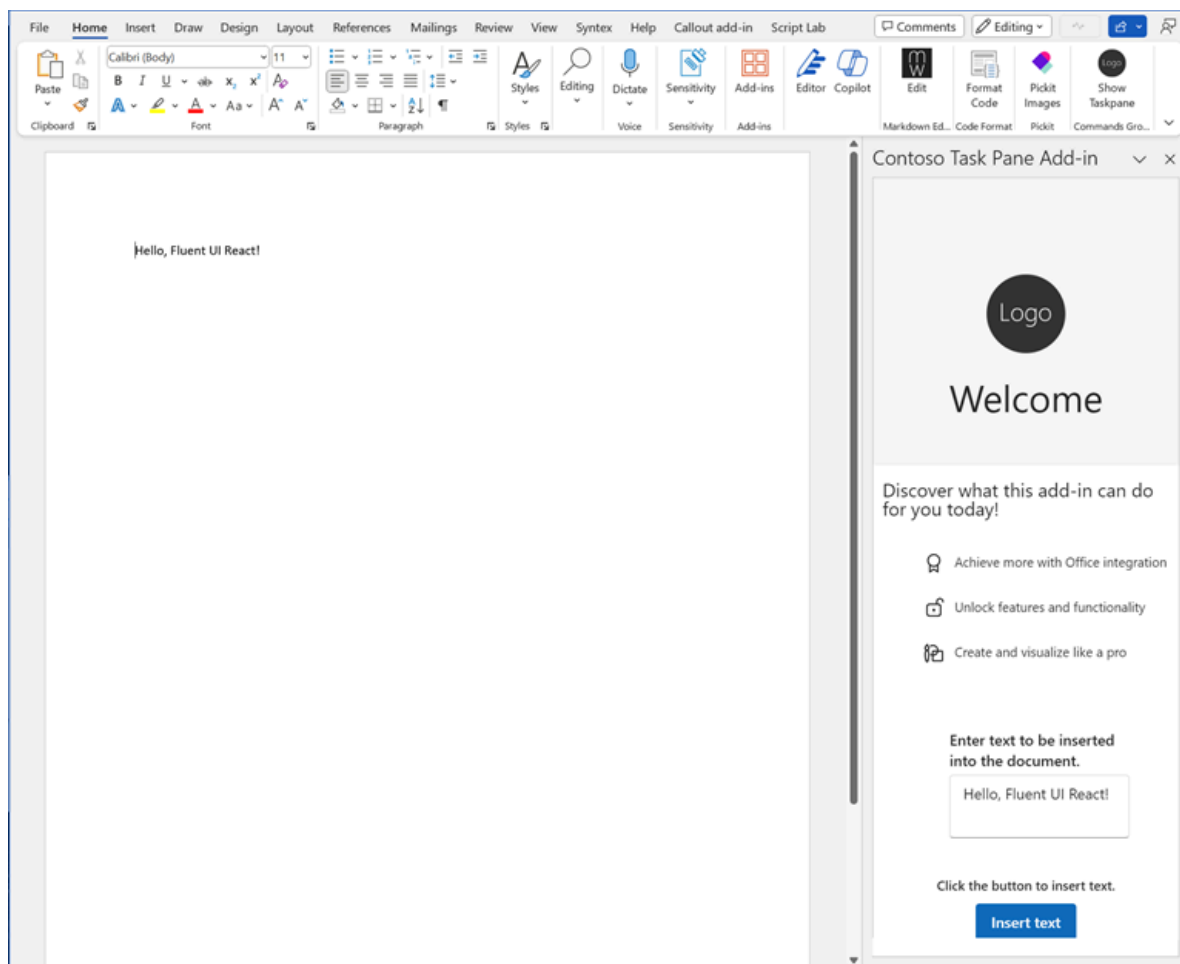
If this is the first time that you have sideloaded an Office add-in on your computer (or the first time in over a month), you're prompted first to delete an old certificate and then to install a new one. Agree to both prompts.

3. A **WebView Stop On Load** prompt appears. Select **OK**.
4. If the "My Office Add-in" task pane isn't already open, choose the **Home** tab, and then choose the **Show Taskpane** button on the ribbon to open the add-in task pane.

ⓘ **Note**

If you're testing your add-in in Outlook, create a new message. Then, navigate to the **Message** tab and choose **Show Taskpane** from the ribbon to open the add-in task pane.

5. Enter text into the text box and then select **Insert text**.



6. When you're ready to stop the dev server and uninstall the add-in, run the following command.

command line

```
npm stop
```

Migrate to Fluent UI React v9

If you have an existing add-in that implements an older version of Fluent UI React, we recommend migrating to Fluent UI v9. For guidance on the migration process, see [Getting started migrating to v9](#).

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

- [UX design patterns for Office Add-ins](#)
- [Fluent UI React](#)
- [Fluent UI GitHub repository](#)

Fabric Core in Office Add-ins

Article • 04/04/2023

Fabric Core is an open-source collection of CSS classes and Sass mixins that's *intended for use in non-React* Office Add-ins. Fabric Core contains basic elements of the Fluent UI design language such as icons, colors, typefaces, and grids. Fabric Core is framework independent, so it can be used with any single-page application or any server-side web UI framework. (It's called "Fabric Core" instead of "Fluent Core" for historical reasons.)

If your add-in's UI isn't React-based, you can also make use of a set of non-React components. See [Use Office UI Fabric JS components](#).

ⓘ Note

This article describes the use of Fabric Core in the context of Office Add-ins, but it's also used in a wide range of Microsoft 365 apps and extensions. For more information, see [Fabric Core](#) and the open source repo [Office UI Fabric Core](#).

ⓘ Note

While Fabric Core is the recommended library to design non-React add-ins, the team is working on [Fluent UI Web Components](#) to provide a newer solution. Built on [FAST](#), the Fluent UI Web Components library allows you to use, customize, and build Web Components to create a more modern and standards-based UI. We invite you to test this library by [completing the quick start](#) and welcome feedback on your experience through [GitHub](#).

Use Fabric Core: icons, fonts, colors

1. Add the content delivery network (CDN) reference to the HTML on your page.

HTML

```
<link rel="stylesheet" href="https://res-1.cdn.office.net/files/fabric-cdn-prod_20230815.002/office-ui-fabric-core/11.0.0/css/fabric.min.css">
```

2. Use Fabric Core icons and fonts.

To use a Fabric Core icon, include the "i" element on your page, and then reference the appropriate classes. You can control the size of the icon by changing the font size. For example, the following code shows how to make an extra-large table icon that uses the themePrimary (#0078d7) color.

HTML

```
<i class="ms-Icon ms-font-xl ms-Icon--Table ms-fontColor-themePrimary">
</i>
```

For more detailed instructions, see [Fluent UI Icons](#). To find more icons that are available in Fabric Core, use the search feature on that page. When you find an icon to use in your add-in, be sure to prefix the icon name with `ms-Icon--`.

For information about font sizes and colors that are available in Fabric Core, see [Typography](#) and the **Colors** table of contents at [Colors](#).

Examples are included in the [Samples](#) later in this article.

Use Office UI Fabric JS components

Add-ins with non-React UIs can also use any of the many components from [Office UI Fabric JS](#), including buttons, dialogs, pickers, and more. See the readme of the repo for instructions.

Examples are included in the [Samples](#) later in this article.

Samples

The following sample add-ins use Fabric Core and/or Office UI Fabric JS components. Some of these repos are archived, meaning that they are no longer being updated with bug or security fixes, but you can still use them to learn how to use Fabric Core and Fabric UI components.


- [Excel Add-in JavaScript SalesTracker](#)
- [Excel Add-in SalesLeads](#)
- [Excel Add-in WoodGrove Expense Trends](#)
- [Excel Content Add-in Humongous Insurance](#)
- [Office Add-in Fabric UI Sample](#)
- [Office-Add-in-UX-Design-Patterns-Code](#)
- [Outlook Add-in GifMe](#)
- [PowerPoint Add-in Microsoft Graph ASPNET InsertChart](#)

- [Word Add-in Angular2 StyleChecker](#)↗
- [Word Add-in JS Redact](#)↗
- [Word Add-in MarkdownConversion](#)↗

Accessibility guidelines

Article • 12/03/2024

As you design and develop your Office Add-ins, you'll want to ensure that all potential users and customers are able to use your add-in successfully. Engineering and implementing inclusive experiences provide better usability and customer satisfaction, as well as a larger market for your solutions. We recommend you become familiar with the Web Content Accessibility Guidelines (WCAG), international web standards that define what's needed for your add-in to be accessible.

- [Explore the WCAG standards and resources](#)
- [Explore the WCAG tutorials](#) 

Apply the following guidelines to ensure that your solution is accessible to all audiences.

Design for multiple input methods

- Ensure that users can perform operations by using only the keyboard. Users should be able to move to all actionable elements on the page by using a combination of the `Tab` and arrow keys.
- On a mobile device, when users operate a control by touch, the device should provide useful audio feedback.
- Provide helpful labels for all interactive controls.
- [Explore more design and UI resources](#).

Make your add-in easy to use

- Don't rely on a single attribute, such as color, size, shape, location, orientation, or sound, to convey meaning in your UI.
- Avoid unexpected changes of context, such as moving the focus to a different UI element without user action.
- Provide a way to verify, confirm, or reverse all binding actions.
- Provide a way to pause or stop media, such as audio and video.
- Don't impose a time limit for user action.

Make your add-in easy to see

- Avoid unexpected color changes.

- Provide meaningful and timely information to describe UI elements, titles and headings, inputs, and errors. Ensure that names of controls adequately describe the intent of the control.
- Verify you UI elements render correctly in the Windows high-contrast themes.
- Follow [standard guidelines](#) [↗] for color contrast.

Account for assistive technologies

- Avoid using features that interfere with assistive technologies, including visual, audio, or other interactions.
- Don't provide text in an image format. Screen readers can't read text within images.
- Provide a way for users to adjust or mute all audio sources.
- Provide a way for users to turn on captions or audio description with audio sources.
- Provide alternatives to sound as a means to alert users, such as visual cues or vibrations.

Test your add-in

- Always use accessibility verification and testing tools like [Accessibility Insights](#) [↗] on your add-in to catch and resolve issues before you ship.
- Verify the screen reading experience using [Windows Narrator](#) [↗], [JAWS](#) [↗], or [NVDA](#) [↗].
- Periodically run the tools to keep up with changes to the international accessibility guidelines. For more information, see [Accessibility testing](#).

See also

- [Accessibility in the Store](#)
- [Web Content Accessibility Guidelines \(WCAG\) 2.2](#) [↗]
- [Developing for Web Accessibility](#) [↗]
- [Accessibility Fundamentals Learning Path](#)
- [European Accessibility Act \(EAA\)](#) [↗]