SVEUČILIŠTE U ZAGREBU

GRAFIČKI FAKULTET

ZAVRŠNI RAD

Daria Prožek

Sveučilište u Zagrebu
Grafički fakultet

Smjer: Tehničko-tehnološki

ZAVRŠNI RAD

PRILAGOĐAVANJE WEB SADRŽAJA RAZLIČITIM REZOLUCIJAMA PREGLEDNIKA

Mentor:

Doc. dr. sc. Tajana Koren Ivančević

Student:

Daria Prožek

Zagreb, 2016.

SAŽETAK

Završni rad obrađuje i rješava problematiku izrade responzivne web stranice, odnosno prikazivanja web sadržaja različitim rezolucijama web preglednika namijenjenim uređajima različitih veličina zaslona – pametnim telefonima, tabletima i računalima.

Korištenjem HTML5 i CSS3 (media queries) tehnologija, o kojima će se govoriti u teorijskom dijelu, manipulira se prikazom sadržaja koji se prikazuje ispravno na svakom zaslonu. Osim problema prikaza samog sadržaja, rješava se i problem navigacije kako bi web stranica bila funkcionalna i praktična za pregled na svim uređajima. Problem navigacije u ovom radu se rješava pomoću CSS i JavaScript tehnologija koristeći jQuery biblioteke. Također, JavaScript i jQuery tehnologija se koristi pri izradi galerije u svrhu njenog olakšanog korištenja.

Krajnji produkt ovog završnog rada je funkcionalna web stranica , prilagođena za danas najviše korištene rezolucije monitora, tableta i pametnih telefona. Web stranica je galerija obrađenih fotografija kataloškog, modnog i umjetničkog karaktera.

KLJUČNE RIJEČI: responzivna web stranica, HTML5, CSS3, JavaScript, jQuery

ABSTRACT

This final thesis deals with the problem of making a responsive web site that is showing the content in different resolution of web browsers, made for devices with various screen sizes- smart phones, tablets or desktop screen.

The content is being adjusted to all screen sizes using the HTML5 and CSS3 (media queries) technologies, which are presented in theoretical part of this thesis. Also, this thesis is dealing the problem of web site navigation to make web site functional and easy to use on every device. The problem of navigation is solved using CSS and JavaScript's library jQuery technologies which are also being used in making of gallery.

The product of this thesis is functional web site, adjusted for the most popular resolutions of desktop screens, tablets or smart phones. Web site is actually a gallery which presents retouched photos.

KEYWORDS: responsive web site, HTML5, CSS3, JavaScript, jQuery

SADRŽAJ

1. UVOD

Internet je danas jedan od najmasovnijih medija koji može pružiti veliki broj različitih vrsta informacija u kratkom vremenu. Kao takav medij, danas dostiže brojku od preko 3 milijarde korisnika u cijelom svijetu. Osim putem računala, razvojem tehnologije omogućen je pristup internetu preko pametnih telefona i tableta.

Internet je naziv za javno dostupnu globalnu podatkovnu mrežu koja zajedno povezuje računala i računalne mreže korištenjem internetskog protokola. Neke od usluge interneta su: razgovori ili *chat*, elektronička pošta, prijenos datoteka i *World Wide Web* kao najpoznatija usluga. Web služi za prezentaciju internetskih stranica te omogućava dohvaćanje hipertekstualnih dokumenata koristeći *HyperText Transfer Protocol - HTTP* kao najčešću metodu prijenosa. Internetske stranice se sastoje od niza HTML dokumenata, a može im se pristupiti uz pomoć internet preglednika, odnosno *browsera*. U engleskom jeziku postoje nazivi *web page* i *web site*. *Web page* je naziv koji označava internetsku stranicu kao HTML dokument koji omogućava prezentaciju teksta, poveznica i multimedijskih elemenata, a dostupan je preko svoje *URL* ili web adrese. *Web site* označava kolekciju internetskih resursa- HTML dokumenata, multimedijskih elemenata, skripti i podataka. Svi dokumenti su spremljeni na istom web serveru, a pristupa im se preko zajedničke web adrese koja se zove *homepage*, odnosno početna stranica.

Sve većom upotrebom interneta i pojavom pametnih telefona razvijaju se različiti koncepti prilagodbe web sadržaja različitim rezolucijama preglednika kao što su responzivan, adaptivan i fluidan web dizajn. Cilj responzivnog dizajna jest korisnicima prikazati istu internet stranicu na različite načine, ovisno o mogućnostima prikaza njihovog uređaja. Načini prikazivanja internetske stranice zadani su preko *CSS*-a uz pomoć *media queries* tehnologije. Fluidan dizajn elemente internet stranice prikazuje u postocima te se sadržaj na taj način prilagođava veličini preglednika. Adaptivan dizajn nudi nekoliko vrsta prikaza internet stranice, a korisnik dobiva prikaz koji najbolje odgovara veličini zaslona, odnosno preglednika. Osim ova 3 načina, postoji i mogućnost izrade posebne web stranice namijenjene za pametne telefone. Kako bi dizajn internetske stranice bio funkcionalan na svim uređajima, potrebno je poznavati gore navedene koncepte, njihove mogućnosti te njihove prednosti i mane kako bi krajnji rezultat bio u potpunosti zadovoljavajući.

2. TEORIJSKI DIO

2.1 HTML

HyperText Markup Language, odnosno HTML je prezentacijski jezik za izradu internet stranica. [1] Pomoću HTML jezika oblikuje se sadržaj te se stvaraju hipertekstualni dokumenti i hiperveze hipertekst dokumenata. Internet preglednici, odnosno *browseri* preko HTML-a dobivaju informacije o sadržaju i strukturi internet stranice te zatim oblikuju stranicu u onakvu kakvu ju korisnik vidi. Dakle, zadaća HTML-a jest uputiti internet preglednik kako prikazati hipertekst dokument.

Hipertekst je naziv za tekstualnu strukturu koja se sastoji od međusobno povezanih informacija te se prikazuje na nekom elektroničkom uređaju. Takva tekstualna struktura se od tradicionalne razlikuje po tome što nema redoslijed čitanja, već ga određuje sam čitatelj zbog čega je hipertekst nesekvencijalan. Također, hipertekst je i modularan što znači da uvijek može biti dorađen te kao takav se nikad ne smatra dovršenim. Hipertekstualni dokumenti se od običnih dokumenata razlikuju po tome što sadrže hiperveze, odnosno *hiperlinkove*, uz pomoć kojih su povezani s drugim hipertekstualnim dokumentima. Kako su *web* stranice danas multimedijalni, a ne samo tekstualni dokumenti, razvio se i pojam *hipermedij* za sve međusobno povezane multimedijske elemente.

HTML se ne smatra programskim jezikom jer se njime ne može izvršiti nikakva operacija. On služi samo za prikaz hipertekstualnih dokumenata. HTML kôd se može pisati u jednostavnim programima za uređivanje teksta, odnosno tekst editorima. HTML datoteka je obična tekstualna datoteka s ekstenzijom *.html* ili *.htm*. Osnovni dio HTML-a su njegovi elementi koji se nalaze unutar svojih oznaka, odnosno *tagova.* Većina elemenata ima svoju otvarajuću i zatvarajuću oznaku. Svaki HTML element internet pregledniku daje informaciju o sadržaju koji se nalazi unutar oznaka.

HTML je jednostavan jezik koji razumije svako računalo. Iako su neke kompanije pokušavale patentirati HTML, on je i dalje besplatan za upotrebu što je njegova velika prednost. Zbog toga je veoma raširen i dostupan te je samim time i olakšano njegovo učenje i korištenje.

2.1.1 Povijest i razvoj HTML-a

Kako bi razvio svoju ideju o jedinstvenim datotekama koje svako računalo može prepoznati, Tim Berners-Lee je 1989. godine razvio hipertekstualni sustav baziran na internetu. 1990. godine je specificirao HTML te je kasnije te godine izradio preglednik i server *softver*. Iste godine nastaje *World Wide Web* kao projekt u CERN-u na kojem su radili Tim Berners-Lee i Robert Cailliau. [2]

Prva javno dostupna verzija HTML-a objavljena je 1991. godine pod nazivom *HTML tags*. Prva verzija HTML-a je sadržavala 18 elemenata te je HTML tada bio veoma jednostavan. 11 od tih elemenata se koriste i u verziji HTML4, a neki elementi se koriste i danas, kao na primjer *tag* <*a*> koji je temeljna ideja HTML-a i WWW-a, a služi za povezivanje hipertekstualnih dokumenata, odnosno kao hiperlink. HTML je baziran na Standard Generalized Markup Language-u, odnosno SGML-u [3]. SGML je definiran ISO 8879 standardom. To je meta-jezik na temelju kojeg su napisana pravila uporabe drugih jezika. Temeljni koncept SGML-a je određivanje vrste teksta oznakama, odnosno *tagovima.* HTML koristi upravo taj koncept uz ideju međusobnog povezivanja hiperteskstualnih dokumenata. Tim Berners-Lee je HTML definirao kao aplikaciju ili primjenu SGML-a.

Osim razvoja jedinstvenog hipertekstualnog jezika, baziranje HTML-a na SGML-u je bila odlična ideja jer je omogućila ljudima da razviju svoj jezik prema uzoru na HTML, prilagođen njihovim potrebama. Iako je to bilo korisno pojedincima, imalo je negativan utjecaj na osnovnu ideju stvaranja HTML-a zbog nerazvijene standardizacije preglednika tog vremena. Kako bi HTML približili korisnicima i njihovim potrebama, Dave Raggett je u suradnji s Timom Berners-Leeijem proširio HTML jezik te stvorio njegovu inačicu *HTML+* s kojom je dovršen *HTML 1.0.* Dave Reggett je pregledavao tiskane medije te je u njima tražio ideje za razvoj nove inačice HTML-a. Razmišljao je kako prilagoditi članke iz tiskanih medija korisnicima web-a i na taj način dodavao nove elemente HTML-u. Budući da su mnogi prepoznali potencijal HTML-a, 1993. je počelo masovno razvijanje internet preglednika, a u to vrijeme najpoznatiji su bili *Arena* i *Mosaic.* Izdavači internet preglednika su željeli doprinijeti razvoju HTML-a pa je većina novih preglednika uvela

svoje novitete vezane uz HTML. HTML 1.0 time postaje nestabilan i nefunkcionalan za svaki internet preglednik. Kao posljedica toga nastaje HTML 2.0 s ciljem uvođenja novih pravila te pokušaja standardizacije HTML-a.

HTML 2.0 razvijen je 1994. Nastao je sakupljanjem svih neslužbeno nastalih oznaka kako bi se zadovoljili svi korisnici i njihove potrebe. Za razvoj nove inačice HTML-a zaslužan je Dan Connolly. On je, također, napisao i svojevrsne upute za HTML 2.0 pod imenom "*Document Type Definition for HTML 2".* Krajem 1994. godine osnovana je *Netscape Communications* korporacija. Netscape izdaje svoj preglednik koji postiže veliki uspjeh. Zbog velikog uspjeha i dobrog plasmana na tržištu, Netscape uvodi svoje HTML oznake neovisno o drugim *web* zajednicama te time počinje upravljati HTML standardom. Kako bi ispravio novu potencijalnu katastrofu za HTML, Tim Berners-Lee iste godine osniva *The World Wide Web Consortium,* ili samo "*W3C",* udruženje čiji je zadatak bio standardizirati HTML kako bi bio u potpunosti bio kompatibilan sa bilo koji preglednikom. [4] Tijekom 1995. godine, masovno dodavanje raznih HTML tagova se nastavilo te se prvi put spominju oznake koje određuju izgled HTML dokumenta, na primjer *bgcolor* kao atribut izvorne oznake <*body*>. Udruženju koje se bavilo razvijanjem HTML-a ovo nije odgovaralo jer je njihova ideja bila izgraditi jezik koji će opisivati kako će sadržaj dokumenta biti organiziran, ali ne i kako će on izgledati.

Dave Raggett je nastavio razvijati HTML te je sve svoje ideje opisao u *Internet Drafts* dokumentu 1995. U novoopisanoj inačici HTML-a, HTML 3.0, Raggett je dodao razne oznake kojima je pokrio nedostatke prijašnjih verzija. Neke od najznačajnijih inovacija u HTML-u 3.0 su bile dodavanje tablica, fusnota i upravljanje obrascima. Također, vrlo bitna stavka HTML-a 3.0 je uvođenje oznaka <*style*> i <*class*> kojima je pružena podrška za uvođenje stilskih jezika. Ova inačica HTML-a je bila sveopće prihvaćena, ali i dalje nije postojala službena standardizacija HTML jezika. 1995. Microsoft izdaje svoj internet preglednik- *Internet Explorer*. Njihov preglednik je bio najveća konkurencija *Netscape-ovom* pregledniku *Navigator*. Time započinje veliko natjecanje između internet preglednika koji se pokušavaju istaknuti dodavanjem novih značajki HTML jezika koje će biti podržane samo od strane preglednika koji ih je uveo. Slični način suzbijanja konkurencije se održao i do danas. U ovo vrijeme se razvijaju i

drugi poznati jezici- *Netscape* uvodi *Javascript, Internet Explorer* uvodi *CSS... World Wide Web Consortium* krajem 1995. osniva *HTML Editorial Review Board*, odnosno odbor koji je trebao pomoći pri standardizaciji HTML-a. U odboru su se nalazili predstavnici najutjecajnijih kompanija svog vremena- *IBM-a, Microsoft-a, Netscape-a, W3C-a...* Zajedničkim dogovorima ukinute su neke oznake poput *Netscape-ovog* <blink> *taga,* ili <marquee> *taga* kojeg je predstavio *Internet Explorer.* Također, neke oznake koje su imale istu funkciju, ali drugačiji naziv, su zamijenjene jednom oznakom za svaki internet preglednik. Primjer takve oznake je oznaka <*object*> koja je zamijenila oznake: *embed, app, applet, dynsrc* koje su imale istu funkciju- ugradnju različitih dodatnih informacija u HTML datoteku. Oznake <*strong*> i <*b*> su obje služile za definiranje podebljanog teksta. Konačnim sporazumom *W3C-a* i predstavnika internet preglednika HTML je standardiziran 1997. Standardizacija HTML-a je značila njegovu jednaku funkcionalnost i uporabu za bilo koji preglednik. Prikupljanjem do sad dogovorenih specifikacija iz verzija *HTML+, HTML 2.0* i *HTML 3.0* stvorena je nova, standardizirana inačica *HTML 3.2* kao konačna i službena verzija *HTML-a 3.0.* U *HTML 3.2* je uključeno dodavanje tablica, indeksa, eksponenata i manjih aplikacija.

Krajem 1997. godine, predstavljen je *HTML 4.0*. Ova inačica je nudila 3 različite verzije: *Strict*- kod koje nije bilo dopušteno koristiti odbačene elemente, *Transitional*- u kojoj je dopušteno korištenje odbačene elemente, i *Frameset*- verziju u kojoj su dopušteni elementi vezani uz izradu okvira, odnosno *frame-ova. HTML 4.0* nastavlja prihvaćati oznake koje su nametnute od strane proizvođača internet preglednika, ali istovremeno čisti standard s ciljem uklanjanja suvišnih oznaka. U prosincu 1999. godine izlazi *HTML 4.01* kao do tad konačna verzija tog jezika s manjim promjenama u specifikaciji standarda.

Nakon što je izdana verzija *HTML 4.01,* novije inačice tog jezika neće biti dugi niz godina, sve do 2014. godine kada je objavljen *HMTL 5.0.* No, unatoč tome što nisu objavljivali novije verzije HTML-a, *W3C* i njihova HTML grupa radili su na novom jeziku- *Extensible Hypertext Markup Language-u,* odnosno *XHTML-u* koji službeno izlazi početkom 2000. godine . *XHTML* je jezik nastao iz HTML-a 4.0, ali za razliku od HTML-a koji je baziran na *SGML-u, XHTML* je baziran na *XML-u. XML* je kratica za *Extensible Markup Language* [5]. To je proširivi označni jezik koji je podskup *SGML* jezika. Razlika

između ta dva jezika je u tome što je *XML* jednostavniji, ali su pravila pisanja ovog jezika stroža što donosi veću mogućnost pogreške. *XHTML* je napravljen s ciljem da HTML postane proširiv te da mu se poveća interoperabilnost sa datotekama različitih formata. XHTML je tako postao poboljšana verzija HTML-a s više mogućnosti.

Krajem 2014. godine predstavljena je nova verzija *HTML-a, HTML 5*. To je prva verzija HTML-a nakon HTML-a 4.01, a nastala je u suradnji *World Wide Web Consortium (W3C)* i *Web Hypertext Application Technology Working Group (WHATWG)* grupa. *HTML 5* preuzima značajke *HTML-a 4.01,* ali i *XHTML-a 1* s ciljem da se HTML jezik proširi i postane podržan za sve oblike multimedije te istovremeno lagan za čitanje i razumijevanje ljudima i internet preglednicima. Ova verzija HTML-a se smatra "živim standardom" jer je zamišljena za stalnu nadogradnju, bez oznake verzije specifikacija. *HTML 5* donosi mnoge mogućnosti koje nisu bile podržane od strane *HTML-a 4.01* i *XHTML-a 1*, na primjer uvođenje videa na stranice bez korištenja *Adobe Flasha* ili *Microsoftovog Silverlight-a,* upravljanje pomoću tipkovnice, "*drag and drop", canvas* i brojne nove elemente. Sintaksa *HTML-a 5* više nije temeljena na *SGML-u.* Unatoč tome, *HTML 5* je osmišljen tako da bude kompatibilan sa starijim verzijama *HTML-a,* odnosno da ih može prepoznati. Sam kôd započinje uvodnom linijom koja sadržava deklaraciju *<!DOCTYPE html>* koja se u prijašnjim verzijama temeljila na *SGML-ovoj* deklaraciji *Document Type Declaration – DTD* te je bila dulja i kompliciranija. Deklaracija internetskom pregledniku omogućava prepoznavanje verzije *HTML-a* kako bi sadržaj stranice bio ispravno prikazan. *HTML 5* je jezik koji se može koristiti za višeplatformske mobilne aplikacije jer sadrži značajke namijenjene uređajima s manjim kapacitetom napajanja.

U sintaksu *HTML-a 5* je uvedeno mnogo novih oznaka. Kako bi se proširilo područje prikaza grafičkog i multimedijskog sadržaja, uvedeni su elementi *<video>, <audio> i <canvas>* kao i podrška *Scalable Vector Graphics (SVG)* vektorskim grafikama te *MathML* za matematiče formule. Uvedeni su i elementi kojim se definira struktura buduće stranice, a neki od njih su: *<main>, <section>, <article>, <header> i <footer>, <nav> i <figure>*. Predstavljeni su novi atributi, a neki elementi sa svojim atributima su izbačeni. Neki od postojećih elemenata su promijenjeni ili standardizirani.

Zbog nekih značajki, *HMTL 5* se često uspoređuje s *Adobe Flash-om*, ali te dvije tehnologije se bitno razlikuju. Iako obje omogućuju dodavanje video i audio datoteka te *SVG* web stranicama, razlika je u tome što *HTML 5* sam ne pruža mogućnost interaktivnosti bez dopune *CSS-om* ili *JavaScript-om.* Danas se *Adobe Flash* sve manje koristi, a *Adobe* je izdao alat kojim se *Flash* konvertira u *HTML 5.*

2.1.2 Struktura HTML-a

HTML kôd se sastoji od slova ili riječi koje se nalaze unutar izlomljenih zagrada, odnosno oznaka ili *tagova* [6]. Izlomljene zagrade od kojih se oznake sastoje često se shvaćaju kao i simboli "manje od" < i "veće od" >. Sve što se nalazi u oznakama, kao i sadržaj između njih, naziva se element. Elementi najčešće imaju dvije oznake- otvarajuću i zatvarajuću, iako postoje elementi koji se sastoje samo od početne, odnosno otvarajuće oznake. Zatvarajuća oznaka se od otvarajuće razlikuje po tome što u njoj ispred elementa dolazi kosa crta "*/*". U *HTML-u* 5 postoje oznake kod kojih nije obavezno njihovo zatvaranje jer se podrazumijeva kako nova oznaka neće funkcionirati ukoliko se prijašnja ne smatra zatvorenom. Neke od takvih oznaka su: *html, head, body, p...* Kôd koji u sebi sadrži zatvarajuće oznake tamo gdje su one nepotrebne će biti mnogo pregledniji te će se eventualne greške lakše naći. Također, postoje i tzv. "samozatvarajući" HTML elementi. Oni se ne zatvaraju zatvarajućom oznakom. Primjer takve oznake je *meta tag* koji se može nalaziti samo unutar otvarajuće zagrade: *<meta charset="UTF-8">*, ili u svojoj oznaci može sadržavati i kosu crtu ispred koje je, prema preporuci *W3C-a,* potrebno staviti razmak: *<meta charset="UTF-8" />*. Sadržaj koji se nalazi između oznaka je onaj koji će se prikazivati u internet pregledniku. Oznake pregledniku daju podatak o tome na koji način prikazati sadržaj između njih. *Slika 1* prikazuje *HTML* element.

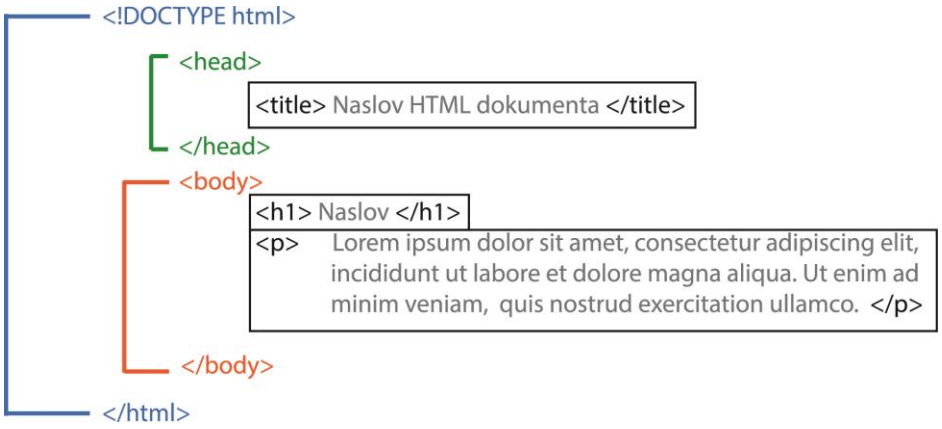
Slika 1. HTML element

Sadržaju HTML elementa moguće je dodijeliti dodatne informacije. Dodatne informacije HTML elementa nazivaju se atributi. Svaki element može imati atribute, ali svaki element ne može imati bilo koji atribut. Također, neki elementi nemaju smisla ukoliko im se ne dodaje atribut. Oni se pišu u otvarajućoj oznaci te se sastoje od dva dijela- naziva i vrijednosti. Opći oblik pisanja atributa je naziv=”vrijednost”, odnosno *name=”value”*. Naziv atributa upućuje na vrstu informacije koja se pridodaje HTML elementu, a vrijednost je informacija dana atributu. Iako HTML 5 pruža mogućnost korištenja velikih slova u pisanju naziva atributa, kao i izostavljanje navodnika, atributi se u kôdu pišu malim slovima te se nakon znaka jednakosti vrijednost smješta u navodne znakove. Neki od vrlo čestih atributa koji se koriste su: *lang* za definiciju jezika HTML dokumenta ili elementa, *class* za specificiranje jedne ili više klasa određenih stilskim jezikom, *id* za određivanje jedinstvenog identifikatora određenog stilskim jezikom, *style* za dodavanje *CSS* stila elementu i tako dalje [7]. *Slika 2* prikazuje primjer atributa *id* i njegove vrijednosti, odnosno imena tekst1 koji su dodani elementu *p*.

Slika 2. atributi HTML elementa

U osnovnim HTML kôdovima nalazimo nekoliko glavnih elemenata. Svaki HTML dokument započinje *html* oznakom koji upućuje na to da je sve između otvarajuće *<html>* i zatvarajuće *</html>* oznake HTML kôd. Kako bi se definirala točna inačica standarda koji se koristi, na samom početku HTML dokumenta postavlja se *<!DOCTYPE html>* oznaka za, u ovom slučaju, HTML 5. Nakon *html* elementa, dolazi element *head.* Ono što se nalazi između otvarajuće oznake “*<head>”* i zatvarajuće oznake “*</head>”* neće biti prikazano na samoj stranici, ali će davati informacije o stranici. Najčešći element koji se nalazi između oznaka *head* je element *title* u kojem stoji naslov HTML dokumenta koji se

prikazuje u pregledniku. Također, u elementu *head* je često smješten i *meta tag.* Sam HTML je skup podataka, a u *meta* oznaci dodaju se podaci o tim podacima, odnosno metapodatke *(metadata).* Kroz *meta tag* moguće je definirati ključne riječi stranice prema kojima će ju biti moguće pronaći putem pretraživača, opis stranice, autora, automatsko osvježavanje stranice te kodiranje znakova kako bi HTML dokument prikazivao posebne znakove. U element *head* moguće je dodati i stilove elemenata, skripte kao i povezivanje dokumenta s vanjskom datotekom putem oznake *link.* Oznaka *link* se kao i oznaka *meta* ne zatvara. Nakon elementa *head,* dolazi element *body* koji se sastoji od otvarajuće <*body*> i zatvarajuće oznake <*/body*>. Svi elementi koji se nalaze između ove dvije oznake su oni koji će se prikazivati u glavnom prozoru internet preglednika. U elementu *body* se nalazi sav sadržaj HTML dokumenta- naslovi, tekstovi, slike itd. Nakon zatvaranja *body* elementa, kako bi se označio kraj prikazivanja sadržaja, zatvara se i *html* element čime se upućuje na kraj HTML kôda, kao što je prikazano na *slici 3.*



Slika 3. Jednostavni HTML kôd

Na *slici 3* se nalazi jednostavni HTML kôd koji se sastoji od svojih osnovnih elemenata: *html, head i body* te sadržajnih elemenata: *h1* i *p.* Vidljivo je kako se neki elementi sastoje samo od oznaka i sadržaja HTML dokumenta, a neki u sebi sadrže i nove elemente. Onaj element koji u sebi sadrži novi element naziva se "*roditelj" (parent),* a svi koji se nalaze unutar nekog elementa nazivaju se "*djeca*" *(children).* Oni zajedno grade obiteljsko stablo Ukoliko element koji je roditelj ima zadane atribute, ti atributi će se

prenositi na svaki element koji se nalazi u njemu. Iznimka je *child* element koji ima zadane svoje atribute, a ova pojava je od velike važnosti kod dodavanja stilova. Osim navedenih osnovnih elemenata, postoji veliki broj elemenata koji mogu utjecati na strukturu HTML dokumenta.

Kod označavanja teksta razlikujemo strukturalne i semantičke elemente. U strukturalnom označavanju teksta postoje dvije glavne vrste elemenata- naslov i paragraf. Naslovi, odnosno *headings* se označavaju u šest razina, oznakama od *<h1>* do *<h6>*. *<h1>* se koristi za glavne naslove, dok se svaka iduća razina koristi za podnaslove. Internet preglednik svaku razinu u naslovu prikazuje u drugačijoj veličini. Iako veličina teksta u naslovu i podnaslovima može varirati od preglednika do preglednika, temeljna ideja njihovog postupnog smanjivanja ostaje ista. Tako će naslov *<h1>* biti najveći, a *<h6>* najmanji. Točne veličine naslova mogu biti definirane stilskim jezikom. Za kreiranje paragrafa, tekstualni sadržaj koji se želi prikazati kao paragraf se umeće između oznaka * * i * *. Svaki zasebni paragraf će u pregledniku biti prikazan u novom redu te će biti odvojen od prijašnjeg. Paragraf i naslov su *block* elementi što znači da se uvijek prikazuju u novom redu te zauzimaju maksimalnu širinu koja im je dostupna, ukoliko drugačija nije zadana. Osim *block* elemenata, postoje i *inline* elementi koji ne započinju u novom redu, a zauzimaju onu širinu koja im je potrebna.

HTML-om je moguće izvršiti vizualna isticanja u tekstu. Ukoliko se željeni tekst umetne između oznaka * i ,* tekst postaje *bold*, odnosno podebljan. Osim vizualnog isticanja, tekst između tih oznaka nema dodatno značenje. Tekst koji se nalazi između oznaka ** i ** će se prikazivati kao *italic,* odnosno ukošeno. Brojke, slova ili dio teksta koji se želi prikazati kao potencija ubacuje se u oznake *^{* i *}*, a ukoliko se žele prikazati kao indeks, koriste se oznake *_{* i *}*.

Ukoliko u kôdu postoji više razmaka, internet preglednici će ih prikazivati kao samo jedan razmak. Također, prelazak u novi red prikazuju kao razmak. Ta pojava se naziva *white space collapsing.* Autori kôdova iskorištavaju tu pojavu ostavljajući praznine u kôdovima kako bi oni bili što pregledniji. U slučaju da autor ne želi da preglednik sam odlučuje o prikazu razmaka, odnosno bjelina, tada se željeni tekst smješta u oznake *<pre>* i

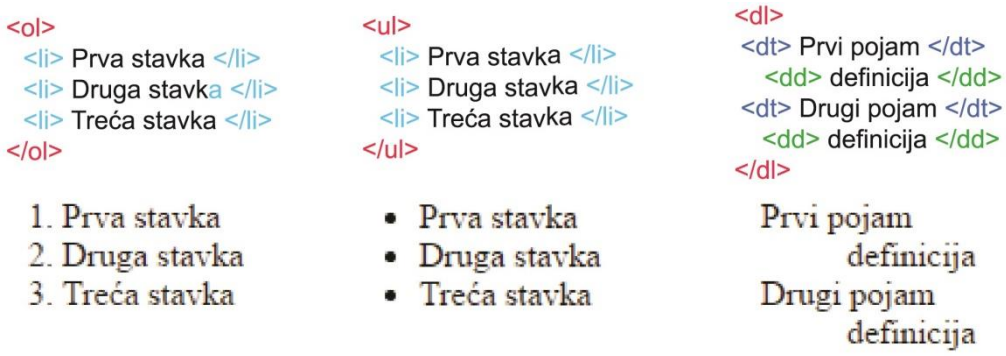
</pre>. Takav tekst će biti prikazan onako kako ga je autor pisao u kôdu, uključujući svaki razmak i prijelaz u novi red, odnosno bjelinu.

Tekst u postojećem paragrafu koji se želi prikazati u novom redu odvaja se elementom *
,* odnosno *break.* Ovaj element se sastoji samo od otvarajuće oznake i nema atribute, a može se pisati i *
*. Ukoliko prijelaz u novi red označava i promjenu teme, ili ako se javlja potreba za vizualnim odjeljenjem, sadržaj je moguće odvojiti horizontalnom linijom koristeći *<hr>,* odnosno *<hr />* oznaku. *<hr>* i *
* su prazni elementi.

Semantički elementi su oni koji nisu zaduženi za strukturu stranice, već dodaju dodatne informacije stranicama. Internet preglednici ih prikazuju na različite načine. Na primjer, element ** naznačuje naglasak pojedinih dijelova teksta. On se u preglednicima prikazuje kao ukošen tekst. Element ** upućuje na važnost sadržaja u elementu, odnosno taj sadržaj mora biti jako naglašen. Preglednici sadržaj između oznaka ** i ** prikazuju kao podebljan tekst, odnosno *bold.* Za označavanje citata postoje dva elementa: *<blockquote>* i *<q>*. U HTML-u 4.01, *<blockquote>* element se koristi za dulje citate koji zauzimaju većinu stranice. U HTML-u 5 ovaj element naglašava citat preuzet s neke druge stranice uz pomoć atributa *cite* čija je vrijednost URL adresa kao izvor citata. Između *<blockquote>* i *</blockquote>* oznaka može nalaziti element *<p>* u kojem se nalazi sadržaj koji se prikazuje kao citat. Tekst prikazan kao *blockquote* citat će biti uvućen od ostatka teksta. Za kraće citate koji su smješteni unutar paragrafa koristi se element *<q>*. Preglednici sadržaju elementa *<q>* dodaju početne i završne navodne znakove, osim *Internet Explorer-a* koji ne dodaje navodne znakove te je zbog toga uporaba ovog elementa umanjena. Također, element *<q>,* kao i element *<blockquote>* koristi atribut *cite*. Osim navedenih semantičkih elemenata, postoji ih još nekoliko: *<abbr>* za objašnjenja kratica, *<cite>* za navođenje autorskih djela, *<dfn>* za definicije novih pojmova itd. Uloga semantičkih elemenata nije promijeniti izgled sadržaju, već preciznije opisati sadržaj *web* stranica. Oni se koriste kako bi drugi programi, kao na primjer razni čitači zaslona ili pretraživači koristili dodatne informacije o sadržaju. Tako će, na primjer, glas čitača zaslona dodati naglasak tamo gdje je to zadano elementom *,* ili će pretraživač prepoznati da stranica sadrži citate koji su zadani elementom *<blockquote>*. Za

promjenu sadržaja stranice koriste se elementi: <*ins*> za novi sadržaj koji je umetnut, <*del*> za sadržaj koji je izbrisan i <*s*> za sadržaj koji više nije točan ili bitan, ali ne bi trebao bti izbrisan. Elementi <*del*> i <*s*> se prikazuju sa crtom posred sadržaja, a element <*ins*> se prikazuje podcrtano.

Ukoliko postoji potreba za listama na *web* stranici, HTML pruža mogućnost korištenja tri vrste lista: uređene liste, neuređene i liste za definicije koje su prikazane na *slici 4*. Uređene liste su one kod kojih je svaka stavka liste označena brojem ili slovom, odnosno sadržaj liste je nabrojen po redoslijedu. Uređena lista se označava oznakom <*ol*> *(ordered list).* Kod neuređene liste ne postoji redoslijed po kojem su stavke navedene, već su one navedene po točkama. Neuređena lista se označava oznakom <*ul*> *(unordered list).* Svaka stavka u listama se označava oznakama <*li*> i </*li*> *(list item).* Liste za definicije su sastavljene od niza pojmova sa pripadajućim definicijama za svaki pojam. One su složenije i sastoje se od tri elementa. Element <*dl*> označava listu za definicije. Unutar tog elementa nalazi se element <*dt*> koji označava pojam čija će definicija biti napisana unutar elementa <*dd*> koji se nalazi u njemu. Jedan pojam može imati više definicija. Liste mogu sadržavati i podliste, odnosno stavka neke liste može biti nova lista.



Slika 4 . Uređena i neuređena lista te lista za definicije

Linkovi su jedna od temeljnih značajki *web-a* jer omogućuju kretanje s jedne stranice na drugu, što je zapravo i sama ideja korištenja *web-a.* Postoji nekoliko vrsta linkova, a to su: linkovi koji povezuju više internet stranica, linkovi koji povezuju nekoliko

stranice na istoj internet stranici, linkovi koji povezuju dijelove iste stranice, linkovi koji se otvaraju u novom prozoru internet preglednika i linkovi koji otvaraju program za slanje elektroničke pošte. Linkovi su označeni elementom *<a>*, odnosno s njegovim pripadajućim otvarajućim i zatvarajućim oznakama. Odredište koje se želi povezati linkom označava se uz pomoć atributa *href* u otvarajućoj oznaci. Sadržaj između oznaka elementa *<a>* je tekst linka. Tekst linka trebao bi uputiti korisnika na to gdje link vodi ukoliko klikne na njega. Prema zadanom, internet preglednici linkove prikazuju kao plavi podcrtani tekst. Ukoliko link vodi na drugu internet stranicu, vrijednost atributa *href* će biti adresa te internet stranice, odnosno njen apsolutni *URL*. Apsolutni *URL* počinje sa imenom domene te internet stranice (*site)*, a može se nastaviti do određene stranice *(page)*. Ako link vodi na druge stranice iste internet stranice, nije potrebno navesti domenu u *URL-u,* nego se koristi relativni *URL*. *Slika 5* prikazuje element *<a>*.

Slika 5. Link koji vodi na stranicu Grafičkog fakulteta

Element *<a>* se koristi i kod kreiranja linkova koji otvaraju program za elektroničku poštu i automatski unosi primatelja. Ovakva vrsta linkova se od do sad nabrojenih razlikuje po vrijednosti atributa *href.* Vrijednost atributa započinje sa "*mailto:",* a u nastavku se navodi *mail* adresa primatelja. Ovakav link se prikazuje isto kao i već navedeni linkovi. Linkovi koji se otvaraju u novom prozoru internet preglednika imaju još jedan atribut. Atribut *target* sa vrijednosti *"_blank"* otvara link u novom prozoru. Linkovi za povezivanje dijelova iste stranice se obično nalaze na vrhu stranice te omogućuju brzo snalaženje po stranici. Prije nego što se određeni element na stranici može povezati linkom, potrebno mu je dodijeliti *id* atribut čija je vrijednost naziv koji će se koristiti u linku. Vrijednost *href* atributa u linku započinje ljestvama *"#"* , a u nastavku sadrži naziv koji je vrijednost *id* atributa onog elementa koji se povezuje linkom.

Postoje informacije koje je najbolje prikazati u tablicama. Kada se informacije prezentiraju u obliku tablice, potrebno je razmišljati na način da se za svaku informaciju moraju točno kreirati red i stupac. Svaki blok u tablici se naziva ćelija, odnosno *table cell.* U HTML-u, tablice se definiraju red po red. Tablice se izrađuju uz pomoć elementa *<table>*. Između otvarajuće i zatvarajuće oznake ovog elementa, dolazi element *<tr> (table row)* koji upućuje na početak novog reda u tablici. Između otvarajuće *<tr>* i zatvarajuće *</tr>* oznake, ubacuje se element *<td> (table data)* za definiranje svake pojedine ćelije (stupce) u redu. Element *<th> (table heading)* se koristi na isti način kao i *<td>*, ali njegova svrha je dodavanje naslova redovima ili stupcima. Internet preglednici obično naslove zadane *<th>* elementom prikazuju kao podebljan tekst, odnosno *bold.* Ukoliko postoji potreba da se neki podatak u tablici smjesti u više stupaca, elementima *<td>* i *<th>* se dodaje atribut *colspan.* Vrijednost atributa *colspan* je broj koji određuje koliko će stupaca biti spojeno u jedan stupac. Ako se neki podatak u ćeliji treba smjestiti u više redaka, oni se također mogu spajati. Redovi se spajaju tako da se elementima *<td>* i *<th>* dodaje atribut *rowspan,* a njegova vrijednost je broj redaka kroz koji se podatak prostire. Kod dugačkih tablica koriste se elementi: *<thead>* za prvi redak tablice u kojem se nalaze naslovi, *<tbody>* za glavni sadržaj tablice i *<tfoot>* za podnožje, odnosno zadnji redak. U starijim verzijama HTML-a, postoje atributi kojima se može definirati širina ćelije, razmak između ćelija, debljina ruba, pozadinska boja itd., ali se ti atributi više ne koriste te se izgled tablice definira uz pomoć *CSS-a. Slika 6* prikazuje tablicu i njen kôd. Zbog boljeg predočenja, izgled tablice definiran je *CSS-om.*

```
<table>
    <tr>   <th> Prvi stupac </th>
           <th> Drugi stupac </th>
    </tr>
    <tr>   <td> A </td>
           <td> B </td>
    </tr>
    <tr>   <td> A </td>
           <td> B </td>
    </tr>
</table>
```

Prvi stupac	Drugi stupac
A	B
A	B

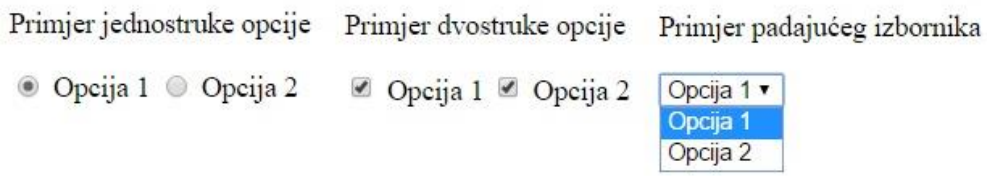
Slika 6. Kôd i pripadajuća tablica

Forme, odnosno obrasci služe za prikupljanje informacija od posjetitelja neke stranice. Postoje razni elementi kojima se na razne načine mogu prikupiti tražene informacije- unosom teksta, mogućnosti odabira ili slanjem postojećih informacija. Svi ti elementi su smješteni unutar elementa <*form*>. Element <*form*> uz sebe uvijek ima atribut *action,* a obično uz taj dolazi i *method*. Vrijednost *action* atributa je *URL* stranice na serveru koji će primati informacije. *Method* služi za slanje formi koje može biti obavljeno na dva načina: *get* ili *post.* Element <*input*> se koristi za nekoliko tipova prikupljanja informacija. Kako bi server znao raspoznati informacije prikupljene u formama, svaki element ima atribut *name.* Na taj način server može razaznati što je od unesenog teksta korisničko ime, a što lozinka. Atribut *type* će, ovisno o vrijednosti koja mu je dodijeljena, odrediti o kojoj se vrsti unosa radi. Vrijednost *text* omogućuje unos jedne linije teksta u prazno polje. Vrijednost *password* kreira polje za jednu liniju teksta, ali simboli koji se unose u to bolje su sakriveni s ciljem očuvanja tajnosti korisničke lozinke. Ukoliko postoji potreba za ograničenjem broja unesenih simbola, dodaje se atribut *maxlength*. Vrijednost ovog atributa je maksimalni broj simbola koji korisnik može unijeti. Element <*textarea*> omogućuje stvaranje polja za više linija teksta. Atributi *cols* i *rows* definiraju veličinu polja na način da je njihova vrijednost broj redova i stupaca teksta koji će biti vidljivi. *Slika 7* prikazuje primjere unosa teksta u forme.

Slika 7. Tekst, lozinka i textarea element

Osim unosom teksta, informacije je moguće pružiti i odabirom jednog od ponuđenih odgovora. Sve mogućnosti, odnosno odgovori na neko pitanje moraju imati isti *name* atribut, a server će njihov odabir raspoznati prema atributu *value* čija je vrijednost odabrana opcija. Ukoliko vrijednost atributa *type* elementa *input* bude *radio,* korisnik može izabrati samo jednu opciju kao odgovor. Vrijednost *checkbox* omogućuje odabir višestrukog odgovora. Ukoliko neka od ponuđenih opcija treba biti postavljena kao označena, toj opciji se dodaje atribut *checked* s vrijednošću *checked.* Izbor opcija se može prezentirati i

padajućim izbornikom elementom <*select*>. Element <*select*> sadrži elemente <*option*>. Zadana opcija može biti postavljena atributom *selected* s vrijednosti *selected.* Atributom *multiple* s vrijednosti *multiple* dozvoljava se mogućnost višestrukih odgovora. Na *slici 8* se nalaze primjeri *radio* i *checkbox inputa* te *select* element.



Slika 8. radio i checkbox input te select element

Ukoliko se korisnicima želi omogućiti *upload* datoteka, elementu *input* se dodaje atribut *type* s vrijednosti *file*. Ovakvim *input* elementom se dobiva prazno polje slično polju za unost teksta i pripadajući gumb. Klikom na gumb korisnik dobiva mogućnost odabira željene datoteke. Kako bi se potvrdio odabir datoteke, ili potvrdila forma, potrebno je dodati *submit button.* Element *input* sa atributom *type* čija je vrijednost *submit* tvore *submit button.* Element <*button*> omogućuje kreiranje gumba. Za naznačavanje pojedinih dijelova forme koristi se element <*label*>. Dijelovi forme se mogu grupirati pomoću <*fieldset*> elementa. <*fieldset*> element može u sebi sadržavati element <*legend*>, a služi za postavljanje natpisa pojedinog grupiranog dijela forme.

HTML 5 uvodi neke nove značajki kod izradi forma. *Form validation,* odnosno potvrđivanje služi kako bi korisnik pronašao izostavljeno obavezno polje u formi. Obaveznom polju se dodaje atribut *required* s vrijednosti *required.* Unos datuma je također predstavljen HTML-om 5. Elementu <*input*> se dodaje atribut *type* i vrijednost *date.* Još neke od novih značajki su: polja za unos *email* adresa, polja za unos *URL* adrese i polje za pretraživanje riječi. Također, za bilo koji unos teksta uveden je atribut *placeholder* čija je vrijednost onaj tekst koji će biti prikazan prije samog unosa teksta.

Osim osnovnih elemenata zaduženih za strukturu HTML dokumenta, HTML pruža mogućnost dodavanja niza elemenata i atributa koji olakšavaju pisanje samog kôda. Za lakše snalaženje u kôdu, moguće je dodati komentar. Komentar je tekst koji neće biti

prikazan u pregledniku, nego isključivo u kôdu. Za pisanje komentara, željeni tekst je potrebno smjestiti između <*!*-- i --> znakova. Već spomenuti atributi *id* i *class* služe za lakše manipuliranje sadržaja označenog tim atributima- dodavanjem stilova ili skripta. Ovi atributi mogu biti dodani bilo kojem elementu, a njihova vrijednost započinje slovom. Bitno je da samo jedan element može imati određenu vrijednost *id* atributa, dok više elemenata može nositi istu vrijednost *class* atributa. <*div*> element služi za grupiranje sadržaja u blokove. Ovaj element može biti prazan s dodanim stilovima ili može nositi određeni sadržaj. Svi stilovi dodani <*div*> elementu se odnose na cijeli skup elemenata u njemu. Element <*span*> također služi za grupiranje sadržaja, ali za razliku od <*div*> elementa, on djeluje *inline.* <*span*> elementom je moguće označiti dio teksta s ciljem dodavanja posebnog stila tom dijelu teksta, ukoliko označavanje nekim drugim elementom nije moguće. Element <*iframe*>, odnosno *inline frame* omogućuje ugrađivanje prozora u kojem je moguće prikazivati drugu stranicu, odnosno dokument putem njegove *URL* adrese. Atributom *src* se definira izvor, odnosno *URL* dokumenta, a atributima *width* i *height* se definira veličina prozora.

HTML dokumentu je moguće dodati multimedijski sadržaj- sliku, video i zvuk. Sliku je moguće dodati koristeći <*img*> element. Atribut *src* pregledniku govori gdje pronaći sliku, a njegova vrijednost je *URL* adresa slike koja se dodaje. Ukoliko slika nije vidljiva, moguće je dodati opis koji će se prikazivati u tom slučaju. Opis se dodaje atributom *alt* čija je vrijednost tekst opisa. Prijelazom miša preko slike prikazuje se naslov slike *(tooltip).* Također <*img*> element može imati atribute *height* i *width* za definiranje veličine slike, iako je ulogu određivanja veličine preuzeo *CSS.* Slika koje se dodaje mora slijediti tri pravila: biti pravog formata (*.jpg, .gif, .png* ili *.webP)*, odgovarajuće veličine i rezolucije *(72 ppi)*. HTML 5 uvodi elemente <*figure*> i <*figcaption*> kojima je moguće dodavati više slika u grupu kojoj je dodan opis. Opis se nalazi ispod slika, a tekst opisa se nalazi između otvarajuće i zatvarajuće oznake <*figcaption*> elementa.

Elementom <*video*> moguće je dodati video zapise. Najkorišteniji video formati su: *.mp4, .ogv* i *.webM.* Atribut *src* definira putanju do videa, odnosno njegovu adresu. Atributom *poster* je moguće dodati sliku koja će se prikazivati prije pokretanja videa. Veličina videa se može zadati atributima *width* i *height*. *Autoplay* atribut zadaje automatsko

pokretanje videa. *Loop* atributom se zadaje ponovno pokretanje videa nakon što je završio. Element <*source*> se dodaje unutar elementa <*video*>, a mijenja atribut *src* te se koristi ukoliko postoji više formata istog videa. Atribut *src* se dodaje svakom <*source*> elementu, a atributom *type* se zadaje format videa. Atributom *preload* moguće je zadati pregledniku što će napraviti kada se stranica učita.

HTML 5 je, uz <*video*> element predstavio i <*audio*> element za dodavanje glazbe ili zvučnih zapisa. Najpoznatiji formati zvučnih zapisa za web su: *.mp3* i *.wav*. Atribut *src* prgledniku govori gdje se nalazi zapis. Kao i <*video*> elementu, moguće je dodati atribut *controls* kojim se prikazuje gumbi za upravljanje. Atribut *controls* nema svoju vrijednost i nije ju potrebno unositi. Sama njegova prisutnost u kôdu zadaje pojavljivanje gumbi za upravljanje. Također, moguće je dodati atribute *autoplay* za automatsko pokretanje i *loop* za ponovno pokretanje, kao i *preload* koji pregledniku govori što činiti kada učita stranicu. Kako i kod <*video*> elementa, preglednicima je moguće ponuditi nekoliko formata zvučnog zapisa uz pomoć elementa <*source*>.

2.2 CSS

Cascading Style Sheets, odnosno *CSS* je stilski jezik koji služi za opis prezentacije dokumenata napisanih pomoću označnog, odnosno *markup* jezika. *CSS* definira kako prikazati *HTML* elemente, njihov raspored, boje, font itd. Iako je *HTML* najrašireniji označni jezik, *CSS* se može koristiti i kod opisa prezentacije drugih označnih jezika, npr: *XHTML, XML* ili *SVG*. Uz *HTML* i *JavaScript, CSS* je jedna od temeljnih tehnologija koju koriste mnoge internet stranice.

CSS je primarno osmišljen kako bi se odvojio pojam sadržaja dokumenta od njegove prezentacije. Ovakvim odvajanjem pojednostavljeno je upravljanje izgledom stranice, smanjena je kompleksnost i ponavljanje s ciljem ostvarenja zamišljenog izgleda stranice. Osim toga, ovim jezikom je vraćena i prvobitna ideja *HTML-a*, koji je bio osmišljen da definira strukturu stranice, a ne i njen izgled pa je tako smanjena potreba osmišljanja neslužbenih elemenata.

Također, velika prednost je uvođenje odvojenog *.css* dokumenta koji definira izgled svih stranica jedne internet stranice čime se ubrzao proces oblikovanja jer više nije bilo

potrebno raditi na svakoj zasebnoj stranici. *CSS-om* je omogućeno vizualno oblikovanje istog dokumenta napisanog označnim jezikom različitim stilovima koji će odgovarati načinu upotrebe dokumenta- za računalni prikaz, u tisku, za čitače zaslona, za uređaje koji koriste Brailleovo pismo. Također, koristi se za prilagodbu prikaza stranice ovisno o veličini zaslona na kojem se prikazuje.

Sintaksa *CSS-a* je jednostavna, a sastavljena je od engleskih riječi kojima su određena imena različitih stilskih svojstava. *CSS-om* je moguće na više načina odrediti stil jednog elementa. Zbog toga su određena pravila kojima je ustanovljena težina pojedinog načina određivanja stila. Ukoliko je stil elementa određen više puta, a način određivanja stila ima istu težinu svaki put, koristi se onaj stil koji je određen tako što je u kôdu napisan zadnji. Svaki element koji je smješten u neki element, odnosno *child* element preuzima stil elementa u kojem se nalazi, odnosno *parent* element, ukoliko nije posebno definiran stil *child* elementa.

CSS se unutar *HTML* dokumenta može pisati na dva načina. Prvi način je pisanje stilova u zaglavlju *HTML* dokumenta, odnosno unutar elementa <*head*> u koji se ubacuje element <*style*>. Drugi način je unutar *HTML* oznaka, odnosno *inline* kao atribut *style.* Također, *CSS* može biti zaseban dokument u kojem su definirani stilovi elemenata, a on se povezuje s *HTML* dokumentom uz pomoć elementa <*link*>. *Slika9* prikazuje tri različita načina dodavanja stilova.

```
...
<style>
p {color:red;}
</style>
...
```

```
<p style="color:red;"> Paragraf </p>
```

```
<link rel="stylesheet" href="stil.css" type="text/css">
```

Slika 9. tri načina dodavanja CSS-a

Zbog razvijanja *CSS-a* i internet preglednika, neki preglednici ne podržavaju sva svojstva koje nudi *CSS.* Kako bi neki preglednik prepoznao takva svojstva, svojstvima se dodaju određeni prefiksi za određene preglednike. Prefiksi su potrebni kod izrade npr.

gradijenata, transformacija, tranzicija i animacija koje će biti objašnjene u daljnjem tekstu. *Tablica 1* prikazuje listu najpoznatijih preglednika i prefikse koje oni koriste.

Tablica 1. popis najpoznatijih preglednika i pripadajućih prefiksa.

Browser	Prefiks
Chrome	*-webkit-*
Firefox	*-moz-*
Internet Explorer	*-ms-*
Opera	*-o-*
Safari	*-webkit-*

2.2.1 Povijest i razvoj CSS-a

CSS prvi spominje *Håkon Wium Lie* krajem 1994. godine koji je surađivao s *Timom Berners-Leeijem.* U to vrijeme je predloženo nekoliko stilskih jezika, ali pregovorima u *W3C-u* je donesena odluka o izdavanju prve predložene verzije *CSS-a, CSS1* koji je izdan u prosincu 1996. godine. *Bert Bos* je svojim prijedlozima imao značajan doprinos razvoju *CSS-a 1* pa se tako smatra i njegovim suautorom te jednim od osnivača *CSS-a*. *CSS* je prvotno nazvan *Cascading HTML Style Sheets,* ali je *HTML* ubrzo izbačen iz naziva jer je *CSS* pružao mogućnost oblikovanja i drugih označnih jezika.

Stilski jezici su se pojavljivali u raznim formama još od samog početka razvijanja *SGML-a*, a *CSS* je razvijen kao stilski jezik koji bi zadovoljio potrebe oblikovanja web-a. Jedan od zahtjeva stilskih jezika za web je bio taj da napisani stilovi dolaze iz različitih izvora na web-u. To nije bilo moguće s dosadašnjim stilskim jezicima kao što su *DSSSL (Document Style Semantics and Specification Language)* ili *FOSI (Formatting Output Specification Instance)* koji su bili namijenjeni oblikovanju *SGML-a.*

1997. godine osnovana je grupa *CSS Working Group* koju je predvodio *Chris Lilley* kao predstavnik *W3C-a.* Ova grupa se bavila rješavanjem problema koji nisu bili ispitani u prvom izdanju *CSS-a.* Njihov rad rezultirao je izdavanjem nove verzije *CSS-a, CSS2. CSS2* je izdan krajem 1997. godine, a službenom preporukom *W3C-a* je izdan 1998. godine. Iste

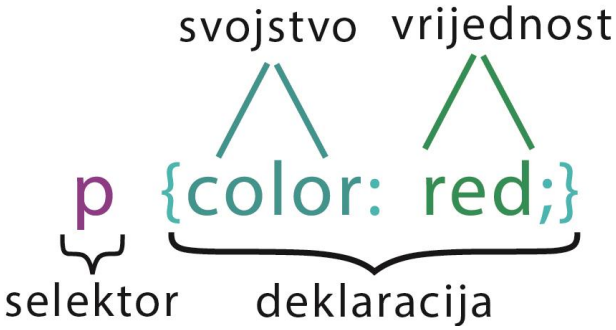
godine započinje rad na novoj verziji *CSS-a, CSS3.* Ova verzija se koristi i danas te se i dalje razvija.

CSS1, odnosno prva verzija *CSS-a,* pružao je nekoliko mogućnosti: svojstva upravljanja fontovima kao što su *typeface* i *emphasis,* promjena boje teksta, pozadina i drugih elemenata, poravnanje teksta, slika, tablica. Osim toga, sadržavao je opcije poput pozicioniranja za većinu elemenata, upravljanje praznim prostorom uz pomoć margina i *padding-a* te oblikovanje rubova (*border).* Također, u ovoj verziji *CSS-a* je bila predstavljena mogućnost dodavanja identifikatora za jedinstvene elemente, kao i klasifikacija grupa elemenata. *CSS2* uvodi nove opcije: *relative, absolute* i *fixed* pozicioniranje, *z-index* za upravljanje rasporeda elemenata, *media types* koja pruža mogućnost prilagodbe sadržaja prema tipu medija, nove opcije za upravljanje tekstovima kao što je dodavanje sjena tekstu i tako dalje. Za razliku od svojih prethodnika koji su predstavljeni kao velika zasebna specifikacija značajki, *CSS3* je podijeljen u nekoliko zasebnih dokumenata, odnosno modula. Svaki od modula dodaje nove mogućnosti ili proširuje i unaprjeđuje značajke predstavljene u *CSS-u 2*. Postoji preko pedeset modula, ali zbog nestabilnosti tek četiri modula imaju službenu preporuku, a to su: *media queries, namespaces, selectors3* i *color.* Iako ne postoji *CSS4,* neki od modula predstavljeni kao *CSS3* su dostigli svoju četvrtu razinu.

2.2.2 Struktura *CSS-a*

CSS je skup pravila koja se odnose na *HTML* elemente. Ta pravila određuju kako će oni biti prikazani. Pravila *CSS-a* se sastoje od dva dijela- selektora i deklaracije. Selektor je dio pravila koji upućuje na koji element se pravilo odnosi. Postoji više vrsta selektora, a ukoliko se neko pravilo odnosi na više elemenata, onda se nazivi elemenata odvajaju zarezom. Drugi dio pravila, koji se nalazi u vitičastoj zagradi, naziva se deklaracija. Deklaracija govori kako se element određen selektorom definira, odnosno kako će se on prikazati. Deklaracije se sastoje od dva dijela- svojstva i vrijednosti, a oni su međusobno odvojeni dvotočkom. Svojstvo deklaracije određuje koji se aspekt nekog elementa definira, na primjer: boja, font, visina, širina itd. Vrijednost definira svojstvo uz koje stoji. Na

primjer, ako se elementu određuje boja, svojstvo će biti *color*, a vrijednost svojstva će biti ona boja koja se želi dodijeliti elementu. Ukoliko se u jednoj deklaraciji definira više svojstava, njih je potrebno odvojiti znakom "točka-zarez". *Slika 10* prikazuje građu *CSS* pravila na primjeru gdje se elementu <*p*>, odnosno paragrafu, dodaje crvena boja teksta.



Slika 10. građa CSS pravila

Postoji više različitih načina definiranja selektora kojim se može dodijeliti *CSS* pravilo nekom *HTML* elementu. Selektori su *case sensitive*, odnosno potrebno ih je pisati točno onako kako su napisana imena elemenata ili atributa u *HTML* kôdu. *Tablica 2* prikazuje neke načine na koje se selektorima mogu odrediti stilovi elemenata.

Tablica 2. vrste selektora CSS-a

Selektor	Značenje	Primjer
Univerzalni selektor	Primjenjuje se na sve elemente u dokumentu	* {}
Element selektor	Primjenjuje se na elemente određene imenom	p {}
Class selektor	Primjenjuje se na elemente koji dijele naziv klase ili samo određene elemente koji dijele naziv klase	.veliki {} p.veliki {}
Id selektor	Primjenjuje se na onaj element čiji naziv *id* atributa odgovara nazivu selektora	#glavni {}
Child selektor	Primjenjuje se na određeni element koji se nalazi unutar određenog elementa	p>a {}
Descendant selektor	Primjenjuje se na element	p a {}

	koji se nalazi unutar određenog elementa, ali ne mora biti *child* element	
Susjedni *sibling* selektor	Primjenjuje se samo na element koji se nalazi odmah nakon nekog elementa	h1+p {}
Opći *sibling* selektor	Primjenjuje se na element koji se nalazi iza nekog elementa	h1~p {}

Ako postoje dva ili više pravila koja se odnose na jedan element, primjenjuje se ono pravilo koje ima određenu prednost prema drugim pravilima. Ukoliko postoje dva selektora koji su isti, primjenjuje se pravilo zadnjeg, odnosno elementu se dodaje onaj stil koji je određen zadnjim napisanim selektorom te vrste. Prema tome bi element selektor imao prednost pred univerzalnim selektorom, *class* selektor bi imao prednost pred element selektorom, *id* bi imao prednost pred *class* selektorom i tako dalje. Ako se neko pravilo želi izdvojiti, odnosno ako se želi staviti kao primarno, u njegovu deklaraciju se dodaje nastavak *!important.* Element koji sadrži druge elemente, odnosno *parent* element prenosi svoje stilove na većinu elemenata u njemu, odnosno *child* elemente. Visok prioritet ima stil koji je dodan kao atribut *style* nekom elementu, odnosno *inline CSS.* Najmanji prioritet imaju zadane postavke internet preglednika koje je odredio *W3C* – linkovi se prikazuju plavom bojom, tekst crnom bojom itd.

Promjena boje je jedno od najrasprostranjenijih svojstava *CSS-a.* Svojstvo *color* omogućuje promjenu boje teksta nekog elementa. Postoji četiri načina na koja je moguće odrediti vrijednost svojstva boje: *RGB* vrijednostima, heksadecimalno, HSL vrijednostima i imenom. *RGB* vrijednosti boje se određuju tako da se svakoj boji (*red, green* i *blue)* dodaje vrijednost od 0 do 255 kao njen udio u konačnoj boji, a 255 je puni ton. Moguće je dodati još jednu vrijednost, vrijednost *alpha* koja je predstavljena *CSS-om 3.* Tada se boja određuje prema *RGBA* vrijednostima, a vrijednosti za *A* parametar određuju transparentnost elementa i kreću se od 0 do 1. Heksadecimalno je boju moguće definirati sa šest znamenki, odnosno dvije za crvenu, dvije za zelenu i dvije za plavu. Vrijednosti mogu biti brojčane, od 0 do 9, ili slovni znakovi, od a do f, gdje je ff puni ton boje. Ispred znamenaka se stavlja

simbol ljestve. *HSL* sustav koristi vrijednosti *hue* (od 0 do 360°) za ton, *saturation* (od 0 do 100%) za zasićenje i *lightness* (od 0 do 100%) za svjetlinu. Također, kao i RGB sustav, može imati i *alpha* (od 0 do 1) vrijednost kojom se definira prozirnost elementa. Posljednji način određivanja boja je njihovim imenom. Postoji 147 definiranih imena boja koje preglednici prepoznaju. Ukoliko se želi postaviti boja pozadine okvira elementa, koristi se *background-color,* ili samo *background,* svojstvo čije se vrijednosti definiraju na isti način kao kod svojstva *color*. *Slika 11* prikazuje tri najčešća načina deklaracije za definiciju boje. Na primjerima je postavljena crvena boja za element naslova <*h1*>.

Slika 11. tri najčešća načina postavljanja boje teksta

Kod uređivanja teksta primarno svojstvo je *font-family* koje omogućuje odabir fonta. Vrijednost ovog svojstva je ime fonta koja se odabire. Korisnik neke stranice mora na svojem računalu imati instaliran font kako bi se ispravno prikazivao. Zbog toga je moguće zadati više fontova istom elementu, ili odrediti je li font serifan ili beserifan. Ukoliko se želi osigurati točan prikaz fonta, koristi se *@font-face* pravilo. Ovim pravilom se određuje mjesto sa kojeg font može biti preuzet ukoliko nije instaliran na računalu korisnika. Kada se nabraja više imena fontova kao vrijednost svojstva, imena se odvajaju zarezom. Ako se ima fonta sastoji od više riječi, piše se unutar navodnika. Veličina fonta se zadaje svojstvom *font-size*. Vrijednost ovog svojstva može biti veličina zadana brojem iza kojeg dolazi nastavak *em* ili *rem* za relativne veličine fontova, ili *px* kao kratica za *pixels.* Također, veličina može biti zadana postotkom prema zadanoj veličini 16px kao 100%. *EMS* je veličina koja se prilagođava zadanoj veličini. Svojstvo *font-weight* definira debljinu teksta. Vrijednosti ovog svojstva mogu biti: *normal, bold, bolder, lighter* itd. *Font-style* je svojstvo kojim se definira ukošenost teksta, a mogu mu se pridodati vrijednosti: *normal, italic* i *oblique*. Svojstvom *text-transform* je moguće neki tekst prebaciti u verzale, kurente ili postaviti početno slovo svake riječi na verzal. Vrijednosti ovog svojstva su: *uppercase,*

lowercase i *capitalize*. *Text-decoration* je svojstvo kojim je moguće istaknuti tekst. Vrijednosti ovog svojstva su: *none* za micanje svih isticanja, *underline* za podcrtavanje teksta, *overline* za dodavanje linije iznad teksta, *line-through* za precrtavanje teksta. *Leading,* odnosno prored se definira svojstvom *line-height*. Razmak između slova ili *kerning* se može regulirati *letter-spacing* svojstvom, a razmak između riječi svojstvom *word-spacing.* Sva ova svojstva kao vrijednosti uzimaju željenu veličinu u nekoj od jedinica veličina teksta. Za poravnavanje teksta koristi se svojstvo *text-align* čije su vrijednosti poravnanja: *left, right, center, justify*. Također, tekst se može poravnati i vertikalno. Neke vrijednosti su: *sub, text-top, top, middle, bottom* i *text-bottom.* Ukoliko se želi napraviti uvlaka u tekstu, ona se radi uz pomoć svojstva *text-indent. Text-shadow* svojstvo omogućuje stvaranje sjene teksta. Iako ovo svojstvo nije podržano u svim internet preglednicima, često se koristi. Vrijednost ovog svojstva se sastoji od brojčanih vrijednosti- dvije koje definiraju položaj sjene (po x i y osi) i jedne koja definira zamućenje, odnosno *blur*. Na kraju vrijednosti stoji boja sjene. *Slika 12* prikazuje primjer oblikovanja teksta.

```
...
h1 {
        font-size:24px;
        font-family:Georgia;
        font-style:italic;
        }
...
<h1> Naslov </h1>
...
```

Naslov

Slika 12. primjer oblikovanja teksta

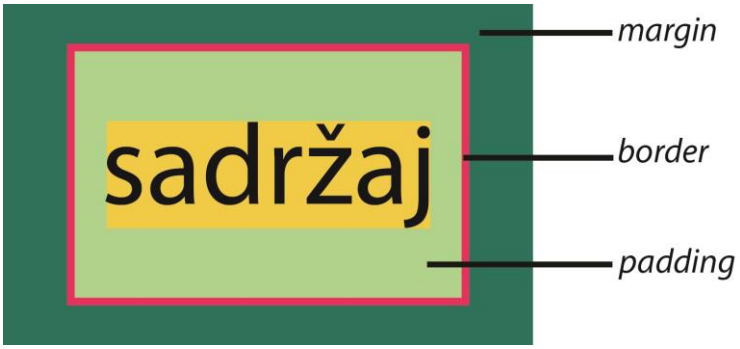
Prema zadanom, internet preglednici linkove prikazuju kao podcrtani tekst u plavoj boji. Ukoliko je stranica na koju link vodi već posjećena, link prema zadanom mijenja svoju boju u ljubičastu. Ova dva zadana svojstva se mogu mijenjati pomoću dvije pseudo-klase. Pseudo-klase se ponašaju kao dodatne vrijednosti za atribut klase. Ukoliko se mijenja izgled linka koji još nije posjećen, koristi se pseudo-klasa *:link*. U tom slučaju će selektor

ovog pravila imati izgled *a:link*. U deklaraciji se definiraju svojstva, odnosno izgled neposjećenog linka. Za link koji je posjećen, izgled se definira pseudo-klasom *:visited.* Također, postoje tri pseudo-klase za oblikovanje linkova u trenutku kada korisnik vrši interakciju s njima, a to su: *:hover* za izgled linka kada se mišem pređe preko njega, *:active* definira izgled linka u trenutku klika na njega i *:focus* koji na link ima sličan utjecaj kao i *:active*, ali stil se zadržava sve do klika na neki drugi element na stranici. Kada se pseudo-klasama dodaju stilovi linku, bitan je redoslijed pseudo-klasa, a on mora biti: *:link, :visited, :hover, :focus, :active.* Svojstvo *cursor* omogućuje promjenu izgleda kursora na linku. Neke od vrijednosti, odnosno neki od izgleda kursora su: *pointer, move, text, help* itd.

Kao što je već spomenuto, *CSS* se prema *HTML* elementima ponaša kao da su smješteni u zasebne okvire (kutije, *boxes*). Postoji nekoliko svojstava koja će utjecati na izgled tih okvira. Njima je moguće definirati dimenzije okvira, dodati rubove, postaviti margine i *padding*, odnosno unutarnju bjelinu i upravljati samim prikazom okvira. Prema zadanom, okviri su onih dimenzija koje zauzima sadržaj u njima. Dimenzije se mijenjaju svojstvima *height* i *width*, a izražavaju se jedinicama: *px*, *em* i postocima. Postoci su relativni, odnosno veličina okvira se mijenja u odnosu s veličinom prozora. Svojstvima *min-width* i *max-width* se definiraju najveće i najmanje širine okvira, a svojstvima *min-height* i *max-heigh* se definiraju najveće i najmanje visine. Ukoliko je sadržaj veći od okvira, svojstvom *overflow* se određuje kako će se prikazati ostatak sadržaja koji ne stane u okvir. Vrijednost *hidden* sakriva sadržaj koji ne stane u okvir, a *scroll* omogućuje "skrolanje" kroz okvir te prikaz cijelog sadržaja.

Svaki okvir ima svoj rub. Rub se može oblikovati uz pomoć vrijednosti *border-width*, a vrijednosti su željene debljine ruba u *px. Border-style* određuje kakva će biti linija ruba: *solid* ravna, *dotted* istočkana, *dashed* iscrtkana itd. *Border-color* definira boju ruba. Sva ova svojstva se mogu skratiti u jedno – *border*, a unose se samo vrijednosti svih spomenutih svojstava, npr. : *border: 3px solid black. Padding* svojstvom se određuje veličina praznine unutar elementa, a *margin* svojstvo određuje veličinu praznine izvan elementa. Oba svojstva se najčešće izražavaju u *px*, a iznosi se određuju za svaku stranu okvira, odnosno nastavcima : *-top, -right, -bottom* i *-left.* Vrijednost *auto* centrira sadržaj. Vrijednosti *padding* i *border* svojstva utječu na dimenzije okvira, odnosno pribrajaju se

postavljenim dimenzijama. Kako bi se zadržale postavljene dimenzija okvira, definira se svojstvo *box-sizing* s vrijednosti *border-box*. *Slika 13* prikazuje primjer margina, ruba i *padding-a.*



Slika 13. margin, border i padding

Inline elementi se mogu pretvoriti u *block* elemente i obrnuto pomoću svojstva *display*. Vrijednosti mogu biti: *inline, block, inline-block* i *none.* Ukoliko se žele zadržati svojstva okvira- njihove dimenzije i raspored sadržaja, moguće ih je sakriti uz pomoć *visibility* svojstva koje može biti *hidden* ili *visible*. Kao i tekstu, i okvirima je moguće dodati sjene uz pomoć *box-shadow* svojstva. Zaobljenost rubova se postavlja *border-radius* svojstvom, a vrijednosti mogu biti različiti iznosi u *px ili %,* a za krug 50%. . Dodavanjem četiri vrijednosti definira se zaobljenost pojedinog ugla, počevši od gornjeg lijevog, u smjeru kazaljke na satu. Tri vrijednosti funkcioniraju na isti način, a zadnjem uglu se dodaje vrijednost njemu suprotnog ugla. Sa dvije vrijednosti se definira zaobljenost dijagonalnih kuteva. Moguće je definirati zaobljenost samo jednog ugla elementa. Ako bi to bio gornji lijevi ugao, svojstvo bi se definiralo kao *border-top-left-radius*.

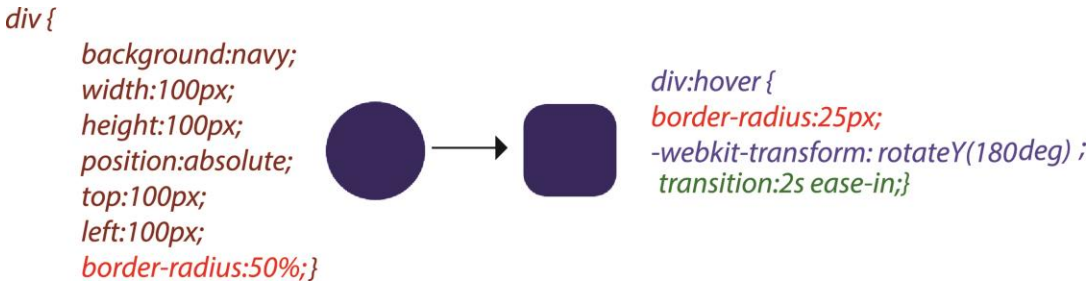
Raspored elemenata na stranici se određuje pozicioniranjem. *CSS* sadrži tri načina pozicioniranja elemenata: *normal flow,* relativno pozicioniranje i apsolutno pozicioniranje. Pozicioniranje se vrši svojstvom *position,* a vrsta pozicioniranja se određuje njegovom vrijednošću. *Normal flow* je vrsta pozicioniranja koja se temelji na izgradnji *block* elemenata. *Block* elementi uvijek započinju u novom redu i vertikalno se nižu jedan iza drugog. Ovaj način pozicioniranja je zadan, ali ukoliko ga je potrebno deklarirati, deklarira

se na način – *position:static;*. Relativno pozicioniranje se vrši deklaracijom *position:relative;*. Kod relativnog pozicioniranja potrebno je odrediti vrijednosti na koje se pozicionira element. One se određuju svojstvima *top* i *left,* a iskazuju se u *px*. Kod relativnog pozicioniranja ishodište koordinatnog sustava za svaki element je ona pozicija na kojoj bi element bio smješten kod *normal flow* pozicioniranja. Ako se relativno pozicioniranje želi prikazati kao horizontalno pozicioniranje, dodaje mu se svojstvo *float* i vrijednost *left.* Apsolutno pozicioniranje se postavlja pomoću deklaracije *position:absolute*. Pozicija se određuje vrijednostima u *px* dodanim svojstvima *top* i *left.* Ishodište koordinatnog sustava za ovakvo pozicioniranje je gornji lijevi ugao prozora preglednika i kao takvo je potpuno neovisno o položaju drugih elemenata. *Fixed* vrijednost određuje pozicioniranje nekog elementa na način na koji to vrši i apsolutno pozicioniranje, ali ovakvim pozicioniranjem taj element zadržava svoju poziciju kod pregledavanja stranice, odnosno "skrolanja". Ukoliko dolazi do preklapanja elemenata, moguće je odrediti koji će element biti na vrhu uz pomoć *z-index* svojstva. Vrijednost ovog svojstva je broj, a element s većim *z-indexom* će biti iznad onog s manjim *z-indexom.*

Osim određene boje, pozadina nekog elementa može biti slika. *Background-image* svojstvo omogućuje postavljanje neke slike kao pozadinu elementa. Vrijednost ovog svojstva je npr. *url("slika.jpg").* Slika se može ponavljati, ili ponavljanje može biti određeno. Svojstvo *background-repeat* sa svojim vrijednostima određuje vrstu ponavljanja. Vrijednosti su: *repeat* za horizontalno i vertikalno ponavljanje, *repeat-x* za horizontalno, *repeat-y* za vertikalno ponavljanje, *no-repeat* za isključenje ponavljanja i *fixed* kako bi slika ostala na istoj poziciji. Također je moguće odrediti poziciju slike u elementu. Ona se određuje svojstvom *background-position,* a može biti *left, center* ili *right* u kombinaciji sa *top, center* ili *bottom.* Osim boje i slike, pozadina može biti gradijent. Gradijent se dodaje svojstvom *background* čija je vrijednost vrsta gradijenta, npr. *linear-gradient* ili *radial-gradient.* U nastavku vrijednost se u zagradi pišu boje koje se prikazuju kao gradijent.

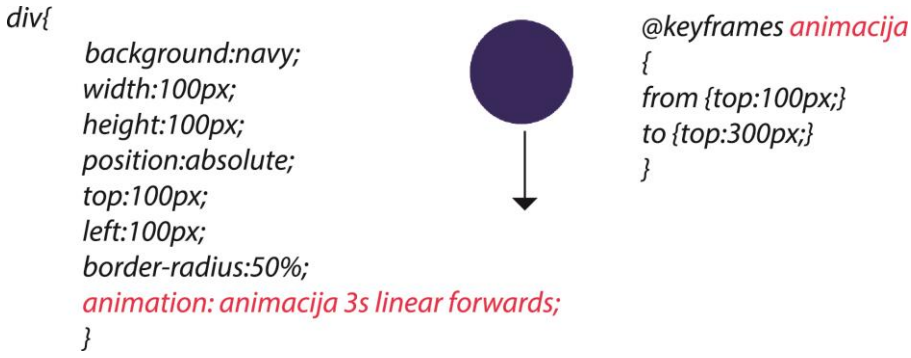
CSS-om je moguće i transformirati HTML elementa. Transformacije mogu biti 2D i 3D i definiraju se svojstvom *transform.* Vrijednosti svojstva *transform* definiraju vrstu transformacije na elementu. Neke od vrijednosti su: *translate(x,y)* za 2D translaciju elementa, *translate3d(x,y,z)* za 3D translaciju elementa, *translateX(x)* za translaciju po X

osi (ili Y ili Z osi), *scale(x,y)* za promjenu veličine elementa, *rotate(angle)* za određivanje kuta zakretanja elementa, *rotate3d(x,y,z, angle)* za određivanje 3D pomaka i kuta rotacije itd. Svakoj transformaciji, ili bilo kojoj promijenjenoj vrijednost elementa preko kojeg se pređe mišem (*:hover)* ili klikom na njega (*:active)* se može odrediti vrijeme i način izvođenja transformacije ili promjene svojstva pomoću svojstva *transition.* Vrijednosti tog svojstva mogu biti: vrijeme trajanja, način izvođenja i početak tranzicije. Način izvođenja može biti: *linear, ease, ease-in, ease-out* i *ease-in-out. Slika 14* prikazuje primjer transformacije elementa i promjene vrijednosti koje se događaju prelaskom miša preko njega.



Slika 14. Primjer transformacije i tranzicije na :hover pseudo-klasi

Pomoću *CSS-a* je moguća i izrada animacija. Animacije, također, moraju imati svoj prefiks kako bi ih preglednik prepoznao. Animacija se definira tako da se elementu koji se animira dodaje svojstvo *animation* s vrijednostima koje određuju ime animacije, trajanje animacije, način izvođenja, početak animacije i njeno ponavljanje. Ukoliko vrijednost ponavljanja bude *forwards,* animacija se izvodi jednom i ostaje u završnoj fazi, za vrijednost *infinite* animacija se izvodi beskrajno mnogo puta, za vrijednost *alternate* animacija se izvodi do kraja i zatim obrnutim smjerom od kraja do početka. Ono što se događa se događa u animaciji se određuje u pravilu *@keyframes.* Koraci animacije mogu biti određeni postotkom trajanja animacije ili kao početni i završni korak naredbama *from* i *to.* U tom pravilu su definirani stilovi za svaki pojedini korak u animaciji. Na *slici 15* je primjer animacije *div* elementa.



Slika 15. Primjer animacije div elementa

2.2.3 Media Query tehnologija i responzivnost

Media Queries je jedan od modula *CSS3* koji omogućava prilagodbu sadržaja različitim rezolucijama internet preglednika, odnosno uređaja na kojima se sadržaj pregledava. Ova tehnologija je temelj responzivnog web dizajna.

Media queries se definira pomoću *@media* pravila. Prije razvoja *CSS3* modula *media queries*, pravilom *@media* se definirao tip uređaja za pregled sadržaja- *media type*. Neki od tipova uređaja su: *aural* za čitače zaslona, *handheld* za manje uređaje, *print* za pisače, *screen* za zaslone stolnih računala itd. *Media queries* tehnologija povećava funkcionalnost prilagodbe sadržaja jer u obzir uzima mogućnosti uređaja, a ne samo njegov tip. Funkcionira na način da postavlja pitanje pregledniku o nekoj od mogućnosti prikaza, a preglednik uzima u obzir onu mogućnost koju smatra istinitom za sebe, odnosno *true.* Neke od mogućnosti uređaja koje *media queries* tehnologija može ispitati su: *width* za moguću širinu prikaza sadržaja, *height* za moguću visinu prikaza sadržaja, *orientation* za prikaz sadržaja u *portrait* ili *landscape* obliku, *aspect-ratio* za odnos visine i širine mogućeg pregleda na zaslonu itd.

Stilovi za svaki *breakpoint* se mogu uvesti pomoću <*link*> elementa kao eksterni *.css* dokument, ili se mogu pisati u postojećem *CSS* kôdu. *Breakpoint* je ona točka na kojoj dolazi do promjene stila, npr. *max-width:x. Slika 16* prikazuje razliku između *media type* i *media queries* tehnologije. Na primjeru za *media queries* tehnologiju (*slika 16, dolje)*

vidljivo je kako se određeni stil odnosi samo na zaslone čija je orijentacija *portrait*, a širina prikaza je minimalno 800px [8].

```
<link rel="stylesheet" type="text/css" media="screen" href="screen-
styles.css">

<link rel="stylesheet" media="screen and (orientation: portrait) and
(min-width: 800px)" href="800wide-portrait-screen.css" />
```

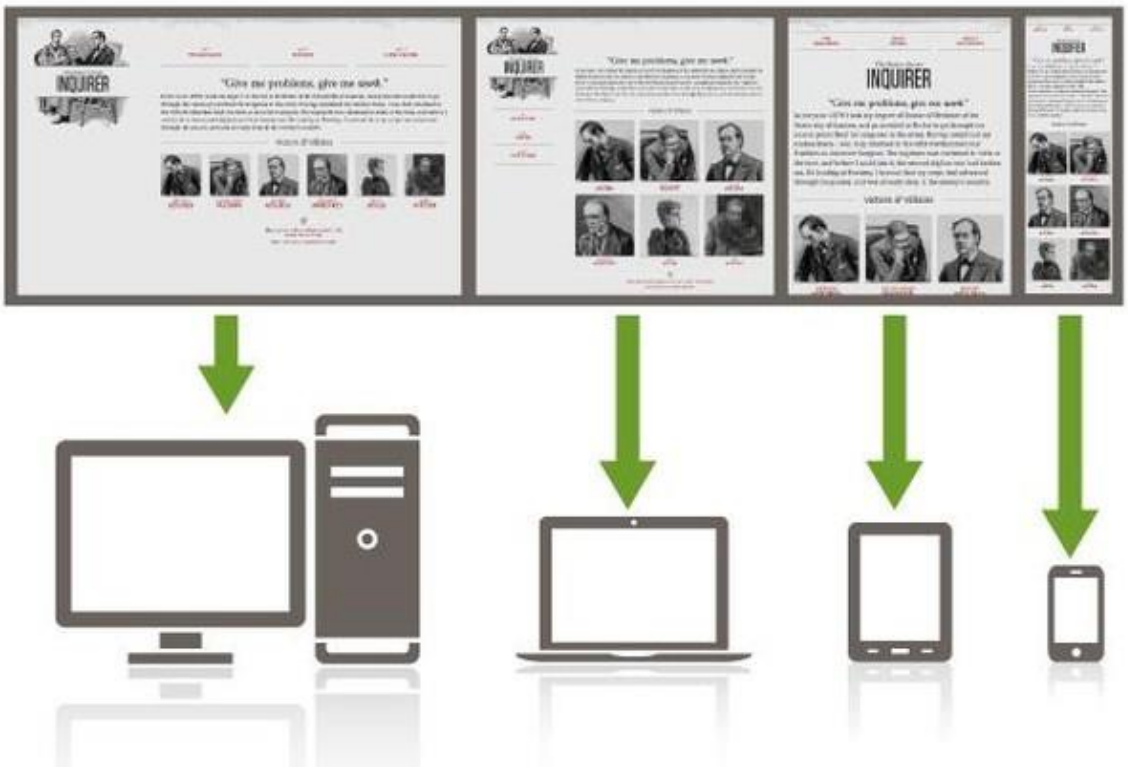
Slika 16. primjeri za: media type (gore) i media queries (dolje)

Kao što je spomenuto, stil za svaki *breakpoint* može biti definiran i unutar *CSS* kôda. Ovakav pristup omogućuje olakšano upravljanjem samo onim svojstvima koje je potrebno prilagoditi značajkama uređaja koji prikazuje sadržaj. *Slika 17* prikazuje stil definiran unutar samog *CSS-a* za određeni *breakpoint.* Na primjeru je vidljivo da je za sve *screen* uređaje maksimalne širine zaslona 400px naslov *h1* prikazan u zelenoj boji.[9]

```
@media screen and (max-device-width: 400px) {
  h1 { color: green }
}
```

Slika 17. stil definiran unutar CSS kôda

Zbog velikog napretka tehnologije i sve veće potrebe korištenja web-a preko pametnih telefona, primjena responzivnog web dizajna je sve šira. [10] Izradom responzivnog dizajna za web stranice, zadovoljava se korisničko iskustvo, a samim time se povećava i broj ljudi koji koriste stranicu. [11] Na *slici 18* je moguće vidjeti primjer responzivnog dizajna web stranice koji je prilagođen svim zaslonima od stolnog računala pa do zaslona pametnog telefona.



Slika 18. Primjer responzivnog web dizajna

2.3 JavaScript i jQuery tehnologija

JavaScript je skriptni programski jezik predstavljen od strane *Netscape-a* 1995. godine. Njegove skripte se izvršavaju u internet preglednicima na strani korisnika. Skripte se izvršavaju naredba po naredba bez prethodnog prevođenja cijelog programa i kreiranja izvršne datoteke što *JavaScript* čini interpreterom. Cilj ovog programskog jezika jest olakšati korištenje stranica i stvoriti komunikaciju korisnika i servera, odnosno dodati interaktivnost *HTML* stranicama. Uz *HTML* i *CSS* ovo je temeljna tehnologija web-a. Iako je *JavaScript* imenom i sintaksom vrlo sličan *Javi*, ova dva programska jezika nemaju drugih sličnosti. *JavaScript* je podržan od strane svih poznatih internet preglednika. Također, on je javno raspoloživ skriptni jezik. [12]

JavaScript omogućuje programiranje u okviru *HTML* stranica, pretvaranje dinamičkog teksta u *HTML* stranicu, reagiranje na događaje, čitanje i pisanje *HTML*

elemenata, validiranje podataka, detektiranje preglednika kojeg koristi korisnik i kreiranje „kolačića“, odnosno *cookies-a*.

Skripte se pišu unutar *HTML* elementa <*script*>. Ovaj element se može nalaziti unutar <*head*> ili <*body*> elementa. Također, *JavaScript* se može uvesti kao zasebna datoteka tako da se elementu <*script*> dodaje atribut *src* kojim se definira mjesto odakle se učitava *JavaScript* datoteka. *JavaScript* datoteke imaju ekstenziju“*.js“*. *Slika 19* prikazuje primjer jednostavne skripte. Ova skripta ispisuje rečenicu u prozoru internet preglednika.

```
<script type="text/javascript">
document.write("Hello, World!");
</script>
```

Slika 19. primjer skripte napisane u JavaScript-u

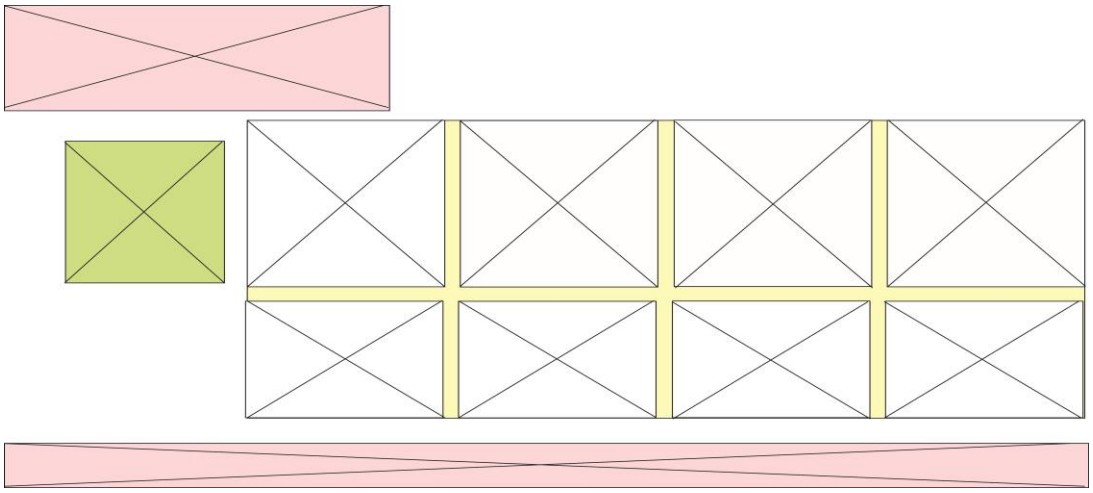
Kako bi se olakšalo korištenje *JavaScript-a,* osmišljene su *JavaScript* biblioteke. [13] To su skupovi prije napisanog *JavaScript-a.* Najpoznatiji primjer takve biblioteke je *jQuery*. On je biblioteka namijenjena za manipulaciju *DOM-om (Document Object Model).* *DOM* je razgranati prikaz svih elemenata web stranice. *jQuery* na jednostavan način omogućuje traženje, selektiranje i manipuliranje tim elementima. *jQuery* je besplatan *open-source* softver. Njegova sintaksa je dizajnirana kako bi pojednostavila sintaksu *JavaScripta* za snalaženje u *HTML* dokumentu, selektiranje elemenata, izvršavanje naredbi itd. *jQuery* je danas podržan od strane svih velikih internet preglednika. *Slika 20* prikazuje opći primjer pisanja *jQuery* biblioteke.

```
<script type="text/javascript">
$(window).load(function() {
        $(".loader").fadeOut("slow");
})
</script>
```

Slika 20. primjer korištenja jQuery biblioteke

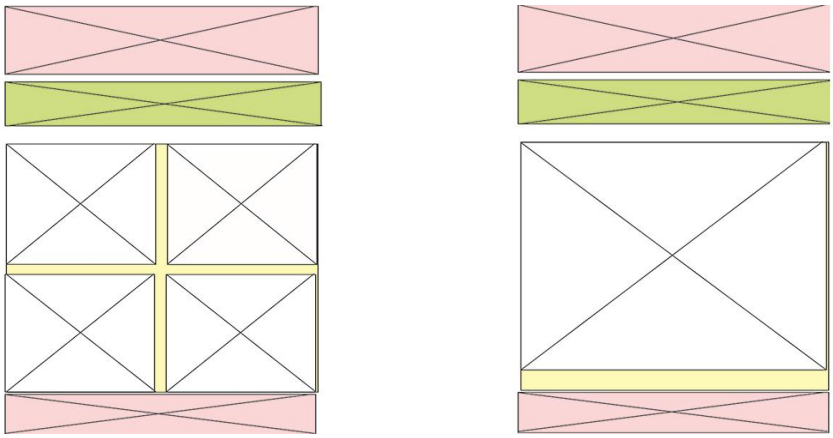
3. EKSPERIMENTALNI DIO

Stranica koja se izrađuje u ovom radu je galerija obrađenih fotografija. Ona mora biti funkcionalna za prikaz na svim uređajima. Kako bi to bilo moguće, prvo je zamišljena skica, odnosno *wireframe*. *Wireframe* je pojednostavljeni prikaz ključnih informacija koje će stranica prikazivati, kao i raspored elemenata – zaglavlje, navigacija i slično. *Slika 21* prikazuje skicu sa ključnim elementima za prikaz na računalu.



Slika 21. raspored elemenata za prikaz na računalu

Kako bi stranica bila funkcionalna za korištenje na zaslonima manje rezolucije, bitno je prilagoditi ključne elemente kako bi željena informacija bila prikazana na ispravan način. Prema zamišljenoj skici izgleda stranice na računalu, razvijene su ideje za izgled stranice na manjim zaslonima za *handheld* uređaje (mobilne uređaje i tablete) koje su prikazane na *slici 22*. Na obje slike elementi su prikazani različitim bojama zbog lakšeg snalaženja- *header* i *footer* su rozi, navigacija je zelena, a slike bijele. Sve slike su smještene u jedan zajednički *div* element. Taj *div* element je na *wireframe-u,* odnosno skici označen žutom bojom.



Slika 22. raspored elemenata prilagođen za mobilne tablete i mobilne uređaje

Nakon što je napravljen plan izgleda stranice, započinje i njena izrada, odnosno pisanje kôda. Za pisanje kôda stranice koristi se tekst *editor Notepad++*. Izrada stranice započinje izradom početne, odnosno *index* podstranice. Ukoliko postoji više podstranica na jednoj web stranici, server prema zadanome otvara datoteku koja se naziva *index.html.*

Sam kôd započinje *<!DOCTYPE html>* oznakom, odnosno deklaracijom, kako bi se definirala točna inačica standarda koji se koristi, a to je *HTML5.* Nakon otvaranja *<html>* oznake, otvorena je oznaka *<head>*. U ovoj se oznaci nalaze elementi koji se ne prikazuju kao sadržaj stranice, ali stranici daju dodatne informacije. Prvi takav element je element *title* kojim je zadan naslov stranice- *Retouch.* Elementom *meta* i njegovim atributom *charset* određena je vrsta prikaza znakova za kodiranje. Kako bi se pravilno prikazivali znakovi hrvatskog jezika, odabrana je vrijednost *UTF-8.* Također, *meta viewport* elementom se određuje prikaz stranice koji ovisi o veličini uređaja na kojem se stranica otvara, te se postavlja zadano povećanje stranice. Ova stavka je bitna za responzivnost stranice i bez nje, stranica ne bi bila responzivna. Nakon *meta* elementa, u *head* elementu se nalazi *link* element. Ovim elementom se definira poveznica između *HTML* dokumenta i *CSS* dokumenta u kojem su definirani stilovi stranice. Atribut *rel* definira vrstu poveznice između ta dva dokumenta. Budući da je riječ o dodanim stilovima, vrijednost atributa je *stylesheet.* Osim navedenih elemenata, u *head* elementu postoji i element *script* u kojem je definiran *URL* kojim je omogućeno korištenje jQuery biblioteke. Nakon što su definirani

svi potrebni elementi u *head* elementu prikazani na slici, on se zatvara kao što je i prikazano na *slici 23*.

```
<!DOCTYPE html>


<head>
    <title> Retouch </title>
    <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" href="a1.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>


</head>
```

Slika 23. definiranje head elementa

Kao što je navedeno u teorijskom dijelu rada, nakon *head* elementa dolazi *body* element. U ovom elementu je definiran sav vidljiv sadržaj stranice. Samom *body* elementu je definiran stil. Kao pozadina mu je zadan gradijent od sive do bijele boje. Svojstvo *background-attachment* određuje hoće li se postavljen gradijent primijeniti duljinom cijele stranice fiksno ili će „putovati“ po stranici tokom *scroll-a*. Zadanom vrijednosti svojstva *fixed* gradijent ostaje na fiksnoj poziciji i ne ponavlja se. *Background* svojstvo je raspisano nekoliko puta sa prefiksima za točan prikaz linearnog gradijenta u svim preglednicima. Na *slici 25* je definiran stil *body* elementa.

```
body {

    background: -webkit-linear-gradient(#F0F0F0, #FFFFFF);
    background: -moz-linear-gradient(#F0F0F0, #FFFFFF);
    background: -o-linear-gradient(#F0F0F0, #FFFFFF);
    background: linear-gradient(#F0F0F0, #FFFFFF);
    background-attachment: fixed;
    margin:0px;
    padding:0px;
}
```

Slika 24. stil body elementa

Prvi element koji se nalazi u elementu *body* je *div* element kojem je dodan *id* atribut *loader*. Ovaj *div* element se prikazuje pri učitavanju stranice nakon čega nestaje. To je ostvareno uz pomoć kratke skripte koja se nalazi u *script* elementu. Stil ovog *div* elementa je definiran u *CSS* dokumentu. Zadana mu je boja pozadine (bijela) te je pozicioniran preko cijelog ekrana. Također, uz pomoć *z-index* svojstva, element se stavlja ispred svih elemenata koji su definirani nakon njega. Ovime se dobiva dojam bljeska pri učitavanju stranice. *Slika 25* prikazuje kako je definiran *div* element koji ima ulogu *loader-a* (lijevo) i njegov stil (desno).

```
<div id="loader"> </div>
<script type="text/javascript">
$(window).load(function() {
    $("#loader").fadeOut("slow");
})
</script>
```

```
#loader {
    background:#FFFFFF;
    position: absolute;
    left: 0px;
    top: 0px;
    width: 100%;
    height: 100%;
    z-index: 10000;
}
```

Slika 25. definiranje div elementa i njegovog stila

Nakon toga, definiran je *header* element, odnosno zaglavlje prikazano na *slici* 26. Ovaj element sadrži naslov prve razine (*h1)* u koji je smješten natpis „GALERIJA“. Ovaj natpis je, također, i link (*a)* koji ponovno učitava početnu stranicu, odnosno *index.html* dokument. Budući da je korišten font preuzet sa interneta, pomoću *@font-face* pravila definiran je *URL* fonta na serveru, odnosno u mapi koja se nalazi na računalu, kako bi korisnicima bio omogućen prikaz ispravnog fonta na njihovim uređajima. *@font-face* pravilo je objašnjeno u teorijskom dijelu rada.

```
@font-face {
    font-family:Constantine;
    src: url(fonts/Constantine.ttf);
    }

header {
    padding:0px 40px;
    width:100%;
    height:140px;
    background: -webkit-linear-gradient(#E0E0E0, #F0F0F0,);
    font-size:40px;
    color:#FFFFFF;
    font-family:Constantine;
    text-align: left;
    text-shadow:2px 2px 1px gray;
    box-sizing:border-box;
}
```

Slika 26. definiranje header elementa

Navigacija je definirana uz pomoć *nav* elementa. Navigacija u sebi sadrži link *about* koji vodi do *footer-a* i link *contact* koji otvara novu podstranicu. Linkovi su smješteni u neuređenoj listi. U *nav* elementu se nalazi i div element sa klasom *handle* koji je vidljiv samo na manjim zaslonima. U tom *div* elementu se nalazi ikona koja označava navigaciju. Izgled *HTML* kôda za izradu navigacije je prikazan na *slici 27.* Kako bi navigacija dobila svoj izgled, u *CSS-u* joj je definiran stil. Stil je definiran za cijelu listu i za svaku stavku u listi posebno. Svim linkovima u kôdu je isključena dekoracija kako se ne bi prikazivali podcrtani. Također, pomoću *list-style-type* svojstva, isključene su oznake stavkama u listi. Prelaskom miša preko elemenata u listi mijenja se boja pozadine.

Za *div* element kojem je određena klasa *handle* je određena boja, kao i *cursor* čija je vrijednost *pointer* te se mijenja prelaskom miša preko div elementa klase *handle* . Cijeli *CSS* korišten za prikaz navigacije, odnosno liste i elemenata u listi navigacije se nalazi na *slici 28.*

```
<nav>
    <ul>
        <a href="#about"> <li> About </li> </a>
        <a href="contact.html"> <li> Contact </li> </a>

    </ul>
    <div class="handle"> <img id="menu" src="contact/menu.png"></div>
</nav>
```

Slika 27. HTML kôd navigacije

```
nav ul {
    font-weight:bold;
    overflow:hidden;
    color:gray;
    padding:10px 10px;
    text-align:center;
    font-size:20px;
    font-family:"Georgia";
    margin:0px;
    -webkit-transition: max-height 0.4s;
    width:200px;
    border-top:2px solid lightgray;
    }

a {
    text-decoration:none;
    color:inherit;
    }

nav ul li {
    list-style-type:none;
    width:200px;
    padding: 20px;
    }
```

```
li {
    margin:10px;}

nav ul li:hover {
    background-color:#B0C4DE;
    }

#menu {
    height:40px;
    width:40px;
    margin:auto auto;
}

.handle {
    width:100%;
    background:#81a7a3;
    text-align:center;
    box-sizing:border-box;
    padding: 15px 10px;
    cursor:pointer;
    color:white;
    display:none;
}
```

Slika 28. Stilovi zadani za elemente navigacije

Kako bi navigacija bila funkcionalna i kako bi se otvarala klikom na nju za sve manje preglednike, potrebno je ubaciti skriptu koja je zadužena za izvršavanje te radnje. Skripta koristi jQuery biblioteke te je kao takva vrlo jednostavna i kratka kao što je vidljivo na *slici 29.*

```
<script type="text/javascript">
    $ ('.handle').on('click', function() {
        $('nav ul').toggleClass('showing');
    });

</script>
```

Slika 29. skripta koja upravlja navigacijom

Svi do sad navedeni elementi koriste se i u drugom *HTML* dokumentu, odnosno podstranici „contact.html“. Osim navedenih elemenata, zajednički im je i element *footer. Footer* element je povezan sa *header* elementom. Element *header* ima *id* atribut naziva *top.* Upravo preko tog *id-a, footer* postaje link kojim se moguće vratiti na vrh stranice. Ova opcija je korisna kod uređaja s manjim zaslonom. *Footer* sadrži paragraf čiji je *id about.* Linkom „*About“* koji se nalazi u navigaciji, moguće je brzo pristupiti dnu stranice, odnosno *footer-u.* Na *slici 30* se nalazi *HTML* kôd i *CSS* za *footer* element.

```
<a href="#top"><footer>
<p id="about"> Copyright 2015 Blisspen
<br> All Rights Reserved </p>
</footer> </a>
```

```
footer {
    height: 80px;
    width:100%;
    position: absolute;
    left: 0px;
    top:130%;
    background: black;
    font-size:20px;
    color:#FFFFFF;
    text-align: center;
    box-sizing:border-box;
    margin-bottom:0px;
    padding:15px;
    z-index:600;
}
```

Slika 30. footer element i pripadajući CSS kôd

U dokumentu *index.html* se prikazuje glavni sadržaj- obrađene fotografije. Zbog lakšeg pozicioniranja slika, sve slike su smještene u *div* element „okvir“. Svaka slika se nalazi u svojem *div* elementu kojem je definirana klasa „maska“. Ti su elementi isključivo

vizualnog karaktera osim toga nemaju bitniju ulogu. Prelaskom miša *hover* preko svake maske, maski se boja pozadine mijenja u crnu. Kako bi se prijelaz izvršavao postepeno, dodano je svojstvo *transition* kojim je određeno vrijeme prijelaza.

Svaki element *img,* odnosno slike su dodane u *div* elemente s klasom „maska“. Svaki element *img* ima svoj *id* atribut te dvije klase- „slika“ i „slikaugaleriji“. Poredak elemenata u *div* elementu „okvir“ je vidljiv na *slici 31* u skraćenom obliku.

```
<div id="okvir">
    <div class="maska"> <img src="final/a1.jpg"
        id="a11" class="slike" class="slikaugaleriji"> </div>
    <div class="maska"> <img src="final/b2.jpg"
        id="b21" class="slike" class="slikaugaleriji"> </div>
</div>
```

Slika 31. skraćeni primjer definiranja slika

Klasa „slika“ definira stil za sve slike koje se nalaze na početnoj *index* stranici. Slike su, kao i maske, pozicionirane relativno i dodano im je svojstvo *float:left* kako bi se pozicionirale horizontalno. Klasa „slikaugaleriji“ se primjenjuje na *img* elemente onda kada se oni otvaraju u galeriji. U isto vrijeme se na *img* element prestaje primjenjivati klasa „slika“. Kako bi to bilo ostvarivo, napisana je skripta. *Id* atribut svake slike zadužen je za promjenu slike prelaskom miša preko nje. Prelaskom miša, prikazuje se neobrađena fotografija. Kako ne bi dolazilo do prikaza neobrađene fotografije u galeriji, klikom na sliku se atribut *id* više ne primjenjuje na *img* element. To je također izvedeno pomoću jQuery biblioteke, a kôd je prikazan na *slici 32*.

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

    $("#galerija").css("display", "block");
    $("#galerija").append($(this).clone().toggleClass("slikaugaleriji"));
    $("#galerija").focus();
    $(".slikaugaleriji").removeAttr('id');
```

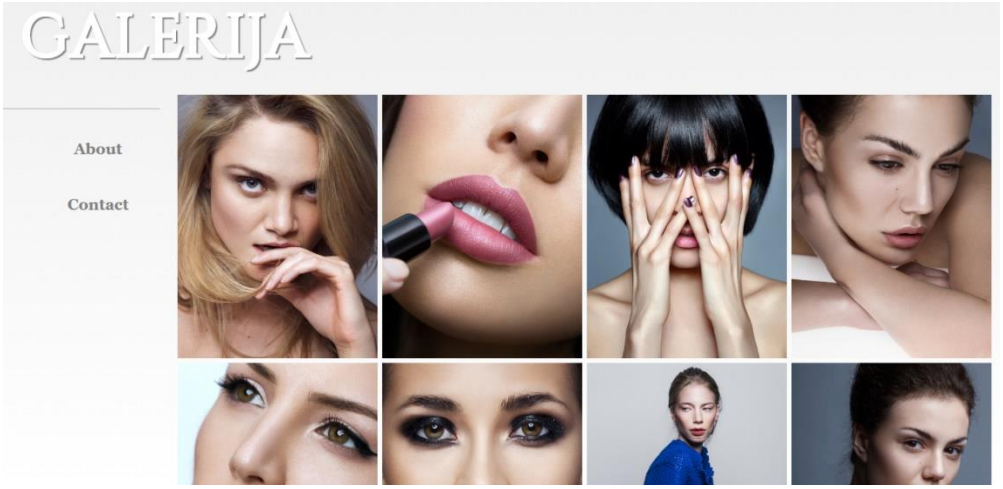
Slika 32. jQuery kojim se ostvaruje prikaz slike u galeriji

Div element „galerija“ nije prikazan cijelo vrijeme već samo onda kad se klikom otvara slika u njemu. Kako bi galerija bila funkcionalna, moguće je listati slike klikom na njih, a galerija se gasi tipkom *esc*. Zbog toga, galerija ima atribut *tabindex* s vrijednosti 1. *Slika 33* prikazuje *div* element „galerija“ te pripadajuće skripte.

```
<div id="galerija" tabindex="1"> </div>

$("#galerija").keydown(function(e) {
if (e.keyCode == 27) {
    $(this).css("display", "none");
    $(this).empty();
    }
});
$("#galerija").click(function () {
        var srcTrenutne = $(this).find(".slikaugaleriji").attr("src");
        var prethodna = $("#okvir").find("img[src$='" + srcTrenutne + "']")
        .parent().prev().find(".slike").attr("src");
        var iduca = $("#okvir").find("img[src$='" + srcTrenutne + "']")
        .parent().next().find(".slike").attr("src");

        //$("#galerija").find(".slikaugaleriji").attr("src", prethodna);
        $("#galerija").find(".slikaugaleriji").attr("src", iduca);
    });
```

Slika 33. definiranje elementa galerija i pripadajućih skripta

U *HTML* dokumentu *Contact* sadržaj je smješten u tablicu. Ikone društvenih mreža su linkovi koji vode na *Facebook* i *Twitter*. Sve ikone su besplatne te su preuzete sa interneta. *Slika 34 i 35* prikazuju konačne izglede stranica *index* i *contact.*



Slika 34. index.html

Slika 35. contact.html

Kako bi stranica bila responzivna, potrebno je definirati *breakpoints,* odnosno točke na kojima dolazi do promjene izgleda stranice pomoću *media queries* tehnologija unutar *CSS* dokumenta. Sadržaj stranice je prilično širok pa je kao prvi *breakpoint* uzeta maksimalna širina od 1024px. Za sve uređaje čiji zaslon prikazuje sadržaj na većoj širini od navedene, sadržaj će biti prikazan kako već definiran. Za uređaje čija je širina prikaza od 1024px ili manja, neki elementi se mijenjaju. Navigacija mijenja svoj izgled te se prikazuje *div* element sa klasom *handle.* Klikom na taj element, otvara se navigacija, a ponovnim klikom se zatvara. Osim navigacije, velika razlika u prikazu je i raspored elemenata. U redu se prikazuju samo dvije slike te je isključena opcija galerije jer ona ne bi bila funkcionalna na uređajima sa zaslonima na dodir.

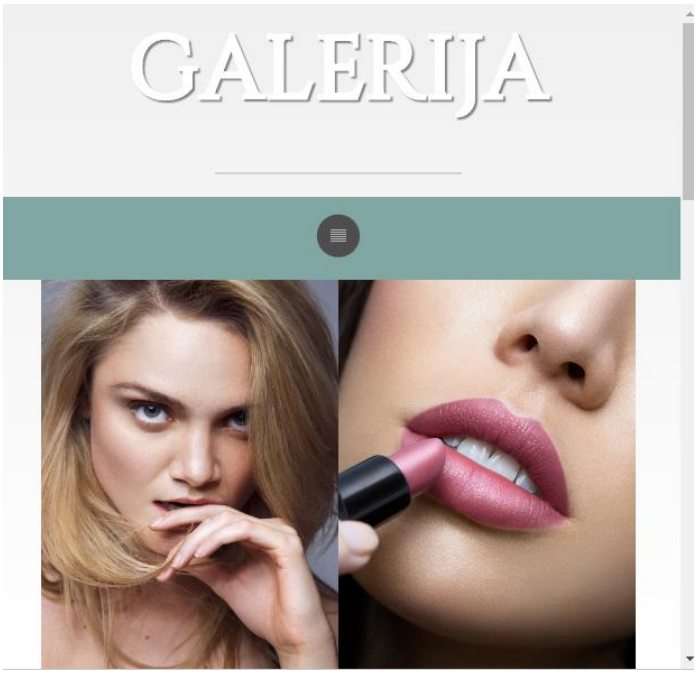
Drugi *breakpoint* je postavljen za maksimalnu širinu preglednika od 580px. Ovakav prikaz stranice namijenjen je mobilnim uređajima. Na svim uređajima maksimalne širine prikaza od 580px, ili manje, prikazivat će se samo jedna slika u redu. Također, mijenjaju se i dimenzije *header* i *footer* elemenata kao i same pozicije elemenata na stranici. Primjer *CSS media queries* koda za oba *breakpoint-a* se nalazi na *slikama 36* i *37.* Konačni izgled početne stranice za manje uređaje je prikazan na *slikama 38* i *39.*

```
@media (max-width: 1024px)

{
    header {text-align:center;}
    nav ul {
        max-height:0px;
        margin:auto;
        }
    .showing {
        max-height: 20em;

    }
    nav ul li {
        width:100%;
        box-sizing:border-box;
        padding: 5px;

    }
```

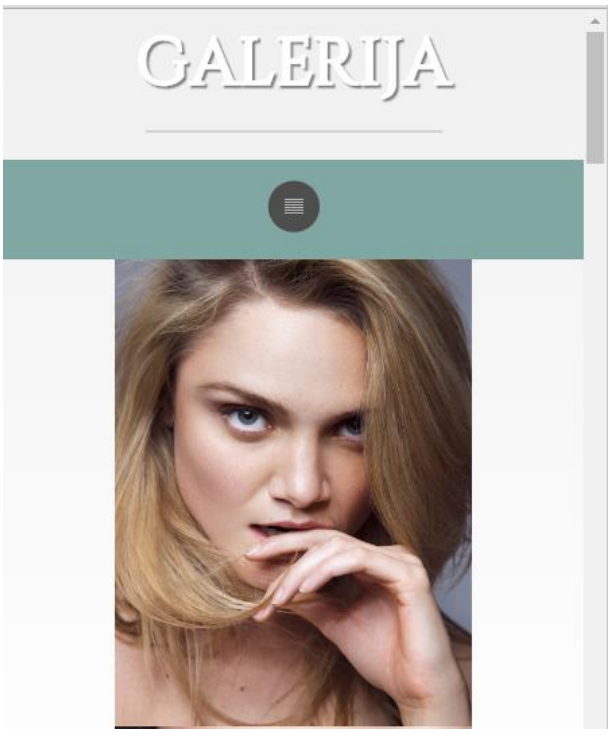
Slika 36. media queries breakpoints za maksimalnu širinu od 1024 px

```
@media(max-width: 580px)
{
    header {
        width:100%;
        font-size:25px;
        height:80px;
    }
    nav ul {
        max-height:0px;
        margin:auto;
    }
    .showing {
        max-height: 20em;

    }
    nav ul li {
        width:100%;
        padding: 5px;
```

Slika 37. media queries breakpoints za maksimalnu širinu od 580 px



Slika 38. Konačni izgled stranice za maksimalnu širinu od 1024 px



Slika 39. Konačni izgled stranice za maksimalnu širinu od 580 px

4. ZAKLJUČAK

Razvojem današnje tehnologije javila se potreba za pregledavanjem sadržaja na webu na različitim uređajima. Zbog toga je pojava responzivnog web dizajna bila od velikog značaja za brojne korisnike. Postotak korisnika koji pretražuju web pomoću mobilnih uređaja i dalje raste, a time raste i popularnost responzivnog dizajna.

Stranica predstavljena u ovom radu je galerija radova obrađenih fotografija. Njena funkcionalnost je ispitana u *Google Chrome* pregledniku opcijom *inspect* te *responsive.* Također, u istom je pregledniku izvršena simulacija prikaza web stranice na popularnim mobilnim uređajima – *Iphone 5, Iphone 6, Samsung S5* i mnogim drugima.

U ovom radu je prikazan primjer stranice koja je prilagođena za različite uređaje. Postupak izrade stranice nije kompleksan jer *media queries* tehnologija omogućava izradu responzivne web stranice na vrlo jednostavan način. Prilagodbom stranice za različite rezolucije internet preglednika dobiva se stranica koja je funkcionalna i dostupna za sve korisnike. Samim time, zbog olakšanog korištenja stranice na svim uređajima, njena posjećenost zasigurno raste.

5. LITERATURA

1. http://www.w3schools.com/html/ - 01.08. 2016.
2. http://infomesh.net/html/history/early/ - 01.08.2016.
3. https://validator.w3.org/docs/sgml.html - 30.07.2016.
4. https://www.w3.org/People/Raggett/book4/ch02.html - 01.08.2016.
5. http://www.ieee.hr/_download/repository/mipro_xml_tekst.pdf - 02.08.2016.
6. Duckett J. (2011.) *HTML & CSS design and build websites,* Indianapolis: John Willey & Sons, Inc.
7. http://www.w3schools.com/html/html_attributes.asp-05.08.2016.
8. Frain B.(2012.) *Responsive Web Design with HTML5 and CSS3,* Birmingham: Packt Publishing Ltd.
9. Crespo G. (2013.) *Responsive Web Design with jQuery*, Birmingham: Packt Publishing Ltd.
10. Jehl S. (2014.) *Responsible responsive design,* New York: A Book Apart
11. Marcotte E. (2011.) *Responsive Web Design*, New York: A Book Apart
12. Duckett J. (2014.) *JavaScript & jQuery*, Indianapolis: John Willey & Sons, Inc.
13. http://www.w3schools.com/jquery/jquery_get_started.asp - 17.08.2016.