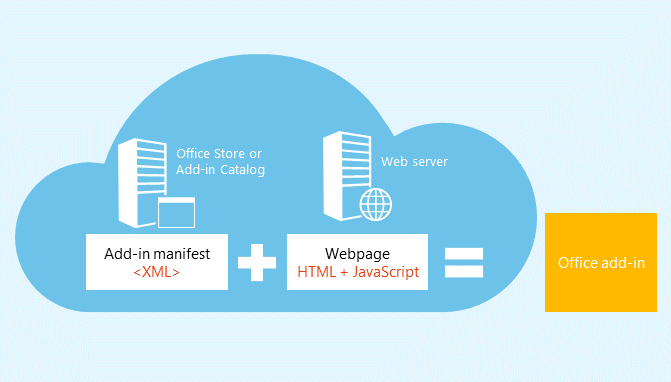
# Office for Mac

Visiting Office Add-ins documentation via <https://msdn.microsoft.com/EN-US/library/office/jj220082.aspx> link you can find lots of information about how to create your own add-ins for different office products. The following image contains the summarized information for general understanding:



You can see that a modern add-in is a set of JavaScript and HTML files that should be described with XML manifest file. In fact, if you are going to publish your own add-in like a task pane, a content add-in or a mail add-in, you need to create all interface elements in HTML documents, implement business logic in JavaScript and publish all these files to Office Marketplace, web site or even network share. In order to install the add-in, you need to provide manifest file that will describe the add-in including the source location and the type for the add-in. Office will use this information to download all content and deploy the add-in.

Of course, you should have a way to work with Office objects in JavaScript and for this Microsoft provides a special Office.js file with all needed interfaces inside.

Due to HTML and JavaScript you can use any text editor to create your own add-in. So, Visual Studio Code should work fine and we simply need to select which type of add-in we will create. But in case of Office for Mac it’s not easy. The problem is that Office 2016 for Mac still doesn’t support add-ins almost for all products there. For example, you can create an add-in for Word for Windows or for Office 365 Word Web Application but you cannot deploy the same add-in to Word for Mac. But the key word is **almost** and Outlook 2016 for Mac already supports modern add-in model. So, if you are going to develop add-ins for Office 2016 and if you want to use your Mac you can create add-ins for Outlook or for Office 365 Web Applications. In any case, the idea is the same and if you have an add-in for Word Web Application, you can deploy it to Word for Windows and if you have an add-in for Outlook 2016 for Mac, you can deploy it to Outlook for Windows and so on.

In this post I decided to use Outlook 2016 for Mac but before repeating all these steps you need to make sure that your Outlook is connected to Exchange Server. It’s Exchange that allows to manage add-ins for particular users or groups. You will not be able to find any special menu items in your Outlook but you can find all needed management panels in Exchange.

Working with Office add-ins documentation you can find many examples that show how to use Office API for different types of add-ins. But in the first stage the hardest thing that you need to create is XML manifest file. Because it contains lots of elements inside I recommend to use Yeoman tool to scaffold Office add-in project. In order to do it, you need to install office generator:

**npm install generator-office**

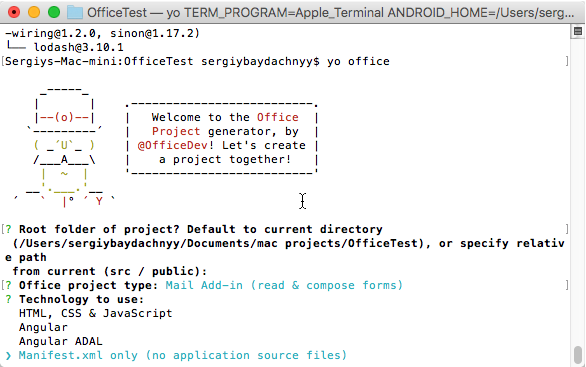
Once generator is installed you can use Yeoman:

**yo office**

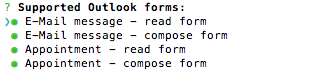
Of course, all these tasks you should execute in the context of your empty working folder.

Yeoman tool will ask several questions like add-in name and type of add-in. Because we are going to create an add-in for Outlook, you need to select E-Mail Add-in.

Not sure if you are going to create a very complex add-in but I decided to ignore existing JavaScript/HTML/Angular templates and simply ask Yeoman to generate XML manifest:



Finally, you need to select purpose for the add-in:



In our case I am going to create an add-in that will be integrated to read e-mail form.

Once manifest file is created you can open it in Code and remove some not needed elements like DisableEntityHighlighting, item edit form, support uri etc. It’s not a problem because the manifest is well structured and it’s easy to understand all elements there.

It’s time to create the add-in itself; and I decided to use an example that you can find in MSDN. This example shows a panel that contains information about sender and recipient. It’s not very useful but it’s easy to implement and using this example you can see all aspects of Office development. Just implement this HTML form:

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="X-UA-Compatible" content="IE=Edge" >

<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.2.min.js"></script>

<script src="https://appsforoffice.microsoft.com/lib/1.1/hosted/Office.js" type="text/javascript"></script>

<script src="default.js" type="text/javascript"></script>

<title>Sender and Recipient</title>

</head>

<body style="margin: 4px">

<h3>Email Details</h3>

<table>

<tr><td>&nbsp;</td><td>Sender</td><td>Recipient</td></tr>

<tr><td>Display Name</td><td id="senderDisplayName" /><td id="recipientDisplayName" /></tr>

<tr><td>Email Address</td><td id="senderEmailAddress" /><td id="recipientEmailAddress" /></tr>

</table>

</body>

</html>

This form is very simple and contains a table with several cells there. Pay special attention that in the beginning we provide a link to Office.js that contains Office API.

Before you start coding you need to enable IntelliSense system for Office API. As usual, we can use tsd tool and execute the following command:

**tsd install office-js**

Right after that we can implement default.js:

var mailbox;

var settings;

Office.initialize = function () {

$(document).ready(function () {

mailbox = Office.context.mailbox;

settings = Office.context.roamingSettings;

document.getElementById("senderDisplayName").innerText = mailbox.item.sender.displayName;

document.getElementById("senderEmailAddress").innerText = mailbox.item.sender.emailAddress;

document.getElementById("recipientDisplayName").innerText = mailbox.item.to[0].displayName;

document.getElementById("recipientEmailAddress").innerText = mailbox.item.to[0].emailAddress;

});

};

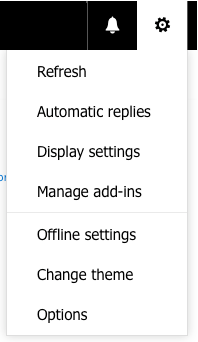
In this code we have implemented the initialize event handler that will be called in order to initialize the add-in instance for the current e-mail.

That’s all, and now it’s time to publish our add-in. Pay special attention that you need to publish add-ins to a place that supports https. The easiest way is publishing all files to Azure Web Apps. We already discussed this topic and it’s easy to do. But I assume that it’s possible to publish all files to public blob. Both things support https for free.

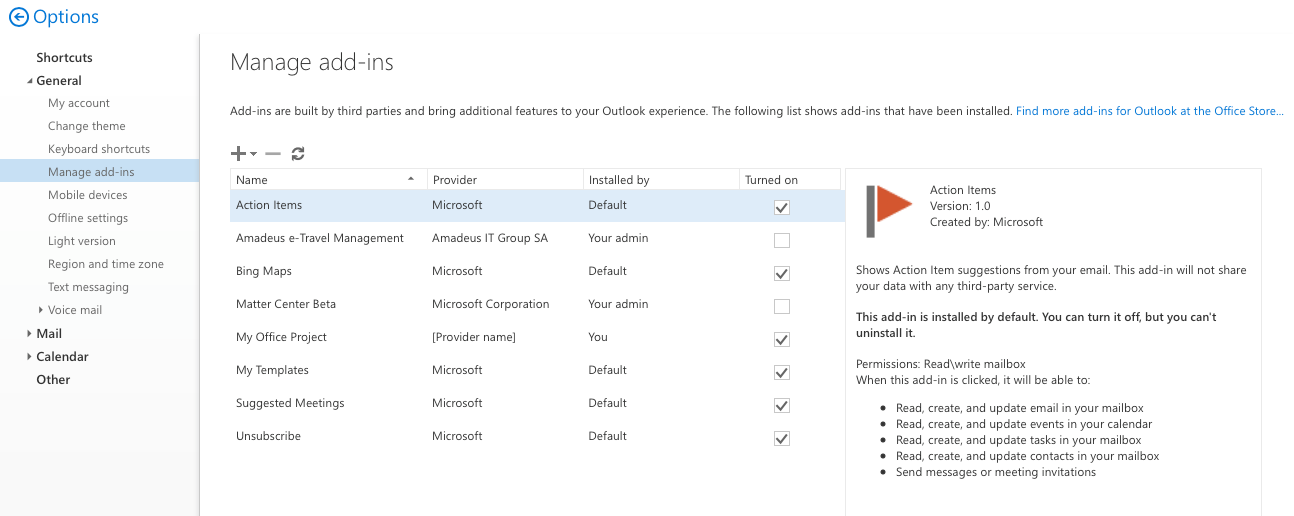
Once you know the location of the add-in, you need to open the manifest and fill source location:

<SourceLocation DefaultValue="**https**://officeaddins.azurewebsites.net/default.htm"/>

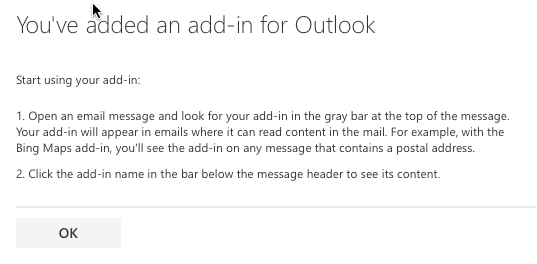
Once all files are published to Azure, you can open Outlook Web interface and click Settings to manage Add-ins:



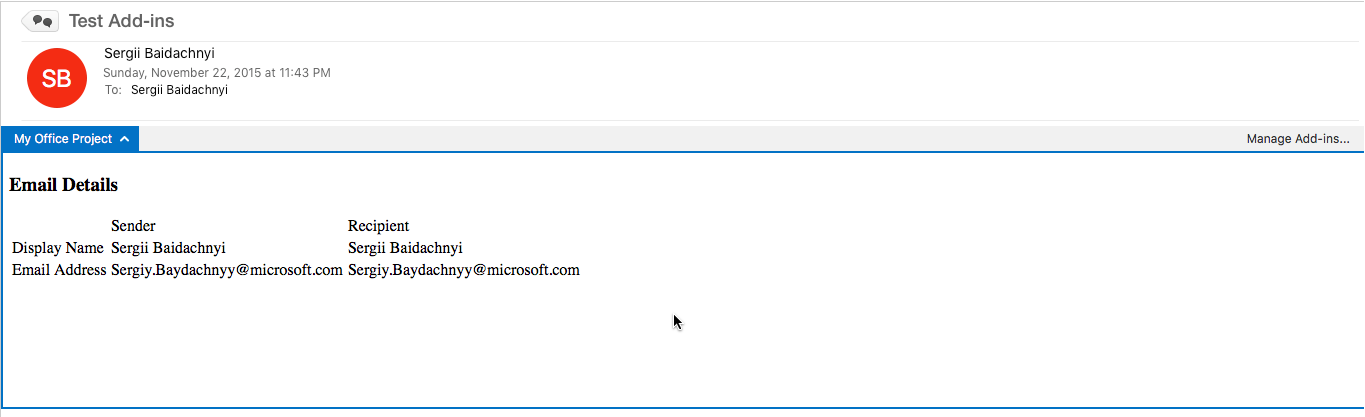
Manage add-ins command will navigate you to a window there you can install your own add-in:



Simply click Add button and provide a link to your manifest file (don’t forget to use https). If everything is OK, you will see the following message:



It’s time to test the add-in. Simply open your Outlook, click any message and you will see a gray bar at the top of the message. There you can find your add-in – click it in order to execute it:



Pay special attention that your add-in works fine in Outlook Web App and in Outlook for Windows as well.

Visit MSDN where you can find lots of examples around Office add-ins development. So, if you have experience in JavaScript, it’s easy to bring it to Office platform.