HTML & CSS

Alen Vlahovljak

Almir Osmanagić

Almir Osmanagić

Alen Vlahovljak

HTML & CSS

Sarajevo, 2019.

Ovaj priručnik su izradili autori u sastavu:

Autori: Almir Osmanagić i Alen Vlahovljak
Print dizajn: Almir Osmanagić
Lektor: Alen Vlahovljak

Verzija priručnika: 1.0

Predgovor

U zadnjih dvadesetak godina su web stranice postale sveprisutne i jedan su od glavnih načina digitalne komunikacije. Za razliku od standardnih medija, web nudi mogućnost objave sadržaja i to na jednostavan način. Preporučeno prethodno znanje za ovaj kurs su osnove Interneta, tj. da se razumije šta je Internet pretraživač (odnosno preglednik), web sajtova (sjedišta), korištenje email-a, pretraživača, itd. Također, smatra se da poznajete osnove rada na računaru, u smislu kreiranja, izmjene i brisanja datoteka kao i poznavanje rada na operativnom sistemu (Windows, macOS ili Linux). Iako priručnik nastoji biti iscrpan, određeni broj rijetko korištenih svojstava i mogućnosti HTML-a i CSS-a nije mogao biti uvršten u kurs. Priručnik se sastoji od: predgovora, XX poglavlja i X praktičnih vježbi za čije su izvođenje potrebni dodatni materijali.

„Uvod u HTML i CSS“ kurs je namijenjen svim osobama koje žele da se upoznaju sa kreiranjem statički web stranica. Kroz ovaj kurs, ćete naučit koristiti razne opcije za kreiranje i uređivanje web stranica, kao i njihovih elemenata na moderan način, pored čega ćete imati priliku naučiti kulturu pisanja koda, što predstavlja: urednost, efikasnost i jednostavnost. Znanje koje usvojite kroz ovaj kurs vas neće ograničiti samo na navedeno u planu i programu. Imat ćete priliku da ovladate korištenjem raznih alata koji su namijenjeni za raznovrsnu primjenu, što znači da vi odlučujete kako i koliko maštovito ćete iste koristiti. Na početku i kraju svakog poglavlja (glave) se nalazi rezime cjeline koja će se obrađivati.

Za sve eventualne greške, kontradiktornosti i slično, molim da se prijave autorima.

Autori:
Almir Osmanagić (almir.osmanagic@live.com)
Alen Vlahovljak (alen_vlahovljak@hotmail.com)

I Web stranice

Po završetku ovog poglavlja, polaznik će moći:

- objasniti što je to HTML;
- objasniti što je web sajt i gdje se web sajtovi nalaze;
- osmisliti informacionu strukturu jednostavnog web sajta;
- objasniti pojmove SEO, pristupačnost i prilagodljivost web stranica.

Web stranice se izrađuju upotrebom HTML jezika pomoću kojeg se definiše struktura i sadržaj. Uz CSS, kojim se web stranice oblikuju i pomoću kojeg poprimaju svoj izgled, HTML je osnovna tehnologija na kojoj se temelji izrada web stranica.

1.1. HTML-u i izradi web-stranica

Osnovni gradivni element svake web stranice je HTML kod. HTML je skraćenica za ***HyperText Markup Language***. HTML je jezik za „označavanje" kojim se određuje struktura, sadržaj i funkcija nekog HTML dokumenta (web stranice ili sajta). Pomoću HTML-a se određuju važni elementi svakog HTML dokumenta kao što su npr. naslov, odlomak, slika, veza (engl. *hyperlink*) i sl. Dakle, **HTML je jezik za označavanje pomoću kojeg se može odrediti struktura elemenata unutar HTML dokumenta**. Osim što je struktura elemenata definisana u HTML-ovom dokumentu, CSS[1] omogućava da se ti elementi stilski/grafički urede. Na taj način se web pregledniku daje do znanja kako će stranica izgledati prilikom prikaza.

HTML je početkom 90-ih godina prošlog vijeka razvio **Tim Berners-Lee**, fizičar koji je zaposlen u CERN[2]-u. HTML je vrlo brzo stekao veliku popularnost koja je tokom 90-ih utjecala na nagli razvoj Interneta. Zbog tako naglog razvoja je razvijeno nekoliko **HTML-ovih standarda**, koji su međusobno bili neusklađeni. Od pojave HTML-a do prve prave upotrebljive verzije; a izdala je **W3C organizacija** (engl. *World Wide Web Consortium*); prošlo je šest godina. HTML je u početku omogućavao jednostavno strukturirane web stranice - odlomke, prijelome redova i zaglavlja, a nije omogućavao unos grafike i multimedijalnih elemenata. HTML se razvijao, a da se pri tome nije pridavala pažnja oblikovanju i prezentaciji stranice, odnosno izgledu, jer je osnovni cilj bio struktuisanje podataka na prost način. Pojavom vizualnih internetskih preglednika, a radi brzog

[1] **CSS** kod se može dodati u HTML dokument, ali ispravnije je CSS izdvojiti u poseban dokument, povezan sa HTML dokumentom.
[2] home.cern

i nekontrolisanog razvoja, pojavili su se problemi oko međusobne nekompatibilnosti različitih preglednika na različitim platformama. Ti su problemi aktualni i danas, ali su sve manje izraženi.

Svaka nova verzija HTML-a je donosila novosti i unaprjeđena, te je predstavljala napredak. Ovaj napredak (većinom) prate i internetski preglednici[3], pa se u najnovijim preglednicima HTML kod ispravno prikazuje. Problem se javlja zbog toga što korisnici često nemaju instalirane najnovije verzije preglednika. Ovaj problem se rješava na način da se prilikom pisanja HTML koda pazi da on bude kompatibilan sa par zadnjih verzija internetskih preglednika. Aktualna verzija HTML-a se zove **HTML5** i podržana je u svim novijim internetskim preglednicima. Učenjem novih verzija tehnologija na kojima se temelji izrada web stranica (HTML5, ali i CSS3), polaznici ovog kursa će biti u stanju da razumiju i starije verzije HTML-a i CSS-a.

Web stranicama se pristupa **korištenjem web preglednika** (eng. *web browser*), kao što su: Firefox, Chrome, IE, Safari, Opera, Microsoft Edge, itd. Da bi se neka web stranica mogla učitati u preglednik, potrebno je u pretraživač upisati web adresu. Pod uslovom da je korisnik spojen na Internet, preglednik šalje zahtjev web poslužitelju, tj. računaru na kojem se tražena stranica nalazi. Poslužitelj šalje pregledniku traženu stranicu, a preglednik stranicu prikazuje korisniku.

Web poslužitelj (engl. *web server*) je računar na kojem se nalaze web stranice, a računar se može nalaziti bilo gdje u svijetu i uvijek je spojen na Internet, kako bi mogao odgovoriti slanjem web stranice korisniku, u bilo kojem trenutku. Iako neke velike kompanije imaju dedicirane web poslužitelje na kojima se nalaze njihove stranice, većina korisnika će prilikom izrade i smještanja vlastitih web stranica ipak zakupiti prostor kod neke kompanije koja ima uslugu „udomljavanja“ web stranica, tj. **hosting** (engl. *web hosting*). Postoji veliki broj različitih uređaja sa kojih se može pristupiti web stranicama kao što su: stolni računar, laptop, tablet, mobitel... Važno je imati na umu da svi ovi uređaji imaju različitu veličinu ekrana tj. rezoluciju, te pri izradi web stranica se mora paziti kako će se one prikazivati na različitim uređajima (engl. *Responsive Design*). U ranijim fazama razvoja web-a, najvažnije je bilo znati napisati HTML kod i objaviti web stranicu. Ipak, danas je izrada web stranica postala više od samog pisanja HTML i CSS koda[4]. Povećanjem dostupnosti i pojeftinjenjem tehnologije i internetske veze, te razvojem mobilnih tehnologija, web je postao dostupan većem broju korisnika. Ovo je dovelo do povećanja broja web stranica

[3] Ukoliko želite da provjerite da li Vaš preglednik podržava određeno HTML ili CSS svojstvo, posjetite sajt: https://caniuse.com.
[4] Neki korisnici za pristupanje web stranicama koriste i čitače (engl. *screen readers*) - uređaj koji izgovara sadržaj web stranice.

iste ili slične tematike, koje su dostupne puno većem broju korisnika nego prije. Zbog povećanja konkurencije, sada je potrebno uložiti i puno više truda pri razvoju web stranica kako bi korisnici došli baš do naše web stranice. Potrebno je pažljivo osmisliti koncepciju web stranice, sadržaj, prezentaciju, jednostavnost snalaženja na stranici, prilagođenost stranice za različite uređaje ili različite kategorije korisnika i web stranica kako bi poruka koju web stranica prenosi došla do krajnjeg korisnika. Naglasak pri izradi se posebno stavlja na korisnika. Potrebno je jasno odlučiti kome je web stranica namijenjena, zašto je korisnici posjećuju, šta žele postići, koje informacije trebaju, koliko često posjećuju stranicu, gdje se zadržavaju i slično.

Male stranice, statičnog sadržaja se većinom izrađuju pisanjem HTML i CSS koda. Veće stranice, pogotovo one koje se često osvježavaju, češće se izrađuju uz pomoć sistema za upravljanje web sadržajem - **CMS**-a (engl. *Content Management System*). CMS omogućava osobi koja izrađuje web stranice[5] korištenje nekih novijih, naprednijih opcija i web tehnologija, sa manje znanja, jednostavnije održavanje sadržaja bez svakodnevnog susreta sa HTML-om i CSS-om. Međutim, znanje HTML-a (i CSS-a) uveliko pomaže čak i kod rada sa sistemima za upravljanje sadržajem, jer omogućava bolje razumijevanje pozadinskih procesa, te bolju kontrolu nad izgledom web stranica. Za izradu web stranice u CMS-u (uglavnom) nije potrebno poznavanje programskih jezika i drugih web tehnologija.

1.2. Organizacija Web stranica

Jednom kad je poznata ciljana skupina korisnika web sajta (sjedišta), možemo da razmišljamo o organizaciji web stranice ili o njezinoj informacionoj strukturi. Informacije se prvo organizuju u sekcije i stranice. Cilj procesa jeste izrada dijagrama stranica koji će se koristiti u struktuisanju web stranice. Na kraju procesa bi trebala nastati mapa web stranica (engl. *site map*), prikazuje strukturu, organizaciju i grupisanje stranica. Tehnika sortiranja kartica može biti dobra pomoć u organizaciji izrade mape. Tehnika se sastoji od toga da se na papiriće zapiše svaka pojedinačna informacija koja bi korisniku mogla biti korisna, te se zatim kartice grupišu u stranice ili u grupe stranica. Ponekad je se, u ovoj fazi, korisno konsultovati sa korisnicima kako bi se informacije lakše grupisale. Uobičajeno, svi web-ovi, kao početnu tačku imaju početnu stranicu - naslovnu (engl. *homepage*). Ako je riječ o kompleksnom web-u, moguće je imati i više početnih stranica,

[5] ***Web* stranica** (engl. *web page*) označava samo jednu web stranicu koja se od početka do kraja može pregledati bez dodatnog klikanja na linkove. Riječ je o jednom HTML dokumentu. ***Web* sjedište** (engl. *web site*) označava određeni broj tematski ili drugačije vezanih stranica. Budući da je riječ o većem broju stranica, ponekad se za web sjedište jednostavno koristi plural web stranice. Web sjedište se sastoji od više povezanih HTML dokumenata.

svaku za jednu temu. Osim organizacije stranica i sekcija, važno je napomenuti da se i sadržaj svake pojedine stranice treba organizovati vizualno. Većina korisnika ne čita cijele web stranice, nego ih samo brzinski pregledava, u potrazi za traženim informacijama. Vizualna hijerarhija se odnosi na put kojim oči korisnika „prolaze" i „doživljavaju" ono što vide, a može se **kontrolisati povećavanjem kontrasta** određenih područja za koja smatramo da su korisniku od interesa. Pažnju korisnika će uvijek privući velika slova, svijetlija područja, slike i kontrasti u tekstu (npr. podebljana slova). Dodatni naglasak na sadržaj se postiže i vizualnim grupisanjem pojedinih informacija na stranici.

1.3. Optimizacija web-stranica za tražilice (SEO)

Dobra **optimizacija** za tražilice (pronalazljivost ili SEO – *Search Engine Optimization*[6]) često može biti presudna kada je u pitanju dolazak korisnika na web stranice sa neke tražilice. SEO služi da se prilikom pretrage određenih pojmova u tražilici, web stranica pojavi bliže prvom rezultatu.

Kako bi se web stranica optimizovala za tražilice, potrebno je koristiti nekoliko tehnika. Jedna od njih je zaključiti koje pojmove korisnici zapravo traže, te pametnim smještanjem tih pojmova na određenim mjestima na web stranici povećati mogućnost da tražilica vrati link na našu web stranicu. Postoji nekoliko mjesta na koja se mogu smjestiti ključne riječi (za koje mislimo da će ih korisnici pretraživati) kako bi se povećala pronalazljivost web stranice:

- Naslov stranice - <title></title> element;
- URL (adresa stranice) – gdje je moguće da se koristite ključne riječi prilikom imenovanja dokumenata;
- Naslovi tekstova – ključne riječi se smještaju u naslov tj. unutar h elementa, tražilica će znati da je taj pojam važan za web stranicu;
- Tekst – ako je prihvatljivo, ključna riječ bi se trebala pojavljivati nekoliko puta u tekstu;
- Tekst poveznice (linka) – ključne riječi bi trebale biti i dio linka na druge stranice, ako je moguće;
- Alternativni tekst za sliku (alt) – svakoj slici bi se trebao staviti alt atribut (opisuje sliku);
- Opis stranice – <meta></meta> element.

[6] **SEO** je zaista jedno kompleksno i opsežno područje, te se ne može detaljno obuhvatiti u ovom kursu. Ako želite saznati nešto više o optimizaciji za tražilice, onda pročitajte knjigu „Razvoj pronalazljivih Web stranica", autora Arona Valtera.

Tražilice ne uzimaju u obzir samo ključne riječi, već i broj linkova sa drugih stranica na našu web stranicu. Važnijim se smatraju linkovi sa stranica koje imaju sličan sadržaj kao i naša. U slučaju da se unutar a elementa nalaze ključne riječi od interesa, link će se smatrati još važnijim. Ako na primjer imamo stranicu o školi koja održava kurseve napredne informatike, relevantniji link će biti onaj na stranici neke druge škole, nego npr. na stranici prodajnog marketa. Još važniji je ako na njemu umjesto adrese web stranice ili riječi „više“ piše tekst „kurs napredne informatike“. **Za uspješan SEO je od iznimne važnosti uspješno određivanje ključnih riječi.**

1.4. Pristupačnost

Čini se gotovo nepotrebnim spominjati pristupačnost informacijama na web stranicama, ali je potrebno uzeti u obzir da postoje ljudi kojima je teže koristiti računar, Internet pa i pregledavati web stranice. Riječ je o osobama sa posebnim potrebama. Cilj svake dobre web stranice bi trebao omogućiti svim korisnicima jednaku mogućnost pristupanja informacijama, nezavisno od njihovih mogućnosti i uređaja koje koriste. Postoji više kategorija osoba sa invaliditetom na koje bi trebalo misliti prilikom izrade web stranica, ali najteže je stranice prilagoditi slijepim i slabovidnim osobama[7]. Slijepe osobe često koriste čitače kako bi pregledavale web stranice. Čitači su uređaji koji, kao što i samo ime kaže, čitaju sve informacije sa web stranica na glas. Slijepe osobe ne mogu „preletjeti“ sadržaj i početi čitati ono što ih zanima, nego čitači za njih čitaju cijeli sadržaj dok ne dođu do sadržaja koji ih zanima. Kako bi se omogućilo „preskakanje“ sadržaja, važno je prilikom izrade web stranica koristiti HTML5 i semantičke oznake. Osim ovoga, vrlo je važno svim slikama i linkovima postaviti jasne i jednoznačne alternativne opise, te paziti da tekst linkova ima jasno značenje (izbjegavati linkove sadržaja „više“, „na stranicama“ i slično). Slabovidnim osobama je važna mogućnost povećanja i smanjenja slova na stranici, općenito, čitljivost teksta (velika slova, jasno odvojeni tekstovi, velike margine između teksta na stranici, veći proredi, povećani kontrasti između teksta i pozadine, itd.). Preporučuje se i izrada dodatnog vizualnog šablona sa povećanim kontrastom (npr. crna pozadina sa bijelim slovima ili plava pozadina sa žutim slovima). Za neke kategorije osoba sa posebnim potrebama, npr. za osobe sa poremećajem koncentracije, izuzetno je važno poravnati cijeli tekst lijevo (nikako obostrano), te imati što jasniju, jednostavniju i dosljedniju navigaciju. Osobama sa posebnim potrebama je važno da pred sobom imaju dobre, jasne i jednostavno napisane, razumljive i sažeta web sjedišta (sajtove), koji su izrađeni smisleno, i jednostavni su za korištenje, tj. stranice su korisne (engl. *high usability*).

[7] Detaljnije informacije o prilagođavanju web-a osobama sa posebnim potrebama je moguće pronaći na specijaliziranim stranicama organizacija koje se bave ovom temom; WAI, WCAG, webaim…

1.5. Prilagodljivost za različite uređaje

Kao što je već rečeno, postoji veliki broj različitih uređaja sa kojih korisnik može pristupiti web stranicama, kao što su: personalni računar, laptop, tablet, mobitel, i dr. Ovi uređaji imaju različitu veličinu ekrana tj. rezoluciju, te je potrebno paziti kako se web sajtovi prikazuju na različitim uređajima. Responzivni web dizajn (RWD) je pojam koji je već postao uobičajen, a odnosi se na izradu web stranica koje osiguravaju optimalno pregledavanje na različitim uređajima. Ovime se omogućava dobra čitljivost, jasna navigacija po stranicama bez potrebe za povećavanjem, smanjivanjem ili micanjem stranice u lijevo ili u desno, kako bi se vidio cijeli sadržaj. Ideja iza responzivnog web dizajna je da se na svim uređajima prikazuje isti sadržaj, ali da se, zavisno od uređaja na kojem se stranica gleda, izgled stranice prilagođava na način da sadržaj ostane jasan i čitljiv. Obično je riječ o nekoliko verzija istog dizajna koji se neprimjetno mijenja zavisno od veličine ekrana na kojem korisnik pregleda web stranice. Responzivnost se, uglavnom, postiže izradom posebnih CSS uslova za različite veličine ekrana. Neke stranice nude popis najčešće korištenih uređaja i veličina ekrana, tj. rezolucija kako bi osobe koje izrađuju web stranice lakše znale za koje veličine trebaju prilagoditi dizajn[8]. Dodatni problem može biti to što osim veličine ekrana postoje još i različite orijentacije (portret ili pejzaž), te ekrani sa različitim omjerima (3:4, 16:9, itd.), što može biti problem kada se izrađuje stranica koja u svojoj pozadini ima sliku.

U poglavlju je obrađeno:

- šta je HTML;
- što je web sajt i gdje se web sajtovi nalaze;
- informacijska struktura jednostavne web sajtova;
- objašnjeni pojmove SEO, pristupačnost i prilagodljivost web sajtova.

[8] Na stranici w3schools.com/browsers/browsers_display.asp je lista najčešće korištenih veličina ekrana.

II Osnove rada sa HTML-om

Po završetku ovog poglavlja, polaznik će moći:

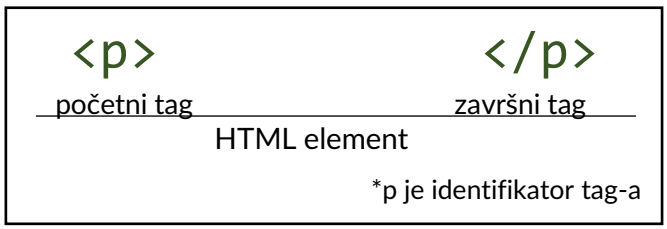
- objasniti od čega se sastoji HTML element;
- objasniti čemu služe HTML atributi;
- izraditi HTML dokument sa osnovnim elementima;
- postavljanje kodiranja u HTML dokumentu.

Kao i bilo koji drugi dokument, HTML dokument ima određenu strukturu. HTML dokument tako može imati naslov, odlomak, slike, tabele i sl. U nastavku se nalazi vrlo jednostavan primjer HTML dokumenta koji ima naslov i odlomak (HTML elementi su označeni zelenom bojom):

```
<html>
    <body>
        <h1>Academy387 Sarajevo</h1>
        <p>AKADEMIJA387 je dizajnirana kao dinamična platforma otvorena za
            nove programe, prijavljivanje na kurseve, upis studenata i
            otvorene komentare od studenata.</p>
    </body>
</html>
```

2.1. Anatomija HTML elementa

HTML element se sastoji od: **početne** i **završne HTML oznake** (engl. *tag*). Većina HTML elemenata se sastoji od početnog tag-a (npr.) i završnog tag-a (npr.). Postoje elementi koji nemaju krajnji tag (npr.
; <hr/>; <img/>, i drugi).



Sadržaj se unosi između dvije HTML oznake, te se one na neki način ponašaju poput spremnika koji pojašnjava koja vrsta informacije se nalazi u istom. Na primjer, unutar HTML oznaka i će se nalazit jedan odlomak teksta. Kako bi razjasnili anatomiju HTML-ovog elementa, potrebno ga je parcijalizovati na osnovne dijelove. Svaka početna HTML oznaka se sastoji od **šiljastih zagrada** i i**dentifikatora**. Šiljaste zagrade su karakteristične za svaki HTML element, tj. svaka HTML oznaka **mora imati dvije šiljaste zagrade**. Identifikator određuje o kojem elementu HTML-a se radi. Završni HTML element se sastoji od: **šiljastih zagrada** u kojima se nalazi **kosa**

linija (engl. *slash*) i **identifikator**. Kosa linija je oznaka koja pokazuje da je to završni element[9], a ne početni. Na primjeru jednostavnog HTML koda koji se sastoji od jednog odlomka teksta se može vidjeti kako izgleda ispravno napisan odlomak teksta u HTML-u:

```
<p>AKADEMIJA387 je zamišljena kao otvorena edukacijska platforma za sve
    pojedince i ustanove sa zajedničnikm ciljem dijeljenja znanja. Primarni
    fokus je na stvaranju intenzivnih i kratkih kurseva, radionica i lekcija
    kreiranih od strane profesionalaca i svjetskih lidera. Programi su
    dizajnirani kao direktan odgovor na porast tržišne potrebe za određenim
    vještinama, poznavanjem tehnologija i znanjem.</p>
```

Kako bi kod bio čitljiviji, piše se na način da svaki HTML element bude u svom redu. Postoje elementi poput naslova h1 ili odlomka p, koji u sebi mogu imati sadržaj, ali postoje i elementi koji nemaju nikakav sadržaj, poput horizontalne linije hr koja se proteže preko cijele stranice i predstavlja tematski prijelom između odlomaka ili oznake za prelaz u novi red. Na prethodnom primjeru se može vidjeti da se element za odlomak otvara na način da se ispred sadržaja napiše , zatim slijedi tekst nakon čega se element naslova zatvara sa . HTML elementi koji nemaju sadržaj imaju samo jednu HTML oznaku. HTML element za vodoravnu liniju, sa obzirom na to da nema sadržaja ima samo jednu oznaku <hr/>, dok HTML element za prelazak u novi red ima samo oznaku <br/>.

2.2. HTML atributi

Neki HTML elementi mogu imati dodatna svojstva (unutar početne HTML oznake) koja daju više informacija o sadržaju elementa, a najčešće se sastoje od dva dijela: **imena** i **vrijednosti**, odvojeni znakom „=“. Ta svojstva se nazivaju **HTML atributima**. Jedan HTML element može imati više atributa. Ime atributa je podatak koji govori o kojoj vrsti informacije je riječ u atributu, dok je svojstvo ili informacija vrijednost atributa. Ime atributa bi se trebalo pisati malim slovima, a vrijednost atributa se uvijek mora nalaziti u dvostrukim navodnicima. Većina atributa se koristi sa tačno određenim HTML elementom, pa nije striktna obaveza. Ipak, postoje HTML elementi koji moraju imati i atribut, jer u suprotnom nije moguća pravilna interpretacija elementa. Primjer neizostavnog atributa je src (engl. *source)*, a koristi se za označavanje putanje do slike. HTML element se sa skupom atributa može vidjeti na primjeru elementa za umetanje slike <img/>. U ovome primjeru oznaka ima tri atributa. Na primjer:

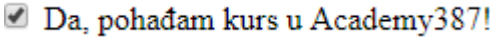
[9] Donedavno su svi HTML elementi (i oni koji imaju po dvije oznake, i oni koji imaju samo jednu) obavezno trebali zatvarati, jer je preporuka bila pisati kod prema **XHTML standardu**. Međutim, sa dolaskom HTML5 standarda, elementi koji imaju samo jednu oznaku se više ne trebaju zatvarati, iako je to i dalje moguće; ako web treba proći strožu validaciju. Budući da je riječ o jednoj oznaci koja čini cijeli HTML element, kosa linija se stavlja iza identifikatora, a ne ispred, pa se u tom slučaju piše <hr/> i <br/>.

```
<img src="img/academy387-logo.png" height="30" width="30"/>
```

Atribut:	Ime atributa:	Vrijednost atributa:
src="img/academy387-logo.png"	src	img/academy387-logo.png
height="30"	height	30
width="30"	width	30

Kod nekih atributa se vrijednost uopće ne navodi. Radi se o atributima koji imaju samo dvije vrijednosti – **istina** (engl. *true*) i **laž** (engl. *false*), odnosno atribut je naveden ili nije naveden. Ovakvi atributi se još nazivaju **Boole-ovim atributima**. U sljedećem primjeru prisutnost atributa checked označava da će kvačica prvobitno biti prikazana kao označena. Na primjer:

```
<input type="checkbox" checked> Da, pohađam kurs u Academy387!
```



Ako želimo da kvačica inicijalno bude isključena, onda je potrebno izostaviti atribut checked. Određene atribute (tzv. globalne atribute) je moguće postaviti na bilo koji HTML-ov element. Objašnjenje najčešće korištenih globalnih atributa je dano u slijedećoj tabeli:

Atribut	Objašnjenje
id	Identifikator elementa koji se koristi za referenciranje elementa u CSS-u i JavaScript-u. Treba biti jedinstven na razini stranice.
class	Klasa elementa koja se koristi za referenciranje elementa u CSS-u i JavaScript-u. Na stranici može biti više elemenata koji imaju istu klasu.
title	Dodatne informacije o elementu. Najčešće se radi o opisu elementa koji će biti prikazan kada korisnik prijeđe mišem preko elementa (u tooltip-u).
style	Atribut preko kojega je moguće postaviti inline CSS pravila na neki element.
lang	Jezik elementa. Najčešće se postavlja na razini cijele stranice tako da se postavi na elementu HTML, ali ga je moguće postaviti i na drugim elementima. Vrijednosti oznake jezika su npr.: bs, sr, en, de,...

2.3. DOCTYPE deklaracija

DOCTYPE deklaracija opisuje koja se verzija HTML-a koristi u dokumentu. Potrebno je navesti na samom početku HTML dokumenta[10], prije svih drugih elemenata. DOCTYPE deklaracija **nije HTML element**, nego instrukcija pregledniku, da bi preglednik znao koju verziju HTML-a treba interpretirati. Dolaskom verzije HTML5, DOCTYPE[11] deklaracija je postala jednostavna. Dakle, dovoljno je na početku dokumenta napisati slijedeće:

```
<!DOCTYPE html>
```

[10] DOCTYPE deklaracija za HTML verziju HTML 4.01 Strict, bi izgledala ovako :`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`

[11] Za pisanje koda koristimo VS Code, možete ga preuzeti na linku: code.visualstudio.com/download

2.4. Osnovna struktura HTML dokumenta

Svi HTML dokumenti, nezavisno od kompleksnosti i sadržaja, imaju neke zajedničke dijelove, pomoću kojih se definiše **struktura HTML dokumenta**. Elementi koji su zajednički svim HTML dokumentima su: <html></html>, <head></head>, <title></title> i <body></body>. Dakle, element koji je obavezan na svakoj stranici je html. To je prvi element, unutar koje se nalazi cijeli sadržaj nekog HTML dokumenta, a služi kako bi web preglednik „znao" prepoznati da li je riječ o HTML dokumentu. Budući da je cijeli sadržaj dokumenta unutar ovog elementa, cijeli dokument počinje i završava oznakama <html> i </html>. Na primjer:

```
<html>
<head>
<title>HTML i CSS</title>
</head>
<body>
</body>
</html>
```

Na vrhu HTML dokumenta se navodi HTML element head, on sadrži informacije o stranici, koje se ne prikazuju kao dio sadržaja stranice. Taj element predstavlja zaglavlje HTML dokumenta. U head elementu se nalazi element title u kojeg pišemo naslov web stranice. Naslov stranice se prikazuje na naslovnoj traci preglednika i ne mora biti identičan naslovu HTML dokumenta koji se pohranjuje lokalno, na računaru. Cijeli sadržaj, koji se nalazi unutar elementa body, je sadržaj koji će se prikazati u internetskom pretraživaču. U prethodnom primjeru će se prikazat prazna stranica, bez sadržaja[12]. Kada se između oznaka body upiše tekst, on će postati sadržaj stranice, pa će u pregledniku biti vidljiv. HTML dokument se čuva, za početak, lokalno na računaru kao datoteka sa ekstenzijom **.html** ili **.htm**; npr. html-i-css.html, odnosno html-i-css.htm. Sa obzirom na to da su HTML dokumenti zapravo obične tekstualne datoteke koje se sastoje od HTML koda, preporuka je da se kod piše struktuisano, tj. uvučeno radi preglednosti, kako bi se kasnije u dokumentu lakše snalazili[13].

[12] Kod bilo koje HTML stranice, i one koju nismo sami izradili, može se vidjeti iz internetskog preglednika, npr. u programu Google Chrome; desnim klikom na stranicu se pristupi tekstualnom izborniku i odabere Izvor (engl. *View Source*).

[13] HTML kod se može pisati uvučeno zbog toga što web preglednik ignoriše dodatne razmake i prelaske u novi red. Ovo je poznato pod nazivom „white space collapsing", a autori HTML dokumenata ga koriste kako bi pisali pregledniji HTML kod.

2.5. Informacije o stranici (meta podaci)

Meta podaci su različiti (dodatni) podaci o web stranici, koji krajnjem korisniku nisu vidljivi, a mogu imati različitu svrhu. Meta podaci mogu da sadržavaju: popis ključnih riječi, opis stranice, informacije o autoru i dr. informacije, koje web stranica šalje tražilicama, radi bolje pretraživosti i optimizacije. Meta podaci se pišu unutar head elementa. Za označavanje meta podataka se koristi HTML element meta koji nema završnu oznaku. Vrijednost meta podatka se upisuje u atribut. Najčešći atributi su name ili content atributi koji se često koriste zajedno i označavaju svojstva cijele HTML stranice. Vrijednost atributa name je svojstvo koje se postavlja, dok je vrijednost atributa content, vrijednost koju želimo dodijeliti tom svojstvu, odnosno specifikacija atributa name. Vrijednost atributa name se može proizvoljno postaviti, ali najčešće se koriste:

- content-type – dodatno **deklarisnje kodne stranice**;
- description – **opis stranice do 155 karaktera** koji tražilicama olakšava „razumijevanje“ sadržaja na stranici, kao rezultat pretrage u tražilicama se ispisuje ime stranice i opis koji je na ovom mjestu postavljen;
- keywords – **popis ključnih riječi**, odvojenih zarezom, za koje pretpostavljamo da će ih korisnik koristiti za pretraživanje u tražilicama, ovo je prije bio vrlo važan korak za SEO, danas sve više gubi na važnosti;
- author – definiše se **autor stranice**.

Primjer korištenja meta podataka može biti slijedeći:

```
<html>
<head>
    <title>Academy387</title>
    <meta name="description" content="Kurs HTML-a i CSS-a u Academy387"/>
</head>
<body>
</body>
</html>
```

2.6. Postavljanje kodiranja stranice

Kodna stranica određuje koja kombinacija nula i jedinica se koristi za zapisivanje znaka. Znakovi engleskog alfabeta imaju iste kombinacije u svim kodnim stranicama, pa se zato uvijek ispravno prikazuju, ali do problema dolazi kada je potrebno prikazati posebne znakove koji se koriste u drugim jezicima. Da bi se bosansko-hercegovački dijakritički znakovi prikazivali ispravno, tada je potrebno postaviti pravu kodnu stranicu koja ih sadrži. Najbolji izbor je kodna stranica **UTF-8**, koja može prikazati skoro sve svjetske nacionalne znakove. Kodna stranica se zadaje pomoću elementa meta sa atributom charset. Ovaj element dolazi unutar elementa <head>. Na primjer:

```
<head>
    ...
    <meta charset="utf-8">
</head>
```

Osim kodne stranica UTF-8[14], za prikaz bosansko-hercegovačkih znakova se često koriste i kodne stranice Windows-1250 i ISO-8859-2. Na primjer:

```
<head>
    ...
    <meta charset="windows-1250">
</head>
```

Ako se HTML datoteka ne prikazuje ispravno, korisnik može i sam promijeniti kodnu stranicu u pregledniku (pomoću opcije View → Text Encoding ili slično).

2.7. Komentari

HTML dokument može sadržavati i **komentare** koji nam omogućavaju unošenje sadržaja koji nije vidljiv krajnjem korisniku. Komentari služe kako bi osoba koja izrađuje web stranicu imala mogućnost zabilježiti pojašnjenja koja su vezana uz neki dio stranice ili koda u samom HTML dokumentu. Na taj način se olakšava naknadni rad na dokumentu. Također, mogu služiti za obilježavanje dijelova koda radi jednostavnije i brže preglednosti. Npr. komentarom se može označiti zaglavlje i podnožje web stranice ili odlomak teksta. Krajnjem korisniku, komentari neće biti prikazani, a osobi koja izrađuje web stranicu će omogućiti lakši pronalazak onog dijela koda u kojem želi napraviti intervenciju. Osim ovoga, komentari mogu biti i prilično korisni u slučaju potrebe testiranja. Ako se želi vidjeti kako bi stranica izgledala bez nekog dijela sadržaja, dovoljno je prije tog sadržaja staviti početnu oznaku komentara, a nakon njega završnu oznaku komentara, kako bi sadržaj koji ne želimo prikazivati bio skriven. Taj dio sadržaja se tada neće prikazati. Čim se **oznake komentara obrišu, on će opet biti vidljiv**. Da bi se dio koda sakrio ili pretvorio u komentare, dodajemo HTML oznake za komentar. Početna oznaka `<!--` se dodaje neposredno prije dijela koda koji se želi pretvoriti u komentare, a završna oznaka `-->` se dodaje poslije dijela koda koji se želi pretvoriti u komentare, tj. sakriti od prikaza krajnjem korisniku. Na primjer:

```
<!-- Ovo je pet stiha iz pjesme "Mala velika moja, večeras ćemo za njih
voljeti" Izeta Kike Sarajlića, ispisani u pet redova -->
<p>A na Kalemegdanima i Nevskim Prospektima,</br>
na Južnim Bulevarima i Kejovima Rastanka,</br>
```

[14] Bitno je da sama HTML datoteka bude spremljena na računaru u istoj kodnoj stranici.

```
na cvijetnim trgovima i Mostovima Mirabo,</br>
divne i kad ne ljube,</br>
čekale su Ane, Zoje, Žanet.</p>
```

A na Kalemegdanima i Nevskim Prospektima,
na Južnim Bulevarima i Kejovima Rastanka,
na cvijetnim trgovima i Mostovima Mirabo,
divne i kad ne ljube,
čekale su Ane, Zoje, Žanet.

U poglavlju je obrađeno:

- od čega se sastoji HTML element;
- čemu služe HTML atributi;
- izrada HTML dokumenta sa osnovnim elementima;
- postavljanje kodiranja u HTML dokumentu.

III Rad sa tekstom

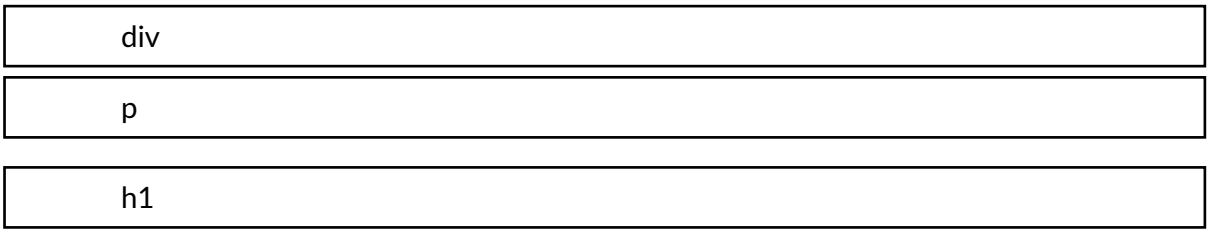
Po završetku ovog poglavlja polaznik će moći:

- objasniti razliku između blok i inline elementa;
- izraditi osnovni tekstualni dokument u HTML-u;
- koristiti posebne znakove i predformatirani tekst;
- razlikovati stilske i semantičke HTML elemente za oblikovanje teksta.

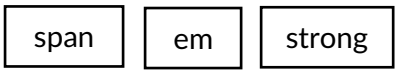
HTML dokumenti se mogu sastojati od teksta, slika, linkova, animacija i drugih sadržaja. U ovom poglavlju će biti objašnjeno kako se u HTML dokumentu radi sa tekstom, kako se taj tekst na kraju prikazuje u web pregledniku, te može li taj tekst nositi i neke dodatne informacije.

3.1. Blok i inline elementi

Prema načinu prikaza, u HTML-u postoje dvije osnovne vrste elemenata[15]. Prva vrsta su **blok elementi** *(*engl. *block elements)*. Oni zauzimaju cijeli red u kojem se nalaze. Moguće im je odrediti visinu i širinu. Čak i ako im se smanji širina, oni će i dalje zauzimati cijeli red, pa će zato naredni element započeti u novom redu.



Primjeri takvih elemenata su div, p i h1. Element div služi za grupisanje sadržaja (teksta i HTML elemenata), element p služi za definisanje odlomka, a element h1 služi da bi postavili naslov prvog nivoa. Element div, odlomak i naslov će da zauzmu cijeli red, odnosno cjelokupnu širinu preglednika. Druga vrsta su **linijski elementi** (engl. *inline elements*). Takvi elementi su prikazani unutar trenutne linije teksta. Oni ne zauzimaju cijeli red nego onoliko prostora koliko zauzima njihov sadržaj:



[15] Ponašanje pojedinog elementa je moguće promijeniti korištenjem CSS svojstva display.

Primjeri takvih elemenata su span, em i strong. Element span služi za označavanje dijela teksta (najčešće kako bi se na njega mogao primijeniti određeni CSS stil), dok elementi em i strong služe također za označavanje, odnosno naglašavanje dijela teksta, te se prikazuje sa stilom koji je predefinisan. Ovi elementi neće uzrokovati prelaženje u novi red (engl. *new line*), već njihova veličina zavisi od sadržaja (prvenstveno teksta) koji se nalazi unutar njih.

3.2. Naslovi – <h1-6> element

Različite tematske cjeline dokumenta se najčešće odvajaju naslovima. HTML ima šest nivoa naslova i to: h1, h2, h3, h4, h5 i h6. **Prvi nivo naslova** h1 se koristi za najvažnije naslove, naslove stranica, kao glavni naslov dokumenta i sl. Prikazuje se najvećom veličinom slova. **Zadnji nivo naslova** h6 se ispisuje najmanjom veličinom slova. Na primjer:

```
<h1>Academy387</h1>
<h2>AKADEMIJA387 pruža neophodne treninge firmama iz oblasti IKT-a</h2>
<h3>AKADEMIJA387 je dizajnirana kao dinamična platforma</h3>
<h4>Web dizajn</h4>
<h5>HTML i CSS</h5>
<h6>Web tehnologije</h6>
```

Academy387

AKADEMIJA387 pruža neophodne treninge postojećim firmama iz oblasti IKT-a

AKADEMIJA387 je dizajnirana kao dinamična platforma

Web dizajn

HTML i CSS

Web tehnologije

3.3. Paragraf – element

Odlomci su tematske, odnosno logičke cjeline sadržaja koje se u HTML dokumentu definišu blok elementom odlomak p (engl. *paragraph*). Element p obavezno mora imati i završnu oznaku . Svaki preglednik će prikazat odlomak u novom redu, a između odlomka i ostalog sadržaja će postaviti razmak. Na primjer:

```
<p>Budi toliko dobar da te ne mogu ignorisati!</p>
```

Budi toliko dobar da te ne mogu ignorisati!

3.4. Prelazak u novi red – </br> element

U slučaju da je unutar jednog odlomka potrebno započeti tekst u sljedećem redu (bez dodatnih razmaka između reda i ostalog teksta), koristit će se element za prelazak u novi red
 (engl. *break row* ili *breaking rule*). Na primjer:

```
<p>Mala velika moja,<br/>
večeras ćemo za njih voljeti.</p>
```

Mala velika moja,
večeras ćemo za njih voljeti.

3.5. Vodoravna linija – <hr/>

Kada je neki dio stranice potrebno logički odvojiti od prethodnog, pa je potrebno odvajanje dodatno naglasiti, može se koristiti **vodoravna linija** – <hr/>. Kao i element za prelazak u novi red, vodoravna linija nema završnu oznaku, već se koristi samo <hr/>. Riječ je o blok elementu, što znači da će se vodoravna linija uvijek postaviti u poseban red. Na primjer:

```
<h3>Web dizajn</h3>
<hr/>
<p>HTML i CSS</p>
```

Web dizajn

HTML i CSS

3.6. Specijalni znakovi i entiteti

Postoje **specijalni znakovi** koji se koriste za oblikovanje i definisanje samog HTML dokumenta, npr. šiljaste zagrade (<>), koje će HTML dokument uvijek prepoznavati kao početak nove HTML oznake. Dodatni primjer specijalnih znakova su i znakovi koji su inicijalno sastavni dio HTML dokumenta, kao što je npr. razmak. U slučaju kada se želi prikazati više od jednog razmaka za redom, potrebno je koristiti specijalne znakove, jer će HTML inače uvijek ignorisati sve osim jednog razmaka, tj. prikazat će se samo jedan razmak. Postavlja se pitanje kako te specijalne znakove zapisati kao dio teksta na web stranici. Takvi specijalni znakovi se prikazuju pomoću **HTML entiteta** – posebnih kodova za zapisivanje tih specijalnih znakova u HTML dokumentu. Ti kodovi će se na web-u prikazati u obliku u kojem ih želimo prikazati. Svaki od tih specijalnih znakova u HTML dokumentu se može zapisati na dva načina: **korištenjem alfanumeričkog koda** i **korištenjem brojčanog koda**[16].

[16] Prilikom upotrebe ovih znakova je potrebno provjeriti kako se prikazuju u web pregledniku, sa obzirom da neki od fontova koji se koriste ne podržavaju ove znakove. U tom slučaju je u CSS-u potrebno postaviti onaj font koji će ove znakove moći prikazati.

Kod	Brojčani kod	Znak	Opis
&quot;	&#34;	"	dvostruki navodnici
&amp;	&#38;	&	engl. *amperstand*
&lt;	&#60;	<	manje od
&gt	&#62;	>	veće od
&pound;	&#163;	£	Funta
&euro;	&#8364	€	Euro
spisak ostalih entiteta možete pronaći na linku: freeformatter.com/html-entities.html			

3.7. Predformatirani tekst – <pre>

U slučaju da se u tekstu želi zadržati velik broj razmaka, radi čega je korištenje specijalnih HTML znakova nezgodno, ili se želi prikazati tekst koji ima tačno onakvo oblikovanje kakvo je imao npr. u Word dokumentu iz kojeg je kopiran, koristiti se predformatirani tekst – pre.

```
<pre>I ne pitaj je li se moglo natrag dok je posljednji put,
    crven kao komunizam, gorio horizont njihovih želja.
            Preko njihovih neljubljenih godina izbodena i uspravna prešla je
budućnost ljubavi.
Nije bilo tajni o polegnutoj travi.
            Nije bilo tajni o raskopčanoj bluzi.</pre>
```

```
I ne pitaj je li se moglo natrag dok je posljednji put,
    crven kao komunizam, gorio horizont njihovih želja.
            Preko njihovih neljubljenih godina izbodena i uspravna prešla je budućnost ljubavi.
Nije bilo tajni o polegnutoj travi.
            Nije bilo tajni o raskopčanoj bluzi.
```

3.8. Supskript i superskript – <sub> i <sup> elementi

Za neke posebne znakove, kao što su hemijske, fizikalne formule ili matematičke formule, oznake indeksa i eksponenta, potrebno je koristiti supskript ili superskript, kako bi dio formule bio ispravno prikazan. Npr. ako na web stranici treba napisati $E = MC^2$, koristit će se **superskript** – sup, dok će se za zapis H_2SO_4 koristi **supskript** – sub.

```
<p>Terorija relativiteta: E = mc <sup>2</sup> </p>
<p>Salicidna kiselina: C <sub>7</sub> H <sub>6</sub> O <sub>3</sub></p>
```

Terorija relativiteta: $E = mc^2$

Salicidna kiselina: $C_7 H_6 O_3$

3.9. Stilski elementi – <b>, <i>, <u>

Ako se želi promijeniti grafički izgled nekog dijela dokumenta ili teksta, koriste se stilski HTML elementi. Tako je tekst moguće podebljati, nakositi ili podcrtati korištenjem stilskih elemenata podebljanje – b, kurziv – i i podvučeno – u.

```
<p>Ovo je <b>podebljani</b> tekst.</p>
<p>Ovo je <i>nakošeni</i> tekst.</p>
<p>Ovo je <u>podvučeni</u> tekst.</p>
```

Ovo je **podebljani** tekst.

Ovo je *nakošeni* tekst.

Ovo je podvučeni tekst.

3.10. Semantički elementi – , , <blockquote>, <q>, <cite>, <abbr> i <s>

Postoje neki elementi čija namjena nije da promijene izgled ili strukturu web stranica, nego da dijelovima stranice pridodaju dodatne informacije. Riječ je o semantičkim HTML elementima. Oni služe za naglašavanje pojedinog dijela teksta, umetanje citata, označavanje skraćenica i slično. Njih će web preglednik, često prikazivati na drugačiji način nego ostatak teksta, ali ih se zbog toga se ne trebaju koristiti kao elementi za oblikovanje teksta. Kao što je to prikladno, preporučuje se koristiti semantičke elemente umjesto stilskih elemenata. Semantički elementi nose informaciju o značenju sadržaja, te su zbog toga vrlo korisni za optimizaciju web stranica. Osim ovoga, druga, vrlo važna namjena semantičkih elemenata je olakšavanje korištenja web stranica u programima koji nisu preglednici.

3.10.1. Semantički element –

Npr. čitači koje koriste slijepe i slabovidne osobe će semantičke elemente interpretirati na način da će npr. strong element snažno naglasiti. HTML element strong označava da sadržaj ima visoku važnost, te da treba na njega obratiti posebnu pažnju. Element se prikazuje na isti način kao i stilski element b, ali nosi dodatno značenje. Na primjer:

```
<p><strong>PAŽNJA: </strong>Visok napon!</p>
```

PAŽNJA: Visok napon!

3.10.2. Semantički element –

Element naglašavanja – em (engl. *emphasis*) označava da bi naglasak trebao biti na tom dijelu teksta. Takav naglasak može suptilno mijenjati značenje rečenice. Preglednik će takav element prikazati na isti način kao i stilski element i (engl. *italic*). Na primjer:

```
<p>Ovo je <em>kurziv.</em></p>
```

Ovo je *kurziv.*

3.10.3. Semantički element – <blockquote>

Za citiranje teksta se koristi blockquote – blok element koji se prikazuje kao uvučeni tekst. Za uvlačenje teksta koji nije citat, treba da se koristiti CSS deklaracija. Unutar blockquote elementa je potrebno koristiti element za oznaku paragrafa p. Na primjer:

```
<p>
    <blockquote cite="https://bs.wikiquote.org/wiki/Titus_Maccius_Plautus">
    Čovjek je čovjeku vuk.
    </blockquote>
</p>
```

Čovjek je čovjeku vuk.

3.10.4. Semantički element – <q>

Za citiranje sadržaja koji se nalazi usred rečenice je potrebno koristiti inline element q. Većina preglednika će oko teksta označenog sa HTML elementom q inicijalno postaviti navodnike. Ali, neki neće, pa je potrebno provjeriti kako se citat prikazuje. Elementi za citiranje: blockquote i q mogu imati atribut cite. Njegova vrijednost je URL koji vodi do izvora citata ili do stranice sa više informacija o citatu. Na primjer:

```
<p>Alan Turing said:
    <q cite="https://www.hollywoodreporter.com/news/alan-turings-5-powerful-
quotes-752669">It seems probable that once the machine thinking method had
started, it would not take long to outstrip our feeble powers," he said during
a lecture in 1951.<q></p>
```

"It seems probable that once the machine thinking method had started, it would not take long to outstrip our feeble powers," he said during a lecture in 1951."

3.10.5. Semantički element – <cite>

Ako se u tekstu referira na neko tuđe djelo, kao što je knjiga ili istraživanje, moguće je označiti naziv tog djela HTML elementom cite, kako bi bilo jasno da je to izvor reference. Dakle, web preglednik će tekst unutar oznaka cite prikazati kao nakošeni tekst. Na primjer:

```
<p><cite>Kratka historija vremena </cite>Stivena Hoskinga je prodana u deset
milijona primjeraka!</p>
```

Kratka historija vremena Stivena Hoskinga je prodana u deset milijona primjeraka!

3.10.6. Semantički element – <abbr>

Ako želimo koristi skraćenicu ili akronim, moguće je to naznačiti elementom abbr. U atribut title je moguće upisati puno značenje skraćenice ili akronima. Tekst u web pregledniku će izgledati nepromijenjeno. Na primjer:

```
<p>dr<abbr title="profesor">Slavko Pokorni</abbr>: Laboratorijske vježbe iz
Osnova elektrotehnike (praktikum), Eleketotehnički falultet Univerziteta u
Sarajevu, 1988.</p>
```

drSlavko Pokorni: Laboratorijske vježbe iz Osnova elektrotehnike (praktikum),
Eleketo profesor falultet Univerziteta u Sarajevu, 1988.

3.10.7. Semantički element – s

Element s označava da nešto više nije tačno ili relevantno, ali da i dalje ne bi trebalo biti obrisano sa web stranice. Na web stranici će se taj sadržaj prikazivati prekriženo. Najčešće u upotrebi je primjer korištenja cijena proizvoda prije sniženja.

```
<p>Ženske čizmice</p>
<p><s>Cijena 500,00 BAM</s></p>
<p>Sada za 350,00 BAM</p
```

Ženske čizmice

~~Cijena 500,00 BAM~~

Sada za 350,00 BAM

U poglavlju je obrađeno:

- razlika između blok i in-line elementa;
- osnovni tekstualni dokument u HTML-u;
- korištenje posebnih znakova i predformatirani tekst;
- razlika između stilskih i semantičkih HTML elementa za oblikovanje teksta.

IV Liste

Po završetku ovog poglavlja, polaznik će moći:

- objasniti razliku između uređene i neuređene liste;
- objasniti čemu služi definicijska lista;
- napraviti različite vrste lista u HTML-u.

Liste (popisi) su nizovi elemenata koji su međusobno povezani u smislenu grupu. Liste po načinu označavanja i svrsi mogu biti:

1. numerisane (uređene) liste – elementi su označeni rednim brojem;
2. neuređene liste – elementi su označeni simbolom;
3. definicijske liste – elementi popisa sastoje se od pojmova i njihovih definicija;
4. ugniježđene liste – popisi koji imaju više nivoa.

4.1. Uređene (numerisane) liste

Numerisani popisi su liste čiji elementi su označeni rednim brojevima ili slovima. U praksi se najčešće koriste kada je potrebno navesti slijed elemenata u popisu, npr. za rang liste, upute za korištenje, kada je potrebno navesti slijed koraka i sl. Numerisani popis se izrađuje uz pomoć HTML elementa ol (engl. *ordered list*), a za definisanje pojedinih elemenata liste tj. popisa, koristi se element li (engl. *list item*). Na primjer:

```
<h3>Smjesa za palačinke:</h3>
<ol>
    <li>2 jajeta,</li>
    <li>150 g brašna,</li>
    <li>300 ml mlijeka</li>
</ol>
```

Smjesa za palačinke:

1. 2 jajeta,
2. 150 g brašna,
3. 300 ml mlijeka

Atributi specifični za numerisani popis:

type – tip numerisane liste: 1 (arapski brojevi); a (mala slova alfabeta); A (velika slova alfabeta); i (mali rimski brojevi); I (veliki rimski brojevi) – preporučeno je za određivanje vrste numeracije koristi CSS;

start – omogućuje promjenu početnog broja;

reversed – omogućuje numeraciju elemenata od većeg prema manjem broju.

4.2. Neuređena lista

Neuređene liste su oni popisi čiji su elementi označeni simbolima ili znakovima. U praksi se najčešće koriste kada nije potrebno navesti slijed elemenata u popisu, tj. kada su svi elementi popisa jednako važni ili nemaju hijerarhijski odnos. Popis sa grafičkim oznakama se izrađuje pomoću HTML elementa ul (engl. *unordered list*), a za definisanje pojedinih elemenata popisa koristi se element li (engl. *list item*). Na primjer:

```
<ul>
    <li>Jaja</li>
    <li>Mlijeko</li>
    <li>Novine</li>
</ul>
```

- Jaja
- Mlijeko
- Novine

Prije se koristio atribut type da bi se odredila vrsta popisa sa grafičkim oznakama, tj. da bi se odredilo da li će se za oznaku slijedećeg elementa koristiti linija, kružić, strelica ili drugi simbol; sada je preporučeno da se za određivanje vrste simbola koristi CSS.

4.3. Definicijska lista

Definicijska lista je onaj popisi u kojim se svaki element popisa sastoji od pojma i definicije koji najčešće dolaze u parovima, tj. definicijskog izraza dt (engl. *definition term*), te same definicije dd (engl. *definition description*). Ponekad je moguće da jedan pojam ima više definicija, ili da se više definicija veže uz jedan pojam. Definicijski popisi se mogu koristiti za kratka objašnjenja i specijalizovane popise, te rječnike na web stranicama. Definicijski popis se izrađuje pomoću HTML elementa dl (engl. *definition list*). Popis će se prikazati tako da će definicija uvijek biti malo uvučenija u odnosu na definicijski izraz, a elementi definicijske liste neće biti dodatno označeni ni brojevima ni simbolima. Na primjer:

```
<dl>
    <dt>Uređena lista</dt>
    <dd>Uređena lista je popis čiji elementi su označeni rednim brojevima ili
        slovima.</dd>
    <dt>Neuređena lista</dt>
    <dd>Neuređene liste su oni popisi čiji su elementi označeni simbolima ili
        znakovima.</dd>
    <dt>Definicijska lista</dt>
    <dd>Definicijska lista su oni popisi u kojima se svaki element popisa
    sastoji od pojma i definicije koji najčešće dolaze u parovima.</dd>
</dl>
```

Vrste lista su:

Uređena lista
Uređena lista je popis čiji elementi su označeni rednim brojevima ili slovima.
Neuređena lista
Neuređene liste su oni popisi čiji su elementi označeni simbolima ili znakovima.
Definicijska lista
Definicijska lista su oni popisi u kojima se svaki element popisa sastoji od pojma i definicije koji najčešće dolaze u parovima.

4.4. Ugnježđivanje lista

Unutar li elementa je moguće dodati novi popis, a se želi izraditi **ugniježđeni popis**. Ugniježđeni popisi se koriste kako bi se dodatno raspisao neki element popisa. Vrlo često se koristi i kao način za izradu navigacije za web stranicu. Ugniježđeni popis će se u pregledniku prikazati uvučenije od osnovnog popisa. Popisi se mogu ugnježđivati do proizvoljnog nivoa. Na primjer:

```
<ul>
    <li>Front End Development</li><ul>
        <li>HTML & CSS</li>
        <li>JavaScript</li>
        <li>React.js, Angular.js,...</li></ul>
    <li>Back End Development</li>
    <li>Database Engineer</li>
    <li>Software Engineer</li>
</ul>
```

- Front End Development
  - HTML & CSS
  - JavaScript
  - React.js, Angular.js,...
- Back End Development
- Database Engineer
- Software Engineer

U poglavlju je obrađeno:

- razlika između uređene i neuređene liste;
- čemu služi definicijska lista;
- različite vrste lista u HTML-u.

V Linkovi

Po završetku ovog poglavlja, polaznik će moći:

- objasniti šta su linkovi i čemu služe;
- izraditi više vrsta linkova u HTML dokumentu (linkovi koji se otvaraju u novom prozoru, likovi koji vode sa jedne na drugu stranicu, likovi koji vode na neko drugo mjesto na istoj stranici,...);
- objasniti kako linkovi utječu na funkcionalnost web sjedišta.

Osim prikazivanja sadržaja, glavna odlika svake web stranice je mogućnost povezivanja dijelova sadržaja, te prelazak sa jedne stranice na drugu stranicu. Bez ovoga svojstva svake web stranice, mogućnost pregledavanja stranica ili surfanje ne bi uopće postojalo. Način na koji se sa jedne stranice može preći na drugu odvija se pomoću linka ili veze (poveznice). Postoji nekoliko vrsta linkova:

- linkovi koji vode sa jedne web stranice na drugu;
- linkovi koji vode sa jedne stranice na drugu, unutar istog web sjedišta;
- linkovi koji vode sa jednog dijela teksta na drugi dio teksta na istoj stranici;
- linkovi koji se otvaraju u novom prozoru preglednika;
- linkovi koji pokreću program za slanje e-pošte.

5.1. Link i njegov atribut – <a> element

Linkovi se izrađuju korištenjem HTML elementa a. Bilo kakav tekst (ili neki drugi element kao na primjer slika) koji se nalazi između početne oznake <a> i završne oznake , postat će link, što znači da će se na taj tekst ili sliku moći kliknuti mišem[17]. Klikom na link prelazi se na neko drugo mjesto. O kojem je mjestu riječ, definiše se korištenjem atributa href. Atribut mora imati formu oblika href = "URL". Na primjer:

```
<a href="http://www.academy387.com">Academy387 Sarajevo</a>
```

Academy387 Sarajevo

Tekst između početne i završne oznake je tekst linka i on će se na web stranici prikazati tako kako je napisan. Gdje god je moguće tekst linka bi trebao biti jasan i objašnjavati na koju stranicu link vodi, a trebalo bi se izbjegavati korištenje riječi kao što su "više" ili "ovdje". Mnogo korisnika

[17] Zbog prilagođavanja web-a za osobe sa posebnim potrebama, potrebno je linkove učiniti razumljivima i izvan konteksta, same za sebe. Pri tome, nije potrebno tekstom dodatno naglašavati da je riječ o linku. Kao tekst linka se nikada ne bi trebao navoditi cijeli URL, zbog toga što čitači koje koriste slijepe i slabovidne osobe, linkove čitaju „slovkajući“, pa bi u ovom slučaju čitali „duplo v, duplo v, duplo v...“.

pregledava veće dijelove teksta na pojedinoj stranici tražeći samo linkove, pa je iz tog razloga važno da su linkovi jasno označeni i vidljivi. Prilikom osmišljavanja teksta za link, potrebno je pokušati razmišljati iz uloge osobe koja gleda web stranicu i smisliti koje bi riječi neko vezao uz stranicu na koju link ide. Npr. ako je riječ o linku na neki turistički objekat ili smještaj u Neumu, bolje bi bilo da tekst linka bude „hotel u Neumu“, umjesto „gdje odsjesti“ i sl. Ako link vodi na neko drugo web sjedište, vrijednost atributa href treba biti apsolutni URL[18]. Apsolutni URL je ona web adresa koja bi se upisala u preglednik kada se želi otvoriti neka stranica. Prvi dio takvog URL-a je ime domene određenog web sjedišta, a može još imati i putanju do tačno određene stranice.

```
<a href="http://www.academy387.com/events">Događaji</a>
```

Ako link vodi na neku stranicu koja je dio našeg web sjedišta, tada će vrijednost atributa href biti relativni URL. Relativni URL nema ime domene, nego samo ime stranice na koju vodi i po potrebi putanju do te stranice. Npr. ako su sve stranice nekog web sjedišta u istoj mapi, onda će relativni URL biti samo ime dokumenta na koji se želi „linkati“.

```
<a href="about_bh.html">Kontakt</a>
```

5.2. Putanje i struktura web sjedišta

Radi preglednosti i lakše organizacije datoteke, **web sjedišta su najvećem broju slučajeva organizovane u hijerarhijsku strukturu**. Ta struktura se postiže na način da se različiti segmenti web-a spremaju u vlastite mape. Ova struktura je razgranata kao stablo i proteže se na nekoliko nivoa, te je zbog toga pisanje relativnih URL-ova kompleksnije. Početna stranica web sjedišta najčešće se naziva index.html. Ako je imenovana na taj način, početna stranica će biti dostupna putem URL-a: www.domena.ba (nije potrebno pisati putanju www.domena.ba/index.html). Ako se unutar mape nalazi datoteka index.html, to će biti početna datoteka za tu mapu, bit će dostupna putem URL-a: www.domena.ba/o-nama (nije potrebno pisati www.domena.ba/o-nama/index.html). Korištenjem relativnih URL-ova prilikom izrade web-a, omogućava se „linkanje“ sa stranice na stranicu koje je **funkcionalno i na lokalnom računaru**, bez imena domene. Na ovaj način se može napraviti navigacija za cijeli web. Ta navigacija će raditi dok su svi HTML dokumenti na lokalnom računalu, ali i kad se prebace na server, pod uslovom da se ne mijenja struktura mapa i lokacija dokumenata u mapama. Ako su sve stranice od kojih se web sastoji u jednoj mapi, tada je dovoljno samo navesti ime stranice u href atributu a elementa.

[18] **URL** (engl. *Uniform Resource Locator*) je putanja do nekog sadržaja na web-u, još naziva i web adresa. Taj sadržaj može biti tekst, slika, HTML dokument, neka datoteka i sl. Svaka stranica, dokument i slika na web-u imaju svoj jedinstveni URL.

Međutim, ako su stranice organizirane po mapama[19], potrebno je odgovarajućom putanjom naznačiti pregledniku kako će doći od stranice na kojoj jest, do stranice na koju treba ići.

Putanja	Primjer
Putanja do dokumenta u istoj mapi	Link sa stranice Osnove CSS-a: (kursevi/web/css.html) na stranicu Uvod u HTML (kursevi/web/html.html) `<a href="html.html">Uvod u HTML</a>`
Putanja do dokumenta u mapi ispod	Link sa stranice Kursevi (kursevi/index.html) na stranicu Uvod u HTML (kursevi/web/uvod-u-html.html) `<a href="web/html.html">Uvod u HTML</a>`
Putanja do dokumenta dvije mape ispod	Link s početne stranice (index.html) na stranicu Uvod u HTML (kursevi/web/html.html) `<a href="kursevi/web/html.html">Uvod u HTML</a>`
Putanja do dokumenta u mapu iznad	Link sa stranice Uvod u HTML (kursevi/web/html.html) na stranicu Kursevi (kursevi/index.html) `<a href="../index.html">Kursevi</a>`
Putanja do dokumenta dvije mape iznad	Link sa stranice Uvod u HTML (kursevi/web/uvod-u-html.html) na početnu stranicu (index.html) `<a href="../../index.html">Kursevi</a>`
Putanja do dokumenta na drugom dijelu stabla	Link sa stranice O nama (o-nama/index.html) na stranicu Uvod u HTML (kursevi/web/html.html) `<a href="../kursevi/web/html.html">Uvod u HTML</a>`

5.3. Otvaranje linka u novom prozoru

Svi linkovi se zadano otvaraju u istom prozoru u kojem se nalaze. Ako želimo omogućiti da se link otvara u novom prozoru potrebno je koristiti target atribut HTML elementa a, i postaviti mu vrijednost na _blank. Na primjer:

```
<a href="http://www.Academy387.com" target="_blank">Academy387 Sarajevo</a>
```

Najčešći razlog za ovakvo otvaranje linka je u slučaju da link vodi sa naših web stranica na neke druge stranice. U tom slučaju će posjetitelj nakon što pregleda novi sadržaj i zatvori prozor, „izgubiti“ prvobitnu stranicu. Ako se link otvori u drugom prozoru, onda će posjetitelj nakon zatvaranja prozora biti vraćen na početnu web stranicu.

[19] U slučaju da se web stranice izrađuju pomoću CMS sistema, postoji mogućnost da svaka stranica nije pojedinačni HTML dokument. Umjesto toga, ovi sistemi često koriste po jedan predložak za jednu vrstu stranice (vijesti, proizvodi, blog...). U slučaju da radimo izmjene koje se ne tiču samog sadržaja na nekome od predložaka, postoji mogućnost da će se izmijeniti i sam predložak, pa samim tim i sve druge stranice koje koriste taj predložak.

5.4. Specijalni linkovi (mailto i tel)

Posebna vrsta linkova su oni koji pokreću vanjske aplikacije za slanje e-pošte ili telefoniranje. Riječ je o mailto i tel linkovima. Da se izradi link koji pokreće korisnički program za slanje e-pošte i popunjava polje sa adresom, koristi se HTML-element a. Ovaj put atribut href započinje sa mailto:, a odmah zatim slijedi adresa e-pošte na koju se želi poslati e-pošta. Na primjer:

```
Za više pitanja obratite se na:
<a href="mailto: contact@academy387.com ">Info</a> e-mail.
```

U novije vrijeme je postalo uobičajno pregledavati Internet na mobitelu, kako bi se korisnicima olakšalo telefoniranje, koristi se protokol tel - omogućava automatsko pokretanje telefoniranja sa već unesenim telefonskim brojem. Na isti se način može koristiti i protokol sms. Na primjer:

```
Ako imate pitanja nazovite nas na <a href="tel:+38733202976">033 202-976</a>.
```

5.5. Link na određeni dio stranice – sidro (engl. *anchor*)

Na vrhu duge stranice sa puno sadržaja se dodaje pregled sadržaja na stranici. U tom slučaju će svaki element u sadržaju biti link koji vodi na tačno određeni dio stranice, na koji se odnosi. Također, kod dugih stranica, nakon pojedinog poglavlja se može dodati i link koji korisnika vraća na vrh stranice, tj. na pregled sadržaja. Prije nego što je moguće napraviti link na određeni dio stranice, prvo se treba identificirati mjesto (ili sidro) na koje će link voditi. Ovo se radi na način da se elementu na koji želimo da link vodi, doda atribut id. Id je atribut koji se može dodati svakom HTML elementu, proizvoljan je, ali bi **trebao započinjati slovom ili podlinijom** (_). Nijedan id atribut ne smije imati istu vrijednost. Jednom kada je određena tačka na koju će link voditi, može se izraditi i sam link. Da bi link išao na element koji koristi id atribut koristi se a element, ali ovaj put se u atribut href doda simbol rešetka (#) i vrijednost id atributa na koji se želi stvoriti veza.

```
<p id="vrh">Kursevi:</p>
<ul>
    <li><a href="#sql">Uvod u SQL</a></li>
    <li><a href="#javascript">Uvod u JAvaScript</a></li>
    <li><a href="#PHP">Uvod u PHP i MySQL</a></li>
</ul>
<h2 id="sql">Uvod u SQL</h2>
<p>Objekti, relacije i baze podataka; kompleksne selekcije; operacije nad
relacijama; elementi teorije normalizacije; oblikovanje relacijske baze
podataka.</p>
<h2 id="javascript">Uvod u JavaScript</h2>
<p>Varijable, funkcije, operatori, metode, objekti.</p>
<h2 id="PHP">Uvod u PHP i MySQL</h2>
```

```
<p>Model klijent-poslužitelj, varijable, operatori, uvjetne strukture, polja,
petlje, funkcije, ugrađene funkcije PHP-a, obrasci, prijenos podataka između
skripti, rad sa datotekama, slanje e-pošte, rad sa bazom,...</p>
<p>Povratak na <a href="#vrh">vrh</a>.</p>
```

Ako se želi napraviti link prema tačno određenom mjestu na nekoj drugoj stranici, i to je moguće korištenjem ove metode. Važno je da mjesto na koje se želi povezati ima definisan id atribut. U primjeru će biti prikazano kako se „linka“ sa neke druge stranice na opis kursa Uvod u PHP i MySQL.

```
<p>Detalje o kursu Uvod u PHP i MySQL pronaći na stranici sa listom<a
href="www.domena.ba/kursevi#PHP">kurseva</a>.</p>
```

U slučaju da se želi napraviti link na tačno određeno mjesto neke stranice na nekom drugom web sjedištu, i to je moguće ako je dostupan id atribut te lokacije. U ova dva slučaja se može koristiti relativni URL za link na drugu stranicu našega web sjedišta ili apsolutni URL za link na tuđe web sjedište.

U poglavlju je obrađeno:

- šta su linkovi i čemu služe;
- vrsta linkova u HTML dokumentu (linkovi koji se otvaraju u novom prozoru, likovi koji vode sa jedne na drugu stranicu, likovi koji vode na neko drugo mjesto na istoj stranici,...);
- kako linkovi utječu na funkcionalnost web-sjedišta.

VI Slike

Po završetku ovog poglavlja, polaznik će moći:

- odabrati i objaviti sliku na web-stranici;
- odabrati format slike koji je najbolji za objavu na web-stranici;
- postaviti opis i druga svojstva slike.

Slike se ubacuju u HTML dokument korištenjem tag-a img. Ovaj tag ima mnogo atributa, a mi ćemo spomenuti neke najbitnije. Slijedeći kod možete ubaciti ispred ili poslije p tag-a ili bilo kojeg drugog tag-a, u zavisnosti gdje želite da prikažete sliku. Obavezni atribut elementa img je src, koji govori pregledniku putanju do slike[20]. Obično se koristi relativni URL, s obzirom na to da se slika nalazi unutar mape na serveru. Slike na web stranicama mogu ispunjavati različite svrhe, pa je prilikom odabira slika za web stranicu potrebno obratiti pažnju na to da:

- budu relevantne,
- prenose određenu informaciju,
- dočaravaju odgovarajuću atmosferu,
- budu jasne,
- budu u skladu sa ostalim bojama na web stranici.

Određene vrste slika mogu vrlo brzo ostaviti određeni dojam o web stranici i prije nego što se stigne pročitati sadržaj stranice. U slučaju nemogućnosti izrade autorskih fotografija i grafika, iste je moguće kupiti preko različitih stranica koje prodaju slike (npr. shutterstock.com); postoje i sajtovi sa slikama koje možete besplatno koristit bez ikakve obaveze prema autoru, jedan od najpoznatijih sajtova je unsplash.com.

6.1. Slika – <img> element

Sve slike koje se koriste na web-stranici trebaju prethodno biti spremljene na nekoj lokaciji unutar web-a. Uobičajena praksa je da se za spremanje slika otvori jedna mapa, najčešće imena "slike", te u njoj odgovarajuće podmape radi lakšeg snalaženja. Za umetanje slika bilo kojeg formata u HTML kodu se koristi element img. Element nema završnu oznaku. Obavezni atribut elementa img je src koji govori pregledniku putanju do slike. Obično se koristi relativni URL, sa obzirom na to da se slika nalazi unutar mape na serveru. Na primjer:

```
<img src="slika-1.png" alt="kabina u šumi"/>
```

[20] Prilikom pripreme slika za objavu na web stranicama, potrebno je misliti na „težinu“ slika, tj. optimizirati fizičku veličinu slike tako da ona bude što manja, bez da se naruši njezina kvaliteta.

Slika

Atribut alt daje opis slike u slučaju da se slika iz bilo kojeg razloga ne može učitati u pregledniku. Važno je da opis bude tačan i jasan, jer je riječ o tekstu koji će se, kada slijepe i slabovidne osobe pristupaju web stranici, pročitati pomoću čitača. U slučaju da slika nema značenje, već služi samo kao dizajnerski element (npr. slika određene boje koja se koristi kao razdjelnik sadržaja), svejedno bi trebala imati alt atribut, bez opisa. Na primjer:

```
<img src="slika.png" alt=""/>
```

Atribut title omogućuje prikazivanje dodatnih informacija o slici kada se mišem pređe preko slike. Na primjer:

```
<img src="slika-1.png" alt="kabina u šumi" title="kabina u šumi"/>
```

Atributi koji služe za određivanje veličine slike u pikselima su: height i width. Ovi atributi se mogu koristiti, ali preporučuje se prethodna optimizacija slika, tj. spremanje slika tačne veličine, te određivanje veličine slika pomoću CSS-a. Nepažljivim korištenjem oba atributa istovremeno, može doći do izobličenja slike. Na primjer:

```
<img src="slika-1.png" alt="kabina u šumi" title="kabina u šumi" width="300"
height="350"/>
```

Slika

Stavljanje img elementa u kodu utječe na to gdje će slika biti smještena na stranici, tj. u odnosu na tekst. Slika može biti smještena:

- prije odlomka,
- na samome početku odlomka,
- u sredini odlomka.

Ako se slika smješta prije odlomka, koji je blok element, slika će ostati u vlastitom redu, a odlomak će započeti u sljedećem redu, jer blok element zauzima cijelu širinu stranice. Npr.:

```
<img src="slika-1.png" alt="kabina u šumi" title="kabina u šumi" width="300" height="350"/>
<p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Quidem recusandae eligendi delectus esse accusamus fugit quasi facilis cumque numquam, qui, odit dolores eius aliquam cupiditate error aspernatur harum animi quibusdam?</p>
```

Slika

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Doloremque
corrupti eligendi aperiam pariatur obcaecati nemo mollitia dolores ve
deserunt exercitationem. Possimus dolor earum molestias, dolorem e

Ako je slika smještena unutar odlomka, bilo na samom početku ili u sredini, tekst u odlomku će ići oko slike. Na primjer:

```
<p>Lorem ipsum dolor sit, amet <img src="slika-1.png" alt="kabina u šumi" title="kabina u šumi" width="300" height="350"/> consectetur adipisicing elit. Quidem recusandae eligendi delectus esse accusamus fugit quasi facilis cumque numquam, qui, odit dolores eius aliquam cupiditate error aspernatur harum animi quibusdam?</p>
```

Slika[21] je inline element što znači da će zauzeti onoliko mjesta na stranici koliko joj je potrebno, a zatim će se nastaviti prikazivati idući element. Smještanje slika unutar odlomka po horizontali ili vertikali izvodi se korištenjem CSS-a, ali se na starim web stranicama može pronaći atribut koji se piše u početnom tag-u, u tabeli u nastavku je predstavljen pregled smještanja slike uz pomoć atributa align.

[21] Odmicanje slova od slike može se napraviti korištenjem CSS-svojstava padding i margin. Ako se želi cijeli tekst omotati oko slike onda se koristi CSS-svojstvo float.

Vrijednost atributa	HTML kod	Primjer
left	`<img src="hmm.jpg" width="400" height="350" alt="kabina u šumi" title="kabina u šumi" align="left"/>`	
right	`<img src="hmm.jpg" width="400" height="350" alt="kabina u šumi" title="kabina u šumi" align="right"/>`	
top	`<img src="hmm.jpg" width="400" height="350" alt="kabina u šumi" title="kabina u šumi" align="top"/>`	
middle	`<img src="hmm.jpg" width="400" height="350" alt="kabina u šumi" title="kabina u šumi" align="middle"/>`	

bottom	`<img src="hmm.jpg" width="400" height="350" alt="kabina u šumi" title="kabina u šumi" align="bottom"/>`	

6.2. Formati slike za web

Slike namijenjene za objavu na web stranicama bi trebale bi biti snimljene u veličini u kojoj će se pojavljivati na samoj web stranici, prije nego ih pohranimo na poslužiteljskom računaru. Ako se slika sprema u manjoj veličini od one koja će biti prikazana, postoji mogućnost da će prikazom u planiranoj veličini na web-u izgubiti na svojoj kvaliteti ili će biti izobličena. A ako je slika veća od veličine predviđene za web, sporije će se učitavati i korisniku, te će mu povećati količinu potrošenog internetskog prometa. U slučaju da na postojećem web-u treba promijeniti sliku, potrebno je prvo provjeriti kolika je bila prethodna slika; može se napraviti klikom desne tipke miša na sliku i otvaranjem u novom prozoru, te provjerom informacija o slici ili na način da se provjeri veličina spremljene slike na serveru. Osim ispravne veličine, važno je da slika bude spremljena i u ispravnom grafičkom formatu. Web stranice uglavnom koriste formate:

- JPEG,
- GIF,
- SVG,
- PNG.

Odabir formata je važan jer odabirom pogrešnog formata može doći do gubitka oštrine ili nepotrebnog opterećenja diska, korisničkog prometa, te sporog učitavanja slike.

JPEG/JPG je format koji se koristi uglavnom za prikaz fotografija, tj. slika koje u sebi imaju veliku količinu boja. JPEG se koristi i za crno-bijele, tj. dvobojne slike koje imaju puno prijelaza. Ovaj format može vjerno prikazati sliku u milionima boja, zbog toga što koristi mogućnost sažimanja sa gubicima. U slučaju da se slika previše sažme

(optimizuje), izgubit će na kvaliteti i postoji mogućnost da ne izgleda dobro. Sažimanje slike i smanjivanje njezinih dimenzija u pikselima nisu povezani, iako će se smanjivanjem slika uvijek zauzimati i manje prostora.

GIF je format koji se koristi za prikaz slika koje imaju veće plohe iste boje kao što su ilustracije, logotipi i slično. Te plohe su jasno razdvojene od ploha drugih boja. GIF prikazuje do 256 boja i ne zahtijeva puno prostora na disku. Animirani GIF je format koji prikazuje nekoliko sličica za redom, te na taj način stvara animaciju.

SVG je noviji grafički format koji omogućava objavljivanje slika u vektorskom formatu. Vektorski format omogućava povećavanje grafika bez ikakvog gubitka kvalitete. Ovo je noviji format, pa se može dogoditi da ga pojedini preglednik ne podržava.

PNG je format koji podržava transparentnost, bilo na dijelu slike (potpuna prozirnost nekih dijelova) ili na cijeloj slici. Zahtijeva vrlo malo prostora na disku. Često se koristi za dugmad i jednostavne slike, ali sa obzirom da omogućava i gradaciju prozirnosti, vrlo je popularan format za prikaz sjena nekih elemenata na web-u.

6.3. Opis slike – <figure> i <figcaption> elementi

Slike na web stranicama mogu imati i opis. Kako bi opis i slika uvijek bili zajedno, uveden je element figure. Unutar elementa je moguće imati više od jedne slike, pod uslovom da sve slike imaju isti opis. Sam opis slike piše se unutar elementa figcaption. Preglednici ponekad sadržaj unutar elementa figure prikazuju uvučeno. Na primjer:

```
<figure>
    <img src="hmm.jpg" width="400" height="350" alt="kabina u šumi"><br/>
    <figcaption> Lorem ipsum, dolor sit amet consectetur adipisicling elit.
    Deleniti, rem!
    </figcaption>
</figure>
```

Slika

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Deleniti, rem!

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae, dolorum doloribu
accusantium quos, eveniet modi illum magnam illo nemo soluta repellendus delectu
cumque dolore numquam, quidem corporis, voluptatem aliquam qui! Quos, nemo ve
Voluptatum nulla quidem veniam accusamus quaerat nesciunt ipsum incidunt quis s
molestiae numquam officiis iure est libero tempore, eaque earum! Voluptas quasi no

U poglavlju je obrađeno:

- odabir i objava slike na web-stranici;
- odabir formata slike koji je najbolji za objavu na web-stranici;
- postavljanje opis i drugih svojstva slike.

VII Multimedijalne datoteke

Po završetku ovog poglavlja, polaznik će moći:

- dodati audio datoteku na web-stranicu;
- dodati video datoteku na web-stranicu.

Video i **audio datoteke** se mogu na web stranice uključiti: **direktno** ili **upotrebom HTML elemenata**, ili na način da se prethodno postavi na neki servis koji je namijenjen dijeljenju videa (npr. YouTube) ili zvuka (npr. Spotify), te da se od tog servisa preuzme kod koji će se uključiti direktno u HTML kod web-stranice.

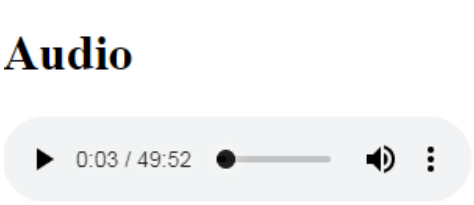
7.1. Dodavanje audio datoteke - <audio> i <source> elementi

Dodavanje audio datoteke putem nekog specijalizovanog servisa namijenjenog dijeljenju audio datoteka je jednostavno, jer će servis, nakon postavljanja audio datoteke korisniku omogućiti kopiranje HTML koda koji se jednostavno umetne u HTML datoteku. HTML5 je uveo novi element audio; omogućava dodavanje audio datoteka na web stranice. U slučaju da preglednik ne podržava postavljeni format, prikazat će što god se nalazi unutar audio elementa. Element audio ima slijedeće atribute:

- src – označava put do audio datoteke;
- controls – postavlja kontrole na audio player, ako se atribut ne navede, kontrole neće biti prikazane;
- autoplay – u slučaju da se ovaj atribut navede, audio datoteka će se automatski pokrenuti, smatra se dobrom praksom da se audio datoteka pokreće tek na zahtjev korisnika;
- preload – u slučaju da nije postavljen atribut autoplay ovaj atribut će odrediti hoće li se audio datoteka automatski preuzeti na korisnički računar ili ne, atribut može imati vrijednost: none – preglednik neće automatski preuzimati datoteku i auto – preglednik će preuzeti datoteku na korisnički računar čim se učita web stranica;
- metadata – preglednik će preuzeti osnovne informacije o datoteci (veličina, ime, trajanje...);
- loop – određuje hoće li se ili neće po završetku audio datoteke ona ponovno pokrenuti.

Na primjer:

```
<audio src="FairyTail.mp3" controls></audio>
```



Budući da neki preglednici ne podržavaju određene vrste audio datoteka, moguće je pripremiti više formata[22] iste audio datoteke[23] i objaviti ih na web stranici, da bi ona vrsta koju preglednik podržava bila dostupna korisniku. Ovo je moguće napraviti korištenjem HTML elementa source. Na primjer:

```
<audio controls>
    <source src="FairyTail.mp3" type="audio/mpeg">
    <source src="FairyTail.ogg" type="audio/ogg">
    Ovaj preglednik ne podržava ovu vrstu audio datoteke.
</audio>
```

Podršku audio formata od strane web preglednika i tipove medija za audio formate možemo vidjeti u tabeli u nastavku:

Preglednik	.mp3	.wav	.ogg	Format datoteke	Tip medija
Internet Explorer	DA	NE	NE	.mp3	audio/mpeg
Chrome	DA	DA	DA	.ogg	audio/ogg
Firefox	DA	DA	DA	.wav	audio/wav
Safari	DA	DA	NE		
Opera	DA	DA	DA		

7.1. Dodavanje audio datoteke – <video> i <source> elementi

Kao i sa audio datotekama, dodavanje video datoteke putem nekog servisa koji služi za dijeljenje video datoteka je prilično jednostavno, jer će takav servis, nakon postavljanja audio datoteke, korisniku omogućiti kopiranje HTML koda koji se jednostavno umetne u HTML datoteku. HTML element video nam omogućava dodavanje video datoteka na web stranice. U slučaju da preglednik ne podržava postavljeni video format, prikazat će šta god se nalazi unutar video elementa. HTML element video omogućava dodavanje video datoteka na web stranice. U slučaju da preglednik ne podržava postavljeni video format, prikazat će što god se nalazi

[22] Većina novih preglednika podržava format MP3, pa zavisno od ciljane publike, nije nužno pripremati datoteku u više formata.
[23] Za razliku od video elementa, audio element nije stekao veliku popularnost. Jedan od razloga mogao bi biti i to što su prvi preglednici koji su implementirali ovaj element imali problema sa kvalitetom zvuka prilikom reprodukcije audio datoteka.

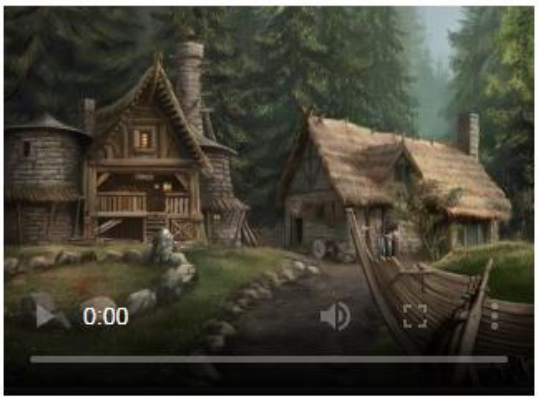
unutar video elementa. Element video ima iste atribute kao i audio, te neke dodatne. Dodatni atributi su:

- width i height – omogućava kontrolu veličine videa u pikselima, moguće je dodijeliti oba atributa, ili samo jedan, ali je kao i kod slika je potrebno paziti da se video ne deformiše korištenjem oba atributa istovremeno;
- poster – omogućava da se tokom učitavanja videa, na mjestu videa pojavi zadana slika. Na primjer: `poster="slike/forest.png"`

Primjer postavljanja video materijala je prikazan u slijedećem primjeru:

```
<video width="320" height="240" poster="hmm.jpg" controls>
    <source src="movie.mp4" type="video/mp4">
    <source src="movie.ogg" type="video/ogg">
    Your browser does not support the video tag.
</video>
```

Audio



Budući da neki preglednici ne podržavaju određene vrste video datoteka, moguće je pripremiti više formata iste video datoteke i objaviti ih na web stranici, kako bi ona vrsta koju preglednik podržava bila dostupna korisniku. Ovo je moguće napraviti korištenjem HTML elementa source.

Atributi ovog elementa su:

- src – putanja do datoteke;
- type – atribut kojim se identificira format video datoteke, ovaj atribut se koristi kako preglednik ne bi započinjao preuzimanje datoteke dok ne sazna o kojem se formatu radi;
- codecs – atribut kojim se označava vrsta softvera za dekodiranje videa koja se koristila pri izradi videa.

Na primjer:

```
<h1>Video</h1>
<video width="600"preload controls autoplay>
<source src="video.mp4" type='video/mp4;codecs="avc1.42E01E, mp4a.40.2"'/>
<source src="video.webm" type='video/webm;codecs="vp8, vorbis"'/>
Your browser does not support the video tag.
</video>
```

Podršku video formata od strane web preglednika i tipove medija za video formate možemo vidjeti u tabeli u nastavku:

Preglednik	.mp3	.wav	.ogg	Format datoteke	Tip medija
Internet Explorer	DA	NE	NE	.mp4	video/mp4
Chrome	DA	DA	DA	.webm	video/webm
Firefox	DA	DA	DA	.ogg	video/ogg
Safari	DA	DA	NE		
Opera	DA	DA	DA		

U poglavlju je obrađeno:

- dodavanje audio datoteke na web-stranicu;
- dodati video datoteku na web-stranicu.

VIII Tabele

Po završetku ovog poglavlja, polaznik će moći:

- praviti jednostavne tabele;
- praviti tabele sa složenijom strukturom, spojenim redovima ili kolonama;
- dodati naslov tabeli.

Tabele omogućavaju prikazivanje informacija na dvije ose (x i y). Koriste se za prikaz finansijskih izračuna, rasporeda, različitih rezultata, statističkih podataka,...

8.1. Semantička struktura tablele – <table>, <tr> i <td> elementi

Za izradu tabele se koristi element table. Sadržaj u tabeli će se ispisivati red po red. Svaki red je definisan elementom tr (engl. *table row*) – označava red tabele. Svaka ćelija u redu se definiše elementom td (engl. *table data*) – označava sadržaj ćelije. Ponekad se automatski prikazuju i linije oko tabele, a nekada ne; u zavisnosti od preglednika. Na primjer:

```
<table>
    <tr>
        <td>Uvod u HTML</td>
        <td>12</td>
    </tr>
    <tr>
        <td>Uvod u PHP i MySQL</td>
        <td>25</td>
    </tr>
    <tr>
        <td>Uvod u SQL</td>
        <td>20</td>
    </tr>
</table>
```

Tabele

Uvod u HTML	12
Uvod u PHP i MySQL	25
Uvod u SQL	20

8.2. Elementi tabele – <th>, <thead>, <tbody> i <tfoot>

Elementi th, thead, tbody i tfoot se koriste kako bi se olakšao prikaz sadržaja korisnicima koji upotrebljavaju čitače za pregledavanje web stranica, kako bi se poboljšalo indeksiranje stranica od strane tražilica, a također olakšavaju kontrolu nad izgledom tabele kada se koristi CSS. Korištenjem elementa th, **tabelama[24] je moguće dodati i naslovni red**. Element th označava naslov reda ili kolone (engl. *table heading*). Preglednici obično prikazuju ovaj element podebljano i centrirano u ćeliji pa je već i na prvi pogled vidljiva njegova svrha[25].

```
<table>
    <thead>
        <tr>
            <th>Naziv kursa</th>
            <th>Trajanje</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Uvod u HTML</td>
            <td>12</td>
        </tr>
        <tr>
            <td>Uvod u PHP i MySQL</td>
            <td>25</td>
        </tr>
        <tr>
            <td>Uvod u SQL</td>
            <td>20</td>
        </tr></tbody></table>
```

Naziv kursa	Trajanje
Uvod u HTML	12
Uvod u PHP i MySQL	25
Uvod u SQL	20

Element thead označava **zaglavlje** tabele (engl. *table head*), element tbody sadržaj tabele (engl. *table body*), element tfoot **podnožje** stranice (engl. *table footer*). Ovi elementi se neće prikazivati na drugačiji način od ostalih elemenata tabele, ali se koriste prilikom definisanja dizajna tablele.

[24] Zaglavlje i podnožje tabele su različiti elementi kako bi se vrlo duge tabele mogle printati ili prikazati na način da je zaglavlje i podnožje uvijek vidljivo. Ovo još nije sistemski implementirano u preglednicima.
[25] Ako je u tabeli potrebno navesti praznu ćeliju, to se radi na način da se svejedno navede element td ili th. U slučaju da se ne navede prazan element, tabela će se pogrešno prikazati.

8.4. Spajanje redova i kolona u tabelama

Ponekad je **potrebno spojiti ćelije u redovima ili u kolonama** kako bi se proširio prostor za sadržaj, ili da bi se uredila hijerarhijska struktura tabele. To se može postići korištenjem atributa colspan za spajanje više ćelija u redu ili rowspan za spajanje više ćelija u koloni. Ovi atributi se mogu dodati elementima th i td. Colspan i rowspan su atributi kojima se dodjeljuje broj ćelija koje se želi spojiti u jednu. Važno je voditi računa o tome da će neke ćelije na ovaj način „nestati“ iz koda. Npr. ako red ima tri ćelije, u kodu se nalaze tri elementa td, ako se dvije ćelije spoje u jednu, u redu će ostati dvije ćelije (jedna duža i jedna kraća), pa će shodno tome u kodu biti i dva td elementa, a ne tri. Na primjer:

```
<head>
    <style>
        td,th{
            border: 1px solid black;
            }
    </style>
</head>
<body>
    <h1>Tabele</h1>
    <table>
        <thead>
            <th></th>
            <th colspan="2">Ime tabele kroz dvije kolone</th>
        </thead>
        <tbody>
            <tr>
                <td rowspan="2">Kolona 1</td>
                <td>Red 1</td>
                <td rowspan="2">Kolona 2</td>
            </tr>
            <tr>
                <td>Red 2</td>
            </tr>
            <tr>
                <td colspan="3">Red po cijeloj dužini tabele</td>
            </tr>
            <tr>
                <td>Ćelija u koloni 1</td>
                <td>Ćelija u koloni 2</td>
                <td>Ćelija u koloni 2</td>
            </tr>
        </tbody>
    </table>
</body>
```

Tabele

	Ime tabele kroz dvije kolone	
Kolona 1	Red 1	Kolona 2
	Red 2	
Red po cijeloj dužini tabele		
Ćelija u koloni 1	Ćelija u koloni 2	Ćelija u koloni 2

8.4. Naslov tabele - <caption> element

Tabele mogu imati i **nazive** (engl. *caption*), kao i slike. Za dodavanje naslova se koristi element caption, isti će se smjestiti iznad tabele, a u kodu se dodaje iznad prvog tr elementa. Do sada se u primjerima kao naslov tabele koristio h1 ili h2, standardni element koji se koristi za naslove svog sadržaja u HTML-u, a sada će biti prikazan primjer sa elementom caption:

```
<table>
    <caption>Trajanje kursa</caption>
    <tr>
        <th>Naziv kursa</th>
        <th>Trajanje kursa</th>
    </tr>
    <tr>
        <td>Uvod u HTML</td>
        <td>20</td>
    </tr>
    <tr>
        <td>Uvod u PHP i MySQL</td>
        <td>35</td>
    </tr>
    <tr>
        <td>Uvod u C#</td>
        <td>50</td>
    </tr>
</table>
```

Trajanje kursa

Naziv kursa	Trajanje kursa
Uvod u HTML	20
Uvod u PHP i MySQL	35
Uvod u C#	50

U poglavlju je obrađeno:

- pravljenje jednostavnih tabela;
- pravljenje tabela sa složenijom strukturom, spojenim redovima ili kolonama;
- dodavanje naslova tabeli.

IX Uvod i sintaksa CSS – a

Po završetku ovog poglavlja, polaznik će moći:

- objasniti šta je CSS;
- razumijeti CSS sintaksu;
- koristiti CSS selektore;
- razumijeti prioritete CSS pravila;
- pseudoklase;
- nasljeđivanje vrijednosti;
- koristiti CSS komentare;
- koristiti CSS uslovne komentare;
- dodati CSS kod direktno u HTML kod;
- uključiti vanjsku CSS datoteku u HTML dokumentu;

CSS je jezik koji služi za oblikovanje web stranica. Uz HTML, CSS je osnovna tehnologija na kojoj se temelji današnji web, a služi za oblikovanje sadržaja. Kroz CSS kod se definiše izgled web stranice i svih elemenata na njoj.

9.1. Pojam CSS-a

CSS je skraćenica za **kaskadnu listu stilova** (engl. *Cascading Style Sheets*). Pojam *style sheet* se često upotrebljava za datoteku koja sadrži CSS kod. Dakle, *style sheet* je datoteka koja definiše stil, odnosno izgled web stranice. Riječ *cascading* označava kaskadnu primjenu CSS pravila. CSS pravilo se može napisati tako da bude primijenjeno na sve elemente ili samo na neke elemente ili tako da vrijedi samo za tačno određeni element. Prije pojave CSS-a, oblikovanje izgleda web stranice. do određenog nivoa je bilo moguće postići i u HTML-u. Ali, time se stvorio problem miješanja sadržaja i strukture sa kodom čija je jedina svrha bila prezentacija. HTML kod za definisanje izgleda se morao ponavljati iznova na svakom elementu i na svakoj stranici u web sjedištu. CSS-om se nastoji riješiti taj problem. Glavna ideja CSS-a je odvajanje prezentacijskog koda u posebne datoteke, te njegovo definisanje uz pomoć jednostavnih pravila koja se mogu odnositi na više elemenata odjednom. Prva verzija CSS-a je definisana krajem 1996. godine. Do usvajanja CSS-a od strane autora web sadržaja i proizvođača web preglednika je prošlo mnogo vremena. Vrlo dugo web preglednici nisu dosljedno implementirali CSS specifikaciju, pa se autori nisu mogli pouzdati da će stranice izgledati približno isto u svim preglednicima. Razvijeni su brojni trikovi u CSS-u čija je namjena bila da isprave neočekivana ponašanja u nekim preglednicima. Današnja situacija je po tom pitanju puno bolja, iako se i danas savjetuje provjera

izgleda stranice u što više preglednika. Trenutna verzija CSS-a je CSS3, koji je donio brojne novitete i poboljšanja. Verzija još nije finalizirana, već se i dalje neprestano razvija i nadograđuje. Razvojem CSS-a se bavi **World Wide Web Consortium** (skr. W3C), tijelo zaduženo za razvoj web standarda u saradnji sa proizvođačima preglednika i zainteresiranim web autorima.

9.2. Uključivanje CSS-a u HTML kod

CSS kod se **tipično piše odvojeno od HTML koda**, tj. u posebnoj datoteci. Da bismo HTML dokument povezali sa CSS datotekom, koristimo HTML-element link:

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

Kada se taj element koristi za uključivanje CSS-a (što je zapravo njegova glavna namjena), atribut rel mora imati vrijednost stylesheet, a atribut type vrijednost text/css. Atribut href postavljamo na putanju do CSS-datoteke koju želimo uključiti. U ovom primjeru uključujemo datoteku moj-stil.css, koja se nalazi u mapi stilovi. U ovom slučaju se radi o relativnoj putanji od HTML datoteke do CSS datoteke. Evo nekoliko najčešćih primjera upotrebe relativne putanje:

Putanja	Opis
`moj-stil.css`	HTML datoteka i datoteka moj-stil.css se nalaze u istoj mapi.
`stilovi/moj-stil.css`	Datoteka moj-stil.css se nalazi u mapi stilovi, HTML datoteka i mapa stilovi nalaze se u istoj mapi.
`../stilovi/moj-stil.css`	Datoteka moj-stil.css nalazi se u mapi stilovi. Mapa u kojoj se nalazi HTML datoteka i mapa stilovi nalaze se u istoj mapi.

Element link se mora uvijek nalaziti unutar HTML elementa head. Ako se HTML stranica koristi većim brojem CSS datoteka, uključit će se tako da se element link navede više puta. Mogući su i načini pisanja CSS koda u samoj HTML datoteci. Za to služi HTML element style, u kojem se direktno mogu pisati CSS pravila. Na primjer:

```
<style type=”text/css”>
     p{color: green;}
</style>
```

Element style se uvijek mora nalaziti unutar elementa head. Atribut type će se u tom slučaju postaviti na vrijednost type/css, ali se može i izostaviti. Drugi način na koji se CSS kod može direktno „ubaciti“ u HTML je preko atributa style:

```
<p style="color: red;">
Ne izuvaj se prije vode.</p>
```

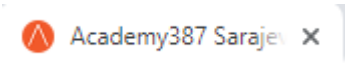
Atribut style se može staviti na skoro svaki HTML element. U ovom slučaju se pišu samo CSS deklaracije, bez selektora, te deklaracije će se primijeniti samo na taj HTML element. Iako se

ponekad može činiti praktičnim, **miješanje CSS-a i HTML-a u istoj datoteci se ne preporučuje**. Najbolji pristup je CSS kod pisati u posebnoj datoteci. Tako se ostvaruje ponovna iskoristivost koda i sažetost – određeno CSS pravilo je dovoljno napisati samo jednom, a ono će vrijediti u svim HTML dokumentima. Također, kad je potrebno nešto promijeniti u nekom CSS pravilu, dovoljno je to napraviti na jednom mjestu, a ne u svakoj HTML datoteci posebno.

9.3. Uključivanje ikone stranice

Ikona stranice (engl. *favicon*) je mala sličica koja se prikazuje u zaglavlju preglednika kako bi se stranica vizualno razlikovala od drugih stranica otvorenih u pregledniku. Ikona stranice se, kao i CSS-datoteke, uključuje pomoću elementa link. Na primjer:

```
<link rel="icon "href="favicon.ico">
```



Atribut rel mora imati vrijednost icon, a pomoću atributa href postavlja se putanja do datoteke. Za ikonu stranice se najčešće koristi .ico datoteka, iako su podržani i drugi formati (na primjer .png i .gif).

9.4. Sintaksa CSS pravila

CSS kod se sastoji od CSS pravila. Primjer jednostavnog CSS pravila:

```
p{
    color: red;
}
```

Ovakvo pravilo postavlja svim p elementima boju teksta na crvenu. Dio pravila koje određuje (odnosno selektuje) elemente na koje se pravilo odnosi zove se **selektor**. Svako pravilo moramo započeti selektorom, a najjednostavniji selektor je upravo naziv elementa:

```
p{    }
```

Nakon selektora dolaze vitičaste zagrade. CSS nije osjetljiv na prazan prostor pa vitičaste zagrade nije obavezno pisati u posebnim redovima, ali se to preporučuje radi čitljivosti. Unutar vitičastih zagrada se prvo navodi svojstvo koje se postavlja. U primjeru se radi o boji teksta (CSS svojstvo color):

```
p{color: }
```

Nakon što se navede svojstvo koje se želi postaviti, dolazi dvotočka, a iza nje **vrijednost** na koju se postavlja to svojstvo (u ovom primjeru radi se o nazivu boje):

```
p{
    color: green;}
```

Svojstvo i **vrijednost** zajedno čine **deklaraciju**. Unutar jednog pravila može biti više deklaracija:

```
p{
    color: green;
    border: 1px solid red;}
```

Deklaracije obavezno moramo razdvojiti pomoću tačke sa zarezom, a poželjno je svaku deklaraciju pisati u novom redu. Iza posljednje deklaracije, u CSS-pravilu nije obavezno navesti tačku sa zarezom, ali se to ipak preporučuje (radi eventualnog naknadnog dodavanja novih deklaracija unutar istog pravila).

9.5. CSS selektori

Kao što je već rečeno, **svako CSS pravilo započinje selektorom**, a selektor određuje za koji HTML element vrijedi to pravilo. U selektoru se mogu koristiti **nazivi**, **klase** ili **identifikatori** elemenata. Najjednostavniji selektor je upravo naziv **elementa**. Ako želimo da se pravilo odnosi na p elemente, napisat ćemo:

```
p{
    color: green;}
```

Ako želimo da se pravilo odnosi na naslov drugog nivoa; umjesto p ćemo napisati h2. Često je potrebno da se CSS pravilo primijeni samo na tačno određene HTML elemente. Da bi se to postiglo, u HTML-u te elemente treba označiti klasom određenog imena, upotrebom atributa class. Na primjer:

```
<p class="zeleni-tekst">
Ne izuvaj se prije vode.</p>
```

Navedeno pravilo će vrijedit samo za elemente koji imaju postavljenu klasu zeleni-tekst. Kada se u selektoru navodi klasa, obavezno se prije imena klase mora staviti tačka:

```
.zeleni-tekst{
    color: green;}
```

Osim pomoću klase, HTML elementi se mogu označiti i identifikatorom, pomoću atributa id.

```
<p id="plavi-tekst">
Ne izuvaj se prije vode.</p>
```

Kada se u selektoru koristi identifikator, prije identifikatora se obavezno mora navesti oznaka #, kao u primjeru:

```
#plavi-tekst{
    color: blue;}
```

Ovo pravilo će se primijenit samo na element sa identifikatorom plavi-tekst. Dakle, **identifikator treba biti jedinstven[26] – na istoj HTML stranici se ne bi trebalo dva puta pisati isti identifikator**. Kod klasa nema tog ograničenja – ista klasa se može upotrijebiti samo jednom, a možemo je upotrijebiti i više puta[27].

Prilikom imenovanja klasa i identifikatora se vodi računa o određenim pravilima. Dopušteni znakovi su slova od A do Z, brojevi od 0 do 9, crtica (-) i donja crta (_). Naziv klase, odnosno identifikatora, ne smije započeti brojem (a ne preporučuje se ni linija ili donja linija). U nazivima klasa i identifikatora se mogu koristiti velika i mala slova. Treba pripaziti na to da su nazivi klasa i identifikatora osjetljivi na velika i mala slova pa tako naziv mojaKlasa nije ekvivalentan nazivu mojaklasa.[28] Ako se naziv klase sastoji od dvije riječi, preporučuje se da se one odvoje crticom (npr. moja-klasa), jer je to stil koji se također koristi i u nazivima CSS svojstava (npr. font-size). Ako nam to ne odgovara, mogu se koristiti i varijante mojaKlasa[29] ili MojaKlasa[30]. U svakom slučaju, poželjno je se dosljedno držati odabranog stila imenovanja. Također, prilikom imenovanja klasa i identifikatora je poželjno da imena budu jasna i razumljiva, ali da ipak ne budu preduga. Ako je potrebno, unutar selektora se može navesti i naziv elementa i klase:

```
p.plavi-tekst{
    color: blue;
}
```

Prethodno pravilo će se primijenit samo na p elemente sa klasom .plavi-tekst, ali ne i na druge p elemente, niti na elemente h1 koji eventualno imaju klasu .plavi-tekst. Slična kombinacija moguća je i sa identifikatorom:

```
p#plavi-tekst{
    color: blue;
}
```

Kombinaciju koju često možemo sresti su dvije (ili više) klasa u identifikatoru. Naime, na HTML element se može postaviti više klasa, što je ponekad korisno (npr. kad se element želi dodatno stilizovati). Postavljanje više klasa na element u HTML-u izgleda ovako:

```
<p class="plavi-tekst crni-okvir">
Ne izuvaj se prije vode.</p>
```

[26] Iako se korištenje istog identifikatora više puta ne preporučuje (jer takav HTML nije validan), ponekad se može sresti u praksi (jer većina preglednika to ipak podržava).
[27] Preporuka je potpuno izbjegavati donju crtu (_) u nazivima klasa i identifikatora, jer to nije podržano u nekim starijim preglednicima. Za imenovanje identifikatora i klasa se najčešće koristi tzv. kebab-case (npr. .moj-stil, #crveni,...)
[28] CSS je case-sensitive.
[29] Camel case
[30] Pascal case

Unutar atributa class su različite klase odvojene razmakom. Taj element ima dvije klase – npr. klasu .poslovica i klasu .istina. Kad se u CSS selektoru žele navesti obje klase, navode se bez razmaka. Važno je zapamtiti da se u selektoru prije naziva klase uvijek navodi tačka. Na primjer:

```
.plavi-tekst.crni-okvir{
    color: blue;
    border: 1px solid black;}
```

Ovo pravilo će se primijenit samo na elemente koji imaju i jednu i drugu klasu. Ovo pravilo se može dodatno „pooštriti“ navođenjem da se ono odnosi samo na elemente h4:

```
<h4>.plavi-tekst.crni-okvir{
    color: blue;
    border: 1px solid black;}
```

Dva (ili više) pravila koja sadrže iste deklaracije se mogu, radi preglednosti napisati kao jedno pravilo. Pogledajmo ova dva pravila:

```
.p{
    color: blue;}
h5{
    color: blue;}
```

Prethodna dva pravila, sa obzirom da sadrže iste deklaracije, mogu se napisati kao jedno pravilo koje ima dva selektora odvojena zarezom:

```
h5, p{
    color: blue;}
```

U sljedećem slučaju se također radi o dva pravila sa istim deklaracijama. Kod prvog pravila je selektor naziv **elementa**, a kod drugog **klasa**:

```
.plavi-tekst{
    color: blue;}
h5{
    color: blue;}
```

I ovdje se ta dva pravila mogu napisati kao jedno pravilo sa dva selektora:

```
h5, plavi-tekst{
    color: blue;}
```

CSS pravilo se može dodatno precizirati navođenjem roditeljskih elemenata. Na primjer:

```
p em{
    color: blue;}
```

Selektor za takvo pravilo se može čitati sa desna na lijevo. Dakle, pravilo će vrijediti za elemente span, ali samo za one koji se nalaze unutar elementa p. Na primjer:

```
<h3><em>Narodne izreke:</em></h3><br/>
<p><span>Ne izuvaj se prije <em>vode</em>!</span>Ispeci pa <em>reci!</em></p>
```

Narodne izreke:

Ne izuvaj se prije *vode*!Ispeci pa *reci!*

To je pravilo primijenjeno na elemente span koji se nalaze unutar elementa p, bez obzira da li im je element p direktni roditelj (element em sa tekstom „reci“) ili se između njih nalazi jedan ili više nivoa (element em sa tekstom „vode“). To pravilo, kao što je i očekivano, nije djelovalo na element em koji se nalazi unutar elementa h3 (element em sa tekstom „Narodne izreke:“). Ako bi željeli da se to pravilo primijeni samo na element em koji se nalazi unutar elementa span, koji se nalazi unutar elementa p (u primjeru bi to bio element em sa tekstom „vode“), navest ćemo slijedeći selektor:

```
p span em{
    color: blue;}
```

Broj nivoa koje možemo navesti nije ograničen, ali se u praksi ne preporučuje navoditi preveliki broj nivoa (zbog performansi i zbog mogućih grešaka). Također, moguće je ograničiti da se to pravilo primijeni samo na elemente em koja su direktna djeca elementa p (u ovom primjeru samo na element em sa tekstom „reci“):

```
p > em{
    color: blue;}
```

Oznaka > u selektoru označava da se on odnosi na elemente em koji su direktna djeca elementa p. Umjesto naziva roditeljskog elementa, moguće je navesti njegovu klasu. Na primjer:

```
plavi-tekst  em{
    color: blue;}
```

Ovaj selektor se odnosi na elemente em koji se nalaze unutar elementa sa klasom .plavi-tekst. Nnaziv drugog elementa se može zamijeniti klasom. U ovom primjeru je naveden selektor koji se odnosi na elemente sa klasom .info koji se nalaze unutar elementa sa klasom .main:

```
.main .info{
    color: blue;}
```

Osim prema klasama i identifikatorima, HTML elementi se mogu selektovati i po nekim drugim atributima. Jedan od najčešćih slučajeva gdje je to korisno su elementi input. Ako se u HTML kodu nalazi element input tipa text; kao u primjeru:

```
<input type="text" value="Unesite prezime: "/>
```

Sljedeće pravilo će se primijeniti samo na elemente input tipa text:

```
input[type="text"]{
color: red;}
```

Dakle, u selektoru je (u uglastim zagradama) potrebno navesti naziv atributa (type), a zatim i njegovu vrijednost (text). U CSS-u postoji i posebni selektor, tzv. **univerzalni selektor** pomoću kojeg se mogu odabrati svi elementi. Oznaka za univerzalni selektor je * (zvjezdica). Pravilo će postavit boju svih elemenata u HTML dokumentu na plavu. Na primjer:

```
*{
    color: blue;}
```

9.6. Prioriteti CSS pravila

Ako želimo postaviti pravilo koje svim elementima p postavlja boju teksta na npr. plavu, napisati ćemo:

```
p{
    color: blue;}
```

Ako želimo postaviti pravilo koje svim elementima sa klasom .crvena-boja postavlja boju teksta na crvenu, može se napisati:

```
.poslovica{
    color: red;}
```

Na taj HTML element će se primijeniti oba, prethodno navedena CSS pravila:

```
<p class="poslovica">
Ne izuvaj se prije vode.</p>
```

Budući da oba CSS pravila postavljaju svojstvo color, koje od njih će imati prioritet je određeno samim selektorom. CSS pravilo čiji selektor ima veću specifičnost će imati veći prioritet, te će ono odlučiti vrijednost svojstva. Navođenje klase je specifičnije od navođenja naziva elementa, pa će stoga drugo pravilo imati viši prioritet i boja teksta u elementu će biti crvena. Još specifičnije od klase je navođenje identifikatora, pa pravilo koje u selektoru navodi identifikator ima viši prioritet od oba navedena pravila. Kada se u selektoru nalazi kombinacija naziva

elementa, klase i identifikatora i kad se u selektoru navode i roditeljski elementi; u takvim slučajevima se može izračunati specifičnost selektora da bi se znalo koje pravilo će imati prioritet. Specifičnost selektora se računa na slijedeći način:

Vrsta selektora	Specifičnost
<p>	1
.klasa	10
#identifikator	100
!importaint	1000

Ako u selektoru postoji više naziva elemenata, klasa ili identifikatora, njihove vrijednosti se sabiraju i tako se dobije ukupna specifičnost selektora. Selektor p, span i strong ima specifičnost 3, jer sadrži tri naziva elementa. Za selektor .poslovica i strong, specifičnost iznosi 11, jer sadrži jednu klasu i jedan naziv elementa. Taj selektor ima veću specifičnost od prethodnog.

9.7. Pseudoklase

Pseudoklase su izrazi u CSS-u koji omogućuju selektovanje elemenata slično kao klase. U CSS pravilima se koriste poput klasa, ali se za razliku od klasa ne postavljaju na elemente u HTML-u.

```
<ul>
    <li>Jagode</li>
    <li>Narandže</li>
    <li>Banane</li>
    <li>Jabuke</li>
</ul>
```

Ako želimo samo na prvoj stavci postaviti boju teksta na crvenu, trebali bi postaviti klasu na prvu stavku:

```
<ul>
    <li class="first">Jagode</li>
    <li>Narandže</li>
    <li>Banane</li>
    <li>Jabuke</li>
</ul>
```

Zatim je potrebno napisati CSS pravilo koje će stavci sa klasom .first postaviti boju na crvenu:

```
li.first{
color: red;}
```

U ovakvim slučajevima, kada se želi primijeniti neko pravilo na temelju položaja elementa, nije potrebno uvijek napraviti novu klasu, nego je moguće koristiti neku od pseudoklasa vezanih

uzpoložaj elementa. Pseudoklasa :first-child selektuj prvi element unutar roditeljskog elementa. Prilikom korištenja pseudoklase u selektoru treba navesti dvotačku (slično kao što se klasa u selektoru uvijek navodi sa tačkom). U gornjem primjeru je moguće izbaciti klasu iz HTML koda, a CSS pravilo napisati na slijedeći način:

```
li:first-child{
color: red;}
```

- Jagode
- Narandže
- Banane
- Jabuke

Ako se želi selektovati samo posljednja stavka liste, može se koristiti pseudoklasa .last-child. Ta pseudoklasa selektuje element koji se nalazi posljednji unutar roditeljskog elementa. Da bi se u posljednjoj stavci liste boja teksta postavi na svijetlo zelenu, navest ćemo slijedeću deklaraciju:

```
li:last-child{
color: yellowgreen;}
```

- Jagode
- Narandže
- Banane
- Jabuke

Ako želimo selektovati neka druga stavka u listi, može se koristiti pseudoklasa :nth-child. Toj pseudoklasi treba postaviti i parametar koji određuje na koji se element unutar roditeljskog elementa odnosi. Ako se želi selektovati samo treću stavku, napiše se:

```
li:nth-child(3){
color: yellowgreen;}
```

- Jagode
- Narandže
- Banane
- Jabuke

Ako želimo selektovati svaku drugu stavku, umjesto 3, kao parametar ćemo predati ključnu riječ even:

```
li:nth-child(even){
color: yellowgreen;}
```

- Jagode
- Narandže
- Banane
- Jabuke

To pravilo će također postaviti boju teksta na svijetlo zelenu svakoj drugoj stavci liste, preciznije svakoj parnoj stavci. Moguće je selektovati i svaku neparnu stavku, ako se umjesto ključne riječi even upotrijebi odd. Na primjer:

```
li:nth-child(odd){
color: yellowgreen;}
```

- Jagode
- Narandže
- Banane
- Jabuke

Pomoću ovih pseudoklasa je moguće različito stilizovati elemente zavisno od korisničkih akcija, te tako dodatno naglasiti informaciju o tome koji je element korisnik odabrao, ili koji je link već posjetio i time mu olakšao navigaciju i interakciju sa web stranicom. Pseudoklasa :link selektuje, bira link (element a) koji korisnik još nije posjetio. Na primjer:

```
a:link{
color: palevioletred;}
```

Pseudoklasa :visited selektuje link koji je korisnik već posjetio:

```
a:visited{
color: slateblue;}
```

Pseudoklasa :active selektuje element na koji je korisnik upravo kliknuo (npr. link ili dugme).

```
a:active{
color: seagreen;}
```

Pseudoklasa :hover selektuje element preko kojeg korisnik trenutno prelazi mišem. To može biti link, ali i drugi element poput elemenata: div, span, p ili li.

```
a:hover{
color: sienna;}
```

Pseudoklasa :focus selektuje element koji je trenutačno u fokusu (npr. element za unos u kojem se nalazi kursor ili link). Na primjer:

```
input:focus{
    color: chocolate;}
```

Ako neki element za unos (ili dugme) trenutno nije omogućen, odnosno vidljiv je, ali iz nekog razloga ne dopuštamo unos teksta ili klik na njega; svakako to želimo vizualno poručiti korisniku. Preglednici imaju svoj zadani stil u kojem prikazuju onemogućene elemente, taj stil se zapravo

može promijeniti pomoću dvije pseudoklase. Pseudoklasa :enabled selektuje elemente koji su omogućeni (npr. linkovi, dugmad ili elementi za unos). Na primjer:

```
input:enabled{
color: hotpink;}
```

Pseudoklasa :disabled selektuje elemente koji nisu omogućeni. Na primjer:

```
input:disabled{
color: sandybrown;}
```

Na dva specijalna elementa za unos (kvačica i dugme za odabir) se mogu primijeniti dvije pseudoklase koje selektuju element zavisno od njegovog stanja: da je li označen (tj. je li stavljena kvačica) ili ne. Pseudoklasa :checked selektuje elemente koji su označeni (elementi input tipa checkbox ili radio). Na primjer:

```
input:checked{
color: red;}
```

Pseudoklasa :indeterminate selektuje elemente koji nisu označeni (također input elementi tipa checkbox ili radio):

```
input:indeterminate{
    color: black;}
```

Pseudoklase imaju istu specifičnost kao i klase (također nose vrijednost 10). Selektor li:first-child će imat ukupnu specifičnost 11 (element (1) + pseudoklasa(10)).

9.8. Nasljeđivanje vrijednosti

Velik broj CSS svojstava se može nasljeđivati. Ako na elementu nije postavljena vrijednost za neko od takvih svojstava, koristit će se vrijednost sa roditeljskog elementa. To omogućuje da se osnovna svojstva kao što su boja, veličina i vrsta fonta postave samo na jednom mjestu (najčešće na elementu body), umjesto da ih treba iznova definisati za svaki element. Na primjer:

```
body{
    color: cadetblue;}
```

Ako je na elementu body postavljena boja teksta na plavu, ona će biti plava i na svim drugim elementima, bez da je treba eksplicitno postaviti, jer su svi elementi djeca elementa body. Na elementima za koje je to potrebno, moguće je postaviti različite vrijednosti:

```
div{
    color: aquamarine;}
```

Ako se za neki od elemenata želi vratiti vrijednost svojstva na naslijeđenu, to se može postići upotrebom ključne riječi inherit:

```
p.izreka{
    color: inherit;}
```

Ključnu riječ inherit je moguće koristiti i na CSS svojstvima čije se vrijednosti ne nasljeđuju. U tom slučaju se mijenja ponašanje CSS svojstva i vrijednost se nasljeđuje s roditeljskog elementa, iako to inicijalno ne bi bilo tako.

9.10. Komentari

U slučaju kad se želi pojasniti namjena nekog CSS pravila ili deklaracije, moguće je u CSS kod dodati komentare – proizvoljni tekst koji opisuje namjenu neke linije ili bloka koda. Komentari započinju slijedom znakova /*, a završavaju slijedom */. Komentar se može nalaziti u jednom redu, ali se može protezati i kroz više njih. Na primjer:

```
span{
    /*first-color*/
    color: bisque;
    /*optional
    color: yellow;*/
}
```

Preporučuje se uvijek stavljati komentare kad namjena napisane CSS deklaracije nije na prvi pogled jasna. Komentari se često koriste za grupisanje više CSS pravila koji imaju zajedničku namjenu. Prilikom testiranja se komentari često koriste da bi se privremeno isključio dio koda. Kod koji se nalazi unutar oznake za komentare se neće primijeniti, već će biti ignorisan (jednako kao da se radi o običnom komentaru).

9.11. Uslovni komentari

Uslovni komentari (engl. *conditional comments*) služe isključivo za pretraživač Internet Explorer, od verzije 5 do verzije 9, sa ciljem da daju dodatne instrukcije tim pretraživačima. Veliki problem kod CSS-a je što pretraživači različito interpretiraju pojedina CSS svojstva. Ovaj problem u IE-u se rješava primjenom uslovnih komentara. Na ovaj način, programer po potrebi piše npr. dva eksterna CSS fajla:

- jedan sa svojstvima i vrijednostima namenjenim grupi pretraživača;
- drugi za Internet Explorer.

Da bi kod „znao“ kada koji CSS fajl da pozove, potrebno je dodati kod za prepoznavanje IE, i učitavanje CSS namenjenog za IE. Kako ne postoji posebna komanda za ovakvu akciju, koristi se proširena verzija komentara – **uslovni komentar**. Primjer primjene uslovnog komentara za IE verzija 8, kojim se učitava eksterni CSS fajl:

```
<!--[if IE 8]>
 <link rel="stylesheet" type="text/css" href="style-special.css" />
<![endif]-->
```

U slučaju da se želi pisati različit CSS kod za različite verzije IE, ova grupa koda se ponavlja za svaku verziju IE-a. U slučaju da se neke posebne napomene žele uraditi od verzije IE10, to je moguće uraditi primjenom skriptnog jezika.

U poglavlju je obrađeno:

- šta je CSS;
- CSS sintaksa;
- CSS selektori i prioriteti CSS pravila;
- pseudoklase;
- nasljeđivanje vrijednosti;
- CSS komentari;
- CSS uslovni komentari
- dodavanje CSS koda direktno u HTML kod;
- uključivanje vanjske CSS datoteke u HTML dokument.

X Modifikacija teksta

Po završetku ovog poglavlja, polaznik će moći:

- mijenjati boju sadržaja;
- mijenjati vrstu, veličinu, „težinu“ i stil fonta;
- skraćeno navoditi svojstva fonta;
- koristiti poravnanje i razmak sadržaja;
- koristiti svojstva velikih i malih slova;
- ukrašavati tekst;
- koristiti web fontove;
- Koristiti Google fontove.

CSS nudi brojne mogućnosti za oblikovanje teksta – od osnovnih kao što je odabir boje, veličine i vrste fonta, pa do onih rjeđe korištenih svojstava pomoću kojih se može upravljati proredima i razmakom između slova. U novije vrijeme CSS nudi i mogućnosti kao što je postavljanje sjene na slova, te korištenje fontova koje korisnik ne mora imati instalirane na svom računaru, što je značajno unaprijedilo tipografiju na web-u.

10.1. Boja teksta

Boju teksta postavljamo pomoću svojstva color, sa kojim smo se već susretali. Ako vrijednost boje nije postavljena, ona će biti naslijeđena od roditeljskog elementa. Vrijednost svojstva color je moguće postaviti na više načina – preko naziva boje, te preko posebnog načina zapisivanja boje – heksadecimalnog koda ili preko zapisa boja u modelima RGB ili HSL. Na slijedećem linku (www.w3schools.com/colors/colors_names.asp) ćete pronaći 140 predefinisanih boja koje su definisane imenom, a podržavaju ih svi moderni pretraživači. Nazivi boja su vrlo praktični i jednostavni za korištenje, ali se na taj način može koristiti ograničen broj boja. Osim ključnih riječi za naziv boje, može se koristiti i ključna riječ inherit – boja će biti postavljebna na vrijednost roditeljskog elementa. Najčešći način postavljanja deklaracije boje je navođenjem heksadecimalnog zapisa boje. Na primjer:

```
color: #a3118a;
```

Takav kod se sastoji od 6 heksadecimalnih brojeva, a prije njega se obavezno navodi oznaka #. Kao cifre se mogu pojaviti brojevi od 0 do 9 i slova a, b, c, d, e ili f. Za prikaz boje na računarima se koristi model boja RGB, gdje je svaka boja predstavljena brojčanom vrijednosti svoje: crvene (engl. *red*), zelene (engl. *green*) i plave (engl. *blue*) komponente. Svaka komponenta može poprimiti vrijednost od 0 do 255. Na ovaj način se može zapisati više od 16 milijona boja.

Heksadecimalni zapis boje se sastoji od vrijednost tih triju komponenti. Prva dva numerička zapisa heksadecimalnog koda boje predstavljaju vrijednost crvene, druge dvije zelene, a posljednje dvije plave komponente boje. Nedostatak takvog načina zapisivanja boja je teško čitljiva vrijednost svojstva (nemoguće prepoznati o kojoj se boji radi). Na sreću, za korištenje heksadecimalnog koda[31] boje nije potrebno poznavati njegov način računanja, već je moguće kopirati heksadecimalni kod iz grafičkog programa ili online alata za odabir boje. Preporučuje se upotreba malih slova u pisanju heksadecimalnih kôdova. Dakle, bolje je pisati #f5412a nego #F5412A, premda će oba načina pisanja funkcionisati. Razlog je dosljednost u pisanju – u cijelom CSS-u se preporučuje upotreba isključivo malih slova.

Osim pomoću heksadecimalnog zapisa, u CSS-u postoji mogućnost postavljanja boje direktnim navođenjem vrijednosti njezinih komponenti (**R**, **G** i **B**). Kao što je navedeno, svaka komponenta može poprimiti vrijednost od 0 do 255, a navode se u pozivu rgb funkcije:

```
color: rgb(77, 86, 58);
```

Osim veće čitljivosti, RGB zapis boje ima još jednu prednost, da se može proširiti dodavanjem još jedne komponente – **prozirnosti**. Takav zapis naziva se **RGBa** zapisom u kojem komponenta **a** (engl. *alpha channel*) je prozirnost boje. Moguće vrijednosti za prozirnost su decimalni brojevi od 0 do 1, koji predstavljaju postotak prozirnosti (0.3 znači 30%, dok 1 znači 100 %). U narednom primjeru se postavlja prozirnosti od 80 % na boju iz prethodnog primjera. RGB vrijednosti boja se također mogu kopirati iz grafičkih programa i koristiti u CSS kodu, vrijednosti se navode u pozivu rgba funkcije:

```
color: rgba(2, 65, 22, 0.8);
```

HSL model je još jedan način zapisivanja boje preko njezinih komponenti, komponente ovog puta su: nijansa (engl. *hue*), zasićenost (engl. *saturation*) i svjetlo (engl. *lightness*). Upotrebom tog modela se lahko mogu dobiti svjetliji i tamniji tonovi iste boje – mijenjanjem zasićenosti i svjetline, dok je to u modelu RGB nepraktično. Kada se u CSS-u koristi model HSL, vrijednost prve komponente (nijansa) se postavlja na broj od 0 do 360, vrijednosti druge dvije komponente (zasićenje i svjetlina) na postotak od 0 do 100% (ovaj put se ne koriste decimalni brojevi nego postotci). Na primjer:

```
color: hsl(7, 91%, 36%);
```

[31] Postoji skraćeni način pisanja heksadecimalnih kodova boja u CSS-u. Ako se heksadecimalni kod sastoji od ponavljajućih brojeva, možemo ga zapisati sa samo 3 broja. Npr. kod #ff3300 u skraćenom zapisu postaje #f30.

Kao i kod modela RGB, i ovdje je moguće postaviti prozirnost kao dodatnu komponentu:

```
color: hsl(7, 91%, 36%, 0.7);
```

Potrebno je voditi računa da se prozirnost i dalje postavlja sa decimalnim brojem od 0 do 1 (a ne sa postotkom). Model HSL je ekvivalent modelu RGB, odnosno može prikazati isti broj boja, a moguća su preračunavanja iz jednog modela u drugi. Treba biti pažljiv prilikom kopiranja HSL vrijednosti iz grafičkih programa, jer se oni često koriste varijacijom tog modela (HSB model) pa dobivena boja nije posve identična. Najbolje je se koristiti alatima specijaliziranim za odabir boje u HSL zapisu[32].

10.2. Vrste fonta

Vrsta fonta se postavlja pomoću svojstva font-family. Naziv svojstva font-family se odnosi na „familiju" fontova, npr. Arial. Unutar „familije" se nalazi više verzija fonta: Helvetica, Regular, Helvetica Bold, Helvetica Italic,... Kao vrijednost se navodi naziv fonta:

```
font-family: Helvetica;
```

Ako naziv fonta sadrži razmake (ili specijalne znakove), potrebno ga je navesti u navodnicima. Mogu se koristiti jednostruki ili dvostruki navodnici.

```
font-family: "Open Sans";
```

Ako font koji je naveden nije instaliran na korisnikovom računaru, preglednik će prikazati svoj predefinisani font (najčešće Times New Roman). Da bi se to izbjeglo, može se navesti jedan ili više zamjenskih fontova (odvojenih zarezom) koji će se upotrijebiti u slučaju da prvi font nije instaliran:

```
font-family: Helvetica, Arial;
```

Ako nijedan od zamjenskih fontova nije instaliran, i dalje će se prikazati preglednikov predefinisani font. Međutim, moguće je odabrati koji će tip fonta biti – sa **serifima** (najčešće *Times New Roman*) ili **bez serifa** (najčešće *Arial*). Na kraju popisa fontova se navodi ključna riječ:

```
font-family: Helvetica, Arial, sans-serif;
```

Na raspolaganju je pet osnovnih fontova: serif (npr. Times New Roman), sans-serif (npr. Arial), monospace (npr. Courier New), cursive (npr. Comic Sans MS) i fantasy (npr. Impact). Međutim, ne preporučuje se oslanjati se na cursive i fantasy. Preglednici (često) prikazuju različite fontove

[32] Zapisi boja: RGB, RGBa, HSL i HSLa su podržani tek u novijim verzijama preglednika (IE9+, Firefox 3+, Opera 10+)

za cursive, a fantasy uopće nije podržan u nekim preglednicima (npr. Firefox). Vrsta fonta se, ako nije postavljena, također nasljeđuje od roditeljskog elementa.

2.3. Veličina fonta

Veličina fonta se postavlja pomoću svojstva font-size. Vrijednost tog svojstva se može postaviti na više načina: pomoću ključnih riječi, pomoću apsolutnih jedinica (pikseli i tačke), te pomoću relativnih jedinica (postotci, em ili rem jedinice). Na primjer:

```
font-size: large;
```

Predefinisane riječi za regulaciju veličine fonta su: xx-small, x-small, small, medium, large, x-large, xx-large, larger (za jedan nivo veća slova od roditeljskog elementa) i smaller (za jedan nivo manja slova od roditeljskog elementa). U starijim preglednicima (IE 8 i stariji) korisnici nisu mogli povećati veličinu teksta postavljenu pomoću piksela, te time sebi olakšati čitanje. Zbog toga se preporučivalo ne koristiti se pikselima; u modernim preglednicima taj problem više ne postoji. Veličina fonta se također, ako nije eksplicitno postavljena, nasljeđuje od roditeljskog elementa. **Ako veličina fonta ne postavi, tekst će se prikazivati u predefinisanoj veličini koja je kod većine preglednika 16 piksela.**

Veličina se fonta se često definiše pomoću piksela. Ovdje se radi o mjernoj jedinici koja je definisana veličinom pojedine tačke (tj. piksela) na ekranu računara. Jedinica se koristi kada je potrebno precizno definisati veličinu teksta (da bi u potpunosti odgovarala zadanom dizajnu). Na primjer:

```
font-size: 24px;
```

Nova veličina teksta koja je definisana u pikselima će ipak varirati od preglednika do preglednika (jer se svaki preglednik koristi vlastitim algoritmom za renderovanje fonta). Neki mobilni uređaji (npr. iPhone i neki Android uređaji) imaju vrlo visoku rezoluciju, pa se na njima tekst veličine 12 piksela prikazuje kao vrlo malen, dok je inače normalno čitljiv na stolnim računarima.

Tačke *(engl. points)* su mjerna jedinica koja se koristi kod definisanja CSS-a za ispis web stranice. Jedna tačka ispisana na papir zauzima 1/72 inča, odnosno 0.035 cm. Tačke su namijenjene za korištenje samo u CSS-u za ispis, te se ne preporučuje da se koriste za veličinu teksta koji će biti prikazan na web-u, jer će rezultat varirati od preglednika do preglednika. Na primjer:

```
font-size: 24pt;
```

Osim apsolutnim definisanjem veličine fonta pomoću piksela (ili tačaka), korištenjem postotaka se veličina fonta može definisati u odnosu na roditeljski element. Taj način postavljanja veličine fonta je sličan kao da se koristimo ključnim riječima smaller i larger, s tim da možemo preciznije definisati razliku u veličini. Na primjer, za veličinu fonta koja je za 37.5% manja od roditeljskog elementa:

```
font-size: 62,5%;
```

Prednost relativnog definisanja veličine fonta je u tome što je na jednom mjestu moguće promijeniti veličinu fonta za sve elemente. Često se na elementu body postavi početna veličina fonta, a za sve druge elemente se veličina fonta postavi relativno u odnosu na njihove roditeljske elemente. Tako se promjenom veličine fonta na elementu body proporcionalno mijenja veličina fonta na svim elementima.

Naziv jedinice em dolazi od slova M – ta jedinica originalno dolazi iz tipografije, a predstavljala je širinu velikog slova M. Jedinica **em** je relativna mjerna jedinica za dužinu koja se koristi slično kao i postotci. 1 em predstavlja veličinu fonta roditeljskog elementa, 0.5 em predstavlja upola manju, a 2 em predstavlja dva puta veću veličinu fonta od roditeljskog elementa. Da bi se postavila veličina fonta na 0.4 (odnosno na 40% od veličine roditeljskog elementa), pisat ćemo slijedeće:

```
font-size: 0.4em;
```

Kod postotaka i kod jedinice em se pojavljuje jedan problem: veličinu fonta uvijek treba zadati u odnosu na roditeljski element. To je problematično za praćenje kod elemenata koji se nalaze na različitom nivou. Ako se za element body definiše veličina od 16 piksela, a potom se za elemente p i span veličina definiše kao 75 % od veličine roditeljskog elementa, time ćemo elementima postaviti veličinu fonta od 12 piksela. Ako unutar elementa p imamo ugniježđen element span, taj element će imati veličinu fonta 75 % od svojeg roditelja, odnosno 9 piksela, što najčešće nije željeno stanje.

Jedinica **rem** (engl. *root em*) je rješenje gore opisanog problema. Radi se također o relativnoj mjernoj jedinici za dužinu, ali veličina fonta se navodi u odnosu na korijenski (engl. *root*) element (što je zapravo element html). Dakle, nije potrebno voditi računa o tome koji se element nalazi unutar kojeg i računati iznos razlike u postocima za svaki, već se sve veličine definišu u odnosu na veličinu fonta postavljenu na elementu html. Rem jedinicu ne podržavaju stariji preglednici

(IE 8 i drugi stariji preglednici). Na primjer, postavit ćemo veličinu fonta koja je za 60% manja od veličine fonta html elementa:

```
font-size: 0.4rem;
```

Pikseli, **tačke**, **em** i **rem** su mjerne jedinice CSS-a za dužinu. Osim za veličinu fonta, koriste se i za definisanje veličine pravougaonih elemenata, te na drugim mjestima gdje treba definisati određenu dužinu. Postotci u CSS-u se također mogu koristiti za definisanje veličine fonta i pravougaonih elemenata, ali ih ipak ne možemo koristiti na svim mjestima gdje i jedinice za dužinu.

10.4 „Debljina“ i stil fonta

„**Težina**“ (pojačanje ili debljina) **fonta** se postavlja pomoću svojstva font-weight. Kao vrijednost se može navesti ključna riječ:

```
font-weight: bold;
```

Predefinisane riječi za „težinu“ fonta su: normal (standardna „težina“ fonta), bold (boldirani font), bolder („težina“ fonta je veća za jednu vrijednost od roditeljskog elementa – ako je dostupna težina fonta), lighter („težina“ fonta je manja za jednu vrijednost od roditeljskog elementa – ako je dostupna težina fonta). Ključna riječ normal će se koristit kada se želi poništiti drugo pravilo koje je postavilo težinu elementa na bold. Vrijednost ovog svojstva se također nasljeđuje sa roditeljskog elementa. Osim ključnih riječi, koriste se i numeričke vrijednosti:

```
font-weight: 600;
```

Vrijednost koje se mogu koristiti su: 100 („slabije“ od normal za tri „težine), 200 („slabije“ od normal dvije „težine“), 300 („slabije“ od normal jedne „težine“), 400 (ekvivalent sa normal), 500 („jače“ od normal jednu „težinu“), 600 („jače“ od normal za dvije „težine“), 700 (ekvivalent bold), 800 („jače“ od bold za jednu „težinu“), 900 („jače“ od bold za dvije „težine“). Ako postavimo vrijednost „težine“ koja je nedostupna na računaru korisnika, u tom slučaju se prikazuje najbliža dostupna „težina“. Da bi korisnik vidio tekst u postavljenoj „težini“, na njegovom računaru mora biti instalirana verzija fonta u toj „težini“. Većina fontova instalirana je samo u „težinama“ normal (400) i bold (700). U praksi se stoga najčešće koriste samo ključne riječi bold i normal.

Stil fonta se postavlja pomoću svojstva font-style. Kao vrijednost se navodi ključna riječ:

```
font-style: italic;
```

Na raspolaganju su slijedeće ključne riječi: normal, italic i oblique. **Italic fontovi** su obično namjenski dizajnirani da izgledaju nakošeno, dok su fontovi **oblique** dobiveni zakošenjem originalnog fonta za određeni broj stepeni. Slično kao i kod „težina“, moguće je koristiti samo stilove koje korisnik ima instalirane na računaru. Većina fontova dolazi zajedno sa verzijom italic, pa se najčešće koristi ta ključna riječ. Ključna riječ normal se koristi kada se želi poništiti drugo pravilo koje je postavilo stil elementa na italic. Vrijednost ovog svojstva se također nasljeđuje sa roditeljskog elementa. Većina fontova dolazi instalirana i sa verzijom Bold i Italic, što čini mogućim da se istom elementu istovremeno postavi težina na bold i stil na italic.

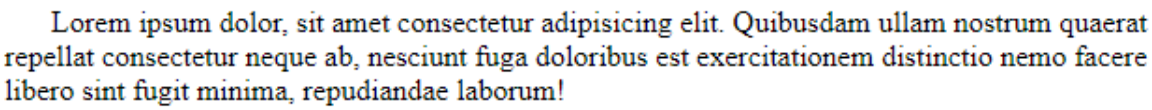
10.5. Poravnanje i razmak

Za određivanje **poravnanja teksta** se koristi svojstvo text-align. Moguće je postaviti jednu od slijedećih ključnih riječi: left (poravnanje uz lijevu ivicu roditelja), right (poravnanje uz desnu ivicu roditelja), center (centrira sadržaj na sredini roditelja) i justify (obostrano poravnanje – zauzima svu moguću dostupnu širinu roditelja). Vrijednost ovog svojstva se nasljeđuje sa roditeljskog elementa. Za razliku od štampanog teksta (npr. novina), na web-u se rijetko susreću uske kolone i tekst kojem je poravnata lijeva i desna ivica (justify). Na web-u prevladava lijevo poravnati tekst, a ostali načini poravnanja se koriste samo u specifičnim situacijama. Na primjer:

```
text-align: justify;
```

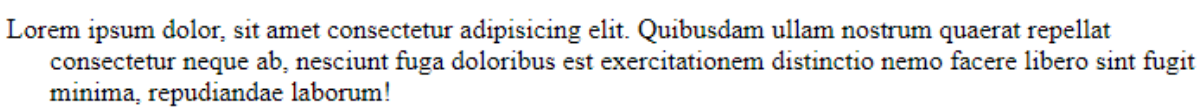
U štampanom tekstu se često uvlači prvi red na svakom odlomku. Na web-u je to rijedak slučaj, ali je to moguće postići upotrebom svojstva text-indent. Svojstvo kao vrijednost prima širinu za koju će tekst biti „uvučen“. Vrijednost ovog svojstva se nasljeđuje sa roditeljskog elementa. Moguće je se koristiti mjernim jedinicama za dužinu u CSS-u: pikselima, tačkama, jedinicama em i rem, kao i postotcima. Na primjer:

```
text-indent: 25px;
```



Ako se definiše širina za koju se tekst uvlači preko postotka, radit će se o postotcima od širine elementa u kojem se nalazi tekst. Ovom svojstvu se može postaviti i negativna vrijednost – u tom slučaju će tekst, umjesto da bude uvučen, biti „izbačen“ ulijevo:

```
text-indent: 25px;
```



Pomoću svojstva letter-spacing je moguće podešavati **razmak između pojedinih slova**. Dakle, vrijednost koja se predaje je širina razmaka koja se želi dodati na postojeći razmak između slova. Za mjernu jedinicu se mogu koristiti pikseli, tačke, jedinice em ili rem, ali ne i postotci.

```
letter-spacing: 3px;
```

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Quibusdam ullam nostrum quaerat repellat consectetur neque ab, nesciunt fuga doloribus est exercitationem distinctio nemo facere libero sint fugit minima, repudiandae laborum!

Osim navođenja iznosa širine, moguće je navesti i ključnu riječ normal, čime se razmak postavlja na standardni razmak u trenutnom fontu. Tako se za neki element može isključiti djelovanje drugog CSS pravila koje postavlja ovo svojstvo. Vrijednost ovog svojstva se također nasljeđuje sa roditeljskog elementa.

Također, možemo podešavati i **razmak između riječi**, upotrebom svojstva word-spacing. Kao vrijednost predajemo željenu širinu razmaka koju možemo definisati u pikselima, tačkama, em ili rem jedinicama (ali ne i u postotcima). Na ovom svojstvu se može navesti ključna riječ normal, čime se razmak postavlja na standardni razmak u trenutnom fontu. Vrijednost ovog svojstva se nasljeđuje sa roditeljskog elementa. Na primjer:

```
word-spacing: 10px;
```

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Quibusdam ullam nostrum quaerat repellat consectetur neque ab, nesciunt fuga doloribus est exercitationem distinctio nemo facere libero sint fugit minima, repudiandae laborum!

Visina reda u kojem se nalazi tekst se može mijenjati pomoću svojstva line-height. Kao vrijednost se postavlja željeni iznos visine. Mogu se koristiti pikseli, tačke, jedinice em ili rem, a također se mogu koristiti postotci ili bezdimenzionalni broj.

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Quibusdam ullam nostrum quaerat repellat consectetur neque ab,

nesciunt fuga doloribus est exercitationem distinctio nemo facere libero sint fugit minima, repudiandae laborum!

Ako se koriste jedinice em, postotci ili broj bez mjerne jedinice, vrijednost visine se računa na temelju veličine fonta elementa u kojem se tekst nalazi. U ovom primjeru ona bi bila dvostruka od veličine fonta na roditeljskom elementu:

```
line-height: 2em;
```

Ista visina reda će se dobiti ako se izostavi jedinica em:

```
line-height: 2;
```

Kao i kod jedinica em, broj koji se navodi može biti decimalni broj:

```
line-height: 1.4;
```

Na ovom svojstvu se također može navesti ključna riječ normal, čime se visina reda postavlja na predefinisanu vrijednost, koja zavisi od preglednika. Vrijednost ovog svojstva se nasljeđuje sa roditeljskog elementa. Postoji razlika između korištenja jedinice em i broja bez jedinice. U slučaju kad se vrijednost svojstva line-height nasljeđuje od roditeljskog elementa, koriste li se jedinice em (ili postotci), one će se računati na temelju veličine fonta roditeljskog elementa, a koriste li se brojevi bez jedinice, oni će se računati na temelju veličine fonta elementa u kojem se tekst nalazi.

Iz estetskih razloga je ponekad potrebno **cijeli tekst u nekom elementu** (npr. naslovi, navigacija) **ispisati velikim slovima.** Da bi se to postiglo, ne mora se ručno mijenjati tekst u HTML datoteci, već se može koristiti CSS-svojstvo text-transform:

```
text-transform: uppercase;
```

Predefinisane ključne riječi za svojstvo text-transform su: uppercase (cijeli sadržaj je ispisan velikm slovima), lowercase (sva mala slova), capitalize (svaka riječ započinje velikim slovom) i none (isključuje se djelovanje nekog od prethodno navedenih pravila). Vrijednost ovog svojstva se nasljeđuje sa roditeljskog elementa.

Uz ovo svojstvo, postoji i svojstvo font-variant, kojim se postavlja **specifičan prikaz teksta**, u kojem su sva slova velika, ali su „prava" velika slova prikazana većima od drugih. Pored small-caps vrijednost svojstava, postoji još i normal za standardan prikaz fonta. Na primjer:

```
font-variant: small-caps;
```

Pomoću CSS-a se tekst može **podvući**, **precrtati** ili se može postaviti **linija iznad teksta**. Za to se koristi svojstvo text-decoration:

```
text-decoration: underline;
```

Pored underline postoje još: overline (tekst sa linijom iznad), line-trough (precrtani tekst), te none (isključuje djelovanje drugog pravila). Vrijednost svojstva se **ne nasljeđuje** sa roditeljskog elementa. Umjesto samo jedne, kod svojstva text-decoration možemo postaviti više ključnih riječi, npr.:

```
text-decoration: underline line-through;
```

~~Lorem ipsum dolor.~~

Svojstvo text-shadow je podržano u većini novijih preglednika, a Internet Explorer ga podržava tek od verzije 10. U novijim preglednicima se može dodati **sjena tekstu**, postiže se svojstvom text-shadow. Vrijednost ovog svojstva se postavlja navođenjem četiri parametara (boja sjene, horizontalni pomak – x osa, vertikalni pomak – y osa, te radijus zamućenja). Na primjer:

```
text-shadow: blueviolet 3px 3px 3px;
```

Parametri koji se navode kao vrijednost svojstva text-shadow se mogu navesti na slijedeći način: horizontalni pomak, vertikalni pomak, radijus zamućenja i boja. Boja sjene se može postaviti pomoću imena boje, pomoću heksadecimalnog koda ili kodova RGBa i HSLa, a ostale tri vrijednosti u jedinicama dužine (pikselima i drugim jedinicama). Vodoravni i okomiti pomak određuju za koliko će sjena biti pomaknuta u odnosu na tekst. Radijus zamućenja je dužina za koju je tekst raširen, što stvara efekt zamućenja. Taj parametar se može izostaviti i tada će njegova vrijednost biti 0. Bez navođenja radijusa zamućenja dobivamo jednoliku boju sjene:

```
text-shadow: blueviolet 3px 3px 3px;
```

Lorem ipsum dolor.

Navede li se radijus zamućenja, sjena je zamućena, što može izgledati efektnije:

Lorem ipsum dolor.

Na tekst se istovremeno može postaviti više sjena, ako se definicija svake sjene odvoji zarezom:

```
text-shadow: blueviolet 3px 3px 3px,
             green 1px 3px 3px,
             yellowgreen 1px 1px;
```

Lorem ipsum dolor.

Ako se za pomak sjene navede negativna vrijednost, sjena će biti smještena sa lijeve, odnosno sa gornje strane slova. U ovom primjeru je dodana bijela sjena sa lijeve i sa gornje strane, čime je postignut efekat odsjaja:

```
text-shadow: blueviolet -2px -2px 2px,
             green 1px 3px 3px,
             yellowgreen 1px 1px;
```

Lorem ipsum dolor.

Za vrijednost tog svojstva se može postaviti i ključna riječ none. To se koristi kada se želi isključiti sjena koja je postavljena nekim drugim CSS pravilom. Vrijednost ovog svojstva se isto nasljeđuje sa roditeljskog elementa.

10. 8. Skraćeno pisanje svojstava fonta

Osnovna svojstva za oblikovanje fonta se mogu postaviti na skraćeni način korištenjem samo jednog svojstva, svojstva font. Svojstva koja se mogu postaviti na taj način su: font-style; font-variant; font-weight; font-size; line-height i font-family. Vrijednosti se postavljaju na način:

```
font: style variant weight size/line-height family;
```

Vrijednosti treba navesti gore navedenim redoslijedom, a neka svojstva se smiju i izostaviti. Obavezna svojstva su: font-size i font-family. U primjeru ćemo postavit sva moguća svojstva:

```
font: italic small-caps bold 14px/20px "Lato";
```

Ako se na jednom mjestu u CSS-u navede: `font-weight: bold`, a zatim kasnije navede `font: 14px Arial`, deklaracija koja je došla kasnije će imat prednost i „težina“ fonta će biti normal (jer je izostavljanjem u drugoj deklaraciji postavljena na normal). Vrijednosti odvojene razmakom, osim vrijednosti svojstava font-size i line-height, koje su odvojene kosom linijom. Ako se npr. izostavi vrijednost za line-height, izostavit će se i kosa linija:

```
font: italic small-caps bold 14px Arial;
```

Kao što je već rečeno, navesti treba samo svojstva font-size i font-family. Ovako bi izgledalo najjednostavnije korištenje svojstva font:

```
font: 14px Arial;
```

Vrijednosti koje nisu navedene, će bit postavljene na vrijednost normal. Kao vrijednost za font-family je moguće navesti više naziva fontova, tada ih odvajamo zarezom (opcionalno ih možemo odvojiti i razmakom):

```
font: 14px Arial, Verdana, sans-serif;
```

10.9. Web fontovi

Korištenje fontova u CSS-u je dugo vremena bilo ograničeno samo na fontove koji su instalirani kod korisnika na računaru. Tu nažalost nije bilo mjesta za neke egzotičnije fontove, već su se dizajneri web-stranica morali ograničiti na tek desetak fontova koje su mogli koristiti relativno sigurno. U posljednje vrijeme se situacija promijenila, pa se danas, osim fontova instaliranih na korisnikovu računaru, mogu koristiti i tzv. web-fontovi[33]. Datoteka sa fontom se postavlja na

[33] Fontovi su autorska djela, pa prilikom korištenja web fontova, obavezno treba voditi računa o licenci pod kojom se koristimo fontom. Ako je font instaliran na našem računaru, imamo samo licencu koja nam dozvoljava da ga koristimo na našem računaru.

web poslužitelj, a korisnički preglednik preuzima tu datoteku i prikazuje tekst u željenom fontu. Za uključivanje web fonta je potrebno koristiti specijalno pravilo @font-face:

```
@font-face {
font-family: Fenix; src: url('Fenix.woff');
}
```

Tamo gdje se nalazi selektor navodi se znak @ i ključna riječ font-face, a u vitičastim zagradama se navode svojstva kojima definišemo font (u ovom primjeru samo font-family i src, a u složenijim primjerima su moguća i druga svojstva). Svojstvom font-family se definiše koji će se naziv koristiti za ovaj font u CSS kodu (u ovom primjeru Fenix), a pomoću svojstva src se navodi putanja do datoteke fonta (Fenix.woff) kojom ćemo se koristiti. Putanja se predaje preko CSS funkcije url. Tamo gdje se želi navesti taj font, upotrijebit će se njegov naziv koji je prije naveden (uz obavezno navođenje zamjenskih fontova). Na primjer:

```
p {
font-family: Fenix, Garamond, serif;}
```

U slučaju da se žele koristiti verzije fonta bold ili italic, mogla bi se koristiti svojstva font-style i font-weight. Da bi **rezultat izgledao podjednako dobro u svim preglednicima**, bolji pristup je upotrijebiti zasebnu font datoteku sa verzijama fonta bold odnosno italic, koja se onda mora uključiti pomoću odvojenog pravila @font-face:

```
@font-face {
font-family: FenixBold; src: url('FenixBold.woff');
}
```

Postoji više tipova web fontova. Format **WOFF**[34] (engl. *Web Open Font Format*) standard je koji podržavaju najnoviji preglednici, ali da bi web fontovi radili za većinu korisnika, potrebno je font pripremiti u više različitih formata i sve ih navesti u pravilu @font-face. U slijedećoj tabeli su popisani mogući formati za web-fontove i njihova podržanost u različitim preglednicima:

Format	Preglednici koji podržavaju
.eot	IE 8
.ttf	Firefox 21, Chrome 21, Safari 5.1, Opera 12.1
.otf	Firefox 21, Chrome 21, Safari 5.1, Opera 12.1
.svg	Chrome 27, Safari 5.1, Opera 23, mobilni preglednici
.woff	IE 9, Firefox 21, Chrome 21, Safari 5.1, Opera 12.1 i mobilni preglednici

[34] developer.mozilla.org/en-US/docs/Web/Guide/WOFF

Problem licence, kao i problem sa velikim brojem potrebnih formata i pouzdanim CSS kodom koji se koristi trikovima da bi riješio probleme sa nekim preglednicima, rješavaju posebna web sjedišta namijenjena korištenju web fontova. Neka od tih web sjedišta nude besplatne fontove, a neka uz pretplatu prodaju licencu za njihovo korištenje na web-u. Prilikom preuzimanja fonta, na većini njih dobijemo font datoteke za sve poznatije preglednike, te gotov CSS kod koji radi u svim poznatijim preglednicima. Najpoznatija web sjedišta za tu namjenu su: fontsquirrel.com i google.com/fonts.

10.10. Google fontovi

Google Fontovi (ili **Google Web Fonts**) je biblioteka od 915 fontova sa slobodnom licencom korištenja. Katalog je pokrenut 2010., a obnovljen 2011. i 2016. Većina fontova je objavljena pod licencom SIL Open Font License 1.1[35], dok su neke objavljene pod licencom Apache License, obje su slobodne licence[36]. Dakle, datoteka Google Fonts je namijenjena za otkrivanje i istraživanje fontova, a usluga se opsežno koristi sa više od 17 triliona puta posluženih fontova, što znači da je svaki od 915 fontova preuzet 19 milijardi puta, dakle svaka osoba na Zemlji je u prosjeku, preuzela svaki font najmanje dva ili tri puta. Biblioteka se održava putem spremišta GitHub Google fontova na https://github.com/google/fonts, gdje se sve datoteke fontova mogu dobiti direktno, dakle izvorne datoteke za fontove su dostupne u Git repozitorijama.

10.10.1. Uključivanje Google fonta u HTML dokument

Da bi smo koristili Google font, najprije ćemo otvoriti internetsku adresu Google font sjedišta (google.com/fonts). Odabrat ćemo font, a nakon toga ćemo (opcionalno) odabrati „težinu“ fonta koju želimo koristiti. Pri odabiru težine moramo voditi računa da odaberemo one koje su nam zaista potrebne, jer svaka odabrana „težina“ usporava brzinu našeg web sjedišta. Da bi font uključili unutar head tag-ova unutar HTML dokumenta. Nakon što odaberemo font i željene „težine“ fonta, kopirat ćemo link tag sa stranice u naš HTML dokument. Na primjer:

```
<link href="https://fonts.googleapis.com/css?family=Heebo" rel="stylesheet"/>
```

STANDARD @IMPORT

```
<link href="https://fonts.googleapis.com/css?family=Heebo" rel="styles
heet">
```

[35] opensource.org/licenses/OFL-1.1
[36] Mogu se koristiti, bez povrede prava autora.

10.10.2. Uključivanje Google fonta u CSS dokument

Pored uključivanja fonta u HTML dokument pomoću tag-ova link, isti možemo uključiti i u CSS datoteku. Da bi to uradili odabrat ćemo opciju @IMPORT, umjesto STANDARD. Font ćemo uključiti na prvu liniju CSS dokumenta. Mana ovakvog uključivanja je da ćemo font moći koristiti samo unutar eksterne CSS datoteke. Na primjer:

```
<style>
    @import url('https://fonts.googleapis.com/css?family=Heebo');
</style>
```

STANDARD @IMPORT

```
<style>
@import url('https://fonts.googleapis.com/css?family=Heebo');
</style>
```

U poglavlju je obrađeno:

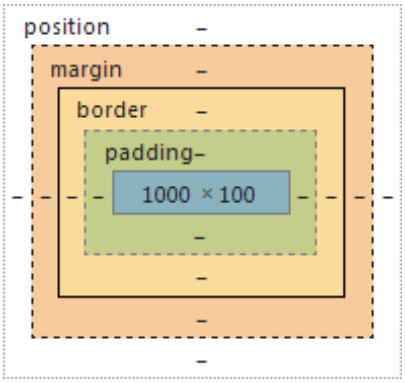
- boja sadržaja;
- vrste, veličina, „težina“, stil fonta;
- skraćeni način navođenja svojstava fonta;
- poravnanje i razmak sadržaja;
- velika i mala slova;
- ukrašavanje teksta;
- web fontovi;
- Google fontovi.

XI Box model

Po završetku ovog poglavlja, polaznik će moći:

- objasniti pojam Box modela;
- upotreba CSS box model svojstva.

Svi HTML elementi se mogu smatrati kao kutije (engl. *box*). U CSS-u, termin „model kutije“ se koristi kada govorimo o dizajnu i izgledu. CSS model kutije je u suštini kutija koja obuhvata svaki HTML element. Sastoji se od margina, okvira, ispune i stvarnog sadržaja. Naredna slika ilustruje model kutije:



Svaki box model se sastoji iz slijedećih dijelova:

1. **Sadržaja** (engl. *content*; označen plavom bojom) - prostor u kojem se nalazi neki sadržaj, tekst, fotografije, itd;
2. **Ispune** (engl. *padding*; označena svijetlo zelenom bojom) – prazan prostor oko sadržaja (unutrašnji dio elementa);
3. **Ivice** (engl. *border*; označena žutom bojom) - okvir ili granica koja se nalazi između padding-a i margine, a označava granicu elementa;
4. **Margine** (engl. *margin*; označena sa narandžastom bojom) – prazan, transparentan prostor izvan granica elementa.

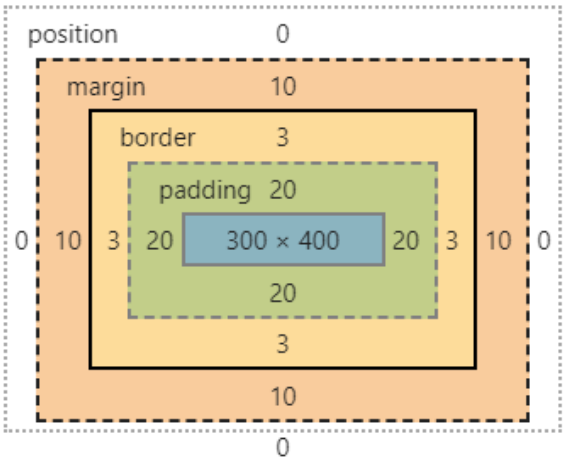
Model kutije nam omogućava dodavanje granice oko elemenata i definisanje prostora između elemenata. Model kutije je najbolje ilustrovati kratkim primjerima. Da bismo izračunali koliko nam je ukupno prostora potrebno za jedan element, računamo to na sljedeći način:

• Ukupna širina = lijeva margina + lijevi okvir + lijevi padding + širina elementa + desni padding + desni okvir + desna margina

• Ukupna visina = gornja margina + gornji okvir + gornji padding + visina elementa + donji padding + donji okvir + donja margina

Pogledajmo jedan HTML i CSS primjer sa pravilima koja sadrže deklaracije za sva svojstva box modela (širina, visina, ispuna, okvir, margina) definisana na jednoj klasi koju smo asocijativno nazvali .box-model:

```
.box-model{
    width: 300px;
    height: 400px;
    padding: 20px;
    border: 3px solid greenyellow;
    margin: 10px;
}
```



Dakle, ukupna širina je:

Širina = 10 + 3 + 20 + 300 + 20 + 3 + 10 = 366 px;

Visina = 10 + 3 + 20 + 400 + 20 + 3 + 10 = 466 px.

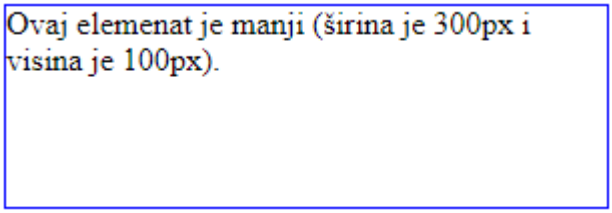
Iz ovog kratkog primjera vidimo da naš element zauzima najmanje 366px u širinu i 466px u visinu. Ako na stranici nema toliko mjesta, naš element će biti pomjeren ili će se „preliti“ izvan svog bloka (engl. *overflow*), čime će se stranica po zadanim postavkama povećati za tu veličinu.

11.1. CSS svojstvo Box modela

CSS svojstvo box-sizinig nam omogućava da u širinu sadržaja (width) uključimo, odnosno isključimo širinu/visinu ispune i ivice. Predefinisana vrijednost svojstva box-sizing je content-box. Dakle ako imamo element klase .content-box sa slijedećim svojstvima:

```
.content-box{
  width: 300px;
  height: 100px;
  border: 1px solid blue;
}
```

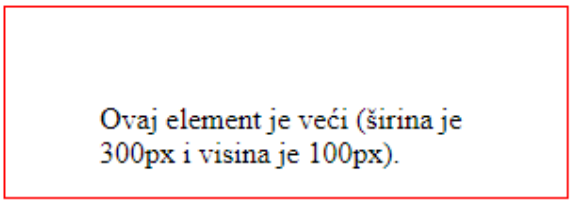
U svojstvima nismo naveli svojstvo box-sizing sa vrijednošću content-box (već smo rekli zašto).



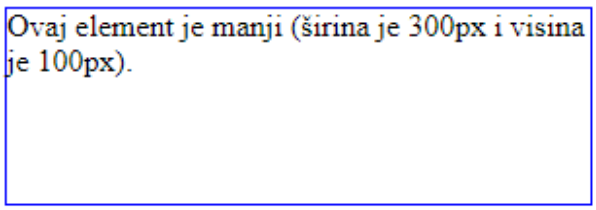
Ako sada napravimo novi element sa klasom .border-box sa slijedećim svojstvima:

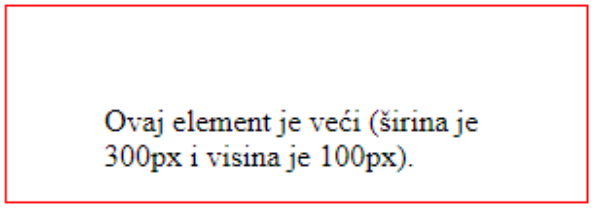
```
.border-box{
  width: 300px;
  height: 100px;
  padding: 50px;
  box-sizing: border-box;
  border: 1px solid red;
}
```

U svojstvima smo naveli svojstvo box-sizing sa vrijednošću border-box, a svojstvo će doprinijeti slijedećem:



Ako uporedimo veličine dva elementa, dolazimo do zaključka:





Veličina je ista, iako element sa klasom .border-box pored širine ima i ispunu. Zašto? Naime, postavljanjem svojstva i vrijednosti `box-sizing: border-box`, u širinu elementa se uračunava ispuna i „debljina“ ivice. Dakle, u našem slučaju širina je 300 piksela, a ispuna od 50 piksela sa lijeve i desne će ući u okvir 300 piksela širine, na uštrb širine sadržaja. U prvom slučaju za širinu

sadržaja je bilo dostupno 300 piksela, dok u drugom slučaju je dostupno 200 piksela (300px – 50px – 50px). Ovakav princip nam omogućava da jednostavno kontrolišemo model kutije. Da sumiramo, kada koristimo svojstvo `box-sizing: border-box`, ukupnu širinu računamo na slijedeći način:

Ukupna širina = širina elementa (ivica + ispuna + širina sadržaja).

A kod svojstva `box-sizing: border-box`, nešto drugačije:

Ukupna širina =širina elementa (širina sadržaja) + ivica + ispuna.

U poglavlju je obrađeno:

- pojam Box modela;
- CSS box model svojstvo.

XII Semantički elementi

Po završetku ovog poglavlja, polaznik će moći:

- objasniti čemu služe semantički elementi;
- objasniti upotrebu div i span elemenata, te kako se razlikuju od ostalih semantičkih elemenata;
- napisati osnovnu semantičku strukturu HTML stranice.

Kako je ranije navedeno, HTML dokument ima određenu strukturu. HTML dokument tako može imati naslove, odlomke, slike, tablice i sl. Osim ovoga, HTML dokument može imati i elemente koji definišu strukturu same web stranice kao što su navigacija, zaglavlje, podnožje, članak,... Ovi elementi se nazivaju **semantičkim** (ili strukturalnim) **elementima**. Semantički elementi su novi HTML elementi koji su se pojavili sa verzijom HTML5 i služe prvenstveno za slaganje osnovnih elemenata stranice u određenu strukturu, tj. slaganje rasporeda (engl. *layout*) stranice. Ranije, dok CSS još nije bio dovoljno razvijen, **struktura HTML stranice se izrađivala na način da se sadržaj stranice smještao u tabele koje nisu imale vidljivu ivicu** (engl. *border*).

Korištenjem colspan i rowspan atributa se mogla izraditi bilo kakva željena struktura tabele, te bi sadržaj umetnut u nju zadržavao željenu strukturu. Ovo je razvojem CSS-a izgubilo svrhu, te se danas smatra izrazito lošim načinom izrade web stranica. Dugo nakon toga, autori web stranica su koristili div elemente kako bi grupisali elemente na stranici. Sa dolaskom HTML5 standarda je uveden novi niz elemenata koji omogućuju podjelu sadržaja i dijelova stranice. Već sama imena tih elemenata na neki način pojašnjavaju čemu koji element služi. Smisao novih elemenata je i opisivanje strukture stranice koje je korisno i za prilagodbu web stranica za osobe sa invaliditetom sa obzirom da će čitači na taj način lakše omogućiti preskakanje zaglavlja prilikom pregledavanja web stranica, tj. skakanje direktno na sam sadržaj i sl. Osobi koja koristi čitač to znatno skraćuje vrijeme pronalaska važnog sadržaja. Također, korisni su i za bolju pronalazljivost stranica od strane tražilica koje će lakše pronaći relevantan sadržaj preskačući na primjer zaglavlje ili podnožje stranice i pregledavajući sadržaj u člancima (element article).

12.1. Semantički elementi za grupaciju sadržaja – i

12.1.1. Semantički element

HTML element div je blok element koji omogućuje grupisanje više elemenata unutar jednog bloka. Moguće je napraviti jedan div element u kojem ćemo postavljati više elemenata koji logički spadaju na isto mjesto, npr. komentari korisnika, slike,... Zatim se na tom div elementu

može primijeniti tačno određeni CSS koji će omogućiti da svi elementi unutar toga div elementa izgledaju jednako. To bi značilo da će npr. svi komentari će korisniku izgledati kao dio sekcije za komentare ili da će sve slike izgledati kao galerija. Bez implementacije CSS-a, sadržaj div elementa neće biti prezentovan na neki poseban način, osim što će obavezno započeti u novom redu, sa obzirom da je riječ o blok elementu. Element div se također koristi i za grupisanje drugih semantičkih elemenata na stranici. Na primjer:

```
<div class="main">
    <div class="featured-article">
        <h2>Raspored kurseva u avgustu</h2>
        <p>Objavljen je raspored kurseva u avgustu. Prijave na kurseve su u
        toku i trajat će do popune slobodnih mjesta.</p>
        <p>Raspored kurseva u avgustu možete pogledati
        <a href="kursevi-raspored.html">ovdje</a>.</p>
    </div>
    <div class="article">
        <h2><em>Novi</em>kurs</h2>
        <p>Putem on-line platforme su dostupni materijali za
        kurseve<strong>Prezentacije (PowerPoint 2019)</strong>.</p>
    </div>
</div>
```

12.1.2. Semantički element

HTML element span je generički inline element koji omogućava označavanje dijela teksta na koji se želi primijeniti specifično CSS svojstvo, tj. označavanje dijela sadržaja u slučaju da nijedan drugi semantički element nije moguće iskoristiti, te za grupisanje drugih inline elemenata. Ako se elementom span označi neki tekst, bez primjene nekog CSS koda, taj tekst će se prikazati kao i ostali tekst na stranici. Na primjer:

```
<p><span class="telefon">telefon:</span> +387 33 783 055</p>
```

12.2. Zaglavlje i podnožje stranice – <header> i <footer>

Elementi koji se koriste za definisanje zaglavlja (header) i podnožja (footer) se mogu koristiti za:

- definisanje zaglavlja i podnožja za cijelu web stranicu,
- definisanje zaglavlja i podnožja nekog članka (article) ili sekcije (section).

12.2.1. Zaglavlje <header>

U navedenim elementima mogu se grupisati svi ostali elementi koji bi bili dio zaglavlja ili podnožja neke stranice, npr. ime web-stranice, navigacija, informacije o autorskim pravima i

slično. U slučaju da se zaglavlje koristi u članku, u njemu može biti naslov članka ili nešto slično. Zaglavlje može izgledati kao u primjeru u nastavku:

```
<header>
    <img src="img/logo.png" alt="Uvod u HTML i CSS" />
    <p>Uvod u HTML i CSS</p>
    <div class="nav">
        <a href="index.html" class="selected">HOME</a>
        <a href="about.html">O nama</a>
        <a href="courses.html">Kursevi</a>
        <a href="schedule.html">Raspored kurseva</a>
        <a href="sign.html">Prijavi kurs</a>
    </div>
</header>
```

12.2.2. Podnožje <footer>

A podnožje na primjer:

```
<footer>
    <a href="#">O nama</a>
    <a href="#top">Nazad na vrh</a>
    <a href="#">Statut</a>
</footer>
```

12.3. Navigacija – element <nav>

Element nav se koristi za „držanje“ navigacije na jednom mjestu, na primjer glavnu navigaciju dobro je smjestiti unutar elementa nav. Na primjer:

```
<nav>
    <a href="index.html" class="selected">HOME</a>
    <a href="about.html">O nama</a>
    <a href="courses.html">Kursevi</a>
    <a href="schedule.html">Raspored kurseva</a>
    <a href="sign.html">Prijavi kurs</a>
</nav>
```

12.4. Sadržaj – elementi <article> i <aside>

12.4.1. Element <article>

Element article se koristi kao spremište za bilo koji dio stranice koji može biti samostalan ili se može preuzeti i distribuirati kao cjelina. To može biti članak, samostalni tekst, forum, galerija... Unutar elementa article se smješta jedan samostalni dio sadržaja. Ako je riječ o komentarima, oni se mogu smjestiti unutar elementa article koji može biti ugniježđen unutar drugog article elementa u kojem se nalazi članak. Na primjer:

```
<article>
    <header>
        <h1 id="top">Uvod u HTML i CSS</h1>
    </header>
    <figure>
        <img src="img/course.png" alt="HTML i CSS logo" title="Polaznici"/>
        <br/>
        <figcaption>Polaznici na kursu</figcaption>
    </figure>
    <p>Uvod u HTML i CSS kurs je namijenjeni svima zainteresovanim za
    stjecanje osnovnih znanja o upotrebi informacijsko-komunikacijskih
    tehnologija; kurs se održava u učionicama (prema rasporedu
    održavanja)</p>
</article>
```

12.4.2. Element <aside>

Element <aside> se može koristiti na dva načina, zavisno od toga gdje je smješten:

- ako je smješten unutar elementa <article> trebao bi sadržavati informacije koje su vezane isključivo uz sadržaj u tom elementu, ali ne i informacije koje se odnose na cjelokupan sadržaj stranice, npr. rječnik vezan uz članak;
- ako je smješten van elementa <article> onda može služiti kao spremnik za informacije koje se tiču cijele stranice, na primjer može sadržavati linkove na druge dijelove stranice, okvire za umetanje vezanih informacija sa društvenih mreža, popis novih članaka i sl.

Na primjer:

```
<article>
    <header>
        <h1 id="top">Uvod u HTML i CSS</h1>
    </header>
    <figure>
        <img src="img/course.png" alt="HTML i CSS logo" title="Polaznici"/>
        <br/>
        <figcaption>Polaznici na kursu</figcaption>
    </figure>
    <p>Uvod u HTML i CSS kurs je namijenjeni svima zainteresovanim za
    stjecanje osnovnih znanja o upotrebi informacijsko-komunikacijskih
    tehnologija; kurs se održava u učionicama (prema rasporedu
    održavanja)</p>
    <aside>
        <a href="#">Raspored održavanja kurseva</a>
        <a href="#">Prijavi kurs</a>
        <a href="#">DL platforma</a>
    </aside></article>
```

12.5. Sekcija – element <section>

Element section omogućava grupisanje povezanog sadržaja u sekcije. Uobičajeno je da svaka sekcija ima vlastito zaglavlje. Na jednoj web stranici može biti više sekcija. Svaka sekcija može imati nekoliko article elemenata koji su međusobno logički povezani, tj. imaju zajedničku temu ili svrhu. Trebalo bi se izbjegavati smještanje cjelokupnog sadržaja web stranice unutar jednog section elementa. Za to je bolje koristiti element div. Na primjer:

```
<section>
    <header>
        <h1 id="vrh">Lista kurseva:</h1>
    </header>
    <ul>
        <li><a href="#html-i-css">Uvod u HTML i CSS</a></li>
        <li><a href="#java">Uvod u programski jezik Java</a></li>
        <li><a href="#PHP-i-SQL">Uvod u PHP i MySQL</a></li>
    </ul>
</section>
<section>
    <article>
        <header>
            <h2 id="html-i-css">Uvod u HTML i CSS</h2>
        </header>
        <p>Šifra kursa: WD-253</p>
        <p>Sintaksa HTML-a i CSS-a, liste, semantički elementi, flex-box,
        grid sistem,...</p>
    </article>
    <article>
        <header>
            <h2 id="java">Uvod u programski jezik Java</h2>
        </header>
        <p>OOP-148</p>
        <p>Osnove jezika Java, uslovi, petlje, klase i objekti,
        upravljanje greškama, moduli, testiranje programskog koda,...</p>
    </article>
    <article>
        <header>
            <h2 id="PHP-i-SQL">Uvod u PHP i MySQL</h2>
        </header>
            <p>WD-523</p>
            <p>Model klijent-poslužitelj, varijable, operatori, uslovne
            strukture, polja, petlje, funkcije, ugrađene funkcije PHP-a,
            autentikacija korisnika,...</p>
    </article>
</section>
<p>Povratak na <a href="#top">Idi na vrh</a>.</p>
```

U poglavlju je obrađeno:

- čemu služe semantički elementi;
- upotreba div i span elemenata, te kako se razlikuju od ostalih semantičkih elemenata;
- pisanje osnovne semantičke strukture HTML stranice

XIII Oblikovanje elemenata

Po završetku ovog poglavlja, polaznik će moći:

- mijenjati boju pozadine;
- mijenjati prikaz elementa;
- postavljati širinu i visinu elementa;
- mijenjati ispunu, margine i ivice elementa;
- izračunati dimenzije elementa;
- resetovati predefinisane vrijednosti;
- postavljati pozadinsku sliku;
- postavljati pozadinski gradijent (prijelaz);
- postavljati sjenu na elemente.

Svi HTML elementi su zapravo pravougli, odnosno zauzimaju pravougaonu površinu unutar ekrana preglednika. U ovom poglavlju ćemo objasnit kako se tim pravougaonicima postavljaju boja, veličina, sjena i druga vizualna svojstva.

13.1. Boja pozadine

Da bi HTML elementi bili vidljivi, može im se postaviti boja (boja pozadine). Za postavljanje boje pozadine koristi se svojstvo background-color. Kao i kod svojstva color, vrijednosti se mogu postaviti na više načina. Može se koristiti naziv boje:

```
background-color: red;
```

Najčešće je korišteni način postavljanja vrijednosti je heksadecimalni kod:

```
background-color: #a0252a;
```

Umjesto heksadecimalnog zapisa mogu se koristiti i RGB vrijednosti:

```
background-color: rgb(15, 88, 32);
```

Ako je potrebno, moguće je postaviti i prozirnost i to tako da se navede vrijednost komponente a (kao decimalni broj između 0 i 1):

```
background-color: rgba(20, 1, 3, 0.5);
```

Također se za odabir boje može koristiti i HSL model:

```
background-color: hsl(17, 25%, 6%);
```

U njemu se također može postaviti prozirnost:

```
background-color: hsl(77, 21%, 86%, 0.5);
```

13.2. Prikaz elemenata

Prema načinu prikaza u HTML-u postoje dvije osnovne vrste elemenata. Prva vrsta su **blok elementi** (engl. *block elements*). Oni zauzimaju cijeli red u kojem se nalaze. Moguće im je odrediti visinu i širinu. Čak i ako im se smanji širina, oni će i dalje zauzimati cijeli red tako da sljedeći element mora započeti u novom redu. Primjeri takvih elemenata su: div, p i h1.

Druga vrsta su **linijski elementi** (engl. *inline elements*). Elementi su prikazani unutar trenutne linije teksta. Oni ne zauzimaju cijeli red, nego onoliko prostora koliko zauzima njihov sadržaj (**njihova širina i visina se ne mogu postaviti**). Primjeri takvih elemenata su span, em i strong.

Pomoću CSS svojstva display je moguće promijeniti tip prikaza na elementu. Postavi li se blok elementima svojstvo na inline, oni će se ponašati kao da su linijski elementi:

```
div, p, h1{
    display: inline;
}
```

div p h1

Postavi li se linijskim elementima vrijednost svojstva display na block, oni će se ponašati kao da su blok elementi:

```
span, em, strong{
    display: block;
}
```

span

em

strong

Mijenjanje tipa prikaza na pojedinom elementu je ponekad potrebno, ako se neki element želi prikazati drugačije. Zanimljiva vrijednost na koju se može postaviti svojstvo display je inline-block, čime se mijenja ponašanje blok elementa tako da on više ne zauzima cijeli red. Takav

element zadržava druga svojstva blok elementa (**može se definisati visina i širina, što je praktično za pozicioniranje elemenata**). Osim blok elemenata i linijskih elemenata, u HTML-u postoje i neki elementi koji prema tipu prikaza zapravo ne pripadaju ni jednoj od pomenute dvije grupe. To su npr. stavke liste (elementi li), te tabele i elementi tablele. U tabeli u nastavku su objašnjene vrijednosti koje se mogu dodijeliti svojstvu display:

Predefinisana riječ	Objašnjenje
none	element neće biti vidljiv
inline	element se prikazuje kao linijski element
block	element se prikazuje kao blok element
inline-block	element zadržava svojstva blok (širina i visina) i linijskog elementa
list-item	element se prikazuje kao element li
table	element se prikazuje kao element tabele (slično blok elementu)
inline-table	element se prikazuje kao tabela (obilježja linijskog elementa)
table-caption	element se prikazuje kao element caption
table-cell	element se prikazuje kao element td
table-column	element se prikazuje element sa svojstvom col
table-column-group	element se prikazuje element sa svojstvom colgroup
table-footer-group	element se prikazuje kao element tfoot
table-header-group	element se prikazuje kao element thead
table-row	element se prikazuje kao element tr
table-row-group	element se prikazuje kao element tbody
flex	element se prikazuje u načinu pozicioniranja flexbox (blok element)
inline-flex	element se prikazuje u načinu pozicioniranja flexbox (linijski element)
grid	element se prikazuje u načinu pozicioniranja grid (blok element)
inline-grid	element se prikazuje u načinu pozicioniranja grid (linijski element)
inherit	element koristi predefinisana svojstva
initial	element nasljeđuje vrijednost od roditeljskog elementa

13.3. Širina i visina

Ako se širina i visina HTML elementa ne postave eksplicitno, njegova širina i visina će zavisiti ili od sadržaja koji se nalazi unutar njega ili od roditeljskog elementa unutar kojeg se nalaze.

Širina se može podešavati samo kod block elemenata (i elemenata inline-block). Ako im se ne postavi određena širina, blok elementi će zauzeti svu širinu roditeljskog elementa. Sa druge strane, širina linijskog elementa je određena njegovim sadržajem. Postavljanje širine na tim elementima neće imati nikakav učinak. Blok elementi inicijalno zauzimaju svu širinu roditeljskog elementa. Za definiranje širine elementa koristi se svojstvo width. Vrijednost tog svojstva

postavlja se u jednoj od mjernih jedinica dužine (pikseli, tačke, jedinice em ili rem) , može i u postotcima. Na primjer:

```
width: 400px;
```

Svojstvo width se može postaviti i na vrijednost auto, što je inicijalna vrijednost tog svojstva. Ta ključna riječ predstavlja vrijednost koju preglednik dodjeljuje elementu (kod blok elemenata je širina roditeljskog elementa). Ključna riječ auto se eksplicitno navodi samo u slučajevima kad se želi zaobići neko drugo CSS pravilo koje je postavilo širinu na neku vrijednost. Ako se vrijednost širine navede u postotcima, širina se računa kao postotak širine roditeljskog elementa. Postavljanjem širine elemenata u postotcima, se može postići da se ona automatski prilagođava količini raspoloživog prostora. Na primjer:

```
width: 50%;
```

Smanjuje li se prozor preglednika, moguće je primijetiti da se i širina ovako definisanog elementa smanjuje. Često se takvo smanjivanje širine elementa želi ograničiti. Pomoću svojstva min-width je moguće definisati **minimalnu širinu** koju element mora zauzimati. Vrijednost minimalne širine se također može postaviti u mjernim jedinicama za dužinu ili u postotcima. Inicijalna vrijednost za ovo svojstvo je 0. Na primjer:

```
min-width: 200px;
```

Osim minimalne širine, elementu se može definisati i maksimalna širina koju smije zauzeti. **Maksimalnu širinu** postavljamo pomoću svojstva max-width. Vrijednost maksimalne širine se također može postaviti u mjernim jedinicama za dužinu ili u postotcima. Inicijalna vrijednost za ovo svojstvo je none, tj. maksimalna širina nije postavljena. Na primjer:

```
max-width: 400px;
```

Visinu možemo podešavati samo kod block elemenata (i kod elemenata inline-block). Visina se postavlja pomoću svojstva height. Vrijednost za svojstvo se postavlja u mjernim jedinicama za dužinu ili u postotcima. Na primjer:

```
height: 200px;
```

Ako visina elementa nije postavljena, imat će vrijednost auto, što znači da će je preglednik sam izračunati na temelju visine sadržaja koji se nalazi unutar elementa (npr. teksta ili slike). Ako je vrijednost visine postavljena u postotcima, onda će visina biti izračunata kao postotak visine

roditeljskog elementa. Dakle, tako se može postići da se visina prilagođava količini raspoloživog prostora. Na primjer:

```
height: 50%;
```

Minimalnu visinu elementa je moguće, kao i minimalnu širinu, postaviti kada se želi ograničiti smanjivanje visine elementa zajedno sa količinom raspoloživog prostora. Minimalna visina se najčešće koristi kada postoji dinamički sadržaj čija visina može varirati. U takvom slučaju se ne želi ograničiti visina elementa, već se želi da se ona automatski prilagođava visini sadržaja. Najčešće je poželjno da element ima neku minimalnu visinu u slučaju kad korisnik unese malu količinu teksta. Tada se neće postaviti svojstvo height na elementu (imat će vrijednost auto), već će se postaviti samo minimalna visina pomoću svojstva min-height. Vrijednost minimalne visine se može postaviti u mjernim jedinicama za dužinu ili u postotcima. Inicijalna vrijednost za to svojstvo je 0, a od ključnih se riječi može koristiti samo ključna riječ inherit. Na primjer:

```
min-height: 100px;
```

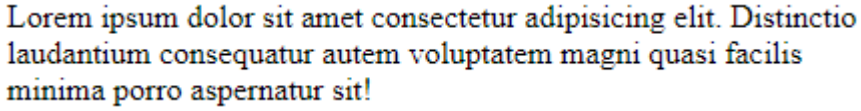
I kod visine se elementu može definirati maksimum koji smije zauzeti. **Maksimalna visina** se postavlja pomoću svojstva max-height:

```
max-height: 500px;
```

Vrijednost maksimalne visine se može također postaviti u mjernim jedinicama za dužinu ili u postotcima. Inicijalna vrijednost za to svojstvo je none, tj. maksimalna širina nije postavljena. Svojstva min-width, max-width, min-height i max-height nisu podržana u IE 6.

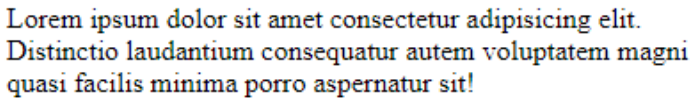
13.4. Ispuna (engl. *padding*)

Iz estetskih razloga je potreban određeni razmak između ivice elementa i teksta (ili drugog sadržaja) koji se nalazi u elementu. Taj razmak se postavlja korištenjem svojstva padding. **Ispuna** se može postaviti i blok elementima i linijskim elementima. Inicijalno, ispuna na elementu nije postavljena (njena vrijednost je 0):



Dakle, svojstvu padding se predaje vrijednost sa mjernom jedinicom za dužinu ili vrijednost u postotcima (u slučaju korištenja postotka veličina padding-a se računa na temelju širine

roditeljskog elementa). Postavi li se preveliki gornji padding kod linijskog (engl. *inline*) elementa, to će uzrokovati pomicanje teksta izvan granica elementa. Na primjer:



Vrijednost svojstva padding se **ne može naslijediti sa roditeljskog elementa**. Ako se svojstvu padding navede samo jedna vrijednost; gornji, desni, donji i lijevi padding će dobit istu vrijednost. Ako je potrebno postaviti različite veličine padding-a na ta četiri mjesta, navedu se četiri različite vrijednosti. Vrijednosti se navode u smjeru kazaljke na satu. Prva je vrijednost za gornji položaj, zatim desni pa donji i na kraju lijevi. Na primjer:

```
padding: 7px 12px 10px 20px;
```

Mogu se navesti i tri vrijednosti. U tom slučaju se prva vrijednost opet odnosi na gornji položaj, druga na desni, a treća na donji. Vrijednost koja nedostaje (lijeva) će biti jednaka naspramnoj vrijednosti (desnoj):

```
padding: 7px 12px 10px;
```

Ako se navedu samo dvije vrijednosti, prva vrijednost se odnosi na gornji položaj, a druga na desni. Donja vrijednost će biti jednaka gornjoj, a lijeva će biti jednaka desnoj:

```
padding: 12px 10px;
```

Čini se potpuno nemogućim zapamtiti na koji se položaj odnosi koja vrijednost, ali nam ovdje pomaže mnemotehnička skraćenica **TRBL** (koju pamtimo pomoću engleske riječi *trouble*). Slova TRBL predstavljaju pojedine položaje i pomažu da se zapamti njihov redoslijed. Kada imamo četiri vrijednosti, prva vrijednost se odnosi na gornji položaj (engl. *top*), druga na desni (engl. *right*), treća na donji (engl. *bottom*), a četvrta na lijevi (engl. *left*). Zapravo, kreće se u smjeru kazaljke na satu, a započinje se sa gornjim položajem. U slučaju kad postoje tri vrijednosti, prva se ponovo odnosi na gornji položaj, druga na desni, treća na donji. Četvrta vrijednost ne postoji, ako vrijednost za lijevi položaj nije navedena, on će imati jednaku vrijednost kao i desni. U slučaju kad postoje samo dvije vrijednosti, prva se odnosi na gornji položaj, a druga na desni. Vrijednosti za bottom i left nisu navedene; lijevi položaj imati istu vrijednost kao i desni, a donji istu kao i gornji. Svojstvo padding zapravo je skraćeni način postavljanja vrijednosti za četiri svojstva koja postoje za svaki položaj: padding-top, padding-right, padding-bottom i padding-

left. Primjer u kojem su postavljene četiri vrijednosti svojstvu padding bi se mogao napisati i na slijedeći način:

```
padding-top: 7px;
padding-right: 12px;
padding-bottom: 10px;
padding-left: 20px;
```

Ako se u ovakvom načinu pisanja izostave neki od položaja, oni neće imati vrijednost 0:

```
padding-top: 7px;
padding-right: 12px;
```

U slijedećem primjeru je praktična primjena ispune:

```
<head>
    <style>
        p{
            background-color: lightblue;
        }
        .padding{
            padding-top: 25px;
            padding-bottom: 25px;
            padding-right: 50px;
            padding-left: 50px;
        }
    </style>
</head>
<body>
    <p>Ovo je padding bez paragrafa.</p>
    <p class="padding">Ovo je paragraf sa padding-om primijenjenim
    preko klase.
</body>
```

Ovo je padding bez paragrafa.

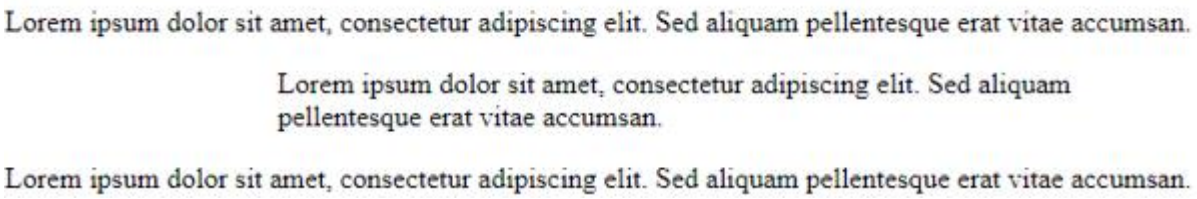
Ovo je paragraf sa padding-om primijenjenim preko klase.

13.5. Margina

Između HTML elemenata je moguće definisati razmak, odnosno marginu. Neki elementi (npr. elementi p i h1) imaju inicijalno postavljenu marginu, dok je drugi (elementi div) nemaju. Margina se može postavljati i blok elementima i linijskim elementima, s tim da je kod linijskih elemenata moguće postaviti samo lijevu i desnu marginu, dok postavljanje gornje i donje margine nema učinka. Margina predstavlja vrijednost koja određuje razmak između elemenata

u HTML dokumentu. Kada određenom elementu odredimo marginu, mi povećavamo prostor koji taj element zauzima, tj. sa vanjske strane elementa (izvan engl. *outline* dijela) dodajemo nevidljivi prostor granica susjednih HTML elemenata. **Elementi bez margina se dodiruju**. Svaki element ima četiri strane, a margine možemo urediti zajedno ili zasebno: gornju, desnu, donju i lijevu. U narednom primjeru smo postavili lijevu marginu drugog paragrafa na 4 cm:

```
<head>
    <style>
        .ml-4{
            margin-left: 4cm;
        }
    </style>
</head>
<body>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    Sed aliquam pellentesque erat vitae accumsan.</p>
    <p class="ml-4">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    Sed aliquam pellentesque erat vitae accumsan.</p>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    Sed aliquam pellentesque erat vitae accumsan.</p>
</body>
```



Ako želimo da margina bude ista sa gornje, desne, donje i lijeve strane:

```
margin: 10px;
```

Oko svakog elementa sad postoji razmak od 10 piksela. Postavljanjem margine je moguće koristiti i ključnu riječ auto, koja znači da će preglednik sam dodijeliti vrijednost margine. Za gornju, desnu, donju i lijevu marginu moguće je navesti različite vrijednosti:

```
margin: 7px 5px 15px 12px;
```

Ovako navedene vrijednosti se odnose na gornju, desnu, donju i lijevu marginu (dakle, u smjeru kazaljke na satu, počevši od gornje margine). Ako se navedu samo tri vrijednosti, prva vrijednost će se odnositi na gornju marginu, druga na desnu, treća na donju, a lijeva će margina biti jednaka desnoj:

```
margin: 7px 5px 20px;
```

Ako se navedu samo dvije vrijednosti, prva vrijednost će se odnositi na gornju marginu, a druga na desnu. Donja margina bit će jednaka gornjoj, a lijeva će biti jednaka desnoj:

```
margin: 7px 20px;
```

Gornja i donja margina susjednih elemenata se spajaju. U primjeru su se spojile donja margina prvog elementa i gornja margina drugog elementa. Razmak između prvog i drugog elementa nije 27px (20px + 7px) nego samo 20px, tj. ostaje prisutan samo razmak od veće margine. Kod lijeve i desne margine to nije slučaj. Kad bi postojao susjedni desni element, susjedne vertikalne margine se bi se spojile, već bi ukupni razmak bio sabirak lijeve i desne margine. Postoje i svojstva pomoću kojih se odvojeno mogu postaviti veličine pojedinih margina. Ta svojstva su: margin-top, margin-right, margin bottom i margin-left. Primjer u kojem su odvojeno postavljene četiri različite vrijednosti izgledao bi ovako:

```
margin-top: 7px;
margin-right: 5px;
margin-bottom: 15px;
margin-left: 12px;
```

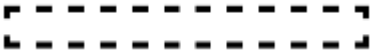
Ako se izostavi neka od gornjih deklaracija, ta margina neće biti postavljena, nego će imati svoju inicijalnu vrijednost (koja će, zavisno od tipa elementa, biti 0 ili će imati neku vrijednost).

13.6. Ivice

Na HTML-elementima se može pomoću CSS-a iscrtati ivica. U tu svrhu se koristi više svojstava CSS-a sa prefiksom border-. Izgled, odnosno stil ivice se definiše pomoću svojstva border-style. Kao vrijednost za to svojstvo navodi se ključna riječ. Najčešće se koristi ključna riječ solid, koja postavlja izgled ivice na punu liniju:

```
border-style: solid;
```

U tabeli u nastavku su ispisane sve moguće predefinisane riječi koje oblikuju dizajn ivice, te pripadajući primjeri:

Predefinisana riječ	Objašnjenje
dotted	
dashed	
solid	
double	
groove	

ridge	
inset	
outset	
hidden	rub se ne vidi (ali se njegova širina računa u dimenzije elementa)
none	bez ivice (inicijalna vrijednost)

Ako ovo svojstvo nije postavljeno, njegova inicijalna vrijednost je none. Dakle, HTML elementi inicijalno nemaju ivicu, dok im se ne postavi vrijednost tog svojstva na neku od ključnih riječi. Vrijednost ovog svojstva se ne nasljeđuje od roditeljskog elementa. Ako se svojstvu border-style postavi samo jedna vrijednost, sve ivice će izgledati jednako. Ako se na primjer postave dvije vrijednosti, prva će se odnositi na gornji i donji, a druga na lijevi i desni rub:

```
border-style: dotted solid;
```

U primjeru je vidljivo da su gornji i donji rub istočkani, dok su desne i lijeve ivice prikazani punom linijom. Ako se ovom svojstvu predaju tri vrijednosti, prva vrijednost će se odnosit na gornji, druga na desni i lijevi, a treća na donju ivicu:

```
border-style: dotted solid dashed;
```

Ako se predaju četiri vrijednosti, za svaku od ivica će se specificirati različita vrijednost. Prva vrijednost će se odnosit na gornji, druga na desni, treća na donji, a četvrta na lijevu ivicu:

```
border-style: dotted solid dashed double;
```

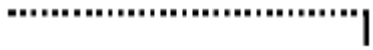
Svojstvo border-style zapravo je skraćeni način postavljanja vrijednosti za četiri odvojena svojstva koja postoje za svaki pojedinačnu ivicu: border-top-style, border-right-style, border-bottom-style i border-left-style. Posljednji primjer, u kojem su postavljene različite vrijednosti na svake od četiri ivice, mogao bi se napisati i ovako:

```
border-top-style: dotted;
border-right-style: solid;
border-bottom-style: dashed;
border-left-style: double;
```

Ako se u ovakvom načinu pisanja izostave neke od vrijednosti, te ivice neće imati postavljenu vrijednost (tj. imat će inicijalnu vrijednost none).

```
border-top-style: dotted;
border-right-style: solid;
```

Kad bi bile navedene samo dvije gornje deklaracije, dobio bi se ovakav rezultat:



Širina ivice se postavlja pomoću svojstva border-width. Kao vrijednosti se može predati ključna riječ ili vrijednost u nekoj od jedinica dužine (pikseli, jedinice em i rem, tačke). Širina ivice se ne može navesti pomoću postotaka. Ovako se postavlja širina ivice pomoću ključne riječi:

```
border-width: thick;
```

Postavljanjem samo ovog svojstva ivica elementa neće biti vidljiva (mora se postaviti i svojstvo border-style). Predefinisane riječi koje se mogu koristiti su date u tabeli u nastavku:

Predefinisana riječ	Objašnjenje
thin	
medium	
thick	

Ako vrijednost nije postavljena, inicijalna vrijednost ovog svojstva je medium. Vrijednost ovog svojstva se ne nasljeđuje od roditeljskog elementa. Širina koja će biti prikazana za ključne riječi thin, medium i thick u praksi može varirati od preglednika do preglednika, pa je za najbolje rezultate bolje postaviti širinu ivice pomoću CSS jedinica dužine. Na primjer:

```
border-width: 2px;
```

I kod ovog svojstva se mogu zadati različite vrijednosti na pojedinim ivicama. Ako se postave dvije vrijednosti, prva vrijednost će se odnositi na gornju i donju, a druga na desnu i lijevu ivicu. Na primjer:

```
border-width: 1px 4px;
```

Ako se postave tri vrijednosti, prva vrijednost će se odnositi na gornju, druga na desnu i lijevu, a treća na donji ivicu. Na primjer:

```
border-width: 1px 4px 7px;
```

Ako se postave četiri vrijednosti, prva će se vrijednost odnositi na gornju, druga na desnu, treća na donju, a četvrta na lijevu ivicu. Na primjer:

```
border-width: 1px 4px 7px 10px;
```

I svojstvo border-width zapravo je skraćeni način postavljanja vrijednosti za četiri odvojena svojstva za svaku pojedinačnu ivicu: border-top-width, border-right-width, border-bottom-width i border-left-width. Svaka od četiri ivice bi se mogle napisati i ovako:

```
border-top-width: 1px;
border-right-width: 4px;
border-bottom-width: 7px;
border-left-width: 10px;
```

Ako se u ovakvom načinu pisanja izostave neke od vrijednosti, te ivice neće imati postavljenu vrijednost (tj. imat će inicijalnu vrijednost medium pa će zbog toga ipak biti vidljivi). Na primjer:

```
border-top-width: 1px;
border-right-width: 4px;
```

Boja ivice se postavlja pomoću svojstva border-color. Kao i kod svojstava color i background-color, vrijednost se može postaviti pomoću naziva boje, heksadecimalnog ili RGBa i HSLa koda. Na primjer:

```
border-color: lightgray;
```


Postavljanjem samo ovog svojstva ivica elementa neće biti vidljiva (mora se postaviti i svojstvo border-style). I ovdje se mogu zadati različite vrijednosti na pojedinim ivicama. Ako se postave dvije vrijednosti, prva vrijednost će se odnositi na gornji i donji, a druga na desnu i lijevu ivicu. Na primjer:

```
border-color: lightgray black;
```

Ako se postave tri vrijednosti, prva vrijednost će se odnositi na gornju, druga na desnu i lijevu, a treća na donju ivicu:

```
border-color: lightgray black gray;
```

Ako se postave četiri vrijednosti, prva vrijednost će se odnositi na gornju, druga na desnu, treća na donju, a četvrta na lijevu ivicu:

```
border-color: lightgray black gray blue;
```

Svojstvo border-color je skraćeni način postavljanja vrijednosti za četiri odvojena svojstva koja postoje za svaki pojedinačni rub: border-top-color, border-right-color, border-bottom-color i border-left-color. Posljednji primjer, u kojem se postavljaju različite vrijednosti na svaku od četiri ivice, mogao bi se napisati i ovako:

```
border-top-color: lightgray;
border-right-color: black;
border-bottom-color: gray;
border-left-color: blue;
```

Ako se u ovakvom načinu pisanja izostave neke od vrijednosti, te ivice neće imati postavljenu vrijednost (tj. imat će inicijalnu vrijednost, a to je crna boja). Na primjer:

```
border-top-color: lightgray;
border-bottom-color: gray;
```

U praksi se rjeđe koriste odvojena svojstva za stil, širinu i boju, nego se najčešće koristi zbirno svojstvo border. Kad je neki od ivica različit od drugih, koriste se svojstva: border-top, border-right, border-bottom i border-left. Osim već spomenutih skraćenih načina pisanja, može se koristiti i svojstvo border pomoću kojeg se istovremeno mogu postaviti stil, širina i boja ivice. Na primjer:

```
border: 1px dashed gray;
```

Redoslijed navođenja vrijednosti nije važan, pa bi ovaj primjer dao isti rezultat:

```
border: gray dashed 1px;
```

Kod svojstva border nije moguće navođenje različite vrijednosti za pojedine ivice, već se moraju koristiti odvojena svojstva: border-top, border-right, border-bottom i border-left. Na primjer:

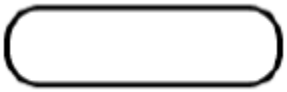
```
border-top: 1px dotted lightgray;
border-right: 4px solid black;
border-bottom: 7px dashed gray;
border-left: 10px double blue;
```

Ako se navedu samo prve dvije deklaracije, vrijednosti za druge dvije ivice neće biti postavljene, nego će oni imati svoje inicijalne vrijednosti (tj. neće biti vidljivi). Na primjer:

```
border-top: 1px dotted lightgray;
border-right: 4px solid black;
```


HTML elementi su pravougli, ali im se uglovi mogu zaobliti. Za postavljanje zaobljenih uglova koristi se svojstvo border-radius, a vrijednost se može postaviti korištenjem mjerne jedinice dužine (pikseli, točke, jedinice em ili rem), a moguće je koristiti i postotke. Na primjer:

```
border-radius: 20px;
```

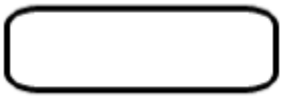


Kao rezultat se dobiva jednoliko zakrivljenje po kružnici radijusa zadane dužine.

Moguće je kreirati i eliptično zakrivljenje, tako da se postave različite vrijednosti za vodoravni i okomiti radijus.

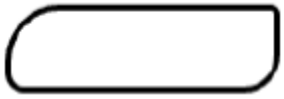
Vodoravni i okomiti radijus treba odvojiti pomoću kose linije. Na primjer:

```
border-radius: 20px / 10px;
```



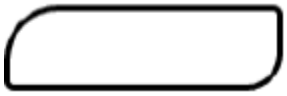
Ako se umjesto mjernih jedinica koriste postotci, horizontalni radijus će biti izračunat kao postotak širine, a vertikalni radijus kao postotak visine elementa. Pomoću svojstva border-radius je moguće definisati različite radijuse na pojedinim uglovima tako da se navedu četiri vrijednosti (bez kose linije). Na primjer:

```
border-radius: 30px 5px 20px 10px;
```



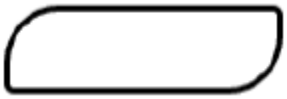
Prva vrijednost se odnosi na gornji lijevi ugao, a slijedeće vrijednosti se odnose na uglove koji slijede u smjeru kazaljke na satu, dakle: gornji desni, donji desni, te donji lijevi ugao. Prilikom navođenja vrijednosti radijusa se mogu navesti samo tri vrijednosti. Prva vrijednost se odnosi na gornji lijevi ugao, druga na gornji desni, a treća na donji desni. Nedostaje vrijednost za donji lijevi ugao, pa ona će biti jednaka vrijednosti za dijametralno suprotni ugao (gornji desni). Npr.:

```
border-radius: 30px 5px 20px;
```



Ako se navedu samo dvije vrijednosti, prva vrijednost se odnosi na gornji lijevi ugao, a druga na gornji desni. Preostali uglovi će imat iste vrijednosti kao uglovi koji su im dijametralno suprotni. Na primjer:

```
border-radius: 30px 5px;
```



Ako se želi postići eliptično zakrivljenje, najprije se navedu vrijednosti za vodoravni radijus za sve uglove, a zatim, nakon kose crte, vrijednosti za okomiti radijus. Na primjer:

```
border-radius: 30px 5px 20px 10px / 20px 15px 10px;
```

Zakrivljenost uglova se može postaviti i pomoću četiri odvojena svojstva koje se odnose na pojedine uglove: border-top-left-radius, border-top-right-radius, border-bottom-left-radius i border-bottom-right-radius. Na primjer:

```
border-top-left-radius: 30px;
border-top-right-radius: 5px;
border-bottom-right-radius: 20px;
border-bottom-left-radius: 10px;
```

Ako se navedu dvije vrijednosti, rezultat je eliptično zakrivljenje. Prva vrijednost predstavlja vodoravni, a druga okomiti radijus elipse (u ovom se slučaju se ne rabi kosa linija za razdvajanje vrijednosti). Na primjer:

```
border-top-left-radius: 30px 20px;
border-top-right-radius: 5px 15px;
border-bottom-right-radius: 20px 10px;
border-bottom-left-radius: 10px 15px;
```

Inicijalna vrijednost za ova svojstva je 0 (svi uglovi su pravougli). Vrijednost se ne nasljeđuje sa roditeljskog elementa. Ako rub elementa nije postavljen (tj. ima inicijalnu vrijednost none), svojstvo border-radius će svejedno imati učinak na pozadinu elementa. Na primjer:

```
border-radius: 20px;
background-color: lightgray;
```


13.7. Kontura elementa

Obris ili **kontura** (engl. *outline*) je linija koja se povlači oko elemenata, izvan granica, kako bi se element „isticao“. Ivice i obrisi su slični. Međutim, konture se razlikuju od granica na sljedeće načine: obrisi nikada ne zauzimaju prostor, jer su izvučeni izvan sadržaja elementa; te prema specifikacijama, konture ne moraju biti pravougaoni, iako su obično. Outline svojstvo CSS-a je skraćenica za postavljanje različitih svojstava obrisa u jednoj deklaraciji za: stil, širinu i boju konture. Ako se ne navede boja konture, primijenjena boja će biti boja teksta. Obris neće biti vidljiv ako njegov stil nije definisan, jer je u tom slučaju vrijednost stila none. Svojstvo konture se može specificirati pomoću jedne, dvije ili tri od dolje navedenih vrijednosti (redoslijed pisanja vrijednosti nije važan). Na primjer:

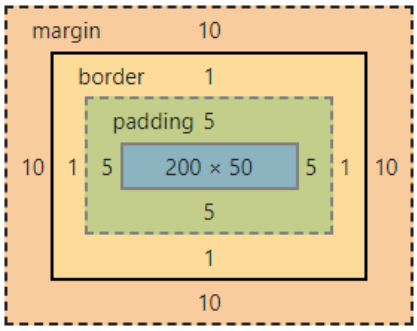
```
outline-width: 3px;
outline-style: solid;
outline-color: royalblue;
```

13.8. Računanje dimenzija elementa

Kad se računaju ukupne dimenzije HTML elementa, tj. prostor koji će on zauzimati, osim postavljene visine i širine se moraju uzeti u obzir ispuna i širina ivice elementa. Za primjer će se koristiti ove dimenzije elementa:

```
width: 200px;
height: 50px;
padding: 5px;
border: 1px solid black;
margin: 10px;
```

Dimenzije elementa se mogu prikazati ovakvim dijagramom:



Vanjski pojas (isprekidana linija) koji je prikazan na ilustraciji je margina elementa. **Žuti pojas** predstavlja ivicu (engl. *border*) elementa. **Vanjski, svijetlo zeleni pojas** (do isprekidane linije) predstavlja ispunu (engl. *padding*) elementa, dok unutrašnje **plavo**, **pravougaono geometrijsko tijelo** predstavlja prostor gdje se nalazi sadržaj. Prilikom postavljanja širine i visine elementa je potrebno imati na umu da se svojstva width i height odnose samo na dio elementa gdje se nalazi sadržaj (unutrašnji plavi pravougaonik). Element neće imati širinu od 200 piksela, nego toj širini dodajemo širinu lijeve i desne ispune i širinu lijeve i desne ivice. Ukupna širina vidljivog dijela:

1px + 5px + 200px + 5px + 1px = 212px

Visina elementa će uključivati ivicu i ispunu:

1px + 5px + 50px + 5px + 1px = 62px

13.9. Resetovanje predefinisanih vrijednosti

Preglednici sami definišu početne vrijednosti za neke elemente. Neki HTML elementi imaju postavljene inicijalne vrijednosti za margine (neki nemaju, što ponekad dovodi do neočekivanih rezultata). Preglednici također definišu inicijalnu veličinu (i vrstu) fonta na nekim elementima, visinu reda (line-height). Budući da se predefinisane vrijednosti razlikuju između preglednika, a često ih zaboravimo sami promijeniti, nastaju sitne razlike u izgledu web stranice zavisno od pregledniku koji koristimo. Zato, na početku CSS dokumenta treba navesti određena pravila

(tzv. CSS reset) kojim se takve vrijednosti postavljaju na inicijalne vrijednosti koje će biti iste u svim preglednicima. Najjednostavnija varijanta CSS reseta je postavljanje margine i ispune svim elementima na nulu, što se može postići pomoću univerzalnog selektora (*):

```
*{
 margin: 0;
 padding: 0;
}
```

Dodatno bi bilo poželjno postaviti visinu reda na prihvatljivu inicijalnu vrijednost:

```
body{
 line-height: 1;
}
```

U ovom je primjeru je visina reda postavljena na vrijednost 1, pa će biti jednaka visini teksta. Umjesto na sve elemente, postavljena je na element body, pa će drugi elementi da naslijede vrijednost sa elementa body. Da je deklaracija visine reda postavljena na 1 unutar pravila koje počinje sa univerzalnim selektorom, bila bi izgubljena mogućnost nasljeđivanja tog svojstva. Kad bi postavili različiti line-height na nekom elementu, njegova djeca ne bi naslijedila tu vrijednost, već bi na njih bila primijenjena vrijednost definisana univerzalnim selektorom. Time bi izgubili očekivano ponašanje, što nije poželjno. Postoji i nekoliko gotovih CSS reseta koji se mogu preuzeti i koristiti u vlastitim projektima[37].

13.10. Pozadinska slika

Osim boje, u pozadinu elementa možemo staviti i sliku. Za postavljanje **pozadinske slike** se koristi svojstvo background-image. Vrijednost se predaje pozivom funkcije url, kojoj predajemo putanju do slike. Na primjer:

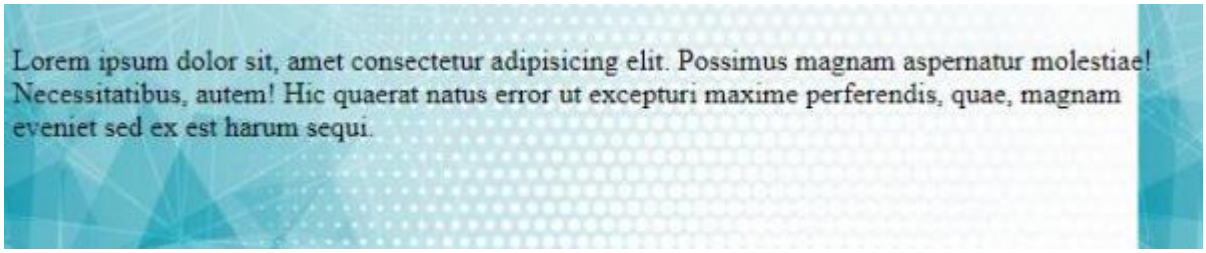
```
background-image: url("plavo.jpg");
```

U mapi u kojoj se nalazi CSS datoteka postoji i slika bckground.png koja izgleda ovako:

[37] https://gist.github.com/DavidWells/18e73022e723037a50d6

U slučaju da u toj CSS datoteci postoji element p na kojem smo već postavili ivicu, visinu i širinu, gornja CSS deklaracija će dati slijedeći rezultat:

Pozadinska slika se prikazuje ispod sadržaja elementa. Ako se u element p doda tekst, tada će izgledati ovako:



Kada se koristiti elementom img, a kada staviti sliku u pozadinu pomoću CSS-a?

Sjetimo se da je namjena CSS-a isključivo prezentacija, dok HTML elementi definišu strukturu i sadržaj stranice. Slike koje služe kao pozadinske slike, koje ne nose neku dodatnu informaciju, nego imaju samo estetsku funkciju, prikazivat ćemo na stranici kao pozadinske slike u CSS-u. Slike koje su dio sadržaja stranice i koje nose informaciju ćemo prikazivati pomoću elementa img. Slika će se na početku ponavljati vodoravno i okomito dok ne popuni cijeli element. Da bi se promijenio način na koji se slika ponavlja, rabi se svojstvo background-repeat, a vrijednosti koje se mogu postaviti tom svojstvu su ključne riječi: no-repeat (slika se ne ponavlja), repeat (slika se ponavlja – predefinisana vrijednost), repeat-x (slika se ponavlja vodoravno), repeat-y (slika se ponavlja okomito), space (slika se ponavlja, ali višak slike neće biti odrezan, već će slike biti razmaknute ujednačeno) i round (slika se ponavlja, ali višak slike neće biti odrezan, nego će slike biti smanjene). Svojstvu background-repeat se mogu predati dvije vrijednosti, pa se u tom slučaju prva vrijednost odnosi na vodoravno, a druga na okomito ponavljanje. Ključne riječi koje se mogu koristiti u tom načinu pisanja su samo: repeat, round i space. Ako želimo isključiti ponavljanje slike, navest ćemo ključnu riječ no-repeat.

Pozadinska slika je inicijalno smještena u gornjem lijevom uglu elementa. Ako je ponavljanje slike isključeno, moguće je promijeniti položaj slike pomoću svojstva background-position. Tom svojstvu se najčešće predaju dvije vrijednosti, pri čemu prva vrijednost predstavlja vodoravni, a druga okomiti pomak od početnog položaja (gornje ivice elementa). Vrijednosti se mogu postaviti u mjernim jedinicama dužine, u postotcima ili se mogu koristiti ključne riječi. Na primjeru vodoravnog pomaka se može objasniti korištenje postotaka. Ako se vodoravni pomak postavi na 0 %, slika će se smjestiti uz lijevu ivicu, a ako se postavi na 100 %, slika će se smjestiti uz desnu ivicu. Vrijednosti između 0 i 100 % se raspodjeljuju jednoliko između tih dvaju krajnjih položaja. Na primjer:

```
background-position: 50% 0;
```

Također se mogu koristiti ključne riječi: bottom (položaj uz donju ivicu - kod okomitog pomaka), center (središnji položaj), left (položaj uz lijevu ivicu - kod vodoravnog pomaka), right (položaj uz desnu ivicu - kod vodoravnog pomaka) i top (položaj uz gornju ivicu - kod okomitog pomaka). Pozadinska slika se prema potrebi može povećati ili smanjiti korištenjem svojstva background-size. Svojstvu se može postaviti vrijednost pomoću jedinica dužine, postotaka (koji se odnose na širinu, odnosno visinu elementa) ili pomoću ključnih riječi. Ovako se može rastegnuti širina slike na pola širine elementa, a visina slike na ukupnu visinu elementa. Ako se postave dvije vrijednosti, prva se odnosi na širinu, a druga na visinu. Ako se postavi samo jedna vrijednost, ona se odnosi na širinu, a visina će biti automatski izračunata tako da se zadrže proporcije slike. Ključne riječi koje koristimo u ovom slučaju su: auto (visina i širina se automatski računa na temelju proporcija slike), cover (slika se povećava - uz zadržavanje proporcija, cijeli element bude pokriven slikom), contain (slika se povećava ili smanjuje - uz zadržavanje proporcija, cijela slika se nalazi unutar elementa). Ključna riječ auto se jedina može koristiti u načinu pisanja kada se navode dvije vrijednosti, a preostale ključne riječi (cover, contain i inherit) određuju i širinu i visinu, pa se ne mogu koristiti na takav način. Pomoću ključne riječi cover se dobiva slijedeći učinak – slika je povećana tako da prekriva cijeli element, pri čemu su proporcije slike zadržane.

Korištenjem ključne riječi contain se može postići da cijela slika bude sadržana unutar elementa uz zadržavanje proporcija.

Na element se može postaviti više različitih pozadinskih slika. Slike se odvajaju zarezom. Na primjer:

```
background-image: url("plavo.jpg"), url("crveno.jpg");
```

Ako želimo da navedemo različite vrijednosti za neka od svojstava koja počinju sa background, navodi se više svojstava odvojenih zarezom. Najčešće se žele navesti različite vrijednosti za svojstvo background-position - **da se slike ne bi nalazile na istim položajima**. Osim navedenih svojstava, pomoću svojstva background se mogu postaviti i vrijednosti za svojstva background-attachment, background-clip i background-origin.

Glavna svojstva za oblikovanje pozadine se mogu postaviti u skraćenom načinu pisanja pomoću svojstva background. To svojstvo objedinjuje ova pojedinačna svojstva: background-color, background-image, background-repeat, background-position i background-size. Sva svojstva su opcionalna, tj. mogu se izostaviti. Redoslijed navođenja svojstava je proizvoljan, osim što se svojstvo background-size mora navesti poslije svojstva **background-position** i biti odvojeno kosom linijom (/). Na primjer:

```
background: color image repeat position/size;
```

Primjer u kojem su navedena sva ta svojstva izgleda ovako:

```
background: greenyellow url('test.png') repeat 20px 10px / auto 30%;
```

Često se može vidjeti da se korištenjem svojstva background postavlja samo pozadinska boja (umjesto korištenjem svojstva background-color). Na primjer:

```
background: rebeccapurple;
```

Moguće je tim svojstvom postaviti samo pozadinsku sliku. Na primjer:

```
background: url('drvo.png');
```

Pomoću svojstva background je također moguće definisati više pozadinskih slika:

```
background: url('test.png') repeat 20px 10px,
            red url('drvo2.png') no-repeat 70px 50px;
```

13.11. Pozadinski prijelaz

Osim jednobojne pozadine, u pozadini elementa se može prikazati i prijelaz između dvije (ili više) boja. **Prijelaz** se može definisati pomoću CSS funkcije linear-gradient. Ta funkcija zapravo u pozadini generiše sliku sa prijelazom, pa se koristi slično kao da se postavlja pozadinska slika – pomoću svojstva background-image (a ne pomoću svojstva background-color). Na primjer:

```
background-image: linear-gradient(#4cd137, #fbc531);
```

U najjednostavnijoj varijanti, funkcija linear-gradient prima dvije boje između kojih se radi prijelaz. Boje se mogu navesti pomoću ključne riječi, heksadecimalnog koda ili kodova RGBa i HSLa (što znači da se mogu koristiti i transparentne boje). Ako se želi postaviti smjer prijelaza, potrebno ga je navesti kao prvi parametar funkcije pomoću ključne riječi. Na primjer:

```
background-image: linear-gradient(to right, #4cd137, #fbc531);
```

Za postavljanje smjera prijelaza se mogu koristiti slijedeće ključne riječi: to top (prijelaz je usmjeren od dna prema vrhu elementa), to right (prijelaz je usmjeren sa lijeve na desnu stranu elementa), to bottom (prijelaz je usmjeren od vrha prema dnu elementa), to left (prijelaz je usmjeren sa desne na lijevu stranu elementa), to top left (prijelaz je usmjeren od donjeg desnog prema gornjem lijevom uglu elementa), to top right (prijelaz je usmjeren od donjeg lijevog prema gornjem desnom uglu elementa), to bottom left (prijelaz je usmjeren od gornjeg desnog prema donjem lijevom uglu elementa), to bottom right (prijelaz je usmjeren od gornjeg lijevog prema donjem desnom uglu elementa). Ako se smjer prijelaza želi definisati preciznije nego što to dopuštaju ključne riječi, potrebno ga je navesti u stepenima (pomoću oznake za stepene u CSS-u – deg). Dopuštene vrijednosti su od 0 do 360, a prijelaz usmjeren na desno (kao u prethodnom primjeru) se postiže postavljanjem vrijednosti od 90 stepeni:

```
background-image: linear-gradient(90deg, #4cd137, #fbc531);
```

Umjesto korištenja samo dvije boje za prijelaz, može se navesti i više boja:

```
background-image: linear-gradient(to left, #4cd137, #fbc531, #00a8ff);
```

Tako se zapravo dodaju dodatne tačke prijelaza. Položaj tih točaka se automatski određuje, ali ga je postaviti navođenjem vrijednosti u jedinicama dužine ili u postotcima

```
background-image: linear-gradient(to left, #4cd137 20%, #fbc531 25%, #00a8ff);
```

Tako je prva tačka prijelaza (između svjetlo plave i žute) postavljena na 20 % širine elementa (s lijeva na desno), a druga tačka prijelaza (između žute i svijetlo zelene) na 70 % širine elementa. Ako se elementu željeli dodati i pozadinska sliku i prijelaz, mogu se navesti dvije vrijednosti za svojstvo background-image odvojene zarezima. Budući da želimo da slika bude prikazana iznad prijelaza, navest ćemo je prvu:

```
background-image: url('plava.png'),
                  linear-gradient(to right, rgb(121, 96, 6), rgb(206, 7,7));
```

Treba da se podesi i svojstvo background-repeat, pa se slika neće ponavljati. Za postavljanje prijelaza se može koristiti skraćeni način pisanja pomoću svojstva background:

```
background: linear-gradient(to right, lightgray, gray);
```

Ako se želi postaviti i pozadinska slika, pomoću svojstva background se može napisati ovako:

```
background: url('test.png') no-repeat,
            linear-gradient(to right, lightgray, gray);
```

13.12. Sjena

Kao što je moguće tekstu dodati sjenu, može se dodati i HTML elementu pomoću svojstva box-shadow.

To svojstvo prima ove vrijednosti, koje treba navesti ovim redoslijedom:

1. **inset** – ključna riječ za postavljanje unutrašnje sjene (opcionalni parametar);
2. **vodoravni pomak** (vrijednost u pikselima ili drugim jedinicama za dužinu);
3. **okomiti pomak** (vrijednost u pikselima ili drugim jedinicama za dužinu);
4. **radijus zamućenja** (opcionalno - vrijednost u pikselima ili drugim jedinicama za dužinu);
5. **radijus proširenja** (opcionalno - vrijednost u pikselima ili drugim jedinicama za dužinu);
6. **boja** (navedena pomoću ključne riječi, heksadecimalnog koda ili kodova RGBa ili HSLa).

Svojstvo box-shadow nije podržano u Interner Explorer 8 i starijim verzijama IE-a. U nastavku je najjednostavniji primjer dodavanja sjene elementu:

```
box-shadow: 5px 5px #2f3640;
```

Radijus zamućenja, slično kao i kod sjene na tekstu, kontroliše koliko će sjena biti oštra ili mutna. Ako se izostavi, njegova vrijednost će biti 0, sjena će biti potpuno „oštra“. Što se postavi veći radijus zamućenja, to će sjena biti mutnija.

```
box-shadow: 5px 5px 8px #2f3640;
```

Radijus proširenja kontroliše koliki će prostor sjena zahvatiti. Ako se izostavi, njegova vrijednost će biti 0, sjena će imati samo inicijalnu veličinu (definisanu vodoravnim i okomitim pomakom).

```
box-shadow: 5px 5px 8px 5px #2f3640;
```

Pomoću ključne riječi none se može isključiti sjena, ako je postavljena nekim drugim pravilom:

```
box-shadow: none;
```

Ako se za vodoravni pomak navede negativna vrijednost, sjena će se postaviti iznad, umjesto ispod elementa, a ako se navede negativna vrijednost za okomiti pomak, sjena će se postaviti lijevo, umjesto desno od elementa.

```
box-shadow: -5px -5px #2f3640;
```

Navođenjem ključne riječi inset, sjena će biti postavljena unutar elementa (uz gornju i lijevu ivicu elementa). U ovom primjeru pokazano je postavljanje unutrašnje sjene:

```
box-shadow: inset -5px -5px #2f3640;
```

Kao i kod sjene na tekstu, moguće je navesti više sjena, odvojenih zarezom:

```
box-shadow: inset 3px 3px 8px #4cd137,
            4px 4px 8px #f5f6fa,
            6px 6px 3px #e84118;
```

U poglavlju je obrađeno:

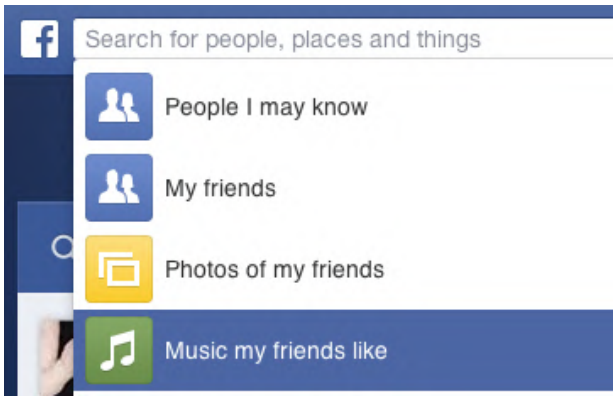
- mijenjanje boje pozadine;
- mijenjanje prikaza elementa;
- postavljanje širine i visine elementa;
- mijenjanje ispune, margine i ivice elementa;
- izračunavanje dimenzije elementa;
- resetovanje predefinisanih vrijednosti;
- postavljanje pozadinske slike;
- postavljanje pozadinskog gradijenta (prijelaza);
- postavljanje sjene na elemente.

XIV Obrasci

Po završetku poglavlja, polaznik će moći:

- pojam, elementi i korištenje obrasca i njegovih atributa;
- korištenje elementa form, input i njihovih atributa;
- grupisanje i validacija elementa obrasca.

Obrasci na web-u služe kako bi korisnici mogli unositi elektronske podatke u preglednik i poslati ih poslužitelju. Ovi podaci mogu biti npr. tekst koji se upisuje u polje za pretraživanje, ili možda podaci koje korisnik upisuje u polje za unos komentara na: web stranici, online anketi, ili u grafičkom interfejsu web aplikacije. Glavna odlika obrazaca na web-u je dugme (engl. *submit button*) čijim pritiskom se podaci šalju poslužitelju. Poslužitelj obrađuje podatke i prikazuje korisniku povratnu informaciju, odnosno rezultate pretrage, ili poruku o uspješnom spremanju podataka i sl. Najpoznatiji primjer obrasca je početna stranica Facebook-a:



14.1. Element <form>

Element form je glavni HTML element web obrasca. Unutar ovoga elementa se nalaze ostali elementi obrasca. Element form ima slijedeće atribute: method – način slanja podataka (sa mogućim vrijednostima: get – podaci se šalju putem URL-a i post – podaci se šalju u tijelu HTTP zahtjeva); action – putanja do stranice (poslužiteljske skripte) koja obrađuje podatke (relativna ili apsolutna putanja) i enctype – način enkodiranja podataka. Na primjer:

```
<form method="post" action="save.php"></form>
```

14.2. Polje za unos teksta

Najčešće korišteni element obrazaca su polje za unos teksta. Većina elemenata obrazaca, pa i polje za unos teksta dobiva se pomoću HTML elementa input. Ovo su osnovni atributi koji se koriste kod polja za unos teksta:

type – postavljen na vrijednost text;
name – naziv polja koji se šalje zajedno sa unesenim podacima;
value – početni tekst upisan u polju;
maxlength – maksimalni dozvoljeni broj znakova;
placeholder – tekst koji daje primjer unosa.

Na primjer:

```
<input type="text" name="ime"/>
```

Ime i prezime:

Kod ovakvog polja za unos teksta je moguće unijeti samo jednu liniju teksta. Pomoću atributa maxlength je moguće ograničiti maksimalan broj karaktera koji se mogu unijeti u polje. Atribut placeholder omogućuje prikaz teksta koji će korisniku dodatno pojasniti namjenu polja. Najčešće se radi o tekstu koji daje primjer unosa, ili dozvoljeni (preporučeni) format unosa. Klikom na polje ovaj tekst će nestati, a korisnik može nesmetano unijeti željeni tekst. Na primjer:

```
Ulica i broj: <input type="text" name="adresa" placeholder="Mis Irbina 3"/>
```

Ulica i broj: Mis Irbina 3

14.3. Dugme (<button> i <input> element)

Obavezni element svakoga obrasca je **dugme** pomoću kojeg se podaci uneseni u obrazac šalju poslužitelju. Dugme za slanje obrasca može se dobiti pomoću elementa input sa atributom type postavljenim na vrijednost submit, ili pomoću elementa button. Osnovni atributi koji se koriste kad se element input koristi kao dugme za slanje obrasca su: type – postavljen na vrijednost submit; name – naziv polja koji se šalje zajedno sa podacima i value – tekst prikazan na dugmetu (i vrijednost koja se šalje ako je dugme pritisnuto). Na primjer:

```
<input type="submit" value="Klikni me"/>
```

Klikni me

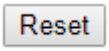
Postoji glavni razlog za upotrebu elementa button umjesto elementa input je zato što je unutar elementa button moguće umetnuti druge HTML elemente, te na taj način dobiti formatirani tekst ili sliku. Također, kod elementa button, vrijednost koja se šalje ako je dugme pritisnuto može biti različita od teksta prikazanog u dugmetu. Ova vrijednost je bitna ako obrazac ima više od jednog dugmeta za slanje, pa je potrebno utvrditi koje dugme je pritisnuto. Kod elementa button se atributi koriste na slijedeći način: type – postavljen na vrijednost submit (može se i

izostaviti); name – naziv polja koji se šalje zajedno sa podacima i value – vrijednost koja se šalje ako je dugme pritisnuto. Na primjer:

```
<button><i>Pošalji</i></button>
```

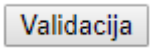
U radu sa obrascima se susrećemo i drugim vrstama dugmeta, npr. dugme za „čišćenje“ obrasca, tj., povratak obrasca na početne vrijednosti. Dugme se može dobiti pomoću elementa input ili pomoću elementa button, kod kojih je type postavljen na vrijednost reset. Na primjer:

```
<input type="reset" value="clear"/>
```



Na kraju, moguća je i treća vrsta dugmeta, koje se također može dobiti pomoću elementa input ili pomoću elementa button, kod kojih je atribut type postavljen na vrijednost button. Ova vrsta dugmeta nema nikakvu ugrađenu funkcionalnost, pa je ponašanje dugmeta potrebno definisati dinamički - pomoću JavaScript-a[38]. Na primjer:

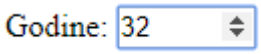
```
<button type="button">Validacija</button>
```



14.4. Polje za unos broja

Polje za unos broja je kreirano pomoću HTML elementa input sa atributom type postavljenim na vrijednost number. Koristi se kada korisnik treba upisati numerički podatak. Osnovni atributi koji se koriste su: type – postavljen na vrijednost number; name – naziv polja koji se šalje zajedno sa unesenim podacima; value – početna vrijednost upisana u polju; min – minimalna vrijednost; max – maksimalna vrijednost i step – korak između dozvoljenih prethodne i slijedeće vrijednosti (predefinisana vrijednost je 1). Na primjer:

```
Godine: <input type="number" name="age" min="1" max="100"/>
```



U ovo polje je moguće upisati samo brojeve (i decimalnu tačku). Uz polje se pojavljuju strelice kojima je moguće povećavati ili smanjivati vrijednost za korak postavljen u atributu step. Pomoću atributa min i max je moguće ograničiti raspon vrijednosti koje je dozvoljeno unijeti. U gornjem primjeru će dozvoljene vrijednosti biti 1, 2, 3 do 100 (uključujući i 100). Ako se atribut step postavi na 3, dozvoljene vrijednosti bit će 1, 4, 7, itd. Ako se atribut step postavi na 0.5,

[38] https://www.w3schools.com/js

dozvoljene vrijednosti bit će 1, 1.5, 2, 2.5, itd. Kod mobilnih preglednika, prilikom unosa u polje se nudi numerička tastatura, što znatno olakšava unos u polje.

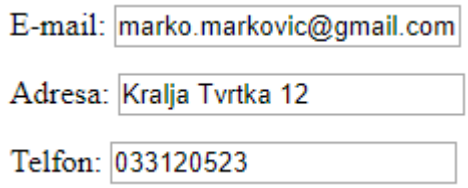
14.5. Polje za unos šifre

Polje za unos šifre je također ostvareno pomoću HTML elementa input. Koristi se kada je potrebno da korisnik upiše šifru ili drugi povjerljivi podatak. Osnovni atributi koji se koriste kod ovog polja su sljedeći: type – postavljen na vrijednost password i name – naziv polja koji se šalje zajedno sa unesenim podacima. Specifičnost polja za unos šifre je u što, radi sigurnosti, znakovi koje korisnik upisuje nisu vidljivi na ekranu. Na primjer:

```
<input type="password" name="šifra"/>
```

14.6. Polje za unos e-mail adrese, URL-a i telefonskog broja

Polja za unos e-mail adrese, URL-a i telefonskog broja su, slično polju za unos broja, elementi za unos specijalne vrste podataka. Osnovni atributi koji se koriste kod ovih polja su sljedeći: type – postavljen na vrijednost email, url ili tel; name – naziv polja koji se šalje zajedno sa unesenim podacima; value – početni tekst upisan u polju; maxlength – maksimalni, dozvoljeni broj znakova i placeholder – tekst koji daje primjer unosa. Na primjer:



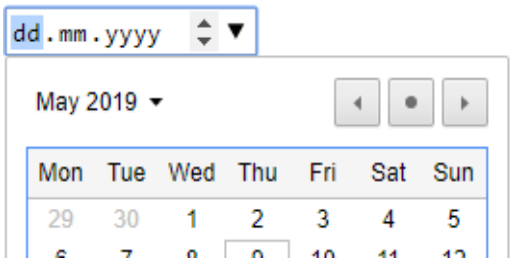
Iako ova polja izgledaju identično kao i obično polje za unos teksta, ona znatno olakšavaju unos ovih specifičnih podataka. Kod polja za unos e-mail adrese i URL-a će biti dozvoljen unos samo ispravne e-mail adrese, odnosno URL-a – zavisno od tipa polja. A ako podatak ne odgovara definisanom formatu, preglednik prikazuje grešku. Na mobilnim uređajima je ponuđena prilagođena tastatura koja nudi samo znakove koji se koriste za unos ovih podataka.

14.7. Polje za unos datuma

Polje za unos datuma je također ostvareno pomoću HTML elementa input. Dakle, koristi se kada korisnik treba unijeti, odnosno odabrati ispravan datum. Datumsko polje nije podržano u svim preglednicima (podržavaju ga: Google Chrome, Microsoft Edge, Opera, Firefox i mobilni Safari). Osnovni atributi koji se koriste su slijedeći: type – postavljen na vrijednost date; name

– naziv polja koji se šalje zajedno sa unesenim podacima; value – početna vrijednost upisana u polju; min – minimalna dozvoljena vrijednost (u formatu gggg-mm-dd) i max – maksimalna dozvoljena vrijednost (u formatu gggg-mm-dd). Klikom na polje za unos datuma pojavljuje se kalendarska kontrola za odabir datuma. Također, preglednik ne dozvoljava unos neispravnog datuma, već u tom slučaju prikazuje grešku. Ovo polje nije implementirano u pregledniku Internet Explorer. Na primjer:

```
<input type="date" name="datum" max="2020-12-31"/>
```



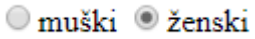
14.8. Polje za pretragu

Za unos teksta za pretragu se može, umjesto običnog polja za unos teksta, koristiti posebno polje za pretragu. Osnovni atributi koji se koriste kod ovog polja su sljedeći: type – postavljen na vrijednost search; name – naziv polja koji se šalje zajedno sa unesenim podacima; maxlength – maksimalni dozvoljeni broj znakova i placeholder – tekst koji daje primjer unosa. U nekim preglednicima ovo polje dobiva specijalne funkcije – pojavljuje se krst za brisanje unesenog teksta, te se prikazuju ranije uneseni pojmovi za pretragu. Iako nema velike razlike u odnosu na obično polje za unos teksta, preporučuj se da koristiti polje za pretragu tamo gdje se radi o pretraživanju, jer se time daje semantička informacija o namjeni polja.

14.9. Dugme za odabir

Dugme za odabir (engl. *radio button*) se upotrebljava kada korisnik treba odabrati vrijednost sa popisa vrijednosti (samo jedna vrijednost može biti odabrana). Naziv „radio button“ dolazi od dugmadi na starim radio aparatima, kod kojih se pritiskom na jedno dugme automatski izbacuje i time isključuje ostalu dugmad. Osnovni atributi koji se koriste kod ovog elementa su: type – postavljen na vrijednost radio; name – naziv polja koji se šalje zajedno sa unesenim podacima; value – vrijednost koja se šalje poslužitelju i checked – postavlja se kada će se dugme unaprijed odabrati. Na primjer:

```
<input type="radio" name="spol" value="male"/>muški
<input type="radio" name="spol" value="female" checked="checked"/>ženski
```

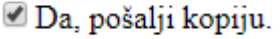


Dugmad za odabir koja se nalazi na istom popisu vrijednosti trebaju imati istu vrijednost atributa name. Na taj način će biti moguće odabrati samo jednu opciju – odabir jedne opcije sa popisa automatski isključuje druge opcije (ako su prethodno bile odabrane).

14.10. Okvir za označavanje

Okvir za označavanje (eng. *checkbox*) služi da bi se uključila / isključila neka opcija. Atributi koji se koriste kod ovog elementa su: type – postavljen na vrijednost checkbox; name – naziv polja koji se šalje zajedno sa unesenim podacima; value – vrijednost koja se šalje poslužitelju (ne ispisuje se) i checked – postavlja se ako kvačica treba biti unaprijed označena. Popis ponuđenih opcija kod koje je moguće odabrati više od jedne opcije najčešće se ostvaruje pomoću popisa okvira za označavanje (po jedna za svaku opciju). Na primjer:

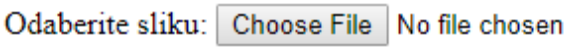
```
<input type="checkbox" name="info" value="true"/>Da, pošalji kopiju.
```



14.11. Polje za odabir datoteke

Polje za odabir datoteke služi da korisnik može odabrati datoteku sa svog računara. Klikom na dugme za slanje obrasca, datoteka će biti poslana na poslužitelj. Ako se koristi polje za odabir datoteke nije moguće koristiti metodu get (putem URL-a) za slanje podataka na poslužitelj. Da bi slanje datoteke na poslužitelj funkcionisalo, potrebno je na elementu form postaviti atribut enctype na vrijednost multipart/form-data. Osnovni atributi koji se koriste kod ovog elementa su: type – postavljen na vrijednost file; name – naziv polja koji se šalje zajedno sa podacima; accept – lista dozvoljenih tipova datoteka (npr. .jpg, .png, .doc, i dr.) i multiple – postavlja se ako se dopušta odabir više datoteka. Na primjer:

```
Odaberite sliku: <input type="file" name="slika"/>
```



Izgled ovog polja, dugmeta za odabir i teksta koji se ispisuje u dugmetu, kao i dijaloga za odabir datoteke, nije moguće mijenjati, on zavisi od preglednika i operativnih podešavanja na sistemu računara.

14.12. Skriveno polje

Skriveno polje se koristi kada se želi poslati vrijednost koja neće biti vidljiva korisniku. Primjer takve vrijednosti može biti identifikator zapisa iz baze ili druge vrijednosti koje su potrebne poslužitelju, a nije ih potrebno prikazati korisniku, koristi će se element input sa atributom type

postavljenim na vrijednost hidden. Atributi koji se koriste kod ovog polja: type – postavljen na vrijednost hidden; name – naziv polja koji se šalje zajedno sa podacima i value – vrijednost koja se šalje poslužitelju. Na primjer:

```
<input type="hidden" name="id" value="111"/>
```

14.14. Polje za višelinijski unos teksta

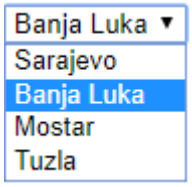
Kada je potrebno omogućiti unos teksta u obrazac, umjesto elementa `<input type="text"/>` se koristi HTML element textarea. Osnovni atributi koji se koriste kod elementa textarea su: name (naziv polja koji se šalje sa unesenim podacima); maxlength (maksimalni dozvoljeni broj znakova) i placeholder (tekst koji daje primjer unosa). Ovaj element uvijek mora imati početnu i završnu oznaku. Ne može se pisati samo `<textarea>`, te je uvijek potrebno napisati `<textarea></textarea>`. Ako je potrebno napisati početnu vrijednost koja će biti prikazana unutar ovoga polja, potrebno ju je ispisati unutar oznaka `<textarea>` . Unutar ovoga elementa korisnik može „prelamati“ tekst u više redova. Na primjer:

```
<textarea name="komentar" maxlength="200"></textarea/>
```

14.13. Lista za odabir

Lista za odabir omogućava odabir jedne (ili više) ponuđenih opcija. U ovu svrhu se mogu koristiti i dugmad za odabir (engl. *radio buttons*), odnosno polja za označavanje (engl. *checkboxes*), ako se radi o većem broju ponuđenih vrijednosti, bolje je upotrijebiti listu za odabir. Lista za odabir se dobiva pomoću HTML elementa select. Njegovi atributi su: name (naziv polja koji se šalje zajedno sa unesenim podacima) i multiple (omogućava odabir više vrijednosti sa liste). Za svaku opciju unutar liste za odabir potrebno je navesti po jedan HTML element option. Njegovi atributi su: value (vrijednost koja se šalje na poslužitelj ako je ta opcija odabrana) i selected (da li je ova opcija unaprijed odabrana). Tekst koji se ispisuje za svaku opciju se piše unutar option elementa. Pomoću atributa multiple je moguće dobiti listu za odabir u kojoj se može odabrati više od jedne vrijednosti. Na primjer:

```
<select name="gradovi">
    <option value="Sa">Sarajevo</option>
    <option value="Bl" selected="selected">Banja Luka</option>
    <option value="Mo">Mostar</option>
    <option value="Tu">Tuzla</option>
</select>
```



14.15. Označavanje elemenata obrasca (<label> element)

Uz svako polje u obrascu je potrebno navesti i njegov naziv tj. oznaku. Takav tekst daje značenje pojedinom polju, te ga je staviti unutar HTML elementa label. Osim vizualno, element label treba i u kodu povezati sa odgovarajućim elementom polja, da bi se olakšala upotreba obrasca osobama koje koriste čitače ekrana (kako bi čitač znao pročitati ispravnu oznaku polja). Ovo povezivanje se postiže tako što se na element label postavi atribut for koji ima vrijednost koju ima atribut id na odgovarajućem elementu input (ili drugom elementu obrasca). Npr.:

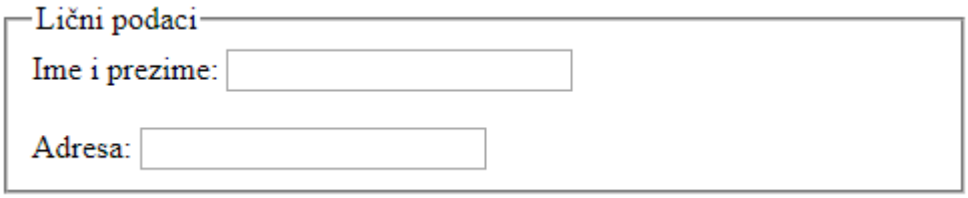
```
<label for="name">Ime i prezime:</label>
<input type="text" id="name" name="name"/>
```

Na prvi pogled slični atributi id i name, imaju različitu namjenu. Atribut id služi za referenciranje elementa iz JavaScript-a i CSS-a (te za povezivanje sa elementom label), te mora biti jedinstven unutar dokumenta. Atribut name služi da bi se postavio naziv koji označava poslani podatak, i više elemenata (na primjer grupa dugmadi za odabir) može imati isti name. Dodatna prednost elementa label je što se klikom na oznaku postavlja fokus na polje za unos teksta. Kod dugmadi za odabir ili polja za označavanja ovo ponašanje je još korisnije – klikom na oznaku (element label) automatski se bira dugme (odnosno polje za označavanje), čime postaje lakše odabrati željenu opciju (područje koje je osjetljivo na klik je znatno veće).

14.16. Grupisanje elemenata obrasca

Kako bi obrasci koji sadrže puno polja bili pregledniji, potrebno je vizualno grupirati srodne elemente u odvojene cjeline. Takve cjeline dobivaju se upotrebom elementa fieldset, unutar kojih se navode elementi obrasca koji pripadaju toj cjelini. Kao prvi element unutar elementa fieldset može doći element legend, unutar kojeg se navodi naslov pojedine cjeline. Na primjer:

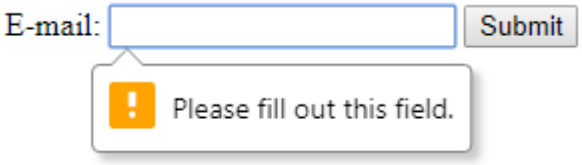
```
<fieldset>
    <legend>Lični podaci</legend>
    <label for="ime">Ime i prezime:</label>
    <input id="ime" type="text" name="ime"><br/><br/>
    <label for="adresa">Adresa:</label>
    <input type="text" id="adresa" name="adresa"/></fieldset>
```



14.17. Validacija podataka unesenih u obrazac

Kako bi se onemogućilo slanje neispravnih podataka, u preglednik su ugrađene određene mogućnosti provjere, odnosno validacije upisanih podataka. Već spomenuti atribut maxlength ograničava dužinu upisanog teksta, te je moguća provjera i ograničenje unosa kod specijalnih tipova podataka (numerički podataka, datumski podatak, URL i e-mail adresa). Glavni i najčešći način validacije polja je provjera je li ono popunjeno (ako se radi o obaveznom polju). To se postiže pomoću atributa required koji je moguće postaviti na većini elemenata obrasca. Polje na kojem je postavljen ovaj atribut mora biti popunjeno kako bi se obrazac mogao poslati poslužitelju. Podatke dobivene iz HTML obrasca je potrebno provjeriti prije spremanja u bazu, bez obzira na eventualne provjere u samom HTML obrascu. Na primjer:

```
E-mail: <input type="text" name="email" required="required"/>
```



Polja za unos broja, e-mail adrese, URL-a, telefonskog broja, datuma i polje za pretragu dio su HTML5 standarda. Ova polja za unos (sa iznimkom datumskog) podržana su u svim modernim preglednicima, a u starim preglednicima ponašaju se kao obično polje za unos teksta: `<input type="text"/>`.

U poglavlju je obrađeno:

- pojam, elementi i korištenje obrasca i njegovih atributa;
- korištenje elementa form, input i njihovih atributa;
- grupisanje i validacija elementa obrasca.

XV Oblikovanje atipičnih elemenata

Po završetku poglavlja, polaznik će moći:

- koristiti CSS svojstva u kontekstu linkova;
- koristiti CSS svojstva u kontekstu polja za unos;
- koristiti CSS svojstva u kontekstu lista;
- koristiti CSS svojstva u kontekstu slika.

U ovom poglavlju će se upotrijebiti dosada obrađena CSS svojstva kako bi se oblikovali HTML elementi sa kojima se svakodnevno susrećemo, ali ipak imaju neke svoje posebnosti što se tiče oblikovanja. Pritom će se obraditi i neka, još neobrađena, CSS svojstva.

15.1. Linkovi

Link, odnosno element a je jedan od osnovnih elemenata u HTML-u (element na kojem se temelji čitav koncept hiperteksta). Inicijalno svi preglednici prikazuju linkove plavom bojom i podcrtano. Kada se definiše osnovni izgled teksta za web-stranicu, najčešće se to radi na elementu body. Svi drugi elementi će naslijediti svojstva fonta sa elementa body. U ovom primjeru će se postaviti vrsta fonta, boja i početna veličina teksta:

```
body{
font-family: "Lato";
font-size: 1rem;
color: gainsboro;
}
```

Najpoznatiji pretraživač je Google.

Iz prethodnog primjera se može zaključiti da je element a naslijedio vrstu i veličinu fonta sa elementa body, ali boja linka je i dalje ostala plava. Razlog je zato što se pretkazivačevo vlastito CSS pravilo za postavljanje plave boje linka primjenjuje direktno na element a i mijenja vrijednost naslijeđenu sa elementa body. Da bi se promijenila boja linka, potrebno je da se u CSS-u web stranice boja postavi direktno na element a:

```
a{
    color: darkcyan;
}
```

Ali, moguće je primijetiti da su linkovi ponekad ljubičasti, iako im je već postavljena boja. Jer, preglednik linkove koje je posjetitelj već posjetio prikazuje ljubičastom bojom. To se ponašanje

može promijeniti pomoću CSS pseudoklase :visited, pseudoklasa se primjenjuje na linkove koji su već posjećeni. Na primjer:

```
a:visited{
    color: burlywood;
}
```

Pomoću pseudoklase :hover se može postići da izgled linka bude promijenjen u trenutku kad korisnik prijeđe mišem preko linka. Tako korisniku dodatno dajemo do znanja da je taj element stranice aktivan, odnosno da se na njega može kliknuti mišem. Najčešće se koriste slijedeći efekti: promjena boje linka, promjena pozadinske boje linka i postavljanje/uklanjanje linije ispod teksta. Na primjer:

```
a:hover{
color: rebeccapurple;
background-color: yellowgreen;
text-decoration: wavy;
}
```

Prema tipu prikaza, element a je linijski (engl. *inline*) element, tj. zauzima samo onoliko mjesta koliko je veliki njegov sadržaj. Zbog toga je i prostor koji je osjetljiv na klik mali. Kada se radi o linkovima koji su izdvojeni iz teksta (npr. linkovi u navigaciji), moguće je povećati prostor na koji korisnik može kliknuti mišem tako da se element pretvori u blok element (ili inline blok), te da se postavi odgovarajuća visina i širina. Na primjer:

```
a{
display: inline-block;
width: 100px;
height: 70px;
}
```

Ako se radi o slici koja je i link (ako unutar elementa a postoji element img), neki preglednici (npr. starije verzije Internet Explorera) prikazat će plavu ivicu oko slike. Da bi se on uklonio, potrebno je postaviti ivicu takve slike na none:

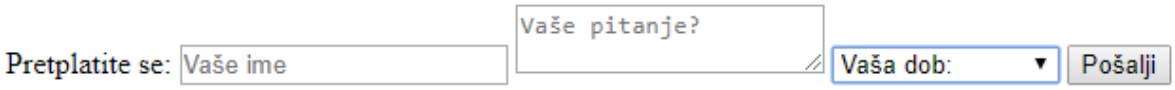
```
a img{
border-style: none;
}
```

15.2. Polja za unos

Polja za unos podataka preglednici prikazuju različito. Postizanje željenog izgleda polja za unos, koji će pritom biti jednak u svim preglednicima, ponekad je zahtjevno, a kod nekih tipova polja

(npr. polje za prijenos datoteke) zapravo i nije moguće. Međutim, nekima od najčešće korištenih polja za unos ipak je moguće prilagoditi izgled u većini preglednika. Ako imamo primjer:

```
<form>
    Pretplatite se:
    <input type="text" placeholder="Vaše ime"/>
    <textarea placeholder="Vaše pitanje?"></textarea>
    <select>
        <option>Vaša dob:</option>
        <option>do 15 godina</option>
        <option>15 - 25 godina</option>
        <option>25 - 30 godina</option>
        <option>preko 30 godina</option>
    </select>
    <button>Pošalji</button>
</form>
```

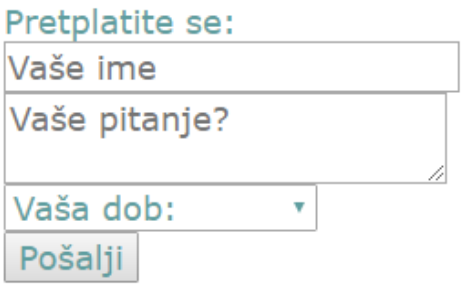


Ako se osnovna svojstva teksta definišu na elementu body, polja za unos će da zadrže vlastitu veličinu, boju i vrstu fonta. Na primjer:

```
body{
font-family: Verdana, sans-serif;
font-size: 18px;
color: cadetblue;
}
input, textarea, section, button{
    display: block;
}
```

Inicijalni tekst u poljima za unos teksta (postavljen pomoću atributa placeholder) se prikazuje u većini preglednika sivom bojom. Boja teksta se može promijeniti samo u nekim preglednicima. Boja teksta se u listi za odabir (select elementu) može promijeniti samo u nekim preglednicima. Za slučaj da postoji više različitih elemenata input, pravilo treba ograničiti samo na polje za unos teksta pomoću selektora input[type='text']. Na primjer:

```
input[type="text"], textarea, select, button{
font-family: inherit;
font-size: inherit;
color: inherit;
}
```

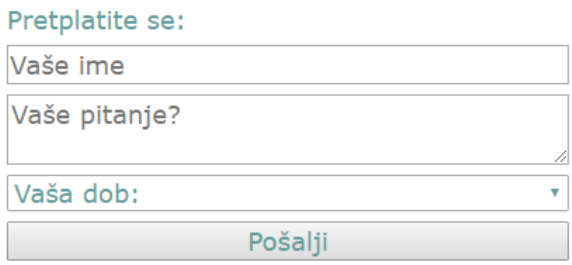


Prednost takvog pristupa je da su vrijednosti za vrstu fonta, veličinu i boju navedene samo na jednom mjestu, što omogućava lakše izmjene. Želimo li na poljima za unos imati npr. različitu veličinu ili boju teksta, možemo postaviti konkretne vrijednosti tim svojstvima. Polja za unos su linijski elementi pa u slučaju da želimo da svaki bude prikazan u svojem redu, treba ih pretvoriti u blok elemente. Kako bi se ujednačio razmak između pojedinih polja u svim preglednicima, potrebno je postaviti i okomitu marginu na elemente. Na primjer:

```
input[type="text"], textarea, select, button{
display: block;
margin: 7px 0px;
}
```

Ako želimo ujednačiti širinu elemenata, možemo im jednostavno postaviti širinu na željenu vrijednost pomoću svojstva width. U nekim preglednicima (npr. IE 9 i starije verzije) to neće rezultirati jednakom širinom, zbog načina kako preglednici inicijalno računaju širinu. Na primjer:

```
input[type="text"], textarea, select, button{
width: 400px;
box-sizing: border-box;
}
```

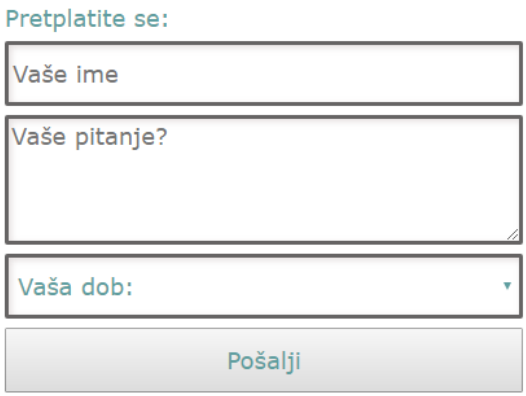


Poljima za unos se može postaviti visina i ispuna. Potrebno je elementu textarea postaviti veću visinu. Na primjer:

```
input[type="text"], select, button{
height: 50px;
padding: 3px;
 }
textarea{
height: 100px;
padding: 2px 2px;
}
```

Da bi polja za unos imala istu boju ivice u svim preglednicima, potrebno je eksplicitno postaviti. Također je moguće zaobliti ivice, te dodati unutrašnju sjenu. Na primjer:

```
input[type="text"], select, textarea{
border: 3px solid rgb(102, 100, 100);
border-radius: 2px;
box-shadow: inset 2px 2px 5px rgb(230, 225, 225);
}
```



Element textarea neće biti sasvim zaobljen u Internet Exploreru zbog strelica za pomicanje koje se nalaze sa desne strane. Strelice se mogu sakriti (barem inicijalno, dok korisnik ne unese tekst u veći od polja) pomoću svojstva overflow, kojim se vrijednost postavlja na auto da bi se strelice pojavile samo kada su potrebne. Dugme za slanje podataka se može uljepšati zaobljivanjem i stavljanjem prijelaza u pozadinu. Prilikom klika mišem na dugme, preglednici obično promijene

njegov izgled tako da dočaraju reakciju dugmeta na klik. Budući da je inicijalni izgled dugmeta u ovom slučaju promijenjen, izgubljena je ta dodatna funkcionalnost pa je potrebno promijeniti izgled pritisnutog dugmeta. Pritisnuto dugme se može selektovati pomoću pseudoklase :active. Preglednici najčešće označavaju trenutno odabrani element dodavanjem plave ivice oko njega. Korištenjem pseudoklase :focus je moguće odrediti kako će se vizualno označiti taj element. Da bi se isključio plavi obris koji preglednik sam postavlja oko odabranog elementa, potrebno je upotrijebiti svojstvo outline. Ono služi za postavljanje obrisa, a slično je svojstvu border za postavljanje ivice. Za razliku od ivice, obris se nalazi izvan elementa i ne uračunava se u njegove dimenzije. Svojstvo outline je skraćeni način pisanja za odvojena svojstva: outline-style; outline-width i outline-color, koja primaju iste vrijednosti kao i svojstva border-style, border-width i border-color. Postavljanjem vrijednosti svojstva outline na none uklonit će se taj plavi obris.

15.3. Liste

Liste u HTML-u se često koriste kad se želi navesti niz međusobno povezanih elemenata. Inicijalni način na koji se lista prikazuje definiše preglednik, međutim CSS omogućava potpunu kontrolu nad tim kako će lista biti prikazana. Izgled oznake stavke se može promijeniti pomoću CSS svojstva list-style-type, kojoj se kao vrijednost predaje ključna riječ. Važno je napomenuti da se to svojstvo postavlja na samu listu (element ul), a ne na stavku liste (elementu li). Ključne riječi koje se mogu koristiti su: disc (krug - inicijalna vrijednost nenumerisane liste); circle (kružnica); square (kvadrat); decimal (decimalni brojevi - inicijalna vrijednost numerisane liste); decimal-leading-zero (decimalni brojevi sa početnom nulom); lower-roman (rimski brojevi malim slovima); upper-roman (rimski brojevi velikim slovima); lower-greek (grčka slova); lower alpha (mala slova latinice); lower-latin (isto kao i lower-alpha); upper-alpha (velika slova latinice); upper-latin (isto kao i upper-alpha); armenian (armenski brojevi); georgian (gruzijski brojevi); none (bez oznake). Prve tri vrijednosti se odnose samo na nenumerisane liste (element ul), a sljedećih 11 vrijednosti se odnosi samo na numerisane liste (element ol). Korištenjem svojstva list-style-type moguće je ne numerisanu listu prikazati kao numerisanu i obrnuto, ali to nije poželjno.

Ako nam oznaka stavke ne odgovara, možemo umjesto nje upotrijebiti vlastitu sliku pomoću svojstva list-style-image. Kao vrijednost treba postaviti poziv funkcije url, kojoj se predaje putanja do slike. Ako je postavljena slika za oznaku stavke pomoću ovog svojstva, ona će biti upotrijebljena čak i ako je postavljeno svojstvo list-style-type. Osim poziva funkcije url, kao vrijednost ovom svojstvu se može predati i ključna riječi none (inicijalna vrijednost).

Pomoću svojstva list-style-position je moguće podešavati položaj oznake. Kao vrijednost tom svojstvu se mogu postaviti ključne riječi: outside (oznaka je izvan prostora koji zauzima tekst) i inside (oznaka je unutar prostora koji zauzima tekst). Djelovanje tog svojstva je vidljivo kada imamo tekst koji se „prelama“ u slijedeći red. Inicijalna vrijednost tog svojstva je outside. Kada bi se postavila vrijednost inside, oznaka stavke bi se nalazila unutar prostora u kojem se nalazi tekst, pa bi zadnja stavka, koja se prelama u novi red, izgledala drugačije.

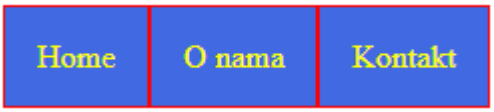
Ta tri svojstva vezana za liste mogu se napisati na skraćeni način pomoću svojstva list-style. Kao vrijednost se navodi vrijednosti za svojstva: list-style-type; list-style-image i list-style-position u proizvoljnom redoslijedu, a neka se mogu izostaviti. Na primjer:

```
ul{
list-style: circle url('head.png') inside;
}
```

Istovremeno postavljanje svojstava list-style-image i list-style-type u načelu nema previše smisla (jer prvo svojstvo poništava drugo), pa se najčešće postavljaju samo dva svojstva. Prema tipu prikaza, liste su blok elementi, tj. inicijalno zauzimaju cijeli red. Stavka liste, tj. element li također zauzima cijeli red, ali on ima vlastiti tip prikaza (list-item), koji je sličan blok elementu, ali prikazuje i oznaku stavke. Ako se promijeni tip prikaza elementu li (u želji da se dobije vodoravna lista, oznaka liste više neće biti vidljive).

Liste se, budući da predstavljaju niz povezanih elemenata, često koriste za definisanje navigacijskog izbornika web stranice. Unutar stavki liste, umjesto običnog teksta, u tom se slučaju se stavljaju elementi a koji vode na pojedine stranice. Tako navigacija dobiva poželjnu strukturu liste, ali najčešće se ne želi zadržati inicijalni izgled liste sa oznakama stavki. Promjenom tipa prikaza na elementu li i dodatnim stilizovanjem je moguće listu oblikovati po želji. Na primjer:

```
li{
    display: inline;
    background-color: royalblue;
    color: yellow;
    padding: 1rem;
    border: 1px solid red;
}
```



15.4. Tabele

Tabele su HTML elementi čija je namjena organizacija sadržaja u tabličnom obliku. U načelu se tu uvijek radi o podacima koje je prirodno prikazati kao redove u tabeli. Tabele su se nekada koristile i za kreiranje rasporeda stranice, zbog toga što je pomoću njih bilo relativno lahko podijeliti stranicu na kolone. Takva praksa je bila dok u CSS-u nije postojalo dobro rješenje za taj problem, danas se nikako ne preporučuju tablice da bi se postiglo željeno pozicioniranje elemenata. Preglednici tabele inicijalno prikazuju bez ivica, a tekst unutar ćelija zaglavlja (elementi th) će biti centriran i podebljan. Da bi tabela bila preglednija, treba prikazati linije koje omeđuju redove i kolone. To se postiže tako da se postavi ivica na ćelije (elemente td i th).

Preglednici inicijalno postavljaju određeni razmak između ćelija. Pomoću svojstva border-spacing se može upravljati veličinom tog razmaka. To svojstvo se postavlja na table, a prima vrijednost definisanu u nekoj od jedinica dužine. Ako se tom svojstvu predaju dvije vrijednosti, prva vrijednost će se odnositi na vodoravni razmak (između kolona), a druga na okomiti razmak (između redova). Kao vrijednost svojstvu se može postaviti i ključna riječ inherit, čime će se naslijediti vrijednost sa roditeljskog elementa. Ako želimo razmak između ukloniti, svojstvo border-spacing se može postaviti na nulu. Prostor između ćelija je nestao, ali zbog „dodirivanja“ susjednih ivica neke linije izgledaju „deblje“. Umjesto upotrebom svojstva border-spacing, ivice pojedinih ćelija se mogu spojiti pomoću svojstva border-collapse, ako se njegova vrijednost postavi na ključnu riječ collapse. Ako je svojstvo border-collapse postavljeno na vrijednost collapse, vrijednost svojstva border-spacing se ignoriše, nije ga potrebno postavljati. Inicijalna vrijednost border-collapse je separate.

Veličina tabele i ćelija unutar nje se automatski prilagođavaju sadržaju. Ako je potrebno, moguće je postaviti širinu ili visinu na tabelu ili na pojedine ćelije, čime se definiše širina kolone, odnosno visina reda. Ako postavimo širinu tabele na 300 piksela, a širina prve kolone želimo postaviti na 33.3 % širine tabele. Za podešavanje širine kolone se može pomoću pseudoklase :first-child selektovati prva ćelija u redu.

Ako želimo da se tekst u ćeliji poravna vodoravno, to se može postići pomoću svojstva text-align. Za okomito poravnanje teksta u ćelijama tabele se koristi svojstvo vertical-align. Svojstvo text-align se koristi za horizontalno poravnanje teksta i unutar ćelija tabela i unutar blok elemenata (npr. ili <p> elemenata). Za razliku od tog svojstva, svojstvo vertical-align se ne može koristiti za vertikalno poravnanje teksta unutar blok elemenata. Kada se primjenjuje na ćelije tabele, mogu se koristiti ključne riječi: top (tekst je okomito poravnat uz gornju ivicu

ćelije); middle (tekst se nalazi u sredini ćelije - inicijalna vrijednost) i bottom (tekst je okomito poravnat uz donju ivicu ćelije).

15.5. Slike

Slike se u HTML-u dodaju pomoću elementa img. Na web stranicama se (najčešće) prikazuju uz tekst, te postoji više načina na koji se slika može uklopiti u tekst. Kao primjer možemo uzeti:

```
<p>
    <img src="rimsko-carstvo.jpg" alt="zastava"/> Rimsko carstvo je uobičajeni
    naziv...
</p>
```

Prema tipu prikaza, element img je linijski element, pa će se slika inicijalno prikazivati u jednom redu teksta:

Rimsko Carstvo je uobičajeni naziv za rimsku državu nakon što ju je preustrojio Oktavijan August u zadnja tri desetljeća prije Krista. Iako je Rim imao imperij stoljećima prije Augustove samovlade, predaugustovska država se obično naziva Rimskom Republikom. Rimsko Carstvo je upravljalo svim heleniziranim državama na Sredozemlju, kao i keltskim područjima sjeverne Europe. Najveću teritorijanu ekspanziju doživjelo je u doba cara Trajana, koji je 114. prisvojio Armeniju i sjevernu Mezopotamiju, a 115. godine osvojio je i Adiabene, Babilon i Ktezifont. Zadnji rimski car zbačen je 476. godine, ali tada je istočnim područjima već vladao drugi car sa sjedištem u Konstantinopolu. Istočno Rimsko Carstvo (Bizant) i dalje je postojalo, iako se postupno smanjivalo, sve do 1453. godine, kad su Turci osvojili Konstantinopol. Kasnije države na zapadu (Franačko Kraljevstvo i Sveto Rimsko Carstvo) i na istoku (ruski carevi) koristile su rimsko državničko nazivlje sve do modernog doba. Golemo naslijeđe Rimskog Carstva vidi se i danas u zapadnjačkim institucijama, pravu, arhitekturi i mnogim drugim područjima života

Da bi se slika smjestila unutar teksta, koristi se CSS svojstvo float. To svojstvo izbacuje element iz normalnog toka dokumenta, postavlja ga uz lijevu (ili desnu) ivicu roditeljskog elementa. Tekst i drugi linijski elementi koji se nalaze pored elementa kojem je dodijeljeno CSS svojstvo float se postavljaju i prelamaju oko njega. Ovako slika se postavlja uz lijevu ivicu, a tekst teče oko slike i prelama se oko nje:

```
img{
    float: left;
}
```

Rimsko Carstvo je uobičajeni naziv za rimsku državu nakon što ju je preustrojio Oktavijan August u zadnja tri desetljeća prije Krista. Iako je Rim imao imperij stoljećima prije Augustove samovlade, predaugustovska država se obično naziva Rimskom Republikom. Rimsko Carstvo je upravljalo svim heleniziranim državama na Sredozemlju, kao i keltskim područjima sjeverne Europe. Najveću teritorijanu ekspanziju doživjelo je u doba cara Trajana, koji je 114. prisvojio Armeniju i sjevernu Mezopotamiju, a 115. godine osvojio je i Adiabene, Babilon i Ktezifont. Zadnji rimski car zbačen je 476. godine, ali tada je istočnim područjima već vladao drugi car sa sjedištem u Konstantinopolu. Istočno Rimsko Carstvo (Bizant) i dalje je postojalo, iako se postupno smanjivalo, sve do 1453. godine, kad su Turci osvojili Konstantinopol. Kasnije države na zapadu (Franačko Kraljevstvo i Sveto Rimsko Carstvo) i na istoku (ruski carevi) koristile su rimsko državničko nazivlje sve do modernog doba. Golemo naslijeđe Rimskog Carstva vidi se i danas u zapadnjačkim institucijama, pravu, arhitekturi i mnogim drugim područjima života S obzirom da je položaj tribuna tradicionalno bio vezan uz narod, Augustova je moć dodatno narasla. Kao drugo, stekao je nove ovlasti u obliku "imperijalne" moći, što je značilo da ima vrhovnu vlast u svim pitanjima teritorijalne uprave. Godina 23. pr. Kr. obično se računa kao godina kad je August postao rimski car. Ipak, više je volio građanske naslove Princeps i "Prvi građanin". Kao car, August je hladnokrvno i efikasno uredio svoje carstvo; njegove su izuzetne sposobnosti velikim dijelom zaslužne što se Rimsko Carstvo održalo tako dugo. Uveo je standardizirani novac i poreze; izgradio je birokratsku strukturu od vitezova i slobodnjaka (bivših robova). Vojnicima je uredio mirovine. Bio je vješt propagandist. Dobio je potporu rimske književnosti kad je postao pokrovitelj pjesnika Horacija, Livija i nadasve Vergilija. Da se svidi narodu, koristio je igre i posebne događaje koji su slavili njega i njegovu obitelj.

Svojstvo float može da poprimi slijedeće vrijednosti: left (element je postavljen uz lijevu ivicu roditeljskog elementa); right (element je postavljen uz desnu ivicu roditeljskog elementa) i none (element je u normalnom toku (engl. *normal flow*) dokumenta – inicijalna vrijednost). Bez obzira što se element u HTML kodu img element nalazi prije teksta, u slučaju da se vrijednost svojstva float postavi na right, tekst se prikazuje na lijevoj strani, a slika na desnoj. Na primjer:

```
img{
    float: right;
}
```

Rimsko Carstvo je uobičajeni naziv za rimsku državu nakon što ju je preustrojio Oktavijan August u zadnja tri desetljeća prije Krista. Iako je Rim imao imperij stoljećima prije Augustove samovlade, predaugustovska država se obično naziva Rimskom Republikom. Rimsko Carstvo je upravljalo svim heleniziranim državama na Sredozemlju, kao i keltskim područjima sjeverne Europe. Najveću teritorijanu ekspanziju doživjelo je u doba cara Trajana, koji je 114. prisvojio Armeniju i sjevernu Mezopotamiju, a 115. godine osvojio je i Adiabene, Babilon i Ktezifont. Zadnji rimski car zbačen je 476. godine, ali tada je istočnim područjima već vladao drugi car sa sjedištem u Konstantinopolu. Istočno Rimsko Carstvo (Bizant) i dalje je postojalo, iako se postupno smanjivalo, sve do 1453. godine, kad su Turci osvojili Konstantinopol. Kasnije države na zapadu (Franačko Kraljevstvo i Sveto Rimsko Carstvo) i na istoku (ruski carevi) koristile su rimsko državničko nazivlje sve do modernog doba. Golemo naslijeđe Rimskog Carstva vidi se i danas u zapadnjačkim institucijama, pravu, arhitekturi i mnogim drugim područjima života. Kao car, August je hladnokrvno i efikasno uredio svoje carstvo; njegove su izuzetne sposobnosti velikim dijelom zaslužne što se Rimsko Carstvo održalo tako dugo. Uveo je standardizirani novac i poreze; izgradio je birokratsku strukturu od vitezova i slobodnjaka (bivših robova). Vojnicima je uredio mirovine. Bio je vješt propagandist. Dobio je potporu rimske književnosti kad je postao pokrovitelj pjesnika Horacija, Livija i nadasve Vergilija. Da se svidi narodu, koristio je igre i posebne događaje koji su slavili njega i njegovu obitelj. August je osnovao i prve profesionalne vatrogasce na svijetu, te je u Rimu uveo profesionalnu policiju. Kao apsolutni vladar carstva, imenovao je svog nasljednika. Tako je vratio običaj koji je odbačen još pri osnutku Rimske Republike kao sramotan. U početku je to trebao biti Marcel, sin njegove sestre, koji se oženio Augustovom kćeri Julijom. Marcel je umro 23. pr. Kr. zbog trovanja hranom. Kasniji su povjesničari tvrdili da je Marcela, kao i razne druge članove carske obitelji, otrovala Augustova žena Livija Drusila, ali to su obična nagađanja.

Da bi se slika odmaknula od teksta, mogu joj se dodati margine. Na primjer:

```
img{
    float: left;
    margin: 10px 20px 10px 0;
}
```

Rimsko Carstvo je uobičajeni naziv za rimsku državu nakon što ju je preustrojio Oktavijan August u zadnja tri desetljeća prije Krista. Iako je Rim imao imperij stoljećima prije Augustove samovlade, predaugustovska država se obično naziva Rimskom Republikom. Rimsko Carstvo je upravljalo svim heleniziranim državama na Sredozemlju, kao i keltskim područjima sjeverne Europe. Najveću teritorijanu ekspanziju doživjelo je u doba cara Trajana, koji je 114. prisvojio Armeniju i sjevernu Mezopotamiju, a 115. godine osvojio je i Adiabene, Babilon i Ktezifont. Zadnji rimski car zbačen je 476. godine, ali tada je istočnim područjima već vladao drugi car sa sjedištem u Konstantinopolu. Istočno Rimsko Carstvo (Bizant) i dalje je postojalo, iako se postupno smanjivalo, sve do 1453. godine, kad su Turci osvojili Konstantinopol. Kasnije države na zapadu (Franačko Kraljevstvo i Sveto Rimsko Carstvo) i na istoku (ruski carevi) koristile su rimsko državničko nazivlje sve do modernog doba. Golemo naslijeđe Rimskog Carstva vidi se i danas u zapadnjačkim institucijama, pravu, arhitekturi i mnogim drugim područjima života S obzirom da je položaj tribuna tradicionalno bio vezan uz narod, Augustova je moć dodatno narasla. Kao drugo, stekao je nove ovlasti u obliku "imperijalne" moći, što je značilo da ima vrhovnu vlast u svim pitanjima teritorijalne uprave. Godina 23. pr. Kr. obično se računa kao godina kad je August postao rimski car. Ipak, više je volio građanske naslove Princeps i "Prvi građanin". Kao car, August je hladnokrvno i efikasno uredio svoje carstvo; njegove su izuzetne sposobnosti velikim dijelom zaslužne što se Rimsko Carstvo održalo tako dugo. Uveo je standardizirani novac i poreze; izgradio je birokratsku strukturu od vitezova i slobodnjaka (bivših robova). Vojnicima je uredio mirovine. Bio je vješt propagandist. Dobio je potporu rimske književnosti kad je postao pokrovitelj pjesnika Horacija, Livija i nadasve Vergilija. Da se svidi narodu, koristio je igre i posebne događaje koji su slavili njega i njegovu obitelj.

Ako se slika ćeli dodatno oblikovati, moguće joj je npr. dodati zaobljene ivice:

```
img{
    float: left;
    margin: 10px 20px 10px 0;
    border-radius: 50%;
}
```

Rimsko Carstvo je uobičajeni naziv za rimsku državu nakon što ju je preustrojio Oktavijan August u zadnja tri desetljeća prije Krista. Iako je Rim imao imperij stoljećima prije Augustove samovlade, predaugustovska država se obično naziva Rimskom Republikom. Rimsko Carstvo je upravljalo svim heleniziranim državama na Sredozemlju, kao i keltskim područjima sjeverne Europe. Najveću teritorijanu ekspanziju doživjelo je u doba cara Trajana, koji je 114. prisvojio Armeniju i sjevernu Mezopotamiju, a 115. godine osvojio je i Adiabene, Babilon i Ktezifont. Zadnji rimski car zbačen je 476. godine, ali tada je istočnim područjima već vladao drugi car sa sjedištem u Konstantinopolu. Istočno Rimsko Carstvo (Bizant) i dalje je postojalo, iako se postupno smanjivalo, sve do 1453. godine, kad su Turci osvojili Konstantinopol. Kasnije države na zapadu (Franačko Kraljevstvo i Sveto Rimsko Carstvo) i na istoku (ruski carevi) koristile su rimsko državničko nazivlje sve do modernog doba. Golemo naslijeđe Rimskog Carstva vidi se i danas u zapadnjačkim institucijama, pravu, arhitekturi i mnogim drugim područjima života S obzirom da je položaj tribuna tradicionalno bio vezan uz narod, Augustova je moć dodatno narasla. Kao drugo, stekao je nove ovlasti u obliku "imperijalne" moći, što je značilo da ima vrhovnu vlast u svim pitanjima teritorijalne uprave. Godina 23. pr. Kr. obično se računa kao godina kad je August postao rimski car. Ipak, više je volio građanske naslove Princeps i "Prvi građanin". Kao car, August je hladnokrvno i efikasno uredio svoje carstvo; njegove su izuzetne sposobnosti velikim dijelom zaslužne što se Rimsko Carstvo održalo tako dugo. Uveo je standardizirani novac i poreze; izgradio je birokratsku strukturu od vitezova i slobodnjaka (bivših robova). Vojnicima je uredio mirovine. Bio je vješt propagandist. Dobio je potporu rimske književnosti kad je postao pokrovitelj pjesnika Horacija, Livija i nadasve Vergilija. Da se svidi narodu, koristio je igre i posebne događaje koji su slavili njega i njegovu obitelj.

Ako sliku želimo dodatno istaknuti, možemo joj dodati ivicu. Da bi ivica bila vidljiva, može se odvojiti od slike pomoću ispune. Na primjer:

```
img{
    float: left;
    margin: 10px 20px 10px 0;
    border-radius: 50%;
    border: 3px solid yellow;
    padding: 10px;
    box-shadow: 8px 5px 5px #747d8c;
}
```

Rimsko Carstvo je uobičajeni naziv za rimsku državu nakon što ju je preustrojio Oktavijan August u zadnja tri desetljeća prije Krista. Iako je Rim imao imperij stoljećima prije Augustove samovlade, predaugustovska država se obično naziva Rimskom Republikom. Rimsko Carstvo je upravljalo svim heleniziranim državama na Sredozemlju, kao i keltskim područjima sjeverne Europe. Najveću teritorijanu ekspanziju doživjelo je u doba cara Trajana, koji je 114. prisvojio Armeniju i sjevernu Mezopotamiju, a 115. godine osvojio je i Adiabene, Babilon i Ktezifont. Zadnji rimski car zbačen je 476. godine, ali tada je istočnim područjima već vladao drugi car sa sjedištem u Konstantinopolu. Istočno Rimsko Carstvo (Bizant) i dalje je postojalo, iako se postupno smanjivalo, sve do 1453. godine, kad su Turci osvojili Konstantinopol. Kasnije države na zapadu (Franačko Kraljevstvo i Sveto Rimsko Carstvo) i na istoku (ruski carevi) koristile su rimsko državničko nazivlje sve do modernog doba. Golemo naslijeđe Rimskog Carstva vidi se i danas u zapadnjačkim institucijama, pravu, arhitekturi i mnogim drugim područjima života S obzirom da je položaj tribuna tradicionalno bio vezan uz narod, Augustova je moć dodatno narasla. Kao drugo, stekao je nove ovlasti u obliku "imperijalne" moći, što je značilo da ima vrhovnu vlast u svim pitanjima teritorijalne uprave. Godina 23. pr. Kr. obično se računa kao godina kad je August postao rimski car. Ipak, više je volio građanske naslove Princeps i "Prvi građanin". Kao car, August je hladnokrvno i efikasno uredio svoje carstvo; njegove su izuzetne sposobnosti velikim dijelom zaslužne što se Rimsko Carstvo održalo tako dugo. Uveo je standardizirani novac i poreze; izgradio je birokratsku strukturu od vitezova i slobodnjaka (bivših robova). Vojnicima je uredio mirovine. Bio je vješt propagandist. Dobio je potporu rimske književnosti kad je postao pokrovitelj pjesnika Horacija, Livija i nadasve Vergilija. Da se svidi narodu, koristio je igre i posebne događaje koji su slavili njega i njegovu obitelj.

U poglavlju je obrađeno:

- korištenje CSS svojstva u kontekstu linkova;
- korištenje CSS svojstva u kontekstu polja za unos;
- korištenje CSS svojstva u kontekstu lista;
- korištenje CSS svojstva u kontekstu slika.

XVII Jedinice mjere u CSS-u

Po završetku poglavlja, polaznik će moći:

- razlikovati relativne i apsolutne jedinice mjere u CSS-u;
- razlikovati specijalne jedinice mjere u CSS-u;

U realnom životu za dužinu imamo usvojenu jednu jedinicu mjere (metar), u CSS-u postoji veliki broj jedinica mjere. Ovim jedinicama se definišu vrijednosti velikog broja svojstava (npr. width, height, margin, padding, font-size, border, itd.). Po pravilu vrijednost svojstva koje opisuje dužinu treba da ima brojnu vrijednost i jedinicu mjere, bez međusobnih razmaka (npr.: 19%, 25px, 2.5em, itd.). Za pojedina svojstva, ove vrijednosti mogu biti i negativne. Jedinice dužine mogu biti **relativne**, **apsolutne** i **specijalne**.

17. 1. Relativne jedinice

Relativne jedinice definišu dužinu u odnosu na veličinu nekog drugog svojstva. Ovaj način se koristi ako je potrebno skaliranje svojstva, njegovo prilagođavanje za različito prikazivanje na raznim medijima:

Jedinica	Opis	Napomena
em	relativna u odnosu na veličinu (visinu) fonta	1em = 16px
ex	relativna u odnosu na visinu slova „x“	
ch	relativna u odnosu na širinu „0“ (nule)	
rem	relativna u odnosu na širinu fonta vršnog svojstva	
vw	relativna u odnosu na širinu viewport-a[39]	1vw je 1% širine prostora
vh	relativna u odnosu na visinu viewport-a	1vw je 1% visine prostora
vmin	relativna u odnosu na manje dimenzije viewport-a	
vmax	relativna u odnosu na veće dimenzije viewport-a	
%	postotak odgovarajuće veličine	

17.2. Apsolutne jedinice

Apsolutne jedinice definišu fiksnu veličinu svojstva:

Jedinica	Opis	Napomena
cm	centimetar	1cm = 0.39in = 37.8px = 28.34pt = 2.36pc
mm	milimetar	1mm = 0.039in = 3.78px = 2.83pt = 0.24pc
in	inč	1in = 2,54cm = 96px = 72pt = 6pc
px	piksel	1px = 1/96 dio inča = 0.26mm
pt	tačka	1pt = 1/72 dio inča = 0.35mm
pc	pika	1pc = 1/6 inča = 12pt = 4.23mm

[39] Vidljivo područje korisniku web stranice

Jedinica piksel je zamišljena kao najmanja jedinica mjere digitalne slike (i najmanji element koji je moguće adresirati) i definisana je kao 96. dio inča (~0,26 mm), tj. jedna tačka na zaslonu kod uređaja niske rezolucije. Zbog toga je piksel uvršten u tabelu apsolutnih jedinica, iako je veličina piksela zavisna od medija.

17.3. Specijalne jedinice

Postoje i specijalne jedinice mjere, koje se koriste u posebne svrhe i zadaju se samo za određene atribute:

Jedinica	Opis	Napomena:
s	sekunda	tranzicije koje traju neko određeno vrijeme.
ms	milisekunda	za preciznije određivanje trajanja (1s = 1000ms)
deg	stepen (°)	krug ima 360deg, pozitivne vrijednosti predstavljaju rotaciju u smjeru kazaljke na satu, a negativne u suprotnom smjeru.

Postoji i funkcija calc() se zadaje na mjestu metrike u CSS deklaraciji, kao parametar prima neki aritmetički izraz. Na primjer:

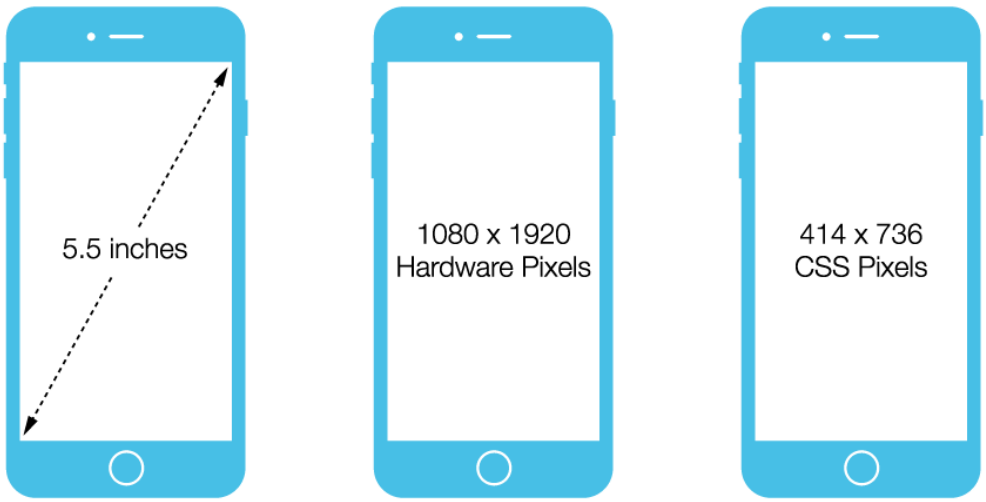
```
width: calc(100% - 150px);
```

U izrazu učestvuju operandi koji mogu biti metrike ili obične numeričke vrijednosti, i operatori (+, -, *, / - obavezni razmaci oko operatora) koji mogu biti matematičke operacije. U izrazu možemo koristiti i zagrade. Kao operand u izrazu je moguće navesti i samu funkciju calc(), s tim što se ona računa kao da je u pitanju običan izraz u zagradama.

17.4. Razlika između hardverskih i CSS piksela

Razlika između veličine prikaza hardverskih i CSS piksela je očigledna. To je vrlo zbunjujuće za većinu dizajnera. U praksi postoji više od desetak načina mjerenja dužine u web programiranju (piksel, tačka, em, ex, pc, %, itd.), a sve to zavisi o web dizajneru. Naime, piksel više nije najmanja tačka koja se prikazuje na ekranu. To znači da pri dizajniranju savremenog, visokokvalitetnog zaslona više nema hardverskih piksela koji upravljaju dimenzijama vaše grafike na „stvarnoj veličini“, sada se to zovu **CSS pikseli**. Broj CSS piksela će biti mnogo manji od broja hardverskih piksela na zaslonima visoke rezolucije, ali isti će biti isti na običnom zaslonu. Tradicionalno, grafički dizajneri bi koristili hardverske piksele kao mjernu jedinicu, a web sjedišta bi bila fiksnih dimenzija. Danas, to više nije slučaj u responzivnim web sjedištima, jer današnja web sjedišta su dizajnirana za skaliranje sadržaja, te koriste tačke loma (engl. *breakpoints*) za preraspodjelu

sadržaja za različite formate zaslona i razlučivost. U nastavku imamo ilustraciju koja jasno pokazuje razliku između hardverskih i CSS piksela:

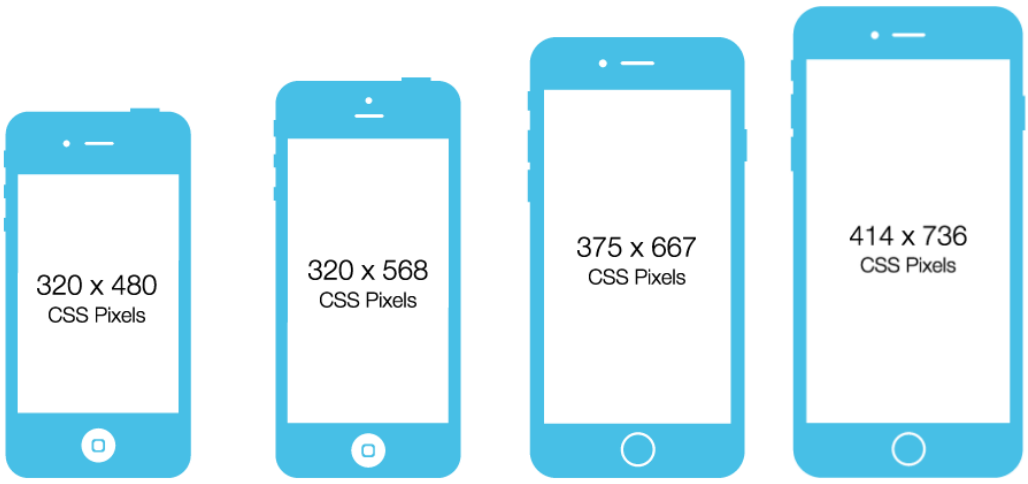


Dijagonala zaslona
Dijagonalna, odnosno mjera zaslona, od ugla do ugla u inčima, baš kao što bi smo izmjerili TV ekran.

Hardverski pikseli
Stvarna rezolucija piksela na fizičkom zaslonu, obično je nazivamo razlučivosti zaslona.

CSS pikseli
De facto veličina koja se koristi u CSS-u ili JavaScript-u za određivanje veličine prikaza na tom uređaju, stvarna veličina grafike na pregledniku.

Prilikom izrade web sjedišta, kao mjernu jedinicu ćemo koristiti CSS piksele, jer definiše naše tačke loma, a i zato što je to jedina precizna mjerna jedinica za projektovanje našeg web sjedišta. CSS pikseli su „stvarna veličina“ na kojoj korisnik vidi grafiku u pregledniku, bez obzira na njegovu rezoluciju zaslona. Uzmimo za primjer iPhone, ovaj primjer vrijedi i za bilo koji mobilni, tablet, desktop uređaj, 4K zaslon, 5K ili HDTV. Jednostavno rečeno, CSS pikseli je „stvarna veličina“ na kojoj korisnik vidi prozor preglednika. Hardverski pikseli predstavljaju „razoluciju zaslona“, a veličina zaslona (engl. *display size*) je veličina našeg uređaja.



iPhone 4
CSS Pixels: 320 x 480
Hardware Pixels: 640 x 960
Display Size: 3.5 inch

iPhone 5
CSS Pixels: 320 x 568
Hardware Pixels: 640 x 1136
Display Size: 4 inch

iPhone 6
CSS Pixels: 375 x 667
Hardware Pixels: 750 x 1334
Display Size: 4.7 inch

iPhone 6 Plus
CSS Pixels: 414 x 736
Hardware Pixels: 1080 x 1920
Display Size: 5.5 inch

Postoji formula pomoću koje preglednik računa logičku (odnosno interpretiranu, odnosno CSS) rezoluciju ekrana;

$$\text{(Logička ili CSS) rezolucija ekrana} = \frac{\text{(Fizička ili Hardverska) rezolucija}}{\text{Omjer piksela uređaja}}$$

Logička rezolucija ekrana predstavlja količnik **hardverske rezolucije** i **omjera piksela uređaja**[40]. Ako želimo da izračunamo logičku rezoluciju ekrana na primjer Apple iPhone 6s telefona, tada ćemo koristit prethodno navedenu formulu:

$$\text{Logička rezolucija ekrana} = \frac{750 \text{ x } 1334}{2} = 375 \text{ x } 667$$

CSS će „misliti" da uređaj ima zaslon rezolucije 375 × 667, a medija upit (engl. *media query*) će „odgovoriti“ kao da je zaslon 375 × 667. Prikazani elementi na zaslonu će biti dvostruko „oštriji“ od stvarnog zaslona 375 × 667, jer na fizičkom zaslonu ima dvostruko više fizičkih piksela.

U poglavlju je obrađeno:

- razlikovati relativne i apsolutne jedinice mjere u CSS-u;
- razlikovati specijalne jedinice mjere u CSS-u;

[40] Na linku www.mydevice.io možete pronaći omjer fizičke i logičke rezolucije (omjer piksela uređaja) za sve mobilne uređaje koji su u upotrebi.

XVIII Pozicioniranje elemenata

Po završetku poglavlja, polaznik će moći:

- koristiti pozicioniranje elemenata pomoću tipa prikaza;
- koristiti pozicioniranje pomoću svojstva float;
- koristiti pozicioniranje elemenata pomoću svojstva postion;
- koristiti pozicioniranje elemenata pomoću tipa prikaza flexbox;

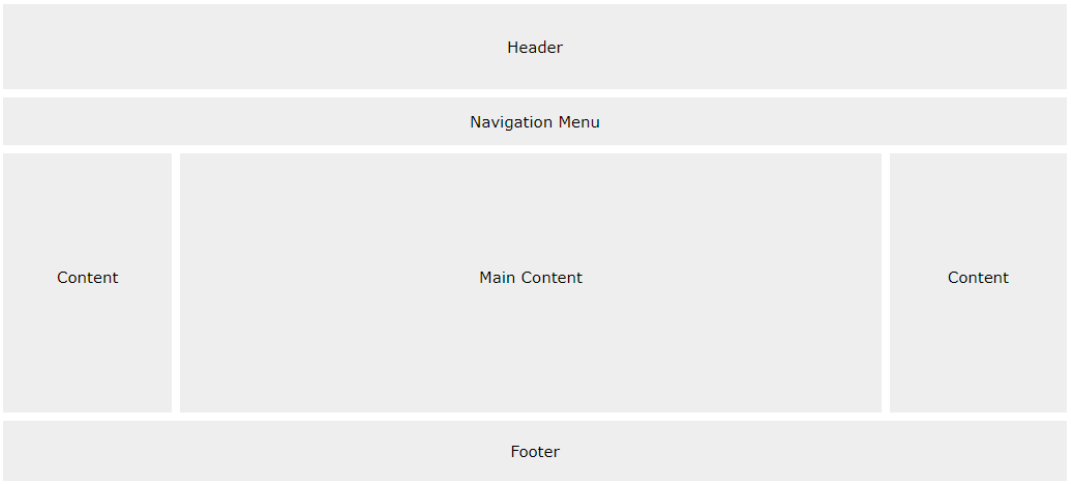
Pozicioniranje HTML elemenata pomoću CSS-a je *de facto* najteži dio CSS-a, a glavni razlog je što do sada CSS nije na jednostavan način omogućivao pozicioniranje kakvo su web dizajneri željeli. Željeni raspored HTML elemenata na web stranici se može postići na više različitih načina, međutim složeniji raspored ponekad zahtijeva i upotrebu nekih trikova. Tehnike koje se koriste (ili su se koristile) za pozicioniranje, u CSS-u su:

1. HTML tabele (ne preporučuje se);
2. CSS-ovo svojstvo float;
3. CSS flexbox;
4. CSS biblioteke (Bootstrap, Materialize CSS, Semantic UI, i dr.);
5. CSS grid.

U novije vrijeme razvile su se (ili su u razvoju) nove mogućnosti pozicioniranja u CSS-u. Modeli **flexbox** i **grid** (bit će tema ovog priručnika), su najnovija dostignuća u pozicioniranju elemenata, te se može reći da je pozicioniranje u CSS-u konačno riješeno na zadovoljavajući način.

18.1. Pozicioniranje pomoću tipa prikaza inline-block

Blok elementi zauzimaju cijeli red, pa čak i kada im se postavi manja širina. Budući da oni uvijek uzrokuju prebacivanje sljedećeg elementa u novi red, nije moguće postaviti dva blok elementa u istom redu. Za razliku od njih, linijski elementi neće uzrokovati prebacivanje u novi red, ali njima se ne mogu podešavati širina i visina. Postavljanjem svojstva display na vrijednost inline-block, elementu je moguće zadati širinu i visinu, a on neće uzrokovati prelazak u novi red. Tako se u istom redu može imati više elemenata, tj. taj red se može podijeliti na više kolona. Inline-block elementi su podržani tek od Internet Explorera 8, pa ih se ne može koristiti za raspored stranice u ranijim preglednicima. Jedan od najčešće korištenih rasporeda na web stranicama sastoji se od: zaglavlja, navigacionog menija, glavnog dijela koji je podijeljen u tri kolone i podnožja. U praksi to izgleda ovako:



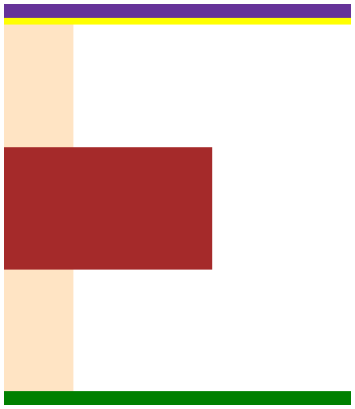
A HTML kod bi trebao izgledati ovako:

```
<header></header>
<nav></nav>
<aside></aside>
<main></main>
<aside></aside>
<footer></footer>
```

U ovom se slučaju elementi element aside, main i aside se žele prikazati u istoj koloni. Da bi se to postiglo, prvo im treba postaviti širinu: aside elementi će imati širinu od 20 %, a main element preostalih 60 %. Odnosno:

```
aside{
    width: 20%;
}
main{
    width: 60%;
}
```

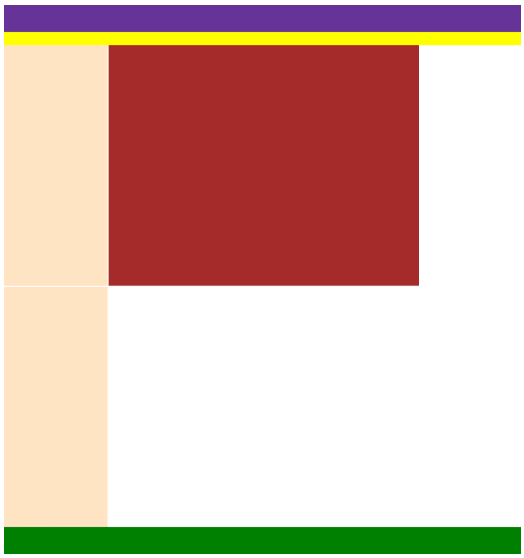
Svaki blok element zauzima cijeli red, bez obzira što mu je postavljena širina. Rezultat ovakve CSS deklaracije je slijedeći:



Elementima koji se žele smjestiti kao kolone treba promijeniti tip prikaza u inline-block:

```
main, aside{
    display: inline-block;
}
```

Međutim, rezultat nije željeni:



Prvi aside i main se zaista nalaze u istom redu, ali čini se da između kolona postoji određeni razmak, pa zbog toga nema dovoljno mjesta da i drugi aside bude u istom redu. Ovaj razmak se pojavljuje zbog toga što se linijski blok elementi djelomično ponašaju kao linijski elementi, pa kada postoji bilo kakav razmak između njih u HTML kodu (pa makar i prelazak u novi red), taj je razmak vidljiv u pregledniku. Da bi se izbjegao taj razmak, HTML kod se mora pisati tako da nakon završetka jednog elementa odmah započinje slijedeći:

```
<aside>
</aside><main>
</main><aside>
</aside>
```

Tako se postiže željeni raspored elemenata (uz malu nepreglednosti HTML koda). U slučaju da se želi zadržati preglednost HTML koda, moguće je osigurati da ne bude razmaka ubacivanjem HTML ovih oznaka za komentare između elemenata, kao u ovom primjeru:

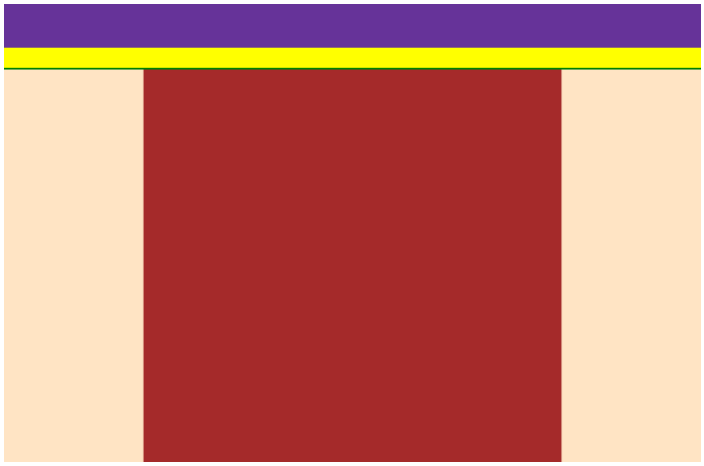
```
<aside><!--
--></aside><!--
--><main>
</main><!--
--><aside>
</aside>
```

18.1. Pozicioniranje pomoću svojstva float

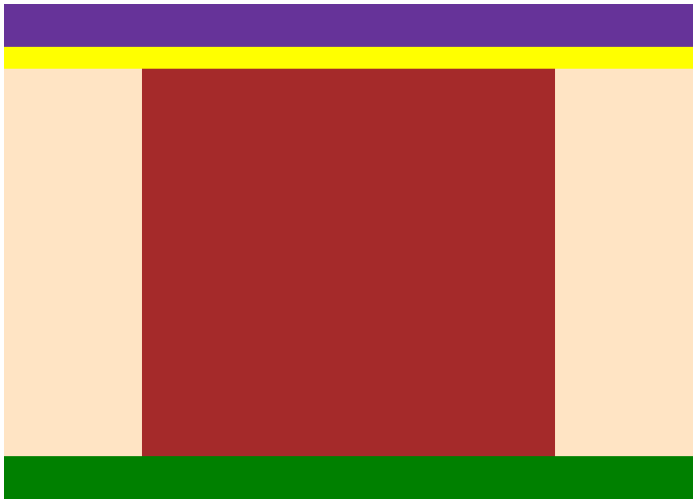
Osnovna namjena svojstva float je prelamanje teksta oko slike. Svojstvo float se može koristiti i za postizanje željenog rasporeda elemenata na stranici. Blok elementima kojim se postavi svojstvo float neće uzrokovati prelazak u novi red, nego će ih biti moguće smjestiti jedan do drugog u istom redu. Elementu kojem se postavi svojstvo float na vrijednost left, bit će pozicioniran uz lijevu ivicu roditeljskog elementa. Ako se i sljedećem elementu postavi svojstvo float na left, on će se pozicionirati uz ivicu prethodnog elementa. Na elemente koje želimo pozicionirati kao kolone, postavit ćemo svojstvo float na vrijednost left:

```
main, footer{
    float: left;
}
```

Kod korištenja svojstva float se ne treba brinuti o razmaku između elemenata u HTML kodu, ali postoje drugi problemi koji se mogu pojaviti. Elementi koji slijede iza elemenata sa svojstvom float se mogu ponašati neočekivano. U ovom primjeru, element footer se pozicionirao uz donju ivicu elementa header, te više nije vidljiv u svojim definisanim dimenzijama:



Da bi se izbjegli takvi problemi, u većini slučajeva je dovoljno postaviti svojstvo clear na prvi naredni element koji dolazi iza elemenata sa svojstvom float. Svojstvom clear će da isključi djelovanje svojstva float na elemente koji slijede, nastavlja prirodno pozicioniranjem elemenata na stranici. Svojstvo clear se može postaviti na ove ključne riječi: left - poništava „plivanje“ (engl. *floating*) elemenata na lijevoj strani; right - prestaje „plivanje“ (engl. *floating*) elemenata na desnoj strani i both – prestaje „plivanje“ (engl. *floating*) elemenata na obje strane. Postavljanjem svojstva clear na vrijednost left na elementu <footer> ćemo dobiti željeni raspored elemenata na stranici:



18.4. Pozicioniranje pomoću svojstva postion (static, relative, absolute i fixed)

HTML elementi se u prozoru preglednika prikazuju redoslijedom kojim su navedeni u HTML kodu. Blok elementi (npr. elementi i <p>) zauzimaju cijeli red i prebacuju slijedeći element u novi red. Linijski elementi (npr. elementi i) ne zauzimaju cijeli red, već samo onoliko mjesta koliko zauzima njihov sadržaj. To je normalni tok elemenata u HTML-u. Elementi slijede jedan iza drugog, odnosno redoslijedom kojim su napisani. Ali, pomoću CSS svojstva position moguće je „izbaciti“ željeni element iz tog slijeda, te ga staviti na neki drugi položaj na stranici, koji ne mora imati veze sa njegovim redoslijedom u HTML kodu. Position se može postaviti na četiri osnovne vrijednosti: static, relative, absolute i fixed.

Vrijednost static je inicijalna vrijednost koju ima svojstvo position, ako nije promijenjeno u neku drugu vrijednosti. Predefinisana riječ static označava da se nalazi unutar normalnog slijeda elemenata. U praksi će se ta vrijednost eksplicitno navoditi samo kad se želi isključiti druga vrijednost (postavljena pomoću drugog CSS pravila). Na primjer:

```
<header></header>
<main></main>
<footer></footer>
```

Postavljanjem vrijednosti relative za svojstvo position se dopušta da element bude pomaknut u odnosu na mjesto koje bi inače zauzimao. Pomak elementa se definiše pomoću CSS svojstava: top, right, bottom i left. Na primjer:

```
main{
    position: relative;
    top: 20px;
    right: 50px;
}
```

Za definisanje pomaka je dovoljno navesti samo dva svojstva: top i left ili bottom i right. Svojstvima: top, right, bottom i left se mogu predati i negativne vrijednosti, što uzrokuje pomak u suprotnom smjeru. Postavljanje svojstava: top, right, bottom i left na elementu kod kojeg svojstvo position ima vrijednost static neće imati nikakav učinak. Važno je napomenuti da kada se koristimo vrijednošću relative, prostor koji bi pomaknuti element zauzimao ostaje sačuvan, a element koji slijedi iza njega ne pomiče se sa svojeg originalnog položaja da bi popunio taj slobodni prostor. Kod preostalih vrijednosti svojstva position (absolute i fixed) nije taj slučaj. Korištenjem vrijednosti absolute se element također pomiče iz slijeda elemenata na stranici, a pomoću svojstava top, right, bottom ili left se određuje pomak elementa u odnosu na ishodišni element. Ishodišni element u odnosu na koji se element pomiče inicijalno je element body. Ako je potrebno, kao ishodišni element se može postaviti i drugi element. Na primjer:

```
main{
    position: absolute;
    top: 30px;
    right: 40px;
}
```

Pomoću svojstva top se navodi udaljenost za koju se element pomiče u odnosu na gornju ivicu ishodišnog elementa, a pomoću svojstva left u odnosu na lijevu ivicu ishodišnog elementa. Ako se koristi svojstvo bottom, element se pomiče za navedenu udaljenost u odnosu na donju ivicu elementa, ako se koristi svojstvo right, u odnosu na desnu ivicu elementa. Ako je ishodišni element body, pomak će biti u odnosu na donji, odnosno desnu ivicu prozora preglednika (jer element body zauzima prostor preglednika). U primjeru se vidi da kad se koristi vrijednost absolute, prostor koji je zauzimao pomaknuti element nije sačuvan, nego se na njega smjestio slijedeći element u nizu. Kao ishodišni element, umjesto elementa body, se može postaviti neki od roditeljskih elemenata našeg elementa koji se želi pomaknuti. Postavlja se tako da se roditeljevo svojstvo position postavi na relative. HTML kod će se izmijenit tako što će se npr. unutar main elementa staviti novi element article:

```
<header></header>
<main>
    <article></article>
</main>
<footer></footer>
```

Elementu <main> ćemo postavit svojstvo position na relative, dakle postavit će se kao ishodišni element:

```
main{
    position: relative;
}
```

Novom elementu article (koje je potomak elementa main) će se postavit svojstvo position na absolute, zatim će se definisati pomak u odnosu na drugi element, na primjer pomoću svojstava top i left:

```
article{
    position: absolute;
    height: 100px;
    width: 100%;
    top: 50px;
    left: 200px;
    background-color: darkorange;
}
```

Vrijednost fixed je slična vrijednosti absolute. I ovdje se element pomiče u odnosu na ishodišni element, sa razlikom da ishodišni element može biti samo element body. Kao i kod vrijednosti absolute, prostor koji bi inače zauzimao pomaknuti element neće biti sačuvan. Vrijednost fixed je podržana tek u relativno novijim preglednicima (od Internet Explorer-a 8[41], Firefox-a 30 i Chrome-a 27). Razlika je u tome što je pozicionirani element na ovaj način „fiksiran" na svom položaju i za razliku od ostalih elemenata ostaje na istom mjestu i kada korisnik koristi klizač za pomjeranje sadržaja.

18.5. Pozicioniranje pomoću tipa prikaza flex

Iako je CSS donio velike novine i olakšanja rada sa kolonama, prethodno navedeni tipovi prikaza nisu jedini koji omogućavaju „prelamanje", odnosno pozicioniranje elemenata. Naime, CSS3 omogućava primjenu novog tipa prikaza flex. Naravno, sa tim mogućnostima dolazi i malo komplikovanija postavka, pošto pored samih flex elemenata koji se redaju, mora postojati i tzv. flex kontejner, unutar koga se ti elementi raspoređuju. U ovom tekstu ćemo se baviti flexbox modelom i njegovim upotrebama. Atributi koji se odnose na: **flex kontejner** i **flex elemente**, obrađeni su u posebnim tekstovima. Međutim, prije nego što krenemo na proučavanje samih atributa, moramo da se upoznamo sa nekim osnovnim pojmovima. **Osnovni pravac** (engl. *main axis*) je pravac redanja elemenata. Ovaj pravac ćemo zadati u flex kontejneru i to u smislu da li će se elementi redati po redovima ili kolonama. Osnovni pravac po redovima predstavlja horizontalno redanje elemenata (jedan za drugim), a po kolonama - vertikalno (jedan ispod drugog). **Ukršteni pravac** (engl. *cross axis*) je pravac koji je pod pravim uglom u odnosu na osnovni pravac. Dakle, ako je zadato da se elementi redaju horizontalno (odnosno po redovima), ukršteni pravac je vertikalan (kako idu sami redovi - u koloni) i obrnuto. Na slici su najbolje ilustrovani pravci:

[41] Internet Explorer podržava fiksno pozicioniranje samo ako postoji !DOCTYPE.

Elementima može biti zadat i raspored u **više linija** (engl. *wrap*). Ako je na primjer osnovni pravac horizontalan, elementi će biti raspoređeni jedan za drugim u redu, a ako je potrebno, i u više horizontalnih redova, koji onda idu jedan ispod drugog. Isto važi ako zadamo vertikalni osnovni pravac, možemo imati više kolona, jednu pored druge. Dakle, linije se smještaju jedna za drugom po ukrštenom pravcu.

Flexbox specifikacija postoji već duže vrijeme, tokom kojeg je prolazila kroz revizije, odnosno unapređenje. Web preglednici su uvijek išli u korak sa specifikacijom, pa su dugo vremena nudili makar parcijalnu implementaciju ovog standarda. U vrijeme pisanja ovog teksta možemo reći da je flex box postao zreo standard i da je veoma dobro podržan u web preglednicima, tako da je bezbjedno koristiti ga. Podržan je od Firefoxa 28 i Chrome-a 29, zatim u njihovim mobilnim verzijama, kao i u Safari i Edge preglednicima. Probleme mogu očekivati jedino korisnici koji još uvijek koriste starije verzije Internet Explorer preglednika.

18.5.1. Flexbox kontejner

Da ponovimo, **osnovni pravac** (engl. *main axis*) je pravac redanja elemenata, a **ukršteni pravac** (engl. *cross axis*) je pravac koji je pod pravim uglom u odnosu na osnovni pravac. Dakle, ako je zadato da se elementi redaju horizontalno (odnosno po redovima), ukršteni pravac je vertikalan (po kolonama) i obrnuto. Elementima može biti zadat i raspored u više linija (engl. *wrap*) u koje se smještaju jedna za drugom po ukrštenom pravcu. Već znamo da se flexbox model zasniva na dva tipa elemenata:

1. flex kontejneru (unutar koga se može nalaziti jedan ili više flex elemenata);
2. flex elemenata (ne postoje prepreke da bilo koji flex element bude i flex kontejner).

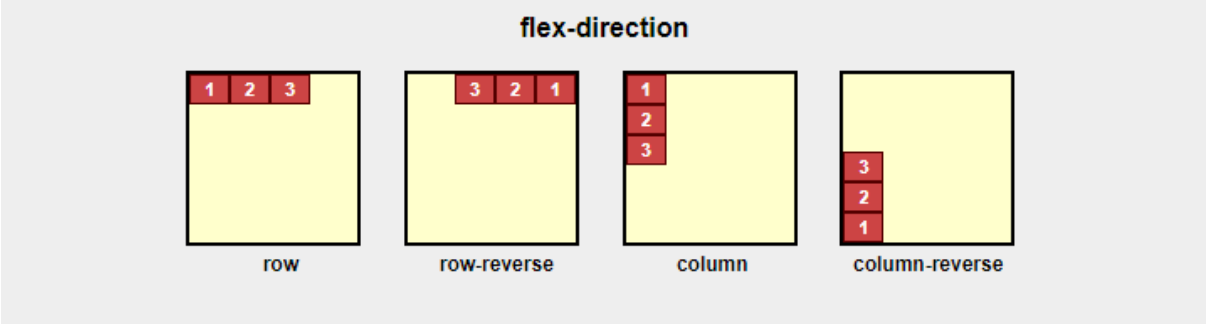
Iako display ne spada u flexbox atribute, bez njega ne možemo. Naime, pomoću ovog atributa definišemo flex kontejner. Postoje dvije vrijednosti koje su nam ovdje značajne:

- flex (element ima sve karakteristike blok prikaza, s tim što postaje flex kontejner);
- inline-flex (element se ponaša kao inline blok element, ali postaje i flex kontejner).

Dakle, flex elementi su *de facto* blokovi koji u normalnom toku elemenata se redaju jedan ispod drugog, dok se inline-flex elementi redaju jedan pored drugog, kao i inline blokovi. Jednom kada element postavimo za flex kontejner, njegovi podelementi (direktno podređeni, ne i elementi unutar elemenata) automatski postaju flex elementi - bez obzira da li su „prirodno“ blok ili inline elementi.

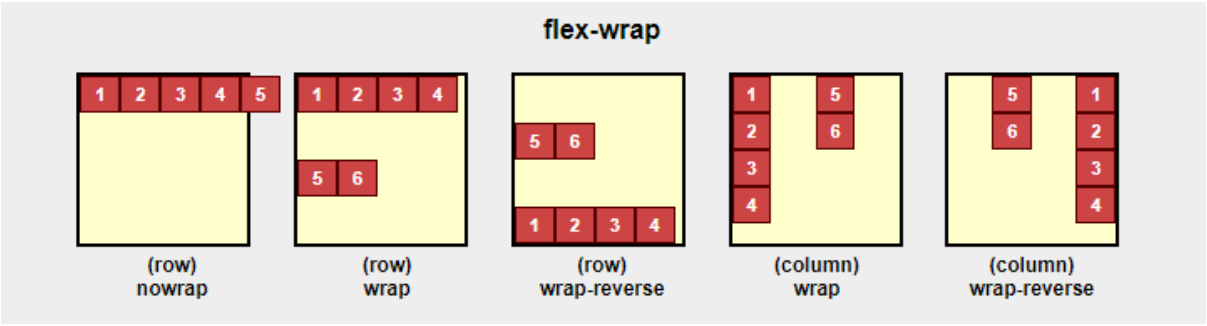
Atributom flex-direction zadajemo osnovni pravac i smjer redanja flex elemenata unutar flex kontejnera. Moguće vrijednosti su:

1. row (elementi se redaju po horizontali, sa lijeve na desnu stranu - inicijalna vrijednost);
2. row-reverse (elementi se redaju horizontalno, ali naopako sa desne na lijevu stranu);
3. column (elementi se redaju vertikalno, sa vrha prema dnu);
4. column-reverse (elementi se redaju vertikalno, ali naopako, od dna prema vrhu).



Atribut flex-wrap je veoma značajan atribut, jer njime definišemo višelinijski prikaz. Samim tim, definišemo i izgled i ponašanje našeg flexbox-a, neki atributi funkcionišu ili ne funkcionišu zavisno od ovog atributa. Moguće vrijednosti ovog atributa su:

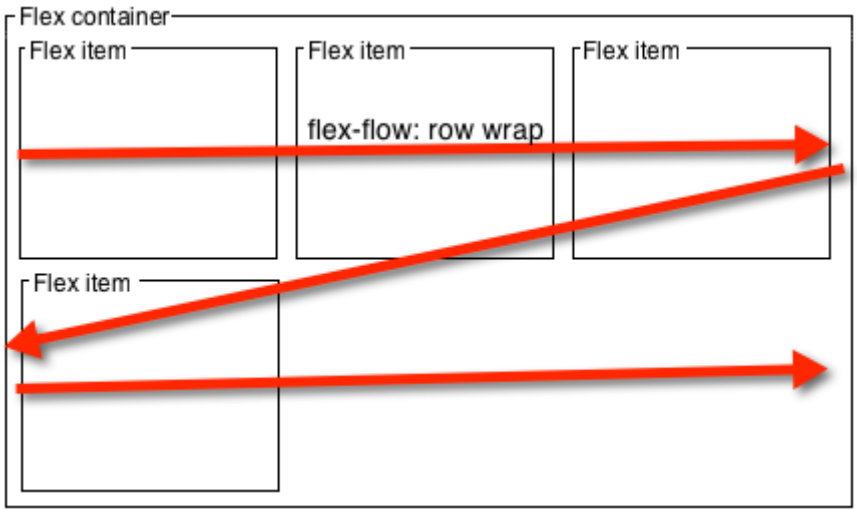
1. nowrap (elementi ne prelaze u slijedeću liniju, tj. kada se stigne do kraja, „ispadaju“ iz flex kontejnera ili se „sabijaju“ ako je tako definisano - inicijalna vrijednost);
2. wrap (flex elementi prelaze u slijedeću liniju, kada „pređu“ dimenzije flex kontejnera);
3. wrap-reverse (elementi prelaze u slijedeću liniju, ali naopako, ako je prikaz u redovima, slijedeći red će biti iznad, a ako je u kolonama, slijedeća kolona će biti lijevo).



Atribut flex-flow je kompleksan atribut koji u suštini zamjenjuje flex-direction i flex-wrap. Može se zadati jedna ili obje vrijednosti. Na taj način, u jednom potezu određujemo osnovni pravac i prikaz u više linija. Atribute flex-flow i flex-direction **ne bismo smjeli da koristimo kao zamjenu za ispravan HTML redoslijed**. Redoslijed elemenata u HTML-u, koji se ne uklapa sa onim što korisnici vide će loše utjecati na čitljivost i razumljivost našeg dokumenta. Npr. aplikacije koje glasovno interpretiraju web stranicu (za slabovidne osobe) će tekst pročitati na pogrešan način.

Na primjer:

```
flex-flow: row wrap;
```

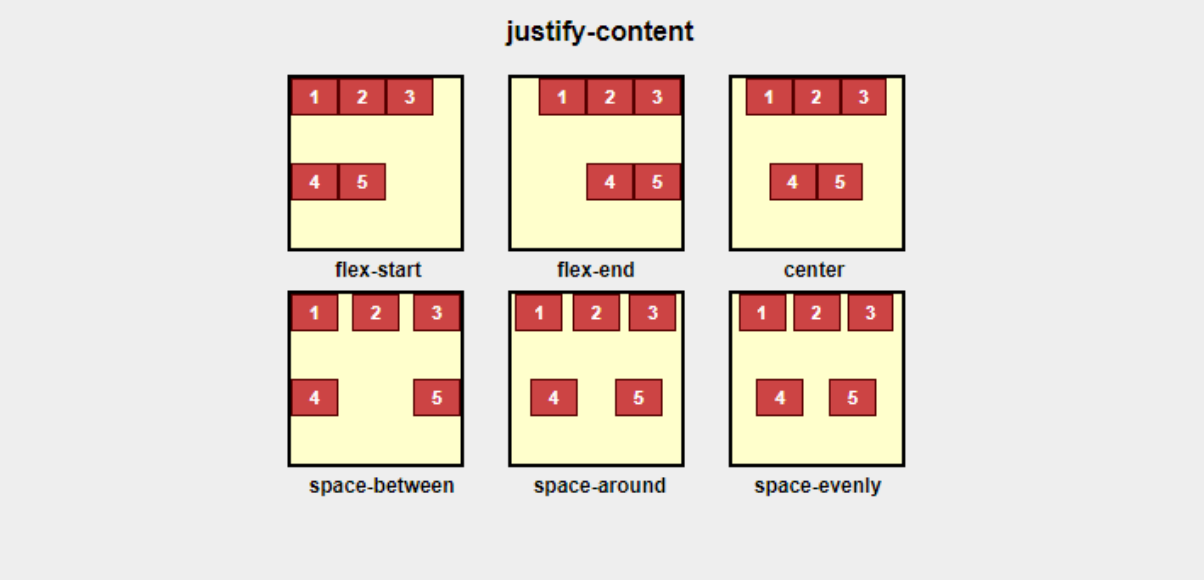


Atribut justify-content predstavlja poravnanje flex elemenata po osnovnom pravcu. Važno je da se sjetimo da osnovni pravac ima i smjer i stranu, tj. „ivicu“ od koje elementi počinju da se redaju, kao i stranu koja bi bila krajnja, tj. prema kojoj se redaju. Za row, početak je lijevo, a za row-reverse je početak desno. Slično, za column, početak je gore, dok je za column-reverse početak dolje. Ovaj uvod nam služi da razjasnimo zašto ovome (i njemu sličnom atributu) ne možemo da zadamo vrijednost npr. left ili right za poravnanje elemenata (kada želimo da radimo sa flexbox-om), zbog toga što se elementi mogu redati sa lijeva na desno ili sa desna na lijevo, nije dobra ideja vezivati se za lijevo i „desno“, već za „početak“ i „kraj“. Vrijednosti koje možemo da koristimo su:

1. flex-start (elementi su poravnati prema početnoj ivici flex kontejnera, tj. po osnovnom pravcu - inicijalna vrijednost);
2. flex-end (elementi su poravnati prema suprotnoj, tj. krajnjoj ivici flex kontejnera, po osnovnom pravcu);
3. center (elementi su centrirani po osnovnom pravcu).

Postoje i tri vrijednosti koje podjednako raspoređuju elemente, a varijante tih vrijednosti su:

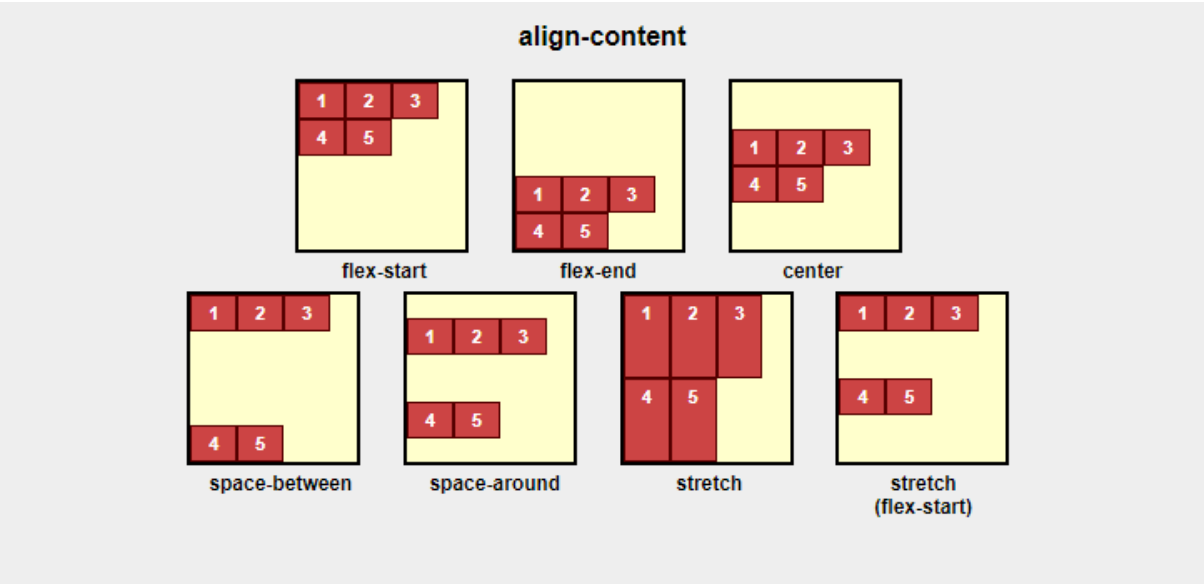
1. space-between (elementi su poravnati po obje strane flex kontejnera, po osnovnom pravcu, od ivice do ivice, sa podjednakim razmacima između njih);
2. space-around (elementi su poravnati po obje strane, sa razmakom između ivica, ali tako da je razmak dvostruko veći između samih elemenata);
3. space-evenly (elementi su poravnati po obje strane, sa razmakom između ivica, ali sada je razmak svugdje podjednako).



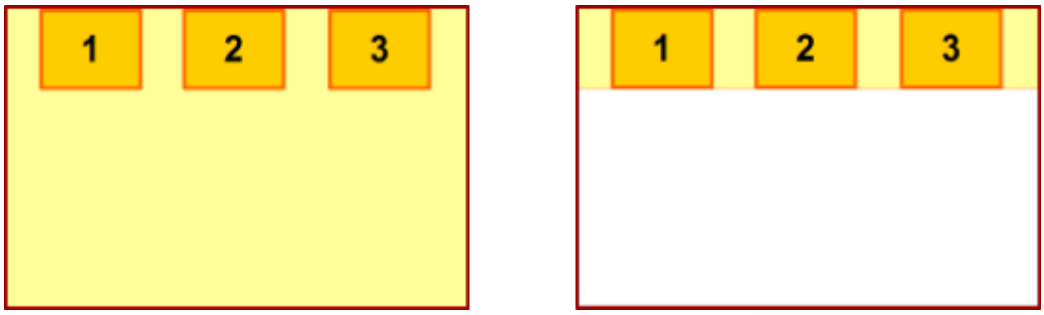
Atribut align-content definiše način na koji će se sve linije poravnati po ukrštenom pravcu. Ovaj atribut se bavi cijelim linijama (redovima ili kolonama). Kako god da se linije rasporede, uvijek „nose“ sa sobom svoje flex elemente, koji mogu biti poravnati po osnovnom (justify-content) i ukrštenom pravcu (align-items) unutar samih linija. Vrijednosti koje atribut align-content može imati:

1. flex-start (sve linije su poravnate uz početak ukrštenog pravca);
2. flex-end (sve linije su poravnate uz kraj ukrštenog pravca);
3. center (linije su centrirane po ukrštenom pravcu);
4. space-between (linije su poravnate po obje strane ukrštenog pravca, a između samih linija postoji podjednak razmak);
5. space-around (linije su poravnate po obje strane ukrštenog pravca, između linija postoji podjednak razmak, kao i između linija i stranica kontejnera);
6. stretch (linije su razvučene po površini flex kontejnera po ukrštenom pravcu - inicijalna vrijednost).

Osim kada važi vrijednost stretch, svaka linija ima dimenziju (po ukrštenom pravcu) najvećeg flex elementa. Vrijednost ovog atributa nema efekta na prikaz ako ne postoji više linija (dakle, obavezan je wrap ili wrap-reverse). Kada višelinijski prikaz nije uključen, jedina postojeća linija je uvijek razvučena preko cijele površine po ukrštenom pravcu. Kada je omogućen višelinijski prikaz (npr. `flex-wrap: wrap`), kada postoji samo jedna linija elemenata, veličina te linije zavisi od ovog atributa. U svim kontejnerima (osim posljednjeg), flex elementi po vertikali pokrivaju cijelu visinu linije, tako da možete vidjeti da su linije samo kod vrijednosti stretch razvučene da pokriju cijelu vertikalu.



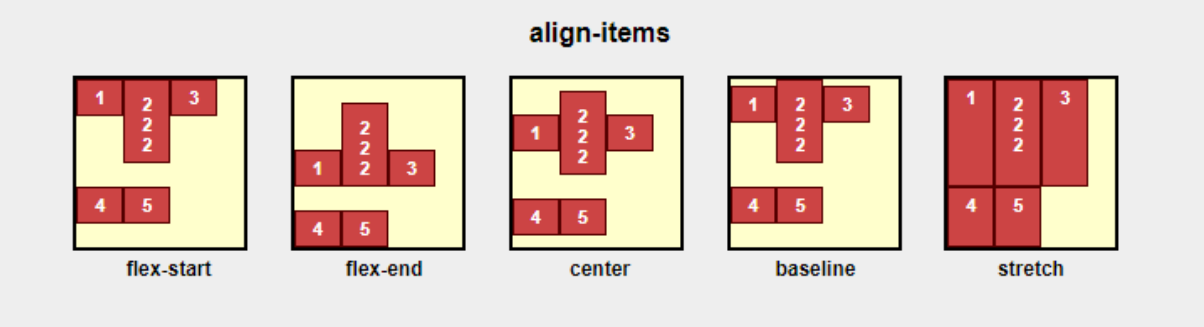
Ako je `align-content: flex-start`, na prvoj slici vidimo kako izgleda linija, kada ne postoji višelinijski prikaz (flex-wrap), a na drugoj kada postoji. Primjetimo razliku, kada nema dovoljno elemenata da se pređe u drugu liniju. U prvom slučaju atributom align-content ništa ne postižemo, a u drugom njime određujemo položaj linije.



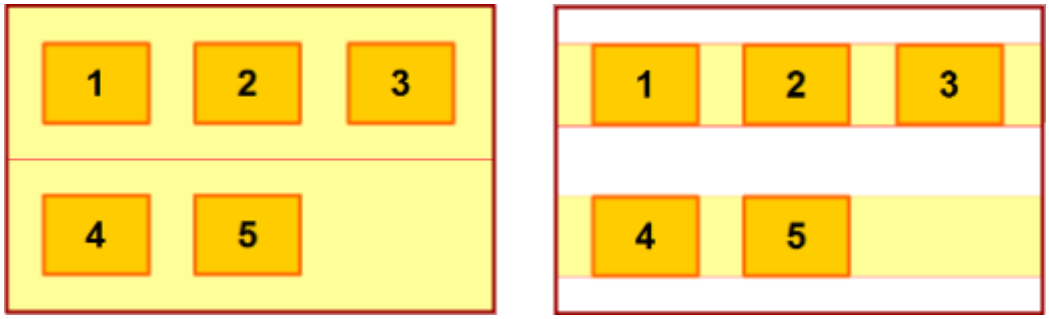
Atribut align-items definiše način na koji će flex elementi biti poravnati unutar svoje linije, po ukrštenom pravcu. Posebno je vidljivo kada objekti imaju različite dimenzije i kada su linije razvučene po ukrštenom pravcu. Moguće vrijednosti su:

1. flex-start (objekti su poravnati uz početak linije po ukrštenom pravcu);
2. flex-end (objekti su poravnati uz kraj linije po ukrštenom pravcu);
3. center (objekti su centrirani unutar linije po ukrštenom pravcu);
4. baseline (objekti su unutar linije poravnati po sadržaju, tj. tekstu);
5. stretch (objekti su razvučeni na cijelu dimenziju linije po ukrštenom pravcu - inicijalna vrijednost).

Teško ćemo vidjeti efekat atributa align-items (ako su objekti jednakih dimenzija), a linije nisu razvučene (align-content nije postavljen na vrijednost stretch).



Da bi smo shvatili kako funkcionišu atributi align-content i align-items kao i razliku između njih, moramo shvatiti razliku između slijedećih slučajeva:



Slučaj kada su linije razvučene da pokriju cijelu vertikalu, te kada su podjednako raspoređene sa razmakom. U prvom slučaju su linije fiksirane (`align-content: stretch`), ali ih možemo da pomjeramo elemente unutar njih (align-items). U drugom slučaju, linije su aranžirane atributom align-content, a elementi unutar njih se mogu pomjerati samo onoliko koliko im visina linija dozvoljava (visina zavisi od najvećeg elementa).

18.5.2. Flexbox elementi

Prije nego što se pozabavimo elementima flexbox-a, posjetit ćemo se terminologije:

Kao što smo rekli, ovdje se bavimo flex elementima, odnosno atributima koji utječu na njihov izgled i raspored unutar flex kontejnera. Naravno, i to su obični elementi, kao i svi drugi. I njima mogu biti zadati: širina, visina, margine, ispuna, pa čak i pozicija. Web preglednik će se držati (koliko je moguće) zadatih dimenzija elementa, ali neke flexbox vrijednosti poput stretch u atributima: align-items, self-align ili flex-basis će imati utjecaja na definisane dimenzije. Zadate margine za flex elemente nekad vrše, a nekad ne vrše utjecaj da se između elemenata pojave razmaci (ako je zadato poravnanje elemenata sa razmakom), dok ispuna uvijek mora da utječe, sa obzirom da se njime mijenjaju dimenzije elemenata. I flex elementi mogu biti apsolutno pozicionirani. U tom slučaju se ne redaju sa ostalim elementima, jer tada „iskaču“ iz normalnog toka elemenata.

Atribut order definiše redoslijed flex elemenata. Naime, ovim atributom možemo postaviti drugačiji redoslijed u odnosu na redoslijed kako su flex elementi navedeni u HTML-u. Dakle, kao vrijednost je moguće navesti:

1. 0 (inicijalna vrijednost);
2. numerik (redoslijed prikaza, očekuje se pozitivan ili negativan cio broj).

Na primjer:

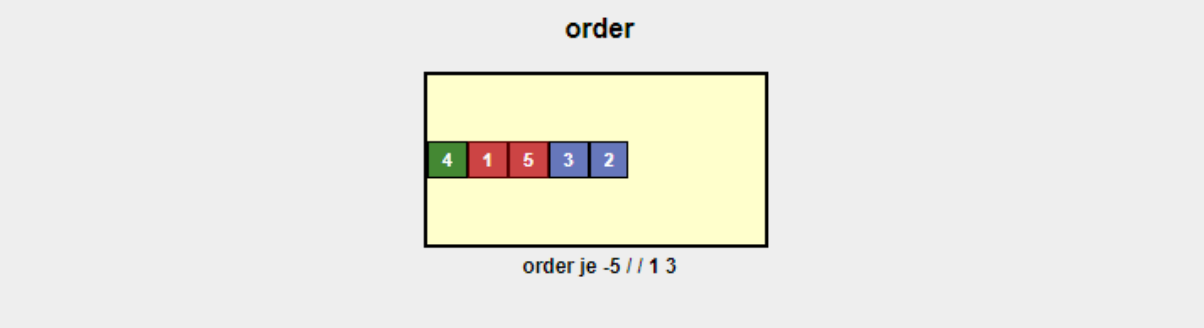
```
order: 3;
order: -5;
```

Kako se redaju elementi sa zadatim order atributom u odnosu na elemente bez ovog atributa?
Pravilo je slijedeće:

- prvo se redaju elementi sa negativnim order vrijednostima;
- zatim se redaju elementi bez order atributa i oni kojima je: `order: 0` (po HTML-ovom redoslijedu);
- na kraju se redaju elementi sa pozitivnim order vrijednostima.

Znači, čak iako bismo imali npr. 100 flex elemenata, i nekom od njih zadamo recimo `order: 5`, on neće biti prikazan na 5. mjestu, već na kraju, pošto se prvo redaju svi elementi koji nemaju zadat order, a tek onda oni koji imaju. Za sve elemente koji nemaju zadatu order vrijednost, podrazumijeva se vrijednost 0. Inače, svi elementi sa jednakom order vrijednošću se redaju po redoslijedu po kojem su navedeni u HTML-u. Isto kao za flex-flow i flex-direction atribute, ne bi smo smjeli da koristimo atribut order umjesto ispravnog HTML redoslijeda. Loš HTML redoslijed će utjecati na čitljivost i razumljivost našeg dokumenta. Dakle, u primjeru vidimo da

je najprije prikazan 4. element, jer ima negativnu order vrijednost, pa se prikazuju 1. i 5. element koji nemaju zadat order, i to po redoslijedu kako su navedeni u HTML-u. Na kraju se prikazuju 3. i 2., i to prema redoslijedu njihovih order vrijednosti.



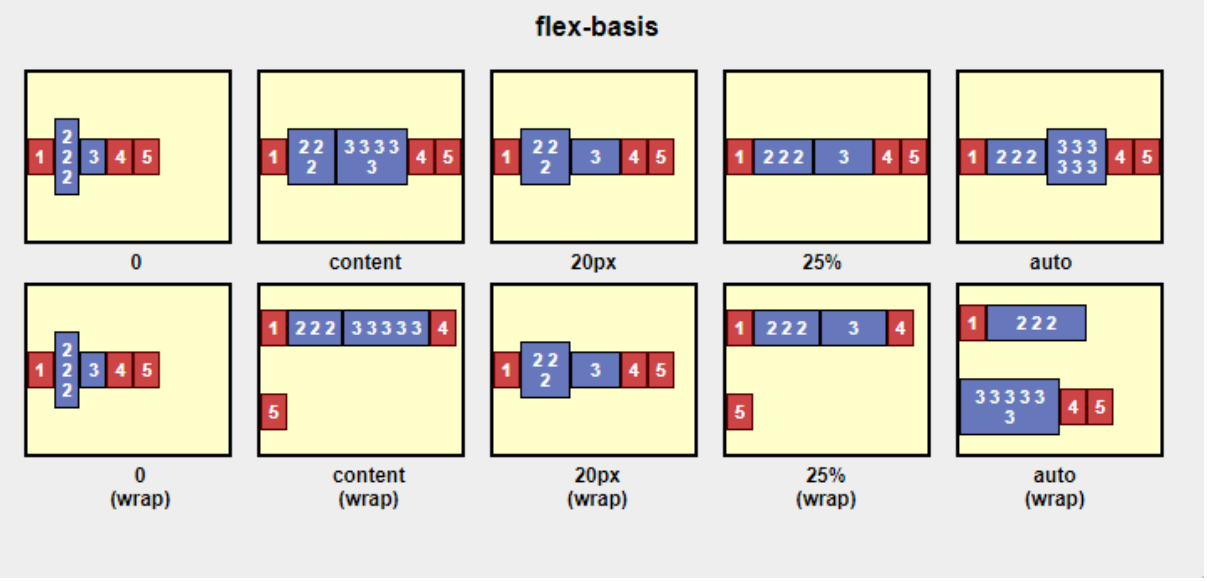
Atribut flex-basis predstavlja inicijalnu veličinu flex elementa po osnovnom pravcu. Znači, ako je osnovni pravac horizontalan, flex-basis označava širinu, a ako je osnovni pravac vertikalan, flex-basis označava visinu elementa. Element će imati tu veličinu (ako je moguće). Vrijednosti koje ovaj atribut može imati su:

1. auto (veličina elementa se preuzima od atributa width ili height; zavisno od glavnog pravca, ako su definisani, a ako ne, onda je vrijednost ista kao da smo zadali content);
2. brojčana vrijednost (elementu se zadaje konkretna veličina u mjernim jedinicama);
3. brojčana vrijednost u procentima (elementu se zadaje relativna veličina u odnosu na flex kontejner);
4. content (element dobija veličinu na osnovu svog sadržaja).

Ako se zada vrijednost auto (ovaj atribut se ne navodi, jer je početna vrijednost auto), gledaju se width ili height. Međutim, ako se „namesti“ da su i dimenzije iz nekog razloga „automatske“, onda se ovaj atribut ne ponaša kako bi se „ponašao“ običan blok, već se veličina elementa određuje na osnovu njegovog sadržaja.

Zašto bismo koristili flex-basis ako je to praktično ista stvar kao kad navodimo width, odnosno height? Dakle, lakše je da imamo u vidu samo jedan atribut veličine po osnovnom pravcu, bez obzira da li je to horizontala ili vertikala, nego da razmišljamo o tome kakav je pravac i da li koristimo width ili height. Posebno što se kasnije osnovni pravac flex kontejnera može promijeniti u programu, a pomoću ovog atributa se sve rješava automatski. Ne zaboravimo da i dalje važi box model! Naime, flex-basis se odnosi na širinu/visinu sadržaja, pa se na tu veličinu dodaje ispuna i ivica. Ovakvo ponašanje se može promijeniti atributom box-sizing. Na primjer:

```
.container-1{
    display: inline-flex;
    flex-direction: row;
    align-items: center;
    align-content: stretch;
  }
  .container-2{
    flex-wrap: wrap;
  }
  .item-1{
    flex-basis: 0;
  }
  .item-2{
    flex-basis: content;
  }
  .item-3{
    flex-basis: 20px;
  }
  .item-4{
    flex-basis: 25%;
  }
  .item-5{
    width: 50px;
    flex-basis: auto;
  }
```



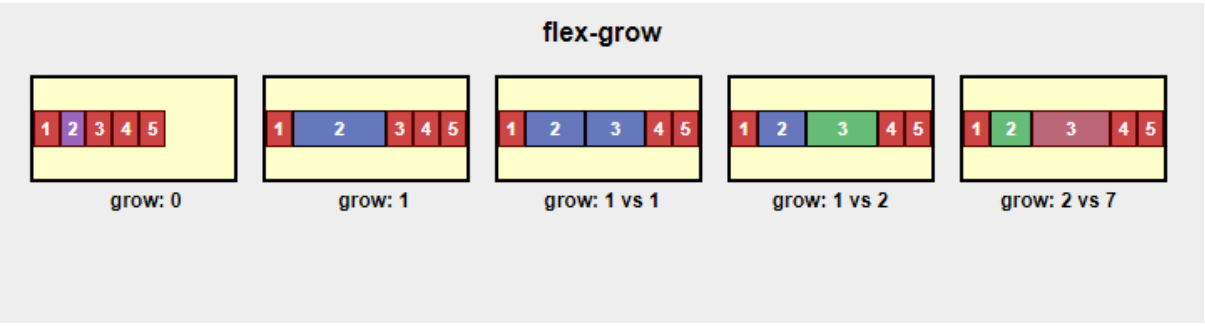
Šta se ovdje sada događa? Zadali smo flex-basis za 2. i 3. element. Također, možemo vidjeti kako se ovi elementi ponašaju u nekoliko različitih situacija, tj. kada nema ili ima prelaska u novi red i kada ima ili nema viška sadržaja. Primjetimo da, kada nije zadan `flex-wrap: wrap` (višelinijski prikaz), flex-basis vrijednost se poštuje samo ako ima dovoljno mjesta u redu. Ovo

u stvari nema puno veze sa atributom flex-basis, već sa fleksibilnosti prikaza flexbox elemenata. Naime, atribut flex-shrink ima inicijalnu vrijednost 1, što omogućava elementima da se „sabiju“ ako treba. Da smo željeli da se isključivo poštuje flex-basis, zadali bismo i `flex-shrink: 0`. Sa druge strane, kada je uključen i višelinijski prikaz, vidimo da se elementi šire onoliko koliko je zadano pošto tada nema razloga da se vrši njihovo „sabijanje“ - kada nema mjesta, jednostavno se pređe u novi red. Vidimo kako vrijednost content zadaje širenje prema sadržaju, a vrijednost auto „prepisuje“ ono što je zadato u atributu width. Kada width ne bi bio zadat, auto bi se ponašao kao content.

Pomoću atributa flex-grow definišemo koliko se veličina elementa može povećati u odnosu na ostale elemente, po osnovnom pravcu. Moguće vrijednosti su:

1. 0 (inicijalna vrijednost);
2. numerik (broj koji označava proporciju proširenja u odnosu na ostale elemente).

Numerik se zadaje bez bilo kakve mjerne jedinice, a dozvoljene vrijednosti su pozitivni brojevi (celi i decimalni). Šta ova vrijednost znači? Sama za sebe ne puno. Ali kada se posmatra u odnosu na druge elemente, označava koliko puta taj element može biti uvećan u odnosu na njih. Ako je svim elementima zadata ista flex-grow vrijednost, onda će svi imati istu veličinu. Dakle ako elementu zadamo `flex-grow: 2`, to može da znači da je element duplo veći od ostalih, ako svi ostali imaju `flex-grow: 1`. Ali isto tako može da znači da je element jednak drugima, ako i svi ostali imaju `flex-grow: 2`.



Slično kao flex-grow, atribut flex-shrink definiše koliko se veličina elementa može smanjiti u odnosu na ostale elemente (po osnovnom pravcu). Moguće vrijednosti su:

1. 1 (svi elementi se mogu podjednako smanjiti - inicijalna vrijednost);
2. numerik (broj koji označava proporciju umanjenja u odnosu na ostale elemente).

Isto kao i kod flex-grow, vrijednost je samo broj bez bilo kakve merne jedinice, a dozvoljene vrijednosti su pozitivni brojevi - cijeli i decimalni. Ako element ima zadatu flex-shrink vrijednost 0, element se neće smanjivati već će zadržati originalnu veličinu. Kada web preglednik odlučuje kako da smanji elemente, on množi flex-shrink faktor sa flex-basis vrijednošću i tako dobija proporciju za umanjenje elemenata kada se raspodjeljuje negativni prostor[42]. To znači da će se „veći“ elementi smanjivati brže od „manjih“. Negativan prostor može postojati samo ako ne postoji višelinijski prikaz. Ako elementi imaju mogućnost da pređu u slijedeću liniju, onda i ne postoji potreba da se smanjuju. Ako na primjer zadamo `flex-wrap: wrap` i `flex-shrink` neće funkcionisati. Atribut flex generalno definiše fleksibilnost elementa. Ovaj atribut je kompleksan i objedinjuje flex-grow, flex-shrink i flex-basis atribute. U svom osnovnom obliku, atribut flex se zadaje kao:

```
flex: grow shrink basis;
```

Postoji značajna razlika kada zadamo flex atribut i kada zadajemo svaki atribut pojedinačno. Kao što možemo zadati samo neke od tri navedena atributa, tako i u flex možemo zadati samo neke vrijednosti. Međutim, podrazumijevane (inicijalne) vrijednosti se razlikuju kada ne zadamo neki atribut i kada ne zadamo neku flex vrijednost. Kombinacije atributa koje se prosljeđuju su slijedeće:

1. initial (tumači se kao 0 1 auto - inicijalna vrijednost);
2. numerik numerik metrička vrijednost (elementu se zadaju faktori uvećanja i umanjenja i početna veličina u mernim jedinicama);
3. numerik numerik content (zadaju se faktori, a veličina je određena sadržajem);
4. numerik numerik auto (zadaju se faktori, a veličina je ono kako je zadato u flex-basis atributu; ako ga ima; inače je opet po sadržaju);
5. auto (tumači se kao 1 1 auto);
6. none (tumači se kao 0 0 auto).

Nisu uvijek potrebne sve tri vrijednosti, pa smo dali moguće kombinacije i kako se one tumače. Naravno, kada se zadaje atribut, za grow i shrink navodimo numerike, a za flex-basis vrijednost navodimo metriku ili definisanu vrijednost. Ako se faktor uvećanja ili faktor umanjenja izostave,

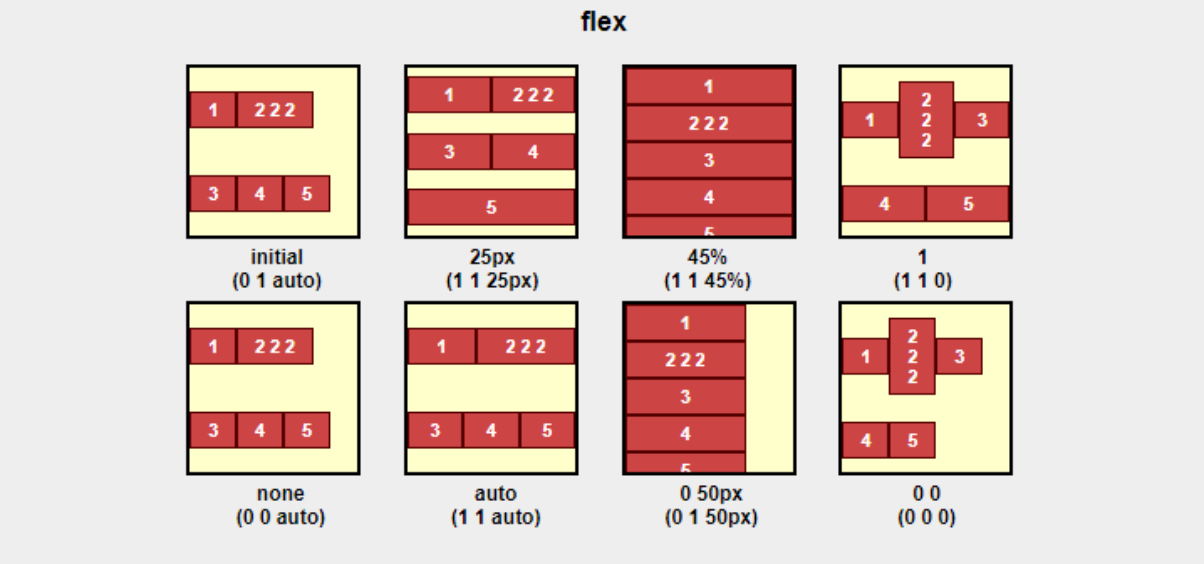
[42] **Negativan prostor** je prostor koji nedostaje, ako elementi po svojoj veličini prevazilaze veličinu linije. Ako imamo negativan prostor, gleda se faktor umanjenja (flex-shrink), pa se elementi proporcionalno sužavaju kako bi se smjestili u liniju. **Pozitivan prostor** je prostor koji ostaje slobodan kada se elementi smjeste u liniju. Ako je zadat faktor uvećanja (flex-grow), elementi se proporcionalno šire da zauzmu taj slobodan prostor. Dakle, prostor koji imamo na raspolaganju je ili pozitivan (ima viška) ili negativan (ima manjka) - ne može biti i jedno i drugo.

njihova podrazumevana vrijednost je 1. Kada se navede `flex: auto`, to predstavlja vrijednost `flex: 1 1 auto`, jer se auto tumači kao flex-basis vrijednost. Ako se izostavi flex-basis, njegova podrazumevana vrijednost je 0. Prema W3C preporuci, uvijek je bolje koristiti atribut flex, nego zadavati pojedinačno atribute flex-grow, flex-shrink i flex-basis. To je zato što flex atribut po svojim inicijalnim vrijednostima „inteligentno“ podešava atribute koji nisu zadati prema tipičnim slučajevima. Iz toga proizilazi da ćemo korištenjem atributa flex prije doći do rezultata koji smo zamislili nego podešavanjem sva tri atributa. Na primjer:

```
.container{
    display: inline-flex;
    flex-direction: row;
    flex-wrap: wrap;
    align-content: stretch;
    align-items: center;
    overflow: hidden;
}
.item-1 > *{
    flex: initial;
}
.item-2 > *{
    flex: 25px;
}
.item-3 > *{
    flex: 45%;
}
.item-4 > *{
    flex: 1;
}
.item-5 > *{
    flex: none;
}
.item-6 > *{
    flex: auto;
}
.item-7 > *{
    flex: 0 50px;
}
.item-8 > *{
    flex: 0 0;
}
```

Napravili smo nekoliko primjera za atribut flex. Vidimo koliko različitih rezultata dobivamo, a flex je u svakom okviru primijenjen na sve elemente. Šta bi se sve postiglo kada bi smo flex primjenjivali na pojedine elemente. Ovo ilustruje koliko je flexbox prikaz fleksibilan. Upravo radi toga kod pisanja grow i shrink faktora ne idemo van zadavanja vrijednosti 0 i 1. Nema svrhe

zadavati veće brojeve ako se vrijednost primjenjuje na sve elemente - veličina nije bitna kada elementi imaju podjednake grow i shrink faktore. Kada bismo primjenjivali flex na pojedinačne elemente, različiti grow i shrink faktori bi utjecali na različite veličine elemenata.

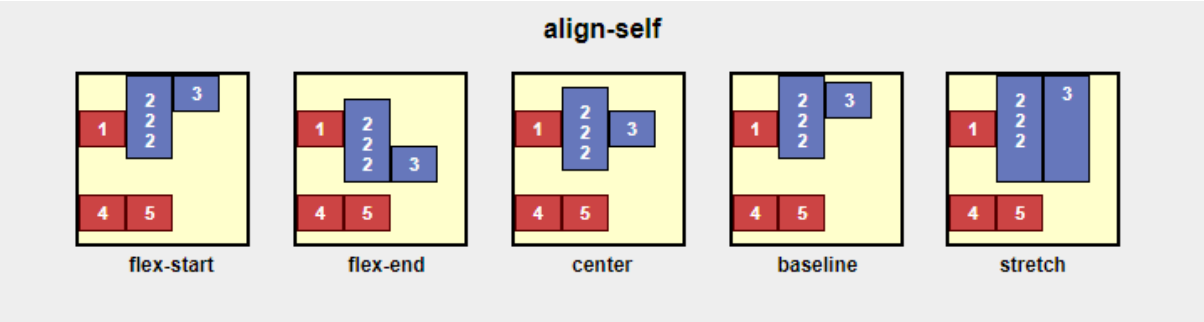


U nastavku ćemo objasniti ponašanje svakog od 8 gore navedenih flex kontejnera i pripadajućih flex elemenata:

- initial - računa se kao da smo zadali 0 1 auto, elementi se ne šire, mogu se suziti ako fali prostora (ne dešava se jer imamo višelinijski prikaz), a veličina im je prema sadržaju;
- 25px - kada zadamo samo flex-basis, kao da je 1 1 25px, što znači da će elementi dobiti širinu od 25px, ako ima slobodnog mjesta, raširit će se da ga popune;
- 45% - opet je zadat samo flex-basis, kao da je 1 1 45%, elementi preširoki da bi stalo više njih u liniju, pa zato imamo samo po jedan, potpuno raširen element po redu;
- 1 – ovo je flex-grow, posmatra se kao 1 1 0, inicijalna širina je minimalna, a onda se svi šire podjednako;
- none - iako se ova vrijednost računa kao 0 0 auto, u ovom slučaju izgleda isto kao initial, jer flex-shrink ne utječe, ne možemo vidjeti razliku u smanjivanju elemenata, jer imamo višelinijski prikaz;
- auto - zadavanje samo flex-basis, isto kao 1 1 auto, elementi prvo dobivaju širinu prema sadržaju, a onda se šire da popune prostor;
- 0 50px - zadavanje numerika i metričke vrijednosti, isto kao da smo zadali flex-grow i flex-basis, što se računa kao 0 1 50px, elementi dobivaju podjednaku fiksnu veličinu, ali se ne šire, a zbog višelinijskog prikaza se ne sužavaju;
- 0 0 - dva numerika su flex-grow i flex-shrink, tumači se kao 0 0 0, širina elemenata će biti minimalna, a elementi se neće ni širiti ni sužavati.

Atribut align-self definiše način na koji će pojedini flex element biti poravnat unutar svoje linije, po ukrštenom pravcu, ovaj atribut radi istu stvar kao align-items, ali se odnosi na pojedinačne elemente. Moguće vrijednosti su:

1. auto (element je u ovom slučaju poravnat po osnovu vrijednosti align-items atributa - inicijalna vrijednost);
2. flex-start (element je poravnat uz početak linije po ukrštenom pravcu);
3. flex-end (element je poravnat uz kraj linije po ukrštenom pravcu);
4. center (element je centriran unutar linije po ukrštenom pravcu);
5. baseline (element je unutar linije poravnat po sadržaju, odnosno tekstu);
6. stretch (element je razvučen na celu dimenziju linije po ukrštenom pravcu).



U poglavlju je obrađeno:

- korištenje pozicioniranja elemenata pomoću tipa prikaza;
- korištenje pozicioniranja pomoću svojstva float;
- korištenje pozicioniranja elemenata pomoću svojstva postion;
- korištenje pozicioniranja elemenata pomoću tipa prikaza flexbox;

XIX Uključivanje vanjskih stranica

Po završetku poglavlja, polaznik će moći:

- kao uključiti drugu web stranicu ili dio web stranice koristeći element iframe.

HTML omogućuje da se kao dio web-stranice uključi neka druga web stranica ili dio iste. To bi se moglo opisati i kao „prozor“ koji gleda na drugu stranicu. Jedan od uobičajenih načina na koji se vanjske stranice uključuju je uključivanje Google mape na stranicu. Uključiti se može bilo koja stranica, bilo sa vlastitog web-sjedišta ili sa sjedišta trećih lica (koje se nalazi na drugom poslužitelju).

19.1. Uključivanje vanjske stranice – <iframe> element

HTML element koji se koristi za uključivanje vanjske stranice je iframe. Ovaj element može imati slijedeće atribute:

1. src – atribut kojem se dodjeljuje URL stranice koja će se prikazati u okviru;
2. height – određuje visinu okvira u pikselima;
3. width – određuje širinu okvira u pikselima;
4. seamless – omogućava korisniku kretanje unutar sadržaja u iframe elementu u slučaju da je on veći od okvira.

HTML5 više ne podržava atribute scrolling i frameborder. Scrolling je atribut koji određuje hoće li okvir imati klizne trake u slučaju da je sadržaj veći od okvira ili ne. Može imati vrijednosti true, false ili auto (trake će se pojavljivati automatski kada je to potrebno). Frameborder je atribut koji određuje hoće li imati okvir ili ne. Može imati vrijednosti 0 ili 1. Npr.:

```
<iframe
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d2876.960050289103!2
d18.394157565416748!3d43.85665669726861!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.
1!3m3!1m2!1s0x4758c923a27c8f8b%3A0xcea818135826c8ac!2sKampus+Univerziteta+u+Sa
rajevu%2C+Zmaja+od+Bosne%2C+Sarajevo+71000!5e0!3m2!1shr!2sba!4v1557499162908!5
m2!1shr!2sba
width="600"
height="450"
frameborder="0"
style="border:0"
allowfullscreen>
</iframe>
```

U poglavlju je obrađeno:

- uključivanje drugih web stranica ili dijela web stranice koristeći element iframe.

XX Responzivni dizajn

U poglavlju je obrađeno:

- uključivanje drugih web stranica ili dijela web stranice koristeći element iframe.

Tradicionalno, prvi korak koji bi grafički dizajner trebao napraviti (pri kreiranju web sjedišta), je da odlučiti da li će pri dizajniranu koristit apsolutne jedinice. Sa obzirom da web sajta moraju svoj sadržaj prilagoditi različitim veličinama sadržaja, za prilagođavanje različitim formatima se koriste tačke prekida (engl. *breakpoints*). Glavno pitanje za većinu dizajnera je: Da li je danas za izradu grafike uopće praktično koristiti fiksni format. Stvarna veličina je ono što je najvažnije, a ne razlučivost zaslona pri dizajniranju.

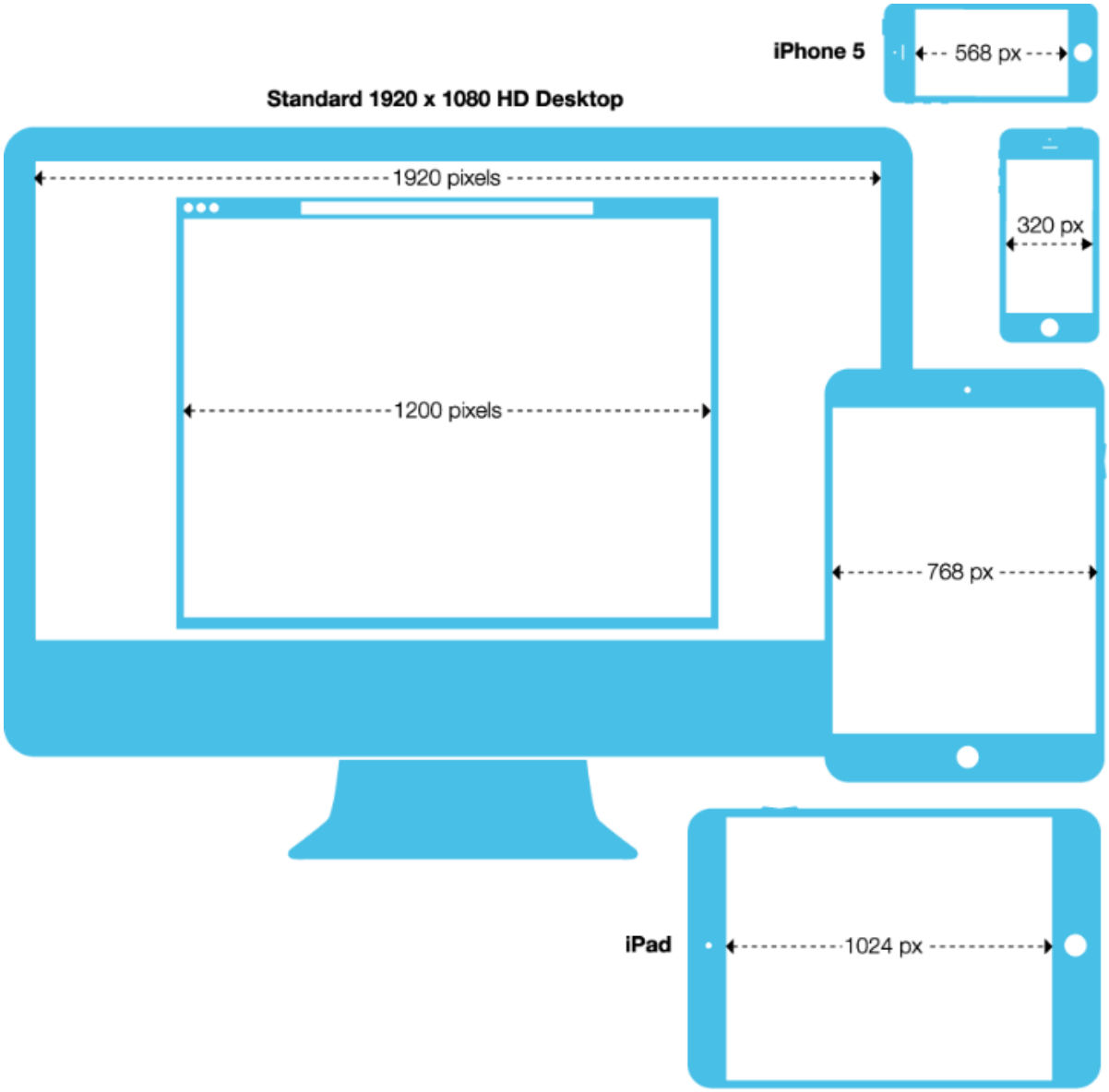
20.1. Korištenje hardverskih piksela za dizajniranje responzivnih stranica

Prilikom izrade web sajta, W3C predlažemo korištenje 3 različite veličine:

1. PC desktop – širina veća od 1200 piksela;
2. Tablet – širina veća od 768 piksela;
3. Pametni telefon – širina veća od 320 piksela.

Postavlja se pitanje: Zašto baš ova tri formata? Odgovor je prilično jednostavan, jer je rezolucija[43]: 1920 x 1080, 1600 x 900 i 1368 x 768 najčešća HD desktop veličina. Fizička širina tableta nije veća od 768 piksela (npr. 768 x 1024 kod iPad-a), a najmanja veličina za koju bi sadržaj trebao biti dizajnirati je fizička širina najmanjeg pametnog telefona, odnosno širina od 320 piksela (npr. 320 x 480 kod iPhone 3). Međutim, mi zapravo dizajniramo za 5 formata (kada uzmemo u obzir da se mobilni uređaj može koristiti vertikalno ili horizontalno). Nekada, CSS pikseli nisu adekvatne jedinice za rješenje problema formiranja grafike. Ako na primjer dizajniramo pomoću uobičajenog zaslona sa radnom površinom, odabir CSS piksela kao jedinice za formiranje grafike je savršen. Ako naš računar ima Retina Screen[44] (npr. Apple iMac 21.5"), 4K ili 5K rezoluciju, moramo biti oprezni, jer naši CSS pikseli mogu biti prikazani na kao hardverski pikseli, što za posljedicu ima prikazivanje sadržaja u manjoj rezoluciji od očekivane. Na primjer, kada radim na Mac-u u Adobe Illustrator-u, bez obzira što koristimo Retina zaslon ili običan zaslon, čini se da su grafike iste veličine, ali kad koristimo Adobe Photoshop na retina zaslonu, grafika je dva puta veća u poređenju sa običnim zaslonom.

[43] Na linku: www.mydevice.io možete provjeriti rezoluciju vaših uređaja.
[44] **Retina Display** je termin koji je definisao Apple, a odnosi se na tehnologiju ekrana visokih rezolucija premijerno predstavljenu na iPhone 4 uređaju (u junu 2010. godine). Riječ retina u slobodnom prijevodu znači mrežnjača (mrežnjača oka).



20.2. Kako radi responzivni raspored?

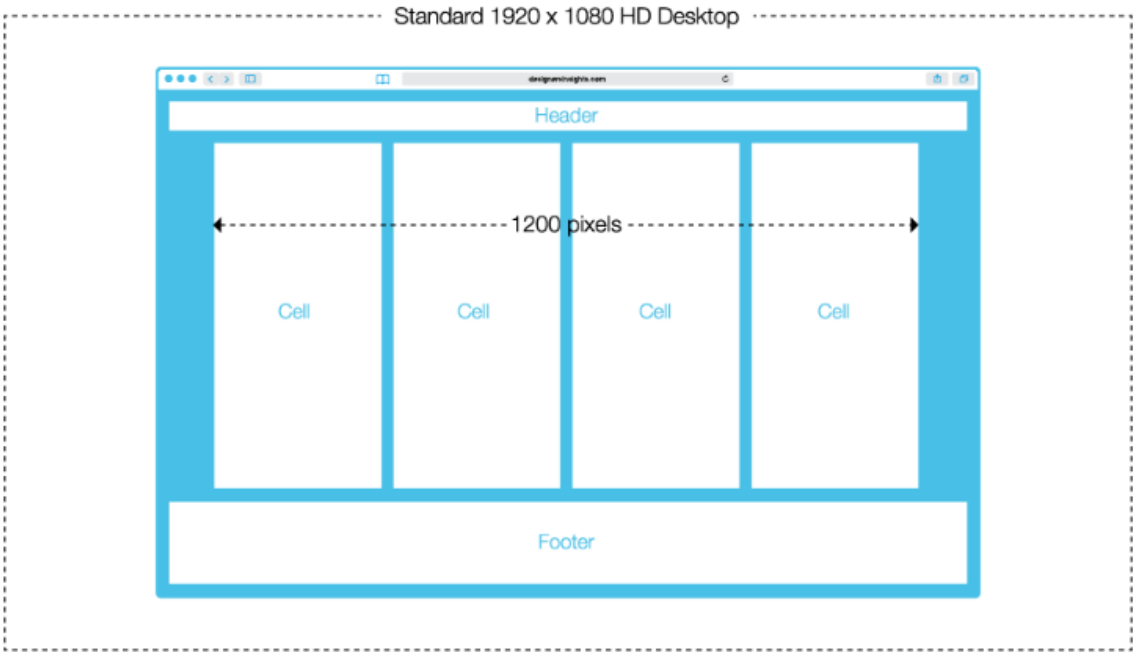
Prilikom izrade responzivnog web sjedišta, imamo za cilj da sadržaj rasporedimo na način da bude što fleksibilniji. Umjesto postavljanja rasporeda (engl. *layout*) fiksne širine, u ovom slučaju moramo odrediti tačke prijeloma (engl. *breakpoints*) u kojima se mijenja raspored, a sadržaj pozicionira na drugi način. Da bi smo testirali responzivnost našeg web sjedišta, nisu nam potrebni nikakvi specijalizovani alati. Nakon što se učita web stranica, ručno, pomoću miša ćemo smanjivati (ili povećavati) veličinu prozora pretraživača. Smanjivanjem prozora ćemo doći do prve tačke prekida (trenutka u kome je trenutna veličina prozora pretraživača ista, odnosno jednaka definisanoj veličini tačke prekida), kada se dostigne tačka prekida; a sajt je responzivan; mijenja se vizualni prikaz sadržaja, tj. mijenja se način prikazivanja HTML elemenata. Nastavimo li dalje da smanjujemo veličinu prozora preglednika, dolazimo do slijedeće tačke prekida, u kojoj

se način prikaza sadržaja, još jednom mijenja. Posljednja tačka prekida je ona koja se definiše za veličinu pametnog telefona. Pored ovih karakterističnih tačaka prekida, mogu postojati i tačke prekida koje se nalaze između pomenutih, ovo zavisi od sadržaja koji se prikazuje. Kao budući web dizajneri profesionalci, trebali bi smo koristiti Google Chrome inspektor, besplatan profesionalni alat koji se nalazi u izborniku alata za razvojne programere preglednika Google Chrome. U osnovi, prilagodljivi rasporedi sadržaja se uvijek svodi se na dvije tehnike:

1. identifikovanje prijelomne tačke, gdje se izgled sadržaj mijenja;
2. proporcionalno skaliranje sadržaja između tačaka prekida.

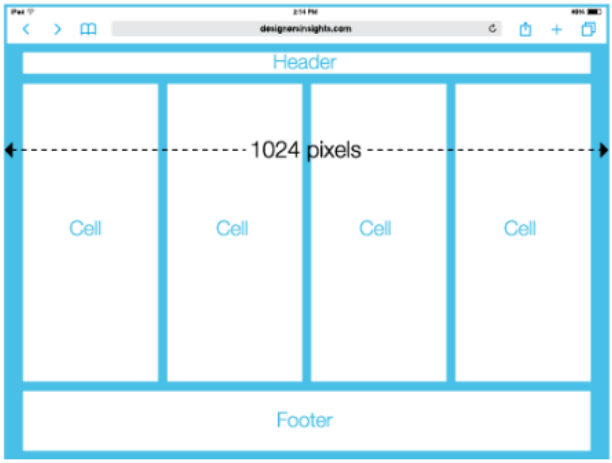
20.3. Desktop raspored

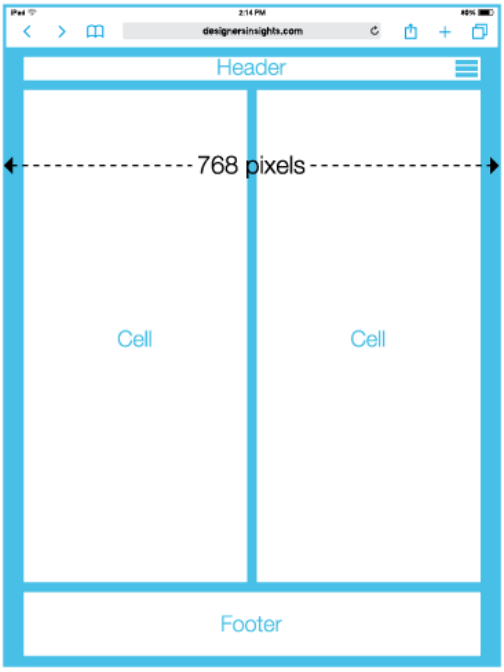
Prva stvar kojom se trebamo pozabaviti je da li naš format radne površine treba da koristiti maksimalnu širinu zaslona ili ćemo odrediti da maksimalnu dostupna širina bude 1200 piksela? Na ovo pitanje ne postoji tačan (ili netačan) odgovor, sve zavisi o sadržaju web stranice i viziji dizajnera. Odgovor koji se sam nameće je da širina sadržaja treba da bude jednaka širini prozora. Ako na primjer imamo sadržaj širine 970 piksela na zaslonu širine 1920 piksela, onda to znači da će polovina zaslona biti prazna, ako uz to dizajn ne bude zadovoljavajući, ovo može da izgleda grozno. Ponekad dizajneri skaliraju na način da „visina“ sadržaja zauzima cijelu širinu prozora preglednika, ali širina sadržaja ostaje ograničena na 1200 piksela. Na taj način web sajt i dalje odgovara bilo kojoj veličinu prozora preglednika, a glavni sadržaj ostaje grupisan u sredini, kako bi ostao dobro organizovan i dizajniran. U nastavku ćemo posmatrati tzv. raspored sa 4 ćelije (kolone), za uređaje sa širinom od 1200 hardverskih piksela. Ovaj raspored se koristi sve dok je širina veća od 768 CSS piksela.



20.3. Tablet raspored

Prilikom dizajniranja rasporeda za tablet uređaj, regularne veličine tableta treba smatrati malom radnom površinom u horizontalnom položaju, zato što ima širinu od (najviše) 1024 CSS piksela, dok mini tablet treba smatrati velikim telefonom. Važno je da naš responzivni dizajn razvijemo u prilagodljivu mrežu sadržaja. Prva točka prijeloma trebala bi biti za veličinu tableta sa načinom prikaza portret (vertikalno) ili kada zaslon preglednika dostigne širinu od 768 CSS piksela. Na 768 piksela, također bi smo trebali razmisliti da počnemo koristiti meni, popularno nazvan po hamburgeru (engl. *hamburger menu*); zbog svoje strukture (sadržaj se reda jedan ispod drugog). Raspored sa 2 ćelije (kolone) se može koristiti za mini tablet sa načinom prikaza horizontalno, ali kada mini tablet prikazuje sadržaj vertikalno, trebali bi prikazivati 1 ćeliju po redu. U našem primjeru ćemo koristiti iPad tablet. Kada se gleda vodoravno, ima raspored od 4 ćelije (kolone), ali kada se gleda vertikalno, dostigne tačku prekida, pa prelazi na izgled od 2 ćelije.

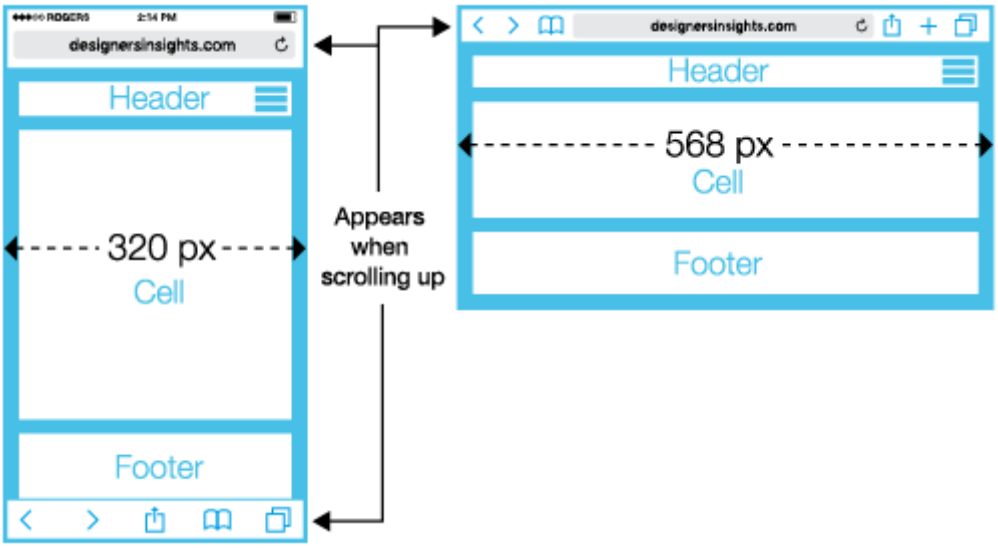




20.4. Telefon raspored

Format telefona je daleko najzahtjevniji format za izradu, jer može biti vrlo mali. Na telefonu se pokazalo da najbolje funkcioniše dizajn jedne ćelije (kolone), jer se vodi se računa da tipografija bude dovoljno velika i čitljiva, a sadržaj dovoljno mali. Interaktivni objekti (dugmad i sl.) trebaju da budu veličine prosječnog prsta. Prinuđeni smo da za navigaciju koristite hamburger meni. Dizajnirajte mora da bude praktičan, da koristi sve funkcije koje dolaze sa modernim telefonom, kao što su karte i sl. Omogućavanje interakcije, dijeljenja sadržaja na društvenih medija se danas očekuje. Izbjegavanje iznimno duge stranice koja se „nikada“ ne završava narušava harmoniju

dizajna. Uzmimo u obzir da će prozor preglednika biti veći kada se sadržaj pomiče prema dolje i manji kada se sadržaj pomiče prema gore, radi menija preglednika, koji će se pojaviti na ekranu. Također, uzmimo u obzir da se pametni telefoni isporučuju u različitim formatima, a najmanji format je iPhone 3, danas mnogi telefoni imaju veličinu iPhone 6 Plus, koji je gotovo mali format tableta. Dakle, dizajnirajte prema tim graničnim formatima. Za primjer smo uzeli iPhone 5:



20.5. Alat za testiranje responzivnosti

Google Chrome inspektor je besplatan, profesionalni alat koji web programerima i dizajnerima omogućava da testiranje web sajtova obavljaju na svojim računarima. Nalazi se u izborniku alata za programere (engl. *Developer Tool Menu*) u pregledniku Google Chrome. Prednost ovog alata je da će simulirati web stranicu, onako kako će se pojaviti na bilo kojem uređaju. Ovaj alat je vrlo koristan za testiranje veličine fonta, jer neki uređaji će učitati i prikazati drugačije. Ali, ovaj alat nikada ne bi trebao zamijeniti testiranje na uređajima, jer Google Chrome inspektor ima ograničenja i ne možemo očekivati da će biti 100% tačan. Sa druge strane, to je brz i učinkovit način testiranja web stranice.

20.6. Odabir prijelomnih tačaka

Odabir prekida za web sjedišta je oduvijek bio zbunjujući za grafičkog i web dizajnera, jer postoje dva glavna konfliktna, odnosno pristupa za odabir prijelomnih tačaka. Odabir tačaka prijeloma (engl. *breakpoints*) nije posao grafičkog dizajnera, već web dizajnera. Također, postoje programeri koji u svom radu objedinjuju teoriju grafičkog dizajna i web development-a.

Tačke prekida (engl. *breakpoints*) su *de facto* širine preglednika. Kada naša web stranica reaguje na određenu širinu prikaza, nje izgled sadržaj se prilagođava novom formatu ekrana. Na primjer, stranica će mijenjati izgled od 1200 piksela do 768 piksela širine. Prosječno, svaki web sajt ima najmanje dvije uobičajne tačke prekida. Jedna je tablete, a druga za mobilne uređaje. Odabir standardne tačke prekida će funkcionisati za većinu web stranica, ali uvijek postoje slučajevi kada je promjena izgleda potrebna. Na primjer određivanje tačaka prekida za uređaje kada su u vertikalnom ili horizontalnom načinu prikaza. Budući da dva web sjedišta nikada nisu dizajnirana na isti način, korištenje standardnih prekidnih tačaka možda neće uvijek raditi savršeno. Stoga je važno utvrditi gdje i kada vršiti prilagođavanje (unutar raspona standardnih tačaka prekida). Da ponovimo, svaki put kada dizajniramo, stvari nisu uvijek kristalno jasne. Kao što smo rekli, postoje dva glavna konfliktna pristupa za odabir tačaka prekida:

1. Postavljane prijelomnih tačaka na osnovu veličine zaslona različitih uređaja ili pomoću najpopularnijih uređaja (Samsung, Apple, LG, itd.);
2. Odabir prekidnih tački na temelju sadržaja, nikada na osnovu uređaja, jer će se veličina tih uređaja u budućnosti stalno mijenjati; dakle, web dizajner bi trebao tačke prekida trebao odrediti tako da počne sa najmanjom mogućom širinom (i visinom) koja se može pojaviti, pa onda da sadržaj prilagođava većim širinama, odnosno visinama, pa će onda biti siguran da je dizajn fluidan.

Preporuka Google-a je da se koristit 2. pristup. Dakle, Google daje slijedeće sugestije:

1. odabir tački prekida se počinje sa odabirom najmanjih;
2. optimizirajte tekst za čitanje;
3. nikada ne skrivajte sadržaj;
4. prijelomne točke na temelju sadržaja, a ne na temelju rezolucije uređaja;
5. dizajn za najmanji mobilni uređaj je na prvom mjestu;
6. 70 do 80 karaktera po redu.

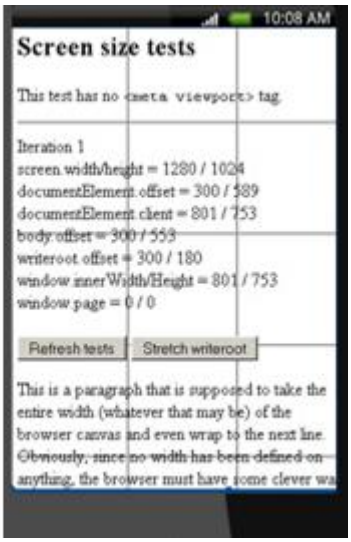
20.7. Početni prikaz (engl. *Viewport*)

Tipično web sjedište optimizirano za mobilne uređaje sadrži nešto poput slijedećeg:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Svojstvo width kontroliše veličinu okvira za prikaz. Može se postaviti na određeni broj piksela na primjer `width=600` ili na posebnu vrijednost device-width, širinu ekrana u CSS pikselima na skali od 100% (postoje odgovarajuće vrijednosti height i device-height, koje mogu biti korisne za stranice sa elementima koji mijenjaju veličinu ili položaj na temelju visine prikaza). Svojstvo

intitial-scale kontroliše nivo zumiranja kada se stranica prvi put učita. Svojstva maximum-scale, minimum-scale i user-scale kontrolišu način po kojem korisnici mogu povećavati ili smanjivati stranicu. Na lijevoj slici možemo vidjeti kako izgleda prikaz na stranici na koju nismo uključili meta tag, a na desnoj slici vidimo istu stranicu na kojoj je uključen pomenuti meta tag.



20.8. Media upit (engl. *Media query*)

Media upit su korisni kada želimo da izmijenim web sjedište ili aplikaciju zavisno od vrsti uređaja ili specifičnog parametra (kao što je rezolucija ekrana ili širina okvira preglednika preglednika). Upiti za medije se koriste za slijedeće:

1. Uslovno promjenu CSS stila (@media i @import pravila);
2. Za odabir multimedijalnog sadržaja za elemente kao što su: link, source i ostale;
3. Za testiranje i praćenje stanja medija pomoću različitih JavaScript metoda.

Ako želimo da se pozadina body elementa promijeni kada je širina prozora preglednika manja od 600 piksela, navest ćemo slijedeće pravilo:

```
@media only screen and (max-width: 600px) {
    body{
    background-color: lightblue;
    }
 }
```

Literatura

1) Ben Frain: „HTML5 i CSS3 – prilagodljiv web dizajn“; Kompjuter biblioteka; Beograd, 2014.

2) Ed Tittel, Chris Minnick: „Beginning HTML5 & CSS3 for Dummuies“; John Wiley & Sons, Inc.; New York, 2013.

3) Ethan Marcotte: „Responsive Web Desing“; A Book Apart; New York, 2011.

4) J. D. Gauchat: „HTML5, CSS3 i JavaScript“; Mikro knjiga; Beograd, 2014.

5) Jon Duckett: „HTML&CSS design and build websites“; John Wiley & Sons, Inc.; Indianapolis, 2011.

6) prof. dr. Nenad Kojić: „WEB DIZAJN: HTML, CSS i JavaScript“; Univerzitet Singidunum; Beograd, 2017.

7) dr. sc. Alen Šimec: „Osnove HTML, XHTML i CSS“; Tehničko veleučilište u Zagrebu; Zagreb, 2011.

8) dr. sc. Alen Šimec: „Website programming and optimization in the HTML5 environment“; Tehničko veleučilište u Zagrebu; Zagreb, 2016.

9) prof. emeritius dr. Terry Ann Felke - Morris: „Web Development and Desing Foundations with HTML5“; Harper Collage; Illinois, 2016.

10) Ashley Menhennett: „A Guide to HTML5 and CSS3“; Zenva Pty Ltd; Brisbane, 2014.

11) Rob Crowther: „Hello! HTML5 & CSS3“; Manning Publications Co; Shelter Island, 2013.