SharePoint モダンUIでスクリプトエディタを利用する

はじめに

「**SharePoint Online**」に於いて、よく耳にするのは、以下のキーワードかと推測します。
- サイトに係わるキーワードとして「**クラシック**と**モダン** エクスペリエンス」、「**モダンサイト**」「**クラシックサイト**」
- サイトを表示するデフォルト画面としてモダンサイトは「**サイトページ**」、クラシックサイトは「**WikiPage**」
- 「**Power PlatForm**」、「**Teams**」、「**Office365グループ**」、「**ハブサイト**」

「**働き方改革**」も「**クラウド環境へ管理データの一元化**」「**メールレス化**」「**ペーパレス化**」の対応が加速されています。
その世の中に流れに合わせて、2年前頃より、従来の「**レガシーシステム**」を「**Microsoft365(「SharePoint Online)**」へ移行検討された組織は、基本ポリシーは「**モダンなエクスペリエンス(User Interface)**」で可能な限りサイト構築を推奨し、「モダンサイト」で「モダンページ」を採用されています。
その理由としては、マイクロソフトが発信している、以下の情報による物と推測します。
- ページの設定・編集は利用可能な機能・内容が少ないため操作も容易
- リストやライブラリのアイテム数の**しきい値が「5,000件」から「20,000件」に拡大**
- 携帯端末「Padやスマートホーン」で観た画面はPC閲覧時と同じでレスポンシブ
- 「PowerPlatForm」は 可能な限りローコードで実現できるツールで「モダンUI」を採用、「Teams」も「モダンUI」を採用

ユーザー側も「クラシックなエクスペリエンス(UI)」が数年後は使えなくなる？という不安もあり、ポリシーを決められたのかと推測します。

しかしながら、サイト展開作業ですぐに直面するのは、「**設定が容易は＝期待するレイアウトや機能が実現不可能**」という現実です。
- サイトテンプレート、リストテンプレート機能が使えない ⇒ サイト展開に**膨大な時間(費用)**が掛かる
- ページレイアウトは**期待通りのレイアウトにならない** ⇒ サイトページで使用できるパーツで可能なレイアウト編集は難しい
- ページ編集機能を簡単にパーツ化できない ⇒ 従来のWebパーツやそれを継承した個別Webパーツは使えない
- 従来、実現できた機能が「PowerApps」でないと実現できない ⇒ コンテンツの作成に**膨大な時間(費用)**が掛かる

上記の問題を解決するため、一部のユーザーは「**モダンサイト**」に対して「**カスタムスクリプト」を有効**にする「**PowerShell**」を１行実行し、設定を変更して「**クラシックサイト」の有用な機能を有効にする**ユーザーも多く見かけられます。
また、上記の１行の「PowerShell」をしてクラシックUI「WikiPage」を使わずにモダンUI「サイトページ」で何とかしたいという事例として、「SharePoint Framework」での対応を検討したいユーザーにその対応手順をご紹介します。

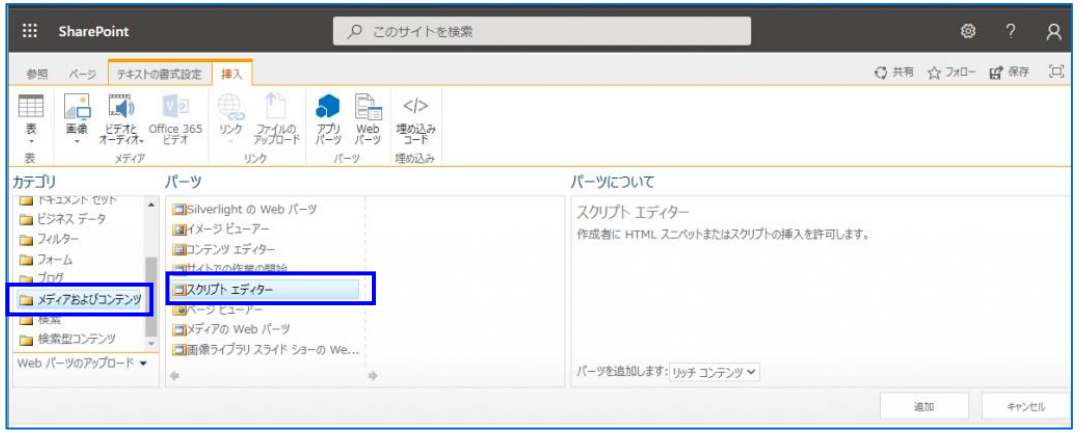
SharePoint モダンUIでスクリプトエディタを利用する

「**SharePoint Online**」が「**クラシックUI**」から「**モダンUI**」が主流になってくるにあたり、どうしても「クラシックUI」で実施していた機能が再現したいというご要望を頂くことが多くなってきました。

そこで、クラシック時代に活躍したWebパーツである「**スクリプトエディタ**」を例にして、モダンUIで利用する迄の手順を紹介します。
「クラシックUI」で一番使用されている？Webパーツ「スクリプトエディタ」を SharePoint Online 環境に反映するまでの手順は以下の通りです。

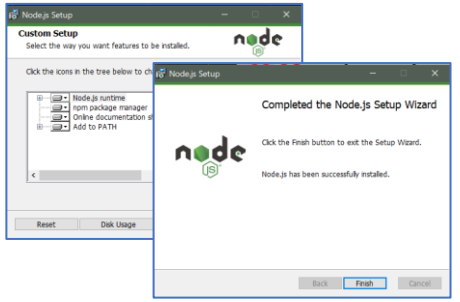
1．スクリプトエディタの概要

- クラシックUIではスクリプトエディタというWebパーツが標準機能として利用できました。
- このスクリプトエディタはJavaScriptを記入でき、SharePoint Online の画面をスクリプトベースでカスタイマイズ可能でした。
- **モダンUI**になり、この**スクリプトエディタ機能は廃止**されてしまっています。
- 代替機能としては **SharePoint Framework** という開発手法で、基本、**TypeScript** でカスタマイズという手順になります。
- **残念ながら従来**のスクリプトエディタでの JavaScript によるカスタマイズよりも、SharePoint Framework によるカスタマイズ(開発)は、かなり**難易度が上がって**います。



SharePoint モダンUIでスクリプトエディタを利用する

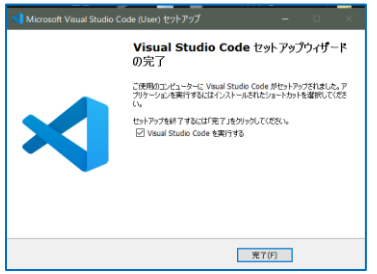
2．事前に必要な環境



- SharePoint Online のサイト　・・・今回スクリプトエディタを利用するサイト
- SharePoint Online のアプリカタログサイト ・・・ スクリプトエディタアプリを登録するサイト
  ※こちらはSharePointの管理者でないとサイト作成ができません。
- SharePoint Framework の開発環境　・・・ SharePoint Framework をビルドできる環境
  ※以下のURLの記載内容を参考にSharePoint Framework 開発環境の設定してください。
  ⇒ https://docs.microsoft.com/ja-jp/sharepoint/dev/spfx/set-up-your-development-environment

★SharePoint Framework の開発環境　・・・ SharePoint Framework をビルドできる環境の準備【操作手順】

① **Node.js. のインストール**
⇒ https://www.nodejs.org/ より「node-v12.14.0-x64.msi」をダウンロードし、
「msi」をクリック　※最新版「node-v18.12.1-x64.msi」では「 npm i」でエラーとなる



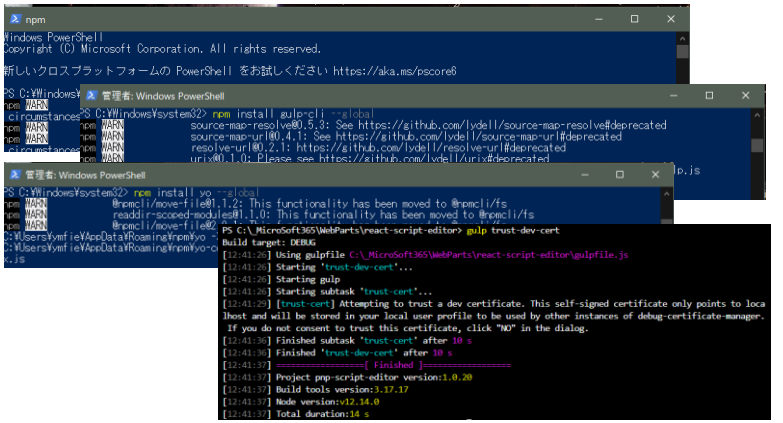
② **コード エディターのインストール**
⇒ https://code.visualstudio.com/docs/?dv=win より
[VSCodeUserSetup-x64-1.73.1.exe] をダウンロードし、[exe]をクリック

③ **開発ツールチェーンの前提条件をインストールする**
1つの行に、次の3つのツールのすべてをインストールできます。
PS > npm install gulp-cli yo @microsoft/generator-sharepoint --global

④ **Gulp のインストール**
PS > npm install gulp-cli --global

⑤ **Yeoman のインストール**
PS > npm install yo --global

⑥ **最新の Web ブラウザーをインストールする**



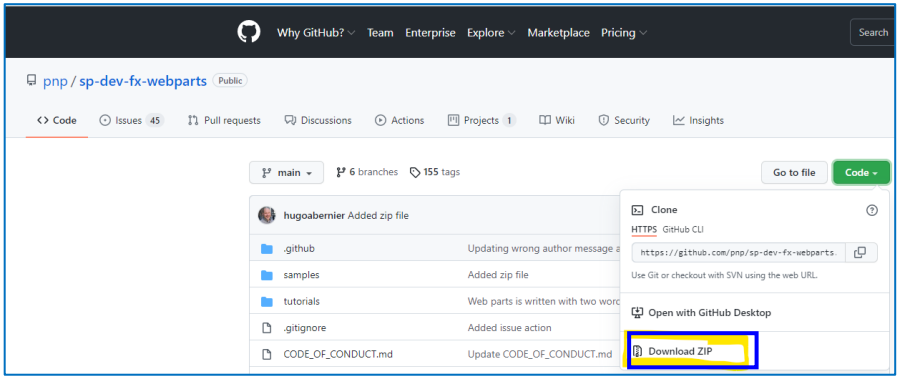
⑦ **開発者用自己署名証明書の信頼**
「3.(4) パッケージファイルの依存関係の反映コマンド実行」完了後、
プロジェクトのルートフォルダー内で以下のコマンドを実行します。
PS> gulp trust-dev-cert

SharePoint モダンUIでスクリプトエディタを利用する

3．スクリプトエディタのビルドから反映まで

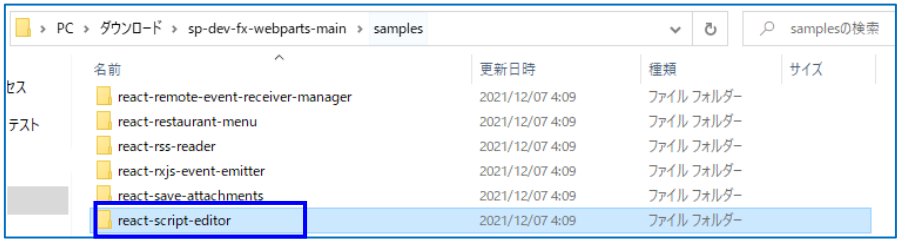
(1) スクリプトエディタのサンプルコード入手

GitHub の **SP-Framework** のサンプルページよりサンプルコードをダウンロードします。
https://github.com/pnp/sp-dev-fx-webparts
すべてのサンプルが含まれているので、11/10時点で圧縮状態でも1.54GBとかなりサイズが大きいです。
sp-dev-fx-webparts-main.zip



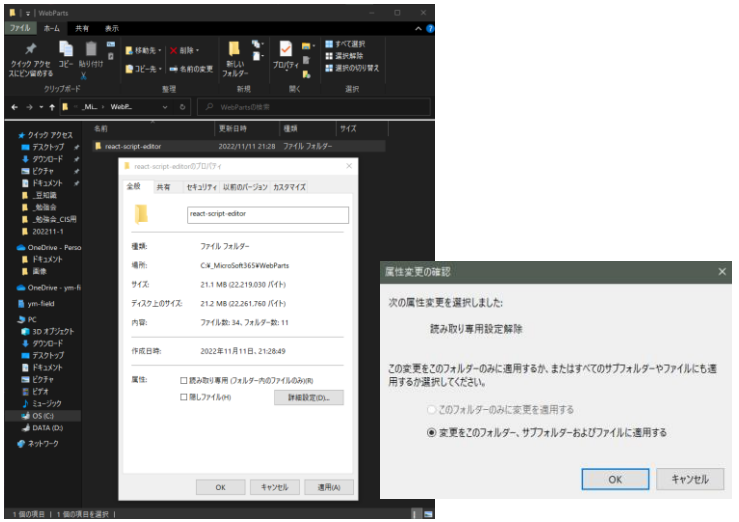
(2) スクリプトエディタのサンプルを取り出す

samplesフォルダから「**react-script-editor**」を取り出して作業フォルダに持っていきます。 作業フォルダは任意の場所(以下のフォルダー)とします。
「C:¥_MicroSoft365¥WebParts¥react-script-editor」



(3) コードの読み取り専用を解除する

任意のフォルダに配置した「react-script-editor」サンプルコードの読み取り専用を解除します。 ※この後の作業でエラーとならないようにするため。



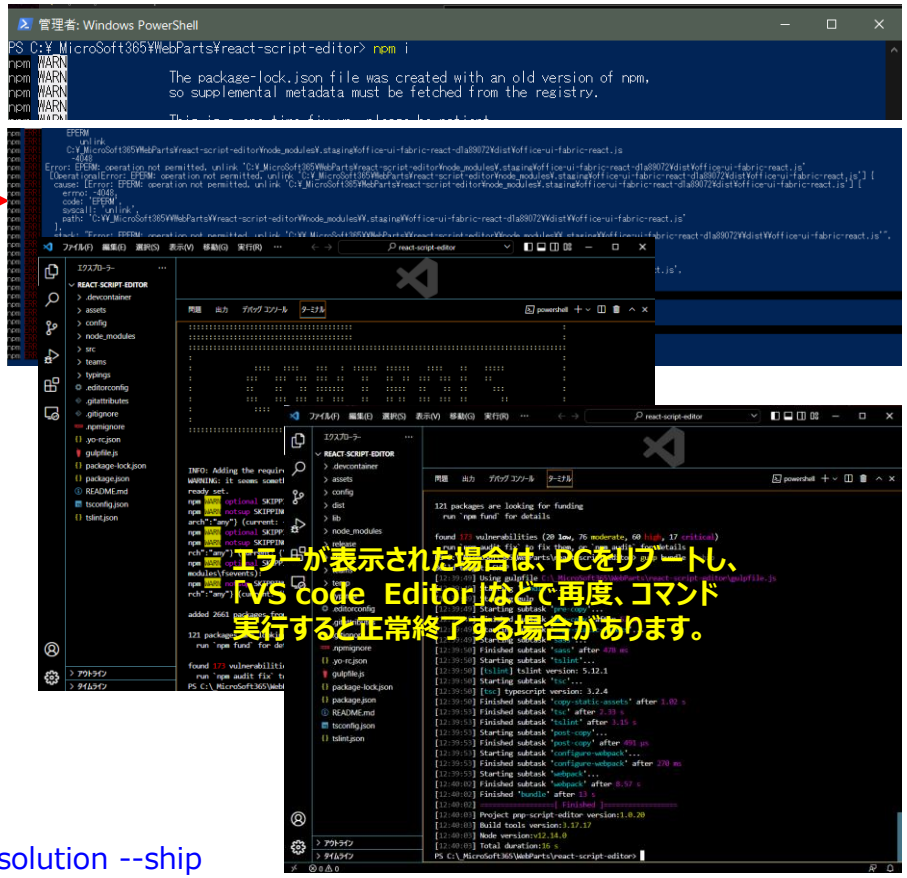
SharePoint モダンUIでスクリプトエディタを利用する

(4) パッケージファイルの依存関係の反映コマンド実行

PowerShellにて「react-script-editor」フォルダを開き、
「react-script-editor」に必要な依存関係のファイルを取得する
以下のコマンドを実行します。
※結構時間がかかります。
警告は出ますが、エラーが出なければ、この後の作業が続行可能です。

PS C:¥_MicroSoft365¥WebParts¥react-script-editor> npm i

※このコマンドが正常終了後は「２.⑦開発者用自己署名証明書の信頼の実行が可能です。



(5) ビルドコマンドの実行

「react-script-editor」をビルドする下記コマンドを実行します。

PS C:¥_MicroSoft365¥WebParts¥react-script-editor　> gulp bundle --ship

(6) ソリューションのパッケージ化コマンドの実行

「react-script-editor」をパッケージする下記コマンドを実行します。

PS C:¥_MicroSoft365¥WebParts¥react-script-editor> gulp package-solution --ship

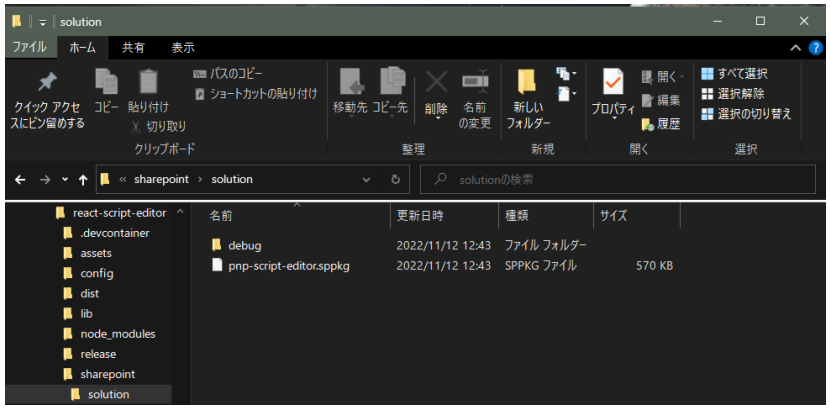
```
PS C:\_MicroSoft365\WebParts\react-script-editor> gulp package-solution --ship
Build target: SHIP
[12:43:58] Using gulpfile C:\_MicroSoft365\WebParts\react-script-editor\gulpfile.js
[12:43:58] Starting 'package-solution'...
[12:43:58] Starting gulp
[12:43:58] Starting subtask 'package-solution'...
```

```
[12:43:59] [package-solution] Done!
[12:43:59] [package-solution]
[12:43:59] [package-solution] ALL DONE!
[12:43:59] [package-solution]
[12:43:59] Finished subtask 'package-solution' after 339 ms
[12:43:59] Finished 'package-solution' after 342 ms
[12:43:59] ================[ Finished ]================
[12:44:00] Project pnp-script-editor version:1.0.20
[12:44:00] Build tools version:3.17.17
[12:44:00] Node version:v12.14.0
[12:44:00] Total duration:3.67 s
```

SharePoint モダンUIでスクリプトエディタを利用する

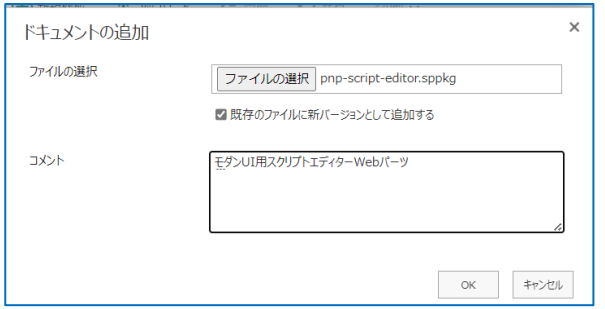
(7) 出来上がったパッケージファイルを確認する

「react-script-editor」のフォルダに「sharepoint¥solution」というフォルダが出来上がるので、そのフォルダに入っている「pnp-script-editor.sppkg」を取得します。



(8) アプリカタログサイトへ登録する

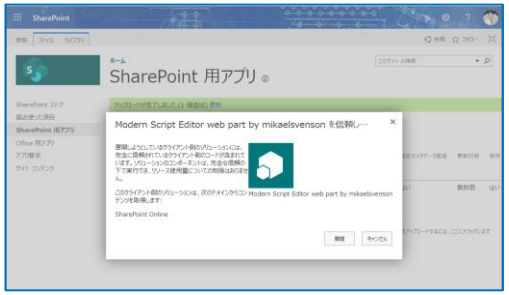
アプリカタログサイトにて、SharePointアプリに「pnp-script-editor.sppkg」をアップロードします。



SharePoint モダンUIでスクリプトエディタを利用する

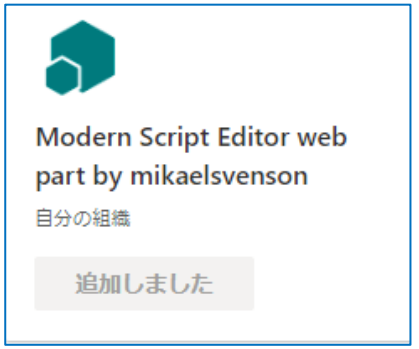
(9) アプリカタログサイトへ展開する

アップロード後に展開画面が表示されるので、展開ボタンを押します。



(10) サイトへの追加

スクリプトエディタを利用したいサイトで「アプリ」クリックで表示される「追加できるアプリ」で作成した「Modern Script Editor Web part」を選択し[追加]をクリックすると、「追加しました。」と表示されると追加成功です。



SharePoint モダンUIでスクリプトエディタを利用する

(11) サイトへのスクリプト有効化コマンドを実行する

スクリプトエディタを利用したいサイトに対してスクリプトを有効化する必要があるため、有効化するスクリプトを実行する。

サイト設定変更コマンド

```
Connect-SPOService -Url https://contoso-admin.sharepoint.com -Credential admin@contoso.com
Set-SPOSite  -DenyAddAndCustomizePages 0
```

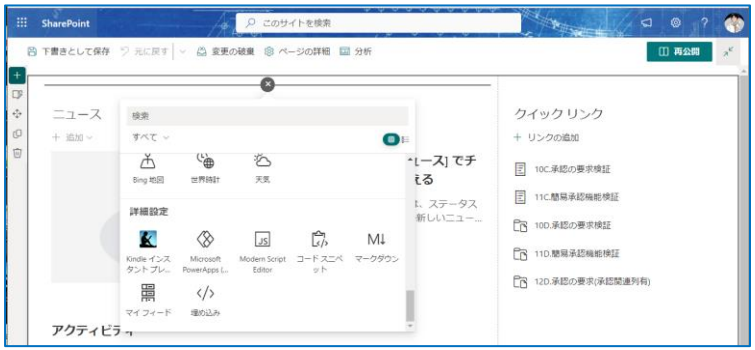
※管理シェルが無い場合はを下記をダウンロードしてインストールしてください。 ※ SharePoint Online Management Shell
https://www.microsoft.com/ja-jp/download/details.aspx?id=35588

※注意

SharePoint管理者権限が必要です。
SharePoint Online ManagementShellモジュールとSharePointClient Components SDKの間には、両方が同じコンピューターにいる場合にモジュールの読み込みに失敗するという既知の問題があります。
この問題が発生した場合は、SharePointクライアントコンポーネントSDKをアンインストールしてください。

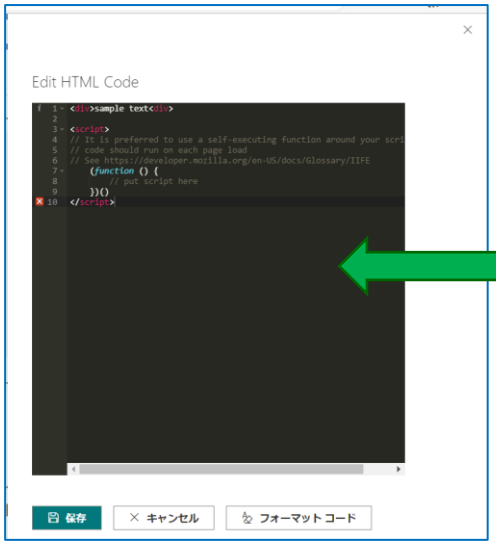
(12) サイトにWebパーツを追加する

サイトにて、「Modern Script Editor Web part」パーツが追加可能となりますので、追加すると下記のようにスクリプトエディタWebパーツが表示されますので、任意のスクリプト「スペニット」を追加してください。



SharePoint モダンUIでスクリプトエディタを利用する

サイトページの事例については、ユーザー要望に合わせ、「Edit HTML Code」プロパティへ、従来の「スクリプトエディターWebパーツ」のスペニットに記述していた「JavaScript」が活用できるかの検証・修正が必要です。




```
<div id="tree"></div>
<script type="text/javascript" src="https://jccunion.sharepoint.com/sites/Study2/ymJsLib/jquery.min.js"></script>
<script type="text/javascript">
// ---------------------------------------------------------------
// スクリプト エディター(c51.TreeViewOfContents2.js)
// コンテンツのツリービュー表示 REST.API 使用サンプル
// ---------------------------------------------------------------
var weburl = _spPageContextInfo.webAbsoluteUrl;
var sMenuString = "";
var rootFolders;
var DocTitle = "11.部署ライブラリ";
var DocTitleEn = "DocLib11";
var sD = "";
var sStr = "";
var ExpCnt = 1;
var ExpNm = "ac1";
$(document).ready(function () {
  TreeViewDisp();
});
// ---------------------------------------------------------------
// ドキュメントライブラリ ツリービューの表示
function TreeViewDisp() {
  sMenuString = "";
  $.ajax({
    url: weburl + "/_api/Web/Lists/GetByTitle('" + DocTitle + "')/items?$select=ID,Title,EncodedAbsUrl,FileRef,FSOb
    type: "GET",
    headers: { "accept": "application/json;odata=verbose", "Content-Type": "application/json;odata=verbose", "x-re
    success: this.onSuccesded,
    error: function (xhr) { alert("bindTreeView Error "+xhr.status + ": " + xhr.statusText); }
  });
}
function onSuccesded(data) {
  var imgU = "";
  var cCnt = 1;
  if (data.d.results.length > 0) {
```

クラシックUIでは、編集者にはコードを意識させない専用Webパーツ化していましたが、モダンUIでは、その機能が容易に実現できないため、コードを意識した作業が必要になります。

おわりに

今回、追加したスクリプトエディタですが、モダンUIにて廃止になっているスクリプトエディタをどうしても利用したいというご要件を頂いた際に選択肢の一つとして対応できる可能性がありますので、ご参考になれば幸いです。

但し、モダンサイトでもクラシックUIの「WikiPage」を使えば上記の作業は不要です。
通常のサイトやコンテンツの設定画面、ドキュメントセット画面など、サイトページ・ビュー一覧、フォーム画面以外は、従来のクラシックUIで提供され画面であり、SharePointの標準機能を最大限活用したい、ユーザー要望を最大限対応したい場合は、モダンUIには固着せず(囚われず)、従来のクラシックUIで提供されたSharePoint標準機能を最大限活用した**「可能な限りモダンUI」**をお勧めします。

【おまけ】モダンUIサイトページの編集(カスタマイズ)について

可能な限り「モダンUI」での対応は、従来の「クラシックUI」と比較し、SharePoint標準で提供される「Webパーツ」が利用できないため、想定されるユーザー要望は殆どはSharePointだけでは実現できません。

① モダンUIのリストやライブラリへは「Webパーツ」を追加不可

・ビューの表示編集は、基本「JSON」設定のみ
→ビューへのアクセス権限設定ができない。 TreeView表示機能の追加などはできない。
・フォームの表示編集や機能追加は「PowerApps」でのみ
→ヘッダー、本文、フッターの表示編集は、基本「JSON」設定のみだが、列ごとのClass指定などによる表示制御は、期待する動作設定が難しい。
→フォームへの表示制御・機能追加は、「PowerApps」での編集作業が必要。
クラシックUIは、「NewForm」「DispForm」「EditForm」ごとに編集か可能だが、モダンUI用の「PowerApps」画面は一つの画面設定で対応は必要であり、専用コードでの記述が必要なため、専任者での対応を推奨。

② モダンUIのサイトのページに限り「Webパーツ」の追加は許可

SharePoint OnlineのJavaScript Object Model(JSOM)は利用できないため、サイトページに限り、大別して２つの対策案が考えられます。
【参考URL(内田洋行 太田さんのブログ):https://idea.tostring.jp/?p=3986】

◆ **SharePoint Framework の利用【本命】**
敷居が高いと思われている「SharePoint Framework」(公開されているソースコード有り)の「スクリプト エディタWebパーツ」をパッケージ化して活用
※参考資料：SharePointを学ぼう_17.【モダンUI】サイトの編集「スクリプトエディタを利用」

◆ **サイトページに標準機能の組み合わせで行う方法**
クラシックUIの標準機能をサイトページへ「埋め込み Web パーツで表示」する内容であり、サイトページに固着しなくても良い場合は、SharePoint標準の「WikiPage」の活用をお勧めします。
① JavaScript を埋め込んだページを作成する。 -「WikiPage」(.aspx)でも代用可能？
② SharePoint Online のデータを利用する場合には REST API を利用する
>SharePoint Online のドキュメントライブラリにアップロードした「.aspx」ファイルからJSOMを利用 ⇒ https://idea.tostring.jp/?p=5462
>ページ上への表示に iframe が利用されるため、CSS も自分で作成しておく必要がある。
>例えば、Office UI Fabric JS ⇒ https://developer.microsoft.com/en-us/fabric-js
この Office UI Fabric を基に自分で「CSS」のカスタマイズを行い、ページを拡張子「.aspx」として保存して任意のドキュメントライブラリにアップロード。
③ 埋め込み Web パーツで表示
これを SharePoint Online モダン ページの埋め込み Web パーツで呼出しは、SharePoint Online のドキュメントライブラリに保存されているページを直接参照できる URL を指定する。
これで、ページ上に先ほど作成したページが埋め込み Web パーツによって iframe のコンテンツとして表示されます。
表示される高さなどを指定したい場合には、埋め込み Web パーツの設定に iframe タグを記述することもできます。