UNIVERZITA PARDUBICE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2019 **David Baláček**

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Automatizování úloh v Adobe Photoshop

David Baláček

Bakalářská práce
2019

PROHLÁŠENÍ

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích, dne 12. 5. 2019

David Baláček

PODĚKOVÁNÍ

Na tomto místě bych velmi rád poděkoval Ing. Zbyňku Kopeckému, za konzultace, cenné rady a odborné vedení při tvoření závěrečné práce. Chtěl bych poděkovat i rodině a přítelkyni za jejich trpělivost a podporu po celou dobu mého studia.

ANOTACE

Tato bakalářská práce se věnuje automatizaci úloh v Adobe Photoshop pomocí skriptování v jazyce JavaScript, VBScript. V teoretické části je představen program Adobe Photoshop a vývojové prostředí. Dále je teoretická část věnována popisu hierarchie objektového modelu, aplikačního programového rozhraní, tříd a komponent. Praktickou část tvoří výukový materiál, jako průvodce automatizováním úloh ve zmíněném grafickém programu. Od úplných základů až po pokročilejší techniky, vytvořený ve webové prezentaci. Jeho obsahem je několik lekcí, které demonstrují teoretickou část bakalářské práce.

KLÍČOVÁ SLOVA

JavaScript, Adobe Photoshop, skript, aplikační programovací rozhraní (API).

TITLE

Procedure automation in Adobe Photoshop

ANNOTATION

This bachelor thesis deals with automating tasks in Adobe Photoshop using JavaScript, VBScript scripting. The theoretical part introduces the program Adobe Photoshop and the development environment. Futhermore, the theoretical part is devoted to the description of the object model hierarchy, application program interface, classes and components. The practical part consists of teaching material as a guide to automating tasks in the aforementioned graphic program. From the very foundation to more advanced techniques. Created in web presentation. It contains several lessons that demonstrate the theoretical part of the bachelor thesis.

KEYWORDS

JavaScript, Adobe Photoshop, script, application programming interface (API).

OBSAH:

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

DTP	Desktop publishing
CS	Creative Suite (služba společnosti Adobe)
CC	Creative Cloud (služba společnosti Adobe)
IDE	Integrated Development Environment (vývojové prostředí)
ID	Identifikace v oblasti IT
Plug-in	Zásuvný modul
UI	User interface (uživatelské rozhraní)
HTML	Hypertext Markup Language (značkovací jazyk)
CSS	Cascading Style Sheets (kaskádové styly)
CD	Compact Disc (kompaktní disk)
API	Application Programming Interface (Aplikační programovací rozhraní)
ESTK	ExtendScript Toolkit
JS	JavaScript
VBScript	Visual Basic Script
DOM	Document Object Model (Objektový model dokumentu)
URL	Uniform Resource Locator (jednotná adresa zdroje)
RGB	Barvový model (červená-zelená-modrá)
CMYK	Barvový model (azurová-purpurová-žlutá-černá)
HSB	Barvový model (odstín-sytost-hodnota jasu)

ÚVOD

Trendem dnešní doby je automatizace úloh, které jsou opakované z důvodu ušetření velkého množství času. V každém odvětví je dnes vidět známka částečné automatizace, aby byla zaměstnancům ušetřena práce a nemuseli dělat celý den ty samé úlohy. Tento trend, či pokrok se nevyhnul ani zpracování grafického obrazu. Potřeba úprav a tvorby grafického obrazu každým dnem narůstá z důvodu, že společnosti potřebují prezentovat své produkty. Úpravy v časopisech jsou již na denním pořádku. Většina lidí si neumí představit, kolik hodin za monitorem musí grafik strávit u grafické tvorby, či úprav a jaké to dá úsilí docílit perfektního vzhledu grafiky. Společnosti si na tuto práci najímají odborníky a z grafické tvorby a úprav se stalo běžné zaměstnání. Tato práce dokumentuje, automatizaci úloh v grafickém programu Adobe Photoshop, které by jinak grafik musel dělat manuálně. Cílena není pouze pro profesionální grafiky, ale i pro běžné uživatele tohoto programu.

1 ZADÁNÍ PRÁCE – CÍL PRÁCE

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**David Baláček**
Osobní číslo:	**I16062**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Automatizování úloh v Adobe Photoshop**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Teoretická část bakalářské práce bude věnována automatizaci pomocí skriptování, zejména automatizaci úloh pomocí skriptovacích jazyků VBScript a JavaScript, která umožňuje provádět téměř všechny operace jinak dostupné přes uživatelské rozhraní. Dále se práce bude věnovat popisu hierarchie objektového modelu, aplikačního programového rozhraní, tříd a komponent. Implementační část bude tvořit výukový materiál v rozsahu cca 10 lekcí vycházející z teoretické části ve formě webové prezentace vytvořené za pomoci jazyků HTML5, CSS, PHP a JS, který bude sloužit jako průvodce skriptováním pro Adobe Photoshop prostřednictvím výše uvedených jazyků, od úplných základů až po pokročilejší techniky. Jeho obsah bude tvořen popisem používaných tříd, příkazů a komponent, jejich syntaxí a možnostmi použití API. Automatizace úloh v příkladech (např. hromadné doplnění vodoznaků, úpravy a konverze fotografií) budou popsány, doplněny obrazovou přílohou před a po úpravě a skripty budou podrobně okomentovány.

Rozsah pracovní zprávy: **(min. 30 stran)**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

SUEHRING, Steve a Jakub ZEMÁNEK. JavaScript krok za krokem. Brno: Brno Computer Press, 2008. ISBN 978-80-251-2241-9
SUMMERFIELD, Mark. VBScript: průvodce vývojáře. Vyd. 1. Brno: UNIS Publishing, 2000. ISBN 80-860-9753-6. Adobe Photoshop CS6 scripting guide [online]. Adobe Systems Inc., 2012. [cit. 2015-10-20]. Dostupné z: https://www.adobe.com/content/dam/acom/en/devnet/photoshop/scripting/Photoshop-CS6-Scripting-Guide.pdf

Vedoucí bakalářské práce: **Ing. Zbyněk Kopecký**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2018**
Termín odevzdání bakalářské práce: **12. května 2019**

Univerzita Pardubice Česká republika
L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

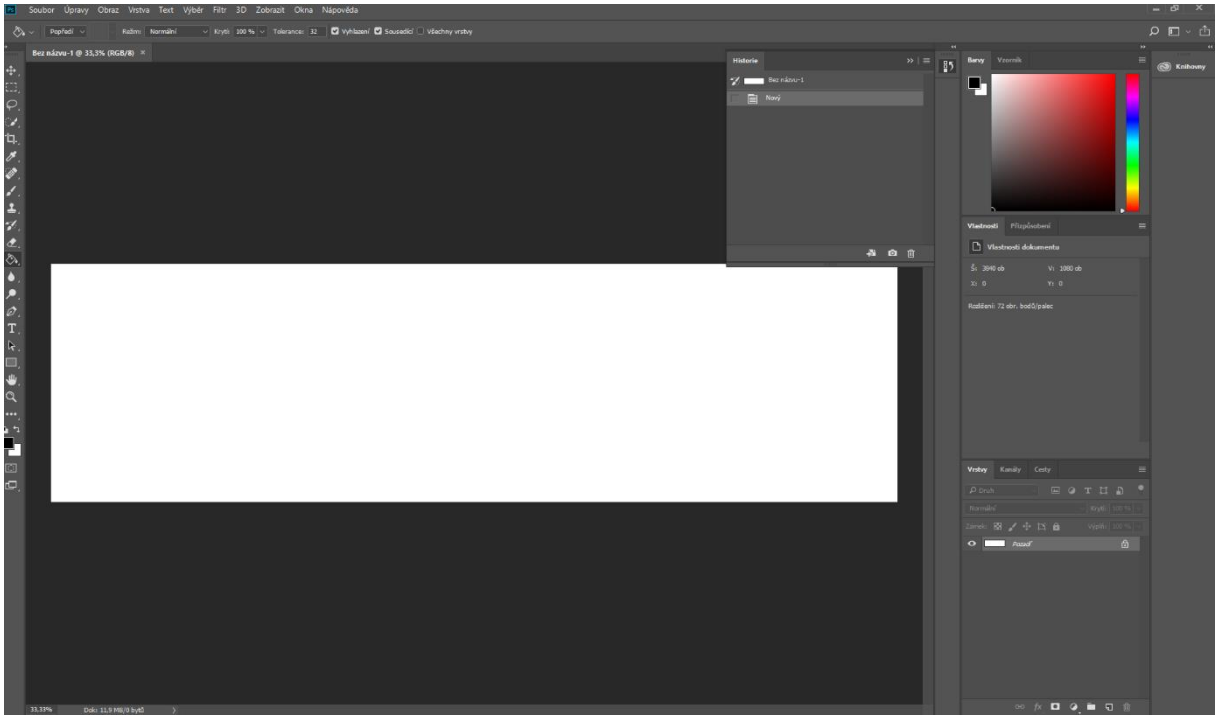
Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 14. prosince 2018

2 TEORETICKÁ ČÁST

2.1 Adobe Photoshop

Adobe Photoshop je software pro úpravy a vytváření rastrového obrazu, vytváření grafického designu a digitální umění. Tento grafický program poskytuje velikou sadu nástrojů pro téměř jakoukoliv manipulaci s nasnímaným obrazem. Jedná se o jeden z nejznámější grafický programů, který je dnes využíván v několika profesních odvětvích a stal se neoficiálně jedním ze standardů pro manipulaci rastrové počítačové grafiky. Spadá do oblasti DTP (Desktop publishing), tedy oblasti zabývající se tvorbou tištěného dokumentu za pomoci počítače. Samotný program má více jak desítku verzí, z nichž nejznámější a dnes využívané jsou: Adobe Photoshop CS6 (Creative Suite), který vyšel v březnu roku 2012 a Adobe Photoshop CC (Creative Cloud).



Obrázek 1 - Adobe Photoshop (zdroj: David Baláček)

2.1.1 Charakteristika funkcí Adobe Photoshop

V programu Adobe Photoshop se můžeme setkat s celou řadou nástrojů pro výběr, ať už geometrický, od ruky, na základě detekce hran a podle barvy. Prostředí dovoluje mít přehlednou hierarchii vrstev, skupin vrstev, aplikovaných výběrů, což přispívá k lehčí práci a přehlednosti. Zároveň hierarchie vrstev umožňuje práci s vrstvami, která se podobá kreslení části obrazu na fólie. Jednotlivé fólie je možno upravovat a přemisťovat, aniž by to ovlivnilo obsah zbylých fólií. Poskládáním fólií na sebe je vidět celou kompozici. Vrstvy nabízí

možnost ikony oka, pomocí kterého lze vrstvu zviditelnit, nebo naopak zneviditelnit. Umožňují vytvoření masky pro vrstvu, která v případě potřeby práce pouze s jednou částí obrazu chrání zbylé části grafiky, podobně jako například lepící páska. Adobe Photoshop disponuje několika barevnými režimy včetně těch základních (RGB, CMYK), je také možnost vybrat paletu barev pouze s bezpečnými webovými barvami. Výslednou práci lze exportovat do mnoha formátů, včetně vlastních formátů programu Adobe Photoshop (*.psd, *.pdd, *.psdt). Grafika je poté připravena pro další využití jinými programy a zároveň je multiplatformní. Zde zmíněné funkce jsou pouze základním přehledem toho programu, ale zároveň jedním z nejvíce využívaných.

2.2 Skriptování

Skript je série po sobě jdoucích příkazů, které říkají, jaké akce má program Adobe Photoshop provést, například jaký použít filtry pro zobrazení výběru v dokumentu či definuje jaký barevný model zvolit. Skript běží sám o sobě pouze na uživatelově počítači a nevyžaduje žádný druh kompilace kódu před spuštěním, přilinkování knihoven ani jakékoliv sestavení řešení před samotným spuštěním. Slouží k automatizaci úloh, které jsou neměnné a opakující se. Využití skriptu je nejlépe zvážit v případě, kdy řešení opakujících se úloh manuálně by bylo vysoce časově náročné. Například v případě, kdy máme vysoké množství obrázků a každý obrázek potřebujeme upravit stejným způsobem, v tomto případě nám napsání skriptu ušetří čas a také velké množství manuální práce. Vytvořený skript může ovlivnit jediný objekt nebo několik objektů v jediném dokumentu, podle uživatelova rozhodnutí, což dodává samotnému skriptu větší možnosti a způsob, jak zkrátit a zpřehlednit zápis, například pokud chceme, aby každá vrstva měla danou průhlednost, stačí definovat průhlednost setu (tedy množině), ve které se vrstvy nachází a není zapotřebí procházet každou vrstvu a nastavovat ji ten samý atribut se stejnou hodnotou. Každá skriptovací třída koresponduje s nástrojem, který najdeme v prostředí programu Adobe Photoshop, což přispívá k tvrzení, že skriptování v programu Adobe Photoshop umožňuje skoro úplnou modifikaci grafických objektů, jako při manuální práci v programu. [2]

2.2.1 Podpora Adobe Photoshop

Photoshop podporuje tři skriptovací jazyky: AppleScript, VBScript a JavaScript. AppleScript a JavaScript lze využít pro skriptování na Mac OS platformě, pro operační systém Windows je třeba využít JavaScript nebo VBScript. Adobe si pro své produkty vytvořilo rozšířenou verzi jazyka JavaScript, nazývaný ExtendScript, který dovoluje uživateli větší možnosti

využití nástrojů prostředí. Z důvodu kompatibility se doporučuje pro skripty v jazyce JavaScript, určené pro prostředí Adobe Photoshop, využívat příponu *.jsx namísto známé *.js přípony. [1]

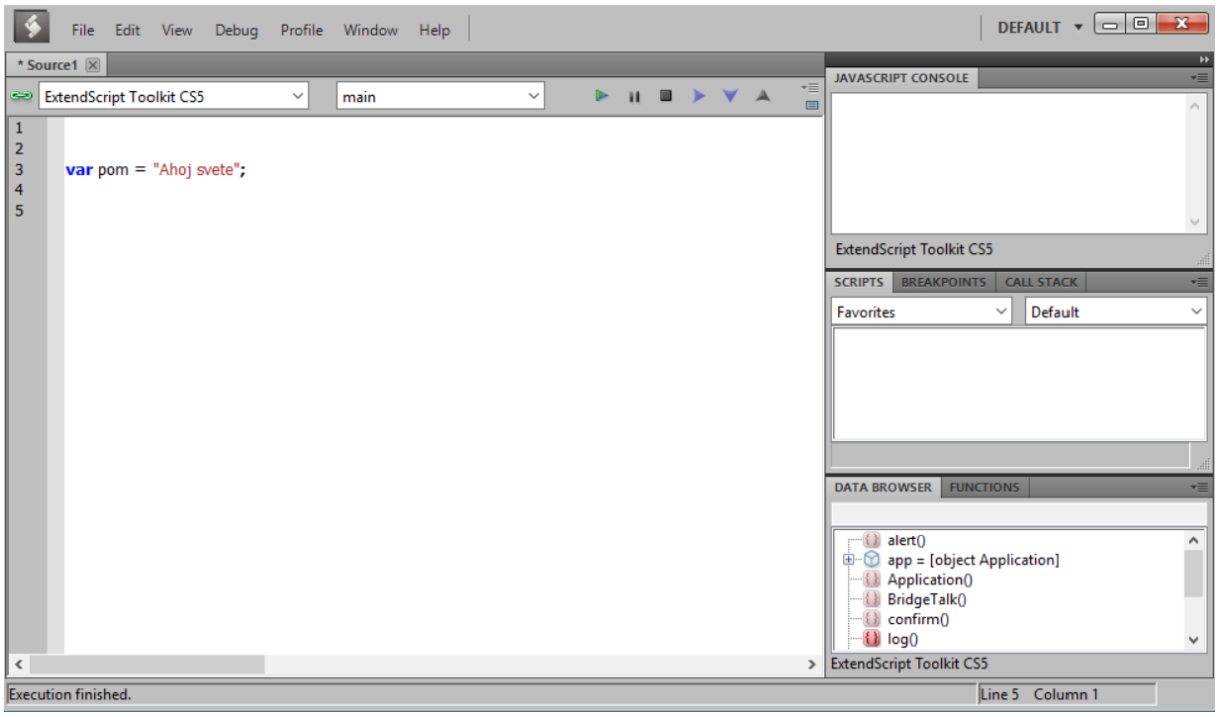
Tabulka 1 – Přehled podporovaných jazyků (zdroj: Adobe Photoshop CC Scripting Guide)

Druh skriptu	Druh souboru	Přípona	Platforma
AppleScript	Kompilovaný skript, OSAS soubor	.scpt, (žádná)	Mac OS
JavaScript, ExtendScript	Textový	.js, .jsx	Mac OS a Windows
VBScript	Textový	.vbs	Windows
Visual Basic	Spustitelný	.exe	Windows

2.2.2 Prostředí

Jako vývojové prostředí pro JavaScript je možné použít jakýkoliv textový editor (např. poznámkový blok, Notepad++), případně jakékoliv IDE (Integrated Development Environment – vývojové prostředí), které podporuje vytváření skriptů v jazyce JavaScrip, například Netbeans, WebStorm. Pro ExtendScript Adobe poskytuje vlastní IDE, a to s názvem ExtendScript Toolkit (ESTK) verze 3.5.1 zahrnuto v Adobe Photoshop CS5, ovšem je možné jej stáhnout zvlášť jako samostatný program. Toto prostředí je přímo připraveno pro psaní skriptů pro produkty od společnosti Adobe a umožňuje i postupné krokování napsaného kódu s možností sledováním změn za běhu, dále je možné v IDE vidět druhy hlavních obalových tříd a napovídání při psaní skriptu pomocí klávesových zkratek. ESTK je možné stáhnout z adresy: https://www.adobe.com/devnet/scripting/estk.html . [3], [5]

Na internetu je možné najít několik vhodných IDE pro VBScript, jedním z nejznámějších a používaných jsou například PrimalScript 2019, VbsEdit. Také jako u JavaScriptu je zde možnost využít jakýkoliv textový editor. [4], [8]



Obrázek 2 - ExtendScript Tool Kit (zdroj: David Baláček)

2.2.3 Instalace skriptu a spuštění neinstalovaného skriptu

Skripty napsané v JavaScriptu jsou jednoduše přístupné z prostředí Adobe Photoshop přes (Soubor > Skripty). Pro instalaci skriptu do prostředí pro přístupnost přímo z menu je nutné skript vložit do adresáře, kde je Photoshop nainstalován. Nejčastěji se jedná o strukturu v tomto uspořádání (C: Program Files/Adobe /Adobe Photoshop CC /Presets /Scripts). Po vložení skriptu do daného adresáře a po restartování programu, budou skripty přístupné pod názvem a bez přípony zobrazeny ve skriptovém menu programu Adobe Photoshop. [1]

Pro provedení skriptu, který není umístěn v adresáři, tedy není dostupný přímo z prostředí je nutno přejít do menu (Soubor > Skripty > Procházet…) a načíst skript manuálně. Ve chvíli vybrání skriptu a odsouhlasení dialogového okna, je skript okamžitě proveden. [2]

Pro spuštění skriptu ve VBScript stačí pouze dvakrát kliknout, nebo stisknout klávesu enter na uložený soubor, před prvním použitím skriptu je nutné mít program Adobe Photoshop zavřený. Po kliknutí na skript se program sám spustí a provede akce napsané ve skriptu. To vše za předpokladu, že koncovka .vbs je registrována na spuštění Windows Script Host. [2]

2.2.4 Spuštění prvního skriptu

Pro spuštění prvního skriptu postupujte podle níže uvedených bodů. Skript má účel po spuštění vytvořit nový předdefinovaný dokument v Adobe Photoshop s výchozí velikostí.

JS[1]

1. Otevřít prostředí ESTK a vlevo v horním části prostředí je přednastavený ExtendScript ToolKit, tuto volbu je třeba změnit na verzi nainstalovaného Photoshopu, v mém případě se jedná o vybrání položky Adobe Photoshop CC 2018 v kombinovaném poli (Combo box).
2. Do konzolové části okna napíši následující kód:

```
app.documents.add()
```

3. Klávesou F5 nebo zvolením Debug > Run, skript spustím.

VBScript[1]

1. Otevřu libovolný textový editor a napíši následující kód:

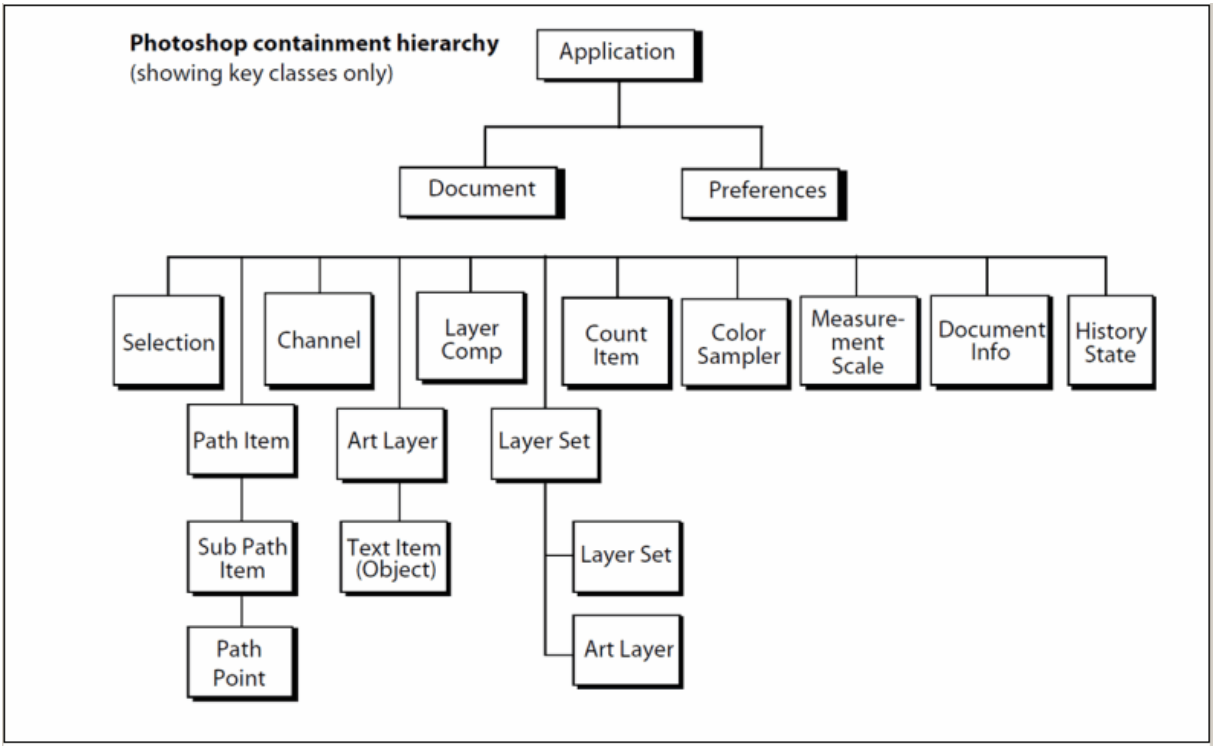
```
Set ref = CreateObject("Photoshop.Application")
ref.Documents.Add()
```

2. Uložím soubor s libovolným názvem s příponou *.vbs.
3. Skript se spustím dvojitým kliknutím na soubor.

2.3 Objektový model

Dokumentový objektový model (DOM) je aplikační rozhraní (API), které umožňuje přístup k rozmanitým komponentám dokumentu a funkcím programu Adobe Photoshop za pomoci skriptovacího jazyka. Cílem modelu je organizovat objekty k jednoduché identifikaci. Principem stojícím za DOM je kontejnerová hierarchie. Jedná se o hierarchii podobné stromové struktuře, jinými slovy horní nejvyšší objekt obsahuje nižší objekty, které také obsahují podmnožinu dalších nižších objektů. V programu Adobe Photoshop je na vrcholu této hierarchie aplikační objekt, který má několik hlavních vlastností a obsahuje v sobě samotný objekt *Document*. Použitím příkazů nebo metod v DOM, je možnost manipulace s objekty, například který přidat, odebrat nebo kterému objektu nastavit individuální vlastnosti, například jako je barva, velikost, tvar. Každý uzel v DOM reprezentuje třídu a některé klíčové třídy jsou seskupeny podle elementů nebo kolekcí. [1], [2]

[1] Zdrojové kódy převzaty z https://www.adobe.com/content/dam/acom/en/devnet/photoshop/scripting/photoshop-cc-scripting-guide.pdf.



Obrázek 3 - Hierarchie objektového modelu (zdroj: Adobe Photoshop CC Scripting Guide)

2.3.1 Elementy a kolekce

Objektový model programu Adobe Photoshopp používá elementy nebo kolekce jako příhodný způsob seskupení tříd. V názorném obrázku nejsou znázorňovány objektové elementy nebo kolekce, jelikož ne všechny třídy jsou spojovány kolekcí. Většina tříd ovšem toto seskupení má. Následuje výčet elementů/kolekcí, které existují v programu Adobe Photoshop: *Art Layers, Channels, Color Samplers, Count Items, Documents, Layers Comps, Layer Sets, History States, Notifiers, Path Items, Path Points, Sub Path Items, Text Fonts.* Dále je nutno upozornit na změnu, kde v jazyce VBScript jsou kolekce indexovány od 1 namísto běžné 0.

2.3.2 Layer třídy

Adobe Photoshop má dva typy vrstev – *Art Layer* (grafická vrstva), která může zahrnovat obsah obrázku a *Layer Set*, která může obsahovat žádnou nebo více vrstev grafických vrstev. Třída *Art Layer* umožňuje práci v dokumentu s jedním elementem obrázku bez narušení ostatních prvků. *Layer Set* je na rozdíl od třídy *Art Layer* skupina několika vrstev. Kompozici obrázku je možné měnit pomocí změny pořadí vrstev a vlastností jednotlivých vrstev. [1]

K následné demonstraci práce s vrstvami vytvořím nový dokument. Do dokumentu přidám novou vrstvu, pojmenuji ji a uložím do proměnné pro další práci. Poté vytvořím *Layer Set* a vytvořenou vrstvu umístím za vytvořenou skupinu vrstev.

JS[1]

```
//vytvoření nového dokumentu
app.documents.add()
//přidání vrstvy, její pojmenování a uložení do proměnné
var layerRef = app.activeDocument.artLayers.add()
layerRef.name = "Moje vrstva"
//vytvoření nové sady vrstev
var newLayerSetRef = app.activeDocument.layerSets.add()
//přesunutí Art Layer třídy do nově vytvořené sady
newLayerSetRef.move(layerRef, ElementPlacement.PLACEAFTER)
```

VBScript[1]

```
' vytvoření nového dokumentu
Dim appRef
Set appRef = CreateObject("Photoshop.Application")
Dim docRef
Set docRef = appRef.Documents.Add()
' přidání vrstvy, její pojmenování a uložení do proměnné
Dim layerObj
Set layerObj = appRef.ActiveDocument.ArtLayers.Add
layerObj.Name = "Moje vrstva"
' vytvoření nové sady vrstev
Dim newLayerSetRef
Set newLayerSetRef = appRef.ActiveDocument.LayerSets.Add
' přesunutí Art Layer třídy do nově vytvořené sady, číselná
' konstanta čtyři pro přesunutí za skupinu vrstev
newLayerSetRef.Move layerObj, 4
```

2.3.3 Selection třída

Selection třída nám umožňuje podobně jako při manuální práci v Adobe Photoshop vytvářet výběry oblasti pixelů aktivního dokumentu a následnou práci s ní. Nabízí možnost aplikovat efekty na výběr nebo jej zkopírovat do schránky a přesunout na jinou pozici.

K vybrané oblasti je možné přistupovat přes vlastnosti třídy *Selection* nebo pomocí metod této třídy. Z nejzákladnějších metod této třídy lze zmínit – *fill* (vyplnění oblasti), *invert* (inverze výběru), *resize* (manipulace s velikostí výběru), *stroke* (nastavení vlastností okraje výběru). Dále je možné s výběry manipulovat pomocí skriptovacích konstant. Třída je hierarchicky uspořádána v kolekci a je možné s ní pracovat na každé úrovni, teda ji využít u masek, cest, výběrů barev. [1], [6]

[1] Zdrojové kódy převzaty z https://www.adobe.com/content/dam/acom/en/devnet/photoshop/scripting/photoshop-cc-scripting-guide.pdf.

Vytvoření výběru lze provést pomocí kartézské soustavy souřadnic, jelikož obrázek je dvoudimenzionální objekt. Počáteční bod obdélníkového výběru je levý horní roh na ose x = 0, na ose y = 0. Protilehlý bod je pravý dolní, který může mít maximální hodnotu šířky a výšky plátna. [3]

V následné demonstraci této třídy dojde k vytvoření nového dokumentu o velikosti 500×500 pixelů, vytvoření pole s body výběru a následného aplikování pole k vybrání definované oblasti. Výběr poté pomocí již zmiňované metody *fill* vybarvím barvou popředí, výchozí nastavení programu Adobe Photoshop je černá.

JS[1]

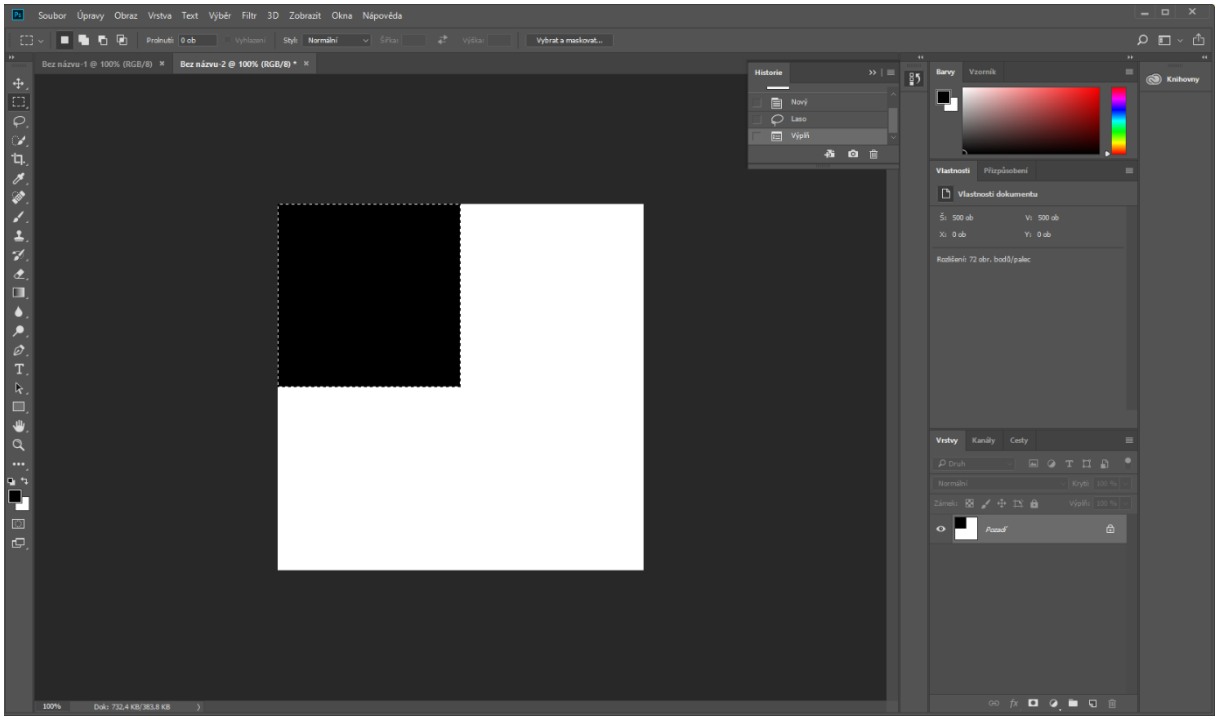
```
// nastavení jednotek na pixelové body, pokud bude
// zakomentováno budou použity výchozí jednotky
app.preferences.rulerUnits = Units.PIXELS
// vytvoření dokumentu 500x500
var docRef = app.documents.add(500, 500)
// definování bodů výběru do pole
var shapeRef = [
        [0,0],
        [0,250],
        [250,250],
        [250,0]
]
// vybrání oblasti
docRef.selection.select(shapeRef)
// vyplnění výběru barvou popředí
app.activeDocument.selection.fill(app.foregroundColor)
```

VBScript[1]

```
' vytvoření dokumentu 500x500
Set ref = CreateObject("Photoshop.Application")
' nastavení jednotek na pixelové body, pokud bude
' zakomentováno budou použity výchozí jednotky
' konstanta pro pixelove body
ref.References.RulerUnits = 1
Set docRef = ref.Documents.Add(500,500)
' definování bodů výběru do pole
ShapeRef =
Array(Array(0,0),Array(0,250),Array(250,250),Array(250,0))
' vybrání oblasti
```

[1] Část zdrojových kódů převzata z https://www.adobe.com/content/dam/acom/en/devnet/photoshop/scripting/photoshop-cc-scripting-guide.pdf, https://www.adobe.com/content/dam/acom/en/devnet/photoshop/pdfs/photoshop-cc-vbs-ref.pdf, nutno podotknout, že zdrojové kódy z druhé příručky (Adobe Photoshop CC VBscript Scripting Reference) jsou nepřesné a musely být mírně upraveny autorem této práce.

```
docRef.Selection.Select ShapeRef
' vyplnění výběru barvou popředí
docRef.Selection.Fill(ref.ForeGroundColor)
```



Obrázek 4 - Výsledek ukázkového kódu (zdroj: David Baláček)

2.3.4 Třídy Path Item, Sub Path Item, Path Point

Hierarchický objektový model je rozdělen na tři třídy cest, z kterých je možno vybrat. Tyto třídy jsou si velice podobné, ale každá je určena k trochu specifičtějšímu účelu a každá z tříd má své vlastnosti a metody, které lze použít pro skriptování. Třídy se nazývají *Path Item*, *Sub Path Item*, *Path Point.*

Třída *Path Item* reprezentuje informaci o nakresleném objektu, jeho tvar, obrys nebo křivku. *Path Item* obsahuje třídu *Sub Path Item*, která přímo poskytuje geometrické tvary. Poslední třída, *Path Point*, uvádí informace o jednotlivých bodových elementech v třídě Sub *Path Item*, dalo by se tedy říci, že tyto třídy jsou provázané jedna na druhou a podtřída obsahuje větší specifikace o elementu.

Následující ukázka použije *Path Item* objekt pro vytvoření rovné přímky vykreslené štětcem. [1]

JS[1]

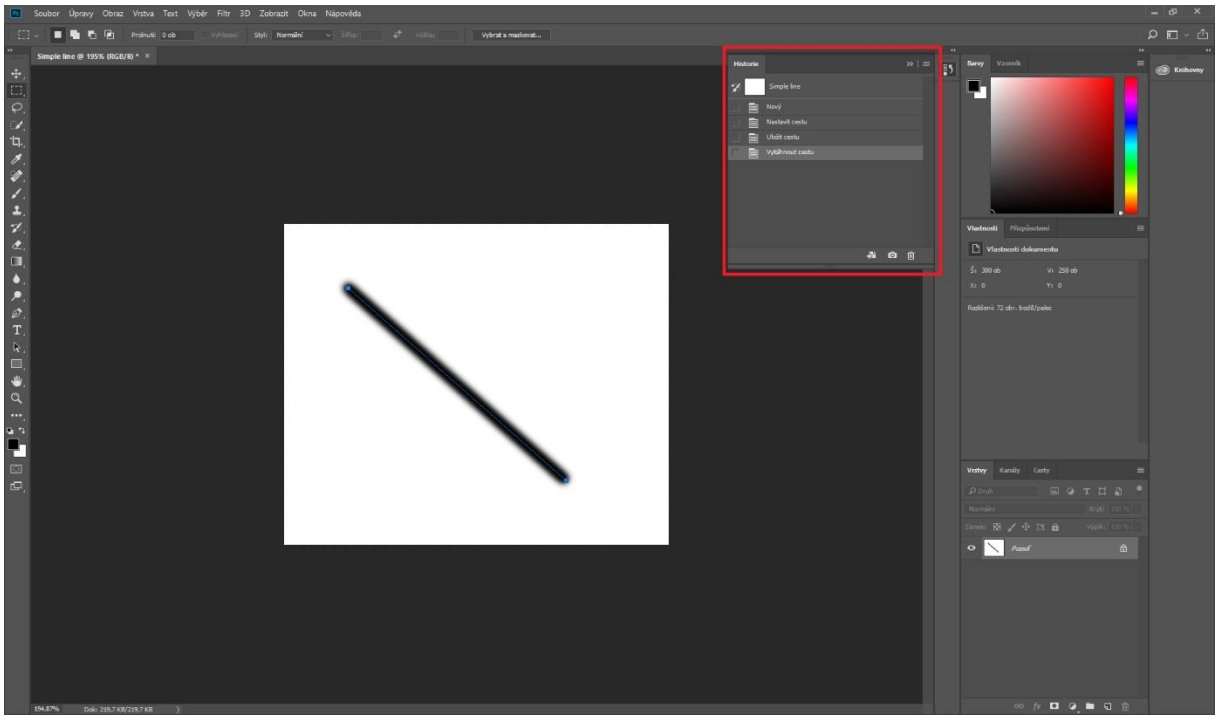
```
//vytvoří nový dokument 300x250 s názvem Simple line
var docRef = app.documents.add(300,250, 72,"Simple line")
//vytvoření pole dvou objektů PathPointInfo, které obsahují
dva body
//a nastavení směru
var lineArray = new Array()
    lineArray[0] = new PathPointInfo
    lineArray[0].anchor = Array(50,50)
    lineArray[0].leftDirection = lineArray[0].anchor
    lineArray[0].rightDirection = lineArray[0].anchor
    lineArray[1] = new PathPointInfo
    lineArray[1].anchor = Array(220,200)
    lineArray[1].leftDirection = lineArray[1].anchor
    lineArray[1].rightDirection = lineArray[1].anchor
//vytvoření objektu SubPathInfo, kterému přiřadím pole bodů
přímky
var lineSubPathArray = new Array()
    lineSubPathArray[0] = new SubPathInfo()
    lineSubPathArray[0].operation = ShapeOperation.SHAPEXOR
    lineSubPathArray[0].closed = false
    lineSubPathArray[0].entireSubPath = lineArray
//vytvoření PathItem objektu a přidání do kolekce PathItems
pomocí metody add()
var myPathItem = docRef.pathItems.add("A line",
lineSubPathArray);
//zobrazení cesty, pro zviditelnění výstupu
myPathItem.strokePath(ToolType.BRUSH)
```

VBScript[2]

```
Dim appref, docRef
Dim lineArray(1), lineArray2(1), lineSubPathArray(0),
myPathItem
Set appRef = CreateObject("Photoshop.Application")
'vytvoří nový dokument 300x250 s názvem Simple line
Set docRef = appRef.Documents.Add(300,250,72, "Simple
line")
'vytvoření pole dvou objektů PathPointInfo, které obsahují
'dva body
'a nastavení směru
Set lineArray(0) = CreateObject("Photoshop.PathPointInfo")
    lineArray(0).Anchor = Array(50,50)
    lineArray(0).LeftDirection = lineArray(0).Anchor
```

[1] Zdrojové kódy převzaty z https://www.adobe.com/content/dam/acom/en/devnet/photoshop/scripting/photoshop-cc-scripting-guide.pdf, kód částečně modifikován autorem práce.
[2] Zdrojové kódy převzaty z https://www.adobe.com/content/dam/acom/en/devnet/photoshop/scripting/photoshop-cc-scripting-guide.pdf, kód částečně modifikován autorem práce.

```
    lineArray(0).RightDirection = lineArray(0).Anchor
Set lineArray(1) = CreateObject("Photoshop.PathPointInfo")
    lineArray(1).Anchor = Array(220,200)
    lineArray(1).LeftDirection = lineArray(1).Anchor
    lineArray(1).RightDirection = lineArray(1).Anchor
'vytvoření objektu SubPathInfo, kterému přiřadím pole bodů
přímky
Set lineSubPathArray(0) =
CreateObject("Photoshop.SubPathInfo")
    lineSubPathArray(0).Operation = 2 'konstanty ve VBS se
'zadávají číselně
    lineSubPathArray(0).Closed = false
    lineSubPathArray(0).EntireSubPath = lineArray
'vytvoření PathItem objektu a přidání do kolekce PathItems
'pomocí metody add()
Set myPathItem = docRef.PathItems.Add("A line",
lineSubPathArray)
'zobrazení cesty, pro zviditelnění výstupu
myPathItem.StrokePath(2) 'opět konstanta 2 = štětec
```



Obrázek 5 - Výstup příkladu použití PathItem, červeně zvýrazněna historie provedených příkazů (zdroj: David Baláček)

2.3.5 Preferences třída

Preferences třída umožňuje nastavit prostředí programu Adobe Photoshop, podle přiřazených hodnot ve skriptu. Umožňuje všechny funkcionality jako v případě manuálního nastavení uživatelem přímo v samotném programu Adobe Photoshop přes menu: (Upravit > Předvolby)

v jednotlivých kategoriích. Nastavení je možné uložit, obnovit na výchozí hodnoty nebo případně načíst uložené nastavení a jednoduchým spuštěním skriptu tak jednoduše připravit specifické prostředí pro uživatele, nejčastěji se používá přednastavení grafických jednotek na obrazové body, nebo palce. [1]

Následující skript uloží uživatelské předvolby do proměnných, aby později bylo možné toto nastavení vrátit do původního stavu. [3]

JS[1]

```
var startRulerUnits = app.preferences.rulerUnits
var startTypeUnits = app.preferences.typeUnits
var statDisplayDialogs = app.displayDialogs
```

VBScript[2]

```
Dim startRulerUnits, startTypeUnits, statDisplayDialogs,
appRef
Set appRef = CreateObject("Photoshop.Application")
startRulerUnits = appRef.Preferences.RulerUnits
startTypeUnits = appRef.Preferences.TypeUnits
statDisplayDialogs = appRef.DisplayDialogs
```

2.3.6 Notifier třída

Tato třída se využívá k interakci s uživatelem a lze přes ni informovat o nastalých událostech. Slouží jako pomocná třída, která se dá navázat ke skriptům a informovat o jejich dokončení. Tato třída obsahuje pouze dvě metody: *add* (přidání události do kolekce), *removeAll* (odstraní všechny události z této kolekce), abychom mohli navázat *Notifier* objekt s událostí, musíme specifikovat přesné ID (identifikátor události), na kterou chceme upozornit. ID těchto událostí lze najít v oficiální příručce o skriptování v programu Adobe Photoshop v příloze A. [3] Druhou možností, jak zjistit ID události nebo třídy, je použít posluchač skriptů o kterém je zmínka v kapitole Manažer Akcí. [1], [3]

Následující ukázka ilustruje, jak nastavit třídu oznámení na otevření nového dokumentu a spuštění skriptu při této akci.

[1] Zdrojové kódy převzaty z http://wwwimages.adobe.com/www.adobe.com/content/dam/acom/en/devnet/photoshop/pdfs/photoshop-cc-javascript-ref.pdf.
[2] Zdrojové kódy převzaty z https://www.adobe.com/content/dam/acom/en/devnet/photoshop/pdfs/photoshop-cc-vbs-ref.pdf.

JS[1]

```
app.notifiersEnabled = true
var eventFile = new File(app.path + "Presets\Scripts\Event
Scripts Only\Welcome.jsx")
//oznámení se projeví, až po restartu programu Photoshop,
//narozdíl od funkce removeAll(), která se projeví okamžitě
app.notifiers.add("Opn ", eventFile)
```

VBScript[2]

```
Dim appRef, eventFile
Set appRef = CreateObject("Photoshop.Application")

appRef.NotifiersEnabled = True
eventFile = appRef.Path & "Presets\Scripts\Event Scripts
Only\Welcome.jsx"
'oznámení se projeví, až po restartu programu Photoshop,
'narozdíl od funkce removeAll(), která se projeví okamžitě
appRef.Notifiers.Add "Opn ", eventFile
```

2.3.7 SolidColor třída

Třída *SolidColor* (třída plných barev) umožňuje veškerou práci s barvami v programu Adobe Photoshop. Tyto třídy představují hexadecimální vyjádření barev v barvových modelech. Třída podporuje několik barvových modelů (RGB, HSB, CMYK) a převod mezi modely. Umožňuje například porovnávání barev pomocí metody *isEqual*, kterou obsahuje rodičovská třída *SolidColor* a všechny její potomky. Třída dále dává možnost převést barvu na nejbližší bezpečnou webovou barvu, která se zobrazí bez rozkladu barev na systémech zobrazujících v 256 barvách.

V následující ukázce demonstruji převod mezi barvovým modelem, porovnání barev a získání nejbližší bezpečné webové barvy. Pro skript je třeba mít otevřený libovolnou grafiku v programu Adobe Photoshop.

JS[3]

```
//převod barvy do modelu
var someColor = foregroundColor.cmyk
//porovnání zdali se barva v popředí rovná barvě pozadí
if(app.foregroundColor.isEqual(backgroundColor))
```

[1] Zdrojové kódy převzaty z https://www.adobe.com/content/dam/acom/en/devnet/photoshop/scripting/photoshop-cc-scripting-guide.pdf.
[2] Zdrojové kódy převzaty z https://www.adobe.com/content/dam/acom/en/devnet/photoshop/pdfs/photoshop-cc-vbs-ref.pdf.
[3] Zdrojové kódy převzaty z https://www.adobe.com/content/dam/acom/en/devnet/photoshop/scripting/photoshop-cc-scripting-guide.pdf.

```
    //proveď příkazy ...
//převod barvy v popření na nejbližší bezpečnou webovou
//barvu
var webSafeColor = new RGBColor()
webSafeColor = app.foregroundColor.nearestWebColor
//vypsání barvových hodnot uživateli
alert("Hodnota červené barvy: " + webSafeColor.red + "\n" +
"Hodnota zelené barvy: " + webSafeColor.green + "\n" +
"Hodnota modré barvy: " + webSafeColor.blue + "\n")
```

VBScript[2]

```
'vytvoření proměnných
Dim someColor, appRef, myWebSafeColor
'uložení reference na objekt do proměnné
Set appRef = CreateObject("Photoshop.Application")
someColor = appRef.ForegroundColor.model
'zjištění zdali se model rovná RGB, podle konstanty
'konstanta odpovídá RGB modelu
If(someColor = 2) Then
    'převod barvy do modelu
    someColor.cmyk
End If
'porovnání, zdali se barva v popředí rovná barvě pozadí
If(appRef.ForegroundColor.IsEqual(appRef.BackgroundColor))
Then
    'proveď příkazy ...
End If
'převod barvy v popření na nejbližší bezpečnou webovou
barvu
Set myWebSafeColor = appRef.ForegroundColor.NearestWebColor
'vypsání barvových hodnot uživateli, vbCrLf je znak
'odřádkování, dále je třeba dát si pozor a umístit příkaz
'na jeden řádek
MsgBox("Hodnota červené barvy: " & webSafeColor.red +
vbCrLf + "Hodnota zelené barvy: " & webSafeColor.green +
vbCrLf & "Hodnota modré barvy: " & webSafeColor.blue)
```

2.3.8 Konstanty

Speciálními komponentami v Adobe Photoshop jsou konstanty, které představují předdefinované hodnoty vlastností jednotlivých tříd. Konstanty či jinak výčtové typy vznikly pro metody tříd, které díky konstantním typům přijímají pouze akceptovatelné hodnoty a nedochází tak k chybám. Kompletní přehled konstant a výčtových typů lze najít v příručce Adobe Photoshop CC/CS JavaScript Scripting Reference nebo Adobe Photoshop CC/CS VBScript Scripting Reference, kde pod kapitolou Scripting Constants lze najít i kompletní

hodnoty a vysvětlení dané konstanty nebo výčtového typu. Nutno podotknout, že zápis konstant se u každého jazyka liší, proto je třeba dbát na správný a korektní zápis. [2], [3]

2.4 Manažer akcí

Adobe Photoshop umožňuje automatizaci opakujících se akcí, pomocí aplikačního rozhraní využívající panel Akce. Manažer akcí umožňuje psát skripty, cílené na funkcionalitu, která není přístupná přes skriptovací rozhraní, jako jsou například filtry třetích stran. Hlavním a také jediným požadavkem pro využití Manažera akcí je, že úlohu, kterou chceme provést musí být schopno načíst. Manažer akcí zahrnuje tři hlavní třídy pro práci s funkcemi: *ActionDescriptor*, *ActionList*, *ActionReference*. [7], [3]

2.4.1 Instalace

Před využíváním samotného Manažera akcí je nutné nainstalovat plug-in (zásuvný modul) posluchače skriptů. Posluchač skriptů vytváří záznam do souboru, který odpovídá akcím provedeným v UI (uživatelském rozhraní). Z důvodu, že posluchač skriptů nahrává většinu provedených akcí, doporučuje se nainstalovat posluchač skriptů pouze tehdy pokud vytváříte skripty přes Manažera akcí. Jestliže posluchač skriptů je nainstalovaný i v případě, kdy není v úmyslu s ním pracovat, bude vytvářet velké soubory na pevném disku a zároveň zpomalovat výkon samotného Photoshopu. Výsledné zdrojové kódy skriptu lze tedy po každé provedené akci najít na své uživatelské ploše. [1]

Jak již bylo zmíněno posluchač skriptů bude vytvářet několik souborů, v případě provedení úlohy nebo více úloh, tyto soubory obsahují kód, který reprezentuje úlohy právě provedené. Soubor *ScriptingListenerJS.log* obsahuje kód v jazyce JavaScript a soubor *ScriptingListenerVB.log* obsahuje VBScript kód (používáte-li systém Windows). Zmíněné soubory můžeme vymazat, aby nezabírali místo na disku. [1]

Posluchač skriptů lze stáhnout z oficiálních stránek firmy Adobe, a to z URL (Uniform Resource Locator): https://www.adobe.com/devnet/photoshop/scripting.html.

Instalace[1]

1. Vyberu soubor *ScriptListener.8li* a zvolím Upravit > Kopírovat.

[1] Návod na instalaci a odinstalování posluchače skriptů čerpán z https://www.adobe.com/content/dam/acom/en/devnet/photoshop/scripting/photoshop-cc-scripting-guide.pdf, nesouhlasila adresářová struktura programu, a proto byla upravena.

2. Vložím zkopírovaný soubor do adresáře ..\Adobe\Adobe Photoshop CC 2018\Plug-ins.
3. Otevřu Photoshop pro nahrání plug-inu.

Odinstalování[1]

1. Ukončím program Adobe Photoshop.
2. Smažu soubor *ScriptListener.8li* z následujícího adresáře ..\Adobe\Adobe Photoshop CC 2018\Plug-ins.
3. Smažu výstupní soubory zdrojových kódů *ScriptingListenerJS.log* a *ScriptingListenerVB.log* z plochy obrazovky, jelikož při každé akci se budou vytvářet nové.

3 PRAKTICKÁ ČÁST

3.1 Nástroje a využité jazyky

V praktické části použiji program Adobe Photoshop, pro který je tato práce určena. Jako další software pro práci využiji od firmy Adobe prostředí Adobe ExtendScript Toolkit CS5, který je již v této práci zmíněný a nabízí velkou část možností zaměřených přímo pro psaní skriptů. Zdrojové kódy jsou psány v jazyce JavaScript, který je stěžejním kamenem pro tuto práci, je platformově nezávislý a zároveň podporovaný již zmíněným prostředím.

Pro tvorbu webových prezentací skládajících se z několika lekcí, kde jednotlivě probírám a komentuji využití skriptů, jsem si vybral open source projekt zvaný WebSlides (https://webslides.tv/#slide=1), díky kterému mám větší možnosti, jak prezentaci graficky vylepšit a zároveň prezentovat vzniklé skripty na přehledné úrovni. Ve webové prezentaci využiji převážně jazyk HTML (Hypertext Markup Language) a pro modifikování některých stylů prezentace jazyk CSS (Cascading Style Sheets).

Všechny zdrojové kódy a webové prezentace lze najít na CD (Compact Disc), které je přílohou bakalářské práce.

3.2 WebSlides

Webslides je open source projekt pro tvorbu webových prezentací, uživatelských příběhů a formulářů. Tato sada již napsaných a předdefinovaných stylů jak v jazyce CSS, tak v jazyce JavaScript obsahuje více než 40 komponent pro tvorbu prezentací, samozřejmostí je počítadlo slidů, bloky, formy, skripty reagující na najetí myši. Všechny tyto skripty jsou volně přístupné a po stažení si je uživatel může modifikovat a upravovat podle své libosti. Soubory jsou dostupné jako komprimovaný soubor ve formátu *.zip, který po rozbalení obsahuje i již vytvořená dema prezentací, které demonstrují a ilustrují chování a možnosti modifikace jednotlivých komponent. V praxi poté tedy v HTML souboru píši kód a využívám již napsané styly, které jsem si dle libosti upravil.

Obrázek 6 – Webová prezentace zobrazující lekci z praktické části za využití WebSlides (zdroj: David Baláček)

3.3 Postupy tvoření

Při vytváření praktické části bakalářské práce, bylo důležité dobře zvolit pořadí lekcí a druhy skriptů, kterým bych se měl věnovat, aby skripty byly využitelné i pro další využití, kromě náplně praktické části. Ve chvíli, kdy druh skriptu byl navržený, začal jsem samotné skriptování a implementaci tříd, které vedly k výsledku. Při nalezení chyb či nesrovnalostí, jsem se opíral o použitou literaturu této práce a diskuzní fóra, kde se řešil případný podobný problém. Daný skript byl sepsán, několikrát ozkoušen a uložen jako výsledný *.jsx soubor. Po této části začala vznikat samotná lekce, jako webová prezentace, kde po úvodu následuje cíl dané lekce, použité třídy, či jiné technologie, poté se prezentace věnuje krok po kroku sestavení skriptu. Skript je sestavován v prezentaci po krocích, ve kterých je vysvětleno, co daná část skriptu zajišťuje a k čemu je důležitá. Daný kód je poté ještě doplněn o komentáře, z důvodu přehlednosti. Na posledních snímcích prezentace je zobrazen výsledný kód, jako celek pro možnosti kontroly a vizualizaci finální podoby skriptu s barevnou syntaxí. Praktickou část bakalářské práce je možné najít na webové adrese http://automatizace-uloh-photoshop.jednoduse.cz/index.html.

3.4 Ukázka zdrojového kódu webové prezentace

```
<!DOCTYPE html>
<html lang="cs">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="description" content="Automatizování úloh v Adobe
Photoshop">
    <meta name="author" content="David Baláček">
    <title>Automatizování úloh v Adobe Photoshop</title>
    <link rel="stylesheet" href="styles/style.css" type="text/css">
```

```
    <link rel="stylesheet" type='text/css' media='all'
href="static/css/prism.css">
    <link rel="stylesheet" type='text/css' media='all'
href="static/css/webslides.css">
    <link rel="stylesheet" href="styles/atom-one-light.css">
    <script src="static/js/highlight.pack.js"></script>
    <script>hljs.initHighlightingOnLoad();</script>
</head>
<header class="bg-white">
    <nav role="navigation" class="navbar">
        <ul>
            <li><a href="lekce_1.html">Lekce 1</a></li>
            <li><a href="lekce_2.html">Lekce 2</a></li>
            <li><a href="lekce_3.html">Lekce 3</a></li>
            <li><a href="lekce_4.html">Lekce 4</a></li>
            <li><a href="lekce_5.html">Lekce 5</a></li>
            <li><a href="lekce_6.html">Lekce 6</a></li>
            <li><a href="lekce_7.html">Lekce 7</a></li>
            <li><a href="lekce_8.html">Lekce 8</a></li>
            <li><a href="lekce_9.html">Lekce 9</a></li>
            <li><a href="lekce_10.html">Lekce 10</a></li>
            <li><a href="lekce_11.html">Lekce 11</a></li>
            <li><a href="index.html">O práci</a></li>
            <li><a href="otazka.html">Otázka k práci</a></li>
        </ul>
    </nav>
</header>
<body>
<div class="wrapper" style="text-align:justify">
    <h1 class="nadpis" style="text-align:center">Seřazení vstrev podle
abecedy</h1>
    <h2 class="nadpis1" style="text-align:center">lekce 6.</h2>
    <h4 class="nadpis1" style="text-align:center">Automatizování úloh
v Adobe Photoshop</h4>
    <h5 class="nadpis1" style="text-align:center; font-weight:
normal;margin-top: -5px;">
    Pomocí skriptovacího jazyka JavaScript
    </h5>
    <hr>
    <div id="content">
        <h3>Cíl lekce</h3>
        <p>Emboss.</p>
        <p>Cílem této krátké lekce je vytvořit emboss. U embossingu dochází
k vytlačování vzoru (obrazců)
        nebo textu do hloubky podkladu. Zárveň bych v této lekci ukázal
práci s třídami <i>ActionReference</i> a <i>ActionDescriptor</i>
            a volání akce z panelu <i>Akce</i> v programu Adobe
Photoshop.</p>
        <a href="./images/lekce_4Puvodni.png" target="_blank">
            <img class="center" src="images/lekce_4Puvodni.png"
alt="lekce_4Puvodni.png"></a>
        <h5><p class="center">Původní obrázek - obrázek 6.1</p></h5>
        <a href="./images/lekce_4Vysledny.png" target="_blank">
            <img class="center" src="images/lekce_4Vysledny.png"
alt="lekce_4Vysledny.png"></a>
        <h5><p class="center">Emboss - obrázek 6.2</p></h5>
        <hr>
        <h3>Využité třídy v této lekci</h3>
        <div>
            <ul class="description">
                <li><span class="text-label">Descriptor:</span>
```

```
                    třída pro popis akce.
                </li>
                <li><span class="text-label">Konstanty:</span>
                    konstanty jako klíče pro <i>ActionDescriptor</i>.
                </li>
            </ul>
        </div>
        <hr>
        <div>
            <h3><i>ActionDescriptor, ActionReference</i></h3>
            <p>ActionDescriptor: udává záznam páru, klíč – hodnota pro
akce, jako jsou například akce dostupné z panelu
                <i>Akcí</i> v prostředí. Je součástí <i>Správce akcí</i>.
                ActionReference: obsahuje data popisující odkazovanou akci.
Referenční objekt akce je také součástí
                funkce <i>Správce akcí</i>. </p>
        </div>
        <hr>
        <h3>Konstanty</h3>
        <p></p>
        <p>Abych mohl provést akci emboss, musím do
<i>ActionDescriptoru</i> vložit hodnoty klíče. Jedná
            se tedy o klíče:
            <i>Angl</i> úhel, <i>Hght</i> výška, <i>Amnt</i> množství. Dále
použiji <i>Embs</i> pro identifikaci
            akce <i>Emboss</i>.</p>
        <hr>
        <h6>Funkce emboss</h6>
        <p></p>
        <p>Nyní už jen zbývá vytvořit funkci emboss s parametry úhel, výška
a množství. Získání ID událostí a
            vložení klíčů a hodnot do <i>ActionDescriptoru</i>
            pomocí metody <i>putIteger()</i>. Pak již mohu zavolat metodu
<i>executeAction</i> s ID události emboss.
        </p>
        <pre class="language-javascript"><code data-pygments="ignore">
function emboss(inAngle, inHeight, inAmount){
    //konstanty
    var keyAngleID = app.charIDToTypeID('Angl');
    var keyHeightID = app.charIDToTypeID('Hght');
    var keyAmountID = app.charIDToTypeID('Amnt')
    var eventEmbossID = app.charIDToTypeID('Embs');

    //vložení hodnot do descriptoru
    var filterDescriptor = new ActionDescriptor();
    filterDescriptor.putInteger(keyAngleID, inAngle)
    filterDescriptor.putInteger(keyHeightID, inHeight)
    filterDescriptor.putInteger(keyAmountID, inAmount)
    //provedení akce
    app.executeAction(eventEmbossID, filterDescriptor)
}
</code></pre>
        <hr>
    <div>
         <h3>Výsledný zdrojový kód</h3>
         <pre class="language-javascript"><code data-pygments="ignore">
if (documents.length) {
    emboss( 60, 10, 60);
}else{
    alert("Žádný dokument není otevřený.", "Varování");
}
```

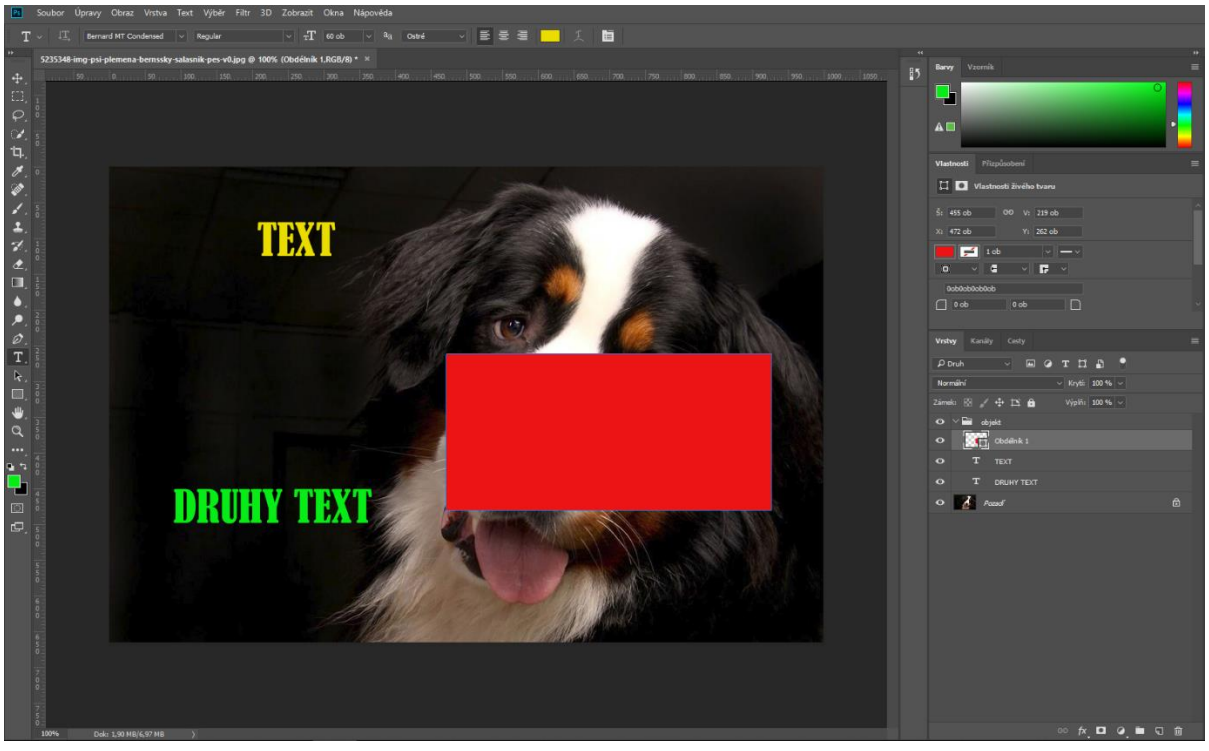
```
function emboss(inAngle, inHeight, inAmount){
    //konstanty
    var keyAngleID = app.charIDToTypeID('Angl');
    var keyHeightID = app.charIDToTypeID('Hght');
    var keyAmountID = app.charIDToTypeID('Amnt')
    var eventEmbossID = app.charIDToTypeID('Embs');

    //vložení hodnot do descriptoru
    var filterDescriptor = new ActionDescriptor();
    filterDescriptor.putInteger(keyAngleID, inAngle)
    filterDescriptor.putInteger(keyHeightID, inHeight)
    filterDescriptor.putInteger(keyAmountID, inAmount)
    //provedení akce
    app.executeAction(eventEmbossID, filterDescriptor)
}
```
```
          </code></pre>
        </div>
    </div>
    <br>
</div>
</body>
</html>
```

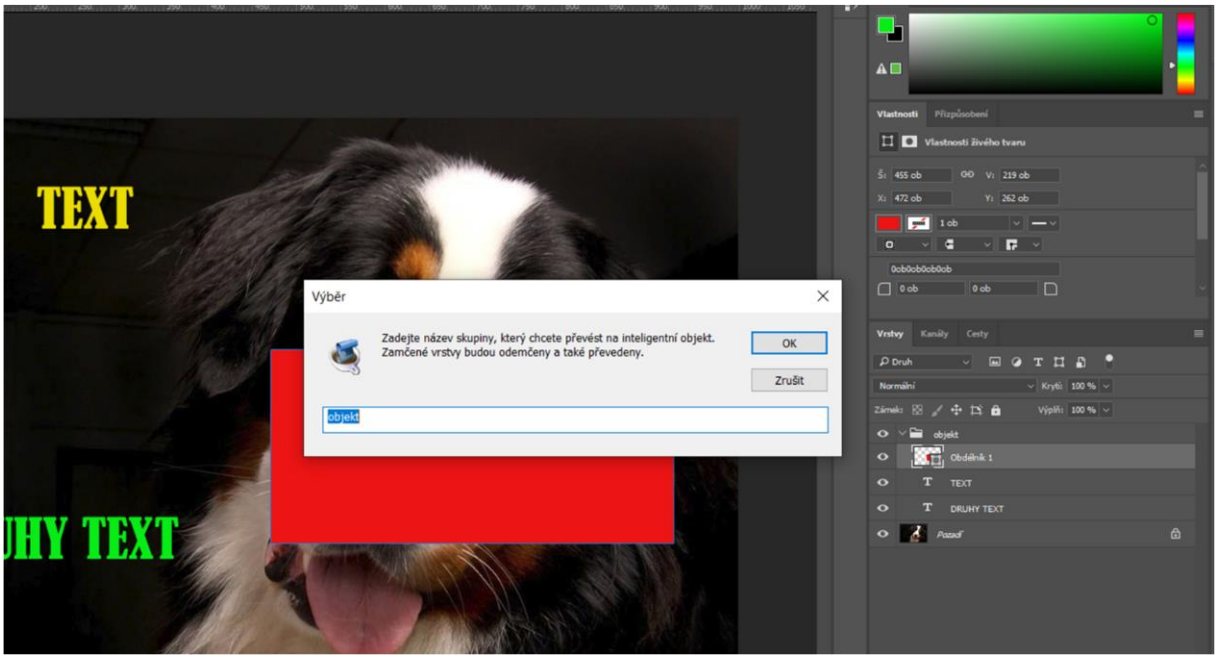
3.5 Lekce praktické části – Vytvoření inteligentního objektu, lekce 9.

3.5.1 Cíl lekce

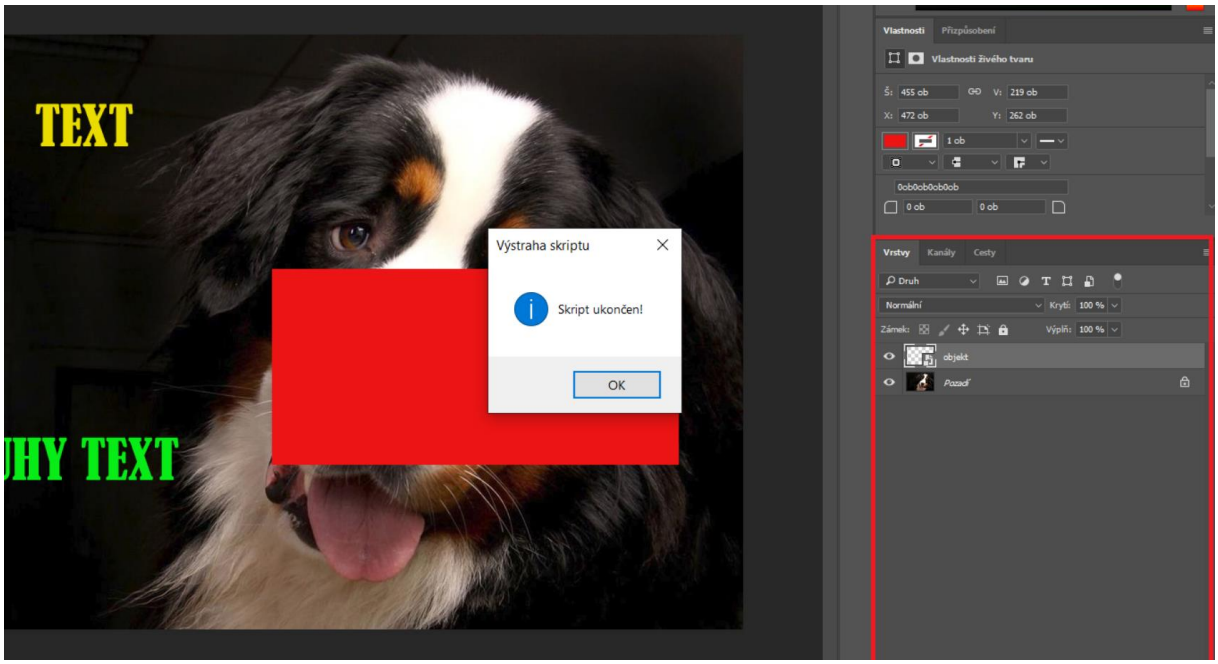
V této lekci vytvořím ze skupiny vrstev, dané uživatelem, jeden inteligentní objekt v prostředí programu Adobe Photoshop. Z počátku zajistím výjimky, které mohou nastat, jako je neotevřený soubor, či existence samotné skupiny vrstev. Uživatel má navíc možnost pomocí dialogového okna, zadat název skupiny vrstev, kterou chce převést do inteligentního objektu.



Obrázek 7 - Původní grafika (zdroj: David Baláček)



Obrázek 8 - Zadávání do dialogového okna (zdroj: David Baláček)



Obrázek 9 - Ukončený skript (zdroj: David Baláček)

3.5.2 Využité třídy, technologie v této lekci

- Application: pro práci s atributy dokumentu
- Konstanty: konstanty výčtových typů pro událost
- Descriptor: pro popis prováděných akcí z rozhraní
- Reference: pro zadání objektů do objektu ActionDescriptor

3.5.3 Vytvoření dialogu a dosazeních do proměnných

Metoda *prompt()* slouží jako dialogové okno s dvěma tlačítky a oblastní do které může uživatel psát vstup. Samotná metoda vrací textový řetězec a v případě prázdného pole hodnotu null. První parametr této metody je zpráva hlavního okna, druhý parametr označuje již předvyplněné textové pole (v tomto případě nastavím první nalezenou skupinu vrstev, aby bylo zřejmé, co má uživatel zadat) a třetí je titulek dialogového okna. Celou metodu poté přiřadím pomocí rovná se do proměnné, jelikož metoda má návratovou hodnotu. Obsah lze jednoduše vypsat pomocí informačního okna, pro kontrolu, že přiřazení je funkční viz. zakomentovaný příkaz. Dále do proměnné *layerSet* přiřadím všechny skupiny vrstev v aktivním dokumentu.

```
var placeHolder;
//získej skupiny vrstev
var layerSet = activeDocument.layerSets;
//pokud existuje více skupin vrstev než 0, nastavím
//předvyplněné pole prvním
//názvem nalezené skupiny vrstev
layerSet.length > 0 ? placeHolder = layerSet[0].name :
placeHolder = "";

//přiřaď název skupiny vrstev
var nameOfSet = prompt("Zadejte název skupiny, který chcete "
+ "převést na inteligentní objekt.\n" + "Zamčené vrstvy budou
" + "odemčeny a také převedeny.", placeHolder, "Výběr");
```

3.5.4 Vytvoření funkce pro kontrolu skupiny vrstev

Jako další krok potřebuji zkontrolovat, pokud uživatelem zadaný název existuje jako skupina vrstev a skript se může provést. To ošetřím jednoduchou podmínkou, kde v cyklu procházím všechny skupiny vrstev a ptám se, či se jejich atribut *name* rovná názvu, který zadal uživatel. V tomto případě se jedná o funkci s dvěma parametry, tedy skupinou a názvem. Při nalezení zadané skupiny vrstev je třeba také ošetřit zamčení jednotlivých vrstev. Proto si nalezenou skupinu vrstev vložím jako parametr do funkce, která ošetří zamčení vrstev.

```
function doesExist(layerSet, nameSet){
    for (var i = 0; i < layerSet.length; i++) {
        if(layerSet[i].name == nameSet){
            unlockLayers(layerSet[i]);
            return true;
        }
    }
    return false;
}
```

3.5.5 Ošetření zamčených vrstev

Na ošetření zamčených vrstev vytvořím jednoduchou funkci *unlockLayers*, která bude mít za úkol jen projít vrstvy skupiny a nastavit atribut *allLocked,* na false. Nastavením tohoto atributu odemknu vrstvu a vyhnu se případným výjimkám u dalšího zpracování vrstev. Tuto funkci ilustruje zdrojový kód níže.

```
function unlockLayers(layerSet){
    for (var i = 0; i < layerSet.artLayers.length; i++) {
         // odemčení vrstvy
         layerSet.artLayers[i].allLocked = false;
    }
}
```

3.5.6 Event ID

Podobně jako v předchozích lekcích potřebují získat event ID z výčtových typů a to ze znakové a řetězcové konstanty. Opět jako u předcházejících lekcí je možné najít konstanty v rozšířených příručkách společnosti Adobe, či na githubovém portálu - https://github.com/pritianka/photoshopwindows/blob/master/pluginsdk/photoshopapi/photoshop/PITerminology.h. Ve zkratce ke konstantám, které použiji: "Lyr " značí vrstvy, "slct" značí výběrovou událost, null prázdná hodnota.

```
//event ID ze znaku
function getIdFromChar(param) {
    return app.charIDToTypeID(param)
}
//event ID z řetězce
function getIdFromString(param) {
    return app.stringIDToTypeID(param)
};
```

3.5.7 Funkce pro vybrání vrstev ve skupině

Nyní vytvořím objekty *ActionDescriptor*, *ActionReference*. Poté do *ActionReference* pomocí funkce *putName()* vložím vrstvy v dané skupině a referenci vložím do objektu *ActionDescriptor*. Na první místě, kde se očekává klíč vložím null a jako hodnotu vložím vytvořenou referenci. Poté pomocí funkce *executeAction()* provedu výběr bez dialogových oken sestupně. Tuto metodu obalím blokem try{ }catch (){} v případě výskytu chyby.

```
function doSelection(nameSet){
    var descriptor = new ActionDescriptor();
    var reference = new ActionReference();
    reference.putName( getIdFromChar( "Lyr " ), nameSet);
    descriptor.putReference( getIdFromChar( "null" ),
reference );
    try{
```

```
        executeAction( getIdFromChar( "slct" ), desc,
DialogModes.NO );
    }catch(exception){
        return false;
    }
}
```

3.5.8 Využití funkcí a sestavení skriptu

Nyní mám vše, co potřebuji a mohu funkce zakomponovat do celého skriptu. Nejprve je ale ještě třeba ošetřit, zda uživatel zadal nějaký název a nenechal pole prázdné, zároveň jestli skupina vrstev s názvem zadaným od uživatele existuje. Dalším důležitým krokem je nastavit aktivní vrstvu, toto je velice zásadní, protože v případě že uživatel má vybranou libovolnou jinou vrstvu než zadal, vytvořil by se inteligentní objekt z této vrstvy a nikoliv z vrstvy zadané. Poté provedu výběr, ten se díky akce transformuje do vrstvy, kterou zviditelním a poté zavolám akci pro vytvoření inteligentního objektu. Poslední část skriptu ilustruje kód níže.

```
if(doesExist(layerSet, nameOfSet)){
        //nastavení activní vrstvy v dokumentu na uživateli
vybranou vrstvu
        app.activeDocument.activeLayer =
app.activeDocument.layerSets.getByName(nameOfSet)

        //vyber vrstvy
        doSelection(nameOfSet);
        //zviditelni aktivní vrstvu
        activeDocument.activeLayer.visible = true;
        //proveď akci pro převedení do chytrého objektu
        executeAction(getIdFromString('newPlacedLayer'),
undefined, DialogModes.NO);

        //informuji o dokočení skriptu
        alert("Skript ukončen!")
}else if(nameOfSet.length > 0 && !doesExist(layerSet,
nameOfSet)){
        alert ("Zadaná skupina neexistuje!", "Upozornění");
}
```

3.5.9 Výsledný zdrojový kód

```
//je otevřený dokument?
if (isDocumentOpened()) {
    var placeHolder;
    //získej skupiny vrstev
    var layerSet = activeDocument.layerSets;
    //pokud existuje více skupin vrstev než 0, nastavím
předvyplněné pole prvním
    //názvem nalezené skupiny vrstev
    layerSet.length > 0 ? placeHolder = layerSet[0].name :
placeHolder = "";
```

```
    //přiřaď název skupiny vrstev
    var nameOfSet = prompt("Zadejte název skupiny, který
chcete převést na inteligentní objekt.\n" +
        "Zamčené vrstvy budou odemčeny a také převedeny.",
placeHolder, "Výběr");

    if (nameOfSet !== null) {
        //existuje tato skupina?
        if (doesExist(layerSet, nameOfSet)) {
            app.activeDocument.activeLayer =
app.activeDocument.layerSets.getByName(nameOfSet)

            //vyber vrstvy
            doSelection(nameOfSet);
            //zviditelni aktivní vrstvu
            activeDocument.activeLayer.visible = true;
            //proveď akci pro převedení do chytrého objektu
            executeAction(getIdFromString('newPlacedLayer'),
undefined, DialogModes.NO);

            alert("Skript ukončen!")
        } else if (nameOfSet.length > 0 &&
!doesExist(layerSet, nameOfSet)) {
            alert("Zadaná skupina neexistuje!", "Upozornění");
        }
    }
} else {
    alert("Žádný dokument není otevřený.", "Varování");
}


function isDocumentOpened() {
    return documents.length;
}


function doesExist(layerSet, nameSet) {
    for (var i = 0; i < layerSet.length; i++) {
        if (layerSet[i].name == nameSet) {
            unlockLayers(layerSet[i]);
            return true;
        }
    }
    return false;
}

function unlockLayers(layerSet) {
    for (var i = 0; i < layerSet.artLayers.length; i++) {
        // odemčení vrstvy
        layerSet.artLayers[i].allLocked = false;
```

```
    }
}

function doSelection(nameSet) {
    var descriptor = new ActionDescriptor();
    var reference = new ActionReference();
    reference.putName(getIdFromChar("Lyr "), nameSet);
    descriptor.putReference(getIdFromChar("null"), reference);
    try {
        executeAction(getIdFromChar("slct"), desc,
DialogModes.NO);
    } catch (exception) {
        return false;
    }
}

function getIdFromChar(param) {
    return app.charIDToTypeID(param)
}

function getIdFromString(param) {
    return app.stringIDToTypeID(param)
}
```

ZÁVĚR

Vytvoření skriptu pro automatizaci úloh v programu Adobe Photoshop, ať už v jazyce JavaScript nebo VBScript, není zdaleka vůbec složité a měl by jej zvládnout i běžný uživatel. Kromě této bakalářské práce, zejména praktické části, je možné najít několik dobrých zdrojů pro skriptování úloh. Nutno podotknout, že je ovšem třeba seznámit se syntaxí jazyka a logicky uvažovat v jakých krocích se skript musí napsat a co by měl obsahovat.

Jestliže tyto základní předpoklady jsou zvládnuty, skriptování úloh je určitě jedna z nejlepších možností, jak ušetřit čas při vytváření grafiky a určitě doporučuji se nad touto variantou zamyslet v případě, kdy máte opravdu velké množství grafických souborů ke zpracování, z důvodu ušetření manuální práce. I přes to, že napsání skriptu a odladění trvá trochu déle, pokud není člověk znalý jazyka a programu, určitě se vyplatí tento čas investovat, jelikož skript poté ušetří i trojnásobek investovaného času a je-li dobře napsán, je možné část zdrojového kódu znovu použít pro jiný skript.

POUŽITÁ LITERATURA

[1] Adobe Systems Incorporated: *Adobe Photoshop CC Scripting Guide* [Online]. 2013 [cit. 2019-03-18]. URL: https://www.adobe.com/content/dam/acom/en/devnet/photoshop/scripting/photoshop-cc-scripting-guide.pdf.

[2] Adobe Systems Incorporated: *Introduction To Scripting* [Online]. 2007 [cit. 2019-03-18]. URL: https://www.adobe.com/content/dam/acom/en/devnet/illustrator/pdf/adobe_intro_to_scripting.pdf.

[3] Adobe Systems Incorporated: *Adobe Photoshop CC JavaScript scripting Reference* [Online]. 2013 [cit. 2019-03-18]. URL: http://wwwimages.adobe.com/www.adobe.com/content/dam/acom/en/devnet/photoshop/pdfs/photoshop-cc-javascript-ref.pdf.

[4] Adobe Systems Incorporated: *Adobe Photoshop CC VBScript scripting Reference* [Online]. 2013 [cit. 2019-03-18]. URL: https://www.adobe.com/content/dam/acom/en/devnet/photoshop/pdfs/photoshop-cc-vbs-ref.pdf.

[5] Adobe Systems Incorporated: *JavaScript Tools Guide* [Online]. 2010 [cit. 2019-03-18]. URL: https://www.adobe.com/content/dam/acom/en/devnet/scripting/pdfs/javascript_tools_guide.pdf.

[6] Adobe Systems Incorporated: *Photoshop 7.0 Scripting Guide* [Online]. 1991-2002 [cit. 2019-04-11]. URL:https://www.manualslib.com/manual/471261/Adobe-23101335-Photoshop-Pc.html?page=84 .

[7] *Ps-scripts forum* [online]. © 2018. URL: https://www.ps-scripts.com/\.

[8] Smashing magazine*: Introduction to Photoshop Scripts* [online]. 2013-07-25 [cit. 2019-04-11]. URL: https://www.smashingmagazine.com/2013/07/introduction-to-photoshop-scripting/.

[9] Davide Barranca: Professional Photoshop Scripting [online]. 2016-2019. Free 50 pages. URL: https://www.ps-scripting.com/.

[10] Creative Bloq Staff: *How to create your own Photoshop scripts* [online]. 2013-04-23. URL: https://www.creativebloq.com/photoshop/scripts-4132441.

[11] Trevor Morris: *Scripting Photoshop, Part 2 – A Practical Example* [online]. © 2014. URL: http://morris-photographics.com/photoshop/tutorials/scripting2.html.

[12] David Flanagan: *JavaScript: The Definitive Guide*. 5. vydání. O'Reilly Media, Inc., 2006. ISBN: 978-05-961-0199-2.

PŘÍLOHY

PŘÍLOHA A – CD

CD přiložené k této bakalářské práci obsahuje zdrojové kódy v jazyce JavaScript, webové prezentace ve formě několika lekcí. Podrobnější popis CD a obsahujících adresářů následuje níže.

- thesis/ obsahuje bakalářskou práci ve formátu *.docx a *.pdf
- scripts/ obsahuje skripty v jazyce JavaScript, které jsou využity v lekcích
- prezentation/ obsahuje webové prezentace, které jsou průvodce napsanými skripty