Izrada web animacija korištenjem CSS, SVG i JavaScript tehnologija

Martines, Valentino

Sveučilište
Sjever

Završni rad br. 15/RINF/2015

Izrada web animacija korištenjem CSS, SVG i JavaScript tehnologija

Valentino Martines, 0336051481

Đurđevac, rujan 2024. godine

Sveučilište Sjever
Sveučilišni centar Koprivnica
Trg dr. Žarka Dolinara 1, HR-48000 Koprivnica

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL: Računarstvo i informatika

STUDIJ: Stručni prijediplomski studij Računarstvo i informatika

PRISTUPNIK: Valentino Martines | MATIČNI BROJ: 0336051481

DATUM: 25. 9. 2024. | KOLEGIJ: Razvoj web aplikacija

NASLOV RADA: Izrada web animacija korištenjem CSS, SVG i JavaScript tehnologija

NASLOV RADA NA ENGL. JEZIKU: Development of web animations using CSS, SVG and JavaScript technologies

MENTOR: Dražen Crčić | ZVANJE: predavač

ČLANOVI POVJERENSTVA:

1. doc.dr.sc. Tomislav Horvat - predsjednik povjerenstva
2. doc.dr.sc. Matija Varga - član povjerenstva
3. pred. Dražen Crčić, mag.ing. - mentor
4. doc.dr.sc. Domagoj Frank - zamjenski član
5.

Zadatak završnog rada

BROJ: 15/RINF/2024

OPIS:

Izrada web animacija može se postići korištenjem različitih tehnologija, a tri glavne tehnologije koje se često koriste za izradu animacija su CSS, SVG i JavaScript. Svaka od tih tehnologija ima svoje prednosti i nedostatke tako da je najbolje rješenje koristiti ih istovremeno kako bi se razvile kompleksnije i interaktivnije web animacije.

U sklopu završnog rada potrebno je:
- dati općeniti opis i osvrt na sve tri navedene tehnologije
- detaljnije objasniti njihovu svrhu i mogućnosti unutar rješenja web animacije
- napraviti usporedbu te objasniti optimalnu primjenu zasebnih tehnologija i u međusobnoj kombinaciji
- kroz konkretna programska rješenja i primjere demonstrirati razvijene funkcionalnosti

ZADATAK URUČEN: 26.9.2024 | POTPIS MENTORA:

SVEUČILIŠTE SJEVER

SIEVER UNIVERSITY NORTH

Očisti obrazac

Sveučilište
Sjever

Odjel za računarstvo i informatiku

Završni rad br. 15/RINF/2024

Izrada web animacija korištenjem CSS SVG i JavaScript tehnologija

Student

Valentino Martines, 0336051481

Mentor

Dražen Crčić, mag. ing. el.

Đurđevac, rujan 2024. godine

Sažetak

Ovaj rad istražuje razne tehnologije animacija u web dizajnu, fokusirajući se na CSS, JavaScript i SVG. Svaka od ovih tehnologija ima svoje prednosti i nedostatke, a njihova kombinacija može značajno unaprijediti korisničko iskustvo. CSS animacije su jednostavne za implementaciju i idealne za osnovne, statične efekte, omogućujući glatke prijelaze i transformacije. Međutim, njihova fleksibilnost može biti ograničena kod složenijih animacija koje zahtijevaju dinamičko računanje.

JavaScript animacije nude visoku kontrolu i fleksibilnost, što ih čini pogodnim za kompleksne i interaktivne animacije. Koristeći razne metode programeri mogu manipulirati animacijama u stvarnom vremenu, prilagođavajući ih korisničkim interakcijama. No, složenost JavaScript animacija može rezultirati većim opterećenjem za preglednike, što utječe na performanse.

SVG animacije, s druge strane, omogućuju visoku kvalitetu vektorske grafike, skalabilnost i mogućnost izrade dinamičkih i interaktivnih elemenata. Koristeći SVG, dizajneri mogu stvarati privlačne vizualne efekte, ali upravljanje složenijim SVG animacijama može predstavljati izazov. Stariji preglednici također mogu imati problema s podrškom za nove SVG značajke.

Rad uspoređuje ove tri tehnologije kroz primjere animacija, demonstrirajući njihove jedinstvene osobitosti. CSS se koristi za jednostavne efekte, dok JavaScript radi kompleksnije zadatke koristeći dinamičke izračune. Izbor odgovarajuće tehnologije za animacije ovisi o specifičnim potrebama projekta, a kombinacija svih triju može rezultirati najefikasnijim rješenjem. Ova analiza ističe važnost razumijevanja mogućnosti svake tehnologije kako bi se postiglo optimalno korisničko iskustvo na webu.

Ključne riječi: CSS, JavaScript, SVG, optimizacija performansi, korisničko iskustvo, skalabilnost.

Summary

This thesis explores various animation technologies in web design, focusing on CSS, JavaScript, and SVG. Each of these technologies has its own advantages and limitations, and their combination can significantly enhance the user experience. CSS animations are easy to implement and ideal for basic, static effects, allowing for smooth transitions and transformations. However, their flexibility may be limited for more complex animations that require dynamic calculations.

JavaScript animations offer high control and flexibility, making them suitable for complex and interactive animations. Using various methods, developers can manipulate animations in real-time, adapting them to user interactions. However, the complexity of JavaScript animations can lead to increased browser load, impacting performance.

SVG animations, on the other hand, allow for high-quality vector graphics, scalability, and the creation of dynamic and interactive elements. Using SVG, designers can create visually appealing effects, though managing more complex SVG animations can be challenging. Older browsers may also struggle with support for newer SVG features.

The thesis compares these three technologies through animation examples, highlighting their unique characteristics. CSS is used for simple effects, while JavaScript handles more complex tasks with dynamic calculations. Choosing the appropriate animation technology depends on the specific needs of a project, and a combination of all three can result in the most efficient solution. This analysis underscores the importance of understanding each technology's capabilities to achieve an optimal user experience on the web.

Keywords: CSS, JavaScript, SVG, performance optimization, user experience, scalability.

Popis korištenih kratica

HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
SVG	Scalable Vector Graphics
XML	Extensible Markup Language
GSAP	GreenSock Animation Platform
RINF	računarstvo i informatika
GIF	Graphics Interchange Format
URL	Uniform Resource Locator (jedinstveni lokator resursa)
API	Aplication Programming Interface (sučelje za programiranje aplikacija)
SMIL	Synchronized Multimedia Integration Language
RAF	Request Animation Frame

Sadržaj

1. Uvod

Od samih početaka razvoja weba animacije su se koristile s ciljem da privuku pažnju korisnika ili pojasne neki dinamički sadržaj koji se ne bi mogao dovoljno dobro predstaviti tekstom i slikom. Animirane GIF datoteke i *Flash* animacije, do nedavno prevladavajuće formate za animacije na webu, danas zamjenjuju nove tehnologije usklađene s HTML 5 standardom, animacije načinjene korištenjem CSS-a, SVG-a i JavaScripta [1].

Izrada web animacija može se postići korištenjem različitih tehnologija, a tri glavne tehnologije koje se često koriste su CSS, SVG i JavaScript. Naravno svaka od tih tehnologija ima svoje prednosti i nedostatke tako da je najbolje rješenje koristiti ih istovremeno kako bismo postigli kompleksnije i interaktivnije web animacije. Tako dobivamo širi spektar i maksimalnu fleksibilnost u dizajniranju i implementaciji animacija. Web animacije su svestrani alat koji se može prilagoditi različitim potrebama i ciljevima web stranica. Integracija različitih tehnologija omogućuje stvaranje vizualno atraktivnih, interaktivnih i skalabilnih animacija koje poboljšavaju korisničko iskustvo na webu.

Web animacije čine ključni element suvremenih web stranica, pružajući interaktivnost, poboljšavajući korisničko iskustvo i dodajući vizualni dojam. Postoje različite tehnologije i pristupi za stvaranje web animacija. Osim gore već navedenih naravno postoji još mnogo ključnih aspekata za izradu web animacija poput biblioteke i raznih okvira, optimizacija perfomansi, rezolucijski prilagodljive animacije, pristupačnost, te samo praćenje budućih trendova.

2. Povijest animacija

Povijest web animacija prati razvoj web tehnologija i napredak u korisničkom iskustvu na internetu. Neki od ključnih trenutaka u povijesti web animacija kreću 90. tih kada počinje sam pojam interneta i GIF animacije koje su bile najpopularniji način za stvaranje animacija na webu. Ove animacije su imale jednostavne sekvence slika koje su se kretale bez potrebe za nekim dodatnim kodiranjem. Kvaliteta i mogućnost su bile ograničene, ali bez obzira na to, zbog svoje jednostavnosti i kompatibilnosti sa svim preglednicima bile su jako korištene.

Adobe Flash tehnologija je u sredini 1990-ih donijela revoluciju web animacija. Omogućila je razvoj sofisticiranih animacija i interaktivnih aplikacija, pružajući bogatu grafiku i animacije koje su bile daleko naprednije od onoga što je bilo moguće sa HTML-om i CSS-om u to vrijeme. Da bi se prikazale *Flash* animacije korisnici su morali instalirati *Flash Player* dodatak internet pregledniku (engl. *plugin*). *Flash* je omogućio stvaranje kompleksnih animacija, igara i video sadržaja, ali je imao problem s performansama.

JavaScript svojim razvojem zamjenjuje upotrebu *Flash-a* zbog lakšeg stvaranja dinamičkih i interaktivnih efekata. Omogućuje manipulaciju sadržaja i stila u stvarnom vremenu. Pojavom biblioteke jQuery krajem 2000-ih olakšava se kreiranje animacija nudeći jednostavne API-je za animiranje objektnih modela dokumenata elemenata i izradu efikasnih korisničkih sučelja.

CSS počinje 2009. sa CSS3 verzijom tranzicija i animacija što je donijelo značajne promjene u načinu na koji se animacije mogu implementirati na web stranicama. Omogućene su bile jednostavne promjene stanja i kompleksnije sekvence animacija. CSS animacije ubrzo su postale popularne zbog svoje učnikovitosti i bolje podrške za računala sa modernim preglednicima, što je omogućilo glatke performanse i manju potrošnju resursa u usporedbi s JavaScript animacijama.

Web animacije su se razvijale od jednostavnih GIF-ova i *Flash* animacija do složenih i optimiziranih CSS i JavaScript animacija. Svaka faza u povijesti web animacija donijela je nova rješenja i mogućnosti, čineći web iskustvo dinamičnijim i interaktivnijim nego ikad prije [2].

Budućnost web animacija donijet će inovacije koje će omogućiti još bogatija, interaktivnija i personalizirana iskustva, uz jaču integraciju s naprednim tehnologijama poput virtualne i proširene stvarnosti. S novim alatima i tehnikama, animacije će postati ključno sredstvo za poboljšanje korisničkog iskustva na webu.

Proširena stvarnost je tehnologija koja omogućava integraciju digitalnih informacija s realnim svijetom u stvarnom vremenu. Koristi kamere i senzore na pametnim telefonima, tabletima ili specijaliziranim naočalama kako bi nadogradila okolinu s digitalnim sadržajem u obliku slike, teksta ili 3D objekata. Kod virtualne stvarnosti računalo generira okoline u koje korisnici mogu uroniti putem specijaliziranih uređaja kao što su virtualne naočale ili slušalice.

3. CSS animacije

CSS animacije su snažan alat za stvaranje dinamičkih i atraktivnih vizualnih efekata na web stranicama bez potrebe za upotrebom JavaScripta ili drugih skriptnih jezika. Animacije omogućavaju postupan prijelaz iz jedne vrijednosti CSS svojstva u drugu. Možemo mijenjati raznolike efekte i CSS svojstava koliko puta želimo te postići raznolike efekte i poboljšati korisničko iskustvo. Kada koristimo CSS animacije moramo postaviti pravila za okvire tzv. @keyframes animacije [3]. Animacije poput *transition, transform, keyframes, animation* i *sprite* su najčešće korištene web animacije.

CSS animacije nude jednostavnu upotrebu kroz implementaciju ključnih okvira (engl. *keyframes*) i osnovnih svojstava kao što su transition i animation. Osim toga, pružaju lagan pristup bez potrebe za dodatnim skriptama ili bibliotekama, što rezultira brzim učitavanjem stranica. Međutim, imaju ograničenu složenost i dinamičnost u usporedbi s JavaScript animacijama. Također, teže je upravljati animacijama na temelju korisničkih događaja, poput klikova ili poteza mišem.

3.1. Tranzicije

Tranzicijske animacije definiraju postupne promjene u svojstvima elemenata, poput boje, veličine ili pozicije. Ključni elementi tranzicijske animacije uključuju tzv. property svojstvo koje se animira (npr. širina, dužina, pozadinska boja), vrijeme trajanja animacije u sekundama ili milisekundama (engl. *duration*). Određivanje brzine mijenjanja animacije tijekom vremena (engl. *timing function*), s opcijama poput *ease, linear, ease-in, ease-out, ease-in-out*). Određivanje odgodbe početka animacije nakon pokretanja (engl. *delay*). Tranzicije imaju mogučnost primjena na više svojstva odjednom tako što se navede više svojstva unutar „*transition-property*“ ili korištenjem „*all*“ kako bi se primjenilo na sva promjenjiva svojstva elemenata.

Razlika između tranzicija i animacija je ta da su tranzicije jednostavne promjene iz jednog stanja elementa u drugo, automatski se pokreću kada se dogodi promjena u stilu elemenata, dok kod CSS animacija moguće je uključivati više ključnih okvira i složenijih promjena.

Primjer animacije prijelaza dan je u nastavku.

```
.element {
  width: 100px;
  height: 100px;
  background-color: blue;
```

```
  transition: width 0.5s ease-in-out;
}
.element:hover {
  width: 200px;
  background-color: red;
  color: white;
  font-size: 20px;
}
```

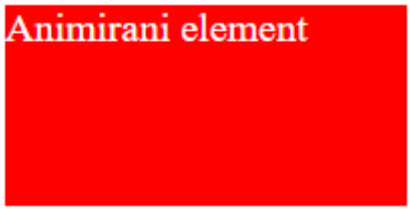
Početno stanje (.element).

Element počinje sa širinom od 100px, visinom od 100px, plavom pozadinom, i primjenjuje se animacija prijelaza samo na širinu (*width*) tijekom 0.5 sekundi s glatkim *ease-in-out* efektom.

Stanje na *hover* (.element:hover).

Kada korisnik postavi miš iznad elementa (*hover*), širina se mijenja na 200px, boja pozadine postaje crvena (background-color: red), boja teksta postaje bijela (color: white), i veličina fonta raste na 20px (font-size: 20px). Ove promjene primjenjuju se tijekom iste animacije prijelaza (0.5 sekundi s *ease-in-out*). Ovaj primjer prikazuje kako možemo koristiti animaciju prijelaza za glatke i elegantne promjene svojstava kada se element nalazi u *hover* stanju. Primjena različitih svojstava tijekom prelaska mišem preko elementa dodaje dinamičnost i interaktivnost web stranici.

Slika 3-1. Početno stanje .element

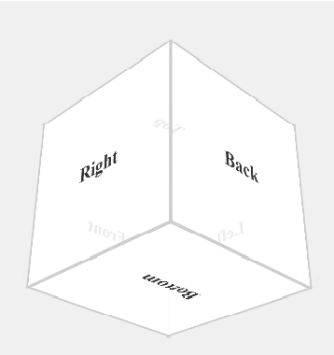


Slika 3-2 Završno stanje element:hover

3.2. Transformacije

CSS svojstvo transformacije (engl. *transform*) koristi se za primjenu transformacija na elemente na web stranici. Ova svojstva omogućuju progrmibilno mijenjanje izgleda i položaja elemenata. Transformacije se primjenjuju bez promjene samog rasporeda stranice, što znači da susjedni elementi neće biti pogođeni transformacijama. Transformacija se koristi za promjenu izgleda elementa, uključujući translaciju, rotaciju, skaliranje, nagib itd.

Translacija pomiče elemente na novu poziciju unutar stranice bez promjene rasporeda okolnih elemenata. Rotacija rotira element za određeni broj stupnjeva oko osi. Skaliranje ima mogućnost povećanja i umanjenja veličine elementa. Nagib deformira element tako što ga naginje u horizontalnom ili vertikalnom smjeru. Također postoje i 3D transformacije koje omogućuju dodavanje dubine elementima uz pomoć translateZ() i perspective().



Slika 3-3 3D transformacija

Primjer transformacije dan je u nastavku.

```
body {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
  margin: 0;
}

.element {
  width: 150px;
  height: 150px;
```

```
  background-color: #3498db;
  color: white;
  text-align: center;
  line-height: 150px;
  font-size: 18px;
  cursor: pointer;
  transition: transform 0.5s ease-in-out;
}

.element:hover {
  transform: rotate(180deg);
}
```

U ovom primjeru, imamo kocku s tekstom "Rotiraj me!". Kada korisnik pređe mišem preko elementa primijenit ćemo transformaciju rotacije za 180 stupnjeva, što će rezultirati polovičnu rotaciju elementa u smjeru kazaljke na satu. Svojstvom „*transition: transform 0.5s ease-in-out*“ omogućuje se glatka transformacija koja traje pola sekunde s efektom „*ease-in-out*“. Korištenjem :*hover* pseudoklase, element se rotira za 180 stupnjeva kada se mišem prođe preko njega.

Slika 3-4 Poćetno stanje transform elementa

Slika 3-5 Završno stanje transform elementa

3.3. Animacije s ključnim okvirima

Animacije s ključnim okvirima (engl. *keyframes*) i stanjima animiranog objekta u CSS-u omogućuju definiranje ključnih trenutaka tijekom animacije, pružajući veću kontrolu nad njenim ponašanjem. Ključevi su definirani pomoću tzv. @keyframes pravila, a animacija se postiže promjenom svojstava elementa između različitih ključnih trenutaka. Takve animacije se koriste kod složenih animacija u različitim medijima, uključujući web dizajn, film i video igre. Animacijski softver koristi interpolaciju za automatsko generiranje prijelaznih okvira. To omogućuje glatke prijelaze između početnog i završnog stanja bez potrebe za ručnim stvaranjem svakog pojedinog okvira. Korisnik postavlja ključne okvire na vremenskoj traci tako što definira važna stanja objekta u različitim točkama vremena. Svaki ključni objekt može od animiranih postavki sadržavati pozicije, rotacije, veličine, boje i razne druge atribute.

Primjer animacije sa ključnim okvirima.

```
body {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
  margin: 0;
  background-color: #ecf0f1;
}

.element {
  width: 100px;
  height: 100px;
  background-color: #3498db;
  animation: rotateColorChange 3s infinite;
  cursor: pointer;
}

@keyframes rotateColorChange {
  0%, 100% {
      transform: rotate(0deg);
```

```
        background-color: #3498db;
    }
    25% {
        transform: rotate(90deg);
        background-color: #e74c3c;
    }
    50% {
        transform: rotate(180deg);
        background-color: #2ecc71;
    }
    75% {
        transform: rotate(270deg);
        background-color: #f39c12;
    }
}
```

U ovom primjeru, kvadrat će neprekidno rotirati i mijenjati boje. Svaka promjena boje događa se u različitim kutovima rotacije. Ključne linije koda su

.element {...}: - stilizira kvadrat.

width: 100px; height: 100px; - postavlja širinu i visinu kvadrata.

background-color: #3498db; - početna boja kvadrata.

animation: rotateColorChange 3s infinite; - dodjeljuje animaciju „*rotateColorChange*“ kvadratu. Animacija će trajati 3 sekunde i izvodit će se beskonačno.

cursor: pointer; - postavlja kursor na strelicu kako bi ukazao na to da je element interaktivan.

@keyframes rotateColorChange {...}: - definira ključeve za animaciju „*rotateColorChange*“.

0%, 100% {...} - kvadrat je u početnom položaju (0 stupnjeva rotacije) i ima početnu boju.

25% {...} - na 25% animacije, kvadrat se rotira za 90 stupnjeva i mijenja boju.

50% {...} - na 50% animacije, kvadrat se rotira za 180 stupnjeva i mijenja boju.

75% {...} - na 75% animacije, kvadrat se rotira za 270 stupnjeva i mijenja boju.

Ovaj primjer bi trebao omogućiti jasnu vizualnu promjenu kada korisnik pređe mišem preko kvadrata.

3.4. Funkcija vremena

Vremenska funkcija animacije (engl. *animation timing function*) u CSS-u određuje kako će se promjene u svojstvima elementa događati tijekom vremena. Postoje različite vrste vremenskih funkcija koje kontroliraju brzinu animacije, uključujući *ease, linear, ease-in, ease-out,* i *ease-in-out*. Ove funkcije omogućavaju fino podešavanje animacija kako bi se postigao željeni vizualni efekt.

Trajanje CSS animacije kontrolira se svojstvom *animation-duration*, određujući ukupno trajanje animacije. Ako se ne postavi trajanje, animacija će se izvršiti odjednom bez postupnog prijelaza. Koraci animacije mogu se definirati u postotcima, omogućujući precizno određivanje stanja CSS svojstava u svakom prolazu.

Brzina animacije prilagođava se svojstvom *animation-timing-function*, koje utvrđuje krivulju brzine animacije. Ovo svojstvo može imati različite vrijednosti, uključujući *ease* (spor početak, ubrzanje i na kraju spor kraj), *linear* (jednaka brzina od početka do kraja), *ease-in* (animacija sa sporim početkom) i *ease-out* (animacija sa sporim krajem). *ease-in-out* – Animacija sa sporim početkom i krajem.

Primjer funkcije vremena dan je u nastavku.

```
body {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
  margin: 0;
  background-color: #ecf0f1;
}
.element {
  width: 100px;
  height: 100px;
  background-color: #3498db;
  animation: colorChangeRotate 2s ease-in-out infinite;
  cursor: pointer;
}
@keyframes colorChangeRotate {
  0% {
```

```
        transform: rotate(0deg);
        background-color: #3498db;
    }
    50% {
        transform: rotate(180deg);
        background-color: #e74c3c;
    }
    100% {
        transform: rotate(0deg);
        background-color: #3498db;
    }
}
```

Ovaj primjer prikazuje kvadrat koji se rotira za 180 stupnjeva i mijenja boje s vremenskom funkcijom *ease-in-out*. Cijela animacija traje dvije sekunde i izvodi se beskonačno. Razlika u odnosu na prethodni primjer je kraće trajanje animacije i manji broj ključnih trenutaka.

3.5. CSS sprite animacija

CSS *sprite* animacija je tehnika koja uključuje korištenje jedne slike (*sprite sheet*) koja sadrži više slika kako bi se postigla animacija na web stranici. Svaka pojedina slika na *sprite sheetu* predstavlja jedan korak animacije. Kroz CSS, određeni dijelovi *sprite sheeta* se prikazuju i animiraju, stvarajući dojam pokreta. Ova tehnika omogućuje prikazivanje različitih dijelova slike kao animirani okvir kroz promjenu pozicije pozadinske slike. *Sprite sheet* je slika koja sadrži više manjih slika raspoređenih u mreži ili nizu. Svaki okvir prikazuje različitu fazu animacije. CSS se ovdje koristi za premještanje pozadine tako da se na ekranu prikazuje samo određeni dio *sprite sheet-a*. Promjena pozicije pozadine u različitim intervalima stvara iluziju animacije.

CSS *sprite* animacija je učinkovita tehnika za smanjenje broja HTTP zahtjeva i poboljšanje performansi web stranica, jer se sve slike potrebne za animaciju učitavaju odjednom u jednom zahtjevu. Ovo smanjuje vrijeme učitavanja stranice, pogotovo kod web stranica s puno vizualnih elemenata. Također, korištenje *sprite* animacija olakšava optimizaciju mobilnih aplikacija i igara, gdje je brzina i responzivnost ključna. Umjesto učitavanja pojedinačnih slika za svaku animaciju, sve faze animacije već su uključene u jedan *sprite sheet*.

Primjer *sprite sheet* animacije dan je u nastavku.

```
body {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
  margin: 0;
  background-color: #ecf0f1; }
.element {
  width: 500px;
  height: 500px;
  background-image: url('sprite-sheet.png');
  background-size: 800%;
  animation: jumpAnimation 1s steps(8) infinite;
}
@keyframes jumpAnimation {
  to {
      background-position: -800%;
  }}
```

.element - stilizacija klase koja predstavlja animirani lik.

width: 100px; height: 100px; - postavlja širinu i visinu lika.

background-image: url('sprite-sheet.png'); - postavlja sliku za pozadinu, koristeći *sprite sheet*.

background-size: 800%; - postavlja veličinu pozadine kako bi bila 8 puta šira od elementa prema broju koraka animacije na *sprite sheetu*.

animation: jumpAnimation 1s steps(8) infinite; - primjenjuje animaciju *jumpAnimation* s trajanjem od 1 sekunde, koristeći 8 koraka (*steps*) i ponavlja je beskonačno.

@keyframes jumpAnimation - definira ključne trenutke animacije.

to - zadnji ključni trenutak u animaciji.

background-position: -800%; - pomiče pozadinu *sprite sheeta* za iznos koji odgovara širini jednog koraka animacije prema lijevo.

4. JavaScript animacije

JavaScript je ključna tehnologija za izradu suvremenih web stranica i web aplikacija, a njegova popularnost i važnost u industriji ne prestaje rasti. Također je i jedan od najpopularnijih programskih jezika. Osim na webu, JavaScript se koristi u raznim drugim okruženjima, što ga čini vrlo svestranim jezikom za učenje i primjenu. JavaScript omogućuje dinamičko mijenjanje elemenata na stranici, uključujući animaciju. Glavni način postizanja animacije u JavaScriptu je kroz upravljanje svojstvima elemenata tijekom vremena. JavaScript je stvoren od strane Brendana Eicha, inženjera u Netscape Communications Corporation kompaniji.

Jezik je prvotno zamišljen kao skriptni jezik za dodavanje dinamičnosti web stranicama. Prva implementacija JavaScripta pojavila se u Netscape Navigatoru 2.0 1995. godine. Unatoč ranim uspjesima u Netscape-u, bilo je izazova u vezi sa standardizacijom. Ecma International preuzela je inicijativu da standardizira jezik, rezultirajući ECMAScript standardom 1997. godine. JavaScript je doživio nagli porast popularnosti s razvojem tehnologija poput Ajax-a. Web stranice su postale dinamičnije i interaktivnije, što je doprinijelo pojmu Web 2.0. jQuery. Biblioteka napisana u JavaScriptu, postala je izuzetno popularna zbog svoje jednostavnosti i sposobnosti rješavanja problema s kompatibilnošću između preglednika. Pojavljuju se moderni okviri kao što su Angular, React i Vue.js, koji olakšavaju razvoj kompleksnih web aplikacija. Node.js, izgrađen na V8 JavaScript *engine-u* tvrtke Google, omogućio je izvođenje JavaScripta na serverskoj strani što je značajno proširilo upotrebu JavaScripta izvan preglednika. Objavljen kao ECMAScript 2015 (poznat i kao ES6), donio je mnoge novitete u jezik, uključujući nove konstrukcije jezika, klase, strelice, destrukturiranje i druge poboljšane značajke. ECMAScript se nastavlja razvijati s novim izdanjima koja donose nove značajke i poboljšanja. Različiti preglednici pokušavaju održavati dosljednost implementacije novih standarda [4].

Dinamičnost omogućuje JavaScriptu prilagodbu animacija na osnovu korisničkih interakcija, vanjskih podataka ili stanja aplikacije, što je teže postići samo CSS-om ili SVG-om. JavaScript pruža visoku razinu kontrole nad tijekom animacija, omogućujući manipulaciju svojstvima elemenata, promjenu putanja ili kontrolu vremenskog tijeka animacije. Asinkronost podržava izvođenje animacija paralelno s drugim zadacima, poboljšavajući ukupnu izvedbu web stranice. Korištenje vanjskih biblioteka poput GSAP pruža napredne značajke i optimizacije koje nisu dostupne u standardnom CSS-u ili SVG-u. JavaScript se integrira s drugim tehnologijama poput WebGL-a za kompleksne 3D animacije ili Web Audio API-ja za sinkronizaciju zvuka s animacijama.

Nedostaci JavaScript animacija uključuju složenost koda, ovisnost o performansama preglednika, vrijeme čekanja zbog analize i izvršavanja JavaScript datoteka, potencijalni gubitak funkcionalnosti ako korisnik onemogući JavaScript, te poteškoće s pristupačnošću. Intenzivne JavaScript animacije mogu povećati potrošnju baterije na mobilnim uređajima, što je bitno za prilagođavanje web stranica mobilnim uređajima.

4.1. Zahtjev za okvir animacije

Zahtjev za okvir animacije (engl. *request animation frame*) je JavaScript API koji omogućuje učinkovito planiranje animacija na web stranici. Umjesto korištenja *setTimeout* ili *setInterval* za animacije, *requestAnimationFrame* pruža optimiziran pristup koji se automatski prilagođava brzini osvježavanja zaslona. Često se koristi za stvaranje rekurzivnih animacijskih petlji. Unutar te petlje, programer može izvršavati promjene u CSS svojstvima, transformacije na elementima, ili druge animacijske operacije. Osim što omogućuje sinkronizaciju s vremenskim ciklusom preglednika, RAF automatski zaustavlja animaciju kada je prozor neaktivan, čime štedi resurse.

Često se koristi u kombinaciji s drugim tehnologijama kao što su CSS animacije i transformacije, WebGL za kompleksne 3D animacije, ili tzv. Web Audio API za sinkronizaciju zvuka s vizualnim elementima. Predstavlja moderni pristup stvaranju glatkih i efikasnih web animacija, prilagođenih karakteristikama i mogućnostima preglednika, uzimajući u obzir optimalnu uporabu resursa i bolje iskustvo korisnika. Metoda *window.requestAnimationFrame()* govori pregledniku da želi izvesti animaciju. Zahtijeva od preglednika da pozove funkciju povratnog poziva koju je donio korisnik prije sljedećeg ponovnog brojanja [5].

Optimizacija performansi osigurava glatke animacije, prilagođavajući automatski osvježavanje slike preglednika. Automatsko zaustavljanje *requestAnimationFrame* smanjuje potrošnju resursa kada korisnik prekine pregledavanje ili prebaci kartice. Ispravno tempiranje temelji se na brzini osvježavanja zaslona, održavajući sinkronizaciju s preglednikom. Manje opterećenje CPU-a postiže se izbjegavanjem nepotrebnih izračuna kada prozor nije aktivan. Nedostaci uključuju ovisnost o pregledniku i hardverskim performansama, izazove u upravljanju vremenskim tokom animacija te poteškoće u implementaciji pristupačnosti.

Primjer RAF koda dan je u nastavku.

```
var textElement = document.getElementById("mydiv");
var leftpos = 0;
var direction = 1; // 1 za pomicanje udesno, -1 za pomicanje ulijevo
```

```
function moveText(timestamp){
 leftpos += 1 * direction;
  textElement.style.left = leftpos + "px";
  // Uvjet za promjenu smjera nakon određenog pomicanja
  if (leftpos > 200 || leftpos < 0) {
    direction *= -1;
  }
  requestAnimationFrame(moveText);
}
requestAnimationFrame(moveText);
```

U ovom primjeru, tekst će se pomicati udesno dok ne dosegne ili premaši 200 piksela, a zatim će se početi pomicati ulijevo sa brzinom od jedne sekunde [6].

4.2. Interpolacija

Interpolacija (eng. *tweening*), skraćenica od "*in-betweening*" (interpolaranje), je tehnika u računalnoj grafici i animaciji koja se koristi za generiranje niza između dvije ključne točke kako bi se postigao glatki prijelaz između njih. Ova tehnika je široko korištena u stvaranju animacija, posebno u kontekstu web dizajna i razvoja igara.

Ključne točke definiraju početno i konačno stanje animacije ili promjene u svojstvima objekta (npr. položaj, veličina, boja). Interpolacija zatim generira niz između tih ključnih točaka, stvarajući glatki prijelaz koji simulira kontinuiranu promjenu svojstva tijekom vremena [7]. Interpolaciju primjenjujemo u izradi web animacija, grafičkom dizajnu, videoigrama, animaciji karaktera, razvoju mobilnih aplikacija itd. Interpolacija između ključnih točaka animacije, dolazi u različite vrste. *Linear Tweening* je najjednostavniji oblik, gdje se vrijednosti svojstava mijenjaju ravnomjerno između ključnih točaka. *Ease In, Ease Out Tweening* uvodi ubrzanje na početku i usporava na kraju animacije, pridonoseći prirodnijim i estetski ugodnijim pokretima. *Elastic Tweening* dodaje elastičnost u animaciju, često koristeći se za postizanje efekta “bounce” ili “o*vershoot*” pri završetku animacije.

Animacija se uvijek bavila iluzijom kretanja. Interpolacija je glavni dio kako bi ta iluzija izgledala stvarno. Ključni okviri su slike na početku i kraju glatkog prijelaza. Na primjer, može se činiti da animirani lik skače s jedne točke na drugu. Taj bi lik bio jasno definiran u svakom ključnom kadru, ali između ta dva kadra, lik bi mogao izgledati iskrivljeno ili rastegnuto kako bi odgovarao pokretu i smjeru. U ranim danima animacije, glavni umjetnik animacije crtao bi ključne kadrove, a drugi umjetnik, između njih, stvarao bi animaciju kadar po kadar između tih ključnih kadrova. Sa sadašnjom

računalnom animacijom, proces je puno lakši i puno manje vremena i rada. Prednosti interpolacije su jednostavna implementacija, povećana pristupačnost, brza iteracija, glatki prijelazi i mogućnost opsežne kontrole. Dok su nedostaci poteškoća sa interakcijom, teška izrada kompleksnih animacija, ovisnost o ključnim točkama i ograničena sposobnost simuliranja [8].

4.3. Canvas

Canvas, u kontekstu web razvoja, odnosi se na HTML5 *Canvas* element. HTML5 *Canvas* pruža mogućnost dinamičkog crtanja grafike na web stranicama pomoću JavaScripta. To je snažan alat koji omogućuje stvaranje raznih vizualnih elemenata, animacija, igara i interaktivnih korisničkih sučelja. *Canvas* element je postao nezamjenjiv alat u arsenalu web programera iz nekoliko razloga. Omogućuje izradu dinamičkih i interaktivnih korisnikovih sučelja bez potrebe za dodacima poput *Flash-a*. Budući da je dio HTML5 standarda, *canvas* je podržan u svim modernim web preglednicima, osiguravajući dosljedno korisničko iskustvo na različitim uređajima. Konačno, s *canvasom* programeri imaju mogućnost izrade bogatih grafičkih sadržaja koji su prilagodljivi i optimizirani za performanse.

Canvas element dolazi s bogatim setom API-ja koji omogućuje crtanje 2D ili čak 3D grafika. Neke od ključnih značajki uključuju crtanje oblika kao što su pravokutnici, kružnice i putanje, upotrebu boja, gradijenta i uzoraka, kao i upravljanje tekstom i slikama unutar *canvasa*. Također, *canvas* podržava transformacije poput rotacije, skaliranja i translacije, što programerima pruža veliku fleksibilnost u dizajniranju složenih grafičkih scena ili animacija.

Upotrebe *canvas* elementa vidljive su u raznim web aplikacijama današnjice. Neki primjeri uključuju igre na webu koje koriste canvas za prikazivanje igre u realnom vremenu, online alate za crtanje i uređivanje fotografija, grafičke simulatore, edukativne alate za vizualizaciju podataka, i mnoge druge. Osim toga, često se koristi u web dizajnu za stvaranje složenih efekata kao što su animirane pozadine ili interaktivni elementi koji reagiraju na korisničke interakcije. Poznat je po tome što podržava crtanje 2D grafike, razne metode crtanja, animacije, manipulaciju pikselima i interakciju sa korisnicima poput hvatanja događaja miša.

Prednosti SVG animacija uključuju dinamičko crtanje, visoke performanse, slobodu dizajna, široku primjenu te mogućnost interakcije s korisnicima. S druge strane, nedostaci obuhvaćaju potrošnju resursa, ograničenja statičkog crtanja te složenost izrade scenarija. HTML5 *Canvas* element je neosporno postao ključna komponenta u izradi modernih, interaktivnih i vizualno privlačnih web stranica i aplikacija. Njegova učikovitost leži u kombinaciji svestranosti, široke podrške preglednika i bogatih mogućnosti za kreativnost programera. S pravim pristupom i tehnikama, programeri mogu

iskoristiti puni potencijal *canvasa* kako bi stvorili fantastična korisnička iskustva koja zadivljuju i angažiraju korisnike širom svijeta [9].

4.4. Animacije proteklog vremena i učitavanja

Animacija učitavanja (engl. *loading animation*) koristi se za informiranje korisnika da se neki proces odvija u pozadini i da mora pričekati dok se taj proces ne završi. Najčešći primjer takve animacije sa kojima se susrećemo je krug koji se stalno vrti dok se određeni podaci i zadaci ne izvrše u potpunosti.

Animacija proteklog vremena (engl. *time elapsed*) animacija prikazuje koliko vremena je prošlo i koliko će još vremena biti potrebno da se završi određeni proces. Takva vrsta animacije ima jasno definirano trajanje i završava kada vrijeme istekne. Najčešće takve animacije prikazujemo nekim načinom odbrojavanja ili prikazivanja preostalog vremena za neki proces kao što je preuzimanje.

Ključne linije koda u JavaScript-u su *startAnimation* funkcija koja pokreće *loading* animaciju, a zatim simulira napredak. *RequestAnimationFrame* računa koliko je vremena prošlo u odnosu na ukupno vrijeme trajanja. Također omogućuje sinkronizaciju animacija s renderiranjem stranice, što osigurava glatke animacije i bolje perfomanse. Animacije se mogu zaustaviti pomoću *cancleAnimationFrame* funkcionalnosti kada se proces završi ili sami korisnik prekine radnju.

```
<script>
    function startAnimation() {
        const loader = document.querySelector('.loader');
        const progress = document.querySelector('.progress');
        const progressText = document.querySelector('.progress-text');
        setTimeout(() => {
            loader.style.display = 'none';
            progress.style.width = '100%';
            // Start updating the percentage
            let duration = 5000; // 5 seconds
            let startTime = Date.now();
            function updateProgress() {
                let elapsedTime = Date.now() - startTime;
                 let percentage = Math.min((elapsedTime / duration) * 100,
100);
                progressText.textContent = Math.floor(percentage) + '%';
```

```
                if (percentage < 100) {
                    requestAnimationFrame(updateProgress);
                }
            }
            updateProgress();
        }, 2000);
    }
    startAnimation();
</script>
```

Slika 4-1 Loading animacija

Slika 4-2 Time elapsed animacija

5. SVG

SVG (engl. *Scaleable Vector Graphic*) je vektorski format temeljen na tekstualnom jeziku za opisivanje 2D grafike u XML-u. SVG specifikacije uključuju efekte animacije temeljene na SMIL jeziku (engl. *Synchronized Multimedia Integration Language*), XML jeziku ili stvarajući sinkronizirane audio, video i animirane elemente.

SVG animacije odličan su način da dodamo razne interaktivnosti i vizualni interes web stranice ili aplikacije. Naravno zbog takvih zahtjeva postoji opširno korisničko iskustvo sa mnogo zanimljivih SVG animacija za razne namjene kao što su *morphing shapes, path drawing, text animations, hover effects, loading spinners, SVG filters, SVG sprites,* logo animacije, itd.

SMIL je jezik koji omogućuje jednostavno definiranje i sinkronizaciju multimedijskih elemenata (video, zvuk i fotografije) na web-u. Svakom medijskom objektu pristupa se jedinstvenim lokatorom resursa (URL-om). SMIL omogućuje pohranjivanje medijskog objekta u više verzija, višejezične verzije zvučnih zapisa i sl. Namijenjeno je da SMIL mogu koristiti svi koji koriste i HTML. Svaka animacija efekt je definiran elementom s atributima za podešavanje (engl. *fine-tuning*). Iako ugrađeni SVG/SMIL animacijski elementi pružaju dobre alate za sve vrste animacijskih zadataka, nedostatak podrške preglednika znači da to nije najbolja opcija. SVG elementi također se mogu animirati pomoću CSS prijelaza i *keyframes-a*.

Treba napomenuti da CSS može animirati samo CSS svojstva, a ne atributne vrijednosti koje mogu biti ograničavajuće za SVG, koji koristi atribute za geometriju i raspored. Ovu tehniku ometa i ograničena podrška preglednika, iako se to i dalje poboljšava za css animacije s jednostavnim nekritičnim efektima. SVG animacije mogu animirati svojstva CSS-a, tako da se animira pomoću postotnih vrijednosti. SVG animacije mogu se kreirati na više načina. Jedan od njih je korištenje oznake *animate*, koja se integrira izravno unutar SVG koda. Ova oznaka omogućuje animaciju različitih atributa SVG elemenata, poput pozicije, boje ili veličine, pružajući dinamičke efekte bez potrebe za dodatnim skriptama [10]. SVG ključno se sastoji od vektorskog formata, raznolike grafike, visoke čitljivosti i jednostavne manipulacije, mogućnosti stilizacije putem CSS-a, animacija, podrške za filtriranje i efekte te prilagodbe ikona i grafičkih elemenata poput logotipa. Prednosti SVG-a obuhvaćaju prilagodljivost, interaktivnost, mogućnosti animacija, visoku čitljivost te skalabilnost bez gubitka kvalitete. S druge strane, nedostaci uključuju izazove u radu s kompleksnijom grafikom, poput 3D grafika, te potrebu za visokim performansama u određenim scenarijima.

Primjer SVG-a dan je u nastavku.

```
<!DOCTYPE html>
<html>
<body>
<svg height="100" width="100" xmlns="http://www.w3.org/2000/svg">
  <circle r="45" cx="50" cy="50" stroke="green" stroke-width="3" fill="red" />
</svg>
</body>
</html>
```

Objašnjenje koda SVG animacije.

<svg height="100" width="100" xmlns="http://www.w3.org/2000/svg"> - Ovdje počinje SVG element. Postavke visine i širine postavljene su na 100 piksela, a xmlns atribut postavljen je na "*http://www.w3.org/2000/svg*" kako bi naznačio da je riječ o SVG formatu.

<circle r="45" cx="50" cy="50" stroke="green" stroke-width="3" fill="red" /> - Ovo je element kruga unutar SVG-a. Atributi kruga su sljedeći.

r="45" - Radijus kruga je postavljen na 45 piksela.

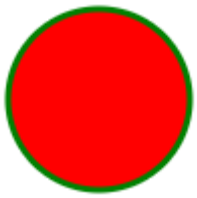
cx="50" - X koordinata centra kruga je postavljena na 50 piksela.

cy="50" -Y koordinata centra kruga je postavljena na 50 piksela.

stroke="green" - Boja obruba kruga je postavljena na zelenu.

stroke-width="3" - Širina obruba kruga je postavljena na 3 piksela.

fill="red" - Boja unutrašnjosti kruga je postavljena na crvenu.



Slika 5-1. SVG element

5.1. Crtanje linija u SVG formatu

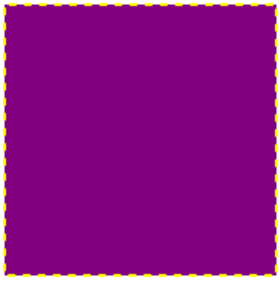
Animacija koja služi za crtanje linija u SVG formatu (engl. *SVG drawing line animation*) omogućuje stvaranje efekata kao da se linija ili određeni oblici postupno iscrtavaju na zaslona. Ova tehnika je vrlo popularna za ilustracije, ikone, logotipe i druge grafičke elemente, dodajući dinamiku u samu interaktivnost web stranica. Funkcionira tako što koristi *path* element za definiranje linija i oblika, dok *"d"* atribut unutar *"<path>"* elementa opisuje točke i krivulje koje čine liniju. CSS animacije možemo koristit za animiranje SVG putanja, omogućujući glatke prijelaze i dinamične efekte. Ključni aspekt animacije crtanja linija u SVG-u temelji se na korištenju CSS svojstva *stroke-dasharray* i *stroke-dashoffset*. *Stroke-dasharray* određuje duljinu crtanja linije, dok *stroke-dashoffset* kontrolira početnu točku crte. Njihovom kombinacijom dobiva se efekt iscrtavanja linije iz početne točke prema kraju.

Primjer crtanja linije SVG formatom dan je u nastavku.

```
<body>
    <svg width="200" height="200" viewBox="0 0 200 200" xmlns="http://www.w3.org/2000/svg">
        <path id="line" d="M10 10 H 190 V 190 H 10 Z" stroke="yellow" stroke-width="2" fill="purple"/>
    </svg>
</body>
#line {
    stroke-dasharray: 6;
    stroke-dashoffset: 0;
    animation: draw 4s linear infinite;
}
@keyframes draw {
    to {
        stroke-dashoffset: -360;
    }}
```

U ovome primjeru koda crta se pravokutnik unutar SVG elementa s početkom u točki (10, 10) i završava u točki (10, 190). Linija je žute boje *stroke="yellow"* i debljine 2 piksela. Pravokutnik je popunjen ljubičastom bojom *fill="purple",* ali tu se mogu ubacivati različite stvari poput slika i raznih oblika koje želimo iscrtati. U CSS dijelu *stroke-dasharray* određuje uzorak crtica i praznine koje čine liniju, kao u ovome primjeru vrijednost 6 znači da se na svakih 6 jedinica puta crta jedna crtica, a

ostatak se ostavlja prazan i time dobivamo isprekidanu liniju. *Stroke-dashoffset* pomiče se u pozitivnom ili negativnom smjeru i tako dobivamo iluziju da se linija pomiče duž putanje. *Stroke-dashoffset* u *@keyframes draw* određuje ključne okvire animacije. Animacija se tako pomiće s 0 na -360 duž putanje oko svoje osi za 4 sekunde uz beskonačno trajanje.

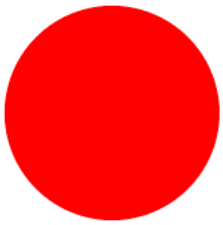


Slika 5-2. SVG drawin line animation

5.2. Transformacije oblika

Transformacije oblika (engl. *morphing shapes*) je SVG tehnika koja svojom animacijom omogućava postupno glatku transformaciju jedne forme u drugu. Ova animacija može stvoriti vrlo privlačne efekte jer oblici mogu prelaziti iz jednog oblika u drugi na fluidan način. Oblici se definiraju pomoću različitih elemenata poput *path, circle, rect"* itd. Kod pretvorbe oblika koristi se <*path*> koji omogućuje definiranje složenih putanja koje predstavljaju oblike. Animaciju postižemo definiranjem putanje sa atributom "d", koji definira putanju između dva ili više stanja. Glavni elementi *morphing shapes* animacija su putanje za definiranje složenih oblika putem niza koordinata i naredbi. *Tweening* proces interpolacije između putanja, gdje se izračunavaju međustanja kako bih se stvorio efekt glatke tranzicije.

Pregled primjera animacije prikazuje prvu putanju (M50,10 A40,40 0 1,1 49.9,10 Z). Ovo je eliptični luk koji počinje od točke (50,10) i završava na vrlo bliskoj točki (49.9,10), tvoreći puni luk. Druga putanja (M50,10 L61,35 ... L39,35 Z). Ovo je oblik zvijezde ili poligona s više linija spojenih između točaka, stvarajući kutove i vrhove. Treća putanja (M50,10 A40,40 0 1,1 49.9,10 Z). Ovdje se vraćamo na početni eliptični luk, završavajući ciklus animacije.



Slika 5-3. Morphing shapes stanje 1

Slika 5-4. Morphing shapes stanje 2

5.3 Crtanje putanje

Odnosi se na tehniku crtanja putanja (engl. *path drawing*) gdje se linije postupno iscrtavaju tijekom određeno definiranog vremena, stvarajući efekt animiranog crtanja. Ova tehnika se često koristi za animiranje rubova crteža i logotipa, ispis rukopisa ili bilo kojih drugih grafičkih elemenata koji se definiraju pomoću SVG *path* elemenata. U SVG grafici *path* element koristi se za definiranje složenih oblika pomoću niza koordinata i naredbi koje definiraju određene krivulje, linije i razne druge segmente. Putanju možemo animirati koristeći „*stroke-dasharray*“ i „*stroke-dashoffset*“ svojstva za stvaranje efekta crtanja.

Stroke-dasharray definira uzorak crtice i razmak koji se koristi za crtanje linija. Postavljanje vrijednosti ovog svojstva jednakoj duljini putanje, linija će se iscrtati od početka do samoga kraja kao jedna puna linija. *Stroke-dashoffset* kontrolira početnu točku crtanja linije. Mijenjanjem ovog svojstva tijekom nekog vremena možemo stvoriti efekt crtanja linije. Ovakav tip animacije se najviše koristi kod animiranja logotipa ili ilustracija, kreiranja rukopisa i interaktivnih crteža.

Slika 5-5. Iscrtavanje srca pomoću Path drawing animacije

6. Prednosti, nedostaci i usporedbe tehnologija

CSS animacije su idealne za jednostavne, glatke prijelaze i transformacije, posebno za statične i osnovne animacije. JavaScript animacije nude najveću fleksibilnost i kontrolu za kompleksne i interaktivne animacije, ali mogu biti teže za optimizaciju i zahtjevaju više linija koda. SVG animacije su izvrsne za vektorske grafike i animacije koje trebaju visoku kvalitetu i skalabilnost, ali mogu biti zahtjevne za složenije efekte. Kombinacijom ovih tehnologija možemo omogućiti izradu bogatih i dinamičnih web animacija, ovisno o potrebama i zahtjevima projekta koji se radi.

6.1. CSS animacije

CSS animacije omogućuju dodavanje dinamičkih vizualnih efekata na HTML elemente isključivo pomoću CSS tehnologije. To uključuje razne prijelaze između stanja transformacije i složene animacije definirane pomoću „*@keyframes*“. Prednost ove vrste animacija je jednostavna implementacija za osnovne animacije, perfomanse su često optimizirane jer se koristi grafički hardver za izvođenje grafičkih operacija koje uključuju prikaz slika i animacija (engl. *hardware-accelerated rendering*) što može poboljšati perfomanse u usporedbi s nekim JS animacijama. Animacije su definirane u CSS-u što rezultira čisti kod.

Nedostaci ove vrste animacija su mala kontrola i fleksibilnost u odnosu na JavaScript, osobito za složenije animacije koje zahtijevaju dinamičko računanje. Neke naprednije animacije jako je teško realizirati samo s CSS-om, osobito one koje zahtijevaju složene dinamičke promjene.

6.2. JavaScript animacije

JavaScript animacije se koriste za stvaranje kompleksnih i dinamičkih animacija koje zahtijevaju više kontrole i fleksibilnosti. Uključuju korištenje raznih metoda poput „*requestAnimationFrame*“, „*tweening*“ i animacija na HTML5 <*canvas*> elementu. Nudi visoku kontrolu i fleksibilnost za dinamičko upravljanje animacijama i složene logike. To je jako korisno kod kompleksnih i interaktivnih animacija. Lako je manipulirati animacijama na temelju korisničkog unosa ili drugih promjena u aplikaciji. Također može koristiti integraciju animacija sa logikom aplikacije ili podacima u realnom vremenu.

Problem dolazi kod animacija koje zahtijevaju mnogo računanja ili su loše optimizirane što dovodi do sporijeg rada i utjeće na perfomanse u aplikaciji. Sama složenost zadatka za dodatnu kontrolu i

upravljanje animacija povećava složenost i veličinu koda. Također određeni internetski preglednici ne podržavaju novije funkcije pa ih je nemoguće koristiti svugdje.

6.3. SVG animacije

SVG animacije su idealne za izradu dinamičkih i interaktivnih web grafika. Svojim iscrtavanjem linija i oblika u stvarnom vremenu stvara efekt crtanih ilustracija i prikazuje proces dizajniranja grafike. Pružaju visoku kvalitetu sadržaja zbog svog vektorskog oblika koji zadržava visoku kvalitetu pri bilo kojoj veličini. Zbog dobre skalabilnosti dobro funkcioniraju na različitim rezolucijama i veličinama ekrana. Sama manipulacija grafike može se lako integrirati sa JavaScriptom za dinamičke efekte.

Kao i kod JavaScript-a kod složenijih animacija i velike grafike SVG postaje težak za optimizaciju i upravljanje. Uređaji koji se koriste su također bitni zbog vrlo kompleksni i brojni animacija koje mogu utjecati na perfomanse preglednika. Stariji internetski ili mobilni preglednici ne podržavaju sve značajke SVG elemenata pa treba voditi računa i o tome gdje se koristi.

6.4. Usporedba CSS, SVG i JS

Ova tri pristupa imaju svoje prednosti i nedostatke u specifičnim scenarijima, tako da ovaj primjer demonstrirati njihovu najbolju svrhu. Obzirom da se SVG koristi za skalabilnu i vektorsku grafiku, CSS za jednostavne i elegantne animacije, te JS za složenije interaktivne animacije. Ovaj primjer prikazuje animaciju dva kvadra sa navedenim osobnostima. CSS dio animacije sastoji se od jednostavnih animacija rotacije i promjena boje kvadra kada se pređe mišem preko njega. Animacija se nadodaje na SVG element i prikazuje idealan primjer kako jedno drugo nadopunjuju.

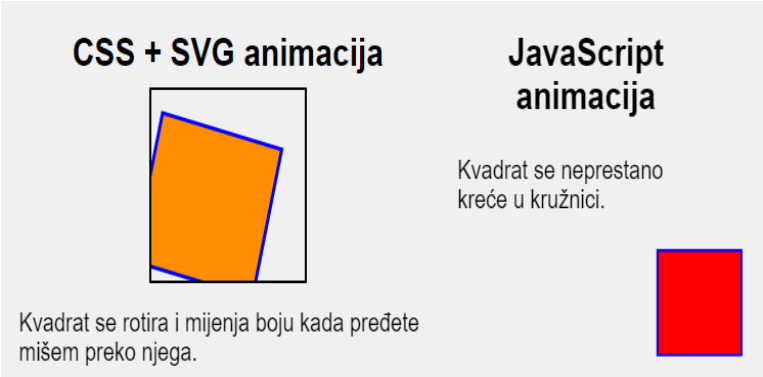
```
/* CSS animacija za jednostavan hover efekt */
.awesome-square {
    transition: transform 1s, fill 0.5s;
    cursor: pointer;
}
/* CSS hover efekt - rotacija i promjena boje */
.awesome-square:hover {
    transform: rotate(360deg);
    fill: orange;
}
```

SVG prikazuje kvadrat koji se skalabilno prilagođava veličini zaslona i ne gubi kvalitetu prilikom zumiranja.

```
<svg viewBox="0 0 100 100" style="width: 200px; height: 200px; border: 1px solid black;">
    <rect
        class="awesome-square"
        x="10" y="10"
        width="80" height="80"
        fill="red" stroke="blue" stroke-width="2">
    </rect>
</svg>
```

JavaScript prikazuje složeniju animaciju koja koristi „*requestAnimationFrame()*“ za kontinuiranu animaciju kvadrada koji se kreće u krugu. Koriste se matematički dinamički izračuni i napredne interakcije za upravljanje pozicijama u realnom vremenu .

```
const jsSquare = document.querySelector('.js-square');
let angle = 0;
function animate() {
    angle += 1;
    const x = 200 + 100 * Math.cos(angle * Math.PI / 180);
    const y = 200 + 100 * Math.sin(angle * Math.PI / 180);
    jsSquare.style.transform = `translate(${x}px, ${y}px)`;
    requestAnimationFrame(animate);
}
animate();
```



Slika 6-1 Usporedba

7. Zaključak

Web animacije predstavljaju ključan element u poboljšanju korisničkog iskustva na webu, dodajući dinamiku, interaktivnost i estetiku. Različite tehnologije, poput CSS-a, SVG-a i JavaScripta, nude različite pristupe i mogućnosti za implementaciju animacija, prilagođene vizualne efekte i interaktivne elemente. CSS animacije su jednostavne za implementaciju i koriste se za osnovne animacije, prijelaze i efekte. One su učinkovite za jednostavne web stranice i elemente koji zahtijevaju minimalnu interakciju. Prednosti uključuju jednostavnost upotrebe i dobru performansu, dok ograničenja mogu biti vidljiva u složenijim animacijama.

JavaScript omogućuje najveću fleksibilnost i kontrolu nad animacijama. Biblioteke poput GSAP pružaju napredne značajke i optimizaciju performansi. JavaScript je posebno koristan za dinamičke animacije, kompleksne interakcije i integraciju s drugim web tehnologijama. SVG pruža vektorski pristup za definiranje grafika, što omogućuje skalabilnost i visoku kvalitetu na različitim uređajima. SVG se često koristi za statične i skalabilne grafike, ikone i animacije. Interaktivnost, jednostavna čitljivost i prilagodljivost čine SVG snažnim alatom za web dizajnere i programere.

Pravi pristup web animacijama ovisi o specifičnim zahtjevima projekta. CSS animacije su brze i jednostavne, SVG pruža skalabilnost i vektorizaciju, dok JavaScript daje potpunu kontrolu i fleksibilnost. Ponekad se kombinacija svih tehnologija koristi za postizanje najboljih rezultata, prilagođenih konkretnim potrebama web stranice ili aplikacije. Važno je uzeti u obzir performanse, pristupačnost i korisničko iskustvo prilikom odabira i implementacije web animacija.

8. Literatura

[1] A. Petrović: Kreiranje animacija na webu, Završni rad, FIDIT, Rijeka, 2020.

[2] https://medium.com/@milberferreira/the-history-of-web-animation-63b106c97fdf, zadnje gledano 30.01.2024.

[3] https://www.mojwebdizajn.net/skriptni-jezici/uvod-u-web-dizajn/css-animacije.php, zadnje gledano 30.01.2024.

[4] https://www.mojwebdizajn.net/skriptni-jezici/uvod-u-web-dizajn/css-animacije.php, zadnje gledano 30.01.2024.

[5] https://en.wikipedia.org/wiki/JavaScript, zadnje gledano 30.01.2024.

[6] https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame, zadnje gledano 30.01.2024.

[7] https://www.w3schools.com/jsref/met_win_requestanimationframe.asp, zadnje gledano 30.01.2024.

[8] https://en.wikipedia.org/wiki/Inbetweening, zadnje gledano 30.01.2024

[9] https://www.adobe.com/creativecloud/video/discover/tweening.html, zadnje gledano 30.01.2024.

[10] https://kakonapravitiwebstranicu.com/koja-je-uloga-canvas-elementa-u-html5-u-u-web-razvoju/, zadnje gledano 30.01.2024.

Popis slika

Sveučilište Sjever

SVEUČILIŠTE SJEVER

IZJAVA O AUTORSTVU

Završni/diplomski/specijalistički rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, Valentino Martines *(ime i prezime)* **pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor završnog rada pod naslovom** Izrada web animacija korištenjem CSS, SVG i JavaScript tehnologija *(upisati naslov)* **te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.**

Student/ica:
(upisati ime i prezime)

Valentino Martines

(vlastoručni potpis)

Sukladno članku 58., 59. i 61. Zakona o visokom obrazovanju i znanstvenoj djelatnosti završne/diplomske/specijalističke radove sveučilišta su dužna objaviti u roku od 30 dana od dana obrane na nacionalnom repozitoriju odnosno repozitoriju visokog učilišta.

Sukladno članku 111. Zakona o autorskom pravu i srodnim pravima student se ne može protiviti da se njegov završni rad stvoren na bilo kojem studiju na visokom učilištu učini dostupnim javnosti na odgovarajućoj javnoj mrežnoj bazi sveučilišne knjižnice, knjižnice sastavnice sveučilišta, knjižnice veleučilišta ili visoke škole i/ili na javnoj mrežnoj bazi završnih radova Nacionalne i sveučilišne knjižnice, sukladno zakonu kojim se uređuje umjetnička djelatnost i visoko obrazovanje.