Primljen: 11.03.2020.
Prihvaćen: 13.05.2020.

Stručni rad
UDK: 004.42InDesign

HTML5 ekstenzije za Adobe InDesign CC aplikacije

HTML5 extensions for Adobe InDesign CC applications

[1]Sanja Brekalo, [2]Klaudio Pap, [3]Željko Knok
[1,3]Međimursko veleučilište u Čakovcu, Bana J. Jelačića 22A, 40000 Čakovec
[2]Grafički fakultet u Zagrebu, Getaldićeva 2, 10000 Zagreb
e-mail: [1]sanja.brekalo@mev.hr, [2]klaudio.pap@grf.hr, [3]zeljko.knok@mev.hr

Sažetak: *Adobe u svojim aplikacijama omogućuje tehnologiju proširivanja, koja daje infrastrukturu za razvoj i dijeljenje ekstenzija koje rade kroz različite Adobe aplikacije. Adobe Application Extension je set datoteka koje zajedno proširuju mogućnosti jedne ili više Adobe aplikacija. Ekstenzije se mogu koristiti kako bi se dodali servisi i integrirale nove mogućnosti kroz aplikacije u CC paketima. Adobe Application Exstension koristi HTML i JavaScript kako bi se kreirala sučelja koja su podržana na različitim platformama. Ekstenzije imaju pristup skriptnom sučelju aplikacije za koju su pisane i mogu raditi sa skriptnim DOM kako bi ostvarile interakciju s dokumentima koji se uređuju u različitim Adobe programima. Adobe programi podržavaju ExtendScript što je Adobe proširena verzija ECMA JavaScript.*

U radu je opisana konfiguracija osnovne ekstenzije za Adobe InDesign te osnove korištenja CEP (Adobe Common Extensibility Platform) pri izradi automatizacije za Adobe InDesign program. Ekstenzije su građene prema zadanoj arhitekturi kojom se bavi ovaj rad. Opisane su mogućnosti izrade ekstenzije kao i njeni sastavni dijelovi.

Ključne riječi: *digitalna grafička priprema, InDesign, ExtendScript, JavaScript, HTML5, CEP*

Abstract: *Adobe provides extension technology in its applications, which provides the infrastructure to develop and share extensions that work across different Adobe applications. Adobe Application Extension is a set of files that together extend the capabilities of one or more Adobe applications. Extensions can be used to add services and integrate new capabilities across applications in CC packages. Adobe Application Extension uses HTML*

and JavaScript to create interfaces that are supported across platforms. Extensions have access to the scripting interface of the application they were written for and can work with the scripting DOM to interact with documents that are edited in different Adobe programs. Adobe programs support ExtendScript which is Adobe's extended version of ECMA JavaScript.

This paper describes the basic extension configuration for Adobe InDesign and the basics of using CEP (Adobe Common Extensibility Platform) when creating automation for Adobe InDesign. The extensions are built according to the default architecture that this work deals with. The possibilities of extensions creation and its parts are described.

Keywords: *digital prepress, InDesign, ExtendScript, JavaScript, HTML5, CEP*

1. Uvod

Adobe u svojim aplikacija omogućuje tehnologiju proširivanja preko CC ekstenzija. U CC ekstenzijama HTML i JavaScript izgrađuju korisničko sučelje. Korisničko sučelje komunicira s aplikacijom za koju je pisano proširenje te se poslovi uređivanja dokumenata i objekata ostvaruju preko EtendScript skriptnog jezika i manipulacije DOM u samoj aplikaciji.

U posljednje vrijeme postoji sve više alata koji nude JavaScript podršku za upravljanje produkcijskim razvojnim okruženjem. JavaScript je postao moćan jezik koji se koristi gotovo svuda, od web i mobilnih aplikacija, nativnih aplikacija do IoT te se može koristiti u programima namijenjenim za pripremu za tisak [1]. Podrška za JavaScript postojala je i ranije u obliku ExtendScript sintakse dok je danas moguće u Adobe okruženju izvršavati standardni JavaScript kod te sve module i biblioteke koji su popularni na webu.

Ekstenzije koje se kreiraju su usko integrirane u Adobe programe što omogućuje kontrolu ekstenzija kao da su ugrađene u aplikaciju. Ekstenzije se pozivaju preko izbornika aplikacije i ovisno o tipu ekstenzije mogu se ponašati kao standardni paneli u programu. Korisnici mogu dodavati i brisati ekstenzije prema željama kako bi podesili svoje Adobe desktop aplikacije prema vlastitim potrebama. [2]

2. Adobe Common Extensibility Platform (CEP)

Adobe Common Extensibility Platform (CEP) je tehnologija ugrađena u Adobe Creative Cloud (CC) *desktop* aplikacije. Prethodno je CEP bio poznat kao CSXS i dostupan je od Adobe Creative Suite 4 (2008. godina). CEP omogućava radno okruženje koje je integrirano s ExtendScript podrškom za skriptiranje. CEP je građen isključivo na web tehnologijama,

korisničko sučelje se izrađuje sa standardnim HTML5 dok se logika izrađuje u JavaScript kodu.

CEP je web-preglednik koji je ugrađen u InDesign i ostale CC aplikacije te je povezan sa skriptnim DOM (engl. *document object model*). CEP je izgrađen oko *Google Chromium Embedded Framework* (CEF). CEF omogućava uključivanje Google Chrome preglednika u aplikaciju te ima podršku za Google V8 JavaScript *engine*. [3]

Tabela 1. Prikazuje CEP verzije u različitim inačicama Adobe InDesign aplikacije. U Tabeli 2. prikazane su podržane verzije CEF, Chrome preglednika te Node.js u različitim verzijama CEP. Node.js je radna i razvojna platforma izgrađena na bazi Chrome Javascript radnog okruženja (engl. *engine).* Node.js nije programski jezik već okruženje (engl. *runtime environment*) za izvršavanje JavaScript koda. Omogućuje izvršavanje zadataka koji su onemogućeni u web-pregledniku kao primjerice čitanje datotečnog sustava, slušanje *http* zahtjeva na pojedinom portu i dr.

Tabela 1. *CEP verzije programa Adobe InDesign ovisno o CC verziji* [4]

Aplikacija	**ID aplikacije**	**CC**	**CC 2014**	**CC 2015**	**CC 2017**	**CC 2018**	**CC 2019**
InDesign	IDSN	9 (CEP 4)	10 (CEP 5)	11 (CEP 6)	12 (CEP 7)	13 (CEP 8)	14 (CEP 9)

Tabela 2. *Podržane CEF verzije u različitim verzijama CEP* [4]

Komponenta	**CEP 6.1 and CEP 7.0**	**CEP 8.0**	**CEP 9.0**
CEF 3	CEF 3 release branch 2272 Commit e8e1f98ee026a62778eb2269c8e883426db645ea	CEF 3 release branch 2987	CEF 3 release branch 3163
Chromium	41.0.2272.104	57.0.2987.74	61.0.3163.91
Node.js	IO.js 1.2.0	Node.js 7.7.4	Node.js 8.6.0
CEF/Node integracija	Node-WebKit 0.12.1 (nw.js)	Node-Webkit 0.21.5	Node-Webkit 0.25

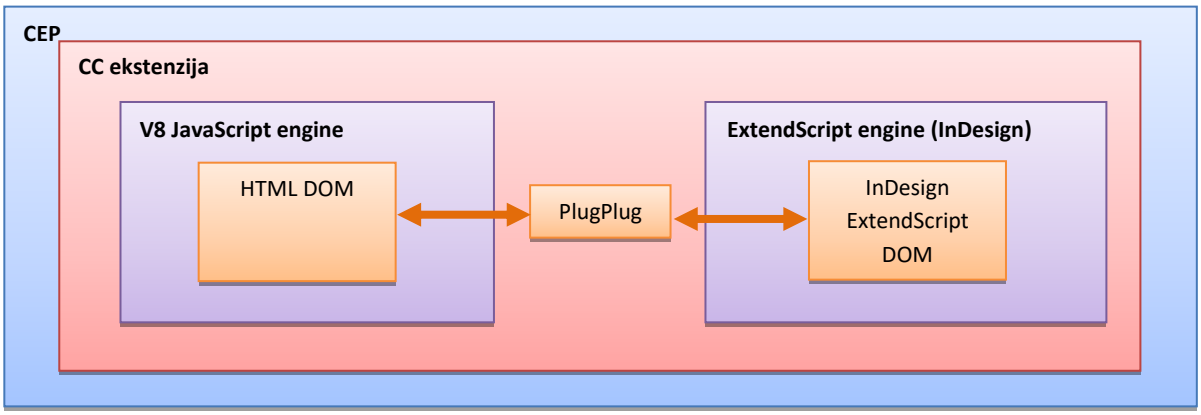
CEP se povezuje s CC aplikacijama komunikacijom preko Adobe PlugPlug komponente. Preko PlugPlug ostvaruje se komunikacija između V8 JavaScript okruženja i ExtendScript DOM. CC ekstenzije građene na temelju CEP tipično uključuju 2 glavna dijela: korisničko sučelje izrađeno u HTML-u i JavaScript-u i jedne ili više ExtendScript datoteka koje manipuliraju InDesign DOM. HTML/JavaScript dio ekstenzije se može povezati na web, baš kao i standardni neugrađeni preglednik. ExtendScript dio omogućuje pisanje skripti koje automatiziraju digitalnu grafičku pripremu [5].

CEP također proširuje CEF dodavanjem Node.js. Node.js daje API (engl. *Application Programming Interface*) za pristupanje datotečnom sustavu, interakciju s webom (npr. čitanje baze, preuzimanje i slanje datoteka), kriptografiju, događaje i ostale funkcije. Node.js je proširiv, te su korištenjem Node.js dostupne tisuće modula (preko *Node Packaged Modules* (NPM)) koji omogućuju razne dodatke kao verifikaciju korisnika, povezivanje s bazama podataka i dr. Glavana prednost korištenja NPM modula je da se mogu koristiti gotovi moduli i isječci koda izrađeni u JavaScriptu koji mogu ubrzati izradu skrpiti za automatizaciju.

CEP integrira i Adobe IPC Toolkit, komunikacijski sistem za Adobe CC aplikacije, preko kojeg se primjerice može kontrolirati Photoshop ili Illustrator iz InDesign ekstenzije. Adobe IPC Toolkit je komponenta koja se može pozivati iz CEP kako bi se omogućila komunikacija i kontrola između različitih CC *desktop* aplikacija korištenjem Vulcan Message Library i Vulcan Control Library.

Slika 1. prikazuje strukturu CEP. CC ekstenzija sadrži dva radna okruženja za izvršavanje koda, te se oni ne izvršavaju u istom zajedničkom globalnom kontekstu. V8 JavaScript *engine* je omogućen pri izradi CC ekstenzija za izradu korisničkog sučelja. CEP V8 JavaScript *engine* je samostalna izvršna datoteka koja omogućuje renderiranje HTML ekstenzija i izvršavanje V8 JavaScript koda. Koristi projekt otvorenoga koda CEF verzije 3[1]. U navedenom radnom okruženju izvršava se i Node.js. ExtendScript *engine* je standardan *engine* u Adobe aplikacijama koji podržava izvršavanje skripti pisanih za pojedine Adobe aplikacije. Također je vidljivo da postoje i dva DOM. Bitno je naglasiti da ta dva dijela aplikacije komuniciraju preko PlugPlug.

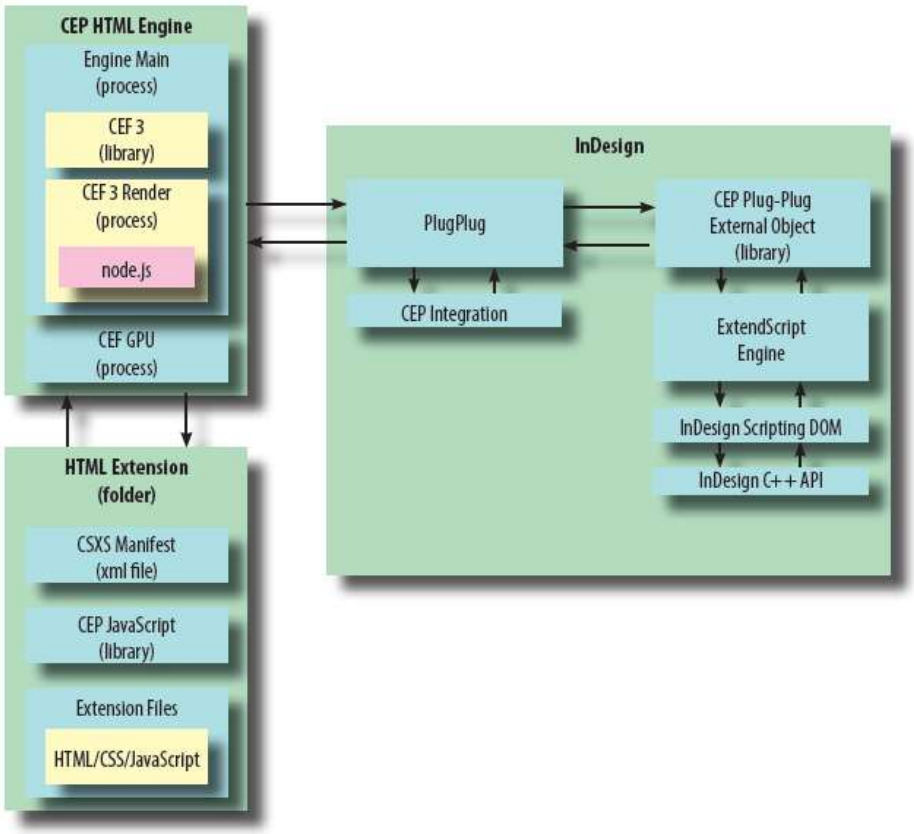
[1] Projekt je opisan na http://code.google.com/p/chromiumembedded/



Slika 1. *Komunikacija u Adobe's Common Extensibility Platform (CEP) tehnologiji*

Izvor: Autori

CEP prikazuje HTML sadržaj CC ekstenzije izrađen za korisničko sučelje unutar prozora aplikacije. Slika 2. prikazuje odnos između dijelova koji čine CEP u Adobe InDesign aplikaciji.



Slika 2. *Dijagram arhitekture komponenti* [3]

PlugPlug je dijeljena komponenta koja omogućuje komunikaciju između aplikacije i CEP komponenti. PlugPlug kontrolira životni ciklus CEP V8 JavaScript *engine*. PlugPlug External Object (*library*) je ExtendScript biblioteka koja omogućuje API za komunikaciju V8 JavaScript *engine* i ExtendScript DOM. Ranije ju je bilo potrebno uključivati u projekte dok je u novijim verzijama InDesign ugrađena u osnovnu instalaciju. PlugPlug pokreće HTML *engine* proces kad se ekstenzija učita i zatvara ga kad se ekstenzija zatvori. Glavni proces HTML *engine*-a je CEF proces preglednika i može imati jedan ili više *Render* i *GPU* procesa. U CEF 3 *Render* procesu izvršava se i Node.js.

HTML ekstenzije su direktoriji koji sadrže *manifest.xml*, jednu ili više HTML, JavaScript, ExtendScript i CSS datoteka. HTML ekstenzija je kao web-stranica koja se može povezati s InDesign dokumentima, stranicama i elementima na stranici.

3. Izrada ekstenzije

Za izradu CC ekstenzije potreban je jedan od Adobe CC programa za koji se želi izraditi ekstenzija. U primjeru izrade ekstenzije korišten je Adobe InDesign CC 2019 verzije 14.0 x64 koji podržava CEP verzije 9. Korišteni OS je Windows 10 Home x64.

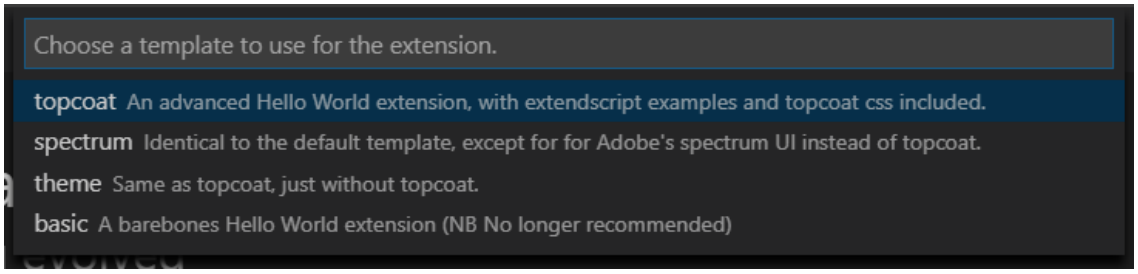
Za izradu CC ekstenzija dovoljan je samo uređivač teksta jer su ekstenzije izrađene u kombinaciji HTML, CSS i JavaScript. Preporuka je koristiti web razvojna okruženja kako bi se olakšala izrada ekstenzije. Proučavanjem dokumentacije u većini slučajeva preporuča se korištenje Extension Builder 3 sa stranice Adobe Labs[2] kao razvojno okruženje za izradu ekstenzija. U navedenom okruženju javlja se problem kod izrade ekstenzija za novije verzije programa. Kako je Extension Builder izdan 2013. godine pri izradi ekstenzije potrebno je dodatno urediti skripte kako bi one ispravno radile u novijim verzijama Adobe programa. Također zahtijeva više podešavanja kako bi ekstenzija ispravno radila u odabranom Adobe programu.

Istraživanjem je utvrđeno da je lakši način izrade početne ekstenzije uz pomoć Visual Studio Code koji danas postaje jedan od najpopularnijih editora koji podržava brojne programske jezike. Također postoje i preporuke za korištenje Brackets ili Sublime Text [6] . Korištenje Visual Studio Code također olakšava web-programeru da ostane u standardnom razvojnom okruženju koje ima dobru podršku za HTML, JavaScript i Node.js. Osim navedenog, u programu Visual Studio Code jednostavno je preuzimanje i dodavanje

[2] https://labs.adobe.com/downloads/extensionbuilder3.html

JavaScript modula preko NPM. Visual Studio Code se može proširiti uz ekstenzije koje olakšavaju izradu početne konfiguracije za CC ekstenziju. CC Extension Builder[3] je ekstenzija za Visual Studio Code koja omogućuje izradu ekstenzija za Adobe Creative Cloud aplikacije.

Nakon instaliranja i omogućavanja alata CC Extension Builder u programu Visual Studio Code izradu početne konfiguracije CC ekstenzije moguće je pokrenuti upotrebom *Command Palette* (*Ctrl+Shift+P*). U *Command Palette* izborniku odabire se *Extension Creator*: *Create a New CC Extension*. U sljedećem koraku potrebno je zadati ID ekstenzije. Sve Adobe ekstenzije moraju imati jedinstveni ID. Adobe preporuča korištenje obrnutog naziva domene. ID ekstenzije upisuje se u *manifest.xml* datoteku ekstenzije [2]. Nakon unošenja ID potrebno je unijeti naziv ekstenzije koji će biti vidljiv u aplikaciji za koju je ekstenzija izrađena. Posljednji korak je odabir predloška (engl. *template*) za izradu ekstenzije što je vidljivo na Slici 3.



Slika 3. *Ponuđeni predlošci za izradu Adobe CC ekstenzije u programu Visual Studio Code korištenjem CC Extension Builder dodatka*

Izvor: Autori

Praćenjem navedenih koraka kreirati će se direktorij na lokalnom disku sa svim potrebnim datotekama. Bitno je napomenuti da aplikacija neće učitati ekstenziju ukoliko ona nije kriptografski potpisna. Kako bi se ekstenzija mogla testirati prije nego što je potpisana potrebno je postaviti *debug flag* u operativnom sustavu. Zastavica se postavlja ovisno o Adobe aplikaciji za koju se izrađuje ekstenzija te se na Windows OS-u u *regeedit* pod HKEY_CURRENT_USER\Software\Adobe\CSXS.9\ postavlja *PlayerDebugMode* na 1.

Novonastali direktorij je potrebno postaviti u jedan od direktorija koje aplikacije pretražuju pri pokretanju. Ekstenzija CC Exstension Builder automatski postavlja izrađeni predložak pod C:\Users\<username>\AppData\Roaming\Adobe\CEP\ extensions\ što je osnovna lokacija za

[3] https://marketplace.visualstudio.com/items?itemName=hennamann.cc-extension-builder

specifičnog korisnika na Windows OS. Ostale moguće lokacije su aplikacijski *exstensions* direktorij (primjerice: C:\Program Files\Adobe\Adobe InDesign CC 2019\Resources\CEP\extensions za InDesign) ili sistemski direktorij za sve Windows OS korisnike (C:\Program Files\Common Files\Adobe\CEP\extensions\). Pri učitavanju aplikacije najprije se pretražuje aplikacijski *extensions* direktorij, zatim sistemski i na kraju korisnički.

Bitno je da se Adobe aplikacija ponovno pokrene pri svakom dodavanju nove ekstenzije kako bi se ona učitala. Ekstenzija se može pokrenuti unutar programa preko *Window\Exstensions* izbornika (napomena: ukoliko ne postoji ni jedna ekstenzija *Extensions* se ne prikazuje u popisu *Window* izbornika).

Prije testiranja aplikacije potrebno je podesiti *manifest.xml* datoteku tako da se pod <*HostList*> elementom podesi oznaka <*Host*> koja omogućuje ekstenziju za specifični program Adobe CC paketa. Slika 4. prikazuje postavku za Adobe InDesign. Na slici je moguće vidjeti i zapis pod *ExtensionManifest* oznakom gdje se nalaze atributi *ExtensionBundleId* koji predstavlja jedinstveni ID paketa ekstenzija kako je zadan prilikom kreacije te *ExtensionBundleName* koji predstavlja naziv ekstenzije kako će se prikazivati u izborniku *Window\Extensions*. Manifest u biti definira paket ekstenzija koje mogu uključivati više od jedne ekstenzije. Atribut *Verison* definira verziju CEP.

manifest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ExtensionManifest Version="9.0" ExtensionBundleId="com.primjer.pozdrav" ExtensionBundleVersion="1.0.0"
        ExtensionBundleName="Prva ekstenzija" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <ExtensionList>
        <Extension Id="com.primjer.pozdrav" Version="1.0.0" />
    </ExtensionList>
    <ExecutionEnvironment>
        <HostList>
            <!-- InDesign -->
            <Host Name="IDSN" Version="[10.0,99.9]" />
```

Slika 4. *Isječak manifest.xml datoteke*

Izvor: Autori

CC ekstenzije zahtijevaju određenu anatomiju. Tipično su sve potrebne datoteke za ekstenziju postavljene u direktorij istog naziva kao i ID ekstenzije. Unutar glavnog direktorija postavljen je CSXS direktorij koji sadrži *manifest.xml* datoteku. U njoj su zapisane putanje do glavne HTML datoteke koja definira sučelje panela (<*MainPath*> oznaka) te putanja do

.jsx datoteke koja sadrži ExtendScript kod koji će se izvršavati u odabranoj Adobe aplikaciji. Slika 5. prikazuje isječak koda s navedenim putanjama.

```
<DispatchInfoList>
    <Extension Id="com.primjer.pozdrav">
        <DispatchInfo >
            <Resources>
            <MainPath>./index.html</MainPath>
            <ScriptPath>./jsx/hostscript.jsx</ScriptPath>
            </Resources>
            <Lifecycle>
                <AutoVisible>true</AutoVisible>
            </Lifecycle>
            <UI>
                <Type>Panel</Type>
                <Menu>Prva ekstenzija</Menu>
                <Geometry>
                    <Size>
                        <Height>500</Height>
                        <Width>500</Width>
                    </Size>
```

Slika 5. *Isječak manifest.xml datoteke koji prikazuje putanje do osnovnih datoteka ekstenzije*

Izvor: Autori

Na slici je također vidljiv odabran tip ekstenzije „*Panel*“ te neke njegove postavke. CEP podržava nekoliko tipova ekstenzija [4]. Tip ekstenzije zadaje se u *manifest.xml*. Mogući tipovi ekstenzija su:

- ***Panel***

  Panel se ponaša kao bilo koja paleta u programu. Može se usidriti, čini *workspace*, može imati padajući meni i ponovo se pokreće pri pokretanju programa (ukoliko je ostao otvoren pri zatvaranju).

- ***Modal Dialog***

  Modal Dialog otvara novi prozor ekstenzije i prisiljava korisnika na interakciju s tim okvirom prije vraćanja kontrole aplikaciji. Tek nakon zatvaranja prozora moguće je raditi s aplikacijom.

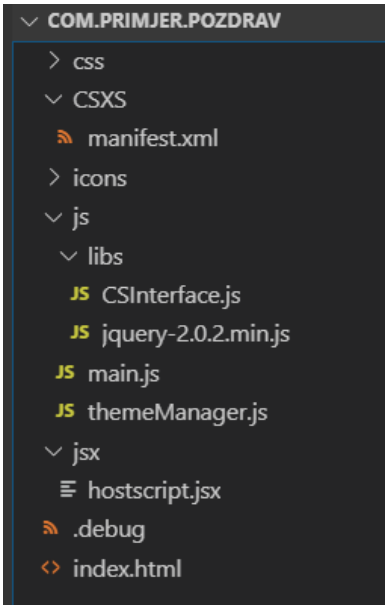
- ***Modeless***

  Modeless Dialog otvara novi prozor ali ne prisiljava isključivu interakciju samo s novootvorenim prozorom.

- ***Custom (od CEP 5.0)***

  Koristi se za nevidljive ekstenzije. Nevidljiva ekstenzija nikad ne postaje vidljiva tijekom cijelog svojeg ciklusa.

Slika 6. prikazuje strukturu i organizaciju datoteka pri izradi CC ekstenzije. HTML datoteka *index.html* opisuje UI[4] ekstenzije. Tipično povezuje JavaScript kod koji omogućuje događaje na stranici te povezivanje s aplikacijom preko CEP. Kad se ekstenzija pokrene, stranica se prikazuje u ugrađenom Chromium pregledniku Adobe aplikacije. Svaka ekstenzija se izvršava u vlastitom radnom okruženju koji se razlikuje od ostalih panela.



Slika 6. *Struktura dokumenata CC ekstenzije*

Izvor: Autori

Js direktorij sadrži potrebne JavaScript datoteke. *Main.js* je glavna datoteka povezana s HTML stranicom i sadrži kod koji će ekstenzija izvršavati. Osim *main.js* koriste se i gotove JavaScript biblioteke kao primjerice *jQuery* i CEP JavaScript biblioteke. CEP JavaScript biblioteke povezuju se s HTML stranicom. Omogućavaju pristup aplikaciji, CEP informacijama i slanje poruka između aplikacije i ekstenzije. Na slici 6. je vidljivo korištenje *CSInterface.js. CSInterface.js* se koristi kako bi se ostvarila komunikacija s Adobe aplikacijom.

[4] Engl. *user interface*- korisničko sučelje

Funkcije u JavaScript biblioteci *CSInterface.js* daju API koji omogućuje komunikaciju V8 JavaScript *engine* s ExtendScript *engine* preko PlugPlug. Komunikacija se ostvaruje korištenjem *evalScript()* metode ili slanjem događaja opisanih u *CSInterface.js*. ExtendScript *engine* komunicira s V8 JavaScript *engine* slanjem događaja preko PlugPlug External Object library. PlugPlug External Object library je ExtendScript biblioteka uključena u ExtendScript DOM[5] te daje API koji omogućuje komunikaciju. Vrijednosti koje se šalju su stringovi. Ukoliko je potrebno prosljeđivati objekte između *engina* potrebno ih je najprije pretvoriti u *string*, što se najčešće čini putem JSON. V8 *engine* podržava JSON a podrška na ExtendScript strani ovisi o verziji i ukoliko nije omogućena potrebno ju je omogućiti putem neke od JSON biblioteka[6].

Na Slici 6. Vidljiva je i *.debug* datoteka koja omogućuje otklanjanje grešaka korisničkog dijela aplikacije u Chromium pregledniku. Otklanjanje pogrešaka je slično kao u slučaju razvoja web-stranica u web-preglednicima korištenjem konzole i ugrađenih alata [7].

4. Proširivanje Adobe InDesign s Node.js

Svaki Adobe program koji podržava HTML ekstenzije sadrži ugrađenu verziju Node.js s kojom paneli mogu biti u interakciji. Budući da Node.js ima moćne API-je i masivan ekosustav paketa trećih strana, izuzetno je koristan pri izradi ekstenzija. Node.js je pokrenut u istom radnom okruženju kao i ekstenzija.

Kako bi se omogućio Node.js u ekstenziji potrebno je promijeniti *manifest.xml* datoteku dodavanjem linijskih naredbi (engl. *Chromium Command Line Switches*). Postoje brojne naredbe koje mijenjanju ponašanje Chromium preglednika [8]. Sve postavke nisu podržane u CEF okruženju [9] te postoje postavke koje je definirao Adobe. Kako bi se u ekstenziji omogućio Node.js potrebno je dodati *CEFCommandLine* oznaku s oznakom *Parameter* koja sadrži vrijednost --*enable-nodejs* što je prikazano na Slici 7.

[5] za starije verzije potrebno ju je preuzeti na https://github.com/Adobe-CEP/CEP-Resources/releases/tag/1 i uključiti u projekt

[6] primjerice *json2.js* koji se može preuzeti s https://github.com/douglascrockford/JSON-js

```
<DispatchInfoList>
    <Extension Id="com.primjer.pozdrav">
        <DispatchInfo >
            <Resources>
            <MainPath>./index.html</MainPath>
            <ScriptPath>./jsx/hostscript.jsx</ScriptPath>
            <CEFCommandLine>
                    <Parameter>--enable-nodejs</Parameter>
            </CEFCommandLine>
            </Resources>
```

Slika 7. *Omogućavanje Node.js u ekstenziji*

Izvor: Autori

Osnovne biblioteke i metode Node.js mogu se pozivati direktno iz koda. Slika 8. prikazuje korištenje *fs (*engl. *File System)* modula za čitanje sadržaja direktorija ekstenzije. Omogućavanjem Node.js direktno se mogu koristiti njegovi ugrađeni moduli.

```
<button onclick="citaj()">Ispiši datoteke</button>
<script>
  function citaj() {
    var cs = new CSInterface();
    var putanja = cs.getSystemPath(SystemPath.EXTENSION);
    const fs = require("fs");
    fs.readdir(putanja, (err, datoteke) => {
      datoteke.forEach(dat => {
        console.log(dat);
      });
    });
  }
</script>
```

Slika 8. *Korištenje fs modula*

Izvor: Autori

Osim osnovnih modula moguće je preuzimati gotove module preko NPM (engl. *Node Package Manager*) ili kreirati vlastite. Instalacijom Node.js[7] dobiva se i NPM CLI (engl. *Command Line Client*) preko kojeg je moguće preuzimati i instalirati razne NPM dodatke.

Slika 9. Prikazuje korištenje *open* modula[8] iz NPM repozitorija koji omogućuje otvaranje URL adresa u pregledniku.

[7] https://nodejs.org/en/download/
[8] https://www.npmjs.com/package/open

```
<button onclick="otvoriHTML()">Posjeti stranicu</button>
<script>
  function otvoriHTML() {
    var open = require("open");
    open("http://www.mev.hr");
  }
</script>
```

Slika 9. *Korištenje open modula*

Izvor: Autori

5. Zaključak

Korištenjem CEP moguće je koristiti web-tehnologije kako bi se prilagodili Adobe alati te povezali s ostalim servisima i platformama. Mogućnost prilagođavanja gotovih alata donosi brojne prednosti. CC ekstenzijama moguće je povećati produktivnost automatizacijom procesa koji su jedinstveni u pojedinim radnim okruženjima ili u komercijalnim programima ne postoje opcije automatizacije pojedinih zadataka. Korištenjem CEP sustava moguće je integrirati Creative Cloud aplikacije s servisima i tehnologijama za koje aplikacije nikad nisu bile zamišljene.

Izrada korisničkoga sučelja u CC ekstenzijama podudara se s razvojem web-stranica. Ne postoje alati i biblioteke koji se zahtijevaju te se mogu koristiti standardni alati koji se koriste pri izradi web-stranica. Postoje određene razlike u izvršavanju napisanoga koda što je i logično jer se ekstenzije izvršavaju unutar Adobe aplikacije a ne web-preglednika.

Korištenjem Node.js i NPM modula moguće je uvelike proširiti mogućnosti Adobe programa. U daljnjem istraživanju potrebno je proučiti prednosti i nedostatke koji nastaju korištenjem web-tehnologija u organiziranju i automatizaciji zadataka digitalne grafičke pripreme.

Literatura

[1] Javascript, a new language for the printing industry; Agile Streams, Dostupno na: https://www.agilestreams.io/en/javascript-a-new-language-for-the-printing-industry/. [Pokušaj pristupa 04.06. 2019.]

[2] USING THE ADOBE EXTENSION SDK; 2014., Dostupno na: http://wwwimages.adobe.com/content/dam/acom/en/devnet/cs-extension-builder/pdfs/CC_Extension_SDK.pdf. [Pokušaj pristupa 28.08.2019.]

[3] CEP for the InDesign Developer; Dostupno na: https://github.com/Adobe-CEP/CEP-

Resources/blob/master/Documentation/CEP%20for%20InDesign%20Developers.pdf. [Pokušaj pristupa 04. 06. 2019.]

[4] Galeran, J.; CEP 9 HTML Extension Cookbook; 27. 11.2018., Dostupno na: https://github.com/Adobe-CEP/CEP-Resources/blob/master/CEP_9.x/Documentation/CEP%209.0%20HTML%20Extension%20Cookbook.md. [Pokušaj pristupa 07. 06. 2019.]

[5] Brekalo, S; Optimizacija digitalne grafičke pripreme korištenjem skriptnih tehnologija, doktorska disertacija, 2015.

[6] Kvern, O.; Publishing Silicon, 14. 06. 2014., Dostupno na: https://www.siliconpublishing.com/blog/2014/07/change-as-a-way-of-life-cc-extensions-and-the-third-party-indesign-developer. [Pokušaj pristupa 02. 09. 2019.].

[7] Client-side debugging; 08. 05. 2018., Dostupno na: https://github.com/Adobe-CEP/Getting-Started-guides/tree/master/Client-side%20Debugging#debugging-in-chrome. [Pokušaj pristupa 02. 09.2019.].

[8] Beverloo, P.; List of Chromium Command Line Switches; 12. 08. 2019. Dostupno na: https://peter.sh/experiments/chromium-command-line-switches/. [Pokušaj pristupa 02. 09. 2019.].

[9] Adobe Exchange and more…, Introducing CEP 5; 25. 04. 2014., Dostupno na: https://blogs.adobe.com/cssdk/2014/04/introducing-cep-5.html. [Pokušaj pristupa 02. 09. 2019.].