Izrada mrežnih animacija: CSS ili JavaScript?

Ledinščak, Iva

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
Ak. god. 2020./ 2021.

Iva Ledinščak

IZRADA MREŽNIH ANIMACIJA: CSS ILI JAVASCRIPT?

Završni rad

Mentor: dr.sc. Kristina Kocijan, izv. prof.

Zagreb 2021.

Izjava o akademskoj čestitosti

Izjavljujem i svojim potpisom potvrđujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

_____

(potpis)

Ovaj rad posvećujem svojoj obitelji, koja mi je velika podrška i uvijek su uz mene, mentorici Kristini Kocijan, kojoj zahvaljujem na strpljenju i pomoći te ekipi ispred knjižnice, koja nestrpljivo čeka da se tamo ponovno okupi.

Sadržaj

1. Uvod

Animacija je stvaranje iluzije o pokretu koji se ostvaruje nizom 2D ili 3D, uzastopno snimljenih fotografija ili crteža (okvira). Norman McLaren, osnivač odsjeka za animaciju na Nacionalnom filmskom odboru Kanade (engl. *National Film Board of Canada*), animaciju je opisao kao umjetnost upravljanja nevidljivim međuprostorima koji se nalaze između tih okvira (Furniss, 2007).

Animacije imaju široku primjenu u svijetu zabave, ali i informiranju ljudi. Tako se one pojavljuju doslovno na svakom koraku: na mreži (engl. *World Wide Web*), u reklamama, filmovima, televizijskim serijama, mobilnim aplikacijama… Osim toga, digitalna animacija se koristi i u vojsci, svemirskim letjelicama, medicini, forenzici, arhitekturi, obrazovanju, likovnoj umjetnosti… (Furniss, 2007)

Pojavom računala omogućen je između ostalog i razvoj animacije koja danas izgleda znatno drukčije od one prije njegove pojave, kada se svaka sličica unutar svake sekunde animacije crtala ručno. U današnje vrijeme animatori mogu birati između više vrsta računalnih programa u kojima lako mogu oponašati upotrebu olovke, ugljena, pastela, vodenih boja ili nekog drugog medija, bez potrebe za ponovnom izradom cijele sličice ukoliko je došlo do sitne pogreške.

Jedan od ciljeva animacije na mrežnim stranicama je privući pažnju promatrača na željeni dio ekrana u pravom trenutku (Whitaker et al., 2009). Međutim, animatori se koriste raznim trikovima kako animacijama ne bi previše odvukli posjetitelja stranice od glavnog sadržaja. Osim za privlačenje pažnje, animacije se mogu koristiti za interakciju ili zabavu, primjerice za vrijeme čekanja učitavanja stranice.

Tema ovog završnog rada je izrada mrežnih animacija CSS-om i JavaScript-om te usporedba postupaka i rezultata korištenja spomenutih tehnologija. Završni rad opisuje istraživanje potaknuto pitanjem: koja tehnologija, CSS ili JavaScript, bolje odgovara izradi animacije u mrežnom prostoru.

U sljedećem poglavlju je izložen kratki povijesni razvoj animacija i tehnologija korištenih za njihovu izradu. U 3. poglavlju su mrežne animacije definirane i detaljnije opisane. Navedene su vrste animacija i njihove razlike. U 4. poglavlju je objašnjen postupak izrade jednostavne animacije koristeći CSS, popraćen fotografijama kôda pisanog za animaciju te fotografijama kadrova iste animacije. U 5. poglavlju je objašnjen postupak izrade iste jednostavne animacije, ali korištenjem JavaScript-a. Poglavlju su također priložene fotografije kôda i kadrova izrađene animacije. U zaključku je rezimirana spoznaja dobivena analizom različitih metoda animiranja, navedene su razlike i prednosti korištenja obje predložene tehnologije za izradu mrežne animacije.

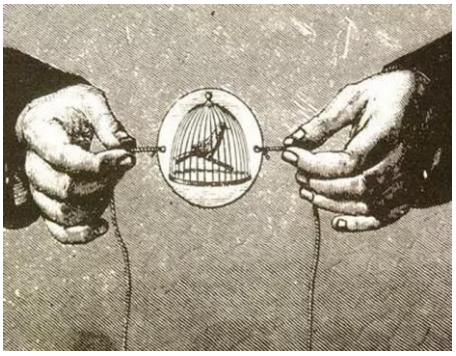
2. Povijest animacija

Za shvaćanje osnovnog koncepta animacije, potrebno je „vratiti se na početak“ i shvatiti korake u razvoju tehnologija koji su doveli do naprednih animacija.

Takozvani „kinematografski preteče“ pojavili su se davno prije prikazivanja pokreta na platnu. Sredinom 17. st., počele su se koristiti oslikane staklene pločice koje su se umetale u ranu vrstu projektora (lat. *laterna magica*) (Furniss, 2007).

Slika 1: Laterna Magica.

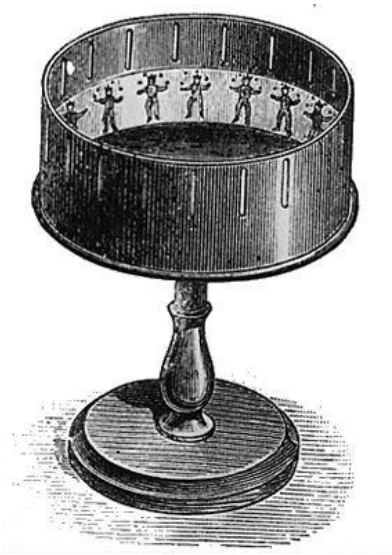
U 19. st. zabilježena je pojava taumatropa (slika 2), kartonske pločice s crtežom sa svake strane koja se okreće pomoću vrpce čime se stvara iluzija pokreta između dvije sličice na pločici u sredini (Furniss, 2007). Zatim fenakistoskop (slika 3), kartonski disk oko čijeg je središta radijalni niz sličica koje prikazuju faze jednog cjelovitog pokreta. Ukoliko se disk zavrti, stvara se iluzija pokreta sličica (Wells, 1998).



Slika 2: Taumatrop.

Slika 3: Fenakistoskop.

Nakon spomenutih „optičkih igračaka“, 1832. godine izumljen je stroboskop, prvi izum koji je prikazivao iluziju slike u pokretu (Whitaker et al., 2009). Odrazi slika na okrećućem disku gledali su se kroz proreze oko ruba diska kako bi se niz statičnih slika brzo nastavljao, jedna slika za drugom (Whitaker et al., 2009). Zoetrop (slika 4) je koristio sličan princip: u njemu su se trake papira promatrale kroz proreze na rotirajućem cilindru (Whitaker et al., 2009).



Slika 4: Zoetrop.

Slika 5: Cinématographe.

Zahvaljujući braći Lumière i njihovom izumu prve kamere (franc. *Cinématographe*) (slika 5), prva snimka slike u pokretu javnosti je prikazana 1895. godine, nakon čega je umijeće stvaranja slika u pokretu doživjelo rast i počelo se pretvarati u veliku industriju (Furniss, 2007).

Nedugo nakon pojave prvih filmova, pojavili su se i animirani filmovi. U ranim godinama animacijske produkcije, animirane filmove su proizvodile male grupe ljudi (Furniss, 2016). Kako je filmska industrija rasla, produkcija je postajala sve kompleksnija, direktorima i drugim umjetnicima bilo je sve teže održavati kreativnu kontrolu nad svojim radom (Furniss, 2007).

Novi izumi koji su donijeli uštedu vremena i novca: nove kamere, uljne boje za pozadinu umjesto vodenih, nove metode osvjetljenja, proces refleksije koji smanjuje broj crtanih okvira filma; pojavili su se 40-ih godina 20. stoljeća (Furniss, 2007).

Krajem 20. st., procijenjeno je da otprilike 75 % igranih filmova koristi neku vrstu digitalno stvorenog efekta ili animacije (Furniss, 2016). U godinama koje su uslijedile nakon toga, digitalne tehnologije su se još više inkorporirale u produkciju ne samo igranih filmova, nego i potpuno animiranih snimaka, televizijskih serija i dr. (Furniss, 2007).

Sredinom osamdesetih godina prošlog stoljeća, svijet animacije se počeo naglo mijenjati. S razvojem digitalne tehnologije, u animacijskoj produkciji su se umjesto papira počeli pojavljivati drugi mediji (Furniss, 2007). Tim Berners-Lee je 1989. godine izradio prvu mrežnu stranicu i internetski preglednik koji omogućava pristup stranicama na Internetu. Radio je na tome da svakome omogući besplatan pristup mreži (Clarke, n.d.).

U početku, računala nisu bila tako usmjerena korisnicima, zbog čega su se umjetnici teško snalazili na njima (Furniss, 2016). Kako je tehnologija danas prilagodljivija i lakše ju je koristiti te pruža nebrojene mogućnosti, nije ni čudo da su animatori odlučili papir i olovku trajno zamijeniti monitorom i mišem.

Računala su postala središte produkcije proizvodnje animacije (Furniss, 2007). Nakon računala, pojavili su se grafički tableti, uređaji za praćenje pokreta i mnoga druga oprema koja olakšava proces animiranja. Softveri za kreiranje mrežnih stranica i animacija svake se godine mijenjaju i nadopunjuju novim mogućnostima, tako da je izbor tehnologija za izradu animacija sve veći.

3. Mrežne animacije

Točka nacrtana olovkom se može smatrati najmanjom slikom u crtanoj animaciji, dok je piksel najmanja komponenta slike na ekranu, pa tako i slike u pokretu, odnosno animacije (Furniss, 2007). Fizički broj piksela zaslona je broj piksela smještenih po širini zaslona (Gasston, 2013). Različiti uređaji imaju različit broj piksela pa im je tako i rezolucija različita. Neće svaki uređaj jednako prikazati slike i druge objekte (Gasston, 2013). Animacije općenito mogu biti dvodimenzionalne (crtanje, slikanje) i trodimenzionalne (stop-animacija). Obje vrste se mogu naći u digitalnoj, mrežnoj produkciji, ali 2D animacije su razvijenije i zastupljenije (Furniss, 2007). Neće svaki monitor jednako prikazati animacije s različitih internetskih preglednika. Prikazi će se možda razlikovati po veličini, rezoluciji i boji (Furniss, 2007).

Mrežne stranice su u osnovi niz HTML dokumenata koji se učitavaju i prikazuju na zaslonu računala, tableta ili mobitela putem internetskog preglednika. Razlikuju se prema sadržaju koji prikazuju, svojoj namjeni i načinu na koji su kreirane. Ovisno o sadržaju na stranici i ciljanoj publici, stranice na mreži mogu sadržavati animacije. Kao što je ranije navedeno, mrežne animacije mogu imati ulogu zaokupljanja posjetitelja stranice, usmjeravajući njegovu pažnju na određeni dio stranice prikazan na ekranu (Whitaker et al., 2009).

U kratkoj povijesti Interneta, web dizajneri su imali nekoliko opcija za dodavanje animacija svojim mrežnim stranicama (McFarland & Sawyer, 2013). Prije CSS-a, za bilo koji oblik animacije bilo je potrebno koristiti animirani GIF ili Flash (Powers, 2010). U početku je dosta popularan bio animirani GIF, više okvira koji stvaraju iluziju kretanja lika, a izvode se u petlji (McFarland & Sawyer, 2013). Adobe Flash program je omogućavao kreiranje kompleksnijih animacija, igrica i mrežnih aplikacija, ali nije ga bilo jednostavno naučiti te nije mogao ''komunicirati'' s HTML-om na stranici, sa slikama, naslovima i paragrafima koji čine većinu sadržaja (Powell, 2001). GIF i Flash animacije

postepeno su zamijenile tehnologije usklađene sa standardom HTML5, a od 31. prosinca 2020. godine, Flash više nije podržan (Adobe, 2021).

Prilagodljivi web dizajn omogućava mrežnoj stranici da funkcionira na različitim uređajima i Internet preglednicima (Frain, 2014). Za termin prilagodljivog web dizajna zaslužan je Ethan Marcotte (Frain, 2014). Marcotte je ujedinio fleksibilan grid raspored, slike i medije i medijske upite u jedinstveni pristup i nazvao ga prilagodljiv web dizajn (Frain, 2014). Ulogu u kreiranju prilagodljivog web dizajna imaju HTML5 i CSS3 (Frain, 2014).

HTML (engl. *Hypertext Markup Language*) je standardni jezik za označavanje koji se prvenstveno koristi za izradu mrežnih stranica. Riječ je o još jednoj u nizu tehnologija za koje je zaslužan Tim Berners-Lee. HTML pruža temelje svakoj stranici na World Wide Web-u (Thomas A. Powell, 2001). On koristi jednostavne oznake za definiranje različitih dijelova mrežne stranice, odnosno upute koje pregledniku govore kako da prikaže mrežnu stranicu (McFarland & Sawyer, 2013). Sve mrežne stranice počinju doctype-om (Frain, 2014). Riječ je o liniji kôda koja mora biti napisana kao prva linija kôda i identificira inačicu HTML-a koja se koristila za pisanje stranice (Gasston, 2013). Bez njega, stranica bi u svakom pregledniku izgledala drugačije. Oznaka meta deklarira raspon Unicode znakova upotrijebljenih za renderiranje teksta na stranici – UTF-8 je podrazumijevani raspon koji se koristi na najvećem dijelu Weba (Gasston, 2013), a uključuje i znakove hrvatske abecede. HTML5 stavlja naglasak na pojednostavljivanje kôda potrebnog za kreiranje stranice koja odgovara W3C standardima i povezuje CSS, JavaScript i grafičke datoteke (Frain, 2014).

CSS3 pruža opciju pomicanja, transformiranja i animiranja bilo kojeg elementa HTML-a na stranici bez potrebe za korištenjem drugih alata i tehnologija. CSS3 može transformirati elemente mrežne stranice tako da ih rotira, skalira, pomiče ili izobliči po horizontalnoj ili vertikalnoj osi elementa (engl. *skewing*) (McFarland & Sawyer, 2013).

JavaScript također omogućuje animiranje svih elemenata koji se nalaze na stranici, samo što je potrebno uložiti više vremena u učenje tog programskog jezika (McFarland

& Sawyer, 2013). Dakle, mrežne animacije na stranicama je moguće prikazati koristeći HTML u kombinaciji sa CSS-om i/ili JavaScript-om, koji su detaljnije objašnjeni u poglavljima koja slijede.

3.1. CSS

Dobar dizajn poboljšava i naglašava poruku mrežne stranice, pomaže posjetiteljima pronaći što traže i određuje kako ju oni doživljavaju (McFarland & Sawyer, 2013). CSS (engl. *Cascading Style Sheets*) pruža kreativnu kontrolu nad rasporedom elemenata i ukupnim dizajnom mrežnih stranica (McFarland & Sawyer, 2013). Stranice se mogu ukrasiti naslovima primamljivih boja i fontova, slike se mogu precizno rasporediti na željena mjesta, poveznice je moguće naglasiti dinamičnim efektima, određene elemente potamniti ili izblijediti, pomicati objekte stranicom ili namjestiti da gumb mijenja boju kada posjetitelj mišem prijeđe preko ili klikne na njega (McFarland & Sawyer, 2013). Osim tih jednostavnih animacija, CSS-om se mogu kreirati i one malo složenije.

CSS ne može ništa bez HTML-a (McFarland & Sawyer, 2013). U CSS-u, formatiranje teksta izvršava se stilom (engl. *style*). Stil je pravilo opisivanja izgleda određenog dijela stranice. Stranica sa stilom (engl. *Style sheet)* u CSS-u označava skup stilova koji se koriste za uređivanje prikaza mrežne stranice (Meyer, 2007). Uporabom CSS stilova moguće je odvojiti strukturu podataka od načina njenog prezentiranja, zatim definirati izgled elemenata stranice bez potrebe vanjskih aplikacija, dodataka i upotrebe velikog broja slika (Powers, 2010). CSS stil se može dodati HTML dokumentu na tri načina. Tako postoji umetnuti, unutarnji i vanjski CSS (W3C, n.d.). Umetnuti CSS nalazi se direktno unutar oznake nekog HTML elementa kao atribut style kojem je dodana jedna ili više vrijednosti (W3C, n.d.). Unutarnji CSS se nalazi u zaglavlju HTML dokumenta unutar oznake <style></style>. Vanjski CSS je posebna datoteka sa skupom stilova koji se primjenjuju na stranicu, a svaka stranica koja sadrži taj stil unutar zaglavlja mora imati poveznicu u oznaci <link> (W3C, n.d.). Vanjskim CSS-om su svi stilovi sakupljeni na jednom mjestu. Svaka promjena učinjena u toj datoteci se odmah primjenjuje na

stranici. Ideja CSS-a je pojednostaviti postupak uređivanja mrežnih stranica, a to je najlakše (osim toga je i tehnički idealno) ostvariti korištenjem vanjske CSS datoteke (Meyer, 2007). Nema potrebe za uređivanjem atributa svakog elementa HTML dokumenta zasebno.

CSS3 pruža mogućnost transformiranja, pomicanja (tranzicije) i animiranja bilo kojeg elementa HTML-a na stranici (McFarland & Sawyer, 2013). U nastavku će biti više riječi o svakom pojedinom modelu.

3.1.1. Transformacije

CSS3 može transformirati elemente mrežne stranice tako da ih rotira (engl. *rotate*), pomiče (engl. *translate*), skalira (engl. *scale*) ili izobliči (engl. *skewing*) po njihovoj horizontalnoj ili vertikalnoj osi (McFarland & Sawyer, 2013). Svojstvo se koristi tako da mu se funkcijom zada željeni tip transformacije i vrijednost koja ukazuje na to koliko se element mijenja. Korištenjem svojstva transform može se napraviti primjerice lagani nagib elementa, rotacija navigacijske trake ili povećanje dimenzija slike kada posjetitelj mišem prijeđe preko nje (McFarland & Sawyer, 2013). Za složenije vizualne efekte moguće je kombinirati više transformacija.

Koristeći funkciju rotate može se zakrenuti element (bez da mu se promijeni oblik) i odrediti broj stupnjeva za koji će se rotirati (Frain, 2014).

Druga funkcija koju svojstvo transform može primiti je funkcija translate (Frain, 2014). Ona jednostavno pomiče element s njegovog trenutnog položaja, lijevo ili desno i gore ili dolje. Funkcija translate uzima dvije vrijednosti: prva određuje horizontalni pokret i druga određuje vertikalni pokret (McFarland & Sawyer, 2013). Kada se neki element pomiče funkcijom translate, preglednik ostavlja prazan prostor tamo gdje bi se oznaka normalno pojavila te postavi element na novi položaj određen funkcijom (McFarland & Sawyer, 2013). Sama po sebi, funkcija nije od velike koristi, ali može dobro doći kada se žele ostvariti suptilni pokreti kao odgovor na klik ili prelazak miša preko nekog elementa (McFarland & Sawyer, 2013).

Funkcije translateX i translateY koriste se za pomicanje elementa samo po osi X (translateX) ili samo po osi Y (translateY) (McFarland & Sawyer, 2013). Funkcija translate se može koristiti s CSS tranzicijama, kako bi se animirao pokret elementa koji vidno putuje ekranom (McFarland & Sawyer, 2013).

Promjena veličine elemenata moguća je jednostavnom promjenom vrijednosti CSS width i height za slike ili promjenom vrijednosti font-size za tekst, ali može se ostvariti i funkcijom scale. Koristeći funkciju scale može se mijenjati veličina elementa (McFarland & Sawyer, 2013).

Broj u zagradi označava koliko puta se trenutna veličina elementa povećava (npr. vrijednost 1 ga ostavlja u izvornoj veličini, .5 ga smanjuje za pola, 2 ga poveća 2 puta) (McFarland & Sawyer, 2013). Najčešća upotreba funkcije scale je kod ostvarivanja dinamične vizualne promjene elementa na stranici. Širina i visina se mogu skalirati odvojeno, tako da se vrijednosti u zagradi navedu odvojene zarezom (McFarland & Sawyer, 2013).

Funkcije scaleX i scaleY koriste se za promjenu samo visine (scaleY) ili samo širine (scaleX) elementa. Korištenjem negativnih brojeva, element se može zrcalno preslikati po osi X ili Y, ili obje ako je tako zadano u zagradi (McFarland & Sawyer, 2013).

Iskrivljavanje elementa (engl. *skewing*) dopušta njegov nagib po horizontalnoj ili vertikalnoj osi, što mu može pružiti dojam trodimenzionalnosti (McFarland & Sawyer, 2013).

Funkcije skewX i skewY koriste se za promjenu elementa po svakoj osi zasebno (McFarland & Sawyer, 2013). Moguće su i višestruke transformacije. Dizajneri nisu ograničeni samo jednom transformacijom. Sliku se može i skalirati i iskriviti, rotirati i translatirati. Redoslijed funkcija određuje kojim redom će preglednik izvršavati efekte. Primjerice, ako je rotate naveden prije scale i translate, preglednik će ga prvo zakrenuti za zadanu vrijednost, potom povećati i pomaknuti (McFarland & Sawyer, 2013).

CSS transformacije ne utječu na elemente oko njih. Ako se neki element rotira za primjerice 45 stupnjeva, možda će prelaziti preko elemenata koji se nalaze oko njega.

Dakle, internetski preglednik, uz taj rotirani element, zadrži izgled stranice onako kako bi elementi bili prikazani kada element ne bi bio rotiran (McFarland & Sawyer, 2013).

3.1.2. Tranzicije

Stranica može dodatno oživjeti ako se transformacije kombiniraju s tranzicijama. Transformacije određuju što će element postati, a tranzicije ublažavaju promjenu iz jednog stanja u drugo (Frain, 2014). Tranzicija je jednostavno animacija od jednog skupa CSS svojstava do drugog kroz određeno vrijeme. Npr. može se odrediti da se neki element zakrene za 360 stupnjeva kroz 2 sekunde. Dodavanjem tranzicije nekoj slici, može se dobiti trenutna, jednostavna animacija (McFarland & Sawyer, 2013).

Za ostvarivanje tranzicije CSS-om potrebna su dva stila (početni i konačni), svojstvo transition i okidač (engl. *trigger*). Jedan stil predstavlja početni izgled elementa, dok drugi predstavlja konačni. Preglednik se brine o procesu animiranja promjena (npr. promjena boje) (McFarland & Sawyer, 2013).

Svojstvo transition omogućuje animaciju. U osnovi, svojstvo transition se primjenjuje na početni stil, stil koji definira izgled elementa prije početka animacije (McFarland & Sawyer, 2013).

Okidač uzrokuje promjenu između dva stila. S CSS-om, može se koristiti više pseudo-klasa koje okidaju animaciju. Najčešća je pseudo-klasa :hover, kojom se može animirati promjena između uobičajenog izgleda elementa i onoga kako on izgleda kada posjetitelj mišem prijeđe preko njega (McFarland & Sawyer, 2013).

Kada okidač više nije u primjeni (primjerice kada miš posjetitelja više nije iznad navigacijskog gumba), preglednik vraća oznaku na prethodni stil i animira cijeli proces (McFarland & Sawyer, 2013). Drugim riječima, potrebno je jednom postaviti tranziciju na element i preglednik se dalje brine o animiranju iz jednog stila do drugog i natrag u početni. Internetski preglednici ne mogu animirati sva CSS svojstva, ali mnogo njih mogu. Uz četiri vrste transformacija, moguće je animirati svojstva height, width, opacity, color, background-color, border-color, border-width, margin, padding, font-

size, letter-spacing, word-spacing, line-height te svojstva položaja: top, left, right i bottom (McFarland & Sawyer, 2013).

U CSS-u je moguće kontrolirati trajanje animacije, tip korištene animacije i odgodu početka animacije (engl. *optional delay*; transition-delay) (McFarland & Sawyer, 2013). Može se specificirati samo jedno svojstvo, ili koristiti ključna riječ all za animiranje svih svojstava koja se mijenjaju ili koristiti zarezom odvojenu listu za specificiranje više od jednog svojstva (McFarland & Sawyer, 2013).

Kako bi se odredilo vrijeme potrebno da se animacija izvrši, koristi se transition-duration. Vrijednost se može izraziti u sekundama ili milisekundama (McFarland & Sawyer, 2013).

Brzina animacije se može kontrolirati svojstvom transition-timing-function. Animacija može krenuti sporo i onda može brzinski završiti, npr. boja pozadine se jedva primjetno mijenja na početku i potom naglo izvrši promjenu boje. Postoji pet varijacija kojima se ovo svojstvo može izraziti: linear, ease, ease-in, ease-out i ease-in-out (McFarland & Sawyer, 2013). Ako se ne odredi vrijeme funkcije, preglednik koristi opciju ease, koja animaciju započinje polako, ubrzava u sredini i usporava na kraju. Opcija linear omogućava ravnomjernu promjenu za vrijeme cijelog trajanja animacije (McFarland & Sawyer, 2013). Za svojstvo transition-timing-function može se koristiti vrijednost cubic-bezier. Određivanjem dviju kontrolnih točaka moguće je kontrolirati put i zakrivljenost linije. Linija će biti strmija što je animacija brža. Linija ne mora biti jednolika. Određivanjem više točaka može se odrediti linija koja je primjerice strma na početku animacije, zatim se izravna na sredini kada animacija uspori i potom nastavlja strmi rast kako se animacija bliži kraju (McFarland & Sawyer, 2013). Ovakva linija može biti opisana sljedećom linijom kôda:

```
transition-timing-function: cubic-bezier(.20, .96, .74, .07);
```

3.1.3. Animacije

CSS animacije omogućuju složenije animirane efekte od transformacija i tranzicija (McFarland & Sawyer, 2013). Za razliku od tranzicija koje omogućavaju animiranje samo iz jednog skupa svojstava u drugo, CSS3 animacije omogućavaju animiranje iz jednog skupa svojstava u drugo, u treće, četvrto itd. (McFarland & Sawyer, 2013). Malo su složenije od tranzicija, ali prednost im je to što ne trebaju nužno okidač za početak (McFarland & Sawyer, 2013). Mogu se ponavljati, zaustaviti kada posjetitelj mišem prijeđe preko njih i čak ići unatrag kada dođu do kraja. Mogu se postaviti u vrijeme učitavanja stranice. Taj efekt dopušta privlačenje pažnje na logo ili neki drugi dio stranice (McFarland & Sawyer, 2013).

Prvi korak u kreiranju CSS animacije je kreiranje skupa okvira (engl. *keyframes*) (McFarland & Sawyer, 2013). U animaciji, *keyframe* je jedan okvir animacije koji govori kako će prikaz izgledati. Dodavanjem drugog okvira može se definirati završna točka animacije (McFarland & Sawyer, 2013).

Internetski preglednik ostvaruje animaciju između dva okvira crtajući međukorake, primjerice stvara pokret figurice s jednog šahovskog polja na drugo (McFarland & Sawyer, 2013). Tranzicije koriste sličnu ideju. U tranziciji se definiraju dva stila i pregledniku se dopušta da animira promjenu iz jednog stila u drugi. Na neki način, stilove možemo promatrati kao okvire (McFarland & Sawyer, 2013).

Kako CSS3 animacije dopuštaju definiranje više od dva okvira, moguće je kreiranje složenijih animiranih efekata (McFarland & Sawyer, 2013).

Iza `@keyframe` dolazi `animationName`, odnosno ime animacije (McFarland & Sawyer, 2013). Ono se kasnije koristi kada se animacija primjenjuje na element na stranici, tako da bi naziv trebao opisivati što animacija radi, npr. „`fadeOut`“ (McFarland & Sawyer, 2013).

Za uspješnu animaciju potrebno je dodati najmanje dva okvira. Za više okvira, koriste se vrijednosti u postocima. Postotak predstavlja gdje se u ukupnoj dužini animacije događa promjena (McFarland & Sawyer, 2013).

CSS3 pruža mnoga svojstva kojima se može kontrolirati kako i kada se animacija odvija (McFarland & Sawyer, 2013). Najjednostavnije, da bi se animacija pokrenula, potrebno je upisati njeno ime i trajanje. Ime animacije jednostavno govori pregledniku koju animaciju da pokrene. S animation-duration postavlja se vrijeme potrebno animaciji za njeno izvršavanje od početka do kraja (McFarland & Sawyer, 2013).

Kao i s tranzicijama, moguće je postaviti određeni tip vremenske funkcije za kontrolu brzine animacije kroz njeno trajanje (McFarland & Sawyer, 2013). Animacija može krenuti polako i završiti brzo, koristeći vrijednost cubic-bezier ili neku od ugrađenih metoda: linear, ease, ease-in, ease-out, ease-in-out, koje rade jednako kao i svojstvo transition-timing-function. Može se koristiti za kontroliranje cijele animacije ili samo određenih okvira (McFarland & Sawyer, 2013).

Nakon definiranja skupa okvira, animaciju je potrebno jedino pridružiti nekom elementu (ili više njih) na stranici kako bi radila. Jedna animacija može biti pridružena više elemenata te jednom elementu može biti pridruženo više animacija. Nakon što je animacija definirana, moguće ju je koristiti neodređeni broj puta u neodređenom broju stilova za bilo koji element na stranici (McFarland & Sawyer, 2013). Animacija će se pokrenuti u trenutku kad se stranica učita. Animacijom se primjerice može približiti (engl. *zoom*) logo ili drugi određeni prostor na stranici kojem posjetitelj treba obratiti pozornost (McFarland & Sawyer, 2013). Animacija se također može primijeniti jednoj od pseudo-klasa poput :hover, :active, :target, :focus kako bi se animacija pokrenula pokretom miša preko elementa kojem je primijenjena (McFarland & Sawyer, 2013). Za jednake animacije moguće je koristiti JavaScript, koji ostvaruje i dinamičnu primjenu određene klase kao rezultat klika mišem na gumb ili neki drugi element na stranici (McFarland & Sawyer, 2013).

3.2. JavaScript

Važno je da mrežne stranice imaju dobru strukturu i oblikovanje, ali moraju biti i funkcionalne (Gasston, 2013). Kada bi mrežne stranice bile pisane samo u HTML-u, čak i

uz dodatak CSS-a, jedina bi im svrha bila prikazati informacije posjetiteljima, uz klik miša na link na drugu stranicu kao jedinu interaktivnost. Pojava kaskadnih opisa stilova je bila ključni element za razvoj dinamičke mrežne stranice izrađene putem JavaScript-a (Powers, 2010).

JavaScript je programski jezik koji je bio zamišljen kao skriptno sučelje između mrežne stranice učitane u klijentski preglednik i aplikacije na poslužitelju (Powers, 2010). On pruža razne mogućnosti poput pisanja funkcija, dinamičkih objekata, deklariranja varijabli bez tipa itd. (Crockford, 2008). Njime se HTML može obogatiti interaktivnošću, dinamičnim vizualnim efektima i animacijama, može dati trenutne povratne informacije putem obrazaca, Internet kupovine itd. Osim toga, može kreirati poruku pogreške ako korisnik na primjer, pokuša predati obrazac kojem nedostaju potrebne informacije. Događaji u JavaScript-u moraju biti pridruženi elementima kako bi se pokrenuli na mrežnoj stranici (Powers, 2010). Takvi događaji se pokreću kada se na mrežnoj stranici pokrene neka aktivnost, npr. kada korisnik pritisne gumb, klikne na link ili kada se stranica učita (Powers, 2010). Brzina je također jedna od dobrih strana JavaScript-a. Zato što ne ovisi o konstantnom i ponovnom učitavanju stranica, JavaScript omogućava mrežnim stranicama da se 'ponašaju' kao aplikacije.

JavaScript je u 10 dana izumio Brendan Eich na Netscape-u 1995. godine. U ranim verzijama JavaScript-a, programeri su mogli manipulirati nekim sadržajima mrežnih dokumenata poput slika i obrazaca (Keith, 2005). Od tada je postao ključna komponenta razvoja za Web (Powers, 2010). On je važan programski jezik jer je to jezik internetskih preglednika (Crockford, 2008). Njegove skripte se lako dodaju mrežnoj stranici te radi u većini preglednika (Chrome (Dale, 2016), Firefox, Internet Explorer, Opera, Safari) (Powers, 2010). JavaScript se prvenstveno koristi za razvoj mrežnih stranica, a tek do nedavno je bio vezan isključivo uz internetske preglednike (Dale, 2016). Dakle, JavaScript se ne koristi samo na mrežnim stranicama. Tim jezikom je moguće kreirati Yahoo! Widget-e i Google aplikacije, pisati programe za iPhone i mnoge Adobe programe kao što su Acrobat, Photoshop, Illustrator i Dreamweaver.

JavaScript se dodaje oznakom <script></script> u HTML dokumente, u njegovo zaglavlje ili u tijelo dokumenta (Powers, 2010). Preporuča se korištenje JavaScript datoteka i navođenje veza do njih u mrežnoj stranici. Takav kôd je jednostavnije mijenjati jer je na jednom mjestu, datoteka je povezana sa svim stranicama gdje se kôd mora izvršavati (Powers, 2010). Preglednik učitava datoteke skripti redoslijedom kojim se pojavljuju na stranici (Powers, 2010).

JavaScript funkcije su ključni dio jezika (Powers, 2010). One su blokovi kôda kreirani tako da izvršavaju neki određeni zadatak (W3C, n.d.). Mogu se dodijeliti varijabli ili proslijediti kao argument pozivu neke druge funkcije (Dale, 2016). Funkcija se definira sljedećim kôdom:

```
function name(parametar1, parametar2) {
 …
}
```

Prvo se navede ključna riječ function, zatim nakon nje slijedi ime funkcije, najbolje tako da opisuje njezino djelovanje, zagrade koje sadrže nula ili više parametara/argumenata i na kraju tijelo funkcije, obavezno unutar vitičastih zagrada. Funkcije komuniciraju s programom koji ih poziva putem argumenata koji su im proslijeđeni i vrijednosti koje vraćaju (Powers, 2010). Unutar funkcije, argumenti se ponašaju kao lokalne varijable (W3C, n.d.). Objekti su funkciji proslijeđeni prema referenci. Promjene izvedene na objektu u funkciji odražavaju se na program koji funkciju poziva (Powers, 2010). Objekt je spremnik svojstava, a svojstva imaju naziv i vrijednost (Crockford, 2008).

U JavaScript-u, varijable su nazivi za spremnike podataka, odnosno način stvaranja reference na njih kako bi im se kasnije moglo pristupiti te imaju ulogu u prijenosu podataka između različitih procesa (Powers, 2010). Podaci mogu biti bilo kojeg tipa, niz znakova, logička vrijednost, numerička, ili bilo koji drugi tip podataka. Tip podataka određuje kako ga skriptni mehanizam JavaScript-a interpretira. Varijable imaju identifikator, doseg i specifičan tip podataka. Važno je da varijabla sadrži ključnu riječ var jer je u suprotnom „maskirana“ ili može „maskirati“ neku drugu varijablu istog imena. Drugim riječima, definiranjem varijable te dodjeljivanjem identifikatora u okviru

atributa „namespace“ osigurava se da nijedan objekt nema isti identifikator (Dale, 2016).

Za razliku od animacija u CSS-u, koje se mogu podijeliti na transformacije, tranzicije i animacije, funkcije koje izvršavaju animacije elemenata napisane u JavaScript-u nisu grupirane u jedinstvenu podjelu. JavaScript dopušta izvršavanje funkcija u intervalima, što znači da može utjecati na stil elemenata u tijeku vremena (Jeremy Keith, 2005). Primjer za ovakve promjene su upravo animacije (Jeremy Keith, 2005).

JavaScript-om je moguće pisanje brojnih funkcija kojim se ostvaruje pojavljivanje i nestajanje elemenata na stranici (McFarland & Sawyer, 2014). Kada se koristi efekt za nestajanje nekog elementa, on zapravo nije uklonjen sa stranice, nego još uvijek postoji u DOM-u (engl. *Document Object Model*) (McFarland & Sawyer, 2014). DOM je, kad se svede na nekoliko riječi, način konceptualizacije sadržaja dokumenta (Jeremy Keith, 2005). Njime se objedinjuju HTML, CSS i JavaScript (Jeremy Keith, 2005). Dakle, kôd elementa je i dalje u memoriji preglednika, samo što mu je display postavljen na none (McFarland & Sawyer, 2014). Zbog toga element ne zauzima vizualni prostor na ekranu i tako ostali sadržaj stranice može doći na mjesto skrivenog elementa (McFarland & Sawyer, 2014).

U početku je pisanje JavaScript kôda bilo vrlo složeno. Samo jedna funkcija je trebala biti napisana u više redaka. Zato su razvijene JavaScript biblioteke, kolekcije napisanih skripti koje pojednostavljuju teške zadatke i rješavaju probleme. S korištenjem biblioteka, kôd za funkcije može biti znatno kraći, a one se i dalje jednako izvršavaju u svim preglednicima (Gasston, 2013). Danas postoji velik izbor JavaScript biblioteka, a neke od njih su jQuery, GSAP, Angular, Babel, Parsley, ReactJS, Slick, TweenJS i mnoge druge. S obzirom na često korištenje biblioteka, začudno je da JavaScript ne posjeduje mehanizam za njihovu upotrebu (Dale, 2016).

Osim funkcija jednostavnog prikazivanja/skrivanja, biblioteka jQuery pruža funkcije fadeIn(), fadeOut(), fadeToggle() i fadeTo() kojima se može mijenjati prozirnost

elemenata, a dodavanjem vremenskog trajanja, ove funkcije mogu ostvariti jednostavne animacije (McFarland & Sawyer, 2014).

Funkcija fadeIn(), postepeno prikazuje element na stranici normalnom brzinom od 400ms ukoliko drugačija vrijednost nije zadana (McFarland & Sawyer, 2014).

Funkcija fadeOut() sakriva vidljiv element tako da se on postepeno gubi iz vida, odnosno 'blijedi'. Za sporije nestajanje, potrebno je upisati vrijednost u zagradu (McFarland & Sawyer, 2014).

Funkcija fadeToggle(), ovisno o trenutnoj vidljivosti elementa, može postepeno sakriti ili prikazati element. Brzina te radnje također ovisi o zadanoj vrijednosti (McFarland & Sawyer, 2014).

Funkcija fadeTO() postavlja određenu prozirnost na element (McFarland & Sawyer, 2014).

Bez primjetne pauze u izvođenju, funkcije se izvršavaju jedna za drugom (Jeremy Keith, 2005). Za kreiranje animacije, potrebno je kreirati tzv. „odgode“ (engl. *delays*) (Jeremy Keith, 2005). Izvršavanje određene funkcije nakon određenog vremena moguće je JavaScript funkcijom setTimeout (Jeremy Keith, 2005). Prvi argument funkcije je željena funkcija tipa niza znakova (engl. *string*), a drugi je broj milisekundi koje će proći prije izvršavanja funkcije (Jeremy Keith, 2005).

Osim promjene razine vidljivosti, elementima na stranici se programom može narediti da se pomiču, odnosno 'klize' (engl. *slide*).

Funkcija slideDown() dovodi skriveni element u vidno polje. Prvo se pojavljuje vrh elementa, potom njegov ostatak prema dolje. Ukoliko se ne zada brzina te radnje, element se pojavi brzinom 400ms (McFarland & Sawyer, 2014). Funkcija slideUp() uklanja element iz vidnog polja tako da prvo sakriva njegov donji dio, potom prema gore sve dok on potpuno ne nestane (McFarland & Sawyer, 2014).

Funkcijom slideToggle() primjenjuje se funkcija slideDown() ukoliko je element skriven, odnosno funkcija slideUp() ako je vidljiv (McFarland & Sawyer, 2014).

Kako bi se element pomicao koristeći svojstva left, right, bottom, top, potrebno je CSS svojstvo position postaviti na absolute ili relative (Powers, 2010). Vrijednost absolute omogućava smještanje elementa bilo gdje na stranici (Jeremy Keith, 2005) te slaganje elemenata u slojevima, jedan iznad drugog, a relative pomiče element iz normalnog položaja s obzirom na elemente koji slijede (Powers, 2010). Osim te dvije vrijednosti, svojstvo position može primiti još tri: static, inherit ili fixed (Powers, 2010). Ako drugačije nije zadano, podrazumijeva se vrijednost static, što znači da drugi elementi na stranici utječu na položaj elementa te on utječe na sve elemente koji slijede (Powers, 2010). Korištenjem div i span elemenata te postavljanjem njihovog svojstva position na absolute, moguće je spriječiti nasumično pomicanje teksta i drugog okolnog sadržaja kada neki element nestaje sa stranice (McFarland & Sawyer, 2014).

Osim korištenja samo ugrađenih efekata u jQuery-ju, moguće je definirati animate() funkciju. Tom funkcijom se može animirati bilo koje CSS svojstvo koje prihvaća numeričku vrijednost (piksel, em ili postotak), npr. veličina teksta, položaj elementa na stranici, prozirnost objekata, širina okvira itd. (McFarland & Sawyer, 2014). Druga CSS svojstva poput boje elemenata se ne mogu animirati tom funkcijom.

Funkcija animate() može primiti nekoliko argumenata. Prvo se navode CSS svojstva koja se žele animirati, zatim njeno trajanje u milisekundama (McFarland & Sawyer, 2014). Animacije ne moraju biti jednake brzine duž cijelog trajanja. Argumentom easing može se kontrolirati brzina u različitim stadijima animacije kako bi animacije postale zanimljivije i dinamičnije (McFarland & Sawyer, 2014). Primjerice element se može početi brzo kretati, potom usporiti i na kraju animacije ponovno ubrzati (McFarland & Sawyer, 2014). Postoje dvije easing metode: swing i linear. Dok linear pruža jednolične pokrete, metoda swing je malo dinamičnija. Njome animacija može početi brzo, zatim usporiti. Ukoliko se ne postavi vrijednost za easing, podrazumijeva se metoda swing (McFarland & Sawyer, 2014). Osim tih metoda, postoje i metode easeInBounce, easeInOutSine, easeInCubic i druge (McFarland & Sawyer, 2014). Neke JavaScript biblioteke kao što su jQuery i GSAP, mogu funkcionirati u kombinaciji s CSS tranzicijama i

CSS animacijama. Primjerice, ako se koristi biblioteka jQuery, slici se može dodati CSS tranzicija tako da slika izblijedi sa 100 % na 0 % u sekundi, jednako kao i funkcija fadeOut() (McFarland & Sawyer, 2014). Ključ je u postavljanju nove klase faded na sliku (McFarland & Sawyer, 2014). Moguće je takav stil primijeniti tek na klik miša posjetitelja tako da se koristi funkcija click(). Na taj način, kada posjetitelj klikne na sliku, jQuery joj jednostavno doda klasu, preglednik tada animira promjenu prozirnosti (McFarland & Sawyer, 2014). Ako se koristi biblioteka GSAP, slici ili nekom drugom elementu se na primjer može dodati transformacija tako da se pomiče za određeni postotak ili broj piksela i istovremeno se pojavljuje na zaslonu u određenom vremenskom trajanju. Za takvu animaciju je ključno da se na željeni element prvo postavi nekoliko CSS svojstava. Svojstvo overflow s vrijednosti hidden, koje će omogućiti nevidljivost sadržaja, svojstvo display s vrijednosti inline-block, koje omogućava namještanje potrebne visine i širine elementa te naravno funkcija transform: translate() kojom se određuje željeni pomak. Eventualni dodatni animacijski pokreti se potom u JavaScript datoteci pridružuju određenom elementu zajedno s vremenskim trajanjem putem ugrađenih GSAP funkcija.

4. Izrada animacije u CSS-u

U svrhu pisanja ovog rada osmišljeno je nekoliko povezanih mrežnih animacija. Ideja je kreirati mrežno sjedište s animacijama određenih elemenata u vremenu učitavanja indeksne stranice, prijelaza na sljedeću stranicu putem linka te učitavanja navigacijske trake i teksta stranice naziva "Mrežne animacije CSS". Uređivač kôda za HTML i CSS koji se koristi za pisanje kôda ovog mrežnog sjedišta je *Visual Studio Code*.

Na početku je potrebno napisati kôd za HTML dokumente stranice (slika 6, slika 7), sa svim potrebnim oznakama i dodijeljenim klasama preko kojih će se povezati s kôdom unutar CSS dokumenta s kojim je povezan.

```
1  <!DOCTYPE html>
2  <html>
3
4      <head>
5          <meta charset="utf-8">
6          <meta http-equiv="X-UA-Compatible" content="IE=edge">
7          <meta name="viewport" content="width-device-width, initial-scale=1.0">
8          <title>Mrežne animacije CSS</title>
9          <link rel="stylesheet" href="stil.css">
10     </head>
11     <body>
12         <div>
13             <nav>
14                 <ul>
15                     <li></li>
16                     <li>
17                         <p class="tekst"><i>CSS<br>Animacije</i></p>
18                         <button><a href="vise.html"><i>Više...</i></a></button>
19                     </li>
20                 </ul>
21             </nav>
22         </div>
23         <main>
24         </main>
25     </body>
26 </html>
```

Slika 6 : Kôd u HTML dokumentu indeksne stranice "Mrežne animacije CSS".

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width-device-width, initial-scale=1.0">
        <title>Mrežne animacije CSS</title>
        <link rel="stylesheet" href="vise_stil.css">
    </head>
    <body>
        <div class="nav">
            <nav id="display">
                <ul>
                    <li></li>
                    <li>
                        <p class="css"><i>CSS<br>Animacije</i></p>
                        <button><i>Više...</i></button>
                    </li>
                </ul>
            </nav>
        </div>
        <nav id="navigacija"></nav>
        <main>
            <div>
                <h1>CSS</h1>
                <p id="sadrzaj">CSS (eng. <i>Cascading Style Sheets</i>) ···
                    one malo složenije.</p>
                <h1>Stil</h1>
                <p id="sadrzaj">U CSS-u, formatiranje teksta izvršava ···
                    elemente stranice...</p>
            </div>
        </main>
    </body>
</html>
```

Slika 7: Kôd u HTML dokumentu stranice sa sadržajem "Mrežne animacije CSS".

U CSS dokumentima je napisan kôd kojim se različitim elementima unutar HTML dokumenata mijenjaju svojstva (Prilog 1, Prilog 2). Osim promjene boje, visine, širine i položaja sadržaja stranice, elementima su dodane i animacije.

Polazeći od početne ideje animiranja elementa prije učitane stranice, napisan je kôd za @keyframes animacije spuštanja navigacijske trake te transformacije elemenata unutar HTML oznaka u CSS datoteci naziva „stil.css“ (slika 8). Prvom od tih elemenata (nav li:first-child {} u CSS-u), bijelom kvadratu, mijenja se veličina (scale()) i prozirnost (opacity()) za vrijeme rotacije za 600° (rotate()), nakon čega se pomiče po

zamišljenoj osi X za 360 % (translateX()). Za vrijeme vraćanja na prvotni položaj, elementu se mijenja prozirnost i veličina po zamišljenoj osi Y, odnosno mijenja se visina elementa (scaleY()). U tom vremenu, pojavljuje se drugi element (nav li:last-child {} u CSS-u) s tekstom i gumbom. Tekst se pojavljuje tako da se spušta na svoj položaj, što je određeno negativnom vrijednošću broja piksela po zamišljenoj osi Y. Podizanje gumba na položaj određeno je pozitivnom vrijednošću broja piksela također po osi Y. Prvi element na svoj konačni položaj dolazi u obliku bijelog, neprozirnog pravokutnika.

```
@keyframes nav-load {
    0% {
        transform: translateY(-100%);
    }
    100% {
        transform: translateY(0);
    }
}
@keyframes nav-li-load {
    0% {
        transform: scale(0)
            rotate(600deg);
    }
    45% {
        transform: scale(1.01);
    }
    50% {
        transform: scale(1);
        opacity: 0.4;
    }
    51% {
        transform: translateX(360%);
    }
    75% {
        opacity: 1;
    }
    100% {
        transform: scaleY(3);
    }
}
```

```
@keyframes tekst-load {
    0% {
        transform: translateY(-400px)
        opacity: 0;
    }
    80% {
        opacity: 0;
    }
    100% {
        transform: translateY(0px);
        opacity: 1;
    }
}
@keyframes button-load {
    0% {
        opacity: 0;
    }
    80% {
        transform: translateY(40px);
        opacity: 0;
    }
    100% {
        transform: translateY(0px);
        opacity: 1;
    }
}
```

Slika 8: Definiranje @keyframes animacija u CSS datoteci "stil.css"(lijevo i desno).

Kako bi @keyframes animacije radile, potrebno im je dodijeliti vlastita imena. Tim jedinstvenom imenom, animacija se može primijeniti neodređenom broju elemenata putem svojstva animation-name. Animaciji se potom može odrediti trajanje (animation-duration) i odgoditi početak njenog izvođenja (animation-delay).

Ukoliko je potrebno povezati više animacija na način da jedna počinje nakon što prethodna završi, moguće je kreirati varijable unutar kojih je zbrojeno vrijeme potrebno za izvođenje prethodnih animacija (calc((var()+var()) ili se broj milisekunda proizvoljno pohrani u varijablu. Varijable se definiraju na početku dokumenta (redak 4-7, slika 9). Primjerice, varijabla –nav-load-time, koja je definirana tako da određuje trajanje animacije (slika 9, redak 4), dodijeljena je animaciji imena nav-load (slika 10, redak 10).

```
1  body {
2      font-family: "Serif";
3      margin: 0;
4      --nav-load-time: 500ms;
5      --nav-li-load-time: 10000ms;
6      --tekst-load-delay: calc(var(--nav-load-time) + 1800ms);
7      --button-load-delay: calc(var(--nav-load-time) + 1800ms);
8  }
```

Slika 9: Definiranje varijabli trajanja za pridruživanje animacijama u CSS datoteci "stil.css".

```
9  nav {
10     animation: nav-load var(--nav-load-time) ease-in-out;
11 }
12 nav li:first-child {
13     animation-name: nav-li-load;
14     animation-duration: var(--nav-li-load-time);
15     animation-timing-function: ease-in-out;
16     animation-delay: var(--nav-load-time);
17     animation-fill-mode: forwards;
18     transform: scale(0);
19 }
20 button {
21     animation: button-load 9000ms ease-out var(--button-load-delay);
22     opacity: 0;
23     animation-fill-mode: forwards;
24 }
25 .tekst {
26     animation: tekst-load 9000ms ease-out var(--tekst-load-delay);
27     opacity: 0;
28     animation-fill-mode: forwards;
29 }
```

Slika 10: Pridruživanje animacija HTML oznakama u CSS datoteci "stil.css".

Nakon animacija na početnoj stranici, klikom na gumb povlače se elementi oznaka , spušta se navigacijska traka i tekst stranice. Naslovi se pojavljuju s vremenskom odgodom nakon pojave teksta što je omogućeno svojstvom animation-fill-mode i vrijednosti forwards. Ono određuje kada elementi poprimaju zadana

obilježja, a vrijednošću forwards elementi ih poprimaju nakon izvođenja animacije. Stranica je otvorena u pregledniku i animacije su podržane. Slike 11-13 prikazuju kôd za CSS animacije u datoteci „vise_stil.css“ za stranicu sa sadržajem, a slike 14-15 kadrove kreirane CSS animacije u pregledniku.

```
1   body {
2       font-family: "Serif";
3       margin: 0;
4       --display-load-time: 1900ms;
5       --nav-li-load-time: 2000ms;
6       --h-load-delay: calc(var(--nav-li-load-time) + 600ms);
7       --nav-load-delay: calc(var(--nav-li-load-time) + 6ms)
8   }
```

Slika 11: Definiranje varijabli trajanja za pridruživanje animacijama u CSS datoteci “vise_stil.css“.

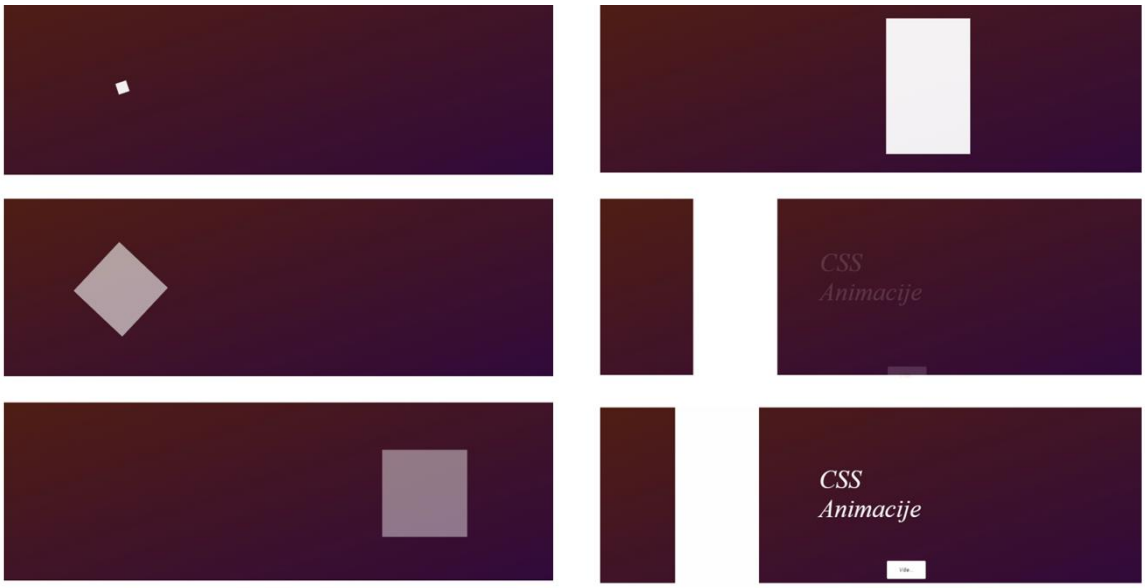
```
38  @keyframes nav-li-load {
39      0% {
40          visibility: visible;
41          transform: scaleY(3.5);
42      }
43      99% {
44          transform: scaleY(3.5)
45              scaleX(10.19);
46      }
47      100% {
48          visibility: hidden;
49      }
50  }
51  @keyframes tekst-load {
52      0% {
53          transform: translateY(-400px);
54          opacity: 0;
55      }
56      80% {
57          opacity: 0;
58      }
59      100% {
60          transform: translateY(0px);
61          opacity: 1;
62      }
63  }
```

```
64  @keyframes h-load {
65      0% {
66          transform: translateY(-20px);
67          opacity: 0;
68      }
69      100% {
70          transform: translateY(0px);
71          opacity: 1;
72      }
73  }
74  @keyframes display-load {
75      0% {
76          visibility: visible;
77      }
78      99% {
79          visibility: visible;
80      }
81      100% {
82          visibility: hidden;
83      }
84  }
85  @keyframes nav-load {
86      0% {
87          transform: translateY(-100%);
88      }
89      100% {
90          transform: translateY(0);
91          opacity: 1;
92      }
93  }
```

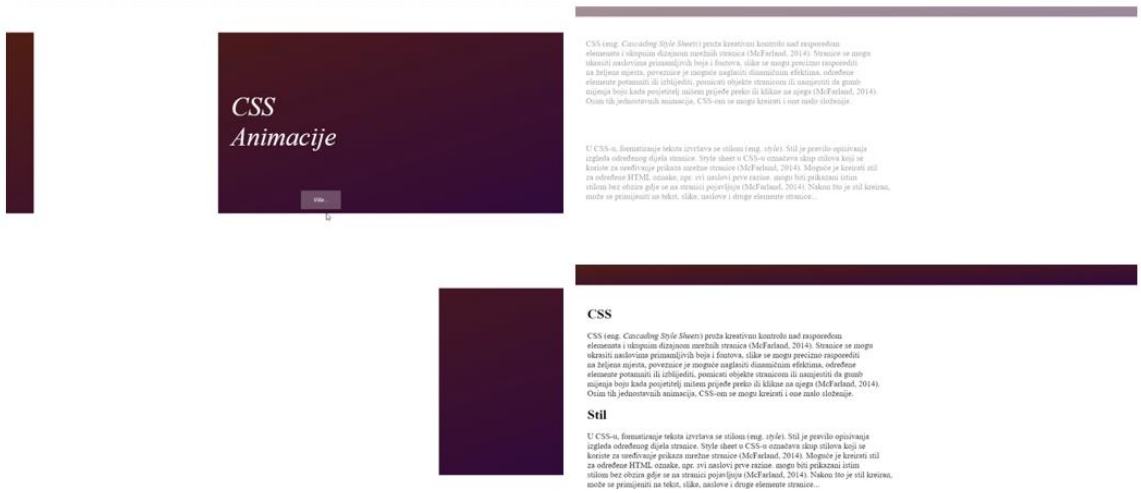
Slika 12: Definiranje @keyframes animacija u CSS datoteci “vise_stil.css“(lijevo i desno).

```
nav li:first-child {
    animation-name: nav-li-load;
    animation-duration: var(--nav-li-load-time);
    animation-timing-function: ease-in-out;
    animation-fill-mode: forwards;
    transform: scale(0);
}
h1 {
    animation: h-load 600ms ease-in-out var(--h-load-delay);
    opacity: 0;
    animation-fill-mode: forwards;
}
#sadrzaj {
    animation: nav-load 1000ms ease-in-out var(--nav-load-delay);
    opacity: 0;
    animation-fill-mode: forwards;
}
#navigacija {
    animation: nav-load 1000ms ease-in-out var(--nav-load-delay);
    opacity: 0;
    animation-fill-mode: forwards;
}
#display {
    animation-name: display-load;
    animation-duration: var(--display-load-time);
    animation-timing-function: ease-in-out;
    animation-fill-mode: forwards;
    visibility: hidden;
}
```

Slika 13: Pridruživanje animacija HTML oznakama u CSS datoteci "vise_stil.css".



Slika 14: Kadrovi kreirane CSS animacije (1).



Slika 15: Kadrovi kreirane CSS animacije (2).

5. Izrada animacije u JavaScript-u

Nakon animiranja pomoću CSS svojstava, jednake animacije su izvedene koristeći JavaScript. Kôd je također pisan u uređivaču kôda *Visual Studio Code*, a zbog jednostavnosti, umjesto pisanja dugih blokova kôda za funkcije koje bi izvodile jednake animacije, korištena je biblioteka GSAP (slika 16, redak 29-31).

Na početku je napisan HTML kôd za mrežno sjedište za indeksnu stranicu i stranicu sa sadržajem pod nazivom "Mrežne animacije JS" (slika 16-17).

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width-device-width, initial-scale=1.0">
        <title>Mrežne animacije JS</title>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <div class="intro">
            <div class="intro-text">
                <h1 class="hide">
                    <span class="text1">JavaScript</span>
                </h1>
                <h1 class="hide">
                    <span class="text1">Animacije</span>
                </h1>
                <h1 class="hide_button">
                    <span class="text2"><button><a href="vise_.html"><i>Više...</i></a></button></span>
                </h1>
            </div>
        </div>
        <ul class="prozirnost">
                <li class="slider"></li>
        </ul>
        <main></main>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.5.1/gsap.min.js"
        integrity="sha512-IQLehpLoVS4fNzl7IfH8Iowfm5+RiMGtHykgZJl9AWMgqx0AmJ6cRWcB+GaGVtIsnC4voMfm8f2vwtY+6oPjpQ=="
        crossorigin="anonymous"></script>
        <script src="./app.js"></script>
    </body>
</html>
```

Slika 16: Kôd u HTML dokumentu indeksne stranice "Mrežne animacije JS".

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width-device-width, initial-scale=1.0">
        <title>Mrežne animacije JS</title>
        <link rel="stylesheet" href="vise_style.css">
    </head>
    <body>
        <main>
            <nav></nav>
            <div class="big-text">
                <h1 class="hide"><span class="podnaslov">JavaScript</span></h1>
                <p class="hide"><span class="sadrzaj">JavaScript je programerski jezik ···
                    stranica učita (Powers, 2010).</span></p>
                <h1 class="hide"><span class="podnaslov">Biblioteke</span></h1>
                <p class="hide"><span class="sadrzaj">JavaScript biblioteke su kolekcije ···
                    za njihovu upotrebu (Dale, 2016)....</span></p>
            </div>
        </main>
        <div class="intro">
            <div class="intro-text">
                <h1>
                    <span class="text1">JavaScript</span>
                </h1>
                <h1>
                    <span class="text1">Animacije</span>
                </h1>
                <h1>
                    <span class="text2"><button><i>Više...</i></button></span>
                </h1>
            </div>
        </div>
        <ul>
            <li class="slider"></li>
        </ul>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.5.1/gsap.min.js"
        integrity="sha512-IQLehpLoVS4fNzl7IfH8Iowfm5+RiMGtHykgZJl9AWMgqx0AmJ6cRWc8+GaGVtIsnC4voMfm8f2vwtY+6oPjpQ=="
        crossorigin="anonymous"></script>
        <script src="./vise_app.js"></script>
    </body>
</html>
```

Slika 17: Kôd u HTML dokumentu stranice sa sadržajem "Mrežne animacije JS".

Iako animacije nisu kreirane u CSS dokumentima, zbog vizualnog dojma prikaza sadržaja stvorena je vanjska CSS datoteka koja je povezana s HTML dokumentima. Kôd unutar CSS datoteka je prikazan u Prilogu 3 i Prilogu 4. Osim toga, neke animacije se ne bi mogle izvesti bez nekih CSS svojstava, kao što je npr. svojstvo overflow s vrijednošću hidden, klase .hide unutar oznaka <h1></h1> (slika 18, redak 84 nadalje).

```
81  .sadrzaj {
82      font-size: 18px;
83  }
84  .hide {
85      overflow: hidden;
86  }
87  .hide span {
88      transform: translateY(-100%);
89      display: inline-block;
90  }
91  .hide_button {
92      overflow: hidden;
93  }
94  .hide_button span {
95      transform: translateY(100%);
96      display: inline-block;
97  }
```

Slika 18: Pridruživanje svojstava overflow i display u CSS datoteci "style.css".

Za korištenje biblioteke GSAP dovoljno je kopirati poveznicu s platforme GreenSock i zalijepiti ju unutar oznake <script> u HTML dokumentu (slika 16, redak 29-31), a moguće je i preuzeti .zip datoteku biblioteke i potrebne datoteke unutar nje jednako označiti u HTML dokumentu (GreenSock, n.d.). Nakon toga, potrebna je još jedna oznaka <script> unutar koje će se navesti put do JavaScript datoteke s kôdom napisanim za izvršavanje animacija na mrežnoj stranici (slika 16, redak 29-32 ; slika 17, redak 52-55).

JavaScript biblioteka GSAP pruža mogućnost kreiranja vremenske crte (engl. *timeline*) u JavaScript datoteci. Korištenjem *timeline*-a jednostavno je kontrolirati sve pokrete elemenata i precizno upravljati vremenom potrebnim za njihovo izvođenje, bilo da je riječ o redoslijedu, trajanju, odgodi ili pak ranijem pokretanju animacije (GreenSock, n.d.). Timeline i zajednička svojstva svih animacija unutar njega pohranjeni su pod jedinstvenom konstantom (const tl), (slika 19 i slika 20, redak 1). Svaku zasebnu animaciju s njoj jedinstvenim kôdom potrebno je dodati konstanti. Prvo se izvodi prva zapisana animacija, potom animacija u sljedećem retku itd. No, za ovo postoji iznimka koja će se objasniti kasnije. Kao i u CSS-u, oznake i klase dodijeljene oznakama u HTML-u imaju veliku ulogu u JavaScript-u. Njima se povezuje sve što mora biti povezano.

Animacija početnog bijelog kvadrata, kojem je u HTML-u pridružena klasa .slider, prije učitane indeksne stranice napisana je kôdom u dva retka (rotacija, translacija,

promjena prozirnosti i visine) (slika 19, redak 3-7). Ako je nekom elementu pridruženo više animacija koje se trebaju izvoditi istovremeno, pišu se u istom retku, odvojene točkom i ključnom riječi bez razmaka. Postoji način za ranije izvođenje animacija bez obzira u kojem retku se nalazi. Posljednjom brojčanom vrijednosti pod navodnicima u zagradi određeno je ranije izvođenje animacije. Dakle, bez navedene negativne vrijednosti broja sekundi, animacija bi krenula tek nakon što bi se izvršila animacija kreirana u retku iznad. Tako je u retku 10 na slici 19 vrijednošću "-=1.25" omogućena pojava elementa klase .text1 za vrijeme trajanja animacije iz prethodnog retka, 1.25 sekundi prije povratka bijelog elementa na početni položaj.

Spuštanje teksta omogućeno je postavljanjem {y:"0%"}, a svojstvo i njemu pridružena vrijednost {stagger: 0.25} omogućava 0.25 sekundi razmaka u izvođenju elemenata kojima je pridružena klasa .text1 (slika 19, redak 10).

```
1   const tl = gsap.timeline({defaults: {ease: 'power1.out'}});
2
3   tl.to(".slider", 5.5, {rotation: 720}).fromTo(".slider", 5.5, {scale: 0},
4       {scale: 1}, "-=5");
5   tl.to(".slider", 1, {x: "800"}).fromTo(".slider", 1.6, {height: "0%"},
6       {height: "100%"}).to(".slider", 1, {y: "-150" }, "-=1.6").to(".slider", 1,
7       {x: "0%"}, "-=1");
8   tl.fromTo(".prozirnost", {opacity: 0.4}, {opacity: 0.7, duration: 5.5},
9       "-=5").fromTo(".prozirnost", {opacity: 0.7}, {opacity: 1, duration: 3}, "-=3");
10  tl.to(".text1", {y: "0%", duration: 1, stagger: 0.25}, "-=1.25");
11  tl.to(".text2", {y:"0%", duration: 1}, "-=1.25");
12  tl.to(".intro", {opacity: 1}, {opacity: 0, duration: 1});
13  tl.fromTo(".big-text", {opacity: 0}, {opacity: 1, duration: 1}, "-=1");
```

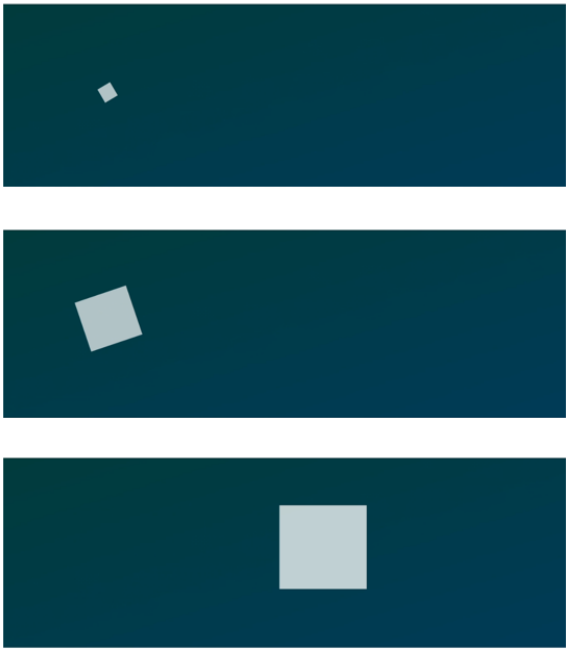
Slika 19: Definiranje animacija u JavaScript datotekama “app.js“.

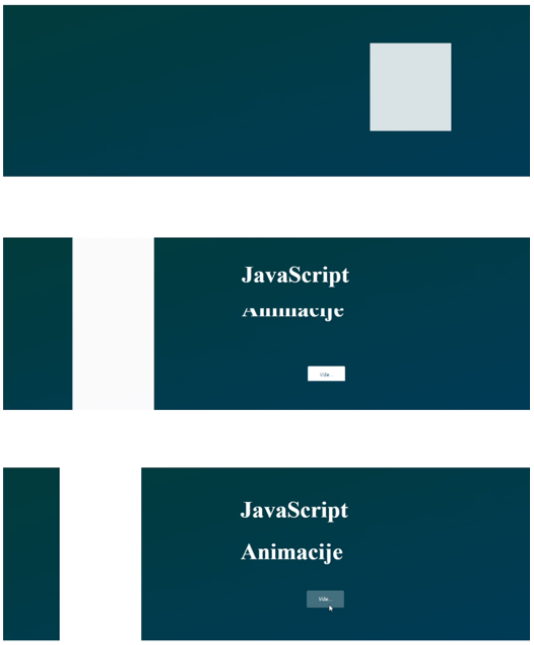
```
1   const tl = gsap.timeline({defaults: {ease: 'power1.out'}});
2
3   tl.fromTo(".slider", 1.6, {width: "0%"}, {width: "100%"}).to(".slider",
4       1, {x: "-200" }, "-=1.6");
5   tl.to(".slider",{y: "-350%", duration: 0.9});
6   tl.to(".intro", {y: "-350%", duration: 0.7}, "-=1");
7   tl.fromTo("nav", {opacity: 0}, {opacity: 1, duration: 1}, "-=0.1");
8   tl.fromTo(".big-text", {opacity: 0}, {opacity: 1, duration: 1}, "-=1");
9   tl.to(".sadrzaj", {y: "0%", duration: 1.5},"-=1.5");
10  tl.to(".podnaslov", {y: "0%", duration: 0.5}, "-=0.5");
```

Slika 20: Definiranje animacija u JavaScript datotekama “vise_app.js“.

Klikom na gumb, bijeli element se širi i pomiče po osi X, potom se pomiče po osi Y zajedno s elementima unutar oznake klase .intro (slika 20: redak 5 i 6), nakon čega se na ekranu pojavljuje navigacijska traka i tekstualni sadržaj stranice. Pojavljivanje navigacijske trake i teksta omogućeno je promjenom prozirnosti sadržaja unutar oznake <nav></nav> i elemenata unutar oznake klase .big-text, a njihovo spuštanje po osi Y omogućeno je postavljanjem vrijednosti {y:"0%"}. Podnaslovi i sam tekstualni sadržaj ne spuštaju se istovremeno jer je u retku 8 zadano da se sadržaj pojavljuje gotovo istovremeno s navigacijskom trakom (vrijednost "-=1" prije zatvaranja zagrade), a u retku 10 je zadana blaga odgoda od pola sekunde (slika 20).

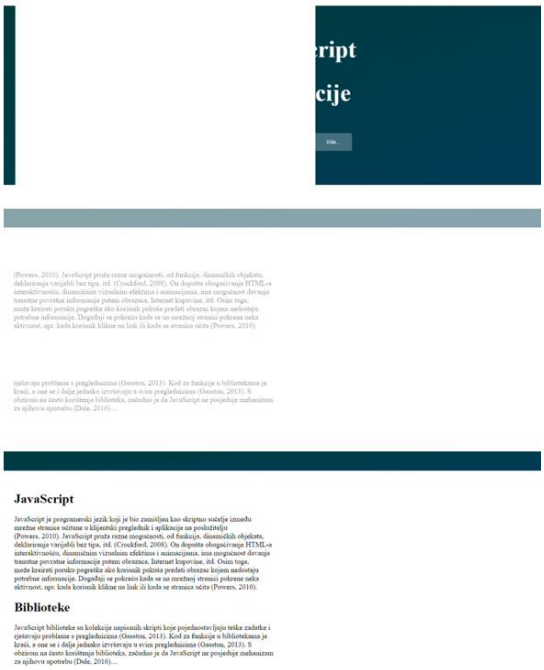
Stranica je otvorena u pregledniku i omogućeno je izvođenje kreiranih animacija. Kadrovi animacija su prikazani na slikama 21-23.





Slika 21: Kadrovi kreirane JS animacije (1).

Slika 22: Kadrovi kreirane JS animacije (2).



Slika 23: Kadrovi kreirane JS animacije (3).

6. Analiza izrade animacija

Zamišljena mrežna animacija uspješno je ostvarena u CSS-u i JavaScript-u. No, postupak kreiranja animacija je različit u nekoliko segmenata. Različita sintaksa, duljina kôda, broj datoteka s kojima je HTML dokument povezan i samo trajanje izrade su prve vidljive razlike. Kôd za animacije u CSS-u je znatno duži od kôda pisanog pomoću biblioteke GSAP u JavaScript datoteci. Jednake animacije se korištenjem JavaScript biblioteka mogu izvršiti u manje redaka nego @keyframes animacije kreirane u CSS-u, koje se moraju imenovati, pridružiti elementima zajedno s varijablama koje su prethodno definirane i u koje je pohranjena određena vrijednost.

Za mrežnu stranicu s JavaScript animacijama, u HTML dokumentu je potrebno navesti vezu do JavaScript biblioteke bez koje se funkcije za mrežne animacije ne bi mogle izvršavati. Osim toga, potrebno je kreirati posebnu JavaScript datoteku, odvojenu od CSS datoteke, u kojoj se piše kôd za animacije i veza do nje koja je navedena unutar oznake <script> u HTML dokumentu.

Za izvršavanje CSS animacija je dovoljno kreirati HTML i CSS datoteke te u HTML-u odrediti njihovu povezanost. U jednoj datoteci, CSS datoteci, tako se nalazi kôd za prikaz sadržaja iz HTML-a i animacije. No, za izvršavanje JavaScript animacija, potrebne su najmanje tri datoteke: HTML, CSS i JavaScript datoteke. CSS datoteka je potrebna jer određuje način prikaza elemenata iz HTML dokumenta na stranici. Osim što se unutar CSS-a određuju svojstva elemenata kao što su npr. njihova boja i veličina, u CSS-u se elementima pridružuju funkcije koje omogućavaju ili olakšavaju izvođenje funkcija pisanih u JavaScript datoteci. Tablica 1 daje usporedni prikaz izrade animacija u CSS-u i JavaScript-u.

Tablica 1: Usporedba izrade animacija u CSS-u i JavaScript-u (s bibliotekom GSAP)

	CSS	**JavaScript (+ GSAP)**
Potreban broj vanjskih datoteka	1	najmanje 2
Uređivanje svojstava sadržaja i kreiranje animacija	na istom mjestu (u jednoj datoteci)	u odvojenim datotekama
Broj redaka kôda za animacije	više	manje
Redoslijed animacija	mogućnost nadovezivanja animacija izračunom vrijednosti trajanja prethodnih animacija	mogućnost izrade *timeline*-a za sve animacije na stranici

7. Zaključak

Bez animacija, slike i tekst bi bili samo nepomične skupine točaka na papiru ili piksela na zaslonu uređaja. Dodavanjem pokreta, dodaje im se živost te one dobivaju potpuno drugačiju ulogu. Ovisno o tome što se želi postići s određenim elementom, mijenja se pristup i način izvršavanja zadatka.

Još godinama unatrag, pokreti su privlačili pažnju ljudima. Važna prekretnica u povijesti prikazivanja iluzija o pokretu, bio je izum prve kamere. Nakon toga je započeo nagli razvoj tehnologija koje su se koristile u ostvarivanju iluzija pokreta, prvo u obliku igranih filmova na platnu, a potom i animiranih filmova na platnu, televiziji i konačno zaslonima računala, mobitela, tableta i drugih uređaja. Animacije su se počele pojavljivati na internetu, mrežnim stranicama i aplikacijama. Njihova je uloga privući pažnju korisnika i naglasiti određeni sadržaj na ekranu. Važno je da se ne pretjera s animacijama na stranici i duljinom njihovog trajanja kako one ne bi ostavile negativan učinak i odvukle korisnika od glavnog sadržaja stranice, zbog kojeg ju je u osnovi i posjetio. Postoji više načina i programa kojima se mogu kreirati animacije u mrežnom prostoru.

U svrhu pisanja ovog rada, detaljnije je proučeno kreiranje animacija u CSS-u i JavaScript-u. Obje tehnologije pružaju gotovo jednake mogućnosti animiranja te se obje moraju povezati s HTML dokumentom na čijem se sadržaju animacije trebaju izvoditi u mrežnom prostoru. CSS i JavaScript, između ostalog, omogućavaju rotaciju, translaciju, promjenu veličine, oblika i prozirnosti elemenata stranice. Također je moguće odrediti željeno vremensko trajanje i odgode početaka izvršavanja animacija. Uz razlike u sintaksi za pisanje kôda, kad se usporedi korištenje obje tehnologije, pojavljuju se razlike u količini pisanog kôda potrebnog za ostvarivanje jednakog pokreta. Ukoliko se koristi neka od JavaScript biblioteka, kao što je GSAP koji se koristio za ovaj rad, broj redaka kôda za JavaScript animacije je mnogo manji od broja redaka kôda za animacije u CSS-u.

Kreiranjem @keyframes animacija u CSS-u omogućeno je kontroliranje pokreta u određenom trenutku trajanja animacije. Ono se određuje postotkom koji se odnosi na postotak ukupnog vremena potrebnog za izvođenje animacije. Za izvršavanje animacija pisanih u CSS-u dovoljno je kreirati samo CSS datoteku s potrebnim kôdom za izvođenje animacija, dok je za JavaScript animacije potrebno kreirati i posebnu JavaScript datoteku sa zapisanim kôdom i CSS datoteku u kojoj se nalazi kôd za promjenu svojstava HTML elemenata. U CSS datoteci se mogu promijeniti boje, veličine, razmaci i brojna druga svojstva elemenata navođenjem oznake kojoj pripadaju ili klase koja im je pridružena unutar oznake. Oznake u HTML-u su tako povezane s CSS i JavaScript datotekama. S druge strane, JavaScript biblioteka GSAP ima mogućnost kreiranja vremenske crte koja objedinjuje sve animacije na mrežnoj stranici, što omogućava jednostavno pridržavanje zadanog redoslijeda animacija.

Na kraju, nema konačnog objektivnog odgovora na pitanje koja je tehnologija bolja od druge za izradu jednostavnih animacija. Danas većina preglednika podržava i CSS i JavaScript animacije, postoje biblioteke koje olakšavaju pisanje JavaScript kôda do te mjere da je za jednaku animaciju potrebno napisati manje redaka kôda nego u CSS-u. Nekome se pisanje animacija u CSS-u može činiti kompliciranijim jer treba paziti na definiranje određenih varijabli i njihovo dodjeljivanje elementima, ali kontroliranje pokreta navođenjem postotaka u trajanju animacije se može činiti lakšim od usklađivanja različitih animacija unutar jedne vremenske crte. Odabir tehnologije ovisi o animatoru i njegovom znanju o njoj ili pak želji i mogućnostima da to znanje proširi i primijeni u svom radu.

8. Literatura

1. "Adobe Flash Player EOL General Information Page." Pristupljeno 29. svibnja 2021. https://www.adobe.com/products/flashplayer/end-of-life.html.
2. "CSS Tutorial." Pristupljeno 10. svibnja, 2021. https://www.w3schools.com/css/.
3. Dale, Kyran. *Data Visualization with Python and JavaScript: Scrape, Clean, Explore & Transform Your Data*. Sebastopol, CA: O'Reilly, 2016.
4. Frain, Ben. *HTML5 i CSS3: Prilagodljivi Web Dizajn. Translated by Slavica Prudkov*. Beograd: Kompjuter biblioteka, 2014.
5. Furniss, Maureen. *Art in Motion: Animation Aesthetics*. Rev. ed. Eastleigh, UK: John Libbey, 2007.
6. Furniss, Maureen. *A New History of Animation*. New York: Thames & Hudson, 2016.
7. Gasston, Peter. *Moderni Web: responzivni Web dizajn uz HTML5, CSS3 i JavaScript*. Footprint : informatičke knjige za kreativce. Zagreb: Dobar plan, 2013.
8. GreenSock. "GSAP." Pristupljeno 16. svibnja, 2021. https://greensock.com/gsap/.
9. "JavaScript Functions." Pristupljeno 10. svibnja, 2021. https://www.w3schools.com/js/js_functions.asp.
10. Keith, Jeremy, and Dave Shea. *DOM Scripting: Web Design with JavaScript and the Document Object Model*. Berkeley, CA: Friendsof, 2005.
11. McFarland, David Sawyer. *CSS3: The Missing Manual.* 3rd ed. The Missing Manual. Sebastopol, CA: O'Reilly, 2013.
12. McFarland, David Sawyer. *JavaScript & JQuery*. 3rd ed. The Missing Manual. Sebastopol, CA... [etc]: O'Reilly, 2014.
13. Meyer, Eric A. *CSS Web Site Design: Includes Exercise Files and Demo Movies*. Lynda Weinman's Hands-on Training. Berkeley: Lynda.com : Peachpit Press, 2007.

14. Powell, Thomas A., Dejan Smiljanić, and Aleksandar Dragosavljević. *Web Dizajn: Kompletan Priručnik*. Beograd: Mikro knjiga, 2001.
15. Powers, Shelley. *Naučite JavaScript*. Translated by Mariana Lisjak and Jasmina Morvaj. Zagreb: Dobar Plan, 2010.
16. World Wide Web Foundation. "Sir Tim Berners-Lee." Pristupljeno 4. svibnja, 2021. https://webfoundation.org/about/sir-tim-berners-lee/.
17. Sito, Tom. *Moving Innovation: A History of Computer Animation*. Cambridge. Mass. ; London: The MIT Press, 2013.
18. Wells, Paul. *Understanding Animation*. London: Routledge, 1998.
19. Whitaker, Harold, John Halas, and Tom Sito. *Timing for Animation*. 2nd ed. Burlington, MA: Focal Press, 2009.

9. Popis priloga

9.1. Prilog 1: CSS za indeksnu stranicu s CSS animacijom

```css
body {
    font-family: "Serif";
    margin: 0;
}
main {
    padding: 30px;
}
nav {
    background-image:
        linear-gradient(
        to bottom right,
        rgb(80, 30, 20),
        rgb(50, 10, 60));
    color: white;
}
ul {
    display: flex;
    justify-content: left;
    text-align: left;
    font-size: 63px;
}
nav li:first-child {
    background: white;
    width: 10px;
    height: 10px;
    list-style: none;
    margin: 0 150px;
    margin-top: 118px;
    padding: 100px;
}
nav li:last-child {
    list-style: none;
    margin-top: 80px;
}
nav a {
    color: inherit;
    text-decoration: none;
}
button {
    background-color: white;
    color: rgb(50, 10, 60));
    border-radius: 3px;
    border-color: white;
    border-style: hidden;
    margin-top: 6px;
    margin-bottom: 10px;
    margin-left: 170px;
    padding: 15px;
    padding-right: 30px;
    padding-left: 30px;
}
button:hover {
    background-color: #FFFFFF44;
    color: white;
    border-color: #FFFFFF44;
}
```

9.2. Prilog 2: CSS za stranicu sa sadržajem s CSS animacijom

```css
body {
    font-family: "Serif";
    margin: 0;
}
nav {
    background-image:
        linear-gradient(
        to bottom right,
        rgb(80, 30, 20),
        rgb(50, 10, 60));
    color: white;
}
ul {
    display: flex;
    justify-content: left;
    text-align: left;
    font-size: 63px;
}
nav li:first-child {
    background: white;
    width: 10px;
    height: 10px;
    list-style: none;
    margin: 0 150px;
    margin-bottom: 150px;
    padding: 100px;
}
nav li:last-child {
    list-style: none;
    margin-top: 80px;
}
nav a {
    color: inherit;
    text-decoration: none;
}
button {
    background-color: #FFFFFF44;
    color: white;
    border-radius: 3px;
    border-color: #FFFFFF44;
    border-style: hidden;
    margin-top: 6px;
    margin-bottom: 10px;
    margin-left: 170px;
    padding: 15px;
    padding-right: 30px;
    padding-left: 30px;
}
main {
    padding: 30px;
}
div {
    position: absolute;
}
.nav {
    width: 100%;
}
#tekst {
    opacity:1;
}
#sadrzaj {
    font-size: 20px;
}
#navigacija {
    background-image:
    linear-gradient(
    to bottom right,
    rgb(80, 30, 20),
    rgb(50, 10, 60));
    color: white;
    width: 100%;
    height: 50px;
}
```

9.3. Prilog 3: CSS za indeksnu stranicu s JavaScript animacijom

```css
body {
    font-family: "Serif";
}
nav {
    background-image:
        linear-gradient(
        to bottom right,
        rgb(0, 60, 60),
        rgb(0, 60, 90));
    width: 100%;
    height: 50px;
    margin: 0;
}
a {
    color: inherit;
    text-decoration: none;
}
ul li:first-child {
    position: fixed;
    list-style: none;
    padding: 100px;
    margin: 0 150px;
    margin-bottom: 100px;
    width: 10px;
    height: 10px;
    background: white;
}
button {
    background-color: white;
    border-radius: 3px;
    border-color: #FFFFFF44;
    border-style: hidden;
    color: rgb(0, 60, 90);
    margin-top: 6px;
    margin-bottom: 10px;
    margin-left: 170px;
    padding: 15px;
    padding-right: 30px;
    padding-left: 30px;
}
button:hover {
    background-color: #FFFFFF44;
    color: white;
    border-color: #FFFFFF44;
}
.big-text {
    position: absolute;
    padding: 30px;
    color: black;
}
.intro {
    background-image:
        linear-gradient(
        to bottom right,
        rgb(0, 60, 60),
        rgb(0, 60, 90));
    position: fixed;
    padding: 70px;
    top: 17%;
    left: 0;
    width: 100%;
    height: 50%;
    display: flex;
    justify-content: center;
    align-items: center;
}
.intro-text {
    top: 10px;
    color: white;
    font-size: 30px;
}
.slider {
    background: white;
    position: fixed;
    top: 35%;
    left: 0;
    width: 10%;
    height: 10%;
    display: flex;
}
```

9.4. Prilog 4: CSS za stranicu sa sadržajem s JavaScript animacijom

```css
body {
    font-family: "Serif";
}
nav {
    background-image:
        linear-gradient(
        to bottom right,
        rgb(0, 60, 60),
        rgb(0, 60, 90));
    width: 100%;
    height: 50px;
    display: flex;
}
a {
    color: inherit;
    text-decoration: none;
}
ul li:first-child {
    background: white;
    width: 10px;
    height: 10px;
    list-style: none;
    margin: 0 150px;
    margin-bottom: 150px;
    padding: 100px;
    transform: scaleY(3);
}
button {
    background-color: #FFFFFF44;
    color: white;
    border-radius: 3px;
    border-color: #FFFFFF44;
    border-style: hidden;
    margin-top: 6px;
    margin-bottom: 10px;
    margin-left: 170px;
    padding: 15px;
    padding-right: 30px;
    padding-left: 30px;
}
.big-text {
    color: black;
    padding: 30px;
}
.intro {
    background-image:
        linear-gradient(
        to bottom right,
        rgb(0, 60, 60),
        rgb(0, 60, 90));
    width: 100%;
    height: 50%;
    position: fixed;
    padding: 70px;
    top: 17%;
    left: 0;
    display: flex;
    justify-content: center;
    align-items: center;
}
.intro-text {
    color: white;
    font-size: 30px;
}
.slider {
    background: white;
    position: fixed;
    top: 30%;
    left: 0;
    width: 10%;
    height: 40%;
    display: flex;
}
.hide {
    overflow: hidden;
}
.hide span {
    transform: translateY(-100%);
    display: inline-block;
}
```

Sažetak

Tema rada je izrada mrežnih animacija. U njemu su objašnjene vrste animacija i njihova uloga na mrežnim stranicama. Rad je fokusiran na izradu animacija u CSS-u i JavaScript-u. Objašnjene su osnove izrade mrežnih animacija pomoću spomenutih tehnologija. Spomenuta je važnost smještaja i vremenskog trajanja animacije na mrežnoj stranici. Prikazani su primjeri jednostavnih animacija (kadrovi animacija u pokretu) te opisan kôd kojim su izrađene. Navedene su i objašnjene razlike između oba načina izrade mrežnih animacija te uočene prednosti svakog od njih. Cilj rada je istražiti mogućnosti koje pružaju CSS i JavaScript i olakšati odabir između navedenih tehnologija za kreiranje animacija.

Ključne riječi: animacija, *HTML, CSS, JavaScript, Web*

Creating Web Animation: CSS or JavaScript?

Abstract

The subject of this paper is making Web animation. Different types of animations and their role on Web sites are explained. The focus of the paper is on the comparison of creating Web animation using two different tools: CSS and JavaScript. Basics of making Web animation with mentioned tools are presented. The importance of the position and time duration of animation are mentioned. The examples of simple animation are shown through photographs of frames and their programming code is described. The main differences of both methods are stated and explained, and the advantages of each method are noticed. The aim is to explore the possibilities of CSS and JavaScript in order to facilitate the choice between these technologies for animation.

Key words: *animation, HTML, CSS, JavaScript, Web*