200 XP

Understand how to customize Office add-ins

7 minutes

The Office add-ins platform enables you to customize your add-in. In this module, you'll explore how to customize your add-in by persisting state, and using Office UI Fabric (Fluent UI) and Microsoft Graph. By the end of this module, you should know how to customize Office add-ins using persisted state, Office UI Fabric (Fluent UI), and Microsoft Graph.

Understand the options for persisting state and settings

The Office add-ins platform provides several ways for your add-in to persist state and settings. Your options depend on the Office applications you plan to support and on the type of add-in you plan to develop.

Options to persist state and settings

The Office JavaScript API provides objects for your add-in to save state across user sessions. The following table lists the options, along with the supported add-in types and Office host applications.

Office API object	Supported add-in types	Supported Office hosts	Storage information
CustomProperties	MailApp	Outlook	Item data is stored on the message or appointment the add-in is working on.*
CustomXmlParts	TaskPaneApp	Excel (host-specific Excel JavaScript API), Word (Office JavaScript Common API)	Data is stored in a custom XML part of the document or workbook.
RoamingSettings	MailApp	Outlook	Data is stored on the user's Exchange mailbox and associated with the specific add-in.*

Office API object	Supported add-in types	Supported Office hosts	Storage information
Settings	ContentApp, TaskPaneApp	Excel, PowerPoint, Word	Data is stored on the document, workbook, or presentation the add-in is working on.*

**Data stored in name/value pairs in a property bag*

You can also use HTML5 web storage and other techniques available through the add-in's underlying browser control.

> **Important**
>
> Don't store passwords and sensitive personally identifiable information (PII) on the user's device.

Understand Office UI Fabric (Fluent UI) in Office add-ins

As you build your add-in, you have many UI design factors to consider. The Office UI Fabric (Fluent UI) provides elements that adhere to Office branding so your add-in looks like a natural extension of Office.

> **Note**
>
> Using UI Fabric is optional but recommended.

About UI Fabric

Office UI Fabric (Fluent UI) has two (2) main areas:

- Fabric Core - Provides basic elements like font, icons, and color
- Fabric React components - Includes Fabric Core elements and adds input, navigation, and notification components, among others

Fabric Core

Fabric Core provides basic design elements that reflect or sync with Office branding.

To start using Fabric Core, reference the CSS in your HTML page, as shown in the following code.

HTML

```html
<link rel="stylesheet"
href="https://static2.sharepointonline.com/files/fabric/office-ui-fabric-
core/9.6.1/css/fabric.min.css">
```

You can then use Fabric icons, fonts, and colors. The following example shows how you can include an extra-large table icon in the Office application's primary theme color.

HTML

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

Fabric components

Fabric React provides UX components for input, navigation, notification, and other categories. It builds on and includes Fabric Core.

Recommended components you can use in your add-in are as follows:

- Breadcrumb
- Button
- Checkbox
- ChoiceGroup
- Dropdown
- Label
- List
- Pivot
- TextField
- Toggle

Tip

You can use the Yeoman generator for Office add-ins to create a project that references Fabric React. An available project type is **Office add-in Task Pane project using React framework**.

Understand when and how to use Microsoft Graph in Office add-ins

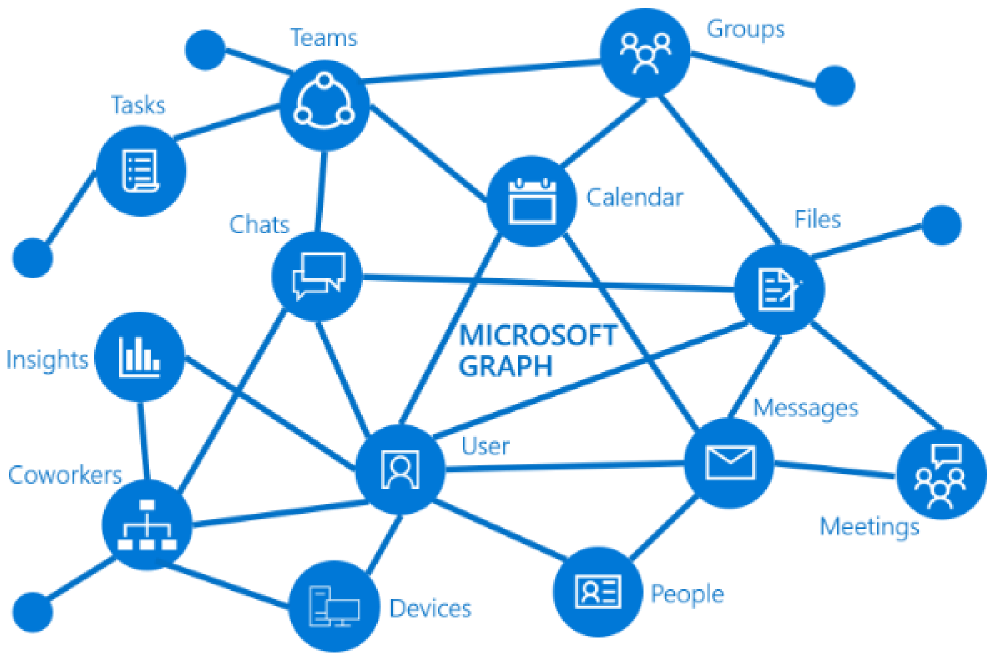
Your add-in can connect to Microsoft Graph and access the user's data so they can accomplish more useful and productive scenarios. Example tasks are:

- Read files from OneDrive
- Fetch email attachments
- Get the user's profile

Why use Microsoft Graph

Microsoft Graph REST APIs provide a way for your add-in to access the user's data in services like:

- Azure Active Directory
- Office 365 services
- Enterprise Mobility and Security services
- Windows 10 services
- Dynamics 365



Microsoft Graph

How to authorize to Microsoft Graph

To connect to and use Microsoft Graph, your add-in needs to:

- Authenticate the user

- Be authorized to act on the user's behalf

Authentication

The add-in can get an access token from Azure Active Directory (Azure AD) when the user has signed in. Azure AD doesn't allow its sign-in page to open in an iframe, and the add-in task pane is an iframe when the add-in is launched in Office on the Web. So, use the Office JavaScript Dialog API to display the Azure AD sign-in form. If your add-in includes custom functions that need authorization to Microsoft Graph, use the custom functions Dialog API to display the sign-in form.

Authorization

After the user signs in, your add-in gets an access token to use in later API calls to Microsoft Graph. The access token can never give the add-in more or greater permissions than the user has. Users typically only have permissions to data about themselves, their own files and email, and objects that have been shared with them. If your add-in gets Microsoft Graph data about multiple users, then it can be used successfully only by users with admin-level permissions.

Recommended libraries

Depending on your development choices, you can use one of the following libraries for authentication and authorization as appropriate.

- Your server-side is on a .NET-based framework (for example, .NET Core or ASP.NET): use MSAL.NET
- Your server-side is node.js-based: use Passport Azure AD
- Your add-in uses Implicit flow: use msal.js

Summary

The Office add-ins platform enables you to customize your add-in. In this module, you explored how to customize your add-in by persisting state, and using Office UI Fabric (Fluent UI) and Microsoft Graph. By the end of this module, you should know how to customize Office add-ins using persisted state, Office UI Fabric (Fluent UI), and Microsoft Graph.

Customize Office add-ins

1. An Outlook add-in needs to store data. What's an Office JavaScript object the add-in developer might use?

- ○ CustomProperties
- ○ CustomXmlParts
- ○ Settings

2. Joe has decided to use Office UI Fabric in an add-in. What is an advantage of doing so?

- ○ Fabric has three (3) main areas.
- ○ Fabric Core is built on Fabric React.
- ○ Fabric reflects Office branding.

3. Joe's connecting to Microsoft Graph in an add-in. Where should the add-in display the Azure Active Directory (Azure AD) sign-in form for a user in Office on the web?

- ○ Custom function
- ○ Dialog
- ○ Task pane

Check your answers