Primjena SVG animacija u web dizajnu

Tomorad, Filip

Master's thesis / Diplomski rad

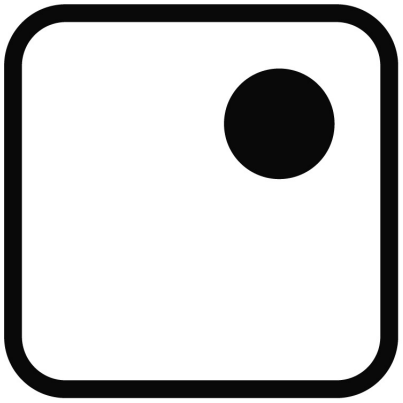
2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Graphic Arts / Sveučilište u Zagrebu, Grafički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:216:508710

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-01-26**



Repository / Repozitorij:

Faculty of Graphic Arts Repository

SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

FILIP TOMORAD

PRIMJENA SVG ANIMACIJA U WEB DIZAJNU

DIPLOMSKI RAD

Zagreb, 2022.

Sveučilište u Zagrebu
Grafički fakultet

FILIP TOMORAD

PRIMJENA SVG ANIMACIJA U WEB DIZAJNU

DIPLOMSKI RAD

Mentor:
Prof. dr. sc. Jesenka Pibernik

Student:
Filip Tomorad

Zagreb, 2022.

Rješenje o odobrenju teme završnog rada

SAŽETAK

Velika raznolikost uređaja putem kojih korisnici danas konzumiraju sadržaj navela je dizajnere i *developere* da stranice razvijaju uporabom fleksibilnih elemenata koji su sposobni mijenjati poziciju, veličinu, boju, i ostale vizualne atribute. Vektorska grafika ovdje pruža jasne prednosti nad rasterskom. Osim neograničene rezolucije i male veličine datoteka, vektori omogućuju razlamanje slike na njene dijelove, kojima se može pojedinačno manipulirati. Takvu je izmjenu atributa i oblika vektora moguće provesti postepeno, čime nastaje animacija. Standard koji omogućuje interakciju web-jezika s vektorskom grafikom je jezik SVG. Cilj je ovoga rada pružiti pregled mogućnosti koje SVG tehnologija pruža u kontekstu web-dizajna, s fokusom na uporabu animacije, te opisati načine na koje se korištenjem popratnih jezika SVG elementi mogu pretvoriti u pokretne grafike.

Ključne riječi: *Vektorska grafika, SVG, web, animacija*

ABSTRACT

The diversity of devices with which today's users consume content requires designers and developers to create websites using flexible elements able to change position, size, colour, and other visual attributes. Vector graphics offer clear advantages over raster ones in this area. Besides their limitless resolution and small file size, vectors allow for manipulation of specific objects within the image. This manipulation can be done gradually, creating animation. The standard which allows for the interactivity of web-languages and vector graphics is the language SVG. The aim of this paper is to give insight into the possibilities that SVG technology offers in the context of web-design, with a focus on use of animation, and to summarize the methods of using other languages to turn SVG elements into motion graphics.

Keywords: *Vector graphics, SVG, web, animation*

SADRŽAJ

1. UVOD 1

2. POVIJEST SVG STANDARDA 4

3. PREGLED FUNKCIONALNOSTI SVG-A 10

3.1 Koncepti i osnovne funkcije 11

3.2 Vektorski oblici 16

3.3 Vizualni prikaz vektora 18

3.4 Ostale funkcije 21

4. METODE ANIMACIJE SVG ELEMENATA NA WEB-STRANICAMA 23

4.1 SMIL 26

4.2 CSS 29

4.3 JavaScript 32

4.4 Eksterni *libraryji* 33

4.5 Web Animations API 35

4.6 Alati s grafičkim sučeljem 37

5. PRIMJENE SVG ANIMACIJA U WEB-DIZAJNU 38

5.1 Animirane ilustracije i pozadinske animacije 40

5.2 UX mikro-interakcije i naglašavanje sadržaja 44

5.3 Animacije prijelaza stranice 47

5.4 Animirane *hero* sekcije 49

5.5 Vizualizacija podataka 51

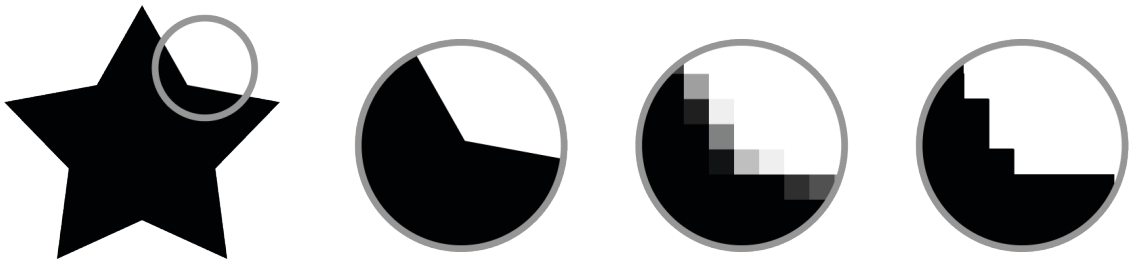
6. ALTERNATIVE SVG ANIMACIJI 53

7. OSVRT I ZAKLJUČAK 55

1. UVOD

Pojam “vektorska grafika” u računalnoj grafici označava grafiku sastavljenu od matematički definiranih krivulja. On je u kontrastu s tzv. “rasterskom grafikom”, onom sastavljenom od mreže piksela, od kojih svaki sadrži informacije o svojoj boji u jednom ili više kanala.[1] Najveća razlika između ovih načina zapisa grafike je rezolucija: dok su rasterske slike (poznate i kao “bitmape”) ograničene dimenzijama spomenute mreže, vektori nemaju maksimalnu rezoluciju, što znači da se vektorske slike mogu neograničeno povećavati bez pojave šuma. Jedna od svakodnevnih primjena vektorske grafike je zapis fontova.

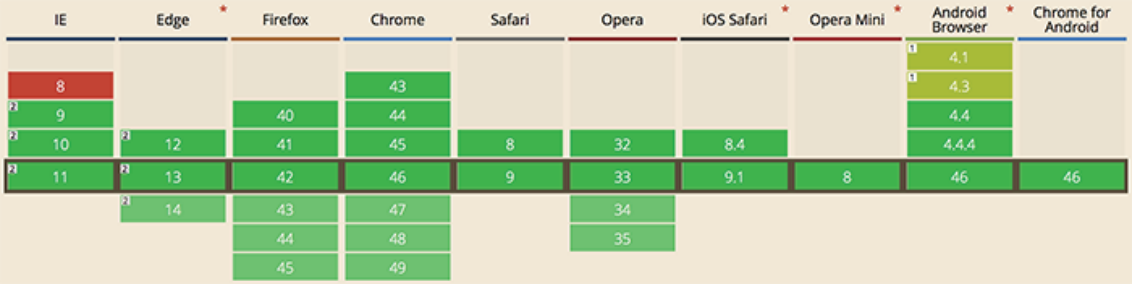
SVG je otvoreni format za zapis vektorskih grafika. Među ostalim je takvim formatima značajan zbog statusa web-standarda, što znači da je čitljiv velikoj većini modernih web-preglednika, i da je sposoban uspostaviti interakciju s ostalim web-jezicima koji sastavljaju web-stranice. SVG datoteke (“.svg”) sastoje se od tekstualnih uputa za iscrtavanje vektora, što ih kod prikaza grafika baziranih na geometrijskim oblicima čini mnogo kompaktnijim od rasterskih verzija istih. Ovo utječe na relativnu “težinu” datoteke, tj. njenu veličinu u bajtima. Na primjer, za prikaz običnog crnog kruga vektorskim zapisom potrebno je samo nekoliko redaka koda koji definiraju njegov polumjer i poziciju centralne točke u koordinatnom sustavu, te boju. Taj bi isti krug u jedno-kanalnom rasterskom zapisu morao sadržavati informaciju o boji svakog piksela unutar definirane mreže. Ako kao primjer uzmemo sliku malih dimenzija 100x100 piksela, ona će uključivati 10,000 informacija o boji piksela, od kojih će velika većina biti ili potpuno crni, ili potpuno bijeli, a manji broj će činiti nijanse sive nastale zbog tzv. *anti-aliasinga* (Slika 1). Tako su za mnoge vrste grafika vektorski formati mnogo kompaktniji. Naravno, kod npr. kompleksnih ilustracija, veličina vektorske datoteke teoretski je neograničena zbog mogućnosti za beskonačnim povećanjem količine detalja, dok rasterska datoteka ima maksimalnu veličinu određenu rezolucijom. Neke su slike stoga prikladnije vektorskom zapisu od drugih. Dobri kandidati za vizualni prikaz putem vektora su tipografija, ikone i piktogrami, logotipi, uzorci, i jednostavne ilustracije.



Slika 1: Primjer povećanja slike vektorskog formata, te rasterskog formata sa i bez *anti-aliasinga*

Prednosti vektorskih formata na webu očite su u mnogim svakodnevnim primjenama. Težnja modernih web-stranica prema responzivnosti (neovisnosti o formatu i orijentaciji ekrana) od grafičkih elemenata zahtijeva fleksibilnost i prilagodljivost skaliranju, dok težnja prema taktilnosti, fluidnosti, i naglašavanju mikro-interakcija zahtijeva kompatibilnost grafika s raznim tranzicijama, animacijama, i ostalim promjenama stanja. Ovdje SVG dolazi u doticaj s drugim jezicima koji su također dio web-standarda: HTML-om, jezikom za definiciju hijerarhije elemenata na stranici, CSS-om, jezikom za opisivanje vizualnih stilova tih elemenata, i JavaScriptom, univerzalnim programskim jezikom. Njihovom kombinacijom moguć je dizajn kompleksnih, ali laganih i interaktivnih animacija koje funkcioniraju u bilo kojem web-pregledniku. Animacije nastaju postepenom, vremenski određenom promjenom vizualnih atributa vektora, kao što su pozicija, veličina, ili boja. Izrađuju se primarno pisanjem koda, no postoje i digitalni alati s grafičkim sučeljem koji pružaju mogućnost zapisa animacije na način razumljiv web-preglednicima.

Među web-tehnologijama, SVG je do relativno nedavno bio slabo poznat, iako postoji već preko dva desetljeća.[2] Zbog nedostatka podrške među tada popularnim web-preglednicima (uglavnom motiviranog financijskim i tržišnim ciljevima), dugo je vrijeme bio zasjenjen drugim formatima. Ipak, pojava mobilnih uređaja izazvala je značajnu promjenu u paradigmi ustaljenih praksi u poljima web-dizajna i -*developmenta*. Povećanje popularnosti otvorenih web-standarda naspram tehnologija u vlasništvu privatnih tvrtki (kao što su Adobe i Microsoft) ponovno je dovelo do spominjanja SVG-a kao aktualne i perspektivne tehnologije koja zaslužuje biti spomenuta u istom dahu kao CSS i HTML. SVG funkcionalnost trenutno podržavaju svi značajni preglednici (Tablica 1).



Tablica 1: Podrška SVG standarda među web preglednicima i njihovim verzijama. [2]

Cilj je ovoga rada stvoriti sveobuhvatan katalog mogućnosti koje SVG tehnologija pruža na webu, s fokusom na njenu ulogu u animaciji. Rad će sadržavati pregled glavnih podržanih metoda izrade animacija, opis prednosti i nedostataka pojedine metode, te popis uloga koje animacije u sklopu web-stranice mogu imati. Prvi dio rada činiti će kratki uvod u povijest SVG jezika i njegovu funkcionalnost, te u popratne web-jezike koji pružaju mogućnost animacije SVG vektorskih elemenata. Ti će jezici biti opisivani s naglaskom na isticanje njihovih mana i vrlina u usporedbi s konkurencijom. Drugi će dio rada pružiti uvid u konkretne primjene SVG grafike određene ulogom unutar web-stranice, uz stvarne primjere. U svakom će podpoglavlju biti opisana jedna od tih uloga, i zadatak koji ona u sklopu te uloge ispunjava. Primjeri će sadržavati one javno dostupne i one iz literature, koji će se koristiti za uvid u raspon ideja koje je u SVG-u moguće realizirati (pogotovo onih čija je funkcionalnost vrlo specifična ili inovativna), te one autorske, izrađene za potrebe ovog rada, koji će biti korišteni za detaljniju analizu koda, kako bi se dublje opisao način na koji SVG animacije funkcioniraju. Kroz autorske će primjere biti nastojano upotrijebiti što više spomenutih metoda animacije, te definirati razloge za njihovu uporabu u pojedinom kontekstu. Cilj je kroz ukupan sadržaj rada pružiti uvid u sve alate koje web-dizajneri poznavanjem SVG-a mogu imati na raspolaganju. Web-inženjer Max Dunn piše: *“Nije uvijek lako definirati najbolje prakse kad se ista stvar može postići na najmanje tri načina.”*[2]

2. POVIJEST SVG STANDARDA

Kratica SVG označava pleonastičnu sintagmu “scalable vector graphics”, tj. “vektorska grafika koja se može skalirati”. Nastala je 1998., kao naziv za projekt World Wide Web Consortiuma (poznatog i kao W3C), glavne međunarodne organizacije za standardizaciju weba i web-jezika. Sredinom devedesetih godina prošlog stoljeća, zajednica ranih web-*developera* prepoznala je potrebu za tehnologijom koja će u uporabama za web naslijediti tada već preko deset godina stari PostScript, Adobeov jezik za prikaz vektorskih grafika, primarno namijenjen i originalno razvijen za uporabu u tisku. Nedostatci PostScripta koji su potakli W3C na potragu za alternativom bili su primarno veličine dokumenata usporedive s rasterskim verzijama istih, te oslanjanje na nefleksibilne definicije dimenzija stranica i mjere u stvarnim jedinicama.[3] Ideja iza ovog projekta definirana je već 1996., u dokumentu “W3C Scalable Graphics Requirements”, gdje su zadani okvirni kriteriji koje će novi vektorski jezik morati zadovoljavati.[4] Oni su podijeljeni u šest kategorija:

“Otvorene specifikacije”, što pokriva javnu dostupnost dokumentacije, prilagodljivost novim tehnologijama, te široku kompatibilnost.

“Grafičke mogućnosti”, gdje su navedeni neki od traženih načina vizualne izvedbe vektora, poput krivulja, fontova, slojeva, maski, itd.

“Interakcija”, koja uključuje interakciju korisnika s grafikom, kao i interakciju jezika s drugim jezicima i sa vanjskim datotekama.

“Metadata”, “metapodatci”, tj. mogućnost umetanja podataka o autoru i specifikacijama dokumenta.

I konačno, kategorije **“Alati za izradu”** i **“Čitljivost”** nalažu da bi u predloženi format trebalo moći zapisati grafike izrađene u najpopularnijim alatima za dizajn vektora, te da bi taj format trebao biti prepoznatljiv najpopularnijim web-preglednicima i čitačima grafike.

Nekoliko godina nakon, 1998., W3C je imao pet zaprimljenih prijedloga za jezik koji odgovara ovim specifikacijama, a predlagatelji koji su potpisivali kandidate uključivali su Adobe, Autodesk, Microsoft, IBM, Xerox i Boeing.[5] W3C je tada

pod nazivom “SVG Working Group” oformio tijelo koje će odlučiti o potencijalnom usvajanju nekog od predloženih standarda. No, ta je komisija, umjesto usvajanja ijednog od prijedloga, odlučila razviti vlastiti standard inspiriran zaključcima i poukama izvedenima iz analize prijedloga, ali ne i direktno temeljen na jednom pojedinom predloženom jeziku. Baza za taj jezik biti će tada relativno nedavno usvojeni XML (Extensible Markup Language), *“text-based”* sustav za strukturirano prikazivanje podataka i njihovih atributa. XML, a tako i njegovi derivati, dijele podatke na dva tipa: sadržaj (*content*), i tzv. *markup*, atribute koji opisuju pojedini blok sadržaja i njegovu poziciju unutar hijerarhije (“document object model”, ili DOM).[6] Takva sintaksa među web-jezicima omogućuje jednostavnu razmjenu ne samo informacija, već i meta-informacija o svakoj pojedinoj informaciji i ukupnoj strukturi dokumenta. U .svg dokumentu sadržaj može biti tekst, rasterska slika, ili matematička definicija vektora, dok *markup* uglavnom opisuje vizualne attribute, kao što su, između ostalog, debljina linije, boje, dimenzije, rotacija, i filteri. Naravno, mogućnosti ove hijerarhije su mnogo šire, što će biti dodatno objašnjeno u idućem poglavlju.

SVG 1.0 usvojen je 5. rujna 2001. službenom preporukom W3C-a.[7] U tadašnjoj zajednici prihvaćen je dosta entuzijastično; portal XML.com taj je tjedan objavio kolumnu s recenzijom novoobjavljene dokumentacije, čiji drugi paragraf glasi: *“Zajedno s W3C XML shemom, SVG je jedna od najbitnijih tehnologija koje su ove godine izašle iz W3C-a. Već je dugo očekivana — prva javna skica je objavljena prije više od dvije i po godine. Iako su mnogi nestrpljivo iščekivali konačnu preporuku, ovaj oduži period sazrijevanja je donio puno koristi za kvalitetu specifikacije SVG-a i broj podržanih implementacija.”*[8] Autor je također opisao i poziciju SVG-a unutar tadašnje sfere aktualnih web-tehnologija. Navodi kako je “najočitija” konkurencija Flash, tada proizvod Macromedije, *“realno jedina ukorijenjena tehnologija za vektorsku ilustraciju i animaciju [na webu]”*, i kako će SVG-u biti teško doseći *“mjesto koje Flash ima među alatima web dizajnera”*. Piše i kako skalabilnost grafike *“ne nudi mnogo prednosti na ekranu osobnog računala, gdje su rezolucije najčešće niske”*. Kao što danas znamo, Macromediju će 2005. kupiti Adobe, koji će petnaest godina kasnije Flash platformu i službeno ugasiti. No, Flash će do tog trenutka provesti skoro dva desetljeća dominirajući nišom koju je dijelio s SVG-om.[9] Jedan je od razloga za to njegova podrška reprodukcije

video sadržaja unutar svog HTML elementa, funkcija koju SVG nije imao. Sam je YouTube godinama ovisio o “Flash Player” ekstenziji za web-preglednike, kao što su ovisile i mnoge web-stranice s kompleksnim interaktivnim grafikama, te većina video-igara igranih u *browseru*. Ovo je značilo da je prije pristupa sadržaju na računalo bilo potrebno instalirati program koji će moći interpretirati kod neshvatljiv običnom pregledniku. Dostupnost sadržaja stoga je ovisila o tzv. *third-party* ekstenzijama izvan web-standarda, poput one koju je izdavao Adobe. SVG je u svojoj fazi rane implementacije naišao na istu prepreku; iako je njegova podrška u grafičkim programima bila dobra još i prije službene W3C preporuke, podrška u web-preglednicima bila je raznolike kvalitete. Stoga su mnogi korisnici bili primorani koristiti nestandardne ekstenzije za pristup sadržaju koji je službeno već bio unutar standarda. Ovaj se je problem najdulje održao kod Microsoftovog Internet Explorera, koji SVG uopće nije podržavao do svoje devete verzije 2011., a čiji udio na tržištu nije uvijek bio zanemariv.

No, taj je problem za SVG od početka bio premostiv. Povećanjem rezolucija i brzina ekrana osobnih računala, sve većim usvajanjem W3C standarda među web-preglednicima, te pojavom mobilnih *touch* uređaja, Flash je sve rjeđe bio najbolji ili najprikladniji alat u paleti web-dizajnera. Dolaskom HTML5 standarda CSS je postao novi “prvi izbor” za animaciju na webu, donijevši sa sobom i mnogo bolje performanse. Usprkos podosta užem spektru funkcionalnosti, uspio je preuzeti veliki udio primjena na kojima je dotad prevladavao Flash, pogotovo na manje kompleksnim aplikacijama poput interaktivnih menija i *video-playera*. Sam će Steve Jobs 2010. napisati otvoreno pismo pod naslovom “Thoughts on Flash”, u kojem navodi šest razloga zbog kojih Apple mobilni uređaji ne podržavaju Flash tehnologiju.[10] Među njima navodi: *“Flash je nastao za vrijeme PC ere — za osobna računala s mišem. [...] Mobilna era je vrijeme niske potrošnje energije, sučelja na dodir i otvorenih web-standarda — sve su to područja u kojima Flash manjka.” “[...] mnoge Flash stranice ovise menijima i elementima koji se prikazuju ulaskom miša u određeno područje. [...] Ako* developeri *moraju promijeniti svoje Flash stranice [kako bi bile kompatibilne s* touch *uređajima], zašto ne bi koristili moderne tehnologije poput HTML-a 5, CSS-a i JavaScripta?” “Iako su operacijski sustavi za [Apple uređaje] zatvoreni, čvrsto vjerujemo da bi svi standardi vezani uz web trebali biti otvoreni.”* Teško je ne primijetiti da su navedene mane Flasha

također i vrline SVG-a. Gledano retrospektivno, moglo bi se reći da je SVG, nažalost, bio "ispred svog vremena". Usprkos početnom entuzijazmu, u svojim ranim danima na webu nije zaživio primarno zbog poteškoća s ulaskom na tržište u kojemu je Flash već bio ustaljeni standard. Ipak, odlaskom Flasha, došlo je do praznine koju CSS ne može popuniti sam. Mnogi su današnji *developeri* i dizajneri tako na SVG naišli kao na alat koju nisu znali da imaju, a koji im uvelike proširuje spektar mogućnosti; dijelom zaboravljeni web-standard koji moderni preglednici već odavno podržavaju, ali čije su primjene slabo razvikane.

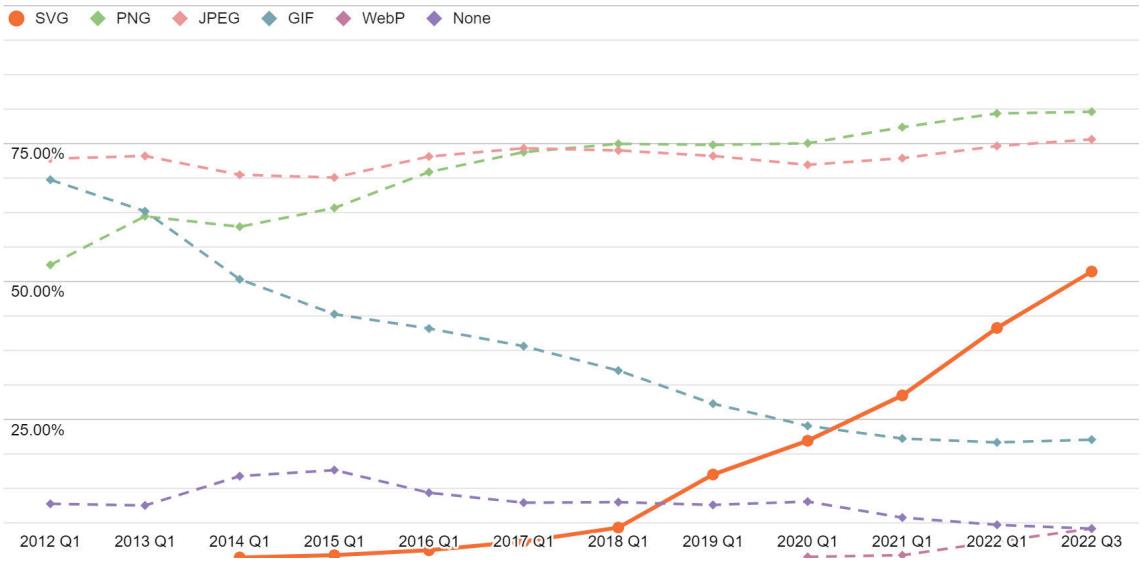
U praksi, današnji je SVG većinski identičan verziji 1.0 usvojenoj 2001.; iako je od tada usvojeno nekoliko novih verzija dokumentacije, one su se fokusirale na povećanje jasnoće teksta, više nego na promjenu bilo kakve funkcionalnosti. SVG kojeg mi danas poznajemo, i na kojem se ovaj rad bazira, je verzija "1.1 (Drugo izdanje)" iz 2011. godine.[11] Uz prvo izdanje SVG-a 1.1 uključene su i dvije pojednostavljene verzije jezika, "SVG Tiny" i "SVG Basic", namijenjene mobilnim telefonima i slanju putem MMS-a. 2003., kada su ti jezici izašli, imali su određene prednosti u kontekstu grafičkih sposobnosti ranih mobilnih uređaja, no u današnjem je web-dizajnu njihova relevantnost vrlo niska. Što se tiče nadolazećih verzija, prijedlog W3C-u pod nazivom "SVG 2" postoji još od 2013. godine,[12] no zasad nije doživio veliku popularnost među *developerima* web-preglednika, a ni među web-*developerima*.[13] Nedostatak interesa za njegovu implementaciju ostavio je SVG 2 u sivoj zoni, gdje je dokumentacija odavno napisana, testirana, i potpisana, ali ne i podržana od strane najpopularnijih web-preglednika. Noviteti naspram SVG-a 1.1 uključuju podosta ambiciozne dodatke poput *gradient mesheva*, video-tekstura, promjene redoslijeda slojeva (*z-index*), i novih tekstualnih efekata. Osim toga, jedan je dio funkcionalnosti SVG-a "predan" popularnijem CSS-u u područjima gdje se one preklapaju.[14] Autori ove dokumentacije, članovi SVG Working Groupa, navode slabu potporu velikih korporacija kao glavni razlog za stagnaciju ovog prijedloga usprkos njegovoj kvaliteti. Urednik dokumentacije i zaposlenik Inkscapea (*open-source* grafičkog programa koji podržava SVG 2 tehnologiju) Tavmjong Bah 2016. piše o poteškoćama pri nalaženju "saveznika" među grafičkim divovima poput Adobea i Canona, istih onih koji su svojedobno potpisali SVG 1.0, te o inzistiranju velikih proizvođača web-preglednika na ograničenju obujma SVG-a 2 isključivo na "popravljanje" problema koji postoje u

SVG-u 1.1, bez implementacije novih (već funkcionalnih i testiranih) *featurea*.[15]

Budućnost SVG standarda zasad je neizvjesna. Relativna zastarjelost sintakse jezika i velika količina atavizama preostalih iz razdoblja ranog weba pretvorila ga je u format koji se uglavnom koristi za zapis vektora izrađenih putem grafičkih programa, a rijetko kao jezik u kojem se grafike izrađuju pisanjem i uređivanjem koda. Iako je doživio porast u popularnosti zbog sve veće potrebe za responzivnim i brzim web-stranicama, na njima se uglavnom nalazi kao statična slika opisana mnogo popularnijim CSS-om, koji je SVG-u kroz godine *"ukrao najzanimljivije featuree"*[15]. Ipak, neki smatraju da je razlog za sporo usvajanje novih funkcionalnosti nedostatak svijesti o postojećima. Web-*developer* i -dizajner Chris Coyier piše: *"Trenutno postoji solidna razina interesa za SVG među developerima, no možda ne isto toliko vikanja o njegovim nedostatcima. U mom iskustvu,* developeri *uglavnom još uvijek pokušavaju razumjeti ono što već postoji."*[16] Činjenica je da je SVG standard, usprkos poodmakloj dobi, još uvijek podosta snažan. Bez obzira na neizvjesnosti o usvajanju noviteta, sadašnja funkcionalnost SVG-a iznimno je dobro podržana, kao i komunikacija SVG elemenata s CSS-om i JavaScriptom. XML baza daje mu mogućnost da unatoč ponekad odavno napuštenoj sintaksi ostane čitljiv web-jezicima i optičkim čitačima. No, usprkos dugoj povijesti, i tome što su moderni uređaji već duže vrijeme sposobni fluidno prikazati SVG slike (bez potrebe za korištenjem pojednostavljenih verzija jezika), čini se da je potencijal ove tehnologije prepoznat tek vrlo nedavno.[17] SVG grafiku trenutno koristi 51.8% stranica[18], što je veoma veliki porast u usporedbi s 1.4% zabilježenih 2016., godine kada je usvojen prvi prijedlog SVG-a 2. Popularnost standarda nesumnjivo raste, a s time i količina inovativnih implementacija istog. Jedan od glavnih izvora inovacija vezanih uz uporabu SVG-a u web-dizajnu, kao i tema ovoga rada, su animacije vektorskih SVG elemenata potpomognute web-jezicima koji će biti dublje istraženi u sljedećem poglavlju. Također, u periodu rasta popularnosti SVG-a vidljiv je jednako značajan pad u popularnosti GIF tehnologije, što može sugerirati da se ona u svojim najčešćim primjenama počinje zamjenjivati SVG-om. SVG tako u kontekstu modernizacije web-grafika nudi zamjenu za bivše GIF primjene, kao i za bivše Flash primjene.

Izvan sfere web-dizajna, povećanje popularnosti SVG-a može nagoviještati i odmak od Adobe Illustratora kao standarda za dizajn vektora. Naime, prelaskom

iz Adobeovog “.ai” formata u .svg određeni se efekti mogu izgubiti, ili mogu biti rastrirani, zbog čega su timovi dizajnera uglavnom primorani koristiti isti alat kako bi mogli naknadno pristupiti postavkama tih efekata. No, usvajanjem SVG-a 2, neki bi od tih efekata (poput *gradient mesha*) mogli postati dostupni bez potrebe za izlaskom izvan prihvaćenog standarda za zapis web grafike, što bi povećalo fleksibilnost dizajnera pri odabiru alata. Također, takvi bi elementi bili kompaktni i univerzalno čitljivi, što bi omogućilo korištenje kompleksnijih vektorskih grafika u web-dizajnu bez potrebe za njihovim prethodnim rastriranjem. Na ovaj bi način noviteti u web-jezicima mogli utjecati na promjene u polju vektorske ilustracije.



Graf 1: Udio popularnih tehnologija zapisa slike u ukupnim web-stranicama; SVG je istaknut. [18]

3. PREGLED FUNKCIONALNOSTI SVG-A

U ovom je poglavlju sadržan sažetak službene i aktualne SVG 1.1 dokumentacije**[11]** po sekcijama, s pojašnjenjem glavnih koncepata i metoda uporabe, te s opisom generalne sintakse jezika. Kod sekcija koje definiraju specifične naredbe, elemente, atribute, i ostale *featuree* SVG jezika biti će priložen primjer u obliku koda, uz objašnjenje. Cilj ovoga poglavlja nije da služi kao opširno uputstvo za korištenje SVG-a, već da čitatelju pruži uvid u njegov opseg, mogućnosti koje daje, i njegovu "gramatiku", te način na koji surađuje s drugim jezicima. Čitatelj koji nije upoznat sa strukturom SVG dokumenta (ili generalnom XML strukturom) kroz ovaj dio rada može steći podlogu koja će mu omogućiti da razumije načine na koje će metode animacije u kasnijim poglavljima funkcionirati. Dokumentacija također uključuje i apendikse od "A" do "Q", koji sadrže sporedne definicije funkcija, izvore, savjete, i kazala pojmova; oni su, kao i druge suviše specifične informacije, iz ovog poglavlja rada izostavljeni.

3.1 Koncepti i osnovne funkcije

Prvih sedam sekcija dokumentacije daju informacije o širim idejama na kojima je SVG standard građen, te o načinu na koji je SVG datoteka sastavljena. U tim je poglavljima definiran prostor u kojemu će se grafički elementi iscrtavati, i sintaksa potrebna da bi se njihovo postojanje u tom prostoru opisalo. To su poglavlja:

Uvod (“Introduction”)
Uvodna rečenica glasi: *“SVG je jezik za opisivanje dvodimenzionalne grafike u XML-u.”* Ovo poglavlje sadrži sažetak *featurea* koje SVG nudi i njegovog sastava, pregled kompatibilnosti s drugim jezicima, i definicije terminologije koja će se koristiti u ostatku dokumentacije. Ovdje je definiran i sufiks datoteke “.svg” , te tri vrste grafike koju ona može sadržavati: vektore, slike, i tekst.

Koncepti (“Concepts”)
Ovo poglavlje potanje opisuje što je vektorska grafika, i koje su njene prednosti na webu. Daje pregled vektorskih oblika koje SVG nudi, koji će kasnije biti detaljno opisani u zasebnim poglavljima. Ovdje je definirano i pet načina na koje se SVG može koristiti u sklopu web-stranice. *“Samostalna SVG web-stranica”* je metoda u kojoj SVG uopće nije dio HTML stranice, već u kojoj web-preglednik otvara čisti SVG dokument. *“*Embedanje *putem reference”* je metoda gdje HTML stranica kao objekt navodi referencu na vanjsku .svg datoteku koja sadrži željeni SVG kod, i koja se tada učitava na namijenjeno mjesto. Metoda *“Referenca CSS ili XLS svojstva”* je slična, no .svg datoteku u tom slučaju ne poziva HTML, već CSS ili XLS. Kod metode *“*Embedanje *u redu”* (“*inline*”), SVG kod je direktno upisan u tekst .html datoteke, unutar koje čini element. Konačno, metoda *“Vanjska poveznica”* uopće ne ugrađuje .svg datoteku u stranicu, već samo nudi poveznicu putem koje se ona može samostalno otvoriti, ili preuzeti. U praksi su od ovih metoda glavne *inline* embedanje i poziv referencom, budući da uključuju komunikaciju SVG-a s drugim jezicima. (Glagol “embed” znači nešto poput “ugraditi” ili “usaditi”.)

Model za iscrtavanje (“Rendering Model”)
Ovdje je ukratko objašnjen sustav putem kojeg se SVG elementi iscrtavaju na ekranu. Element koji je zadnji definiran unutar .svg datoteke je “na vrhu”.

Osnovni tipovi podataka i sučelja (“Basic Data Types and Interfaces”)
Ovo poglavlje sadrži popis najčešćih vrsta podataka koje SVG prepoznaje, kao što su vrijeme, kut, boja, ili broj, te jedinice putem kojih određeni podatci mogu biti izraženi. Ovdje je sadržan i popis naziva boja koje SVG podržava (npr. “orange”, “salmon”, “hotpink”) uz RGB vrijednosti koje im pripadaju.

Struktura dokumenta (“Document Structure”)
U ovom su poglavlju definirane osnove sintakse SVG-a. SVG dokument sastoji se od elemenata koje određuje ključna riječ, tzv. *tag*, unutar zagrada. Ovo je primjer pojednostavljenog SVG koda koji sadrži dva kruga i tekst “Primjer”:

```
<svg>
      <desc>Dva kruga i tekst</desc>
      <g>
            <circle id="krug1" />
            <circle id="krug2" />
      <g>
      <text font-family="Futura">Primjer</text>
</svg>
```

Kao što je vidljivo, hijerarhija je definirana na način da neki elementi sadrže druge. Početak elementa označava se korištenjem šiljastih zagrada s njegovim nazivom, a kraj istim takvim zagradama s kosom crtom prije naziva. Tako je cijela grafika sadržana unutar `svg` elementa kojeg omeđuje kod `<svg></svg>`. Te odnose elemenata nazivamo “*parent*”(“roditelj”) i “*child*”(“dijete”), onaj koji sadrži, i onaj koji je sadržan. Kao što je vidljivo kod elemenata `circle` u primjeru, elemente bez djece možemo završiti unutar jednih zagrada. Također, elementi mogu imati tzv. atribute, koji se sastoje od naziva i vrijednosti u navodnicima, poput atributa `font-family="Futura"` u elementu `text`. Konačno, element u sebi može sadržavati i sadržaj izvan zagrada, poput teksta u *tagovima* `desc` i `text`.

U ovom su poglavlju dokumentacije navedeni česti i univerzalni elementi i njihovi atributi: Glavni element `svg` označava početak SVG koda unutar HTML stranice. Element `g` označava grupu, univerzalnu oznaku koja se koristi za primjenu zajedničkih atributa na više elemenata. Element s tekstom `desc` ne prikazuje se korisniku, već služi za potrebe indeksiranja, optičkih čitača, i web-preglednika koji

ne podržavaju SVG. On u pravilu sadrži opis sadržaja svog roditelja. Elementi `circle` i `text` neki su od tipova vektorskih podataka koji kasnije u dokumentaciji imaju vlastita poglavlja. `id` je univerzalni atribut koji pojedinom elementu dodjeljuje unikatni naziv putem kojeg se kasnije može pozvati drugdje u dokumentu. To se odvija putem atributa `xlink:href`; na primjer, prvi se krug iz primjera može pozvati kodom `xlink:href="#krug1"`. Ovako se mogu pozvati i vanjski dokumenti, na primjer u elementu slike koji poziva JPG sliku "slika1.jpg":

```
<image xlink:href="slika1.jpg">
```

Konačno, u ovom su poglavlju opisani načini na koji se u SVG dokumentu može definirati grafika namijenjena za kasnije korištenje i instanciranje. To se odvija umetanjem grupa vektora s `id` atributom unutar `defs` elementa, najčešće u sklopu `symbol` elementa. Ovi se elementi neće odmah prikazati korisniku; njihove vidljive verzije referencom poziva `use` element, unutar kojeg nastaje "klon" koda sadržanog u originalnoj definiciji, kao u primjeru:

```
<svg>
      <defs>
            <symbol id="krug"><circle /></symbol>
      </defs>
      <use xlink:href="#krug" />
</svg>
```

Korisniku će od ovog koda biti vidljiv samo `use` element, kojeg će web-preglednik interpretirati kao `g` element sa istim sadržajem i atributima kao `symbol` element kojega poziva.

Stiliranje ("Styling ")

SVG omogućuje definiranje "stilova", kao što su boja, debljina linije, ili `font-family` iz prvog primjera. Stilovi su atributi koji određuju izgled pojedinog elementa. Ovo je funkcionalnost preuzeta iz CSS jezika; dio je atributa u SVG standardu identičan onima u CSS-u. Osim *inline* upisivanjem atributa direktno u element, stilovi se mogu definirati i unutar `style` *tagova*. Kao što `symbol` element sadrži vektore za kasniju uporabu, `style` element sadrži grupe stilova koje se kasnije primjenjuju na željene elemente, najčešće određene klasom. Klasa, tj. `class` atri-

but je sličan `id` atributu, no ne definira unikatni element, već se koristi za označavanje više elemenata istog, proizvoljno definiranog, tipa. Primjer ispod sadrži četiri `text` elementa, među kojima su kroz dodijeljene klase samoglasnici obojeni crveno, a suglasnici plavo.

```
<svg>

      <style>
            .samoglasnik {color: red;}
            .suglasnik {color: blue;}
      </style>
      <text class="suglasnik">B</text>
      <text class="samoglasnik">O</text>
      <text class="samoglasnik">E</text>
      <text class="suglasnik">P</text>
</svg>
```

Element se kao meta za CSS stiliranje, osim klasom, može specificirati i drugim atributima, na primjer *tagom*, relativnim odnosima, `id` atributom, i tzv. pseudo-klasama poput `:hover`. CSS stilovi mogu biti definirani i kroz eksterni .css dokument zvan "*stylesheet*". CSS će u kasnijem poglavlju biti dodatno istražen kao jezik za animaciju.

Koordinatni sustavi, transformacije i jedinice ("Coordinate Systems, Transformations and Units")
Pozicioniranje elemenata unutar SVG dokumenta odvija se u beskonačnom koordinatnom sustavu. No, iscrtavanje se tih elemenata odvija u definiranom pravokutnom isječku tog sustava, kojeg nazivamo "*viewport*" ("okno"). Manipulacija elementima stvara nove koordinatne sustave unutar glavnog. Na primjer, kroz translaciju, rotaciju, i ostale tipove transformacije definirane `transform` atributom:

```
<svg>
      <g>
            <rect />
            <rect />
      </g>
      <g transform="translate(10,10) rotate(30) scale(2)">
            <rect />
            <rect />
      </g>
</svg>
```

Ovaj primjer sadrži dvije grupe od dva kvadrata. Dok prva grupa obitava u krovnom koordinatnom sustavu dokumenta, druga se grupa zbog transformacija iscrtava u sustavu koji je u usporedbi s glavnim pomaknut za deset u obje dimenzije, rotiran trideset stupnjeva, te povećan na dvostruku veličinu. Neki elementi, poput `use` i `svg`, automatski stvaraju novi sustav, i bez aplikacije transformacija. *Viewport* se može dodatno kontrolirati kroz `viewBox` atribut. On se sastoji od četiri podatka: početnih koordinata, te visine i širine. Uzmimo za primjer sljedeći svg element:

```
<svg width="100" height="100" viewBox="50 50 100 200" preserveAspectRatio="none" />
```

Ovaj će element unutar SVG dokumenta biti kvadrat sa stranicama duljine sto piksela, no njegovo će okno prikazivati pravokutni prostor koji ne uključuje ishodište sustava, i koji je dulji nego što je širok. Točnije, ovaj će kod rezultirati prikazom područja koordinatnog sustava od 50 do 150 u x-osi, te od 50 do 250 u y-osi, automatski "stisnutog" kako bi stalo u kvadratni format. Prilagodbu okna zadanom formatu određuje atribut `preserveAspectRatio`, koji omogućuje i očuvanje proporcija, te poravnanje sadržaja. Ovo je jedna od glavnih funkcionalnosti SVG-a za postizanje responzivnosti.

3.2 Vektorski oblici

Iduće tri sekcije dokumentacije određuju sintaksu za definiciju i iscrtavanje vektorskih grafičkih elemenata. Različite vrste vektorskih elemenata imaju različite atribute; na primjer, dok kružnicu definira radijus, pravokutnik definiraju visina i širina. Neovisno o tome, konačni je rezultat svih ovih elemenata skup točaka u prostoru i njihovih odnosa. Te je oblike kasnije moguće prikazati putem vizualnih stilova. Vrste vektora definirane u SVG dokumentaciji su:

Linije (“Paths”)
Najopširniji način za definiciju vektorske grafike koji SVG nudi je element `path`. On može definirati bilo kakav oblik, zbog čega je najčešća metoda zapisa vektora kompleksnijih od običnih geometrijskih likova. Njegov atribut `d` sadrži koordinate točaka kroz koje vektor prolazi, s naredbama koji određuju ponašanje u određenoj točki. U sljedećem je primjeru opisan kvadrat veličine 100, s početkom u točki “100,100”:

```
<path d=”M 100 100 L 200 100 L 200 200 L 100 200 z />
```

Naredba `M` označava pomak “olovke” na određenu točku, naredbe `L` označavaju crtanje linije do određene točke, a naredba `z` označava zatvaranje vektorskog oblika. Putem kompleksnijih naredbi moguća je i uporaba nekoliko vrsta krivulja, ne samo ravnih crta.

Osnovni oblici (“Basic Shapes”)
Ovo poglavlje opisuje šest “osnovnih oblika” koje SVG prepoznaje. Sljedeći ih primjer sadrži sve, sa svim atributima koje pojedini oblik podržava:

```
<circle cx=”0” cy=”0” r=”10” />

<ellipse cx=”0” cy=”0” rx=”10” ry=”20”

<rect x=”0” y=”0” width=”20” height=”10” rx=”2” ry=”4” />

<line x1=”0” y1=”0” x2=”10” y2=”20” />

<polyline points=”0,20 10,0 20,20” />

<polygon> points=”0,20 10,0 20,20” />
```

Prvi oblik je `circle`, krug. On je definiran koordinatama centra i polumjerom. Elipsa, `ellipse`, definira radijus u dvije dimenzije. Pravokutnik, `rect`, nudi šest atributa: koordinate ishodišta, visinu i širinu, te radijus rubova u dvije dimenzije, ako želimo pravokutnik sa zaobljenim rubovima. `line` je ravna linija definirana sa samo dvije točke. `polyline` je, pak, niz točaka spojenih ravnim linijama, određen skupom koordinata. Na isti je način određen i `polygon`, no on automatski spaja prvu i zadnju točku u nizu.

Teoretski, svi bi ovi oblici mogli biti definirani i `path` elementom. Ipak, korištenje sintakse za iscrtavanje osnovnih oblika kod čini mnogo kompaktnijim, a animaciju mnogo lakšom. Kao i uvijek u vektorskoj grafici, kombinacijom raznih jednostavnih oblika možemo kreirati kompleksnu, ali i dalje modularnu grafiku. Mnogi alati za dizajn vektora nude zapis u .svg format, no rijetki nude razinu optimizacije koja se može dobiti direktnim pisanjem SVG koda. Mana strojno zapisanih .svg datoteka može biti pretjerano oslanjanje na `path` element.

Tekst (“Text”)
Tekst je u SVG dokumentu sadržan unutar `text` elementa. Može biti upisan *in-line* ili *embedan* putem reference, a njegove su značajke, poput veličine i fonta, određene atributima. Tekst je, u suštini, samo skup krivulja, te time ima iste mogućnosti kao bilo koji drugi vektorski element. Istovremeno, SVG tekst zadržava semantičku funkcionalnost, te se u preglednicima može selektirati i kopirati.

Iako SVG ne podržava upisivanje paragrafa teksta unutar vektorskih okvira (za to je potreban CSS), on podržava ispis teksta po proizvoljnoj liniji, kroz `textPath` atribut s referencom na `path` element definiran u `defs` sekciji. SVG tekst također podržava transformacije pojedinih slovnih znakova unutar rečenice.

3.3 Vizualni prikaz vektora

Sljedećih pet poglavlja opisuju načine na koje se spomenuti vektorski elementi mogu stilski definirati. To uključuje jednostavno stiliranje putem atributa, kao kod npr. boje i obruba, te kompleksnije metode prikaza grafičkih informacija, poput maski, uzoraka, i filtera. Kroz ove se metode SVG kod iz matematičkih funkcija pretvara u sliku vidljivu i razumljivu korisniku.

Bojanje… (“Painting: Filling, Stroking and Marker Symbols”)
Vektori mogu imati ispunu i obrub, putem atributa `fill` i `stroke`. To se naziva “*painting*”, ili bojanje. Ono može biti detaljnije definirano atributima poput `stroke-width`, koji određuje debljinu obruba. Također, neki vektori, poput linija, mogu imati tzv. “marker”, simbol koji se nastavlja na njihove kuteve ili završetke. Njihova je glavna primjena crtanje strelica. Markeri se definiraju unutar `defs` elementa, putem `marker` *taga*, i pozivaju kroz `id`.

Boja (“Color“)
Boje SVG elemenata definirane su vrijednostima u RGB sustavu. Boja pojedinog elementa određena je `color` atributom.

Gradijenti i uzorci (“Gradients and Patterns“)
Osim bojom, SVG elementi mogu biti ispunjeni i gradijentima ili uzorcima ranije definiranima u `defs` sekciji dokumenta. U prvom je primjeru krug ispunjen linearnim gradijentom koji prelazi iz crne u zelenu:

```
<svg>
      <defs>
            <linearGradient id=”gradijent”>
                  <stop offset=”0” stop-color=”black” />
                  <stop offset=”1” stop-color=”green” />
            </linearGradient>
      </defs>
      <circle fill=”url(#gradijent)”/>
</svg>
```

Gradijent se tako aplicira na vektor putem `id` reference, u sklopu atributa koji prihvaćaju boju kao vrstu podatka. Boje se gradijenta određuju `stop` elementi-

ma koji su postotkom pozicionirani na relaciji između njegove početne i završne točke. Kod linearnih je gradijenata još moguće i promijeniti kut, a kod radijalnih žarište. Također je moguće odabrati ponašanje gradijenta nakon dolaska do njegove krajnje točke: ponavljanje, zrcaljenje, i slično. Za razliku od CSS-a, SVG ne podržava konusne gradijente.

Uzorci su vektorske grafike koje poput "pločica" tvore ispunu odabranog elementa. U idućem je primjeru definiran uzorak krugova u mreži od deset stupaca i redova, rotiranoj dvadeset stupnjeva.

```
<defs>
      <pattern id="uzorak" width="0.1" height="0.1"
      patternTransform="rotate(20)>
                  <circle />
      </pattern>
<defs>
```

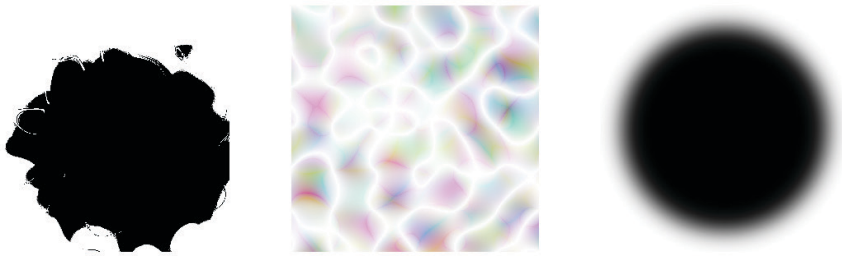
Uzorak se na element aplicira na isti način kao i gradijent. Atributom `patternTransform` moguće je upotrijebiti bilo koji od tipova transformacije na cijeloj mreži uzorka.

Maske… ("Clipping, Masking and Compositing")
Maske su grafike koje svojim oblikom "izrezuju" drugu grafiku. Njihov oblik služi kao "prozor" kroz koji gledamo sadržaj. Mogu biti vektori ili bitmape, te mogu biti poluprozirni. Definiraju se unutar `defs` elementa, kroz `clipPath` ili `mask` *tagove*, te se pozivaju atributima `clip-path` i `mask`. U primjeru je na kvadrat primijenjena maska određena slikom "maska.png".

```
<svg>
      <defs>
            <mask id="maska">
                  <image xlink:href="maska.png" />
            </mask>
      </defs>
      <rect mask="url(#maska)" />
</svg>
```

Filterski efekti (“Filter Effects“)

SVG omogućuje izradu rasterskih filtera koji su primjenjivi na bilo koji element, i koji mu tom primjenom mijenjaju izgled. Svaki se filter sastoji od serije osnovnih grafičko-matematičkih operacija definiranih u SVG dokumentaciji zvanih “primitivi”, poput Gaussovog zamućenja, dodavanja šuma, i tzv. *blending modeova* poput “*darken*” i “*screen*”. Oni se također pozivaju iz `defs` elementa. Zadaća im je na webu omogućiti upotrebu efekata ustaljenih u grafičkim alatima, bez gubitka semantičke strukture stranice. Budući da je je primitiva puno, i da je njihova uporaba relativno kompleksna, neće biti objašnjeni kroz kod, već samo kroz priložene primjere konačnih rezultata filtriranja:



Slika 2: Tri različita filtera primijenjena na obični crni krug

SVG filteri su su, dakle, vrlo moćan alat za stvaranje inovativnih vizualnih efekata. No, izvor te snage im je također i najveća mana: njihova rasterska priroda. Iako se rastriranjem u ovom slučaju ne kompromitira rezolucija, ono još uvijek izaziva znatno povećanje utroška prostora i energije za prikaz slike. To je pogotovo bitno za animacije, gdje svaka promjena atributa zahtijeva ponovno učitavanje rezultata filtriranja. Ovaj nedostatak ne bi trebao odvratiti web-dizajnere od korištenja takvih animacija, ali znači da je kod izrade grafika s SVG filterima potrebno posvetiti posebnu pažnju optimizaciji i testiranju.

Još jedan nedostatak SVG filtera je samo djelomična podrška njihovog korištenja na HTML elementima. Razlog tome je pojava CSS filtera, koji su usprkos boljoj podršci podosta malobrojni, i koji ne omogućuju izradu vlastitih efekata. Ipak, podrška za SVG filtere u CSS dokumentaciji postoji, tako da je moguće da će se *browserska* kompatibilnost u budućnosti poboljšati.

3.4 Ostale funkcije

Završnih osam poglavlja objašnjavaju funkcionalnosti izvan samog grafičkog prikaza vektora. U njima su opisani načini na koje SVG u suradnji s ostalim web-standardima može sudjelovati u stvaranju funkcionalne i multimedijske web-stranice. To su poglavlja:

Interaktivnost (“Interactivity“)
Ovdje su definirane vrste interakcija s elementima koje SVG prepoznaje, poput klika ili prelaska mišem. Ti su standardi podosta šturi, a uz to i pisani prije doba *touch* uređaja. Funkcionalnosti koje omogućuju su od tada modernizirane u sklopu CSS-a i Javascripta.

Povezivanje (“Linking”)
Ovo poglavlje opisuje način na koji funkcioniraju reference na vanjske objekte, te na definirane objekte unutar dokumenta. Moguće je referencirati i pojedini element sadržan u vanjskom .svg dokumentu, ako mu je dodijeljen `id`. Sljedeći kod poziva element s atributom `id="Mrkva"` iz slike “povrce.svg”:

```
<image xlink:href="povrce.svg#Mrkva" />
```

Skriptiranje (“Scripting”)
Ovdje je objašnjen `script` element, unutar kojega je moguće upisati skriptu koju će interpretirati programski jezik određen atributom. Usprkos podršci, ovo se u izradi web-stranica uglavnom ne radi u SVG dokumentu, već u krovnom HTML dokumentu koji također podržava `script` *tag*.

Animacija (“Animation“)
SVG ima mogućnost animacije elemenata putem modificiranog SMIL jezika, također W3C standarda. Osim toga, XML struktura SVG dokumenta omogućuje korištenje vanjskih jezika za animaciju. Ovo je područje detaljno opisano u idućem poglavlju ovoga rada.

Fontovi (“Fonts”)
Još jedan element koji se može umetnuti u `defs` sekciju SVG-a je `font`. On omogućuje definiciju seta vektora koji se kasnije može koristiti za prikaz slovnih znakova u `text` elementima. Nažalost, ova je funkcionalnost iz većine aktualnih

web-preglednika potpuno uklonjena zbog konsenzusa zajednice o postojanju boljih alternativa.[19]

Metapodatci (“Metadata”)
“Metapodatci su strukturirani podatci o podatcima.” Ovo poglavlje opisuje `metadata` *tag*, koji se koristi za umetanje podataka o autorima, datumu izrade, izvorima, itd.

Kompatibilnost unatrag (“Backwards Compatibility”)
Ovdje je definiran način za umetanje SVG koda u preglednike koji ga ne podržavaju, uz pretpostavku da ti preglednici imaju mogućnost ekstenzije, poput one definirane u idućoj sekciji dokumentacije.

Mogućnost ekstenzije (“Extensibility”)
Zadnje poglavlje definira način kojim se kroz element `foreignObject` u SVG datoteku može umetnuti kod kojeg ona samostalno ne podržava. Ovime se teoretski može proširiti funkcionalnost SVG jezika. Pretpostavka je da će preglednik kojim se ta datoteka prikazuje imati način interpretacije “stranog” koda.

4. METODE ANIMACIJE SVG ELEMENATA NA WEB-STRANICAMA

Glavni će izvor u ovom poglavlju biti knjiga “SVG Animations: From Common UX Implementations to Complex Responsive Animation” američke web-inženjerke Sare Drasner.[20] Sarah u ovom priručniku nudi vrlo opširan pregled mogućnosti velikog broja metoda animacije SVG grafika na webu, uključujući i za kompleksne primjene poput grafičke vizualizacije podataka. Osim uputa za korištenje SVG sintakse, ova knjiga sadrži mnoge vrlo aktualne naputke direktno temeljene na Sarinom iskustvu u radu na stvarnim projektima, kao i općenite savjete vezane uz dizajn-proces i proces izrade animacija. Već spomenuti Chris Coyier u predgovoru te knjige piše: *“SVG je jedinstveno kvalificiran za animaciju. On je definitivno najjači postojeći alat za animaciju na webu. [...] SVG animacija je toliko zanimljiva i zbog mnogobrojnih načina kroz koje se može izvesti. Postoje razne* native *tehnologije među kojima se može birati, kao i pomoćni* libraryji *izgrađeni nad njima. Kako znati koju odabrati? To zahtijeva nešto znanja i razmatranja.”*

Prije nego što krenemo s dubljim pojašnjenjem ovih tehnologija, potrebno je razlikovati ugrađene, tzv. *native* (“domaće”) tehnologije, te tzv. *libraryje* (knjižnice). Etiketa *native* označava već spomenute web-standarde čitljive svim web-preglednicima; izraditi web-stranicu unutar okvira *native* jezika znači koristiti samo resurse koje web-preglednici sami mogu interpretirati. Glavna je prednost takvih stranica jako dobra optimizacija, dijelom zbog kvalitete sintakse tih jezika, a dijelom i zbog truda koji su *developeri* web-preglednika uložili u njihovu efikasnu interpretaciju.

Pojam *library*, pak, označava skup specijaliziranih funkcija sagrađen na temelju jednog ili više *native* jezika, koji “sažima” određene kompleksne naredbe u mnogo jednostavnije “prečace”. Na webu je *native* podloga najčešće programski jezik JavaScript, a *library* .js datoteka. Ona *“ispod haube”* sadrži tekst koda kojim se veći broj naredbi definira kao jedna funkcija koja se uporabom ključnih riječi može pozvati, upotpuniti parametrima, te povezati s elementom na stranici. Ona može biti sadržana na lokalnom serveru, istom onom koji sadrži i web-stranicu, ili može biti “ugrađena” poveznicom na javno dostupni server, najčešće u vlasništvu *developera* tog *libraryja*. U kontekstu ovoga rada, radi se specifično o *libraryjima* za

animaciju, koji kompleksne funkcije poput relativnog tempiranja, prekidanja animacije, *randomizacije*, i *touch* interakcije specijaliziranom sintaksom omogućuju kroz samo par linija koda. Njihovo korištenje stoga omogućuje delegaciju jednog dijela rješavanja problema autorima *libraryja*. U suštini, korištenje *libraryja* znači korištenje gotovih rješenja za probleme koji su web-zajednici već neko vrijeme poznati. Kao i standardi, mnogi su *libraryji* djela timova volontera; ipak, neki od njih zahtijevaju licencu za komercijalnu uporabu. 82% web-stranica koristi barem jedan *library*.[21]

Generalno, animacija SVG elementa označava vremenski definiranu promjenu njegovih atributa. To su uglavnom transformacija (pozicija, rotacija, itd.), boja, i nešto rjeđe oblik, ali sva je dosad navedena SVG svojstva moguće animirati bar jednom od tehnologija navedenih u ovom poglavlju. Izbor metode animacije ovisi o dosta faktora; svaka od njih ima svoje mane i vrline, čije relativne važnosti variraju ovisno o pojedinoj primjeni. Metaforički rečeno, izbor između dvije metode SVG animacije je kao izbor između noža za putar i kuharskog noža: oba su alata uglavnom sposobna izvršiti zadaću, ali su rijetko u tome jednako efikasni. Tako je i većina web-jezika zamišljeno kao alat sa specifičnom namjenom, ali koji može uključivati i funkcionalnosti izvan te namjene. Stoga, samo zato što je nekim jezikom moguće doći do zadovoljavajućeg rješenja, ne znači da je taj jezik za to dobar. Također je moguće u dokumentaciji naići na definiciju funkcionalnosti koja u trenutnom web-standardu nije podržana, kao što je slučaj kod SVG fontova.

Općenito, funkcionalnost pojedinih *native* tehnologija varira ovisno o korisnikovom odabiru *browsera*. Kod pojave takvih problema najčešće se radi o Internet Exploreru, bivšem službenom pregledniku Microsoftovog operacijskog sustava Windows, koji je u ranom 21. stoljeću imao sličan udio u tržištu kakav danas ima Google Chrome. Microsoft je u više slučajeva u taj *browser* odbio integrirati među konkurencijom prihvaćene standarde kako bi “progurao” svoje autohtone tehnologije, te je time izazvao stagnacije u polju web-inovacija. Ipak, trenutna je situacija mnogo bolja, zbog čega kompatibilnost koda među preglednicima više nije svakodnevna briga web-*developera*. No, unatoč sve boljoj standardizaciji, još uvijek postoji mogućnost pojave varijacija u tajmingu, poravnanju, ili u relativnim pozicijama objekata. Iz tog se razloga preporuča svaku SVG animaciju prije plasmana na web testirati u svim najpopularnijim *browserima*.

Izrada kompleksnijih animacija najčešće uvjetuje i uporabu kompleksnijih jezika. U ostatku ovoga poglavlja biti će navedene razne tehnologije za pokretanje SVG grafika, definirane njihove primjene i ograničenja, te konačno i preporučena njihova uporaba u određenim slučajevima.

4.1 SMIL

SMIL (“Synchronized Multimedia Integration Language”, izgovara se *“smile”*) je jezik nekoliko godina stariji od SVG-a, koji je na sličan način u ranim danima W3C-a usvojen kao preporuka, a kasnije uglavnom zanemaren među širom zajednicom *developera*. Također temeljen na XML-u, originalno je zamišljen kao alat za izradu prezentacija koji će *“omogućiti autorima da plasiraju sadržaj poput televizijskog na web”*, te (kao što mu ime nalaže) kao način za lakšu integraciju i sinkronizaciju multimedijskog sadržaja na web-stranicama u doba prije modernih multimedijskih standarda.[22] U SVG je dokumentaciju integriran u potpunosti, preuzet i ugrađen kao pod-jezik, zbog čega je inherentno podržan među aktualnim web-preglednicima. Putem njega je moguće odabrati bilo koji atribut SVG elementa, te mu odrediti promjenu definiranu vremenom. U sljedećem je primjeru putem SMIL-a animiran radijus crvenog kruga, koji se varijabilnom brzinom povećava i smanjuje:

```
<circle xml:id="krug"
      cx="0" cy="0" r="10"
      fill="red" />

<animate xlink:href = “krug”
      attributeName="r" from="10" to="50"
      dur="5s" accelerate=".3" decelerate=".3"
      autoReverse="true" repeatCount="indefinite" />
```

Kao što je vidljivo, SMIL funkcionira umetanjem `<animate>` elementa u SVG dokument, i definiranjem animacije putem atributa tog elementa.[23] Krug je u ovom slučaju kao objekt na kojeg se primjenjuje animacija definiran putem `id` atributa, no moguće je i `animate` element usaditi unutar `circle` elementa, čime će na njega biti primijenjena animacija bez potrebe za referencom. Značajkom `attributeName` biramo atribut koji želimo animirati, u ovom slučaju radijus, te putem `from` i `to` određujemo početnu i konačnu vrijednost tog atributa. Ovim je putem moguće definirati i promjenu atributa koji sadrži nekoliko brojeva, ili koji ima tekstualnu vrijednost. Značajkom `dur` određujemo trajanje animacije, u ovom slučaju

pet sekundi, te način na koji ona ubrzava i usporava. Putem značajki `accelerate` i `decelerate` određujemo dio ukupnog trajanja animacije u kojem će se odvijati promjena brzine. U ovom su slučaju obje vrijednosti 0.3, to jest 30% ukupnog trajanja. Konačno, `autoReverse` i `repeatCount` su neki od načina definicije ponašanja animacije nakon njenog završetka; u ovom slučaju određuju animaciju koja se beskonačno ponavlja, i kojoj je svako drugo ponavljanje obrnuto, to jest koja prije vraćanja u početno stanje izvodi inverznu verziju definirane animacije. Ovaj je primjer vrlo osnovan; SMIL podržava i kompleksnije funkcije poput sekvenciranja animacija, definiranja brzine krivuljama, kretanja po krivulji, i transformacije vektora putem matrica. Zašto, onda, uopće razmatrati druge metode animacije kada postoji preporučeni W3C standard koji je razvijen paralelno s SVG-om, i koji može animirati bilo koju njegovu funkcionalnost? Razloga za to ima nekoliko. Kao prvo, SMIL je podosta star, zbog čega bi mogao frustrirati programere navikle na aktualne standarde. Na primjer, u kodu iznad vidljiva je zastarjela sintaksa "`xlink:href`", koja danas glasi samo "`href`", kao i danas nepotrebna kosa crta za završetak elementa. Ovo nije veliki problem (mnogi će *browseri* na isti način interpretirati obje verzije sintakse), no ukazuje na određeno kaskanje u modernizaciji standarda. Zbog ovog i drugih atavizama javila se želja za dodjelom tzv. *deprecated* statusa SMIL-u. *Deprecatanje* u kontekstu web-standarda predstavlja oznaku da je određena tehnologija "u prošlosti": da se ne preporuča njeno korištenje, i da je njena funkcionalnost u potpunosti zamjenjiva objektivno boljim alternativama. [24] Sama je Sarah Drasner odlučila uključiti SMIL u *"Not Suggested"* metode animacije SVG-a: *"Hrpa je vrlina i mana u radu sa SMIL-om, no najveća će me navesti da ga u potpunosti izostavim: gubi podršku."* Sarah ovdje referira na planirano napuštanje SMIL standarda od strane Google Chromea, najavljeno 2015. [25] Ipak, *developeri* su Chromea u međuvremenu odustali od te namjere, nakon što su im komentari zajednice sugerirali da SMIL u određenim primjenama još uvijek nije adekvatno zamijenjen. Tako, unatoč strahovima, *deprecatanje* ostaje potencijalna sudbina SMIL-a, ali mu nije u neposrednoj budućnosti. I konačno, SMIL nije idealan za komunikaciju s ostatkom weba. Iako podržava neke vrste interakcije, poput pokretanja ili pauziranja animacija klikom miša (detaljnije opisane u prošlom poglavlju), uvjerljivo nije najbolji izbor za interakciju i ostale primjene koje zahtijevaju reagiranje na unose korisnika. I za ovo je djelomično razlog slaba popularnost SMIL-a, zbog koje nije bio prioritet u integraciji u nove standarde,

kao ni u *third-party libraryje*.

Gdje onda, ako igdje, koristiti SMIL? U realnosti, čak i kada bi se njegova preporuka ukinula, nema presedana koji bi sugerirao da će time nestati i podrška koju trenutno ima. Google to ne spominje, a i sam W3C web-preglednicima preporuča da *deprecatane* tehnologije nastave podržavati, kako ne bi izgubili kompatibilnost sa stranicama izrađenim u vrijeme starijeg standarda. Ukratko, *deprecatana* tehnologija nastavlja funkcionirati, uz napomenu da se više ne koristi. Tako se u trenutku pisanja ovoga rada SMIL za određene primjene može preporučiti, čak i kada se u obzir uzmu pitanja o dugovječnosti standarda. Web-dizajner ili -*developer* koji smatra da prednosti uporabe SMIL-a nadilaze njegovu zastarjelost može se na svoju odgovornost odlučiti na uporabu ove metode animacije, koja je i dio samog SVG standarda. Jedna se tu prednost ističe kao glavna: naime, SMIL može biti vrlo praktičan zbog potpune integracije u .svg datoteku. SMIL kod sadržan je u sklopu SVG koda, i funkcionira neovisno o ostatku stranice. Ovim se putem animirana grafika u web-stranicu može usaditi na isti način kao i bilo koja druga slika, na primjer, u sklopu `<img>` HTML oznake i ili `background-image` CSS svojstva. Ovo je jedina metoda animacije SVG grafike koja ne zahtijeva izmjenu koda same stranice, ili dodavanje reference na vanjski kod; cijela je animacija već sadržana u slici. Uporaba integriranih SMIL animacija tako može olakšati iteraciju dizajna, a i sam kod web-stranice može učiniti kompaktnijim. Naravno, ovakav će se pristup koristiti samo u primjenama koje nisu sputane spomenutim problemima u komunikaciji s informacijama izvan `<img>` elemenata. U praksi, to su uglavnom ukrasne animacije koje se ponavljaju, i koje nemaju potrebu reagirati na akcije korisnika. One mogu biti animirane ilustracije i ikone, ili drugi elementi kojima je cilj privući pažnju. Također, to mogu biti i animirane pozadine u kojima je animacija više suptilna. Najbolji kandidati za uporabu SMIL-a su elementi koji bi inače bili statični, ali čija se zanimljivost uključenjem jednostavne animacije može nezanemarivo povećati.

4.2 CSS

CSS (Cascading Style Sheets) je jezik za opisivanje stilova XML elemenata, *de facto* standard za vizualni dio web-*developmenta*.[26] Na sličan način kao SMIL, može biti ugrađen u element koji opisuje, ili može biti referencom povezan s njim. Njegova snaga leži u velikom broju službeno prepoznatih atributa koje može definirati. Oni su slični SVG atributima (ponekad i identični), ali su detaljniji i mnogobrojniji: uključuju gradijente, sjene i zaobljenost rubova, određuju poravnanje s obzirom na roditelja, opisuju tekst, itd. Također, kao i SVG, CSS sadrži sintaksu za animaciju promjene svojih atributa. No, elementi koje CSS na webu najčešće opisuje nisu geometrijski likovi bilo kakvog oblika, to su u suštini pravokutnici: blokovi sadržaja unutar HTML hijerarhije. Tako CSS usprkos snažnim mogućnostima za opisivanje vizualnog izgleda nema mogućnost definicije vektorskih grafičkih elemenata. Na sreću, CSS sintaksa SVG element unutar HTML hijerarhije tretira kao bilo koji drugi, što interakcijom tih dvaju jezika omogućuje upotrebu CSS sustava animacije za pokretanje SVG grafike, kao i CSS spomenutih stilova za njeno opisivanje.

U sklopu CSS-a postoje dva tipa animacije: “animacije” i “tranzicije”. Kod tranzicija se ne određuju koraci kroz koje će vizualni izgled animiranog elementa prolaziti, već samo njegovo ponašanje u slučajevima gdje mu dolazi do promjene atributa. CSS tako na vrlo jednostavan način omogućuje responzivne prijelaze stanja. Sljedeći primjer koda sadrži gumb koji na korisnikov prelazak mišem mijenja boju iz bijele u žutu:

```
.gumb {color: white; transition: 2s;}
.gumb:hover {color: yellow}
```

Prvi red definira stilove za svaki element klase `gumb`; kao i boja, tranzicija je također atribut. Drugi red definira žutu boju za elemente klase `gumb` s pseudoklasom `hover`. To je pseudoklasa koja nastaje kada korisnikov pokazivač miša uđe u prostor elementa, te koja nestaje njegovim odlaskom. Kada ne bi bila definirana tranzicija, ova bi se promjena dogodila instantno, no zbog dodatka atributa `transition` u stilove ona će trajati dvije sekunde. Tranziciju je kroz atribute također moguće i odgoditi, te joj odrediti krivulju brzine.

CSS animacije, pak, više sliče SMIL principu animiranja, uz nešto moderniju i kompaktniju sintaksu:

```
@keyframes boja {

      0% {color: white;}
      60% {color: blue;}
      100% {color: white;}
}

.kvadrat {animation: boja 3s linear 2s infinite;}
```

Prva sekcija primjera definira animaciju kroz naredbu `@keyframes`, koju slijedi naziv animacije, te njen tok određen postotkom ukupnog trajanja: na početku je animacije boja definirana kao bijela, na 60% kao plava, te na kraju ponovo kao bijela. U drugoj je sekciji ta animacija pod nazivom `boja` željenom elementu pridružena kroz CSS stil. U ovom slučaju, animiran će biti svaki element na stranici s klasom `kvadrat`. Animacija će trajati tri sekunde, početi će dvije sekunde nakon učitavanja stranice, izvoditi će se linearno, te će se ponavljati beskonačno puta. Atribut `animation` može se dodatno dopuniti podatcima o odgodi i smjeru izvođenja.

Ove dvije metode animacije elemenata putem CSS-a imaju sličnu sintaksu, no dosta različite slučajeve uporabe. Dok animacijom možemo izazvati promjene stanja, tranzicija na njih može samo reagirati. Za jednostavne promjene između dva stanja, koje će se dogoditi jednom, preporučuje se upotreba tranzicija, a za promjene između više od dva stanja, pogotovo koje se ponavljaju, preporučuju se animacije. Ipak, obje metode imaju svoje nedostatke. Dok su tranzicije ograničene mogućnošću prelaska isključivo iz točke “A” u točku “B”, animacijama je glavna mana nedostatak responzivnosti. Za postizanje korisničke interakcije s CSS animacijama najčešće je potrebna uporaba JavaScripta, što će biti opisano u sljedeća tri poglavlja.

Sarah Drasner CSS naziva “Ockhamovom oštricom” web-animacije: *“najjednostavnije rješenje je ponekad i najbolje”.* Stoga se CSS za animaciju SVG-a na webu uvijek preporučuje kao prvi izbor, zbog daleko najveće jednostavnosti i

odlične optimizacije. Uporaba drugih tehnologija preporučena je u situacijama kada opseg i kompleksnost animacije izlazi izvan ograničenja metoda opisanih u ovome poglavlju.

4.3 JavaScript

JavaScript je programski jezik kojeg koristi velika većina web-stranica. Počeo je kao *native* jezik preglednika *Netscape* 1995., od kada je polako, ali sigurno postao standard za izvođenje koda u *browseru* korisnika.[27] Budući da on je pravi programski jezik, s njime je u teoriji moguće "bilo što", ali budući da nije zamišljen kao jezik za animaciju, može biti nezgrapan za takvu uporabu. Kao i u većini programskih jezika, moguće je definirati tzv. "petlju" ("*loop*") koja ponavlja zadane naredbe; tako je moguće napisati skriptu koja određeni element na stranici postepeno pomiče u stranu, i koja se ponavlja dvadeset puta u sekundi, dovoljno brzo da bi se gledatelju stvorio privid kretanja. Kako bi se ova metoda animacije olakšala, u JavaScript standard dodana je naredba `requestAnimationFrame`, optimiziranija vrsta petlje čija frekvencija ponavljanja ovisi o ekranu korisnikovog uređaja, i koja staje ako stranica na kojoj se nalazi nije u fokusu (na primjer, ako je u drugom *tabu* preglednika).[28] JavaScript animacije, dakle, imaju najširi spektar funkcionalnosti, no čak i uz olakšanja njihova izrada ostaje teže pristupačna, pogotovo dizajnerima i animatorima. Osim što zahtijeva dublje poznavanje kodiranja, korištenje JavaScripta za animaciju SVG grafika uključuje puno veću količinu apstrakcije nego druge metode, te ne uključuje mogućnost korištenja određenih "prečaca" koje animacijski jezici s razlogom sadrže. No, usprkos nezgrapnosti izrade animacija univerzalnim programskim jezikom, opširnost mogućnosti koje ta metoda nudi dopušta realizaciju rješenja koja izlaze izvan okvira standardnih funkcija web-preglednika. U kompleksnim projektima, pogotovo onima koji uključuju interakciju i povezivanje više sustava animacije u jednu cjelinu, animacija SVG-a putem JavaScripta može biti najbolji izbor.

Ipak, za takve se slučajeve najčešće ne koristi "obični" JavaScript, već neki od već spomenutih animacijskih *libraryja*. Njihova je zadaća upravo omogućiti korištenje efikasne animacijske sintakse unutar JavaScript skripte. Tako se za kompleksne primjene animirane grafike može preporučiti uporaba jednog od prikladnih JavaScript *libraryja* za animaciju. Primjene koje nisu dovoljno kompleksne da bi zahtijevale uporabu *libraryja* vrlo su vjerojatno CSS-om izvedive na kompaktniji način nego JavaScriptom. Više o dostupnim *libraryjima* biti će napisano u idućoj sekciji.

4.4 Eksterni *libraryji*

Ova metoda animacije dolazi u obzir kada mogućnosti ponuđenih *native* jezika nisu dovoljne, ili su oviše nepraktične. Sarah Drasner navodi nekoliko JavaScript *libraryja* za animaciju, od kojih najviše preporuča GreenSock, te kao alternativu Mo.js. Glavni *feature* koji oba nude, i koji u CSS-u nedostaje, je tzv. "*timeline*", sekcija koja definira redoslijed kojim će se animacije izvoditi. Ovo je pojednostavljeni primjer GreenSock *timelinea*:

```
var tl = new TimelineMax();
tl.to(animacija1});
tl.to(animacija2});
tl.to(animacija3}, "-=1");
```

Svaki *library* ima svoju sintaksu, umetnutu unutar JavaScript sintakse, zbog čega podsjeća više na programski kod nego na XML *markup* iz prošlih poglavlja. Prvi red koda ovdje stvara *timeline* kroz varijablu `tl`, dok daljnji redovi definiraju animacije u kronološkom nizu. Uz animacije mogu biti pridruženi atributi koji ih pomiču u vremenu: na primjer, `animacija3` će početi jednu sekundu ranije nego što `animacija2` završi. Moguće je također korištenje relativnih vremenskih relacija kako bi veći broj animacija počeo odjednom, kao i sekvenciranje više nezavisnih *timelinea* unutar krovnog *timelinea*. *Timeline* je moguće i pauzirati, te mu odrediti način ponavljanja.

GreenSock nudi i mnoga olakšanja koja česte kompleksne primjene web-animacije sažimaju u funkcije. Na primjer, za pretvaranje jednog SVG vektora u drugi potrebna je samo jedna linija koda:

```
TweenMax.to("#zvijezda", 1, {morphSVG:"#krug"});
```

Ova će animacija kroz period od jedne sekunde pretvoriti vektorsku zvijezdu u krug pomicanjem njenih krivulja i točaka. Funkcionira kroz naredbu `morphSVG`, koja referira na dva SVG elementa definirana `id` atributom. Postoji i funkcija `drawSVG`, kroz koju se linije vektora na jednako koncizan način mogu postepeno iscrtati u određenom vremenskom periodu. Još jedna mogućnost je tzv. `Draggable`, koji omogućuje translaciju ili rotaciju bilo kojeg web-elementa od strane korisnika, povlačenjem klikom miša ili putem dodira. Ovo uvelike olakšava *deve-*

lopment interaktivnih grafika i korisničkih sučelja. Sljedeći primjer kroz jednu liniju koda stvara volan koji korisnik može rotirati:

```
Draggable.create("#volan", {type: "rotation"});
```

Ova se funkcija može upotpuniti kako bi brzina vrtnje volana odgovarala fizičkoj simulaciji inercije. Moguće je i JavaScriptom prepoznati trenutnu vrijednost rotacije, čime se volan može koristiti za kontrolu drugog objekta na stranici. Sarah Drasner u svojoj knjizi prilaže primjer u kojem korisnik okretanjem zupčanika kroz *timeline* kontrolira odvijanje popratne animacije. Konačno, GreenSock, kao i SMIL, nudi i mogućnost pomicanja elementa po krivulji, ali za to nudi i grafičko korisničko sučelje izvedeno kroz *browser* ekstenziju.

Uz sve ove mogućnosti i prečace, lako je vidjeti zašto web-animatori često preferiraju korištenje *libraryja* nad mnogo rječitijim *native* izvedbama istih primjena. Stoga se alati poput GreenSocka mogu preporučiti za primjene SVG animacije koje zahtijevaju kompleksniju interakciju s korisnikom, pogotovo na mobilnim uređajima, te koje zahtijevaju dinamično tempiranje uvelike olakšano kroz *timeline*. *Developeri* koji zaključe da preferiraju sintaksu pojedinog *libraryja* nad npr. CSS-ovom mogu se odlučiti i na njihovo korištenje u jednostavnijim primjenama, no u tom bi slučaju trebali pripaziti na utjecaj koji to može imati na veličinu i optimizaciju stranice.

Uz *libraryje* za animaciju, postoje i oni koji upotpunjuju mogućnosti SVG-a na druge načine. Na primjer, *library* D3 nudi mnoge mogućnosti vezane uz grafički prikaz podataka, čime omogućuje stvaranje interaktivnih i iznimno detaljnih grafova, karata, dijagrama, i ostalih načina vizualizacije.[29] *Library* Raphaël omogućuje iterativnu manipulaciju SVG elementima kroz JavaScript, na dosta pristupačniji način nego D3.[30] SVG.js je animacijski *library* čiji je cilj biti maksimalno brz i kompaktan, te sadržavati samo najneophodnija poboljšanja nad *native* jezicima.[31] Konačno, postoje i vrlo specijalizirani mikro-*libraryji*, kao što je Vivus, čija je jedina funkcija ekvivalentna `drawSVG` funkciji unutar Greensocka.[32]

4.5 Web Animations API

Nadolazeći W3C standard za web-animaciju je koncizno nazvani Web Animations API. To je pokušaj uključivanja nekih od mogućnosti popularnih u animacijskim *libraryjima* u web-standard; primarno, to su *timeline* i kontrola animacije. Iako je trenutno još uvijek u fazi skice, njegova je podrška među *browserima* jako dobra; problemi postoje samo kod korištenja kompleksnijih funkcija na manje popularnim mobilnim preglednicima. U praksi je u velikoj većini slučajeva u potpunosti podržan. Njegova je sintaksa:

```
document.getElementById(“trokut”).animate(
      [
            { color: “red” },
            { color: “yellow”, offset: 0.7 },
            { color: “green” },
      ], {
            duration: 5000,
            iterations: Infinity
            easing: “ease”,
      }
);
```

Ovaj kod primjenjuje funkciju `animate` na element s `id` atributom `trokut`. Ta funkcija kao prvi atribut unutar uglatih zagrada sadrži *timeline*, a nakon toga unutar vitičastih podatke o tajmingu. Kod u primjeru će kroz pet sekundi (u sintaksi izraženo u milisekundama) trokutu promijeniti boju iz crvene, u žutu, te u zelenu. Animacije unutar *timelinea* biti će ravnomjerno raspoređene unutar zadanog vremena, osim ako im je dodan atribut `offset`. Tako će se promjena boje u žutu dogoditi na 70% trajanja animacije. API omogućuje i kontrolu reprodukcije animacija (pauziranje, ubrzavanje, itd.), te povezivanje te kontrole s proizvoljno definiranim akcijama korisnika na stranici. Ovaj dio funkcionalnosti također nudi i integraciju s CSS-om, čime proširuje i mogućnosti tog jezika.[33]

Neki od ovih *featurea* mogu eliminirati potrebu za izlaskom iz *native* jezika kod izrade dinamičnih i interaktivnih animacija. Tako će se Web Animations API, kada dobije puni status W3C preporuke, moći preporučiti za sve aplikacije za koje bi

se moglo preporučiti i korištenje JavaScript *libraryja*. *Developeri* koji to preferiraju mogu ga početi koristiti već i sad, no u tom bi slučaju na umu trebali imati da će se standard prije potpunog usvajanja najvjerojatnije još mijenjati i evoluirati. Najveći nedostatak koji trenutno ima naspram *libraryja* poput GreenSocka je nemogućnost promjene SVG vektorskih atributa poput koordinata točaka i krivulja.

4.6 Alati s grafičkim sučeljem

Za animatore s averzijom prema kodiranju postoje grafički alati s mogućnošću zapisa u .svg datoteku. Na primjer, za Adobe After Effects, popularni program za izradu pokretnih grafika, postoji nekoliko ekstenzija koje mu daju ovu funkcionalnost.[34] Iako je ovo legitiman način izrade SVG animacija, u ovome radu neće biti preporučen iz nekoliko razloga. Prvo, ti su alati ograničeni ne samo mogućnostima SVG-a, već i mogućnostima pojedinog grafičkog alata. To znači da će rezultat u najboljem slučaju biti isti kao i kod izrade pisanjem koda, ali najvjerojatnije lošije optimiziran. Drugo, ova metoda može opstruirati uključivanje interakcije u animaciju. Iako to nije nemoguće, dodavanje funkcionalnosti strojno zapisanim SVG datotekama uglavnom zahtijeva ručnu optimizaciju nakon izvoza, što u konačnici može oduzeti više vremena nego izrada kompletne animacije u web-jeziku. Treće, alati poput After Effectsa nude mogućnosti izvan SVG standarda, koje nije moguće ugraditi u .svg datoteku. Ovo bi moglo frustrirati animatore koji su navikli na određeni ustaljeni *workflow*. Konačno, kvaliteta ovih alata znatno varira, a mnogi se i plaćaju, te su time teže pristupačni od standardnih W3C jezika, čija je dokumentacija besplatna. Postoji mogućnost, pogotovo ako se popularizacija SVG animacije na webu nastavi, da će u budućnosti ti alati napredovati dovoljno da budu preporučljivi kao alternativa kodu, no trenutno je spektar njihovih mogućnosti suviše ograničen da bi mogli nuditi značajne prednosti animatorima koji već poznaju neki od spomenutih jezika. Ipak, ova metoda još uvijek može imati velike prednosti nad video ili GIF datotekama, te može biti korisna kao način da animatori koji ne namjeravaju naučiti SVG kod uživaju neke od prednosti korištenja .svg datoteka; primarno, njihovu malu veličinu na disku. Stoga se za takve slučajeve može preporučiti zamjena rasterskih animacija strojno zapisanim vektorskim, ako se time ne gube vizualne značajke.

5. PRIMJENE SVG ANIMACIJA U WEB-DIZAJNU

Nakon što je u prethodnim poglavljima opisana funkcionalnost SVG-a i načini njegove animacije u sklopu web-stranica, u ovom će poglavlju primjene spomenutih tehnologija biti podrobnije istražene kroz primjere uporabe. Primjena ovdje označava ulogu koju grafika u stranici zauzima; zadaću koju ispunjava. Kroz analizu više izvora, ove su primjene podijeljene na pet kategorija: “animirane ilustracije i pozadinske animacije”, “UX mikro-interakcije i naglašavanje sadržaja”, “animacije prijelaza stranice”, “animirane *hero* sekcije”, te “vizualizacija podataka”.[20][35][36][37][38][39][40][41] Unutar svake od ovih kategorija moguće je korištenje jednostavnih i laganih rješenja za čiju je izradu dovoljan CSS ili SMIL, te kompleksnih interaktivnih rješenja koja zahtijevaju uporabu nekoliko JavaScript *libraryja*, i koji se protežu kroz više kadrova. Nakon pojašnjenja raspona i značajki animacija koje pojedina kategorija uključuje, biti će priložen autorski primjer animirane web-stranice koji spada u tu kategoriju, izrađen specifično kako bi služio za analizu i pojašnjenje koda. Isječci tog koda biti će istaknuti i opisani u tekstu, dok će puni kod stranice biti priložen. Osim autorskog, biti će navedeni i javno dostupni primjeri iz literature, koji će služiti za demonstraciju kompleksnijih grafika, odviše opširnih i sofisticiranih da bi njihov kod mogao biti ukratko objašnjen.

Kod dizajna animacija namijenjenih za uporabu na web-stranicama postoje neke općenite smjernice, neovisne o specifičnoj funkciji pojedine grafike. Trenutno se u sferi web-dizajna i UX-a minimalna količina pokretnog dizajna smatra neophodnom, ali se uz to preporuča i umjerenost. Za uključivanje animacije u dizajn mora postojati razlog; cilj joj je uvijek poboljšati iskustvo korisnika i naglasiti ili podupriječeti najbitnije informacije. Animator stoga na umu mora imati cjelinu iskustva koje korisniku stranica želi pružiti. Izbor vrste, izgleda, i trajanja animacije ovisi o identitetu pojedinog brenda, kao i tipu stranice na kojoj se korisnik nalazi. Na primjer, početnoj je stranici najčešće cilj privući pažnju, te ona stoga može sadržavati veliku količinu animacija, čak i ako im je svrha većinski ukrasna. S druge strane, stranica na kojoj se nalaze npr. korisničke postavke ima određenu funkcionalnost od koje se pažnja ne bi trebala odvraćati. Stoga će u takvu stranicu biti uključene isključivo animacije koje pomažu u navigaciji i usmjeravanju korisnika prema traženom sadržaju. Na kraju dizajn-procesa, ključno je testirati stranicu i prilagoditi animacije saznanjima dobivenima od korisnika. *“Korištenje web animacije samo*

kako bi se pratili trendovi nije dobra ideja"[42]

Animacija se mora uklopiti u ostatak vizualnog stila i tona komunikacije. Kroz odabir trajanja pojedinih pokreta i njihovih krivulja brzine moguće je animaciji dati "karakter": Je li ona ozbiljna i hladna, ili je više zaigrana? Kvalitetan odabir ovih aspekata povezuje korisnika s vizualnim jezikom brenda, bilo to kroz dekoraciju, ili kroz funkcionalnost. Dobro aplicirana animacija "vodi" korisnika do sadržaja koji želi vidjeti, ali istovremeno ne skriva ostali sadržaj koji mu je ponuđen. Ovo je pogotovo bitno za primjene gdje se očekuje da će se korisnik više puta vraćati na istu stranicu.

Uspješnost uporabe animacije u nekim se slučajevima može mjeriti objektivnim metrikama poput broja klikova, vremena provedenog na stranici, itd. Ovako mjerljiva efektivnost često je rezultat pomno namještenog tempiranja. Općenito, podatci nalažu da je potencijal za zadržavanje pažnje korisnika na animiranim stranicama značajno veći nego kod statičnih.[43] Na ekranima, kao i u stvarnome svijetu, ljudska je percepcija sposobna primijetiti jako male promjene stanja, na koje će se gledatelj tada fokusirati više nego na prominentnije, ali statične elemente. Cilj je animacije, stoga, usmjeriti pažnju korisnika na željeni objekt ili dio stranice, ili mu zadržati pažnju tokom prijelaza između blokova sadržaja, npr. kod *loading* animacija. Istraživanja reakcije i održanja pažnje korisnika nalažu da se periodi kraći od desetinke sekunde percipiraju kao instantni, dok kod perioda duljih od deset sekundi korisnik gubi pozornost i tok misli vezan uz iskustvo na stranici. Trajanja animacija bi, dakle, trebala biti između te dvije krajnosti, kako korisnik animaciju ne bi propustio, odnosno kako mu ne bi dosadila. Pokreti animiranih elemenata trebali bi biti fluidni i logični, kako ne bi bili percipirani kao pogrešni ili neprirodni.

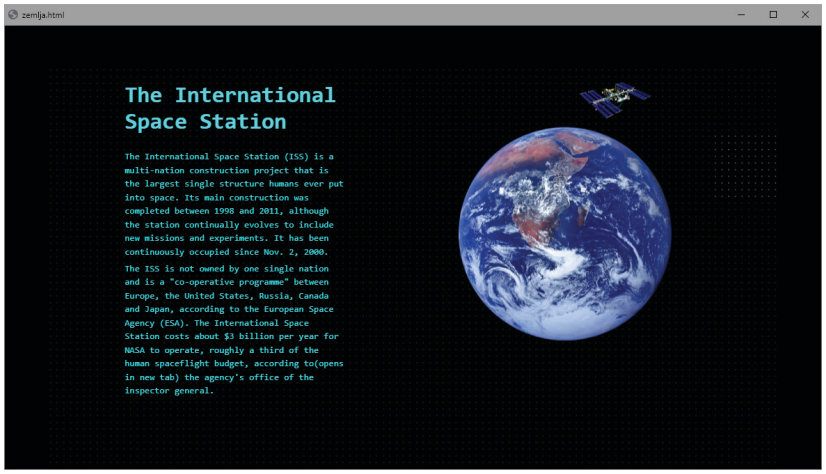
Animacije također mogu biti interaktivne, što će biti dodatno istraženo unutar pojedinih podpoglavlja. Kod izrade interaktivnih animacija za web potrebno je predvidjeti njihovu uporabu na *desktop*-, kao i na mobilnim preglednicima.

5.1 Animirane ilustracije i pozadinske animacije

Animacijom SVG elemenata moguće je na vrlo jednostavan način podići razinu "zanimljivosti" grafika koje se na stranici nalaze. Iako ova kategorija animacija najčešće ne nudi dodatnu funkcionalnost, ona može znatno povećati estetsku kvalitetu stranice, kao i tok korisnika kroz nju. Također, na ovaj se način stranica može istaknuti među konkurencijom: animacijom ilustracija, pozadina, ikona, te ostalih vizualnih elemenata koji su u većini slučajeva statični.

Ipak, iako im je primarni cilj uglavnom estetske prirode, ove animacije mogu biti i informativne, tj. mogu korisniku predočiti veći broj podataka od svojih statičnih verzija. Uz to, one mogu prenijeti i emocionalne informacije o identitetu brenda. *"Čak bi i animacije čisto vizualne prirode trebale imati cilj."*[37] Dok je animiranim ilustracijama cilj privući pažnju, pozadinske animirane grafike trebale bi pridodati glavnom sadržaju stranice, bez da odvraćaju pozornost od njega.

Autorski primjer za ovu kategoriju sadrži blok teksta o Međunarodnoj svemirskoj postaji, animiranu fotografsku ilustraciju, te interaktivnu pozadinu s uzorkom:



Slika 3: Prvi rad, statični kadar i QR kod s poveznicom

Ilustracija se sastoji od dva rasterska elementa, te od kružnice. Ona je u HTML stranicu *embedana* kao poveznica na vanjsku .svg datoteku, koja u sebi sadrži poveznice na vanjske .png datoteke. Postaja je unutar SVG dokumenta definirana unutar `defs` sekcije, gdje joj je u SMIL-u atributom `animateTransform` animirano rotacijsko ljuljanje, te joj je dodana i animirana kružnica koja se širi i nestaje:

```
<defs>

      <symbol id="iss">
            <image href="iss-sd.png" [...]>
                  <animateTransform
                        attributeName="transform"
                        type="rotate"
                        [...]/>
            </image>
            <circle [...]>
                  <animate attributeName="r" [...] />
                  <animate attributeName="opacity" [...] />
            </circle>
      </symbol>

      <path id="orbita" d="[...]" />

</defs>
```

Kao što je vidljivo u kodu, `defs` sekcija sadrži i `path` element koji se `id` atributom kasnije poziva kao putanja kojom će se Postaja kretati. Budući da SVG standard još ne podržava promjenu rasporeda slojeva, postaja je pozvana dvaput: jednom ispred slike Zemlje, i jednom iza. Ta su dva sloja izmjenično vidljiva animacijom `opacity` atributa uz atribut `calcMode="discrete"` koji određuje instantne prijelaze između stanja, te su rotirani kako bi orbita bila nakošena:

```
<g id="donji-iss" transform="rotate(-45)">

      <use href="#iss" [...]>
            <animateMotion [...]>
                  <mpath href="#orbita"/>
            </animateMotion>
      </use>
```

```
      <animate attributeName="opacity" from="0" to="1"
      calcMode="discrete" [...]/>

</g>
```

Ove će se animacije beskonačno ponavljati zbog animacijskog atributa `repeatCount="indefinite"`. Pozadina stranice je jednostavni kvadrat sa SVG `pattern` uzorkom sitnih kružnica, koji je više puta dodan u HTML dokument kako bi popunio pozadinu. Pojedino je "polje" točkica u početku poluprozirno, no ulaskom pokazivača miša u njegovo područje ono postaje jače vidljivo. Ova je animacija izvedena kroz CSS, pseudoklasom `hover`, a efekt "traga" koji pokazivač ostavlja postignut je razlikom između atributa `transition` u kvadratu s i bez pseudoklase:

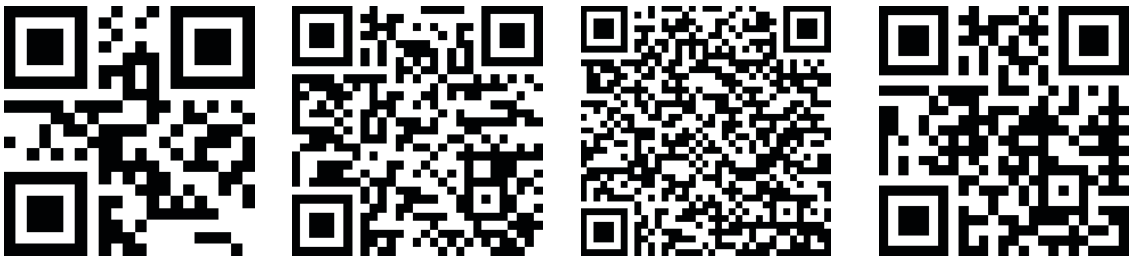
```
.nebo object {
      opacity: 30%;
      transition: opacity 1s ease .3s;
[...] }

.nebo object:hover {
      opacity: 100%;
      transition: opacity .4s ease 0s;
[...] }
```

U ovom je primjeru statična stranica na vrlo minimalan način pretvorena u pokretnu; bez potrebe za implementacijom kompleksnih interaktivnosti, i većinski kroz vanjske datoteke. Iako korisnik u primjeru ne može utjecati na sadržaj stranice, interakcija miša s pozadinom dodaje joj emocionalnu dimenziju. Također, ilustracija u primjeru sadrži rasterske elemente, čime se pokazuje svestranost SVG formata, ali i umanjuje jedna od njegovih vrlina: brze i male datoteke. Ovakve su grafike primjenjive na stranicama poput *blogova* ili članaka, gdje nije predviđena interakcija sa sadržajem, ali je poželjno uključivanje vizualno atraktivnih elemenata, a gdje je većina popratnog sadržaja izvedena fotografski.

Ovakvi "statični" pokretni elementi česta su primjena SVG animacija. Općenito, bilo koja statična grafika može biti i animirana, ako se time ne narušava funk-

cionalnost ili preglednost. Na primjer, *hero* sekcija tvrtke Zil Global sadrži vrlo jednostavnu pozadinsku animaciju sastavljenu od SVG kvadrata ispunjenih gradijentima.[44] Agencija BFF koristi animirane SVG ilustracije kako bi upotpunila početnu stranicu, dok agencija Baunfire radi isto za podstranicu sa godišnjim izvještajem.[45][46] Početna stranica tečaja Smart Interface Design Patterns ispunjena je animiranim SVG ilustracijama koje upotpunjuju tekst, a pokreću se samo jednom.[47] Dizajner Tom Ten Voorde animirao je vektorske ilustracije portreta zaposlenika za *"about us"* podstranicu svoje agencije.[34] Stranica "svgbackgrounds.com" nudi desetke animiranih i statičnih SVG pozadina prilagodljivih bilo kojoj stranici.[48]

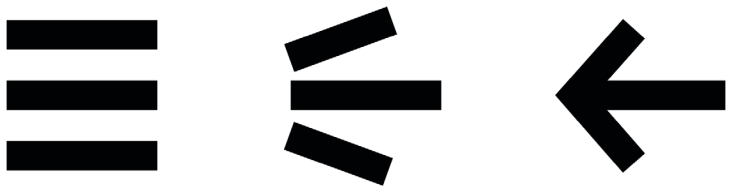


Slika 4: QR kodovi s poveznicama na primjere [44], [47], [34], i [48]

5.2 UX mikro-interakcije i naglašavanje sadržaja

U ovoj je kategoriji animacije najbitnija funkcionalnost. Iako je u svakoj primjeni potrebno na umu imati estetsku kvalitetu rješenja, kod UX animacija i animiranih mikro-interakcija ključno je korisnika ne “zbuniti” prevelikom količinom i isticanjem pokretnih elemenata, zbog čega je ovdje forma podređena funkciji. Ovim se animacijama može značajno poboljšati navigacija i korisničko iskustvo, pogotovo u slučajevima gdje grafike reagiraju na promjene stanja ili na interakcije. U slučaju gdje animacije znatno utječu na brzinu i performanse stranice, korist dobivena njima može biti zasjenjena nezadovoljstvom i frustracijom korisnika. Stoga UX i UI animacije trebaju biti jednostavne, lagane, fluidne, i logične.

Na *desktop* stranicama, najčešća je vrsta ovakve animacije tzv. *hover* animacija: ona koja se pokreće prelaskom mišem preko elementa. Takve animacije na UI elementima naglašavaju interaktivnost pojedinih dijelova stranice, te privlače pozornost korisnika prema funkcijama koje su mu najbitnije. Mana im je nekompatibilnost s *touch* sučeljima, zbog čega stranice koje se oslanjaju na *hover* interakciju mogu izgledati statično na mobilnim uređajima. Kako bi se funkcionalnost stranice održala, na mobilnim se formatima *hover* informacije uglavnom sažimaju u lebdeće izbornike ili skrivene menije.[49] Jedan je od ustaljenih klišeja u vektorskoj UI animaciji postala ikona *hamburger* menija koja se na klik pretvara u ikonu za povratak. Ova i slične animirane ikone nešto su što je već dulje vrijeme ustaljeno u mobilnim aplikacijama, a na webu je najlakše izvedivo kroz SVG. Za ovo postoje razne metode, od kojih je najjednostavnija uporaba *libraryja* koji lako može pretvoriti jedan vektor u drugi.[50] Ipak, za stvaranje prirodnije animacije potrebno je razmotriti načine na koje bi taj prijelaz mogao biti “uvjerljiv”. U primjeru *hamburger* menija, tri paralelne linije vrlo praktično mogu prijeći u strelicu, koja je također sastavljena od tri linije. Osim ovoga, postoje još mnogi domišljati prijelazi *hamburger* ikona.[51]



Slika 5: Prelazak iz ikone *hamburger* menija u ikonu strelice

Osim ikona, i cijeli meniji često dobivaju animirane tranzicije. Oni najčešće izlaze s jedne strane ekrana, ali mogu biti i skočni izbornici, koji kroz uporabu vektora poprimaju dinamične oblike. Web-dizajn blog Codrops daje više primjera "elastičnih" SVG menija.[52][53] Animacije progresije su još jedna UX primjena. One komuniciraju status određenog procesa povezivanjem SVG-a s JavaScriptom.[54] Animacijom se mogu prikazati i druge povratne informacije, pogotovo one vezane za akcije korisnika poput klika ili upisivanja teksta. Na primjer, promjenom boje ikone ili gumba može se simbolizirati promjena stanja. Dizajner Roman Bulakh dizajnirao je vektorsku ikonu koja komunicira napredak i status prijenosa datoteke.[55] Dizajnerica Mariana Beldi dijeli kod za formu koja slanje poruke komunicira animiranom ikonom sandučića za poštu.[56]

Slika 6: QR kodovi s poveznicama na primjere [52], [53], [54], i [55]

Slika 7: Drugi rad, statični kadar i QR kod s poveznicom

Autorski primjer u ovom poglavlju sastoji se od gumba za preuzimanje datoteke i ikone koja prikazuje status tog preuzimanja. SVG grafika sadrži sliku dokumenta, koja je statična, te strelicu, koja će se animirati:

```
<div id="ikona">
      <svg id="dokument" [...]>[...]</svg>
```

```
      <svg id="strelica" [...]>[...]</svg>
</div>
<div onclick="preuzmi()" id="gumb">[...]</div>
```

Klikom na gumb poziva se JavaScript funkcija `preuzmi`, koja pokreće animaciju. Animacija odabire elemente na stranici putem `id` atributa, funkcijom `getElementById`, te ih animira kroz Web Animations API, funkcijom `animate`:

```
function preuzmi() {
      var strelica = document.getElementById("strelica");
      var animacija = strelica.animate([
            {transform: "translate(0, 0)",},
            {transform: "translate(0, -30%)", fill: "lawngreen",},
            ], {
            duration: 1000, iterations: 8,
            direction: "alternate", easing: "ease-in-out"});
[...]};
```

Ovaj isječak koda pokreće strelicu gore-dolje, te joj mijenja boju, čime komunicira da je preuzimanje datoteke u tijeku. Završetkom preuzimanja strelica se kroz translaciju svojih sastavnih elemenata pretvara u kvačicu zelene boje. Izgled gumba također komunicira progresiju promjenom pozadine, izvedenom kroz animaciju oštrog CSS gradijenta. Ove su animacije za potrebe primjera tempirane specifičnim vremenskim vrijednostima, no u stvarnoj bi uporabi one bile dinamično vezane za proces preuzimanja.

5.3 Animacije prijelaza stranice

Prijelaze između stranica poželjno je korisniku vizualno pojasniti. Ovo uključuje dvije vrste animacije: tranzicijske animacije i tzv. *loading* animacije.

Tranzicije stranica, osim što su estetski privlačne, mogu i poboljšati korisnikovo shvaćanje trenutne pozicije unutar web-mjesta. One mogu biti potpuni prijelaz na novu web-adresu, ili mogu biti dio cjeline u obliku npr. *carousela*. Neki izvori animirane tranzicije nazivaju *storytelling* animacijama, jer im je cilj kroz pokret naglasiti "priču" koju se kretanjem kroz web-stranicu želi ispričati. Jako česta vrsta takvog *storytellinga* su animacije koje se pokreću *skrolanjem* kroz sadržaj. Ipak, tranzicije mogu imati i UX svrhu, gdje naglašavaju prelazak na novu vrstu sekcije; na primjer, u procesu kupnje putem internetske trgovine. Osim toga, pokazano je i da prebrzi ili nagli prijelazi između stranica mogu za korisnike biti zbunjujući, što dodatno ukazuje na potrebu za pomno tempiranim tranzicijskim animacijama.

Loading animacije popunjavaju vrijeme između učitavanja stranica. One najčešće dolaze u obliku ikona u žargonu poznatih kao "*throbber*" ili *"spinner"*.[42] To su kratke animirane ikone čiji se pokret ponavlja, te koje najčešće uključuju neki od zaštitnih vizualnih elemenata brenda. Kroz istraživanja je pokazano da ove ikone smanjuju šansu da korisnik tijekom učitavanja stranice odustane od čekanja, jer im one komuniciraju da se "nešto" događa, to jest da stranica nije zamrznuta, i da je učitavanje u procesu.[43] Stranica "loading.io" nudi veliki izbor animiranih *loading* ikona koje se mogu lako ugraditi u bilo koju web-stranicu.[57] Još jedna vrsta *loading* animacija su tzv. "*skeleton screen*" animacije, koje sliče na *wireframe* verziju stranice koja se učitava. Cilj im je korisniku sugerirati raspored elemenata na stranici, kako bi lakše mogao očekivati poziciju željenog sadržaja.

UI dizajner i *developer* Abbas Monfared svoju je stranicu u potpunosti koncipirao kroz SVG tranzicije među podstranicama.[58] Stranica 100 Years of National Parks Service prikazuje progresiju čitanja kroz animirane linije koje prate sadržaj, te kroz navigacijsku traku s lijeve strane.[59] Stranica poslovnog prostora Coal House radi isto, ali se linije koje naglašavaju tranziciju uz to i pretvaraju u ilustracije.[60] Web-aplikacija Zest tranzicije naglašava iscrtavanjem pozadinskih SVG elemenata.[61] Platforma Sales Hunter isto radi njihovom izmjenom i rotacijom.[62] Marketinška agencija Wavemaker cijelu uvodnu priču na početnoj stranici

komunicira kroz SVG tranzicije između animiranih kadrova.[63]

Slika 8: QR kodovi s poveznicama na primjere [58], [61], [62], i [63]

Za autorski je primjer izrađena jednostavna tranzicija koja se sastoji od dvije crne plohe koje na kratko vrijeme popune prostor ekrana. To su SVG `polygon` elementi pokrenuti CSS transformacijama, klikom na gumb koji poziva funkciju `animacija`:

```
function animacija() {
      var gornji = document.getElementById(“gornji”);
      var animacija1 = gornji.animate([...]);
      var donji = document.getElementById(“donji”);
      var animacija2 = donji.animate([...]);
      var body = document.getElementById(“body”);
      body.classList.toggle(“crvena”);
};
```

Dok poligoni popunjavaju ekran, `body` element dobiva klasu `crvena` koja mu mijenja boju pozadine. Kroz CSS `transition` atribut, ta je promjena odgođena kako bi se dogodila u periodu kada je pozadina sakrivena. Ova je tranzicija također i responzivna; “razvlači” se na bilo koji format ekrana.

Slika 9: Treći rad, statični kadar i QR kod s poveznicom

5.4 Animirane *hero* sekcije

U web-dizajnu, *hero* sekcija je uočljiva grafika plasirana na sam vrh početne stranice. Najčešće sadrži tzv. "*call-to-action*" ili CTA, interaktivni element poput gumba koji korisnika vodi na daljnje informacije, ili u proces registracije. Ova je kategorija vrlo općenita; sadržaj koji se animira specifičan je za pojedini brend i sektor tržišta u kojem se on nalazi. Ipak, sve *hero* sekcije, bile animirane ili statične, imaju neke zajedničke značajke. U praksi, cilj im je kroz impresivan vizualni sadržaj u vrlo kratkom roku uhvatiti pažnju gledatelja. Ovdje animirana grafika ima znatnu prednost nad statičnom, a interaktivna grafika nad objema. SVG animacijom može se povećati količina sadržaja kojoj je korisnik izložen već od samog ulaska na stranicu, čime se na intuitivan način komuniciraju vrijednosti i značajke brenda, te atmosfera i emocije kojima je on definiran. Ona također služi i kao odskočna daska za daljnju navigaciju.

Dizajner Cyril Conton na svojoj portfolio-stranici putem SVG-a animira tipografiju, kao i ikone sekcija koje se nalaze u kutovima.[64] Početna stranica e-knjige Go Freelance u *hero* sekciji sadrži ilustriranu naslovnicu koja se iscrtava pri ulasku na stranicu.[65] Platforma Ooki za kriptovalute na svojoj *hero* sekciji sadrži animiranu SVG maskotu, uz ostale animirane SVG ilustracije.[66] Agencija za *development* Skysoft u SVG elemente uključuje i interakciju, te dinamično pozicioniranje.[67] Agencija Sparky koristi slično rješenje, a dodaje i SVG pokazivač miša.[68]

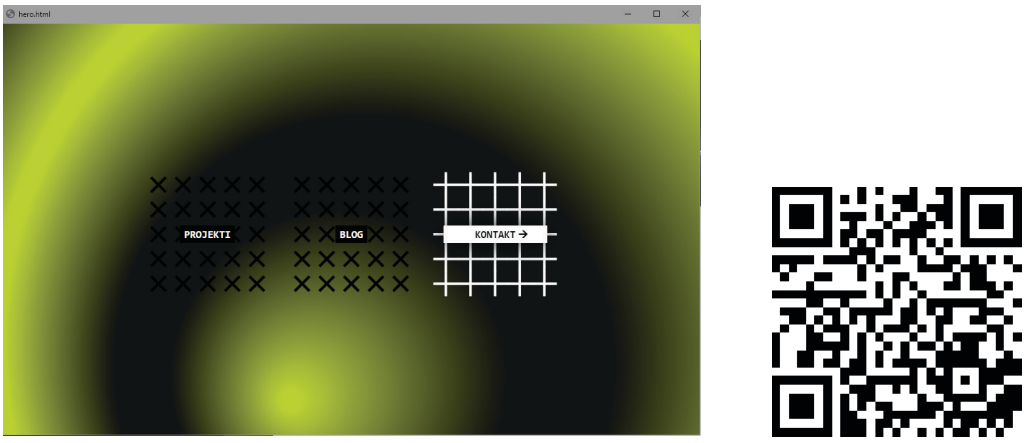
U nekim je slučajevima granica između *hero* sekcije i ostatka stranice tanka, ili je nema. Određeni brendovi mogu se odlučiti na animaciju kompletne početne stranice, koja tada poprima oblik neprekinute animirane cjeline. Reklamne stranice Apple proizvoda često ulaze u ovu kategoriju.[43]

Slika 10: QR kodovi s poveznicama na primjere [64], [66], [67], i [68]

Autorski primjer *hero* sekcije sadrži SVG pozadinu animiranu kroz SMIL, te izbornik s *hover* efektima. Dok je pozadina samostalna .svg datoteka koja se sastoji samo od kvadrata ispunjenog pokretnim gradijentom, vektorski elementi u pozadinama izbornika u HTML su dokument uključeni *inline*, te su animirani CSS tranzicijama i pseudoklasom `hover`. Prelaskom mišem preko pojedinog izbornika crni elementi oblika “X” rotacijom postaju bijela mreža linija:

```
.blok:hover svg .X {
      transform: rotate(45deg) scale(1.2);
      stroke: white;
}
```

Izbornici sadrže i okvir s tekstom, u kojem se na *hover* prikazuje strelica.

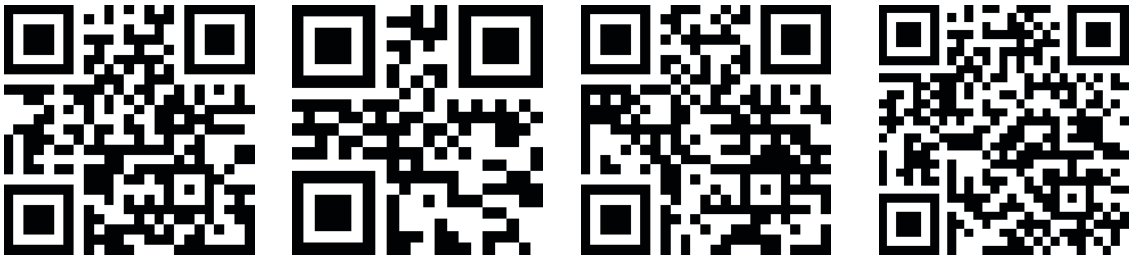


Slika 11: Četvrti rad, statični kadar i QR kod s poveznicom

5.5 Vizualizacija podataka

Vizualizacija podataka je najkompleksnija primjena SVG animacije, jer zahtijeva poznavanje programskog koda, kao i matematičkih operacija. Stranice koje sadrže ovakvu animaciju često moraju iščitavati informacije iz određene baze podataka u stvarnom vremenu. Neki od tipova grafika putem kojih se ti podatci prikazuju su grafovi svih vrsta, mentalne mape, dijagrami toka, te interaktivne karte. Glavni ustaljeni alat za ovu primjenu već je spomenuti D3, koji nudi iznimno široku paletu opcija.[20] Bilo se koji podatak na ovaj način može vezati uz bilo koji SVG parametar, čime se stvaraju animirane infografike.

Investment Calculator putem SVG grafova prikazuje novčane informacije.[69] Stranica NYC Foodiverse vizualizira podatke o sanitarnim inspekcijama restorana u New Yorku.[70] Promo-stranica aplikacije Hello, Sun kroz animirane SVG ilustracije prikazuje nebo i poziciju sunca na različitim lokacijama.[71] Stranica koja obilježava osamdesetu godišnjicu Konferencije u Wannseeu kroz interaktivnu SVG kartu Europe prikazuje podatke o Holokaustu.[72] Još je jedan primjer stranica The Divide, koja kroz SVG animaciju točaka u dvije boje prikazuje informacije o razlikama između prihoda spolova.[73]



Slika 12: QR kodovi s poveznicama na primjere [69], [71], [72], i [73]

Autorski primjer za posljednju kategoriju je karta sjeverne Hrvatske, podijeljena na županije. Vektori koji ju sačinjavaju grafičkim su programom izolirani iz javno dostupnih SVG karti, te je taj strojno zapisani kod kasnije manualno uređen i *inline* kopiran u HTML dokument. Županije su `polygon` elementi određeni `id` atributom.

```
<div[...]><svg class="karta" [...]>
      <polygon id="krapinsko-zagorska" [...] />
      <polygon id="varazdinska" [...] />
```

```
      <polygon id="zagrebacka" [...] />
      [...]
</svg></div>

<div id="podatci">[...]</div>
```

Kao što je vidljivo, dokument sadrži i `div` element s podatcima koji se prikazuju odabirom pojedine županije. To je izvedeno JavaScriptom, izmjenom CSS klasa kroz funkciju `info` pozvanu klikom miša. Za potrebe primjera funkcionalnost je uključena samo u vektor Zagrebačke županije.

```
function info() {
      var zupanija = document.getElementById("zagrebacka");
      zupanija.classList.toggle("odabrano");
      var podatci = document.getElementById("podatci");
      podatci.classList.toggle("vidljivo");}
```

Funkcija županiji pridodaje klasu `odabrano`, a podatcima klasu `vidljivo`, koje im mijenjaju vizualne atribute. Tako se odabrana županija ističe, a podatci postaju vidljivi. Na ponovni klik te se klase elementima oduzimaju. Atributi koje mijenjaju su sljedeći:

```
polygon.odabrano {
      transform: translate(-5px, -5px);
      stroke-width: 1px;
      [...]}
#podatci.vidljivo {width: 100%; [...]}
```

Slika 13: Peti rad, statični kadar i QR kod s poveznicom

6. ALTERNATIVE SVG ANIMACIJI

Budući da je SVG jedini aktualni web-standard za zapis vektorskih grafika, među *native* jezicima u polju animacije vektora nema konkurenciju. Ipak, on ima svoje granice koje za vrlo kompleksne primjene mogu biti suviše uske: na primjer, ako se radi o programiranju i animaciji cijele video-igre unutar preglednika. Ovo je niša koju je u bliskoj povijesti popunjavao Flash, ali je to mjesto izgubio jer, nakon dolazaka novih standarda, nije nudio dovoljno *featurea* da opravda svoj utrošak resursa. No, neki alternativni jezici u pojedinim primjenama zbilja nude značajno jače alate. Oni mogu biti preporučeni ako projekt unutar SVG-a uvjerljivo nije izvediv. S druge strane, neke su primjene toliko jednostavne da uopće ne zahtijevaju korištenje vektora, ili uključuju foto-materijal, zbog čega u njima mogućnosti SVG-a ne igraju ulogu. U ostatku će ovog poglavlja biti opisane neke grafičke primjene u kojima SVG nije najbolji izbor.

Prva su vrsta alternativa animirani rasterski formati. Ovo uključuje sve vrste video-formata, kao i formate “animirane slike” kao što je GIF. GIF animacija na mnogo je načina bila začetak animiranog weba.[42] Ipak, povećanjem rezolucija ekrana veličine su .gif datoteka postale prevelike za održavanje zadovoljavajućih brzina učitavanja na većim formatima. Danas je GIF uglavnom asociran s animacijama niske kvalitete i rezolucije, zbog čega se u mnogim uporabama zamjenjuje novim video-formatima koji su istovremeno “lakši” i bolje rezolucije. Najveća je prednost ovih tehnologija u spomenutim primjenama koje zahtijevaju uporabu fotografskog materijala, ili čak video-materijala. Iako SVG podržava uporabu rasterskih elemenata unutar koordinatnog sustava, ne nudi mogućnosti za njihovu manipulaciju usporedive s modernim alatima za *video-editing*. Neke je efekte nemoguće izvesti vektorski, poput tzv. “specijalnih efekata” koji se mogu vidjeti u filmovima. Druge je, pak, moguće izvesti putem SVG filtera, ali se time stranicu čini “težom” nego što bi ju činilo učitavanje ekvivalentnog video-sadržaja. Naravno, taj će video za razliku od vektora biti komprimiran, što u nekim slučajevima može izazvati vidljivu razliku u izgledu grafike, i čime se gubi mogućnost transparentnosti slike. (I SVG je moguće komprimirati, ali zbog vektorske prirode time ne gubi na kvaliteti.) Također, animirane rasterske datoteke su snimljene unaprijed; nisu responzivne. Responzivnost se u video- i foto-sadržaju uglavnom postiže dinamičnim pauziranjem i pokretanjem, ili alternacijom između prikazanih

datoteka unutar istoga okvira.[43] Konačno, rastriranjem sadržaja gubi se njegova semantička struktura: nije ga moguće rastaviti na elemente.

Tehnologija Canvas je *native* način za crtanje grafika u JavaScriptu.[74] Iako je on rasterski element, između ostalog podržava i uporabu vektora. Kao i ostale primjene programskog jezika za crtanje grafika, može biti prilično neintuitivan, ali je u praksi moćniji nego jezici kojima je crtanje grafika jedina funkcija. Stoga, kao što *libraryji* za animaciju uglavnom počivaju na `requestAnimationFrame` funkciji JavaScripta, grafički *libraryji* uglavnom počivaju na funkciji `canvas`. Primjer je Paper.js, koji sebe naziva *"švicarskim nožićem"* vektorskog skriptiranja, a namjena mu je direktna kontrola točaka i krivulja vektora, te njihove međusobne interakcije.[75] Širina mogućnosti takvih *libraryja* čini ih najboljim izborom za spomenute vrlo kompleksne primjene, poput video igara. Kroz Canvas tehnologiju realizirana je i većina implementacije 3D objekata u web-preglednik. Na primjer, WebGL (Web Graphics Library), standard za izvedbu trodimenzionalne grafike na webu, temeljen je na Canvasu.[76] Tako se Canvas i *libraryji* građeni na njemu mogu preporučiti kod izrade vizuala koji nadilaze definiciju vektorske grafike, te koji ulaze u sferu interaktivnog iskustva ili 3D grafike.

Lottie je jezik čija je funkcija primarno kao način zapisa datoteka izrađenih u programu sa grafičkim sučeljem, poput After Effectsa ili Figme. Ona je JavaScript *library* koji sadrži potpuno zaseban vektorski jezik, s mnogo sličnih ideja kao i SVG.[77] Budući da podržava izradu animacija pisanjem koda, Lottie može biti legitimna alternativa, no njeno korištenje podrazumijeva izlazak iz *native* standarda, a animatorima koji ne koriste grafičke programe ne nudi mnogo prednosti nad SVG-om.

Konačno, neke su primjene dovoljno jednostavne da bi mogle biti izvedene samim CSS-om. Tu se uglavnom radi o grafikama baziranim na pravokutnicima, za koje nije potrebna implementacija vektorskih krivulja. CSS je također preporučen za većinu animacija tekstualnog sadržaja.

7. OSVRT I ZAKLJUČAK

Nakon što je dugo vrijeme u svojoj sferi bila zanemarivana, SVG tehnologija u relativno je kratkom roku doživjela vrlo značajan porast u popularnosti. U suradnji s ostalim web-standardima, kao i samostalno, SVG može biti podloga za izradu kompleksnih, ali brzih, responzivnih, i interaktivnih animacija. U polju vektorskih web-grafika, za veliki je broj primjena uvjerljivo najbolji izbor, dok alternative koje su ga godinama mijenjale sada značajno gube na popularnosti. Jedan od razloga za ovako brzo usvajanje može biti njegova lakoća prilagodbe raznim kontekstima; uporaba ne zahtijeva integraciju nikakvih nestandardnih tehnologija u stanicu, a struktura sintakse je pristupačna onima koji su se već susreli s drugim W3C jezicma.

Ipak, za mnoge je web-*developere* i -dizajnere SVG još uvijek nepoznanica, usprkos odličnoj podržanosti i širokom spektru mogućnosti. Stoga je vrlo vjerojatno da je trenutni status SVG-a još uvijek rana faza usvajanja, i da će njegova popularnost još rasti. Veliki pomak mogao bi se dogoditi potpunim usvojenjem standarda SVG 2, koje je u procesu, a koje bi mu znatno proširilo funkcionalnost i moderniziralo određene dijelove sintakse. Autori SVG animacija za web mogu biti optimistični o budućnosti jezika, i mogu očekivati pojavu sve većeg broja resursa, kao i poboljšanje postojećih. Iz perifernog i ponekad zaboravljenog standarda, SVG je prerastao u jednu od primarnih *frontend* web-tehnologija.

8. LITERATURA

Slike za koje nisu navedeni izvori su autorski rad. Autorske animacije su dostupne na poveznici: *codepen.io/dr_tomasaki*

[1] guides.lib.umich.edu/c.php?g=282942&p=1885352 - University of Michigan Library - Raster vs. Vector Images

[2] siliconpublishing.com/blog/2015-12-the-fall-and-rise-of-svg/ - Silicon Publishing - The Fall and Rise of SVG

[3] https://www.irt.org/articles/js176/ - Irt.org - SVG Brings Fast Vector Graphics to Web, 21.9.2022.

[4] w3.org/Graphics/ScalableReq.html - W3C - W3C Scalable Graphics Requirements

[5] w3.org/Graphics/SVG/WG/wiki/Secret_Origin_of_SVG - W3C - The Secret Origin of SVG

[6] w3.org/standards/xml/core.html - W3C - XML Essentials

[7] w3.org/Graphics/SVG/History - W3C - W3C Scalable Vector Graphics (SVG) - History

[8] www.xml.com/pub/a/2001/09/12/svg.html - XML.com - Picture Perfect

[9] techcrunch.com/2017/07/25/get-ready-to-say-goodbye-to-flash-in-2020 - TechCrunch - Get ready to finally say goodbye to Flash — in 2020

[10] web.archive.org/web/20100502021750/https://www.apple.com/hotnews/thoughts-on-flash/ - Apple - Thoughts on Flash

[11] w3.org/TR/SVG11 - W3C - Scalable Vector Graphics (SVG) 1.1 (Second Edition)

[12] w3.org/blog/news/archives/3112- W3C - Scalable Vector Graphics (SVG) 2 Draft Published

[13] librearts.org/2016/11/is-svg-2-really-on-life-support - Libre Arts - “Is SVG 2 really on life support?”

[14] w3.org/TR/SVG2 - W3C - “Scalable Vector Graphics (SVG) 2

[15] tavmjong.free.fr/svg2_status.html - Tavmjong Bah - SVG 2 is on life support.

[16] css-tricks.com/svg-2-conundrum/ - CSS-Tricks - The SVG 2 Conundrum

[17] redmonk.com/kholterhoff/2022/02/07/svgs-whats-the-hype/ - RedMonk - SVGs: What’s the Hype?

[18] w3techs.com/technologies/history_overview/image_format/all/y - W3Techs - Historical yearly trends in the usage statistics of image file formats for websites

[19] chromestatus.com/feature/5930075908210688 - Google - Feature: SVG (1.1) Fonts (Removed)

[20] Drasner S. (2017.) *SVG Animations*, O’Reilly

[21] w3techs.com/technologies/overview/javascript_library - W3Techs - Usage statistics of JavaScript libraries for websites

[22] w3.org/Press/1998/SMIL-REC - W3C - Press release: W3C Issues SMIL as a W3C Recommendation

[23] w3.org/TR/REC-smil/ - W3C - Synchronized Multimedia Integration Language (SMIL 3.0)

[24] www.w3.org/TR/html4/conform.html- W3C - HTML4 - Conformance: requirements and recommendations

[25] developer.mozilla.org/en-US/docs/Web/SVG/SVG_animation_with_SMIL - Mozilla - SVG animation with SMIL

[26] w3.org/Style/CSS/Overview.en.html - W3C - Cascading Style Sheets
[27] w3schools.com/js/js_history.asp - W3Schools - JavaScript History
[28] css-tricks.com/using-requestanimationframe/ - CSS-Tricks - Using requestAnimationFrame
[29] d3js.org/ - D3.js
[30] raphaeljs.com/ - Raphaël.js - An Intro to Raphaël
[31] svgjs.dev/docs/3.0/ - SVG.js
[32] maxwellito.github.io/vivus/ - vivus.js
[33] w3.org/TR/web-animations-1/ - W3C - Web Animations
[34] elnino.tech/blog/what-i-learned-from-making-svg-animations-with-after-effects-and-bodymovin - El Niño - What I learned from making SVG animations with After Effects and Bodymovin
[35] blog.tubikstudio.com/web-animation/ - Tubik Studio -Motion in UX Design: 6 Effective Types of Web Animation, 21.9.2022.
[36] lvivity.com/web-animations-use-cases - Lvivity - Web Animations: 7 Use Cases in Web App Design and Great Examples, 21.9.2022.
[37] designshack.net/articles/graphics/an-introduction-to-animation-in-web-design/ - Design Shack - An Introduction to Animation in Web Design, 21.9.2022.
[38] fireart.studio/blog/10-best-website-animation-techniques-for-your-web-design/ - 15 Best Website Animation Techniques for Your Web Design - Fireart, 21.9.2022.
[39] constructive.co/insight/web-animation-delightful-or-distracting/ - Constructive - Web Animation: Delightful or Distracting?, 21.9.2022.
[40] explain.ninja/blog/animation-in-web-design-all-a-client-needs-to-know/ -Explain Ninja - Animation in Web Design: All a Client Needs to Know, 21.9.2022.
[41] toptal.com/designers/web/animating-the-web-in-the-post-flash-era - Toptal - Web Animation In The Post-Flash Era, 21.9.2022.
[42] techopedia.com/definition/23814/throbber - Techopedia - What is a Throbber?, 21.9.2022.
[43] css-tricks.com/lets-make-one-of-those-fancy-scrolling-animations-used-on-apple-product-pages/ - CSS-Tricks - Let’s Make One of Those Fancy Scrolling Animations Used on Apple Product Pages, 21.9.2022.
[44] zil.global - Zil Global - Zil Global, 21.9.2022.
[45] bff.design - BFF - BFF, 21.9.2022.
[46] eoy.baunfire.com/2017 - Baunfire - Baunfire End-Of-Year 2017, 21.9.2022.
[47] smart-interface-design-patterns.com - Smashing Magazine - Smart Interface Design Patterns, 21.9.2022.
[48] svgbackgrounds.com - SVG Backgrounds - SVG Backgrounds, 21.9.2022.
[49] bootcamp.uxdesign.cc/mobile-doesnt-have-hover-dude-b37e8e0b586e - Bootcamp - Mobile doesn’t have hover, dude!, 21.9.2022.
[50] tympanus.net/codrops/2013/11/05/animated-svg-icons-with-snap-svg/ - Codrops - Animated SVG Icons with Snap.svg, 21.9.2022.
[51]sliderrevolution.com/resources/css-hamburger-menu/ - Slider Revolution - Cool CSS Hamburger Menu Icons and Their Animations, 21.9.2022.
[52] tympanus.net/Development/ElasticSVGElements/collapseexpand.html - Codrops - Elastic SVG Elements, 21.9.2022.

[53] tympanus.net/Tutorials/ShapeHoverEffectSVG/index3.html - Codrops - Shape Hover Effect with SVG, 21.9.2022.
[54] alvarotrigo.com/blog/progress-bar-css/ - Álvaro Trigo - 20 Amazing CSS Progress Bars [With Examples], 21.9.2022.
[55] dribbble.com/shots/1989304-Attaching-File - Roman Bulakh - Attaching File, 21.9.2022.
[56] medium.com/leniolabs/contact-form-with-animated-feedback-51df60d2bfb6 - Leniolabs - Contact form with animated feedback, 21.9.2022.
[57] loading.io - PlotDB Ltd. - Loading.io, 21.9.2022.
[58] monfa.red - Abbas Monfared - Abbas Monfared, 21.9.2022.
[59] 100yearsofnps.com - National Park Service - 100 Years of National Parks Service, 21.9.2022.
[60] coalhouse.co.uk - Create Real Estate - Coal House, 21.9.2022.
[61] landing.zest.is - Zest - Zest, 21.9.2022.
[62] saleshunter.com - SalesHunter - SalesHunter, 21.9.2022.
[63] wavemakerglobal.com - Wavemaker - Wavemaker, 21.9.2022.
[64] cyrilconton.com - Cyril Conton - Portfolio, 21.9.2022.
[65] novolume.co.uk/gofreelance - Dave Ellis - Go Freelance, 21.9.2022.
[66] ooki.zajno.com - Ooki - Ooki, 21.9.2022.
[67] skysoft.tech/en - SkySoft - SkySoft, 21.9.2022.
[68] sparky.us - Sparky - Sparky, 21.9.2022.
[69] investmentcalculator.io - Investment Calculator - Investment Calculator, 21.9.2022.
[70] nycfoodiverse.com - Will Su - NYC Foodiverse, 21.9.2022.
[71] hellosunapp.com - Small Multiples - Hello, Sun., 21.9.2022.
[72] ghwk.de/statisticsandcatastrophe - House of the Wannsee Conference -Questioning Eichmann's numbers, 21.9.2022.
[73] dataexplorer.womenwill.com/intl/en/thedivide - Women Will - The Divide, 21.9.2022.
[74] css-tricks.com/when-to-use-svg-vs-when-to-use-canvas/ - CSS-Tricks - When to Use SVG vs. When to Use Canvas, 21.9.2022.
[75] paperjs.org/about/ - "Paper.js
[76] khronos.org/news/press/khronos-releases-final-webgl-1.0-specificationa - Khronos Group - Khronos Releases Final WebGL 1.0 Specification, 21.9.2022.
[77] lottiefiles.com/blog/engineering/what-why-how-to-convert-files-from-svg-to-lottie - LottieFiles - SVG to Lottie: the what, the why, and the how, 21.9.2022.