/\*\*

\* =============================================================================

\* @file 00\_initializer.gs

\* @module GAssistant.Initializer

\* @description

\* هذا الملف هو نقطة الانطلاق وحجر الأساس للمشروع بأكمله.

\*

\* الوظائف الرئيسية:

\* 1. \*\*إنشاء مساحة الاسم:\*\* يقوم بتعريف الكائن العام `GAssistant` مرة واحدة فقط،

\* لتجنب تلوث النطاق العام (Global Scope) وضمان تنظيم الكود.

\* 2. \*\*تهيئة الوحدات الفرعية الديناميكية:\*\* يقوم بقراءة `loadOrder.json` واستدعاء

\* وظائف التهيئة لكل وحدة بالترتيب المحدد ديناميكيًا، لضمان تحميل سليم للتبعيات.

\* 3. \*\*نقطة الدخول عند الفتح:\*\* يحتوي على دالة `onOpen` التي يتم استدعاؤها

\* تلقائيًا بواسطة Google Sheets، والتي بدورها تشغل عملية التهيئة الكاملة.

\*

\* @architectural\_note يجب أن يكون هذا الملف هو أول ملف يتم تحميله في

\* مشروع Google Apps Script لضمان تهيئة البيئة بشكل صحيح.

\* =============================================================================

\*/

// تعريف الكائن العام للمشروع. هذا هو المدخل الوحيد في النطاق العام.

var GAssistant = typeof GAssistant !== "undefined" ? GAssistant : {};

// تهيئة مساحات الأسماء الأساسية لضمان وجودها.

// هذه مجرد تهيئة فارغة لتجنب أخطاء 'undefined' عند محاولة إضافة خصائص لاحقًا.

GAssistant.AI = GAssistant.AI || {};

GAssistant.UI = GAssistant.UI || {};

GAssistant.Tools = GAssistant.Tools || {};

GAssistant.Utils = GAssistant.Utils || {};

GAssistant.Security = GAssistant.Security || {};

GAssistant.Analytics = GAssistant.Analytics || {};

GAssistant.Tests = GAssistant.Tests || {};

GAssistant.Agents = GAssistant.Agents || {};

GAssistant.API = GAssistant.API || {};

/\*\*

\* دالة Apps Script القياسية التي تُنفذ عند فتح المستند.

\* هي نقطة الدخول الوحيدة التي تُشغل تهيئة الوحدات ومن ثم الواجهة.

\* @public

\*/

function onOpen() {

// \*\*التصحيح\*\*: استدعاء دالة التهيئة الديناميكية للنظام أولاً.

// هذه الدالة ستقوم بتهيئة GAssistant.Utils أولاً، لذا ستكون متاحة.

\_initializeGAssistantSystemDynamically();

// بعد التأكد من أن جميع الوحدات قد تم تحميلها وتهيئة IIFEs الخاصة بها،

// يمكننا الآن استخدام GAssistant.Utils.log بأمان.

const Utils = GAssistant.Utils;

Utils.log('G-Assistant: onOpen triggered. UI initialization follows...');

// وأخيراً، استدعاء دالة onOpen الخاصة بوحدة الواجهة GAssistant.UI.

const UI\_Module = GAssistant.UI;

if (UI\_Module && typeof UI\_Module.onOpen === 'function') {

UI\_Module.onOpen();

} else {

// هذا السيناريو يجب ألا يحدث أبداً مع هذا التصميم والترتيب الأبجدي للملفات.

Utils.error("CRITICAL: GAssistant.UI.onOpen is not available. UI initialization failed.");

SpreadsheetApp.getUi().alert("خطأ فادح: فشل تهيئة واجهة المستخدم.");

}

}

/\*\*

\* دالة تهيئة النظام الرئيسية. تقوم بقراءة loadOrder.json

\* وتقوم بتهيئة الوحدات ديناميكيًا بالترتيب المحدد.

\* @private

\*/

function \_initializeGAssistantSystemDynamically() {

// \*\*التصحيح\*\*: لا يمكننا الاعتماد على GAssistant.Utils هنا بعد.

// سنقوم بتهيئة Utils يدوياً أولاً.

GAssistant.Utils = GAssistant.Utils || {}; // تأكيد الوجود

if (typeof \_Utils\_Module === 'function') { // \_Utils\_Module هي الدالة التي تُعرف Utils في 02\_utils.gs

GAssistant.Utils = \_Utils\_Module();

} else {

console.error("CRITICAL: \_Utils\_Module not found. Cannot initialize basic utilities.");

throw new Error("Initialization failed: Core utilities not found.");

}

const Utils = GAssistant.Utils; // الآن Utils متاح.

Utils.log('Initializer: Starting dynamic system initialization based on loadOrder.json...');

let loadOrder = null;

try {

// قراءة loadOrder.json. هذا يتطلب أن تكون الملفات النصية متاحة.

// Apps Script لا يسمح بقراءة ملفات .json مباشرة من الكود.

// لذا، يجب أن نقوم بتحميلها كملف نصي.

// هذا هو الحل المعتاد: قراءة الملف كـ text Blob ثم parse.

const loadOrderFile = DriveApp.getFilesByName('loadOrder.json').next();

loadOrder = JSON.parse(loadOrderFile.getBlob().getDataAsString());

Utils.log('Initializer: loadOrder.json loaded successfully.');

} catch (e) {

Utils.error('Initializer: Failed to load or parse loadOrder.json.', e);

throw new Error('Initialization failed: Could not load loadOrder.json.');

}

// المرور على الأقسام والوحدات حسب الترتيب في loadOrder.json

for (const sectionKey in loadOrder) {

if (loadOrder.hasOwnProperty(sectionKey)) {

const modulesInOrder = loadOrder[sectionKey];

Utils.log(`Initializer: Processing section: ${sectionKey}`);

// تهيئة مساحة الاسم إذا لم تكن موجودة (مثلاً GAssistant.AI)

if (sectionKey !== 'Utils' && sectionKey !== 'Config' && sectionKey !== 'UI' && sectionKey !== 'API' && sectionKey !== 'Tools' && sectionKey !== 'AI' && sectionKey !== 'Agents' && sectionKey !== 'Security' && sectionKey !== 'Analytics' && sectionKey !== 'Tests') {

GAssistant[sectionKey] = GAssistant[sectionKey] || {};

}

for (const moduleName of modulesInOrder) {

// بناء الاسم المتوقع للدالة (مثلاً \_Config\_Module, \_AI\_Core\_Module)

// يجب أن نكون حذرين مع الأسماء هنا.

let functionName = `\_${moduleName}\_Module`;

// للتعامل مع "API" و "UI" و "Tools" و "AI" و "Agents" و "Security" و "Analytics" و "Tests" التي هي في المستوى الأول مباشرة

if (['Config', 'Utils', 'UI', 'AI', 'Agents', 'Tools', 'Security', 'Analytics', 'Tests', 'API'].includes(moduleName)) {

// هذا يعني أن الوحدة هي القسم نفسه وليست جزءاً منه.

// مثال: GAssistant.Config تُعرّف بواسطة \_Config\_Module

functionName = `\_${moduleName}\_Module`;

} else {

// هذا يعني أنها وحدة فرعية داخل قسم.

// مثال: GAssistant.AI.Core تُعرّف بواسطة \_AI\_Core\_Module

functionName = `\_${sectionKey}\_${moduleName}\_Module`;

}

// الوصول إلى الدالة العامة لتهيئة الوحدة.

const loaderFunction = GAssistant.Utils.getFunctionByPath(functionName);

if (typeof loaderFunction === 'function') {

// استدعاء دالة التهيئة وتخزين الناتج في مساحة الاسم GAssistant

// إذا كان moduleName هو 'AI' أو 'UI' أو 'Tools' أو غيرها، فسيتم تعيينها مباشرة.

// إذا كان moduleName هو 'Core' داخل 'AI'، فسيتم تعيين GAssistant.AI.Core.

// هذا يتطلب مرونة في طريقة التعيين.

let targetNamespace = GAssistant;

if (sectionKey === 'AI' || sectionKey === 'Agents' || sectionKey === 'Tools' || sectionKey === 'UI') {

// الوحدات في هذه الأقسام هي وحدات فرعية (مثل AI.Core, Tools.Catalog)

// يجب التأكد أن القسم الأب (GAssistant.AI, GAssistant.Tools) موجود.

if (typeof GAssistant[sectionKey] === 'undefined') {

GAssistant[sectionKey] = {}; // تهيئة القسم الأب إذا لم يكن موجوداً.

}

targetNamespace = GAssistant[sectionKey];

}

// إذا كان moduleName هو نفس sectionKey (مثال: "Utils" في قسم "Utils")

if (moduleName === sectionKey) {

targetNamespace[moduleName] = loaderFunction();

} else {

// هذا لـ AI.Core, Tools.Catalog, etc.

targetNamespace[moduleName] = loaderFunction();

}

Utils.log(`Initializer: ✅ Loaded module: GAssistant.${sectionKey}.${moduleName}`);

} else {

Utils.warn(`Initializer: ⚠️ Loader function not found for: ${functionName}. Module GAssistant.${sectionKey}.${moduleName} might not be initialized.`);

}

}

}

}

Utils.log('Initializer: Dynamic system initialization completed successfully.');

}