// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// --- START OF FILE: 02\_utils.gs ---

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*

\* @file 02\_utils.gs

\* @module GAssistant.Utils

\* @description

\* وحدة الأدوات المساعدة العامة والموحدة مع نظام حقن التبعيات وفحص البيئة.

\* تشمل أدوات تسجيل، فحص، تحليل، وتنفيذ آمن لدوال النظام.

\*/

// تأكيد وجود الكائن الرئيسي قبل أي تنفيذ

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

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

// تعريف الوحدة الفعلية كـ IIFE تُنفّذ فورًا عند تحميل السكريبت

GAssistant.Utils = (() => {

'use strict';

// ==================== تحميل التكوين (بشكل آمن) ====================

// يتم الوصول إلى GAssistant.Config هنا بشكل آمن، بعد أن يكون قد تم تعريفه كـ IIFE.

function \_getConfig() {

return (GAssistant.Config && typeof GAssistant.Config.getAll === 'function') ? GAssistant.Config.getAll() : {

DEBUG\_MODE: true,

OPERATION\_LOG\_SHEET: "ErrorLog",

MAX\_LOG\_ENTRIES: 500,

PRIMARY\_HEADER\_COLOR: "#4a5568" // قيمة افتراضية لتجنب الخطأ

};

}

// ==================== أدوات التسجيل ====================

function log(message, data) {

console.log(`[INFO] ${message}`, data ? JSON.stringify(data) : '');

if (\_getConfig().DEBUG\_MODE) \_logToSheet('INFO', message, data);

}

function warn(message, data) {

console.warn(`[WARN] ${message}`, data ? JSON.stringify(data) : '');

if (\_getConfig().DEBUG\_MODE) \_logToSheet('WARN', message, data);

}

function error(message, err) {

const details = err instanceof Error ? { message: err.message, stack: err.stack } : err;

console.error(`[ERROR] ${message}`, JSON.stringify(details));

\_logToSheet('ERROR', message, details);

}

function \_logToSheet(level, message, data) {

try {

const config = \_getConfig();

// تأكد أن getSheet و GAssistant.Config متاحتان

if (typeof getSheet !== 'function' || !config.OPERATION\_LOG\_SHEET) {

console.error(`Utils: Cannot log to sheet - getSheet or config missing.`);

return;

}

const sheet = getSheet(config.OPERATION\_LOG\_SHEET, ["Timestamp", "Level", "Message", "Data"]);

if (!sheet) return;

const compact = data ? JSON.stringify(data).substring(0, 1000) : '';

sheet.appendRow([new Date(), level, message, compact]);

const maxRows = config.MAX\_LOG\_ENTRIES || 500;

if (sheet.getLastRow() > maxRows) {

sheet.deleteRows(2, sheet.getLastRow() - maxRows);

}

} catch (e) {

console.error(`Utils: Failed to log to sheet - ${e.message}`);

}

}

// ==================== فحص الورقة ====================

function getSheet(name, headers) {

if (!name || typeof name !== 'string') {

error('getSheet: اسم الورقة غير صالح.');

return null;

}

const ss = SpreadsheetApp.getActiveSpreadsheet();

let sheet = ss.getSheetByName(name);

if (sheet) {

const isHidden = sheet.isSheetHidden?.() ?? false;

// ✅ تصحيح: فحص وجود getProtections قبل الاستدعاء

const isProtected = sheet.getProtections ? (sheet.getProtections(SpreadsheetApp.ProtectionType.SHEET)?.length > 0) : false;

if (isHidden || isProtected) {

warn(`getSheet: الورقة "${name}" مخفية أو محمية.`);

return null;

}

} else {

sheet = ss.insertSheet(name);

if (Array.isArray(headers) && headers.length > 0) {

const range = sheet.getRange(1, 1, 1, headers.length);

const headerColor = \_getConfig().PRIMARY\_HEADER\_COLOR;

range.setValues([headers])

.setFontWeight('bold')

.setBackground(headerColor)

.setFontColor('#ffffff');

}

}

return sheet;

}

// ==================== تنفيذ آمن ====================

function executeSafely(func, args = [], context = '') {

const start = Date.now();

try {

const result = func(...args);

log(`Execution Time: '${func.name || context}'`, { duration\_ms: Date.now() - start });

return result;

} catch (e) {

const duration = Date.now() - start;

error(`Execution failed: ${func.name || 'anonymous'} ${context}`, e);

log(`Execution Time (Failed): '${func.name || context}'`, { duration\_ms: duration });

const Dialogue = GAssistant?.UI?.Dialogue; // الوصول الآمن لوحدة الحوار

return Dialogue ? Dialogue.createError(`حدث خطأ: ${e.message}`) : { type: 'error', text: e.message };

}

}

// ==================== أدوات إضافية ====================

function getFunctionByPath(path) {

// هذا يسمح بالوصول إلى وحدات متداخلة مثل GAssistant.AI.Core

// يعتمد على أن GAssistant تم تهيئته بالكامل في النطاق العام.

return path.split('.').reduce((obj, key) => obj?.[key], GAssistant);

}

function safeParse(str) {

try { return JSON.parse(str); } catch (\_) { return null; }

}

// ==================== حقن التبعيات ====================

const Injector = {

get: (...moduleNames) => {

const result = {};

for (const fullName of moduleNames) {

// الوصول للوحدات عبر GAssistant بعد أن تكون قد تم تعريفها في النطاق العام.

const mod = getFunctionByPath(`GAssistant.${fullName}`);

if (typeof mod === 'undefined' || mod === null) {

const msg = `CRITICAL DEPENDENCY ERROR: Module GAssistant.${fullName} is not loaded or does not exist.`;

// نستخدم error() الخاص بـ Utils نفسه هنا

error(msg);

throw new Error(msg);

}

// لا يوجد دعم لـ Lazy Initializers هنا، لأن كل الوحدات IIFE وتُنفذ فورًا

const key = fullName.split('.').pop();

result[key] = mod;

}

return result;

}

};

// ==================== دالة تعريف الوحدة ====================

/\*\*

\* دالة مساعدة لتعريف الوحدات. تُستخدم لتوحيد عملية حقن التبعيات وتسجيل الوحدة.

\* @param {string[]} deps - مصفوفة بمسارات التبعيات المطلوبة (e.g., ['Utils', 'UI.Dialogue']).

\* @param {function(object): any} factory - دالة المصنع التي تنفذ منطق الوحدة وتستقبل التبعيات.

\* @param {string} label - تسمية الوحدة للتسجيل (e.g., 'AI.Core').

\* @returns {any} - الناتج الذي تُرجعه دالة المصنع (الوحدة نفسها).

\*/

function defineModule(deps, factory, label) {

// استخدام Injector لجلب التبعيات

const resolvedDeps = Injector.get(...deps);

// تنفيذ دالة المصنع مع التبعيات المحقونة

const output = factory(resolvedDeps);

// تسجيل الوحدة عند تهيئتها بنجاح

if (label) {

log(`📦 Module Ready: ${label}`);

}

return output;

}

// ==================== أدوات فحص هيكل التحميل ====================

/\*\*

\* يتحقق من صحة هيكل loadOrder.json.

\* @param {object} structure - كائن loadOrder.json الذي تم تحميله.

\* @returns {void}

\*/

function validateLoadOrder(structure) {

const reservedTopLevelNames = [

'Utils', 'Config', 'UI', 'AI', 'Agents', 'Tools', 'Security', 'Analytics', 'Tests', 'API'

];

// تأكد أن '\_note' يمكن أن يتواجد كخاصية

if (structure && typeof structure === 'object') {

for (const key in structure) {

if (structure.hasOwnProperty(key) && key !== '\_note') { // تجاهل خاصية الملاحظة

if (!reservedTopLevelNames.includes(key)) {

warn(`validateLoadOrder: Unrecognized top-level namespace in loadOrder.json: "${key}".`);

}

if (!Array.isArray(structure[key])) {

warn(`validateLoadOrder: Section "${key}" in loadOrder.json is not an array.`);

}

}

}

} else {

error(`validateLoadOrder: loadOrder.json structure is invalid or empty.`);

}

}

// ==================== تصدير ====================

return {

log, warn, error,

getSheet, executeSafely,

getFunctionByPath, safeParse,

Injector,

defineModule,

validateLoadOrder // ✅ تصدير الدالة الجديدة

};

})();

/\*\*

\* ✅ يتحقق من وجود كل الوحدات الحيوية داخل GAssistant

\* ويسجل حالة كل واحدة (موجود / مفقودة / دالة مؤجلة)

\*/

function verifyCoreModules() {

// ✅ تصحيح: الوصول لـ GAssistant.Utils بشكل آمن هنا

const Utils = GAssistant.Utils;

if (!Utils || typeof Utils.log !== 'function' || typeof Utils.getFunctionByPath !== 'function') {

// هذا السيناريو يجب ألا يحدث، ولكن هذا هو خط الدفاع الأخير.

console.error("❌ وحدة Utils غير جاهزة. لا يمكن تنفيذ الفحص.");

return;

}

const requiredModules = [

'AI',

'AI.Core',

'AI.Memory',

'Agents',

'Tools',

'Tools.Catalog',

'UI',

'UI.Dialogue',

'Config',

'Utils', // يجب أن تكون Utils متاحة هنا

'Security',

'Analytics',

'Tests',

'API',

'Types' // هذا ليس موديول JavaScript عاديًا، هو فقط لـ JSDoc

];

Utils.log("🔍 بدء فحص وحدات GAssistant الأساسية...", { total: requiredModules.length });

requiredModules.forEach((path) => {

let module = Utils.getFunctionByPath(`GAssistant.${path}`);

// بما أن كل الوحدات IIFE، لا يوجد مفهوم Lazy Initializer هنا.

// const isLazy = typeof module === 'function';

if (!module) {

Utils.error(`❌ الوحدة "GAssistant.${path}" غير محملة`);

} else {

Utils.log(`✅ الوحدة "GAssistant.${path}" محملة`);

// هذا الجزء يخص Types فقط، يجب استثناؤه من الفحص العام للوحدات

if (path === 'Types') {

if (typeof GAssistant.Types !== 'undefined') {

Utils.log(`✅ الوحدة "GAssistant.Types" (JSDoc) موجود`);

} else {

Utils.warn(`⚠️ الوحدة "GAssistant.Types" (JSDoc) غير موجودة.`);

}

}

}

});

Utils.log("✅ تم الانتهاء من فحص الوحدات.");

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// --- END OF FILE: 02\_utils.gs ---

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*