

Visoka škola za informacijske tehnologije

Oblikovanje web stranica
skripta

Jurica Đurić

Zagreb, 2013

Sadržaj

1	Povijest web dizajna	10
1.1	Prvi dizajni	11
1.2	Programi za izradu web stranica	13
2	Strategija dizajna	14
3	Dizajn sučelja.....	15
3.1	Pravila dizajna	16
3.1.1	Jednostavan dizajn	16
3.1.2	Sadržaj stranice	17
3.1.3	Čitljivost teksta.....	17
3.1.3.1	Fontovi.....	17
3.1.3.2	Boje	19
3.1.4	Jednostavnost korištenja – navigacija	20
3.1.4.1	Horizontalna navigacija	20
3.1.4.2	Vertikalna navigacija	20
3.1.5	Podrška za sve preglednike.....	21
3.1.5.1	Mobilni web – mobile web.....	22
3.1.6	Ažuriranost stranice	23
3.1.7	Ključne riječi.....	23
3.1.8	Pružanje informacija korisniku.....	23
3.1.9	Razlog za vraćanje na stranicu	24
3.1.10	Testiranje	24
4	Oblikovanje stranica	25
4.1	Dizajn pomoću frame-ova.....	25
4.2	Dizajn pomoću tablica	25
5	Multimedija	26
5.1	Formati za slike.....	26
5.1.1	Razlike rasterske i vektorske grafike	26
6	CSS	28
6.1	Povezivanje HTML-a i CSS-a	28
6.1.1	Linijski način pisanja stilova	28

6.1.2	Pisanje CSS-a unutar oznake <style>	28
6.1.3	Pisanje stilova u vanjskoj datoteci	29
6.2	Pravila CSS-a.....	31
6.3	Grupiranje selektora.....	32
6.4	Nasljeđivanje svojstva	32
6.6	Svojstva za uređivanje teksta.....	34
6.6.1	Boja teksta	34
6.6.2	Veličina teksta	34
6.6.3	Debljina slova	34
6.6.4	Oblik slova.....	35
6.7	Svojstva za uređivanje elemenata.....	35
6.7.1	Pozadinska boja elemenata	35
6.7.2	Obrub elemenata	35
6.7.3	Margina.....	36
6.7.4	Padding	36
6.7.5	Širina i visina elemenata	37
6.7.6	Svojstvo overflow	37
6.7.6.1	Overflow: visible	38
6.7.6.2	Overflow:auto	38
6.7.6.3	Overflow: hidden	39
6.7.6.4	Overflow:scroll.....	39
6.7.7	Postavljanje pozadinske grafike elementa	40
6.8	Model kutije (Box model)	41
6.9	Klasa i ID selektori	42
6.9.1	Klasa.....	42
6.9.2	ID atribut.....	43
6.10	Elementi <div> i	44
6.11	Pseudoklase i pseudoelementi	45
6.12	Pseudoelementi	46
6.13	Selektori atributa.....	47
6.14	Specificity	48
6.14.1	Important.....	49

6.15 Razmještaj elemenata	50
6.15.1 Izrada stranice koristeći jedan stupac	52
6.15.2 Izrada stranice koristeći dva stupca	55
6.15.2.1 Svojstvo float i clear	58
6.15.3 Izrada stranice koristeći tri stupca	59
6.16 Pozicioniranje elemenata	62
6.16.1 Static	62
6.16.2 Absolute	62
6.16.3 Relative	62
6.16.4 Fixed	63
6.16.5 z-indeks	63
6.16.6 Display	66
6.16.6.1 None	66
6.16.6.2 Block	67
6.16.6.3 Inline	67
6.16.7 Visibility	68
6.17 CSS3	70
6.17.1 Oznake proizvođača	70
6.17.2 Svojstvo border-radius	71
6.17.3 Svojstvo text-shadow	72
6.17.4 Svojstvo box-shadow	73
6.17.5 RGBA boje	73
6.17.6 HSLA boje	74
6.17.7 Svojstvo font-face	74
6.17.8 Gradijenti	75
6.17.9 Višestruke pozadinske grafike	77
6.17.10 Višestruki stupci	77
6.17.11 Transformacija elemenata	78
6.17.11.1 Rotacija elemenata	78
6.17.11.2 Promjena veličine	79
6.17.11.3 Promjena položaja	80
6.17.12 Prijelazi	81

6.17.13	Animacije.....	83
6.17.14	Korisničko sučelje	84
6.17.14.1	Promjena veličine.....	84
7	HTML.....	85
7.1	Općenito o HTML-u	85
7.2	HTML5.....	86
7.2.1	Novi doctype i skup znakova elementi.....	87
7.2.2	Novi elementi.....	87
7.2.2.1	Novi media elementi.....	89
7.2.2.2	Novi <canvas> element	89
7.2.2.3	Novi elementi forme	89
7.2.2.4	Izbačeni elementi.....	89
7.2.3	Multimedija i grafika	90
7.2.3.1	Canvas	91
7.2.3.1.1	Koordinate canvasa.....	91
7.2.3.2	Audio element	93
7.2.3.3	Video element	94
7.2.3.4	Dodatni elementi	95
7.2.3.4.1	<meter>.....	95
7.2.3.4.2	<progress>	96
7.2.3.5	Novi elementi unutar forme.....	96
7.2.3.5.1	Input Type: color.....	96
7.2.3.5.2	Input Type: date	96
7.2.3.5.3	Input Type: datetime-local	97
7.2.3.5.4	Input Type: email	97
7.2.3.5.5	Input Type: number	98
7.2.3.5.6	Input Type: range	98
7.2.3.5.7	<datalist>	99
7.2.3.5.8	<keygen>	99
7.2.3.5.9	<output>	100
7.2.3.5.10	<fieldset>	100
7.2.3.5.11	<figcaption>	101
7.2.3.5.12	<optgroup>	101
7.2.3.6	Atributi forme	102

7.2.3.6.1	required.....	102
7.2.3.6.2	autofocus.....	103
7.2.3.6.3	autocomplete.....	103
7.2.3.6.4	novalidate.....	104
7.2.3.6.5	multiple	104
7.2.3.6.6	placeholder.....	104
7.2.4	Offline i pohrana	105
7.2.4.1	Manifest datoteka.....	107
7.2.4.1.1	CACHE MANIFEST.....	107
7.2.4.1.2	NETWORK	107
7.2.4.1.3	FALLBACK.....	108
7.2.4.2	Resursi u aplikacijskom međuspremniku	110
7.2.4.3	Stanja aplikacijskog međuspremnika	111
7.2.5	Razlika lokalne pohrane u odnosu na međuspremnik preglednika.....	111
7.2.6	Web storage.....	111
7.2.6.1	LocalStorage	112
7.2.6.2	SessionStorage.....	113
7.2.6.3	Web SQL.....	114
7.2.7	Drag&drop	115
7.2.8	Web Workers	116
7.2.9	Server Sent Events – SSE.....	118
7.2.10	Geolokacija.....	120
7.2.10.1.1	Povratne vrijednosti funkcije getCurrentPosition()	121
7.2.11	Izrada igrica pomoću HTML5	123
8	Javascript	124
8.1	Funkcije i događaji JavaScripte	125
8.2	Manipulacija HTML elementima	125
8.3	Izrazi i varijable.....	127
8.3.1	Varijable	127
8.3.2	Tipovi podataka.....	127
8.4	JavaScript objekti.....	128
8.4.1	Kreiranje objekta	128
8.4.2	Pristupanje metodama	129

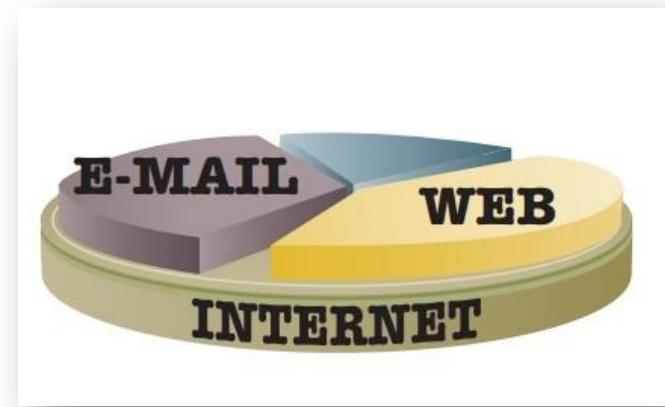
8.5	JavaScript funkcije	129
8.6	Vidljivost varijabli	130
8.7	Operatori.....	130
8.8	Uvjetni izrazi.....	131
8.9	Petlje.....	132
8.10	JavaScript greške	132
8.11	JavaScript validacija	133
8.12	Regуларни изрази	134
8.13	DOM.....	135
8.13.1	Dohvaćanje elementa.....	135
8.13.1.1	Dohvaćanje elementa po id-u.....	135
8.13.1.2	Dohvaćanje elementa po imenu	136
8.13.1.3	Dohvaćanje elementa po klasi	136
8.14	Događaji.....	136
8.15	Kreiranje i brisanje elemenata	138
8.16	JavaScript BOM	138
8.16.2	Window objekt.....	139
8.16.3	Window screen objekt.....	139
8.16.4	Window Location	139
8.16.5	Window history objekt	139
8.16.5.1	History.back().....	139
8.16.5.2	History.forward().....	140
8.16.6	Window Navigator	140
8.16.7	JavaScript poruke (Popup Boxes)	141
8.16.7.1	Upozorenje – alert box	141
8.16.7.2	Potvrda – confirm box	141
8.16.7.3	Upit - (prompt box)	141
8.16.8	JavaScript vremenski događaji (Timing events).....	142
8.16.8.1	Metoda setInterval()	142
8.16.8.2	Metoda clearInterval()	143
8.16.8.3	Metoda setTimeout()	143
8.16.9	Cookies.....	144
8.17	JavaScript biblioteke.....	145

9	XML.....	145
9.1	XML i HTML	145
9.2	Pravila XML-a	146
9.3	XMLHttpRequest.....	146
10	AJAX.....	147
10.1	Slanje zahtjeva serveru.....	148
10.1.1	GET i POST.....	148
10.1.1.1	GET	149
10.1.1.2	POST	149
10.2	Odgovor servera.....	150
10.2.1	ResponseText.....	150
10.2.2	ResponseXML.....	150
10.3	onreadystatechange događaj	150
11	jQuery.....	151
11.1	jQuery instalacija	152
11.2	Osnove jQuery-a.....	153
11.2.1	jQuery selektori.....	153
11.2.2	jQuery funkcije	154
11.2.3	jQuery događaji.....	154
11.3	Napredne opcije jQuery-a.....	155
11.3.1	jQuery efekti	155
11.3.1.1	jQuery hide() i show()	155
11.3.1.2	jQuery toggle()	156
11.3.1.3	jQuery fading	156
11.3.1.3.1	fadeIn()	156
11.3.1.3.2	fadeOut()	157
11.3.1.3.3	fadeToggle().....	157
11.3.1.3.4	fadeTo()	158
11.3.1.4	jQuery sliding	158
11.3.1.4.1	slideDown().....	158
11.3.1.4.2	slideUp()	159
11.3.1.4.3	slideToggle().....	159
11.3.1.5	jQuery animation	159

11.3.1.6	Manipulacija više svojstava	160
11.3.1.7	Korištenje relativnih vrijednosti.....	161
11.3.1.8	Korištenje predefiniranih vrijednosti	161
11.3.1.9	Metoda stop()	161
11.3.1.10	Povezivanje više akcija/metoda.....	162
11.3.2	JQuery HTML DOM.....	162
11.3.2.1	Dodavanja sadržaja	164
11.3.2.1.1	append()	164
11.3.2.1.2	prepend().....	165
11.3.2.1.3	before().....	165
11.3.2.1.4	after().....	166
11.3.2.2	Brisanje elemenata/sadržaja	166
11.3.3	CSS klase	168
11.3.4	jQuery HTML DOM – Dimensions	170
11.3.5	jQuery AJAX.....	171
11.3.6	Metoda noConflict()	173
12	Optimizacija stranice za tražilice	174
12.1	Optimizacija stranica	175
12.1.1	Odabir ključnih riječi.....	175

1 Povijest web dizajna

Samo 20-tak godina nakon što je izmišljen, World Wide Web ili www ili web kako se još naziva promijenio je naše živote u svakom pogledu. Promijenio je način rada, način učenja, način življena. Prvobitno web nije bio zamišljen za ono što ga mi danas koristimo tako prije daljnog učenja treba se upoznati sa prošlosti web-a. Web treba razlikovati od interneta. Web je samo jedan od servisa koji se koristi na internetu slično kao i e-mail ili FTP ili bilo koji drugi servis.

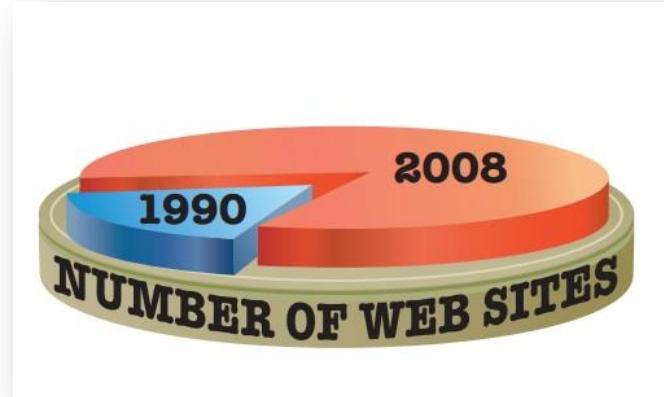


1990. godine u ¹CERN-u engleski inženjer i znanstvenik Tim Berners-Lee i belgijski znanstvenik Robert Cailliau rade na projektu koji je kasnije nazvan World Wide Web ili www ili web te predlažu da se koristi hipertekst za povezivanje i dohvaćanje informacija. 1994. Tim Berners-Lee napušta CERN i osniva World Wide Web Consortium (W3C), međunarodno tijelo koje nadgleda razvoj standarda za Web kako bi stranice izrađene na takav način bile dostupnije korisnicima Interneta. HTML, CSS su neki od standarada koje nadzire W3C. Uz te standarde W3C nadzire i promovira nove grafičke, video i zvučne formate koji se koriste na web-u.



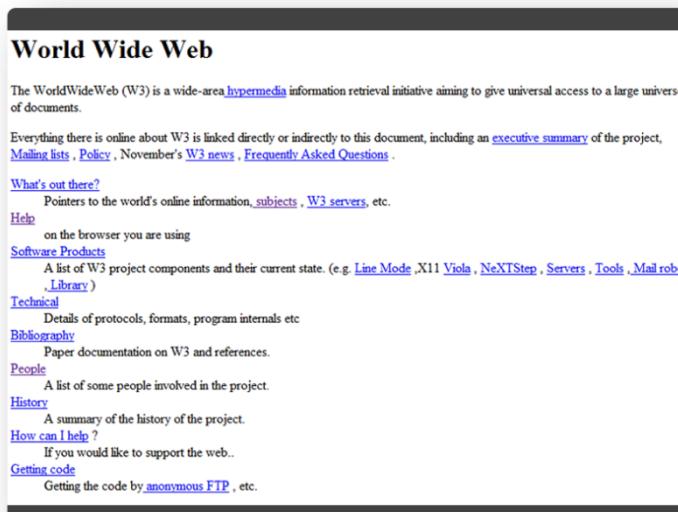
¹ Conseil Européen pour la Recherche Nucléaire

Broj web stranica je vrlo teško točno definirati no većina procjena govori o nekoliko bilijuna. 2008. godine Google je objavio kako je indeksirao jedna trilijun jedinstvenih web adresa. Kako je prošlo tek 20 godina od osnivanja web-a razvoj je stvarno vrlo velik.



1.1 Prvi dizajni

Prve web stranice nisu koristile prebogati grafički dio, razloga je bilo nekoliko. Jedan od razloga je bio taj što prve web stranice su se koristile samo za prikaz tekstualnih informacija te samim time potreban za multimedijskim sadržajem nije postojala.



Slika 1. Prva web stranica

Prva web stranica je bila stranica koja je sadržavala samo tekst i nekoliko poveznica na druge stranice. Kasnije većina drugih stranica je izgledala vrlo slično sa dodatkom jednostavne grafike. Dalnjim razvojem postojala je sve veća potreba za uvođenjem raznih multimedijskim sadržajem. Kako u to vrijeme osim nerazvijenosti web-a nije bila razvijena niti infrastruktura koja je mogla podržati prijenos multimedijskog sadržaja multimedije je u početku bila vrlo malo korištena. No dalnjim razvojem tehnologija multimedija je postala osnovni dio web-a.



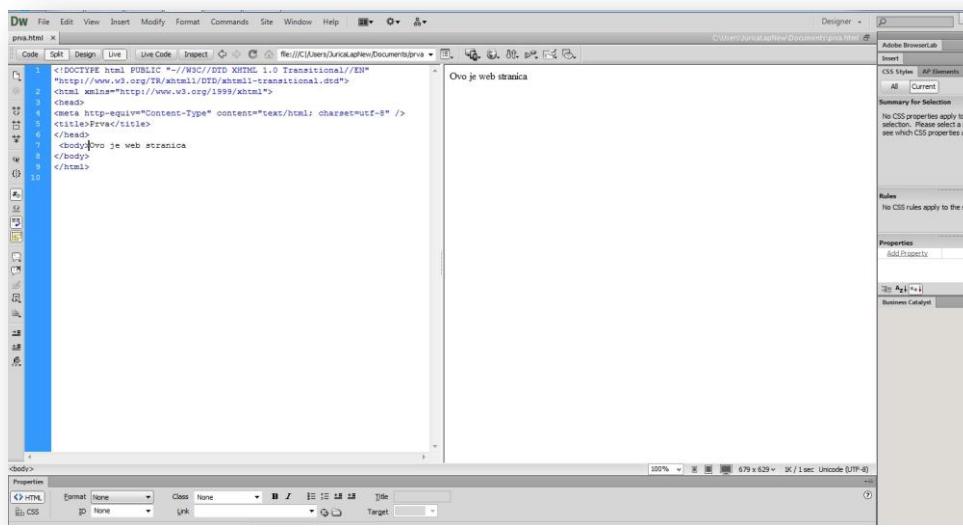
Slika 2. Web stranica Amazon-a



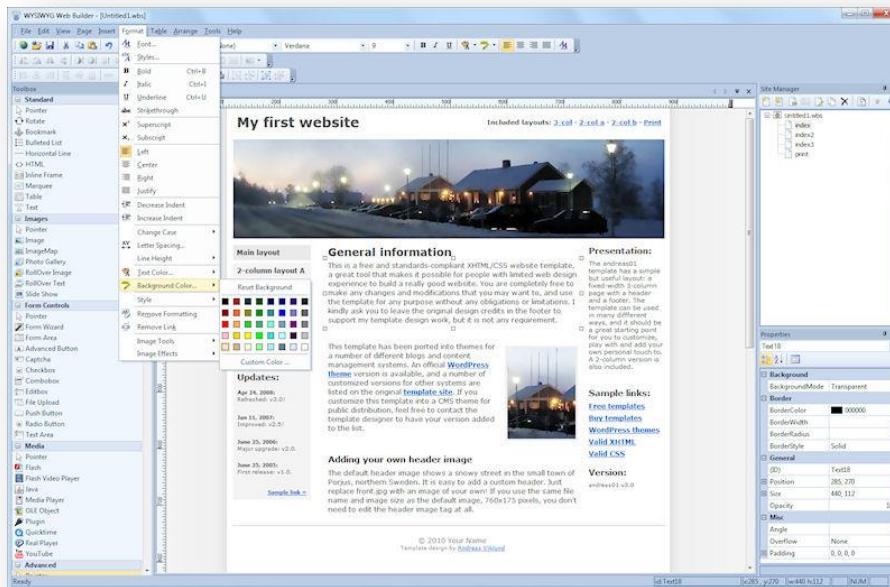
Slika 3. Stranica Google-a iz 1997. godine

1.2 Programi za izradu web stranica

Neki od programa za izradu web stranica su Adobe Dreamweaver kojeg je razvila tvrtka Macromedia, no 2005. postaje dio Adobe Creative Suite paketa. DreamWeaver pruža i prikaz dizajna i prikaz kôda te mješoviti prikaz. Pomoću njega moguće je vidjeti kako će stranica izgledati u pregledniku prilikom izrade bez potrebe otvaranja preglednika.



Također postoji niz raznih programa koji omogućavaju izradu i dizajn web stranica pomoću gotovih predložaka. Jedan od takvih programa je **WYSIWYG** (eng. **What You See Is What You Get**) Web Builder koji omogućava vrlo jednostavno dizajniranje web stranica i ljudima koji nemaju prevelikog znanja u području izrade web stranica. Naravno takve programe mogu koristiti i oni koji posjeduju napredna znanja izrade web stranica.



Programi koji omogućavaju brzu izradu web stranica ne znače da su najbolji izbor prilikom

izrade web stranica. Većina njih automatski ubacuje kôdove za neke dijelove stranica koje kasnije mogu usporiti očitavanje stranice ili donijeti probleme prilikom održavanja stranica. Iz tog razloga najbolje je ovisno o potrebama izrade web stranica donijeti pravilnu odluku s kojim programom kreirati web stranicu. Ponekad je puno bolje i kvalitetnije napraviti web stranicu koristeći program tipa Notepad-a iako je u startu potrebno više vremena no na kraju se pokaže da je brže jer sav kôd koji se napiše zna se gdje je i koju svrhu na web stranici ima.

2 Strategija dizajna

Web stranica (eng. Web page) je HTML dokument koji omogućava prezentaciju teksta i poveznica, a dostupan je preko svojeg URL adrese. Na web stranici uz tekst moguće je prikazivati i razne multimedejske sadržaje, slike, zvukove, video i drugi sadržaji.

Web sjedište (eng. Web site) je mjesto (direktorij) na računalu (serveru) na kojem se nalazi skup web stranica koje su povezane međusobno preko poveznica te čine jednu cjelinu.

Prije izrade samih web stranica prvo je potrebno isplanirati web sjedište kako kasnije kada se krene u izradu ne bi došlo do nepotrebnih problema. Planiranje se izvodi kroz 4 koraka koja su prikazana sljedećom slikom.



Prilikom izrade web sjedišta prvi korak je definiranje ciljeva u kojem se određuju ciljevi izrade web sjedišta te namjena istog. U drugom koraku razmatra se ciljana publika koja će primarno koristiti web sjedište. Ovisno o ciljanoj publici zasniva se daljnji dizajn stranica jer nije isti dizajn ako želi privući mlađa publiku ili starija, isto tako nije isto ako se radi o web sjedištu neke tvrtke ili ako se radi o web sjedištu koji služi za razmjenu primjerice fotografija. U ovom koraku također je vrlo bitno saznati i tehničke detalje vezano za način spajanja na web sjedište kako bi se znalo u kolikoj mjeri je moguće implementirati multimedijijske sadržaje. Ako ciljana publiku koristi vrlo spore veze za spajanje na internet bilo kakva multimedija na web stranici bi učinila stranicu vrlo teškom za učitavanje i pregled. Kako su danas brzine spajanja na Internet i po nekoliko desetaka puta brže u odnosu na vrijeme kada su nastale prve web stranice multimedijijski sadržaj danas spada u nezaobilazni dio svake web stranice. Internet preglednici su također jedan od mogućih problema te se prilikom ovog koraka mora paziti koji će preglednik koristiti ciljana publiku kako se ne bi koristile tehnologije koje ti preglednici ne podržavaju. Sljedeći korak u razvoju web sjedišta je prikupljanje informacija u kojem se prikupljaju svi potrebni podaci kako bi se bez ikakvih problema u potpunosti kreirati web sjedište. U tom koraku potrebno je prikupiti sve tekstove, sve slike i ostale multimedijijske sadržaje kako bi se u sljedećem koraku moglo pristupiti što prije. Sljedeći korak u razvoju je kreiranje idejnog rješenja web sjedišta. U tom koraku se stvaraju mogući izgledi web sjedišta koji se prezentiraju naručiteljima te ovisno o njihovim zahtjevima ili se prepravlja ili s kreće u finalnu izradu cijelog web sjedišta. Naravno u ovom koraku je vrlo bitno držati se pravila grafike i tipografije kako bi konačno web sjedište bilo funkcionalno i kako bi služilo svojoj svrsi. Također u ovom koraku je bitna i optimizacija konačnih web stranica kako zbog bržeg učitavanja tako i zbog veće mogućnosti izlistavanja prilikom pretraživanja u tražilicama.

3 Dizajn sučelja

Do 2007. godine web dizajneri su mogli biti sigurni da se korisnici koji pregledavaju web stranice koriste računalo sa velikim ekranom. Tada su sve stranice trebale biti široke oko 960 piksela što je bio neki standard. Jedini problemi koje su mučili web dizajnere su bili prikazi u različitim internet pretraživačima. Iako se i prije 2007. godine stranicama moglo pristupiti preko mobilnog telefona većina korisnika je za pregledavanje web stranica koristilo računalo. Pojavom pametnih uređaja (iPhone-a i Android operacijskog sustava) sve više korisnika koristi i svoj mobilni uređaj za pregledavanje web stranica. Samim time javlja se novi veliki problem koji web dizajneri moraju riješiti, različite rezolucije ekrana kako samih mobilnih uređaja tako sada velika razlika između ekrana na mobilnim uređajima i ekrana koja koriste stolna računala. Uz mobilne uređaje u zadnje vrijeme na tržištu ima sve više i tablet-a koji također imaju različite rezolucije ekrana.



U počecima izrade web stranica, web dizajn se odnosio samo na statičke HTML (Hypertext Markup Language) stranice koje su koristile veze i grafiku. Danas web stranicu je gotovo nemoguće zamisliti bez dinamičke web stranice koja u sebi ima dijelove koristeći programske jezike kao na primjer .NET, JSP(JavaServer Pages) ili PHP.

3.1 Pravila dizajna

Prilikom dizajna trebalo bi paziti na nekoliko pravila koja bi se trebala poštivati zbog uspješnosti web stranice.

1. Jednostavan dizajn
2. Sadržaj stranice
3. Čitljivost teksta
4. Jednostavnost korištenja – navigacija
5. Podrška za sve preglednike
6. Ažuriranost stranice
7. Ključne riječi
8. Pružanje informacija korisniku
9. Razlog za vraćanje na stranicu
10. Testiranje

3.1.1 Jednostavan dizajn

Jednostavan dizajn je jedan prvo pravilo koje bi svaki web dizajner trebao znati. Korisnici ne vole komplikirane i kompleksne web dizajne, posjetitelj dođe na stranicu pogledati ono što ga zanima i što je stranica jednostavnija to je veća vjerojatnost da će se vratiti na istu stranicu. Primjer je recimo Google pretraživač koji ima vrlo jednostavan dizajn koji gotovo svi posjetitelji prihvataju. Posjetitelj koji dođe na Google traži informaciju i želi ju što prije dobiti.



3.1.2 Sadržaj stranice

Istraživanjem je dokazano kako ljudi čitaju web stranice u obliku slova F. Što znači da posjetitelji najviše gledaju gornji lijevi dio prema desnoj strani i odozgore po lijevoj strani prema dolje. Samim time dizajnerima je to naputak gdje trebaju staviti najvažnije informacije koje žele kako bi korisnici što prije došli do informacija koje ga zanimaju.



3.1.3 Čitljivost teksta

3.1.3.1 Fontovi

Veličinu i odabir fonta treba prilagoditi namjeni web stranice. Postoje dvije veće podijele font-ova (stilova pisma) a to su Serif i Sans-serif. U serif skupinu fontova spadaju oni koji imaju malene ukrasne poteze, zaobljenja na rubovima slova. Zbog te karakteristike najčešće se upotrebljavaju za naslove i tamo gdje treba naglasiti velike dijelove teksta jer kod manjih veličina serif fontovi smanjuju čitljivost zbog ukrasa koji oduzimaju bijeli prostor između slova. To vrijedi samo za web dok za tisk na papiru to nije pravilo. Rezolucija ima najviše

utjecaja na čitljivost teksta pa je tako rezolucija kod web-a 72dpi-a dok se za tisk koristi najmanje 300dpi-a. Kod nižih rezolucija slova postaju manje čitljiva jer pikseli dolaze do izražaja. Primjer serif fonta su: Times, Times New Roman, Courier, Courier New, Georgia, Garamond.



Slika 4. Serif font

Sans-serif fontovi nemaju rubne ukrase na slovima te samim time se povećava čitljivost jer povećava bjelinu između slova. Sans-serifni fontovi su najidealniji za ispisivanje sadržaja stranica te su zbog toga najčešće korišteni na web-u. Primjer sans-serif fonta su: Arial, Tahoma, Verdana, Trebuchet.



Slika 5. Sans serif font

Većina fontova dolazi zajedno sa operacijskim sustavom no može se dogoditi da web dizajner na stranicu stavi neki font koji je sam kreirao ili skinuo sa interneta. Na to se treba obratiti pažnja jer se može dogoditi da krajnji korisnik nema instaliran taj font te će se prikazivati zamjenski font što može narušiti cijeli dizajn stranice. Prilikom izrade web stranica trebaju se koristiti takozvani web safe fontovi koji su zapravo sistemski fontovi koji su sastavni dio svakog operacijskog sustava.

Uz vrste fontova postoje i grupe fontova od kojih su najpoznatije:

- Serif (serifni fontovi)
- Sans-serif (sans serifni fontovi)
- Cursive (ukošeni fontovi)
- Fantasy (dekorativni fontovi)
- Monospace (fontovi jednake širine)

Primjer grupe fontova: font-family: Arial, 'Trebuchet MS', Helvetica;

U navedenom primjeru ako se dogodi da korisnik koji pregledava stranicu nema instaliran prvi font, kao zamjenski koristiti će se drugi a ako njega nema onda će se koristiti treći. Naravno uvjek bi se trebalo paziti na to da obavezno jedan bude sistemski kako bi se na kraju stranica mogla bez problema pregledavati. Ukoliko se dogodi da ne postoji niti jedan od navedenih fontova kao zadnji font može se staviti cijela grupa fontova koja će se tada koristiti kao zamjenski font.

Primjer grupe fontova: font-family: Arial, Helvetica,sans-serif;

Veličinu fonta je bitno prilagoditi namjeni stranice. Naslovi, navigacija te bitne informacije potrebno je naglasiti sa većim fontom dok sami sadržaj stranice može biti manjeg fonta. Većina današnjih preglednika omogućava povećanje prikaza tako da ako korisnik ne može sa lakoćom čitati može povećati prikaz cijele stranice.

3.1.3.2 Boje

Boje su vrlo bitne u web dizajnu te pravilnim odabirom boja web stranica može privući ili odbiti posjetitelja. Većina tvrtki koji ulaze u svoj imidž vode brigu oko pravilnog odabira boja te na kraju boje su njihov simbol prepoznavanja. Na primjer Milka (ljubičasta), Coca-Cola (crvena), McDonald's (žuta- crvena), Facebook (plava-bijela).

Postoje dva sustava boja a to su RGB (**R**ed, **G**reen, **B**lue) i CMYK (**C**yan, **M**agenta, **Y**ellow, **B**lack). RGB sustav boja koriste ekrani, fotoaparati, kamere dok CMYK sustav boja koriste pisači. Sustav boja koji se koristi na webu je RGB. Miješanjem te tri boje moguće je dobiti bilo koju nijansu boje od potpuno crne do potpuno bijele. Svaki dio boje ima raspon od 0 do 255 gdje 0 predstavlja minimalni udio te boje dok 255 predstavlja maksimalni udio te boje.

Unutar kôda boja se može definirati na tri načina:

- Ime boje,
- RGB vrijednost boje, te
- Heksadecimalni kôd boje.

Ime boje podrazumijeva korištenje naziva boje koja se želi koristiti recimo „blue“ je primjerice za plavu boju. Na ovaj način moguće je vrlo jednostavno koristiti boje bez potrebe znanja rgb ili heksadecimalne vrijednosti boje. Drugi način korištenja boje je upisivanjem pojedine vrijednosti boja koje želimo, na primjer za potpuno crnu boju treba napisati rgb(0,0,0), za potpuno bijelu (255,255,255) dok primjerice za plavu boju treba napisati rgb(0,0,255). Treći način korištenja boja je pomoću heksadecimalnog kôda koji zapravo predstavlja heksadecimalni prikaz rgb broja. Heksadecimalni kôd sastoji se od znaka # i 6 znakova (brojki i/ili slova). Tih 6 znakova su heksadecimalne vrijednosti – kombinacija osnovnih boja (crvene, zelene i plave). Prva dva heksadecimalna znaka se odnose na crvenu boju, srednja dva na zelenu dok zadnja dva na plavu boju. Heksadecimalno označavanje kreće se od #00 (0) do #FF (255) i predstavljaju kao i kod RGB označavanja udio određene osnovne boje u konačnoj boji.

Kod web dizajna najčešće se koristi heksadecimalno označavanje boja.

Naravno nije potrebno pamtiti heksadecimalni kôd određene boje jer svi programi koji omogućavaju rad sa bojama imaju ugrađene palete boja ili takozvane Color Mixer-e koji nude mogućost odabira određene boje klikom miša te se nakon toga prikazuje i rgb i heksadecimalna vrijednost odabrane boje.

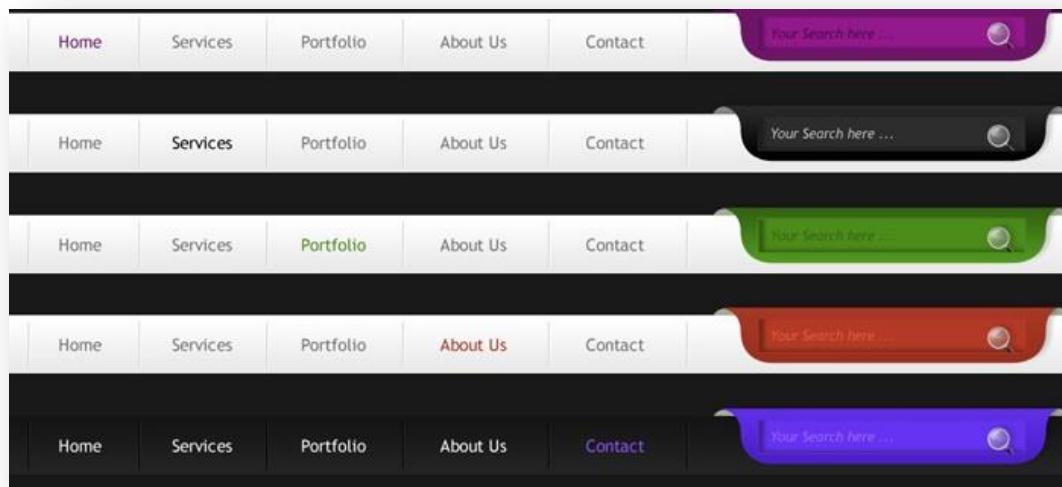
3.1.4 Jednostavnost korištenja – navigacija

Svaka web stranica ima poveznice na druge stranice unutar istog web sjedišta ili vanjske poveznice. Prilikom dizajna vrlo je bitno osmisiliti pravilnu navigaciju koja će posjetitelju omogućiti jednostavno korištenje i navigaciju između nekoliko stranica. Kod dužih stranica potrebno je postaviti poveznicu koja vodi na vrh stranice kako bi posjetitelj vrlo brzo i jednostavno se vratio na početak stranice. Postoje dvije vrste navigacija a to su:

- Horizontalna, te
- Vertikalna

3.1.4.1 Horizontalna navigacija

Najčešće se nalazi na vrhu stranice i sadrže glavnu navigaciju. Unutar glavne navigacije može se nalaziti i pomoćna tj. sekundarna koja ima podelemente svakog glavnog elementa. Na primjer ako se radi o stranici neke web trgovine koja prodaje računala i pripadajuću tehniku, jedan podelement u grupi računala može biti stolno računalo a drugi podelement može biti prijenosno računalo. Nakon toga se može sve dalje granati koliko je potrebno i koliko se želi.



Slika 6. Primjer horizontalne navigacije

3.1.4.2 Vertikalna navigacija

Vertikalna navigacija je najstarija navigacija koja se koristi na web stranicama. Može biti pozicionirana na lijevoj ili desnoj strani stranice. Isto kao i kod horizontalne navigacije mogu se koristiti izbornici u nekoliko razina ovisno o namjeni stranice.



Slika 7. Primjer vertikalne navigacije

Uz ove dvije vrste navigacija postoje i navigacije koje su prilagođene namjeni koje ne spadaju u niti jednu vrstu navigacije. Primjerice jedno vrijeme stranica od Nike-a je imala navigaciju koja je izgledala kao daljinski upravljač. Takva vrsta navigacije se najčešće koristi tvrtki koje prodaju razne proizvoda kako bi privukli što više posjetitelja koji će na kraju kupiti proizvod koji se prikazuje na stranici.



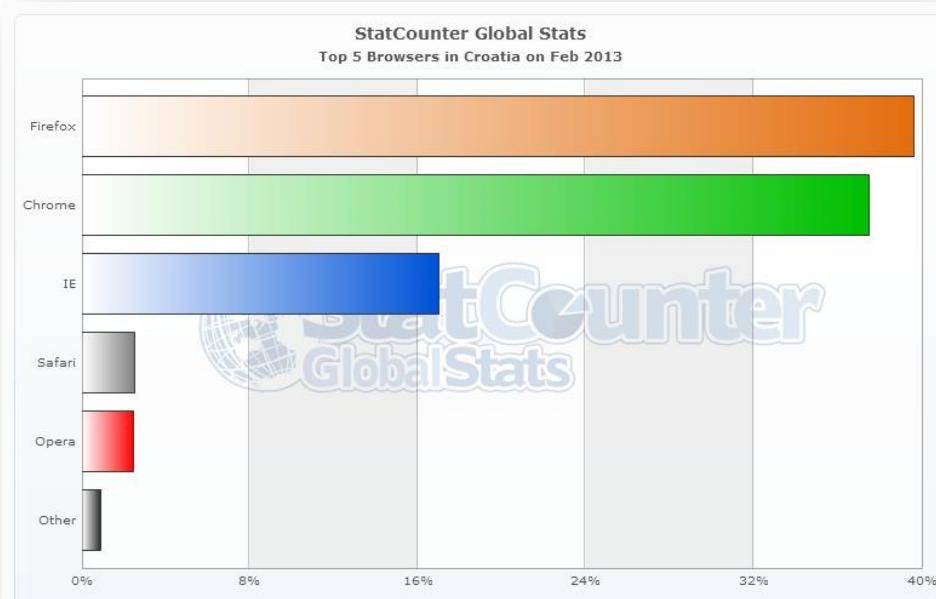
Slika 8. Primjer navigacije koja izgleda kao daljinski upravljač

Koja god navigacija se odabere treba biti konzistentna tokom izrade cijelog web sjedišta, odnosno kroz svaku web stranicu treba biti pozicionirana na istom mjestu i imati isti dizajn kako bi posjetitelj znao gdje se nalazi i lakše se snalaziti na stranici.

3.1.5 Podrška za sve preglednike

Početkom razvoja interneta nije bilo previše preglednika koji su se koristili za pregledavanje web stranica. Razvojem tehnologija pojavilo se sve više preglednika pomoću kojih su krajnji

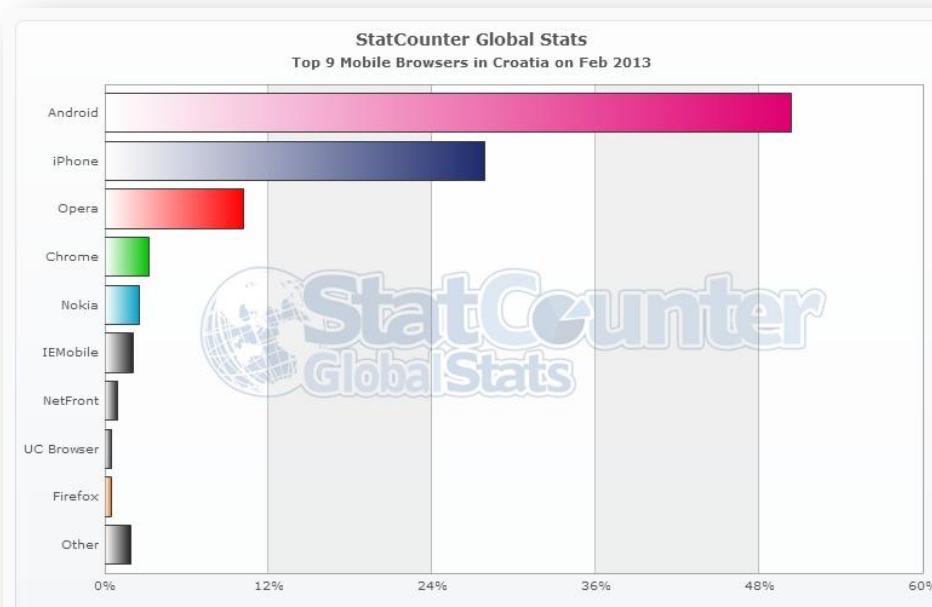
korisnici mogli pregledavati web stranice. Najpopularniji internet preglednici koji se danas koriste su Internet Explorer, Mozilla Firefox, Opera, Google Chrome, Safari i drugi. Internet Explorer dolazi kao dio Windows-a te ga mnogi korisnici koriste kao jedini internet preglednik. Mozilla Firefox i Google Chrome su trenutno najpopularniji internet preglednici.



Slika 9. Korištenje internet preglednika

3.1.5.1 Mobilni web – mobile web

Pregledavanje web stranica u zadnje vrijeme je sve popularnije i preko mobilnih uređaja koji koriste razne vrste internet preglednika. Općenito pojam mobilnog weba ne predstavlja podvrstu weba nego znači da se istim web stranicama pristupa preko mobilnih uređaja. Naravno zbog različitih rezolucija ekrana potrebno je prilagoditi stranicu mobilnom uređaju no to je dalje ista web stranica koju je moguće pregledavati i preko bilo kojeg drugog internet preglednika. Ponajprije ono na što se mora paziti prilikom izrade stranica koje će se pregledavati i preko mobilnih uređaja je navigacija koja je zbog manjih ekrana mobilnih uređaja otežana u odnosi na velike ekrane koje koriste računala.



Slika 10. Korištenje internet preglednika na mobilnim uređajima

3.1.6 Ažuriranost stranice

Stranicu bi redovito trebalo osvježavati sa novim informacijama. Ako je riječ o stranici koja prodaje neki proizvod potrebno je uvijek na stranici imati aktualne cijene, opise proizvoda koji su u trenutnoj ponudi. Isto tako potrebno je osvježavati i kontakt informacije u slučaju promjene adrese, promjeni broja telefona ili bilo kakvih drugih bitnih informacija koje su bitne krajnjem posjetitelju stranice. Ako se radi o stranici koja izvještava o vijestima i događajima potrebno je imati najnovije vijesti koje su aktualne. Isto tako vrlo je bitno imati i arhivu starih vijesti kako bi se u bilo kojem trenutku moglo doći do vijesti i događaja koji su se dogodili u prošlosti. Uvijek je dobro na stranicu staviti i tražilicu koja omogućava pretraživanje informacija unutar stranice ili cijelog web sjedišta ili u nekim slučajevima čak direktno pretraživanje interneta vez otvaranja tražilice.

3.1.7 Ključne riječi

Ključne riječi su vrlo važne najviše zbog internetskih tražilica koje upravo pomoću toga pronalaze web stranice. Što je više ključnih riječi korišteno unutar samo web stranice to je veća šansa da će se pojaviti prije u tražilici. Naravno potrebno je odabrati ključne riječi koje imaju doticaja sa sadržajem teksta koji se nalazi na stranici ili ključne riječi koje imaju veze sa proizvodima i uslugama koje se nalaze na stranici.

3.1.8 Pružanje informacija korisniku

Cilj web stranice je brzo, efikasno i točno posjetitelju pružiti informaciju zbog koje je i došao. Svaki posjetitelj ne želi trošiti puno vremena tražeći informaciju na stranici i samim time dizajn web stranice mora biti takav da posjetitelj odmah uoči ono što ga zanima.

Sve bitne informacije trebaju biti istaknute na vidljivim mjestima te se treba izbjegavati predugačka ili preširoka stranica na kojoj posjetitelj mora koristiti ili vertikalne ili horizontalne klizače tzv. scroll. Naravno nije uvijek moguće napraviti stranicu gdje se ne moraju koristiti klizači no ako je baš neophodno onda bi bilo bolje kada bi to bili samo vertikalni klizači.

3.1.9 Razlog za vraćanje na stranicu

Razlog za vraćanje posjetitelja na stranicu može biti mini blog koji se vodi sa novi aktualnostima, forum preko kojeg posjetitelji komuniciraju međusobno ili sa vlasnikom web stranice. Jedan od mogućih načina privlačenja posjetitelja je mogućnost pretplate na novosti takozvani newsletter. To je vrsta pretplate u kojoj je posjetitelj koji želi primati novosti vezano za novosti na stranici, novosti o proizvodima ili nove usluge ostavlja svoju e-mail adresu te se nakon toga automatski šalju novosti na koje se posjetitelj pretplatio.

3.1.10 Testiranje

Svaku stranicu na cijelom web sjedištu treba provjeriti kako se ne bi dogodilo da bilo koja poveznica sa drugom stranicom nije dobra tj. da ne postoje takozvane „broken links“. Uz provjeru valjanosti svih poveznica potrebno je stranice testirati koristeći W3C validatore. W3C validatori mogu testirati HTML i CSS kôd te na taj način dizajneru dati potpuni izvještaj da li je sve napravljeno po standardima koje bi trebalo poštivati.



Slika 11. W3C validator

A screenshot of the W3C Validator interface showing the validation output. It starts with a section titled "Notes and Potential Issues" which includes a note about experimental features and character encoding. Below this is a "Validation Output: 24 Errors" section. One specific error is highlighted: "Line 9, Column 2182: The bgcolor attribute on the body element is obsolete. Use CSS instead." with the corresponding code snippet: "... </head><body dir="ltr" bgcolor="#ffff"><script>(function(){var src='/images/sr...". The page also shows a "Top" button at the bottom right.

Slika 12. W3C validator - pregled grešaka

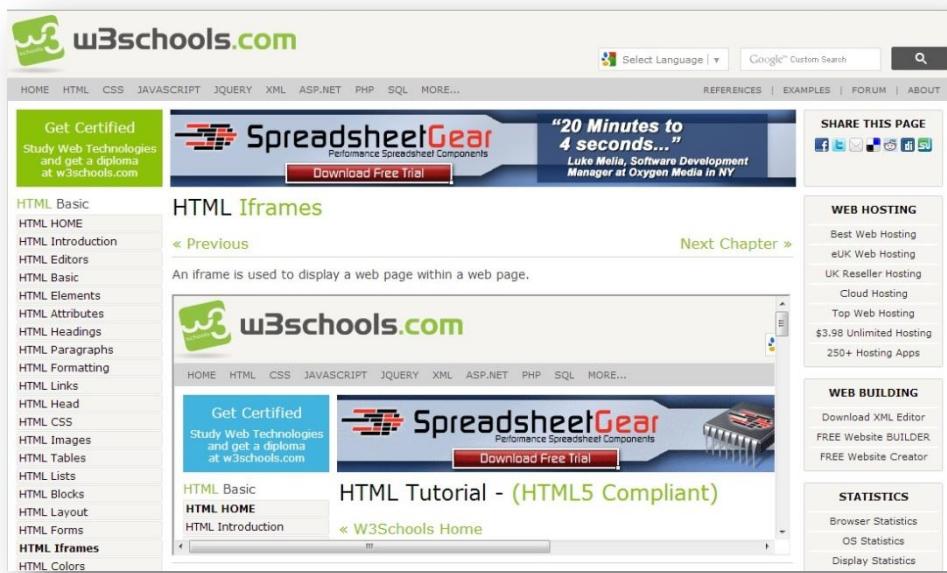
Također svakako je potrebno provjeriti stranice na različitim rezolucijama kako na većim ekranima tako i na mobilnim ekranima te na različitim preglednicima kako se ne bi dogodilo da posjetitelj koji koristi stariji preglednik ne može pregledavati stranicu.

4 Oblikovanje stranica

Prilikom oblikovanja web stranica vrlo je bitno znati postaviti pravilni dizajn te način na koji će se taj dizajn i kreirati. Također vrlo je bitno znati postaviti navigaciju unutar stranice i veze prema drugim web stranicama.

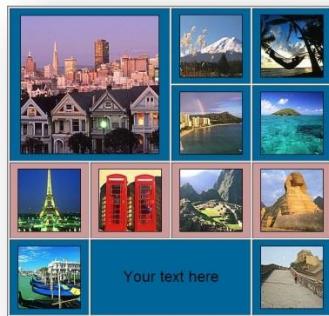
4.1 Dizajn pomoću frame-ova

Stranice koje koriste okvire su se koristile kada su nastale web stranice i oni se više ne koriste previše. Naprednija verzija okvira su i-frame-ovi oni se mogu koristiti kada se želi ograničiti duljina stranice no i dalje se želi imati relativno dugačka „podstranica“ unutar stranice. To se recimo može koristiti kada se želi unutar stranice staviti nekakva tablica koja može biti vrlo dugačka te kako se povećanjem tablice ne bi povećala cijela stranica.



4.2 Dizajn pomoću tablica

Još jedan od starijih načina dizajniranja web stranica koji koristi tablice za pozicioniranje elemenata. Ovaj način oblikovanja stranica može se i danas koristiti kada se radi recimo jednostavna galerija slika u kojoj se žele staviti slike jednakih veličina.



5 Multimedija

Od prve pojave web stranica multimedija čini osnovni dio svake web stranice, grafika, zvuk i video su danas nezaobilazni dio svake web stranice. Upravo zato vrlo je bitno znati osnove grafike kako bi se mogla napraviti što kvalitetnija web stranica koja se brzo učitava. Kod izrade web stranica vrlo je bitna kvalitetna kompresija multimedije kako bi se što brže stranice mogle učitavati no naravno bitno je i da kompresija ne bude prevelika kako se ne bi previše izgubila kvaliteta.

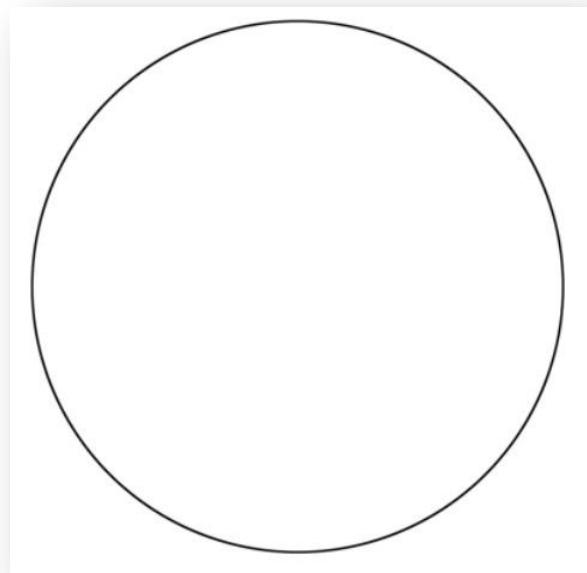
5.1 Formati za slike

Postoji nekoliko formata za slike koje svakodnevno koristimo no neki od njih nisu primjereni za web stranice zbog svoje veličine i neoptimiziranosti. Primjer takvog formata je BMP koji je format za nekompresiranu slikovnu datoteku. Neki od primjera kompresiranih slikovnih datoteka su JPG, PNG i GIF.

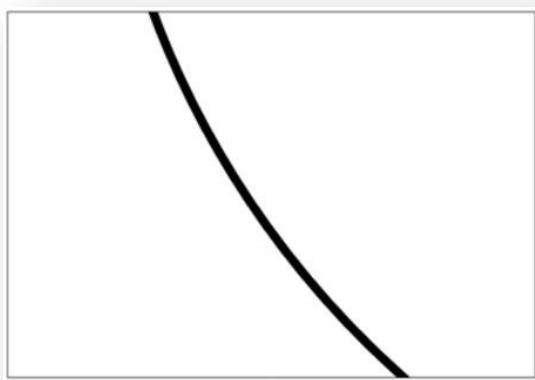
5.1.1 Razlike rasterske i vektorske grafike

Većina poznatih formata za slike kao što su JPG, PNG, GIF su rasterski tipovi grafike i oni se najčešće koriste za web stranice no kako je to rasterska grafika imaju veliku manu.

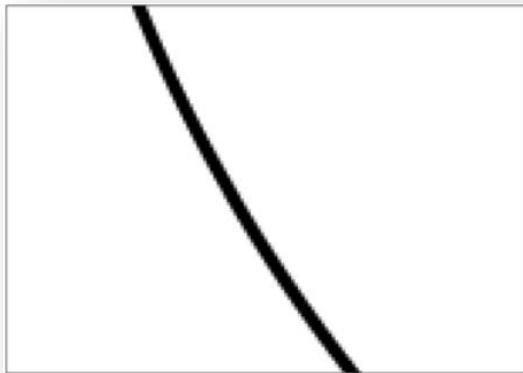
Povećanjem slike gubi se njihova kvaliteta te pri velikom povećanju izgledaju kockasto. Kod vektorske grafike to nije slučaj i one su uvijek iste kvalitete bez obzira na njihovu veličinu recimo SVG format. Sljedeći primjer će to najbolje prikazati, ako se napravi krug u vektorskem i rasterskom formatu i nakon što se poveća samo dio kruga dobiti će se različiti rezultati.



Krug koji je napravljen u vektorskog grafici iako je povećan nije izgubio na kvaliteti prikaza, što znači da bez obzira povećavanje ili smanjivanje kvaliteta ostaje ista što je vrlo bitno kod web stranica gdje se koriste ekran različitih rezolucija.



Krug koji je napravljen u rasterskog grafici kada je povećan kvaliteti prikaza se promijenila, tj. rubovi su postali zamućeni i kada bi se povećalo još više vidjelo bi se kako je crna boja na rubovima postala siva. Iz ovoga se vidi kako u rasterskoj tehnologiji promjena rezolucije mijenja kvalitetu prikaza no u većini slučajeva gubitak kvalitete nije toliko jako vidljiv jer se slike optimiziraju na određenu rezoluciju na kojoj se toliko ne vidi gubitak kvalitete.



6 CSS

CSS (*Cascading Style Sheets*) je tehnologija koja služi za dizajniranje HTML dokumenta. CSS je uveden zbog otežanog dizajniranja grafičkih elemenata unutar HTML-a te zbog odvajanja strukture sadržaja i prezentacije.

6.1 Povezivanje HTML-a i CSS-a

CSS stilove je moguće povezati sam HTML-om na tri načina:

- Linijski (atribut style unutar svake HTML oznake),
- Unutar `<style>` HTML oznake u `<head>` oznaci HTML-a te
- U vanjskoj datoteci koju povezujemo s HTML dokumentom.

Najčešće upotrebljavani način je odvajanje stilova u zasebnu datoteku jer se na taj način omogućava pristup svim stilovima u više HTML datotekama.

6.1.1 Linijski način pisanja stilova

Linijski način naziva se pisanje CSS-a unutar HTML oznake. Svakoj HTML oznaci se dodaje atribut **style** pomoću kojeg se definiraju pojedine CSS oznake.

Na primjer:

```
<p style="color:blue; font-size:20px">Ovdje će se prikazati tekst plave boje veličine 20 px</p>
```

U ovom primjeru koristi se **style** atribut HTML oznake `<p>` (oznaka paragrafa) unutar kojeg će se prikazati tekst plave boje i veličine slova 20px.

6.1.2 Pisanje CSS-a unutar oznake `<style>`

Pisanjem CSS stilova unutar `<head>` HTML oznake omogućava lakši pregled svih CSS oznaka unutar jedne HTML datoteke jer se sve oznake stavljaju na jedno mjesto unutar `<style>` HTML oznake.

Na primjer:

```
<html>
  <head>
    <title>Naslov</title>
    <style type="text/css">
      h2 {font-size: 12px;}
      p {font-size: 10px;
          color: #222222;}
    </style>
  </head>
  <body>
    <h2>Ovo je tekst h2</h2>
    <p>Tekst paragrafa</p>
  </body>
</html>
```

Primjer korištenja CSS oznaka unutar `<style>` HTML oznake koja omogućava korištenje nekoliko stilova unutar iste `<style>` oznake. Za razliku od linijskog načina pisanja na ovaj način moguće je vrlo lako upravljati izgledom stranice koristeći samo jednu `<style>` oznaku.

6.1.3 Pisanje stilova u vanjskoj datoteci

Ovaj način omogućava pisanje stilova koji se mogu koristiti u nekoliko HTML datoteka. Na taj način moguće je napraviti isti dizajn za sve stranice koje se žele koristiti. Također na taj način vrlo je lako promijeniti izgled cijele stranice ili nekoliko stranica promjenom samo jedne CSS oznake koja se nalazi u vanjskoj CSS datoteci. CSS datoteka je obična tekstualna datoteka koja ima nastavak `.css` te ju je moguće uređivati u bilo kojem tekstu editoru koji se koristi na računalu.

```
h1 {  
    font-size:24px;  
    color:yellow;  
}  
  
h2 {  
    font-size:18px;  
    color:#ff0010;  
}  
  
p {  
    font-size:10px;  
    color:#00ee00;  
}
```

```
<html>  
    <head>  
        <title>1. primjer </title>  
        <link href="stil.css" media="all" type="text/css" rel="stylesheet" />  
    </head>  
    <body>  
        <h1>Ovo je naslov h1</h1>  
        <p>Tekst paragrafa koji koristi stilove iz css datoteke</p>  
        <p>Drugi tekst paragrafa koji koristi stilove iz css datoteke</p>  
    </body>  
</html>
```

```
<html>  
    <head>  
        <title>2. primjer </title>  
        <link href="stil.css" media="all" type="text/css" rel="stylesheet" />  
    </head>  
    <body>  
        <h1>Ovo je naslov h1</h1>  
        <h2>Ovo je podnaslov</h2>  
        <p>Tekst paragrafa koji koristi stilove iz css datoteke</p>  
    </body>  
</html>
```

Izgled prvog primjera HTML dokumenta koji je uređen koristeći CSS iz vanjske datoteke.



Izgled drugog primjera HTML dokumenta koji je uređen koristeći CSS iz vanjske datoteke.



6.2 Pravila CSS-a

Svako pravilo unutar CSS-a sastoji se od:

- selektora,
- svojstva te
- vrijednosti

Selektor služi za određivanje HTML elementa na koji će se stil primjenjivati (body, p,h1, itd.)

Na primjer ako se želi primijeniti stil na HTML oznaku <p> CSS oznaka će biti **p**.

Svojstvo određuje koje se svojstvo određenog selektora želi urediti. Svojstva su **color** za boju, **font-size** za veličinu slova, **border** za obrub.

Vrijednost predstavlja vrijednost određenog svojstva kao što za svojstvo **color** vrijednost može biti **yellow** ili za svojstvo **font-size** može biti vrijednost **18px**.

Deklaracija stila sastoji se od svojstva i vrijednosti na primjer:

```
color: yellow;
```

Svojstvo i vrijednost odvajaju se dvotočkom, dok se deklaracije odvajaju točkom-zarez. Sve deklaracije koje pripadaju nekom selektoru grupiraju se vitičastim zagradama.

Na primjer ako se želi oblikovati `<p>` HTML element kojem se želi postaviti boja teksta u plavu i postaviti veličinu slova na 20px to se može napraviti ovako:

```
h1 {  
    color: blue;  
    font-size: 20px;  
}
```

6.3 Grupiranje selektora

Ponekad postoji potreba da se više različitih HTML elemenata urede sa istim stilom. Kako se ne bi ponavljao kôd za isti stil nekoliko puta, postoji način da se svi potrebni elementi grupiraju te se nakon toga uredi ta grupa kao jedan stil. Selektori se grupiraju tako da se između njih stavi zarez.

```
h1, h2 {  
    color: #222222;  
    font-size: 14px;  
}
```



Na ovaj način je moguće grupirati neograničen broj selektora što smanjiti veličinu css datoteke u kojoj se nalaze stilovi te na taj način ubrzati učitavanje stranice.

6.4 Nasljeđivanje svojstva

Grupiranje selektora podrazumijeva korištenje nekoliko identičnih vrijednosti svojstava za sve grupirane selektore. Ponekad se može dogoditi da određeni selektori imaju vrijednosti nekih svojstava te se tada koristi nasljeđivanje svojstava. Na primjer:

```
h1, h2 {  
    color: #222222;  
}  
h2 {  
    font-size: 14px;  
}
```

Selektor h2 u primjeru se pojavljuje na dva mesta. Prvi puta je grupiran sa selektorm h1 i definirano im je zajedničko svojstvo boja. Drugi puta selektor h2 se pojavljuje kako bi se definiralo svojstvo veličine slova pri čemu boja teksta ostaje ista. Ovim primjerom se vidi kako je selektor h2 naslijedio boju teksta iz prethodne deklaracije. U slučaju da se neka svojstva preklapaju veći prioritet ima svojstvo koje je zadnje napisano.

```
h1, h2 {  
    color: #222222;  
    font-size: 24px;  
}  
h2 {  
    font-size: 14px;  
}
```



Grupiranje selektora zajedno sa nasljeđivanjem svojstava su vrlo bitna za optimiziranje veličine css datoteke. Grupiranje se obično radi nakon što se deklariraju sva svojstva koja su potrebna za dizajn stranica.

6.6 Svojstva za uređivanje teksta

6.6.1 Boja teksta

Za određivanje boje teksta unutar HTML elementa može se koristiti svojstvo **color**.

Vrijednost svojstva može se postaviti na neki nekoliko načina:

- Imenom boje (blue, red, yellow),
- RGB vrijednošću boje (rgb(255,200,200)) te
- Heksadecimalnim brojem (#00ff00).

Najčešći način postavljanja boje je korištenjem heksadecimalnim brojem.

```
color: blue;  
color:rgb(0,0,255);  
color: #0000ff;  
color: #00f;
```

6.6.2 Veličina teksta

Za promjenu veličine teksta koristi se svojstvo **font-size**. Svojstvo se može postaviti na nekoliko načina od kojih je najpopularniji unošenje absolutne vrijednosti u pikselima (px).

```
font-size: 12px;  
font-size: x-small;  
font-size: 200%;  
font-size: 2em;
```

W3C preporuča korištenje ili % ili em za veličinu slova jer se na taj način smanjuje mogućnost problema kod različitih rezolucija.

6.6.3 Debljina slova

Svojstvo za debljinu slova je **font-weight** a vrijednosti svojstva mogu biti sljedeće:

- normal,
- bold,
- bolder te,
- lighter.

Umjesto tih vrijednosti moguće je napisati i brojčanu vrijednost (100, 200, 300, 400, 500, 600, 700) s tim da 400 predstavlja normalnu debljinu slova dok 700 predstavlja bold debljinu slova.

```
font-weight: normal;  
font-weight: 400;
```

6.6.4 Oblik slova

Oblik slova je također dio koji se može uređivati unutar svojstava fonta. U oblike slova spadaju razne grupe fontova ili **font-family**. Postoje dvije osnovne verzije oblika slova a to su serif i sans-serif. Slova sa serifima koriste ukrase na rubovima slova te nisu toliko čitljiva kao što su slova bez serifa.

```
font-family: arial, verdana, sans-serif;  
font-family: courier, serif;
```

6.7 Svojstva za uređivanje elemenata

6.7.1 Pozadinska boja elemenata

Pozadinske boje elemenata se postavljaju na vrlo sličan način kao što se postavlja boja slova. Boja se može postaviti koristeći nekoliko načina:

- imenom boje,
- RGB vrijednosti boje te,
- Heksadecimalnim brojem boje.

```
background-color: blue;  
background-color:rgb(0,0,255);  
background-color: #0000ff;  
background-color: #00f;
```

Kao što je vidljivo iz prethodnih primjera imena svojstava u CSS-u su riječi engleskog jezika.

Kada se naziv svojstva sastoji od dvije riječi međusobno se spajaju koristeći znak '-'.

6.7.2 Obrub elemenata

Svakom HTML elementu se može postaviti obrub svojstvom **border**. Vrijednost svojstva sastoji se od nekoliko dijelova:

- debljine obruba (najčešće u pikselima),
- vrsta obruba (puna linija, točkasta linija, itd.) te
- boja obruba.

```
border: 2px solid #252525;
```

Ovo je naslov h1

Ovo je podnaslov h2

Kako je svaki HTML element pravokutnog oblika moguće je postaviti obrub na samo jednu stranicu pravokutnika (lijevu, desnu, gornju ili donju).

```
border-left: 2px solid #252525;  
border-right: 2px solid #252525;
```

Ovo je naslov h1
Ovo je podnaslov h2

6.7.3 Margina

Razmak između dva HTML elementa naziva se margina (**margin**). Kao i kod obruba margini se mogu postaviti na svakoj stranici posebno. Također moguće je staviti marginu za sve stranice samo sa jednom vrijednosti.

```
margin: 15px;
```

Ovo je naslov h1
Ovo je podnaslov h2

Pojedinačne margini je moguće postaviti kao i obrube koristeći sljedeće:

- margin-top (gornja margina),
- margin-left (lijeva margina),
- margin-right (desna margina) i
- margin-bottom (donja margina).

Sve margini je moguće postaviti unutar jedne linije koristeći jednu oznaku margin. Prvi broj se odnosi na gornju marginu, drugi na desnu, treći na donju te zadnji na lijevu marginu.

```
margin: 15px 10px 10px 15px;
```

6.7.4 Padding

Padding je svojstvo koje određuje udaljenost sadržaja HTML elemenata od obruba. Način na koji se deklarira je isti kao i kod margina.

```
padding: 15px;  
padding-left: 10px;  
padding: 15px 10px 15px 10px;
```

Padding je svojstvo koje se često koristi kako bi se tekst odvojio od obruba i na taj način olakšao čitljivosti teksta.

```
Padding
```

6.7.5 Širina i visina elemenata

Prepostavljena vrijednost širine većine HTML elemenata je 100% širine <body> elementa, točnije širina prozora preglednika. Za podešavanje širine koristi se svojstvo **width**, dok se za visinu koristi svojstvo **height**. Prepostavljena visina HTML elemenata je **auto**, odnosno povećava se proporcionalno količini sadržaja.

```
width: 400px;  
width: 50%  
  
height: 200px;  
height:10%;
```

Naslov h1

Paragraf visine 100px i širine 250px

Paragraf visine 100px i širine 250px

6.7.6 Svojstvo overflow

Svojstvo **overflow** definira na koji način će se ponašati element ako se unutar njega nalazi više teksta nešto što je visinom zadano. Ukoliko je postavljena visina elementa te sadržaji

koji se nalazi unutar tog elementa prelazi zadanu visinu koristeći ***overflow*** svojstvo ***auto***, automatski će se pojaviti klizna traka (scrollbar) koji će omogućiti pregled cijelog sadržaja tog elementa. Ako visina nije postavljena elementu unutar kojeg se nalazi sadržaj će se automatski povećati kako bi se mogao vidjeti cijeli sadržaj. Predefinirana vrijednost ***overflow*** svojstva je ***visible***.

6.7.6.1 Overflow: visible

```
height: 100px;  
overflow: visible;
```

Overflow svojstvo: visible

Kao što je vidljivo ako se postavi vrijednost na ***visible*** to znači da će se sadržaj prikazivati i izvan zadanih veličina.

6.7.6.2 *Overflow:auto*

```
height: 100px;  
overflow: auto;
```

Naslov h1

Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Cras mollis,
elit rhoncus pretium adipiscing, turpis
neque pretium erat, et aliquam nibh
augue sed lorem. Lorem ipsum dolor
sit amet, consectetur adipiscing elit.

```
height: auto;  
overflow: auto;
```

Naslov h1

Etiam non possumus. *Aliquam* enim est
adipiscing elit. *Cras* mollis, elit rhoncus
premium adipiscing, turpis neque premium
erat, et aliquam nibh augue sed lorem.
Etiam non possumus. *Aliquam* enim est
adipiscing elit. Nulla pharetra interdum
nisi id adipiscing. Duis commodo feugiat
nisi quis ullamcorper. Nunc vel lorem sed
est porta lacinia. Sed vel nisl vitae lectus
posuere semper. Suspendisse laoreet,
turpis interdum cursus vestibulum, arcu
augue congue sapien, non mattis ligula
quam non nibh.

6.7.6.3 Overflow: hidden

Hidden vrijednošću svojstva **overflow** sadržaj koji prelazi izvan granica elementa neće se prikazati tj. neće biti vidljiv sadržaj koji je izvan dimenzija elementa.

```
height: 100px;  
overflow: hidden;
```

Overflow svojstvo: hidden

Etiam et urna eu ligula fermentum pulvinar. Quisque quis justo lorem. Suspendisse porttitor lobortis erat ut posuere. Vestibulum hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a elit. Aliquam quis nisl sit amet nibh

6.7.6.4 Overflow:scroll

Koristeći ***scroll*** vrijednost svojstva ***overflow*** pojavljuju se i vertikale i horizontalne pomične trake (scroll) i kada je potrebno i kada nije. To znači da bez obzira na veličinu sadržaja i količinu teksta koji se nalazi unutar elementa trake će biti vidljive.

```
height: 100px;  
overflow: scroll;
```

Overflow svojstvo: hidden

```
 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam  
et urna eu ligula fermentum pulvinar. Quisque quis justo lorem.  
Suspendisse porttitor lobortis erat ut posuere. Vestibulum  
hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed  
< >
```

6.7.7 Postavljanje pozadinske grafike elementa

Za postavljanje pozadinske grafike elementu koristi se svojstvo ***background-image***. Kao vrijednost svojstva potrebno je postaviti relativnu ili absolutnu putanju do željene grafike. Postoji i opcija ponavljanja grafike, postavljanje pozicije te postavljanje veličine. Kod postavljanja pozadine ovo svojstvo se postavlja na `<body>` element.

```
background-image: url(slika1.jpg);  
background: url(pozadina.jpg) repeat-x right;
```

Naslov h1

```
 Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Cras mollis, él rhoncus  
prettium adipiscing, turpis neque pretium  
erat, et aliquam nibh augue sed lorem.  
Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Nulla pharetra interdum  
nisi id adipiscing. Duis commodo feugiat  
nisi quis ullamcorper. Nunc vel lorem sed  
est porta lacinia. Sed vel nisl vitae lectus  
posuere semper. Suspendisse laoreet,  
turpis interdum cursus vestibulum, arcu  
augue congue sapien, non mattis ligula  
quam non nibh.
```

VsiteVsiteVsiteVsiteVsiteVsiteVsiteVsiteVsite

```
background: url(pozadina.jpg) no-repeat right;
```

Naslov h1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras mollis, elit rhoncus pretium adipiscing, turpis neque pretium erat, et aliquam nibh augue sed lorem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla pharetra interdum nisi id adipiscing. Duis commodo feugiat nisi quis ullamcorper. Nunc vel lorem sed est porta lacinia. Sed vel nisl vitae lectus posuere semper. Suspendisse laoreet, turpis interdum cursus vestibulum, arcu augue congue sapien, non mattis ligula quam non nibh.

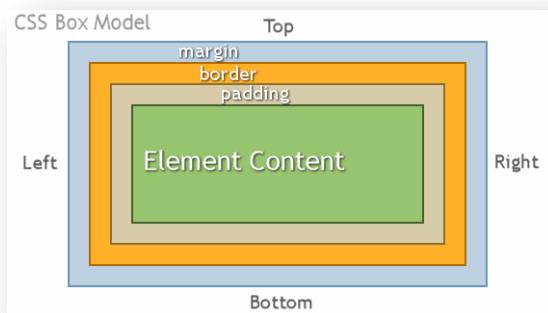


6.8 Model kutije (Box model)

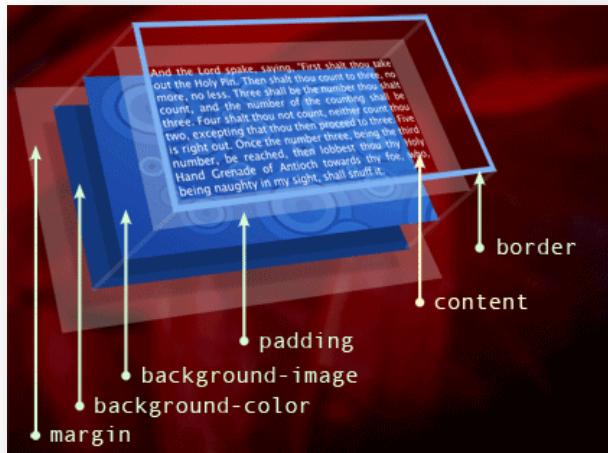
Svi HTML elementi mogu se smatrati "kutijama". CSS "box model" (model kutije) je u biti okvir koji uokvirava HTML elemenata, a sastoji se od:

- sadržaja - sadržaj HTML elementa (tekst, slika, drugi elementi),
- margina - prostor između ruba HTML elementa i ostalih elemenata,
- bordera - obrub HTML elementa te
- paddinga - prostor između sadržaja i obruba).

CSS "box model" (model kutije) omogućava postavljanje obruba oko elemenata, vanjske razmake između različitih elemenata i razmake između obruba elemenata i sadržaja elemenata. Razumijevanje box modela ima uvelike veze sa layout-om stranica



Uz sve navedeno u modelu kutije nalazi se i pozadinska boja i pozadinska slika.



6.9 Klasa i ID selektori

U radu sa HTML-om dokumenata često nije dovoljno upotrebljavati samo selektori koji su uključeni u HTML. U tim slučajevima potrebno je definirati vlastite stilove za jedan HTML element. Klase su ekstenzije selektora koje omogućavaju da se za jedan selektor koristi više stilova.

6.9.1 Klasa

Klasa nekom selektoru se pridjeljuje tako što se iza imena selektora stavi točka i naziv klase

```
p.pocetak {color:#252525;}
```

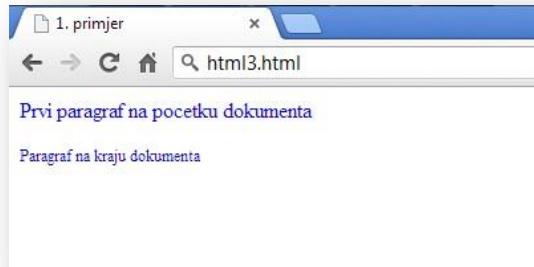
Nakon kreiranja klase unutar CSS-a potrebno je tu klasu i pozvati unutar HTML dokumenta.

```
<p class="pocetak">Prvi paragraf</p>
```

Na ovaj način je moguće definirati poseban stil za svaki HTML element unutar dokumenta.

```
p {color: #252525; font-size: 14px;}  
p.pocetak {color: #0000ff; font-size: 16px;}  
p.kraj {color: #0000dd; font-size: 12px;}
```

```
<body>  
    <p class="pocetak">Prvi paragraf na pocetku dokumenta</p>  
    <p class="kraj">Paragraf na kraju dokumenta</p>  
</body>
```



Klase nužno se ne moraju pisati kao ekstenzija nekog selektora već mogu biti samostalne.

```
.tekst {  
    color:#252525;  
    font-size: 16px;  
    padding-left: 50px;  
}
```

```
<body>  
    <p class="pocetak">Prvi paragraf na pocetku dokumenta</p>  
    <p class="tekst">Tekst koji se nalazi unutar HTML dokumenta. </p>  
    <p class="kraj">Paragraf na kraju dokumenta</p>  
</body>
```



6.9.2 ID atribut

Uz klase vrlo često se koriste i ID atributi koji su jedinstveni identifikatori elemenata. U CSS-u se pišu tako što se ispred njih stavi znak **#**. Jedan ID atribut može se koristiti samo za jedan element za razliku od klase koja se može koristiti na više elemenata.

```
#prviparagraf {background-color:#999999; color:yellow;}
```

```
<p id="prviparagraf">Prvi paragraf</p>
```

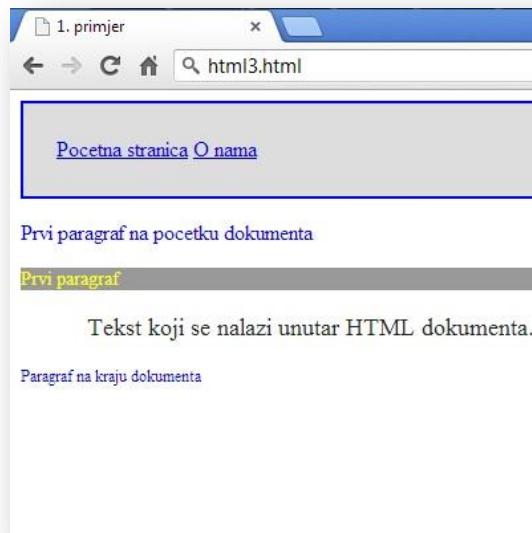


6.10 Elementi <div> i

Prilikom uređivanja izgleda HTML dokumenata potrebno je grupirati više elemenata u logičke cjeline kao što su na primjer zaglavlje gdje se nalazi logo i navigacija. Takvo grupiranje moguće je napraviti koristeći **<div>** element. Koristeći taj element moguće je postaviti ista svojstva za niz elemenata koji se nalaze unutar pojedine cjeline.

```
<div id="zaglavlje">
    <a href="index.html">Pocetna stranica</a>
    <a href="o_nama.html">O nama</a>
</div>
```

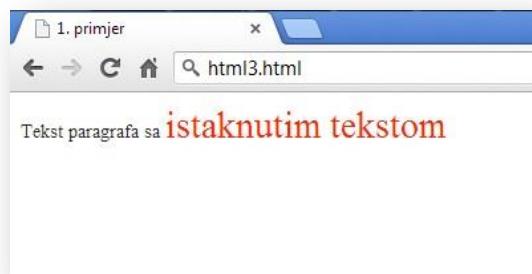
```
#zaglavlje {
    background-color: #dddddd;
    padding: 25px;
    border: 2px solid #0000fe;
}
```



Element **** se također može koristiti za odvajanje logičkih cjelina ali za razliku od div elementa koji stvara blokove, span je linijski element.

```
<p>Tekst paragrafa sa <span class="istaknuto"> istaknutim tekstrom </span></p>
```

```
.istaknuto {  
    color: #ff2500;  
    font-size: 28px;  
}
```



6.11 Pseudoklase i pseudoelementi

Korištenje pseudoklasa je uvedeno verzijom CSS-a 2.1. Time je omogućena veća fleksibilnost pri uređivanju elemenata. Pomoću njih je moguće promijeniti izgled prve linije unutar teksta ili postavljanja prvog slova velikog unutar odlomka (drop cap).

Pseudoklasama moguće je selektirati element ili stanje elemenata koji se nalaze u html datoteci. Tako je recimo moguće urediti izgled veze koji se može mijenjati ovisno na položaj

pokazivača miša. Neke od pseudoklasa koje se mogu koristiti u vezama su:

- a:link {color:#FF0000;}
- a:visited {color:#00FF00;}
- a:hover {color:#FF00FF;}
- a:active {color:#0000FF;}

Koristeći pseudoklase moguće je u potpunosti promijeniti izgled veza ovisno o njihovim stanjima.

- **:link** – se koristi za prikazivanje veza koje još nisu posjećene tj. korisnik nije nikad koristio tu vezu.
- **:visited** – se koristi za prikazivanje veza koje je korisnik posjetio.
- **:hover** – se koristi za prikazivanje veza u trenutku kada se pokazivačem miša prolazi preko veze.
- **:active** – se koristi za prikazivanje veza koje su aktivne veze.

Postoji još nekoliko pseudoklasa od kojih je zanimljiva **lang()** koja se može koristiti ako se želi napraviti više jezična stranica.

```
:lang(en) {color: #880088;}  
:lang(hr) {color: #550055;}  
<body lang="hr">
```

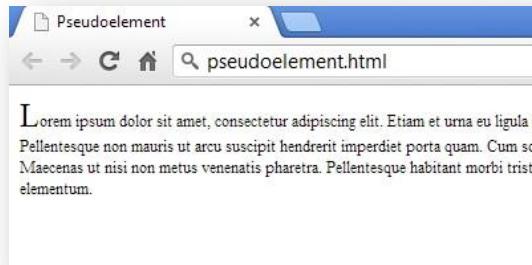
6.12 Pseudoelementi

Pseudoelementi su dijelovi html dokumenta koji nisu određeni elementom ali su njegovi dijelovi. Zbog toga pseudoelementi moraju stajati uz selektor elementa. Neki od pseudoelemenata su **:first-line** i **:first-letter**. Pseudoelement **:first-line** selektira prvi red teksta unutar nekog elementa, dok će pseudoelement **:first-letter** selektirati će prvo slovo unutar teksta elementa.

```
body {  
    font-size: 12px;  
}  
p:first-line {  
    font-size: 16px;  
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam et urna eu ligula fermentum pulvinar. Quisque quis justo lorem. Suspendisse porttitor lobortis erat ut posuere. Vestibulum hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a elit. Aliquam quis nisi sit amet nibh malesuada sagittis. Pellentesque non metus ut arcu suscipit hendrerit imperdiet porta quam. Cum sociis natoque perniciens et magnis dis parturient montes, nascetur ridiculus mus. Nulla facilisi. Pellentesque nec elit dolor, nec mollis urna. Quisque volutpat, mi sed datus fermentum blandit, leo quam convallis purus, sed scelerisque arcu lacus id odio. Maecenas ut nisi non metus venenatis pharetra. Pellentesque habitant morbi tristis et netus et malesuada fames ac turpis egestas. Vestibulum aliquet, neque vel tempor vulputate, tortor ligula auctor justo, in scelerisque dolor ante vel tortor. Morbi ut enim justo, a semper nibh. Cras eget orci eu massa adipiscing elementum.

```
body {  
    font-size: 12px;  
}  
p:first-letter {  
    font-size: 26px;  
}
```



6.13 Selektori atributa

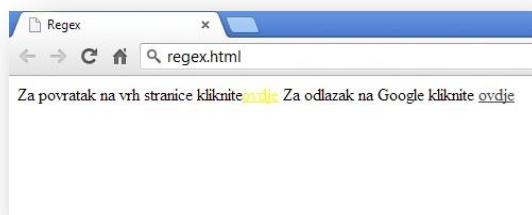
Od verzije CSS2 omogućeno je selektiranje nekog elementa s obzirom na njegove atribute. S time je omogućeno uređivanje istih elementa sa različitim atributima. Selektori atributa označavaju se uglatim zagradama [i] unutar kojih ide ime i vrijednost atributa.

```
a [title="Naslov"] {color: yellow;}
```

U navedenom primjeru selektirani su svi elementi koji imaju atribut title riječ naslov, boja tih elemenata će biti žuta.

Koristeći se sintaksom regularnih izraza moguće je selektirati i posebne slučajevе vrijednosti atributa (na primjer atributi koji sadrže ili počinju nekim tekstrom)

```
a {color: yellow;}  
a[href^="http:"] {color: #00ffff;} /*boja vanjskih veza*/
```



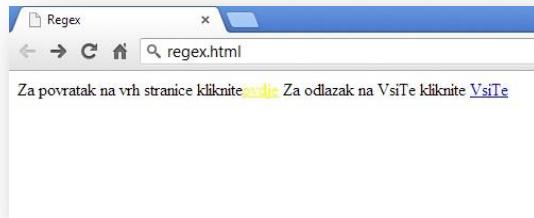
Oznaka ^ (kapica) ispred znaka jednakosti označava tekst na početku vrijednosti atributa

mora odgovarati tekstu u navodnicima selektora.

[atribut \sim =vrijednost] odgovara elementima koji u vrijednosti atributa imaju više vrijednosti od kojih je jedna vrijednost jednaka „vrijednost“.

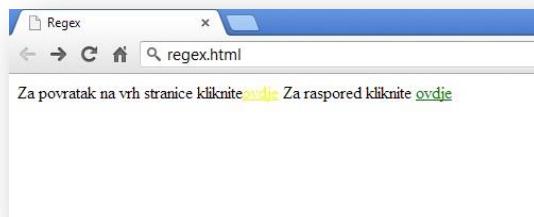
```
a[rel~="tehnologije"] {color: blue;}
```

Ovaj primjer će selektirati sve veze koje u svom rel atributu imaju riječ „tehnologije“.



Ukoliko se želi selektirati element čija vrijednost završava sa određenom vrijednošću koristi se znak \$.

```
a[href$=".pdf"] {color: green;}
```



Ovim primjerom selektira se veza koja vodi na pdf datoteku.

6.14 Specificity

Specificity je mehanizam unutar CSS-a koji služi za lakše rješavanje mogućih problema u slučaju nasljeđivanja. Koristeći specificity moguće je odrediti način nasljeđivanja svojstava. Kao što je prikazano na slici najmanju vrijednost imaju elementi, nakon toga idu klase, pseudoklase i atributi, nakon toga ID, te najveću vrijednost imaju inline atributi unutar <style> oznake. Vrijednosti se računaju kao kod pretvorbe binarnih brojeva u dekadske, skroz desno su brojevi manje vrijednosti dok su sa lijeve strane brojevi veće vrijednosti. Na temelju izračuna konačne vrijednosti postavljaju se CSS pravila.

```
p { color: #fff; /*0,0,0,1*/
.intro { color: #98c7d4; /*0,0,1,0*/
#header { color: #444245; /*0,1,0,0*/
<h1 style="color:#000;">Naslov</h1> /*1,0,0,0*/
```

```
#content p {
    color: blue;
}
p {
    color: lightblue;
}
#content .intro {
    color: yellow;
}
```

```
<div id="content">
    Sadržaj
    <p class="intro">Tekst unutar klase</p>
    <p>Tekst unutar p elementa</p>
</div>
```

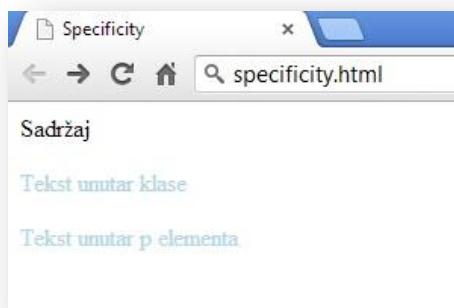


6.14.1 Important

Svojstvo ***!Important*** može promijeniti redoslijed nasljeđivanja unutar CSS pravila. Ponekad je potrebno koristiti to svojstvo kako bi bili sigurni da će se neko svojstvo sigurno postaviti no nije ga preporučljivo koristiti uvijek. Kada se koristi mora se posebno paziti da li se zaista želi postaviti ili se problem koji se želi izbjegći korištenjem svojstva important može drugačije riješiti.

```
#content p {
    color: blue;
}
p {
    color: lightblue !important;
}
#content .intro {
    color: yellow;
}
```

```
<div id="content">
    Sadržaj
    <p class="intro">Tekst unutar klase</p>
    <p>Tekst unutar p elementa</p>
</div>
```



6.15 Razmještaj elemenata

Prije pojave CSS-a postojala su dva načina razmještaja elemenata. Koristeći framesetove te koristeći tablice. Oba načina su imali velike probleme prilikom pregledavanja stranica na ekranima različitih rezolucija. Uz to niti jedna od ta dva načina nije bio u potpunosti prihvatljiv jer su koristili previše internetskog prostora što je uzrokovalo prevelike promete koji su u to vrijeme bili vrlo ograničeni zbog brzina veza sa internetom. Pozicioniranje elemenata može se podijeliti u nekoliko grupa. Jedna podjela je prema širini cjelokupne stranice:

- stranice absolutne širine (širina je točno zadana nekom mjernom jedinicom npr. pikselima) te
- stranice relativne širine (širina ovisi o prozoru preglednika ili ekrana računala).

Druga podjela je prema broju stupaca u kojima se nalazi sadržaj stranice pa tako postoje stranice koje imaju:

- jedan stupac,
- dva stupca,
- tri stupca te
- više stupaca.

Kod izrade web stranica vrlo je bitno pozicionirati sadržaj u sredinu stranice odnosno ekrana. Postoji nekoliko načina za to no jedan od jednostavnijih je postaviti isti razmak od lijevog i desnog ruba stranice do sadržaja stranice. Za pozicioniranje stranica se koriste **<div>** HTML elementi koji uvelike olakšavaju uređivanje cijelih web stranica.

```
<h1>Centriranje stranice</h1>
<div id="wrapper">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam et urna eu ligula fermentum
    pulvinar. Quisque quis justo lorem. Suspendisse porttitor lobortis erat ut posuere. Vestibulum
    hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a elit. Aliquam
    quis nisl sit amet nibh malesuada sagittis. Pellentesque non mauris ut arcu suscipit hendrerit
    imperdiet porta quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur
    ridiculus mus. Nulla facilisi. Pellentesque nec elit dolor, nec mollis urna. Quisque volutpat, mi
    sodales fermentum blandit, leo quam convallis purus, sed scelerisque arcu lacus id odio. Maecenas
    ut nisi non metus venenatis pharetra. Pellentesque habitant morbi tristique senectus et netus et
    malesuada fames ac turpis egestas. Vestibulum aliquet, neque vel tempor vulputate, tortor ligula
    auctor justo, in scelerisque dolor ante vel tortor.
</div>
```

```
#wrapper {
    background-color: #eeeeee;
}
```

Centriranje stranice

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam et urna eu ligula fermentum pulvinar. Quisque quis justo lorem. Suspendisse porttitor lobortis erat ut posuere. Vestibulum hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a elit. Aliquam quis nisl sit amet nibh malesuada sagittis. Pellentesque non mauris ut arcu suscipit hendrerit imperdiet porta quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla facilisi. Pellentesque nec elit dolor, nec mollis urna. Quisque volutpat, mi sodales fermentum blandit, leo quam convallis purus, sed scelerisque arcu lacus id odio. Maecenas ut nisi non metus venenatis pharetra. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum aliquet, neque vel tempor vulputate, tortor ligula auctor justo, in scelerisque dolor ante vel tortor.

Ovim primjerom je vidljivo da je stranica već centrirana, no razlog tomu je div element koji kao i bilo koji drugi element uvijek zauzima 100% širine ako se ne navede proizvoljna širina elementa.

```
#wrapper {  
    background-color: #eeeeee;  
    width: 400px;  
}
```

Ukoliko se postavi širina div elementa na neku vrijednost stranica više nije centrirana te je prema automatskim postavkama pomaknuta prema lijevom rubu ekrana.

Centriranje stranice

Lore ipsum dolor sit amet, consectetur adipiscing elit. Etiam et urna eu ligula fermentum puhinar. Quisque quis justo lorem. Suspendisse porttitor lobortis erat ut posnere. Vestibulum hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a élit. Aliquam quis nisl sit amet nibh malesuada sagittis. Pelentesque non mauris ut arcu suscipit hendrerit imperdēt porta quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla facilisi. Pelentesque nec élit dolor, nec mollis urna. Quisque volutpat, mi sodales fermentum blandit, leo quam convallis purus, sed scelerisque arcu lacus id odio. Maecenas ut nisi non metus venenatis pharetra. Pelentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum aliquet, neque vel tempus vulputate, tortor ligula auctor justo, in scelerisque dolor ante vel tortor.

Ako se sadržaj želi postaviti u sredinu stranice potrebno je podesiti margin. Ukoliko se margin postane na vrijednost **auto** time se postiže automatsko centriranje sadržaja.

```
#wrapper {  
    background-color: #eeeeee;  
    width: 400px;  
    margin: auto;  
}
```

Centriranje stranice

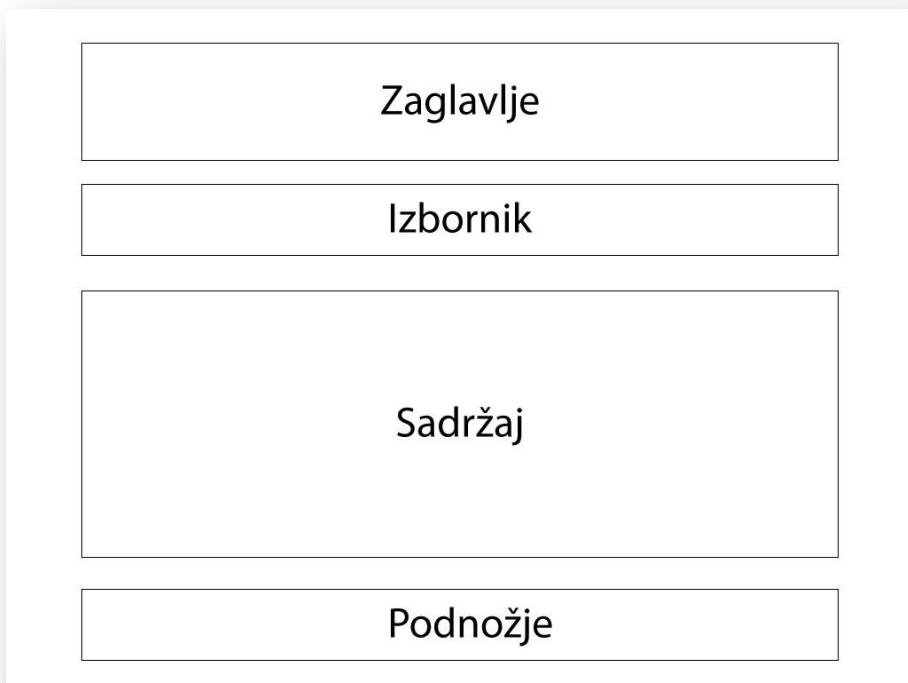
Lore ipsum dolor sit amet, consectetur adipiscing elit. Etiam et urna eu ligula fermentum puhinar. Quisque quis justo lorem. Suspendisse porttitor lobortis erat ut posnere. Vestibulum hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a élit. Aliquam quis nisl sit amet nibh malesuada sagittis. Pelentesque non mauris ut arcu suscipit hendrerit imperdēt porta quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla facilisi. Pelentesque nec élit dolor, nec mollis urna. Quisque volutpat, mi sodales fermentum blandit, leo quam convallis purus, sed scelerisque arcu lacus id odio. Maecenas ut nisi non metus venenatis pharetra. Pelentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum aliquet, neque vel tempus vulputate, tortor ligula auctor justo, in scelerisque dolor ante vel tortor.

U primjeru za apsolutnu širinu elementa postavljena je na 400px no u praksi je puno veće i obično iznosi 964 (za ekrane širine 1024px) no sve češće se koriste i veće širine jer ekran imaju veću rezoluciju.

6.15.1 Izrada stranice koristeći jedan stupac

Prije početka izrade dizajna potrebno je skicirati dizajn stranice te na taj način olakšati

kasniju izradu i pozicioniranje elemenata. Skiciranje nije nužno uvijek potrebno no preporučljivo je jer može znatno olakšati kasniji rad.



Na skici se vidi da je stranica podijeljena na 4 logička dijela:

- zaglavlje,
- izbornik,
- sadržaj te
- podnožje.

U zaglavljtu se najčešće stavlja naslov stranice, logotip te ponekad izbornik koji se može nalaziti i kao poseban element ispod zaglavljta. Izbornik se sastoji od poveznica na druge stranice unutar istog web site-a ili vanjske poveznice. U sadržaju se nalazi sadržaj stranice i sve važne informacije koje se mijenjaju ovisno o stranici koja je odabrana. U podnožju se nalaze podaci o autoru i autorskim pravima stranice te obično o godini izrade stranice.

Logičke cjeline na skici potrebno je pretvoriti u HTML div elemente koji se mogu nazvati:

- wrapper (container unutar kojeg će biti postavljena cijela stranica),
- header (prostor za zaglavlje),
- navigation (prostor za izbornik),
- content (prostor za sadržaj) i
- footer (prostor za podnožje).

```
<div id="wrapper">
    <div id="header"></div>
    <div id="navigation"></div>
    <div id="content"></div>
    <div id="footer"></div>
</div>
```

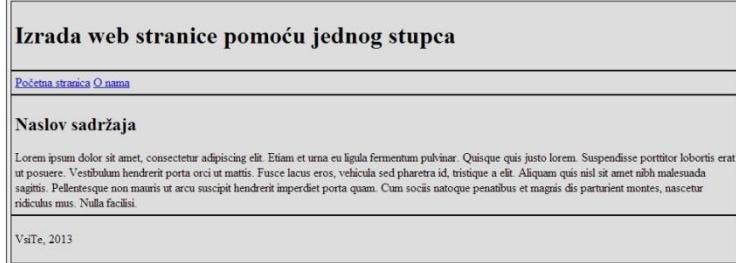
Nakon kreiranja HTML strukture potrebno je napraviti i CSS kôd. Wrapper ima fiksnu dimenziju od 964px jer je stranica predviđena za ekran veličine 1024px. Svi elementi su odmaknuti od ruba 5px dok su međusobno elementi odmaknuti isto 5px.

```
#wrapper {
    background-color: #eeeeee;
    width: 964px;
    margin: auto;
    padding: 5px;
    border: 1px solid black;
}
#header, #navigation, #content, #footer {
    background-color: #dddddd;
    padding: 5px;
    border: 1px solid black;
}
```

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Centriranje</title>
    </head>
    <body>
        <div id="wrapper">
            <div id="header">
                <h1>Izrada web stranice pomoću jednog stupca</h1>
            </div>
            <div>
                <a href="#">Početna stranica</a>
                <a href="#">O nama</a>
            </div>
            <div id="content">
                <h2>Naslov sadržaja</h2>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam et urna eu ligula fermentum pulvinar. Quisque quis justo lorem. Suspendisse porttitor lobortis erat ut posuere. Vestibulum hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a elit. Aliquam quis nisl sit amet nibh malesuada sagittis. Pellentesque non mauris ut arcu suscipit hendrerit imperdiet porta quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla facilisi.
            </div>
            <div id="footer">
                <p>VsiTe, 2013</p>
            </div>
        </div>
    </body>

```



6.15.2 Izrada stranice koristeći dva stupca

Dizajn sa dva stupca je puno češće korišteni u odnosu na dizajn sa jednim stupcem. Koristeći dizajn sa dva stupca unutar dodatnog stupca koji se može nalaziti ili sa lijeve ili sa desne strane sadržaja obično se nalazi ili vertikalni izbornik ili oglasi tj. reklame.



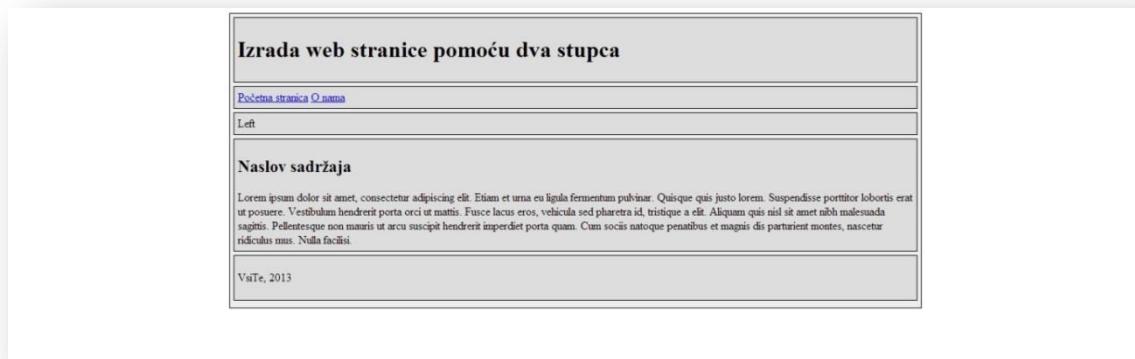
Izgled HTML kôda može izgledati ovako:

```
<div id="wrapper">
    <div id="header"></div>
    <div id="navigation"></div>
    <div id="left"></div>
    <div id="content"></div>
    <div id="footer"></div>
</div>
```

```

#wrapper {
    background-color: #eeeeee;
    width: 964px;
    margin: auto;
    padding: 5px;
    border: 1px solid black;
}
#header, #navigation, #content, #footer, #left {
    background-color: #dddddd;
    padding: 5px;
    border: 1px solid black;
}

```

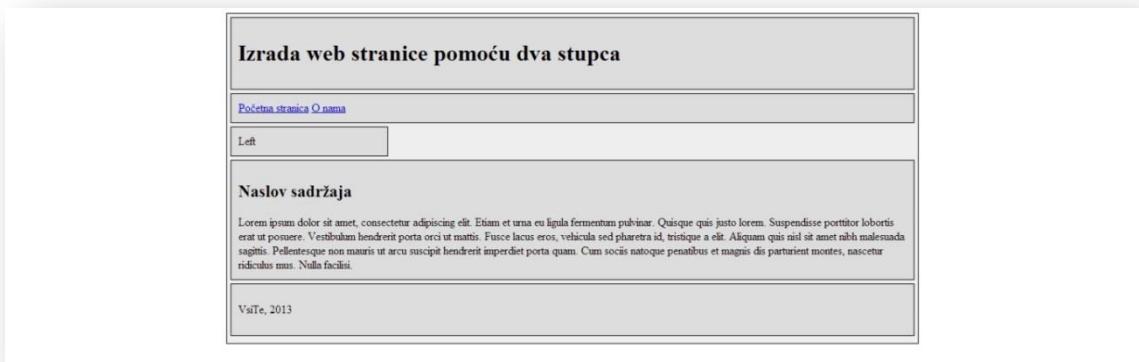


Kao što je vidljivo na slici ukoliko se ne postavi širina lijevog stupca automatski se postavlja na 100% roditelja, tj. postavlja se punu širinu ekrana ukoliko ne postoji roditelj. Ukoliko se postavi fiksna širina lijevog stupca izgled stranice se mijenja.

```

#wrapper {
    background-color: #eeeeee;
    width: 964px;
    margin: auto;
    padding: 5px;
    border: 1px solid black;
}
#header, #navigation, #content, #footer, #left {
    background-color: #dddddd;
    padding: 5px;
    border: 1px solid black;
}
#left {
    width: 200px;
}

```

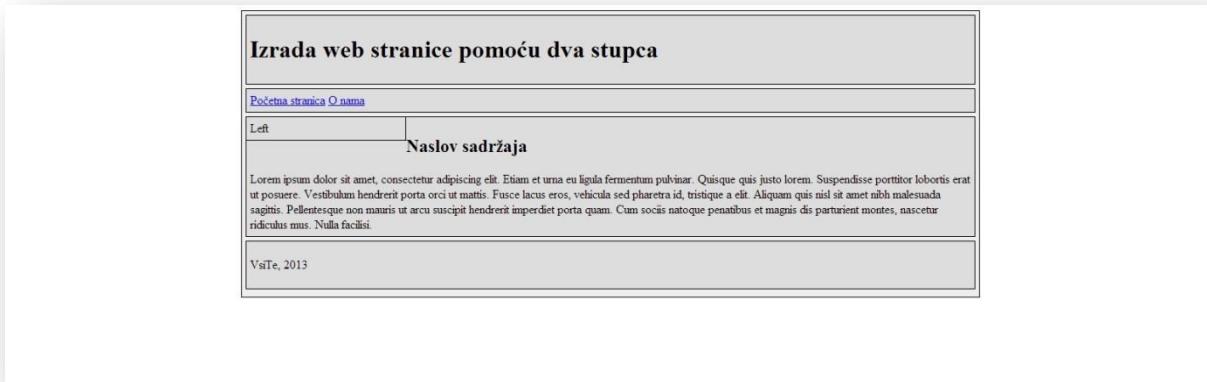


Kao što je vidljivo na slici iako je postavljena fiksna širina lijevog stupca i dalje stupac sa sadržajem se nalazi ispod njega.

6.15.2.1 Svojstvo **float** i **clear**

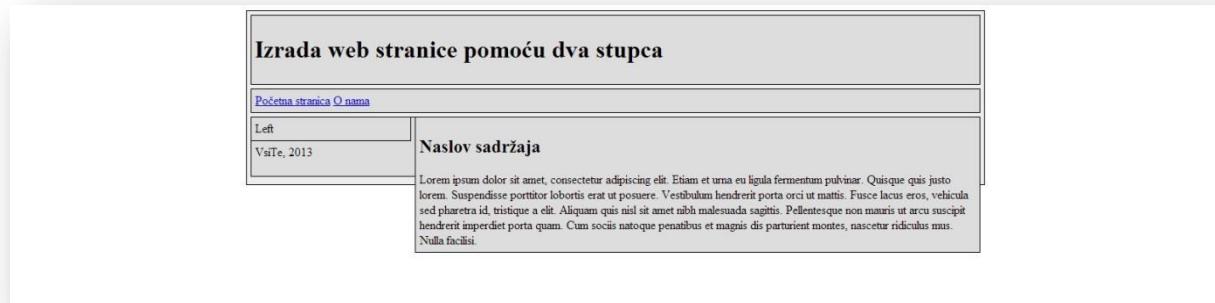
Svojstvo **float** omogućuje kako bi se ciljani element postavio ili uz lijevi ili uz desni rub ekranra. Ukoliko na više elemenata postavimo svojstvo **float** to neće imati utjecaja na konačni izgled. Ako se na lijevi stupac postavi svojstvo **float** na vrijednost **left** rezultat će biti kao na slici.

```
#left {  
    width: 200px;  
    float: left;  
}
```



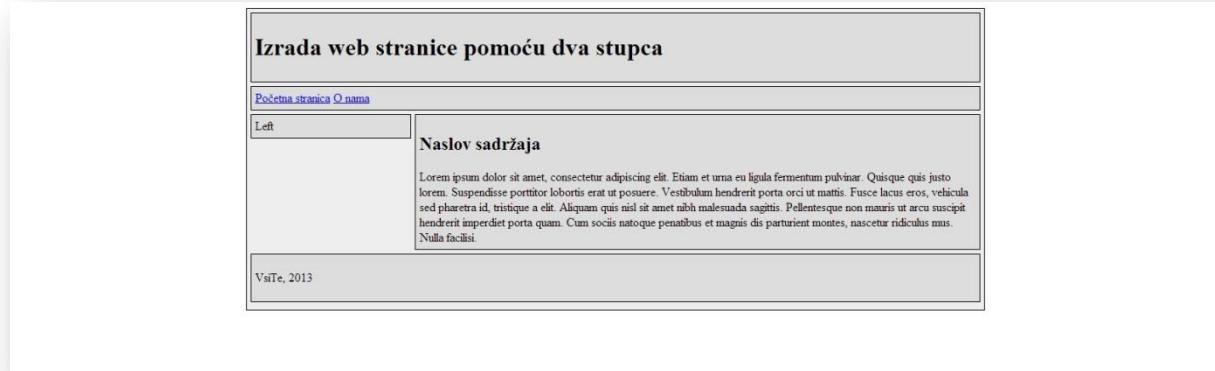
Kako bi se postavio sadržaj s desne strane potrebno je postaviti i širinu elementa unutar kojeg se nalazi sadržaj. Širina elementa se može izračunati koristeći sljedeću formulu: $964 - 200 - 4 - 20 = 740$ od čega je 200 piksela širina lijevog stupca 4 je obrub po 2 piksela sa svake strane i 25 je padding, 15 piksela lijeve i 10 piksela sa desne strane.

```
#content {  
    width: 735;  
    float: right;  
}
```



Korištenjem fiksnih širina oba stupca postiže se postavljanje obaju stupaca kako je planirano no zadnji dio odnosno footer se nalazi ispod sadržajnog stupca što se želi izbjegići. Kako bi se to izbjeglo potrebno je koristiti svojstvo **clear** koje ima tri vrijednosti **left**, **right** i **both**. Postavljanjem svojstva **clear** i vrijednosti **both** na footer postiže se konačni izgled stranice.

```
#footer {  
    clear: both;  
}
```



6.15.3 Izrada stranice koristeći tri stupca

Na internetu najčešće susrećemo stranice koje koriste dizajn sa tri stupca. Kao i kod dizajna sa dva stupca, dodatni stupci sadrže dodatne informacije kao što su reklame ili kalendarili drugo. Uobičajeno je da se dodatni stupac nalazi ili s lijeve ili s desne strane sadržajnog stupca.



Izgled HTML kôd može izgledati ovako:

```
<div id="wrapper">
    <div id="header"></div>
    <div id="navigation"></div>
    <div id="left"></div>
    <div id="content"></div>
    <div id="right"></div>
    <div id="footer"></div>
</div>
```

Kao i u slučaju sa dizajnom koji koristi dva stupca pozicioniranje elemenata se izvodi koristeći **float** i **clear** svojstvima. Lijevi i sadržajni stupac koriste svojstvo **float** sa vrijednostima **left** dok desni stupac koristi **right** vrijednost svojstva **float**.

```

#wrapper {
    background-color: #eeeeee;
    width: 964px;
    margin: auto;
    padding: 5px;
    border: 1px solid black;
}

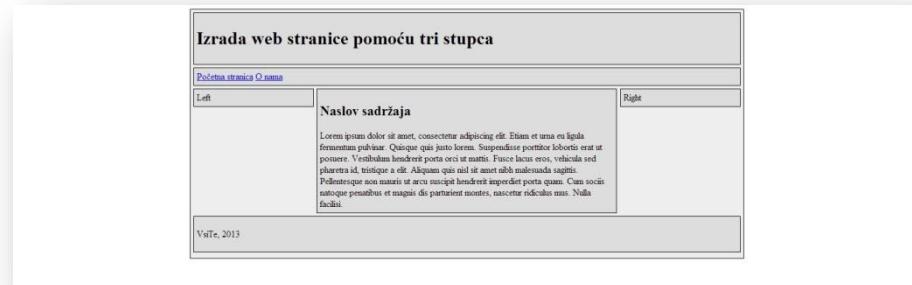
#header, #navigation, #content, #footer, #left, #right {
    background-color: #dddddd;
    padding: 5px;
    margin-bottom: 5px;
    border: 1px solid black;
}

#left {
    width: 200px;
    margin-right: 5px;
    float: left;
}

#content {
    width: 517px;
    float: left;
}

#right {
    width: 200px;
    margin-left: 5px;
    float: right;
}

```



6.16 Pozicioniranje elemenata

Koristeći svojstvo **position** moguće je pozicionirati pojedini element na stranicu ovisno o njegovoј vrijednosti. Uz ovo svojstvo također se koriste svojstva **left**, **right**, **top** i **bottom** pomoću kojih se pože odrediti pozicija nekog elementa u odnosu na navedene strane.

Svojstvo **position** može imati sljedeće vrijednosti:

- static,
- absolute,
- relative te
- fixed.

6.16.1 Static

Vrijednost **static** je predefinirana vrijednost svojstva **position**, na taj način se elementi dodaju jedan za drugim na stranicu. Svojstva **left**, **right**, **top** i **bottom** nemaju utjecaja prilikom korištenja ove vrijednosti.

6.16.2 Absolute

Korištenjem vrijednosti **absolute** elementi se pozicioniraju apsolutno u odnosu na druge elemente tj. na prvi element koji ima poziciju različito od **static** ukoliko takav element ne postoji onda se element pozicionira u odnosu na <html> oznaku.

```
h2 {  
    position: absolute;  
    left: 10px;  
    top: 40px;  
}
```



6.16.3 Relative

Korištenjem svojstva **relative** elementi se pozicioniraju relativno na normalnu poziciju.

```
h2.pos_left {  
    position: relative;  
    left: -20px;  
}
```



6.16.4 Fixed

Kod korištenja svojstva **fixed** potrebno je znati se element postavlja na fiksnu poziciju te će taj element uvijek biti na točno toj poziciji. Prilikom pomicanja kliznih traka (scrollanja) element uvijek ostaje na istoj poziciji.

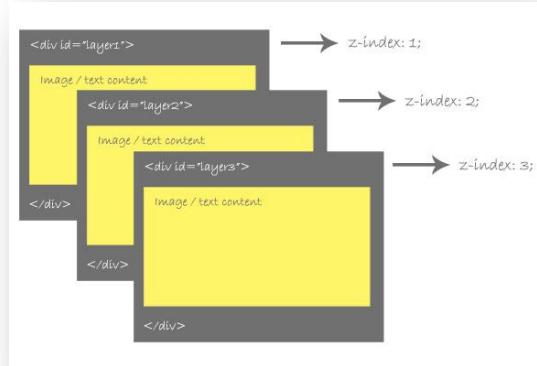
```
h2.pos_fixed {  
    position: fixed;  
    left: 20px;  
    top: 50px;  
}
```



6.16.5 z-index

Uobičajeno HTML stranice su dvodimenzionalne, no ponekad je potrebno elemente složiti i u tri dimenzije, svojstvo **z-index** to omogućava. **Z-index** se može primjenjivati samo kada se primjenjuje svojstvo **position** sa vrijednostima **absolute**, **relative** i **fixed**. Vrijednosti **z-indexa** prikazuju koliko je element „blizu“ korisniku tj. koliko je udaljen.

Naime vrijednosti mogu biti ili negativne što znači da je element u pozadini drugih elemenata ili mogu biti pozitivne vrijednosti što znači da je element iznad svih. Predefinirana vrijednost je **auto**; Ukoliko se koristi nekoliko elemenata koji imaju vrijednost **auto** elementi će se slagati jedan na drugi.



```
#prvi {
    position:fixed;
    left: 20px;
    top: 55px;
    z-index: auto;
    background-color: lightblue;
}

#drugi {
    position:fixed;
    left: 30px;
    top: 70px;
    z-index: auto;
    background-color: red;
}

#treci {
    position:fixed;
    left: 40px;
    top: 85px;
    z-index: auto;
    background-color: lightgreen;
}
```



```
#prvi {  
    ...  
    z-index: 0;  
    ...  
}  
#drugi {  
    ...  
    z-index: 2;  
    ...  
}  
#treci {  
    ...  
    z-index: 1;  
    ...  
}
```



```
#prvi {  
    ...  
    z-index: 3;  
    ...  
}  
#drugi {  
    ...  
    z-index: 0;  
    ...  
}  
#treci {  
    ...  
    z-index: 2;  
    ...  
}
```



6.16.6 Display

Svojstvo **display** određuje na koji će se način HTML element prikazivati. Najčešće vrijednosti ovog svojstva su:

- none,
- block te
- inline.

6.16.6.1 None

Koristeći **none** vrijednost element se ne prikazuje na stranici.

```
#prvi {  
    display: none;  
    border: 1px solid black;  
}
```

```
<h1>Display: none</h1>
<div id="prvi">Prvi element </div>
<div id="drugi">Drugi element </div>
<div id="treci">Treći element </div>
```

Display: none

Drugi element
Treći element

Kao što je vidljivo na slici prvi element koji postoji unutar HTML dokumenta se ne prikazuje na stranici jer mu se postavljeno svojstvo **display** na vrijednost **none**.

6.16.6.2 Block

Koristeći **block** vrijednost element se prikazuje kako je inače i predviđeno tj. širina se postavlja na širinu elementa koji je iznad njega. Elementi div, p, ul, ol, li i od h1 do h6 imaju predefinirane vrijednosti **block**. Ponašaju je kao da je postavljen prijelom retka prije i poslije elementa.

```
#prvi {
    display: block;
    border: 1px solid black;
}
```

Display: block

Prvi element
Drugi element
Treći element

6.16.6.3 Inline

Koristeći **inline** vrijednost element zauzima mesta samo onoliko koliko mu je potrebno tj. koliko je velik sadržaj elementa. Obično se nalaze unutar nekog „block“ elementa. Elementi span, a, img, em, strong imaju predefinirane vrijednosti **inline**. Ne rade prijelom retka prije i poslije elementa.

```
#prvi {  
    display: block;  
    border: 1px solid black;  
}  
.element {  
    display: inline;  
    border: 1px solid black;  
}  
p {  
    border: 1px solid black;  
}
```

```
<h1>Display: inline</h1>  
<div id="prvi">Prvi element  
    <div class="element">Drugi element unutar prvog elementa </div>  
    <p>Treći element</p>  
</div>
```

Display: inline

Prvi element	Drugi element unutar prvog elementa
Treći element	

6.16.7 Visibility

Svojstvo **visibility** za razliku od svojstva **display** ostavlja element „u toku“ bez obzira da li je elementi vidljiv ili ne. Postoji nekoliko mogućih vrijednosti:

- visible,
- hidden te
- inherit.

Visible je predefinirana vrijednost s kojom je element vidljiv.

```
#prvi {  
    visibility: visible;  
    border: 1px solid black;  
}
```

Visibility: visible

Prvi element
Drugi element
Treći element

Ukoliko se postavi vrijednost **hidden** element više nije vidljiv no i dalje zauzima mjesto na stranici i dalje je „u toku“ stranice.

```
#prvi {  
    visibility: hidden;  
    border: 1px solid black;  
}
```

Visibility: hidden

Drugi element
Treći element

Inherit vrijednost je vrijednost koja je naslijedjena od roditelja tj. od elementa koji je iznad elementa koji se trenutno koristi.

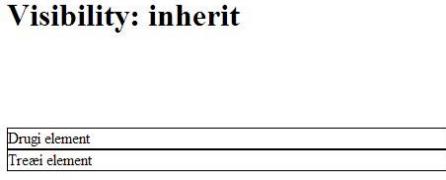
```
#prvi {  
    visibility: visible;  
    border: 1px solid black;  
}  
  
p {  
    color: yellow;  
    visibility: inherit;  
}
```

Visibility: inherit

Prvi element
Neki tekst unutar p elements
Drugi element
Treći element

```
#prvi {  
    visibility: hidden;  
    border: 1px solid black;  
}  
  
p {  
    color: yellow;  
    visibility: inherit;  
}
```

Visibility: inherit



Drugi element
Treći element

6.17 CSS3

Verzija CSS3 donosi mnoge novosti u svakom dijelu dizajna. U prošlosti korisnici su morali biti prisiljeni duže čekati učitavanje stranice ako su htjeli vizualno bogatu stranicu ili su mogli imati stanicu koja se brzo otvarala no nije imala previše multimedije. Uz pomoću CSS verzije 3 više nema potrebe za kompenzacijom. Sa samo nekoliko linija kôda (bez slika) moguće je napraviti zaobljene rubove elemenata, pozadinske gradijente, sjene nad tekstrom i elementima prilagođenu tipografiji u drugo. Također moguće je napraviti i osnovnu interakciju koja je do sada bila moguća uz korištenje JavaScript-a. CSS3 je u potpunosti kompatibilan sa prethodnim verzijama te se time ne gubi funkcionalnost web stranica koje su napisane starijim verzijama CSS-a. Od nove verzije CSS je podijeljen na module dok su prošle verzije bile podijeljene na manje dijelove. Najvažniji moduli unutar CSS3 su:

- selektori,
- model kutije,
- pozadina i obrubi,
- tekstualni efekti,
- 2D i 3D transformacije,
- animacije,
- više stupčasti dizajn te
- korisničko sučelje.

Većina današnjih preglednika ima već u sebi uključenu podršku za veliki broj CSS3 svojstava.

6.17.1 Oznake proizvođača

Unutar CSS3 postoje oznake koje označavaju proizvođača preglednika te na taj način

omogućuju svojstvima da se prikazuju jednakim u svakom pregledniku. Neke od oznaka:

- -khtml- Konqueror (Linux),
- -rim- RIM (BlackBerry),
- -ms- Microsoft,
- -o- Opera,
- -moz- Mozilla (Firefox) te
- -webkit- Webkit (Safari i Chrome).

6.17.2 Svojstvo border-radius

Jedna od novih svojstava koji se mogu koristiti unutar CSS3 je zaobljenje rubova okvira elemenata.

```
#box {  
    border-radius: 5px;  
}
```

Svojstvo border-radius

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam et urna eu ligula fermentum pulvinar. Quisque quis justo lorem. Suspendisse porttitor lobortis erat ut posuere. Vestibulum hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a elit. Aliquam quis nisi sit amet nibh malesuada sagittis. Pellentesque non mauris ut arcu suscipit hendrerit imperdiet porta quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla facilisi. Pellentesque nec elit dolor, nec mollis urna.

Ukoliko se želi napraviti zakrivljenost rubova nad nekim elementom uz potpunu sigurnost da će se tako prikazivati na svakom pregledniku potrebno je napraviti svojstvo **border-radius** iza oznake proizvođača preglednika.

```
#box {  
    background-color: #00eeee;  
    padding: 10px;  
    border-radius: 15px;  
    -webkit-border-radius: 15px; /*Chrome, Safari*/  
    -moz-border-radius: 15px; /*Mozilla*/  
    -ms-border-radius: 15px; /*IE*/  
    -o-border-radius: 15px; /*Opera*/  
}
```

Ukoliko se želi primijeniti zakrivljenost obruba na svakom kutu pojedinačno to se može napraviti na sljedeći način.

```
#box {  
    background-color: #00eeee;  
    padding: 10px;  
    border-radius: 5px 10px 15px 20px;  
}
```

Svojstvo border-radius

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam et urna eu ligula fermentum pulvinar. Quisque quis justo lorem. Suspendisse porttitor lobortis erat ut posuere. Vestibulum hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a elit. Aliquam quis nisl sit amet nibh malesuada sagittis. Pellentesque non mauris ut arcu suscipit hendrerit imperdiet porta quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla facilisi. Pellentesque nec elit dolor, nec mollis urna.

6.17.3 Svojstvo text-shadow

Koristeći svojstvo **text-shadow** moguće je postaviti sjenu ispod teksta te na taj način dobiti izgled elementa kao da je korištena slika. Korištenjem slike povećava se veličina stranice a samim time i povećava vrijeme učitavanja stranice.

Kako bi se moglo koristiti **text-shadow** potrebno je postaviti vrijednosti:

- odmak sjene po x osi,
- odmak sjene po y osi,
- radius zamućenosti sjene te
- boju sjene.

```
text-shadow: x[px] y[px] r[px] #xxxxxx;
```

```
text-shadow: 3px 3px 10px #000000;
```

Svojstvo text-shadow

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam et urna eu ligula fermentum pulvinar. Quisque quis justo lorem. Suspendisse porttitor lobortis erat ut posuere. Vestibulum hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a elit. Aliquam quis nisl sit amet nibh malesuada sagittis. Pellentesque non mauris ut arcu suscipit hendrerit imperdiet porta quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla facilisi. Pellentesque nec elit dolor, nec mollis urna.

6.17.4 Svojstvo box-shadow

Svojstvo ***box-shadow*** je vrlo slično svojstvu kao i ***text-shadow*** no sjena se primjenjuje na element a ne na tekst unutar elementa. Kako bi se moglo koristiti ***box-shadow*** potrebno je postaviti vrijednosti:

- odmak sjene po x osi,
 - odmak sjene po y osi,
 - radijus zamućenosti sjene te
 - boju sjene.

box-shadow: 3px 3px 10px #000000;

Svojstvo box-shadow

Etiam et urna eu ligula fermentum pulvinar. Quisque quis justo lorem. Suspendisse portitor lobortis erat ut posuere. Vestibulum hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a elit. Aliquam quis nisi sit amet nibh malesuada sagittis. Pellentesque non mauris ut arcu suscipit hendrerit imperdiet porta quam. Cum sociis natum penibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla facilisi. Pellentesque nec elit dolor, nec mollis urna.

Ukoliko se koristi zaobljenje rubova okvira sjene će se prilagoditi zaobljenosti kutova.

Svojstvo box-shadow

Etiam et urna eu ligula fermentum pulvinar. Quisque quis justo lorem. Suspendisse porttitor lobortis erat ut posuere. Vestibulum hendrerit porta orci ut mattis. Fusce lacus eros, vehicula sed pharetra id, tristique a elit. Aliquam quis nisl sit amet nibh malesuada sagittis. Pellentesque non mauris ut arcu suscipit hendrerit imperdiet porta quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla facilisi. Pellentesque nec elit dolor, nisl urna.

6.17.5 RGBA boje

Do pojave CSS3 sustav boja koji se koristio u web dizajnu je RGB no u novoj verziji moguće je koristiti i RGBA sustav boja. RGBA je sustav koji uz RGB boje koristi i alfa kanal koji služi za transparentnost. Vrijednost rgb boja je od 0 do 255 dok je vrijednost alfa kanala od 0 (potpuno prozirno) do 1(potpuno neprozirno).

background-color: rgba(xxx,xxx,xxx,0.x);

```
<div style="background-color: rgba(0,0,255,0);">Prvi</div>
<div style="background-color: rgba(0,0,255,0.25);">Drugi</div>
<div style="background-color: rgba(0,0,255,0.5);">Treći</div>
<div style="background-color: rgba(0,0,255,0.75);">Četvrti</div>
<div style="background-color: rgba(0,0,255,1);">Peti</div>
```



6.17.6 HSLA boje

Još jedna novost uz RGBA postavljanja boja je HSLA način postavljanja boja. HSLA koristi nijanse, zasićenost te svjetlinu (parametri hue, saturation/lightness) uz dodatni parametar alfa kanala. Parametar boje se određuje broje od 0 do 359, dok zasićenost i svjetlina koristi postotak od 0 do 100. Alfa kanal kao i kod RGBA se postavlja kao decimalni broj u rasponu od 0 do 1.

```
#box {
    background: hsla(150, 25%, 25%, 0.7);
}
```



6.17.7 Svojstvo font-face

Svojstvo **font-face** je predstavljeno još u verziji 2 no unutar CSS3 donosi niz mogućnosti. Koristeći ovo svojstvo omogućeno je uključivanje vlastitog fonta unutar web stranice što predstavlja velike mogućnosti. Podržani formati za fontove su EOT (Embedded Open Type), TTF (True Type), OTF(Open Type), WOFF (Web Open Font Format) i SVG (Scalable Vector Graphics). Svaki font koji se želi uključiti u web stranicu mora imati se deklarirati svojstvom **@font-face**. Kako bi se mogao deklarirati font potrebno je navesti ime i putanju do datoteke željenog fonta. Putanja datoteke mora biti vidljiva svim stranicama koje žele koristiti novi font.

```

@font-face {
    font-family: 'DS-DIGI';
    src: url('fonts/DS-DIGI.ttf') format('truetype');
    font-weight: normal;
    font-style: normal;
}

#box {
    font-family: 'DS-DIGI';
    font-size: 18px;
}

```

Font-face

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. ETIAM ET URNA EU LIGULA FERMENTUM PULVINAR. QUI SOUE QVIS JUSTO LOREM. SUSPENDISSE PORTTITOR LOBORTIS ERAT UT POSUERE VESTIBULUM HEMORERIT PORTA ORCI UT MATTIS. FUSCE LACUS EROS, VEHICULA SED PHARETRA ID, TRISTIQUE A ELIT. ALIQUAM QUIS NISL SIT AMET NIBH MALESUADA SAGITTIS. PELLentesque non MAURIS UT ARCU SUSCIPIT HEMORERIT IMPERDIT PORTA QUAM. CUM SOCIS MATOQUE PENATIBUS ET MAGNIS DIS PARTURIENT MONTES, NASCETUR RIDICULUS MUS. NULLA FACILISI. PELLentesque NEC ELIT DOLOR, NEC MOLLIS URNA.

6.17.8 Gradijenti

Gradijenti prijelazi boja često se koriste kao pozadina web stranica. Prije postojanja CSS3 jedini način kako se mogla napraviti takva pozadina je bilo korištenje grafičkog elementa sa ponavljanjem po x ili y osi, no CSS3 to znatno pojednostavljuje.

```

#box {
    font-family: 'DS-DIGI';
    font-size: 18px;
    background: linear-gradient(top, #FFFFA6 0%, #BDF271 50%, #01A2A6 100%);
    background: -moz-linear-gradient(top, #FFFFA6 0%, #BDF271 50%, #01A2A6 100%);
    background: -webkit-linear-gradient(top, #FFFFA6 0%, #BDF271 50%, #01A2A6 100%);
}

```

Linear-gradient

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. ETIAM ET URNA EU LIGULA FERMENTUM PULVINAR. QUI SOUE QVIS JUSTO LOREM. SUSPENDISSE PORTTITOR LOBORTIS ERAT UT POSUERE VESTIBULUM HEMORERIT PORTA ORCI UT MATTIS. FUSCE LACUS EROS, VEHICULA SED PHARETRA ID, TRISTIQUE A ELIT. ALIQUAM QUIS NISL SIT AMET NIBH MALESUADA SAGITTIS. PELLentesque non MAURIS UT ARCU SUSCIPIT HEMORERIT IMPERDIT PORTA QUAM. CUM SOCIS MATOQUE PENATIBUS ET MAGNIS DIS PARTURIENT MONTES, NASCETUR RIDICULUS MUS. NULLA FACILISI. PELLentesque NEC ELIT DOLOR, NEC MOLLIS URNA.

Ovim primjerom je prikazan primjer upotrebe višestrukog linearног gradijenta koristeći tri boje koji je predviđen za rad u preglednicima koji podržavaju oznaku **webkit** i **moz**.

```
#box {  
    font-family: 'DS-DIGI';  
    font-size: 18px;  
    background: -webkit-radial-gradient(center, ellipse cover, #BDF271 72%, #01A2A6 100%);  
}
```

Radial-gradient

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. ETIAM ET URNA EU LIGULA FERMENTUM PULVINAR. QUI SOQUE QVIS JUSTO LOREM.
SUSPENDEOISSE PORTTITOR LOBORTIS ERAT UT POSUERE VESTIBULUM
HEMORERIT PORTA ORCI UT MATTIS. FUSCE LACUS EROS, VEHICULA SED
PHARETRA ID, TRISTIQUE A ELIT. ALIOQAM QVIS NISI SIT AMET NIBH
MALESUADA SGITTIS. PELLENTESQUE NON MAURIS UT ARCU SUSCIPIT
HEMORERIT IMPERDIEIT PORTA QURAM CUM SOCIS NATODQUE PENATIBUS ET
MAGNIS DIS PARTURIENT MONTES, NASCETUR RIDICULUS MUS. NULLA
FACILISI. PELLENTESQUE NEC ELIT DOLOR, NEC MOLLIS URNA.

Ovim primjerom je prikazan radijalni gradijent koji koristi centar kao središte gradijenta koji ide prema krajevima u obliku elipse.

```
#box {  
    font-family: 'DS-DIGI';  
    font-size: 18px;  
    background: -webkit-repeating-radial-gradient(2px 2px, ellipse, hsla(0,0%,100%,1)  
    2px, hsla(0,0%,95%,1) 10px, hsla(0,0%,93%,1) 15px, hsla(0,0%,100%,1) 20px);  
}
```

Repeating-radial-gradient

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. ETIAM ET URNA EU LIGULA FERMENTUM PULVINAR. QUI SOQUE QVIS JUSTO LOREM.
SUSPENDEOISSE PORTTITOR LOBORTIS ERAT UT POSUERE VESTIBULUM
HEMORERIT PORTA ORCI UT MATTIS. FUSCE LACUS EROS, VEHICULA SED
PHARETRA ID, TRISTIQUE A ELIT. ALIOQAM QVIS NISI SIT AMET NIBH
MALESUADA SGITTIS. PELLENTESQUE NON MAURIS UT ARCU SUSCIPIT
HEMORERIT IMPERDIEIT PORTA QURAM CUM SOCIS NATODQUE PENATIBUS ET
MAGNIS DIS PARTURIENT MONTES, NASCETUR RIDICULUS MUS. NULLA
FACILISI. PELLENTESQUE NEC ELIT DOLOR, NEC MOLLIS URNA.

Ovim primjerom je prikazan ponavljaјућi radijalni gradijent kojim se postižu složeniji grafički oblici.

6.17.9 Višestruke pozadinske grafike

CSS3 omogućuje korištenje višestruke pozadinske grafike te na taj način nije potrebno koristiti dodatna svojstva. Korištenjem svojstva **background-image** moguće je navesti nekoliko slika koje se žele postaviti u pozadinu. Svojstvo **background-position** određuje poziciju pojedine slike tako što prva riječ podešava horizontalnu poziciju dok druga podešava vertikalnu poziciju. Svaka od slika može se posebno podesiti, svojstva pojedine slike odvajaju se zarezom. Ukoliko se ne želi omogućiti ponavljanje slike niti po jednoj osi potrebno je postaviti svojstvo **background-repeat** na vrijednost **no-repeat**.

```
body {  
    background-image: url(images.jpg),url(logo.jpg);  
    background-position: center center, right top;  
    background-repeat: no-repeat;  
    color: #666666;  
}
```



6.17.10 Višestruki stupci

Koristeći ovo svojstvo moguće je napraviti nekoliko stupaca kao što je često vidljivo u časopisima ili novinama unutar elementa. Kako bi se koristilo ovo svojstvo potrebno je postaviti nekoliko parametara.

- broj stupaca (column-count),
- širina stupca (column-width),
- razmak između stupaca (column-gap) te
- obrub između stupaca (column-rule).

Ukoliko se podesi prevelika širina stupca automatski će se postaviti broj stupaca koliko stane po širini ekrana. Isto ukoliko se ne odredi fiksni broj stupaca a postavi se zadana širina stupca, prilikom promjene širine ekrana broj stupaca će se automatski mijenjati.

```
#box {
    -webkit-column-count: 2;
    -webkit-column-gap: 50px;
    -webkit-column-rule: 1px solid black;
}
```

Multiple column

LOREM IPSUM DOLOR SIT AMET,
CONSECTETUR ADIPISCING ELIT.
ETIAM ET URNA EU LIGULA
FERMENTUM PULVINAR. QUISQUE
QUIS JUSTO LOREM SUSPENDISSE
PORTTITOR LOBORTIS ERAT UT
POSUERE VESTIBULUM
HENDERERIT PORTA ORCI UT
MATTIS. FUSCE LACUS EROS,
VEHICULA SEO PHARETRA ID.
TRISTIQUE A ELIT. ALIQUAM QUIS

MISL SIT AMET NIBH MALESUARD
SAGITTIS. PELLentesque non
MAURIS UT ARCU SUSCIPIT
HENDERERIT IMPERDIT PORTA
QUAM. CUM SOCIIS Natoque
PENATIBUS ET MAGNIS DIS
PARTURIENT MONTES, MASCETUR
RIDICULUS MUS. NULLA FACILISI.
PELLentesque nec ELIT DOLOR,
NEC MOLLIS URNA.

```
#box {
    -webkit-column-width: 100px;
    -webkit-column-gap: 50px;
    -webkit-column-rule: 1px solid black;
}
```

Multiple column

LOREM IPSUM DOLOR
SIT AMET,
CONSECTETUR
ADIPISCING ELIT.
ETIAM ET URNA EU
LIGULA FERMENTUM
PULVINAR. QUISQUE
QUIS JUSTO LOREM.
SUSPENDISSE
PORTTITOR
LOBORTIS ERAT UT
POSUERE
VESTIBULUM

HENDERERIT PORTA
ORCI UT MATTIS.
FUSCE LACUS EROS,
VEHICULA SEO
PHARETRA ID.
TRISTIQUE A ELIT.
ALIQUAM QUIS MISL
SIT AMET NIBH
MALESUARD
SAGITTIS.
PELLentesque non
MAURIS UT ARCU
SUSCIPIT

HENDERERIT
IMPERDIT PORTA
QUAM. CUM SOCIIS
Natoque
PENATIBUS ET
MAGNIS DIS
PARTURIENT
MONTES, MASCETUR
RIDICULUS MUS.
NULLA FACILISI.
PELLentesque nec
ELIT DOLOR, NEC
MOLLIS URNA.

6.17.11 Transformacija elemenata

6.17.11.1 Rotacija elemenata

Pomoću svojstva ***transform*** moguće je izvoditi razne transformacije elemenata. Svojstvu ***transform*** postavlja se vrijednost ***rotate*** kojemu se kao parametar proslijedi iznos rotacije u stupnjevima (deg), gradijanima (grad), radijanima (rad) ili u broju okreta. Kod ovog svojstva bitno je navesti i oznaku proizvođača kako bi se željena transformacija izvršila bez obzira na preglednik.

```
#box {  
    transform:rotate (-15deg);  
    -webkit-transform: rotate(-15deg);  
    -moz-transform: rotate(-15deg);  
}
```

Transform: rotate

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT ETIAM ET
URMA EU LIGULA FERMENTUM PULVINAR. QUISQUE QUS JUSTO LOREM.
SUSPENDISSE PORTTITOR LOBORTIS ERAT UT POSUERE VESTIBULUM.
HOMERERI PORTA ORU UT MATTIS FUSCE LACUS EROS. VEHICULA SE
PHARETRA ID TRISTIQUE A ELIT ALIQUAM QUS NISL SIT AMET NISH
MALESUADA SAGITTIS. PELLentesque NON MAURIS UT ARCA SUSCIPIT
HOMERERI IMPERDIT PORTA QUAH. CUN SOCHIS MATROQUE PENATIBUS ET
MAGNIS DIS PARTURIENT MONTES. MAGNETUR RIDICULUS MUS. NULLA
FRAILISI. PELLentesque NEC ELIT DOLOR. NEC ROLLIS URMA.

6.17.11.2 Promjena veličine

Koristeći vrijednost **scale** svojstva **transform** moguće je promijeniti veličinu elementa na vrlo jednostavan način. Mijenjanje veličine je proporcionalno za širinu i visinu. Parametar koji se prosljeđuje vrijednosti **scale** je decimalni broj. Ukoliko je parametar veći od 1 to znači da će se element povećati, ukoliko je manji od 1 element će se smanjivati.

```
#box {  
    transform:scale (0.5);  
    -webkit-transform: scale(0.5);  
    -moz-transform: scale(0.5);  
}
```

Transform: scale(0.5)

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT ETIAM ET
URMA EU LIGULA FERMENTUM PULVINAR. QUISQUE QUS JUSTO LOREM.
SUSPENDISSE PORTTITOR LOBORTIS ERAT UT POSUERE VESTIBULUM.
HOMERERI PORTA ORU UT MATTIS FUSCE LACUS EROS. VEHICULA SE
PHARETRA ID TRISTIQUE A ELIT ALIQUAM QUS NISL SIT AMET NISH
MALESUADA SAGITTIS. PELLentesque NON MAURIS UT ARCA SUSCIPIT
HOMERERI IMPERDIT PORTA QUAH. CUN SOCHIS MATROQUE PENATIBUS ET
MAGNIS DIS PARTURIENT MONTES. MAGNETUR RIDICULUS MUS. NULLA
FRAILISI. PELLentesque NEC ELIT DOLOR. NEC ROLLIS URMA.

```
#box {  
    transform:scale(1.1);  
    -webkit-transform: scale(1.1);  
    -moz-transform: scale(1.1);  
}
```

Transform: scale(1.1)

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. ETIAM ET
URNA EU LIGULA FERMENTUM PULVINAR. QUISQUE QUIS JUSTO LOREM.
SUSPENDISSE PORTTITOR LOBORTIS ERRAT UT POSUERE. VESTIBULUM
HENDERIT PORTA ORCI UT MATTIS. FUSCE LACUS EROS, VEHICULA SED
PHARETRA ID, TRISTIQUE A ELIT. ALIQUAM QUIS NISL SIT AMET NIBH
MALESUADA SAGITTIS. PELLENTESQUE NON MAURIS UT ARCU SUSCIPIT
HENDERIT IMPERDIET PORTA QUA. CUM SOCII NATOQUE PENATIBUS ET
MAGNIS DIS PARTURIENT MONTES, NASCETUR RIDICULUS MUS. NULLA
FACILISI. PELLENTESQUE NEC ELIT DOLOR, NEC MOLLIS URNA.

6.17.11.3 Promjena položaja

Koristeći vrijednost **translate** svojstva **transform** moguće je promijeniti položaj originalnog elementa po x i y osi. Prvi broj određuje pomicanje po x osi dok drugi broj određuje pomicanje po y osi.

```
#box {  
    transform: translate(20px,50px);  
    -webkit-transform: translate(20px,50px);  
}
```

Transform: translate

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. ETIAM ET
URNA EU LIGULA FERMENTUM PULVINAR. QUISQUE QUIS JUSTO LOREM.
SUSPENDISSE PORTTITOR LOBORTIS ERRAT UT POSUERE. VESTIBULUM
HENDERIT PORTA ORCI UT MATTIS. FUSCE LACUS EROS, VEHICULA SED
PHARETRA ID, TRISTIQUE A ELIT. ALIQUAM QUIS NISL SIT AMET NIBH
MALESUADA SAGITTIS. PELLENTESQUE NON MAURIS UT ARCU SUSCIPIT
HENDERIT IMPERDIET PORTA QUA. CUM SOCII NATOQUE PENATIBUS ET
MAGNIS DIS PARTURIENT MONTES, NASCETUR RIDICULUS MUS. NULLA
FACILISI. PELLENTESQUE NEC ELIT DOLOR, NEC MOLLIS URNA.

6.17.12 Prijelazi

Koristeći ovo svojstvo moguće je napraviti prijelaze iz jednog stila u drugi bez korištenja Flash-a ili JavaScripti. CSS3 prijelazi su efekti koji omogućavaju elementima promjenu stila iz jednog u drugi kroz određeni period vremena. Za to je potrebno postaviti:

- odrediti CSS svojstvo nad kojim se želi primijeniti efekt te
- odrediti vrijeme trajanja efekta.

Postoji nekoliko načina postavljanja prijelaza a oni su sljedeći:

- transition (omogućava podešavanje svih parametara unutar isto reda - shorthand),
- transition-property – element nad kojim se želi izvršiti prijelaz),
- transition-duration – trajanje prijelaza, predefinirana vrijednost je 0),
- transition-timing-function – označava kako će se izračunati brzina prijelaza, predefinirana vrijednost je „ease“ te



- transition-delay – definira vrijeme kada će početi prijelaz, predefinirana vrijednost je 0.

```
#box {  
    width: 100px;  
    height: 100px;  
    background:#7777dd;  
    transition:width 2s;  
    -moz-transition:width 2s; /* Firefox 4 */  
    -webkit-transition:width 2s; /* Safari and Chrome */  
    -o-transition:width 2s; /* Opera */  
}  
#box:hover {  
    width: 400px;  
}
```

Transition



Transition



Isto tako moguće je u isto vrijeme napraviti promjenu nekoliko različitih svojstava.

```
#box {  
    width: 50px;  
    height: 50px;  
    background:#7777dd;  
    transition: width 2s, height 2s, transform 2s;  
    -moz-transition: width 2s, height 2s, -moz-transform 2s;  
    -webkit-transition: width 2s, height 2s, -webkit-transform 2s;  
    -o-transition: width 2s, height 2s,-o-transform 2s;  
}  
#box:hover {  
    width: 100px;  
    height: 100px;  
    transform:rotate(180deg);  
    -moz-transform:rotate(180deg); /* Firefox 4 */  
    -webkit-transform:rotate(180deg); /* Safari and Chrome */  
    -o-transform:rotate(180deg); /* Opera */  
}
```

Transition



Transition



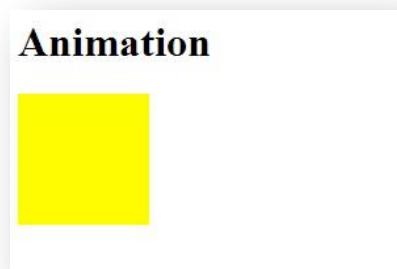
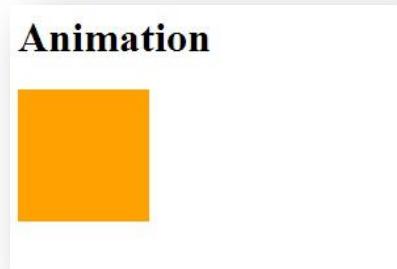
6.17.13 Animacije

Unutar CSS3 moguće je napraviti animacije koje mogu zamijeniti animirane slike, Flash ili JavaScript-e u velikom broju web stranica. Postoje dvije komponente animacije a to su deklaracija keyframe-a i korištenje istog unutar animacijskog svojstva. Prvo je potrebno definirati pravila nakon toga je moguće koristiti animaciju unutar elementa koji se želi animirati.

```
@keyframes myfirst {  
    from {background: red;}  
    to {background: yellow;}  
}  
@-webkit-keyframes myfirst {  
    from {background: red;}  
    to {background: yellow;}  
}  
#box {  
    width:100px;  
    height:100px;  
    background:red;  
    animation:myfirst 5s;  
    -webkit-animation:myfirst 5s;  
}
```

Animation





Kada je kreirana animacija unutar @keyframe-a potrebno ju je povezati sa željenim elementom jer inače se neće ništa dogoditi. Za povezivanje potrebno je navesti ime i duljinu animacije. Duljina animacije se mora unijeti jer u protivnom se neće pokrenuti. Animacija je efekt koji omogućava postepeni promjenu stila elementa iz jednog u drugi. Moguće je promijeniti nebrojeno stilova nebrojeno puta. Postoje dva načina na koji se mogu prikazati animacije:

- koristeći (from to), te
- koristeći (0% 100%).

From predstavlja početak animacije a **to** predstavlja kraj animacije, dok koristeći % 0% predstavlja početak a 100% kraj animacije.

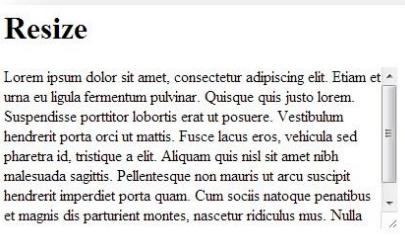
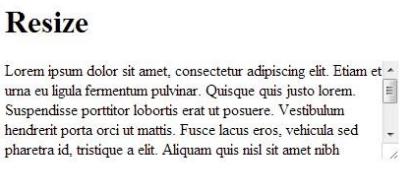
6.17.14 Korisničko sučelje

Unutar CSS3 postoji nekoliko novih mogućnosti prilikom korištenja elemenata za grafičko sučelje.

6.17.14.1 Promjena veličine

Korištenjem svojstva **resize** omogućeno je promjena veličine određenih elemenata.

```
#box {  
    width: 400px;  
    height: 100px;  
    resize: both;  
    overflow: auto;  
}
```



7 HTML

7.1 Općenito o HTML-u

HTML (Hypertext Markup Language) predstavlja prezentacijski jezik za izradu web stranica. Prvi javno dostupan opis HTML-a je dokument zvan HTML tags (oznake), prvi put se spominje na internetu od strane Tim Berners-Leeja krajem 1991. Taj opis se sastoji od 20 elemenata početnog, relativno jednostavnog dizajna HTML-a. Trinaest tih elemenata još uvijek postoji u HTML4. Postanak mnogih svojih oznaka duguje jednom od ranih jezika za formatiranje teksta, runoff-u. Runoff je razvijen u ranim 1960-im za CTSS (Kompatibilni Time-Sharing System) operacijski sustav. Runoff je kasnije inkorporiran u UNIX operativni sustav u naprednije formatirajuće programe kao što su roff, nroff i troff. Svaka nova verzija HTML-a je razvijana tako da ostane čitljiva na svim web preglednicima. Tim Berners-Lee je, nakon što je u listopadu 1994. napustio CERN (Europsku organizaciju za nuklearno istraživanje), osnovao organizaciju World Wide Web Consortium koja se bavi standardizacijom tehnologija korištenih na webu poznatija kao W3C.

Prva verzija HTML jezika objavljena je 1993. godine. U to je vrijeme bio još poprilično ograničen, pa nije bilo moguće čak ni dodati slike u HTML dokumente.

Razvoj HTML-a nastavljen je prvom "imenovanom" verzijom – 2.0, no ni ona nije postala standardom. U ožujku 1995. W3C objavljuje verziju 3.0, koja donosi mogućnosti definicije tablica. Daljnji razvoj ove verzije HTML-a označilo je prihvaćanje "specifičnih" oznaka podržanih u tada najvećim i najprihvaćenijim web preglednicima. Tako su nastale mnoge duplikacije, pa je postojalo više oznaka koje su imale istu funkciju. Podebljani test, primjerice bilo je moguće definirati oznakom ****, ali i oznakom ****.

HTML4 predstavljen je u prosincu 1997., nastavio je s prihvaćanjem oznaka nametnutih od strane proizvođača različitih web preglednika, no istovremeno je pokrenuto i "čišćenje" standarda proglašavanje nekih od njih suvišnim. Manje promjene u specifikaciji ovog standarda predstavljene su u prosincu 1999., kada je predstavljena konačna verzija ovog jezika HTML4.01.

HTML 5 je prva nova revizija standarda od HTML 4.01, koji je izdan 1999. Nastao u suradnji World Wide Web Consortium (W3C) i Web Hypertext Application Technology Working Group (WHATWG). Do 2006. godine su ove dvije grupe radile odvojeno, WHATWG je radio sa web formama i aplikacijama, a W3C sa XHTML 2.0. Na svu sreću odlučili su udružiti snage i kreirati novu verziju HTML-a. Izdavanje konačnih specifikacija standarda HTML5 u suprotnosti je s inicijativom Web Hypertext Application Technology Working Group (WHATWG) prema kojoj bi HTML trebao biti "živi" standard koji se stalno nadograđuje, bez oznake verzije specifikacija. HTML5 standard nalazi se u statusu radnog dokumenta (draft), a očekuje se da će postati službeno objavljen sredinom 2012. godine, dok bi konačne specifikacije trebale biti gotove u drugom kvartalu 2014 . Zanimljivo je da već sada veliki broj preglednika ima implementiran sustav koji omogućuje interpretaciju HTML5.

HTML5 donosi brojne nove mogućnosti koje HTML 4.01 i XHTML 1.x nisu imali, kao što je mogućnost reprodukcije videa na stranicama bez korištenja Adobe flasha ili Microsoftovog silverlighta, mogućnost upravljanja pomoću tipkovnice i opcijama za bilo koju vrstu manipulacija, drag and drop, canvas kao i ostali novi elementi.

7.2 HTML5

HTML 5 je donio revoluciju u smislu web dizajna zajedno sa CSS3 i JavaScript-om koji će biti opisan kasnije. HTML5 ne pripada niti jednoj tvrtci niti pojedinom pregledniku te ga koriste velike kompanije poput Google-a, Microsoft-a, Apple-a, Mozilla, Facebook-a, IBM-a, HP-a, Adobe-a i drugih. Uvođenjem ove verzije HTML-a dobio je potpuno drugo značenje jer donosi web stranicama nove vizualne, audio i video mogućnosti koje prije nitko nije niti mislio da će biti moguće napraviti koristeći samo HTML. Programerima nova verzija omogućava kreiranje aplikacija i web stranica koje imaju brzinu, funkcionalnost i performanse desktop aplikacija. No za razliku od desktop aplikacija, aplikacije koje se rade za web platformu su lakše dostupne krajnjim korisnicima na različitim uređajima. Korisnici također imaju dobrobiti od nove verzije HTML-a jer više ne moraju instalirati svoje omiljene aplikacije na nekoliko različitih uređaja i brinuti se oko nadogradnje na najnoviju verziju iste. Njihovi podaci, posao zabava i sve što rade je dostupno bez obzira s kojeg uređaja se spajaju na stranicu.

Sljedeća generacija web aplikacija može pokretati grafiku visokih performansi, raditi bez spajanja na internet, spremati velike količine podataka na korisnički uređaj, vrlo brzo izvoditi računske operacije te postaviti interakciju na totalno novu razinu.

7.2.1 Novi doctype i skup znakova elementi

Element **doctype** bi trebao biti prvi element unutar HTML dokumenta jer na temelju tog elementa preglednik određuje na koji način će pregledavati dokument. U starijim verzijama element je morao izgledati ovako:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

Unutar HTML5 potrebno je navesti puno kraći zapis:

```
<!DOCTYPE html>
```

Skup znakova je potrebno koristiti kako bi se mogli prikazivati svi znakovi unutar stranice. Ovo je najbitnije koristiti kada se na stranicama koriste posebni znakovi ili neki drugi jezik koji sadrži posebna slova u to spada i naš jezik. Najčešće se koristi UTF-8 jer se unutar njega nalaze gotovo svi znakovi koji se mogu pojaviti u gotovo svakom svjetskom jeziku. Korištenje određenog skupa znakova (charset) u starijim verzijama HTML-a izgledalo je ovako:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Koristeći HTML5 potrebno je navesti samo naziv skupa znakova ili načina kodiranja stranice:

```
<meta charset="utf-8">
```

7.2.2 Novi elementi

Internet i korištenje interneta se puno primjenilo od 1999. godine kada je predstavljena verzija 4.01. Nova verzija HTML-a donosi niz novih elemenata koji su navedeni u tablici.

Tag	Opis
<article>	Defines an article
<aside>	Defines content aside from the page content

<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside it
<command>	Defines a command button that a user can invoke
<details>	Defines additional details that the user can view or hide
<summary>	Defines a visible heading for a <details> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<figcaption>	Defines a caption for a <figure> element
<footer>	Defines a footer for a document or section
<header>	Defines a header for a document or section
<hgroup>	Groups a set of <h1> to <h6> elements when a heading has multiple levels
<mark>	Defines marked/highlighted text
<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links
<progress>	Represents the progress of a task
<ruby>	Defines a ruby annotation (for East Asian typography)
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<rp>	Defines what to show in browsers that do not support ruby annotations
<section>	Defines a section in a document
<time>	Defines a date/time
<wbr>	Defines a possible line-break

7.2.2.1 Novi media elementi

Tag	Opis
<audio>	Defines sound content
<video>	Defines a video or movie
<source>	Defines multiple media resources for <video> and <audio>
<embed>	Defines a container for an external application or interactive content (a plugin)
<track>	Defines text tracks for <video> and <audio>

7.2.2.2 Novi <canvas> element

Tag	Opis
<canvas>	Used to draw graphics, on the fly, via scripting (usually JavaScript)

7.2.2.3 Novi elementi forme

HTML5 nudi nove elemente forme koji pružaju veću funkcionalnost.

Tag	Opis
<datalist>	Specifies a list of pre-defined options for input controls
<keygen>	Defines a key-pair generator field (for forms)
<output>	Defines the result of a calculation

7.2.2.4 Izbačeni elementi

Sljedeći elementi iz HTML 4.01 su uklonjeni:

- <acronym>
- <applet>
- <basefont>
- <big>
- <center>
- <dir>
-
- <frame>
- <frameset>

- <noframes>
- <strike>
- <tt>

Uvođenjem novih elemenata za pozicioniranje olakšava se izrada dizajna cijele web stranice.

Novi elementi koji služe za pozicioniranje elementa su:

- <header> - sadržaj zaglavlja (za stranicu ili dio stranice),
- <footer> - sadržaj podnožja (za stranicu ili dio stranice),
- <section> - dio web stranice,
- <article> - nezavisni sadržaj članka,
- <aside> - povezani sadržaj,
- <nav> - navigacija.

Korištenjem novih elemenata gotovo je nepotrebno koristiti div elemente za pozicioniranje.

Svi navedeni elementi se mogu uređivati unutar CSS-a što omogućuje potpunu prilagodbu dizajna. Primjer jedne stranice uz korištenje novih navigacijskih elemenata.



7.2.3 Multimedija i grafika

Svi mi volimo razne efekte, 3D efekti, laseri, eksplozije. Na to nas je naučila televizija na kojoj gledamo u filmovima razne efekte koji nas oduševljavaju. Od samog početka web je vizualni medij bio ograničen. Programeri koji su htjeli napraviti igre, brze animacije ili kompleksne vizualne efekte morali su prijeći na druge platforme i dodatne plugin-ove. Sa HTML5, web preglednik je postao potpuna platforma za igre, animacije, filmove, sve što je grafičko. Detalji poput osvjetljenja, sjena, refleksije i bogatih tekstura rezultiraju realističnim izgledom. Visoke video performanse poput 3D CSS-a, vektorske grafike (canvas, SVG) i

WebGL-a omogućavaju vrlo brze web aplikacije sa predivnom 3D grafikom i specijalnim efektima. Bogati audio dio i mala latencija mreže zajedno sa grafičkim dijelom pruža predivno iskustvo korisnicima.

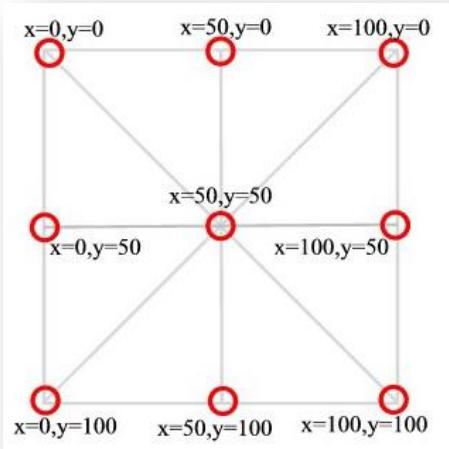


7.2.3.1 Canvas

Canvas je element koji omogućuje postavljanje dinamičkih grafičkih elemenata na web stranicama. Također moguće je i napraviti vlastitu grafiku koristeći razne elemente i JavaScript. Kako bi se mogao koristiti canvas element potrebno je željeni kod staviti unutar <canvas> oznaka.

7.2.3.1.1 Koordinate canvasa

U radu sa **canvas-om** vrlo je bitno poznavanje koordinatnog sustava jer je to temelj rada u korištenju **canvasa**.



Slika pokazuje canvas područje veličine 100 sa 100 piksela.

- gornji lijevi kut ima koordinate x=0, y=0.
- x vrijednost povećava se horizontalno dok se y vrijednost povećava vertikalno.
- donji desni kut ima koordinate x=100, y=100.
- točka u centru ima koordinate x=50, y=50.

Predefinirana vrijednost **canvas-a** je pravokutnih veličine 300 sa 150 piksela no veličinu je moguće postaviti na bilo koju drugu vrijednost koristeći svojstva **height** i **width**. Svaki canvas element može imati i HTML atribute kao što su klasa, id ili ime. Koristeći JavaScript i id canvas-a povezuje se u jedno cjelinu te se na taj način omogućuje korištenje svih mogućnosti.

Kreiranje canvas objekta:

```
<canvas id="diagonal" style="border: 1px solid;" width="200" height="200"></canvas>
```

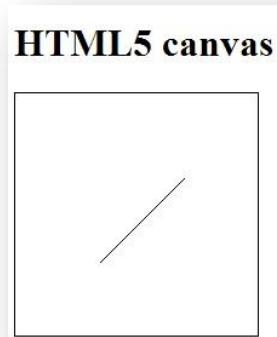
Pomoću JavaScript naredbi moguće je kontrolirati canvas elemente koristeći `document.getElementById()`.

```
var canvas = document.getElementById('diagonal');
var context = canvas.getContext('2d');
```

```

<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>HTML5 basics</title>
        <script>
            function drawDiagonal() {
                // Get the canvas element and its drawing context
                var canvas = document.getElementById('diagonal');
                var context = canvas.getContext('2d');
                // Create a path in absolute coordinates
                context.beginPath();
                context.moveTo(70, 140);
                context.lineTo(140, 70);
                // Stroke the line onto the canvas
                context.stroke();
            }
            window.addEventListener("load", drawDiagonal, true);
        </script>
    </head>
    <header>
        <h1>HTML5 canvas</h1>
    </header>
    <article>
        <canvas id="diagonal" style="border: 1px solid;" width="200" height="200"></canvas>
    </article>
</html>

```



7.2.3.2 Audio element

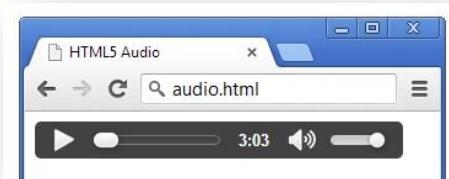
Novost u HTML5 je i korištenje audio elementa moguće je bez dodatnih dodataka (plugin-ova).

Najjednostavnija primjena audio elementa:

```
<audio controls src="johann_sebastian_bach_air.ogg">  
    Vaš preglednik ne podržava audio element.  
</audio>
```

Također moguće je koristiti i više izvora audio datoteka te je moguće staviti vidljive audio kontrole uz pomoću kojih je moguće regulirati glasnoću i poziciju željene audio datoteke.

```
<audio controls>  
    <source src="johann_sebastian_bach_air.ogg" type="audio/ogg; codecs=vorbis">  
    <source src="johann_sebastian_bach_air.mp3" type="audio/mpeg">  
        Vaš preglednik ne podržava audio element.  
</audio>
```



7.2.3.3 Video element

Slično kao i audio element, video element omogućuje direktnu reprodukciju video datoteka unutar HTML dokumenta bez potrebe dodatnih dodataka.

```
<video width="320" height="240" controls>  
    <source src="Intermission-Walk-in_512kb.mp4" type="video/mp4">  
        Vaš preglednik ne podržava video element.  
</video>
```



7.2.3.4 Dodatni elementi

Uz sve navedene novosti uvedeno je niz novih elemenata koje do sada nije bilo moguće koristiti direktno unutar HTML dokumenta. Većina elemenata donosi novosti po pitanju grafičkih elemenata za koje više nije potrebno uključivati razne dodatke kako bi stranica izgledala funkcionalno i moderno.

7.2.3.4.1 <meter>

Uz pomoću elementa <meter> moguće je prikazivati određenu veličinu grafički a ne samo brojčano.

```
<meter min="0" max="100" low="40" high="80" optimum="90" value="100"></meter>
```



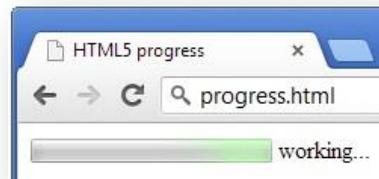
Kao što je vidljivo iz slike moguće je postaviti nekoliko graničnih vrijednosti na kojima će se boja prikazivanja unutar elementa promijeniti te će se na taj način razlika u vrijednostima još više uočiti. Unutar elementa postoji nekoliko svojstava koje je moguće postaviti:

- min – minimalna dopuštena vrijednost elementa,
- max – maksimalna dopuštena vrijednost elementa,
- low – najniža razina,
- high – viša razina,
- optimal – optimalna razina te
- value – vrijednost elementa.

7.2.3.4.2 <progress>

Element **progress** je vrlo sličan elementu **meter** no uz pomoć njega moguće je prikazati radnju koja se odvija (na primjer učitavanje neke video datoteke na stranicu).

```
<progress>working...</progress>  
working...
```



7.2.3.5 Novi elementi unutar forme

7.2.3.5.1 Input Type: color

Element koji služi za odabir boje.

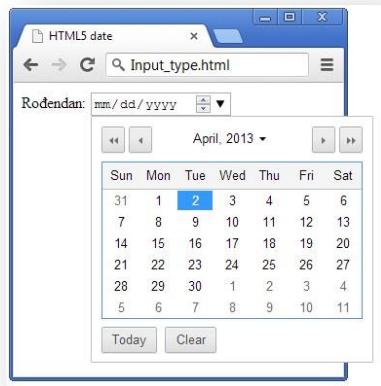
```
Odaberite boju: <input type="color" name="favcolor">
```



7.2.3.5.2 Input Type: date

Element koji služi za odabir datuma. Automatski provjerava datume.

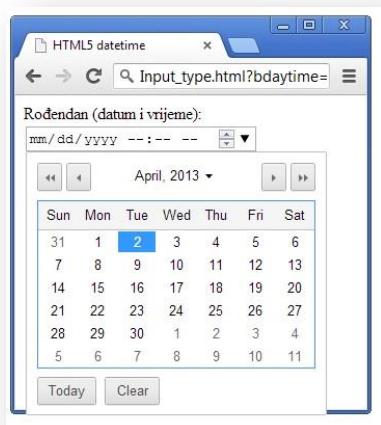
```
Rođendan: <input type="date" name="bday">
```



7.2.3.5.3 Input Type: datetime-local

Element koji služi za odabir lokalnog datuma i vremena. Kao i kod **date** automatski provjerava datume i vremena.

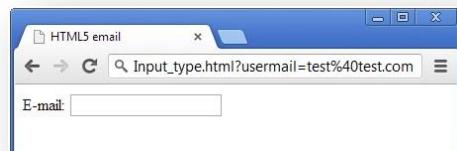
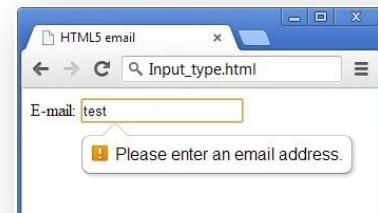
Rođdan (datum i vrijeme): <input type="datetime-local" name="bdaysime">



7.2.3.5.4 Input Type: email

Element koji služi za unos e-mail adrese. Nakon unosa automatski se provjerava da li je unesena adresa valjana.

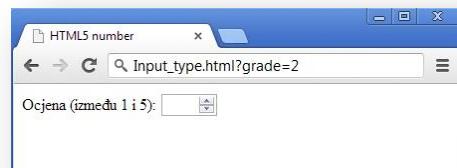
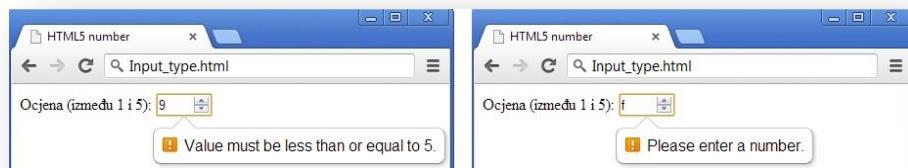
E-mail: <input type="email" name="usermail">



7.2.3.5.5 Input Type: number

Element koji služi za unos samo numeričkih vrijednosti. Unos drugih znakova nije dozvoljen. Moguće je postaviti i ograničenja u kojem rasponu se mogu unašati brojevi.

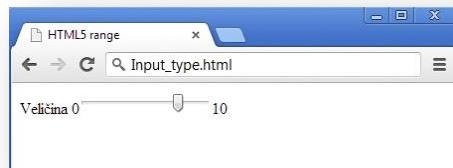
Ocjena (između 1 i 5): <input type="number" name="grade" min="1" max="5">



7.2.3.5.6 Input Type: range

Elementom **range** moguće je ograničiti odabir brojeva iz nekog raspona kao na primjer brojeve od 1 do 10.

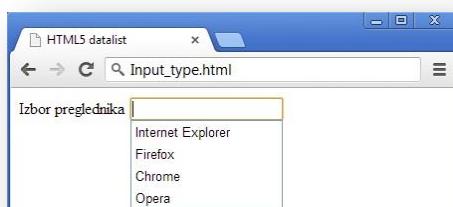
```
Veličina <input type="range" name="velicina" min="1" max="10">
```



7.2.3.5.7 <datalist>

Element **datalist** omogućuje odabir predefiniranih vrijednosti. Također omogućava automatsku ponudu vrijednosti. Vrijednosti koje se nude kao izbor upisane se u atributima elementa.

```
Izbor preglednika <input list="preglednici">
    <datalist id="preglednici">
        <option value="Internet Explorer">
        <option value="Firefox">
        <option value="Chrome">
        <option value="Opera">
    </datalist>
```



7.2.3.5.8 <keygen>

Svrha **keygen** elementa je pružiti korisniku sigurniji način provjere korisnika. Kada se forma potvrди kreiraju se dva ključa, privatni i javni. Privatni ključ je pohranjen lokalno dok se javni ključ šalje serveru. Javni ključ se može iskoristiti za certifikata kako bi klijent mogao u budućnosti pristupati stranici.

```
<form>
    Username: <input type="text" name="usr_name">
    Encryption: <keygen name="security">
    <input type="submit">
</form>
```

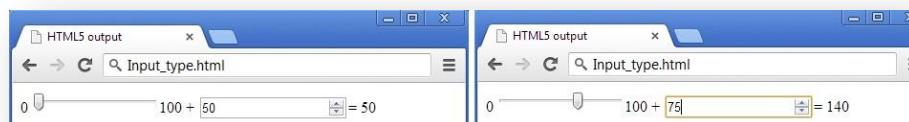


Kao rezultat forma će vratiti par ključ-vrijednost i to vrijednost enkriptiranu ovisno o odabranoj vrijednosti unutar forme.

7.2.3.5.9 <output>

Ukoliko se ne želi koristiti JavaScript-a za izračun nekih jednostavnih izraza moguće je koristiti HTML5 oznaku **output**.

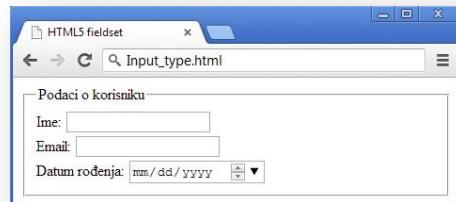
```
<form oninput="x.value=parselnt(a.value)+parselnt(b.value)">
    <input type="range" id="a" value="50">100 +
    <input type="number" id="b" value="50">=
    <output name="x" for="a b"></output>
</form>
```



7.2.3.5.10 <fieldset>

Koristeći element **fieldset** moguće je grupirati niz elemenata tako da čine jednu cjelinu.

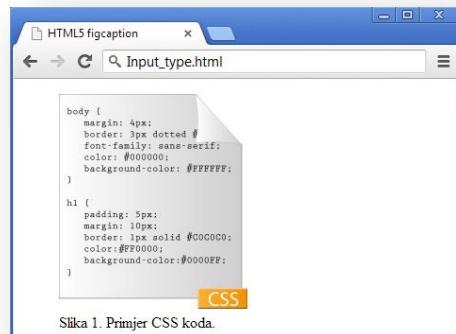
```
<fieldset>
    <legend>Podaci o korisniku</legend>
    Ime: <input type="text"><br>
    Email: <input type="email"><br>
    Datum rođenja: <input type="date">
</fieldset>
```



7.2.3.5.11 <figcaption>

Koristeći element **figcaption** moguće je postavi opis slike koja se nalazi unutar web stranice.

```
<figure>
    
    <figcaption>Slika 1. Primjer CSS koda.</figcaption>
</figure>
```



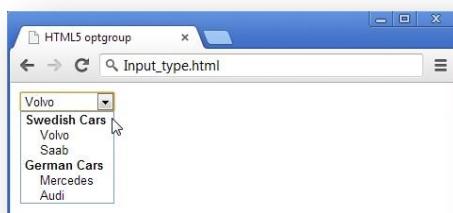
7.2.3.5.12 <optgroup>

Ukoliko postoji potreba za grupiranjem selekcijskog odabira korištenjem elementa **optgroup** moguće je grupirati skupine koje se mogu odabrati iz padajućeg izbornika.

```

<select>
    <optgroup label="Swedish Cars">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
    </optgroup>
    <optgroup label="German Cars">
        <option value="mercedes">Mercedes</option>
        <option value="audi">Audi</option>
    </optgroup>
</select>

```



7.2.3.6 Atributi forme

Novom verzijom HTML-a uvedeno je i nekoliko novih atributa koji se mogu koristiti unutar forme. Neki od njih su:

- ***required***,
- ***autofocus***,
- ***autocomplete***,
- ***novalidate***,
- ***multiple*** te
- ***placeholder***.

7.2.3.6.1 required

Koristeći ovaj atribut moguće je odrediti koji su elementi obavezni za unos te se morma neće moći potvrditi ukoliko se ti elementi ne ispune.

```

<form>
    Korisničko ime: <input type="text" name="username" required>
    <input type="submit">
</form>

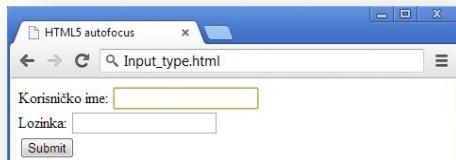
```



7.2.3.6.2 autofocus

Koristeći atribut **autofocus** moguće je odrediti na koji se element želi postaviti fokus prilikom otvaranja stranice.

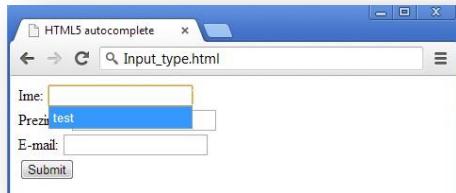
```
<form>
    Korisničko ime: <input type="text" name="username" required autofocus><br>
    Lozinka: <input type="password" name="pass" required><br>
    <input type="submit">
</form>
```



7.2.3.6.3 autocomplete

Koristeći atribut **autocomplete** moguće je zapamtiti prethodne unose u elemente na razini cijele forme ili samo nekog elementa. Isto tako moguće je postaviti pamćenje prethodnih unosa za cijelu formu osim za neke elemente. Ovaj atribut napravo spremi do sada unašane vrijednosti i automatski nudi odabir postojećih vrijednosti prilikom unosa vrijednosti u element.

```
<form autocomplete="on">
    Ime: <input type="text" name="fname"><br>
    Prezime: <input type="text" name="lname"><br>
    E-mail: <input type="email" name="email" autocomplete="off"><br>
    <input type="submit">
</form>
```



7.2.3.6.4 novalidate

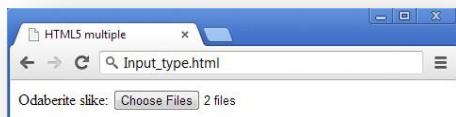
Ukoliko se želi isključiti automatska provjera unesenih podataka (email element ima predefinirano automatsku provjeru unesenih podataka) to je moguće napraviti koristeći atribut **novalidate**.

```
<form novalidate>
    E-mail: <input type="email" name="user_email">
    <input type="submit">
</form>
```

7.2.3.6.5 multiple

Prilikom podizanja (uploadanja) datoteka na stranicu u većini slučajeva moguće je odabrati samo jednu datoteku. To ograničenje može usporiti rad krajnjeg korisnika ukoliko je potrebno odabratи nekoliko datoteka i postaviti ih na stranicu. Koristeći atribut **multiple** omogućeno je označavanje više datoteka odjednom.

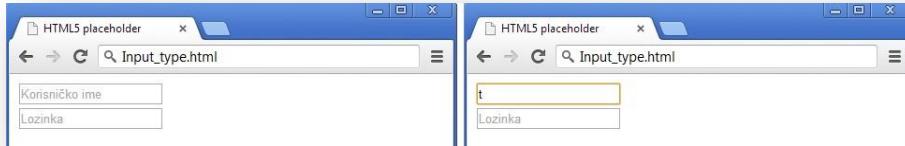
```
Odaberite slike: <input type="file" name="img" multiple>
```



7.2.3.6.6 placeholder

Placeholder je atribut koji prikazuje kratki opis koju vrijednost element očekuje. Opis se prikazuje dok je element prazan i nestaje čim element dobije fokus.

```
<input type="text" name="username" placeholder="Korisničko ime"><br>
<input type="password" name="pass" placeholder="Lozinka"><br>
```



7.2.4 Offline i pohrana

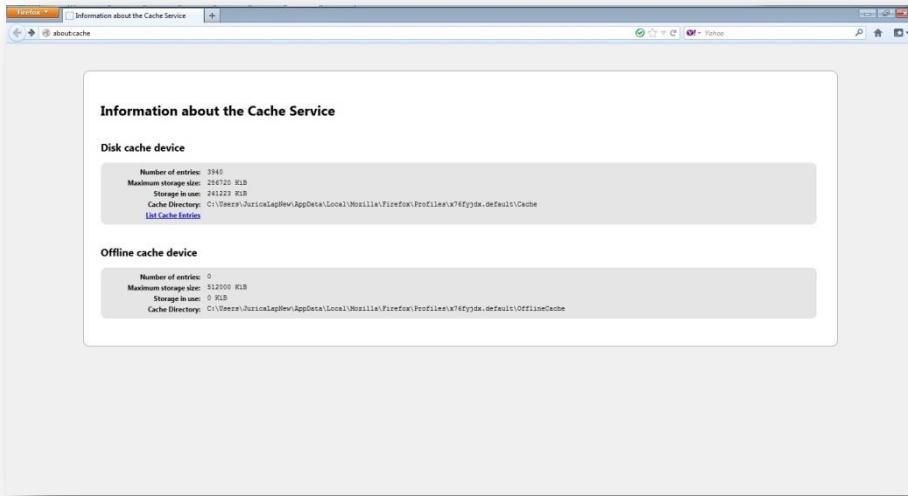
Web i offline su dva termina koja puno korisnika ne mogu staviti u istu rečenicu, no pojavim HTML5 to je moguće. Koristeći lokalnu pohranu aplikacije se mogu koristiti i kada ne postoji veza prema internetu tako što se ili dio podataka pohrani lokalno ili sve kako bi se moglo pregledavati kasnije. Korisnici koriste sve više mobilne uređaje za pristup internetu te samim time može se dogoditi da prilikom putovanja dođe do prekida veze i samim time prekida mogućnosti korištenja web aplikacija no uz pomoć lokalne pohrane, aplikacijskog spremnika ti problemi postaju prošlost.

Primjena pohrane je velika te se može koristiti u aplikacijama kao što su:

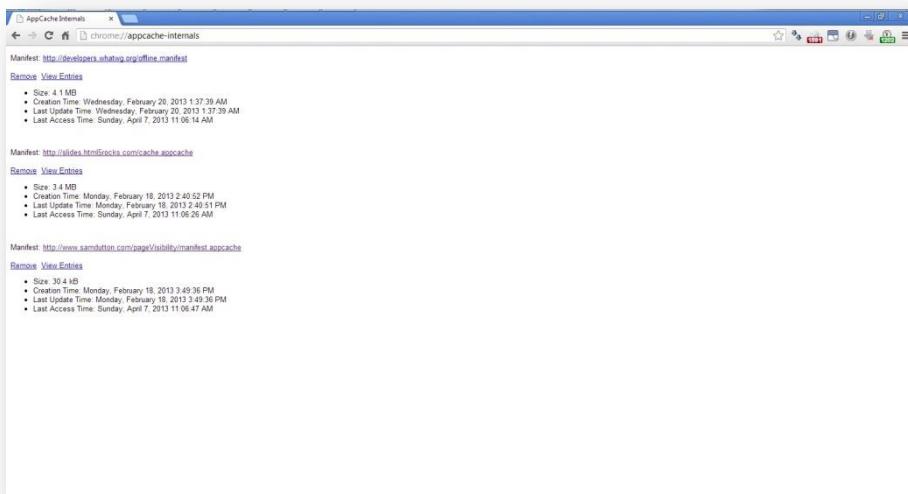
- čitanje i pisanje e-mailova,
- uređivanje dokumenta,
- uređivanje i prikaz prezentacija te
- razne igrice ili manje aplikacije koje ne trebaju stalnu vezu.

Koristeći offline pohranu moguće je izbjegći normalne pozive preko mreže prilikom učitavanja aplikacija. Ako je međuspremnik aktualan i nema novih podataka, preglednik zna kako ne treba ponovno učitavati resurse te je na taj način ponovno učitavanje stranice puno brže jer se sve učitava iz lokalne pohrane. Uz to učitavanjem resursa iz lokalne pohrane smanjuje se promet kroz mrežu što je vrlo bitno kod pristupanja stranici preko mobilnih uređaja. Upravo sporije učitavanje čini veliku razliku između web i desktop aplikacija, no lokalna pohranu tu razliku smanjuje. Datoteka **cache manifest** omogućuje resursima grupiranje u logičke cjeline. Ovakav vrlo moćan koncept pruža web aplikacijama karakteristike desktop aplikacija.

Koristeći preglednik Firefox moguće je pregledati njegov spremnik offline podataka. Pristup tim informacijama je moguć ako se u adresnu traku preglednika upiše **about:cache**.



Ukoliko se koristi preglednik Chrome pristup sličnim podacima moguć je preko adrese ***chrome://appcache-internals/***.



Za korištenje lokalne pohrane potrebna su dva koraka:

- izrada datoteke sa popisom datoteka koje se žele spremiti – (cache manifest),

CACHE MANIFEST

```
/theme.css
/logo.gif
/main.js
```

- pozivanje datoteke gdje se nalazi popis spremljenih datoteka.

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

Svaka stranica koja ima postavljen atribut **manifest** će biti spremljena kada ju korisnik posjeti. Ukoliko atribut **manifest** nije postavljen stranica se neće spremati u međuspremnik (osim ako se posebno navede u datoteci gdje se nalazi popis svih datoteka koje se spremaju). Preporučeni nastavak za **manifest** datoteku je „.appcache“.

7.2.4.1 Manifest datoteka

Manifest datoteka je tekstualna datoteka koja u sebi ima popis svih datoteka koje se trebaju spremati na lokalni disk, popis svih datoteka koje se ne spremaju te popis datoteka koje će se koristiti ukoliko pristup određenoj datoteci nije moguć. Manifest datoteka se sastoji od:

- CACHE MANIFEST – ispod je popis datoteke koje će se lokalno pohraniti nakon što su skinute sa stranice prvi puta,
- NETWORK – ispod je popis datoteka koje se nikada neće lokalno pohraniti te datoteke zahtijevaju stalnu vezu sa serverom i
- FALLBACK – ispod je popis datoteka koje će se otvoriti ukoliko određene stranice nije moguće otvoriti.

7.2.4.1.1 CACHE MANIFEST

U prvom redu datoteke mora obavezno biti ovaj parametar unutar kojega se nalaze sve datoteke koje se žele lokalno pohraniti.

```
CACHE MANIFEST
/theme.css
/logo.gif
/main.js
```

U popisu datoteka se nalazi jedna CSS datoteka, jedna gif datoteka te jedna JavaScript datoteka. Prilikom učitavanja manifest datoteke te će tri datoteke biti pohranjene na disku te će se prilikom sljedećeg poziva stranice učitati sa diska.

7.2.4.1.2 NETWORK

U dijelu nakon ključne riječi **network** nalazi se popis datoteka koje se nikada neće pohranjivati na disk. U ovom primjeru stranica **login.asp** se nikada neće pohranjivati na disk te neće biti dostupna ukoliko ne postoji veza sa serverom.

```
NETWORK:  
login.asp
```

Znak * (zvjezdica) se može koristiti ukoliko se žele postaviti svi resursi/datoteke na određenu razinu (pohranjivati ili ne pohranjivati). U ovom primjeru niti jedan resurs niti datoteka se neće pohranjivati na disk.

```
NETWORK:  
*
```

7.2.4.1.3 Fallback

U dijelu nakon ključne riječi **fallback** nalazi se popis datoteka koje će se otvoriti ukoliko pristup određenim datotekama ili direktorijima nije moguć.

```
FALLBACK:  
/html/ /offline.html
```

U ovom primjeru ukoliko nije moguće pristupiti direktoriju **html** otvoriti će se stranica **offline.html**. Prvi dio je resurs a drugi dio nakon razmaka je stranica koja će se otvoriti.

Nakon što su se stranice pohranile na disk one ostaju pohranjene dok se ne dogodi jedan od slučaja:

- korisnik izbriše međuspremnik preglednika,
- datoteka sa popisom se ne promijeni te
- međuspremnik se promijeni programski.

Redovi koji počinju sa znakom # se smatraju komentarom no mogu se koristiti i za druge primjene. Međuspremnik se ponovno ažurira tek nakon što se promijeni manifest datoteka. Ukoliko se promijeni bilo koja datoteka koja se nalazi unutar manifest datoteke to neće uzrokovati automatsko obnavljanje međuspremnika. Promjena datuma i verzije unutar zakomentiranog reda je način na koji će preglednik osvježiti međuspremnik.

```
CACHE MANIFEST
# 2013-04-04 v1.0.0
/theme.css
/logo.gif
/main.js
```

NETWORK:
login.asp

FALLBACK:
/html/ /offline.html

```
<!DOCTYPE html>
<html manifest="demo_html.appcache">
<body>
<script src="demo_time.js">
</script>
<p id="timeParametar"><button onclick="getTime()">Get Date and Time</button></p>
<p></p>
<p>Try opening <a href="tryhtml5_html_manifest.htm" target="_blank">this page</a>, then go offline, and
reload the page. The script and the image should still work.</p>
</body>
</html>
```

```
/*time.js*/
function getTime()
{
    var d=new Date();
    document.getElementById('timeParametar').innerHTML=d;
}
```

```
#demo_html.appcache
CACHE MANIFEST
/index.html
/demo_time.js
/logo.jpeg
```



Navedeni primjer pokazuje primjenu lokalne pohrane koja daje točan datum i vrijeme koje se dobiva uz pomoć korištenja JavaScripte. Ukoliko i ne postoji veza sa serverom podatak od datumu i vremenu će se prikazati.

7.2.4.2 Resursi u aplikacijskom međuspremniku

Aplikacijski međuspremnik sadrži barem jedan resurs. Svi resursi su podijeljeni u jednu od kategorija:

- Master entries – tu se nalaze svi resursi koje preglednik sam automatski sprema sa stranica koje su posjećivane,
- Explicit entries – tu se nalaze svi resursi koji su eksplicitno naglašeni za pohranu,
- Network entries – tu se nalaze svi resursi koji su unutar manifest datoteke navedene pod opcijom **network**,
- Fallback entries – tu se nalaze svi resursi koji su unutar manifest datoteke navedene pod opcijom **fallback**.

The screenshot shows the 'AppCache Internals' panel in Google Chrome DevTools. It displays the manifest URL: http://www.w3schools.com/html/demo_html.appcache. Below the manifest URL, there are two buttons: 'Remove' and 'View Entries'. Under the 'View Entries' section, there is a list of resources with their flags, URLs, and sizes:

Flags	URL	Size (headers and data)
Manifest	http://www.w3schools.com/html/demo_html.appcache	396 B
Explicit	http://www.w3schools.com/html/demo_time.js	554 B
Explicit	http://www.w3schools.com/html/img_logo.gif	3.4 kB
Master	http://www.w3schools.com/html/tryhtml5_html_manifest.htm	773 B

7.2.4.3 Stanja aplikacijskog međuspremnika

Svaki aplikacijski spremnik ima svoje stanje koje prikazuje njegovo trenutno stanje.

- UNCACHED – posebna vrijednost aplikacijskog međuspremnika koja pokazuje kako spremnik nije u potpunosti inicijaliziran,
- IDLE – aplikacijski međuspremnik nije u procesu ažuriranja,
- CHECKING - manifest je u procesu dohvaćanja i spremen za ažuriranje,
- DOWNLOADING – resursi su u procesu preuzimanja te se dodaju u međuspremnik, uslijed promjene manifesta,
- UPDATEREADY – nova verzija aplikacijskog međuspremnika je dostupna. Događaj **updateready** je vezan uz ovo stanje,
- OBSOLUTE – skupina aplikacijskog međuspremnika je zastarjela, nije aktualna.

7.2.5 Razlika lokalne pohrane u odnosu na međuspremnik preglednika

Svi poznatiji preglednici imaju neku vrstu međuspremnika. No potrebno je razlikovati međuspremnik preglednika i lokalnu pohranu. Preglednici u međuspremnik spremaju stranice i resurse sa stranica koje se često posjećuju te na taj način ubrzavaju ponovno učitavanje stranice. Lokalna pohrana omogućava programerima pohranjivanje resursa i stranica koje prije nisu posjećene. Isto tako međuspremnik je obično nečitljiv i nepouzdan te nema sigurnosti da li će se stranica ili resursi moći pregledavati i kada ne postoji veza sa serverom.

7.2.6 Web storage

Sigurnija i bolja pohrana nego kod pohrane koristeći cookies (kolačićima). Koristeći HTML5 moguće je pohranjivati korisničke podatke unutar preglednika. Prije su se za pohranu koristili cookie-si (kolačići). Web storage je puno sigurniji i bolji način pohrane jer pohranjuje podatke samo kada se to traži za razliku od cookie-sa koji kontinuirano pohranjuju podatke svakim pozivom prema serveru. Također je moguće spremiti velike količine podataka bez utjecaja na performanse cijele web stranice. Podaci se spremaju u paru ključ/vrijednost i pristup tim podacima moguć je samo iz te web stranice što još više pridonosi sigurnosti.

Postoji tri načina spremanja podataka a to su:

- localStorage,
- sessionStorage te
- web SQL.

Prije početka korištenja web storage-a potrebno je provjeriti da li preglednik podržava navedenu opciju.

```

if(typeof(Storage)!=="undefined"){
    //Yes!localStorage and sessionStorage support!
    // Some code.....
}
else{
    // Sorry! No web storage support..
}

```

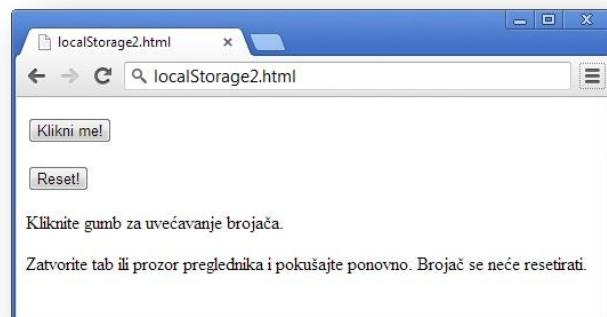
7.2.6.1 LocalStorage

LocalStorage spremi podatke lokalno bez vremena isteka. Podaci se ne brišu zatvaranjem kartice preglednika ili prozora preglednika. Podaci mogu biti vidljivi dan, tjedna ili mjesec dana dok god se ne obrišu ručno.

```

function clickCounter(){
if(typeof(Storage)!=="undefined") {
    if (localStorage.clickcount) {
        localStorage.clickcount=Number(localStorage.clickcount)+1;
    }
    else {
        localStorage.clickcount=1;
    }
    document.getElementById("result").innerHTML="Kliknuli ste gumb " + localStorage.clickcount + " put(a).";
    }
else {
    document.getElementById("result").innerHTML="Sorry, your browser does not support web storage...";
    }
}
function cleanLocalStorage(){
    localStorage.clear();
}

```





7.2.6.2 SessionStorage

SessionStorage sprema lokalno podatke koji se nalaze unutar jedne sesije. Zatvaranjem kartice unutar preglednika ili prozora preglednika brišu se podaci koji su spremljeni.

```
function clickCounter(){
if(typeof(Storage)!=="undefined") {
    if (sessionStorage.clickcount) {
        sessionStorage.clickcount=Number(sessionStorage.clickcount)+1;
    }
    else {
        sessionStorage.clickcount=1;
    }
    document.getElementById("result").innerHTML="Kliknuli ste gumb " + sessionStorage.clickcount + " put(a) unutar ove sesije.";
}
else {
    document.getElementById("result").innerHTML="Sorry, your browser does not support web storage...";
}
}
```

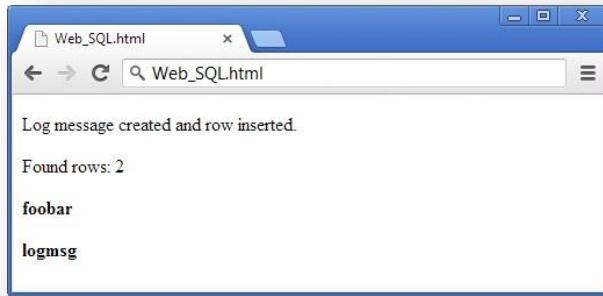




7.2.6.3 Web SQL

Web SQL je način pohrane podataka koji se najviše razlikuje u odnosu na **localStorage** i **sessionStorage** jer nije u potpunosti zaživio u HTML5 standardu iako ima niz prednosti u odnosu na druge načine pohrane. Najveća prednost je pohrana podataka u obliku baze podataka koristeći SQL sintaksu. Na taj način je moguće vrlo lako i jednostavno spremiti podatke ili pročitati podatke koji su od prije spremjeni.

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
var msg;
db.transaction(function (tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
    tx.executeSql('INSERT INTO LOGS (id, log) VALUES (1, "foobar")');
    tx.executeSql('INSERT INTO LOGS (id, log) VALUES (2, "logmsg")');
    msg = '<p>Log message created and row inserted.</p>';
    document.querySelector('#status').innerHTML = msg;
});
db.transaction(function (tx) {
    tx.executeSql('SELECT * FROM LOGS', [], function (tx, results) {
        var len = results.rows.length, i;
        msg = "<p>Found rows: " + len + "</p>";
        document.querySelector('#status').innerHTML += msg;
        for (i = 0; i < len; i++) {
            msg = "<p><b>" + results.rows.item(i).log + "</b></p>";
            document.querySelector('#status').innerHTML += msg;
        }
    }, null);
});
```

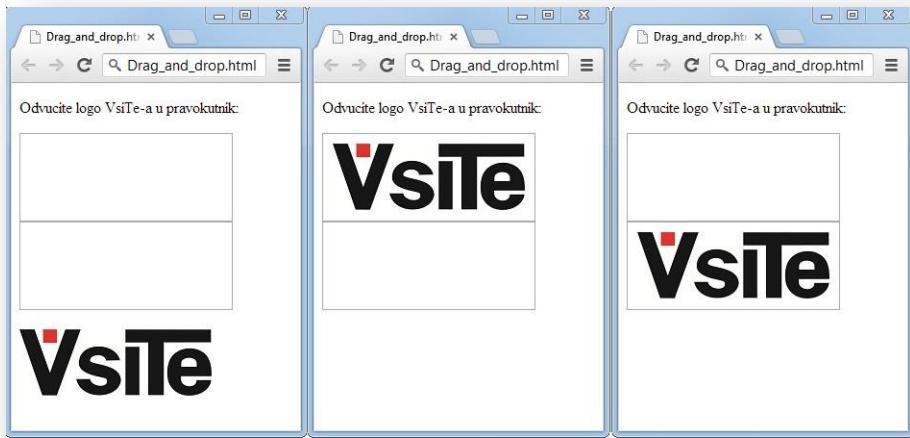


7.2.7 Drag&drop

Koristeći drag&drop (odvuci i spusti) moguće je napraviti potpuno interaktivno sučelje koje omogućuje korisniku pomicanje elemenata po stranici. Kako bi se moglo koristiti pomicanje elemenata, elementu je potrebno postaviti atribut **draggable** na vrijednost **true**.

```
<!DOCTYPE HTML>
<html>
    <head>
        <style type="text/css">
            #div1 {width:200px;height:70px;padding:10px;border:1px solid #aaaaaa;}
            #div2 {width:200px;height:70px;padding:10px;border:1px solid #aaaaaa;}
        </style>
        <script>
            function allowDrop(ev){
                ev.preventDefault();
            }
            function drag(ev){
                ev.dataTransfer.setData("Text",ev.target.id);
            }
            function drop(ev){
                ev.preventDefault();
                var data=ev.dataTransfer.getData("Text");
                ev.target.appendChild(document.getElementById(data));
            }
        </script>
    </head>
    <body>
        <p>Odvucite logo VsiTe-a u pravokutnik:</p>
        <div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
        <div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
        <br>
    </body>
</html>
```

Također potrebno je odrediti i element na koje mjesto se želi odvući željeni element. Za taj postupak potrebno je koristiti atribute **ondrop** i **ondrageevent** unutar kojih je potrebno postaviti željene funkcije uz pomoću kojih će se cijeli postupak pomicanja i odvijati.



7.2.8 Web Workers

Česti je slučaj korištenja JavaScripti koji se izvršavaju na stranicama. To može uzrokovati usporavanje stranice ili čak potpunu blokadu stranice. Koristeći ovu opciju moguće je kreirati poseban mehanizam koji će omogućiti nezavisno izvršavanje JavaScripte u odnosu na web stranicu i na taj način totalno odvojiti zahtjevne dijelove koda i sučelje. Koristeći **web Workers-e** (web radnike) izrada web stranice se još više približava izradi desktop aplikacija gdje postoje slični problemi te se to rješava korištenjem niti (thread-ova). Prije korištenja potrebno je ispitati da li preglednik podržava korištenje web workers-a.

```
if(typeof(Worker)!=="undefined") {  
    // Yes! Web worker support!  
    // Some code.....  
}  
else {  
    // Sorry! No Web Worker support..  
}
```

Potrebno je kreirati datoteku unutar koje će se nalaziti JavaScript kod koji se treba izvršavati. Datoteka mora imati nastavak .js koji predstavlja JavaScript datoteku. Vrlo je bitno koristiti metodu **postMessage()** koja služi za vraćanje podataka HTML stranici. Obično JavaScript datoteka nije ovako jednostavna već se obično nalazi puno zahtjevniji i složeniji kod.

```
var i=0;
function timedCount(){
    i=i+1;
    postMessage(i);
    setTimeout("timedCount()",500);
}
```

Nakon što se kreira datoteka unutar koje se nalazi kod koji se treba izvršavati potrebno je kreirati objekt unutar HTML stranice s kojime se povezuje HTML stranica i JavaScript datoteka.

```
if(typeof(w)=="undefined")  {
    w=new Worker("demo_workers.js");
}
w.onmessage = function (event) {
    document.getElementById("result").innerHTML=event.data;
};
```

Nakon toga moguće je primati i slati poruke koje dolaze iz JavaScripte koja se izvršava u pozadini. Dodavanjem **onmessage** funkcije omogućava se izvršavanje želenog koda svaki puta kada se prime podaci. Nakon što je kreiran objekt kod u pozadini se izvršava kontinuirano iako je završen. Jedini način na koji se može u potpunosti zastaviti je ručno gašenje tj. uništavanje objekta koristeći **terminate()** funkciju. Na ovaj način se oslobođaju i svi resursi preglednika odnosno računala koji su bili korišteni za izvođenje koda. Nakon zaustavljanja više nije moguće ponovno pokrenuti izvršavanje koda. Jedini način na koji se može ponovno pokrenuti izvršavanje je ponovnim učitavanjem web stranice.



```

<!DOCTYPE html>
<html>
<body>
<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>
<script>
var w;
function startWorker(){
if(typeof(Worker)!=="undefined"){
if(typeof(w)=="undefined") {
w=new Worker("demo_workers.js");
}
w.onmessage = function (event) {
document.getElementById("result").innerHTML=event.data;
};
}
else{
document.getElementById("result").innerHTML="Sorry, your browser does not support Web Workers...";}
}
function stopWorker(){
w.terminate();
}
</script>
</body>
</html>

```

7.2.9 Server Sent Events – SSE

U današnje vrijeme sve više stranica primaju kontinuirane podatke sa servera. To je najviše vidljivo na socijalnim mrežama gdje se podaci u pregledniku prikazuju bez potrebe za ponovnim učitavanjem stranice ili bilo kakvom akcijom korisnika. To je moguće napraviti koristeći SSE – Server Sent Events (server šalje događaje) metodom. Za korištenje potrebno je kreirati kod koristeći pomoću serverskih tehnologija (PHP,ASP) koje se izvršavaju na serveru te koda koji se izvršava na klijentu koristeći HTML5 i JavaScript elemente. Prije korištenja potrebno je provjeriti da li preglednik podržava korištenje ove opcije.

```

if(typeof(EventSource)!=="undefined") {
// Yes! Server-sent events support!
}
else {
// Sorry! No server-sent events support..
}

```

Primjer PHP koda:

```
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');
$time = date('r');
echo "data: The server time is: {$time}\n\n";
flush();
?>
```

```
<!DOCTYPE html>
<html>
<body>
<h1>Getting server updates</h1>
<div id="result"></div>
<script>
if(typeof(EventSource)!=="undefined") {
    var source=new EventSource("demo_sse.php");
    source.onmessage=function(event) {
        document.getElementById("result").innerHTML+=event.data + "<br>";
    };
}
else {
    document.getElementById("result").innerHTML="Sorry, your browser does not support server-sent
events... ";
}
</script>
</body>
</html>
```

Getting server updates

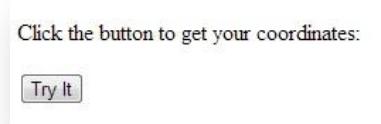
The server time is: Sun, 14 Apr 2013 08:49:15 -0400
The server time is: Sun, 14 Apr 2013 08:49:19 -0400

Navedeni primjer pokazuje dohvaćanje datuma i vremena sa servera. Klijent (HTML stranica) periodički dohvaća nove vrijednosti koje server šalje te ih prikazuje. Prikazana je automatsko dohvaćanje podataka sa servera bez korištenja ponovnog učitavanja stranice ili bilo koje druge akcije korisnika.

7.2.10 Geolokacija

Korištenje navigacije i GPS-a danas je svakodnevica, većina današnjih mobilnih uređaja ima ugrađene GPS senzore uz pomoću kojih je moguće točno odrediti lokaciju korisnika. GPS senzori nisu baš česta pojava računalima. Zahvaljujući opciji geolociranja korisnici koji nemaju GPS senzor također mogu koristiti mogućnosti navigacije i lociranja. Unutar HTML-a geolokacija se može vrlo jednostavno koristiti za razne primjene od jednostavnog pozicioniranja te traženja raznih usluga u okolini pa čak do potpuno funkcionalne navigacije ukoliko se korisni i GPS senzor uređaja. Prije prikazivanja lokacije, korisnik mora potvrditi da želi podijeliti lokaciju sa serverom, ukoliko korisnik odbije podijeliti svoju lokaciju neće se moći odrediti njegova pozicija. Isto kao i za prethodne opcije potrebno je provjeriti da li preglednik podržava korištenje geolokacije.

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Click the button to get your coordinates:</p>
<button onclick="getLocation()">Try It</button>
<script>
var x=document.getElementById("demo");
function getLocation() {
if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition(showPosition);
}
else{x.innerHTML="Geolocation is not supported by this browser.";}
}
function showPosition(position) {
x.innerHTML="Latitude: " + position.coords.latitude +
"<br>Longitude: " + position.coords.longitude;
}
</script>
</body>
</html>
```

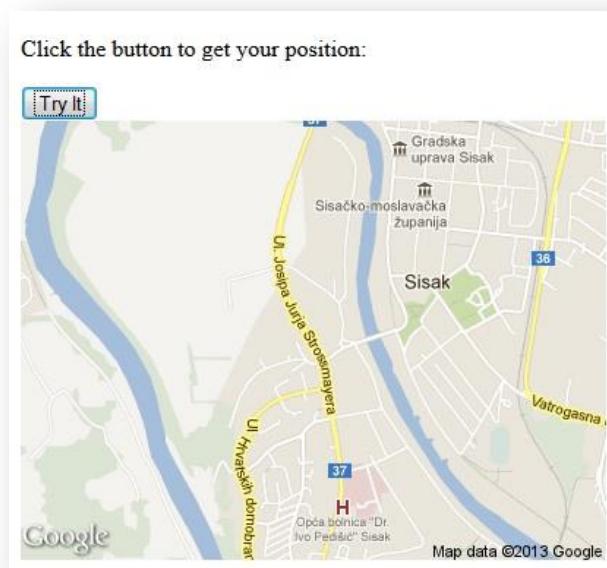


Klikom na tipku prikazuje se lokacija korisnika u obliku zemljopisne širine i dužine. Ti podaci govore o lokaciji korisnika no uz pomoću tih podataka korisnik i dalje neće znati gdje se nalazi jer nema kartu koja pokazuje njegovu lokaciju. Sljedeći primjer prikazuje i kartu gdje se korisnik nalaze.

```

var x=document.getElementById("demo");
function getLocation() {
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(showPosition,error);
}
else{x.innerHTML="Geolocation is not supported by this browser.";}
}
function showPosition(position) {
var latlon=position.coords.latitude+","+position.coords.longitude;
var img_url="http://maps.googleapis.com/maps/api/staticmap?center="
+latlon+"&zoom=14&size=400x300&sensor=false";
document.getElementById("mapholder").innerHTML=<img src='"+img_url+"'>";
}

```



Lokacija korisnika je na srediti karte. Na ovaj način korisnik jednim pogledom na kartu može znati gdje se nalazi i što se nalazi u njegovoj blizini. Prilikom dohvaćanja podataka postoji mogućnost pojave greške. Greške koje se mogu pojaviti su:

- PERMISSION_DENIED – ukoliko korisnik ne želi podijeliti lokaciju,
- POSITION_UNAVAILABLE – nije moguće odrediti položaj,
- TIMEOUT – vrijeme unutar kojega se mora odrediti položaj je isteklo te
- UNKNOWN_ERROR - nepoznata greška.

7.2.10.1.1 Povratne vrijednosti funkcije getCurrentPosition()

Funkcija **getCurrentPosition()** može vratiti nekoliko atributa. Funkcija uvijek vraća zemljopisnu širinu i dužinu te točnost, no moguće je vratiti i niz drugih.

Svojstvo	Opis
coords.latitude	The latitude as a decimal number
coords.longitude	The longitude as a decimal number
coords.accuracy	The accuracy of position
coords.altitude	The altitude in meters above the mean sea level
coords.altitudeAccuracy	The altitude accuracy of position
coords.heading	The heading as degrees clockwise from North
coords.speed	The speed in meters per second
timestamp	The date/time of the response

Uz sve navedeno moguće je koristiti i funkcije **watchPosition()** i **clearWatch()** koje se mogu koristiti kao navigacija. Funkcija **watchPosition()** prati pozicije korisnika te ih prikazuje na karti te se uz pomoću toga može napraviti prikaz putovanja ili mesta gdje je korisnik bio. Funkcijom **clearWatch()** zaustavlja se praćenje pozicije tj. zaustavlja se navigacija.

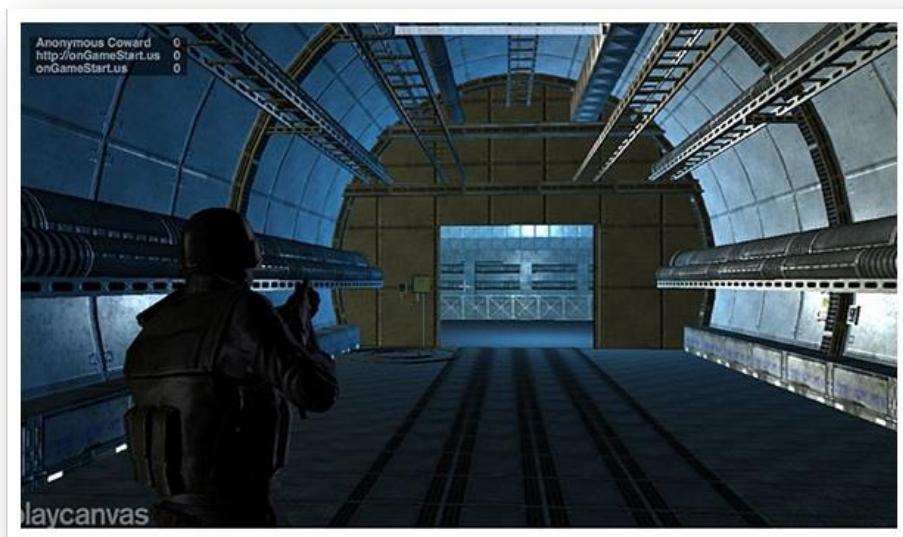
```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Click the button to get your coordinates:</p>
<button onclick="getLocation()">Try It</button>
<script>
var x=document.getElementById("demo");
function getLocation() {
if (navigator.geolocation) {
  navigator.geolocation.watchPosition(showPosition);
}
else{x.innerHTML="Geolocation is not supported by this browser.";}
}
function showPosition(position) {
x.innerHTML="Latitude: " + position.coords.latitude +
"<br>Longitude: " + position.coords.longitude;
}
</script>
</body>
</html>
```

7.2.11 Izrada igrica pomoću HTML5

Igranje igri na internetu nije novost, ta opcija postoji još od pojave HTML verzije 4.

Uvođenjem HTML 5 moguće je kreirati i grafički vrlo zahtjevne igre koje koriste sve dostupne resurse klijentskog računala a pojaviše grafičke kartice. Uz korištenje HTML5 postoji mogućnost korištenja WebGL-a koji daje absolutnu slobodu 2D i 3D grafike. Pojavom ovih tehnologija izrada vrlo zahtjevnih igara koji daju osjećaj da se uopće ne radi o igrama koje se nalaze na internetu već da je riječ o igrama koje su instalirane na računalu korisnika.

Problemi svih igara koji se igraju preko interneta su i dalje prisutni i ovdje. Brzina pristupa internetu, vrijeme kašnjenja neki su od problema koji se mogu pojaviti, no korištenjem lokalne pohrane i drugih opcija moguće je navedene probleme svesti na minimum.



8 Javascript

JavaScript je skriptni jezik koji se koristi na internetu. Treba ga razlikovati od programskog jezika **Java** koji je programski jezik i platforma dok je **JavaScript** skriptni jezik koji se izvršava na korisničkom računalu. **JavaScript** se koristi na velikom broju stranica za razne dodatne funkcionalnosti web stranica kao što su provjera unesenih podataka, komunikaciju sa serverom, razne vizualne efekte i drugo.

```
<!DOCTYPE html>
<html>
    <head>
        <title>JavaScript</title>
    </head>
    <body>
        <h1>JavaScript</h1>
        <p id="demo">
            JavaScript može mijenjati sadržaj HTML elementa.
        </p>
        <script>
            function myFunction(){
                x=document.getElementById("demo");
                x.innerHTML="Ja sam JavaScript!";
            }
        </script>
        <button type="button" onclick="myFunction()">Klikni me!</button>
    </body>
</html>
```



Primjer prikazuje vrlo jednostavan način korištenja **JavaScripte**, prilikom klika na tipku poziva se funkcija **myFunction()** unutar koje se mijenja sadržaj HTML elementa koji ima id „**demo**“.

JavaScript kod mora biti napisan između **<script>** i **</script>** HTML elemenata.

JavaScript je vrlo sličan programskom jeziku C te je većina naredbi se piše isto te imaju isto značenje.

8.1 Funkcije i događaji JavaScripte

JavaScript-a se može izvršavati prilikom učitavanja stranice no puno je češća primjena izvršavanja nakon nekog događaja.

Unutar HTML stranice moguće je staviti neograničen broj skripti. JavaScript kod moguće je staviti ili unutar **<head>** ili **<body>** HTML elementa. Koja god pozicija bila skripta će se izvršavati na isti način. Uobičajen je način umetanja skripte unutar **<head>** HTML elementa kako bi se povećala preglednost koda.

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction(){
document.getElementById("demo").innerHTML="My First JavaScript Function";
}
</script>
</head>
<body>
<h1>My Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```

Također moguće je uključiti i gotovu JavaScript datoteku unutar HTML-a koristeći **src** atribut.

```
<!DOCTYPE html>
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

8.2 Manipulacija HTML elementima

Pristup svakom HTML elementu moguć je koristeći metodu **document.getElementById(id)**.

Id je id elementa kojem se želi pristupiti. U sljedećem primjeru je prikazan pristup **p** HTML elementu koji ima id „demo“.

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p id="demo">My First Paragraph</p>
<script>
document.getElementById("demo").innerHTML="My First JavaScript";
</script>
</body>
</html>
```

Kako se JavaScript kod izvršava unutar preglednika moguće je pristupiti svakom HTML elementu koristeći **innerHTML** svojstvo.

Ukoliko se želi ispisati tekst na stranicu to je moguće koristeći metodu **document.write()**.

Pomoću te metode moguće je ispisati tekst direktno na stranicu.

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<script>
document.write("<p>My First JavaScript</p>");
</script>
</body>
</html>
```

Ukoliko se za ispis na stranicu koristi metoda **document.write()** potrebno je znati kako se koristeći tom metodom slučajno može obrisati cijeli sadržaj stranice ukoliko se pozove nakon što se učita cijela stranica.

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My First Paragraph.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction(){
document.write("Oops! The document disappeared!");
}
</script>
</body>
</html>
```

8.3 Izrazi i varijable

JavaScript kod se izvršava kao slijed naredbi koji se izvršavaju u pregledniku. Kao što je već ranije spomenuto **JavaScript** je jezik koji je vrlo sličan programskom jeziku C te ima jako puno sličnosti kao što su:

- svaka naredba završava sa „;“,
- blok koda počinje sa „{“ i završava sa „}“ ,
- JavaScript je case sensitive točnije razlikuje mala i velika slova (if i IF su dvije različite naredbe),
- jednolinijski komentar „//“ te
- više linijski počinje sa „/*“ a završava sa „*/“.

8.3.1 Varijable

Imena varijabli moraju početi sa slovom ili znakom „\$“ ili „_“ no najčešće se koriste slova. Ukoliko se žele u nazivu varijable koristiti brojevi mogu biti tek od drugog znaka (npr. i2).

8.3.2 Tipovi podataka

Kao i u bilo kojem programskom jeziku tako i u **JavaScriptu** postoje različiti tipovi varijabli neki od tipova varijabli su string, number, boolean, object itd. Deklaracija varijable započinje ključnom riječi **var**, nakon toga ide ime varijable i vrijednost varijable ukoliko se želi postaviti. Ako se ne postavi vrijednost varijable ona je postavljena na vrijednost **undefined**.

```
var ime = "Pero Perić";
var broj = 2;
var prezime="Perić", mjesto="Zagreb";
```

```
var x;           // x je undefined
var x = 5;        // x je Number
var x = "Pero";    // x je String
var osoba={ime:„Pero”, prezime:„Perić”, id:5566};
name=osoba.prezime;
name=osoba[“prezime”];
```

Ukoliko se želi poništiti vrijednost varijable mora se postaviti na vrijednost **null**. Ukoliko se želi eksplisitno postaviti varijabla na određeni tip potrebno je napraviti sljedeće:

```
var ime = new String;
var x = new Number;
var y = new Boolean;
var auti = new Array;
var osoba = new Object;
```

8.4 JavaScript objekti

Svi tipovi varijabli su zapravo objekti. Kada se kreira varijabla koristeći ključnu riječ **new** kreira se objekt tog tipa.



Svaki objekt ima svoje metode i svojstva preko kojih komunicira sa drugim objektima. Svojstva objekta su podaci koji svaki objekt jedinstveno identificiraju.

8.4.1 Kreiranje objekta

Prije korištenja svaki objekt prvo je potrebno kreirati. Objekt se kreira koristeći ključnu riječ **new**. U primjeru je prikazano kreiranje objekta osobe i pristup njegovim svojstvima.

```
osoba=new Object();
osoba.ime="Pero";
osoba.prezime="Perić";
osoba.godine=25;
osoba.boja_ociju="plava";
```

8.4.2 Pristupanje metodama

Pristupanje metodama objekata je slično kao i svojstvima koristeći točku i nakon toga ime metode koja se želi pozvati. Svaka metoda mora imati zagrade () što ju i razlikuje od svojstva. Ukoliko metoda prima argument taj argument se upisuje prilikom poziva unutar zagrade.

```
var message="Hello world!";
var x=message.toUpperCase();
```

Povratna vrijednost ove metode će biti „HELLO WORLD!“.

8.5 JavaScript funkcije

JavaScript funkcije su vrlo slične funkcijama unutar programskog jezika C osim što se ne piše povrati tip funkcije.

```
function ime_funkcije(){
    tijelo funkcije
}
function ime_funkcije(var1,var2){
    tijelo funkcije
}
function myFunction(){
    var x=5;
    return x;
}
```

```
function myFunction(a,b){
    return a*b;
}

document.getElementById("demo").innerHTML=myFunction(4,3);
```

Rezultat funkcije će biti 12 jer će se vratiti rezultat množenja dva ulazna parametra (4 i 3).

8.6 Vidljivost varijabli

Postoje dvije vrste varijabli koje se koriste unutar JavaScripta a to su lokalne i globalne.

Lokalne varijable su vidljive samo unutar bloka naredbi, unutar petlji ili unutar funkcije.

Izlaskom iz funkcije varijabla više ne postoji. Globalne varijable su vidljive unutar cijele web stranice. To su varijable koje su deklarirane na razini cijele skripte te im je moguće pristupiti iz bilo koje funkcije koja se nalazi unutar te skripte. Zatvaranjem stranice gube se podaci koje se nalaze unutar globalnih varijabli.

8.7 Operatori

Postoji niz operatora koji se mogu koristiti sa varijablama unutar JavaScript koda. Većina njih se ne razlikuje u odnosu na bilo koji drugi programski jezik. Jedina iznimka je operator „+“ koji ima dvostruko značenje a to je zbrajanje numeričkih vrijednosti i spajanje tekstualnih tipova podataka.

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
What a very nice day
x=5+5;
y="5"+5;
z="Hello "+5;
10
55
Hello 5
```

Operator	Description	Example	Result of x	Result of y
+	Addition	x=y+2	7	5
-	Subtraction	x=y-2	3	5
*	Multiplication	x=y*2	10	5
/	Division	x=y/2	2.5	5
%	Modulus (division remainder)	x=y%2	1	5
++	Increment	x=++y	6	6
		x=y++	5	6
--	Decrement	x=--y	4	4
		x=y--	5	4

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
=	x=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

Operator	Description	Comparing	Returns
==	is equal to	x==8	false
		x==5	true
====	is exactly equal to (value and type)	x===="5"	false
		x====5	true
!=	is not equal	x!=8	true
!==	is not equal (neither value nor type)	x!=="5"	true
		x!==5	false
>	is greater than	x>8	false
<	is less than	x<8	true
>=	is greater than or equal to	x>=8	false
<=	is less than or equal to	x<=8	true

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
	or	(x==5 y==5) is false
!	not	!(x==y) is true

8.8 Uvjetni izrazi

Uvjetni izrazi su isti kao i kod programskog jezika C te se ne razlikuju u niti jednoj naredbi.

Uvjete je moguće ispitati koristeći neki od ovih načina:

- if (uvjet) izraz_true
- if (uvjet) izraz_true else izraz_false
- if (uvjet) izraz_true else if (uvjet) izraz_true else izraz_false
- switch (uvjet) case a, case b...

Ako se prilikom korištenja **switch** ispitivanja ne zadovolji niti jedan uvjet, ukoliko postoji naredba **default** taj blok naredbi će se izvršiti.

8.9 Petlje

Slično kao i uvjetni izrazi petlje koje se koriste unutar JavaScripta previše se ne razlikuju u odnosu na druge programske jezike.

Postoji nekoliko petlji koje se mogu koristiti a to su:

- for petlja
 - for/in petlja

```
var person={fname:"John",lname:"Doe",age:25};

for (x in person) {
  txt=txt + person[x];
}
```

- while petlja
- do/while petlja

Iz svake od navedenih petlji moguće je izaći prije nešto što uvjet bude/ne bude zadovoljen koristeći ključnu riječ **break** uz pomoć koje se izlazi iz petlje odmah. Koristeći ključnu riječ **continue** omogućeno je preskakanje izvođenja petlje, točnije moguće je pokrenuti sljedeću iteraciju petlje i prije nego se u potpunosti izvrši sav kod koji se nalazi u bloku petlje.

8.10 JavaScript greške

Vrlo je čest slučaj pojave neke greške tokom izvođenja bilo kojeg koda pa tako i JavaScript koda. Kako bi se smanjila mogućnost pojave greške moguće je koristiti blokove za hvatanje grešaka. Ovu mogućnost je JavaScript naslijedila iz viših programskih jezika kao što su C++, C# i Java. Unutar **try** bloka koda ide dio unutar kojeg bi se moglo dogoditi greške, nakon toga unutar **catch** bloka ide dio u kojem će se „uloviti“ greške te ih ili pokušati ispraviti ili javiti korisniku koja je greška nastala.

```
try {  
    //Run some code here  
}  
catch(err) {  
    //Handle errors here  
}
```

Također je moguće „baciti“ grešku namjerno, točnije kada se želi namjerno pozvati greška to je moguće koristeći ključnu riječ **throw**.

```
<script>  
function myFunction(){  
    try {  
        var x=document.getElementById("demo").value;  
        if(x=="") throw "empty";  
        if(isNaN(x)) throw "not a number";  
        if(x>10) throw "too high";  
        if(x<5) throw "too low";  
    }  
    catch(err) {  
        var y=document.getElementById("mess");  
        y.innerHTML="Error: " + err + ". ";  
    }  
}</script>
```

8.11 JavaScript validacija

Koristeći JavaScript validaciju moguće je provjeriti da li je korisnik unio valjane podatke u polja za unos te ukoliko je negdje unesena vrijednost koja se ne očekuje prikazati korisniku poruku kako bi mogao ispraviti grešku. Moguće je provjeriti sljedeće:

- da li je korisnik unio obavezan podatak,
- da li je korisnik unio valjanu e-mail adresu,
- da li je korisnik unio valjni datum te
- da li je korisnik unio slovo u polje gdje se zahtjeva broj.

Provjera unosa obaveznog podatka koristeći JavaScript može izgledati ovako:

```

function validateForm(){
    var x=document.forms["myForm"]["fname"].value;
    if (x==null || x=="") {
        alert("First name must be filled out");
        return false;
    }
}
<form name="myForm" action="demo_form.asp" onsubmit="return validateForm()" method="post">
First name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>

```

U navedenom primjeru ispituje se vrijednost tekstualnog polja gdje se očekuje unos imena te ukoliko se ne unese ništa a pritisne se tipka za potvrdu forme JavaScript će javiti grešku. Provjera e-mail adrese može se izvršiti na ovaj način:

```

function validateForm(){
    var x=document.forms["myForm"]["email"].value;
    var atpos=x.indexOf("@");
    var dotpos=x.lastIndexOf(".");
    if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length) {
        alert("Not a valid e-mail address");
        return false;
    }
}
<form name="myForm" action="demo_form.asp" onsubmit="return validateForm()" method="post">
First name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>

```

8.12 Regularni izrazi

Regularni izrazi omogućavaju pretraživanje teksta ili nalaženje samo dijela teksta. Pretraživanje je moguće napraviti na razini nekoliko riječi ili samo jednog znaka.

```

var patt=new RegExp(pattern,modifiers);
var patt=/pattern/modifiers;

```

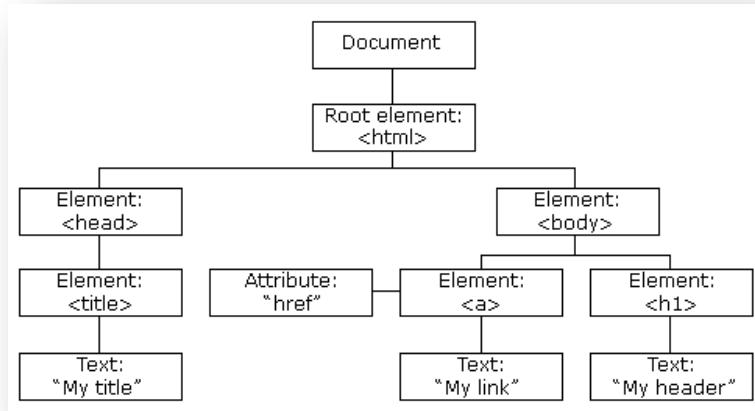
- pattern – određuje uzorak izraza
- modifiers – određuje da li će pretraga biti globalna, osjetljiva na velika i mala slova i slično.

```
var str = "Visit W3Schools";
var patt1 = /w3schools/i;
document.write(str.match(patt1));
```

Rezultat ovog primjera će biti „W3Schools“. Izrazi mogu biti i puno složeniji ukoliko postoji potreba za pretraživanjem više znakova unutar teksta.

8.13 DOM

DOM (Document Object Model) definira standarde pristupanja i manipulaciju HTML elemenata.



Koristeći DOM JavaScript kod može promijeniti HTML element, atribut elementa, promijeniti CSS stilove ili reagirati na događaje koji se događaju na stranici.

8.13.1 Dohvaćanje elementa

Svaki element je moguće dohvatiti na tri načina:

- po id-u,
- po imenu elementa te
- po klasi.

Na ove načine moguće je mijenjati ne samo HTML elemente već i CSS elemente i svojstva.

8.13.1.1 Dohvaćanje elementa po id-u

Ukoliko se unutar JavaScript koda želi dohvatiti HTML element kojem se zna njegov id atribut to je moguće napraviti vrlo jednostavan način. Ako HTML element postoji metoda **getElementById()** će vratiti objekt koji će se pohraniti unutar variable. Ukoliko element ne postoji povratna vrijednost metode će biti **null**.

```
<p id="intro">Hello World!</p>
<script>
x=document.getElementById("intro");
document.write("<p>The text from the intro paragraph: " + x.innerHTML + "</p>");
</script>
```

8.13.1.2 Dohvaćanje elementa po imenu

Ukoliko se želi dohvatiti HTML element kojem se zna ime to je moguće napraviti koristeći metodu **getElementsByName()**. U primjeru je prikazan način dohvaćanja HTML elementa preko imena. U ovom slučaju bit će dohvaćen samo **p** HTML element koji se nalazi unutar div elementa koji ima id “main”.

```
<p>Hello World!</p>
<div id="main">
<p>The DOM is very useful.</p>
</div>
<script>
var x=document.getElementById("main");
var y=x.getElementsByTagName("p");
document.write('First paragraph inside "main" is ' + y[0].innerHTML);
</script>
```

8.13.1.3 Dohvaćanje elementa po klasi

Kada postoji potreba dohvaćanja HTML elementa po klasi to je moguće na vrlo sličan način kao i dva prethodna, koristeći metodu **getElementsByClassName()**. Na ovaj način je moguće dohvatiti sve HTML elemente koji imaju određenu klasu.

8.14 Događaji

Samo dohvaćanje pojedinog elementa ponekad nije dovoljno za potpunu upotrebu JavaScript koda te je potrebno pratiti događaje koji se događaju na stranici. Neki od događaja koji se mogu pojaviti na stranici su:

- kada korisnik klikne tipku miša,
- kada se učita web stranica,
- kada se učita slika,
- kada se pokazivačem miša prođe preko elementa,
- kada se promijeni tekst unutar elementa,
- kada se potvrdi HTML forma te
- kada korisnik pritisne tipku na tipkovnici.

Jednostavan primjer događaja prikazan je u sljedećem primjeru. Kada se klikne na element promijeniti će se njegov tekst.

```
<h1 onclick="this.innerHTML='Ooops!'">Click on this text!</h1>
```

```
<button id="myBtn">Try it</button>
<script>
document.getElementById("myBtn").onclick=function(){displayDate()};
function displayDate()
{
document.getElementById("demo").innerHTML=Date();
}
</script>
<p id="demo"></p>
```

U ovom primjeru na događaj **onclick** HTML elementa koji ima id "myBtn" poziva se korisnički kreirana funkcija koja prikazuje datum unutar HTML elementa koji ima id "demo".

```
<div onmouseover="mOver(this)" onmouseout="mOut(this)" style="background-color:#D94A38;width:120px;height:20px;padding:40px;">Mouse Over Me</div>
<script>
function mOver(obj){
obj.innerHTML="Thank You"
}
function mOut(obj){
obj.innerHTML="Mouse Over Me"
}
</script>
```



8.15 Kreiranje i brisanje elemenata

Ponekad postoji potreba za kreiranjem novih elemenata ili brisanjem elemenata unutar postojeće strukture stranice. JavaScript i to omogućava koristeći metode ***createElement()***, ***createTextNode()*** i ***appendChild()***. Koristeći te metode moguće je kreirati nove elemente dinamički koristeći samo JavaScript kod te bez ikakve potrebe otvaranja HTML dokumenta unutar editora.

```
<div id="div1">
    <p id="p1">This is a paragraph.</p>
    <p id="p2">This is another paragraph.</p>
</div>
<script>
    var para=document.createElement("p");
    var node=document.createTextNode("This is new.");
    para.appendChild(node);
    var element=document.getElementById("div1");
    element.appendChild(para);
</script>
```

Na vrlo sličan način moguće je i obrisati postojeći element koristeći metodu ***removeChild()***.

```
<div id="div1">
    <p id="p1">This is a paragraph.</p>
    <p id="p2">This is another paragraph.</p>
</div>
<script>
    var parent=document.getElementById("div1");
    var child=document.getElementById("p1");
    parent.removeChild(child);
</script>
```

8.16 JavaScript BOM

JavaScript BOM (Browser Object Model) nije službeni standard no većina preglednika koristi mogućnosti BOM-a. BOM je veza između JavaScript-a i preglednika te uz pomoć njega možemo saznati podatke o pregledniku ili veličini ekrana na kojem korisnik pregledava stranicu.

8.16.2 Window objekt

Uz pomoć **window** objekta možemo upravljati korisničkim prozorom preglednika pa je tako moguće otvoriti novi prozor, zatvoriti postojeći ili promijeniti položaj prozora na ekranu.

Window objekt se razlikuje od preglednika do preglednika pa tako recimo za veličinu ekrana u Internet Exploreru od verzije 5 do 8 moguće je dobiti koristeći `document.documentElement.clientHeight` ili `document.body.clientHeight` za visinu te `document.documentElement.clientWidth` ili `document.body.clientWidth` za širinu prozora, dok se za ostale preglednike koristi `window.innerHeight` za visinu i `window.innerWidth` za širinu prozora.

8.16.3 Window screen objekt

Window screen objektom moguće je dohvatiti veličinu korisničkog ekrana. Ovaj objekt treba razlikovati sa prethodnim objektom u kojem se saznaju informacije o veličini prozora preglednika koji se može mijenjati dok se veličina ekrana fiksna za jedno računalo.

Ukoliko se želi dohvatiti maksimalna visina i širina ekrana to je moguće napraviti na sljedeći način:

```
<script>
document.write("Available Width: " + screen.availWidth);
document.write("Available Height: " + screen.availHeight);
</script>
```

8.16.4 Window Location

Koristeći **Window location** objekt moguće je saznati informacije o adresi stranice(URL) ili niz drugih informacija kao što su:

- `location.hostname` – vraća ime domene
- `location.pathname` - vraća putanju i ime datoteke trenutne stranice
- `location.port` – vraća port web host-a (80 ili 443)
- `location.protocol` – vraća web protokol koji se koristi (`http://` or `https://`)
- `location.href` - vraća URL trenutne stranice

8.16.5 Window history objekt

Sadrži povijest posjećenih stranica koja se nalazi pohranjena u pregledniku. Kako bi se zaštitili korisnici postoje ograničenja pristupa podacima pregledanih stranica. Neke od metoda koje se koriste su **history.back()** i **history.forward()**.

8.16.5.1 History.back()

History.back() metoda učitava zadnju posjećenu stranicu. Ima istu akciju kao i tipka za povratak u prozoru preglednika.

```
<script>
function goBack(){
    window.history.back()
}
</script>
<input type="button" value="Back" onclick="goBack()">
```

8.16.5.2 *History.forward()*

History.forward() metoda učitava sljedeću stranicu na popisu stranica koje se nalaze u pregledniku. Ima istu akciju kao i tipka za naprijed u prozoru preglednika.

```
<script>
function goForward(){
    window.history.forward()
}
</script>
<input type="button" value="Forward" onclick="goForward()">
```

8.16.6 Window Navigator

Window navigator objekt sadrži informacije o pregledniku, platformi računala i druge informacije. Ove informacije ponekad nisu u potpunosti točne te ih ne treba uzimati kao sigurne i provjerene.

```
<div id="example"></div>
<script>
    txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
    txt+= "<p>Browser Name: " + navigator.appName + "</p>";
    txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
    txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
    txt+= "<p>Platform: " + navigator.platform + "</p>";
    txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";
    txt+= "<p>User-agent language: " + navigator.systemLanguage + "</p>";
    document.getElementById("example").innerHTML=txt;
</script>
```

8.16.7 JavaScript poruke (Popup Boxes)

Postoje tri vrste poruka koje se mogu prikazati korisniku a to su: upozorenje (alert box), potvrda (confirm box) i upit (prompt box). Koristeći ove poruke moguće je prikazati poruku korisniku te ovisno o tipu poruke i tražiti unos korisnika neku vrijednost. Ukoliko postoji potreba za prekidanjem reda tj. ako se koristi više teksta koji se želi prelomiti kroz nekoliko redova prebacivanje je moguće koristeći „\n“.

8.16.7.1 Upozorenje - alert box

Ova vrsta poruke se koristi samo kada se želi prikazati poruka korisniku.

```
<script>
function myFunction(){
    alert("I am an alert box!");
}
</script>
...
<input type="button" onclick="myFunction()" value="Show alert box">
```

8.16.7.2 Potvrda - confirm box

Ova poruka se koristi kada se korisnika želi upitati nešto te on može kliknuti tipku „OK“ za potvrdu ili „Cancel“ za prekid.

```
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction(){
    var x;
    var r=confirm("Press a button!");
    if (r==true){
        x="You pressed OK!";
    }
    else{
        x="You pressed Cancel!";
    }
    document.getElementById("demo").innerHTML=x;
}
</script>
```

8.16.7.3 Upit - (prompt box)

Ova poruka se koristi kada se od korisnika zahtjeva unos. Ova poruka je vrlo slična kao i

prethodna (potvrda) jer ima isto tipke za „OK“ i „Cancel“.

```
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction(){
var x;
var person=prompt("Please enter your name","Harry Potter");
if (person!=null){
x="Hello " + person + "! How are you today?";
document.getElementById("demo").innerHTML=x;
}
}
</script>
```

8.16.8 JavaScript vremenski događaji (Timing events)

Unutar JavaScripta moguće izvršavati dijelove koda u određenim vremenskim intervalima.

Postoje dvije vrste vremenskih događaja a to su:

- `setInterval()` – izvršava funkciju kontinuirano u zadanim vremenskim razmacima,
- `setTimeout()` – izvršava funkciju jednom nakon određenog vremena čekanja.

8.16.8.1 Metoda `setInterval()`

Metoda `setInterval()` izvršava zadanu funkciju kontinuirano u određenim vremenskim razmacima te je pomoću nje moguće napraviti obnavljanje sadržaja dijela stranice.

Vremenski razmaci se pišu u milisekundama.

```
window.setInterval("javascript function",milliseconds);
```

```
<button onclick="myFunction()">Try it</button>
<script>
function myFunction(){
setInterval(function(){alert("Hello")},5000);
}
</script>
```

U primjeru se poziva JavaScript funkcija „MyFunction“ svakih 5 sekundi te se ispisuje tekst „Hello“. Koristeći ovaj događaj moguće je napraviti vrlo jednostavan digitalni sat koji će ažurirati vrijeme svake sekunde.

```
<p id="demo"></p>
<script>
var myVar=setInterval(function(){myTimer()},1000);
function myTimer(){
var d=new Date();
var t=d.toLocaleTimeString();
document.getElementById("demo").innerHTML=t;
}
```

8.16.8.2 Metoda *clearInterval()*

Metoda ***clearInterval()*** služi za zaustavljanje pokrenutog timer-a. Nakon što se pokrene metoda ***clearInterval()*** zaustavlja se daljnje pozivanje funkcije koja se nalazi unutar metode ***setInterval()***.

```
<p id="demo"></p>
<button onclick="myStopFunction()">Stop time</button>
<script>
var myVar=setInterval(function(){myTimer()},1000);
function myTimer(){
var d=new Date();
var t=d.toLocaleTimeString();
document.getElementById("demo").innerHTML=t;
}
function myStopFunction(){
clearInterval(myVar);
}
</script>
```

8.16.8.3 Metoda *setTimeout()*

Koristeći metodu moguće je pokrenuti JavaScript funkciju nakon što istekne zadano vrijeme. Vrijeme nakon kojega će se pokrenuti funkcija piše se u milisekundama.

```
window.setTimeout("javascript function",milliseconds);
```

```
setTimeout(function(){alert("Hello")},5000);
```

Za zaustavljanje pokretanja funkcije koristi se metoda ***clearInterval()***.

8.16.9 Cookies

Cookies (kolačići) je varijabla koja pohranjuje podatke o posjetitelju stranice te ih spremi na njegovo računalo. Svaki put kada isto računalo pristupi stranici šalju se cookie podaci.

Koristeći JavaScript moguće je pohranjivati i dohvaćati cookie informacije. Prilikom spremanja podataka na korisničko računalo često se spremi ime i datum kako bi se kasnije prilikom čitanja podataka mogao prepoznati korisnik. U sljedećem primjeru je prikazano pohranjivanje korisničkog imena, vrijednosti i datuma do kada su podaci valjni.

```
function setCookie(c_name,value,exdays){  
    var exdate=new Date();  
    exdate.setDate(exdate.getDate() + exdays);  
    var c_value=escape(value) + ((exdays==null) ? "" : "; expires="+exdate.toUTCString());  
    document.cookie=c_name + "=" + c_value;  
}
```

Kada posjetitelj sljedeći puta posjeti stranicu koristeći i dalje neće biti poznati podaci koji su se pohranili prilikom prvog posjeta. Kako bi to bilo omogućeno potrebno je pročitati podatke koji su pohranjeni na korisničkom računalu. Sljedeći primjer prikazuje način čitanja pohranjenih podataka.

```
function getCookie(c_name){  
    var c_value = document.cookie;  
    var c_start = c_value.indexOf(" " + c_name + "=");  
    if (c_start == -1) {  
        c_start = c_value.indexOf(c_name + "=");  
    }  
    if (c_start == -1) {  
        c_value = null;  
    }  
    else {  
        c_start = c_value.indexOf("=", c_start) + 1;  
        var c_end = c_value.indexOf(";", c_start);  
        if (c_end == -1) {  
            c_end = c_value.length;  
        }  
        c_value = unescape(c_value.substring(c_start,c_end));  
    }  
    return c_value;  
}
```

Nakon što su uspješno pročitani podaci koji su spremljeni na korisničkom računalu potrebno je provjeriti aktualne podatke sa podacima koji su pročitani.

```
function checkCookie(){
var username=getCookie("username");
if (username!=null && username!=""){
alert("Welcome again " + username);
}
else {
username=prompt("Please enter your name:","");
if (username!=null && username!=""){
setCookie("username",username,365);
}
}
}
```

8.17 JavaScript biblioteke

Ponekad samo pisanje JavaScript koda može biti vrlo zamorno jer se želi postići potpuna funkcionalnost na svim preglednicima te napraviti naprednije grafičke sučelje koristeći razne komplikirane funkcije. Kako bi se olakšalo korištenje JavaScripta postoje gotove biblioteke koje u sebi sadrže sve što je potrebno za korištenje i izradu stranice koristeći naprednije JavaScript funkcije. Neke od poznatih biblioteka su:

- jQuery,
- Prototype te
- MooTools.

Korištenje bilo koje biblioteke uvelike olakšava korištenje JavaScripta jer je dovoljno pozvati samo jednu funkciju koja će napraviti komplikirani JavaScript kod u pozadini bez imalo znanja što se točno radi u pozadini. Jedna od prednosti korištenja gotovih biblioteka je ta što se ne mora brinuti oko kompatibilnosti JavaScript koda sa svim preglednicima jer većina poznatijih biblioteka je kompatibilna sa gotovo svim modernim preglednicima.

9 XML

XML (eXtensible Markup Language) je jezik koji je prvenstveno predviđen za pohranu i prijenos podataka. XML je jezik koji je preporučen za korištenje od strane W3C-a što znači da je opće prihvaćen kao standard za pohranu i prijenos podataka.

9.1 XML i HTML

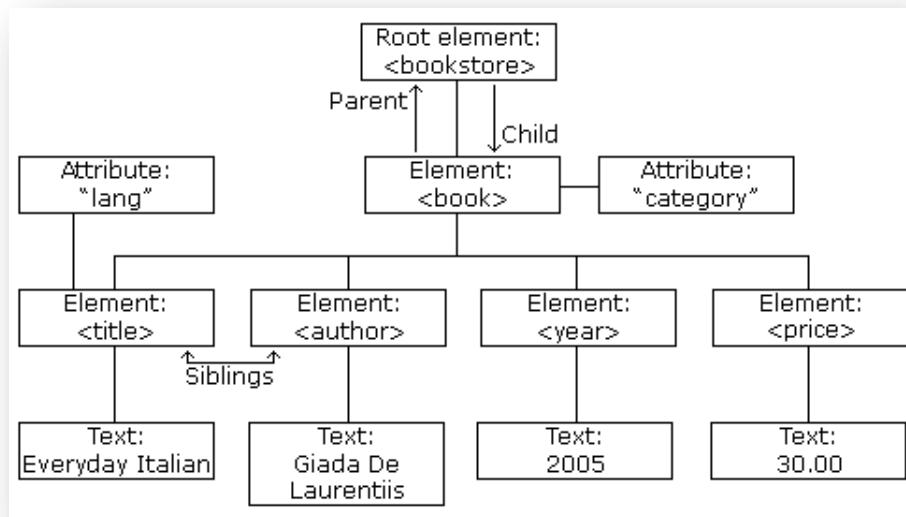
XML je jezik za označavanje te je po tomu vrlo blizak HTML-u. Sa HTML-om ga još vežu i oznake koje se koriste kao osnovni elementi strukture XML-a no za razliku od HTML-a gdje su oznake određene i svaka ima svoje značenje u XML-u korisnik sam definira svoje oznake. Svaka oznaka ima svoje značenje podatka koji se nalazi između otvorene i zatvorene oznake. XML služi samo za pohranu i prijenos podataka a ne za prikaz podataka, za to služi HTML.

Za razliku od HTML-a unutar kojeg se na temelju oznaka elemenata u pregledniku prikazuju i mijenjaju podaci, XML ne radi ništa točnije sam po sebi XML je statička datoteka koja služi kao jednostavna baza podataka.

9.2 Pravila XML-a

Kao i kod svakog standarda postoje pravila za korištenje koja bi se trebala poštivati kako bi svaka XML datoteka se mogla na isti način pročitati i koristiti. Neka od pravila su:

- unutar XML-a svaka otvorena oznaka mora se zatvoriti,
- oznake su osjetljive na mala i velika slova (case sensitive),
- moraju se poštivati pravila ugnježđivanja,
- XML dokument mora imati root element te
- ukoliko postoji atribut elementa on se mora staviti unutar navodnih znakova (<note date="12/11/2007">).



```
<?xml version="1.0"?>
<note>
  <to>John</to>
  <from>Bob</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

9.3 XMLHttpRequest

XMLHttpRequest objekt se koristi za razmjenu podataka sa serverom u pozadini.

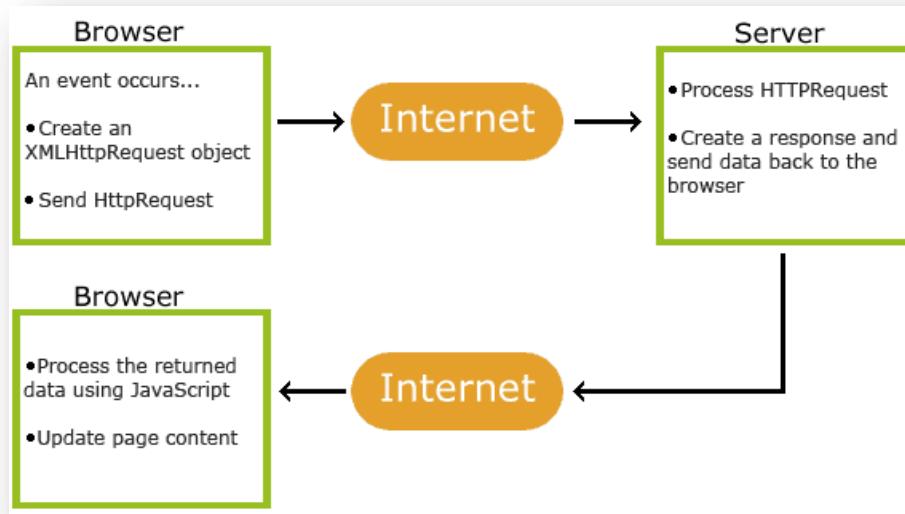
Koristeći XMLHttpRequest objekt moguće je napraviti sljedeće:

- ažurirati web stranice ili dio stranice bez potrebe za ponovnim učitavanjem stranice,
- tražiti podataka od servera nakon što se stranica već učitala te
- primati podataka od servera nakon što se stranica već učitala.

```
var xmlhttp;
if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
}
else {// code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
```

10 AJAX

AJAX (Asynchronous JavaScript and XML) je novi način korištenja postojećih tehnologija te nije novi programski jezik. AJAX je način izmjene podataka sa serverom bez potrebe za ponovnim učitavanjem stranice. Način na koji AJAX šalje podatke serveru je asinkrono, što znači da se podaci između servera i klijenta izmjenjuju bez potrebe za eksplisitnim akcijama slanja podataka. AJAX je opće prihvaćen način razmjene podataka pa ga tako koriste i web servisi kao što su Google Maps, Gmail, Youtube, Facebook i drugi.



Za potpunu funkcionalnost AJAX koristi i sljedeće tehnologije:

- XMLHttpRequest objekt – za asinkronu razmjenu podataka sa serverom),
- JavaScript/DOM – za prikaz/interakciju informacija,
- CSS – za uređivanje prikaza podataka,
- XML – često se koristi za formatiranje podataka za prijenos.

Temelj AJAX je **XMLHttpRequest** objekt koji se koristi za razmjenu podataka između klijenta i servera. AJAX je neovisan o pregledniku i platformi na kojoj se pokreće.

```
function loadXMLDoc(){  
    var xmlhttp;  
    if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari  
        xmlhttp=new XMLHttpRequest();  
    }  
    else {// code for IE6, IE5  
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  
    }  
    xmlhttp.onreadystatechange=function() {  
        if (xmlhttp.readyState==4 && xmlhttp.status==200) {  
            document.getElementById("myDiv").innerHTML=xmlhttp.responseText;  
        }  
    }  
    xmlhttp.open("GET","ajax_info.txt",true);  
    xmlhttp.send();  
}  
...  
<div id="myDiv"><h2>Let AJAX change this text</h2></div>  
<button type="button" onclick="loadXMLDoc()">Change Content</button>
```

10.1 Slanje zahtjeva serveru

Za slanje zahtjeva serveru koriste se dvije metode a to su:

- open(method, url, async) i
- send(string).

Metoda open služi za otvaranje zahtjeva prema serveru kao argumente prima tip zahtjeva (GET ili POST), URL (lokaciju datoteke na serveru), te način slanja podataka asinkrono ili sinkrono. Metoda send šalje zahtjev serveru kao argument može primiti tekst koji se koristi samo kod slanja podataka POST tipom.

10.1.1 GET i POST

Klijent serveru može poslati zahtjeva ili koristeći tip GET ili POST. GET tipom zahtjeva je jednostavniji i brži te se koristi u većini slučajeva. GET tipom zahtjeva se ne bi smjeli slati osjetljivi podaci jer su vidljivi u adresi stranice.

POST tip zahtjeva se koristi kada se treba poslati velika količina podataka pošto GET ima ograničenje u količini podataka koji se mogu poslati. POST se koristi kada se trebaju dohvatiti podaci na serveru koji se ažuriraju kao što je baza podataka ili neka datoteka koja se ažurira periodički.

10.1.1.1 GET

```
xmlhttp.open("GET","demo_get.asp",true);
xmlhttp.send();
```

Korištenjem GET tipom zahtjeva mogući je problem dohvaćanja pohranjenih (cached) datoteka iako se želi dohvatiti zadnja ažurirana datoteka. Kako bi se to spriječilo moguće je koristiti sljedeći primjer koda.

```
xmlhttp.open("GET","demo_get.asp?t=" + Math.random(),true);
xmlhttp.send();
```

Na ovaj način svaki puta kada se pozove željena datoteka kao parametar se pošalje i slučajni broj kako bi se svaki puta dohvatiла zadnja verzija datoteke. Ukoliko se serveru žele poslati podaci to je moguće napraviti na sljedeći način.

```
xmlhttp.open("GET","demo_get2.asp?fname=Henry&lname=Ford",true);
xmlhttp.send();
```

10.1.1.2 POST

```
xmlhttp.open("POST","demo_post.asp",true);
xmlhttp.send();
```

Za slanje podataka kao što je HTML forma potrebno je dodati HTTP header koristeći **setRequestHeader()**.

```
xmlhttp.open("POST","ajax_test.asp",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send("fname=Henry&lname=Ford");
```

10.2 Odgovor servera

Kada server primi i obradi zahtjev podatke može vratiti na dva načina koristeći ove metode:

- `responseText` – vraća odgovor kao tekst i
- `responseXML` – vraća odgovor kao XML.

10.2.1 ResponseText

Ova metoda se koristi ukoliko server treba vratiti podatke u obliku teksta. Ova metoda se često koristi za vraćanje podataka jer server najčešće šalje odgovor u obliku teksta.

```
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

10.2.2 ResponseXML

Ova metoda se koristi ukoliko server treba vratiti podatke u XML obliku. Metoda se koristi kada se podaci dohvaćeni od servera trebaju ili spremiti u XML datoteku ili prikazati u obliku tablice jer je puno lakše izraditi tablicu koristeći XML prikaz podataka nego ukoliko server vrati samo tekst.

```
xmlDoc=xmlhttp.responseXML;
txt="";
x=xmlDoc.getElementsByTagName("ARTIST");
for (i=0;i<x.length;i++){
    txt=txt + x[i].childNodes[0].nodeValue + "<br>";
}
document.getElementById("myDiv").innerHTML=txt;
```

10.3 onreadystatechange događaj

Ponekad kada se zahtjev pošalje serveru potrebno je ovisno o odgovoru izvršiti neke akcije. Svaki put kada se promijeni stanje **readyState** svojstva pokrene se događaj **onreadystatechange**. **ReadyState** svojstvo sadrži informacije os statusu **XMLHttpRequest** objekta.

Postoje tri važna svojstva **XMLHttpRequest** objekta a to su:

- onreadystatechange – pohranjuje funkciju (ili ime funkcije) koja će biti automatski pozvana svaki put kada se promijeni svojstvo readyState,
- readyState - sadrži status XMLHttpRequest:
 - 0: zahtjev nije pokrenut,
 - 1: veza sa serverom uspostavljena,
 - 2: zahtjev je zaprimljen,
 - 3: obrada zahtjeva te
 - 4: zahtjev gotov i odgovor spremam.
- Status – sadrži status stranice
 - 200 - „OK“ te
 - 404 - „Page not found“.

Kada je readyState 4 i status 200 odgovor je tada spremam te se nakon toga može koristiti odgovor koji je primljen od servera.

```
xmlhttp.onreadystatechange=function(){  
    if (xmlhttp.readyState==4 && xmlhttp.status==200){  
        document.getElementById("myDiv").innerHTML=xmlhttp.responseText;  
    }  
}
```

11 jQuery

jQuery je „lagana“ JavaScript biblioteka koja u sebi sadrži većinu JavaScript gotovih funkcija koje se mogu koristiti na puno lakši i jednostavniji način nego korištenje samo JavaScripta. jQuery koristi filozofiju „write less, do more“. Za korištenje jQuery-a nije potrebno potpuno poznавanje JavaScripta jer je većina komplikiranih funkcija već napisanih te ih je potrebno samo pozvati. jQuery je vrlo lako za naučiti je se koristi vrlo jednostavna sintaksa. Za potpunu upotrebu jQuery-a potrebno je poznavati HTML, CSS i JavaScript. jQuery pojednostavljuje korištenje JavaScript tehnika kao što su DOM i AJAX. jQuery biblioteka sadrži sljedeće značajke:

- HTML/DOM manipulaciju,
- CSS manipulaciju,
- HTML metode događaja,
- Efekte i animacije,
- AJAX te
- Ostalo.

Uz jQuery postoji i niz drugih JavaScript biblioteka no jQuery je jedna od popularnijih i najviše korištenih. Koriste ga i mnoge velike tvrtke kao što su: Google, Microsoft, IBM, Wikipedia, Wordpress i drugi.

11.1 jQuery instalacija

Postoje dva načina instalacije jQuery-a to su:

- skidanje jQuery biblioteke sa službene stranice jquery.com te
- uključivanjem jQuery-a sa CDN-a (Content Delivery Network).

Odlaskom na službenu stranicu jquery.com moguće je skinuti dvije verzije jQuery-a.

Produkcijska verzija koja se koristi na gotovim stranicama i razvojna verzija koja se koristi prilikom izrade i razvoja stranice. Produkcijska verzija je kompresirana i optimizirana verzija kako bi se što više ubrzalo učitavanje stranica.

```
<head>
<script src="jquery-1.9.1.min.js"></script>
</head>
```

Ukoliko se ne želi skidati jQuery biblioteka moguće ju je direktno uključiti koristeći CDN (Content Delivery Network) koristeći ili Google-ov web servis ili Microsoft-ov web servis.

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">
</script>

<script src="//ajax.aspnetcdn.com/ajax/jQuery/jquery-1.9.1.min.js">
</script>
```

Koristeći ovaj način uključivanja jQuery-a na stranicu ima nekoliko prednosti a to su:

- uvijek dostupna zadnja verzija bez potrebe ručnog ažuriranja biblioteke,
- brže učitavanje (ukoliko nekoliko stranica koristi jQuery automatski se skida samo prvi puta te se prilikom sljedećeg pozivanja dohvaća iz privremene memorije a ne interneta).
- dohvaćanje sa najbližeg servera.

11.2 Osnove jQuery-a

Uz pomoć jQuery-a označavaju se HTML elementi i nad njima se izvršavaju akcije. Osnovna sintaksa izgleda ovako: **`$(selector).action()`**

- \$ – znak za definiranje/pristup jQuery-a,
- (selector) – za pronalaženje HTML elementa,
- action() – akcija koja će se dogoditi nad elementom.

Kako bi se spriječilo pokretanje jQuery koda prije nego se stranica u potpunosti učita može se koristiti sljedeći kod.

```
$(document).ready(function(){  
    //jQuery methods go here...  
});
```

11.2.1 jQuery selektori

jQuery selektori su najvažniji dio jQuery biblioteke jer oni služe za pronalaženje elemenata nad kojim se želi izvršavati određena akcija. jQuery selektori omogućavaju označavanje i manipulaciju HTML elemenata. Elemente je moguće označiti preko njihovog id-a, klase, tipa, atributa, vrijednosti i drugih svojstava. Većina jQuery selektora je bazirano na CSS selektorima. Primjer korištenja selektora:

- Element selektor
 - `$("p")` – dohvaćanje svih `<p>` elemenata
 - `$(this)` – dohvaćanje trenutnog elementa
- `#id` selektor
 - `$("#test")` – dohvaćanje elementa sa `id=„test”`
- `.class` selektor
 - `$(".test")` – dohvaćanje elementa sa `class=„test”`

```
$(this).hide() – skriva trenutni element  
$("p").hide() - skriva sve <p> elemente  
$(".test").hide() – skriva sve elemente sa klasom=“test”  
$("#test").hide() – skriva sve elemente sa id=“test”
```

Syntax	Description
<code>\$("")</code>	Selects all elements
<code>\$(this)</code>	Selects the current HTML element
<code>\$("p.intro")</code>	Selects all <code><p></code> elements with <code>class="intro"</code>
<code>\$("p:first")</code>	Selects the first <code><p></code> element
<code>\$("ul li:first")</code>	Selects the first <code></code> element of the first <code></code>
<code>\$("ul li:first-child")</code>	Selects the first <code></code> element of every <code></code>
<code>\$("[href]")</code>	Selects all elements with an <code>href</code> attribute
<code>\$("a[target='_blank']")</code>	Selects all <code><a></code> elements with a <code>target</code> attribute value equal to <code>"_blank"</code>
<code>\$("a[target!='_blank']")</code>	Selects all <code><a></code> elements with a <code>target</code> attribute value NOT equal to <code>"_blank"</code>
<code>\$(":button")</code>	Selects all <code><button></code> elements and <code><input></code> elements of <code>type="button"</code>
<code>\$("tr:even")</code>	Selects all even <code><tr></code> elements
<code>\$("tr:odd")</code>	Selects all odd <code><tr></code> elements

11.2.2 jQuery funkcije

Ukoliko je potrebno napraviti veliki broj stranica i ako se žele omogućiti jednostavno održavanje jQuery funkcija moguće je sve jQuery funkcije prebaciti u zasebne .js datoteke.

```
<head>
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">
</script>
<script src="my_jquery_functions.js"></script>
</head>
```

11.2.3 jQuery događaji

jQuery ima već ugrađenu podršku za razne događaje koji se mogu dogoditi prilikom korištenja web stranica. Neki od događaja su koji se mogu dogoditi su:

- Pomicanje miša preko elementa,
- Označavanje checkbox-a te
- Klik na neki element.

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

Kod korištenja jQuery-a moguće je koristiti sljedeće događaje:

- `$(document).ready()` – poziva se kada se web stranica u potpunosti učita,
- `click()` – poziva se kada se napravi klik lijeve tipke miša,
- `dblclick()` – poziva se kada se napravi dvoklik lijeve tipke miša,
- `mouseenter()` - poziva se kada pokazivač miša dođe u područje elementa,
- `mouseleave()` – poziva se kada pokazivač miša napusti element,
- `mousedown()` – poziva se kada se pritisne lijeva tipka miša,
- `mouseup()` – poziva se kada se otpusti lijeva tipka miša,
- `hover()` – poziva se kada se mišom prođe iznad elementa (kombinacija sa `mouseenter()` i `mouseleave()`),
- `focus()` – poziva se kada element dobije fokus te
- `blur()` – poziva se kada element izgubi fokus.

11.3 Napredne opcije jQuery-a

Uz osnovne opcije koje pruža jQuery, unutar jQuery biblioteke postoje i naprednije funkcionalnosti kao što su razni efekti, dio koji služi za povezivanje sa HTML-om i CSS-om te dio za korištenje AJAX-a.

11.3.1 jQuery efekti

jQuery efekti su jedna od naprednjih opcija koje se mogu koristiti unutar jQuery-a kako bi se poboljšao grafički izgled stranice. Neki od efekata su već poznati iz CSS-a no neki su potpuno novi vizualni efekti.

11.3.1.1 *jQuery hide() i show()*

Dva nova vizualna efekta su `hide()` koji služi sa skrivanje elemenata te `show()` za prikazivanje elemenata.

```
$(selector).hide(speed,callback);
```

```
$(selector).show(speed,callback);
```

Opcijski argumenti **speed** određuje kojom će se brzinom elementi sakriti/prikazati. Moguće vrijednosti argumenta su: „slow“, „fast“ ili vrijeme u milisekundama. Drugi opcijски argument **callback** je funkcija koja se može pokrenuti nakon što se metoda `show()` ili `hide()` izvrši.

```
$("#hide").click(function(){
    $("p").hide();
});

$("#show").click(function(){
    $("p").show();
});
```

```
 $("button").click(function(){
    $("p").hide(1000);
});
```

11.3.1.2 jQuery *toggle()*

Kako su metode **hide()** i **show()** međusobno povezane ponekad bi bilo komplikirano koristiti prvo jednu pa nakon toga drugu metodu. Umjesto toga može se koristiti metoda **toggle()** koja mijenja obje metode te služi prebacivanje iz jednog stanja u drugo.

```
$(selector).toggle(speed,callback);
```

```
 $("button").click(function(){
    $("p").toggle();
});
```

11.3.1.3 jQuery *fading*

Postoji nekoliko fading metoda koje se mogu koristiti sa jQuery-em to su:

- **fadeIn()**,
- **fadeOut()**,
- **fadeToggle()** te
- **fadeTo()**.

Koristeći fading efekte moguće je postići polagano prikazivanje ili skrivanje elemenata što može uvelike dopridonijeti poboljšanju grafičkog dizajna cijele stranice.

11.3.1.3.1 fadeIn()

Metoda koji služi za polagano prikazivanje elemenata. Opcionalni parametar **speed** ima redefiniranu vrijednost koja je ekvivalent 400 milisekundi.

Za vrijednost brzine moguće je staviti ključne riječi „slow“ čija je vrijednost ekvivalenta 600 milisekundi ili ključnu riječ „fast“ čija je vrijednost ekvivalenta vremenu od 200 milisekundi. Ukoliko niti jedna od predefiniranih vrijednosti ne zadovoljava potreba moguće je postaviti bilo koje vrijeme kroz koje će efekt izvršavati. Vrijeme koje se želi upisati unaša se u milisekundama.

```
$(selector).fadeIn(speed,callback);
```

```
 $("button").click(function(){
  $("#div1").fadeIn();
  $("#div2").fadeIn("slow");
  $("#div3").fadeIn(3000);
});
```

11.3.1.3.2 fadeOut()

Metoda koji služi za polagano skrivanje elemenata. Vrlo je slična kao i metoda **fadeIn()** koja ima iste parametre **speed** i **callback**.

```
$(selector).fadeOut(speed,callback);
```

```
 $("button").click(function(){
  $("#div1").fadeOut();
  $("#div2").fadeOut("slow");
  $("#div3").fadeOut(3000);
});
```

11.3.1.3.3 fadeToggle()

Metoda koji služi za prebacivanje između fadeIn i fadeOut metoda. Kada je element prikazan prilikom izvršavanja efekta pokrenuti će se **fadeOut()** metoda a kada je objekt skriven pokrenuti će se **fadeIn()** metoda.

```
$(selector).fadeOut(speed,callback);
```

```
$( "button" ).click(function(){
  $( "#div1" ).fadeToggle();
  $( "#div2" ).fadeToggle("slow");
  $( "#div3" ).fadeToggle(3000);
});
```

11.3.1.3.4 fadeTo()

Metoda koji služi za postavljanje elementa do određene vrijednosti transparentnosti između 0 i 1. Ova metoda se razliku po još jednom parametru koji prima a to je **opacity** koji predstavlja neprozirnost.

```
$(selector).fadeTo(speed,opacity,callback);
```

```
$( "button" ).click(function(){
  $( "#div1" ).fadeTo("slow",0.15);
  $( "#div2" ).fadeTo("slow",0.4);
  $( "#div3" ).fadeTo("slow",0.7);
});
```

11.3.1.4 jQuery sliding

Postoji nekoliko metoda koji omogućavaju klizne efekte a to su:

- slideDown(),
- slideUp() te
- slideToggle().

Ponekad je potrebno kreirati klizne efekte koji omogućavaju pomicanje elemenata prema dolje ili prema gore.

11.3.1.4.1 slideDown()

Metoda koji služi za polagano spuštanje elementa prema dolje.

```
$(selector).slideDown(speed,callback);
```

```
$("#flip").click(function(){
    $("#panel").slideDown();
});
```

11.3.1.4.2 slideUp()

Metoda koji služi za polagano dizanje elementa prema gore.

```
$(selector).slideUp(speed,callback);
```

```
$("#flip").click(function(){
    $("#panel").slideUp();
});
```

11.3.1.4.3 slideToggle()

Metoda koji služi za prebacivanje između metoda slideDown() i slideUp(). Ukoliko se element nalazi gore metodom **slideToggle()** pozvati će se metoda **slideDown()** koja će spustiti element prema dolje. Ukoliko se element nalazi dolje pozvati će se metoda **slideUp()** koja će podignuti element prema gore.

```
$(selector).slideToggle(speed,callback);
```

```
$("#flip").click(function(){
    $("#panel").slideToggle();
});
```

11.3.1.5 jQuery animation

Koristeći jQuery mogu je kreirati vlastite animacije, za to se može koristiti metoda **animate()**.

```
$(selector).animate({params},speed,callback);
```

Metoda ima nekoliko parametara od kojih je najbitniji prvi parametar unutar kojega se mora postaviti CSS atribut koji se želi animirati.

Predefinirana pozicija HTML elemenata je **static** što znači da im nije moguće mijenjati poziciju. Ukoliko se želi promijeniti pozicija elementa potrebno je prvo promijeniti CSS svojstvo **position** na **relative**, **absolute** ili **fixed**. Ukoliko postoji nekoliko animacija koje se žele izvršiti jedna za drugom jQuery sve animacije spremi u red izvođenja te izvodi jednu po jednu nakon što se prethodna završi.

```
$( "button" ).click(function() {
    var div=$("div");
    div.animate({height:'300px',opacity:'0.4'},"slow");
    div.animate({width:'300px',opacity:'0.8'},"slow");
    div.animate({height:'100px',opacity:'0.4'},"slow");
    div.animate({width:'100px',opacity:'0.8'},"slow");
});
```

```
$( "button" ).click(function() {
    var div=$("div");
    div.animate({left:'100px'},"slow");
    div.animate({fontSize:'3em'},"slow");
});
```

11.3.1.6 Manipulacija više svojstava

Ukoliko postoji potreba ta manipulacijom nekoliko svojstava to je također moguće napraviti koristeći metodu **animate()**. Važno je znati kako bi se mogla koristiti sva CSS svojstva koja se sastoje od dvije ili više riječi moraju biti spojena te se moraju koristiti nazivi u obliku camelCase, na primjer **padding-left** se mora pisati **paddingLeft**. Ako se želi koristiti animacija boja potrebno je koristiti dodatak **Color Animations** koji se može skinuti sa jQuery.com stranice.

```
$( "button" ).click(function(){
    $("div").animate({
        left:'250px',
        opacity:'0.5',
        height:'150px',
        width:'150px'
    });
});
```

11.3.1.7 Korištenje relativnih vrijednosti

Koristeći **animate()** metodu moguće je koristiti relativne vrijednosti što znači da će se vrijednosti mijenjati relativno na trenutne vrijednosti. Za to se mogu koristiti operatori `+=` i `-=`.

```
 $("button").click(function(){
    $("div").animate({
        left:'250px',
        height:'+=150px',
        width:'+=150px'
    });
});
```

11.3.1.8 Korištenje predefiniranih vrijednosti

Moguće je koristiti i predefinirane vrijednosti kao što su „show“, „hide“ ili „toggle“.

```
 $("button").click(function(){
    $("div").animate({
        height:'toggle'
    });
});
```

11.3.1.9 Metoda stop()

Svaka animacija koje se pokrene ne može se zaustaviti dok se u potpunosti ne izvrši. Ukoliko se animacija želi završiti ranije moguće je koristiti metodu **stop()** koja će odmah zaustaviti daljnje izvođenje animacije. Metoda može primiti dva opcionalna parametra, prvi parametar je opcija ukoliko se žele zaustaviti sve animacije, dok je drugi opcija da se nakon zaustavljanja animacije animacija dođe na kraj zadanih postavki.

```
 $(selector).stop(stopAll,goToEnd);
```

```
 $("#stop").click(function(){
    $("#panel").stop();
});
```

11.3.1.10 Povezivanje više akcija/metoda

Ponekad je potrebno povezati više akcija/metoda u jedno, to je moguće napraviti nizanjem metoda.

```
$("#p1").css("color","red").slideUp(2000).slideDown(2000);
```

Popis svih efekata koji se mogu koristiti unutar jQuery-a.

Metoda	Opis
animate()	Runs a custom animation on the selected elements
clearQueue()	Removes all remaining queued functions from the selected elements
delay()	Sets a delay for all queued functions on the selected elements
dequeue()	Removes the next function from the queue, and then executes the function
fadeIn()	Fades in the selected elements
fadeOut()	Fades out the selected elements
fadeTo()	Fades in/out the selected elements to a given opacity
fadeToggle()	Toggles between the fadeIn() and fadeOut() methods
finish()	Stops, removes and completes all queued animations for the selected elements
hide()	Hides the selected elements
queue()	Shows the queued functions on the selected elements
show()	Shows the selected elements
slideDown()	Slides-down (shows) the selected elements
slideToggle()	Toggles between the slideUp() and slideDown() methods
slideUp()	Slides-up (hides) the selected elements
stop()	Stops the currently running animation for the selected elements
toggle()	Toggles between the hide() and show() methods

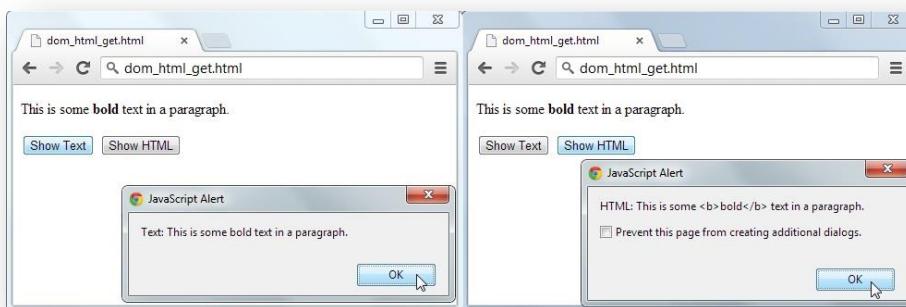
11.3.2 JQuery HTML DOM

Unutar jQuery-a moguće je mijenjati i manipulirati HTML elementima i atributima koristeći razne metode. Za dohvaćanje i postavljanje sadržaja koriste se metode:

- `text()` – postavlja ili vraća tekst koji se nalazi unutar označenog elementa,
- `html()` – postavlja ili vraća sadržaj označenog elementa (uključujući i HTML oznake) te
- `val()` – postavlja ili vraća vrijednosti polja formi.

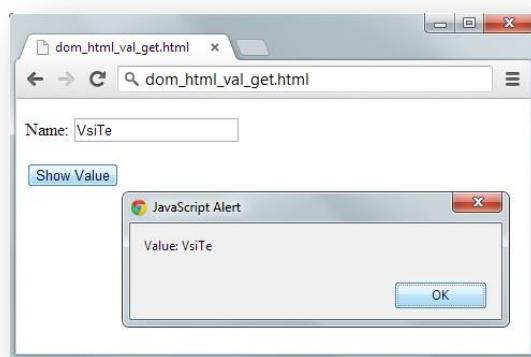
Dohvaćanje teksta i html:

```
e$("#btn1").click(function(){
    alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
    alert("HTML: " + $("#test").html());
});
```



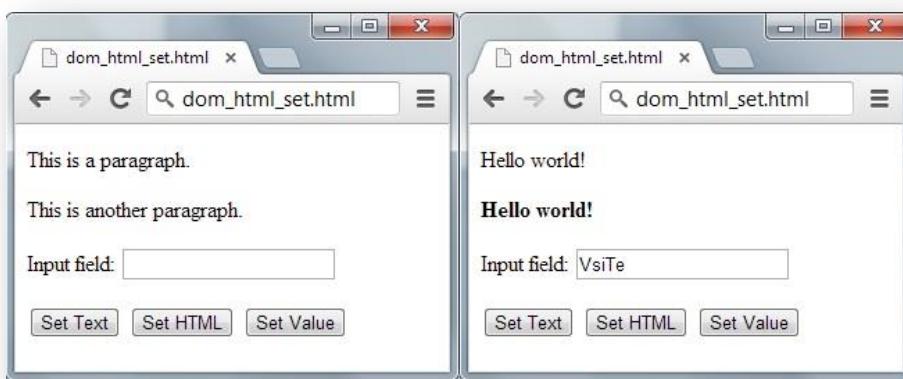
Dohvaćanje vrijednosti elementa:

```
$("#btn1").click(function(){
    alert("Value: " + $("#test").val());
});
```



Postavljanje sadržaja:

```
$("#btn1").click(function(){
    $("#test1").text("Hello world!");
});
$("#btn2").click(function(){
    $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
    $("#test3").val("VsiTe");
});
});
```



11.3.2.1 Dodavanja sadržaja

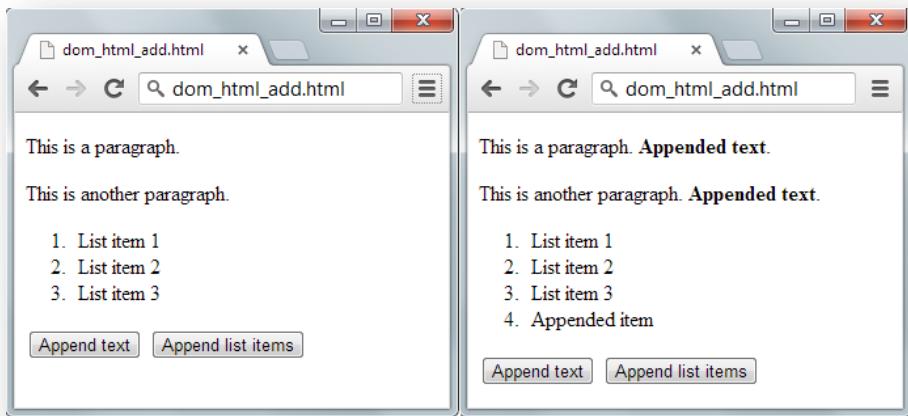
Za dodavanje sadržaja mogu se koristiti sljedeće metode:

- `append()` – ubacuje sadržaj na kraj označenog elementa,
- `prepend()` – ubacuje sadržaj na početak označenog elementa,
- `after()` – ubacuje sadržaj nakon označenog elementa te
- `before()` – ubacuje sadržaj prije označenog elementa.

11.3.2.1.1 append()

jQuery **append()** metoda dodaje sadržaj na kraj označenog HTML elementa.

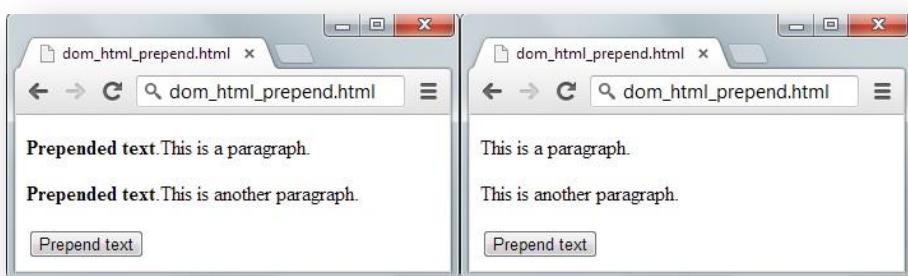
```
$("#btn1").click(function(){
    $("p").append("<b>Appended text</b>.");
});
$("#btn2").click(function(){
    $("ol").append("<li>Appended item</li>");
});
```



11.3.2.1.2 **prepend()**

jQuery **prepend()** metoda dodaje sadržaj na početak označenog HTML elementa.

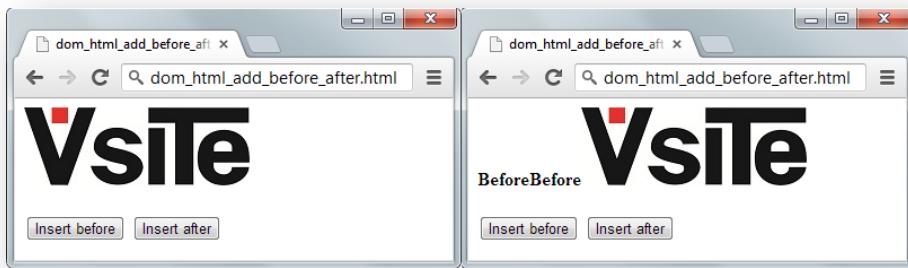
```
$("#btn1").click(function(){
    $("p").prepend("<b>Prepended text</b>.");
});
```



11.3.2.1.3 **before()**

jQuery **before()** metoda dodaje sadržaj prije označenog HTML elementa.

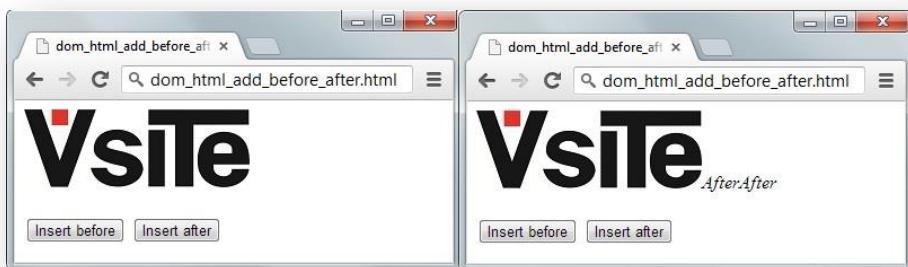
```
$("#btn1").click(function(){
    $("img").before("<b>Before</b>");
});
```



11.3.2.1.4 after()

jQuery **after()** metoda dodaje sadržaj nakon označenog HTML elementa.

```
$("#btn1").click(function(){
    $("img").after("<i>After</i>");
});
```

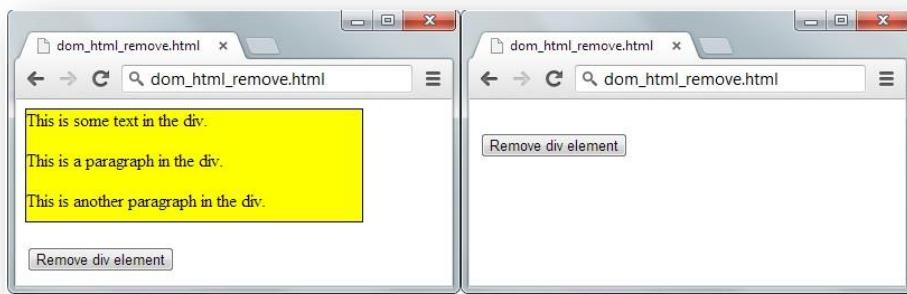


11.3.2.2 Brisanje elemenata/sadržaja

Pomoću jQuery-a moguće je brisati elemente ili sadržaj elemenata. Za to se koriste sljedeće metode:

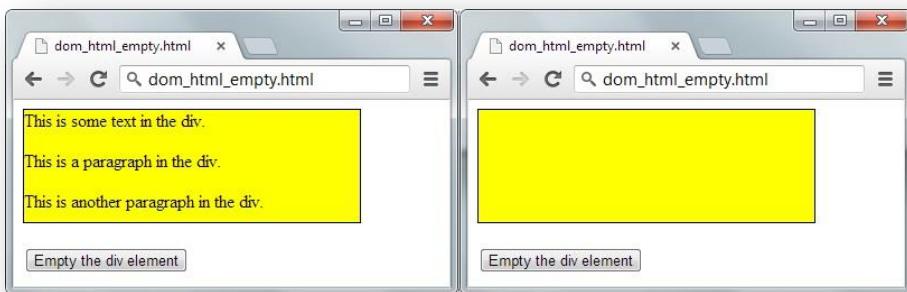
- `remove()` – briše označene elemente i sve njihove potomke te

```
$("button").click(function(){
    $("#div1").remove();
});
```



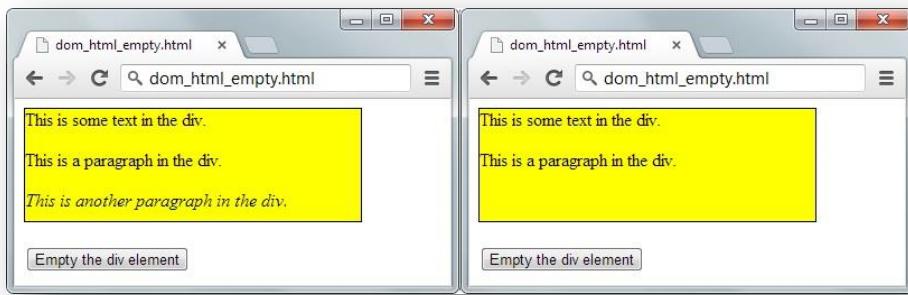
- empty() – briše sve potomke označenih elemenata.

```
$(“button”).click(function(){  
    $(“#div1”).empty();  
});
```



Ukoliko postoji potreba za filtriranjem brisanja elemenata ili sadržaja to je moguće napraviti koristeći parametar unutar metode.

```
$(“button”).click(function(){  
    $("p").remove(“.italic”);  
});
```



11.3.3 CSS klase

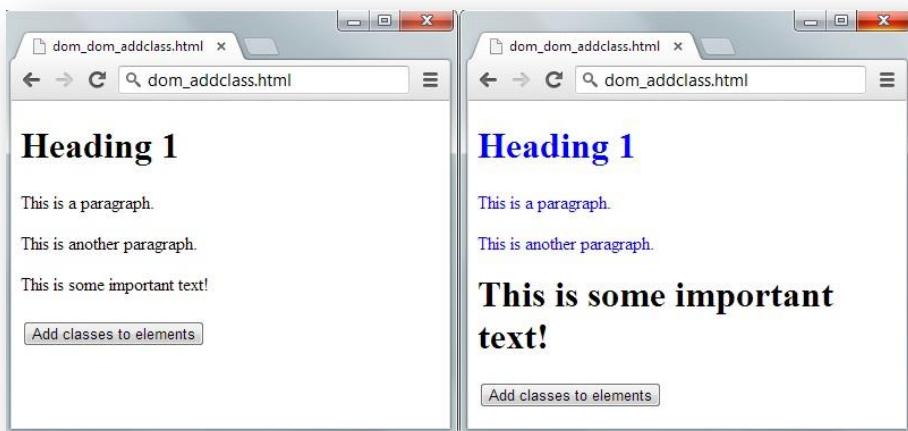
Pomoću jQuery-a moguće je vrlo lako manipulirati CSS klasama. Za to se najčešće koriste sljedeće metode:

- `addClass()` – dodaje jednu ili više klasa označenom elementu,

```

$("button").click(function(){
    $("h1,h2,p").addClass("blue");
    $("div").addClass("important");
});

```

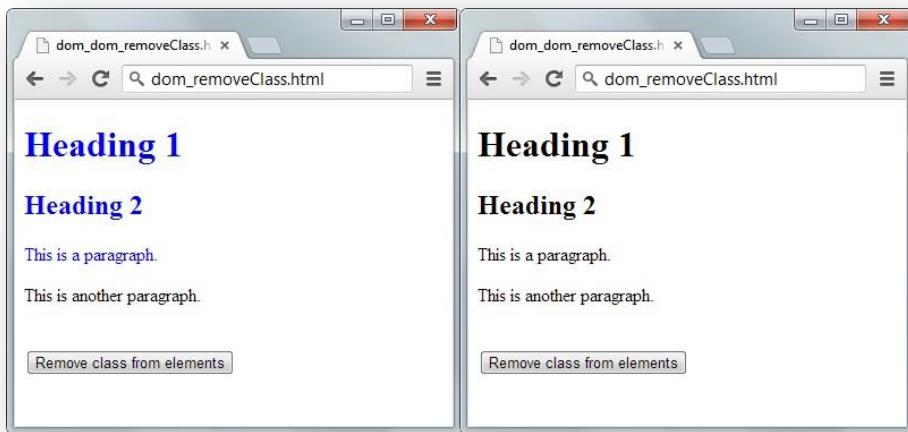


- `removeClass()` – briše jednu ili više klasa označenom elementu,

```

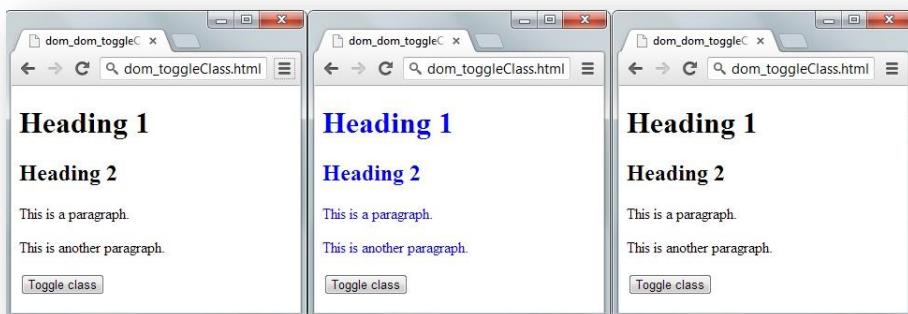
$("button").click(function(){
    $("h1,p").removeClass("blue");
});

```



- `toggleClass()` – prebacuje između dodavanja ili brisanja klase označenog elementa te

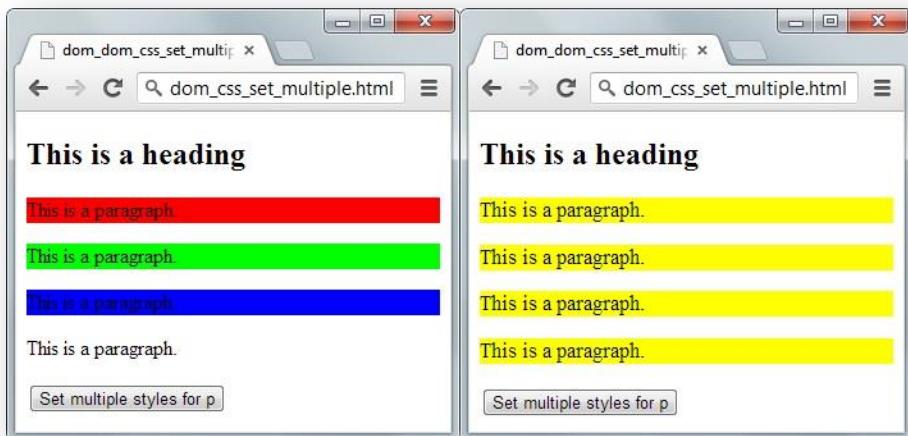
```
$(“button”).click(function(){  
    $("h1,h2,p").toggleClass(“blue”);  
});
```



- `css()` – Postavlja ili vraća svojstva stila.

```
css(“propertyname”, “value”);
```

```
$(“button”).click(function(){  
    $("p").css({“background-color”:“yellow”, “font-size”:“105%”});  
});
```



11.3.4 jQuery HTML DOM - Dimensions

jQuery omogućava dohvaćanje ili postavljanje dimenzija elementa uključujući i dimenzije ekrana i dimenzije prozora preglednika. Neke od metoda koje je moguće koristiti su:

- `width()` – postavlja ili dohvaća širinu elementa (bez padding-a, border-a i margina),

```
$("#div1").width();
```

- `height()` – postavlja ili dohvaća visinu elementa (bez padding-a, border-a i margina),

```
$("#div1").height();
```

- `innerWidth()` – postavlja ili dohvaća širinu elementa (uključujući padding),

```
$("#div1").innerWidth();
```

- `innerHeight()` – postavlja ili dohvaća visinu elementa (uključujući padding),

```
$("#div1").innerHeight();
```

- `outerWidth()` – postavlja ili dohvaća širinu elementa (uključujući padding i border),

```
$("#div1").outerWidth();
```

- `outerHeight()` - postavlja ili dohvaća visinu elementa (uključujući padding i border),

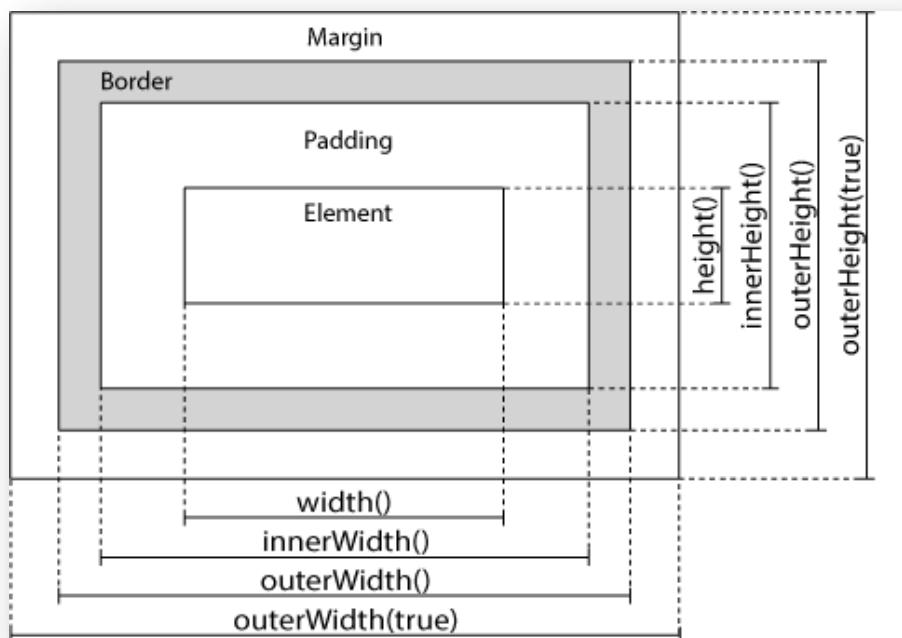
```
$("#div1").outerHeight();
```

- `outerWidth(true)` – postavlja ili dohvaća širinu elementa (uključujući padding, border i margine) te

```
$("#div1").outerWidth(true);
```

- `outerHeight(true)` - postavlja ili dohvaća visinu elementa (uključujući padding, border i margine).

```
$("#div1").outerHeight(true);
```



11.3.5 jQuery AJAX

JQuery metoda **`load()`** uvelike olakšava korištenje AJAX-a na stranicama. Za korištenje AJAX tehnologije koristeći jQuery potrebna je samo **`load`** metoda koja je vrlo jednostavna ali vrlo moćna. Služi za slanje zahtjeva serveru te prihvati odgovora koji je stigao od servera.

```
$(selector).load(URL,data,callback);
```

Load metoda prima tri parametra a tu su:

- Obavezni URL parameter određuje URL od kuda se žele dohvatiti podaci,
- Opcionalni parametar određuje querystring par ključ/vrijednost koji se šalje zajedno sa zahtjevom te
- Opcionalni parametar callback je ime funkcije koja će se pozvati nakon što se izvrši **load()** metoda.

```
demo_test.txt
<h2>jQuery and AJAX is FUN!!!</h2>
<p id="p1">This is some text in a paragraph.</p>

jQuery_ajax_test.html
$("#div1").load("demo_test.txt");
...
$("#div1").load("demo_test.txt #p1");
```

Opcionalni callback parameter određuje funkciju koja će se izvršiti nakon što se do kraja izvrši metoda **load()**. Callback funkcija može imati sljedeće parametre:

- responseTxt – sadrži rezultirajući sadržaj ako je poziv uspio,
- statusTxt – sadrži status poziva te
- xhr – sadrži XMLHttpRequest objekt.

```
 $("button").click(function(){
  $("#div1").load("demo_test.txt",function(responseTxt,statusTxt,xhr){
    if(statusTxt=="success")
      alert("External content loaded successfully!");
    if(statusTxt=="error")
      alert("Error: "+xhr.status+": "+xhr.statusText);
  });
});
```

Podaci koji se žele poslati serveru mogu se poslati metodama:

- GET – slanje podataka GET metodom je najčešći način slanja podataka jer je brže i u većini slučajeva najoptimalnije. Metoda koristi HTTP GET zahtjev za slanje podatka prema serveru.

```
$.get(URL,callback);
```

```
$(“button”).click(function(){
  $.get(“demo_test.asp”,function(data,status){
    alert(“Data: ” + data + “\nStatus: ” + status);
  });
});
```

- POST – slanje podataka POST metodom se najčešće koristi kada se šalju osjetljivi podaci kao što su korisnički podaci ili velika količina podatka. Metoda koristi HTTP POST zahtjev za slanje podatka prema serveru.

```
$.post(URL,callback);
```

```
$(“button”).click(function(){
  $.post(“demo_test_post.asp”, {
    name:”VsiTe”,
    city:”Zagreb”
  },
  function(data,status){
    alert(“Data: ” + data + “\nStatus: ” + status);
  });
});
```

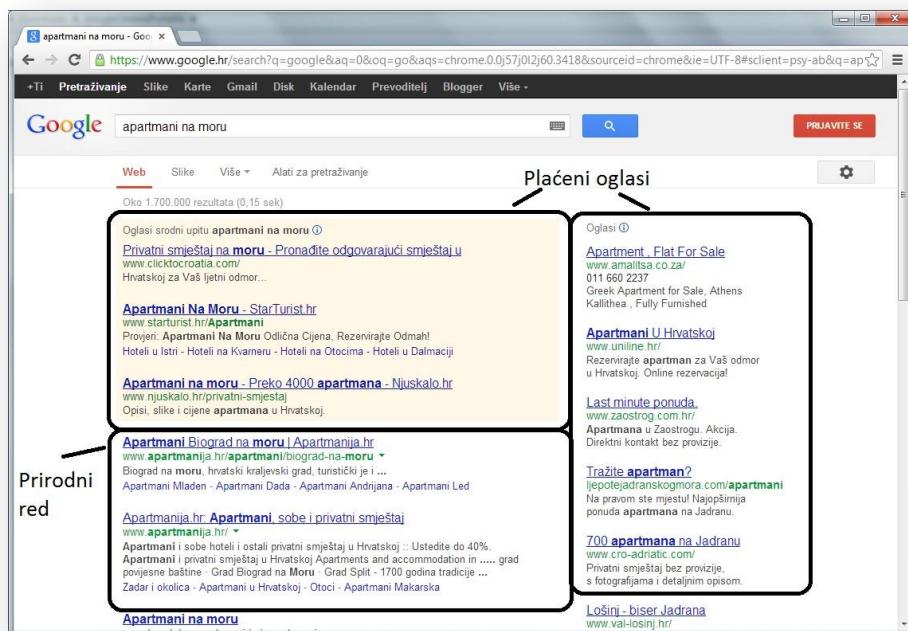
11.3.6 Metoda **noConflict()**

Čest je slučaj korištenja nekoliko različitih biblioteka koje mogu koristiti posebni znak kao što jQuery koristi znak **\$**. Kako bi se spriječili problemi ova metoda može pomoći na vrlo jednostavan način **noConflict()** metoda otpušta **\$** oznaku identifikatora, kako bi ga druge skripte mogle koristiti.

```
var jq = $.noConflict();
jq(document).ready(function(){
  jq(“button”).click(function(){
    jq(“p”).text(“jQuery is still working!”);
  });
});
```

12 Optimizacija stranice za tražilice

Pojam SEO je skraćeni naziv za pojam Search Engine Optimization odnosno optimizacija stranica za tražilice. SEO je proces prilagodbe web stranica, odnosno dio internetskog marketinga koji se bavi pozicioniranjem web stranica na rezultatima pretrage, za ključne riječi koje najviše odgovaraju sadržaju stranice. Cilj optimizacije web stranica je što bolji položaj na rezultatu pretrage tzv. SERP (Search engine Results Page) kako za strogo specifične ključne riječi tako i za širi spektar riječi koje su vezane za ključne riječi. Dobro optimizirane stranice ostvariti će visoke pozicije na rezultatima pretrage za gotovo sve riječi relevantne za tematiku koju web stranica obrađuje. Optimiziranjem web stranica dovode se kako ciljani tako i relevantni posjetitelji na web stranicu.



Kako je Google najpoznatija web tražilica dovoljno je naučiti na koji način se može optimizirati web stranica za google te će se ta pravila moći primijeniti i za druge tražilice kao što su yahoo.com ili bing.com. Google kao i ostale tražilice koriste automatizirane alate koji prate, čitaju, analiziraju uspoređuju i rangiraju web stranice. Tražilice rangiraju stranice po relevantnosti sadržaja i po važnosti i to padajućim redoslijedom. Ukoliko stranica nije među prvih 20 rezultata mala je vjerojatnost da će posjetitelj doći do stranice je vrlo malo korisnika pregledava drugu, treću ili veću stranicu pretrage.

Kako bi se nona stranica počela prikazivati u rezultatima pretrage, Google je prvo mora posjetiti tj. indeksirati sadržaj web stranice. Google posjećuje stranice automatiziranim programima koji se zovu **roboti** ili **spideri (bots)**. Oni u detalje pregledavaju sve stranice i podstranice. Najčešće počinju od naslovne i prate linkove u dubinu sadržaja na pojedinoj domeni te im ovisno o sadržaju dodjeljuje važnost. Poslije što više linkova s drugih stranica vodi u dubinu sadržaja raste važnost sadržaja te će se podizati na rezultatima pretrage.

Nova stranica će biti posjećena od strane tražilice tek kada Google pronađe link na nekoj drugoj stranici prema novonastaloj stranici. Učestalost posjete Google spider-a poznatog i kao Googlebot ovisi o popularnosti web stranice, starosti kvaliteti sadržaja, broju dolaznih linkova, učestalosti promjena i slično. Važnije web stranice bit će posjećivane češće na primjer portali sa vijestima, blogovi, forumi dok će druge stranice koje nemaju učestale promjene sadržaja biti posjećivane možda jednom mjesечно.

Tehnike optimizacije podijeljene su na dvije osnovne kategorije, poželjnu „**White hat**“ i nepoželjnu „**Black hat**“. Ukoliko se koristi nepoželjna metoda, nakon nekog vremena tražilica će izbaciti stranicu iz svojeg indeksa jer će primjetiti sumnjiva ponašanja. Kod izrade stranica bitno je prilagoditi sadržaj posjetitelju a ne tražilicama.

Tražilice skrivaju svoje algoritme i povremeno objave pojedinosti na koji način rangiraju web stranice no istovremeno izdaju se naputci o pravilima koje je potrebno poštivati, tako je Google napravio cijeli set sugestija naziva „**Search Engine Optimization Starter Guide**“.

Ukoliko se samo jednom iskoristi nepoželjna „**Black hat**“ metoda postoji mogućnost kažnjavanja i spuštanja rangiranja na najnižu razinu.

Najstarija **Black hat** metoda koja se i danas može vidjeti je postavljanje tekstova bogatih ključnim riječima bijele boje na bijeloj pozadini. To je ujedno i najčešći način spamanja tražilica. Skrivanje teksta u div elementima bogatim ključnim riječima a veličine 1x1 px je također jedna od nepoželjnih metoda. Kreiranje **Doorway** stranica koje služe za preusmjeravanje korisnika ili koje sadrže samo jednu stranicu koja je bogata linkovima prema drugim stranicama je također zabranjeno. Nedavno postalo je popularno izrađivati stranice sa duplim sadržajem, jedan sadržaj za prikaz informacija korisniku a drugi za tražilice, to se također kažnjava. Sve ove metode mogu imati vrlo kratkotrajan učinak no nakon što se otkriju potpuno gube svoj smisao.

12.1 Optimizacija stranica

Optimizacija počinje i prije nego se kreće u izradu web stranica. Nulti korak u izradi web stranica je pronaći ključne riječi koje će biti vezane za sadržaj stranica. Najveća greška prilikom izrade web stranica je kopiranje gotovih sadržaja sa postojećih stranica. Optimizacija ima dva osnovna dijela a to su:

- Onsite optimizacija – promjene/prilagodbe koje je potrebno napraviti na samoj web stranici i
- Offsite optimizacija – izrada dolaznih linkova sa relevantnih članaka, osmišljavanje kvalitetnih članaka s kojih će se povezivati vlastite web stranice.

12.1.1 Odabir ključnih riječi

Važno je znati da se svaka pojedina stranica optimizira za nekoliko ključnih riječi. Naslovna se optimizira za nekoliko najvažnijih, zatim svaka podstranica samo za najvažnije riječi koje se spominju u sadržaju. Pronalazak pravih ključnih riječi važan je segment jer ključne riječi koje imaju velik broj pretraga ne jamče nužno i kvalitetne posjetitelje koji će rezultirati kupnjom/registracijom/kontaktom.

Ključne riječi koje imaju velik broj pretraga vrlo vjerojatno imaju veliku konkureniju što znači i manju vjerojatnost da će posjetitelj otići upravo na novostvorenu stranicu. Google nudi besplatni alat uz pomoć kojega je moguće potražiti ključne riječi. Na stranici:

https://adwords.google.com/o/Targeting/Explorer?_c=1000000000&_u=1000000000&o=cues&ideaRequestType=KEYWORD_IDEAS

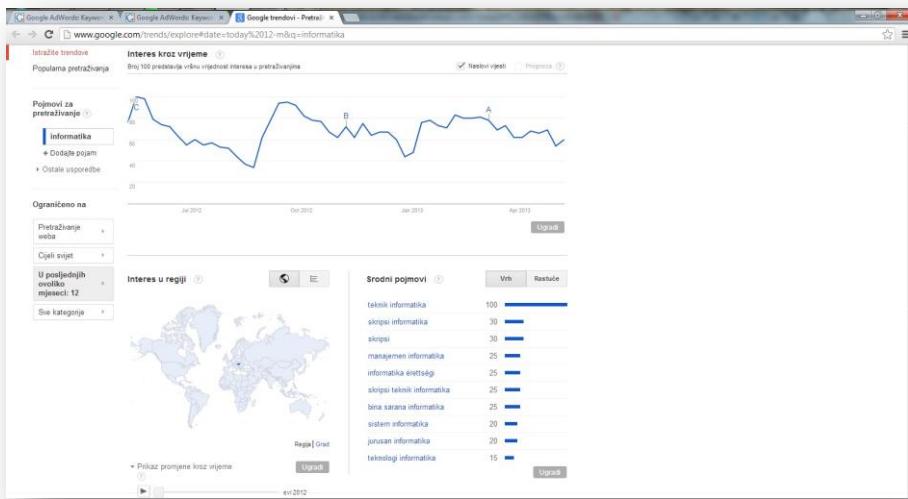
moguće je testirati ključne riječi ili fraze ovisno o lokaciji i jeziku na kojem će se stranice prikazivati. Alat će prikazati koliko često se traži upisana ključna riječ ili fraza te kolika je konkurentnost.

The screenshot shows the Google AdWords Keyword Planner interface. In the search bar, the term 'informatica' is entered. Below the search bar, there are filters for 'Website' (set to 'www.google.com/page.html') and 'Category' (set to 'Apparel'). On the left, a sidebar shows 'Saved ideas (0)' with options for 'My keyword ideas' and 'My ad group ideas'. The main area displays a list of suggested keywords under the heading 'Find keywords'. One suggestion, 'informatica', is highlighted. To the right of the suggestions, there are dropdown menus for 'Locations and languages' and 'Include specific content'. A note at the bottom says 'Letters are not case-sensitive'. At the bottom of the page, there are buttons for 'Search' and 'Sign in with your AdWords login information to see the full set of ideas for this search'.

This screenshot shows the same Google AdWords Keyword Planner interface as the previous one, but it has been expanded to show detailed search results. The search term 'informatica' is still in the search bar. The results table is titled 'Search terms (1)' and contains one entry: 'informatica'. This entry is further broken down into 'Keyword ideas (21)'. The table includes columns for 'Keyword', 'Competition', 'Global Monthly Searches', and 'Local Monthly Searches'. The first few rows of the table are as follows:

Keyword	Competition	Global Monthly Searches	Local Monthly Searches
informatica	Low	1,220,000	27,100
instar informatica	Low	1,900	1,900
medicinska informatica	Low	480	140
www.informatica	Low	58	28
in star informatica	Low	2,400	1,900
postiona informatica	Low	1,900	390
racunalne komponente	Low	260	260
gd informatica	Low	530	390

Alat također može prikazati i podatke iz kojeg dijela svijeta se najviše traži upisana ključna riječ ili fraza te grafički prikazati podatke ovisno o vremenu. Uz to se prikazuju i srodni pojmovi koji mogu olakšati odabir ključne riječi ili fraze.



Nakon što se odaberu najbolje ključne riječi potrebno ih je koristiti unutar sadržaja stranice, no ne samo unutar sadržaja već i unutar naslova. Uz naslove i sadržaj ključne riječi je potrebno koristiti i unutar title, meta i alt svojstvima elemenata koji se nalaze unutar stranice.

```
<head>
<meta name="description" content="Stranica za osnove informatike">
<meta name="keywords" content="osnove informatike, računalo">
<meta name="author" content="Test">
</head>
```

Također vrlo je bitna lokacija stranice u odnosu na korijenski tj. root direktorij domene. Tražilice sadržaj koji je bliže korijenskom direktoriju domene smatraju važnijim od onoga koji se nalazi dalje tako da recimo www.domena.com/vijesti/2013/05/05/naslov-clanka.html manje važan od www.domena.com/vijesti/naslov-clanka.html.

Dobra praksa SEO-a:

- Naslov web stranice - 10-70 znakova (preporučeno: do 7-15 riječi)
- Opis web stranice - do 200 znakova /preporučeno do 160)
- Ključne riječi po web stranici - do 10 riječi
- Ponavljanje ključnih riječi u sadržaju (tekstu stranice)- 5-20% za sve ključne riječi
- Pojedine ključne riječi - 1-6% pojedina ključna riječ
- Broj linkova na pojedinoj web stranici - manji od 100 (Google preporuka)
- Sitemap/Mapa stranice - do 100 linkova (ako je veći, razdijeliti na nekoliko dijelova)
- Url adresa stranica - dobro da je do 100 znakova

Loša praksa SEO-a:

- Tekst u slici/grafici - tekstovi u grafikama na internet stranicama nisu čitljivi web tražilicama
- Razmjena linkova sa farmama linkova i plaćanje ranga - "loše susjedstvo"
- Over optimization penalty (OOP) - prekomjerna "optimizacija" - penali za prečesto ponavljanje ključnih riječi i izraza u raznim dijelovima koda i tekstova
- Preusmjeravanje stranice putem meta tagova - npr. zakup nekoliko domena koje se preusmjeravaju na jednu stranicu
- Krađa tekstova i slika sa drugih internet stranica - Google to ne voli i kažnjava
- Dinamične internet stranice - stranice kojima se adresa mijenja
- Korištenje skrivenih likova i automatskog preusmjeravanja
- Flash internet stranice (animirane stranice) - Google i druge tražilice još uvijek ne mogu čitati sadržaj koji se nalazi u Flash stranicama