Code for Documents

function extractUrlsAndPasteToSheet() {

// Get the active spreadsheet

const spreadsheet = SpreadsheetApp.getActiveSpreadsheet();

// Get the active sheet within the spreadsheet

const sheet = spreadsheet.getActiveSheet();

const documentIds = sheet.getRange(1, 1, 1, sheet.getLastColumn()).getValues()[0];

for (let i = 0; i < documentIds.length; i++) {

const documentId = documentIds[i];

if (!documentId) continue; // Skip empty cells

try {

const urls = extractUrlsFromDocument(documentId);

// Paste the extracted URLs into the column under the document ID

const pasteRange = sheet.getRange(2, i + 1, urls.length, 1);

pasteRange.setValues(urls.map(url => [url]));

} catch (error) {

Logger.log(`Error processing document ID ${documentId}: ${error}`);

sheet.getRange(2, i + 1).setValue("Error");

}

}

Logger.log("URLs extraction and pasting completed.");

}

function extractUrlsFromDocument(documentId) {

const documentContent = Docs.Documents.get(documentId).body.content;

const urls = [];

JSON.parse(JSON.stringify(documentContent), (key, value) => {

if (key === "url") urls.push(value);

});

return urls;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code for Sheets

function extractUrlsAndPasteToSheet() {

// Get the active spreadsheet

const spreadsheet = SpreadsheetApp.getActiveSpreadsheet();

// Get the active sheet within the spreadsheet

const sheet = spreadsheet.getActiveSheet();

const sheetIds = sheet.getRange(1, 1, 1, sheet.getLastColumn()).getValues()[0];

for (let i = 0; i < sheetIds.length; i++) {

const sheetId = sheetIds[i];

if (!sheetId) continue; // Skip empty cells

try {

const urls = extractUrlsFromSheet(sheetId);

// Paste the extracted URLs into the column under the sheet ID

const pasteRange = sheet.getRange(2, i + 1, urls.length, 1);

pasteRange.setValues(urls.map(url => [url]));

} catch (error) {

Logger.log(`Error processing sheet ID ${sheetId}: ${error}`);

sheet.getRange(2, i + 1).setValue("Error");

}

}

Logger.log("URLs extraction and pasting completed.");

}

function extractUrlsFromSheet(sheetId) {

const sheet = SpreadsheetApp.openById(sheetId).getActiveSheet();

const rangeValues = sheet.getDataRange().getValues();

const urls = [];

for (let i = 0; i < rangeValues.length; i++) {

for (let j = 0; j < rangeValues[i].length; j++) {

const cellValue = rangeValues[i][j];

if (typeof cellValue === 'string') {

const urlRegex = /(https?:\/\/[^\s]+)/g;

const matches = cellValue.match(urlRegex);

if (matches) {

urls.push(...matches);

}

}

}

}

return urls;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code for Slides

function extractUrlsAndPasteToSheet() {

// Get the active spreadsheet

const spreadsheet = SpreadsheetApp.getActiveSpreadsheet();

// Get the active sheet within the spreadsheet

const sheet = spreadsheet.getActiveSheet();

const sheetIds = sheet.getRange(1, 1, 1, sheet.getLastColumn()).getValues()[0];

for (let i = 0; i < sheetIds.length; i++) {

const sheetId = sheetIds[i];

if (!sheetId) continue; // Skip empty cells

try {

const urls = extractUrlsFromSlides(sheetId);

// Paste the extracted URLs into the column under the sheet ID

const pasteRange = sheet.getRange(2, i + 1, urls.length, 1);

pasteRange.setValues(urls.map(url => [url]));

} catch (error) {

Logger.log(`Error processing sheet ID ${sheetId}: ${error}`);

sheet.getRange(2, i + 1).setValue("Error");

}

}

Logger.log("URLs extraction and pasting completed.");

}

function extractUrlsFromSlides(slideId) {

const presentation = SlidesApp.openById(slideId);

const slides = presentation.getSlides();

const urls = [];

slides.forEach(slide => {

const shapes = slide.getShapes();

shapes.forEach(shape => {

const text = shape.getText();

const links = text.getLinks();

links.forEach(link => {

const url = link.getTextStyle().getLink().getUrl();

urls.push(url);

});

});

});

return urls;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Complete code

function extractUrlsAndPasteToSheet() {

// Get the active spreadsheet

const spreadsheet = SpreadsheetApp.getActiveSpreadsheet();

// Get the active sheet within the spreadsheet

const sheet = spreadsheet.getActiveSheet();

const documentIds = sheet.getRange(1, 1, 1, sheet.getLastColumn()).getValues()[0];

for (let i = 0; i < documentIds.length; i++) {

const documentId = documentIds[i];

if (!documentId) continue; // Skip empty cells

let urls;

try {

if (isGoogleSheet(documentId)) {

urls = extractUrlsFromSheet(documentId);

} else if (isGoogleDoc(documentId)) {

urls = extractUrlsFromDocument(documentId);

} else if (isGoogleSlide(documentId)) {

urls = extractUrlsFromSlides(documentId);

} else {

urls = ["No URLs"];

}

// Paste the extracted URLs into the column under the document ID

const pasteRange = sheet.getRange(2, i + 1, urls.length, 1);

pasteRange.setValues(urls.map(url => [url]));

} catch (error) {

Logger.log(`Error processing document ID ${documentId}: ${error}`);

sheet.getRange(2, i + 1).setValue("Error");

}

}

Logger.log("URLs extraction and pasting completed.");

}

function isGoogleSheet(sheetId) {

try {

SpreadsheetApp.openById(sheetId);

return true;

} catch (error) {

return false;

}

}

function isGoogleDoc(docId) {

try {

Docs.Documents.get(docId);

return true;

} catch (error) {

return false;

}

}

function isGoogleSlide(slideId) {

try {

SlidesApp.openById(slideId);

return true;

} catch (error) {

return false;

}

}

function extractUrlsFromSheet(sheetId) {

const sheet = SpreadsheetApp.openById(sheetId).getActiveSheet();

const rangeValues = sheet.getDataRange().getValues();

const urls = [];

for (let i = 0; i < rangeValues.length; i++) {

for (let j = 0; j < rangeValues[i].length; j++) {

const cellValue = rangeValues[i][j];

if (typeof cellValue === 'string') {

const urlRegex = /(https?:\/\/[^\s]+)/g;

const matches = cellValue.match(urlRegex);

if (matches) {

urls.push(...matches);

}

}

}

}

return urls.length > 0 ? urls : ["No URLs"];

}

function extractUrlsFromDocument(documentId) {

const documentContent = Docs.Documents.get(documentId).body.content;

const urls = [];

JSON.parse(JSON.stringify(documentContent), (key, value) => {

if (key === "url") urls.push(value);

});

return urls.length > 0 ? urls : ["No URLs"];

}

function extractUrlsFromSlides(slideId) {

const presentation = SlidesApp.openById(slideId);

const slides = presentation.getSlides();

const urls = [];

slides.forEach(slide => {

const shapes = slide.getShapes();

shapes.forEach(shape => {

const text = shape.getText();

const links = text.getLinks();

links.forEach(link => {

const url = link.getTextStyle().getLink().getUrl();

urls.push(url);

});

});

});

return urls.length > 0 ? urls : ["No URLs"];

}

Code for extraction of the documents from folder to columns (Extracting URLs 2)

function listGoogleDocsInFolder() {

// Define the Google Drive folder ID to search within

const folderId = "1QszzSwGz904z8dR\_mg3inLpXcjbdFyYu";

const folder = DriveApp.getFolderById(folderId);

// Get the active spreadsheet

const spreadsheet = SpreadsheetApp.getActiveSpreadsheet();

const sheet = spreadsheet.getActiveSheet();

// Write headers to the spreadsheet

sheet.getRange(1, 1).setValue("Name");

sheet.getRange(1, 2).setValue("URL");

sheet.getRange(1, 3).setValue("Document ID");

// Recursively search through the folder and its subfolders

searchFolder(folder, sheet);

}

function searchFolder(folder, sheet) {

const files = folder.getFiles();

while (files.hasNext()) {

const file = files.next();

const type = getFileType(file.getMimeType());

if (type === "Google Sheet" || type === "Google Doc" || type === "Google Slides") {

const name = file.getName();

const url = file.getUrl();

const id = getIdFromUrl(url);

const column = sheet.getLastColumn() + 1;

sheet.getRange(1, column).setValue(name);

sheet.getRange(2, column).setValue(url);

sheet.getRange(3, column).setValue(id);

}

}

const folders = folder.getFolders();

while (folders.hasNext()) {

const subfolder = folders.next();

searchFolder(subfolder, sheet);

}

}

function getFileType(mimeType) {

switch (mimeType) {

case MimeType.GOOGLE\_SHEETS:

return "Google Sheet";

case MimeType.GOOGLE\_DOCS:

return "Google Doc";

case MimeType.GOOGLE\_SLIDES:

return "Google Slides";

default:

return "Other";

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code for retrieving the URLs from row 3 (Extracting URLs)

function extractUrlsAndPasteToSheet() {

// Get the active spreadsheet

const spreadsheet = SpreadsheetApp.getActiveSpreadsheet();

// Get the active sheet within the spreadsheet

const sheet = spreadsheet.getActiveSheet();

const documentIds = sheet.getRange(3, 1, 1, sheet.getLastColumn()).getValues()[0];

for (let i = 0; i < documentIds.length; i++) {

const documentId = documentIds[i];

if (!documentId) continue; // Skip empty cells

let urls;

try {

if (isGoogleSheet(documentId)) {

urls = extractUrlsFromSheet(documentId);

} else if (isGoogleDoc(documentId)) {

urls = extractUrlsFromDocument(documentId);

} else if (isGoogleSlide(documentId)) {

urls = extractUrlsFromSlides(documentId);

} else {

urls = ["No URLs"];

}

// Paste the extracted URLs into the column under the document ID

const pasteRange = sheet.getRange(4, i + 1, urls.length, 1);

pasteRange.setValues(urls.map(url => [url]));

} catch (error) {

Logger.log(`Error processing document ID ${documentId}: ${error}`);

sheet.getRange(4, i + 1).setValue("Error");

}

}

Logger.log("URLs extraction and pasting completed.");

}

function isGoogleSheet(sheetId) {

try {

SpreadsheetApp.openById(sheetId);

return true;

} catch (error) {

return false;

}

}

function isGoogleDoc(docId) {

try {

Docs.Documents.get(docId);

return true;

} catch (error) {

return false;

}

}

function isGoogleSlide(slideId) {

try {

SlidesApp.openById(slideId);

return true;

} catch (error) {

return false;

}

}

function extractUrlsFromSheet(sheetId) {

const sheet = SpreadsheetApp.openById(sheetId).getActiveSheet();

const rangeValues = sheet.getDataRange().getValues();

const urls = [];

for (let i = 0; i < rangeValues.length; i++) {

for (let j = 0; j < rangeValues[i].length; j++) {

const cellValue = rangeValues[i][j];

if (typeof cellValue === 'string') {

const urlRegex = /(https?:\/\/[^\s]+)/g;

const matches = cellValue.match(urlRegex);

if (matches) {

urls.push(...matches);

}

}

}

}

return urls.length > 0 ? urls : ["No URLs"];

}

function extractUrlsFromDocument(documentId) {

const documentContent = Docs.Documents.get(documentId).body.content;

const urls = [];

JSON.parse(JSON.stringify(documentContent), (key, value) => {

if (key === "url") urls.push(value);

});

return urls.length > 0 ? urls : ["No URLs"];

}

function extractUrlsFromSlides(slideId) {

const presentation = SlidesApp.openById(slideId);

const slides = presentation.getSlides();

const urls = [];

slides.forEach(slide => {

const shapes = slide.getShapes();

shapes.forEach(shape => {

const text = shape.getText();

const links = text.getLinks();

links.forEach(link => {

const url = link.getTextStyle().getLink().getUrl();

urls.push(url);

});

});

});

return urls.length > 0 ? urls : ["No URLs"];

}