SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

ZAVRŠNI RAD

Andrija Prelec

Sveučilište u Zagrebu
Grafički fakultet

Smjer: Tehničko-tehnološki

ZAVRŠNI RAD

IZRADA RESPONZIVNE INTERNETSKE STRANICE UPOTREBOM HTML5 i CSS3 TEHNOLOGIJA

Mentor:
doc. dr. sc. Tajana Koren Ivančević

Student:
Andrija Prelec

Zagreb, 2015.

SAŽETAK

Ovaj završni rad bavit će se HTML i CSS tehnologijama za izradu internetskih stranica, od nastanka pa sve do danas s naglaskom na aktualne HTML5 i CSS3 inačice te svime što su one promijenile i uvele što se tiče standarda.

Poseban naglasak je na problemu prilagodbe sadržaja (responzivnost stranice) svim uređajima i mogućim veličinama zaslona koji danas omogućavaju pregled internetskih stranica pomoću web preglednika (*browsera*), od pametnih telefona do tableta i osobnih računala. Cilj rada jest ustanoviti koliko je zapravo moguće prilagoditi sadržaj prosječne internetske stranice suvremenim uređajima i internetskim preglednicima bez korištenja ostalih tehnologija poput JavaScripta ili izrade više različitih stranica posebno kreiranih za svaki od navedenih uređaja.

Sve dosad spomenuto bit će prikazano na praktičnom primjeru osobne internetske stranice s prezentacijskim radovima (*portfoliom*) u eksperimentalnom dijelu završnog rada gdje će stranica biti kreirana od samog početka i svaki će korak biti postepeno prikazan i obrazložen.

KLJUČNE RIJEČI: internetska stranica, HTML, CSS, responzivnost

ABSTRACT

This final thesis deals with HTML and CSS technologies which have been used for the creation of websites since their foundation. Emphasis is placed on the newest HTML5 and CSS3 editions and everything they have introduced or changed when it comes to the standards of the Web.

It especially deals with the problem of responsiveness of websites and the adjustment of content to every possible kind of screen size and device used for browsing the Web. Main objective of this thesis is to establish how possible it is to adjust the content of an average website to the modern devices and browsers without using other technologies like JavaScript or creating more separate websites for every type of device.

Everything already mentioned will be shown on a working example of a personal portfolio website in the experimental section of the thesis. A website will be created from the beginning and every step of the process will be shown and explained along the way.

KEYWORDS: website, HTML, CSS, responsive

SADRŽAJ

1. UVOD

Internet je javno dostupna globalna paketna podatkovna mreža koja zajedno povezuje računala i računalne mreže korištenjem internetskog protokola. Web, kao jedna od najkorištenijih usluga Interneta koja omogućava dohvaćanje hipertekstualnih dokumenata, je jedan od servisa Interneta koji služi za prezentaciju internetskih stranica pomoću HTTP (HyperText Transfer Protocol) protokola, glavne i najčešće metode prijenosa informacija na Webu.

Internetske stranice se sastoje od niza dokumenata kojima se može pristupiti uz pomoć web preglednika (*browsera*). Internetska stranica se u hrvatskom jeziku istovremeno upotrebljava i za samostalne HTML dokumente (*web page*) i za kolekciju dokumenata i resursa koji čine mnogo kompleksniji *web site*.

Internetska stranica kao HTML dokument (*web page*) omogućava prezentaciju teksta i poveznica, a dostupna je preko svoje URL adrese. Na internetskoj se stranici osim teksta mogu prikazati i multimedijski elementi kao što su slike, zvukovi, animacije i slično. Internetska stranica kao *web site* je kolekcija internetskih resursa – HTML dokumenata, multimedijskih sadržaja, podataka i skripti. Dokumentima unutar internetskih stranica se pristupa preko zajedničke korijenske adrese (internetske domene). Složenije internetske stranice koriste web aplikacije napisane u nekom od programskih jezika za stvaranje sadržaja.

Danas se, kao posljedica pojave pametnih telefona i tableta, internetske stranice više ne posjećuju samo preko preglednika na osobnim računalima. Samim time za dizajnere internetskih stranica javila se potreba prilagodbe sadržaja veličinama zaslona svih uređaja koji podržavaju pristup internetu.

Za kvalitetnu izradu modernih internetskih stranica, dizajner mora poznavati podjednako informatičku tehnologiju kao i dizajn. Pravi je web dizajner istovremeno inženjer, programer i umjetnik.

2. HTML

U najjednostavnijem smislu, internetska stranica je tekstualni dokument koji se sastoji od posebnih uputa koje govore o sadržaju stranice. Te upute su napisane HTML (HyperText Markup Language) jezikom. HTML je stoga prezentacijski jezik za izradu internetskih stranica kojim se oblikuje sadržaj i stvaraju hiperveze *hipertekst* dokumenta.

Hipertekst je tekstualna struktura koja se sastoji od međusobno povezanih jedinica informacije prikazana na nekom elektroničkom uređaju. Za razliku od tradicionalnog teksta, hipertekst nema jedinstven redoslijed čitanja, nego ga čitatelj dinamički određuje. Zato kažemo da je nesekvencijalan. Osim toga je i modularan, čime želimo naglasiti da on u određenom smislu nikada nije dovršen pošto se uvijek može dodati novi modul. On ne mora biti potpun kako bi bio funkcionalan, te se može objaviti u stanju za koje autor smatra da je prikladno.

HTML je jednostavan za uporabu i relativno se lako uči, što je jedan od razloga njegove opće prihvaćenosti i popularnosti. Prikaz hipertekst dokumenta omogućuje web preglednik. Temeljna zadaća HTML jezika je uputiti preglednik kako prikazati hipertekstualni dokument. HTML nije programski jezik pošto njime ne možemo izvršiti nikakvu zadaću poput zbrajanja, on služi samo za opis naših hipertekstualnih dokumenata.

HTML datoteke su zapravo obični tekstualni dokumenti s ekstenzijom .html ili .htm. Osnovni građevni element u HTML jeziku su *tagovi* koji opisuju kako će se nešto prikazati u web pregledniku. Poveznice unutar HTML dokumenata povezuju dokumente u uređenu hijerarhijsku strukturu i time određuju način na koji posjetitelj doživljava sadržaj internetskih stranica.

2.1. Razvoj HTML-a

1989. Tim Berners-Lee je razmišljao o problemima s kojima su se znanstvenici u CERN-u (Europska organizacija za nuklearno istraživanje) susretali tijekom dijeljenja svojih radova i rezultata istraživanja. Svaki je znanstvenik imao svoje računalo, svoj

način pisanja i program za pisanje kojeg je preferirao. Kada je netko zatražio dokument svojeg kolege trebao je ili naučiti koristiti drugo računalo ili program za pisanje ili je trebao *konvertirati* (pretvoriti, promijeniti) tu datoteku kako bi odgovarala njegovom programu. Berners-Lee je vjerovao da bi hipertekstualni sistem bio idealno rješenje tom problemu, ali su svi do tad bili prekompleksni za tako nešto stoga je započeo dizajniranje svog jednostavnog hipertekstualnog jezika baziranog na SGML-u (Standard Generalized Markup Language).

Sve je kulminiralo na Božić 1990. kada je pokrenut WorldWideWeb preglednik i server. On je dopuštao svima da objave svoje dokumente u standardnom formatu kojeg svatko kasnije može pronaći i čitati preko preglednika. HTML se u početku sastojao od dvadesetak *tagova* koji su bili označeni izlomljenim zagradama, npr. <p> je označavao te i danas označava paragrafe teksta. Ideja se brzo proširila akademskim svijetom te se pojavilo sve više preglednika i svaki je implementirao nove ideje. Uvođenje svih tih novih ideja u različite preglednike prije standardizacije zaprijetilo je povratkom na početak pošto su dokumenti lagano postajali kompatibilni samo s određenim preglednicima. Kako bi to spriječili, 1993. su Tim Berners-Lee i Dave Raggett napisali prvu probnu verziju HTML-a koju su nazvali „Hypertext Markup Language Ver 1.0“ koja je trebala postati standard.

HTML 1.0 doduše nije u tome uspio zbog prevelikog rasta broja preglednika, ali potreba za standardom nije nestajala stoga je HTML 1.0 ubrzo bio zamijenjen HTML-om 2.0. Tim Berners Lee napustio je CERN 1994. i organizarao je World Wide Web Consortium (W3C) koji se i dan danas bavi standardizacijom tehnologija korištenih na webu. Započelo je vrijeme poznato kao „*Browser Wars*“ (ratovi preglednika) gdje su se proizvođači preglednika borili za prevlast na tržištu. Najveći suparnici bili su Netscape Navigator i Microsoftov Internet Explorer koji nastavljaju s trendom implementacije različitih standarda, pa tako i uvođenjem JavaScripta koji je naposlijetku bio jedini uspješan način rješavanja problema inkompatibilnosti, ali je bio na lošem glasu zbog produljenja vremena učitavanja stranice.

W3C je pokušao opet standardizirati HTML svojom 3.0 inačicom koja je donijela mogućnost tablica. Najveća posljedica ratova preglednika je prihvaćanje specifičnih oznaka podržanih u najpoznatijim preglednicima. Tako su nastale mnoge

duplikacije, to jest više je oznaka imalo istu funkciju. Podebljani tekst je primjerice bilo moguće definirati oznakom `<b>`, ali i oznakom `<strong>`.

HTML 4 je predstavljen u prosincu 1997. te je nastavio s prihvaćanjem oznaka (*tagova*) nametnutih od strane proizvođača web preglednika, no istovremeno je pokrenuto i „čišćenje“ standarda od suvišnih oznaka. Manje promjene predstavljene su u prosincu 1999. kada je predstavljena HTML 4.01 verzija.

Internet Explorer je stagnirao nakon pobjede nad Netscapeom (koji je prestao s razvojem preglednika) i sve što se nije moglo HTML-om i CSS-om nadoknađivalo se JavaScriptom i Flashom. Doduše, zbog toga su mnogi ljudi predviđali da budućnost weba nije vezana uz HTML. Dva su bitna događaja najavila novi pristup web aplikacijama – pojava Firefox preglednika (Netscapeov nasljednik), te Google-ova email aplikacija, Gmail. Gmail se nakon učitavanja rijetko ponovno učitavao što je bila potpuna novost. Umjesto da je link vodio na novu stranicu, JavaScript je uskočio i poslao XHR zahtjev serveru i ažurirao već učitanu stranicu. XHR je jezik koji se koristi za slanje http ili https zahtjeva serveru i učitavanje odgovora u skriptu. Iako nije bila prva web aplikacija koja je koristila XHR ili slične tehnike, Gmail je obnovio interes u JavaScript i započeo trend XHR-baziranih web aplikacija.

Za to vrijeme W3C je radio na mnogim drugim standardima, ali i na novoj inačici HTML-a. Odlučili su da budućnost leži u XML-u (Extensible Markup Language) koji je bio jako sličan HTML-u osim u dvije bitne razlike: greške nikada ne prolaze neprimjetno što je sa sobom nosilo mnogo stroža pravila, te nadogradivost u smislu da se mogu dodati novi elementi koje se samo treba opisati u odvojenoj datoteci i povezati tu datoteku sa svojim glavnim dokumentom. 2000. je W3C kao standard predstavio XHTML, poboljšanu verziju HTML-a baziranu na XML-u.

2.2. HTML 5

HTML 5 je nova revizija standarda HTML-a, nasljednik inačice HTML 4.01. Nastao je u suradnji World Wide Web konzorcija (W3C) i WHATWG Grupe (Web Hypertext Application Technology Group) koje su do 2006. radile odvojeno (WHATWG s web formama i aplikacijama, a W3C s XHTML 2.0). Glavni cilj bio je napredak jezika u smislu podrške najnovijih multimedijskih standarda dok istovremeno

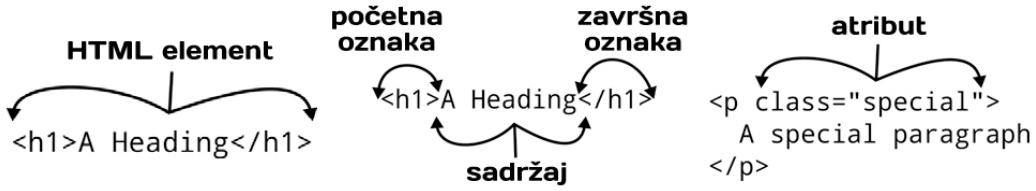
ostaje jednostavan za čitanje ljudima i razumijevanje računalima i drugim uređajima koji podržavaju pregledavanje internetskih stranica. HTML 5 je uveo nove sintakse poput `<video>`, `<audio>` i `<canvas>` elemenata, ali i integrirao je SVG (Scalable Vector Graphics) za vektorske grafike na webu te MathML za matematičke formule.

Iako su neke opcije HTML 5 često uspoređivane s Adobe Flash-om, te dvije tehnologije su jako različite. Obje dopuštaju audio i video datoteke i SVG, ali HTML5 sam po sebi ne može biti upotrebljen za animacije ili interaktivnost, već uz sebe treba biti dopunjen drugim tehnologijama poput JavaScripta ili CSS3 tehnologije. Čak i tada nije moguće realizirati sve što se može pomoću Adobe Flash-a. Unatoč tome, Apple je među prvima onemogućio Flash u svojim iPhone pametnim telefonima zbog čestih problema koje je Flash imao s internet preglednicima, a danas i najpopularniji preglednici poput Chrome-a i Firefox-a blokiraju Adobe Flash.

28.10.2014. HTML 5 je službeno postao standardom, iako se već u 2012. objavila i prva „skica“ inačice HTML 5.1. koja će u 2016. nakon posljednjih preinaka postati preporučenim standardom.

2.3. Struktura HTML dokumenta

Svaki se HTML dokument sastoji od elemenata, a svaki se element sastoji od oznaka (*tagova*). Oznaka je dio koda koji služi za razgraničavanje, a sastoji se od izlomljenih zagrada i alfanumeričkih znakova. Većina elemenata se sastoji od početne i završne oznake (završna oznaka ispred svojeg imena ima i kosu zagradu) unutar kojih se nalazi sadržaj elementa, iako se određeni elementi sastoje samo od početne oznake. Osim toga element može imati i atribute kojima se definiraju svojstva tog elementa, a atribut se sastoji od imena nakon kojeg slijedi znak jednako te vrijednost atributa unutar navodnika, kao što je prikazano na slici 1.

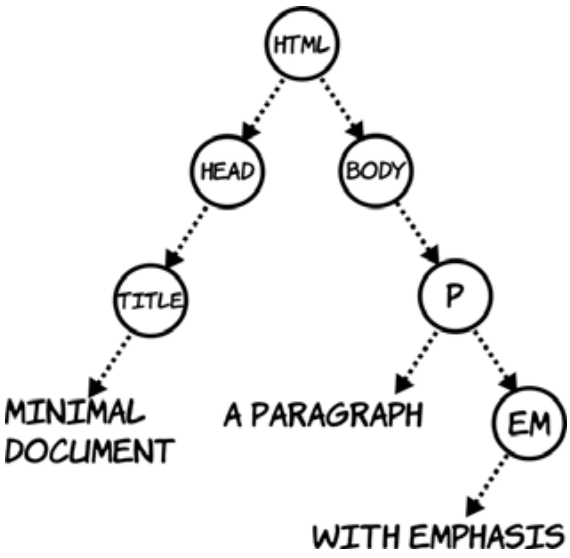


Slika 1. Dijelovi HTML elementa

Neki elementi moraju imati barem jedan atribut kako bi imali svrhu, npr. `<link>` element mora imati atribut koji sadrži adresu stranice na koji poveznica vodi. Dva najčešća atributa su `id` (jedinstveni identifikator) i `class` (klasa, kategorija).

Elementi najčešće sadrže tekst, iako mogu sadržavati i druge elemente. Svaki element koji sadrži neke druge elemente je roditelj (*parent*) svim elementima koje sadrži pošto elementi formiraju strukturu sličnu obiteljskom stablu. Tako je i svaki HTML dokument „obiteljsko stablo“, kao što je prikazano na slici 2., pošto započinje `<html>` oznakom koja u sebi sadržava `<head>` i `<body>` elemente. Sve unutar `<html>` elementa jest HTML kod. Unutar `<head>` elementa nalazi se sve što nije direktan, ali je i dalje vrlo bitan sadržaj internetske stranice (poput naslova stranice, CSS stilova ili ključnih riječi koje mogu biti korisne kod pretraživanja), a unutar `<body>` elementa se nalazi sve što nam preglednik prikazuje u svom prozoru.

```
<html>
<head>
<title>Minimal
document</title>
</head>
<body>
<p>A paragraph
<em>with emphasis</em></p>
</body>
</html>
```



Slika 2. Prikaz obiteljskog stabla jednostavne HTML internetske stranice

2.4. HTML elementi

Osim već spomenutih `<html>`, `<head>` i `<body>` elemenata koji su sastavni dijelovi internetskih stranica postoji mnogo drugih elemenata koji se koriste po potrebi.

Za označavanje teksta postoje dvije vrste elemenata: naslovi i paragrafi. Postoji šest razina naslova, od `<h1>` do `<h6>`. `<h1>` se generalno koristi za glavne naslove,

<h2> za podnaslove i tako dalje. Paragrafi se označavaju oznakom. Naslove i paragrafe potrebno je zatvoriti završnim oznakama (</h1> ili).

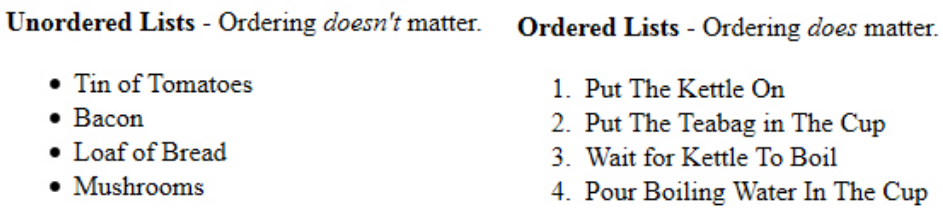
Kako bismo tekst formatirali drugačije možemo se služiti s nekoliko opisnih elemenata: <b> element će podebljati sav tekst koji se nalazi unutar početne i završne oznake, dok će <i> tekst prikazati u kurzivu. <sup> se koristi za potencije, a <sub> za indekse.

Svaki paragraf će automatski biti prikazan u novom redu s razmakom u odnosu na protekli paragraf. Ukoliko želimo prebaciti u novi red tekst unutar paragrafa ili naslova koristimo
 element. On je prazan element, dakle sastoji se samo od početne oznake i ne zahtijeva završnu pošto ionako nema nikakav sadržaj. <hr> element dodaje horizontalnu liniju i također je prazan element.

Postoje i elementi čija svrha nije utjecati na izgled ili strukturu internetskih stranica i to su tzv. semantički (*semantic*) elementi. Npr. element naglašava dio teksta i preglednici često prikazuju sadržaj tog elemenata drugačije (najčešće u kurzivu). Doduše glavni cilj elementa je naglasiti taj dio teksta kao bitan tražilicama ili npr. čitačima ekrana što je vrlo bitno kod slijepih osoba koje ovise o takvim čitačima. element ima sličnu svrhu, doduše naglasak je još jači nego kod elemenata. <blockquote> služi za citate koji mogu zauzeti cijeli paragraf, a <q> služi za kraće citate koji se nalaze unutar paragrafa. Za akronime ili skraćenice koristi se <abbr> element koji sadrži title atribut čija je vrijednost pun naziv. Kada se mišem pređe preko dijela teksta koji je unutar <abbr> elementa pojavi se vrijednost title atributa. <dfn> se koristi kada se prvi put spominje neka nova terminologija i služi kao oznaka da se na tom mjestu definira neki novi pojam. <address> element označava podatke o autoru internetske stranice ili članka. Ukoliko želimo prikazati promjene u sadržaju stranice koristimo <ins> za novi sadržaj koji smo umetnuli, <del> za sadržaj koji više ne vrijedi i <s> za sadržaj koji također ne vrijedi, ali ga ne želimo izbrisati (npr. stara cijena pored nove unutar nekog web shopa).

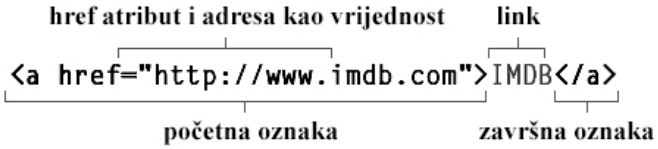
Postoji mnogo situacija kada se u HTML-u koriste liste, a poznajemo uređene i neuređene liste, te liste za definiranje. Uređene koristimo kada nešto ima određeni redoslijed, a neuređene koristimo za nabrajanje bez redoslijeda. element označava uređene (*ordered*) liste, a <ul> neuređene (*unordered*). Oni funkcioniraju

kao roditelj (*parent*) stavkama liste. Razlika između ove dvije vrste listi u pregledniku vidi se na slici 3. Svaka stavka liste se označava `<li>` elementom (neovisno da li je lista uređena ili neuređena). Treća vrsta listi je lista koja se koristi za definiranje. Kod njih je `<dl>` element koji označava samu listu, `<dt>` označava pojam koji se definira, a `<dd>` sadrži definiciju pojma i nalazi se neposredno nakon `<dt>` elementa.



Slika 3. Neuređena (*unordered*, lijevo) i uređena (*ordered*, desno) lista

Linkovi su jedni od glavnih opcija weba jer dozvoljavaju prijelaz s jedne na drugu internetsku stranicu te tako omogućavaju samu ideju surfanja. Linkovi se kreiraju pomoću `<a>` elementa. Sve što se nalazi unutar oznaka jest ono što će na samoj stranici biti prikazano i funkcionirati kao link, dakle klikom na to (bilo to tekst ili npr. slika) korisnik će biti preusmjeren na adresu linka. Adresa se specificira kao vrijednost `href` atributa. Ukoliko link vodi na neku drugu stranicu koristimo apsolutni URL (Uniform Resource Locator) link kao adresu. S druge strane, ako link vodi na neku podstranicu iste internetske stranice koristimo relativne URL linkove. To su skraćene verzije apsolutnih linkova jer nije potrebno specificirati domenu (npr. vrijednost `href` atributa je samo `contact.html`). HTML kod `<a>` elementa objašnjen je na slici 4.



Slika 4. Link u HTML-u; `<a>` element

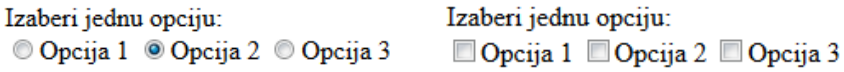
Ukoliko želimo da link otvori korisnikov program za slanje e-pošte vrijednost href atributa će započeti s `mailto:` te nakon dvotočke upisujemo adresu na koju želimo da dođe elektronička pošta. Moguće je link otvoriti i u drugom prozoru, a to ostvarujemo pomoću `target` atributa kojemu vrijednost postavimo na `_blank`. Ako s

druge strane želimo da naš link vodi na određeni dio naše stranice to možemo učiniti preko `id` atributa. U tom se slučaju nekom dijelu stranice (npr. paragrafu) uvede atribut `id` s nekom vrijednošću (npr. `paragraf1`), a linku se kao vrijednost `href` atributa postavi vrijednost `id` atributa (u našem slučaju `href="#paragraf1"`).

Neki sadržaji internetskih stranica poput sportskih rezultata ili rasporeda tramvajskih stanica se prikazuju u tablicama. Tablice se sastoje od stupaca i redova. U HTML-u tablice označavamo `<table>` elementom unutar kojeg se može sastojati bezbroj `<tr>` i `<td>` elemenata. `<tr>` elementi predstavljaju redove, dakle ukoliko imamo samo jedan stupac ne trebamo koristiti `<td>` elemente koji predstavljaju stupce. Ali ako želimo imati više stupaca tada moramo unutar `<tr>` elementa uvrstiti `<td>` elemente. `<th>` element je gotovo jednak `<td>` elementu, ali se koristi za naslove stupaca i redaka. U slučaju da polje nema nikakvu vrijednost ipak se treba pisati `<td>` ili `<th>` elemente kako bi se polje točno prikazalo kao prazno. Redove se može proširiti (da zauzmu površinu dva ili više prazna polja) pomoću `colspan` atributa, a stupce pomoću `rowspan` atributa. Vrijednost tih atributa se postavlja na željen broj polja koje bi redak ili stupac trebao zauzeti i koristi se samo kod `<td>` i `<th>` elemenata.

Forme su korisne zbog toga što omogućavaju prikupljanje informacija od posjetitelja internetske stranice. Mogu služiti za unos teksta, odabir između više opcija i slanje podataka. Forme označavamo `<form>` elementom koji bi uvijek trebao imati `action` atribut čija je vrijednost URL stranice na serveru koja će primiti informacije unutar forme kada je ona poslana. Metoda primanja podataka iz forme označava se `method` atributom unutar `<form>` elementa i postoje dvije moguće vrijednosti: `get` za kratke forme i `post` za naprednije forme koje omogućavaju *upload* datoteka ili sadrže osjetljive podatke poput lozinki. `<input>` je prazan element i koristi se za unos podataka. Ukoliko je vrijednost `type` atributa `text` tada element služi za unos teksta, a ukoliko je vrijednost `type` atributa `password`, tada element služi za unos lozinke. Vrijednost tog atributa može biti i `radio` ili `checkbox`, a razlika između ove dvije opcije je prikazana na slici 5. Drugi atribut koji se često pojavljuje kod unosa podataka je `maxlength` čija je vrijednost broj koji označava koliko maksimalno znakova

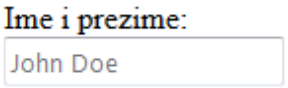
korisnik može unijeti. <textarea> se također koristi za unos teksta, ali ovog puta multilinijski.



Slika 5. Type atribut i neke moguće vrijednosti: radio (lijevo) i checkbox (desno)

<select> element se koristi za padajući izbornik i sadrži <option> elemente koji služe kao opcije na izborniku. Poželjno je sadržaj <option> elementa nazvati jednako kao i vrijednost value atributa jer je vrijednost atributa ono što se šalje serveru, a sadržaj je ono što korisnik vidi kao opciju na listi. Drugi atribut koji se koristi kod <option> elementa je selected kojim možemo odrediti jednu od stavki kao onu koja je odabrana pri učitavanju stranice. U tom slučaju se vrijednost atributa postavlja na selected. Ukoliko ne koristimo taj atribut tada se prva stavka na listi postavlja kao odabrana pri učitavanju. Za upload datoteka koristimo već spomenuti <input> element koji ima type atribut, a vrijednost se u tom slučaju postavlja na file. HTML tada kreira polje koje izgleda kao polje za unos teksta nakon kojeg se pojavljuje *browse* gumb (za pretraživanje), a klikom na gumb otvara se prozor koji omogućava pronalazak i odabir željene datoteke na računalu. <button> element je bio uveden kako bi dizajnerima omogućio dizajniranje samog gumba. Elementi unutar <form> elementa mogu se grupirati u više grupa pomoću <fieldset> elementa koji ih odvaja i djeluje kao roditelj elementima koji se nalaze unutar njega. Direktno nakon otvaranja <fieldset> elementa možemo otvoriti i <legend> element koji služi kao identifikator grupe te se pojavljuje poput naslova prije ostalih elemenata unutar grupe.

HTML 5 je uveo mnogo novih korisnih vrijednosti za type atribut <input> elementa poput date za datume, email za e-poštu, url za adrese internetskih stranica i search za pretragu. Također je uveden i atribut placeholder čija vrijednost je tekst koji se nalazi u polju za unos teksta prije nego što korisnik klikne na njega (primjer na slici 6).



Slika 6. Atribut placeholder kojemu je vrijednost John Doe

Jedan od elemenata koji je prije bio puno kompliciraniji za definiranje je `<!DOCTYPE>` element. Pomoću njega se označavalo koja se inačica HTML-a koristi u dokumentu, iako bi se stranica prikazivala i bez tog elementa, doduše možda nepravilno. U HTML 5 se taj element definira na početku samog koda i to vrlo jednostavno kao `<!DOCTYPE html>`.

Komentari se u HTML kodu ostavljaju pomoću `<!--` i `-->` znakova. Sve što je unutar tih znakova neće biti vidljivo posjetitelju stranice, a sami komentari su vrlo korisni zbog toga što čine kod razumljivijim u slučaju da se vratite kodiranju stranice nakon nekog vremena ili niste jedina osoba koja radi na stranici.

Elementi u HTML-u mogu biti *block* (blokovi) ili *inline* (u redu) elementi. *Block* elementi uvijek započinju u novom redu (npr. `<h1>`, `<p>` ili `<ul>`) dok se *inline* elementi nastavljaju u istom redu kao susjedni elementi (npr. `<a>`, `<b>` ili `<em>`). Elemente se također može grupirati unutar *block* ili *inline* elemenata. Ukoliko želimo elemenete grupirati unutar *block* elementa koristimo `<div>` element. `<span>` koristimo za grupiranje elemenata *inline*.

`<iframe>` element koristimo ukoliko unutar naše stranice želimo prikazati drugu stranicu. Jedan od najboljih primjera gdje se to koristi je prikaz Google Maps karte s određenom lokacijom unutar stranice koju trenutno posjećujemo. Za točan prikaz <iframe> elementa trebamo nekoliko atributa, a to su `src` za specificiranje željene URL adrese, te `height` i `width` (visina i širina).

Informacije o stranici se nalaze unutar `<head>` elementa. Tu koristimo prazni `<meta>` element koji nije vidljiv posjetitelju stranice, ali izvršava razne uloge pomoću svojih atributa. Neki od najkorištenijih atributa su `description` za opis stranice, `keywords` za ključne riječi (vrlo bitno kod tražilica) i `author` za vlasnika internetske stranice. Također se koriste i `<link>` elementi koji upućuju na neke vanjske resurse poput CSS-a te `<script>` elementi koji specificiraju kod kojeg browser treba pokrenuti.

Novi značajni elementi koji su uvedeni u HTML 5 inačici su `<header>` za vrh stranice, `<nav>` za navigaciju, `<section>` za određene odjeljke stranice, `<article>` za članke na stranici (funkcionira slično kao i `<section>`), `<main>` za glavni dio stranice unutar kojeg se nalaze odjeljci i članci, `<aside>` koji se koristi za

sadržaj koji je na strani (nije glavni) te `<svg>` koji se koristi za uvođenje SVG (Scalable Vector Graphics) tehnologije u internetsku stranicu.

2.5. Multimedijski sadržaji

Slike dodajemo pomoću praznog `<img>` elementa. `<img>` element mora sadržavati `src` atribut koji će pregledniku reći gdje se slika nalazi (najčešće je to relativni URL i baš zbog toga je dobra praksa svakog dizajnera da kreira zasebnu mapu za slike koje će se na samoj stranici prikazivati). Ostali atributi koji se koriste su `alt` koji daje detaljan opis slike u slučaju da se ona ne može učitati (ili u slučaju čitača ekrana za slijepe osobe) i `title` koji daje naslov slici te za vrijeme prelaska preko slike mišem se isti taj naslov pojavljuje kao *tooltip* (oblačić). Visinu i širinu slike definiramo `height` i `width` atributima čija je vrijednost broj u pikselima, iako se to danas sve češće radi pomoću CSS-a. Za vektorsku grafiku koristimo već spomenuti `<svg>` element.

Video se dodaje pomoću `<video>` elementa koji ima mnogo atributa koji se koriste za kontrolu samog videa: `src` specificira gdje se video nalazi, `poster` omogućava odabir slike koja je prikazana dok se video skida ili dok korisnik ne pokrene video, `width` i `height` kontroliraju visinu i širinu, `autoplay` pokreće video pri učitavanju stranice, `loop` označava konstantno ponavljanje videa i na posljetku postoji `preload` koji pregledniku govori što učiniti pri učitavanju stranice. To je atribut kojim možemo omogućiti učitavanje videa istovremeno kada se učitava i sama stranica, čak i prije pokretanja videa. `<source>` element se nalazi unutar `<video>` elementa ukoliko imamo više formata i u tom slučaju ne koristimo `src` atribut unutar `<video>` elementa.

Glazbu možemo dodati preko `<embed>` elementa ukoliko je sama datoteka već uploadana na nekoj drugoj stranici ili pomoću `<audio>` elementa koji je uveden u petoj inačici HTML-a. Atributi koje koristi `<audio>` element su većinom jednake atributima koje koristi `<video>` element samo nemamo `poster`, `width` i `height` atribute.

3. CSS

CSS (Cascading Style Sheets) je jezik kojim opisujemo izgled i formatiramo dokumente napisane pomoću *markup* jezika (npr. HTML, XML i SVG dokumenti). Osmišljen je primarno kako bi se napravila separacija izgleda od sadržaja samog dokumenta te kako bi se omogućila bolja fleksibilnost, snalaženje unutar dokumenta i kontrola izgleda dokumenta, a najveći je plus to što je moguće formatirati više dokumenata na jednak način (cijela internetska stranica sa svim svojim podstranicama ujednačeno je definirana pomoću jedne .css datoteke).

Ta ista separacija formatiranja i sadržaja omogućuje prikaz istog markup dokumenta pomoću različitih stilova za različite potrebe (računalo, smartphone, tisak, uređaji s Braillovim pismom ili čitači ekrana). Isto tako, ukoliko se želi promijeniti neka vrijednost (npr. povećati se veličina fonta koji se koristi za `<h1>` naslove) to se može učiniti vrlo lagano i promjena će se odnositi na sve `<h1>` elemente. Prije CSS-a se trebalo proći kroz sve datoteke i u njima pronaći sve te elemente i mijenjati ih jedan po jedan.

CSS također specificira način na koji odlučuje koji se stil odnosi na određeni element ukoliko je jedno pravilo kontradiktorno drugom. Stilovi se primjenjuju u kronološkom redoslijedu, ali i prema specifičnosti. Dakle ako je neka vrijednost deklarirana više nego jednom, zadnje deklarirana vrijednost se koristi osim u slučaju kad je neka deklaracija bila specifičnija.

CSS se može pisati direktno unutar HTML elementa pomoću `style` atributa (*inline* način), interno unutar `<style>` elementa koji se nalazi unutar `<head>` elementa u HTML dokumentu ili eksterno. Ukoliko koristimo eksterni način (CSS je zasebna .css datoteka) CSS moramo povezati s našim HTML dokumentom preko sljedeće sintakse: `<link href="css/style.css" type=“text/css“ rel="stylesheet">`. `Link` elementom govorimo HTML dokumentu gdje se nalazi CSS datoteka pomoću `href` atributa (adresa), `type` atribut mu govori da se radi o css datoteci, te naposljetku imamo `rel` atribut koji specificira odnos između HTML dokumenta i dokumenta na koji link vodi.

3.1. Razvoj CSS-a

CSS je prvi predložio Håkon Wium Lie 10. listopada 1994. godine, a CSS1 je postao preporuka W3C konzorcija dvije godine kasnije, u prosincu 1996. Njime su se mogli odrediti fontovi, boje teksta i pozadine, razmak između riječi, slova i linija teksta, pozicioniranje teksta, slika i tablica, margine, *border* (rubovi, okviri) i *padding*. Tek tri godine kasnije kreiran je prvi preglednik koji je gotovo u cijelosti podržavao tu inačicu.

4. studenog 1997. kreiran je CSS2 koji je trebao biti rješenje za mnoge probleme s kojima se suočavao CSS1. On je postao preporuka u svibnju sljedeće godine. U toj inačici uvedene su opcije poput apsolutnog i relativnog pozicioniranja te z-index (z os koordinatnog sustava) i sjene na tekstu. Iste godine je počeo rad na trećoj inačici CSS-a, koja se i danas koristi. Ona je, za razliku od svojih prethodnika, odvojena na više modula. Svaki modul dodaje nove mogućnosti ili unapređuje staru verziju odvojeno od ostalih, tako da svaki modul ima drugačiju stabilnost i status. Trenutno ima preko pedeset modula koji su objavljeni od strane CSS grupe, ali samo četiri imaju formalnu preporuku. Neki moduli poput Pozadina i okvira kandidati su za preporuku te se stoga smatraju stabilnima.

Ne postoji potpuna CSS4 inačica pošto je svaki modul postepeno i odvojeno razvijen, ali postoje moduli koji su već na četvrtoj razini razvoja. Dakle, iako samostojeća verzija CSS4 neće biti objavljena niti postati preporuka, svi moduli koji su na četvrtoj razini kolektivno se nazivaju CSS4 modulima.

3.2. CSS sintaksa

Slika 7. CSS sintaksa

CSS dokument se sastoji od pravila, a svako pravilo se sastoji od selektora i liste deklaracija koje se nalaze unutar vitičastih zagrada. Svaka deklaracija ima svojstvo i

vrijednost svojstva. Svi navedeni pojmovi su prikazani na slici 7. Točka sa zarezom označava kraj deklaracije.

Pomoću selektora određujemo na koji dio HTML-a se stil odnosi. Selektori mogu biti elementi (npr. naslovi), specificirani elementi (npr. svi naslovi druge razine, `h2`) ili elementi specificirani atributom (najčešće jedinstvenim identifikatorom `id` ili klasom koju može imati više elemenata na stranici). Selektori su detaljnije pojašnjeni uz primjere unutar tablice 1. Atributi `class` i `id` kao identifikatori moraju započeti sa slovnim znakom i mogu sadržavati samo slova engleske abecede, brojeve i podvlake.

Selektor	Značenje	Primjer
Univerzalni selektor	Odnosi se na sve elemente u dokumentu	`* {}`
Selektor elementa	Odnosi se na točno taj element	`p {}`
Class selektor	Odnosi se na sve elemente s atributom class čija vrijednost je jednaka odabranoj (u drugom primjeru samo na `p` elemente koji uz to imaju i klasu `plavi`)	`.plavi {}` `p.plavi {}`
Id selektor	Odnosi se na element s atributom `id` čija vrijednost je jednaka odabranoj (`prvi`)	`#prvi {}`
Selektor djeteta	Odnosi se na element koji je dijete definiranog elementa (npr. svi `a` elementi unutar `li` elemenata, ali ne i svi ostali `a` elementi u dokumentu)	`li>a {}`
Selektor nasljednika	Odnosi se na elemente koji su nasljednici definiranog elementa (npr. bilo koji `a` element razgranat unutar `p` elementa)	`p a {}`
Selektor naknadnog brata	Odnosi se samo na element definiran direktno nakon određenog elementa (npr. odnosi se samo na prvi `p` nakon `h1` elementa, ali ne i naknadne `p` elemente)	`h1+p {}`
Generalni selektor naknadne braće	Odnosi se na elemente koji su definirani nakon određenog elementa (npr. odnosi	`h1~p {}`

	se na sve `p` elemente koji su definirani nakon `h1` elementa u dokumentu, a nisu mu nasljednici već „braća“)	
Atribut selektor postojanja	Odnosi se na element s atributom, neovisno o vrijednosti atributa	`p[class] {}`
Atribut selektor jednakosti	Odnosi se na element s atributom točno određene vrijednosti	`p[class=“dog“]`
Atribut selektor razmaka	Odnosi se na elemente s atributom koji ima više vrijednosti odvojenih razmakom od kojih je jedna definirana u selektoru (npr. ovaj se odnosi na sve p elemente koji u klasi sadrže vrijednost d1)	`p[class~=“d1“]`
Atribut selektor prefiksa	Odnosi se na elemente s atributom čija vrijednost započinje definiranim znakovima	`p[attr^“d“]`
Atribut selektor podniza (*substring*)	Odnosi se na elemente s atributom čija vrijednost sadrži definirane znakove	`p[attr*“do“]`
Atribut selektor sufiksa	Odnosi se na elemente s atributom čija vrijednost završava definiranim znakovima	`p[attr$“g“]`

Tablica 1. Lista selektora s definicijama i primjerima

Pseudoklase se koriste u selektorima da omoguće formatiranje informacija koje se ne nalaze u obiteljskom stablu dokumenta. Jedan od najboljih primjera pseudoklase je `:hover`, koji se primjenjuje samo kada korisnik prijeđe mišem preko vidljivog elementa (u CSS-u se direktno nadovezuje na element na koji se odnosi, npr. `a:hover` označava da će se deklaracije odnositi samo na linkove kada korisnik preko njih prijeđe mišem). Drugi primjeri su `:active` koji se primjenjuje kada je element aktiviran od strane korisnika (npr. neki *button* na stranici), `:focus` označava da je fokus na tom elementu (često se koristi kod formi), `:link` se koristi za još neposjećene linkove, a `:visited` za linkove na koje je korisnik kliknuo (te se stoga smatraju posjećenima).

3.3. Prioriteti i ograničenja u CSS-u

Kao što je prije spomenuto, CSS specificira način na koji odlučuje koja se pravila odnose na koji element ukoliko dolazi do kontradikcije između dva ili više pravila. Tablica 2. prikazuje koje pravilo nadvladava koje u slučaju kontradikcije.

Prioritet	Vrsta CSS-a	Opis
1	Važnost	`!important` pribilješka pokraj pravila nadvladava ostale prioritete
2	*Inline*	Stil definitan unutar HTML elementa pomoću `style` atributa
3	Media tip	Definicija vrijedi samo za taj media tip, za sve ostale vrijedi opća deklaracija
4	Definirano od strane korisnika	Korisnik može definirati svoj CSS u pregledniku koji će nadjačati neka CSS pravila
5	Određeniji selektor	Određeniji selektor nadjačava opća pravila (npr. div s klasom ima svojstva definirana klasom, a ne općenita svojstva div elementa; isto tako `id` ima prioritet nad `class` atributom pošto je specifičniji)
6	Poredak pravila	Zadnja deklaracija nadjačava prijašnje
7	Nasljedstvo od roditelja	Ako svojstvo nije definirano preuzima se od roditelja
8	CSS svojstvo	Definirani CSS nadvladava CSS preglednika
9	Zadano od strane preglednika	Najniži prioritet, osnovne vrijednosti svojstava koje je definirao W3C (npr. osnovna boja paragrafa je crna)

Tablica 2. CSS pravila prioriteta u slučaju kontradikcije

Jedno od najčešće kritiziranih ograničenja u CSS-u je to da trenutno ne nudi mogućnost selektiranja roditelja ili pretka, iako nudi obrnut proces (izbor djeteta ili nasljednika pomoću roditelja ili pretka). Osim toga, horizontalno pozicioniranje elemenata je generalno lagano kontrolirati, dok je vertikalno pozicioniranje često

nemoguće većinom zbog problema koje preglednici imaju s podržavanjem CSS-a. Također je nemoguće specificirati vrijednosti pomoću jednostavnih matematičkih izraza (npr. `margin-left: 10% - 3em + 4px;`), iako se radi na skici za `calc` vrijednost koja bi trebala omogućiti nešto slično tome.

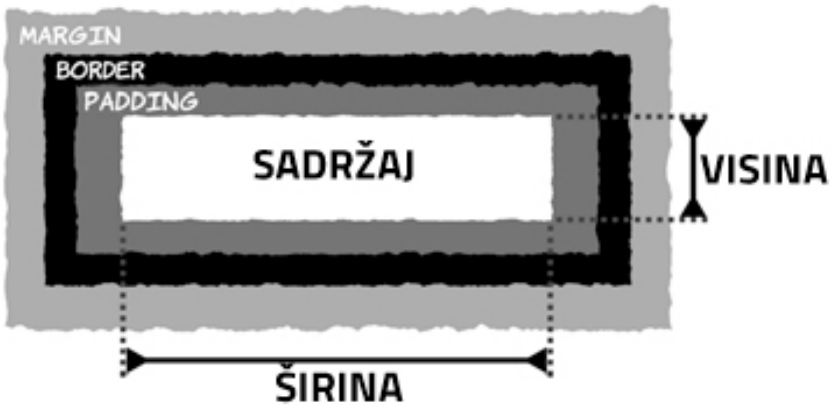
3.4. CSS deklaracije

Boja je najčešće svojstvo koje se deklarira pomoću CSS-a. Boju možemo odrediti na više načina: pomoću RGB vrijednosti (upisuje se koliki udio crvene, zelene i plave postoji u željenoj boji npr. `color: rgb(100,100,90);`), heksadekadskog koda (npr. `color: #ee3d80;`) i naziva boje na engleskom jeziku (npr. `color: DarkCyan;`). Prozirnost se može definirati odvojeno pomoću `opacity` svojstva (npr. `opacity: 0.5;` označava 50% prozirnosti) ili pomoću `rgba` vrijednosti boje (npr. `color: rgba(0,100,90,0.5);` gdje četvrti broj označava prozirnost). Boja pozadine se definira pomoću `background` (ili `background-color`) svojstva, a ne `color` svojstva.

Tekst oblikujemo pomoću više svojstava: `font-family` omogućava odabir pisma (npr. `font-family: Georgia;`), `font-style` može biti `italic` ili `oblique` (tekst ukošen pod određenim kutem, većina „italic“ tekstova na webu je zapravo oblique), `font-weight` nam omogućuje da definiramo debljinu teksta (npr. `light`, `medium`, `bold`, `black` ili numeričke vrijednosti), a pomoću `font-size` određujemo veličinu teksta u raznim mjernim jedinicama. Mjerne jedinice koje CSS prepoznaje su pikseli (px), točke (pt), centimetri (cm), inči (in), emovi (em) i postotci (%). Pomoću `text-transform` možemo tekst prebaciti u `uppercase` (verzali), `lowercase` (kurenti) ili `capitalize` (prvo slovo veliko). Možemo i dekorirati tekst pomoću `text-decoration` svojstva. Vrijednost `none` označava da tekst nema nikakvu dekoraciju, `underline` podcrtava tekst, `overline` dodaje liniju iznad teksta, a `line-through` precrtava tekst. Prored (*leading*) se definira s `line-heigth`, razmak između slova (*kerning*) s `letter-spacing`, a razmak između riječi s `word-spacing` svojstvom. Pomoću `text-align` svojstva definiramo poravnavanje teksta, a moguće vrijednosti su `left` (lijevo), `right` (desno), `center`

(centrirano) i `justify` (obostrano). `Text-shadow` je relativno novo i ne još potpuno podržano svojstvo koje tekstu daje sjenu. Definira se pomoću tri brojčane vrijednosti (od koje prva označava koliko je sjena udaljena u horizontalnom smjeru, druga u vertikalnom smjeru, a treća označava zamagljenje) i četvrte vrijednosti koja označava boju sjene (npr. `text-shadow: 1px 1px 3px #000000;`).

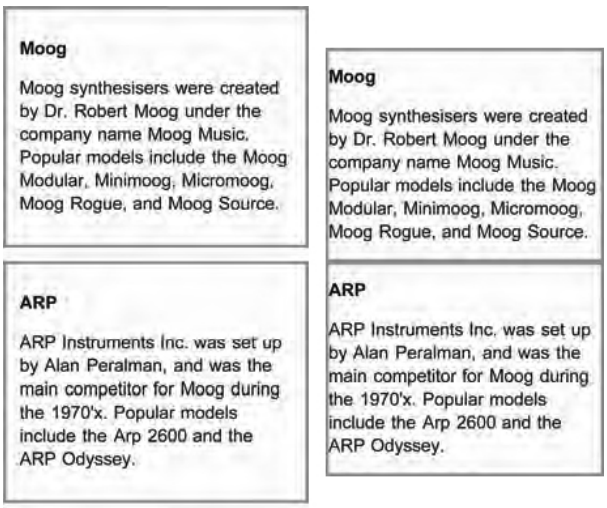
Pošto CSS elemente tretira kao da svaki postoji u zamišljenoj „kutiji“ (*box elements*) moguće je kontrolirati mnoga svojstva tih kutija. Visina i širina se definiraju pomoću `width` i `height` svojstava čije su vrijednosti u ranije spomenutim mjernim jedinicama (najčešće pikselima ili postocima). Mnoge stranice formatiraju svoj sadržaj ovisno o veličini ekrana ili preglednika i zbog toga postoje `min-width` i `max-width` svojstva. Prvo svojstvo služi da se zamišljena kutija nikada ne prikaže manja od zadane numeričke vrijednosti, a drugo da se nikada ne proširi više od zadanog (što npr. blokira sadržaj da se proširi u preduge linije i tako smanji čitljivost teksta). One kontroliraju sadržaj horizontalno, ali postoje i `min-height` te `max-height` koje čine istu stvar, ali vertikalno (dakle kontroliraju visinu, a ne širinu elemenata). U slučaju da je sadržaj elementa prevelik za element koji sadrži sadržaj koristimo `overflow` svojstvo koje ima dvije vrijednosti: `hidden` (skriva sav višak teksta) ili `scroll` (dodaje kliznu traku). Svaka „kutija“ ima tri svojstva koja mogu biti regulirana kako bi se promijenio izgled same kutije, a slika 8 prikazuje spomenuta svojstva.



Slika 8. Svojstva *box* elemenata: margine, rubovi (*border*) i *padding*

Ukoliko definiramo visinu i širinu elementa sva tri svojstva se nadodaju na tu visinu i širinu. Svaka „kutija“ ima svoj rub (eng. *border*, čak iako je nevidljiv ili velik 0 piksela). Margina je prostor s vanjske strane ruba koju koristimo kako bi povećali

razmak između dva elementa. *Padding* je s druge strane prostor koji se nalazi imeđu ruba i sadržaja kutije. Njima možemo povećati čitljivost sadržaja, što možemo vidjeti na slici 9. gdje se u lijevom primjeru koristi *padding* za stvaranje prostora između sadržaja i ruba te margina za stvaranje prostora između dva elementa, a na desnoj strani je prikaz istih elemenata bez upotrebe ovih svojstava.



Slika 9. Upotreba margina i *paddinga* za poboljšavanje čitljivosti sadržaja

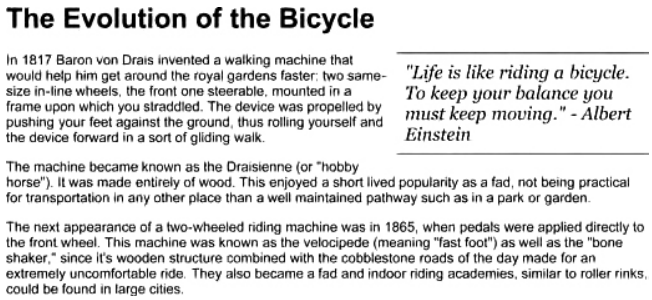
Na rubove možemo utjecati s mnogo svojstava: širinu kontroliramo `border-width` svojstvom, način na koji se rub prikazuje `border-style` svojstvom (neki od vrijednosti su `solid` (ravna crta), `dotted` (točkasto), `dashed` (isprekidane linije), `double` (dvostrukim ravnim linijama) te `hidden` ili `none` koji označavaju da se rub ne vidi, to jest ne postoji), boju `border-color` svojstvom. Jednostavniji način jest zapisivanje svih svojstava u jednoj liniji koda pomoću `border` svojstva (npr. `border: 3px dotted black;`). Padding i margine funkcioniraju slično i mogu se definirati svaki razmak zasebno (iznad (`top`), desno (`right`), ispod (`bottom`) i lijevo (`left`)) ili u jednoj liniji koda. Ukoliko npr. želimo definirati samo marginu iznad elementa koristimo `margin-top` svojstvo i vrijednost je neka numerička vrijednost s mjernom jedinicom. Ukoliko želimo sve definirati u jednoj liniji koda tada moramo imati četiri vrijednosti (jedna za svaki razmak, npr. `padding: 10px 5px 3px 1px;` bi značilo da element ima padding iznad velik 10 piksela, desno 5 piksela, ispod 3 piksela i lijevo 1 piksel, dakle vrijednosti se primjenjuju u smjeru kazaljke na satu). U slučaju da su nam gornja i donja, te lijeva i desna vrijednost jednake možemo

upisati samo dvije vrijednosti od kojih bi se prva odnosila na razmak gore i dolje, a druga na razmak lijevo i desno. Isto tako, u slučaju da su sve vrijednosti *paddinga* ili margina jednake možemo upisati samo jednu vrijednost koja se tada odnosi na sve razmake. Centriranje nekog *box* elementa na stranici ili unutar nekog drugog elementa postiže se definiranjem širine i postavljanjem lijeve i desne margine na vrijednost `auto`. CSS3 je dodao nekoliko novih svojstava, jedno od njih je `box-shadow` koje funkcionira slično kao `text-shadow` svojstvo samo stvara sjenu okolo box elementa, a ne okolo teksta. Drugo, relativno novo svojstvo jest `border-radius` koji je uveo opciju da se rubovi elementa zaoble. Vrijednost je veličina radijusa izražena u pikselima ili postotkom. Pomoću `border-radius` svojstva možemo četvrtasti `div` element pretvoriti u zaobljene pravokutnike, elipse i konačno krugove (`border-radius: 50%;`). Ukoliko ne želimo da *padding* i rubovi utječu na željenu definiranu visinu i širinu elementa koristimo `box-sizing` svojstvo i vrijednost `border-box` kako bi element ostao u zadanim dimenzijama.

Kao što je ranije spomenuto, elementi u HTML-u mogu biti *block* (blokovi) ili *inline* (u redu) elementi koji se specifično ponašaju. Pomoću CSS-a možemo natjerati da se *block* element ponaša kao *inline* element i obratno. To činimo pomoću `display` svojstva čiju vrijednost postavljamo na jednu od sljedećih opcija: `inline` označava da će se element ponašati kao *inline* element, `block` kao *block* element, a `none` će element maknuti sa stranice (iako je on i dalje vidljiv u kodu). Na vidljivost elementa možemo utjecati i pomoću `visibility` svojstva i `hidden` vrijednosti, iako oni ne uklanjaju element sa stranice već ga samo čine nevidljivim (prostor kojeg bi element zauzeo još uvijek postoji, samo je prazan).

Svaka internetska stranica ima svoj način pozicioniranja elemenata, a koordinatni sustav dokumenta započinje u gornjem lijevom kutu stranice. Elemente možemo pozicionirati na nekoliko načina pomoću `position` svojstva. Normalan (`static`) način je standardno početno stanje pozicioniranja gdje svaki *block* element pada u novi red, a *inline* elementi se prikazuju u redu s prethodnim i naknadnim elementima. Relativnim (`relative`) pozicioniranjem možemo element pomaknuti u odnosu na njegovo normalno stanje pomoću `left`, `right`, `bottom` i `top` svojstava. Taj pomak ne utječe na elemente koji ga okružuju, oni ostaju u svom položaju. Apsolutnim (`absolute`) pozicioniranjem pomičemo element u odnosu na element koji

ga sadrži, on je u ovom slučaju izvučen iz normalnog pozicioniranja te se elementi koji bi ga okruživali jednostavno ponašaju kao da taj element ne postoji i ignoriraju prostor koji bi on trebao zauzeti. U ovom slučaju elementi klize zajedno s ostatkom stranice. Kako bismo element fiksirali i izbjegli spomenuto klizanje koristimo fiksirano (`fixed`) pozicioniranje. To je vrsta apsolutnog pozicioniranja gdje element pozicioniramo ovisno o prozoru preglednika. Tako pozicionirani elementi ne utječu na okolne elemente i ne klize kada pomičemo stranicu gore-dolje. Plutajućim (`float`) svojstvom koji je prikazan na slici 10. elemente vadimo iz njihovog normalnog „toka“ i postavljamo na desni ili lijevi rub elementa koji ih sadržava. Element dobiva *block* svojstva i oko njega se postavljaju ostali elementi. Ukoliko dođe do preklapanja elemenata koristimo `z-index` svojstvo koje funkcionira kao z-os koordinatnog sustava i pomoću tog svojstva određujemo koji se element nalazi iznad.



Slika 10. *Float* svojstvo (citat se nalazi desno, a tekst se pozicionira oko njega)

Većina internetskih stranica danas kao pozadinu koristi sliku što postižemo `background` svojstvom koje se slično kao i `border` svojstvo zapravo sastoji od više svojstava spojenih u jednu liniju koda. Primjer jednog svojstva je `background: #ffffff url("img_tree.png") no-repeat fixed right top;` gdje prva vrijednost označava boju (`background-color`), druga govori pregledniku gdje se slika nalazi, treća govori pregledniku da se slika ne ponavlja (`background-repeat`), četvrta da slika ne kliže zajedno sa sadržajem (`background-attachment`) i na posljetku imamo pozicioniranje slike (`background-position`).

3.5. Transformacije, tranzicije i animacije

Transformacije u CSS3 inačici mogu biti sljedeće: rotacija, translacija, skaliranje i iskrivljavanje. Rotacijom (`rotate`) rotiramo element u smjeru kazaljke na satu, translacijom (`translate`) ga pomičemo iz početnog stanja u kojem bi se on nalazio bez nje, skaliranjem (`scale`) ga smanjujemo ili povećavamo u odnosu na njegovu početnu veličinu, a iskrivljavanjem (`skew`) mu transformiramo kuteve u odnosu na x i y osi koordinatnog sustava. Transformacije su moguće u dvodimenzionalnom i trodimenzionalnom prostoru. Transformacije se zapisuju pomoću `transform` svojstva čija je vrijednost željena transformacija nakon koje slijedi zagrada sa zadanom numeričkom vrijednosti ili kutem (npr. `transform:translate(45px,-30px);` će element pomaknuti po x osi za 45 piksela, a po y osi za 30 piksela, ali u negativnom smjeru). Više transformacija se može zapisati u istoj liniji koda na sljedeći način: `transform: rotate(-10deg) translateX(40px) scale(1.5);` i ova bi linija koda element rotirala za 10 stupnjeva u smjeru suprotnom od kazaljke na satu, translatirala samo po x osi za 40 piksela i povećala element s 1 (početno stanje) na 1.5 početne veličine.

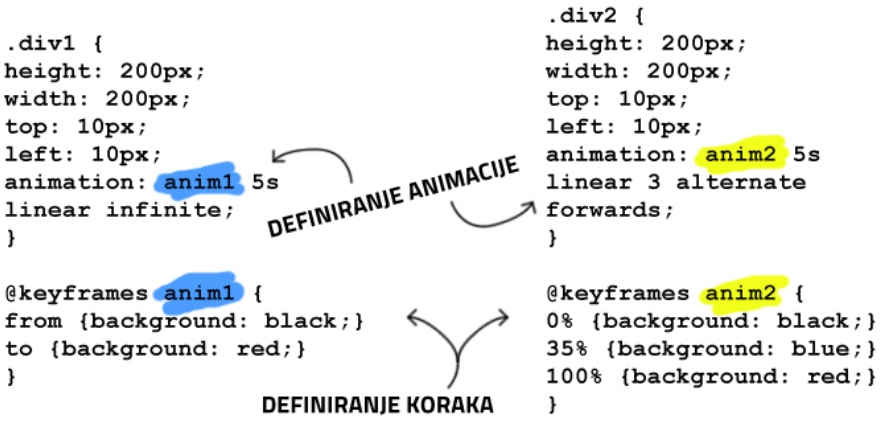
Tranzicije (prijelazi) nam omogućavaju da nekom elementu promijenimo vrijednosti svojstava u slučaju interakcije s elementom (najčešće koristeći `:hover` pseudoklasu). Prijelaz ne mora biti trenutan, već može biti i postupan slično kao i kod animacija. Tranziciju definiramo pomoću `transition` svojstva. Primjer jedne linije koda je: `transition: 2s ease-in-out 50ms;` gdje je prva vrijednost trajanje prijelaza (dvije sekunde), druga vrijednost je tijek prijelaza (`linear` bi označavao jednolik prijelaz, `ease-in` sporiji početak, `ease-out` usporavanje na kraju prijelaza, a `ease-in-out` usporen početak, normalan tijek u sredini i ponovno usporavanje na kraju prijelaza), dok zadnja vrijednost označava vrijeme odgode tranzicije (nakon 50 milisekundi). Ukoliko želimo da prijelaz vidimo samo kada prolazimo pokazivačem preko elementa, a ne kada s njega odlazimo liniju koda upisujemo samo u selektor s `:hover` pseudoklasom, u drugom slučaju liniju koda pišemo u selektor samog elementa te se tada prijelaz vrti kada držimo pokazivač na elementu i kad pokazivač s elementa maknemo. Pomoću CSS selektora možemo prijelaze imati na elementima čak

i ako je pokazivač na drugom elementu koji će u tom slučaju pokrenuti prijelaz, dakle možemo kontrolirati promjene na jednom elementu pomoću drugoga.

Tranzicijama možemo animirati elemente tijekom nekog vremena, ali smo poprilično limitirani (ovisimo o početnom i završnom stanju). Za sve ostalo postoje CSS animacije kojima možemo kontrolirati postepene prijelaze pomoću koraka animacija, dakle osim početnog i završnog stanja imamo mogućnost definirati sve točke animacije između ta dva stanja. Animacija se u CSS-u kreira pomoću `animation` svojstva unutar selektora elementa kojeg želimo animirati. Primjer jedne takve linije koda je `animation: anim1 10s ease-in 3 alternate 5s backward;` gdje je prva vrijednost željeno ime animacije, druga vrijeme trajanja animacije od početnog do završnog stanja (ove prve dvije vrijednosti su obavezne, ostale se koriste po potrebi), treća je jednaka tijeku prijelaza u tranzicijama, četvrta vrijednost označava koliko će se puta animacija ponavljati prije zaustavljanja, peta vrijednost se koristi samo kada animacija ima više ponavljanja kao u našem primjeru gdje imamo tri ponavljanja (`alternate` vrijednost označava da će animacija npr. u prvom prolazu ići od početnog prema završnom, pa zatim od završnog prema početnom i na kraju opet od početnog prema završnom stanju; `reverse` označava da će svako ponavljanje animacija ići od završnog prema početnom stanju, a `alternate-reverse` je ista kao i `alternate` vrijednost u smislu da se naizmjence mijenjaju početno i završno stanje, ali u ovom slučaju animacija započinje završnim i kreće prema početnom), šesta vrijednost nam ukazuje koliko će se animacija odgoditi od trenutka učitavanja stranice, a zadnja vrijednost definira što se događa prije početka i nakon kraja animacije (zadana vrijednost je `none` koja označava da element prije i nakon animacije izgleda neovisno o koracima animacije; pomoću `forwards` će element nakon animacije ostati u zadnjem stanju animacije koje ovisi o tome da li se animacija provodila naizmjence ili ne; pomoću `backwards` element ostaje u stanju s kojim je animacija započela, također ovisno o provođenju animacije). Animaciju možemo pauzirati npr. kada pokazivačem pređemo preko elementa pomoću svojstva `animation-play-state` i vrijednosti `paused`.

Za svaku animaciju moramo definirati *keyframeove* koje možemo zamisliti kao korake unutar početnog i završnog stanja animacije. Oni se definiraju izvan elementa na kojeg se animacija odnosi. Definiranje započinjemo pomoću `@keyframes` selektora

nakon kojeg upisujemo ime animacije koju smo definirali u `animation` svojstvu unutar selektora za element kojeg želimo animirati. Zatim unutar vitičastih zagrada definiramo postotke za korake animacije u kojima će se dešavati promjene na način kako je prikazano na slici 11. Ukoliko imamo samo početno i završno stanje ne moramo koristiti postotke već možemo koristiti `from` i `to` koji označavaju početno i završno stanje animacije.



Slika 11. Definiranje animacije i koraka (*keyframes*) u CSS-u

4. PRILAGODBA SADRŽAJA (RESPONZIVNOST)

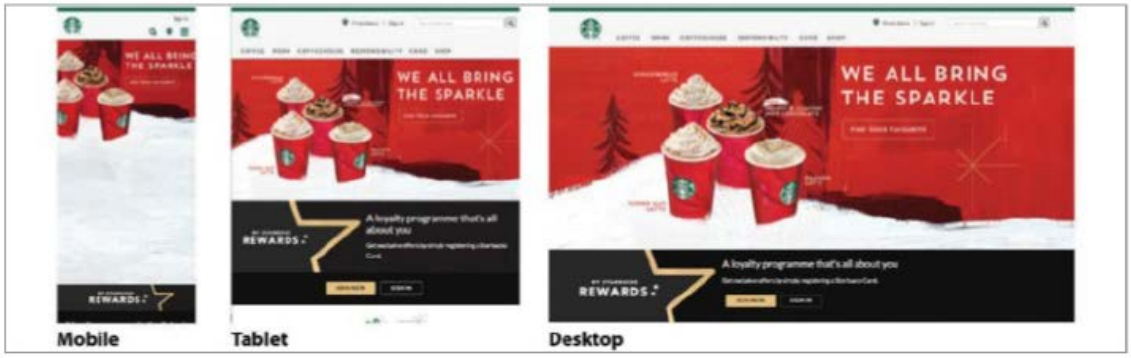
2009. godine u prosjeku je samo 1% posjetitelja internetskih stranica iste posjećivao preko mobilnih uređaja i tableta. Sredinom 2014. ta se brojka popela na više od 35%. To znači da se do nedavno stranice moglo izrađivati u fiksnoj širini (npr. 960px što bi značilo da je sav sadržaj stranice bio sadržan unutar elemenata širokih najviše 960 piksela) i bilo je razumno očekivati da će svi korisnici imati poprilično ujednačeno iskustvo sa stranicom.

Danas zbog povećanog broja posjetitelja internetskih stranica preko pametnih mobilnih telefona, tableta ili prijenosnih računala s malim zaslonima, ali isto tako i korisnika s ekranima većim od 30 inča, internetske stranice izrađene u fiksnoj širini postaju stvar prošlosti. Rješenje ovog problema mnogi dizajneri pronalaze u tzv. responzivnom dizajnu (izraz je osmislio Ethan Marcotte) koji pomoću HTML5 i CSS3 tehnologija omogućava stranicama da funkcioniraju podjednako dobro na svim zaslonima.

U početku je najveći problem responzivnog dizajna bio Internet Explorer i ostali zastarjeli preglednici čije verzije ne podržavaju HTML5 i CSS3. Stoga se često izrađivala zasebna stranica za mobilne preglednike, a zasebna za preglednike na računalima ili se pomoću JavaScripta pokušavalo riješiti problem prilagodbe sadržaja. Srećom, danas se broj korisnika IE preglednika smanjuje jednakom brzinom kojom raste broj korisnika koji stranice posjećuju preko mobilnih uređaja pa i taj problem postaje dijelom prošlosti.

Pitanje koje se često postavlja je zašto onda jednostavno ne koristiti responzivan dizajn stalno? Odgovor je jednostavan, ponekad postoje situacije i internetske stranice za koje je logičnije izraditi zasebnu mobilnu verziju – kada želimo istaknuti potpuno drugačiji sadržaj, kada nam statistika posjetitelja govori o tome da korisnike zanima samo određeni dio sadržaja ako posjećuju preko mobilnih uređaja ili tableta, kada nam statistika govori da većina posjetitelja ipak još uvijek koristi neke od zastarjelih preglednika, kada lokacija utječe na sadržaj ili kada razne varijable poput brzine prijenosa podataka ili mogućnosti samog uređaja kvare doživljaj prekompleksne stranice korisnicima.

Responzivan dizajn se sastoji od tri prije poznate tehnike: fleksibilnog rasporeda sadržaja u *gridu* (eng. rešetkama, mreži), fleksibilnih slika i multimedije te *media queries* (upita) koji su sastavni dio CSS3 tehnologije. Na slici 12. prikazana je jedna responzivna stranica i njena prilagodba sadržaja različitim uređajima i veličinama zaslona.



Slika 12. Primjer responzivne stranice i prilagodbe sadržaja veličini preglednika

4.1. Media Queries

Prije CSS3 inačice dizajner je morao specificirati tip medija pomoću `media` atributa te nije bilo neobično naići na `<link>` element koji je izgledao ovako: `<link rel="stylesheet" href="print.css" media="print">` te se tako mogla definirati drugačija CSS datoteka za svaku vrstu preglednika (uključujući i tisak). Jedna od najvećih mana ovog načina je bila to što iako su preglednici prepoznavali da se samo jedna CSS datoteka odnosila na njih, preuzeli bi sve priložene CSS datoteke što je odužilo učitavanje internetske stranice na što se moralo utjecati pomoću JavaScripta.

Danas uz pomoć *media queries* (CSS3 modul) možemo promijeniti način na koji se sadržaj na stranici prezentira pomoću samo nekoliko linija koda. *Media queries* šalju definirani upit pregledniku i prepoznaju mnogo bitnih faktora za responzivan dizajn od širine preglednika, omjer visine i širine ekrana, rezoluciju te orijentaciju preglednika (portret ili pejzaž).

Sintaksa je vrlo jednostavna: `@media handheld and (max-width: 380px){}` bi se na primjer odnosila samo na sve *handheld* (uređaje koje držimo u rukama, npr. tableti ili mobilni uređaji) čija širina ekrana nije veća od 380 piksela. Sav

CSS koji želimo da se odnosi samo na te uređaje se piše unutar vitičastih zagrada i on nadjačava CSS kojeg smo prije definirali. Tako je danas preko CSS3 moguće definirati općeniti CSS kojeg po potrebi izmijenimo unutar same *media queries* sintakse gdje promijenimo samo ono što želimo (npr. naša stranica ima sliku visoke rezolucije kao pozadinsku, ali u slučaju da stranicu posjećujemo preko uređaja čija je širina preglednika manja od 480 piksela slika se ne učitava i pozadina stranice postaje boja definirana unutar vitičastih zagrada *media queries* sintakse).

4.2. Responzivan sadržaj i tipografija

Jedan od bitnih faktora u responzivnom dizajnu je fluidno skaliranje slika i videa tako da odgovaraju širini ekrana. Jedan od načina na koji možemo postići da se slika skalira ovisno o širini ekrana je sljedeći: kreiramo `<div>` element unutar kojeg se nalazi `<img>` element. U CSS tada uvedemo `div` selektor kojemu zadamo željenu širinu i `img` selektor kojemu definiramo maksimalnu širinu pomoću `max-width` svojstva čiju vrijednost postavimo na 100% te definiramo i visinu pomoću `height` svojstva čiju vrijednost postavimo na `auto`. Tada slika zauzima 100% svog roditelja, a kada se njegova širina promjeni slika se skalira ovisno o njegovoj širini. Visinu smo postavili na `auto` zbog očuvanja omjera visine i širine naše slike.

Pomoću HTML5 je također vrlo lagano video učiniti responzivnim pošto `video` oznaka podržava korištenje postotaka kao za svojstvo širine. Jedino ograničenje je da video mora biti *hostan* (prenesen u direktoriju zajedno s ostalim datotekama na server) zajedno sa stranicom.

Drugi jako bitan faktor je tipografija pošto tekst definiran u pikselima na različitim uređajima izgleda različito, a često postaje i nečitljiv. Recept za rješavanje ovog problema mnogo dizajnera pronalazi u upotrebi rem mjernih jedinica. Pomoću njih veličina fonta je relativna u odnosu na glavnu definiranu veličinu fonta. EM mjerne jedinice čine sličnu stvar, ali ovise o roditelju. Unutar `html` selektora definiramo `font-size` svojstvo na željenu vrijednost (npr. 12px). Ostatak teksta na stranici možemo mijenjati pomoću raznih selektora tako da im `font-size` svojstvo definiramo pomoću rem mjernih jedinica (npr. `p {font-size: 1rem;}` bi značilo

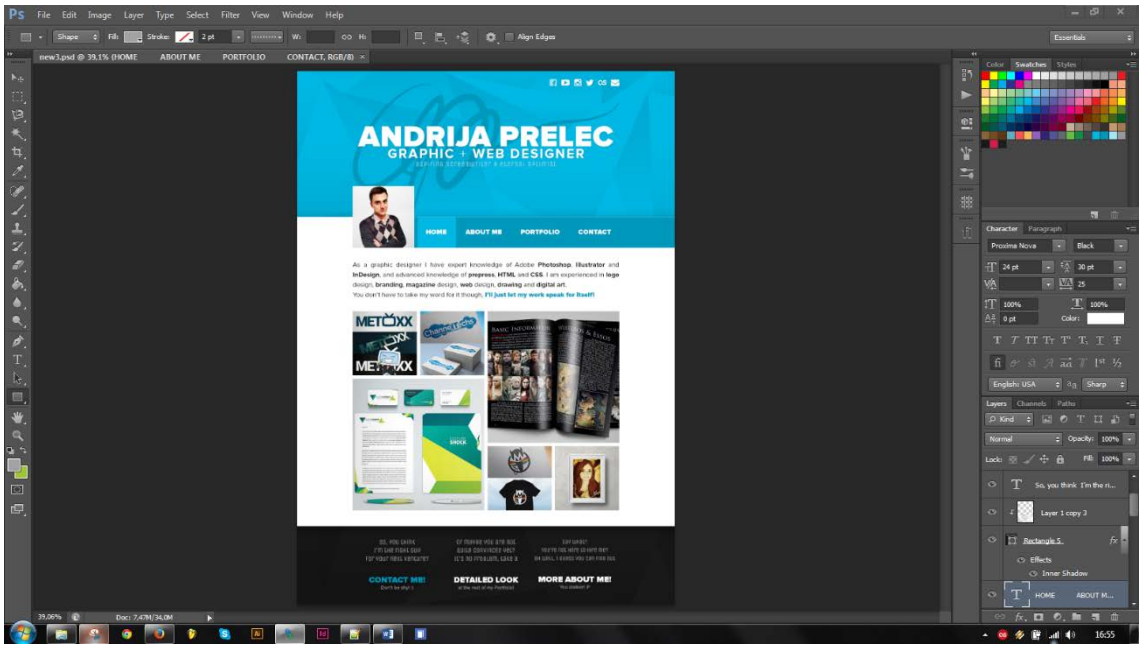
da su svi paragrafi na stranici veliki 12 piksela). Ukoliko u *media queries* dodamo drugačiju vrijednost tada će jednak tekst na mobilnom uređaju biti npr. velik 2rem ili 24 piksela u našem slučaju čime povećavamo čitljivost na malim ekranima.

4.3. Alternativni izbori za responzivni dizajn

Kreiranje nove responzivne stranice često može biti obeshrabljujuće zbog mnogih problema koji se pojavljuju vezanih uz responzivan dizajn. Neki od tih problema su računanje kolumni, kreiranja tzv. mreže internetske stranice, računanja postotaka i kontrolnih točaka u dizajnu (točke koje koristimo pri određenoj širini preglednika koje kažu dizajnu da više ne prikazuje dvije kolumne, nego da druga npr. padne ispod prve) te još mnogo toga. Mnogi dizajneri stoga rješenje prolaze u već kreiranim i isprobanim „kosturima“ stranica.

Bootstrap je jedan od najpopularnijih alternativnih izbora za responzivan dizajn zbog toga što je *open source* (otvoren kod, svima dostupan za korištenje i izmjene kao što je npr. i Firefox preglednik). Kreirao ga je Mark Otto, *developer* u Twitteru, u kolovozu 2011. godine. Foundation je druga popularna alternativa za brzo kreiranje responzivnih stranica.

5. IZRADA RESPONZIVNE OSOBNE INTERNETSKE STRANICE



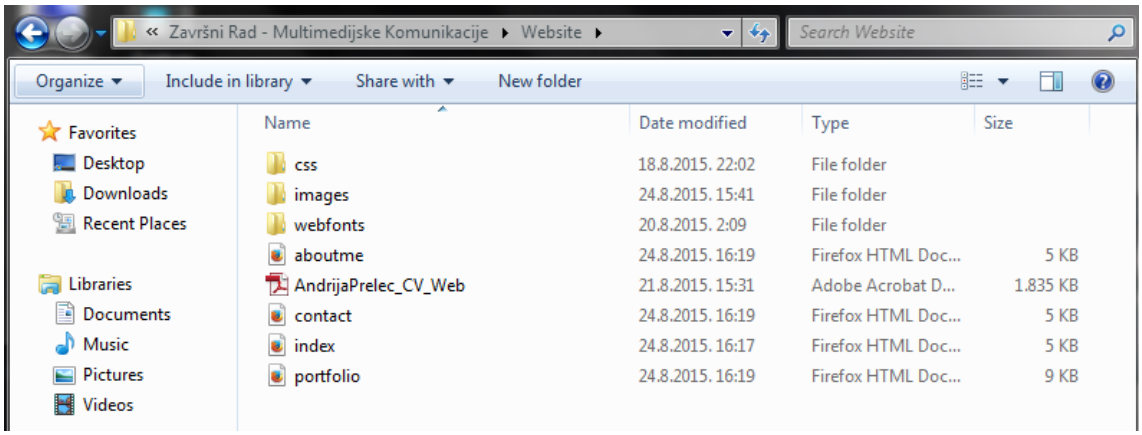
Slika 13. Skica internetske stranice

Za potrebe eksperimentalnog dijela završnog rada dizajnirana je skica izgleda internetske stranice u programu Adobe Photoshop CS6 koja je prikazana na slici 13. Ona prikazuje željeni izgled stranice na zaslonu računala, a kasnije će elementi biti prilagođeni ostalim zaslonima pomoću već spomenutih tehnika prilagodbe sadržaja. Moguće je krenuti i obrnutim putem, to jest započeti kodiranje za mobilne uređaje malih veličina zaslona te kasnije prilagoditi sadržaj većim ekranima poput onih na osobnim računalima.

Radi se o osobnoj internetskoj stranici s prezentacijskim radovima (*portfoliom*) te su stoga osim naslovne stranice planirane i sljedeće podstranice: *About me* (o meni, to jest biografski dio osobne stranice), *Portfolio* (podstranica s prezentacijskim radovima) te *Contact* (podstranica koja sadrži kontakt podatke i životopis).

Za početak je potrebno kreirati mapu na računalu koja će sadržavati sve što će se nalaziti na internetskoj stranici i koja će biti postavljena na server. Podmape koje bi gotovo svaka stranica trebala sadržavati su css podmapa te podmapa sa slikama. Sve slike koje će se koristiti u dizajnu postavljaju se u podmapu images. Pošto su u dizajnu korišteni nestandardni fontovi kreirana je i mapa webfonts koja ih sadrži kako bi se

osigurao željen doživljaj stranice. Na posljetku su kreirana četiri HTML dokumenta za glavnu stranicu (*index*) i sve podstranice te CSS dokument koji se nalazi u css podmapi. Sadržaj mape je prikazan na slici 14.



Slika 14. Izgled mape s datotekama i podmapama potrebnim za internetsku stranicu

Kodiranje stranice započinje `index.html` datotekom te definiranjem vrste dokumenta pomoću `!DOCTYPE` elementa. Zatim se otvara i zatvara `html` element unutar kojeg se isto tako otvaraju i zatvaraju `head` i `body` elementi. Unutar `head` elementa definira se naslov internetske stranice `title` elementom. Pomoću `meta` elemenata i raznih atributa i svojstava definiraju se vrsta znakova koja se koristi za kodiranje (`charset` atribut), opis stranice (`description`), ključne riječi potrebne za tražilice (`keywords`), upisuje se ime autora stranice (`author`) te se naposljetku definira i vrsta prikaza koja će biti potrebna kasnije zbog responzivnosti (`viewport`). Posljednji element koji se nalazi unutar `head` elementa jest `link` element kojim se HTML dokument upućuje na lokaciju CSS dokumenta u kojem je definirano oblikovanje stranice. Sav dosad definirani kod može se kopirati i primijeniti na sve `.html` dokumente (dakle na sve podstranice). Za razliku od `head` elementa, unutar `body` elementa nalazi se sav vidljivi sadržaj stranice. Stranica je raspodijeljena na *header* (vrh, glava stranice), navigaciju, sadržaj te *footer* (dno, podnožje stranice). Početni izgled osnovnog HTML dokumenta vidljiv je na slici 15.

Unutar CSS datoteke za početak se definiraju fontovi pomoću `@font-face` selektora što prikazuje slika 16. Potrebno je definirati lokaciju pomoću `src` svojstva za sve debljine koje se koriste na stranici pošto se svaka debljina gleda kao novi font. Postoje i mnoge internetske stranice (kao što je na primjer:

http://www.fontsquirrel.com/tools/webfont-generator) koje same kreiraju deklaracije za *uploadane* fontove koje se zatim samo kopiraju i zalijepe unutar `.css` datoteke.

index.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4
5      <meta charset="utf-8">
6      <title> Andrija Prelec - Personal Website + Portfolio </title>
7
8      <meta name="description" content="Graphic and Web Designer">
9      <meta name="keywords" content=
       "web,design,portfolio,designer,graphic,html,css,Croatia,grafički,dizajner,Hrvatska">
10     <meta name="author" content="Andrija Prelec">
11
12     <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
13     <link rel="stylesheet" href="css/style.css">
14
15 </head>
16 <body>
17
18 </body>
19 </html>
```

Slika 15. Početni izgled koda i definiranje glave (`head` elementa) stranice (HTML #1)

index.html style.css

```
1  @font-face {
2      font-family: 'Proxima Nova';
3      src: url('../webfonts/proximanova-black-webfont.eot');
4      src: url('../webfonts/proximanova-black-webfont.eot?#iefix') format(
       'embedded-opentype'),
5           url('../webfonts/proximanova-black-webfont.woff2') format('woff2'),
6           url('../webfonts/proximanova-black-webfont.woff') format('woff'),
7           url('../webfonts/proximanova-black-webfont.ttf') format('truetype'),
8           url('../webfonts/proximanova-black-webfont.svg#proxima_novablack') format(
            'svg');
9      font-weight: 900;
10     font-style: normal;
11 }
```

Slika 16. Primjer `@font-face` svojstva i potrebnih deklaracija (CSS #1)

Kao što je vidljivo na skici stranice (slika 13.) unutar *headera* se nalaze ikone s linkovima na društvene mreže, te nekoliko linija teksta i 2 pozadinske slike (plava pozadina te prozirna slika potpisa iznad pozadine). Stoga se unutar `body` elementa kreira `header` element koji sadrži `div` s klasom `container` (koja nam služi da sadržaj stranice centriramo i držimo u željenoj širini), `div` koji sadrži ikone te `div` sa željenim tekstom. Slika 17. prikazuje kreirani kod objašnjen u ovom koraku.

U CSS-u unutar `body` selektora deklariraju se fontovi, pozadina stranice, boja teksta i `overflow` kojim se kontrolira što se događa ukoliko sadržaj izlazi iz *box* elemenata. Osim toga definiraju se i svojstva linkova (`a`) i naslova (`h1-h3`) koje koristimo u *headeru*. U header selektoru definira se širina, visina, slika pozadine (pomoću `url` veze), centriranje teksta i mnogo toga poput `box-shadow` svojstva kojim stvaramo željenu sjenu unutar *headera*. Sve spomenuto vidljivo je na slici 18.

```
15  </head>
16  <body>
17
18      <header>
19          <div class="container header">
20              <div class="socials">
21                  <a href="https://www.facebook.com/andrija.prelec" target="_blank">C</a>
22                  <a href="https://www.youtube.com/user/Prelec93" target="_blank">W</a>
23                  <a href="https://instagram.com/lecpre" target="_blank">H</a>
24                  <a href="https://twitter.com/lecpre" target="_blank">F</a>
25                  <a href="http://www.last.fm/user/prelec93" target="_blank">O</a>
26                  <a href="mailto:info@andrijaprelec.com" target="_blank">o</a>
27              </div>
28              <h1>ANDRIJA PRELEC</h1>
29              <h2>graphic + web designer</h2>
30              <h3>aspiring screenwriter & eternal optimist</h3>
31          </div>
32      </header>
```

Slika 17. Definiranje `header` elementa (HTML #2)

index.html style.css

```
60
61  body {
62      font-family: 'Proxima Nova', sans-serif;
63      font-size: 16px;
64      padding: 0px;
65      margin: 0px;
66      background: #fff;
67      color: #333;
68      overflow-x: hidden;
69      overflow-y: auto;
70      -moz-user-select: none;
71      -webkit-user-select: none;
72      cursor: default;
73  }
74  a {
75      text-decoration: none;
76      color: #fff;
77      transition: 0.25s ease-in-out;
78      -moz-transition: 0.25s ease-in-out;
79      -webkit-transition: 0.25s ease-in-out;
80  }
81  a:hover {
82      color: #a6e6f6;
83  }
84
85  header {
86      width: 100%;
87      height: 524px;
88      background: url(../images/blueback.jpg) no-repeat;
89      background-size: cover;
90      text-align: center;
91      color: #fff;
92      box-shadow: inset 0 0 70px #0098bc;
93  }
130 .header {
131     background: url(../images/sig.png);
132     height: 524px;
```

```
94  .socials {
95      font-family: fontello;
96      font-size: 2rem;
97      letter-spacing: 5px;
98      padding-top: 15px;
99      text-align: right;
100     text-shadow: 2px 2px 2px #0098bc;
101 }
102
103 h1 {
104     font-weight: 900;
105     font-size: 6.75rem;
106     padding-top: 140px;
107     margin: 0px;
108     line-height: 4rem;
109     text-shadow: 2px 2px 2px #0098bc;
110 }
111 h2 {
112     font-weight: 800;
113     font-size: 3rem;
114     margin: 0px;
115     text-transform: uppercase;
116     letter-spacing: 10px;
117     line-height: 4rem;
118     text-shadow: 2px 2px 2px #0098bc;
119 }
120 h3 {
121     font-family: 'Unica One';
122     font-size: 1.5rem;
123     margin: 0px;
124     line-height: 0.5rem;
125     letter-spacing: 6px;
126     color: #a6e6f6;
127     text-shadow: 2px 2px 2px #0098bc;
128 }
134 .container {
135     width: 960px;
136     margin: 0px auto;
137 }
```

Slika 18. Deklariranje potrebnog CSS-a (CSS#2)

Slika 19. Izgled internetske stranice u pregledniku (#1)

Nakon definiranja *headera*, koji je u potpunosti dovršen i prikazan na slici 19., na red dolazi navigacija, to jest `nav` element (*menu* internetske stranice) koji sadrži profilnu sliku te četiri poveznice koje vode na podstranice. Definiran je i `div` s klasom `clearfix` koji se također nalazi unutar `nav` elementa. HTML dio koda koji se odnosi na navigaciju vidljiv je na slici 20.

U CSS-u osim pozadine navigacije (`nav` element) deklariraju se i svojstva `.menu` klase. Ona ima `float` svojstvo s vrijednošću `left` koje sve elemente unutar tog `div` elementa postavlja jedan nakon drugoga umjesto jedan ispod drugoga. Također se definiraju i svojstva koje linkovi imaju kada se preko njih pređe mišem pomoću `:hover` pseudoklase. Sav CSS potreban za željen prikaz navigacije prikazan je na slici 21., a trenutan izgled stranice vidljiv je na slici 22.

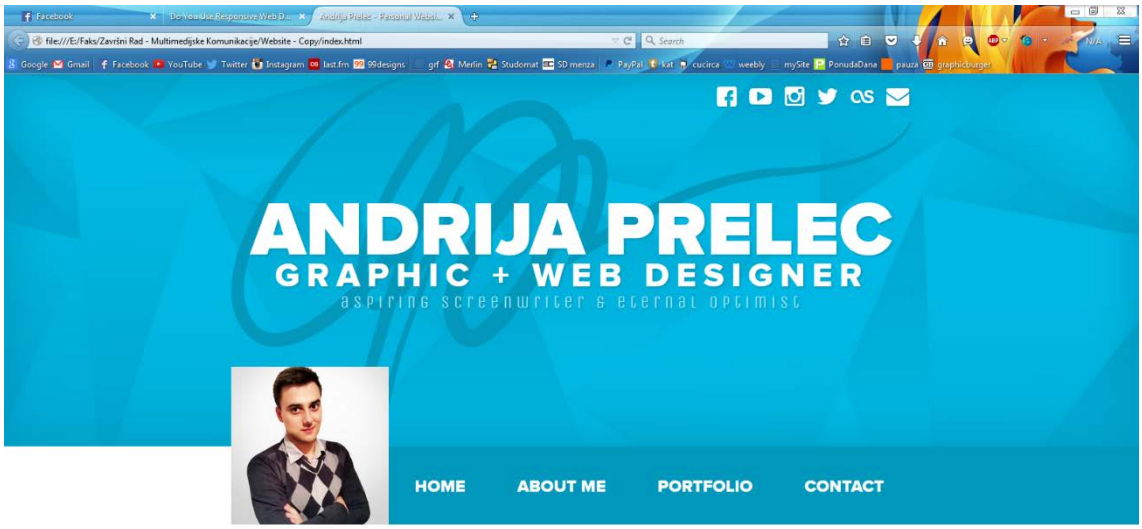
```
46              <h3>aspiring screenwriter & eternal optimist</h3>
47          </div>
48      </header>
49      <nav>
50          <div class="container">
51              <img src="images/profile2.jpg" class="profile-img" alt="profile picture">
52              <div class="invisible"></div>
53              <a href="index.html" class="menu">HOME</a>
54              <a href="aboutme.html" class="menu">ABOUT ME</a>
55              <a href="portfolio.html" class="menu">PORTFOLIO</a>
56              <a href="contact.html" class="menu">CONTACT</a>
57          </div>
58          <div class="clearfix"></div>
59      </nav>
```

Slika 20. Definiranje navigacije unutar HTML dokumenta (HTML #3)

```
164  nav {
165      background: url(../images/nav_bg.png) no-repeat center center;
166  }
167  .menu {
168      box-sizing: border-box;
169      background: #0098bc;
170      height: 110px;
171      padding: 42px 36.5px 0px;
172      display: table-cell;
173      float: left;
174      vertical-align: middle;
175      font-weight: 800;
176      font-size: 1.5rem;
177      transition: 0.25s ease-in-out;
178      -moz-transition: 0.25s ease-in-out;
179      -webkit-transition: 0.25s ease-in-out;
180  }
181
182  a.menu:hover {
183      background: url(../images/blueback.jpg) no-repeat center;
184      color: #fff;
185      box-shadow: inset 0 0 20px #0098bc;
186      padding: 36px 36.5px 0px;
187  }
188
189  .profile-img {
190      float: left;
191      margin-top: -112px;
192      width: 222px;
193      height: 222px;
194  }
```

Slika 21. Deklariranje potrebnog CSS-a (CSS#3)

`.clearfix` klasa ima samo jednu deklaraciju, a to je `clear: both;` koja omogućava da roditelj dobije definiranu širinu i kraj pošto inače sadržava samo `float` elemente koje zbog tog svojstva HTML ne vidi kao da imaju širinu i visinu što često može stvarati probleme.



Slika 22: Izgled internetske stranice u pregledniku (#2)

Sadržaj stranice nalazi se unutar jednog `div` elementa koji ima dvije klase: `.content` i već prije deklariranu `.container` klasu (vidljivo na slici 23.). Osim toga potrebno je deklarirati i paragrafe (`p` element), podebljan tekst (`b`) te `div` elemente koji će sadržavati prezentacijske radove u obliku slika. Zbog specifične pozicije tih `div` elemenata kreirano je nekoliko klasa kojima se oni korektno pozicioniraju na stranici.

Sadržaj podstranica je jedini dio koji se razlikuje pošto su *header*, navigacija i *footer* na svim podstranicama jednaki. Ukoliko se dobro definiraju željene klase, iste se mogu koristiti na svim podstranicama što uvelike umanjuje vrijeme potrošeno na deklariranje CSS-a koji je prikazan na slici 24., te je jedini preostali posao ubacivanje sadržaja unutar `div` elementa.

```
    </nav>

    <div class="content container">
    <p>As a graphic designer I have expert knowledge of Adobe <b>Photoshop</b>, <b>
    Illustrator</b> and <b>InDesign</b>, and advanced knowledge of <b>prepress</b>,
    <b>HTML</b> and <b>CSS</b>. I am experienced in <b>logo</b> design, <b>branding
    </b>, <b>magazine</b> design, <b>web</b> design, <b>drawing</b> and <b>digital art
    </b>.
    <br>You don't have to take my word for it though, <b><a class="linkin" href=
    "portfolio">I'll just let my work speak for itself!</a></b></p>
    <div class="portf small L"><img src="images/Logo_04.jpg" class="porslik" alt=
    "Logo for Metoxx"></div>
    <div class="portf small M"><img src="images/Art_01.jpg" class="porslik" alt=
    "Portrait of a girl"></div>
    <div class="portf big R"><img src="images/Magazine_02.jpg" class="porslik" alt=
    "Game of Thrones magazine"></div>
    <div class="portf big L"><img src="images/Branding_01.jpg " class="porslik" alt=
    "Culture Shock Rebranding"></div>
    <div class="portf small M"><img src="images/Logo_03.jpg" class="porslik" alt=
    "Marina Orsag Logo"></div>
    <div class="portf small R"><img src="images/Photoshop_01.jpg" class="porslik" alt=
    "Photoshop Retouching"></div>
    <div class="clearfix"></div>
    </div>
```

```
    <div class="content container">
    <img src="images/malija.jpg" class="abaut">
    <p>I was born on <b>December 16, 1993</b> in <b>Krizevci, Croatia</b> to Darko
    and Marina Prelec and I have five siblings (two older brothers, one younger
    brother and two younger sisters). <br><br>
    I have attended <b>Ivan Zakmardij Dijankovecki Gymnasium</b> (high school with
    primary focus on Math and Science) in my hometown from 2008 until 2012. I am
    currently in my <b>third year of college</b> (I am studying at the <b>Faculty of
    Graphic Arts</b> in Zagreb, Croatia) and I am aspiring to become a <b>graphic
    designer</b> as I have loved drawing and all sorts of other creative stuff for as
    long as I can remember. At the age of 12 I have also started playing around with
    <b>web design</b> and image editing programs like <b>Photoshop</b> and my passion
    about them didn't stop since.<br><br>
    My second passion is <b>writing/screenwriting</b> as I love reading and watching
    TV shows and movies. When I was younger I have worked on a site which released
    shows created with the Sims 2 and 3 (stategic life simulation games) and I have
    also released some solo projects on <b>YouTube</b> afterwards (including a
    somewhat popular show called Bridgeport Hills, also made by using the Sims 3).
    <br>
</p>

    <div class="clearfix"></div>
    </div>
```

Slika 23: Definiranje sadržaja naslovne stranice (iznad) i *about me* podstranice (ispod) (HTML #4)

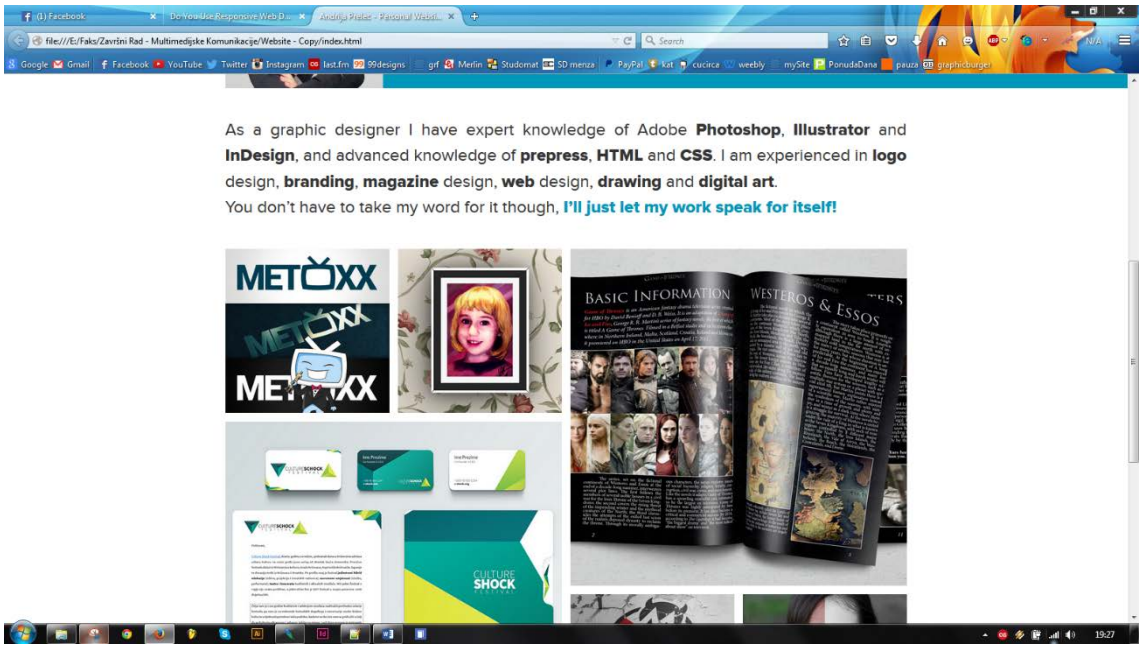
index.html style.css

```
200  p {
201      font-size: 1.5rem;
202      line-height: 2.25rem;
203      padding: 10px 0px;
204      text-align: justify;
205      color: #333;
206      font-weight: 400;
207      margin: 30px 0px 30px 0px;
208  }
209
210  b {
211      font-weight: 800;
212  }
213
214  .linkin {
215      color: #0098bc;
216  }
217
218  .linkin:hover {
219      color: #00b8e4;
220  }
221
222  .portf {
223      float: left;
224      overflow: hidden;
225  }
226
227  .big {
228      height: 474px;
229      width: 474px;
230  }
231
232  .small {
233      height: 231px;
234      width: 231px;
235  }
```

index.html style.css

```
236
237  .L {
238      margin: 0px 6px 12px 0px;
239  }
240  .M {
241      margin: 0px 6px 12px 6px;
242  }
243  .R {
244      margin: 0px 0px 12px 6px;
245  }
246
247  .big.R + .big.L {
248      margin-top: -243px;
249  }
250
251  .porslik {
252      max-width: 100%;
253      height: auto;
254      transition: 0.5s;
255      -moz-transition: 0.5s;
256      -webkit-transition: 0.5s;
257  }
258
259  .porslik:hover {
260      transform:scale(1.1);
261  }
```

Slika 24. Deklariranje potrebnog CSS-a (CSS#4)



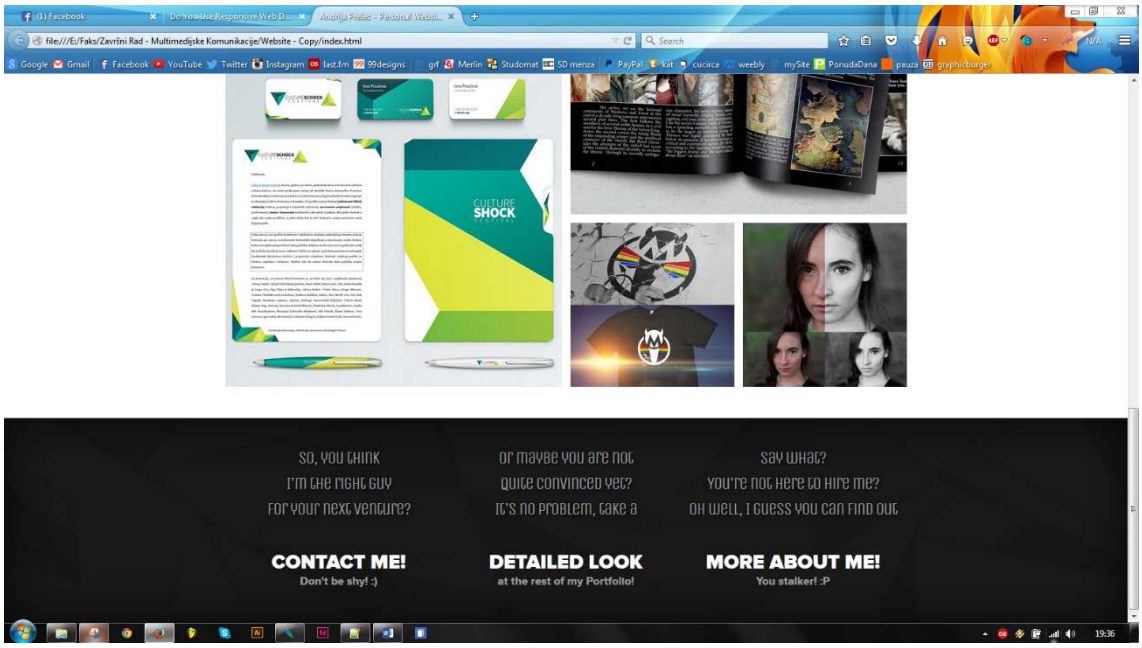
Slika 25. Izgled internetske stranice u pregledniku (#3)

Nakon izrade svih podstranica (od kojih je trenutan izgled početne prikazan na slici 25.) na red dolazi `footer` koji sadrži 3 div elementa jednake širine unutar kojih se nalaze paragrafi s različito stiliziranim tekstovima (pomoću klasa). HTML i CSS kod prikazan na slici 26. je potreban za željen izgled *footera* koji je vidljiv na slici 27.

```
<footer>
<div class="container">

<div class="d320">
<p class="footext1">So, you think <br>I'm the right guy <br>for your next venture?
</p>
<a class="footext2" href="contact.html">CONTACT ME!</a>
<p class="footext3">Don't be shy! :)</p>
</div>

<div class="d320">
<p class="footext1">Or maybe you are not <br>quite convinced yet? <br>It's no
problem, take a</p>
<a class="footext2" href="portfolio.html">DETAILED LOOK</a>
<p class="footext3">at the rest of my Portfolio!</p>
</div>

<div class="d320">
<p class="footext1">Say what? <br>You're not here to hire me? <br>Oh well, I
guess you can find out</p>
<a class="footext2" href="aboutme.html">MORE ABOUT ME!</a>
<p class="footext3">You stalker! :P</p>
</div>
<div class="clearfix"></div>
</div>
</footer>
```

```
.d320 {
    float: left;
    width: 320px;
    text-align: center;
}

.footext1 {
    text-align: center;
    font-family: 'Unica One';
    font-size: 1.5rem;
    letter-spacing: -1.2px;
    color: #999999;
    text-shadow: 1px 1px 2px #000;
    margin-top: 28px;
}

.footext2 {
    font-weight: 900;
    font-size: 1.75rem;
}

.footext2:hover {
    color: #00b8e4;
}

.footext3 {
    text-align: center;
    font-size: 1rem;
    font-weight: 800;
    margin-top: -20px;
    color: #999999;
}
```

Slika 26. Definiranje *footera* unutar HTML-a, te deklariranje potrebnog CSS-a (HTML+CSS #5)



Slika 27. Izgled dovršene internetske stranice u pregledniku (#4)

Responzivnost sadržaja postiže se pomoću `media queries` sintakse unutar samog CSS-a, dakle jednake HTML datoteke će se prikazivati drugačije ovisno o zaslonu i uređaju na kojem se pregledavaju. Sav sadržaj stranica nalazi se unutar `div` elemenata koji sadrže `.container` klasu kojoj je pripisana širina od 960px. Stoga je potrebno je definirati što se događa sa sadržajem ukoliko je zaslon manji od 960 piksela.

```
@media (max-width: 999px) {
header {
    height: 480px;
}
.container {
    width: 750px;
}
.socials {
    text-align: center;
}
h1 {
    font-size: 5rem;
    padding-top: 75px;
    line-height: 3.5rem;
}
h2 {
    font-size: 2.5rem;
    letter-spacing: 9px;
    line-height: 3rem;
}
h3 {
    line-height: 1rem;
}
.header {
    width: 100%;
}
nav {
    background: #0098bc;
}
.menu {
    height: 48px;
    padding: 12px 18px 0px;
    font-size: 1.25rem;
}
a.menu:hover {
    padding: 12px 18px 0px;
}
.profile-img {
```

```
@media (max-width: 768px) {
header {
    height: 250px;
}
.header {
    height: 250px;
    background-size: cover;
}
.container {
    width: 100%;
    box-sizing: border-box;
    padding: 20px;
}
h1 {
    font-size: 3rem;
    padding-top: 35px;
    line-height: 2.5rem;
}
h2 {
    font-size: 1.5rem;
    letter-spacing: 3px;
    line-height: 1.25rem;
}
h3 {
    display: none;
}
.menu {
    text-align: center;
    float: none;
    display: block;
}
.profile-img {
    width: 100%;
    height: auto;
    margin: 0 0;
}
p {
    font-size: 1.2rem;
```

Slika 28. `@media queries` i definiranje promjena pri određenim širinama zaslona (CSS #6)

Nakon `@media` selektora nalazi se zagrada unutar koje se upisuje pri kojoj će se veličini dogoditi neka promjena. Prve promjene na stranici događat će se ukoliko je širina stranice manja od 999 piksela stoga se deklarira `(max-width: 999px)` zagrada nakon koje se otvaraju vitičaste zagrade unutar kojih se nalaze sve promjene u CSS-u koje tek tada počinju vrijediti. Neke od važnijih promjena su npr. te da klasa `.container` počinje biti široka 750px, mijenjaju se veličine fontova, smanjuje se navigacija, te div elementi koji sadrže prezentacijske radove mijenjaju pozicije i veličine.

Sljedeći niz promjena počinje vrijediti ukoliko je širina ekrana manja od 768 piksela (prilagodbe za mobilne zaslone). Klasa `.container` sada zauzima 100% širine ekrana, ali dobiva i `padding` od 20 piksela kako tekst ne bi bio potpuno uz rub preglednika. Naslovi se još više smanjuju, a `div` elementi sa slikama prezentacijskih radova zauzimaju cijelu širinu ekrana te se nalaze jedan ispod drugoga. Čak i elementi unutar navigacije više nemaju `float` svojstvo već padaju jedan ispod drugoga radi lakše preglednosti. Obje sintakse su prikazane na slici 28.

Uz pomoć `media queries` i samo nekoliko novih linija koda kreirana je u potpunosti responzivna stranica koja je vidljiva na slici 29. na nekoliko različitih uređaja. Cijeli svoj sadržaj prilagođava svim aktualnim veličinama zaslona bez ikakve upotrebe drugih tehnologija.

Slika 29: Potpuna prilagodba sadržaja svih veličinama zaslona

6. ZAKLJUČAK

Tehnologija neprestano napreduje, a zajedno s pozitivnim promjenama dolaze i negativne koje se trebaju nadvladati. Do nedavno je prilagodba sadržaja internetskih stranica velikom broju različitih zaslona (od pametnih telefona malih zaslona do monitora osobnih računala visoke rezolucije) bila mukotrpan posao koji je uključivao razne tehnologije (ponajviše JavaScript) ili kreiranje zasebnih stranica za različite uređaje.

Cilj ovog rada bio je provjeriti je li moguće većinu, ako ne i sav sadržaj, u potpunosti učiniti responzivnim pomoću aktualnih inačica HTML i CSS tehnologija. Iako postoji mnogo besplatnih alata i stranica koje nude već unaprijed responzivne kosture ili teme za internetske stranice u ovom radu je u potpunosti kreirana osobna stranica s prezentacijskim radovima korištenjem samo HTML5 i CSS3 tehnologija.

Kreirana stranica je testirana na mnogim drugim internetskim stranicama koje su specijalizirane za simulaciju responzivnosti na modernim uređajima manjih rezolucija (www.responsinator.com, www.responsivetest.net i www.responsivedesignchecker.com su samo neke od korištenih), te u različitim preglednicima na mobilnim uređajima (Samsung Galaxy S5, Samsung Galaxy Trend Plus, iPhone 5) i osobnim računalima. Sav sadržaj se uvijek prilagodio sukladno sa širinom zaslona.

Time je dokazano da se sav sadržaj prosječnih internetskih stranica može prilagoditi svim zaslonima uz samo nekoliko dodatnih linija koda i ponajviše razumijevanjem već spomenutih tehnologija.

7. LITERATURA

1. Crowther R. (2013.) *Hello! HTML5 & CSS3*, Manning Publications, Shelter Island
2. Duckett J. (2011.) *HTML & CSS - Design and Build Websites*, John Wiley & Sons, Inc., Indianapolis
3. Goldstein A., Lazaris L., Weyl E. (2015.) *HTML5 & CSS3 For the Real World (Second Edition)*, SitePoint Pty. Ltd., Melbourne
4. LaGrone B. (2013.) *HTML5 and CSS3 Responsive Web Design Cookbook*, Packt Publishing Ltd., Birmingham
5. Shenoy A., Guarini G. (2013.) *HTML5 and CSS3 Transition, Transformation and Animation*, Packt Publishing Ltd., Birmingham
6. Imagine Publishing Ltd. (2014.) *HTML5 & CSS3 – The Complete Manual*, Imagine Publishing Ltd., Bournemouth
7. Keith J. (2010.) *HTML5 for Web Designers*, A Book Apart, New York
8. Frain B. (2012.) *Responsive Web Design with HTML5 and CSS3*, Packt Publishing Ltd., Birmingham
9. Dawson A. (2009.) *Getting StartED Building Websites*, Apress Co., New York
10. Nixon R. (2014.) *Learning PHP, MySQL, JavaScript, CSS & HTML5 (Third Edition)*, O'Reilly Media, Inc., Sebastopol
11. http://www.creativebloq.com/rwd/pros-guide-responsive-web-design-71515692 – Pro's Guide to Responsive Web Design, 14.7.2015.
12. https://hr.wikipedia.org/wiki/Internet – Internet, 17.7.2015.
13. https://hr.wikipedia.org/wiki/Web_stranice – Web stranice, 17.7.2015.
14. https://en.wikipedia.org/wiki/Web_page – Web page, 17.7.2015.
15. https://hr.wikipedia.org/wiki/World_Wide_Web – World Wide Web, 17.7.2015.
16. https://en.wikipedia.org/wiki/HTML – HTML, 18.7.2015.
17. https://hr.wikipedia.org/wiki/Hipertekst – Hipertekst, 18.7.2015.
18. https://en.wikipedia.org/wiki/Cascading_Style_Sheets – Cascading Style Sheets, 22.7.2015.
19. https://hr.wikipedia.org/wiki/CSS – CSS, 22.7.2015.
20. http://www.w3.org/TR/css-cascade-3/ - CSS Cascading and Inheritance Level 3, 23.7.2015.