УДК 004.414.2:37.091.3 **DOI: https://doi.org/10.30837/PMW.2025.T1.044**

ВИКОРИСТАННЯ СКРИПТІВ У PHOTOSHOP В МЕЖАХ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ «ПЗКВС»

Бізюк А.В.
к.т.н., професор, кафедра «Медіасистеми та технології»,
Харківський національний університет радіоелектроніки

Анотація. *В дослідженні розглянуті способи використання програмних засобів автоматизації роботи у Photoshop, а саме – скриптів мовою JavaScript та, відповідно, впровадження описаних засобів в процесі лабораторних занять з навчальної дисципліни «Програмні засоби комп'ютерних видавничих систем» кафедри МСТ ХНУРЕ.*

Ключові слова: *СКРИПТ, PHOTOSOP, АВТОМАТИЗАЦІЯ, НАВЧАЛЬНИЙ ПРОЦЕС.*

Вступ

У сучасному світі цифрових технологій графічний редактор Adobe Photoshop є невід'ємною частиною навчальних програм у сферах дизайну, медіа, реклами та цифрового мистецтва. Цей потужний інструмент широко застосовується не лише у професійному середовищі, а й у закладах освіти для формування в учнів та студентів практичних навичок роботи з графічними зображеннями. Одним з важливих напрямів ефективного використання Photoshop у навчальному процесі є автоматизація рутинних дій за допомогою скриптів. Цей підхід дозволяє оптимізувати робочі процеси, заощадити час, а також навчити студентів думати алгоритмічно – що є ключовим елементом сучасної цифрової грамотності.

Мета та задачі дослідження

Впровадження скриптів у навчальні курси відкриває нові можливості для міждисциплінарного підходу, де поєднуються знання з інформатики, програмування та дизайну. Викладачі можуть не лише навчати базовим функціям Photoshop, а й показувати, як за допомогою скриптів можна налаштувати програму під власні потреби, створювати власні інструменти й навіть розробляти повноцінні автоматизовані робочі процеси.

Мета дослідження – розкрити можливості використання скриптів Photoshop в межах навчального процесу, а також надати практичні приклади того, як їх можна інтегрувати у заняття для підвищення ефективності навчання та зацікавленості студентів.

Суттєвим аспектом поточного дослідження є поліпшення методики викладання навчальної дисципліни «Програмні засоби комп'ютерних видавничих систем» (ПЗКВС) кафедри МСТ ХНУРЕ, в межах відведеного часу відповідно до нагальних потреб видавництв та поліграфічних виробництв регіону. Оптимальним

з точки зору поєднання теоретичного та практичного навчання буде опрацювання типових ситуацій, які зустрічаються у повсякденній практиці видавництв стосовно автоматизації роботи на прикладі Photoshop.

В рамках дослідження планується розглянути типові завдання, в яких виникає потреба автоматизації роботи та можливий шлях виконання таких завдань через створення скриптів мовою JavaScript.

Основна частина

У галузі «Видавництво та поліграфія» існують публікації, які розглядають використання скриптів у Adobe Photoshop для автоматизації процесів додрукарської підготовки. Одна з таких статей – «Апаратні та програмні засоби виготовлення електронних оригінал-макетів журнальної продукції» авторства І.М. Стельмаха, О.В. Зоренко та Т.Г. Осипової [5]. У цій роботі досліджуються макрокоманди та скрипти як засоби автоматизації процесу додрукарської підготовки видань, а також надаються рекомендації щодо вдосконалення технології верстання журнальної продукції.

Крім того, навчальний посібник «Технології видавництв та поліграфії. Курсова робота» під редакцією С.М. Зигулі, О.В. Зоренко, Н.Л. Талімонової та К.О. Чепурної містить рекомендації щодо виконання курсових робіт з дисципліни «Технології видавництв та поліграфії» [6]. Посібник охоплює питання конструювання видань, розробки технологічних процесів додрукарської підготовки, друкарських та післядрукарських процесів, що є важливими аспектами для студентів спеціальності «Видавництво та поліграфія».

Також корисною може бути стаття «Вчимося писати скрипти для Photoshop» від студії Віталія Комлєва [4], яка пояснює, як автоматизувати робочі процеси у Photoshop за допомогою написання скриптів, що може бути корисним для дизайнерів та спеціалістів у галузі поліграфії.

З матеріалів студентів та викладачів кафедри МСТ ХНУРЕ можна згадати статтю авторів Н.С. Дідик, А.В. Бізюк «Використання Java Script в програмі Adobe Photoshop» [3]. У цій статті обговорюється питання автоматизації рутинних операцій при обробці великої кількості фотографій. Автори розглядають інструменти автоматизації, такі як екшени (actions), скрипти (scripts) та дроплети (droplets), які доступні в Adobe Photoshop.

Офіційна документація Adobe також надає детальну інформацію про можливості використання скриптів у Photoshop, що може бути корисним для фахівців видавничо-поліграфічної галузі.

Ці матеріали можуть бути корисними для розуміння та впровадження скриптів у процеси додрукарської підготовки та верстання у сфері видавництва та поліграфії.

1 Що таке скрипти в Photoshop

Скрипти Photoshop – це невеликі спеціальні програми, які дозволяють автоматизувати повторювані або складні завдання. Вони виконуються в середовищі Photoshop і взаємодіють із різними його компонентами: документами, шарами, виділеннями, фільтрами тощо. Скрипти значно розширюють можливості користувача, дозволяючи створювати гнучкі й настроювані інструменти.

На відміну від «Дій» (Actions), які фіксують послідовність команд у графічному інтерфейсі, скрипти пишуться мовами програмування і можуть включати:

- умови (if, else);
- цикли (for, while);
- логіку перевірки об’єктів;
- взаємодію з файловою системою;
- створення графічних елементів з нуля.

Photoshop підтримує три основні мови для створення скриптів.

JavaScript (ExtendScript) для Photoshop – це потужний інструмент для автоматизації та розширення можливостей цього графічного редактора. ExtendScript є спеціальною версією JavaScript, адаптованою для роботи з продуктами Adobe. Вона дозволяє розробляти скрипти, які автоматизують рутинні процеси в Photoshop, такі як масове оброблення зображень, зміну розмірів, конвертацію форматів, налаштування стилів шарів та інше. Всі ці операції можна виконати через пряме керування об’єктами Photoshop, такими як документи, шари та маски, використовуючи стандартний синтаксис JavaScript.

Ключовою особливістю ExtendScript є його можливість працювати з об'єктною моделлю Photoshop (DOM), що дає доступ до всіх аспектів програми – від основних функцій, таких як відкриття та збереження файлів, до складних операцій, наприклад, зміна властивостей шарів або застосування фільтрів. Вбудоване середовище ExtendScript Toolkit дозволяє не тільки писати та тестувати скрипти, але й забезпечує інтерактивну налагоджувальну панель для виведення логів та відстеження помилок. Завдяки цьому розробка скриптів стає значно простішою, а сам процес автоматизації – більш доступним навіть для новачків у програмуванні.

VBScript для Photoshop – це один із методів автоматизації завдань в Adobe Photoshop, хоча цей підхід використовується значно рідше, ніж JavaScript (ExtendScript). VBScript є мовою сценаріїв, яка традиційно використовується для автоматизації процесів у середовищі Windows, і може бути застосована для взаємодії з Photoshop через COM-об'єкти. Скрипти на VBScript дозволяють користувачам автоматизувати такі операції, як обробка зображень, зміна їх властивостей, додавання текстів чи графічних елементів, а також масова обробка документів. VBScript надає змогу автоматизувати такі завдання, як відкриття, редагування та збереження файлів, але на відміну від ExtendScript, ця мова обмежена лише середовищем Windows.

Основною перевагою VBScript є його простота для користувачів, знайомих з Windows-операційними системами та бажаючих використовувати сценарії для автоматизації простих задач у Photoshop без потреби занурюватися в складніші мови програмування. Однак VBScript має значні обмеження у порівнянні з JavaScript, зокрема, відсутність доступу до більш потужних та гнучких можливостей об'єктної моделі Photoshop. Також важливим обмеженням є те, що VBScript працює лише в Windows-середовищі, що значно знижує його універсальність у порівнянні з JavaScript, який є кросплатформним.

AppleScript для Photoshop – це мова сценаріїв, яка використовується на платформах macOS для автоматизації завдань в Adobe Photoshop. AppleScript інтегрується з Photoshop через Apple Event – механізм, що дозволяє зовнішнім додаткам взаємодіяти один з одним. Скрипти на AppleScript дозволяють автоматизувати численні процеси в Photoshop, такі як обробка зображень, налаштування параметрів зображень, створення шаблонів, а також виконання складних обчислень та експорту файлів у різних форматах. Ця мова є відмінним інструментом для користувачів macOS, які хочуть автоматизувати специфічні завдання, не вдаючись до програмування на більш складних мовах.

AppleScript має високу зручність для користувачів Mac завдяки інтеграції з операційною системою, дозволяючи створювати сценарії для запуску додатків, а також взаємодіяти з іншими програмами в екосистемі Apple. Однак вона має певні обмеження, оскільки її використання обмежене тільки операційною системою macOS і не є кросплатформним. Крім того, хоча AppleScript має доступ до основних функцій Photoshop, він не так гнучкий та потужний, як JavaScript (ExtendScript), який забезпечує більш широкі можливості для програмування та автоматизації складних задач. Проте для користувачів macOS AppleScript є простим і ефективним інструментом для автоматизації повторюваних завдань у Photoshop.

JavaScript/ExtendScript має найбільшу гнучкість і підтримку, тому є основним вибором у навчальному процесі. Вибір JavaScript/ExtendScript для студентів-дизайнерів має кілька вагомих переваг, особливо в контексті їхнього навчання на програмуванні мовою C++ на другому курсі. Хоча C++ є потужною мовою, яка формує фундаментальні знання в програмуванні, для студентів-дизайнерів вона може бути занадто складною та абстрактною. Зазвичай, для творчих спеціальностей важче орієнтуватися в мовах, які вимагають глибокого розуміння концепцій, таких як управління пам'яттю, вказівники та складні структури даних. Замість цього, JavaScript/ExtendScript дає змогу студентам зосередитись на практичних завданнях, що безпосередньо пов'язані з їхньою професією, без необхідності занурюватись у складні технічні аспекти.

JavaScript/ExtendScript має набагато простішу синтаксичну структуру, порівняно з C++, що робить його доступнішим для студентів, які ще не мають великого досвіду в програмуванні. Ця мова дозволяє студентам швидко розпочати автоматизацію завдань в Photoshop, таких як зміна розмірів зображень, створення графічних елементів або обробка масових даних. Таким

чином, студенти можуть отримати безпосередні результати своєї праці, що стимулює їхню мотивацію до навчання програмуванню. Використання JavaScript дозволяє зв'язати творчу складову дизайну з технічною, що робить навчання більш інтегрованим і корисним.

Крім того, JavaScript є універсальним інструментом, який можна застосовувати не лише в Photoshop, а й у багатьох інших галузях веб-розробки та програмування. Для студентів-дизайнерів це відкриває додаткові можливості для розвитку їхніх навичок у майбутньому. Якщо вони освоять JavaScript, вони будуть мати перевагу при роботі з іншими програмами Adobe або веб-розробкою, де ця мова є стандартом. ExtendScript дає змогу студентам поступово освоювати основи програмування, використовуючи реальні практичні завдання, що допомагає їм поступово підвищувати рівень своїх знань і розвивати навички, які можуть бути корисними в їхній кар'єрі дизайнерів.

2 Різниця між скриптами, діями та плагінами

Скрипти, дії та плагіни – це три різні способи автоматизації процесів у Photoshop, кожен з яких має свої особливості та застосування. Дії – це набір команд, які записуються в Photoshop і виконуються в певній послідовності. Вони дозволяють автоматизувати рутинні завдання, такі як корекція кольорів, застосування фільтрів або конвертація файлів. Дії зручні для простих процесів, але вони не дозволяють налаштовувати логіку чи умови виконання команд. Вони виконуються без додаткового коду і є найлегшим способом автоматизації для користувачів Photoshop.

Скрипти, на відміну від дій, дають змогу більш детально контролювати процеси в Photoshop за допомогою програмування. Скрипти пишуться на мовах програмування, таких як JavaScript (ExtendScript), і дозволяють створювати складні алгоритми, які можна адаптувати до різних умов. Вони відкривають можливості для гнучкої автоматизації та інтеграції з іншими додатками. Плагіни ж є розширеннями, які можуть змінювати функціональність Photoshop на рівні інтерфейсу або додавати нові інструменти та ефекти. Вони потребують інсталяції та часто створюються за допомогою більш складних мов програмування, таких як C++ або JavaScript. Плагіни можуть бути потужними інструментами для дизайнерів, але їхній процес створення значно складніший, ніж написання скриптів або запис дій.

Отже, скрипти займають середню нішу між простотою дій і складністю плагінів, і тому чудово підходять для навчання основ програмування в контексті дизайну. Скрипти допомагають автоматизувати рутинні завдання, що економить час і зменшує ймовірність людських помилок. Вони корисні для:

– масової обробки зображень (зміна розміру, конвертація форматів, додавання водяних знаків);

– генерації складних ефектів або дизайнів, які важко або неможливо зробити вручну;

– оптимізації роботи дизайнерів, фотографів та ретушерів.

3 Де знайти та як запускати скрипти

Скрипти у форматі .jsx або .js можуть бути запущені в Photoshop через спеціальне меню програми, що дозволяє автоматизувати різноманітні завдання та оптимізувати робочий процес. Підтримка цих форматів в Adobe Photoshop надає користувачам можливість інтегрувати власні автоматизовані рішення без необхідності глибокого втручання в інтерфейс програми. Важливими є як стандартні, так і розширені можливості запуску скриптів, що сприяє гнучкості у роботі з програмою.

У Photoshop є вбудоване меню, яке дозволяє запускати скрипти безпосередньо з інтерфейсу програми. Для цього потрібно зайти в меню Файл > Сценарії > Запустити сценарій (File > Scripts > Run Script). Цей спосіб запуску дозволяє користувачам, навіть без досвіду програмування, швидко і просто виконати заданий сценарій. Вибір відповідного скрипту у форматах .jsx або .js дозволяє Photoshop виконувати автоматизовані завдання, наприклад, обробку зображень, застосування ефектів, створення шарів, зміну розмірів та багато інших операцій:

File → Scripts → Browse… – для запуску зовнішнього скрипта;

File → Scripts → [назва] – якщо скрипт попередньо збережено в папці Presets/Scripts.

Цей процес дозволяє користувачам не лише запускати готові скрипти, а й використовувати власні або сторонні скрипти для автоматизації рутинних задач, що значно економить час і підвищує ефективність роботи в Photoshop.

Інший варіант запуску скриптів у Photoshop – це через використання панелі Дії (Actions). У панелі Дії можна створювати та зберігати дії, що містять послідовність інструментів і команд Photoshop, а також скриптів. Це дає можливість автоматизувати складні робочі процеси за допомогою одного натискання кнопки. Хоча дії переважно використовуються для запису послідовності маніпуляцій у Photoshop, вони також дозволяють запускати скрипти, що значно розширює функціональність стандартних дій.

Для запуску скрипта через Дії відкрийте панель Дії (Window > Actions). Створіть нову дію, натиснувши іконку нового набору дій. У процесі запису дій виберіть Запустити сценарій (Run Script) і додайте скрипт до запису. Тепер при запуску цієї дії, буде виконано не лише стандартні команди Photoshop, але й заданий скрипт. Цей спосіб особливо корисний для професіоналів, які регулярно використовують одну й ту саму комбінацію дій і скриптів для досягнення бажаного результату.

Для більш досвідчених користувачів і розробників Adobe також надає можливість працювати зі спеціалізованою панеллю Сценарії (Script Panel), що дозволяє завантажувати і запускати скрипти безпосередньо з інтерфейсу Photoshop. Цей підхід дає більший контроль над сценаріями, оскільки дозволяє користувачам інтерфейсу програми не тільки запускати, а й редагувати, тестувати та відлагоджувати свої скрипти безпосередньо в середовищі Photoshop.

Одним із основних переваг запуску скриптів є значна економія часу. Якщо скрипт містить серію повторюваних операцій, таких як обробка зображень, зміна розмірів, створення шаблонів або автоматична корекція кольору, запуск цього скрипта може повністю автоматизувати процес, що в іншому випадку зайняв би кілька хвилин або годин ручної роботи. Це дозволяє користувачам зосередитись на більш креативних завданнях і зменшує ймовірність людських помилок.

Ще однією важливою перевагою є використання однакових налаштувань при роботі з багатьма зображеннями. Якщо є необхідність обробити великий обсяг зображень з однаковими параметрами (наприклад, масштабути всі зображення до певного розміру або змінити їх формат), скрипти значно полегшують цей процес, виконуючи операції на великій кількості файлів за лічені хвилини.

Отже, запуск скриптів у форматах .jsx або .js через меню Photoshop (рис. 1) є потужним інструментом для автоматизації та розширення функціональності програми. Це дає змогу користувачам значно підвищити ефективність роботи та знизити час, необхідний для виконання рутинних завдань. За допомогою стандартних або кастомізованих скриптів можна автоматизувати величезну кількість процесів, що не тільки економить час, але й допомагає уникнути помилок, що можуть виникнути при ручному виконанні однотипних операцій.

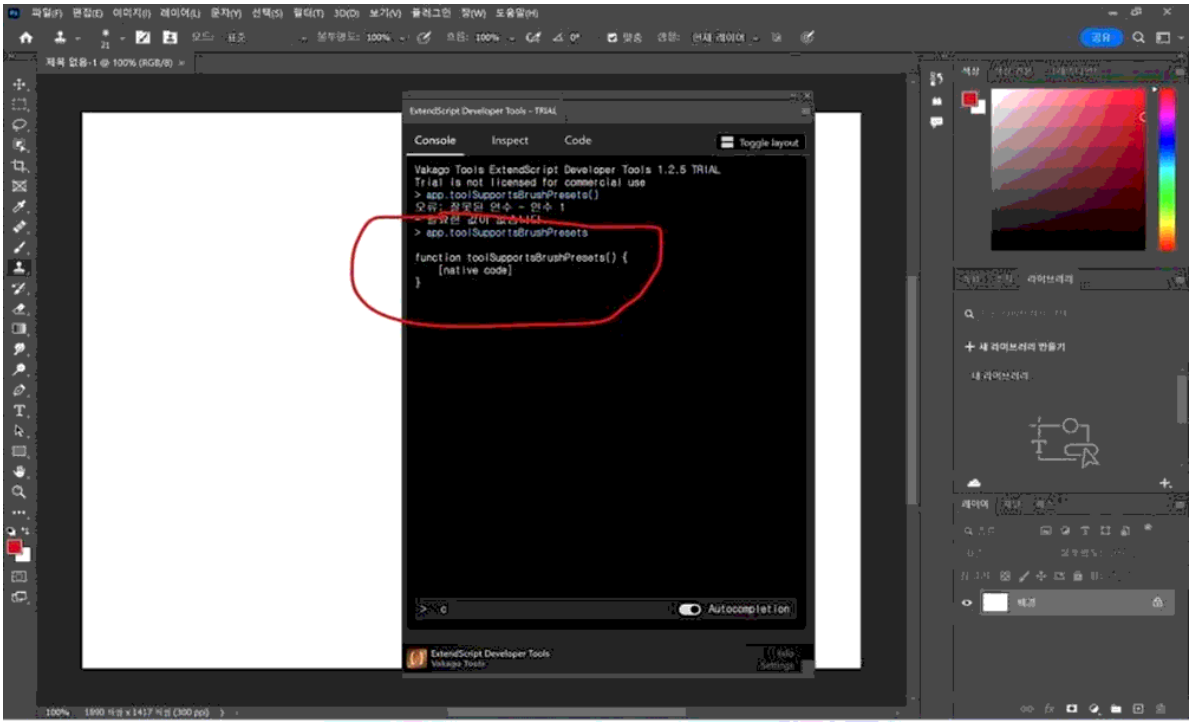

Рисунок 1 – Приклад застосування скрипта в середовищі Photoshop

Існує вбудований Script Editor (ExtendScript Toolkit), де можна писати, тестувати та налагоджувати скрипти (рис. 1). Це вбудоване середовище для написання та налагодження скриптів для програм Adobe, включаючи Photoshop. Воно дозволяє створювати, тестувати і відлагоджувати скрипти на JavaScript (ExtendScript), а також взаємодіяти з програмами Adobe за допомогою об'єктної моделі. Завдяки ExtendScript Toolkit можна писати код, перевіряти його

безпосередньо в середовищі програми, а також виводити налагоджувальні повідомлення, що робить процес розробки значно зручнішим і швидшим.

ExtendScript Toolkit є частиною стандартного пакету Adobe Creative Suite/Cloud, але в останніх версіях Creative Cloud цей інструмент може бути відсутній або замінений на нове середовище (наприклад, Visual Studio Code з додатковими розширеннями для Adobe).

Якщо у користувача встановлено ExtendScript Toolkit, його можна знайти у списку програм Adobe, а саме в папці Adobe Utilities (Windows) або в програмних додатках (Mac). Для запуску достатньо просто вибрати додаток у списку.

Після запуску ExtendScript Toolkit, відкривається інтерактивне середовище, де можна писати код, відлагоджувати його за допомогою панелі налагодження і виконувати безпосередньо на Photoshop або інших програмах Adobe. Для цього в меню «Target» вибирається потрібна програма (наприклад, Photoshop), і код можна тестувати в реальному часі. Цей інструмент також дозволяє виводити налагоджувальні повідомлення через панель "JavaScript Console", що дозволяє перевіряти правильність виконання скрипта та відстежувати помилки.

4 Освітня цінність скриптів

Використання скриптів у Photoshop має не лише технічну, а й важливу освітню цінність. Їхнє впровадження у навчальний процес сприяє розвитку критичного мислення, формує міждисциплінарні навички та допомагає краще зрозуміти принципи цифрової творчості та автоматизації.

Формування алгоритмічного мислення. Скрипти змушують студентів мислити поетапно, структуровано, з урахуванням логіки виконання команд. Це сприяє розумінню алгоритмів і послідовностей дій, здатності вирішувати завдання з використанням умов, циклів, змінних, перенесенню принципів програмування на практичні дизайнерські задачі.

Скрипти є містком між дисциплінами, такими як інформатика (програмування, логіка, структура коду), комп’ютерна графіка (робота з шарами, кольорами, композицією), проєктна діяльність (створення скриптів як частина творчого завдання). Такі завдання сприяють розвитку універсальних компетентностей: аналітичного мислення, здатності до самонавчання, дослідницької діяльності.

Навчання роботі зі скриптами дозволяє скорочувати час на рутинні дії під час виконання практичних завдань, таких, як пакетна обробка зображень, автоматичне перейменування чи розміщення елементів, створення шаблонів і макетів. Це підвищує ефективність роботи студентів і дозволяє зосередитися на творчих аспектах дизайну.

Застосування скриптів сприяє самостійному навчанню. Оскільки багато скриптів доступні у відкритому доступі, студенти можуть самостійно вивчати чужі приклади, адаптувати та модифікувати готові скрипти під свої потреби, створювати власні проєкти, орієнтуючись на потреби реального користувача. Це

стимулює дослідницьку активність та інтерес до програмування навіть у тих, хто навчається у творчих спеціальностях.

У сучасній професійній діяльності дизайнерів часто виникає потреба обробляти великі масиви даних (наприклад, сотні фото), працювати за встановленими шаблонами або брендовими гайдлайнами, інтегрувати Photoshop у більші виробничі процеси. Знання скриптів надає студенту конкурентну перевагу на ринку праці, оскільки він може працювати швидше, ефективніше й креативніше.

5 Практичне використання скриптів у навчальному процесі

Однією з найсильніших сторін використання скриптів у Photoshop є їхня практичність. Навчання через реальні приклади допомагає студентам не лише засвоїти інструменти, а й побачити їхню цінність у реальних робочих ситуаціях. Скрипти можна легко інтегрувати у навчальні проєкти, курсові завдання або лабораторні роботи.

У межах навчального курсу можна почати з простих, але корисних задач:

– автоматичне перейменування шарів. Скрипт, який перейменовує всі шари за заданим шаблоном: Шар_1, Шар_2 і т.д. Таке завдання розвиває навички роботи з циклами та об’єктною моделлю Photoshop;

– масова зміна розміру зображень. Скрипт, що відкриває всі зображення з папки, змінює розмір (наприклад, до 800x800 px) і зберігає в нову папку. Завдання може стати корисним для практик у вебдизайні, маркетингу або верстці;

– конвертація в інший формат (наприклад, PNG у JPG). Завдання для засвоєння базової структури скрипта, роботи з файлами та форматами.

У рамках проєктного або курсового навчання скрипти можуть стати основою для реалізації повноцінних практичних кейсів або мініпроєктів:

– створення шаблонів з динамічними елементами. Наприклад, скрипт, який автоматично вставляє логотип, додає текстову інформацію (ім’я, дату, позицію) та експортує готові зображення – це корисно при створенні бейджів, сертифікатів, афіш тощо;

– генератор колажів чи сіток зображень. Завдання для студентів: створити скрипт, що розміщує обрані фото в задану структуру (наприклад, 3x3) з автоматичним масштабуванням і вирівнюванням;

– аналіз структури документа. Скрипт, який виводить у вікні інформацію про кількість шарів, їх тип, розміри тощо. Таке завдання відзначається більшою складністю, але може бути корисним для розуміння внутрішньої логіки PSD-файлів.

Аналіз готових скриптів є важливою формою навчання для студентів-дизайнерів, особливо коли мова йде про освоєння автоматизації процесів в Photoshop. Вивчення готових скриптів дозволяє студентам з першого погляду зрозуміти, як саме працюють різні функції Photoshop, та отримати знання щодо того, як їх можна автоматизувати за допомогою коду. Аналізуючи структуру скриптів, студенти можуть побачити, як можна змінювати параметри, обробляти зображення, працювати з шарами та застосовувати фільтри. Це також допомагає

зрозуміти, як функціонує об'єктна модель Photoshop і як кожен елемент програми взаємодіє з іншими через скрипт.

Одним із основних переваг такого підходу є покрокове вивчення коду: студенти починають з простих скриптів і поступово переходять до складніших. Спостерігаючи за тим, як невеликі фрагменти коду змінюють роботу програми, студенти здобувають практичні навички і краще розуміють, як реалізувати власні сценарії для автоматизації завдань. Цей підхід дозволяє зняти психологічну перешкоду програмування, роблячи його більш доступним для студентів, що мають творчі спеціальності і не звикли до складних програмістських мов.

Аналіз готових скриптів також сприяє розвитку критичного мислення, адже студенти мають оцінювати код на ефективність, зрозумілість і можливість його вдосконалення. Вони вчаться не лише користуватися готовими рішеннями, а й шукати шляхи їх оптимізації або адаптації під свої потреби. В результаті цього процесу студенти можуть швидше та ефективніше застосовувати отримані знання у реальних проєктах. Такий підхід до навчання створює тісний зв'язок між теорією і практикою, дозволяючи студентам здобути досвід роботи з реальними інструментами, які вони можуть використовувати у своїй професійній діяльності як дизайнери.

Таку задачу можна подати як частину практичного модуля: «Розберіть роботу скрипта й адаптуйте його під індивідуальне завдання». Комбінування власного скрипта з готовими заскриптованими діями є потужним методом для підвищення ефективності обробки зображень у Photoshop. Використовуючи вже створені заскриптовані дії, такі як "copy", "rotate", або "scale", студенти можуть створити свої власні сценарії для автоматизації складних завдань. Наприклад, можна створити скрипт, який спочатку копіює зображення за допомогою команди app.activeDocument.copy(), потім повертає його на певний кут з використанням app.activeDocument.rotateCanvas(90) і після цього масштабує зображення до потрібного розміру за допомогою app.activeDocument.resizeImage(800, 600). Комбінуючи ці прості дії, студент може розробити більш складний процес обробки зображень, що виконуватиметься одним натисканням кнопки.

Таким чином, власний скрипт дозволяє не лише автоматизувати окремі дії, але й поєднувати їх у більш складний робочий процес, який можна адаптувати до різних потреб. Наприклад, для проекту, що вимагає одночасної обробки кількох зображень, можна створити скрипт, який застосовуватиме ці дії до кожного зображення в пакетному режимі. Студенти можуть використовувати цей підхід для створення набору автоматизованих дій для серійного оброблення зображень, що значно зекономить час і зменшить кількість помилок. Крім того, можливість комбінувати свої власні скрипти з існуючими діями дозволяє студентам гнучко налаштовувати процеси під конкретні завдання та створювати більш професійні рішення в Photoshop.

Студенти можуть працювати в командах над створенням власного скрипта, а потім презентувати його класу для обговорення, пояснити логіку роботи,

продемонструвати результат у реальному часі. Це стимулює розвиток soft skills – комунікації, командної роботи та вміння пояснювати технічний матеріал доступно.

6 Розробка власного скрипта як навчальне завдання

Створення власного скрипта у Photoshop – це не лише практичне програмування, а й цілісний навчальний процес, який включає аналіз задачі, логічне планування, кодування, тестування та демонстрацію результату. Це завдання можна інтегрувати у формат курсової, лабораторної роботи або командного мініпроєкту.

На початковому етапі студентам пропонується реальна або змодельована проблема, яку можна вирішити за допомогою скрипта.

Приклади типових задач:

- згенерувати серію сертифікатів із різними іменами;
- створити колаж із вибраних зображень у заданій сітці;
- автоматично розмістити логотип та водяний знак на фотографіях;
- масово зменшити розмір зображень та експортувати в інший формат.

Важливо, щоб завдання мало чітку ціль і передбачало автоматизацію дій, які вручну виконуються довго або неефективно.

Планування та алгоритм дій. Перед написанням коду студент має визначити послідовність дій, які виконує скрипт (алгоритм); зрозуміти, які об’єкти Photoshop будуть задіяні (документи, шари, текст, зображення), скласти в довільній формі блок-схему або псевдокод для візуалізації логіки. Цей етап формує аналітичне мислення та закладає основу структурованого програмування.

Робота зі скриптами в Adobe Photoshop за допомогою мови JavaScript (ExtendScript) надає студентам можливість створювати ефективні автоматизовані процеси для виконання повторюваних завдань. Однак, замість того, щоб створювати код з нуля, студенти часто мають можливість взяти готовий код та адаптувати його під свої конкретні потреби. Це підхід дозволяє заощадити час і швидко впроваджувати автоматизацію без необхідності глибокого занурення в програмування.

Одним із основних елементів при роботі з кодом є створення функцій. Студенти можуть використовувати вже готові функції для обробки зображень або маніпуляцій з шарами, таких як resizeImage(), rotateCanvas(), duplicate(), і адаптувати їх для своїх завдань. Наприклад, можна взяти код, який повертатиме зображення на певний кут, і змінити його так, щоб він працював для масового оброблення кількох файлів. Це дозволяє створити більш складний робочий процес, використовуючи уже готові частини коду, що робить програмування простішим для студентів, які ще не мають великого досвіду.

Важливими елементами роботи зі скриптами є оголошення змінних, що дозволяє зберігати значення для подальшої обробки, а також використання умов та циклів для вибору певних операцій в залежності від результатів виконаних дій. Студенти можуть адаптувати ці конструкції до своїх задач. Наприклад, для

повертання зображення на певний кут, типова змінна може бути оголошена як var rotationAngle = 90;, де значення 90 – це кут, на який потрібно повернути зображення. В подальшому цю змінну можна використати в методі app.activeDocument.rotateCanvas(rotationAngle);, щоб повернути зображення на вказаний кут. Також, для масштабування зображення, може бути використана змінна, наприклад, var newWidth = 800; та var newHeight = 600;, що задають нові розміри зображення в пікселях. Згодом ці змінні можна передати в метод app.activeDocument.resizeImage(newWidth, newHeight);, щоб змінити розміри поточного документа. Використовуючи змінні для таких операцій, студенти мають можливість налаштовувати параметри обробки зображень і легко змінювати їх у майбутньому для різних проектів чи умов.

Такий підхід дозволяє студентам поступово розвивати свої навички в програмуванні, одночасно отримуючи практичні результати, що можна використовувати у їхній професійній діяльності. Цей етап можна поєднувати з індивідуальними консультаціями або роботою в парах.

7 Тестування та усунення помилок

Навчання програмуванню для студентів-дизайнерів включає не лише написання кодів скриптів, але й освоєння базових принципів налагодження (дебагінгу). Це важливий аспект, оскільки навіть досвідчені програмісти часто стикаються з помилками, які потребують ретельного аналізу і виправлення. Один із перших кроків у процесі налагодження – це перевірка логіки виконання коду. Під час виконання скриптів студенти мають вміти визначити, чи працюють усі функції вірно, чи відбувається те, що очікується, чи виконуються всі команди в правильному порядку. Це важливо для того, щоб зрозуміти, чому певні частини коду не дають бажаного результату або не виконуються зовсім.

Одним із найбільш ефективних інструментів для налагодження скриптів є виведення інформації за допомогою команд для дебагу, таких як alert() чи $.writeln(). Студенти можуть використовувати ці функції для відстеження значень змінних, перевірки того, які частини коду виконуються, і виявлення точок, де виникають проблеми. Наприклад, використання alert() дозволяє вивести вікно з повідомленням, що містить значення певної змінної чи результат виконання функції, що є корисним для перевірки роботи коду під час розробки. Команда $.writeln() є більш зручним варіантом для виведення даних у консолі, що дозволяє побачити всі повідомлення в текстовому вигляді і здійснювати відслідковування процесу виконання коду без переривання роботи програми.

Під час навчання студенти також ознайомлюються з методами виправлення синтаксичних та логічних помилок. Виявлення помилок у коді може бути складним завданням, особливо для тих, хто тільки починає працювати з програмуванням. Студенти навчаються розрізняти синтаксичні помилки, які виникають через неправильне використання мови програмування (наприклад, забуті крапки з комами чи неправильний порядок аргументів), та логічні помилки, коли код працює, але дає неправильний результат через помилкову

структуру чи неправильну обробку даних. Вміння знайти і виправити такі помилки є ключовим для успішної розробки робочих скриптів.

8 Демонстрація результату

Після завершення роботи зі створенням і налаштуванням скрипта в Photoshop, важливим етапом є демонстрація результату та пояснення його роботи. Цей процес не лише підсумовує виконану роботу, а й дозволяє студентам зрозуміти, як їхній код взаємодіє з програмою, а також допомагає користувачам швидко освоїти інструмент, розроблений студентами. Пояснення і демонстрація дозволяють з'ясувати, як конкретний скрипт вирішує поставлене завдання, та дають можливість передати інші важливі відомості користувачам, які будуть працювати зі скриптом.

Перше, що має зробити студент після завершення розробки скрипта – це продемонструвати, як він працює. Студент запускає скрипт у реальному середовищі Photoshop і показує, як саме відбуваються зміни в документі після виконання коду. Наприклад, якщо скрипт створює новий шар або змінює розмір зображення, студент має показати, що ці операції виконуються автоматично без необхідності ручного втручання з боку користувача. Також важливо продемонструвати, які саме параметри були використані в коді – наприклад, розмір зображення, рівень яскравості чи застосовані фільтри.

Демонстрація включає покрокову ілюстрацію того, як скрипт може бути корисним у реальному робочому процесі. Для цього студент може відкрити кілька різних зображень, щоб показати, як один і той самий скрипт працює з різними типами файлів або даних. Також важливо звернути увагу на те, що скрипт може бути адаптований до інших завдань, якщо це передбачено.

Другим етапом є пояснення дій, які виконує скрипт. Студент повинен наочно продемонструвати, як кожен етап коду відповідає за конкретну операцію в Photoshop. Це включає в себе опис, які об'єкти програми змінюються за допомогою скрипта: чи це шари, чи зображення, чи параметри зображення, такі як розмір або колірні налаштування.

Пояснюючи код, студент може навести конкретні приклади і обговорити кожну змінну або функцію, що використовуються в скрипті. Наприклад, якщо скрипт змінює розмір зображення, студент повинен пояснити, як працює функція resizeImage() і які параметри передаються їй (ширина, висота). Якщо скрипт працює з шарами, важливо пояснити, як використовується об'єкт app.activeDocument.layers для доступу до шарів документа.

Крім технічного розуміння того, як працює скрипт, студент повинен вміти чітко і зрозуміло пояснити, які саме завдання виконує його код у середовищі Photoshop. Демонстрація результату, пояснення кожного етапу виконання коду, а також надання інструкції користувача – це важливі елементи навчання, які допомагають не тільки розробляти ефективні скрипти, але й передавати їх іншим користувачам, що зробить їх використання ще більш зручним та доступним.

Оцінювання роботи студентів в межах лабораторної роботи щодо розробки скриптів для Photoshop є важливим етапом навчання, оскільки дозволяє не лише перевірити технічні навички, але й оцінити здатність студента до творчого вирішення задач. Оцінювання зазвичай базується на кількох критеріях, кожен з яких важливий для успішного освоєння програмування та розробки ефективних рішень для автоматизації процесів у Photoshop. Основними критеріями є відповідність скрипта поставленій задачі, коректність та ефективність роботи скрипта, рівень самостійності та креативності, а також оформлення звітного опису.

Першим і найважливішим критерієм є відповідність скрипта поставленій задачі. Студент має чітко зрозуміти, яке завдання він повинен вирішити, і правильно реалізувати його за допомогою скрипта. Для цього важливо, щоб скрипт не лише виконував основну задачу, а й враховував усі нюанси, які були зазначені в інструкціях чи вимогах. Наприклад, якщо задача полягала в автоматизації зміни розміру зображень, то скрипт повинен виконати цю операцію швидко та ефективно, працюючи з кількома файлами або в певному форматі.

Цей критерій перевіряється через порівняння постановки завдання у звіті та реального функціонування скрипта. Оцінюючи цей критерій, варто звернути увагу на те, чи реалізовано всі функціональні можливості, які були передбачені для скрипта. Якщо студент створив скрипт, що відповідає вимогам, і успішно вирішує поставлену задачу, то оцінка за цим критерієм буде високою. Наприклад, скрипт, який дозволяє швидко змінювати розмір зображення без втрати якості, є правильним рішенням, якщо це була основна мета завдання.

Другим важливим критерієм є коректність та ефективність роботи скрипта. Скрипт має працювати без помилок, не викликати збій програми і не генерувати непотрібних помилок під час виконання. Коректність коду забезпечує, що всі функції та методи працюють належним чином, а не призводять до аварійних ситуацій або непередбачуваних результатів. Оцінка за цим критерієм залежить від того, чи наявні в коді ділянки дублювання операцій або невикористані операції.

Крім технічних аспектів, важливо оцінити рівень самостійності та креативності студента. Важливо, щоб студент проявив ініціативу та здатність розв’язувати складні завдання самостійно, а не просто слідував за запропонованим шаблоном або копіював готові рішення з Інтернету. Оцінка цього критерію залежить від того, наскільки студенти можуть ставити власні задачі, а надалі адаптувати та модифікувати готові скрипти для вирішення конкретних проблем. Це включає як використання стандартних функцій Photoshop, так і можливість написати оригінальний код для вирішення більш складних завдань. В наведеному на рис. 2 прикладі студентка виконала копіювання початкового декоративного елементу з восьмиразовим обертанням та масштабуванням. Обертання виконане навколо центральної точки.

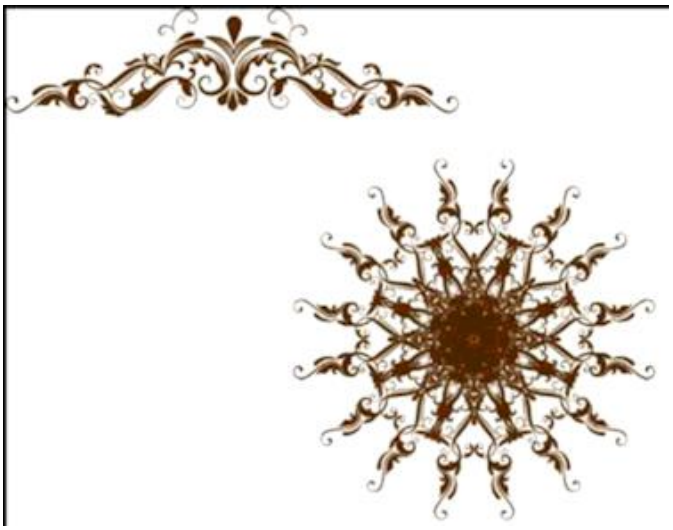

Рисунок 2 – Приклад 1 студентської роботи

Аналогічне завдання виконане в другому прикладі, але в цьому випадку обертання налаштоване навколо кутової точки, закладено 4 ітерації, по 45^{o} (рис. 3).

Рисунок 3 – Приклад 2 студентської роботи

Креативність також оцінюється за тим, як студент додає власні інноваційні елементи до свого проєкту. Наприклад, студент може створити скрипт, який не лише автоматизує рутинні операції, але й пропонує нові, нестандартні рішення або покращення для автоматизації складних задач у Photoshop. Це можуть бути, наприклад, елементи інтерфейсу користувача, які дозволять швидко змінювати налаштування скрипта.

Останнім критерієм оцінювання є оформлення звіту з роботи. Це включає в себе постановку завдання, алгоритм роботи скрипта, структуру коду, коментарі, а також презентацію результату. Добре структурований код з чіткими коментарями дозволяє не тільки іншому розробнику, а й самому студенту згодом легко зрозуміти, як працює скрипт. Коментарі мають бути чіткими та зрозумілими, пояснюючи складні частини коду, особливо якщо вони виконують специфічні або нетривіальні операції.

Оформлення звіту може включати також динамічну презентацію. Студент має представити свою роботу у зрозумілій та професійній формі, описавши

основні моменти та особливості розробленого скрипта. Це може бути коротка презентація в PowerPoint, яка пояснює роботу скрипта на анімованих прикладах.

Загальна оцінка роботи студента базується на балансі між технічними характеристиками скрипта, його функціональністю та творчим підходом. Важливим є те, щоб студент зміг не лише вирішити конкретну задачу, а й продемонструвати розуміння процесу розробки скриптів, вміння оптимізувати код і правильно оформляти звіт. Водночас оцінка за самостійність та креативність стимулює студентів до пошуку нових рішень і адаптації існуючих інструментів для виконання нестандартних завдань.

9 Технічні аспекти та інструменти

Для ефективної роботи зі скриптами в Photoshop студентам необхідно ознайомитися з технічними особливостями середовища розробки, файлами скриптів, методами тестування та базовими інструментами для створення коду. Цей розділ допомагає закріпити практичні знання на надійній технічній основі.

Основний формат скриптів для Photoshop – .jsx (JavaScript Extension). Це текстовий файл, який містить програмний код, написаний мовою JavaScript, адаптованою для Adobe-продуктів.

У типовому скрипті можна зустріти:

- змінні – для збереження параметрів;
- функції – для логічної організації коду;
- об’єкти Photoshop – app, documents, layers тощо;
- оператори умов і циклів – для керування логікою.

Ось приклади простих скриптів для Photoshop, написаних мовою JavaScript (ExtendScript), які виконують масштабування або поворот зображення.

Приклад 1. Масштабування зображення.

Цей скрипт змінює розміри поточного документа в Photoshop на задані значення:

```
// Отримуємо активний документ
var doc = app.activeDocument;
// Задаємо нові розміри (в пікселях)
var newWidth = 800; // нова ширина
var newHeight = 600; // нова висота
// Масштабуємо зображення
doc.resizeImage(newWidth, newHeight);
// Збереження файлу (необов'язково, можна додати додатковий код для збереження)
```

У цьому скрипті використовується метод resizeImage(), який масштабує зображення до вказаних ширини та висоти. Студенти мають звернути увагу на змінні newWidth та newHeight та змінити значення відповідно до своїх потреб.

Приклад 2. Поворот зображення.

Цей скрипт повертає зображення на певний кут (в даному випадку, на 90 градусів):

```
// Отримуємо активний документ
var doc = app.activeDocument;
// Задаємо кут повороту
var rotationAngle = 90; // кут у градусах
// Повертаємо зображення
doc.rotateCanvas(rotationAngle);
// Збереження файлу (необов'язково, можна додати додатковий код для збереження)
```

В другому скрипті метод rotateCanvas() дозволяє повернути зображення на заданий кут, в цьому випадку на 90 градусів. Як і в попередньому прикладі, студенти налаштовують змінну rotationAngle під свої потреби. Як ускладнення це може бути частина циклу, що дасть можливість накопичення повторюваних елементів, повернутих відносно один одного.

Ці скрипти можна запускати в Adobe Photoshop для автоматизації процесу масштабування та повороту зображень. Вони можуть бути легко адаптовані для більш складних завдань, наприклад, для пакетної обробки кількох зображень.

Photoshop автоматично виявляє скрипти, що зберігаються в папці (рис. 4):

📂 *Adobe Photoshop > Presets > Scripts.*

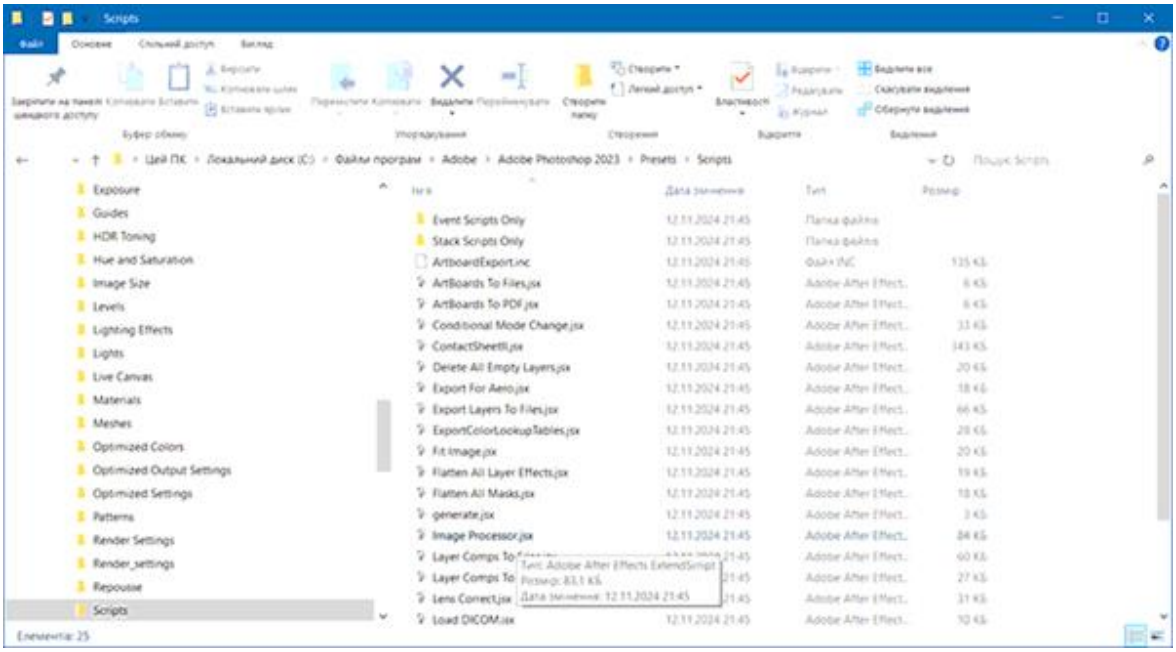

Рисунок 4 – Розташування скриптів в папках Windows

Після збереження файлу .jsx у цій папці, скрипт з’являється в меню: File → Scripts. Також можна запускати скрипти безпосередньо через: File → Scripts → Browse...

9.1 Інструменти для написання і тестування коду.

Офіційне середовище Adobe ExtendScript Toolkit для роботи зі скриптами (рис. 5) є потужним інструментом для розробки, тестування та налагодження скриптів для програм Adobe, таких як Photoshop, Illustrator, After Effects та інші.

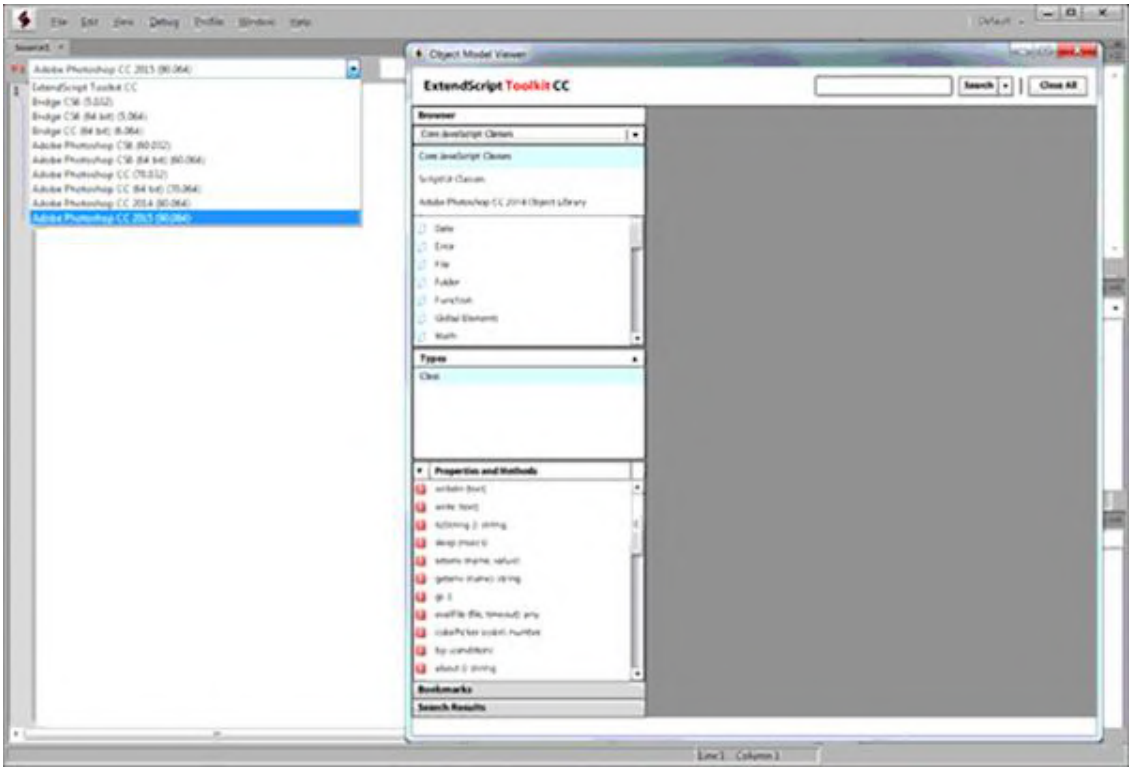

Рисунок 5 – Приклад роботи середовища ExtendScript

Хоча ExtendScript Toolkit більше не підтримується, сучасні рішення, такі як Adobe Visual Studio Code (VSCode) API, забезпечують комфортне середовище для розробки скриптів, включаючи зручні можливості для підсвічування синтаксису, доступ до документації об'єктної моделі та функції крокового виконання з виведенням логів.

Одна з основних переваг офіційного середовища Adobe для роботи зі скриптами – це підсвічування синтаксису. Ця функція значно полегшує процес програмування, оскільки дозволяє розпізнавати помилки в коді ще на етапі написання. У середовищі Adobe VSCode за допомогою спеціального розширення для ExtendScript синтаксис JavaScript (який використовується для написання скриптів для Adobe) підсвічується відповідно до стандартів мови. Ключові слова, змінні, функції та об'єкти мають різні кольори, що допомагає швидко орієнтуватися в коді. Наприклад, об'єкти Photoshop, такі як app.activeDocument, будуть відображатися іншим кольором, що допомагає уникнути помилок через неправильне використання об'єктів чи функцій.

Ще однією важливою перевагою є доступ до документації об’єктної моделі прямо в середовищі розробки. Об'єктна модель Adobe є комплексною та включає безліч об'єктів, методів та властивостей, які необхідно використовувати при написанні скриптів. В офіційному середовищі, таким як VSCode з розширенням для Adobe ExtendScript, студенти та розробники можуть легко знайти необхідні

функції та ознайомитися з їхнім синтаксисом. Документація забезпечує детальні описи кожного елементу об'єктної моделі, приклади використання та можливі параметри, що знижує ризик помилок. Крім того, це полегшує швидке освоєння нових функцій та методів при роботі з програмами Adobe.

Однією з ключових можливостей офіційного середовища є можливість крокового виконання та виведення логів через методи, такі як $.writeln(). Це забезпечує ефективне налагодження та перевірку логіки виконання коду. В середовищі VSCode для ExtendScript можна використовувати консоль для виведення логів під час виконання скриптів, що дозволяє побачити реальний хід виконання коду, значення змінних і інші важливі деталі. Наприклад, студенти можуть додавати в код рядки, як $.writeln("Розмір зображення: " + app.activeDocument.width);, що дозволяє вивести в консоль розміри поточного зображення в Photoshop. Це дуже корисно для налагодження, оскільки дає змогу на кожному етапі бачити, чи правильно працюють команди і чи відповідають вони очікуваним результатам.

Крокове виконання також дозволяє покроково проходити через кожну команду, зупиняючись на певних етапах виконання для детального аналізу. Це особливо корисно для виявлення логічних помилок, коли програма працює, але дає неправильний результат. Студенти можуть вручну пройти через кожен крок, побачити, де саме виникає проблема, і виправити її, що робить процес навчання більш інтерактивним і зрозумілим.

Отже, офіційне середовище Adobe для роботи зі скриптами надає студентам і розробникам безліч потужних інструментів для ефективного створення, тестування та налагодження скриптів. Підсвічування синтаксису, доступ до документації об’єктної моделі та можливість крокового виконання з виведенням логів дозволяють зменшити кількість помилок, швидше навчатися і ефективно працювати з кодом. Використання сучасного середовища, такого як Adobe VSCode, значно полегшує розробку скриптів і робить їх більш доступними для дизайнерів, які хочуть освоїти програмування для автоматизації процесів у Photoshop та інших програмах Adobe.

Visual Studio Code з розширенням Adobe (рис. 6). У новіших версіях Adobe Photoshop рекомендовано використовувати офіційне середовище Adobe для роботи зі скриптами, що включає використання Visual Studio Code (VS Code) з плагінами Adobe UXP Developer Tools або Adobe ExtendScript Debugger, надає потужний та сучасний інструмент для розробки, налагодження та тестування скриптів для програм Adobe, таких як Photoshop, Illustrator, InDesign тощо. Це середовище значно полегшує роботу з кодом завдяки таким функціям, як підсвічування синтаксису, інтегрована документація об'єктної моделі та можливість крокового виконання скриптів з виведенням логів.

Однією з основних переваг використання VS Code з плагіном Adobe ExtendScript Debugger або Adobe UXP Developer Tools є підсвічування синтаксису.

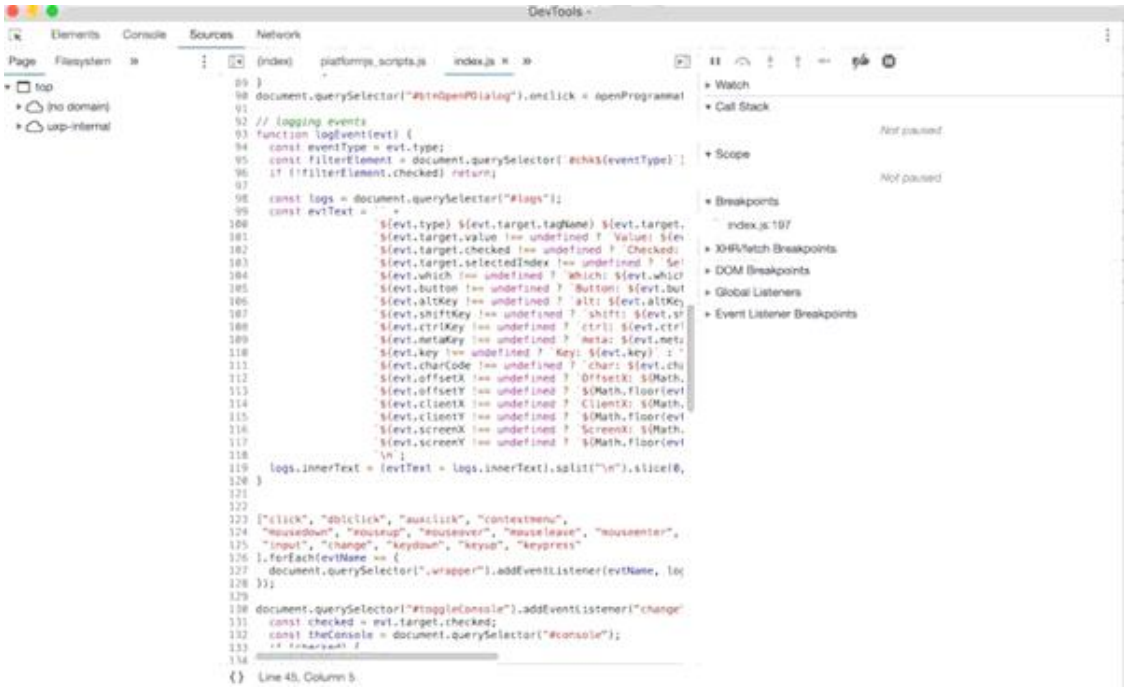

Рисунок 6 – Приклад роботи середовища VS Code

Плагіни для VS Code забезпечують підтримку мови JavaScript (ExtendScript) та JavaScript для UXP, що дозволяє автоматично підсвічувати синтаксичні елементи в коді. Завдяки цьому студенти можуть швидко розпізнати помилки, пов'язані з некоректним використанням змінних, функцій або об'єктів, а також легше орієнтуватися у великому коді. Наприклад, ключові слова та об'єкти Adobe, такі як app.activeDocument, Document, Layer тощо, будуть підсвічені різними кольорами, що дозволяє швидко ідентифікувати їх і уникати синтаксичних помилок.

Ще одним потужним інструментом є доступ до документації об’єктної моделі прямо в середовищі розробки. Плагін Adobe ExtendScript Debugger або Adobe UXP Developer Tools надає можливість інтеграції документації Adobe з Visual Studio Code, що дозволяє студентам і розробникам отримувати доступ до повної інформації про всі доступні об'єкти, методи та властивості. Вони можуть швидко дізнатися, як працює конкретний метод, яке значення параметрів йому передавати, а також подивитися приклади використання цих методів. Завдяки такому доступу студенти не лише автоматично отримують допомогу під час написання коду, а й зменшують ризик помилок завдяки чітким вказівкам документації.

Одна з найбільших переваг при роботі з VS Code – це можливість покрокового виконання коду та виведення логів, що особливо корисно для налагодження (дебагінгу). Плагін Adobe ExtendScript Debugger дає змогу виконувати скрипти по кроках, що дозволяє зупиняти виконання на кожному етапі та ретельно перевіряти, чи все працює так, як треба. Студенти можуть відслідковувати змінні, параметри та значення, що передаються в функції, і на основі цього виправляти помилки. Наприклад, за допомогою методу $.writeln() можна вивести в консоль значення змінних або результат виконання функцій на

кожному етапі виконання, що робить процес дебагінгу набагато зручнішим. Завдяки цьому студенти можуть покроково аналізувати код і знаходити помилки в логіці, що особливо важливо, коли мова йде про складніші скрипти з кількома умовами або циклами.

Отже, використання Visual Studio Code з плагінами Adobe ExtendScript Debugger або Adobe UXP Developer Tools є відмінним рішенням для студентів і розробників, які працюють зі скриптами для програм Adobe. Підсвічування синтаксису, доступ до документації об'єктної моделі та можливість крокового виконання з виведенням логів значно полегшують процес програмування і налагодження, роблячи його більш зручним і швидким. Завдяки цим функціям студенти можуть швидко знайти та виправити помилки, краще розуміти структуру коду та більш ефективно працювати з об'єктами Photoshop, Illustrator та інших продуктів Adobe. Використання VS Code з цими плагінами – це сучасний та потужний інструмент для розробки скриптів, що відкриває нові можливості для автоматизації та оптимізації робочих процесів у програмному забезпеченні Adobe.

9.2 Документація та ресурси для навчання.

Надання студентам доступу до офіційних та спільнотних ресурсів є важливою частиною навчального процесу при вивченні програмування для Photoshop та інших програм Adobe. Оскільки студенти, особливо ті, хто не має досвіду в програмуванні, можуть стикатися з труднощами в освоєнні основних концепцій та інструментів, правильно організовані ресурси допомагають їм швидше засвоїти необхідні навички. Відповідні ресурси включають як офіційну документацію Adobe, так і спільнотні платформи, де студенти можуть знайти відповіді на запитання або навіть отримати допомогу від більш досвідчених розробників.

Один з найважливіших ресурсів для студентів, які навчаються програмуванню для Photoshop, це Adobe Photoshop Scripting Guide (PDF) [1]. Це офіційна детальна документація об'єктної моделі, яка містить повний опис всіх об'єктів, методів і властивостей, що доступні в середовищі Photoshop для скриптів. Цей документ дозволяє студентам зрозуміти, як працюють різні компоненти програми, як маніпулювати шарами, зображеннями, стилями тощо, за допомогою коду. Оскільки в програмуванні важливо мати чітке уявлення про те, з якими об'єктами працює код, доступ до такого документа може бути вирішальним для успішного розв'язання задач. Студенти можуть ознайомитися з прикладами використання основних методів і параметрів, що дасть їм змогу швидко освоїти основи скриптування для Photoshop.

Ще одним важливим інструментом для розробки скриптів є Adobe Developer Console [2] (рис. 7). Це офіційний ресурс для розробників, які працюють із UXP (Unified Extensibility Platform), новою платформою для розширень і плагінів в Adobe. Оскільки UXP є наступником ExtendScript, розробка за допомогою цього інструменту дозволяє створювати розширення для

Photoshop, які інтегруються безпосередньо в програму з сучасними можливостями та інтерфейсами. Студенти, які працюють з UXP, можуть використовувати Developer Console для реєстрації своїх розширень, доступу до API та інструментів для тестування та налаштування своїх рішень. Цей ресурс є особливо корисним для тих, хто хоче розробляти більш складні та інтерактивні плагіни, що використовують сучасні можливості Photoshop.

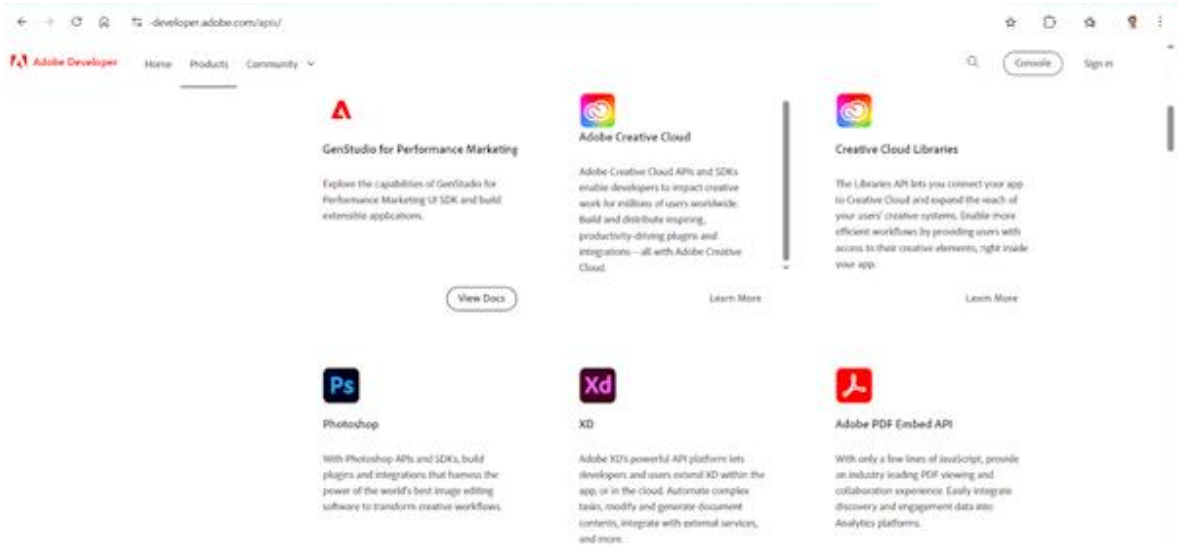

Рисунок 7 – Елементи ресурсу Adobe Developer Console

Спільноти та форуми є важливою частиною навчального процесу, оскільки на них студенти можуть звертатися за допомогою або обговорювати свої ідеї з іншими розробниками. Stack Overflow є однією з найбільших платформ для програмістів, де студенти можуть знайти відповіді на питання, що стосуються конкретних помилок у коді або кращих практик програмування для Photoshop. Оскільки на Stack Overflow вже існують багаточисельні відповіді на різноманітні запитання щодо скриптів для Adobe, студенти можуть швидко знайти рішення на свої проблеми.

GitHub – це не тільки платформа для зберігання та спільної роботи над кодом, але й місце, де студенти можуть знаходити безліч готових проектів та скриптів для Photoshop. Вони можуть переглядати чужі репозиторії, вивчати код, вносити власні зміни або навіть долучатися до відкритих проєктів (рис. 8). Це дає студентам можливість побачити, як працюють більш досвідчені програмісти, а також отримати зворотний зв'язок щодо своїх власних розробок.

Reddit також є корисною платформою для обговорення тем, пов'язаних зі скриптами для Photoshop, зокрема на спеціалізованих субреддітах, таких як r/Photoshop або r/Adobe. Тут студенти можуть задавати питання, обговорювати нові фічі або отримувати поради від інших користувачів, що працюють в Adobe-середовищі.

Не менш важливим ресурсом для студентів є відеоуроки на YouTube, де можна знайти багато покрокових інструкцій з програмування для Photoshop. Запит за ключовими словами, такими як "Photoshop scripting basics" або "Adobe ExtendScript tutorial", надасть доступ до широкого спектра відео, де показано, як

створювати прості та складні скрипти для автоматизації різних процесів в Photoshop. Відеоуроки часто мають візуальну складову, що дозволяє студентам краще засвоїти матеріал, бачачи, як виконується код і як результат відображається в самому Photoshop. Крім того, на YouTube є багато відео, де обговорюються типові проблеми, з якими можуть зіткнутися новачки, що робить цей ресурс чудовим доповненням до теоретичного матеріалу.

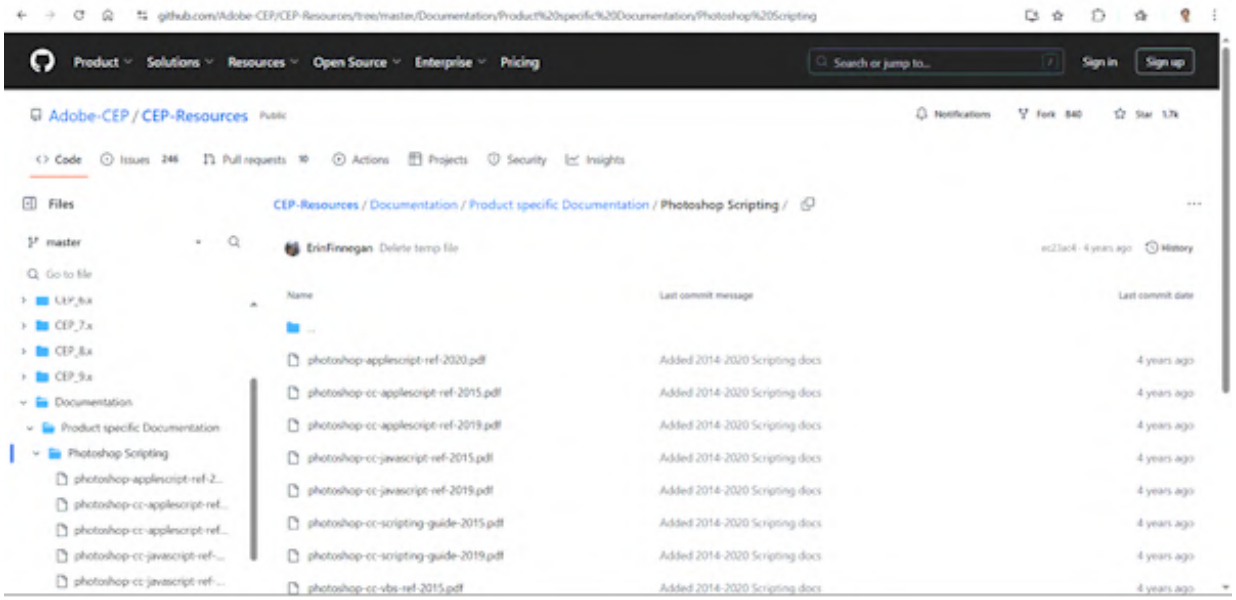

Рисунок 8 – Елементи ресурсу GitHub

Таким чином, надання студентам доступу до офіційних та спільнотних ресурсів є важливим кроком на шляху до успішного освоєння програмування для Photoshop. Adobe Photoshop Scripting Guide та Adobe Developer Console дають чітке розуміння інструментів і об'єктної моделі, а форумі, такі як Stack Overflow, GitHub та Reddit, забезпечують підтримку та можливість навчатися на досвіді інших розробників. Відеоуроки на YouTube пропонують наочні покрокові інструкції, що особливо корисно для тих, хто краще сприймає матеріал візуально. Усі ці ресурси допомагають студентам не тільки навчитися писати скрипти, але й долати труднощі, що виникають на шляху, що робить навчальний процес більш ефективним і доступним.

10 Типові помилки та поради

Порада для студентів: завжди починати зі збереження документа перед запуском скрипта, щоб уникнути втрати даних. Технічна грамотність – ключ до ефективного використання скриптів. Освоєння базових інструментів дозволяє студентам працювати впевнено, самостійно експериментувати та створювати більш складні проєкти.

Навчання студентів програмуванню для Photoshop має враховувати не тільки основні принципи написання скриптів, але й типові помилки, які можуть виникнути в процесі розробки. Знання цих помилок і вміння їх виправляти допомагає студентам більш ефективно працювати з кодом, знижуючи кількість

непотрібних перешкод на шляху до розв’язання завдань. Ось кілька основних труднощів, з якими можуть стикатися студенти.

Неправильна структура документа (activeDocument відсутній). Однією з найпоширеніших помилок є спроба доступу до активного документа activeDocument, коли такий документ відсутній. Це може статися, коли студент намагається виконати скрипт, не відкривши жодного файлу в Photoshop. У такому випадку код, що намагається взаємодіяти з документом, призведе до помилки. Приклад помилки:

```
var doc = app.activeDocument;
doc.resizeImage(1000, 1000);
```

Помилка: There is no document open.

Щоб уникнути цієї помилки, перед виконанням будь-яких операцій з активним документом необхідно перевірити, чи існує відкритий документ:

```
if (app.documents.length > 0) {
   var doc = app.activeDocument;
   doc.resizeImage(1000, 1000);
} else {
   alert("No document is open!");
}
```

Цей код додає перевірку, чи є відкриті документи, перш ніж намагатися взаємодіяти з ними.

Помилки при роботі з шарами (невірне посилання на об'єкт). Ще одна поширена помилка виникає при спробі звертатися до шарів. Студенти можуть помилково припустити, що шар завжди існує або що він доступний під певним індексом, що може призвести до помилки при виконанні коду. Особливо це стосується ситуацій, коли шар може бути відсутнім або неправильно названим.

З узагальненого досвіду перебігу виконання лабораторних робіт можна зазначити, що такі помилки виникають, коли студент використовує готовий текст скрипту, але не виправив назви шарів відповідно до своєї ситуації. Деякі з запропонованих студентам завдань містили вставлення елементів зображень в файл, що зазвичай виконується Photoshop з утворенням нового шару. Приклад помилки:

```
var layer = app.activeDocument.layers["LayerName"];
layer.visible = false;
```

Помилка: The layer "LayerName" was not found.

Для уникнення таких помилок слід перевіряти, чи існує шар із заданим ім'ям або індексом, перш ніж виконувати операції з ним:

```
var doc = app.activeDocument;
var layer = doc.layers.getByName("LayerName");
if (layer) {
```

```
    layer.visible = false;
} else {
    alert("Layer not found!");
}
```

Цей код перевіряє, чи існує шар із заданим ім'ям, перед тим як змінити його видимість.

Відсутність перевірок (if перевірки перед виконанням дії). Іншою типовою помилкою є відсутність перевірок перед виконанням певних дій, що може призвести до непередбачуваних результатів. Наприклад, якщо код спробує виконати операцію на непідготовленому об’єкті, це може спричинити помилку або неправильне виконання скрипту. Приклад помилки:

```
var doc = app.activeDocument;
doc.saveAs(new File("/path/to/file.jpg"));
```

Помилка: Якщо документ не було змінено, або якщо шлях до файлу некоректний, результат може бути непередбачуваним. Така помилка часто трапляється, коли студенти використовують вже готові елементи програмного тексту без адаптації під свої задачі. Щоб уникнути цієї помилки, важливо використовувати перевірки перед виконанням дій:

```
var doc = app.activeDocument;
if (doc.saved) {
    doc.saveAs(new File("/path/to/file.jpg"));
} else {
    alert("Document is not saved!");
}
```

Цей код перевіряє, чи був документ збережений, перед тим як виконати операцію збереження.

Шляхи до файлів у різних ОС (Windows / macOS). Оскільки Photoshop може бути встановлений як на Windows, так і на macOS, важливо правильно враховувати різницю в форматах шляхів до файлів. Водночас Windows використовує зворотні косі риски (\), тоді як macOS використовує прямі косі риски (/). Недотримання цих стандартів може призвести до помилок при збереженні або відкритті файлів. Приклад помилки:

```
var file = new File("C:\\Users\\Documents\\image.jpg");
```

Цей код працюватиме лише на Windows і не буде працювати на macOS.

Щоб уникнути помилки, слід використовувати універсальний формат:

```
var filePath = (File.fs == "Windows" ? "C:\\Users\\Documents\\image.jpg" : "/Users/Documents/image.jpg");
var file = new File(filePath);
```

Цей код адаптує шлях до файлу відповідно до операційної системи.

Права доступу до системних папок. Останньою поширеною проблемою є права доступу до системних папок. На macOS та Windows користувач може зіткнутися з обмеженнями доступу до певних директорій, наприклад, до папок "Program Files" або "System". Якщо скрипт намагається звертатися до таких директорій без належних прав, це призведе до помилки. Приклад помилки:

```
var file = new File("C:\\Program Files\\image.jpg");
file.open("r");
```

Помилка: Access denied або Permission error.

Щоб уникнути цієї проблеми, слід перевірити, чи має програма доступ до цієї папки, і обробляти можливі помилки за допомогою блоку try/catch:

```
try {
   var file = new File("C:\\Program Files\\image.jpg");
   file.open("r");
} catch (e) {
   alert("Access denied to the specified file or directory.");
}
```

Цей код допомагає обробити ситуацію, коли доступ до файлу або папки заблоковано.

Розуміння і вміння уникати типових помилок є важливою частиною процесу навчання програмуванню для Photoshop. Студенти повинні враховувати можливі труднощі, такі як відсутність активного документа, невірне посилання на об'єкти, відсутність перевірок перед виконанням дій, проблеми з різними шляхами до файлів у різних операційних системах та обмеження прав доступу до системних папок. Знання цих помилок і вміння їх виправляти допоможе студентам створювати надійніші та ефективніші скрипти.

11 Переваги та виклики впровадження

Інтеграція скриптів у викладання Photoshop є перспективною та корисною ініціативою, однак її реалізація вимагає врахування як сильних сторін, так і потенційних труднощів. Впровадження нових підходів до навчання завжди потребує адаптації з боку і викладача, і студентів.

Переваги впровадження скриптів можна звести до наступних – вказані на рис. 9.

Виклики та труднощі, які виникають під час впровадження завдань з використання скриптів, можна узагальнити таким чином (рис. 10).

Рекомендований підхід до подолання викликів. Досвід застосування завдань на вміння працювати з засобами автоматизації в Photosop свідчить, що вводити такі елементи (дії, скрипти) треба поетапно, починаючи з редагування готових рішень. Пояснювати кожен крок можна через аналогії – наприклад, порівнювати структуру скрипта з рецептом.

Рисунок 9 – Переваги впровадження скриптів в навчальний процес

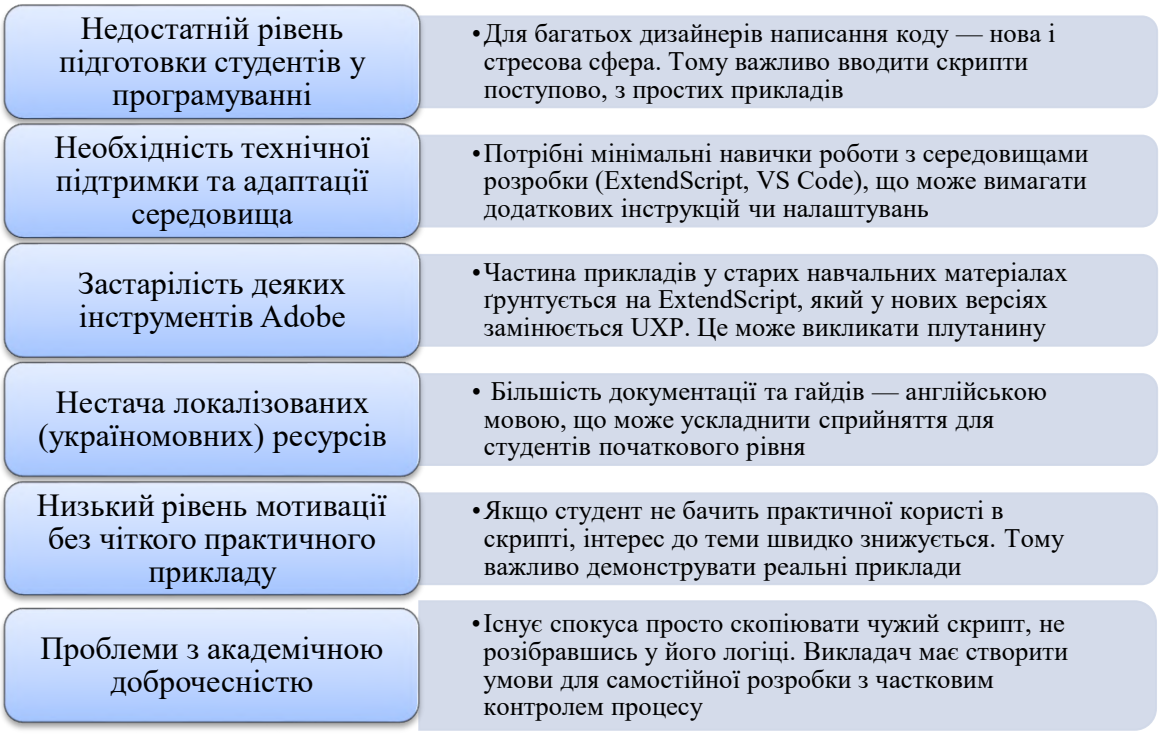

Рисунок 10 – Виклики та труднощі під час впровадження скриптів в навчальний процес

Бажано використовувати реальні практичні завдання, пов'язані з дизайном афіш, сертифікатів, презентацій тощо. Підтримувати студентів навчальними шаблонами, відеоінструкціями, чек-листами. Застосовувати командну роботу, де студенти вчаться один в одного.

Переваги скриптів значно переважають виклики – за умови грамотного педагогічного впровадження та адаптації під рівень студентів.

Висновки та рекомендації

В підсумку можна зробити висновок, що використання скриптів у Photoshop відкриває нові можливості як для викладання, так і для засвоєння матеріалу студентами. Це не лише інструмент автоматизації, а й ефективний засіб розвитку аналітичного, алгоритмічного та креативного мислення.

Скрипти дозволяють оптимізувати рутинні задачі, що особливо актуально в умовах обмеженого навчального часу. Студенти розвивають навички, які поєднують дизайн і програмування, що є важливою перевагою на сучасному ринку праці. Завдяки використанню скриптів у навчальних проєктах формується цілісний підхід до вирішення задач, від ідеї до реалізації.

Як переваги для викладачів слід згадати можливість створювати більш варіативні та інтерактивні завдання. Формування навчального середовища, що стимулює дослідження та самостійність. Легке масштабування завдань: від редагування готових скриптів до написання власних рішень.

Рекомендації щодо впровадження більш детально розкриті пунктом вище. Це порада починати з простого: запропонувати студентам перегляд і адаптацію готових скриптів. Надалі використовувати мікропроєкти, які можна реалізувати за кілька занять. Заохочувати групову роботу, де кожен учасник відповідає за окремий фрагмент коду. Поступово впроваджувати UXP-орієнтований підхід, щоб студенти орієнтувались на нові стандарти Adobe. Включати елементи самоаналізу та презентації результатів – це розвиває soft skills і відповідальність.

Скрипти – це міст між технічною точністю та творчою свободою. І саме освіта може навчити балансувати між цими двома сторонами.

Список літератури.

1. Adobe Developer Website. (n.d.). Adobe Developer Website. https://developer.adobe.com/.

2. Adobe-CEP/CEP-Resources. (n.d.). Photoshop Scripting. GitHub. https://www.adobe.com/content/dam/acom/en/devnet/photoshop/pdfs/photoshop-cc-javascript-ref-2020.pdf.

3. Дідик, Н.С., & Бізюк, А.В. (2014). Використання Java Script в програмі Adobe Photoshop. Інформаційні системи та технології. (с. 188-189). http://openarchive.nure.ua/handle/document/6460.

4. Комлєв, В. Вчимося писати скрипти для Photoshop. https://komliev.studio/p/practice/photoshop-scripting.html.

5. Стельмах, І.М., Зоренко, О.В., & Осипова, Т.Г. (2011). Апаратні та програмні засоби виготовлення електронних оригінал-макетів журнальної продукції. Технологія і техніка друкарства, 1(31), 14-24. https://doi.org/10.20535/2077-7264.1(31).2011.53221.

6. Зигуля, С.М., Зоренко, О.В., Талімонова, Н.Л., & Чепурна, К.О. (2023). Технології видавництв та поліграфії. Курсова робота. https://ela.kpi.ua/handle/123456789/55418.