Office アドインの概要と開発

Microsoft MVP for Office Development
@kinuasa

Ver.20191112

＊　改訂履歴

バージョン	改訂内容
20181019	未完成版暫定公開
20181022	第 4 章、第 5 章追加
20181023	第 6 章追加
20181026	第 7 章追加
20181029	第 2 章(2-2　手順 7.)修正
20191101	参考 Web サイト修正、参考書籍追加
20191111	第 8 章追加
20191112	第 8 章(8-3、8-4、8-5)修正

＊ 注意事項

1. 本稿は Windows デスクトップ版の Excel を主な対象としており、Outlook 上で動作するアドインについては取り扱っておりません。
2. 本稿記載の説明やコードは、Windows 10 Pro 64 ビット版＋Office 365 ProPlus バージョン 1803 環境を元に検証を行っており、他のバージョンでは、アプリケーション上の表記や操作が異なる可能性があります。
3. 本稿では、煩雑なコードの記述を避けるために jQuery を使用しています。
4. 本資料、及びコードは【 https://github.com/kinuasa/OfficeAddInsHandsOn 】にアップしてありますので、必要に応じてコピーしてお使いください。
5. オンライン版の Office のことを「Web 用 Microsoft Office」「Office on the web」を表記することがありますが、本稿では「Office Online」表記で統一します。

内容

1. Office アドインの概要 1
1.1. Office アドインとは？ 1
1.2. Office アドインの構成要素 1
1.3. Office アドインの利点 2
1.4. Office アドインの種類 2
2. はじめての Office アドイン開発 3
2.1. 使用するアプリケーション 3
2.1.1. XAMPP Portable のインストール 3
2.1.2. デジタル証明書のインポート 9
2.2. 共有フォルダカタログの準備(サイドロード) 13
2.3. マニフェストファイルの作成 16
2.4. アドイン本体となる Web ページの作成 20
2.5. アドインの動作確認 22
2.6. コードの解説 23
2.7. Office 2013 の場合 24
2.8. Office アドインのデバッグ方法 25
3. アドイン コマンド 27
3.1. アドイン コマンドのマニフェストファイル 27
3.2. 作業ウィンドウ用、および関数実行用の Web ページの作成 32
3.3. アドイン コマンドの動作確認 34
4. YO OFFICE!(Yeoman)によるひな形の作成 37
4.1. Node.js のインストール 37
4.2. YO OFFICE!のインストール 40
4.3. YO OFFICE!による Office アドインのひな形作成 40
4.4. ローカルサーバーの起動と自己署名証明書の追加 42
4.5. Office アドインの動作確認 50
5. Script Lab の紹介 52
6. Excel カスタム関数(Excel Custom Functions) 56
6.1. カスタム関数のマニフェストファイル 56
6.2. カスタム関数を定義する JSON ファイル 58
6.3. カスタム関数の本体となるスクリプトファイル 59
6.4. カスタム関数の HTML ファイル 60
6.5. カスタム関数の動作確認 61
7. Office Online によるサイドロードと SharePoint カタログへの発行 63
7.1. 動作確認用マニフェストファイル 63

7.2. Office Online によるサイドロード 63
7.3. SharePoint カタログへの Office アドインの発行 65
8. Visual Studio Code による Office アドイン開発 70
8.1. Visual Studio Code のインストール 70
8.2. Visual Studio Code の日本語化 72
8.3. インテリセンスの有効化(npm 使用) 73
8.4. インテリセンスの有効化(npm 未使用) 74
8.5. Visual Studio Code＆YO OFFICE!＆Office Online による Office アドイン開発 75
9. おわりに 81
10. 参考 Web サイト 82
11. 参考書籍 85

1. Office アドインの概要

1.1. Office アドインとは？

OneDrive に文書を保存する、動画共有サービスの動画を文書に埋め込む、Office 2013 以降 Microsoft Office はクラウドサービスとの連携が当たり前になりました。

「**Office アドイン**」(旧名：Office 用アプリ)も Office 2013 で追加された Web 連携機能の一つで、HTML や CSS、JavaScript などの Web 標準技術を使って、Word や Excel、PowerPoint といった Office アプリケーションの機能を拡張することができます。

例えば、Word や Excel の校閲タブにある「翻訳」を使うと、画面右側にある作業ウィンドウから、文書内の文字列を Microsoft Translator でシームレスに翻訳できますが、この機能の実体は Office アドインです。

1.2. Office アドインの構成要素

Office アドインの実体は、ザックリと言ってしまうと「**Office アプリケーション上で動作する Web アプリ**」です。その Web アプリの中で、Microsoft が提供している専用の JavaScript ライブラリ(JavaScript API for Office ライブラリ、Office.js)を読み込むことで、Office 文書へ文字列の読み書きを行ったり、図を挿入したり、といったことができるようになっています。

Office アドインは、本体となる Web アプリと、アプリの各種情報が記載されたマニフェストファイル(XML)の 2 つで構成されます。Web アプリの場所や ID、説明やバージョンといった、必要な情報が記載されたマニフェストファイルを読み込むことで、Word や Excel、PowerPoint などの Office アプリケーション上で、サーバーにある Web アプリが実行されます。

1.3. Office アドインの利点

先述の通り、Office アドインの実体は Web アプリです。すなわち、HTML や JavaScript といった Web 標準技術で開発できるため、専用の技術を習得する必要がありません。

従来の Office 開発技術である VBA や VSTO と違って、プラットフォームに依存しない、クロスプラットフォームである点も大きな特徴です。Windows 用、Mac 用、スマートフォン用、Office Online、Office の環境は様々ですが、Office アドインはそれらすべてで実行可能です。もちろん、プラットフォームによって実行できる機能に違いはありますが。

また、Office アドインには一元的に管理・展開する仕組みが用意されているため、組織内で使用するアドインを端末ごとに配布する手間が掛かりません。さらに、Microsoft が用意しているプラットフォーム「AppSource」(旧 Office ストア)にアドインを提出すれば、開発したアドインを国内だけではなく、AppSource が対応している国外に向けても公開することができます。

1.4. Office アドインの種類

Office アドインは大別すると、下記の 3 種類に分けられます。

1. 作業ウィンドウ アドイン：Word、Excel、PowerPoint、Outlook といった、Office アプリケーションの右側のウィンドウ(作業ウィンドウ)で動作するアドインです。下記コンテンツ アドインのように、機能を Office 文書に直接埋め込む必要が無い場合は、作業ウィンドウ アドインを使用します。
2. コンテンツ アドイン：Word、Excel、PowerPoint 文書に直接埋め込む形で動作するアドインです。
3. Outlook アドイン(メール アドイン)：Outlook や Outlook.com 上で動作するアドインです。本稿では取り扱いません。

作業ウィンドウ アドイン画面

コンテンツ アドイン画面

2. はじめての Office アドイン開発

これまで述べてきた通り、Office アドインの実体は Web アプリであるため、最低限メモ帳や TeraPad(https://tera-net.com/)、Visual Studio Code(https://code.visualstudio.com/)といったテキストエディタと、Web サーバーさえあればアドインを開発することができます。

もちろん、Microsoft が提供している統合開発環境「Visual Studio」でも開発することができますが、本項では、Windows に標準搭載されている「**メモ帳**」を使って Office アドインを作る方法について解説します。

2.1. 使用するアプリケーション

最初の Office アドインを作成するにあたり、下記のアプリケーションを使用することにします。

1. テキストエディタ：メモ帳
2. Web サーバー：Apache(XAMPP Portable)

Web サーバーには、容易に導入でき、環境も汚さない Portable 版の XAMPP(Apache(Web サーバー)や MariaDB(データベース)といった、Web アプリの実行に必要なアプリケーションをまとめたパッケージ)を選定しました。

2.1.1. XAMPP Portable のインストール

下記手順で Portable 版の XAMPP をインストールします。

1. XAMPP のサイト(https://www.apachefriends.org/jp/)にアクセスします。	
2. ページ上部にある「**ダウンロード**」をクリックします。	

3. 「**Windows 向け XAMPP**」から「**その他のダウンロード**」をクリックします。

4. 「**XAMPP Windows**」をクリックします。

5. 最も新しいバージョン番号(本稿執筆時点では 7.2.9)をクリックします。

6. 「**xampp-portable-win32-(バージョン番号)-VC15.zip**」をクリックして ZIP ファイルをダウンロードします。7z 形式に対応したファイルアーカイバーをお使いの場合は 7z 形式でも問題ありません。

7. 自動的にファイルのダウンロードが始まるので、任意の場所にファイルを保存します。

8. 手順 7.で保存した ZIP ファイル内にある「**xampp**」フォルダを「**C:¥**」に展開します。

※ 以下、XAMPP のインストールフォルダを「C:¥xampp」として話を進めます。

9. XAMPP のインストール先フォルダにある「**setup_xampp.bat**」ファイルを実行し、環境設定を行います。「続行するには何かキーを押してください」とのメッセージが表示されたら、Enter キーを押して画面を閉じます。

10. XAMPP のインストール先フォルダにある「**xampp-control.exe**」を実行します。「現在、SmartScreen を使用できません」画面が表示された場合は、「**実行**」ボタンをクリックします。

11. 言語選択画面が表示されるので、「**英語**」を選択し、「**Save**」ボタンをクリックします。	
12. XAMPP のコントロールパネルが立ち上がるので、Apache の横にある「**Start**」ボタンをクリックします。	
13. Windows Defender ファイアウォールによるブロック画面が表示された場合は、「**アクセスを許可する**」ボタンをクリックします。	

14. 「Apache」の背景色が緑色になっていることを確認し、「**Admin**」ボタンをクリックします。ブラウザーが立ち上がり、Apache のダッシュボード画面が表示されれば、無事に Web サーバーが動作しています。

15. XAMPP を終了する際は、「**Stop**」ボタンをクリックして Apache を終了した後、「**Quit**」ボタンをクリックしてください。

以上で XAMPP のインストール作業は完了です。

2.1.2. デジタル証明書のインポート

Office アドインは、仕様上、作成したアドインを https サーバー上に配置する必要があります。XAMPP は標準で SSL が有効になっていますが、そのままだと https サイトを表示した際に警告画面が表示されてしまうため、証明書のインポート作業を行う必要があります。

1. XAMPP のコントロールパネルから Apache を起動します。	
2. Internet Explorer で【 https://localhost/ 】を開きます。「このサイトは安全ではありません」警告画面が表示されるので、「**詳細情報**」から「**Web ページに移動(非推奨)**」をクリックします。	
3. アドレスバーにある「**証明書のエラー**」から「**証明書の表示**」をクリックします。	

4. 証明書画面が表示されるので、「**証明書のインストール**」ボタンをクリックします。

5. 証明書のインポート ウィザード画面が表示されるので、「**現在のユーザー**」を選択し、「**次へ**」ボタンをクリックします。

6. 「**証明書をすべて次のストアに配置する**」を選択し、「**参照**」ボタンをクリックします。	
7. 「**信頼されたルート証明書機関**」を選択し、「**OK**」ボタンをクリックします。	

8. 証明書ストアが「信頼されたルート証明書機関」になっていることを確認し、「**次へ**」ボタンをクリックします。	
9. 「**完了**」ボタンをクリックします。	

10. セキュリティ警告ダイアログが表示されたら、「**はい**」ボタンをクリックして、証明書をインストールします。	
11. 「正しくインポートされました」メッセージが表示されたら、証明書のインポート作業は完了です。Internet Explorer を一度終了し、再度【 https://localhost/ 】を開くと、警告画面が表示されないことを確認できます。	

2.2. 共有フォルダカタログの準備(サイドロード)

SharePoint カタログや AppSource、ネットワーク共有(共有フォルダカタログ)等、Office アドインには様々な展開方法が用意されています。今回はローカル環境で簡単にアドインのテストを行えるよう、共有フォルダを使ってマニフェストファイルを配置することにします。

1.	新規フォルダ(**C:¥Share**)を作成します。
2.	1.のフォルダのプロパティから「**共有**」タブを開きます。

3. 「**詳細な共有**」ボタンをクリックします。	
4. 「**このフォルダーを共有する**」にチェックを入れ、同時に共有できるユーザー数を設定します。	

5. 「アクセス許可」ボタンをクリックし、**Everyone にフルコントロール許可**を与えます。

6. 「OK」ボタンをクリックして、詳細な共有ダイアログを閉じます。

7. 「**ネットワーク パス**」をコピーした後、プロパティ画面を閉じます。

8. Excel(もしくは Word)を立ち上げ、オプション画面を開きます。

9. 「セキュリティ センター」から「**セキュリティ センターの設定**」ボタンをクリックします。

10. 「信頼できるアドイン カタログ」から「カタログの URL」欄に、手順 7.でコピーした**ネットワーク パス**(もしくは**¥¥localhost¥Share**)を入力し、「**カタログの追加**」ボタンをクリックします。	
11. 「**メニューに追加する**」にチェックを入れ、「OK」ボタンをクリックします。	
12. 「設定を保存しました。次回 Office を起動したときに適用されます。」メッセージボックスが表示されるので、「OK」ボタンをクリックします。	
13. 「OK」ボタンをクリックしてオプション画面を閉じ、一度アプリケーションを終了します。	

以降、設定した共有フォルダにマニフェストファイルを保存すると、Office アプリケーションからアドインとして読み込めるようになります。

2.3. マニフェストファイルの作成

共有フォルダの準備ができたので、次はアドインの設定を記述するマニフェストファイル(XML)を作成します。下記コードをメモ帳に貼り付け、XML ファイルとして上記手順で作成した共有フォルダに保存します(文字コード：**UTF-8**)。

＊ manifest_2-3.xml(文字コード：UTF-8)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="TaskPaneApp">
3    <Id>{GUID}</Id>
4    <Version>1.0</Version>
5    <ProviderName>{アドイン提供者}</ProviderName>
```

```
6   <DefaultLocale>ja-jp</DefaultLocale>
7   <DisplayName DefaultValue="{Office アドインの名前}" />
8   <Description DefaultValue="{Office アドインの説明}"/>
9   <Hosts>
10    <Host Name="Workbook" />
11  </Hosts>
12  <DefaultSettings>
13    <SourceLocation DefaultValue="https://localhost/OfficeAddInsHandsOn/src/2-3/index.html" />
14  </DefaultSettings>
15  <Permissions>ReadWriteDocument</Permissions>
16 </OfficeApp>
```

XML の要素と属性の概要は下表の通りで、Id 要素、ProviderName 要素、DisplayName 要素の DefaultValue 属性、Description 要素の DefaultValue 属性の値をそれぞれ書き換えます。

要素	説明
OfficeApp	Office アドインのマニフェストのルート要素です。「xsi:type」属性でアドインの種類を指定します。 1. TaskPaneApp：作業ウィンドウ アドイン 2. ContentApp：コンテンツ アドイン 3. MailApp：Outlook アドイン
Id	一意の ID を GUID で指定します。GUID の生成には、Visual Studio 付属の「GUIDGEN.EXE」やオンラインの GUID ジェネレーター(https://www.guidgen.com/ 等)、あるいは次ページにあるようなスクリプトを使用することができます。
Version	Office アドインのバージョンを指定します。
ProviderName	Office アドインの提供者名を 125 文字以内の文字列で指定します。
DefaultLocale	既定のロケール(地域)を「ja-JP」等の言語タグで指定します。
DisplayName	Office アドインの名前を 125 文字以内の文字列で指定します。
Description	Office アドインの説明を 250 文字以内の文字列で指定します。
Hosts	クライアントアプリケーションを指定する「Host」要素を含みます。
Host	「Name」属性で Office アドインが動作するクライアントアプリケーションを指定します。 1. Document：Word 2. Workbook：Excel 3. Presentation：PowerPoint 4. Database：Access 5. Mailbox：Outlook 6. Notebook：OneNote 7. Project：Project
DefaultSettings	作業ウィンドウ アドイン、またはコンテンツ アドインの、既定のソースの場所や設定を指定します。
SourceLocation	「DefaultValue」属性で Office アドイン本体となる Web ページの場所を指定します(1~2018 文字)。ソースファイルは「https」アドレスである必要があります。
Permissions	Office アドインのアクセス許可レベルを指定します。アドインが正しく機能するために必要最小限のレベルを指定するのが望ましく、たとえば、ドキュメント内の値を読み込むだけで良いのであれば、「ReadDocument」レベルで十分です。

	1. Restricted : コンテンツ アドインまたは作業ウィンドウ アドインで要求することができる、最小のアクセス許可レベルです。 2. ReadDocument : Restricted アクセス許可によって使用可能となる API に加え、ドキュメントの読み取りとバインディングの管理に必要な API メンバーへのアクセス権を追加します。 3. ReadAllDocument : Restricted、ReadDocument アクセス許可によって使用可能となる API に加え、OOXML にアクセスするためのメソッドへのアクセスも許可されます。 4. WriteDocument : Restricted アクセス許可によって使用可能となる API に加え、ドキュメントに書き込むためのメソッドへのアクセスも許可されます。 5. ReadWriteDocument : ReadAllDocument、WriteDocument アクセス許可を含む、最上位のアクセス許可レベルです。

＊ GUID を生成する VBScript コード(GUIDGEN.vbs)

```
Option Explicit

Dim ret

Do
  ret = InputBox("Please copy a generated GUID." & vbNewLine & _
                 "You can then press Enter to regenerate GUID." & vbNewLine & vbNewLine & _
                 "Click the Cancel button or press ESC to finish.", _
                 "GUID Generator", _
                 LCase(Mid(CreateObject("Scriptlet.TypeLib").GUID, 2, 36)))
Loop Until Len(Trim(ret)) < 1
```

＊ 必要な要素・属性の値を書き換えたマニフェストファイル

manifest_2-3.xml - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
<?xml version="1.0" encoding="UTF-8"?>
<OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="TaskPaneApp">
  <Id>d6fc035d-de41-4669-b680-c465c3278522</Id>
  <Version>1.0</Version>
  <ProviderName>@kinuasa</ProviderName>
  <DefaultLocale>ja-jp</DefaultLocale>
  <DisplayName DefaultValue="最初のOffice アドイン" />
  <Description DefaultValue="テスト用作業ウィンドウ アドインです。"/>
  <Hosts>
    <Host Name="Workbook" />
  </Hosts>
  <DefaultSettings>
    <SourceLocation DefaultValue="https://localhost/OfficeAddInsHandsOn/src/2-3/index.html" />
  </DefaultSettings>
  <Permissions>ReadWriteDocument</Permissions>
</OfficeApp>
```

2.4. アドイン本体となる Web ページの作成

マニフェストファイルの準備ができたので、最後にアドイン本体となる Web ページを作成します。下記コードをメモ帳に貼り付け、Apache の公開フォルダに HTML 形式で保存(C:¥xampp¥htdocs¥OfficeAddInsHandsOn¥src¥2-3¥index.html、フォルダが無い場合は作成)します。

＊ index.html

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8" />
5      <meta http-equiv="X-UA-Compatible" content="IE=Edge" />
6      <title>Sample Office Add-ins</title>
7      <script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.3.1.min.js"></script>
8      <script src="https://appsforoffice.microsoft.com/lib/1/hosted/office.js"></script>
9      <script>
10       Office.onReady(function(){
11         //Office is ready
12         $(document).ready(function(){
13           //The document is ready
14           $("#run").click(run);
```

```
15          });
16        });
17
18        function run() {
19          Excel.run(function(context) {
20            var sheet = context.workbook.worksheets.getActiveWorksheet();
21            var range = sheet.getRange("A1");
22            range.values = [["Hello World."]];
23            return context.sync();
24          }).catch(function(error) {
25            console.log("Error: " + error);
26          });
27        }
28      </script>
29    </head>
30    <body>
31      <button id="run">OK</button>
32    </body>
33  </html>
```

index.html - メモ帳

ファイル(F)　編集(E)　書式(O)　表示(V)　ヘルプ(H)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=Edge" />
    <title>Sample Office Add-ins</title>
    <script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.3.1.min.js"></script>
    <script src="https://appsforoffice.microsoft.com/lib/1/hosted/office.js"></script>
    <script>
      Office.onReady(function(){
        //Office is ready
        $(document).ready(function(){
          //The document is ready
          $("#run").click(run);
        });
      });

      function run(){
        Excel.run(function(context){
          var sheet = context.workbook.worksheets.getActiveWorksheet();
          var range = sheet.getRange("A1");
          range.values = [["Hello World."]];
          return context.sync();
        }).catch(function(error){
          console.log("Error: " + error);
        });
      }
    </script>
  </head>
  <body>
    <button id="run">OK</button>
  </body>
</html>
```

2.5. アドインの動作確認

マニフェストファイル、アドイン本体となる Web ページの準備ができたので、動作確認を行います。

1. Excel を起動します。	
2. 「挿入」タブから「**個人用アドイン**」ボタンをクリックします。	
3. Office アドイン画面が表示されるので、「**共有フォルダー**」を選択します。	
4. 挿入する Office アドインを選択し、「**追加**」ボタンをクリックします。	
5. 画面右側の作業ウィンドウに Office アドインが表示されます。	

6. 「OK」ボタンをクリックすると、選択中のシートのセル A1 に「Hello World.」と入力されます。	

2.6. コードの解説

```
1  Office.onReady(function(){
2    //Office is ready
3    $(document).ready(function(){
4      //The document is ready
5      $("#run").click(run);
6    });
7  });
```

Office.onReady()は、Office.js ライブラリが完全に読み込まれているかどうかを確認しながら Promise オブジェクトを返す非同期のメソッドです。また、下記のようにハンドラーを割り当てる従来の方法「Office.initialize」もまだサポートされています。

```
1  Office.initialize = function(reason){
2    $(document).ready(function(){
3      $("#run").click(run);
4    });
5  };
```

Office.initialize イベントは、ライブラリが読み込まれ、アドインが、アプリケーションやホストされたドキュメントとの対話を開始する準備ができたときに発生します。initialize イベントリスナー関数の reason パラメーターは、初期化の発生方法を指定する Office.InitializationReason 列挙値を返します。

```
1  Excel.run(function(context){
2  .........
3  }).catch(function(error){
4  .........
5  });
```

Excel.run は、Excel オブジェクト モデルに対して行う処理を指定する関数を実行します。Excel オブジェクトと対話するための要求コンテキスト(context)を自動的に作成し、処理が完了すると、Promise が解決され、実行時に割り当てられたすべてのオブジェクトが自動的に解放されます。

```
1  var sheet = context.workbook.worksheets.getActiveWorksheet();
2  var range = sheet.getRange("A1");
3  range.values = [["Hello World."]];
4  return context.sync();
```

アドインで宣言して使用する Excel JavaScript オブジェクトは、プロキシオブジェクトです。呼び出すメソッドや、プロキシオブジェクトに設定・取得を行うプロパティは、単純に保留中のコマンドのキューとして追加されます。要求コンテキスト上で sync()メソッドを呼び出すと、キューに入れられたコマンドが、Excel に送られて実行されます。

Excel JavaScript API では、基本的にバッチ処理が中心となります。必要とする変更内容を、要求コンテキストのキューとして登録してから sync()メソッドを呼び出すと、キューに入れられたコマンドがバッチ的に実行されます。

たとえば上記コードでは、アクティブシートから A1 セルを取得し、取得したセルに「Hello World.」と入力する処理を行いますが、context.sync()メソッドが呼ばれるまで処理は反映されません。

sync()メソッドを呼び出すと、プロキシオブジェクトと Excel ドキュメント内のオブジェクトの状態が同期されます。sync()メソッドは、要求コンテキストのキューに登録されたすべてのコマンドを実行し、また、プロキシオブジェクトに読み込まれるプロパティの値を取得します。sync()メソッドは非同期で実行され、処理が完了すると解決される Promise を返します。

2.7. Office 2013 の場合

Office 2016 で導入された「**Excel JavaScript API**」では、ワークシートや Range、表やグラフ等にアクセスできる、Excel オブジェクトが用意されています。

残念ながら、Office 2013 ではこれら API が利用できないため、Word や Excel、PowerPoint などの複数の種類のホストアプリケーションに共通する「**Shared API**」を使って、Office ドキュメントにアクセスすることになります。

たとえば、Excel 2013 で上記コードのようにセルに値を入力する場合は、下記コードのように「setSelectedDataAsync」メソッドを使用します(下記コードの動作確認を行う場合は、2.3 項で作成したマニフェストファイルの、SourceLocation 要素、DefaultValue 属性の値を変更してください)。

＊ index_2013.html

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8" />
```

```
5     <meta http-equiv="X-UA-Compatible" content="IE=Edge" />
6     <title>Sample Office Add-ins</title>
7     <script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.3.1.min.js"></script>
8     <script src="https://appsforoffice.microsoft.com/lib/1/hosted/office.js"></script>
9     <script>
10      Office.initialize = function(reason){
11        $(document).ready(function(){
12          $("#run").click(run);
13        });
14      };
15
16      function run(){
17        Office.context.document.setSelectedDataAsync("Hello World!", function(asyncResu
lt){
18          if(asyncResult.status === Office.AsyncResultStatus.Failed){
19            console.log("Error: " + asyncResult.error.message);
20          }
21        });
22      }
23    </script>
24  </head>
25  <body>
26    <button id="run">OK</button>
27  </body>
28 </html>
```

2.8. Office アドインのデバッグ方法

Windows 10 に含まれている F12 開発者ツール(C:¥Windows¥System32¥F12¥**IEChooser.exe**)を使って、Office アドインをデバッグすることができます。

1. Excel を起動し、上記手順で作成した Office アドインを読み込みます。

2. F12 開発ツール(IEChooser.exe)を実行し､｢デバッグするターゲットの選択｣画面で Office アドインを指定します。	
3. 「デバッガー」タブの「**ドキュメントを開く**」(フォルダアイコン)から､デバッグするファイルを選択します。	
4. 処理と一時的に停止したい位置にブレークポイントを設定することができます。	
5. アドイン側で処理を実行すると、指定したブレークポイントで処理が停止します。	

詳しい使用方法については､｢Inspect running JavaScript with the Debugger｣や｢Microsoft Edge Developer Tools｣をご参照ください。

3. アドイン コマンド

Office アドインは「**アドイン コマンド**」呼ばれる機能によって、リボン上にボタンを配置したりして、UI を拡張することができます。UI 要素は、マニフェストファイルの VersionOverrides 要素以下で定義することになりますが、具体的な記述方法は下記のサンプルファイルをご参照ください。各要素の説明もコード内に記載しています。

3.1. アドイン コマンドのマニフェストファイル

一見すると冗長なコードですが、UI を拡張する際は、(1)アドイン コマンドに関する情報は「VersionOverrides」要素以下に格納、(2)コントロールの名前や説明、読み込むソースファイルの URL といったリソースの情報は「Resources」セクション(Resources 要素)に格納、この 2 点がポイントとなります。

＊ manifest_3.xml(文字コード：UTF-8)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <OfficeApp
3    xmlns="http://schemas.microsoft.com/office/appforoffice/1.1"
4    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5    xmlns:bt="http://schemas.microsoft.com/office/officeappbasictypes/1.0"
6    xmlns:ov="http://schemas.microsoft.com/office/taskpaneappversionoverrides"
7    xsi:type="TaskPaneApp">
8
9    <Id>{GUID}</Id>
10   <Version>1.0</Version>
11   <ProviderName>{アドイン提供者}</ProviderName>
12   <DefaultLocale>ja-JP</DefaultLocale>
13   <DisplayName DefaultValue="{Office アドインの名前}" />
14   <Description DefaultValue="{Office アドインの説明}" />
15   <IconUrl DefaultValue="https://localhost/OfficeAddInsHandsOn/src/3/img/icon-32.png" />
16   <HighResolutionIconUrl DefaultValue="https://localhost/OfficeAddInsHandsOn/src/3/img/icon-64.png"/>
17   <Hosts>
18     <Host Name="Workbook" />
19   </Hosts>
```

```
20  <DefaultSettings>
21    <SourceLocation DefaultValue="https://localhost/OfficeAddInsHandsOn/src/3/index.html" />
22  </DefaultSettings>
23  <Permissions>ReadWriteDocument</Permissions>
24
25  <!-- アドイン コマンド設定開始 -->
26  <VersionOverrides xmlns="http://schemas.microsoft.com/office/taskpaneappversionoverrides" xsi:type="VersionOverridesV1_0">
27    <Hosts>
28      <!-- ホスト指定 Excel:Workbook, Word:Document, PowerPoint:Presentation -->
29      <Host xsi:type="Workbook">
30        <!-- Form ファクター -->
31        <DesktopFormFactor>
32          <!-- 正常にアドインが読み込まれた際に表示可能なメッセージの設定 -->
33          <GetStarted>
34            <!-- メッセージのタイトル。「resid」属性で、Resources セクションの ShortStrings 要素にある有効な ID を参照 -->
35            <Title resid="Contoso.GetStarted.Title" />
36
37            <!-- メッセージの説明。「resid」属性で、Resources セクションの LongStrings 要素にある有効な ID を参照 -->
38            <Description resid="Contoso.GetStarted.Description" />
39
40            <!-- アドインの詳細を説明するページの URL。「resid」属性で、Resources セクションの Urls 要素にある有効な ID を参照 -->
41            <LearnMoreUrl resid="Contoso.GetStarted.LearnMoreUrl" />
42          </GetStarted>
43
44          <!-- UI を表示せずに実行する JavaScript 関数のソースコードファイルを指定。「resid」属性で、Resources セクションの Urls 要素にある有効な ID を参照 -->
45          <FunctionFile resid="Contoso.DesktopFunctionFile.Url" />
46
47          <!-- リボンの拡張。「PrimaryCommandSurface」は Office のリボン -->
48          <ExtensionPoint xsi:type="PrimaryCommandSurface">
49            <!-- 既定のリボンのタブを拡張。「id」属性で指定する値は【 https://docs.microsoft.com/ja-jp/office/dev/add-ins/reference/manifest/officetab 】参照 -->
```

```
50          <OfficeTab id="TabHome">
51            <!-- id 属性でグループの ID を一意の値で指定(最大 125 文字の文字列) -->
52            <Group id="Contoso.Group1">
53              <!-- グループ名。「resid」属性で、Resources セクションの ShortStrings 要素にある有効な ID を参照 -->
54              <Label resid="Contoso.Group1Label" />
55              <!-- ボタンやメニューのアイコンを定義。「resid」属性で、Resources セクションの Images 要素にある有効な ID を参照 -->
56              <!-- size 属性で画像のサイズをピクセル単位で指定。5 つのサイズ(20,24,40,48,64 ピクセル)がサポートされていますが、3 つのサイズ(16,32,80)を必ず指定 -->
57              <!-- PNG 形式のアイコンを使用。Resources セクションの URL はすべて https を使用する必要があります。 -->
58              <Icon>
59                <bt:Image size="16" resid="Contoso.tpicon_16x16" />
60                <bt:Image size="32" resid="Contoso.tpicon_32x32" />
61                <bt:Image size="80" resid="Contoso.tpicon_80x80" />
62              </Icon>
63
64              <!-- アクションを実行したり、作業ウィンドウを表示する JavaScript 関数を定義 -->
65              <!-- xsi:type 属性で[Button]もしくは[Menu]を指定 -->
66              <!-- id 属性でコントロールの ID を一意の値で指定(最大 125 文字の文字列) -->
67              <Control xsi:type="Button" id="Contoso.TaskpaneButton">
68                <!-- コントロール名。「resid」属性で、Resources セクションの ShortStrings 要素にある有効な ID を参照 -->
69                <Label resid="Contoso.TaskpaneButton.Label" />
70                <!-- コントロールのヒント -->
71                <Supertip>
72                  <!-- ヒントのタイトル。「resid」属性で、Resources セクションの ShortStrings 要素にある有効な ID を参照 -->
73                  <Title resid="Contoso.TaskpaneButton.Label" />
74                  <!-- ヒントの説明。「resid」属性で、Resources セクションの LongStrings 要素にある有効な ID を参照 -->
75                  <Description resid="Contoso.TaskpaneButton.Tooltip" />
76                </Supertip>
77                <Icon>
```

```
78                      <bt:Image size="16" resid="Contoso.tpicon_16x16" />
79                      <bt:Image size="32" resid="Contoso.tpicon_32x32" />
80                      <bt:Image size="80" resid="Contoso.tpicon_80x80" />
81                    </Icon>
82                    <!-- ユーザーがボタンやメニューを選択したときに実行する操作を指定 -->
83                    <!-- xsi:type 属性で「ShowTaskpane」または「ExecuteFunction」を指定します。 -->
84                    <Action xsi:type="ShowTaskpane">
85                      <TaskpaneId>ButtonId1</TaskpaneId>
86                      <!-- xsi:type 属性の値が「ShowTaskpane」のときに必ず指定する要素 -->
87                      <!-- この操作のソースコードファイルを指定。「resid」属性で、Resources セクションの Urls 要素にある有効な ID を参照 -->
88                      <SourceLocation resid="Contoso.Taskpane.Url" />
89                    </Action>
90                  </Control>
91
92                  <Control xsi:type="Button" id="Contoso.FunctionButton">
93                    <Label resid="Contoso.FunctionButton.Label" />
94                    <Supertip>
95                      <Title resid="Contoso.FunctionButton.Label" />
96                      <Description resid="Contoso.FunctionButton.Tooltip" />
97                    </Supertip>
98                    <Icon>
99                      <bt:Image size="16" resid="Contoso.tpicon_16x16" />
100                     <bt:Image size="32" resid="Contoso.tpicon_32x32" />
101                     <bt:Image size="80" resid="Contoso.tpicon_80x80" />
102                   </Icon>
103                   <Action xsi:type="ExecuteFunction">
104                     <!-- xsi:type 属性の値が「ExecuteFunction」のときに必ず指定する要素 -->
105                     <!-- FunctionFile 要素で指定したソースファイルに含まれる、実行する関数名を指定 -->
106                     <FunctionName>setValue</FunctionName>
107                   </Action>
108                 </Control>
109
110               </Group>
```

```
111             </OfficeTab>
112           </ExtensionPoint>
113         </DesktopFormFactor>
114       </Host>
115     </Hosts>
116
117     <!-- 各リソース設定 -->
118     <Resources>
119       <bt:Images>
120         <bt:Image id="Contoso.tpicon_16x16" DefaultValue="https://localhost/OfficeAddIn
sHandsOn/src/3/img/icon-16.png" />
121         <bt:Image id="Contoso.tpicon_32x32" DefaultValue="https://localhost/OfficeAddIn
sHandsOn/src/3/img/icon-32.png" />
122         <bt:Image id="Contoso.tpicon_80x80" DefaultValue="https://localhost/OfficeAddIn
sHandsOn/src/3/img/icon-80.png" />
123       </bt:Images>
124       <bt:Urls>
125         <bt:Url id="Contoso.Taskpane.Url" DefaultValue="https://localhost/OfficeAddInsH
andsOn/src/3/index.html" />
126         <bt:Url id="Contoso.GetStarted.LearnMoreUrl" DefaultValue="https://docs.microso
ft.com/ja-jp/office/dev/add-ins/design/add-in-commands" />
127         <bt:Url id="Contoso.DesktopFunctionFile.Url" DefaultValue="https://localhost/Of
ficeAddInsHandsOn/src/3/function.html" />
128       </bt:Urls>
129       <!-- ShortStrings : 最大 125 文字 -->
130       <bt:ShortStrings>
131         <bt:String id="Contoso.TaskpaneButton.Label" DefaultValue="作業ウィンドウ表示"
/>
132         <bt:String id="Contoso.FunctionButton.Label" DefaultValue="関数実行" />
133         <bt:String id="Contoso.Group1Label" DefaultValue="コマンド グループ" />
134         <bt:String id="Contoso.GetStarted.Title" DefaultValue="ようこそ、アドイン コマ
ンドのサンプルへ！" />
135       </bt:ShortStrings>
136       <!-- LongStrings : 最大 250 文字 -->
137       <bt:LongStrings>
138         <bt:String id="Contoso.TaskpaneButton.Tooltip" DefaultValue="クリックすると、作
業ウィンドウを表示します。" />
```

```
139         <bt:String id="Contoso.FunctionButton.Tooltip" DefaultValue="クリックすると、指定した関数を実行します。" />
140         <bt:String id="Contoso.GetStarted.Description" DefaultValue="アドインの読み込みに成功しました。" />
141       </bt:LongStrings>
142     </Resources>
143   </VersionOverrides>
144   <!-- アドイン コマンド設定終了 -->
145
146 </OfficeApp>
```

3.2. 作業ウィンドウ用、および関数実行用の Web ページの作成

上記マニフェストファイルで定義した、作業ウィンドウ用の Web ページ(index.html)、及び拡張したリボンのボタンから実行する関数用の Web ページを作成します。

＊ index.html

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8" />
5      <meta http-equiv="X-UA-Compatible" content="IE=Edge" />
6      <title>Sample Office Add-ins</title>
7      <script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.3.1.min.js"></script>
8      <script src="https://appsforoffice.microsoft.com/lib/1/hosted/office.js"></script>
9      <script>
10       Office.onReady(function(){
11         //Office is ready
12         $(document).ready(function(){
13           //The document is ready
14           $("#run").click(run);
15         });
16       });
17
18       function run(){
19         Excel.run(function(context){
20           var range = context.workbook.getActiveCell();
21           range.load("text"); //text プロパティ読み込み
```

```
22            return context.sync().then(function(){
23              $("#results").val(range.text);
24            });
25          }).catch(function(error){
26            console.log("Error: " + error);
27          });
28        }
29      </script>
30    </head>
31    <body>
32      <button id="run">OK</button>
33      <textarea id="results" cols="40" rows="4"></textarea>
34    </body>
35  </html>
```

＊ function.html

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="UTF-8" />
5       <meta http-equiv="X-UA-Compatible" content="IE=Edge" />
6       <script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.3.1.min.js"></script>
7       <script src="https://appsforoffice.microsoft.com/lib/1/hosted/office.js"></script>
8       <script>
9         Office.onReady(function(){
10          //Office is ready
11        });
12
13        function setValue(){
14          Excel.run(function(context){
15            var range = context.workbook.getActiveCell();
16            range.values = [["TEST"]];
17            return context.sync();
18          }).catch(function(error){
19            console.log("Error: " + error);
20          });
21        }
```

```
    </script>
  </head>
  <body></body>
</html>
```

3.3. アドイン コマンドの動作確認

マニフェストファイルと、アドイン本体となる Web ページの準備ができたので、動作確認を行います。

1. マニフェストファイル(manifest_3.xml)の Id 要素、ProviderName 要素、DisplayName 要素の DefaultValue 属性、Description 要素の DefaultValue 属性の値をそれぞれ書き換え、共有フォルダ(C:¥Share)に保存します。
2. アドイン本体となる Web ページを、Apache の公開フォルダ(C:¥xampp¥htdocs¥OfficeAddInsHandsOn¥src¥3)に保存し、Apache を起動します。
3. Excel を起動します。
4. 「挿入」タブから「**個人用アドイン**」ボタンをクリックします。

5. Office アドイン画面が表示されるので、「**共有フォルダー**」を選択します。

6. 挿入する Office アドインを選択し、「**追加**」ボタンをクリックします。

7. 「ホーム」タブにマニフェストファイルで定義した「コマンド グループ」が追加され、読み込み成功時のメッセージが表示されます。	
8. 追加されたボタン上にカーソルを持っていくと、マニフェストファイルで定義したヒントが表示されます。	
9. 「**関数実行**」ボタンをクリックすると、FunctionName 要素で指定した「**setValue**」関数が呼び出され、選択中のセルに「TEST」と入力されます。	

10. 「**作業ウィンドウ表示**」ボタンをクリックすると、マニフェストファイルで指定したページ(index.html)が作業ウィンドウとして表示されます。	
11. 作業ウィンドウ上の「**OK**」ボタンをクリックすると、選択中のセルの値がテキストエリアに表示されます。	

以上で、アドイン コマンドの動作確認ができました。

4. YO OFFICE!(Yeoman)によるひな形の作成

Office アドインには、マニフェストファイルを含めた、アドインのひな形を生成するためのツール「YO OFFICE!」(Yeoman)が用意されています。このツールを使うと、任意のテキストエディタを使って簡単にアドイン開発を行うことができます。

4.1. Node.js のインストール

YO OFFICE!は、Node.js のパッケージ管理マネージャである npm で導入するので、まずは Node.js をインストールする必要があります(npm は Node.js と一緒にインストールされます)。

1. Node.js の公式サイト(https://nodejs.org/ja/)にアクセスします。	
2. ダウンロード画面から「**推奨版**」のインストーラーをダウンロードします。	
3. 手順 2.でダウンロードしたインストーラーを実行し、指示に従ってインストールを行います。インストール時のオプションは特に変更する必要はありません。	

Node.js Setup
End-User License Agreement
Please read the following license agreement carefully
Node.js is licensed for use as follows:
Copyright Node.js contributors. All rights reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject
I accept the terms in the License Agreement
Print
Back
Next
Cancel

Node.js Setup
Destination Folder
Choose a custom location or click Next to install.
Install Node.js to:
C:¥Program Files¥nodejs¥
Change...
Back
Next
Cancel

Node.js Setup
Custom Setup
Select the way you want features to be installed.
Click the icons in the tree below to change the way features will be installed.
Node.js runtime
npm package manager
Online documentation shortcuts
Add to PATH
Install the core Node.js runtime (node.exe).
This feature requires 21MB on your hard drive. It has 2 of 2 subfeatures selected. The subfeatures require 20KB on your hard drive.
Browse...
Reset
Disk Usage
Back
Next
Cancel

4. インストール完了後、コマンドプロンプトから「**node -v**」「**npm -v**」コマンドを実行して、それぞれのバージョンが表示されれば、問題なくインストールが行われたことが確認できます。

4.2. YO OFFICE!のインストール

npm インストール後、コマンドプロンプトから下記コマンドを実行すれば、自動的に YO OFFICE!(Yeoman)がインストールされます。

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

4.3. YO OFFICE!による Office アドインのひな形作成

インストールが終わったら後は簡単です。YO OFFICE!を実行して質問に答えていくだけで、Office アドインのひな型ができあがります。

1. コマンドプロンプトを実行し、Office アドインのひな型を出力するフォルダ(例：C:¥Addins)に移動します。	
2. 「**yo office**」コマンドを実行すると、ひな型作成を手助けしてくれる紳士のアスキーアートが表示されます。 初回起動時には使用統計を匿名で送信しても良いかどうか、確認のメッセージが表示されますが、拒否する場合は「**n**」キーを押してください。	

3. プロジェクトの種類(Angular、jQuery、React、Excel カスタム関数、マニフェストファイルのみ)を選択します。今回は「**jQuery**」のプロジェクトを作成します。	
4. スクリプトの種類(TypeScript、JavaScript)を選択します。今回は「**JavaScript**」にします。	
5. アドイン名を入力します。	
6. アドインのホストとなる Office クライアントアプリケーション(Excel、OneNote、Outlook、PowerPoint、Project、Word)を選択します。今回は「**Excel**」にします。	

7. 最後の質問に答えると、手順 1.のフォルダ以下に Office アドインのひな型一式が出力されます(例：C:¥Addins¥My Sample Addin)。

※ 質問項目は YO OFFICE!のバージョンによって異なります。また、一部パッケージの作成には「Git」(https://git-scm.com/download)のインストールが必要となります。

4.4. ローカルサーバーの起動と自己署名証明書の追加

作成したひな型を使えば、すぐに Office アドインの開発ができるのですが、このままだとアドイン実行時に証明書エラーが表示されてしまうので、「**信頼されたルート証明機関**」に自己署名証明書を追加する必要があります。

1. コマンドプロンプトでひな型を出力したフォルダ(例：C:¥Addins¥My Sample Addin)に移動します。

2. 「**npm start**」コマンドを実行すると、ローカルサーバーが立ち上がります。このとき、他の Web サーバー(Apache)がすでに実行されている場合は、事前に終了しておきます。	
3. Internet Explorer で【 https://localhost:3000/ 】を開きます。「このサイトは安全ではありません」警告画面が表示されるので、「**詳細情報**」から「**Web ページに移動(非推奨)**」をクリックします。	
4. アドレスバーにある「**証明書のエラー**」から「**証明書の表示**」をクリックします。	

5. 証明書画面が表示されるので、「**証明書のインストール**」ボタンをクリックします。	
6. 証明書のインポート ウィザード画面が表示されるので、「**現在のユーザー**」を選択し、「**次へ**」ボタンをクリックします。	

7. 「**証明書をすべて次のストアに配置する**」を選択し、「**参照**」ボタンをクリックします。	
8. 「**信頼されたルート証明書機関**」を選択し、「**OK**」ボタンをクリックします。	
9. 証明書ストアが「**信頼されたルート証明書機関**」になっていることを確認し、「**次へ**」ボタンをクリックします。	

10. 「**完了**」ボタンをクリックします。	
11. セキュリティ警告ダイアログが表示されたら、「**はい**」ボタンをクリックして、証明書をインストールします。	
12. 「正しくインポートされました」メッセージが表示されたら、「**OK**」ボタンをクリックします。	

13. 再び Internet Explorer で【 https://localhost:3000/ 】を開くと、警告画面が表示されるので、手順 3.～4.と同様に証明書を表示します。

14. 「**証明書のパス**」タブを開くと、CA ルート証明書が信頼されていないことが確認できます。

15. CA ルート証明書(localhost-ca)を選択し、「**証明書の表示**」ボタンをクリックします。	
16. 手順 5.~12.と同様に、CA ルート証明書のインストールを行います。	

証明書のインポート ウィザード
証明書のインポート ウィザードの開始
このウィザードでは、証明書、証明書信頼リスト、および証明書失効リストをディスクから証明書ストアにコピーします。
証明機関によって発行された証明書は、ユーザー ID を確認し、データを保護したり、またはセキュリティで保護されたネットワーク接続を提供するための情報を含んでいます。証明書ストアは、証明書が保管されるシステム上の領域です。
保存場所
現在のユーザー(C)
ローカル コンピューター(L)
続行するには、[次へ] をクリックしてください。
次へ(N)
キャンセル

証明書のインポート ウィザード
証明書ストア
証明書ストアは、証明書が保管されるシステム上の領域です。
Windows に証明書ストアを自動的に選択させるか、証明書の場所を指定することができます。
証明書の種類に基づいて、自動的に証明書ストアを選択する(U)
証明書をすべて次のストアに配置する(P)
証明書ストア:
参照(R)...
次へ(N)
キャンセル

証明書ストアの選択
使用する証明書ストアを選択してください(C)
個人
信頼されたルート証明機関
中間証明機関
信頼された発行元
信頼されていない証明書
サード パーティ ルート証明機関
物理ストアを表示する(S)
OK
キャンセル

証明書のインポート ウィザード
証明書ストア
証明書ストアは、証明書が保管されるシステム上の領域です。
Windows に証明書ストアを自動的に選択させるか、証明書の場所を指定することができます。
証明書の種類に基づいて、自動的に証明書ストアを選択する(U)
証明書をすべて次のストアに配置する(P)
証明書ストア:
信頼されたルート証明機関
参照(R)...
次へ(N)
キャンセル

証明書のインポート ウィザード
証明書のインポート ウィザードの完了
[完了] をクリックすると、証明書がインポートされます。
次の設定が指定されました:
ユーザーが選択した証明書ストア
信頼されたルート証明機関
内容
証明書
完了(F)
キャンセル

セキュリティ警告
発行者が次であると主張する証明機関 (CA) から証明書をインストールしようとしています:
localhost-ca
証明書が実際に "localhost-ca" からのものであるかどうかを検証できません。"localhost-ca" に連絡して発行者を確認する必要があります。次の番号はこの過程で役立ちます:
拇印 (sha1): E9EA83D0 C10878C3 19287D17 91226D5E 379411FB
警告:
このルート証明書をインストールすると、この CA によって発行された証明書は自動的に信頼されます。確認されていない拇印付きの証明書をインストールすることは、セキュリティ上、危険です。[はい] をクリックすると、この危険を認識したことになります。
この証明書をインストールしますか?
はい(Y)
いいえ(N)

17. Internet Explorer で【 https://localhost:3000/ 】を開くと、警告画面が表示されないことを確認できます。	

4.5. Office アドインの動作確認

YO OFFICE!によって出力したひな型一式には、アドインのマニフェストファイル(例：C:¥Addins¥My Sample Addin¥manifest.xml)が含まれています。このマニフェストファイルを**“共有フォルダ”**にコピーすると、YO OFFICE!で指定した Office アプリケーション(Word や Excel 等)でアドインとして読み込むことができます。

また、ローカルサーバーを起動した状態(npm start)で、「**npm run sideload**」コマンドを実行することでも、Office アドインの動作確認を行うことができます。

以上のように、YO OFFICE!を使うことで、簡単に Office アドインの土台を作ることができます。そ

のままアドインを動かすこともできるので、Visual Studio を使わずに Office アドイン開発を始めるのに、YO OFFICE!は非常に役立つツールと言えます。

コマンドラインオプションなどの詳細な情報については、「Microsoft Office Add-in Project Generator - YO OFFICE!」(https://github.com/OfficeDev/generator-office)をご参照ください。

5. Script Lab の紹介

Microsoft Garage(Microsoft の実験的なプロジェクト)発の「**Script Lab**」(https://www.microsoft.com/en-us/garage/profiles/script-lab/)という Office アドインがあります。このアドインを使うことで、Office アドインの開発を簡単に体験することができます。

1. Office アプリケーション(Excel、Word、PowerPoint)を起動します。	
2. 「挿入」タブの「アドイン」グループから「**ストア**」をクリックします。	
3. 検索ボックスに「**script**」と入力して検索すると、Script Lab がヒットするので、「**追加**」ボタンをクリックします。	
4. アドインが読み込まれ、「**Script Lab**」タブが追加されます。	
5. 「Script Lab」タブの「Script」グループから「**Code**」ボタンをクリックします。	

6. 作業ウィンドウにアドインが表示され、「**Samples**」タブから様々なサンプルコードを表示できます。

7. 「▸」ボタンから「Run in this pane」をクリックすると、作業ウィンドウ上からサンプルコードを実行することができます。

8. コードは自分で編集することもでき、入力画面ではインテリセンス(コードの入力補完)も効きます。

<table>
<tr>
<td></td>
<td>
</td>
</tr>
<tr>
<td>9. 作業ウィンドウを「移動」することで、Office アドインのウィンドウを分離することもできます。</td>
<td>
</td>
</tr>
</table>

Script Lab アドインは、その他にコードスニペットのインポートやエクスポートを行うこともでき、簡単なプロトタイプを作成する際にも使える便利なツールです。詳細な使い方については、「Script Lab, a Microsoft Garage project」(https://github.com/OfficeDev/script-lab)をご参照ください。

6. Excel カスタム関数(Excel Custom Functions)

Office アドインには、Excel VBA で言うところの「ユーザー定義関数」のようにワークシート上から使える独自関数を、JavaScript で定義できる機能があります。それが、「**Excel カスタム関数(Excel Custom Functions)**」です。

カスタム関数は 2018 年 10 月時点ではまだ Preview 版で、実行するには Windows 版 Excel の場合、ビルド番号 10827 以降の環境が必要です(Insider 版 Office のインストール方法は「office 365 の一般法人向けユーザーは、新しい office 機能にいち早くアクセスする方法を説明します」(https://support.office.com/ja-jp/article/4dd8ba40-73c0-4468-b778-c7b744d03ead)参照)。また、Office Insider プログラム(https://products.office.com/office-insider)にも参加する必要があります。

カスタム関数を利用するには、マニフェストファイルの他に、カスタム関数を定義する JSON メタデータ、カスタム関数のコードを記述するスクリプトファイル、スクリプトファイルへの参照を定義する HTML ファイルの 3 つのファイルが必要となります。

6.1. カスタム関数のマニフェストファイル

カスタム関数のマニフェストファイルです。各要素の説明は、コード内に記載しています。

＊ manifest_6.xml(文字コード：UTF-8)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:bt="http://schemas.microsoft.com/office/officeappbasictypes/1.0" xmlns:ov="http://schemas.microsoft.com/office/taskpaneappversionoverrides" xsi:type="TaskPaneApp">
3    <Id>{GUID}</Id>
4    <Version>1.0</Version>
5    <ProviderName>{アドイン提供者}</ProviderName>
6    <DefaultLocale>ja-JP</DefaultLocale>
7    <DisplayName DefaultValue="{Office アドインの名前}" />
8    <Description DefaultValue="{Office アドインの説明}" />
9    <Hosts>
10     <Host Name="Workbook" />
11   </Hosts>
12   <DefaultSettings>
13     <SourceLocation DefaultValue="https://localhost/OfficeAddInsHandsOn/src/6/index.html" />
14   </DefaultSettings>
```

```
15  <Permissions>ReadWriteDocument</Permissions>
16  <VersionOverrides xmlns="http://schemas.microsoft.com/office/taskpaneappversionoverrides" xsi:type="VersionOverridesV1_0">
17    <Hosts>
18      <Host xsi:type="Workbook">
19        <AllFormFactors>
20          <!-- アドインが機能を公開する場所を定義。xsi:type 属性で「CustomFunctions」を指定 -->
21          <ExtensionPoint xsi:type="CustomFunctions">
22            <Script>
23              <!-- JavaScript のソースコードファイルを指定。「resid」属性で、Resources セクションの Urls 要素にある有効な ID を参照 -->
24              <SourceLocation resid="JS-URL" />
25            </Script>
26            <Page>
27              <!-- HTML ファイルを指定。「resid」属性で、Resources セクションの Urls 要素にある有効な ID を参照 -->
28              <SourceLocation resid="HTML-URL" />
29            </Page>
30            <Metadata>
31              <!-- JSON ファイルを指定。「resid」属性で、Resources セクションの Urls 要素にある有効な ID を参照 -->
32              <SourceLocation resid="JSON-URL" />
33            </Metadata>
34            <!-- 関数の名前の前に追加される名前空間を指定。「resid」属性で、Resources セクションの ShortStrings 要素にある有効な ID を参照 -->
35            <Namespace resid="namespace" />
36          </ExtensionPoint>
37        </AllFormFactors>
38      </Host>
39    </Hosts>
40    <Resources>
41      <bt:Urls>
42        <bt:Url id="JS-URL" DefaultValue="https://localhost/OfficeAddInsHandsOn/src/6/customfunctions.js" />
43        <bt:Url id="HTML-URL" DefaultValue="https://localhost/OfficeAddInsHandsOn/src/6/index.html" />
```

```
44          <bt:Url id="JSON-URL" DefaultValue="https://localhost/OfficeAddInsHandsOn/src/6/customfunctions.json" />
45        </bt:Urls>
46        <bt:ShortStrings>
47          <bt:String id="namespace" DefaultValue="CONTOSO" />
48        </bt:ShortStrings>
49      </Resources>
50    </VersionOverrides>
51  </OfficeApp>
```

6.2. カスタム関数を定義する JSON ファイル

カスタム関数を定義する JSON メタデータです。

＊ customfunctions.json

```
1  {
2    "$schema": "https://developer.microsoft.com/en-us/json-schemas/office-js/custom-functions.schema.json",
3    "functions": [
4      {
5        "id": "ADD", //関数の ID
6        "name": "ADD", //関数名。マニフェストファイルで指定した名前空間が接頭語として付されます。
7        "description": "2 つの数値を足した値を返します。", //関数の説明
8        "helpUrl": "https://www.ka-net.org/", //ヘルプページの URL
9        //関数によって返される情報の種類を定義
10       "result": {
11         "type": "number",
12         "dimensionality": "scalar"
13       },
14       //関数の入力パラメーターを定義
15       "parameters": [
16         {
17           "name": "first",
18           "description": "最初の値",
19           "type": "number",
20           "dimensionality": "scalar"
21         },
```

```
22          {
23            "name": "second",
24            "description": "2 番目の値",
25            "type": "number",
26            "dimensionality": "scalar"
27          }
28        ]
29      },
30      {
31        "id": "INCREMENT",
32        "name": "INCREMENT",
33        "description": "一定間隔で値が増えていきます。",
34        "helpUrl": "https://www.ka-net.org/",
35        "result": {
36            "type": "number",
37            "dimensionality": "scalar"
38      },
39      "parameters": [
40          {
41              "name": "increment",
42              "description": "増えていく値",
43              "type": "number",
44              "dimensionality": "scalar"
45          }
46      ],
47      "options": {
48          "cancelable": true,
49          "stream": true
50        }
51      }
52    ]
53  }
```

6.3. カスタム関数の本体となるスクリプトファイル

カスタム関数の本体となるスクリプトファイルです。

＊ customfunctions.js

```
1  function add(first, second){
2    return first + second;
3  }
4
5  function increment(incrementBy, callback) {
6    var result = 0;
7    var timer = setInterval(function() {
8      result += incrementBy;
9      callback.setResult(result);
10   }, 1000);
11
12   callback.onCanceled = function() {
13     clearInterval(timer);
14   };
15 }
16
17 //JSON メタデータで定義した「ADD」に関数「add」が対応するように指定
18 CustomFunctionMappings.ADD = add;
19 //JSON メタデータで定義した「INCREMENT」に関数「increment」が対応するように指定
20 CustomFunctionMappings.INCREMENT = increment;
```

6.4. カスタム関数の HTML ファイル

カスタム関数を定義する JavaScript ファイルを参照する HTML ファイルです。

＊ index.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="UTF-8" />
5    <meta http-equiv="X-UA-Compatible" content="IE=Edge" />
6    <meta http-equiv="Expires" content="0" />
7    <title>Custom Functions Sample</title>
8    <script src="https://appsforoffice.microsoft.com/lib/1/hosted/office.js"></script>
9    <script src="customfunctions.js"></script>
10   <!-- <script src="https://appsforoffice.microsoft.com/lib/beta/hosted/office.js"></sc
ript> -->
```

11	<!-- <script src="https://officedev.github.io/custom-functions/lib/custom-functions-runtime.js"></script> -->
12	</head>
13	<body></body>
14	</html>

6.5. カスタム関数の動作確認

1. マニフェストファイル(manifest_6.xml)の Id 要素、ProviderName 要素、DisplayName 要素の DefaultValue 属性、Description 要素の DefaultValue 属性の値をそれぞれ書き換え、共有フォルダ(C:¥Share)に保存します。
2. アドイン本体となる Web ページを、Apache の公開フォルダ(C:¥xampp¥htdocs¥OfficeAddInsHandsOn¥src¥6)に保存し、Apache を起動します。
3. Excel を起動します。
4. 「挿入」タブから「**個人用アドイン**」ボタンをクリックします。

5. Office アドイン画面が表示されるので、「**共有フォルダー**」を選択します。

6. 挿入する Office アドインを選択し、「**追加**」ボタンをクリックします。

7. マニフェストファイルで定義した名前空間(=CONTOSO)を入力すると、カスタム関数として定義した関数が表示され、関数を実行することができます。	

Office アドインを介して Cognitive Services(https://azure.microsoft.com/ja-jp/services/cognitive-services/)などの強力な API を利用でき、ユーザー側では SUM 関数や AVERAGE 関数といった標準関数と同様の操作性で簡単に扱うことができる、Excel カスタム関数は非常に強力な機能です。

ただし、先述の通り、2018 年 10 月時点ではまだ Preview 版となっていますので、今後仕様が大きく変わる可能性がある点については注意が必要です。

7. Office Online によるサイドロードと SharePoint カタログへの発行

これまで共有フォルダカタログを使用してアドインの動作確認を行ってきましたが、本項では、共有フォルダを使わない動作確認方法、および SharePoint を使った組織内へのアドインの配布方法について説明します。

7.1. 動作確認用マニフェストファイル

本項で使用するマニフェストファイルです。

＊ manifest_7.xml(文字コード：UTF-8)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="TaskPaneApp">
3    <Id>{GUID}</Id>
4    <Version>1.0</Version>
5    <ProviderName>{アドイン提供者}</ProviderName>
6    <DefaultLocale>ja-JP</DefaultLocale>
7    <DisplayName DefaultValue="{Office アドインの名前}" />
8    <Description DefaultValue="{Office アドインの説明}" />
9    <IconUrl DefaultValue="https://kinuasa.github.io/OfficeAddInsHandsOn/apps/img/icon-32.png" />
10   <HighResolutionIconUrl DefaultValue="https://kinuasa.github.io/OfficeAddInsHandsOn/apps/img/icon-64.png"/>
11   <Hosts>
12     <Host Name="Workbook" />
13   </Hosts>
14   <DefaultSettings>
15     <SourceLocation DefaultValue="https://kinuasa.github.io/OfficeAddInsHandsOn/apps/index.html" />
16   </DefaultSettings>
17   <Permissions>ReadWriteDocument</Permissions>
18 </OfficeApp>
```

7.2. Office Online によるサイドロード

Office アドインは、Office Online 上でマニフェストファイルをアップロードすることにより、アドインカタログにマニフェストファイルを配置しなくても、アドインの動作確認を行うことができます。

1. マニフェストファイル(manifest_7.xml)の Id 要素、ProviderName 要素、DisplayName 要素の De

faultValue 属性、Description 要素の DefaultValue 属性の値をそれぞれ書き換えます。	
2. Office Online(https://www.office.com/)にアクセスし、Excel ファイルを新規作成します。	
3. 「挿入」タブから「**Office アドイン**」ボタンをクリックします。	
4. Office アドイン画面が表示されるので、「**個人用アドイン**」を選択します。	
5. 「**個人用アドインの管理**」から「**マイ アドインのアップロード**」をクリックします。	

6. アドインのアップロード画面が表示されるので、手順 1.で保存したマニフェストファイルを選択し、「**アップロード**」ボタンをクリックします。	
7. アドインが読み込まれ、動作確認を行うことができます。	

アドイン本体の場所(SourceLocation 要素の DefaultValue 属性)をローカルサーバー(localhost)にした場合、ブラウザーの設定によってはエラーが発生してアドインを読み込むことができないため、Office Online を使って動作確認を行う場合は、https に対応した適当なホスティングサービス、あるいは「Git Hub Pages」(https://pages.github.com/)等にファイルをアップすることをお薦めします。

7.3. SharePoint カタログへの Office アドインの発行

本項では、Office 365 に含まれる SharePoint Online を元に、SharePoint カタログへの Office アドインの発行方法について説明します。

1. マニフェストファイル(manifest_7.xml)の Id 要素、ProviderName 要素、DisplayName 要素の DefaultValue 属性、Description 要素の DefaultValue 属性の値をそれぞれ書き換えます。

2. Office 365(https://www.office.com/)にサインインし、管理センターを開きます。

3. 「**管理センター**」 から「**SharePoint**」をクリックします。

4. SharePoint 管理センター画面が表示されるので、「**アプリ**」をクリックします。	
5. 「**アプリ カタログ**」をクリックします。	
6. 「**Office 用アプリ**」をクリックします。	

7. 「**アップロード**」をクリックします。	
8. ドキュメントの追加画面が表示されるので、手順 1.で保存したマニフェストファイルを選択し、「**OK**」ボタンをクリックします。	
9. アップロードが完了し、問題なくマニフェストファイルが読み込まれると、カタログにアドインが追加されます。	
10. Excel Online でファイルを新規作成し、「**Office アドイン**」を開きます(前項[7. 2.]参考)。	
11. 「**自分の所属組織**」を開くと、上記手順で追加したアドインが表示されていることが確認できます。	

以上の手順で、組織内のユーザーに対して Office アドインを配布することができます。

8. Visual Studio Code による Office アドイン開発

本項では、Microsoft が公開しているエディタ「**Visual Studio Code**」を使って、より効率的に Office アドイン開発を行う方法を説明します。

8.1. Visual Studio Code のインストール

1. Visual Studio Code のダウンロードサイト(https://code.visualstudio.com/download/)にアクセスします。

2. 「**Windows**」ボタンをクリックして、インストーラーをダウンロードします。

3. 手順 2.でダウンロードしたインストーラーを実行し、指示に従ってインストールを行います。

Microsoft Visual Studio Code (User) セットアップ
プログラムグループの指定
プログラムアイコンを作成する場所を指定してください。
セットアップはスタートメニューにプログラムのショートカットを作成します。
続けるには「次へ」をクリックしてください。違うディレクトリを選択するには「参照」をクリックしてください。
Visual Studio Code
参照(R)
プログラムグループを作成しない(D)
< 戻る(B)
次へ(N) >
キャンセル

Microsoft Visual Studio Code (User) セットアップ
追加タスクの選択
実行する追加タスクを選択してください。
Visual Studio Code インストール時に実行する追加タスクを選択して、「次へ」をクリックしてください。
アイコンを追加する:
デスクトップ上にアイコンを作成する(D)
その他:
エクスプローラーのファイル コンテキスト メニューに [Code で開く] アクションを追加する
エクスプローラーのディレクトリ コンテキスト メニューに [Code で開く] アクションを追加する
サポートされているファイルの種類のエディターとして、Code を登録する
PATH への追加(再起動後に使用可能)
< 戻る(B)
次へ(N) >
キャンセル

Microsoft Visual Studio Code (User) セットアップ
インストール準備完了
ご使用のコンピュータへ Visual Studio Code をインストールする準備ができました。
インストールを続行するには「インストール」を、設定の確認や変更を行うには「戻る」をクリックしてください。
インストール先:
C:¥Users¥ ¥AppData¥Local¥Programs¥Microsoft VS Code
プログラムグループ:
Visual Studio Code
追加タスク一覧:
アイコンを追加する:
デスクトップ上にアイコンを作成する(D)
その他:
PATH への追加(再起動後に使用可能)
< 戻る(B)
インストール(I)
キャンセル

8.2. Visual Studio Code の日本語化

Visual Studio Code のインストールが終わったら、使いやすいようにインターフェースを日本語化します。

1. Visual Studio Code を起動します。
2. 「**Extensions**」から「**japanese**」と検索し、上位でヒットする「**Japanese Language Pack for Visual Studio Code**」をインストールします。

3. インストール完了後、「**Restart Now**」ボタンをクリックして、Visual Studio Code を再起動します。	
4. 再起動後、メニューバー等のインターフェースが日本語になっていることが確認できます。	

8.3. インテリセンスの有効化(npm 使用)

効率よくコーディングを進めるため、インテリセンス機能を有効化する方法を説明します。本項では「**npm**」を使用するため、事前に Node.js をインストールしておいてください(4.1 項を参照)。

1. Visual Studio Code でフォルダを開き、ファイルの編集画面を開きます。	
2. 何も設定していない状態だと、「Offi…」と入力しても「Office」が表示されず、インテリセンスが効いていないことが確認できます。	
3. 「**ターミナル**」メニューから「**新しいターミナル**」をクリックします。	
4. 画面右下にターミナルが表示されるので、「**npm install @types/office-js**」コマンドを実行して、Office アドインの型定義ファイルをインストールします。	
5. インテリセンスが有効になっていることが確認できます。	

8.4. インテリセンスの有効化(npm 未使用)

npm を使用しなくても、Office アドインの型定義ファイルさえあればインテリセンスを有効にすることができます。ただし、実際に開発を進める際は前項の npm を使用する方法をお薦めします。

1.	GitHub(https://github.com/DefinitelyTyped/DefinitelyTyped/blob/master/types/office-js/index.d.ts)から型定義ファイルをダウンロードし、編集対象のファイルと同じフォルダにコピーします。
2.	Visual Studio Code でフォルダを開き、ファイルの編集画面を開きます。

3. 文頭にトリプルスラッシュディレクティブ「**/// <reference path="./index.d.ts" />**」を追加し、型定義ファイルを参照することで、インテリセンスが有効になります。	

8.5. Visual Studio Code＆YO OFFICE!＆Office Online による Office アドイン開発

第 4 章で紹介した「YO OFFICE!」を Visual Studio Code のターミナルから実行することで効率よく Office アドインを開発でき、また、Office Online を使用することで簡単に動作確認が行えます(YO OFFICE!のインストールは 4.2 項を参照)。

1. Visual Studio Code でフォルダを開きます。	
2. ターミナルで「**yo office**」コマンドを実行します。	
3. ジェネレーターの指示に従って、Office アドインのテンプレートを作成します。 (A) Choose a project type: Office Add-in Task Pane project (B) Choose a script type: JavaScript (C) What do you want to name your add-in?: My Office Add-in (D) Which Office client application would you like to support?: Excel	

4. YO OFFICE!によって、マニフェストファイルや本体となるスクリプト等のファイル一式が、手順 3.で指定したアドイン名フォルダに出力されるので、CD コマンドでフォルダを移動します。

(例) cd "C:¥apps¥My Office Add-in"

5. 「**npm run start:web**」コマンドを実行すると、ローカルサーバーが立ち上がります。

<table>
<tr><td></td><td>
</td></tr>
<tr><td>6. Edge で Office Online(https://office.live.com/start/Excel.aspx)にアクセスし、Microsoft アカウントでサインインします。</td><td>
</td></tr>
</table>

7. 新規 Excel ブックを作成します。	
8. 7.2 項を参考に、手順 3.で出力されたマニフェストファイル(manifest.xml)をアップロードすると、アドインが読み込まれてホームタブに「Commands Group」グループが表示されます。	

9. 「Show Taskpane」ボタンで表示された作業ウィンドウから「**Run**」ボタンをクリックすると、スクリプトが実行されます。

10. 手順 9.で実行されたスクリプトは「src/ taskpane.js」に記載されています。Visual Studio Code 上でファイルを編集して上書き保存すると、Office Online 上で実行しているアドインも自動的に更新されます。

11. YO OFFICE!では、インテリセンス機能も自動的に有効になるため、前項で紹介している手順で手動でインテリセンスを有効化する必要はありません。	

9. おわりに

これまで述べてきた通り、Office アドインは、VBA や VSTO といった従来の Office 開発とは全く異なる、新しい『Office 開発手法』です。登場から数年経った今でも情報は少なく、AppSource(https://appsource.microsoft.com/ja-jp/)で公開されているアドインでさえも決して多くはありませんが、標準技術による開発、クロスプラットフォーム、Web API やクラウドサービスとの親和性の高さといった、Web アプリの利点を活かしつつ、Excel や Word などの Office アプリケーションと連携が取れるメリットは非常に大きく、今後活用する場面は一層増えていくだろうと思います。

本稿では、Office アドインの“ごく一部”の機能についてまとめてみました。あくまでも Office アドインの概要の把握を目的としているため、説明を省略している部分も多く、また、Visual Studio による開発方法(IDE)や、Office UI Fabric を使った画面設計等々、本稿だけではカバーしきれていない部分も多々あります。

しかし、それでも、本稿がこれから*「Office アドインの開発をはじめてみよう！」*と思われる方の一助となるのであれば、筆者として非常に嬉しく思います。

10. 参考 Web サイト

1. 初心者備忘録 - Office アドインカテゴリー
   https://www.ka-net.org/blog/?cat=130
2. Office ライブラリの JavaScript API をそのコンテンツ配信ネットワーク (CDN) から参照する
   https://docs.microsoft.com/ja-jp/office/dev/add-ins/develop/referencing-the-javascript-api-for-office-library-from-its-cdn
3. Microsoft Ajax Content Delivery Network
   https://docs.microsoft.com/ja-jp/aspnet/ajax/cdn/overview
4. Office アドインを展開し、発行する
   https://docs.microsoft.com/ja-jp/office/dev/add-ins/publish/publish
5. テスト用に Office アドインをサイドロードする
   https://docs.microsoft.com/ja-jp/office/dev/add-ins/testing/create-a-network-shared-folder-catalog-for-task-pane-and-content-add-ins
6. Office アドインの XML マニフェスト
   https://docs.microsoft.com/ja-jp/office/dev/add-ins/develop/add-in-manifests
7. コンテンツ アドインと作業ウィンドウ アドインでの API 使用についてアクセス許可を要求する
   https://docs.microsoft.com/ja-jp/office/dev/add-ins/develop/requesting-permissions-for-api-use-in-content-and-task-pane-add-ins
8. JavaScript API for Office について
   https://docs.microsoft.com/ja-jp/office/dev/add-ins/develop/understanding-the-javascript-api-for-office
9. Office の JavaScript API のリファレンス
   https://docs.microsoft.com/ja-jp/javascript/api/overview/office
10. setSelectedDataAsync(data, options, callback)
    https://docs.microsoft.com/ja-jp/javascript/api/office/office.document?view=office-js#setselecteddataasync-data--options--callback-
11. ドキュメントやスプレッドシート内のアクティブな選択範囲へのデータの読み取りおよび書き込み
    https://docs.microsoft.com/ja-jp/office/dev/add-ins/develop/read-and-write-data-to-the-active-selection-in-a-document-or-spreadsheet
12. Windows 10 で F12 開発者ツールを使用してアドインをデバッグする
    https://docs.microsoft.com/ja-jp/office/dev/add-ins/testing/debug-add-ins-using-f12-developer-tools-on-windows-10
13. Inspect running JavaScript with the Debugger
    https://docs.microsoft.com/ja-jp/previous-versions/windows/internet-explorer/ie-developer/samples/dn255007(v=vs.85)
14. Microsoft Edge Developer Tools

https://docs.microsoft.com/ja-jp/microsoft-edge/devtools-guide
15. Excel Online または Word Online からアドインをデバッグする
https://docs.microsoft.com/ja-jp/office/dev/add-ins/testing/debug-add-ins-in-office-online
16. Excel アドインの概要
https://docs.microsoft.com/ja-jp/office/dev/add-ins/excel/excel-add-ins-overview
17. Excel JavaScript API を使用した基本的なプログラミングの概念
https://docs.microsoft.com/ja-jp/office/dev/add-ins/excel/excel-add-ins-core-concepts
18. Word JavaScript API の概要
https://docs.microsoft.com/ja-jp/office/dev/add-ins/reference/overview/word-add-ins-reference-overview
19. Excel、Word、PowerPoint のマニフェストにアドイン コマンドを作成する
https://docs.microsoft.com/ja-jp/office/dev/add-ins/develop/create-addin-commands
20. VersionOverrides 要素
https://docs.microsoft.com/ja-jp/office/dev/add-ins/reference/manifest/versionoverrides
21. あなたの Web 開発人生を変える Yeoman、Bower、Yo のインストールと使い方
http://www.atmarkit.co.jp/ait/articles/1407/02/news040.html
22. Excel アドインのチュートリアル
https://docs.microsoft.com/ja-jp/office/dev/add-ins/tutorials/excel-tutorial
23. Word アドインのチュートリアル
https://docs.microsoft.com/ja-jp/office/dev/add-ins/tutorials/word-tutorial
24. PowerPoint のアドインのチュートリアル
https://docs.microsoft.com/ja-jp/office/dev/add-ins/tutorials/powerpoint-tutorial
25. チュートリアル: Excel でのカスタム関数の作成
https://docs.microsoft.com/ja-jp/office/dev/add-ins/tutorials/excel-tutorial-create-custom-functions
26. Develop connected tools and add-ins across Office
https://channel9.msdn.com/Events/Connect/2017/T192
27. Microsoft Garage: Script Lab - an Office add-in
https://www.microsoft.com/en-us/garage/profiles/script-lab/
28. Script Lab, a Microsoft Garage project
https://appsource.microsoft.com/ja-jp/product/office/WA104380862
29. Excel でカスタム関数を作成する
https://docs.microsoft.com/ja-jp/office/dev/add-ins/excel/custom-functions-overview
30. Excel カスタム関数のランタイム
https://docs.microsoft.com/ja-jp/office/dev/add-ins/excel/custom-functions-runtime
31. カスタム関数のメタデータ
https://docs.microsoft.com/ja-jp/office/dev/add-ins/excel/custom-functions-json
32. Custom functions in Excel
https://github.com/OfficeDev/Excel-Custom-Functions

33. Create custom functions in Excel
https://github.com/OfficeDev/office-js-docs-pr/blob/master/docs/excel/custom-functions-overview.md
34. Office アドインでの Office UI Fabric
https://docs.microsoft.com/ja-jp/office/dev/add-ins/design/office-ui-fabric
35. テスト用に Office Online で Office アドインをサイドロードする
https://docs.microsoft.com/ja-jp/office/dev/add-ins/testing/sideload-office-add-ins-for-testing
36. 作業ウィンドウ アドインとコンテンツ アドインを SharePoint カタログに発行する
https://docs.microsoft.com/ja-jp/office/dev/add-ins/publish/publish-task-pane-and-content-add-ins-to-an-add-in-catalog

11. 参考書籍

ここでは、Office アドインを開発する際の参考となる書籍を紹介します。

1. Building Office Add-ins using Office.js / Michael Zlatkovsky [著]

	本稿でも紹介している「Script Lab」の開発者である Michael Zlatkovsky 氏によって書かれた電子書籍です。 いわゆる“中の人”が書いた書籍だけあって内容は濃密で、Office アドインの概要から具体的な開発方法、VBA や VSTO との違いにわたるまで詳細に書かれています。 Office アドインの開発者であれば、手元にあって絶対に損は無い資料ですが、唯一の難点は英語であることでしょうか？ 下記リンクから、内容を試し読みできるようになっています。
価格：$15.99 ～ $19.99 ※ 最低価格は $15.99 ですが、推奨価格として $19.99 が設定されています。 ※ 支払いはクレジットカード、PayPal に対応しています。	
URL： https://buildingofficeaddins.com/ https://leanpub.com/buildingofficeaddins/	

2. マイクロソフト Office 用アプリ開発スタートアップガイド / 富士ソフト [著]

 (画像は Amazon より)	Office 用アプリ(現 Office アドイン)について日本語でまとめられた唯一の書籍です。 Office 用アプリのことを本書一冊で大まかに把握できるという点ではお薦めできますが、出版が 2013 年であるため、情報が古くなっていることは留意すべきでしょう。解説に多くのページが割かれた開発ツール「Napa」も現在では使用できなくなっています。 下記リンクから、内容を試し読みできるようになっています。
価格：￥3,168	
URL： https://book.mynavi.jp/manatee/books/detail/id=56193 https://book.mynavi.jp/support/pc/4827/	