

Teams & Office Add-Ins Intro

© 2019 - 2020 ALEXANDER.PAJER@INTEGRATIONS.AT

A solid blue horizontal bar spanning the width of the slide at the bottom.

Introduction to Teams

Introduction to Office Add-Ins

Introduction to Teams

Microsoft Teams

Microsoft Teams is a persistent chat-based collaboration platform complete with document sharing, online meetings, and many more extremely useful features for business communications.

It provides

- Teams and channels
- Conversations within channels and teams
- Document storage in SharePoint
- Online Meetings & Online Video Calling and screen sharing

It is extendable using State-of-Art Web Technologies



Microsoft Teams platform UX elements

The Microsoft Teams Platform provides flexible UI elements for apps to take advantage of.

Microsoft Teams platform elements:

- Cards and card actions.
- Task modules.
- Deep links.
- Web content pages.

Extensible points in the Teams client

There are multiple places where the Microsoft Teams client can be extended to allow users to interact with the app.

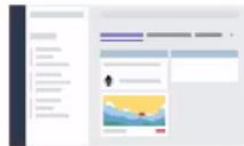
Options for extending the Teams client:

- Teams, channels and group chats
- Personal apps
- Messages



Immersive experience

The platform places content and conversations side by side.



Tabs and conversations

Tabs give your app a dedicated canvas that allows people to jump between conversations and your experience.



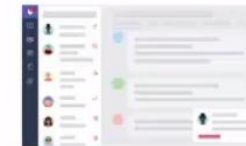
Customizable cards

Cards let you surface important tasks, information, and reminders in actionable ways.



Bot framework

The Bot framework offers a natural language UX that can interact with your services.



Instant notifications

The activity feed displays notifications for your users.



Compose extensions

Compose Extensions are quick and easy shortcuts to key components of your service.

App Studio for Microsoft Teams app development

App Studio streamlines the process of creating manifests and packages for Teams apps.

Use App Studio to:

Create and edit app manifests

Design and preview cards

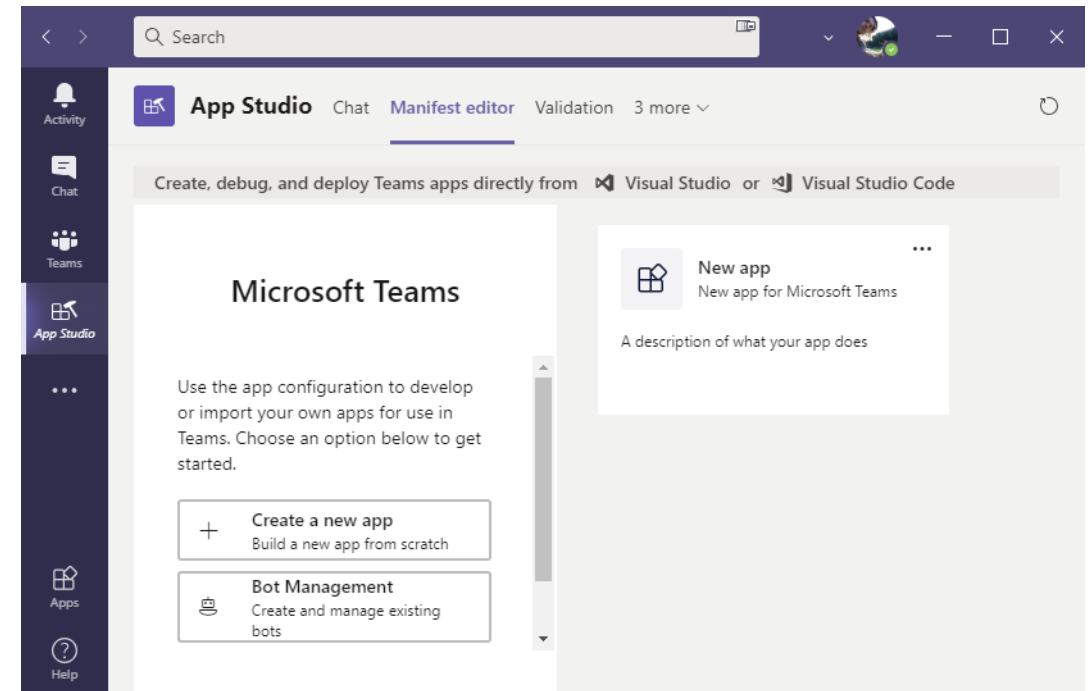
Find documentation

Access UI controls in the React control library



App Studio
Developer tools

Create new Microsoft Teams apps, design and preview bot cards, and explore documentation using App Studio.



Scaffolding & Basic Commands

Teams Apps are scaffolded using Yeoman Generator for Teams

- `npm install -g yo generator-teams`
- `yo teams`

Create the app package

- `gulp manifest`

Build your application

- `gulp build`

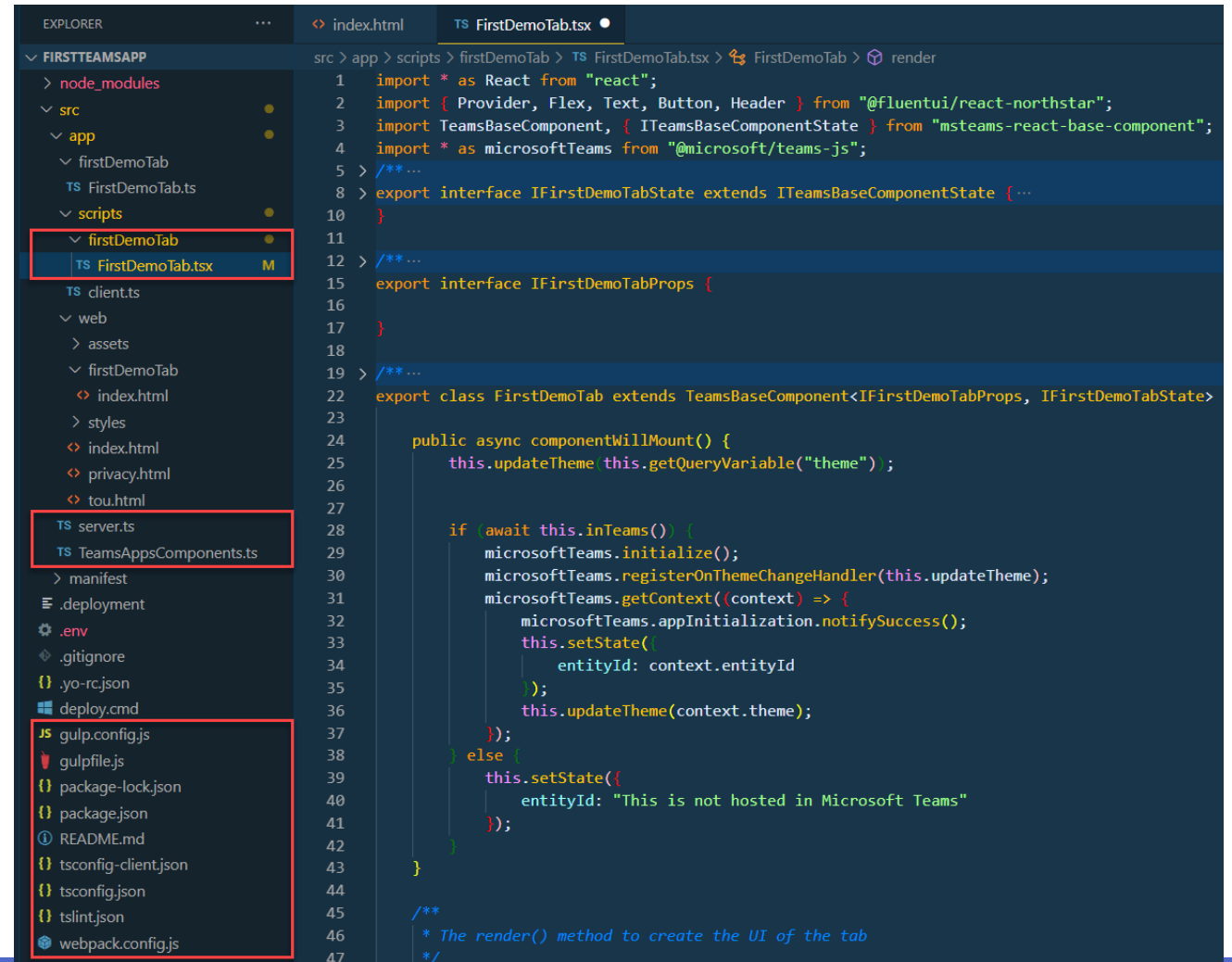
Establish an exposed, secure tunnel to your tab

- `gulp ngrok-serve`
- ngrok URL is changed every time you run ngrok-serve ... unless you have a paid ngrok account

Project Structure

File & Folder structure:

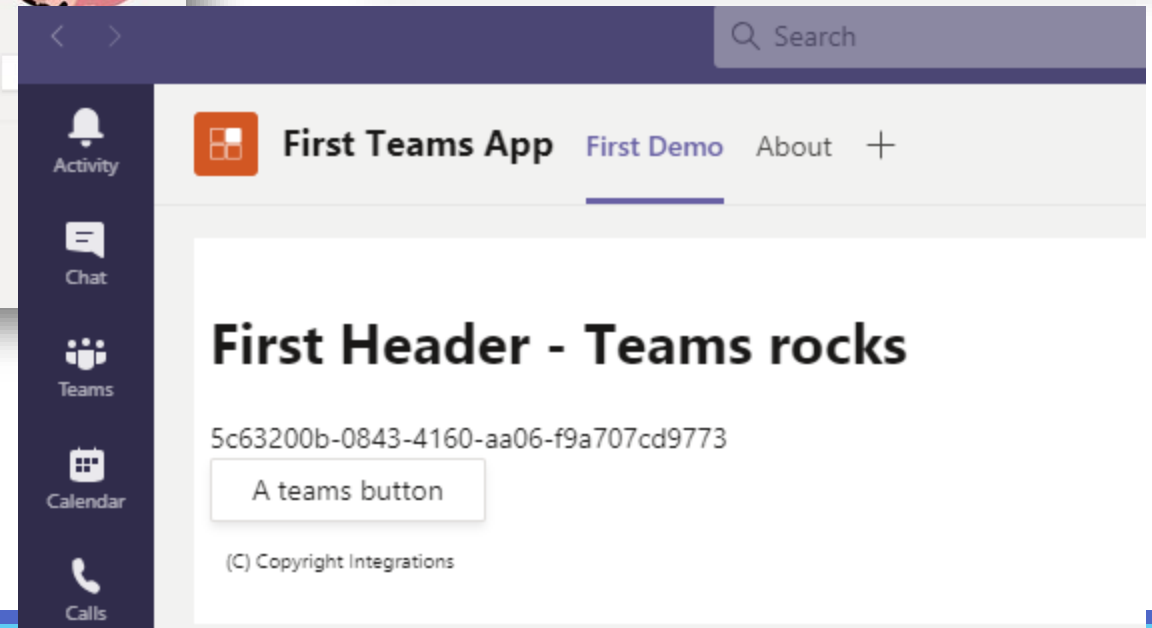
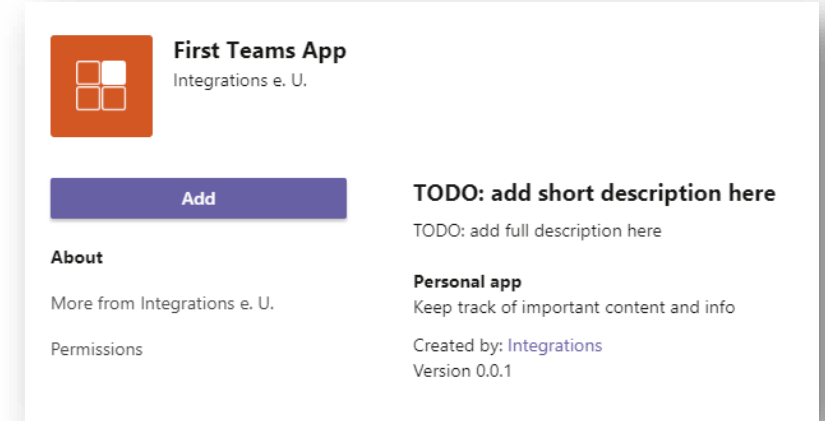
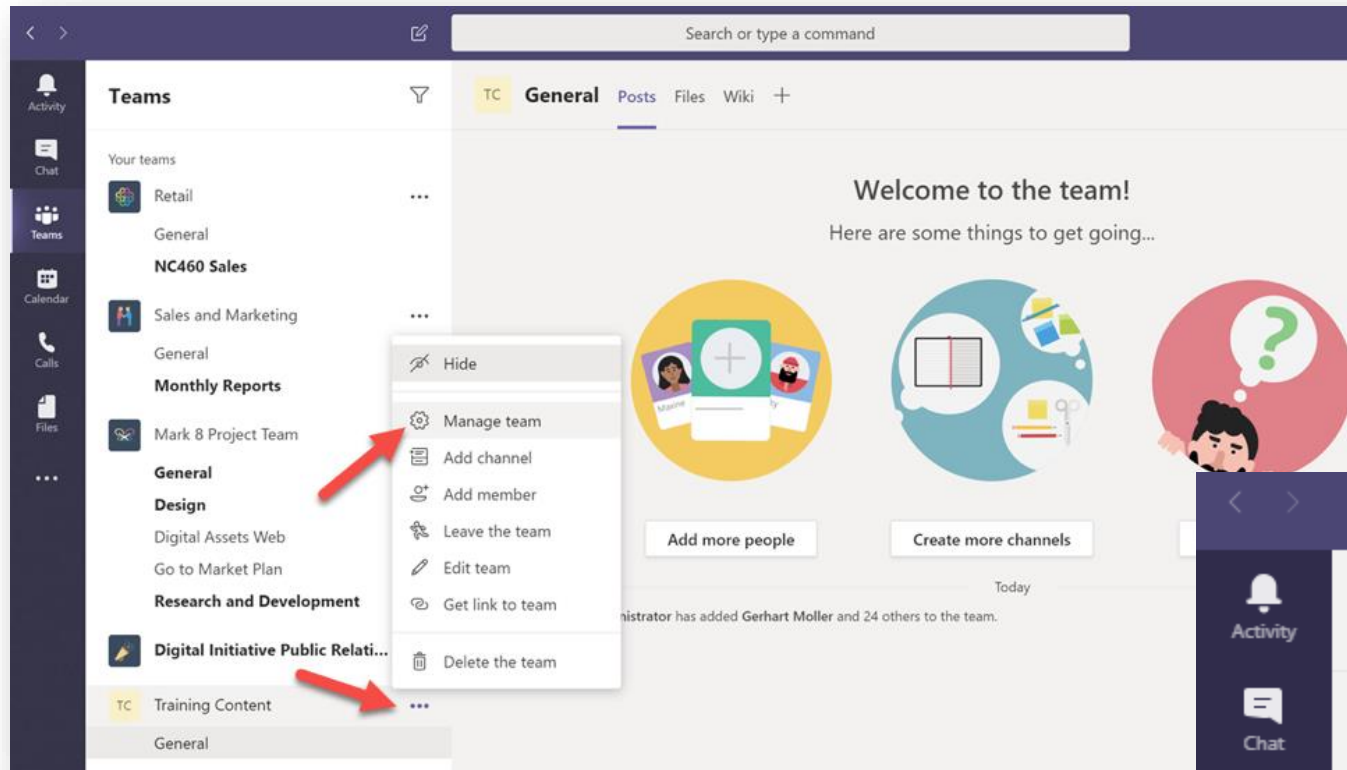
- root folder
- src folder
 - Manifest folder
 - Build Manifest: gulp manifest
 - App folder
 - Build App: gulp build
 - Serve App: gulp serve
 - Serve App using ngrok:
 - gulp ngrok-serve



The screenshot displays the Visual Studio Code interface. On the left, the Explorer sidebar shows the project structure for 'FIRSTTEAMSAPP'. The 'src' folder is expanded, showing subfolders 'app' and 'scripts'. The 'firstDemoTab' folder under 'scripts' is selected, containing 'TS FirstDemoTab.tsx' and 'TS client.ts'. The 'Manifest' folder is also visible under 'app'. The main editor shows the content of 'TS FirstDemoTab.tsx', which includes imports for React, Fluent UI components, and Teams SDK, followed by interface definitions for state and props, and a class component 'FirstDemoTab' that implements 'componentWillMount' and 'render' methods.

```
1 import * as React from "react";
2 import { Provider, Flex, Text, Button, Header } from "@fluentui/react-northstar";
3 import TeamsBaseComponent, { ITeamsBaseComponentState } from "msteams-react-base-component";
4 import * as microsoftTeams from "@microsoft/teams-js";
5 /** ...
8 > export interface IFirstDemoTabState extends ITeamsBaseComponentState { ...
10 }
11
12 > /** ...
15 export interface IFirstDemoTabProps {
16 }
17
18
19 > /** ...
22 export class FirstDemoTab extends TeamsBaseComponent<IFirstDemoTabProps, IFirstDemoTabState>
23
24   public async componentWillMount() {
25     this.updateTheme(this.getQueryVariable("theme"));
26
27     if (await this.inTeams()) {
28       microsoftTeams.initialize();
29       microsoftTeams.registerOnThemeChangeHandler(this.updateTheme);
30       microsoftTeams.getContext((context) => {
31         microsoftTeams.appInitialization.notifySuccess();
32         this.setState({
33           entityId: context.entityId
34         });
35         this.updateTheme(context.theme);
36       });
37     } else {
38       this.setState({
39         entityId: "This is not hosted in Microsoft Teams"
40       });
41     }
42   }
43
44   /**
45    * The render() method to create the UI of the tab
46    */
47   render() {
```

Add App



Microsoft Teams Toolkit

The Microsoft Teams Toolkit extension enables you to create, debug and deploy Teams apps directly from Visual Studio Code

The screenshot shows the Microsoft Teams Toolkit interface within Visual Studio Code. On the left is a dark sidebar with a menu containing: 'Welcome' (highlighted), 'Edit app package', 'Validate', 'Publish', and 'Bot management'. The main area has a dark blue header with the Microsoft Teams logo and the text 'Microsoft Teams' and 'Easily create your own apps and extensions for Microsoft Teams.' Below this is a 'Document Library' section with three columns: 'Setup', 'Development', and 'Publishing'. Each column contains a list of tasks with external link icons. At the bottom, there is a footer with the text 'Got issues, ideas, or suggestions? [Share feedback](#)' and icons for GitHub and Twitter.

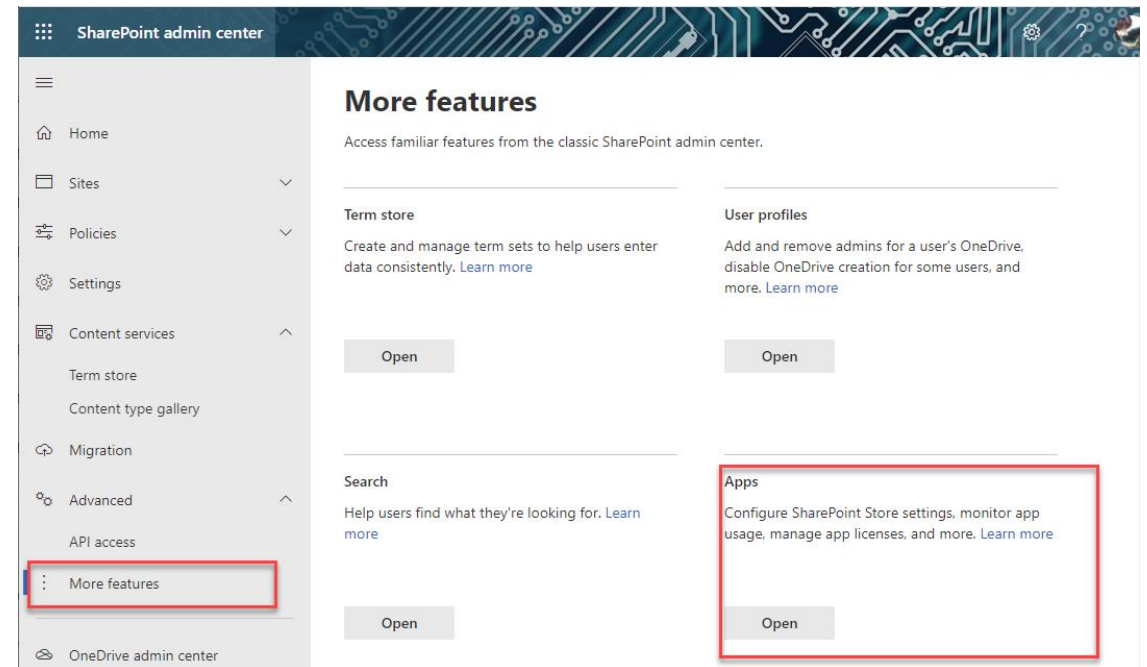
Setup	Development	Publishing
<ul style="list-style-type: none">Create a developer accountEnable custom apps for your developer tenantInstall and setup ngrok	<ul style="list-style-type: none">Package your app with App StudioTunnel to your local development server with ngrokCreate an embedded web experiences with tabsCreate a bot for TeamsCreate a messaging extensionHow to collection user input	<ul style="list-style-type: none">Sideload your appPublish your app to your tenant catalogPublish your app to public store

Options for distributing a Teams app

Three options for distributing a Microsoft Teams app.

- Share app package directly.
- Publish app to organizational app catalog.
- Publish app to public App Store.

All app installations in Microsoft Teams are context-specific



Introduction to Office Add-Ins

Yeoman Generator for Office Add-ins

Includes Office JavaScript API requirement sets.

Use Office JavaScript APIs.

Make asynchronous calls using proxy objects.

Suports common SPAs

- Angular = AngularJS
- To use Angular 10
 - Scaffold Angular Proj
 - Execute yo office from same folder

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

```

C:\>yo
Microsoft Windows [Version 10.0.19041.572]
(c) 2020 Microsoft Corporation. All rights reserved.
Clink v0.4.9 [git:2fd2c2] Copyright (c) 2012-2016 Martin Ridgers
http://mridgers.github.io/clink

H:\Projects>yo office

  Welcome to the Office
  Add-in generator, by
  @OfficeDev! Let's create
  a project together!

? Choose a project type: (Use arrow keys)
> Office Add-in Task Pane project
  Office Add-in Task Pane project using Angular framework
  Office Add-in Task Pane project using React framework
  Office Add-in Task Pane project supporting single sign-on
  Office Add-in project containing the manifest only
  Excel Custom Functions Add-in project

```

Office Add-ins manifest

An Office Add-in's XML manifest file defines the settings and capabilities of the add-in. It describes how your add-in should be activated when an end user installs and uses it with Office documents and applications.

Role of the manifest.

Elements defined in a manifest:

- Add-in metadata.
- Information about how add-in integrates with Office.
- Location of images add-in should use for branding & command iconography.
- Permissions add-in requires.
- Dimensions of add-in.
- Rules that specify when add-in should activate in a message or appointment.

Office Add-ins manifest

```
EXPLORER
...
manifest.xml X

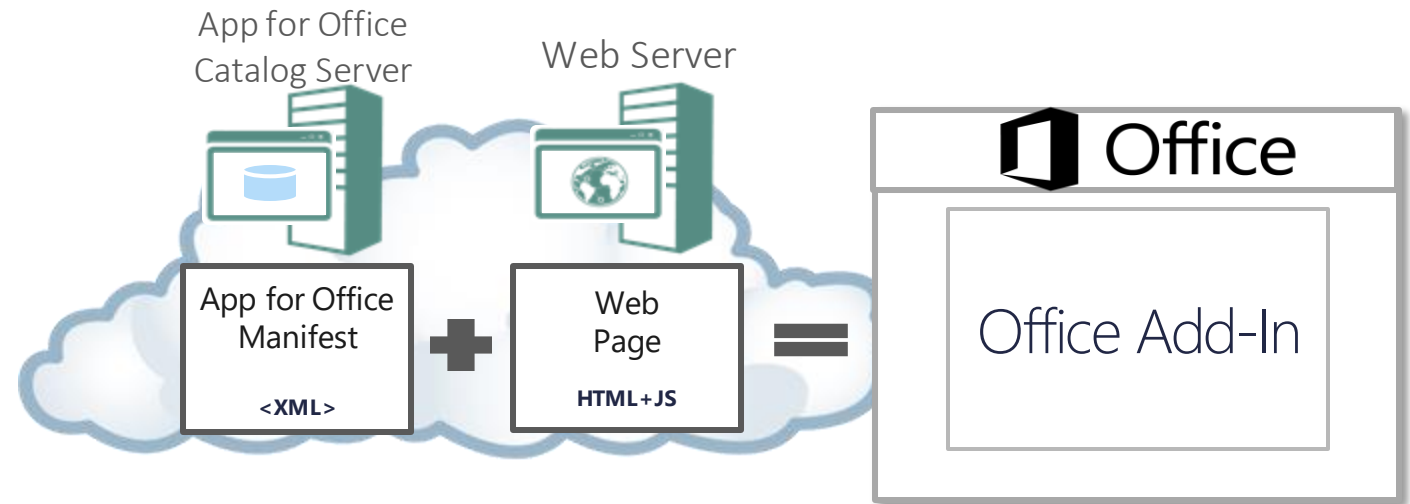
FIRSTOUTLOOKADDIN
  .vscode
  assets
  node_modules
  src
    commands
    commands.html
    commands.js
    taskpane
      taskpane.css
      taskpane.html
      taskpane.js
      .eslintrc.json
      CONTRIBUTING.md
      LICENSE
      manifest.xml
      package-lock.json
      package.json
      README.md
      tsconfig.json
      webpack.config.js

manifest.xml
10 <ImageSourceLocation DefaultValue="https://localhost:3000/assets/icon-60.png"/>
11 <SupportUrl DefaultValue="https://www.contoso.com/help"/>
12 <AppDomains>
13   <AppDomain>https://www.contoso.com</AppDomain>
14 </AppDomains>
15 <Hosts>
16   <Host Name="Mailbox"/>
17 </Hosts>
18 <Requirements>
19   <Sets>
20     <Set Name="Mailbox" MinVersion="1.1"/>
21   </Sets>
22 </Requirements>
23 <FormSettings>
24   <Form xsi:type="ItemRead">
25     <DesktopSettings>
26       <SourceLocation DefaultValue="https://localhost:3000/taskpane.html"/>
27       <RequestedHeight>250</RequestedHeight>
28     </DesktopSettings>
29   </Form>
30 </FormSettings>
31 <Permissions>ReadWriteItem</Permissions>
32 <Rule xsi:type="RuleCollection" Mode="Or">
33   <Rule xsi:type="ItemIs" ItemType="Message" FormType="Read"/>
34 </Rule>
```


Anatomy of an Office Add-In

Each App for Office is based on XML-based manifest

- Manifest points to a Web page
- Manifest defines the type of the App for Office
- Manifest defines which Office applications it supports
- Manifest defines required capabilities



Office Add-in programming model

The Office Add-in programming model relies on two JavaScript object models:

Host-specific JavaScript API

- Host-specific APIs for Excel and Word providing strongly-typed objects that you can use to access specific elements in host application.

Common API

- Enables you to access features such as:
 - UI.
 - Dialogs.
 - Client settings that are common across multiple types of Office applications.