Obogaćivanje web stranice animacijama

Butko, Tino

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University North / Sveučilište Sjever**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:122:121696

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-08-06**

Repository / Repozitorij:

University North Digital Repository

Sveučilište
Sjever

Završni rad br. 812/MM/2022

Obogaćivanje web stranice animacijama

Tino Butko, 4082/336

Varaždin, rujan 2022. godine

Sveučilište Sjever

Multimedija, oblikovanje i primjena

Završni rad br. 812/MM/2022

Obogaćivanje web stranice animacijama

Student

Tino Butko, 4082/336

Mentor

mr. sc. Vladimir Stanisavljević

Varaždin, rujan 2022. godine

Predgovor

Kao student hibridnog tehničko-umjetničkog smjera multimedije, ovim radom nastojim prikazati rezultat stečenih vještina na studiju koje ujedinjuju kreativni izražaj ostvaren tehničkim putem.

Duboko se zahvaljujem svim profesorima na studiju, osobito mentoru Vladimiru Stanisavljeviću, te obitelji i kolegama na pruženoj podršci i vodstvu kroz studiranje.

Sažetak

U ovom završnom radu je prikazan i opisan osnovni proces izrade web stranice, odnosno njenih *front-end* komponenti kao što su korisničko sučelje i grafički elementi, na demonstrativni način su ukomponirane određene funkcije očekivane od web stranice (*web shopa*), te naposljetku je implementirano mnoštvo animacija i efekata koji unaprjeđuju korisnički doživljaj i iskustvo na stranici. Rad je podijeljen na dva dijela; prvi, teorijski dio je okrenut prema povijesnom razvoju interneta i mrežnih sustava, te općoj teoriji računala, kodiranja i prirodi web stranica, dok je u drugom dijelu rada opisan praktični postupak izrade web shopa, navedeni su korišteni alati i metode te povučene paralele s prvim dijelom završnog rada. Svaki ključni trenutak izrade dokumentiran je tekstualno i vizualno (koristeći *screenshot*).

Ključne riječi: front-end, web shop, funkcije, animacije, Internet, mrežni sustavi, kodiranje, web stranica

This thesis showcases and describes the basic process of creating a web page, mainly its front-end components such as the user interface and other graphical elements, demonstratively incorporates some basic functions expected of a web page (web shop), and also implements various animations and effects which further enhance the user experience. The thesis is composed of two sections, the first of which focuses primarily on the historical development of the Internet and various networking systems, while also addressing the general theory of computers, coding and the nature of web pages, while the next section features detailed descriptions of the practical application of said methods to create a web shop. In this section, the tools and methods that were used in the process are also named and described, and a parallel to the first section of the thesis is drawn to ensure the completion of the abstract concept. Each crucial moment regarding the creation of the web page is documented textually and visually (using screenshots).

Keywords: front-end, web shop, functions, animations, Internet, networking systems, coding, web page

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

UNIVERSITY NORTH

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL Odjel za multimediju

STUDIJ preddiplomski stručni studij Multimedija, oblikovanje i primjena

PRISTUPNIK Tino Butko | JMBAG 4082/336

DATUM 27.09.2022. | KOLEGIJ Programski alati 3

NASLOV RADA Obogaćivanje web stranice animacijama

NASLOV RADA NA ENGL. JEZIKU Enhancement of a web page with animation

MENTOR Vladimir Stanisavljević | ZVANJE viši predavač

ČLANOVI POVJERENSTVA

1. doc.dr.sc. Andrija Bernik - predsjednik povjerenstva
2. Dražen Crčić, dipl.ing. predavač - član povjerenstva
3. mr.sc. Vladimir Stanisavljević, v.pred. - mentor
4. doc.dr. sc. Domagoj Frank - zamjenski član
5.

VŽ KC

MMI

Zadatak završnog rada

BROJ 812/MM/2022

OPIS

Dinamički elementi većine web stranica svode se na izmjenjivanje niza slika. U tu svrhu na raspolaganju je veliki broj izmjenjivača slika iz galerije. Za neke specifične potrebe, poput vizualne poduke raznih vještina ili nastavu, potrebne su tehnologije koje glatko izmjenjuju slike, mijenjaju dio slike ili preko slike dodaju razne dinamičke elemente tako da se dobije dojam animacije visoke kvalitete. Animacije na web stranicama se mogu izvesti pomoću standardnih HTML5 komponenti ili pomoću raznih programskih dodataka.

U radu je potrebno:
- izraditi web stranicu za koju je poželjno da ima animirane elemente,
- prikazati i usporediti osnovne tehnologije pomoću kojih se na stranicu dodaju animacije,
- na prethodno izrađenu statičku stranicu dodati animirane elemente koji unaprijeđuju korisnički doživljaj,
- demonstrirati razne mogućnosti animacija nekim od prikazanih tehnologija.

Detaljno opisati izvedbu web rješenja i programski kod korišten u radu. Dokumentirati sve potrebne korake pri izvedbi cjelovitog rješenja i opisati iskustva stečena pri kodiranju stranice. U zaključku razmotriti druge moguće varijante izvedbe sustava i eventualna naknadna poboljšanja.

REPUBLIKA HRVATSKA
SVEUČILIŠTE SJEVER

ZADATAK URUČEN 25.10.2022 | POTPIS MENTORA

Popis korištenih kratica

TCP/IP	Transmission Control Protocol/Internet Protocol
WWW	World Wide Web
HTTP	HyperText Transfer Protocol
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
IDE	Integrated Development Environment
IBM	International Business Machines (Corporation)
ENIAC	Electronic Numerical Integrator and Computer
COBOL	Common Business Oriented Language
CMS	Content Management System

Sadržaj

1. Uvod 1
2. Računala, Internet i web stranice 3
   2.1. Uloga informatičkih koncepta u web developmentu 5
   2.2. Front-end, back-end i CMS 6
   2.3. IDE i tehnička implementacija ideja 8
3. Web animacije i efekti 9
   3.1. Animacije u standardnim web jezicima 10
   3.2. Napredniji načini animiranja web elemenata 12
4. Praktični dio 14
   4.1. Dizajniranje logotipa 16
   4.2. Izrada glavnih sekcija i elemenata stranice, podjela 18
   4.3. Ostvarivanje funkcionalnosti pomoću JavaScripta 31
   4.4. Implementacija animiranih elemenata - CSS3 38
   4.5. Implementacija animiranih elemenata - jQuery 47
5. Analiza rezultata 55
   5.1. Zaglavlje 55
   5.2. Lijevi odjeljak 55
   5.3. Navigacija 57
   5.4. Prikaz proizvoda 58
   5.5. Konačni rezultat 58
6. Zaključak 61
7. Literatura 62

1. Uvod

Povijest računala kao pomagala ljudskoj djelatnosti seže sve do drevnih civilizacija, kao što su Kinezi ili civilizacije bliskog istoka, koji su koristili tzv. *abakus* za provođenje kalkulacija. Napretkom civilizacije i razvojem tehnologije ljudi otkrivaju i stvaraju sofisticiranije strojeve koji, između ostaloga, omogućuju i lakše baratanje informacijama kao i širenje istih. Prva računala imala su relativno jednostavnu svrhu; olakšavanje računanja i kalkulacija, te se kao takva prema funkciji mogu usporediti sa suvremenim kalkulatorima. Koncept medija, koji se razvio pojavom pisma u starom vijeku, tada još nije imao nikakve korelacije s računalima. Razvoj medija kroz povijest pratio je razvoj tehnologije; prelazak pisane riječi s kamene ploče na papir označilo je pojavu novog medija na sličan način kao i inicijalna pojava prvih modernih medija, kao što su radio ili televizija. Isto tako razvojem računala od njihove inicijalne pojave pa do danas može se uočiti proširenje upotrebe računala za različite svrhe i djelatnosti, jedna od kojih je i pojava Interneta kao medija, pa direktno s tim i pojava web stranica.

Web i web stranice jedan su od najbrže rastućih i danas najzastupljenijih medija, uvelike zahvaljujući širenju i brzom rastu tijekom 90-ih godina 20. stoljeća i početku 21. stoljeća. Potencijalno najveći korak u razvoju internetskih tehnologija bila je kreacija TCP/IP protokola početkom 1983. godine, koja je konačno omogućila međusobnu komunikaciju između različitih vrsta računala te tako stvorila univerzalni jezik interneta. Krajem tog desetljeća u Švicarskoj je razvijen World Wide Web, koji će nedugo zatim postati glavni mrežni informacijski sustav u svijetu. Paralelno pojavi World Wide Weba pojavile su se i prve web stranice kakve ih poznajemo danas, za što je primarno zaslužan Hypertext Transfer Protocol (HTTP). Tim protokolom vrši se razmjena i prikaz informacija na različitim računalima. Za upotpunjavanje funkcija HTTP-a osmišljen je i HTML (Hypertext Markup Language), koji u suštini definira elemente koji se prikazuju na web stranici. Neki od elemenata koje HTML podržava i može prikazati su slike, videozapisi i audiozapisi koji su ugrađeni u web stranicu, obični tekst, hiperveze i ostalo. Uzimajući u obzir važnost grafičkog prikaza i prezentacije elemenata, te zbog činjenice da je HTML po tom pitanju relativno ograničen, stvoren je i CSS (Cascading Style Sheets) opisni jezik. Svrha CSS-a sužava se na modificiranje grafičkih komponenti raznih elemenata definiranih u HTML-u web stranice, te tako omogućuje dizajneru web mjesta da grafički obogati svoju stranicu i poboljša njen opći reprezentativni grafički dojam. Još jedan od bitnih HTML elemenata su tzv. skripte, koje omogućuju dizajneru da provodi sekvencirani programski kod s ciljem modificiranja i dinamičke promjene pojedinih elemenata HTML koda. Programski (skriptni) jezik koji se koristi u ovu svrhu jest JavaScript, koji je radi lakšeg korištenja dizajniran da bude sličan programskom jeziku Java, pa otud proizlazi i njegovo ime.

JavaScript se izvodi u web pregledniku (softver za pregledavanje Interneta i web stranica) klijenta, odnosno korisnika Interneta, što ga čini *client-side* jezikom, za razliku od primjerice PHP-a, koji se izvodi na poslužitelju (*serveru*). Kod tradicionalne izrade web stranica, *front-end* dio kreacije obično se odnosi na ta tri jezika, HTML, CSS (opisni jezici) i JavaScript (skriptni). Za svrhe ovog rada, web stranica je izrađena na taj način, odnosno pisanjem koda u razvojnu okolinu (IDE, *Visual Studio Code*). Svaki dio koda podijeljen je prema funkciji i jeziku koji se koristi, pa je tako glavna okosnica (*framework*) stranice podijeljena na tri datoteke s tri različita proširenja; .html, .css i .js.

Što se tiče animacija, potrebno je implementirati razne tipove animiranih elemenata web stranice radi poboljšanja korisničkog doživljaja i lakše prezentacije informacija i stavki dostupnih na stranici. Za te svrhe dostupno je mnogo animacijskih tehnologija, neke naprednije i neke jednostavnije. Sve te tehnologije, međutim, mogu pronaći adekvatnu primjenu u izradi web shopa ili obične web stranice. Neke od značajnijih tehnologija za izradu animacija na web mjestima su standardne CSS tranzicije, kao i malo naprednije i novije CSS3 *keyframes* animacije, zatim jQuery animacije, koje se baziraju na standardnim JavaScript animacijama, ali s olakšanim pristupom i načinom kodiranja, a povećanom funkcionalnošću. Postoje i razne manje zastupljene tehnologije animiranja web stranica, kao što su WebGL, Scalable Vector Graphics, obični GIF-ovi, ali u fokusu ovog rada bit će one najzastupljenije i najnaprednije tehnologije animiranja.

Web stranica je dizajnirana s podjednakim fokusom na dizajn, kao i na osnovnu funkcionalnost. Kroz opis rada i koraka izrade ću slikovnim prikazima dočarati proces kodiranja određenih elemenata web stranice, te usporedno jedno pored drugog postaviti programski kod u IDE-u i rezultat koji isti daje na web stranici u pregledniku. Kod izrade web stranice mi je uz dizajn glavni cilj bila osnovna funkcionalnost, odnosno da na čim jednostavniji ali funkcionalni način implementiram mogućnosti kupovine prave robe na web shopu. Naravno, bez *back-end* komponenti stvarna kupovina nije moguća, ali kroz JavaScript funkcije imitiram dodavanje stavki u kupovnu košaricu, te se time elementi stranice dinamički mijenjaju i modificiraju. U procesu izrade stranice sam osim vlastitog znanja stečenog na studiju te specifično kolegijima Programski alati 2 i 3, također konzultirao i web stranice vezane uz kodiranje web stranica, primarno W3Schools, koje su mi osvježile znanje o opisnim i skriptnim jezicima web stranica, kao i dale uvid u određene specifičnosti tih jezika te mi omogućile da lakše tehnički ostvarim svoju ideju i viziju web stranice. Međutim, kao što je bilo navedeno u sažetku, prije demonstracije izrade web stranice potrebno je detaljno proći kroz teorijsku pozadinu Interneta, web stranica i općeg razvoja tehnologije koja je u početku i omogućila status weba kao vodećeg medija, čime će se baviti početni odlomci rada koji slijede u nastavku.

2. Računala, Internet i web stranice

U kontekstu informatičke pozadine weba i web stranica bitno je početi od samog početka, odnosno zajedničkog nazivnika cijele web tehnologije; računala. Povijest strojeva, naprava i pomagala (s iznimkom prije spomenutog abakusa) koja uživaju slične primjene kao suvremena računala može se datirati unatrag 200 godina. Početkom 19. stoljeća, točnije 1801. godine, francuski trgovac Joseph Marie Jacquard napravio je vlastitu modifikaciju stroja za tkanje koja je koristila bušene kartice kako bi se postupak tkanja olakšao i automatizirao. Njegov izum, koji se danas naziva žakardni tkalački stroj, jedan je od prvih povijesnih primjera automatizacije proizvodnog postupka koji zamjenjuje ljudski napor, te se zbog činjenice da su u dizajnu stroja ključnu ulogu imale bušene kartice, koje su kasnije bile korištene i u ranim računalima, može direktno povezati i sa modernim konceptom računala. Dvadeset godina kasnije je engleski matematičar Charles Babbage osmislio prvi stroj koji je mogao računati s tablicama brojeva te rješavati polinomne izraze. Njegov tzv. diferencijalni stroj pokretala je para. Međutim, unatoč potpori britanske vlade, zbog ograničenja tehnologije njegovog vremena njegov dizajn nije bio u potpunosti implementiran u funkcionalan stroj. Također vrijedna spomena je engleska matematičarka Ada Lovelace, koja je 1848. kod prevođenja teksta o Babbageovom stroju iz francuskog u engleski, napisala vlastite bilješke u kojima je korak po korak opisala proces računanja Bernoullijevih brojeva koristeći Babbageov stroj, čime je zapravo stvorila i opisala prvi programski algoritam u povijesti. Ponovna uporaba bušenih kartica u primijenjenom strojarstvu pojavljuje se pred kraj stoljeća, kada je Herman Hollerith izumio stroj koji mu je pomogao s kalkulacijama američkog popisa stanovništva te godine. Hollerith je također poznat i kao osnivač tvrtke koja je kasnije postala IBM.

Na prijelazu u 20. stoljeće bitno je spomenuti jednog od glavnih pionira informatičke tehnologije i razvoja računala kroz cijelu povijest, Alana Turinga. Turing je bio britanski znanstvenik i matematičar čije je životno djelo bio tzv. Turingov stroj. Ključna svrha Turingovog stroja je obračunavanje mogućnosti i sposobnosti komputacije određenih problema, odnosno istraživanje granica teoretskog računala i što ono može izračunati. Turingove teorije su postavile temelje za modernu računalnu znanost i osigurale i ojačale razvoj informacijske tehnologije. Gradeći na Turingovim postignućima, ostatak dvadesetog stoljeća doživio je eksponencijalni razvoj tehnoloških napredaka koji se nastavlja i dan danas. Nakon drugog svjetskog rata u upotrebi se pojavljuju prva prava računala, prvo od kojih je tzv. ENIAC (*Electronic Numerical Integrator and Calculator*), koje se obično povezuje sa standardnim slikama prvih računala kao glomaznih strojeva koji su zauzimali cijele prostorije u poslovnim objektima.

Nekoliko godina nakon prve praktične uporabe računala kao ENIAC se pojavljuje i prvi programski jezik, COBOL (*Common Business Oriented Language*) pod pionirstvom američke računalne znanstvenice Grace Hopper. Kao prvi programski jezik, COBOL je postavio temelje za razvoj softverskog inženjerstva i uspostave adekvatne komunikacije između znanstvenika ili programera te računalnog hardvera, te je kao takav i zajednički predak našim opisnim jezicima HTML-u i CSS-u i skriptnom jeziku JavaScriptu. Pred kraj 50-ih godina 20. stoljeća također je dizajniran i prvi integrirani sklop, te tako i prvi računalni čip. Desetljeće nakon toga bilo je značajno za razvoj računala i interneta, s obzirom da su se u znanstvenim krugovima pojavljivala prva komercijalna računala kakve ih danas poznajemo, te prvi mrežni sustavi poput Interneta.

Komercijalizacijom računalnih sustava i internetskih usluga, njihova se upotreba proširila s isključivo vojne i vladine na osobnu uporabu, s obzirom da su računala dospjela u domove građana te su dobili pristup internetu pomoću World Wide Weba. S rastućim brojem korisnika interneta, potrebno je bilo imati standardiziran način prikaza informacija na njemu, te je stoga 1993. godine Tim Berners-Lee finalizirao prvu verziju HTML jezika za korištenje na web stranicama. CSS je uslijedio godinu dana kasnije, kao produkt kolaborativnog rada između norveškog programera Hakona Wiuma Liea, Tima Bernersa-Leeja i Roberta Cailliaua, dok je JavaScript patentirao Brendan Eich u 1995. Kombinacijom tih dviju opisnih jezika i skriptnim jezikom JavaScripta, razvoj web stranica (web development) dosegao je novu razinu napretka, te su postavili temelje za današnji web.

Slika 1 Logotipovi HTML-a, CSS-a i JavaScripta

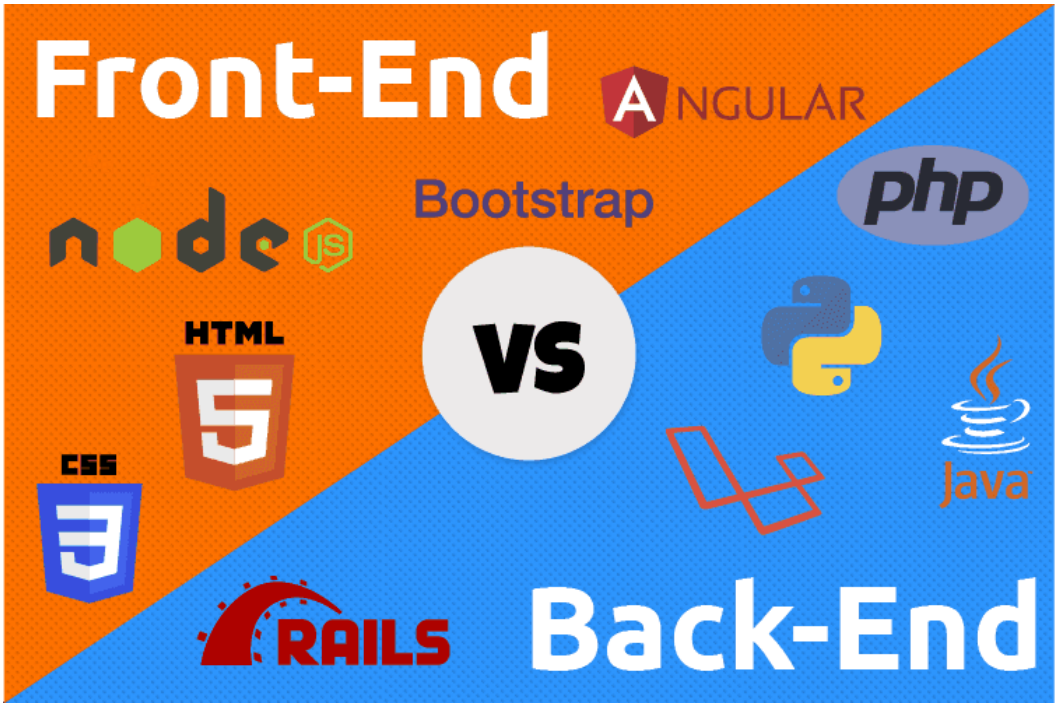
2.1. Uloga informatičkih koncepta u web developmentu

Zbog same arhitekture i dizajna računala, neizbježno je da i svi derivativni koncepti direktno vezani uz informatiku budu bazirani na istim principima i teorijskim načelima, pa su stoga i temelji web developmenta ustanovljeni u kodiranju, odnosno strukturiranim tekstualnim uputama koje su direktno upućene entitetu koji te upute može interpretirati, bio to hardver kao u programiranju niske razine ili interpretirajući softver kao u jednostavnijim programskim jezicima više razine. Specifično u kontekstu web developmenta, te standardizirane upute prima web preglednik koji prema njima vrši konstrukciju stranice, pritom kombinirajući razne elemente iz sva tri *front-end* jezika te dopunjujući ih njihovim odgovarajućim *back-end* komponentama. Kod usporedbe standardnih računalnih programskih jezika koji se koriste primjerice za izradu desktop aplikacija s *front-end* jezicima web developmenta, prema sintaksi i funkciji su u nekim aspektima značajno drugačiji i u nekim slučajevima kompliciraniji, s obzirom da je velika većina sofisticiranijih programskih jezika bazirana na objektno orijentiranom programiranju dok su HTML, CSS i JavaScript relativno jednostavni jezici s manjim krajnjim ciljevima uporabe. S druge strane, sličnosti između spomenutih jezika se također ne mogu zanemariti; s obzirom da obje grupe jezika imaju zajedničke korijene i zajedničke pretke (COBOL, BASIC itd.), pa su stoga obje naslijedile i određene načine korištenja i primjene logičkih načela, te će korisnicima koji su rabili obje grupe ovih programskih jezika sličnosti odmah postati evidentne. Oba stila jezika zahtijevaju isti logički orijentiran način razmišljanja te formuliranja ideja i algoritama (u kontekstu algoritma HTML i CSS su izuzetak), te će tako bilo tko vrstan u pisanju programskog koda u naprednijim jezicima automatski lako savladati i *front-end* web development jezike.

Slika 2 Najpoznatiji programski jezici današnjice

2.2. Front-end, back-end i CMS

Paralelno komercijalizaciji Interneta i širenju njegovog utjecaja počeli su se razvijati i napredniji načini prikaza informacija na webu, koje je predvodio Tim Berners-Lee sa svojom kreacijom HTML jezika. HTML te kroz nekoliko godina CSS i JavaScript s vremenom su se ujedinili u funkcionalnu cjelinu koja danas podrazumijeva *front-end* komponentu web developmenta. S druge strane svake funkcionalne stranice na webu postoji i *back-end* komponenta, čija je svrha podržavati ključnu „programsku" podršku web stranice. *Back-end* dio web stranica uvijek se nalazi na poslužiteljskoj strani, te se programski kod *back-end* dijela isto tako uvijek vrši na serveru (poslužitelju), umjesto na računalu i pregledniku klijenta kao *front-end* komponente. *Back-end* funkcionalnosti ključne su i obvezatne za sve web stranice koje su stvorene sa svrhom korisničke interaktivnosti; tipične značajke web stranica koje zahtijevaju *back-end* funkcionalnosti su registracija te prijava i odjava (*login*), povezivanje s bazama podataka, spremanje i dobavljanje podataka i informacija (direktna korelacija s prijašnjim značajkama), te naposljetku osiguravanje potpune funkcionalnosti *front-end* komponenti. Sve u svemu, *front-end* dio web stranica uvijek podrazumijeva „grafičke" komponente, odnosno ono što se korisniku na ekranu prikazuje te uz to vezane skriptne funkcije (JavaScript), dok se *back-end* komponente odnose na samu funkcionalnost stranice, kako ona komunicira i vrši interakciju sa serverom, te koje se informacije skladište i kako se one procesuiraju. U kontekstu mojeg rada, kao što je već bilo spomenuto ranije, fokus je u potpunosti na *front-end*, te s dodanim JavaScript funkcijama koje imitiraju određene *back-end* komponente, specifično kupovinu proizvoda, dodavanjem u košaricu te dinamičnom promjenom stranice preko korisničkog unosa.



Slika 3 Front-end/back-end

Još jedna bitna značajka web developmenta koju je ključno spomenuti jest tzv. Content Management System (CMS). CMS je, barem u usporedbi s tradicionalnim *front-end* jezicima, relativno novi fenomen. CMS se u osnovi odnosi na softverske pakete koji korisniku odnosno dizajneru omogućuju izradu web mjesta bez uporabe HTML-a, CSS-a i JavaScripta, na način da stranicu dizajnira pomoću intuitivnijeg (u usporedbi s kodiranjem) grafičkog sučelja. Neki od najpoznatijih CMS-a današnjice su WordPress, Wix, SquareSpace i slični paketi. Još jedna ključna razlika između CMS-a i tradicionalnog *front-end* developmenta jest ta što većina CMS sustava nudi opciju *hostinga* (poslužiteljskih usluga) korisničkih stranica na njihovim *Cloud* servisima, što može uključivati plaćene planove serverskih servisa ili besplatne, svaki sa svojim određenim značajkama, te je korisniku stoga ponuđeno više opcija i na njemu je da izabere onu adekvatnu za svoje ciljeve i misiju. U kontrastu prema tome, klasični *front-end* sam po sebi ne posjeduje te napredne značajke čija je svrha prosječnom korisniku olakšati cjelokupni postupak, pa stoga od njega zahtijeva aktivniji i izravniji pristup tom aspektu developmenta. Nakon kompletne izrade stranice, s pristupačnim *front-end* dijelom i funkcionalnim *back-endom*, kreator web mjesta mora odabrati adekvatan poslužiteljski servis na kojem će njegova stranica biti postavljena. Da ponovno povučem paralelu na svoj vlastiti rad; moj izbor je bio ne oslanjati se na CMS usluge za izradu stranice jer smatram da klasičnim načinom izrade, koristeći HTML, CSS i JavaScript, kreator web mjesta dobiva najpotpuniji oblik direktne kontrole nad sadržajem koji stvara. Klasični *front-end* opisni i skriptni jezici dizajneru stranice daju mogućnost izravne manipulacije elementima web mjesta, kao i osjećaj potpune kontrole nad svime što se na stranici pojavljuje, što je moja preferencija u tom aspektu i što je glavni razlog zašto sam odabrao izraditi web stranicu klasičnim putem naspram CMS metode.

Slika 4 Primjeri CMS servisa

2.3. IDE i tehnička implementacija ideja

Priroda kodiranja web stranica jamči da korisnik odnosno dizajner mora odabrati adekvatnu razvojnu okolinu kako bi sam sebi olakšao i pojednostavio proces izrade web stranice. Iako je činjenica da se HTML kod može pisati i u najobičnijem uređivaču teksta poput Microsoftovog *Notepada*, pisanjem koda na takav način i u takvom okruženju kreator stranice potencijalno ograničava vlastite sposobnosti i na posljetku izgled i funkcionalnost svoje stranice. U današnjem svijetu web developmenta te programiranja općenito postoje mnoga razvojna okruženja, od kojih su neka dizajnirana s ciljem općenite i širokopojasne funkcionalnosti, dok su određena druga okruženja stvorena sa specifičnim tipom developmenta na umu. Neka od poznatijih razvojnih okruženja današnjih programerskih krugova, posebice u web developmentu, su Visual Studio (Code ili IDE), NetBeans i Atom. Integrirano razvojno okruženje korisniku omogućuje instalaciju raznih dodataka i *plugina* koji mogu biti specifični za određen tip pisanja koda, odnosno orijentirani na cilj korisnika, ili mogu pružati opću podršku koja znatno olakšava dopunjavanje i dovršavanje koda. Neke od značajki tih dodataka su automatsko dovršavanje koda koji korisnik upisuje, na sličan način kao što to rade i tražilice na web preglednicima, zatim otvaranje izbornika s dostupnim kodiranim naredbama i dodacima za određeni atribut ili element u kodu, i slično. Takvi dodaci značajno olakšavaju pisanje koda, te u usporedbi s pisanjem koda u običnim uređivačima teksta pružaju veliku prednost programeru i omogućavaju otključavanje punog potencijala njegovih sposobnosti i znanja.

Razvojne okoline striktno u kontekstu web dizajna i developmenta moraju imati podršku prvotno za sve formate datoteka prisutne u web dizajnu, koje podrazumijevaju sve *front-end* formate (.html, .css, .js), te podršku za PHP i *back-end* programiranje. Isto tako, korisno je da te okoline imaju „sekundarne“ značajke koje bi mogle doći u ruku programerima koji ih koriste, kao što je direktan pristup GitHubu kako bi programer mogao direktno iz IDE-a spremati svoj napredak na server ili *cloud*. Kroz učenje kodiranja i programskih jezika, kao na fakultetu tako i u slobodno vrijeme, koristio sam i stekao iskustva u nekoliko razvojnih okruženja, od kojih su najznačajniji bili NetBeans i Visual Studio Code. Web stranicu sam odlučio izraditi u Visual Studio Codeu, uz instalirane *plugine* za web development za olakšanje samog procesa izrade.

3. Web animacije i efekti

Prvom pojavom Interneta i njegovim probojem u svijet medija, kao nova i inovativna usluga uživao je u širokom doživljaju noviteta i jedinstvenosti, te se po svojem načinu korištenja i prenošenju informacija razlikovao i izdvajao od ostalih konkurentnih medija. S vremenom, međutim, taj novitet je počeo lagano isparavati, te su se stoga počele javljati potrebe za daljnjim poboljšanjima i unaprjeđenjima internetskih tehnologija. Jedan aspekt, odnosno smjer tih poboljšanja bio je korisnički doživljaj i iskustvo na web mjestu; s obzirom da su prve web stranice bile konstruirane isključivo u HTML dokumentima, personalizacija, dizajn i općeniti doživljaj korisničkog iskustva na web stranici bio je u sekundarnom planu. Primarni cilj tih prvih web stranica bio je izravan prijenos i dostupnost informacija. Širenjem popularnosti Interneta i web usluga te stizanjem istih u domove svakodnevnih ljudi, sljedeći korak u razvoju spomenutih tehnologija bio je očit; implementacija grafičkih i animiranih elemenata, proširenje programskih mogućnosti za dodavanje podrške za slike, fotografije, ilustracije i slične grafičke elemente te općenito unaprjeđivanje korisničkog iskustva.

Prvi korak u ovom smjeru bilo je osmišljavanje revolucionarnog grafičkog formata GIF (Graphics Interchange Format). Ovaj format je stvorila tvrtka CompuServe 1987. godine, te je njegova prvotna svrha bila dodavanje i sadržavanje boja u slikovnim datotekama, koje se zatim mogu lakše preuzeti s Interneta za razliku od konkurentnih formata tvrtka Apple ili Microsoft. Svojstva GIF formata za koje je danas poznat, kao što su animacija i podrška za prozirnu pozadinu, nisu bila prisutna u prvoj verziji, međutim dodana su nedugo kasnije u sljedećim revizijama formata. Animacija koristeći GIF format brzo je zaživjela, te su mnoge tvrtke iskoristile ovu tehnologiju za poboljšanje svojih logotipa i njihov prikaz na webu, primjerice s animacijama rotacije ili pomicanja, ili iluzijom 3D efekta pomoću animacije. GIF format zadržao je primarnu ulogu u web animacijama relativno dugo, sve do pojave Adobeove *Flash* tehnologije krajem 90-ih godina 20. stoljeća, nakon čega su animacije na webu postale sveprisutne. Adobe Flash kreiran je za svrhe stvaranja web animacija, Internet i desktop aplikacija, jednostavnijih video igara, bilo na desktopu, mobilnim uređajima ili na web mjestima. Zbog malog broja potencijalnih konkurenata, Adobe Flash zauzeo je vodeću ulogu u tadašnjem web *developmentu* i postao je najpopularniji alat za svrhe za koje je kreiran. Jačanjem popularnosti mobilnih uređaja, međutim, Flash je naišao na problem; mobilni uređaji nisu posjedovali jednake tehničke sposobnosti kao računala, pa su kod učitavanja i prikazivanja Flash sadržaja trošili više tehničkih resursa, pa tako i bateriju. Zbog toga je tvrtka Apple, primjerice, isključila podršku za Flash na vlastitim uređajima i zamijenila je vlastitim sustavima. Nedugo zatim su i programeri ostalih sustava uočili nedostatke Adobeovog Flasha i počeli s vremenom smanjivati i ukidati potporu.

3.1. Animacije u standardnim web jezicima

Nestankom Flasha dodatno je ojačala potreba za nadoknadom izgubljenih značajki web animacija, aplikacija i ostalih interaktivnih i grafičkih elemenata, pa su time tradicionalni jezici web developmenta, odnosno velika trojka HTML-a, CSS-a i JavaScripta u ovom aspektu unaprijedili vlastite tehnologije kako bi se utjecaj tog nedostatka minimizirao. U HTML jezik su u međuvremenu implementirani mnogi noviteti, kao što su podrška za dodavanje multimedijskih sadržaja poput slika, videozapisa i audio datoteka. JavaScript skriptni jezik unaprjeđen je na načine koji omogućavaju animiranje HTML elemenata na mnoge načine, neki od kojih su pomicanje, rotiranje, sakrivanje i prikazivanje elemenata. Za tu svrhu razvijene su određene programske biblioteke (engl. *library*), koje proširuju funkcionalnost programskog jezika i olakšavaju pristup određenim funkcijama te poboljšavaju kompatibilnost između istih. Jedna od tih biblioteka je tzv. „inalan“ (Interactive Algorithm Animations JavaScript Library), stvoren na Sveučilištu u Trnavi u Slovačkoj. Ova specifična biblioteka kreirana je da bude jednostavna za korištenje, ali efikasna i napredna u svojim tehničkim principima. Njena svrha je prvotno služiti kao alat za učenje i lakše razumijevanje određenih programskih struktura i koncepta, te je izrađena za uporabu od strane studenta ovog sveučilišta, ali je također kao javna publikacija dostupna i široj javnosti. Biblioteka nastoji podučiti studente o konceptu jednodimenzionalnih programskih polja, odnosno programskih nizova, te uspostavlja primjer ovog koncepta kroz primjer animacije. Neke od individualnih značajki ove programske biblioteke i njezinog povezanog primjera animacije su korištenje stupaca za prikaz polja i tipa polja, koja se dijele na sortirana, nesortirana i usporedna polja. Indeksi ovih polja i varijable koje te indekse spremaju su animirani, te ih korisnik može pokretati pomoću sučelja animacije. Ovaj specifični primjer odnosi se na autorovu animaciju korištenu za svrhe edukacije svojih studenata. Što se tiče kreacije same animacije, prvi korak jest izrada objekta koji sadrži sve interaktivne gumbove za kontrolu animacije. Nakon toga je potrebno definirati algoritam animacije uz nekoliko jednostavnih JavaScript funkcija, te po potrebi urediti objekt kontrolne ploče i dodati određene naknadne objekte. Ovom bibliotekom moguće je stvoriti nekoliko vrsta interaktivnih animacija vezanih uz svrhu koju je autor osmislio; prvi tip sadrži olakšanje vizualizacije određenih algoritama, odnosno algoritama koji primjerice zamjenjuju dvije varijable, zbrajaju elemente poretka/niza (polja), te ostale jednostavne operacije s jednodimenzionalnim programskim poljima. Druga dva tipa autorove animacije odnose se na rekurzivne i nerekurzivne algoritme te njihove načine funkcioniranja. Vrijedno je ponovno spomenuti da je ova specifična biblioteka JavaScript animacija stvorena specifično za potrebe ovog sveučilišta, te za njihove edukacijske taktike olakšavanja razumijevanja programskih struktura svojim studentima.

Ostale standardne primjene animacijskih tehnologija u tradicionalnim web development jezicima odnose se na osnovne animacije kreirane u CSS i JavaScript jezicima. HTML u samostalnoj primjeni ne sadrži podršku za animirane elemente, pa je stoga potrebno ove funkcionalnosti ostvariti u komplementarnim jezicima. CSS u svojoj osnovnoj strukturi sadrži nekoliko načina animiranja, prvi te relativno primitivniji od kojih su tranzicije. Osnovna svrha CSS tranzicija jest glatka promjena određenih stilskih atributa elementa kroz vremenski period. Naredbe vezane uz CSS tranzicije su `transition`, `transition-delay`, `transition-duration`, `transition-property`, `transition-timing-function`. *Delay* naredba određuje vremenski interval nakon kojeg će efekt tranzicije započeti kada se izvrši inicijalna akcija koja tranziciju pokreće, kao što je klik na gumb ili prelazak miša preko gumba (*hover*). *Duration* direktno određuje vremensko trajanje tranzicijskog efekta, dok *property* određuje po kojem atributu elementa će se tranzicija provesti. Konačno, *timing-function* određuje varijaciju u izvođenju brzine te tranzicije. CSS tranzicije relativno su jednostavnije od standardnih CSS3 animacija, koje često koriste *keyframes* atribute za preciznije određivanje tijeka i trajanja animacije. Tranzicije se koriste u situacijama u kojima je potrebno primijeniti jednostavnu ali efektivnu transformaciju elementa uz animaciju.

JavaScript primarno se koristi za ostvarivanje funkcionalnosti web mjesta te za mogućnosti dodavanja interaktivnih elementa, kao što su gumbovi i kontrolna sučelja. Zbog činjenice da je JavaScript skriptni programski jezik, a ne opisni jezik kao njegovi komplementarni jezici HTML i CSS, posjeduje jedinstvena svojstva, glavno od kojih je mogućnost pisanje sekvencijalnog programskog koda koji se također može svrstati u funkcije koje se zatim pozivaju naknadno u HTML kodu. U JavaScriptu je također moguće kodirati animacije i njihove elemente, međutim zbog činjenice da je tip jezika različitih od HTML-a i CSS-a, taj proces se vrši na drugačiji način. Za početak opisa JavaScript animacija potrebno je opisati osnovni način njihovog funkcioniranja; JavaScript je sposoban direktno mijenjati, modificirati i stvarati dinamički HTML kod, zbog činjenice da se kod kreacija bilo koje HTML web stranice generira njezina struktura, tzv. *Document Object Model*. Ovakav tip strukture skriptnom jeziku kao što je JavaScript daje direktnu moć nad dinamičkim mijenjanjem stranice. Ovaj isti princip dinamičkog mijenjanja HTML koda primjenjuje se i u implementaciji JavaScript animacija. Prije kodiranja same animacije potrebno je imati HTML elemente na koje će se animacija primjenjivati, te je poželjno da su oni stilizirani pomoću CSS-a. Animacija se kodira na način da se kroz određeni vremenski period pomoću funkcija mijenjaju vrijednosti stila elementa koji se animira. Ove promjene su tempirane, te se izvode ovisno o trenutnom položaju i stanju funkcije. Kada je ukupni vremenski interval dovoljno malen, spomenuta animacija izgleda fluidno i kontinuirano.

3.2. Napredniji načini animiranja web elemenata

Tradicionalni jezici web developmenta u tehničkom aspektu su sposobni za implementaciju i dinamičko dodavanje animacija u HTML kod samostalno, ali također imaju i određena ograničenja, osobito u usporedbi s naprednijim načinima obavljanja ovih zadaća. Prvi od tih naprednijih načina dodavanja animacija koji je vrijedan spomena jest „ekstenzija“ CSS opisnog jezika koja se javila kao novitet u CSS3 inačici, a to su CSS3 *keyframes* animacije. Ovim novitetom u CSS jeziku omogućeno je dinamičko mijenjanje svojstva animacije tijekom njezinog izvođenja. *Keyframes* su kontrolne točke u tijeku animacije na kojima se modificiraju atributi vezani uz CSS klasu na koju se animacija primjenjuje. Sličan princip kontrolnih točaka također se pojavljuje u ostalim medijskim platformama, primjerice kod uređivanja video ili audio materijala i primjene efekata na iste. Koncept *keyframesa* u CSS-u direktno je povezan sa svojstvima animacije, koje su analogne već spomenutim CSS tranzicijama. Slične su inicijalne strukture, ali uz blago različite primjene te razliku u naprednosti tehnologije. CSS3 animacije su blago naprednije od standardnih CSS tranzicija. Svojstva animacija slična su svojstvima tranzicija, odnosno također posjeduju svojstva kao što su *delay* i *duration*, međutim, kod animacija također se pojavljuju neka nova svojstva kao što su broj ponavljanja iste animacije, smjer animacije i identificiranje animacije dodavanjem imena za upotrebu *keyframes* tehnologije.

Kod svake kontrole točke (*keyframea*) moguće je dodati nove CSS atribute za promjenu postojeće klase, kao i modificirati i mijenjati stare. Prateći ovaj postupak možemo primjerice izmijeniti transparentnost (*opacity*) na stranici, pa napraviti tranziciju iz potpuno vidljive stranice na nevidljivu, s također mogućim varijacijama tijekom animacije. Ovo je glavna stavka po kojoj se CSS3 *keyframes* animacije razlikuju od standardnih CSS tranzicija te po čemu su naprednije od njih. Programer može mijenjati svojstva animacije u bilo kojem trenutku same animacije. Pozicija kontrolne točke u tijeku animacije određuje se postocima, pa će se tako početak animacije označiti s 0%, dok će se kraj animacije označiti sa 100%. Sve između ove dvije točke označuje se odgovarajućim postotkom, pa će tako primjerice polovina animacije biti označena s 50%, a četvrtina s 25%. Tip animacije određuje se u kodu klase na koju se animacija primjenjuje, dok se sama animacija kao i njezine kontrolne točke deklariraju i definiraju u zasebnoj klasi koja se naznačuje s oznakom @keyframes *ime-klase*.

Sljedeći napredniji tip animacije jesu jQuery animacije. jQuery je još jedna od biblioteka za JavaScript skriptni web development jezik. Za razliku od već spomenute „inalan“ biblioteke koja je razvijena isključivo za potrebe jednog zadatka u sklopu jednog sveučilišta, ova biblioteka razvijena je za širokopojasnu i globalnu upotrebu i olakšanje korištenja i pisanja JavaScript koda, te je zbog toga jQuery i postala jedna od najpopularnijih i najviše korištenih programskih

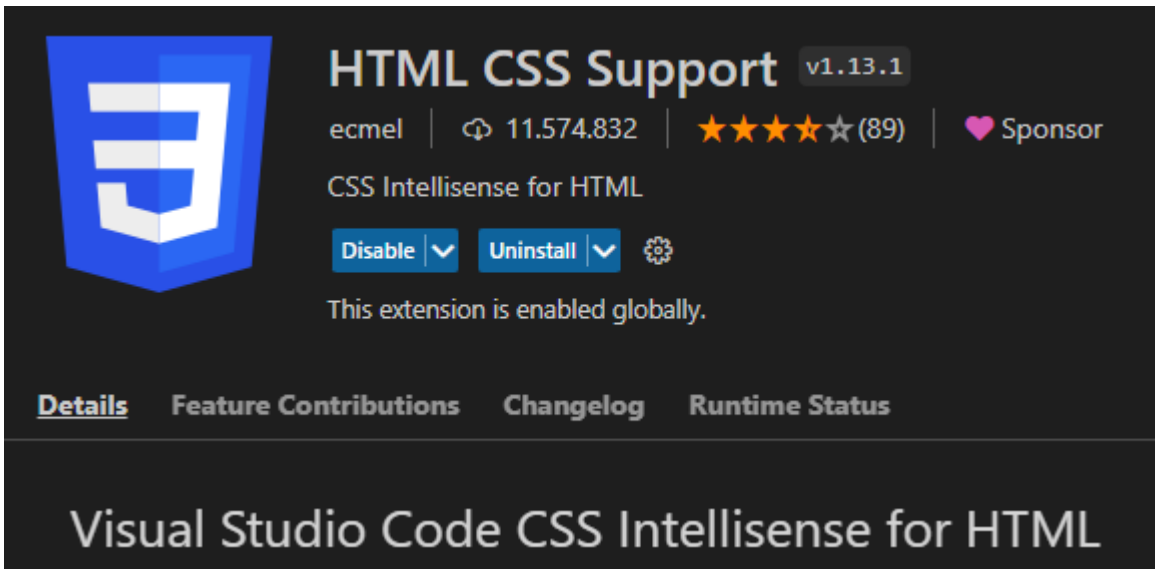
biblioteka za JavaScript. Sama biblioteka koristi se za još nekoliko primjena osim dinamičkog animiranja HTML elemenata, neki od kojih su povezivanje s bazama podataka, AJAX programiranje, te opće olakšavanje pisanja JavaScript koda kao što je već bilo spomenuto. Specifično u kontekstu jQuery animacija, sama biblioteka ima ukomponirane mnoge metode i funkcije za laganu implementaciju animacija na HTML elemente. Neke od tih metoda su tzv. *slideToggle, animate, fadeIn, fadeOut, fadeToggle* i ostali slični efekti. Za razliku od JavaScripta, ove animacije i efekti mnogo su lakši za implementirati u kod, s obzirom da su već od prije ukomponirani u sklop biblioteke i sve što je potrebno je pozvati ih pomoću njihovih odgovarajućih naredbi. Ekvivalente ovih animacija u JavaScriptu bi bilo potrebno individualno kodirati, dok je taj posao za programera u jQueryu već od prije riješen. Zbog ove činjenice, može se argumentirati da je jQuery znatno bolji kod implementacije animacija u HTML dokument od tradicionalnog JavaScripta, i njegovo korištenje kao direktne ekstenzije JavaScript funkcionalnosti uvijek je opravdano u kontekstu kodiranja animacija. CSS3 i jQuery animacije zajedno se mogu svrstati u isti rang u hijerarhiji programskih rješenja za dodavanje animacija na web mjesta.

Postoji mnoštvo naprednijih načina dodavanja animacija na web mjesta, te oni glavni su već spomenuti i opisani. Neki manje korišteni, ali i dalje napredni načini implementacija animacija u web stranice su primjerice WebGL (Web Graphics Library), koji je također direktno vezan s JavaScriptom kao i jQuery te programerima omogućuje dodavanje interaktivnih grafičkih elemenata i animacija. Najčešće se koristi za dodavanje raznih 3D elemenata, logotipa i ostalih sličnih grafičkih struktura.

Slika 5 jQuery logotip

4. Praktični dio

Praktični dio rada započinje pripremanjem svih potrebnih preduvjeta za uspješno obavljanje zadatka. Prvi od tih ključnih preduvjeta jest instalacija razvojnog okruženja; potrebno je odabrati IDE koji bi bio najprikladniji za rad na izradi web stranice, gdje se može birati između više opcija, neke od kojih su spomenute iznad (NetBeans, Atom…). Izabrao sam Visual Studio Code zbog osobnih preferencija i iskustva u izradi u tom razvojnom okruženju. Nakon instalacije samog softvera, to jest razvojnog okruženja, potrebno ga je „upotpuniti“ dodavanjem *plugina* koji će olakšati proces razvoja stranice.

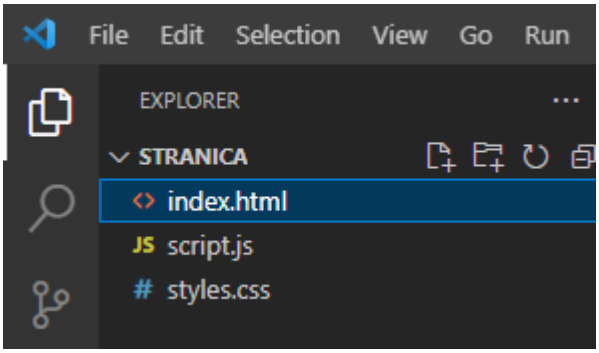


Slika 6 HTML plugin

Slika 7 JavaScript plugin

Ove softverske ekstenzije za razvojno okruženje omogućuju korisniku (programeru) automatsku implementaciju programskih jezika i njihovih programskih knjižnica (*library*), te je u njihove *plugine* također ukomponiran tzv. IntelliSense koji je glavna značajka koju sam spominjao ranije; automatski dovršava kod za programera te znatno olakšava proces samog pisanja.

Nakon instalacije ekstenzija u razvojno okruženje, potrebno je kreirati datoteke u koje će se pisati izvorni kod web stranice. S obzirom da se radi o klasičnom *front-end* kodiranju, potrebno je kreirati zasebne datoteke za HTML kod, za CSS kod i za JavaScript kod. Budući da je HTML okosnica cijele stranice i služi kao osnovni temelj, preuzet će glavnu ulogu te će se CSS i JavaScript datoteke vezati na njega. Također, za optimalnu funkcionalnost i minimiziranje potencijalnih problema, poželjno je postaviti sve tri datoteke u istu mapu na disku ili serveru. Datoteke se mogu kreirati u operativnom sustavu, primjerice u Windowsu, na sljedeći način; kreira se nova tekstualna datoteka te se zatim promijeni njezino ime (npr. u index) pa zatim proširenje, iz .txt u .html. Konačni rezultat je datoteka imena index.html, što je i standardni naziv za početne HTML datoteke. Također, datoteke se mogu kreirati direktno iz razvojnog okruženja koristeći izbornike File -> New File… gdje zatim odaberemo tip datoteke i naziv.



Slika 8 Zasebne datoteke stranice

Sljedeći ključni korak u pripremi izrade stranice jest povezati sve tri datoteke u jednu kako bi nesmetano funkcionirale u konjunkciji. Kao što je već spomenuto, HTML datoteka je okosnica cijele stranice, stoga se CSS i JavaScript datoteke direktno povezuju u stranicu preko HTML koda. Kako bismo spojili CSS datoteku s glavnom HTML datotekom, potrebno je u HTML kod upisati sljedeće:

```
<!DOCTYPE html>

<html>
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="styles.css">
</head>
```

Slika 9 Povezivanje CSS datoteke s HTML okosnicom

CSS datoteku poželjno je uključiti u dokument na samom vrhu, odnosno u *headu* dokumenta. Za to se koristi HTML atribut `link`, koji zahtijeva argument `rel` kojim određujemo tip dokumenta na koji se atribut veže, te zatim preko argumenta `href` određujemo točno ime i proširenje datoteke na koju se vežemo.

S druge strane, uspostavljanje veze na skriptni dio stranice, odnosno na JavaScript datoteku, se vrši malo drugačije. Za glavnu skriptu dokumenta poželjno je da se stavlja na kraju samog dokumenta, kako bi se osigurala optimalna funkcionalnost i izbjegli potencijalni konflikti s ostalim skriptama prisutnima u HTML kodu. Veza na JavaScript datoteku prikazana je na slici ispod:

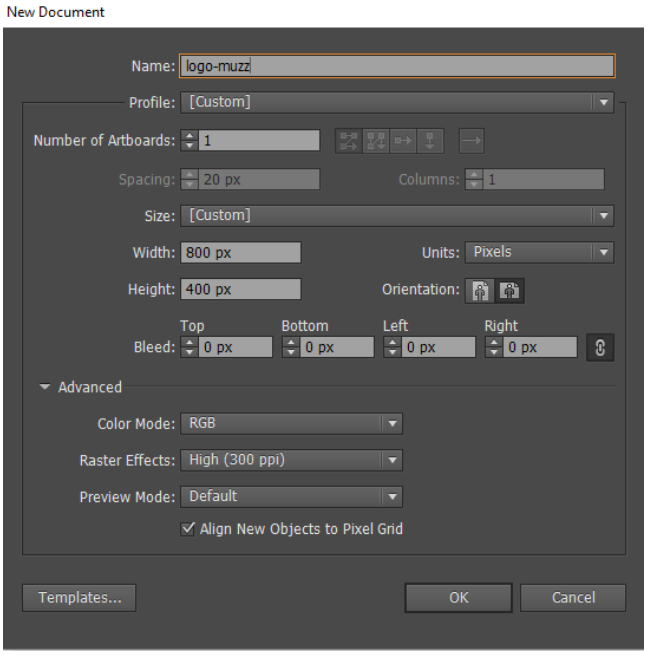
```
</body>

<script src="script.js"></script>
</html>
```

Slika 10 Povezivanje JavaScript datoteke s HTML okosnicom

4.1. Dizajniranje logotipa

Nakon uspostave samih temelja web stranice te prije početka izrade elemenata iste, potrebno je dizajnirati logotip za web shop. Najprikladniji alat za taj dio zadatka je Adobe Illustrator.

Za logo sam odabrao jednostavan dizajn, te je glavni element logotipa njegova shema boja, a cilj je da se vizualno podudara s ostatkom web stranice. Za početak potrebno je otvoriti novi dokument u Illustratoru te postaviti optimalne postavke rezolucije, orijentacije i ostalih bitnih elemenata početnog *artboarda*.



Slika 11 Početne postavke dokumenta logotipa

Glavne početne postavke dokumenta su sljedeće: širina dokumenta postavljena je na 800 piksela, dok visina iznosi 400 piksela. Orijentacija dokumenta je vodoravna, dok je *bleed* postavljen na 0px jer dokument nije primjeren za tisak. Model boja je RGB s visokim brojem piksela po inču (300).

S obzirom da sam se prije započinjanja dizajna loga odlučio za jednostavni dizajn, prvi korak kod izrade jest odabir dobrog fonta. Pretragom na stranici 1001fonts koja sadrži mnoštvo fontova besplatnih za osobnu i komercijalnu uporabu, odlučio sam se za font po imenu *ethnocentric* zbog njegovog futurističkog dizajna. Preuzeo sam .otf (OpenType font) datoteku fonta te ju instalirao na svoje računalo. Nakon toga sam u Illustratoru jednostavno iskoristio *Type Tool* te napisao ime web shopa koristeći preuzeti font, te sam potom promijenio boju teksta u narančastu koristeći *Fill*. Nakon toga sam označio tekst, kliknuo na izbornik *Type* i na funkciju *Create Outlines*, čime sam tekst pretvorio u objekte koji se mogu dalje uređivati. Time sam također otvorio mogućnost stvaranja dodatne vanjske konture na tekstu, što mi je bio i cilj. Konturama teksta dodao sam još jedan *Stroke*, te ga postavio u crnu boju, podebljao ga i na kraju dobio sljedeći rezultat, za što sam na posljetku i odlučio da će biti konačan logotip.

Slika 12 Gotov logotip web shopa

Nakon završetka dizajna logotipa, spremio sam dokument kao Illustrator projekt (.ai format) kako bi, ukoliko je to potrebno, mogao bez problema vektorski mijenjati logotip tako da se ne gubi kvaliteta. Također sam izvršio *Export* logotipa u .png format za implementaciju u HTML kod i korištenje na web stranici, i to u dvije varijante, sa crnim konturama teksta i sa bijelima, u slučaju potrebe stavljanja logotipa na crnu pozadinu.

4.2. Izrada glavnih sekcija i elemenata stranice, podjela

Nakon uspješno provedene izrade logotipa, može započeti rad na samoj web stranici. S obzirom da su sve tri potrebne datoteke za izradu već stvorene, jedino što je sad potrebno jest otvoriti ih u razvojnom okruženju (Visual Studio Code). Ovo specifično razvojno okruženje, uz standardno otvaranje svake datoteke individualno, također nudi opciju otvaranja cijele mape radi bolje pristupačnosti. Nakon otvaranja svih datoteka sve je spremno za početak rada. Početni korak u izradi stranice jest definiranje glavnih elemenata podjele stranice u HTML kodu. Ti elementi dijele stranicu na standardne dijelove poznate u bilo kojem obliku formatiranja, neki od kojih su *header*, *footer*, *body* i slično, te oni sveukupno čine kompletnu cjelinu i upotpunjuju funkcionalnost stranice u kontekstu samog HTML koda. U ovoj stranici nije bilo specifične potrebe za implementacijom *headera* i *footera,* pa su jedini glavni djeljivi elementi prisutni u kodu *head*, čija je jedina svrha povezivanje HTML koda s CSS datotekom, zatim *title,* u kojem je napisan naslov stranice koji se pojavljuje na prikazu kartice u web pregledniku, te najvažnije, *body*, u kojem se nalaze svi ključni elementi web mjesta.

```
<body>

    <div id="glavno">
        <div id="nav" class="nav">
            <div class="t-slika">
                <img src="../grafika/barbell.png" id="brb1">
            </div>
            <div class="peding-p" id="muzz1">
                Dobrodošli na web stranicu Muzz Shopa!
            </div>
```

Slika 13 Početni body atribut uz nekoliko div elemenata

Sljedeća bitna stavka za spomenuti kod podjele elemenata na web mjestu jesu *div* atributi, što su značajke HTML koda koje označavaju „diviziju“ odnosno podjelu HTML elemenata na sekcije. U kontekstu *div* elemenata na slici 12, oni označuju glavni dio, navigaciju, određenu sliku te krajnje *padding* na određenom dijelu stranice. Prema položaju koda u dokumentu, jasno je razaznati određenu hijerarhiju; *div* koji nosi identifikator („id“) „glavno“ je na vrhu hijerarhije, te mu je *div* s identifikatorom „nav“ direktno podređen, dok su njemu direktno podređeni *div* elementi ispod njega. Standardna sintaksa označavanja atributa i elemenata u HTML-u također se vrši na način prikazan na slici 12, odnosno početna oznaka („tag“) elementa/atributa označava se ključnom riječju elementa (tipa *body, div*) pozicioniranom u strelicama, na ovakav način: `<body>`. Završna oznaka HTML elementa se označava na sličan način, ali s dodatnim znakom: `</body>`.

Prva glavna sekcija stranice označena *div* elementom koju sam izradio jest zaglavlje, koje je na stranici označeno crnom bojom te dizajnirano da bude uočljivo. Svrha ovog *diva* zaglavlja je da sadrži glavnu poruku, odnosno moto web shopa te da inicijalno privuče pozornost posjetitelju na glavni cilj i misiju web stranice. *Div* element zaglavlja definiran je u HTML kodu, te u svojoj deklaraciji direktno ima poziv i na klasu u CSS-u, čija je svrha grafičko oblikovanje tog elementa. Koristeći primjer CSS klase za ovaj specifični *div* zaglavlja, opisat ću sam koncept CSS klasa i kako se one koriste za uređivanje HTML elemenata.

```
.zaglavlje{

    position: absolute;
    top: 0px;
    left: 20%;
    width: 100%;
    height: 300px;
    background-color:black;
}
```

Slika 14 CSS klasa zaglavlja

Klase u CSS opisnom jeziku mogu se definirati tako da se odnose na sve elemente koje povezuje isti atribut, primjerice ako se kao naziv klase upiše `body` tada će se ta klasa odnositi na cijeli *body* element u HTML kodu, ili ako je ime klase `button` tada će se ta klasa odnositi na sve gumbe stvorene u HTML dokumentu. S druge strane, klase se također mogu definirati koristeći korisnički definirano ime, kao u primjeru na slici 13. U tim slučajevima kod deklaracije je potrebno prije imena klase staviti točku, jer prema sintaksi jezika, točka prije imena klase je signal jeziku da prepozna klasu kao korisnički definiranu. Takvu klasu je zatim u HTML kodu potrebno pozvati koristeći atribut `class` (pr. sintakse: `<div class="gumb"></div>`, ta klasa bi u CSS-u bila definirana ovako: `.gumb{}`). Što se tiče klase iz primjera slike 13, neki atributi koje sam u toj klasi definirao jesu `position`, koji određuje tip pozicioniranja elementa u klasi; može biti *fixed*, *absolute* ili *relative*. Sljedeći atributi koje sam mijenjao su direktno povezani s pozicijom, atribut `top` određuje položaj elemenata od vrha tog *div* elementa, dok `left` određuje položaj slijeva. Ti atributi mogu se definirati koristeći piksele, točke (*point*, *pt*) ili postotke. Sljedeći atributi, `width` i `height` određuju širinu i visinu *div* elementa, dok `background-color` postavlja boju pozadine. Nakon definiranja svih bitnih atributa klase, u HTML-u sam ju pozvao direktno unutar *div* elementa uz pomoć atributa `class`. Time se spomenuti *div* element direktno povezuje s klasom definiranom pod tim imenom, te se svi atributi deklarirani u klasi primjenjuju na njega.

S obzirom da u samom *div* elementu zaglavlja nema potrebe za bilo kakvim skriptnim funkcijama, za sada je dovoljno definirati ga u HTML kodu te urediti ga u CSS-u koristeći poziv klase. U elementima podređenim zaglavlju, međutim, potrebno će biti definirati određene funkcije. Za početak ih je potrebno definirati u HTML-u i dodati ih ispod *div* elementa zaglavlja, kako bi njemu bili direktno podređeni. Neki od tih elemenata imaju svrhu dopunjavanja grafičkog dojma zaglavlja, dok drugi imaju skriptnu funkciju. Prvi podređeni element dodan zaglavlju jest ilustracija, pozicionirana s desne strane zaglavlja (izvor; fotografirano kod kuće i izrađeno u Photoshopu prije izrade završnog rada za osobne potrebe). Ilustracija je povučena iz zasebne mape u kojoj se nalaze i ostali grafički elementi korišteni na web stranici. Sintaksa pozivanja slike u HTML kod, specifično za ovu ilustraciju, glasi ovako:

```
<img src="../grafika/zyzz1.png" class="prvaslika">
```

Kod definiranja slika vrijedi uočiti da završna oznaka nije potrebna; za prikaz slike u HTML kodu potrebno je samo iskoristiti atribut *img* te zatim navesti izvor slike koristeći atribut *src*. Klasa kako je prikazana u primjeru nije obvezatna.

```
<div class="zaglavlje">

        <img src="../grafika/zyzz1.png" class="prvaslika">
        <img src="../grafika/zyzz2.png" class="drugaslika">
        <div class="trokut"></div>
        <div class="krug">
            <img src="../grafika/soping.png" onclick="cart()" class="cart">
        </div>
```

Slika 15 Prva polovica diva zaglavlja

Preostali *div* elementi podređeni glavnom elementu zaglavlja uočljivi su na slici 14; element označen klasom trokut, koja u CSS-u stvara trokut, te element označen klasom krug, koja stvara krug.

```
.trokut{

    background-color: #FF6800;
    width: 400px;
    height: 400px;
    transform: rotate(330deg);
    position: absolute;
    top: -300px;
    left: -10%;
}
```

Slika 16 CSS klasa trokuta

```css
.krug{
    position: absolute;
    top: -30px;
    right: 320px;
    background-color: #FF6800;
    border-radius: 50%;
    display: inline-block;
    height: 150px;
    width: 150px;
}
```

Slika 17 CSS klasa kruga

U *div* elementu označenim klasom „krug“ možemo primijetiti dodanu sliku atributom *img*. Slika se nalazi direktno u div elementu, kako bi bila ograničena na njegove granice i ne ih prelazila. Put do slike određen je u *src* atributu kao i u prijašnjim primjerima, te gledajući kod iza atributa primjećujemo dvije točke, što HTML sintaksi signalizira da se datoteka koja se traži nalazi u mapi jednu razinu iznad mape u kojoj se nalazi HTML dokument. Nakon kose crte određujemo točno ime mape u kojoj se slika nalazi, te nakon još jedne kose crte određujemo ime i format slike kako bi se ona mogla implementirati u HTML kod. Atribut *onclick* određuje HTML akciju koja će se provesti kada posjetitelj stranice klikne na element koji je određen *onclick* atributom. U kontekstu ove specifične slike, *onclick* atribut pokrenut će JavaScript funkciju po imenu `cart()`. Ovu specifičnu funkciju kao i ostale JavaScript funkcije stvorene za ovaj web shop opisat ću kasnije, kao i načine na koji one funkcioniraju.

```html
<div class="zaglavlje">

        <img src="../grafika/zyzz1.png" class="prvaslika">
        <img src="../grafika/zyzz2.png" class="drugaslika">
        <div class="trokut"></div>
        <div class="krug">
            <img src="../grafika/soping.png" onclick="cart()" class="cart">
        </div>
        <div class="promocija">OŽIVITE SVOJ SPORTSKI DUH</div>
        <div class="promocija2">Isprobajte naše proizvode i uvjerite se u
vrhunsku kvalitetu</div>
        <div class="gumbovi">
            <button type="button" class="gumb"
onclick="prikazip()">Prehrana</button>
            <button type="button" class="gumb"
onclick="prikazio()">Oprema</button>
            <button type="button" class="gumb"
onclick="prikazit()">Treninzi</button>
            <button type="button" class="okrugli" onclick="isprazni()">Isprazni
košaricu</button>
        </div>
```

Slika 18 Kompletni prikaz cijelog diva zaglavlja

Dalje analizirajući zaglavlje, uočit ćemo još nekoliko *div* elemenata, od kojih svaki ima svoju zasebnu klasu i funkciju. Element s klasom „promocija“ je element u kojem je sadržan i zapisan moto/poruka web stranice, element s klasom „promocija2“ ima sekundarnu poruku, dok element s klasom „gumbovi“ sadrži sve funkcionalne gumbove u zaglavlju. Kod dizajna klase za moto stranice sam se odlučio koristiti isti font kao i u logotipu web shopa, a s obzirom da je to font preuzet s interneta pa tako prema zadanim postavkama nije instaliran na bilo kojem računalu, potrebno je posebno ga ukomponirati u CSS datoteku stranice kako bi ispravno funkcionirao. To se vrši na sljedeći način:

```
@font-face {
    font-family: EthnocentricItalic;
    src: url("../grafika/ethnocentricrgit.otf") format("opentype");
}
```

Slika 19 Implementacija fonta u CSS

Ova specifična CSS klasa drugačija je od ostalih, prvotno po identifikatoru klase; koristi se simbol @ za označavanje specifične funkcije koju je potrebno provesti u CSS kodu, te je u ovom slučaju ta funkcija uvoženje fonta treće strane. U samoj klasi se imenom definira `font-family` („obitelj“ fonta, grupa istih stilova fonta), te zatim izvorna datoteka fonta, koja je u ovom slučaju stavljena na isto mjesto kao i ostali grafički elementi web mjesta. Nakon toga ključno je definirati tip fonta, te s obzirom da je ovaj font sastavljen u formatu .otf, potrebno je specificirati font kao format „opentype“.

CSS klase preostalih dviju nespomenutih elemenata u zaglavlju nisu odviše različite od dosadašnjih; klasa „promocija2“ kojom se prikazuje sekundarna poruka stranice jest definirana na način da se taj tekst prikazuje direktno ispod glavne poruke stranice („OŽIVITE SVOJ SPORTSKI DUH“) te da se prikazuje u drugačijem fontu ali s istom bojom. Klasa za gumbove je specifična po tome što definira drukčiji tip elementa; gumb.

```
.gumb{
    font-family: arial black;
    background-color: #FF6800;
    border-style: solid;
    border-color: #DCDCDC;
    border-radius: 5pt;
    border-width: 3pt;
    color: white;
    padding: 16px 80px;
    text-align: center;
    display: inline-block;
    font-size: 16px;
    margin-right: 24px;
}
```


Slika 20 CSS klasa gumb

U ovoj klasi bilo je osim standardnih elemenata potrebno definirati i neke nove, poput stila konture elementa (`border-style`) i njemu direktno povezanih atributa (`border-radius`, `border-width`, `border-color`). Ovi elementi određuju stil konture (solidna linija, isprekidana i sl.), njenu boju, zakrivljenost kutova i širinu. Kod gumbova ove stavke su bitne radi postizanja određene estetike i grafičkog dojma samih gumbova. Atribut `padding` odnosi se na „ispunjenost“ samog elementa, odnosno koliko prostora on zauzima, te se kao i neki ostali atributi može definirati pomoću piksela, postotaka ili točaka.

Klasa „okrugli“ također se odnosi na jedan od gumbova, međutim pošto je taj gumb različit od ostalih po stilu i funkciji, odlučio sam ga dizajnirati na način da se istakne od ostalih pa je zbog toga bilo potrebno izraditi i posebnu klasu.

S time je u suštini završen kod za definiranje zaglavlja. Sljedeći bitni odjeljak stranice jest divizija s lijeve strane, koja će prikazivati sat, odnosno vrijeme i datum pristupanja stranici, zatim logotip, te nekoliko rotirajućih slika i kontaktne informacije. Za ovu sekciju odlučio sam se za kontrastni stil zaglavlju, tako da se oba dijela stranice mogu istaknuti na svoj način. Na ovaj odjeljak stranice u kodu referiram nazivom „strana“, pa sam tako nazvao i glavnu klasu ovog odjeljka.

```
.strana{

    position: absolute;
    background-color: #DCDCDC;
    top: 0px;
    left: 0px;
    width: 20%;
    height: 120%;
}
```

Slika 21 CSS klasa odjeljka s lijeve strane

Jedina ključna razlika u ovoj klasi u usporedbi s klasom zaglavlja jest pozadinska boja, koja je u ovom slučaju bjelkasto siva, za razliku od crne u zaglavlju. Na ovom primjeru možemo uočiti da se boje u HTML i CSS kodu mogu definirati pomoću njihovih uobičajenih naziva na engleskom jeziku (black, blue, white itd.) ili pomoću HEX vrijednosti boja, a također i pomoću RGB vrijednosti. Kada tražimo specifičnu nijansu određene boje, prikladnije je koristiti HEX ili RGB vrijednosti zbog njihove uvećane preciznosti.

```
<div class="strana">

        <button class="sat">
            <div id="datum" class="datum"></div>
        </button>
        <img src="../grafika/logo-muzz-small-1.png" class="logo">
        <div class="stock1" id="fade">
            <img src="../grafika/fade1.jpg" id="fade1" class="fade1">
            <img src="../grafika/fade2.jpg" id="fade2" class="fade2">
        </div>
        <div class="kontakt">
            42000 Varaždin<br>
            Ulica Ivana Kukuljevića Sakcinskog 24<br>
            095 8011 543<br>
            kontakt@muzz-shop.hr
        </div>
    </div>
```

Slika 22 Kompletni "strana" div

Na slici 21 u potpunosti je vidljiv *div* označen klasom „strana“, sa svim ostalim njemu podređenim elementima. Kao što vidimo na slici, prvi podređeni element mu je gumb označen klasom „sat“. U ovom slučaju se takav element može označiti običnim *divom* sa *paddingom*, međutim ja sam se odlučio za gumb u slučaju da se odlučim kasnije dodati neke dodatne funkcionalnosti. Klasa sat određuje položaj toga gumba unutar nadređenog *diva*. Ispod nje je element identificiran imenom „datum“, koja služi specifično za uporabu i poziv u JavaScript funkciji za dobivanje datuma i vremena pristupa stranici, koje ću, kao i ostale JavaScript funkcije, opisati kasnije. Klasa „datum“ opisuje izgled samog teksta unutar gumba. Sljedeći element unutar glavnog *diva* „strana“ jest sam logotip, koji je kao i ostale slike u radu pozvan atributom *img*, te je njegov položaj unutar elementa definiran klasom logo. Sve te klase koje u pravilu definiraju isključivo položaj elemenata u ovom *divu* su u većini slučajeva napisane vrlo sličnim kodom, pa smatram da nema potrebe prikazivati taj kod ovdje zbog redundancije. Međutim, sljedeći element je specifičan; *div* s klasom „stock1“ i identifikatorom „fade“ označuje ukrasne fotografije pozicionirane u ovom elementu. Ovaj element zanimljiv je po tome što se fotografije naizmjenično prikazuju i nestaju, što je postignuto postavljanjem obje fotografije u istu poziciju te zatim naizmjeničnim postavljanjem transparentnosti obje fotografije s 0 na 1 (0% do 100%). Taj efekt kodiran je i postignut u CSS klasi direktno povezanoj s obje fotografije. Kod efekta može se vidjeti ispod:

```
#fade img{
    position: absolute;
    left: 0%;
    transition: opacity 1s ease-in-out;
}

@keyframes fading {
    0% {
    opacity:1;
  }
  45% {
  opacity:1;
  }
  55% {
  opacity:0;
  }
  100% {
  opacity:0;
  }
  }

#fade img.fade2 {
    animation-name: fading;
    animation-timing-function: ease-in-out;
    animation-iteration-count: infinite;
    animation-duration: 5s;
    animation-direction: alternate;
    }
```

Slika 23 Kodirani efekti transparentnosti

Ovim CSS klasama postavljene su pozicije obje slike, kao i efekti, vrijeme i stil tranzicije (`#fade img{}`). Možemo primijetiti da je i naziv ove klase također malo drugačiji; ljestve ispred naziva označavaju da se klasa odaziva na identifikator elementa („id“ atribut), te *img* opisnom jeziku signalizira da modificira sve *img* elemente koji su podređeni tom identifikatoru. Klasa `@keyframes fading {}` određuje ključne vremenske točke u tranziciji te što će se na njima dogoditi, pa se tako na svakoj ključnoj točci mijenja transparentnost (*opacity*) elemenata. U trećoj klasi se klasa *keyframes* primjenjuje na određene elemente u HTML-u, te se određuje broj njezinih iteracija kao i trajanje same animacije.

Posljednja klasa u odjeljku „strana“ jest klasa za kontaktne informacije. Kao što je vidljivo u slici 21, tekst *div* elementa je napisan direktno u HTML-u, kao i neki elementi u zaglavlju stranice. S obzirom da je poželjno da se ovakve informacije podijele na rijetke, svaki redak teksta u HTML-u završava naredbom `<br>` koja označava kraj retka te početak novog, slično kao naredba `/n` ili `endl` u nekim ostalim programskim jezicima.

```
.kontakt{
    position: absolute;
    top: 950px;
    font-family: arial black;
    background-color: #FF6800;
    border-style: solid;
    border-color: black;
    border-radius: 5pt;
    border-width: 3pt;
    color: white;
    padding: 16px 80px;
    text-align: center;
    display: inline-block;
    font-size: 16px;
}
```

Slika 24 Klasa za kontakt

Na primjeru ove klase važno je uočiti pozicioniranje elemenata unutar postojećeg *div* elementa, odnosno s obzirom da kontakt informacije želimo da budu na dnu stranice, potrebno je iskoristi jedno od dvije opcije; `bottom` atribut s niskom vrijednošću, ili `top` atribut s visokom vrijednošću. U ovom slučaju odlučio sam se za `top` atribut. Stil fonta je isti kao i na ostatku ostalih bitnih informacija na stranici, te su konture bloka konfigurirane na sličan način kao i gumbovi u zaglavlju. `color` atribut određuje boju teksta u elementu, u ovom slučaju bijela, s obzirom da imamo narančastu pozadinu. Također je bitno napomenuti `text-align` atribut, koji postavlja sav tekst u sredinu *div* elementa.

Time je grafički završen i lijevi odjeljak stranice. Sada se fokus rada orijentira na glavni dio stranice, u kojem se nalaze sekundarni navigacijski elementi i glavni prikaz kupovnih proizvoda. Naziv ovog odjeljka je „glavno“, pa se tako naziva i klasa, isto kao i kod lijevog odjeljka opisanog u prijašnjem dijelu rada. Početni dio ovog odjeljka bio je prikazan na početku ovog poglavlja, točnije na slici 12, pa ću ovdje ponovo napomenuti sve elemente prikazane na toj slici: glavni div (`id="glavno"`), njemu podređen element „nav“ (`id="nav" class="nav"`), te ispod toga slika i tekst u oblačićima ispod navigacije.

```
<div id="nav" class="nav">
            <div class="t-slika">
                <img src="../grafika/barbell.png" id="brb1">
            </div>
            <div class="peding-p" id="muzz1">
                Dobrodošli na web stranicu Muzz Shopa!
            </div>
            <div class="peding-p" id="muzz2">
                Kliknite na neki od gornjih gumbova kako biste pregledali
dostupne proizvode.
                Kupujte sigurno i ugodno uz Muzz Shop!
            </div>
            <div style="position: relative; top: 38%; display: none;" id="peding-
p">
                <div class="peding-p">
                    PREHRANA
                </div>
```

Slika 25 Div element identifikatora nav

Na slici 24 prikazan je detaljniji i proširen prikaz početka „nav“ elementa, gdje možemo uočiti specifične klase za tekst u HTML-u. Na oba oblačića teksta primijenjena je ista klasa, tako da oba izgledaju isto i posjeduju isti grafički stil. Međutim, svaki od ta dva *diva* ima posebni identifikator, primarno zbog mogućnosti primjene različitih JavaScript funkcija na svaki oblačić individualno. Sljedeći element nalazi se na samom dnu slike, s identifikatorom „peding-p“, koji kao i ostali identifikatori *div* elemenata dosada, služi radi reference u funkcijama JavaScript datoteke. U opisu tog elementa također možemo uočiti jednu specifičnost; umjesto primjene zasebne CSS klase, primijenio sam tzv. *inline* CSS stil, koji se obično koristi za primjenu jako kratkih i specifičnih linija stila za jedan element. U standardnoj uporabi obično se preporuča koristiti klase u svim slučajevima, radi njihove bolje čitljivosti i funkcionalnosti, ali u ovakvim slučajevima kada je potrebna primjena samo nekoliko linija CSS koda na jedan specifični element, uporaba *inline* stila također se može tolerirati. Točne linije CSS stila koje sam dodao tom elementu jesu pozicija, koju sam odredio da bude relativna, s obzirom da mi je potrebno pomicanje elemenata u tom *divu* ovisno o korisničkom unosu, zatim sam odredio točnu poziciju elementa da bude 38% od vrha, te da na početku otvaranja stranice, bez ikakvih korisničkih unosa, bude nevidljiva uz pomoć `display: none;` koda.

```
<div style="position: relative; top: 38%; display: none;" id="peding-p">
                <div class="peding-p">
                    PREHRANA
                </div>
                <div class="klik-gumbovi">
                    <button type="button" onclick="protein()">Protein</button>
                    <button type="button" onclick="kreatin()">Kreatin</button>
                    <button type="button"
onclick="preworkout()">Preworkout</button>
                    <button type="button" onclick="dodaci()">Dodaci</button>
                    <button type="button" onclick="ostalo()">Ostalo</button>
                </div>
            </div>
            <div style="position: relative; top: 38%; display: none;" id="peding-
o">
                <div class="peding-p">
                    OPREMA
                </div>
                <div class="klik-gumbovi">
                    <button type="button" onclick="utezi()">Utezi</button>
                    <button type="button" onclick="sipke()">Šipke</button>
                    <button type="button" onclick="stalci()">Stalci za
utege</button>
                    <button type="button" onclick="dodacx()">Dodaci</button>
                    <button type="button" onclick="oprema()">Ostala
oprema</button>
                </div>
            </div>
            <div style="position: relative; top: 38%; display: none;" id="peding-
t">
                <div class="peding-p">
                    TRENINZI
                </div>
                <div class="klik-gumbovi">
                    <button type="button" onclick="teretana()">Teretana</button>
                    <button type="button" onclick="kuca()">Kod kuće</button>
                    <button type="button"
onclick="kalistenika()">Kalistenika</button>
                    <button type="button" onclick="muski()">Muški</button>
                    <button type="button" onclick="zenski()">Ženski</button>
                </div>
            </div>
            <div class="klik-gumbovi">
                <button type="button" onclick="trik()">Napravi trik!</button>
            </div>
            <div class="p-slika">
                <img src="../grafika/barbell.png" id="brb2"
            </div>
        </div>
```

Slika 26 Ostatak glavnog div elementa

Na slici 25 prikazan je ostatak glavnog odjeljka web stranice, te najupečatljivija značajka ovog dijela koda jesu žuto obojani pozivi na JavaScript funkcije. Svaki *div* odjeljak s klasom „peding-p“ označava jedan od glavnih odjeljaka u navigaciji koji se otvara klikom na gumbove u zaglavlju, čiji je kod prikazan na slici 17. Klikom na bilo koji od tih gumbova sa stranice se skrivaju inicijalni tekstualni oblačići pod imenom „muzz1“ i „muzz2“, te se na njihovom mjestu pojavljuju oblačići s odgovarajućom otvorenom sekcijom, ovisno o gumbu koji je posjetitelj pritisnuo. Primjerice, ako je kliknuo na gumb Prehrana, na mjestu sada skrivenih oblačića otvorit će se oblačić s tekstom „PREHRANA“ i ispod njega funkcionalni gumbovi navigacije koje korisnik može dalje pritiskati kako bi otvorio proizvode podijeljene u te određene kategorije. Primjećujemo da je klasa ovih oblačića jednaka inicijalnim oblačićima koji su bili otvoreni na stranici od početka, radi održavanja istog tipa stila. Također, svi funkcionalni gumbovi posloženi su u isti *div* element, te se na klik pojavljuju zajedno s oblačićem u čiju su kategoriju smješteni. S obzirom da se svi gumbovi nalaze pod istim elementom s jednom primijenjenom klasom, nije potrebno na svaki gumb posebno primjenjivati zasebnu klasu.

Posljednji odjeljak u HTML kodu koji treba spomenuti prije prelaska na objašnjenje JavaScript funkcija jest dio koda u kojem se dio tih funkcija još jednom poziva, a to je dio stranice u kojem se prikazuju slike proizvoda, kao i njihovi funkcionalni gumbovi za kupovinu.

```
<img src="../grafika/suplementi.jpg" id="suplementi">

            <div id="protein1" style="display: none;">
                <img src="../proizvodi/protein1.png" id="p1m" class="ppr">
                <div class="prvi-d" id="p1y">
                    Proteini Optimum Nutrition GOLD STANDARD WHEY
                    <button class="novigumb" onclick="kupip1()"
id="g1">Kupi</button>
                </div>
            </div>
            <div id="protein2" style="display: none;">
                <img src="../proizvodi/protein2.png" id="p2m" class="ppr">
                <div class="drugi-d" id="p2y">
                    Proteini Optimum Nutrition WHEY
                    <button class="novigumb" onclick="kupip2()"
id="g1">Kupi</button>
                </div>
            </div>
            <div id="kreatin" style="display: none;">
                <img src="../proizvodi/kreatin.png" id="km" class="ppr">
                <div class="jedan" id="ky">
                    Kreatin Optimum Nutrition
                    <button class="novigumb" onclick="kupik()"
id="g1">Kupi</button>
                </div>
            </div>
```

Slika 27 Odjeljak s funkcionalnim gumbovima kupovine

Na slici 26 vidimo primjer glavnog *diva* vezanog uz ovaj odjeljak stranice, kao i njemu podređene elemente u kojima se nalaze spomenuti gumbovi vezani uz JavaScript funkcije kupovine. Glavni element ovog odjeljka nosi identifikator „promo“ te istoimenu klasu, koja ga pozicionira u donji desni dio stranice, ispod zaglavlja i desno od lijevog odjeljka i navigacije. Kod otvaranja stranice je po zadanim postavkama na tom dijelu stranice vidljiva samo jedna slika, označena identifikatorom „suplementi“. Klikom na bilo koji od dostupnih gumbova pod navigacijskom podjelom, ta slika se sakriva te se pojavljuju odgovarajući proizvodi, ovisno o gumbu koji je korisnik pritisnuo. Svaki od tih gumba ima svoj posebni *div* element, s tri različite klase: ukoliko se u kategoriji nalaze dva proizvoda, proizvod ima klasu „prvi-d“ ako je prvi od ta dva proizvoda, te „drugi-d“ ako je drugi, a ako je to jedini proizvod u kategoriji, onda posjeduje klasu „jedan“. S druge strane, svaki zasebni proizvod ima svoj specifični identifikator, kako bi se osiguralo ispravno provođenje JavaScript funkcije nad tim *div* elementom HTML koda. Ispod toga možemo primijetiti tekst napisan u običnom HTML-u, koji se pojavljuje iznad gumba za kupovinu, od kojih je svaki definiran klasom „novigumb“, s obzirom da se mora razlikovati od prijašnjih vrsta gumbova. Svaki od gumbova vrši zasebnu funkciju kada se na njega klikne, kako bi se adekvatni elementi HTML koda prikazali ili sakrili. Budući da svaka glavna kategorija (prehrana, trening, oprema) ima 5 kategorija podređenih sebi, te svaka ta „pod-kategorija“ sadrži 1 ili 2 proizvoda, bilo je potrebno napraviti mnoštvo sličnih JavaScript funkcija kako bi se na adekvatan način prikazali i sakrili odgovarajući elementi. U sljedećem poglavlju rada ću te JavaScript funkcije opisati.

4.3. Ostvarivanje funkcionalnosti pomoću JavaScripta

HTML dio web stranice ima svrhu definirati sve elemente prisutne na stranici, kao što su odjeljci stranice definirani pomoću atributa poput *div, body* itd., dok je svrha CSS-a opisivanje tih istih elemenata i dodavanje raznih grafičkih funkcionalnost i elemenata. S druge strane, JavaScript kao skriptni programski jezik proširuje funkcionalnosti web stranica na način da dizajneru omogućava brojna programska rješenja unaprjeđenja web stranice. U kontekstu mog rada, iskoristio sam JavaScript primarno za dodavanje funkcija prikazivanja i skrivanja raznih elemenata, te za dodavanje funkcionalnog ispisa vremena pristupa stranici, kao i funkcionalne *front-end* demonstrativne košarice za kupovinu.

U kontekstu samog pisanja koda u JavaScriptu, koristio sam primarno funkcije te ih zatim pozivao u HTML kodu. U nekoliko slučajeva sam dodao i globalne varijable za uporabu u tim funkcijama, ali veliki udio svog JavaScript koda nalazi se unutar funkcija. Prva funkcija koju sam implementirao u JavaScript datoteku je funkcija za dinamičko mijenjanje HTML koda u identifikatoru „datum“, koji se nalazi u odjeljku „strana“ (glavni odjeljak sivkaste boje s lijeve strane sučelja). Kod svakog otvaranja ili osvježavanja stranice taj kod se osvježi te ponovno prikazuje vrijeme i datum pristupa stranici prema lokalnoj vremenskoj zoni. Kod za tu funkciju nalazi se na slici ispod:

```
/* Funkcija za prikaz datuma i vremena pristupa stranici */
var dat = new Date();
document.getElementById('datum').innerHTML=dat;
```

Slika 28 JavaScript funkcija datuma

Sama funkcija je relativno jednostavna. Zeleni tekst iznad funkcije je komentar, odnosno tekst koji jezik ne gleda kao kod zbog posebnih oznaka između kojih ga je programer smjestio, što su u slučaju JavaScripta kosa crta i zvjezdica (`/* Ovo je primjer komentara u JavaScriptu. */`). Komentari se pišu radi lakšeg referenciranja podataka za programera te bolju čitljivost i snalažljivost u kodu. U sljedećoj liniji koda stvaram varijablu pod imenom „dat“ i postavljam njezinu vrijednost tako da stvara novi datum naredbom `Date()`. Nakon toga zatvaram tu liniju koda i u novoj dozivam identifikator elementa „datum“ iz HTML koda (*document*), te u drugom dijelu linije modificiram postojeći HTML (*.innerHTML*) te postavljam da njegova vrijednost bude jednaka varijabli *dat*, čija je vrijednost jednaka datumu stvorenom pri otvaranju ili osvježavanju stranice.

Sljedeća funkcija u mom JavaScript kodu odnosi se prvotno na mijenjanje HTML elemenata u navigacijskom dijelu stranice.

```
/* Funkcije (sljedeće tri) za mijenjanje div-a klikom na gumb */

function prikazip(){
    var x = document.getElementById("peding-p");
    var y = document.getElementById("peding-o");
    var z = document.getElementById("peding-t");
    var m = document.getElementById("muzz1");
    var n = document.getElementById("muzz2");
    if (m.style.display !== "none" && n.style.display !== "none"){
        m.style.display = "none";
        n.style.display = "none";
    }
    if (y.style.display === "none" && z.style.display === "none"){
        if (x.style.display === "none") {
            x.style.display = "block";
          } else {
            x.style.display = "none";
          }
    } else {
        (y.style.display === "none" && z.style.display === "none");
    }
}
```

Slika 29 JavaScript funkcija za navigacijski dio

U ovoj funkciji stvaram 5 varijabli, svaka od kojih referencira jedan od glavnih elemenata u navigacijskom dijelu web stranice. Nakon toga koristim uvjetni izraz, odnosno *if* funkciju; ukoliko je jedan od uvjeta točan/netočan, uz još nekoliko mogućih kombinacija do koje se dolazi operatorima i/ili/ne, postavljam uvjete za prikazivanje i sakrivanje elemenata navigacije. U ovoj specifičnoj funkciji, početni uvjet glasi da ako su oba oblačića navigacije vidljiva (dupla negacija u izrazu), da ih je potrebno postaviti da budu nevidljivi. Sljedeća selekcija ispituje jesu li sekundarni izbornici koji bi se pojavili pritiskom na gumb vidljivi, te ako oba nisu vrši se sljedeća selekcija koja gleda je li izbornik prehrane nevidljiv, te ako je onda postaje vidljiv, a ako nije onda ostaje nevidljiv, te se sada vraćamo na početnu selekciju čiji *else* glasi da izbornici opreme i treninga budu nevidljivi. *Else* će se provesti ako inicijalni uvjet nije točan, odnosno ako su izbornici opreme i treninga vidljivi, a ne nevidljivi. Sljedeće dvije funkcije u kodu funkcioniraju na isti princip kao i ova, samo sa zamijenjenim varijablama tako da njihova funkcija može raditi za pritisak na gumbove za trening i opremu, za razliku od ove koja proradi kada se klikne na gumb za prehranu, pa stoga smatram da ih ne vrijedi posebno prikazivati. Poslije tih funkcija u kodu sam naveo globalne varijable koje sam iskoristio kasnije u funkcijama za prikaz i sakrivanje gumbova kupovine proizvoda, što izgleda ovako:

```
/* Funkcije i varijable vezane uz kupovinu */
var kq = 0;
var p1q = 0;
var p2q = 0;
var pwq = 0;
var dod1q = 0;
var dod2q = 0;
var ostq = 0;
var uq = 0;
var ssq = 0;
var stq = 0;
var dodx1q = 0;
var dodx2q = 0;
var opr1q = 0;
var opr2q = 0;
var t1q = 0;
var t2q = 0;
var kuq = 0;
var ksq = 0;
var mskq = 0;
var znskq = 0;
```

Slika 30 Globalne varijable s postavljenim vrijednostima

```
var ky = document.getElementById("ky");
var p1y = document.getElementById("p1y");
var p2y = document.getElementById("p2y");
var pwy = document.getElementById("pwy");
var dod1y = document.getElementById("dod1y");
var dod2y = document.getElementById("dod2y");
var osty = document.getElementById("osty");
var uy = document.getElementById("uy");
var ssy = document.getElementById("ssy");
var sty = document.getElementById("sty");
var dodx1y = document.getElementById("dodx1y");
var dodx2y = document.getElementById("dodx2y");
var opr1y = document.getElementById("opr1y");
var opr2y = document.getElementById("opr2y");
var t1y = document.getElementById("t1y");
var t2y = document.getElementById("t2y");
var kuy = document.getElementById("kuy");
var ksy = document.getElementById("ksy");
var msky = document.getElementById("msky");
var znsky = document.getElementById("znsky");
```

Slika 31 Globalne varijable koje dohvaćaju gumbove u HTML-u

Slika 29 prikazuje globalnu varijablu za svaki od proizvoda, npr. „kq“ predstavlja kreatin, „p1q“ proteine #1, „p2q“ proteine #2, „pwq“ pre-workout i tako dalje. Sufiks q sam iskoristio kako ne bi morao izmišljati potpuno nova imena za iste varijable koje služe drugačiju funkciju ali na istom HTML elementu, s obzirom da sam u drugom dijelu koda već iskoristio ista imena varijabli bez sufiksa. Isti princip vrijedi i za varijable na slici 30, gdje se također varijable odnose na iste elemente u HTML-u, ali služe drugačiju funkciju, odnosno sakrivanje s drugim gumbom. Globalne varijable u osnovi se koriste kada istu varijablu moramo pozvati u nekoliko funkcija, tako da se onda ta ista varijabla ne deklarira u svakoj funkciji zasebno. Kod deklaracija varijabli sa sufiksom q cilj je bio postaviti inicijalnu vrijednost kupovine, odnosno kod otvaranja stranice da svi proizvodi budu „nekupljeni“, a kasnije pritiskom na gumb koji je povezan s određenom funkcijom da se njihova vrijednost promijeni na 1 i da proizvod postane „kupljen“ i da se njegova ilustracija pojavi kada se otvori košarica kupovine. S druge strane, varijable na slici 30 deklarirao sam isključivo sa svrhom dozivanja HTML elemenata, što je odmah i vidljivo u sintaksi (`document.getElementById`). U sljedećem dijelu koda JavaScripta upisao sam svaku funkciju zasebno za svaki gumb s kojim se ona podudara, tako da svaki proizvod ima svoju funkciju kupovine i dodavanja proizvoda u košaricu (slika ispod).

```
function kupipw(){
    pwq = 1;
}

function kupidod1(){
    dod1q = 1;
}

function kupidod2(){
    dod2q = 1;
}
```

Slika 32 Funkcije za kupovinu

Kao što je bilo opisano na prijašnjoj stranici, ove funkcije mijenjaju inicijalne vrijednosti globalnih varijabli sa sufiksom q kako bi se na klik gumba koji te funkcije poziva, odgovarajući proizvodi vezani uz funkciju pojavili u košarici za kupovinu. Svaki proizvod ima vlastitu funkciju tako da ove funkcije koje su po prirodi iste, samo s drukčijim varijablama, zauzimaju nekoliko desetaka linija koda sve dok ne dođemo do sljedećeg tipa funkcije; punjenje i pražnjenje košarice. S obzirom da je ova funkcija duga, prikazat ću slike na sljedećoj stranici, a na ovoj ju opisati. Funkcija započinje pozivanjem varijabli svih proizvoda (bez sufiksa), te zatim na svaku dodaje `document.getElementById` koji uzima elemente iz HTML-a kako bi ih

modificirao. Svaka varijabla proizvoda poziva svoj odgovarajući proizvod u HTML kodu (npr. varijabla za trening plan za žene poziva HTML element tog istog trening plana za žene). Zatim sam za svaku varijablu u toj istoj funkciji postavio da se dinamički mijenja CSS kod njenog HTML ekvivalenta (primjer sintakse: `s.style.display = „none“;`) , pa tako prvi dio naredbe, prije točke, što je u ovom slučaju s, označuje varijablu koja poziva svoj HTML kod, zatim nakon točke se piše atribut *style* što signalizira da je riječ o CSS dijelu tog elementa, te nakon još jedne točke određujemo točno koji CSS atribut se mijenja, u ovom slučaju *display*, te se postavlja na vrijednost *none*, kako bi ga se u potpunosti sakrilo. Isti proces se ponavlja za svaku varijablu proizvoda, da svi budu sakriveni, te to postavlja inicijalnu vrijednost košarice kada još nijedan proizvod nije kupljen. Sljedeći korak je postavljanje selekcije za svaku varijablu; ukoliko posjetitelj klikne na gumb koji mijenja vrijednost globalne varijable proizvoda iz 0 u 1, tada se smatra da je taj proizvod kupljen te ga se dodaje u košaricu, odnosno pritiskom na gumb košarice će se pojaviti jer je globalna varijabla jednaka 1.

```
function cart(){
    var k = document.getElementById("kreatin");
    var p1 = document.getElementById("protein1");
    var p2 = document.getElementById("protein2");
    var pw = document.getElementById("preworkout");
    k.style.display = "none";
    p1.style.display = "none";
    p2.style.display = "none";
    pw.style.display = "none";
    if (kq === 1){
        k.style.display = "block";
        ky.style.display = "none";
    } if (p1q === 1){
        p1.style.display = "block";
        p1y.style.display = "none";
    } if (p2q === 1){
        p2.style.display = "block";
        p2y.style.display = "none";
    } if (pwq === 1){
        pw.style.display = "block";
        pwy.style.display = "none";
```

Slika 33 Prikaz funkcije za dodavanje elemenata u košaricu

Selekcija (*if* naredba) prikazana na kraju slike 32 ponavlja se za svaku varijablu posebno. Ostatak funkcija strukturiran je na sličan način, pa ću u slike stavljati samo bitniji dio funkcija koji i dalje prenosi cijelu poantu. Sljedeća funkcija u JavaScript kodu odnosi se na pražnjenje košarice za kupovinu, te funkcionira na sljedeći način: vrijednosti svih deklariranih globalnih varijabli sa sufiksom q se postavljaju na 0, kako bi njihov status kupovine bio resetiran te da se ne računaju više kao da su kupljene. Nakon toga se njihov standardni CSS prikaz (*display*), koji je zapisan u varijablama sa sufiksom y, ponovno postavlja na „block“ kako bi bio vidljiv kada se ti proizvodi otvore u izbornicima u navigaciji.

```
    opr1q = 0;
    opr2q = 0;
    t1q = 0;
    t2q = 0;
    kuq = 0;
    ksq = 0;
    mskq = 0;
    znskq = 0;
    ky.style.display = "block";
    p1y.style.display = "block";
    p2y.style.display = "block";
    pwy.style.display = "block";
```

Slika 34 Dio funkcije isprazni()

Posljednji tip funkcije u mom JavaScript kodu, koji se također kao i neke ostale već navedene funkcije ponavlja zasebno za svaki proizvod, jest tip funkcije kojim se sam proizvod prikazuje na ekranu kada se klikne na njegov gumb u navigacijskom oknu stranice. Funkcija obavlja rad na sljedeći način: učitava sve HTML elemente proizvoda u varijable (`document.getElementById`), te zatim sve elemente preko varijabli sakriva (pr. `t1.style.display = „none“;`), osim elementa kojim ta funkcija barata, primjerice ako je funkcija za prikaz kreatina onda će sakriti sve proizvode osim kreatina. Nakon sakrivanja svih ostalih elemenata, funkcija uvjetnom selekcijom provjerava je li *default* slika suplemenata (slika koja se pojavljuje prije svega kod otvaranja stranice) prikazana ili skrivena. Ukoliko je prikazana, tada se u oknu selekcije ona sakriva sa `display: none` a proizvod koji se želi prikazati se prikazuje koristeći `display: block`. U protivnom (*else*) se *default* slika postavlja da bude vidljiva a proizvod za koji funkcija vrijedi da se sakriva. Kao što je već bilo navedeno, ovaj tip funkcije ponavlja se za svaki proizvod, sa promijenjenim varijablama kako bi se dobili odgovarajući rezultati i prikazali ili sakrili ispravni proizvodi.

```
function protein(){
    var s = document.getElementById("suplementi");
    var k = document.getElementById("kreatin");
    var p1 = document.getElementById("protein1");
    var p2 = document.getElementById("protein2");
    var pw = document.getElementById("preworkout");
    var dod1 = document.getElementById("dodaci");
    var dod2 = document.getElementById("dodaci2");
    var ost = document.getElementById("ostalo");
    var u = document.getElementById("utezi");
    var ss = document.getElementById("sipke");
    var st = document.getElementById("stalci");
    var dodx1 = document.getElementById("dodx1");
    var dodx2 = document.getElementById("dodx2");
    var opr1 = document.getElementById("opr1");
    var opr2 = document.getElementById("opr2");
    var t1 = document.getElementById("teretana1");
    var t2 = document.getElementById("teretana2");
    var ku = document.getElementById("kuca");
    var ks = document.getElementById("kalistenika");
    var msk = document.getElementById("muski");
    var znsk = document.getElementById("zenski");
    t1.style.display = "none";
    t2.style.display = "none";
    ku.style.display = "none";
    ks.style.display = "none";
    msk.style.display = "none";
    znsk.style.display = "none";
    u.style.display = "none";
    ss.style.display = "none";
    st.style.display = "none";
    dodx1.style.display = "none";
    dodx2.style.display = "none";
    opr1.style.display = "none";
    opr2.style.display = "none";
    k.style.display = "none";
    pw.style.display = "none";
    dod1.style.display = "none";
    dod2.style.display = "none";
    ost.style.display = "none";
    if (s.style.display !== "none"){
        s.style.display = "none";
        p1.style.display = "block";
        p2.style.display = "block";
    } else {
        s.style.display = "block";
        p1.style.display = "none";
        p2.style.display = "none";
    }
}
```

Slika 35 Kompletna funkcija za prikaz proteina

4.4. Implementacija animiranih elemenata – CSS3

Ostvarivanjem svih prethodnih ciljeva obogaćivanja *web* mjesta omogućavamo optimalnu funkcionalnost svih dostupnih elemenata, kao što su responzivni gumbovi koji prikazuju i sakrivaju traženi sadržaj, podjela elemenata na kategorije od kojih svaka ima vlastitu funkciju prikaza, te općenit *front-end* dio sklopljen pomoću osnovnih funkcija HTML i CSS opisnih jezika. Sljedeća bitna komponenta u izradi web stranice jest realizacija implementacije animacija i efekata, čiji je cilj daljnje poboljšanje korisničkog iskustva na web mjestu, kao i doživljaja te dojma koji određeni elementi stranice ostavljaju.

Postoji nekoliko osnovnih načina i tehnologija koje se mogu iskoristiti u ove svrhe, neke od kojih su tzv. *keyframes* funkcionalnosti opisnog jezika CSS, koje su svoju prvu pojavu uočile u prvom kvadrantu 2009. godine, kada su mnogi preglednici krenuli s implementacijama CSS animacija kroz vlastite različite načine. U kontekstu mog rada, CSS *keyframes* animacije prvotno sam iskoristio za izmjenjivanje polaznih fotografija s lijeve strane, čiji je proces implementacije i korišteni kod opisan ranije u radu. Kao što je već bilo opisano, u različitim trenucima animacije mijenja se vrijednost transparentnosti (*opacity*) fotografija tako da se *fade-in* i *fade-out* animacijama fotografije naizmjenično zamjenjuju. Sam koncept *keyframesa* odnosi se na točke u vremenskom periodu, koje su definirane različitim promjenama atributa nekih elemenata. Isti koncept pojavljuje se i u ostalim medijima, primjerice u uređivanju videozapisa; kod primjene raznih efekta i animacija na videozapis, definiraju se različite točke u vremenskoj crti, na svakoj od kojih se vrši različita promjena nad sadržajem, tako da se u konačnici dobiva određen efekt tranzicije ili promjene. U CSS-u su te točke definirane postocima, odnosno u kodu *keyframes* animacije postotkom se definira vremenska točka, te ćemo stoga promjenu koju želimo izvršiti na polovini ukupnog trajanja animacije označiti s 50%. Za uočljivu promjenu primjenom animacije, potrebno je definirati barem dvije *keyframes* točke. Za jednostavnu i konstantnu promjenu se mogu iskoristiti dvije točke, npr. 0% i 100%, koje će omogućiti linearni prikaz animacije jednostrukim potezom. Kod definiranja *keyframes* animacija u CSS-u također je potrebno odrediti trajanje animacije u sekundama, što se vrši pozivom na animaciju u klasi koja tu animaciju koristi (pr. `animation: ime-animacije 1s;`). Neke od ostalih naredbi direktno vezanih uz CSS3 animacije jesu *animation-timing-function*, *animation-duration*, *animation-direction* i sl., svaki od kojih ima i vlastite pozivne argumente koji se trebaju definirati (npr. *animation-direction* određuje smjer animacije, koji mogu biti *normal*, *reverse*, *alternate* i sl.).

U kontekstu mojeg rada, CSS3 *keyframes* animacije glavnu primjenu pronalaze u prikazivanju i nestajanju elemenata, kao i pomaku određenih elemenata ili promjeni njihovih atributa i značajki. Prva implementacija *keyframes* animacije, kao što je bilo spomenuto, jest promjena glavnih fotografija s lijeve strane web mjesta, kroz trajanje od 5 sekundi i 4 točke mijenjanja definiranje u *keyframes* klasi koja se primjenjuje na te elemente. Taj kod vidljiv je na slici 22. To je bila jedina animacija definirana prije završetka ostatka stranice; svi ostali animirani elementi delegirani su za dovršavanje nakon završetka kostura i funkcionalnosti stranice.

Sljedeće „poglavlje“ u definiranju animacija na web mjestu pomoću CSS3 opisnog jezika odnosit će se na animiranje gumbova. Kod primjene animacija na gumbove primarni cilj mi je bio blaga promjena određenih atributa gumba kod prijelaza mišem (*hover*). Definiranje animacije na *hover* akciju također se vrši preko CSS klase, specifično dodavanjem sufiksa *:hover* na ime klase (pr. `.nav:hover`). Time CSS jeziku signaliziramo da prelaskom miša preko elementa koji nasljeđuje zadanu klasu želimo da se izvrši određen CSS kod, definiran pod zasebnom „podklasom“ *hover*. Za početak sam definirao *hover* klase za „glavne“ gumbove, odnosno gumbove na čiji se pritisak prikazuju glavne kategorije na web stranici (prehrana, oprema, trening). Na te gumbove primijenio sam jednostavniju, lakšu tranziciju; s obzirom da se pojavljuju u zaglavlju koje je crne boje, na *hover* tih gumbova dodao sam promjenu boje okvira gumba na crnu uz tranziciju od 0.7 sekundi, vidljivo u kodu ispod.

```
.gumb:hover, .j-gumb:hover, .gumb1:hover,.gumb2:hover, .pomak-gumb:hover{
    transition: 0.7s;
    border-color: black;
}
```

Slika 36 Kod tranzicije gumbova

Kao što je vidljivo u izvatku koda, jednaki CSS kod primjenjuje se na više različitih klasa, svaka od kojih odnosi se na zasebni tip gumba na web mjestu. Kod se vrši na svaki prijelaz mišem preko tog gumba (*hover*). Pritiskom na gumb vrši se JavaScript funkcija opisana u prijašnjem dijelu rada. Ovaj CSS kod odnosi se na sve gumbove koji se nalaze u zaglavlju, tako da se održi konzistentnost smjera dizajna u tom dijelu stranice.

Nakon što smo animirali „glavne“ funkcionalne gumbove stranice, potrebno je skrenuti fokus na ostatak ključnih elemenata u zaglavlju. S obzirom da je naslov, odnosno glavni moto stiliziran fontom treće strane, te radi zadržavanja već postignutog stila u zaglavlju, smatram da nije potrebno posebno animirati tekstualne elemente zaglavlja. S tim na umu, pažnju ćemo okrenuti na gumb prikaza košarice za kupovinu te ilustracije ispod nje. Na gumb košarice primijenit će se

animacija smanjivanja, ponovno koristeći *hover* atribut te ovaj put i zasebne animacije iz posebne klase, umjesto tranzicije unutar klase kao u prijašnjem primjeru. U inicijalnoj klasi određuje se točno koja animacija će se primijeniti na ovu klasu, te njezino trajanje i koliko puta će se ona ponoviti.

```
.cart{

    animation: povecaj3 1s;
    animation-iteration-count: 1;
}

.cart:hover{
    animation: povecaj2 1s forwards;
    animation-iteration-count: 1;
}
```

Slika 37 Klase animiranja košarice

U svrhu pravilnog animiranja elemenata, stvorene su dvije klase: klasa `.cart` i klasa `.cart:hover`. Prva klasa odnosi se na „bazalno“ odnosno inicijalno stanje elementa, odnosno što se s njim događa kada se na njemu ne nalazi pokazatelj miša. Ta klasa odnosno animacija također će se izvesti kod svakog osvježavanja ili ponovnog pokretanja web mjesta. Druga klasa odnosi se na akcije koje će se provesti kada se preko elementa označenog tom klasom prijeđe potezom miša, odnosno kad se na njega *hovera*. U tom slučaju provest će se druga varijanta animacije. U kodu ispod vidljive su obje animacije te kako funkcioniraju.

```
@keyframes povecaj2{
    0%{
        transform: scale(1, 1);
    }
    100%{
        transform: scale(0.8, 0.8);
    }
}

@keyframes povecaj3{
    0%{
        transform: scale(0.8, 0.8);
    }
    100%{
        transform: scale(1, 1);
    }
}
```

Slika 38 Animacije korištene kod prikaza košarice

Na slici 37 vidljiv je kod korišten kod prikaza košarice za kupovinu, te možemo uočiti inverznu relaciju između obje klase; obje funkcioniraju na isti način, ali sa zamijenjenim vrijednostima na istim *keyframe* točkama. Inicijalna funkcija, *povecaj3*, mijenja vrijednost veličine elementa s 0.8 natrag na 1, te se ta klasa vrši nakon što se pokazatelj miša makne s elementa koji nasljeđuje ove klase. Klasa koja se izvodi na *hover* miša preko elementa jest *povecaj2*, koja mijenja inicijalnu i originalnu veličinu elementa (`scale(1, 1)`) na vrijednost 0.8, kako bi se taj element smanjio. Kada se pokazatelj miša pomakne s elementa, izvede se klasa *povecaj3* te se njena veličina vrati na originalne vrijednosti.

Sljedeći element zaglavlja nad kojim se vrše animacije jest ilustracija poze, koja se nalazi direktno ispod gumba za košaricu te desno od glavnog teksta i gumbova. Način na koji je ova slika implementirana u dokument je opisan na samom početku, te će u ovom dijelu biti opisano kako ćemo na nju primijeniti određene animacije. S obzirom da je u dokument dodana kao obična slika, mogućnost implementacija animacija na nju su ograničene, te je stoga optimalan postupak za sljedeći korak stvaranje varijacije iste slike, odnosno stvaranje kopije slike te mijenjanje određenih atributa iste. Odlučio sam modificirati kopiju na način da izmijenim smjer gradijenta u Photoshopu, tako da na određen način bude kontrastna inicijalnoj verziji. Nakon toga implementirao sam obje verzije slike u kod, svaka sa svojom posebnom klasom i identifikatorom. Sa dvije različite inačice iste slike proširujemo mogućnosti primjene animacija i efekata, na sličan način kao i polazne fotografije s lijeve strane web stranice. Prvi korak kod animiranja ovih ilustracija jest izrada klasa u CSS datoteci stranice te direktno povezivanje tih klasa sa HTML elementom ilustracija. S obzirom da se obje slike nalaze na istoj poziciji i dijele iste atribute, osim animacije, obje klase bit će gotovo jednake, s jedinim razlikama u animaciji koje se primjenjuju na njih.

```
.prvaslika{
    position: absolute;
    height: 300px;
    right: 20%;
    animation: zyzz1 2s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
}

.drugaslika{
    position: absolute;
    height: 300px;
    right: 20%;
    animation: zyzz2 2s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
}
```

Slika 39 Klase animiranja ilustracije

Jedna klasa primjenjuje se na jednu od inačica ilustracije, dok se druga primjenjuje na drugu, te kao što se može uočiti jedina razlika jest primjena različite animacije. Osim već spomenutih atributa pozicije, bitno je napomenuti i linije koda koje se odnose direktno na tip animacije i načine na koji se ona provodi, koji su u ovom slučaju broj iteracije, odnosno broj ponavljanja animacije, i smjer animacije. S obzirom da su ove animacije perpetualne i ponavljaju se cijelo vrijeme od pokretanja stranice do zatvaranja, njihov broj iteracija potrebno je postaviti da bude beskonačan (*infinite*), te smjer animacije je potrebno postaviti na *alternate*.

```
@keyframes zyzz1{
    0%{
        opacity: 0;
    }
    100%{
        opacity: 1;
    }
}

@keyframes zyzz2{
    0%{
        opacity: 1;
    }
    100%{
        opacity: 0;
    }
}
```

Slika 40 Definiranje animacija prijelaza ilustracije

Kod definiranja *keyframes* točaka animacija bitno je samo izmjenjivati transparentnost elemenata kako bi se postigao efekt prijelaza između obje ilustracije, na sličan način kao i polazne fotografije. U isto vrijeme kada je transparentnost (*opacity*) jedne ilustracije postavljen na vrijednost 1, ta vrijednost druge ilustracije je postavljena na 0, i obrnuto. Time se postiže glatka tranzicija i oku ugodan efekt prijelaza gradijenata jedne ilustracije na drugu. Ovaj efekt animacije, zajedno s efektom primijenjenim na prijelaz fotografija, specifičan je po tome što se izvodi cijelo vrijeme, od pokretanja odnosno posjeta stranici pa sve do njenog zatvaranja, što je postignuto neograničenim brojem iteracija odnosno ponavljanja sekvence animacije. Također, ova animacija ne mijenja se i ne pokreće se pritiskom na bilo koji od interaktivnih gumbova, te je kao takva posve nezavisna o korisničkom unosu.

Sljedeći tipovi animacija relativno su jednostavniji te ne zahtijevaju značajno opširna objašnjenja. Prva od tih animacija je tzv. „trik“, odnosno demonstrativna animacija nad dva manja elementa stranice koja nastoji prikazati drugačiji tip efekta mogućih CSS *keyframes* animacijama. Animacija efekta vrši se nad slikama šipki s utezima koje se nalaze na vrhu i na dnu navigacijskog izbornika. Ova animacija za razliku od prijašnjih vrši se primarno na klik gumba, te nema nikakvih direktnih *hover* atributa, ali njen gumb ima, što ću opisati kasnije zajedno sa svim ostalim sporednim gumbovima. Klikom na gumb vrši se CSS klasa koja je definirana animacijom zapisanom u zasebnoj *keyframes* klasi, kao i dosada, te ta animacija traje 2 sekunde i broj njezinih ponavljanja postavljen je na 1.

```
@keyframes trik{
    0%{
        transform: rotate(0deg);
    }
    100%{
        transform: rotate(360deg);
    }
}
```

Slika 41 Animacija trika

U kodu prikazanom iznad vidi se način na koji su definirane *keyframe* točke animacije; na prvoj točci, 0%, je definirano početno stanje elementa. CSS funkcija *transform* odnosi se na transformaciju elementa na razne načine, te je u ovom slučaju korištena transformacija *rotate* koja rotira HTML element po unesenoj vrijednosti. Početno stanje definirano je rotacijom od 0 stupnjeva, a konačno stanje, koje se nalazi na drugoj *keyframe* točki, 100%, je definirano rotacijom od 360°, tako da se izvrši potpuna kružna rotacija cijelog elementa i da se on nakon animacije vrati na početno stanje.

Sljedeća jednostavnija animacija koja se mora zasebno izdvojiti prije prelaska na ostale, jest *hover* preko prikazanih proizvoda. Prelaskom miša na sliku proizvoda nakon što smo ga pritiskom na određeni gumb prikazali na ekranu, dobit ćemo animaciju povećavanja proizvoda. Ova animacija vrši se na isti način kao i animacije smanjivanja ikone košarice za kupovinu, koristeći *scale* naredbu i postavljanje dviju inverznih klasa za dva slučaja; kada se pokazatelj miša nalazi na elementu na koji je primijenjena klasa, i kada se ne nalazi na njemu. Ovim načinom postiže se glatka tranzicija povećavanja *hoverom* i inverzna animacija pomicanjem miša na neko drugo mjesto u dokumentu.

Za glatki prijelaz kod otvaranja stranice, dodana je i animacija otvaranja kako se stranica ne bi prikazala abruptno i odjednom, već uz laganu i ugodnu tranziciju. To je postignuto dodavanjem klase za standardni HTML selektor *body* (nije potrebno dodavanje točke prije definiranja klase jer se odnosi na već postojeću HTML komponentu). U klasu za *body* dodaje se atribut animacije, gdje se poziva animacija koja će kasnije u kodu biti definirana pomoću *keyframeova*, kao i sve CSS3 animacije dosad. S obzirom da je ova animacija potrebna samo kod otvaranja stranice, broj iteracija postavljen je na 1. U klasi *keyframes* definiran je prijelaz transparentnosti cijelog *body* elementa s 0 na 1 kako bi se postigao blagi *fade-in* efekt cijele stranice.

Posljednja sporedna animacija koju je potrebno zasebno spomenuti jest efekt kojim se proizvodi pojavljuju na ekranu pritiskom na njihov odgovarajući gumb. Ova animacija također je različita od ostalih, zbog čega i zahtijeva posebni spomen. Kao što je bilo definirano JavaScript funkcijama, pritiskom na gumb na ekranu se pojavljuju odgovarajući proizvodi, međutim bez ikakvih animacija. U ovom dijelu rada nadograđujem već postojeće funkcije stranice tako da dodajem animacije na ove elemente. Klikom na gumb, osim samog prikaza proizvoda, također će se pokrenuti animacija kojom će se taj proizvod prikazati na ekranu korisnika. U kontekstu ovih specifičnih elemenata, odnosno proizvoda koji se prikazuju, koriste se dvije klase; jedna za pomak proizvoda (*slide*), te jedna za prikaz odgovarajućeg teksta i gumba za kupovinu.

```
@keyframes slide{
    0%{
        transform: translate(400%);
    }
    100%{
        transform: translate(0%);
    }
}

@keyframes appear{
    0%{
        opacity: 0;
    }
    50%{
        opacity: 0;
    }
    100%{
        opacity: 1;
    }
}
```

Slika 42 Animacije prikaza proizvoda

Kao što je vidljivo na slici 41, u klasi *slide* prvi *keyframe* definiran je na način da se prikazani element premjesti pomoću naredbe `transform: translate`, te ga se time pozicionira 400% na horizontalnoj osi, van korisniku vidljivog prostora. Na kraju animacije element se pomoću iste naredbe premjesti natrag na početnu poziciju na kojoj bi bez animacije bio od početka. Time se postiže animacija pomicanja elementa, odnosno „klizanja“, od desna prema lijevo. Sljedeća animacija, koja je definirana u klasi *appear*, primjenjuje se na tekst koji se pojavljuje zajedno sa slikom proizvoda (na koju se odnosi klasa *slide*). Trajanje animacija razlikuje se; trajanje animacije *slide* postavljeno je na 1 sekundu, dok animacija *appear* traje 2 sekunde. Stoga je kod definiranja *keyframeova* animacije potrebno postaviti da se tekst vezan uz proizvod počne pojavljivati tek nakon završetka prve animacije, kako bi se izbjeglo preklapanje elemenata i postigao optimalan efekt tranzicije. Nakon što je prva animacija (klizanje proizvoda) izvršena, druga animacija započinje te se tekst i gumb za kupovinu vezan uz proizvod lagano pojavljuje *fade-in* animacijom, koja je kodirana koristeći transparentnost kao i određene ostale animacije opisane prije.

Najveći dio CSS3 animacija korištenih u izradi rada nalazi se kod efekata i animacija primijenjenih na sekundarne gumbove, odnosno gumbove koji prikazuju proizvode, gumbove kojima se ti proizvodi dodaju u košaricu te slične male gumbove sa sekundarnim funkcijama. Glavna stavka unaprjeđenja doživljaja za korisnike u kontekstu ovih gumbova jest promjena njihove boje i veličine prelaskom miša, odnosno na *hover*. Kao i kod ostalih animacija, potrebno je definirati ulaznu i izlaznu komponentu, odnosno da se na *hover* preko gumba provede određena animacija, te da se prelaskom miša na neki drugi element izvršni inverzna animacija koja element vraća u inicijalno stanje. Na stranici postoji mnogo takvih „sekundarnih“ gumbova, ali na sve su primijenjeni isti principi te vrlo slične CSS klase za provedbu animacija.

```
@keyframes povecaj{
    0%{
        transform: scale(1, 1);
        background-color: #FF6800;
    }
    100%{
        transform: scale(0.8, 0.8);
        background-color: #FF8C00;
    }
}
```

Slika 43 Primjer animiranja gumba

Animacija definirana *keyframes* klasom *povecaj* kao što je vidljivo na slici 42 koristi se za mijenjanje veličine i boje gumba. Funkcija `transform: scale` radi na isti način kao i prije opisanim primjerima, odnosno smanjuje veličinu gumba, a `background-color` mijenja boju svakog elementa koji koristi ovu klasu, u ovom slučaju gumbovi na stranici. Kroz animaciju ovi se atributi elementa glatko mijenjaju.

```
.klik-gumbovi button:hover, #g1:hover{

    animation: povecaj 0.5s forwards;
    animation-iteration-count: 1;
}
```

Slika 44 Klasa animiranih gumbova

U kodu prikazanom iznad može se iščitati način na koji je animacija iz slike 42 primijenjena u samoj klasi; trajanje joj iznosi pola sekunde te se na to nadodaje atribut *forwards*, koji osigurava da ta animacija ostane u posljednjem stanju definiranom u *keyframes* klasi animacije, a ne da se izvrši jednom i da se odmah vraća na početno stanje. Time se također osigurava da kad god je pokazatelj miša još uvijek na elementu koji treba vršiti animaciju na *hover*, da animacija ostane u konačnom stanju, te da se vrati na početno pomakom miša na neko drugo mjesto. Kao i druge animacije ovog tipa, potrebno je također definirati inverznu animaciju za vraćanje na to početno stanje, pa tako postoji klasa koja ovaj element vraća vrijednost *scalea* na 1 i boju natrag na originalnu.

Ostale animacije koje se primjenjuju na sekundarne gumbove slične su prirode kao i već spomenute animacije, ali s manjim razlikama. Ove animacije primarno se odnose na stanja elemenata nakon promjene sheme boje cijele stranice, koja će detaljnije biti opisana u sljedećem odjeljku, koji je fokusiran na jQuery animacije. Za svrhe ovog argumenta, bitno je napomenuti da se na *hover* boje također mijenjaju te se *scale* postavlja na isti način kao i sa narančasto obojanim gumbovima definiranim na prijašnjem primjeru.

```
@keyframes povecajx{

    0%{
        transform: scale(1, 1);
        background-color: #4466aa;
    }
    100%{
        transform: scale(0.8, 0.8);
        background-color: #92a8d3;
    }
}
```

Slika 45 Animacije nakon promjene boja

4.5. Implementacija animiranih elemenata – jQuery

Za svrhu dodavanja animiranih elemenata i efekata na web stranice postoje mnoge tehnologije koje dizajnerima i programerima otvaraju razne mogućnosti u ovom aspektu razvoja web mjesta. Dosad korištene CSS3 *keyframes* animacije relativno su jednostavan ali moćan način dodavanja vizualne interaktivnosti i obogaćivanja stranice efektima, te s obzirom da je po prirodi CSS opisni a ne programski jezik, također su za prosječnog korisnika lakše za implementirati i koristiti u usporedbi s naprednijim metodama dodavanja animacija. Jedno od spomenutih programskih rješenja za dodavanje animiranih elemenata su jQuery animacije. jQuery se može opisati kao programska biblioteka za unaprjeđenje i pojednostavljenje JavaScript funkcionalnosti, koja uživa veoma široku uporabu na cijelom webu. Implementacijom jQuery biblioteke proširuju se mogućnosti JavaScript koda te se njenim korištenjem olakšava posao pisanja skriptnih i programskih rješenja na web mjestima.

S obzirom da je jQuery derivativni produkt JavaScripta, velikim dijelom nasljeđuje njegove funkcionalnosti i sintaksu pisanja koda. jQuery biblioteka implementira se u HTML stranicu pomoću direktne URL poveznice, ili ubacivanjem skriptne datoteke jQuery biblioteke u isti direktorij u pohrani kao i glavne datoteke web stranice. Kada je biblioteka implementirana u web stranicu, njen kod moguće je pisati u bilo kojoj kreiranoj skriptnoj datoteci koja se također veže za taj HTML dokument, pa se stoga JavaScript i jQuery kod može pisati paralelno u istoj datoteci bez konflikta. Glavne značajke korištenja jQuerya jesu simplifikacija određenih JavaScript funkcija koje obično zahtijevaju više linija koda, te skraćivanje tih funkcija u određene metode koje obavljaju iste zadatke s manje linija koda. Kao i JavaScript, jQuery biblioteka sposobna je vršiti rad na više aspekta web komponenti, kao što su manipulacija i direktna izmjena HTML i CSS koda, baratanje HTML *event* metodama, AJAX za komunikaciju sa serverom, te od najveće važnost za ovaj rad, kreacija i manipulacija animacijama i efektima korisničkog sučelja na web stranici. Za te svrhe jQuery također koristi mnoštvo *plugina* (programskih dodataka) za vršenje i pojednostavljenje određenih zadaća. Jedan specifični *plugin* korišten u ovom radu jest jQuery color *plugin*, koji omogućuje programeru da vrši razne radnje nad HTML i CSS kodom koji u standardnoj *stand-alone* biblioteci jQuerya nisu moguće, kao što je promjena *background-color* atributa CSS koda. Ovaj *plugin*, kao što mu i samo ime govori, odnosi se na sve HTML i CSS atribute koji barataju raznim bojama korištenim na stranici, koje same po sebi nisu ukomponirane u standardni jQuery *library*.

Početni korak u korištenju jQuery biblioteke za obogaćenje web stranice jest već spomenuta implementacija biblioteke u sam HTML dokument. Može se izvršiti na dva načina; dodavanjem preuzete jQuery skriptne datoteke, ili dodavanjem URL poveznice na biblioteku s određenog *Content Delivery Networka*, kao što je Google APIs. U oba slučaja, datoteka se dodaje u *header* dokumenta radi optimalne funkcionalnosti i izbjegavanja konflikta s ostalim skriptama. U svom radu sam glavnu jQuery biblioteku dodao pomoću Google APIs sustava, dok sam jQuery color *plugin* dodao koristeći preuzetu datoteku biblioteke. Nakon što su biblioteke dodane u dokument na adekvatan način, odmah su dostupne za korištenje i pisanje koda. Prva komponenta jQuery sintakse koju je potrebno opisati jesu selektori, koji se blago razlikuju od svojih JavaScript ekvivalenta. U jQueryu selektor predstavlja dio koda koji označuje i dohvaća određeni HTML ili CSS element, što se koristeći JavaScript može izvršiti primjerice deklariranjem varijable te korištenjem naredbe `document.getElementById(„identifikator-elementa“);`, dok se ista sintaksa u jQueryu vrši pomoću simbola američkog dolara te zatim „uzimanja“ oznake elementa u zagradi s navodnicima; `$(„p“)`. Ovaj selektor označit će sve <p> elemente u HTML kodu, odnosno sve odlomke teksta. Selektor se od ostatka koda odvaja točkom, te nakon točke uslijedi akcija koja će se vršiti, što može biti *click, hover* ili slične akcije. Nakon toga se otvara funkcija na sličan način kao i u JavaScriptu, te se u nju unosi kod koji će se vršiti kada se izvrši akcija koju tu funkciju inicira.

```
$(".j-gumb").click(function(){

    $(".peding-i1").slideToggle();
    $(".peding-i2").slideToggle();
});
```

Slika 46 Primjer jednostavne jQuery funkcije

U primjeru na slici 45 vidljiv je tipični kod jednostavne jQuery funkcije. Selektor je CSS klasa „j-gumb“, što znači da će se sav kod napisan u ovoj jQuery funkciji odnositi na sve elemente u HTML dokumentu koji nasljeđuju CSS klasu „j-gumb“. Nakon selektora dolazi akcija *click*, što znači da će se funkcija izvršiti kad korisnik klikne na gumb koji je povezan s jQuery funkcijom, što se definira zasebno u HTML dokumentu. Nakon toga se otvara funkcija koja selektorima dohvaća klase „peding-i1“ i „peding-i2“ te nad njima vrši funkciju *slideToggle*. Završetkom ovih akcija funkcija se zatvara, te će se na ponovni klik gumba povezanog s ovim kodom funkcija ponovno aktivirati.

Prijašnji primjer jQuery koda također se odnosi specifično na određeni gumb na stranici koji sakriva i prikazuje sadržaj. Naredba *slideToggle* korištena u funkciji vrši tu zadaću, na način da na pritisak gumba animacijom sakriva kod te da na ponovni pritisak gumba ga ponovno prikazuje, ili obrnuto, ovisno o početnom stanju elementa, odnosno je li element inicijalno sakriven ili prikazan (`display: none` ili `visibility: hidden`). Analogno naredbi *slideToggle* također postoji jednostavnija naredba *toggle*, koja vrši isti zadatak, ali bez animacije i prisutnog efekta.

Animiranje elemenata pomoću naredbi *toggle* može se izvesti pomoću nekoliko varijanta sličnog koda, sljedeća od kojih je korištena kod sakrivanja i prikazivanja elemenata zaglavlja.

```
$("#pomak-gumb").click(function(){

    $(".gumbovi").animate({
        height: "toggle"
    });
    $(".sat").slideToggle();
    $(".kontakt").slideToggle();
});
```

Slika 47 Drugačiji način animiranja uz pomoć toggle naredbe

U ovoj funkciji odmah je uočljivo da je kod u veliku ruku sličan prijašnjoj funkciji, uz nekoliko blagih razlika. Inicijalni selektor elementa poziva se na identifikator elementa, umjesto na klasu kao u prijašnjem primjeru. To se postiže korištenjem simbola ljestvi (#) ispred naziva identifikatora elementa. Nakon toga vrši se ista akcija kao i u prijašnjem primjeru, odnosno na klik će se izvršiti funkcija, a klik je definiran u HTML deklaraciji gumba. Funkcija će na selektor klase „gumbovi“ izvršiti animaciju, što se definira naredbom *animate(),* koja programeru dozvoljava direktnu manipulaciju CSS atributima tog elementa ili te klase. Uz pomoć ove naredbe element se može pomaknuti koristeći *left, right, top* ili *bottom* atribute, može mu se promijeniti visina ili širina, te ostali slični postupci manipulacije CSS kodom. Također postoji nekoliko načina na koji se ovi postupci mogu provesti; postavljanjem jednostruke brojčane vrijednosti u navodnike naredbe koja pomiče element, jeziku signaliziramo da je ta vrijednost „apsolutna“, odnosno da element pomakne točno na to mjesto u dokumentu. S druge strane, ako prije vrijednosti dodamo aritmetičke znakove + ili -, takvim kodom ćemo pomaknuti element od trenutne pozicije na novu poziciju za toliku numeričku vrijednost kolika je zadanu u navodnicima. U svakom slučaju, s obzirom da je zadana funkcija *animate,* koja god promjena da se vrši nad elementom koda, ta promjena će biti animirana, pa će tako uz svaki pomak ili promjenu veličine biti prisutan i odgovarajući animirani efekt. U specifičnom kontekstu ove

funkcije, za CSS klasu „gumbovi“, koju nasljeđuju svi funkcionalni gumbovi koji otvaraju glavne kategorije proizvoda te se nalaze u zaglavlju, će se primijeniti animacija koja mijenja visinu selektiranog elementa (gumbovi) na „toggle“. To znači da će se klikom na odgovarajući gumb označen u HTML-u visina svih elemenata s klasom „gumbovi“ postaviti na 0, te ponovnim klikom postaviti na inicijalnu vrijednost, te pošto je funkcija definirana animacijom, također će se taj postupak animirati. Sljedeći dio cjelokupne funkcije isti je kao i ona u prijašnjem primjeru, odnosno koristi se *slideToggle* naredba za sakrivanje i prikazivanje elemenata. Jedina razlika je što se ista naredba odnosi na druge HTML elemente, također označene klasama (u ovom slučaju su to klase „sat“ i „kontakt“), te će taj gumb sve zajedno sakriti ili prikazati sve elemente označene klasama „gumbovi“, „sat“ i „kontakt“, ali na blago različit način.

Glavna jQuery funkcija vezana uz animacije u ovom radu jest kompletna promjena sheme boja stranice, s originalno narančaste na jarko plavu. Preduvjet za funkcionalnost ovakve funkcije jest implementacija jQuery color biblioteke, s obzirom da standardna jQuery biblioteka nema mogućnost manipulacije CSS elementima koji mijenjaju boje (s izuzetkom *color* atributa za promjenu boje teksta). Color *plugin* za jQuery bilo je potrebno implementirati direktnim preuzimanjem biblioteke te postavljanjem u direktorij web stranice, pošto ova biblioteka nije javno dijeljena na serveru kao što je Google APIs. Nakon implementacije *plugin* biblioteke u *header* HTML dokumenta, moguće je mijenjati CSS atribute koji manipuliraju raznim bojama HTML elemenata. Ovaj proces vrši se na sličan način kao i prijašnje dodavanje animacija uz jQuery, s obzirom da color *plugin* ne dodaje eksplicitno nove funkcije, već otvara mogućnost manipuliranja određenim CSS elementima koje jQuery inicijalno ne mijenja. Primjer koda koji mijenja boju elementa nalazi se ispod.

```
$(".gumb1").click(function(){

    $(".trokut").animate({
        backgroundColor: "#FF6800"
    }, 1500 );
    $(".gumb").animate({
        backgroundColor: "#FF6800"
    }, 1500 );
    $(".sat").animate({
        backgroundColor: "#FF6800"
    }, 1500 );
    $(".j-gumb").animate({
        backgroundColor: "#FF6800"
    }, 1500 );
```

Slika 48 Promjena boje pomoću jQuerya

U kodu na slici 47 vidljiva je slična sintaksa kao što je primijenjena u prijašnjim funkcijama za prikazivanje i sakrivanje elemenata; koristi se selektor koji poziva određeni gumb u HTML dokumentu, na čiji se klik provode razne akcije. U ovom slučaju su to označavanje svakog individualnog elementa u HTML stranici koji je narančaste boje (na slici ih je prikazano samo nekoliko), kojima se zatim animacijom mijenja pozadinska boja (backgroundColor) u onu koju želimo. Na slici je prikazana funkcija koja te elemente mijenja natrag u narančastu boju (#FF6800). Nakon što se odredi akcija *animate()*, također se može odrediti i trajanje te akcije, u ovom slučaju animacije. U prijašnjim primjerima trajanje animacije nisam mijenjao, što ju zatim ostavlja na zadanim postavkama trajanja (*default*). Za ove animacije sam odlučio da će imati trajanje od 1500 milisekundi. Trajanje, odnosno brzina animacije, može se konfigurirati predodređenim vrijednostima (*fast*, *slow*) ili specifičnom numeričkom vrijednošću određenom milisekundama. Također, nakon dodavanja brzine može se dodati još jedan opcionalni parametar, a to je tzv. naredba *callback*, što označava akciju koja će se izvršiti nakon što animacija izvrši svoje trajanje. *Callback* funkciju iskoristit ću u kasnijim primjerima animacija pomoću jQuery tehnologije.

Nakon što je boja svakog narančastog elementa promijenjena u plavo ili obrnuto, potrebno je u istoj funkciji promijeniti boju loga. Za tu svrhu stvorio sam novu varijantu logotipa, u odgovarajućim bojama nove sheme boja stranice. Nakon toga taj logotip potrebno je ukomponirati u HTML dokument koristeći određene jQuery naredbe. To je izvršeno na sljedeći način:

```
var logo = $(".logo");

    logo.fadeOut('slow', function(){
        logo.attr('src', '../grafika/logo-muzz-small-1.png');
        logo.fadeIn('slow');
    })
```

Slika 49 jQuery zamjena logotipa

Deklarira se nova varijabla u koju će se pomoću jQuery selektora učitati odgovarajući HTML element, ili u ovom slučaju CSS klasa koja je direktno povezana s HTML elementom logotipa na stranici. S obzirom da je selektor već inicijaliziran u varijabli, u ostatku koda koristi se samo ime varijable, pa se tako u prvoj liniji nakon deklaracije postavlja *fadeOut* efekt, koji će selektor u varijabli animacijom sakriti. U zagradi naredbe se definira brzina animacije (postavljamo ju na *slow*), te se zatim otvara nova funkcija koja služi kao upravo spomenuti *callback* nakon inicijalne animacije. Ta funkcija mijenja HTML atribute varijable logo i direktnom promjenom HTML

koda mijenja njen izvor, pa tako i cijelu sliku koja se prikazuje. Nakon toga se vrši animacija *fadeIn*, također sporom brzinom.

Glavna neželjena pojava kod ovog procesa promjene kompletne sheme boja na stranici bilo je resetiranje određenih postavljenih CSS3 animacija, specifično animacija vezanih za gumbove i njihova promjena boja i veličine. Stoga je optimalno rješenje ovog problema bilo ponovno postavljanje tog specifičnog CSS koda pomoću iste jQuery funkcije.

```
$(".klik-gumbovi button").css("animation", "povecaj1 0.5s");

    $(".klik-gumbovi button").css("animation-iteration-count", "1");
    $(".novigumb").css("animation", "povecaj1 0.5s");
    $(".novigumb").css("animation-iteration-count", "1");
    $(".klik-gumbovi button").hover(function(){
        $(this).css("animation", "povecaj 0.5s forwards");
    }, function(){
        $(this).css("animation", "povecajr 0.5s forwards")
    })
    $('[id="g1"]').hover(function(){
        $(this).css("animation", "povecaj 0.5s forwards");
    }, function(){
        $(this).css("animation", "povecajr 0.5s forwards")
    })
})
```

Slika 50 Automatizirano ponovno postavljanje CSS koda

Svaki element koji je potrebno promijeniti poziva se sa jQuery selektorom, kao i do sada. Nakon što je adekvatni element selektiran, koristi se naredba *css* koja modificira *css* kod tog elementa (koji je u većini slučajeva na primjeru CSS klasa) na način da povratimo funkcionalnost CSS3 *keyframes* animacija. To se postiže označavanjem specifičnog atributa koji mijenjamo (u ovom slučaju *animation*) te zatim odijelimo element i njegovu vrijednost zarezom, koju nakon toga definiramo. Vrijednost atributa mora biti jednaka onoj postavljenoj u CSS dokumentu, kako bi se adekvatno povratila funkcionalnost animacija. S obzirom da naredba *css* može promijeniti samo jednu liniju CSS koda, potrebno je upisati novu liniju koda za svaku CSS liniju koja se modificira, pa tako u sljedećoj liniji koda ponovno postavljamo broj iteracija CSS3 animacije na vrijednost zadanu u CSS datoteci i kodu. Nakon toga isti proces ponavljamo za drugu klasu interaktivnih gumbova, „novigumb“. Zatim je potrebno ponovno definirati *hover* funkciju. S obzirom da je potrebno definirati što se događa kad je pokazatelj miša na gumbu, te također što se događa nakon što se pokazatelj makne s gumba, potrebno je u istoj funkciji obje definirati; *hover* u početnoj sintaksi te zatim *callback* funkciju koja će se vršiti kada se kursor makne s gumba. Također, u ovoj funkciji uočljiva je uporaba selektora *this*, koji označava da se

radi o selektoru definiranom kod same deklaracije, u ovom slučaju klasa „klik-gumbovi button“. Nakon toga se na isti način vrši promjena CSS koda kao i u gornjim linijama, te se na vrijednost animacije dodaje atribut „forwards“ kako je bilo i u originalnom CSS kodu. Atribut „forwards“, kao što je bilo opisano ranije u radu, osigurava da animacija neće stati direktno nakon *hovera*, već da će ostati u krajnjem, odnosno konačnom položaju sve dok se kursor miša ne pomakne s tog određenog elementa. Posljednji dio ove funkcije odnosi se na sve specifične HTML elemente s određenim identifikatorom. U ovom slučaju, kada više HTML elemenata ima jednaki identifikator, da bi funkcija obuhvaćala sve te elemente potrebno je koristiti sintaksu kako je prikazana na primjeru; u selektoru uz jednostruke navodnike i uglate zagrade napisati točan HTML kod koji definira taj identifikator. U ovom slučaju je to identifikator „g1“, koji označuje sve gumbove za kupovinu proizvoda. Ako je kod selektor deklariran kao na primjeru, tada će se sljedeći kod odnositi na sve elemente s ovim identifikatorom, dok će se u protivnom kod korištenja ljestava što je standardna procedura s identifikatorima koristiti samo prvi element s tim identifikatorom. Sljedeći dio koda označuje da se na *hover* tog elementa pokreće funkcija koja na isti način kao i prijašnje linije koda modificira CSS kod identifikatora ove klase kako bi se održala puna funkcionalnost animacija. Slike 49, 48 i 47 čine jednu cjelinu, odnosno kompletnu funkciju, s tim da je sa slike 47 izuzeto većina elemenata narančaste boje, s obzirom da ih ima puno.

Posljednje jQuery animacijske funkcije odnose se na demonstrativni prikaz korištenog koda. Njihove sintakse praktički su identične prijašnje korištenim *slideToggle* funkcijama, samo uz drugačiji poziv selektora.

```
$('[name="primjer1"]').click(function(){

    $("#kod1a").slideToggle();
    $("#kod1b").slideToggle();
})

$('[name="primjer2"]').click(function(){
    $("#kod2a").slideToggle();
    $("#kod2b").slideToggle();
})
```

Slika 51 Prikaz koda pomoću jQuerya

Razlika kod poziva elementa u selektoru jest ta da sam zbog već korištenih identifikatora i klasa bio prisiljen ovim specifičnim elementima dodati *name* atribut kako bi ih mogao na funkcionalan način pozvati u jQuery funkciji, te s obzirom da jQuery nema točnu naredbu za poziv elemenata preko *name* atributa, morao sam iskoristiti sličan kod kao u prijašnjem primjeru;

uporaba jednostrukih navodnika i uglatih zagrada za pozivanje točnog HTML elementa prema identičnom kodu. U obje funkcije se zatim klikom na gumb istovremeno prikazuje jedan HTML a sakriva drugi, odnosno onaj koji je inicijalno sakriven se prikazuje i obrnuto.

5. Analiza rezultata

Dodavanjem svih ključnih funkcija u JavaScript datoteku te povezivanjem istih s HTML kodom konačno ostvarujemo očekivanu i željenu funkcionalnost web stranice, koja zajedno s elementima dizajna opisanih u CSS datoteci, CSS3 *keyframes* animacijama i jQuery animacijama te samom strukturom stranice definiranoj u HTML datoteci dodaje konačnom ukupnom dojmu same stranice. U ovom poglavlju prikazat ću rezultate svakog dijela koda opisanog u prijašnjem poglavlju rada.

5.1. Zaglavlje

Prvi ključni element stranice koji sam odradio je bilo zaglavlje, koje je prvi element opisan tekstom i slikama u poglavlju 3.2, te ću kao podsjetnik ovdje reiterirati ključne stavke ovog elementa pa zatim prikazati krajnji rezultat upisanog koda. Zaglavlje sam definirao na vrhu stranice, postavio pozadinsku boju na crnu, te postavio nekoliko geometrijskih oblika i ilustraciju sebe modificiranu u Photoshopu radi poboljšanja grafičkog dojma. Rezultat:

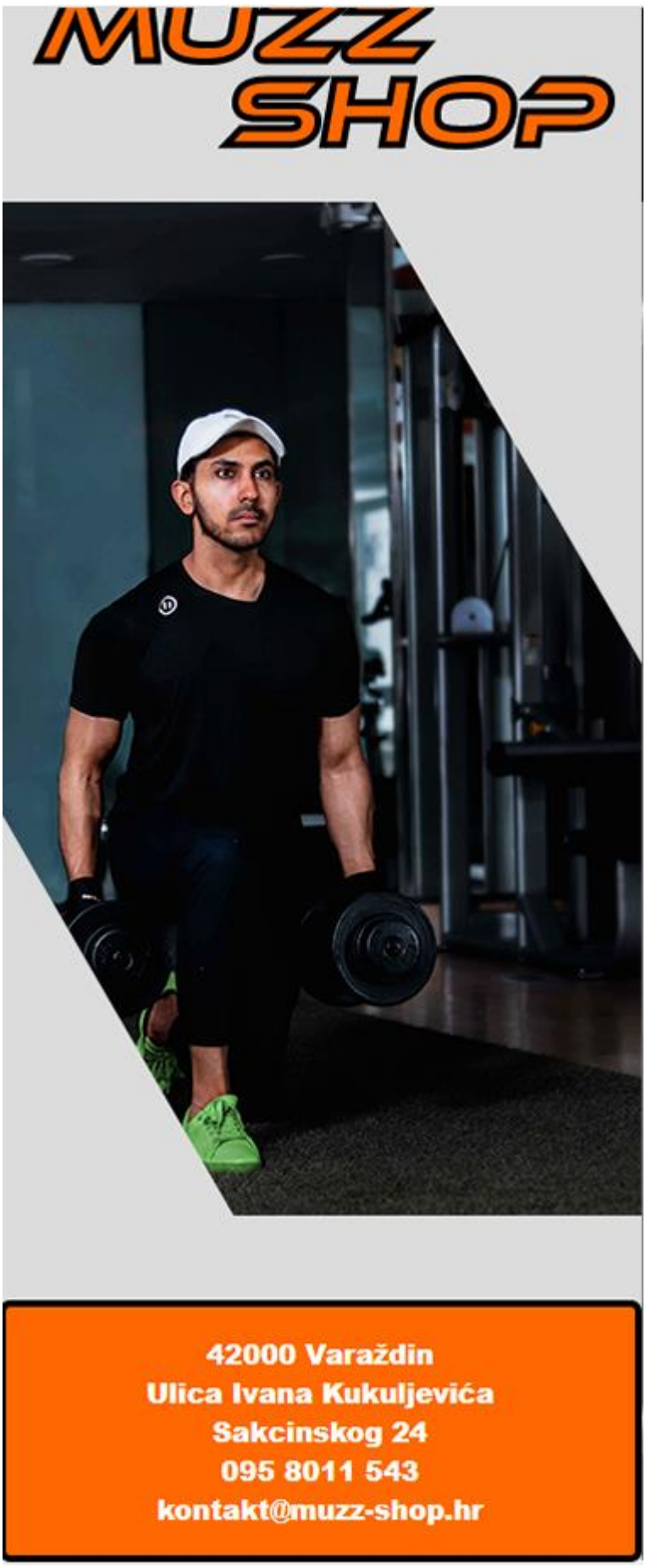
Slika 52 Konačni izgled zaglavlja

Napomena: funkcionalni gumbovi su u konačnom rezultatu dio zaglavlja, ali u kodu nisu direktno podređeni *div* elementu zaglavlja.

5.2. Lijevi odjeljak

Sljedeći glavni dio stranice jest odjeljak s lijeve strane stranice. On sadrži vrijeme pristupa stranici u gornjem lijevom kutu koje je sadržano unutar okvira, zatim glavni logotip stranice, te ispod logotipa dvije izmjenjujuće i animirane slike, a ispod tih slika informacije za kontakt web shopa, konstruirane na sličan način kao i gornji prikaz vremena pristupa stranici. Pristup vremena poziva se na JavaScript funkciju kako bi prikazao vrijeme i datum, dok se animirane slike ispod njega i logotipa oslanjanju na CSS klase s *keyframe* značajkama.

Slika 53 Gornja polovica odjeljka sa satom

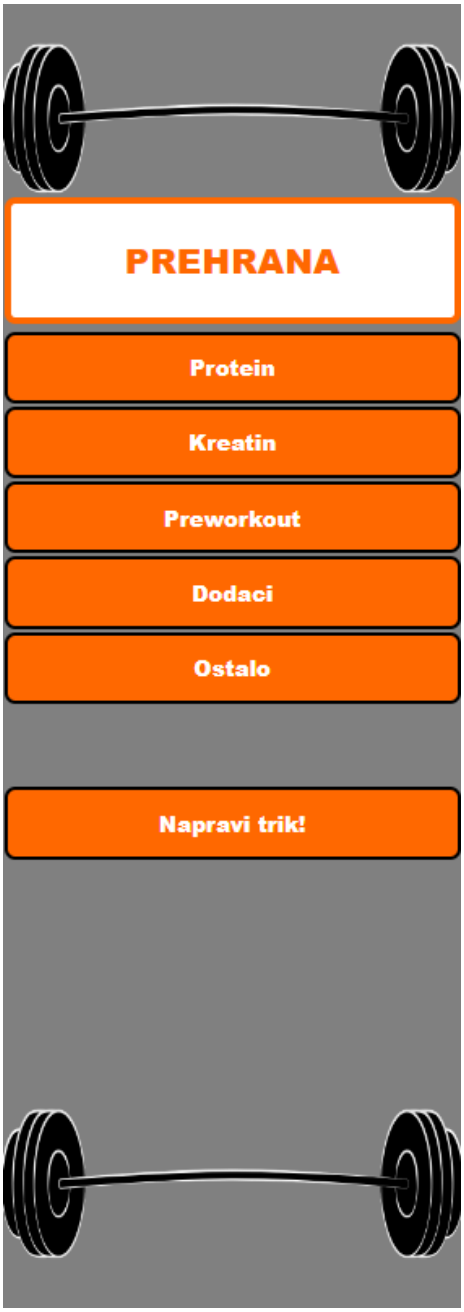


Slika 54 Donja polovica odjeljka s kontakt informacijama

5.3. Navigacija

Sljedeća ključna komponenta stranice jest navigacija, koja se nalazi desno od lijevog odjeljka i direktno ispod zaglavlja. Kod navigacijskog okna sam za pozadinsku boju odabrao malo tamniju nijansu sive od lijevog odjeljka, radi boljeg kontrasta i očitavanja. Na vrhu i na dnu okna sam postavio ilustraciju šipke za bolji grafički dojam, te između njih sam postavio glavne oblačiće navigacijskog okna, koji nestaju pritiskom na tipke i zamjenjuju ih navigacijski izbornici. Gumb „Napravi trik!“ vrši animaciju opisanu u klasi trik.

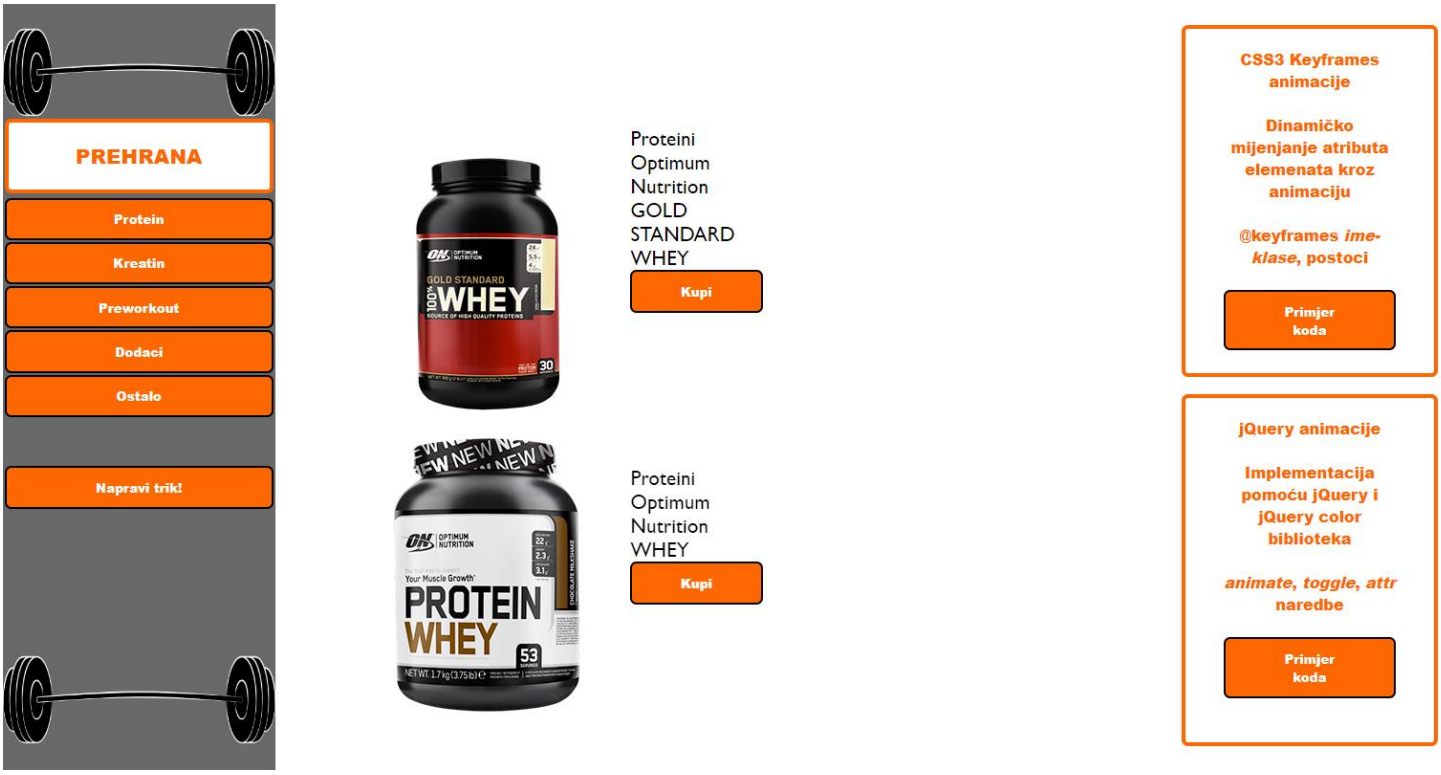
Slika 55 Standardno navigacijsko okno



Slika 56 Navigacijsko okno nakon pritiska gumba

5.4. Prikaz proizvoda

U odjeljku stranice koji zauzima donji lijevi bijeli okvir je dizajnirano da se pojavljuju proizvodi. Pod zadanim postavkama, odnosno kod otvaranja stranice, na tom mjestu se otvara generička slika suplemenata. Nakon klika na neku od kategorija proizvoda u navigacijskom oknu, otvaraju se prikazi odgovarajućih proizvoda sukladno kategoriji koju je posjetitelj odabrao.

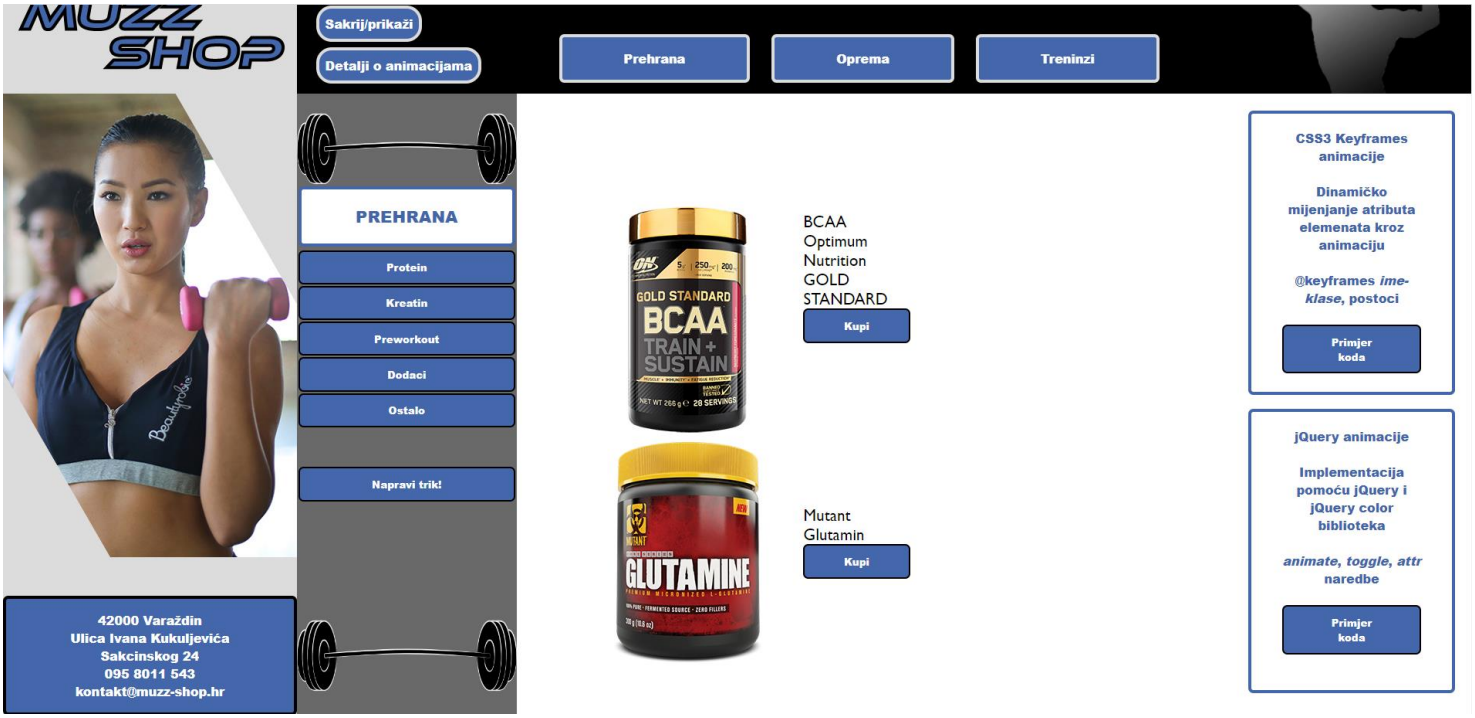


Slika 57 Otvaranje odgovarajućih prikaza proizvoda klikom na kategoriju

5.5. Konačni rezultat

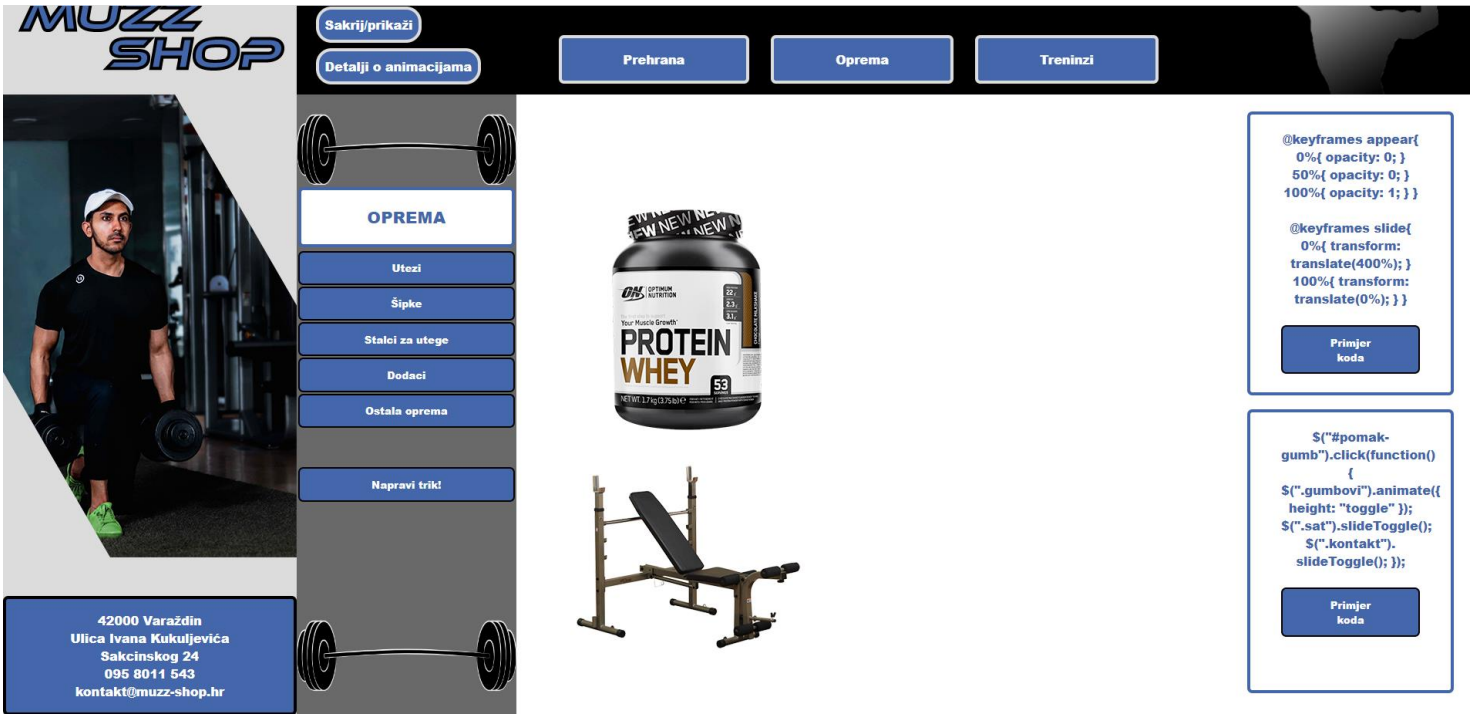
U sljedećih nekoliko stranica pokazat ću kompletni izgled stranice kod otvaranja te nekoliko mogućih izgleda stranice, ovisno o korisničkim unosima, odnosno ovisno o tome koje funkcionalne gumbove je korisnik pritisnuo. S obzirom da u PDF dokumentu nije moguće prikazati animirane elemente stranice, već samo statičke elemente preko snimaka zaslona, za pregledavanje animacija poželjno je otvoriti stranicu na Arwen poslužitelju Sveučilišta, link na koji se može pronaći na kraju ovog poglavlja.

Slika 58 Izgled stranice kod otvaranja



Slika 59 Izgled stranice kada se promijene boje i otvori jedna od kategorija

Napomena: da je moguće pritisnuti gumb dodaci, prvo se mora otvoriti izbornik „PREHRANA“ koristeći gumb Prehrana u zaglavlju.



Slika 60 Otvorena košarica s dva kupljena proizvoda i prikazom koda

Slika 61 Stranica nakon povratka na narančastu shemu i sakrivanja gumbova

Kompletni rezultat koda nalazi se na Arwen serveru Sveučilišta. Otvaranjem stranice vidljive su sve animacije i elementi stranice. Napomena: za optimalno pregledavanje stranice, otvorite ju u rezoluciji 1920x1080 piksela. Stranicu otvorite ovdje:

http://arwen.unin.hr/~tibutko/zavrsni/stranica/

6. Zaključak

Kod izrade web stranice koristeći *front-end* alate potrebno je uspostaviti adekvatnu i smislenu strukturu stranice, prvotno u HTML kodu, kako bi se osigurala optimalna čitljivost i dobar grafički dojam. HTML postavlja temelje svake web stranice, te preostali *front-end* jezici služe kako bi ga upotpunili i proširili. Uporabom CSS opisnog jezika stranici dajemo život, proširujemo njezine grafičke sposobnosti i osiguravamo da stranica bude atraktivna i pristupačna. S druge strane, JavaScript skriptni jezik omogućuje nam implementaciju raznih programskih funkcija koje dodatno upotpunjuju našu web stranicu, tako da možemo dinamičkim putem i korisničkim unosima mijenjati određene HTML i CSS elemente stranice, a također otvara i određene mogućnosti u svezi animacija, koje su još jedan ključan aspekt dojmljive i impresivne web stranice.

Prvotna i primarna svrha animacija jest unaprjeđenje korisničkog iskustva i dojma cjelokupne stranice. Glavne tehnologije animacija korištene u ovom radu su CSS3 *keyframes* i jQuery tehnologije animacija, koje su mi omogućile znatno poboljšanje doživljaja stranice i pretvorile statičku stranicu u dinamično web mjesto s atraktivnim efektima. CSS3 animacije omogućile su jednostavnu implementaciju naprednih tipova animacija, s mnogim mogućnostima modifikacije na mjestu i bez previše muke. Uz pomoć ove tehnologije animacija, lagano sam mijenjao i dodavao razne efekte za poboljšanje korisničkog iskustva na stranici. S druge strane, jQuery animacije omogućile su različito, ali jednako efikasno dodavanje prijelaznih efekata te su istovremeno znatno olakšale direktno povezivanje tih animacija i efekata s interaktivnošću stranice, s obzirom da je jQuery dizajniran za lakše baratanje JavaScript funkcijama. Sve u svemu, ove dvije tehnologije značajno su olakšale dodavanje atraktivnih efekata i omogućile mi da daleko unaprijedim korisničko iskustvo na prijašnje statičkoj stranici.

Nadovezujući se na animacije, potencijalna poboljšanja u ovom aspektu mogla bi se postići Scalable Vector Graphics elementima, koji su implementirani u HTML5 jezik te uživaju nativnu potporu. WebGL također je tip animacija zastupljen na brojnim web stranicama koji može na atraktivne načine poboljšati korisnička iskustva pristupa stranicama.

7. Literatura

[1] L. Végh: Creating Interactive JavaScript Animations for Demonstrating Algorithms on One-Dimensional Arrays, XXX. DIDDMATECH, Trnava, 2017., str. 75-78

[2] https://www.usg.edu/galileo/skills/unit07/internet07_01.phtml, dostupno 26.09.2022.

[3] https://www.livescience.com/20718-computer-history.html, dostupno 26.09.2022.

[4] https://www.internetsociety.org/internet/history-internet/brief-history-internet/, dostupno 26.09.2022.

[5] https://plato.stanford.edu/entries/turing-machine/, dostupno 26.09.2022.

[6] https://www.washington.edu/accesscomputing/webd2/student/unit1/module3/html_history.html, dostupno 27.09.2022.

[7] http://css3.bradshawenterprises.com/cfimg, dostupno 20.09.2022.

[8] https://www.w3schools.com/howto/howto_js_toggle_hide_show.asp, dostupno 22.09.2022.

[9] https://www.w3schools.com/css/css3_animations.asp, dostupno 27.09.2022.

[10] https://www.w3schools.com/jquery/jquery_animate.asp, dostupno 27.09.2022.

[11] https://api.jquery.com/category/effects/, dostupno 27.09.2022.

[12] https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function/translate, dostupno 26.09.2022.

[13] https://www.30secondsofcode.org/css/s/zoomin-zoomout-animation, dostupno 26.09.2022.

[14] https://www.w3schools.com/jquery/jquery_animate.asp, dostupno 26.09.2022.

[15] https://www.w3schools.com/css/css3_animations.asp, dostupno 27.09.2022.

[16] https://stackoverflow.com/questions/12991164/maintaining-the-final-state-at-end-of-a-css3-animation, dostupno 27.09.2022.

[17] https://www.w3schools.com/howto/howto_js_toggle_class.asp, dostupno 26.09.2022.

[18] https://medium.com/@milberferreira/the-history-of-web-animation-63b106c97fdf, dostupno 27.09.2022.

[19] https://computer.howstuffworks.com/web-animation.htm, dostupno 27.09.2022.

Popis slika

Slika 1 Logotipovi HTML-a, CSS-a i JavaScripta. 4
Slika 2 Najpoznatiji programski jezici današnjice. 5
Slika 3 Front-end/back-end. 6
Slika 4 Primjeri CMS servisa. 7
Slika 5 jQuery logotip..... 13
Slika 6 HTML plugin. 14
Slika 7 JavaScript plugin. 14
Slika 8 Zasebne datoteke stranice. 15
Slika 9 Povezivanje CSS datoteke s HTML okosnicom. 15
Slika 10 Povezivanje JavaScript datoteke s HTML okosnicom. 16
Slika 11 Početne postavke dokumenta logotipa. 16
Slika 12 Gotov logotip web shopa. 17
Slika 13 Početni body atribut uz nekoliko div elemenata. 18
Slika 14 CSS klasa zaglavlja. 19
Slika 15 Prva polovica diva zaglavlja. 20
Slika 16 CSS klasa trokuta. 20
Slika 17 CSS klasa kruga. 21
Slika 18 Kompletni prikaz cijelog diva zaglavlja. 21
Slika 19 Implementacija fonta u CSS. 22
Slika 20 CSS klasa gumb. 22
Slika 21 CSS klasa odjeljka s lijeve strane. 23
Slika 22 Kompletni "strana" div. 24
Slika 23 Kodirani efekti transparentnosti. 25
Slika 24 Klasa za kontakt. 26
Slika 25 Div element identifikatora nav. 27
Slika 26 Ostatak glavnog div elementa. 28
Slika 27 Odjeljak s funkcionalnim gumbovima kupovine. 29
Slika 28 JavaScript funkcija datuma. 31
Slika 29 JavaScript funkcija za navigacijski dio. 32
Slika 30 Globalne varijable s postavljenim vrijednostima. 33
Slika 31 Globalne varijable koje dohvaćaju gumbove u HTML-u. 33
Slika 32 Funkcije za kupovinu. 34
Slika 33 Prikaz funkcije za dodavanje elemenata u košaricu. 35

Slika 34 Dio funkcije isprazni().....36
Slika 35 Kompletna funkcija za prikaz proteina. 37
Slika 36 Kod tranzicije gumbova. 39
Slika 37 Klase animiranja košarice. 40
Slika 38 Animacije korištene kod prikaza košarice.....40
Slika 39 Klase animiranja ilustracije. 41
Slika 40 Definiranje animacije prijelaza ilustracije. 42
Slika 41 Animacija trika. 43
Slika 42 Animacije prikaza proizvoda.....44
Slika 43 Primjer animiranja gumba. 45
Slika 44 Klasa animiranih gumbova. 46
Slika 45 Animacije nakon promjene boja.....46
Slika 46 Primjer jednostavne jQuery funkcije. 48
Slika 47 Drugačiji način animiranja uz pomoć toggle naredbe. 49
Slika 48 Promjena boje pomoću jQuerya. 50
Slika 49 jQuery zamjena logotipa. 51
Slika 50 Automatizirano ponovno postavljanje CSS koda. 52
Slika 51 Prikaz koda pomoću jQuerya. 53
Slika 52 Konačni izgled zaglavlja. 55
Slika 53 Gornja polovica odjeljka sa satom. 56
Slika 54 Donja polovica odjeljka s kontakt informacijama. 56
Slika 55 Standardno navigacijsko okno. 57
Slika 56 Navigacijsko okno nakon pritiska gumba. 57
Slika 57 Otvaranje odgovarajućih prikaza proizvoda klikom na kategoriju. 58
Slika 58 Izgled stranice kod otvaranja. 59
Slika 59 Izgled stranice kada se promijene boje i otvori jedna od kategorija. ...59
Slika 60 Otvorena košarica s dva kupljena proizvoda i prikazom koda. 60
Slika 61 Stranica nakon povratka na narančastu shemu i sakrivanja gumbova. 60

Prilozi

Ovdje su priloženi svi vanjski resursi korišteni kod izrade web stranice.

https://www.1001fonts.com/ethnocentric-font.html

https://www.pexels.com/photo/man-holding-black-dumbbells-2105493/

https://www.pexels.com/photo/woman-wearing-black-and-gray-sport-bra-869243/

https://www.naturalproductsinsider.com/sports-nutrition/sports-nutrition-innovation-often-reinvention

https://favpng.com/png_view/shopping-cart-shopping-cart-royalty-free-stock-photography-png/d04qLvjK

https://www.subpng.com/png-fdykk7/

https://www.subpng.com/png-292aft/

https://www.subpng.com/png-xlkqxs/

https://www.subpng.com/png-vunwnv/

https://www.subpng.com/png-5my755/

https://www.subpng.com/png-43ddk9/

https://www.subpng.com/png-keljz9/

https://www.subpng.com/png-4n5pkq/

https://www.subpng.com/png-0hp10h/

https://www.subpng.com/png-ydpvku/

https://www.subpng.com/png-n5gfq2/

https://www.subpng.com/png-32pkqx/

https://www.subpng.com/png-zsz0is/

https://www.subpng.com/png-xb4w9l/

https://www.subpng.com/png-uoil68/

https://www.subpng.com/png-4tfear/

https://www.subpng.com/png-thssky/

https://www.subpng.com/png-ej32ls/

https://www.subpng.com/png-eahvp9/

https://www.subpng.com/png-jl4pgt/

https://jquery.com/

Sveučilište
Sjever

UNIVERSITY NORTH

SVEUČILIŠTE SJEVER

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski 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, Tina Butka _____ *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom Obogaćivanje web stranice animacijama *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)
Tina Butka
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, Tina Butka _____ *(ime i prezime)* neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog/~~diplomskog~~ (*obrisati nepotrebno*) rada pod naslovom Obogaćivanje web stranice animacijama *(upisati naslov)* čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)
Tina Butka
(vlastoručni potpis)