**Main Objective:**

You are a prompt engineer expert in creating prompts for other AI Developer Agent, your goal will always be to provide structured instructions for another AI to do the work that your user inputs to you. Therefore, you will grab the provided user information, process it and deliver a step-by-step instruction for the next agent to follow.

Focus task: User wants to create Photoshop Plugins to help automate the repetitive tasks of the designing team in a furniture company.

User Input: The designing team had this script with different .js files that automated the process of creating different layers of the final edited product images to later export them into a new folder manually. However, the script only works in Photoshop 2023 version or older ones. The goal is to adapt the script to the new Photoshop 2025 version to make it compatible, therefore, creating a simple one file script that can be run with a simple interface plugin. Below in the context I provide you with two things; 1. The old script files. 2. An example of a simplistic plugin interface.

Analyse the old script, investigate what is the problem with it and provide me a one file script with the exact same functionality as it is now but adapted to be compatible with the newest Photoshop version. Then, create a simple plugin as the one provided with only one button to run this script when pushed.

**Context Index:**

1. Old script files (The repository of all the old files needed to be adapted to be compatible while maintaining the same exact functionalities):
   1. “add suffix.js”:
   2. “BesoLUX editing.atn”:
   3. “CandM 2.js”:
   4. “collapse.js”:
   5. “rename.js”:
   6. “slelect\_layers.js”:
   7. “SQUARES.atm”:
   8. “two\_up.js”:
2. Simple plugin Example:
   1. “index.html”:
   2. “manifest.json”:
   3. “package.json”:
   4. “README.md”:

**Context:**

1. Old script files:
   1. “add suffix.js”:

function main() {

    // user settings

    var prefs = new Object();

    prefs.nameSeparator = '-';  // character to insert between the layer name and number (default: ' ')

    prefs.topToBottom   = false; // rename layers top to bottom (false) or bottom to top (true)

    // prompt for layer name

    prefs.layerPattern = prompt('Enter the view number:\n');

    // rename layers

    if (prefs.layerPattern) {

        addSuffix(activeDocument, prefs);

    }

}

///////////////////////////////////////////////////////////////////////////////

// addSuffix - rename layers, top to bottom, or bottom to top

///////////////////////////////////////////////////////////////////////////////

function addSuffix(ref, prefs) {

    // declare local variables

    var len = ref.layers.length;

    // rename layers top to bottom

    if (prefs.topToBottom) {

        for (var i = 0; i < len; i++) {

            rename();

        }

    }

    // rename layers bottom to top

    else {

        for (var i = len - 1; i >= 0; i--) {

            rename();

        }

    }

    // rename - rename layer

    function rename() {

        var layer = ref.layers[i];

        var vis = layer.visible;

        // check for groups

        if (layer.typename == 'LayerSet') {

    //      addSuffix(layer, prefs);

    return 0;

        }

        // rename layer

        else {

            layer.name = layer.name + prefs.nameSeparator + prefs.layerPattern;

            if (!vis) {

                layer.visible = false;

            }

            prefs.countFrom++;

        }

    }

}

///////////////////////////////////////////////////////////////////////////////

// isCorrectVersion - check for Adobe Photoshop CS (v8) or higher

///////////////////////////////////////////////////////////////////////////////

function isCorrectVersion() {

    if (parseInt(version, 10) >= 8) {

        return true;

    }

    else {

        alert('This script requires Adobe Photoshop CS or higher.', 'Wrong Version', false);

        return false;

    }

}

///////////////////////////////////////////////////////////////////////////////

// isOpenDocs - ensure at least one document is open

///////////////////////////////////////////////////////////////////////////////

function isOpenDocs() {

    if (documents.length) {

        return true;

    }

    else {

        alert('There are no documents open.', 'No Documents Open', false);

        return false;

    }

}

///////////////////////////////////////////////////////////////////////////////

// showError - display error message if something goes wrong

///////////////////////////////////////////////////////////////////////////////

function showError(err) {

    if (confirm('An unknown error has occurred.\n' +

        'Would you like to see more information?', true, 'Unknown Error')) {

            alert(err + ': on line ' + err.line, 'Script Error', true);

    }

}

///////////////////////////////////////////////////////////////////////////////

// test initial conditions prior to running main function

///////////////////////////////////////////////////////////////////////////////

if (isCorrectVersion() && isOpenDocs()) {

    try {

        main();

    }

    catch(e) {

        if (e.number != 8007) { // don't report error on user cancel

            showError(e);

        }

    }

}

* 1. “BesoLUX editing.atn”:

Attached as a document.

* 1. “CandM 2.js”:

var doc = app.activeDocument; //references current document

var index = doc.layers.length - 1;

for(var index; index > 0; index--)

app.doAction("create and merge 2.0","BesoLUX editing");

doc.activeLayer = doc.layers[index];

doc.activeLayer.visible = false; //keeps layer invisible

* 1. “collapse.js”:

var doc = app.activeDocument; //references current document

app.runMenuItem(stringIDToTypeID('collapseAllGroupsEvent'));

doc.activeLayer.visible = false; //keeps layer invisible

* 1. “rename.js”:

function main() {

    // user settings

    var prefs = new Object();

    prefs.countFrom     = 1;    // number to start counting from (default: 1)

    prefs.zeroPadding   = 1;    // number of digits to use for the layer number (defaul: 3)

    prefs.nameSeparator = '';  // character to insert between the layer name and number (default: ' ')

    prefs.topToBottom   = false; // rename layers top to bottom (false) or bottom to top (true)

    // prompt for layer name

    prefs.layerPattern = prompt('Enter the name.\n');

    // rename layers

    if (prefs.layerPattern) {

        renameLayers(activeDocument, prefs);

    }

}

///////////////////////////////////////////////////////////////////////////////

// renameLayers - rename layers, top to bottom, or bottom to top

///////////////////////////////////////////////////////////////////////////////

function renameLayers(ref, prefs) {

    // declare local variables

    var len = ref.layers.length;

    // rename layers top to bottom

    if (prefs.topToBottom) {

        for (var i = 0; i < len; i++) {

            rename();

        }

    }

    // rename layers bottom to top

    else {

        for (var i = len - 1; i >= 0; i--) {

            rename();

        }

    }

    // rename - rename layer

    function rename() {

        var layer = ref.layers[i];

        var vis = layer.visible;

        // check for groups

        if (layer.typename == 'LayerSet') {

    //      renameLayers(layer, prefs);

    return 0;

        }

        // rename layer

        else {

            layer.name = prefs.layerPattern + prefs.nameSeparator +

            (prefs.countFrom.toString());

        //      (prefs.countFrom + Math.pow(10, prefs.zeroPadding)).toString().substr(1);

            if (!vis) {

                layer.visible = false;

            }

            prefs.countFrom++;

        }

    }

}

///////////////////////////////////////////////////////////////////////////////

// isCorrectVersion - check for Adobe Photoshop CS (v8) or higher

///////////////////////////////////////////////////////////////////////////////

function isCorrectVersion() {

    if (parseInt(version, 10) >= 8) {

        return true;

    }

    else {

        alert('This script requires Adobe Photoshop CS or higher.', 'Wrong Version', false);

        return false;

    }

}

///////////////////////////////////////////////////////////////////////////////

// isOpenDocs - ensure at least one document is open

///////////////////////////////////////////////////////////////////////////////

function isOpenDocs() {

    if (documents.length) {

        return true;

    }

    else {

        alert('There are no documents open.', 'No Documents Open', false);

        return false;

    }

}

///////////////////////////////////////////////////////////////////////////////

// showError - display error message if something goes wrong

///////////////////////////////////////////////////////////////////////////////

function showError(err) {

    if (confirm('An unknown error has occurred.\n' +

        'Would you like to see more information?', true, 'Unknown Error')) {

            alert(err + ': on line ' + err.line, 'Script Error', true);

    }

}

///////////////////////////////////////////////////////////////////////////////

// test initial conditions prior to running main function

///////////////////////////////////////////////////////////////////////////////

if (isCorrectVersion() && isOpenDocs()) {

    try {

        main();

    }

    catch(e) {

        if (e.number != 8007) { // don't report error on user cancel

            showError(e);

        }

    }

}

* 1. “slelect\_layers.js”:

var doc = app.activeDocument;

var myLayers = doc.layers;

for(var i = 0; i < myLayers.length; i++){

    //alert(myLayers[i].name);

    if(myLayers[i].kind == LayerKind.NORMAL){

        selectFunc(myLayers[i].name);

        }

    else{

        deselectFunc(myLayers[i].name);

        }

    }

function selectFunc(name){

    var idslct = charIDToTypeID( "slct" );

    var pepsi = new ActionDescriptor();

    var idnull = charIDToTypeID( "null" );

    var google = new ActionReference();

    var idLyr = charIDToTypeID( "Lyr " );

    google.putName( idLyr, name );

    pepsi.putReference( idnull, google );

    var idselectionModifier = stringIDToTypeID( "selectionModifier" );

    var idselectionModifierType = stringIDToTypeID( "selectionModifierType" );

    var idaddToSelection = stringIDToTypeID( "addToSelection" );

    pepsi.putEnumerated( idselectionModifier, idselectionModifierType, idaddToSelection );

    var idMkVs = charIDToTypeID( "MkVs" );

    pepsi.putBoolean( idMkVs, false );

    executeAction( idslct, pepsi, DialogModes.NO );

    }

function deselectFunc(name){

    var idslct = charIDToTypeID( "slct" );

    var pepsi = new ActionDescriptor();

    var idnull = charIDToTypeID( "null" );

    var google = new ActionReference();

    var idLyr = charIDToTypeID( "Lyr " );

    google.putName( idLyr, name );

    pepsi.putReference( idnull, google );

    var idselectionModifier = stringIDToTypeID( "selectionModifier" );

    var idselectionModifierType = stringIDToTypeID( "selectionModifierType" );

    var idremoveFromSelection = stringIDToTypeID( "removeFromSelection" );

    pepsi.putEnumerated( idselectionModifier, idselectionModifierType, idremoveFromSelection );

    var idMkVs = charIDToTypeID( "MkVs" );

    pepsi.putBoolean( idMkVs, false );

    executeAction( idslct, pepsi, DialogModes.NO );

    }

* 1. “SQUARES.atm”:

Attached as a document.

* 1. “two\_up.js”:

var doc = app.activeDocument; //references current document

var index = 0;

for(var i = 0; i < doc.layers.length; i++){

  if(doc.activeLayer.name === doc.layers[i].name){

    index = i - 2;

    if(index < 0)

      index = 0;

    break;

  }

}

doc.activeLayer = doc.layers[index];

doc.activeLayer.visible = false; //keeps layer invisible

1. Simple plugin Example:
   1. “index.html”:
2. <!DOCTYPE html>
3. <html>
4. <head>
5. <script src="index.js"></script>
6. </head>
7. <style>
8. body {
9. color: white;
10. padding: 0 16px;
11. }
12. li:before {
13. content: '• ';
14. width: 3em;
15. }
16. #layers {
17. border: 1px solid #808080;
18. border-radius: 4px;
19. padding: 16px;
20. overflow: scroll;
21. height: 160px;
22. }
23. footer {
24. display: flex;
25. justify-content: space-around;
26. }
27. </style>
28. <body>
29. <sp-heading id="layer-title">Layers</sp-heading>
30. <sp-body id="layers">
31. No layers
32. </sp-body>
33. <footer>
34. <sp-button id="btnAlert">Show Alert</sp-button>
35. <sp-button id="btnPopulate">Populate Layers</sp-button>
36. <sp-button id="btnCLear">Clear Layer List</sp-button>
37. </footer>
38. </body>
39. </html>

* 1. “index.js”:

function showLayerNames() {

    const app = require("photoshop").app;

    if (app.documents.length == 0) {

      showAlert("Please open at least one document.");

      return;

    }

    const activeDocTitle = app.activeDocument.title;

    document.getElementById("layer-title").innerHTML = `Layers for document ${activeDocTitle}`;

    // Collection classes are proxies on arrays. So if you want to push on them and alter them

    // you need to copy them into a proper array

    let allLayers = Array.from(app.activeDocument.layers);

    const allLayerNames = [];

    // We do not have a way to get a flat list of all layer names in a document

    // but you can use layers and Array.reduce to recursively collect names

    while (allLayers.length > 0) {

      const layer = allLayers.shift();

      allLayerNames.push(layer.name);

      if (layer.layers) {

        layer.layers.forEach(l => allLayers.push(l));

      }

    }

    const sortedNames = allLayerNames.sort((a, b) => a.toUpperCase() < b.toUpperCase() ? -1 : a.toUpperCase() > b.toUpperCase() ? 1 : 0);

    document.getElementById("layers").innerHTML = `

      <ul>${

        sortedNames.map(name => `<li>${name}</li>`).join("")

      }</ul>`;

}

async function showAlert(message) {

  // the "async" is required here because we're doing an "await" for the showAlert function

  const app = require('photoshop').app;

  await app.showAlert(message);

}

function clearList() {

  document.getElementById("layers").innerHTML = "";

}

document.getElementById("btnPopulate").addEventListener("click", showLayerNames);

document.getElementById("btnAlert").addEventListener("click", function () {showAlert("hello world")});

document.getElementById("btnCLear").addEventListener("click", clearList);

* 1. “manifest.json”:

{

  "id": "com.adobe.hello-world-panel-sample",

  "name": "Sample: Hello World Panel",

  "version": "1.0.0",

  "main": "index.html",

  "host": [

    {

      "app": "PS",

      "minVersion": "23.0.0"

    }

  ],

  "manifestVersion": 4,

  "entrypoints": [

    {

      "type": "panel",

      "id": "helloworld",

      "minimumSize": {

        "width": 230,

        "height": 200

      },

      "maximumSize": {

        "width": 2000,

        "height": 2000

      },

      "preferredDockedSize": {

        "width": 230,

        "height": 300

      },

      "preferredFloatingSize": {

        "width": 400,

        "height": 300

      },

      "label": {

        "default": "Hello World Sample"

      },

      "icons": [

        {

          "width": 23,

          "height": 23,

          "path": "icons/dark.png",

          "scale": [

            1,

            2

          ],

          "theme": [

            "darkest",

            "dark",

            "medium"

          ]

        },

        {

          "width": 23,

          "height": 23,

          "path": "icons/light.png",

          "scale": [

            1,

            2

          ],

          "theme": [

            "lightest",

            "light"

          ]

        }

      ]

    }

  ],

  "icons": [

    {

      "width": 48,

      "height": 48,

      "path": "icons/plugin-icon.png",

      "scale": [

        1,

        2

      ],

      "theme": [

        "darkest",

        "dark",

        "medium",

        "lightest",

        "light",

        "all"

      ],

      "species": [

        "pluginList"

      ]

    }

  ]

}

* 1. “package.json”:

{

  "name": "hello-world-panel-sample",

  "version": "1.0.0",

  "description": "Sample Photoshop plugin demonstrating a panel with buttons.",

  "author": "Adobe Inc",

  "license": "Apache-2.0"

}

* 1. “README.md”:

**# Starter Photoshop Plugin (Vanilla)**

This sample plugin shows a panel that interacts with Photoshop to get layer names from the active document. It also shows how to add text at runtime to your panel's html

**## Loading in Photoshop**

You can load this plugin directly in Photoshop by using the UXP Developer Tools application. Once started, click "Add Plugin...", and navigate to the "manifest.json" file in this folder. Then click the ••• button next to the corresponding entry in the developer tools and click "Load". Switch over to Photoshop, and the plugin's panel will be running.

**Response Style:**

* Use the best advanced prompting techniques (prompt structure, prompt coherence, prompt clarity, …) For the next Agent “Plugin Programmer Agent” to perfectly understand the task and provide the best response possible reducing errors.
* Always investigate the latest information and revise your response to make sure it’s accurate to accomplish the goal.