Emmet

Većina IDE razvojnih okruženja dozvoljava programeru da napiše kôd koji učestalo koristi i da ga sačuva kao šablon. Za manje blokove kôda ti šabloni se zovu „snippets“. Problem sa ovakvim načinom rada je da iako ubrzava pisanje kôda, šablon nije dinamički promenljiv, odnosno za poziv snippet-a dobija se uvek isti kôd čija varijacija mora da se menja.

Ideja iza Emmet-a je da se programeru omogući da korišćenjem CSS-olike sintakse za selektore kreira izraze koji prilikom pritiska dirke ili kombinacije dirki generiše HTML ili CSS kôd.

Emmet čita uneti izlaz sve dok ne dođe do razmaka ili novog reda. Da biste napisali bilo koji element dovoljno je da unesete ime elementa. Za unetu reč div dobija se div → , za uneti karakter p dobija se p → . Emmet nema ograničenja za tagove tako da je moguće ukucati i sopstvene tagove. Za unetu reč foo dobija se foo → <foo></foo>.

Operatori ugnježđavanja sadržaja

Emmet podržava sledeće operatore za ugnježđavanje sadržaja:

- Operator za dete (>)
- Operator za susedni (sledeći) element (+)
- Operator za prethodni nivo (^) – ovaj operator može da se koristi i više puta u zavisnosti koliko nivoa iznad želite da kreirate element.
- Operator za ponavljanje (*)
- Operator za grupisanje (())

Pogledajmo operatore u akciji:

Izraz **div>ul>li** generisaće sledeći kôd:

```
<div>
    <ul>
        <li></li>
    </ul>
</div>
```

Izraz **div+p+bq** generisaće sledeći kôd:

```
<div></div>
<p></p>
<blockquote></blockquote>
```

Izraz **div+div>p>span+em^bq** generisaće sledeći kôd:

```
<div></div>
<div>
    <p><span></span><em></em></p>
    <blockquote></blockquote>
</div>
```

Izraz **div+div>p>span+em^^^bq** generisaće sledeći kôd:

```
<div></div>
<div>
```

```
3.          <p><span></span><em></em></p>
4.      </div>
5.      <blockquote></blockquote>
```

Izraz ul>li*3 generisaće sledeći kôd:

```
1.      <ul>
2.          <li></li>
3.          <li></li>
4.          <li></li>
5.      </ul>
```

Identifikatori, klasni selektori, atributi i tekst

Za pisanje identifikatora možete koristiti sintaksu **#naziv** ili **element#naziv**. Ukoliko ne navedente element, podrazumevani element koji se generiše je div element. Za pisanje elemenata sa klasnim selektorom možete koristiti sintaksu **.naziv** ili **element.naziv**. Princip generisanje je isti kao kod identifikatora.

Za definisanje atributa elementu koristi se sintaksa **element[atribut=vrednost]** ili **element[atribut="vrednost"]** ukoliko je vrednost atributa tekstualna. Da biste definisali tekst nekog elementa morate tekst navesti unutar simbola **{}**. Nije neophodno stavljati tekst pod navodnike za tekstualni prikaz. Operator **$** koristi se sa operatorom * i na njegovoj poziciji ispisuje se vrednost iteratora. Ukoliko ispred operatore stoji **@-** numerički rednosled je obrnut.

Ukoliko želite da ispišete lorem ipsum tekst potrebno je da unesete reč **lorem**.

Izraz **ul>(li.menuitem>a[href="#"]{Link $})*3** generisaće sledeći kôd:

```
6.  <ul>
7.      <li class="menuitem"><a href="#">Link 1</a></li>
8.      <li class="menuitem"><a href="#">Link 2</a></li>
9.      <li class="menuitem"><a href="#">Link 3</a></li>
10. </ul>
```

Emmet je podržan na svim većim razvojnim okruženjima i za željeno okruženje možete ga preuzeti sa stranice http://www.emmet.io/.

HTML

World Wide Web (WWW)

World Wide Web (Web) je bio ključan za razvoj informacionog doba i primarni je način za interakciju između korisnika širom sveta. Web je svetska mreža računara koja predstavlja izvor digitalnih informacija (resource) predstavljenih uopšteno kao WEB dokumenti (HTML, XML, tekst, slika, zvuk, video, program).

Izvor informacije na WEB-u ima jednoznačnu adresu URI-a (Universal Resource Identifier) koja se sastoji od tri dela (scheme://host.domain:port/path/filename):

- protokola pristupa izvoru (npr. http, https, ftp,...),
- imena sajta na kom je izvor informacije (npr. www.skola.edu.rs)
- staze do resursa (path, npr: studijskiprogaram).

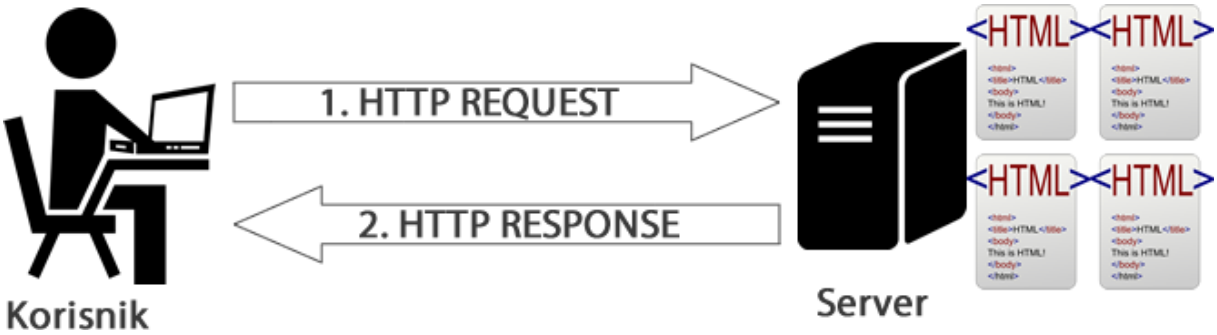
Kombinacijom svih delova dobija se putanja (adresa) http://www.skola.edu.rs/studijskiprogram.html

Sve web stranice su primarno tekstualni dokumenti koji su struktuirani korišćenjem Hypertext Markup Language (HTML). Pored tekstualnog sadržaja stranice, stranica može da sadrži i slike, video, audio, i multimedijalni sadržaj. Ranije su podržavane tehnologije kao što su Flash i Java Applets, ali je zbog bezbednosti kao i politike giganata u svetu interneta sve više akcenat da se funkcionalnosti ovih tehnologija definišu HTML i Javascript standardima. Kolekcije HTML stranica koje su međusobno povezane hyperlink-ovima nazivaju se web sajtovi ili web prezentacije. Websajtovi sa naprednim funkcionalnostima (ažuriranje sadržaja, upravljanje sadržaja, servisi...) nazivaju se web aplikacije.

U opštem slučaju se za pristup izvoru informacija na mreži se koriste: jedinstvena šema imena za lociranje izvora na mreži (protokol, ime sajta, staza do resursa), navedeni protokol (skup pravila za uspostavu, održavanje i raskidanje veze kao i oporavka u slučaju otkaza) hipertekst (Hypertext) za jednostavnu navigaciju između izvora informacija, kao što je na primer HTML (HyperText Markup Language).

Komunikacija između browser-a i web sajta

Komunikacija kada se unese zahtev za posetu određenom statičkom sajtu dat je na slici 1. Web sajtovi mogu biti statički i dinamički. Statički sajtovi imaju fiksnu strukturu koja jedino može da se menja izmenom izvornog kôda same stranice.



Slika 1. Komunikacija između korisnika i servera

Sa druge strane dinamički sajtovi svoj sadržaj dobijaju generisanjem sadržaja. Sam sadržaj može biti generisan korišćenjem argumenata - sadržaj se formira na osnovu unosa korisnika (kalkulatori, kursna lista, konverzije dokumenata...) ili sadržaj stranice se nalazi u tekstualnim dokumentima (txt, yaml, twig...) ili iz baze podataka. Na ovom kursu za dinamički sadržaj biće korišćen JSON koji će se nalaziti u tekstualnim datotekama.

Na slici 1 posmatramo komunikaciju koja se odvija putem TCP/IP protokola. Korisnik šalje HTTP Request, odnosno unosi adresu stranice koju želi da poseti (npr. www.skola.edu.rs/kontakt.html) a server zatim traži dokument koji se nalazi na traženoj lokaciji i sa HTTP Response vraća korisniku HTTP Response koji u sebi sadrži struktuirane digitalne informacije.

Za publikaciju digitalne informacije potreban je mehanizam koji je platformski nezavisan u hardverskom smislu i u smislu operativnog sistema. Ovaj problem je rešen definisanjem opisnog (markup) jezika (HTML-a) kao hipertekstualnog standarda za browser-e koji prerasta u xHTML (HTML+XML) a sada u HTML5. HTML5 je markerski jezik koji se koristi za strukturiranje i prezentovanje sadržaja za World Wide Web. To je peta verzija HTML standarda nastala 2012. godine kao preporuka od strane W3C. Osnovna zamisao je unapređenje podrške za multimedijalni sadržaj uz očuvanje lakoće korišćenja kako za program (veb čitač, parser) tako i za čoveka. HTML5 stranice imaju ekstenziju .html (treba koristiti ovu ekstenziju), a nalaze se u određenom direktorijumu servera vezanog na Internet, što ih čini dostupnim na web-u. Pomoću HTML5 jezika se generišu dokumenti tipa hipertekst.

HTML dokumenti

HTML5 je ovog trenutka (2015) poslednji HTML standard. Ovim je HTML dobio na funkcionalnosti, obzirom da su prethodne verzije koristile pomoć drugih markerskih ili skript jezika, što su ujedno bili njihovi glavni nedostaci. Dizajniran je da prikaže veb sadržaj bez eventualnih dodataka (plugins). Nezavisan je od platforme i predviđen je za korišćenje na svim uređajima. Fokusiran je više na strukturu nego na izgled strane i sadrži informacije i komande.

HTML5 komande se pišu u vidu TAG-ova unutar oznaka manje (<) i veće (>) i koji utiču na prikaz elemenata i podataka koji su unutar opsega taga. Tag je komanda koja govori browseru na koji način da prikaže sadržaj stranice odnosno elementa na koga taj tag utiče.

Na početku svakog HTML5 dokumenta nalazi se tag definicije tipa dokumenta **<!DOCUMENT html>** koji daje informaciju browser-u da fajl predstavlja HTML5 dokument te da ga tako i tretira i prikaže. Tagovi <html> i </html> uokviruju html dokument tako da predstavljaju respektivno početak i kraj HTML dokumenta. Skoro svi tagovi u HTML5 imaju i početni i završni tag (nemaju npr. <meta>, <img>). Između početnog i završnog taga se nalazi sadržaj na koji ovaj tag utiče. Završni tag se dobija dodavanjem znaka / ispred naziva taga i označava mesto na kom prestaje dejstvo tog taga. Prosti tagovi (nekontejnerski tagovi) služe za opisivanje jednostavnih elemenata logičke strukture dokumenta. Oblika su: **<tag>**. Složeni tagovi (kontejnerski tagovi) su oblika **<tag>** ***sadržaj*** **</tag>** kojima je opisan izgled elemenata koji se nalaze između oznaka za početak i kraj datog taga. Moguće je koristiti atribute u okviru početnog taga <tag a1=a a2=b ...> y </tag> koji pružaju dodatne informacije o uticaju taga na elemente ograđene tagom.

Svi HTML dokumenti sastoje se iz dva primarna odeljka: zaglavlje strane (ograničeno je tagovima <head> i </head>) - odeljak zaglavlja strane čini sadržaj koji se ne prikazuje na stanici (ne vidi ga korisnik) i telo strane (ograničeno tagovima <body> i </body> - sav sadržaj na stranici, odnosno, ono što se vidi u prozoru browser-u i na šta se deluje nalazi se u telu HTML5 dokumenta. Elementi <html>,<head> i <body> nisu obavezni, ali ih je dobro koristiti zbog jasnoće koda.

Sledi minimalni preporučeni HTML5 kod:

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta charset="UTF-8">
5.         <title>Naslov dokumenta</title>
6.     </head>
7. <body>
8.         Sadržaj dokumenta...
9. </body>
10.</html>
```

U zaglavlju se mogu navoditi meta tag <meta> koji nema završni tag. Služi za opis dokumenta (ključne reči, sadržaj, ponašanje kao npr. Osvežavanje stranice itd).

Meta podaci nemaju više važnost koju su imali pre nekoliko godina, ali i dalje postoje neki aktuelni tagovi koji su obavezni zbog SEO optimizacije. Tagovi su keywords, description, i author. U zaglavlju takođe treba da bude i title tag za naslov stranice. Tagovi koji mogu da stoje u zaglavlju dati su u nastavku:

Tag	Opis
<title>	Definiše naslov dokumenta
<base>	Definiše podrazumevanu adresu ili odredište za sve linkove na stranici.
<link>	Definiše relaciju između dokumenta i eksternog izvora.
<meta>	Definiše meta-podatke o HTML dokumentu
<script>	Definiše skript na klijentskoj strani
<style>	Definiše informaciju o stilu za dokument

Body HTML elementi

Za vizuelni prikaz postoji veliki broj HTML elemenata. Na Mozilla Developer stranici[1] nalazi se lista svih elemenata kao i lista elemeneta koji je preporučeno da se više ne koriste. U sledećim tabelama grupisani su HTML5 elementi po svojim funkcionalnostima:

Grupisanje sadržaja

Tag	Opis
<address>	Definiše kontakt informacije.
<article>	Definiše nezavisnu celinu u dokumentu, stranici, čija se struktura može ponovo koristiti (vesti za blog, članci za kuvanje...). Preporučuje da svaki article ima svoje zaglavlje (h1... h6).
<aside>	Definiše sekciju stranice koja dopunjuje prezentovani sadržaj, ali stoji kao izdvojena celina (meni za linkovanje u dokumentu,

[1] Mozilla Developers – List of active and inactive elements in HTML5 - https://developer.mozilla.org/en/docs/Web/HTML/Element

	reference, izdvojen sadržaj...)
	Definiše nepreciziran kontejner za elemente.
<footer>	Definiše kontejner za elemente podnožija stranice.
<header>	Definiše kontejner za elemente zaglavlja stranice.
<hgroup>	Grupiše elemente koji predstavljaju grupu zaglavlja dokumenta ili sekcije (npr. Naslov i apstrakt, naslov i autori..)
<main>	Definiše glavni deo <body> elementa ili aplikacije.
<nav>	Definiše kontejner za navigaciju stranice ili sekcije
<section>	Definiše određenu sekciju dokumenta.

Prikaz tekstualnog sadržaja

Tag	Opis
<h1>...<h6>	Definiše sadržaj zaglavlje sekcije. Elementi sa oznakom h1 imaju najveći prioritet, dok h6 imaju najmanji.
<dd>	Definiše opis u definicionoj listi.
<dl>	Definiše kontejner u definicionoj listi.
<dt>	Definiše pojam u definicionoj listi.
<figcaption>	Definiše opis ili legendu koja je povezana sa figurom. Element je opcioni i koristi se sa <figure> tagom.
<figure>	Definiše sadržaj koji može biti slika, ilustracija, dijagram ili izdvojeni kôd. Obično sadrži i opis definisan <figcaption> elementom.
<hr>	Definiše tematsku podelu unutar članka i vizuelno se prikazuje kao horizontalna linija.
	Definiše element uređene ili neuređene liste
	Defniše uređenu listu
<p>	Definiše paragraf
<pre>	Defniše preformatirani tekst. Obično se sadržaj prikazuje korišćenjem monospace fonta.
	Definiše neuređenu listu.

Izmene sadržaja unutar grupacije

Tag	Opis
<a>	Definiše link
<abbr>	Definiše opis skraćenice. Daje korisne informacije browser-ima, sistemima za prevođenje i pretraživačima (search-engines).
<b>	Definiše boldovani tekst.
<bdi>	Definiše tekst koji ne podleže promeni <bdo> elementa.
<bdo>	Omogućuje bidirekcioni ispis
<blockquote>	Citiranje bloka teksta (obično browser uradi indent ovog bloka)
 	Označava prelaz unovi red
<cite>	Definiše naslov rada (u italiku)
<code>	Omogućuje pisanje teksta koji predstavlja programski kod. Ovaj tag ne čuva beline u kodu.
<dfn>	Definiše definiciju termina ili skraćenice. Ako se koristi atribut title onda on predstavlja definiciju termina.
	Definiše tekst koji treba biti istaknut. Prikazuje se kao italic tekst.
<i>	Definiše italic tekst.
<kbd>	Definiše font tastaturnog ulaza
<mark>	Definiše označeni tekst.
<q>	Postavljanje teksta pod znakove navoda.
<s>	Definiše precrtani tekst.

<samp>	Definiše font računarskog izlaza.
<small>	Smanjuje tekst za jednu veličitnu manje u odnosu na veličinu teksta sekcije ili dokumenta (large->medium, medium->small...)
	Definiše generički kontejner za elemente u nekoj selekciji bez njihovog odvajanja od sadržaja kom pripadaju.
	Daje važnost tekstu. Obično se prikazuje kao boldovani tekst.
<sub>	Definiše tekst koji treba zbog tipografskih razloga da bude spušten u odnosu regularni tekst.
<sup>	Definiše tekst koji treba zbog tipografskih razloga da bude podignut u odnosu regularni tekst.
<time>	Definiše dvadesetčetvoročasovno vreme.
<u>	Definiše podvučeni tekst.
<var>	Definiše promenljivu za matematičke ili programske izraze.
<wbr>	Definiše poziciju gde browser može da prelomi opciono tekst u zavisnosti od veličine ekrana ili ostatka teksta na stranici.

Tabele

Tag	Opis
<caption>	Definiše naslov tabele
<col>	Definiše kolonu tabele
<colgroup>	Definiše kontejner kolona tabele
<table>	Definiše tabelu
<tbody>	Definiše telo (sadržaj) tabele. Obično mu prethodi <thead> sa zaglavljima kolona.
<td>	Definiše kolonu ćelije
<tfoot>	Definiše podnožije tabele
<th>	Definiše sadržaj zaglavlja tabele
<thead>	Definiše kontejner zaglavlja tabele
<tr>	Definiše red tabele

Forma

Tag	Opis
<button>	Definiše dugme
<datalist>	Definiše listu <option> elemenata koji predstavljaju vrednosti dostupne za druge kontrole (elemente forme).
<fieldset>	Definiše kontejner za kontrole i elemente.
<form>	Definiše formu.
<input>	Definiše polja za unos korisnika. U zavisnosti od atributa „type“ postoje različite vrste unosa.
<label>	Definiše labelu koja ide uz polje za unos.
<legend>	Definiše naslov za <fieldset> element.
<meter>	Definiše prikaz mere unutar poznatog opsega (npr. iskorišćenost diska)
<optgroup>	Definiše grupe opcija za <select> element.
<option>	Definiše opciju u elementima <datalist> i <select>.
<output>	Definiše izlaz koji je nastao kao ishod računa korisnika.
<progress>	Definiše vizuelni prikaz odrađenog procesa. Različito se prikazuje u svakom browser-u.
<select>	Definiše padajuću listu ili listu sa višestrukim izborom. Ukoliko stavite atribut multiple, dobija se lista sa višestrukim izborom.
<textarea>	Definiše polje za višelinijski unos teksta.

Linkovi

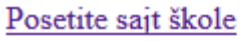
Linkovi imaju višestruku primenu. Služe da preusmere korisnika na drugu stranicu, za preuzimanje multimedijalnog sadržaja, za pozicioniranje korisnika na tekućoj stranici itd.

Tag za link je <a> i njegov zatvarajući tag . Obavezni atribut je href (hyperlink reference) koji definiše ***lokaciju odnosno putanju*** na koju će korisnik biti preusmeren ili fajl koji korisnik treba da preuzme. Između otvarajućeg linka se nalazi sadržaj koji je linkovan, odnosno element čijim će klikom biti izvršena aktivnost preusmeravanja / preuzimanja.

Primer linka izgleda:

```
1. <a href="http://www.viser.edu.rs/">Posetite sajt škole</a>
```

Kada otvorite stranicu dobićete izgled koji je prikazan na slici 2.



Slika 2. Vizuelni prikaz primera za link

Drugi atribut koji možete dodati je *target* atribut. Target atribut govori browser-u gde da otvori novi link. Uz napredak tehnologija dve vrednosti su još uvek aktuelne - _blank koja otvara link u novom prozoru / tabu (zavisi od browsera) i _self koja je podrazumevana i otvara link u aktivnom prozoru.

Ukoliko želite da link ne vodi ni na jednu lokaciju, umesto linka u href atributu unesite znak #. Praktična primena je kada pišete navigaciju ali još uvek ne znate lokacije vaših stranica, ili će u aplikaciji Javascript biti primenjen za interakciju sa korisnikom kada on klikne na link.

Linkovi ne moraju da vode na spoljne stranice (da u sebi imaju http://) nego mogu voditi i na lokalne stranice. U sledećem primeru biće pojašnjeno linkovanje po relativnim adresama.

Kada se otvaraju html stranice na računaru, u address bar-u je dat link kao file:///C:/Users/korisnik/Desktop/stranica.html

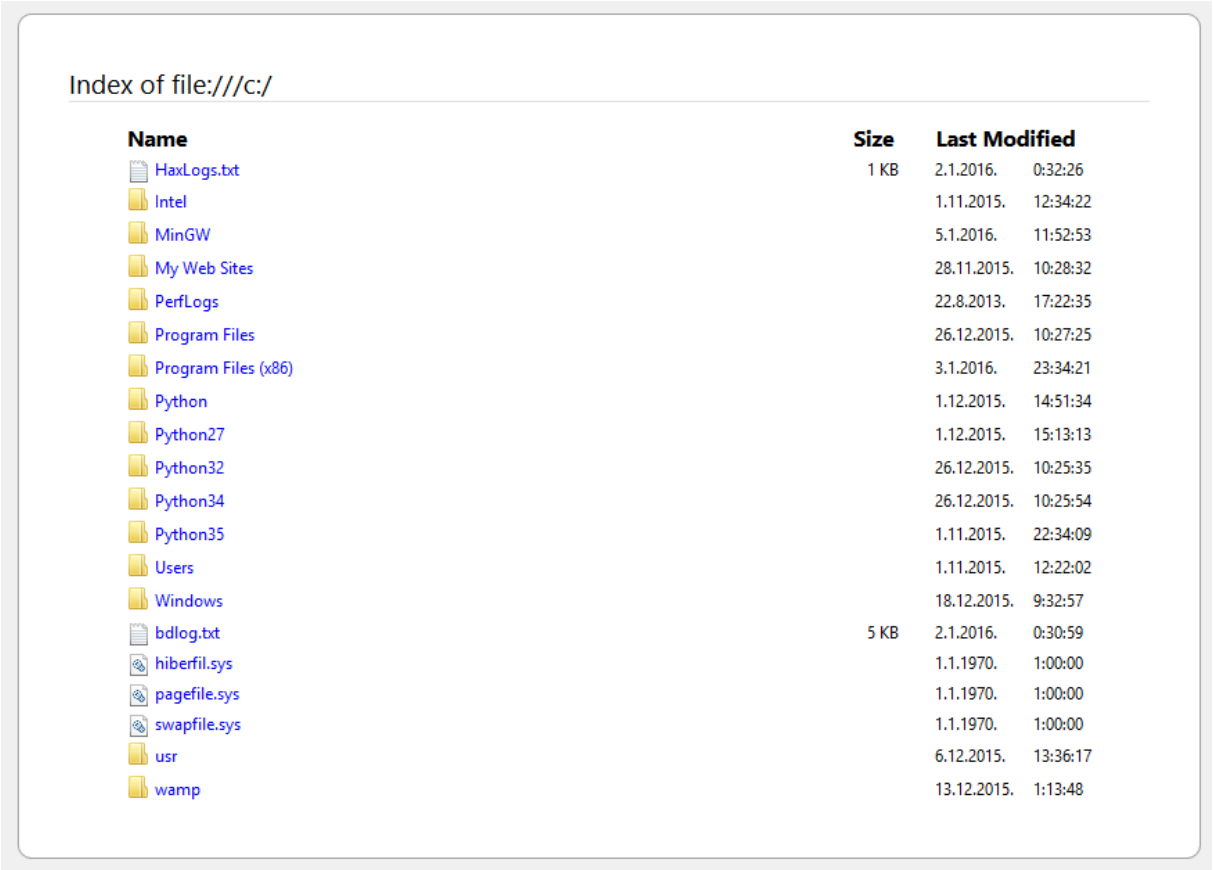
Ovakvi linkovi kao i linkovi http://www.viser.edu.rs/ nazivaju se absolutne putanje, zato što se od linka ne očekuje da promeni svoju lokaciju. Verovatno vam se događalo da otvorite neki članak i autor članka se poziva na neki link na koji kada ste kliknuli vodi na stranicu koja više ne postoji, bilo ona na istoj adresi kao dati sajt ili na drugoj adresi. Ove adrese su absolutne i ako se dokument prebaci sa date lokacije, taj link može da ne radi (postoje tehnike slanja korisnika na neku drugu stranicu sa objašnjenjem da link ne postoji i da može neki sličan link izabrati u ponuđenim opcijama).

Relativni linkovi za razliku od absolutnih podrazumevaju da su dokumenti kojima se pristupa na istom računaru. Sajt će imati uvek glavni (root) direktorijum. To je direktorijum kome pristupaju korisnici kada posećuju sajt. Taj direktorijum mora imati index stranicu. Ona može biti različitih ekstenzija (index.html, index.php, index.jsp, ...).

Kada unesete link http://www.viser.edu.rs/ vi zapravo unosite skraćeno http://www.viser.edu.rs/index.php

Kada direktorijum nema index stranicu, web server sam generiše indeks stranicu koja prikazuje sadržaj.

Kada u Firefox browseru unesete absolutnu putanju kao C:\ možete dobiti strukturu vaše lokalnog diska (slika 3). Link će biti pretvoren u file:///c:/ kako bi browser mogao strukturu da interpretira kao stranice.



Slika 3. Firefox prikaz absolutne putanje kao strukture sajta

Slike

Slike nisu fizički definisane kao podaci u dokumentu već su posebni tagovi koji vezuju lokaciju slike i govore browseru da treba da prikaže vizuelnu sliku sa datog linka. Tag za slike je <img> koji je samozatvarajući tag i u okviru njega je obavezni atribut src (source) koji definiše link ka slici. Link može kao i kod linkova da bude absolutni (sa nekog sajta) ili relativni (u okviru samog sajta).

Primer tag slike sa relativnim linkom:

Primer taga slike sa absolutnim linkom:

Problem sa linkovima iznad je da ako se slika zove „aaa.jpg“ search engine botovi ne znaju šta predstavlja data slika. Ukoliko želite da vam se bolje rangira sajt, sa atributom alt search engine botovima dajete opis šta se nalazi na slici. Takođe funkcija alt atributa je da prikaže tekst ukoliko slika ne postoji na datoj putanji.

Primer taga slike sa alt atributom

Umesto linka kao izvora slike moguće prikazati slike korišćenjem byte kôda. U tom slučaju u atributu source treba da stoji src="**data:image/jpg;base64**,*byte array*">. Tip slike može biti i bilo kog drugog formata image/png, image/gif itd. U zavisnosti od tipa zavisiće i kvalitet slike koja se prikazuje.

Liste

Liste predstavljaju liste elementa označene nekom vrstom buletina (nenabrojive) ili brojeva (nabrojive). Mogu biti uređenje ili neuređene. U zavisnosti od tipa liste koristimo tagove <ol> (ordered list) za uređenu listu, odnosno (unordered list) za neuređenu. Imaju svoje zatvarajuće tagove. U okviru sadržaja nalaze se elementi liste. Svaki element se nalazi unutar <li> (list item) taga. elementi mogu imati tekst ili druge HTML elemente.

Primer jedne neuređene liste dat je kroz sledeći kôd:

```
1.   <ul>
2.       <li>Unordered List Item 1</li>
3.       <li>Unordered List Item 2</li>
4.       <li><h3>Header as List Item</h3></li>
5.   </ul>
```

Prikaz je dat na slici 4.

- Unordered List Item 1
- Unordered List Item 2

- **Header as List Item**

Slika 4. Prikaz primera neuređene liste

Primer uređene liste dat je kroz sledeći kod:

```
1.   <ol>
2.       <li>Ordered List Item 1</li>
3.       <li>Ordered List Item 2</li>
4.       <li>
5.           <ul>
6.               <li>Unordered List Item 1</li>
7.               <li>Unordered List Item 2</li>
8.               <li>Unordered List Item 3</li>
9.           </ul>
10.      </li>
11.  </ol>
```

Vizuelni prikaz dat je na slici 5. Možete videti da liste mogu da sadrže i druge liste u sebi, i mogu biti numeričke ili buletin liste.

1. Ordered List Item 1
2. Ordered List Item 2
3.
    - Unordered List Item 1
    - Unordered List Item 2
    - Unordered List Item 3

Slika 5. Prikaz primera uređene liste sa elementom neuređene liste

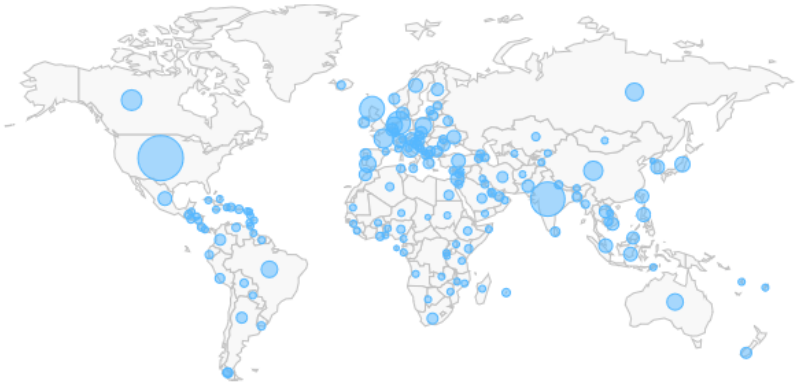
Od HTML5 standarda ne postoje atributi za način prikaza buletina listi. Za definisanje prikaza koristi se CSS o kome će biti reči u sledećem poglavlju.

Napisati kôd za novinski članak koji je dat na slici.

JavaScript Popularity Surpasses Java, PHP in the Stack Overflow Developer Survey

18 Mar 2016 9:53am, by David Cassel

JavaScript, primarily known as a front-end language for Web applications, is quickly surpassing the popularity of not only other Web languages such as PHP, but even traditional back-end programming languages such as Java, at least according to the results of the latest annual survey from Stack Overflow, the popular question-and-answer site for developers and weekend hackers.



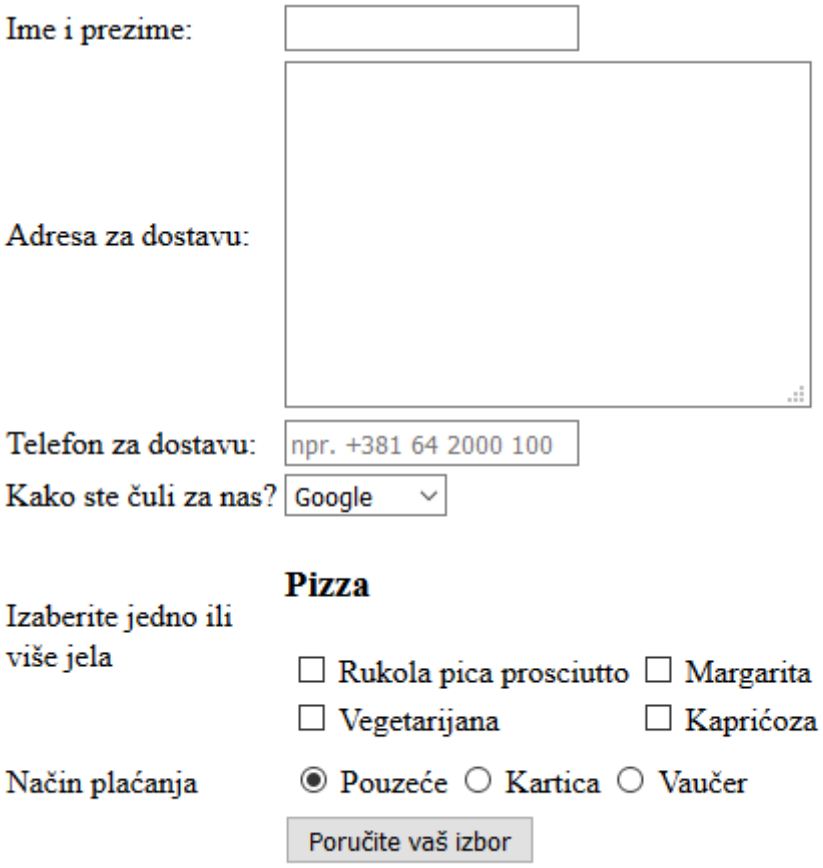
Described as *"the most comprehensive developer survey ever conducted,"* the Stack Overflow survey tallied the responses from 56,033 developers in 173 countries.

HTML kôd:

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.      <meta charset="UTF-8">
5.      <title>JavaScript Popularity Surpasses Java, PHP in the Stack Overflow
    Developer Survey</title>
6.  </head>
7.  <body>
8.      <h1>JavaScript Popularity Surpasses Java, PHP in the Stack Overflow
    Developer Survey</h1>
9.      <p>18 Mar 2016 9:53am, by <a href="#">David Cassel</a></p>
```

```
    <p>JavaScript, primarily known as a front-end language for Web applications, is quickly surpassing the popularity of not only other Web languages such as PHP, but even traditional back-end programming languages such as Java, at least according to the results of the latest annual survey from Stack Overflow, the popular question-and-answer site for developers and weekend hackers.</p>
    <p><img src="http://thenewstack.io/wp-content/uploads/2016/03/Stack-Overflow-developer-locations-2016.png"></p>
    <p>Described as <i>“the most comprehensive developer survey ever conducted,”</i> the Stack Overflow survey tallied the responses from 56,033 developers in 173 countries.</p>
</body>
</html>
```

Napisati kôd za formu za naručivanje proizvoda korišćenjem tabela i elemenata forme koja je data na sledećoj slici.



HTML kôd:

```
<!DOCTYPE html>
<html lang="rs">
<head>
    <meta charset="UTF-8">
    <title>Narudžbenica</title>
</head>
<body>
```

```
    <table>
        <tr>
            <td>Ime i prezime:</td>
            <td><input type="text"></td>
        </tr>
        <tr>
            <td>Adresa za dostavu:</td>
            <td><textarea cols="30" rows="10"></textarea></td>
        </tr>
        <tr>
            <td>Telefon za dostavu:</td>
            <td><input type="tel" placeholder="npr. +381 64 2000 100"></td>
        </tr>
        <tr>
            <td>Kako ste čuli za nas?</td>
            <td>
                <select>
                    <option value="Google">Google</option>
                    <option value="Facebook">Facebook</option>
                    <option value="Twitter">Twitter</option>
                </select>
            </td>
        </tr>
        <tr>
            <td>Izaberite jedno ili<br>više jela</td>
            <td>
                <h3>Pizza</h3>
                <table>
                    <tr>
                        <td><input type="checkbox"> Rukola pica
prosciutto</td>
                        <td><input type="checkbox"> Margarita</td>
                    </tr>
                    <tr>
                        <td><input type="checkbox"> Vegetarijana</td>
                        <td><input type="checkbox"> Kapričoza</td>
                    </tr>
                </table>
            </td>
        </tr>
        <tr>
            <td>Način plaćanja</td>
            <td>
                <table>
                    <tr>
                        <td><input name="placanje" type="radio" checked>
Pouzeće</td>
                        <td><input name="placanje" type="radio">
Kartica</td>
```

```
                        <td><input name="placanje" type="radio"> Vaučer</td>
                    </tr>
                </table>
            </td>
        </tr>
        <tr>
            <td></td>
            <td><button>Poručite vaš izbor</button></td>
        </tr>
    </table>
</body>
</html>
```

Emmet

Većina IDE razvojnih okruženja dozvoljava programeru da napiše kôd koji učestalo koristi i da ga sačuva kao šablon. Za manje blokove kôda ti šabloni se zovu „snippets“. Problem sa ovakvim načinom rada je da iako ubrzava pisanje kôda, šablon nije dinamički promenljiv, odnosno za poziv snippet-a dobija se uvek isti kôd čija varijacija mora da se menja.

Ideja iza Emmet-a je da se programeru omogući da korišćenjem CSS-olike sintakse za selektore kreira izraze koji prilikom pritiska dirke ili kombinacije dirki generiše HTML ili CSS kôd.

Emmet čita uneti izlaz sve dok ne dođe do razmaka ili novog reda. Da biste napisali bilo koji element dovoljno je da unesete ime elementa. Za unetu reč div dobija se div → , za uneti karakter p dobija se p → . Emmet nema ograničenja za tagove tako da je moguće ukucati i sopstvene tagove. Za unetu reč foo dobija se foo → <foo></foo>.

Operatori ugnježdavanja sadržaja

Emmet podržava sledeće operatore za ugnježdavanje sadržaja:

- Operator za dete (>)
- Operator za susedni (sledeći) element (+)
- Operator za prethodni nivo (^) – ovaj operator može da se koristi i više puta u zavisnosti koliko nivoa iznad želite da kreirate element.
- Operator za ponavljanje (*)
- Operator za grupisanje (())

Pogledajmo operatore u akciji:

Izraz **div>ul>li** generisaće sledeći kôd:

```
<div>
    <ul>
        <li></li>
    </ul>
</div>
```

Izraz **div+p+bq** generisaće sledeći kôd:

```
<div></div>
```

```
2.    <p></p>
3.    <blockquote></blockquote>
```

Izraz **div+div>p>span+em^bq** generisaće sledeći kôd:

```
1.    <div></div>
2.    <div>
3.        <p><span></span><em></em></p>
4.        <blockquote></blockquote>
5.    </div>
```

Izraz **div+div>p>span+em^^^bq** generisaće sledeći kôd:

```
1.    <div></div>
2.    <div>
3.        <p><span></span><em></em></p>
4.    </div>
5.    <blockquote></blockquote>
```

Izraz ul>li*3 generisaće sledeći kôd:

```
1.    <ul>
2.        <li></li>
3.        <li></li>
4.        <li></li>
5.    </ul>
```

Identifikatori, klasni selektori, atributi i tekst

Za pisanje identifikatora možete koristiti sintaksu **#naziv** ili **element#naziv**. Ukoliko ne navedente element, podrazumevani element koji se generiše je div element. Za pisanje elemenata sa klasnim selektorom možete koristiti sintaksu **.naziv** ili **element.naziv**. Princip generisanje je isti kao kod identifikatora.

Za definisanje atributa elementu koristi se sintaksa **element[atribut=vrednost]** ili **element[atribut="vrednost"]** ukoliko je vrednost atributa tekstualna. Da biste definisali tekst nekog elementa morate tekst navesti unutar simbola **{}**. Nije neophodno stavljati tekst pod navodnike za tekstualni prikaz. Operator **$** koristi se sa operatorom * i na njegovoj poziciji ispisuje se vrednost iteratora. Ukoliko ispred operatore stoji **@-** numerički rednosled je obrnut.

Ukoliko želite da ispišete lorem ipsum tekst potrebno je da unesete reč **lorem**.

Izraz **ul>(li.menuitem>a[href="#"]{Link $})*3** generisaće sledeći kôd:

```
6.  <ul>
7.      <li class="menuitem"><a href="#">Link 1</a></li>
8.      <li class="menuitem"><a href="#">Link 2</a></li>
9.      <li class="menuitem"><a href="#">Link 3</a></li>
10. </ul>
```

Emmet je podržan na svim većim razvojnim okruženjima i za željeno okruženje možete ga preuzeti sa stranice http://www.emmet.io/.

Kanvas

Вежба 2

Kanvas element HTML5 je kontejner za grafiku i služi za crtanje grafičkih elemenata. Predstavlja dvodimenzionalnu mrežu piksela čiji je koordinatni početak gornji levi ugao. (0,0). Postoje metode za crtanje različitih oblika kao što su putanje, polja, crtanje teksta, krugova, dodavanje slika. <canvas> je pravougaona oblast unutar HTML5 dokumenta, nema ivicu i sadržaj. U jednom HTML5 fajlu moguće je imati više kanvas elemenata.

Za početak ćemo definisati jedan kanvas element sa atributima width =200px; height=200px; i postavićemo id=“myKanvas“

```
<!DOCTYPE html>
<html lang="en">
<body>
< canvas  id=“myKanvas“ width=“200“ height=“200“ >  </ canvas >
</body>
</html>
```

Pošto kanvas podrazumevano nema ivicu i sadržaj, ovaj kod ništa neće prikazati. Ukoliko želimo da kanvas-u definišemo ivicu to ćemo uraditi korišćenjem STYLE atributa.

```
< canvas id=“myKanvas“ width=“200“ height=“200“ style=“border:1px solid black“>  </ canvas >
```

Crtanje u kanvasu realizuje se pomoću JavaScript-a. Prvo je potrebno pronaći <kanvas> element preko dodeljenog ID-a.

```
var c=document.getElementById("myKanvas");
```

Zatim pozivamo njegovu getContext() metodu i prosleđujemo j string “2d” jer ćemo crtati u 2D okruzenju. Objekat getContext(“2d”) je ugrađen u HTML te sadrži mnoga svojstva i metode za crtanje linija,kruga, teksta, okvira.

```
Var ctx=c.getContext(“2d”)
```

Sada ćemo nacrtati jedan crveni pravougaonik dimenzija 150x75px.

```
ctx.fillStyle=”FF0000”;
ctx.fillRect (0,0,150,75) ;
```

Svojstvo *fillStyle* može biti boja ili gradijent. Podrazumevana vrednost je #000000 (crna boja)

Metoda *fillRect(x,y,width,height)* crta pravougaonik popunjen prethodno definisanim stilom popunjavanja. Prvo je potrebno definisati svojstvo *fillStyle*, a onda nacrtati objekat pomoću metode *fillRect*. Ukoliko *fillStyle* nije definisan, objekat će imati podrazumevanu popunu crne boje. U našem slučaju metoda *fillRect* počev od tačke (0,0) crta pravougaonik širine 150px i visine 75px. Ukoliko želimo da nacrtamo još jedan objekat, potrebno je ponoviti prethodne dve linije koda, izmena *fillStyle* stojstva će se odnositi samo na elemente koji se crtaju nakon definisanja/promene *fillStyle* svojstva.

Metode za crtanje linija:

-moveTo(x,y)
-lineTo(x,y)

moveTo() definiše početnu tačku linije, a *lineTo*() definiše krajnju tačku linije. Ovaj kod ne crta već samo definiše liniju. Za samo iscrtavanje linije koristi se metoda *stroke();*

Primer:

```
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
```

Iz tačne (0,0) definišemo liniju čija je krajnja tačka (200,100), a potom metoda *stroke()* iscrta tu liniju podrazumevnaom crnom bojom. Svojstvo *strokeStyle* definiše boju linije ili ivične linije nekog drugog objekta. Sa *fillStyle* definiše se boja popune objekta, a sa *strokeStyle* ivična linija.

Metode za crtanje kruga:

-beginPath() - govori da će se crtati staza
-arc(x,y,r, početni ugao,krajnji ugao, smer) - x,y tačka je CENTAR kruga

Primer:

```
ctx.beginPath();
ctx.arc(50,50,30,0,2*Math.PI);
ctx.stroke();
```

Ovaj primer crta pun krug sa centrum u tački (50,50) , čiji je poluprečnik 30. Metoda *stroke()* će nacrtati krug bez popune, samo sa ivčnom linijom, a ukoliko dodamo *ctx.fill()* krug će biti popunjen bojom definisan u *fillStyle*.

Videli smo da metoda *fillRect(x,y,širina,visina)* crta popunjen pravougaonik. Postoji i metoda *strokeRect(x,y,širina,visina)* koja crta nepopunjen pravougaonik i metoda *clearRect(x,y,širina,visina)* koja briše pravougaono područje.

Boje za *fillStyle* i *strokeStyle* se mogu zadavati korišćenjem heksadecimalnog koda, korišćenjem unapred definisanog naziva boje (red,green…) ili pomoću RGB komponente.

Gradijenti

Postoje 2 tipa gradijenta, linearni i radijalni.

createLinearGradient(x1,y1, x2,y2);

createRadialGradient(x1,y1,r1, x2,y2,r2)
Kada se definiše gradijent, potrebno je dodati bar 2 graničnik boje pomoću metode: *addColorStop(pozicija,boja). ctx.createLinearGradient* kreira gradijent objekat i smešta ga u promenjivu “grd”.

```
grd=ctx.createLinearGradient(0,0,100,0);
grd.addColorStop(0,"red");
  grd.addColorStop(0.7,"green");
  ctx.fillStyle=grd;
  ctx.fillRect(10,10,100,100);
```

Promenjiva "grd" je gradient objekat**.** Grd ima metodu *addColorStop* kojom smo definisali dve crvenu i zelenu boju. Potom smo za *fillStyle* svojstvo postavili umesto obične boje promenjivu "grd" odnosno postavili smo prethodno definisani gradijent za popune i na kraju obojili pravougaonik dimenzija 100x100 tim gradijentom.

Zadatak: Kreirati radijalni gradijent po istom principu.

Linije- putanje

Primer crtanja otvorene putanje:

ctx.beginPath();
ctx.moveTo(100,100);
ctx.lineTo(200,200);
ctx.lineTo(100,200);
ctx.stroke();

Ukoliko posle poslednje *lineTo()* metode dodamo *ctx.closePath()* zatvorićemo putanju na način da će se pravom linijom spojiti početna i krajnja tačka. Npr. kod crtanja trougla, nacrtamo dve stranice trougla I sa closePath() dobijemo treću stranicu.

Primer crtanja trougla:

ctx.beginPath();
ctx.moveTo(100,100);
ctx.lineTo(200,200);
ctx.lineTo(100,200); -umesto ove linije koda mogli smo da napišemo ctx.closePath();
ctx.stroke();

closePath() zatvara liniju**,** *stroke()* boji ivičnu liniju. U slučaju da je umesto *stroke()* stajalo *fill(),* popuna bi bila obojena definisanom bojom/gradijentom, a putanja bi automatski i bez korišćenja *closePath()* bila zatvorena.

-Linewidth definiše debiljinu linije.
-**LineCap** definiše oblik završetka linije. Isprobati moguće vrednosti butt,round,square.
-**LineJoin** definiše oblik veze između dve linije (round,bevel,miter)

ctx.beginPath();
ctx.lineCap="round"
ctx.moveTo(100,100);
ctx.lineTo(200,200);
ctx.lineTo(100,200);
ctx.stroke();

Tekst

strokeText(tekst,x,y) - ispisuje zadati tekst kao konturu na poziciji (x,y)

fillText(tekst,x,y) - ispisuje zadati tekst sa popunom na poziciji (x,y)

measureText(tekst) -vraća info o veličini teksta koji se prosleđuje

ctx.font=”30px Arial”;
*ctx.textAlign=”start” //*startna pozicija teksta će biti 10,10, u slučaju da je vrednost textAlign=”end”, završna pozicija teksta bi bila 10,10.*//*
ctx.fillText(“popunjen tekst”,10,10);
ctx.strokeText(“Nepopunjen tekst”, 50,50);

Senke

Postoje sledeća svojstva za senke:

-shadowColor -definiše boju senke

-s**hadowOffsetX** - definiše koliko će senka biti udaljena od objekta po x osi (horizontalno)

-shadowOffsetY - definiše koliko će senka biti udaljena od objekta po x osi (vertikalno)

-shadowBlur - zamućenje senke

Transformacije

Metode za transformaciju:

- translate (x,y) –premešta koordinarni početak koji je podrazumevano (0,0)
- rotate (alfa) – rotira kanvas oko koordinatnog početka za datu vrednost u radijanima
- scale (x,y) – skalira sadržaj kanvasa

```
ctx.font="bold 20px verdana";
ctx.fillText("TEST",50,20);
ctx.translate(150,70);

ctx.rotate((Math.PI/180)*45);
ctx.fillText("TEST",0,0);
ctx.rotate((-Math.PI/180)*45);
ctx.translate(70,100);
ctx.scale(2,2);
ctx.fillText("TEST",0,0);
```

Nakon definisanja svojstva “font”, ispisujemo popunjen tekst metodom *fillText* na poziciji (50,20). Nakon toga metoda *translate(150,70)* pomera koordinatni početak za prethodno ispisan tekst kao i za dalja ispisivanja teksta. Metoda *rotate()* rotira tekst za 45 stepeni. *Metoda fillText()* ispisuje tekst na koordinati (0,0) , ali to nije više gornji levi ugao kanvasa, već pozicija do koje smo došli primenom metode *translate()* koja je izmestila koordinatni početak na novu poziciju.

Nakon toga metoda *translate()* opet menja koordinatni početak, ali u odnosu na trenutni koordinatni početak. Metoda *scale()* skalira tekst. Ukoliko želimo simetrično uvećanje, skaliraćemo po x i po y osi jednako npr. scale(2,2), scale(3,3). Svaka transformacija je kumultativna. Ako 2 puta pozovemo scale(2,2) učetvorostručićemo razmeru objekta.

Postoje i metode za snimanje i vraćanje stanja:
-save()
-restore();

```
ctx.font="bold 20px verdana";
ctx.save(); SAČUVAMO POČETNO STANJE TEKSTA!
ctx.fillText("TEST",50,20);
ctx.translate(150,70);
ctx.rotate((Math.PI/180)*45);
ctx.fillText("TEST",0,0);
ctx.rotate((-Math.PI/180)*45);
ctx.translate(70,100);
ctx.scale(4,4);
ctx.restore();  - VRAĆAMO (PONIŠTAVAMO) SVE PRIMENJENE TRANSFORMACIJE
```

Ispisujemo tekst na poziciji (0,20) koja ide od koordinatnog početka a ne od tačke (70,100) jer smo metodom RESTORE() poništili sve transformacije i vratili se na nivo gde smo poslednji put pozvali metodu save();
ctx.fillText("TEST",0,20); - ispisuje tekst za 20px udaljen po y osi od gornjeg levog ugla kanvasa.

save() snima:
transformacije koje su primenjene, vrednosti svojstva stilizovanja (promena boje, fonta i sl).

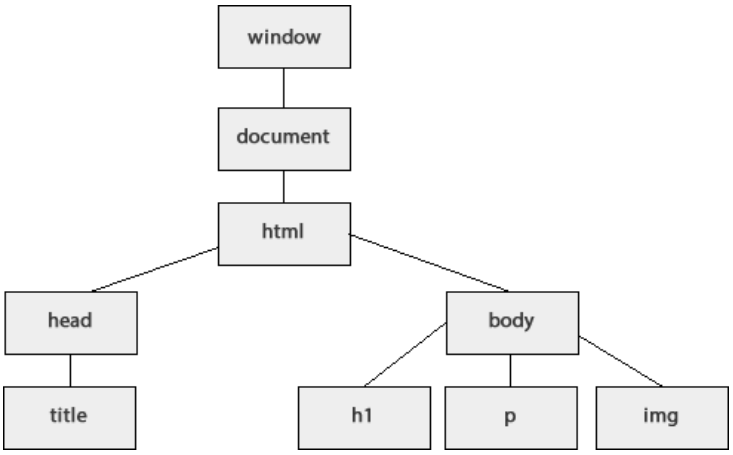
DOM struktura i Canvas

DOM (Document Object Model) struktura

Pre rada sa elementima za grafiku, potrebno je razumeti DOM. DOM je interfejs za HTML i XML dokumenta i definiše strukturu koju mogu menjati programi. Sa DOM-om programeri mogu kreirati dokumente, menjati njihovu strukturu (dodavati, ažurirati i brisati sadržaj). Svemu iz HTML ili XML strukture može pristupiti korišćenjem node-ova.

Node je svaki element HTML/XML strukture. Sam dokument je *document node*. Svi HTML/XML elementi su *element nodes.* Svi HTML/XML atributi su *attribute nodes*. Tekst unutar HTML/XML elemenata je *text node.* U praktikumu objašnjenja i primeri odnosiće se na strukturu HTML elemenata. Svi elementi mogu sadržati druge elemente koji se zovu *child nodes.*

Jezik kojim možemo upravljati strukturom stranice je Javascript (JS). DOM nije programski jezik ali bez njega Javascript ne bi mogao da prepozna kako izgleda struktura elementa. Da bi znali kako do da pristupimo elementima potrebno je da znamo kako izgleda struktura DOM-a. Grafički prikaz dat je na slici 1.



Slika 1. Prikaz strukture DOM-a dokumenta

Glavni element je **window** element. Window element predstavlja prozor browser-a. Ulog window elementa je da detektuje događaje koji su vezani za browser kao što su scroll, focus, load itd. Drugi element je **document** element. Document element predstavlja strukturu HTML dokumenta i služi za pristupanje drugim node elementima. Window i document su definisani kao posebne globalne promenljive u Javascript-u. Ostalim node-ovima pristupa se korišćenjem **document** elementa.

Svi node elementi u Javascript-u posmatraju se kao objekti i imaju u sebi ugrađene događaje i metode koji se mogu modifikovati.

Rad sa HTML elementima

Da biste dohvatili elemente postoji nekoliko načina da to uradite:

- Dohvatanje elementa po identifikatoru (atribut id)
- Dohvatanje elemenata po nazivu taga
- Dohvatanje elemenata korišćenjem klasnog selektora (atribut class)

- Dohvatanje elemenata korišćenjem CSS selektora
- Dohvatanje elemenata korišćenjem HTML kolekcija objekata

Dohvatanje elementa po identifikatoru

Kreirate HTML stranicu „index.html“. Na stranici u <body> tagu unesite sledeći kôd:

```
1.    <p id="paragraf">Ovo je tekstualni node</p>
```

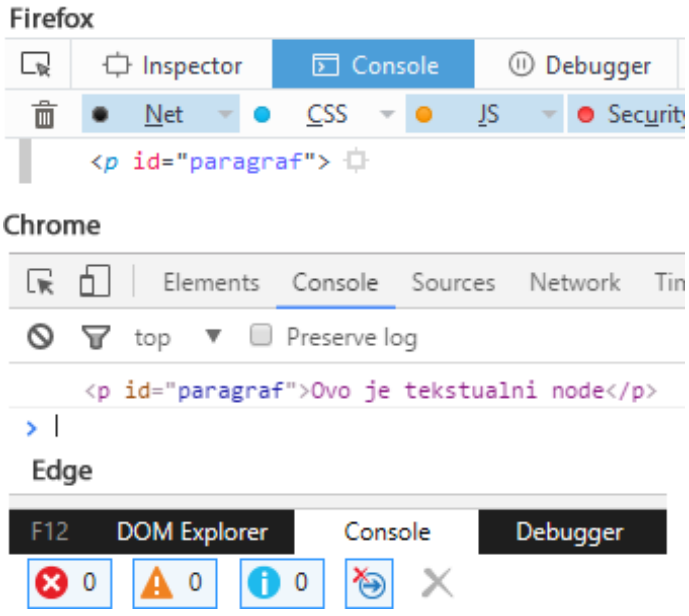
Ispod elementa napisaćemo JS kod koji nam dozvoljava da vidimo kako izgleda node element. Najlakši način za dohvatanje elementa je po njegovom identifikatoru. Document node sadrži sve ostale elemente kao i metode za njihove dohvatanje. Jedna od metoda je **getElementById**. Unesite sledeći kôd ispod paragrafa:

```
1.  <script>
2.      var paragraf = document.getElementById("paragraf");
3.      console.log(paragraf);
4.  </script>
```

Na liniji 2 deklarisana je promenljiva „paragraf“ kojoj će biti dodeljen node element koji vraća metoda getElementById.

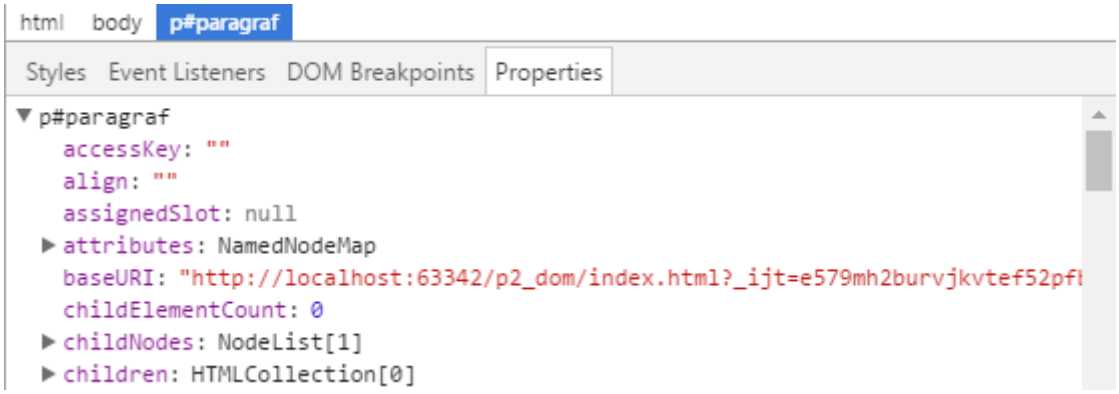
Na liniji 3 nalazi se naredba console.log(). Ova naredba nije deo standardne JS sintakse već je vezana za browser. Većina modernih browser-a ima razvojne alate koji pomažu web programerima da debug-uju svoj kôd.

U Mozilla Firefox-u potrebno je da kliknete desnim klikom miša i izaberete opciju „Inspect Element“, u Google Chorome potrebno je da kliknete desnim klikom miša i izaberete opciju „Inspect“ ili na tastaturi pritisnete kombinaciju Ctrl + Shift + I, dok u Microsoft Edge iz padajućeg menija izaberete Developer Tools ili pritisnete taster F12. Prikaz konzolnog elementa dat je na slici 2.



Slika 2. Console tabovi razvojnih alata popularnih browser-a

U Google Chrome razvojnom alatu u tabu Elements, kada izeberemo neki element videćemo da postoji i nova sekcija sa dodatnim tabulatorima. Jedan od njih je „Properties“. Za element koji ste kreirali na stranici možete videti svojstva i metode koje sadrži (slika 3).



Slika 3. Svojstva paragraf node elementa

Svaki element sadrži dosta svojstava i metoda, od kojih ćemo pomenuti najvažnije stribute.

- Attributes – svojstvo koje je lista node attribute elementa.
- children – lista HTML node elemenata koje element sadrži
- className – lista naziva klasa koje su dodeljene elementu
- id – identifikator elementa
- innerHTML – HTML sadržaj elementa
- style – lista CSS svojstava koje možemo menjati

Metode koje element ima počinju sa on a zatim u nazivu imaju akciju na koju će se izvršiti. Svi događaji su JS funkcije. Neki od događaja su onclick, onhover, onkeyup, onkeydown, onfocus itd.

Dohvatanje elemenata po nazivu taga

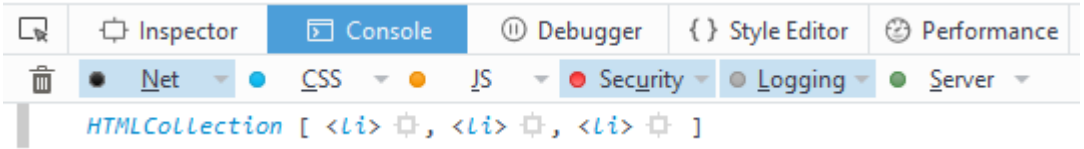
Kada želite da dohvatite više elemenata, JS će vam vratiti niz elemenata kojima možete pristupiti po indeks poziciji elementa. Kreirajte neuređenu listu u dokumentu.

```
1.    <ul>
2.        <li>Item 1</li>
3.        <li>Item 2</li>
4.        <li>Item 3</li>
5.    </ul>
```

Metod za dohvatanje elemenata po nazivu taga je **getElementsByTagName**. Unesite sledeći kod ispod elementa:

```
1.    <script>
2.        var lielementi = document.getElementsByTagName("li");
3.        console.log(lielementi);
4.    </script>
```

Kao ishod u konozoli browser-a dobićete HTML kolekciju li elemenata (slika 4). HTML kolekcija je skup HTML node elemenata koji su dobijeni po određenom kriterijumu (tag, klasa, atribut, ...).



Slika 4. HTML kolekcija li node elemenata

Nad kolekcijama nije moguća direktna primena svojstava već se mora pristupati elementima pojedinačno. U navedenom primeru prvom elementu pristupa se sa *lielement[0]*. Za pristup svim elementima koristi se for petlja. Broj elemenata u kolekciji dobija se iz svojstva **length**.

Za dohvatanje elemenata korišćenjem klasnog selektora koristi se metoda **getElementByClassName.** Povratna vrednost funkcije je HTML kolekcija.

Dohvatanje elemenata putem CSS selektora biće razmatrano u poglavlju koje se bavi CSS jezikom.

HTML kolekcije

HTML kolekcije su predefinisane kolekcije. Kolekcije kojima se može pristupiti su:

- document.anchors
- document.body
- document.documentElement
- document.embeds
- document.forms
- document.head
- document.images
- document.links
- document.scripts
- document.title

Pogledajmo primer kolekcije za linkove. Data je sledeća HTML struktura:

```
1.    <a href="#">Link 1</a>
2.    <a href="#">Link 2</a>
3.    <a href="#">Link 3</a>
4.    <a href="#">Link 4</a>
5.    <a href="#">Link 5</a>
```

Da bi se pristupilo linkovima koristiće se document.links kolekcija. Ispo kôda za linkove dodajte sledeći kôd:

```
1.    <script>
2.        var linkovi = document.links;
3.        console.log(linkovi);
4.    </script>
```

Kada otvorite konzolu u browser-u dobićete HTML kolekciju od pet linkova.

Kada ste se upoznali sa DOM strukturom, u HTML-u postoji tag koji zavisi od manipulacije JS-om. Element koji će biti objašnjen u nastavku je canvas element.

Canvas

Canvas je HTML5 element koji omogućava grafička iscrtavanja vizuelnih elemenata (slike, linije, ...) i teksta u okviru kontejnera koji ima tag <canvas>. Za rad sa <canvas> elementom potrebno je osnovno razumevanje HTML i Javascript koda.

Za početak potrebno je definisati <canvas> element.

```
<canvas id="mojkanvas" width="150" height="150"></canvas>
```

Ukoliko se atributi „width“ i „height“ ne definišu, podrazumevana veličina canvas-a je 300x150px u Firefox-u, a može varirati u zavisnosti od browser-a. Canvas je podržan na svim modernim browserima (IE9+). Kada je definisan element, dalje korišćenje kanvasa je kroz Javascript. Prvi korak je da izaberete kanvas element koji želite da koristite. U javascript-u postoji metoda ***getElementById*** koja će dohvatiti element čiji atribut „id“ ima vrednost navedenu kao argument. U primeru inicijalizacije kanvasa rečeno je da vrednost „id“ atributa je „mojkanvas“. Unutar <script> taga napišite sledeći kôd:

```
1. <script>
2. var canvas = document.getElementById('mojkanvas');
3. </script>
```

Da bismo dohvatili kanvas metodu pozivamo nad objektom „document“ koji se odnosi na ceo strukturu stranice od otvarajućeg <html> taga do njegovog zatvaranja. Sada smo dobili objekat. Kanvas kao objekat nema funkciju. Zamislite kanvas kao platno. Element koji smo izabrali je trenutno samo fizička predstava prostora na kome ćemo iscrtavati vizuelne objekte. Da bismo pristupili tom platnu odnosno površini za crtanje, moramo je definisati. Metoda koju poseduje canvas element je ***getContext*** koja dohvata površinu za crtanje. Kao argument se šalje tip grafike. Tip koji će se koristiti u primerima je ’2d’ a moguće je dodavanjem WebGL biblioteke raditi i 3D sadržaj. Dodajte još jednu liniju u vašem kôdu:

```
1. var canvas = document.getElementById("mojcanvas");
2. var ctx = canvas.getContext("2d");
```

Crtanje pravougaonika

Canvas podržava samo jednu vrstu primitivnih objekata – pravougaonik (rectangle). Ostali oblici se kreiraju korišćenjem linija koje se spajaju u navedenim tačkama. Postoje već ugrađene metode koje omogućavaju crtanje kompleksnih oblika.

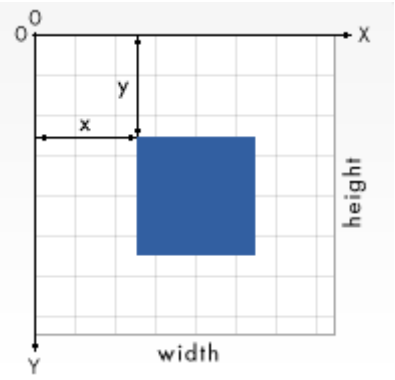
Metode koje su nam na raspolaganju su:

- fillRect(x, y, širina, visina) – crta popunjeni pravougaonik.
- strokeRect(x, y, širina, visina) – iscrtava ivice pravougaonika
- clearRect(x, y, širina, visina) – briše sadržaj za zadate dimenzije.

Nacrtati popunjeni pravougaonik dimenzija 100x100.

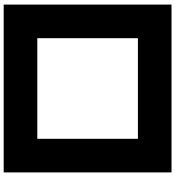
```
3.    <canvas id="mojkanvas" width="150" height="150"></canvas>
4.    <script>
5.        var canvas = document.getElementById("mojkanvas");
6.        var ctx = canvas.getContext("2d");
7.        ctx.fillRect(0,0,100,100);
8.    </script>
```

Kako u zadatku nije navedena početna pozicija za crtanje pravougaonika, pretpostavićemo da su početne koordinate 0 i 0. Koordinatni sistem za kanvas dat je na slici 1.



Slika 1. Koordinatni sistem za iscrtavanje na kanvas elementu

Argumente za „width“ i „height“ možete da posmatrate kao transliranje tačke x odnosno tačke y. Nareba fillRect formira četiri tačke $T_1(0,0)$, $T_2(100,100)$, $T_3(0,100)$ i $T_4(100,0)$ i popunjava prostor između njih. Ukoliko se ne navede boja ili tekstura za popunjavanje, podrazumevana vrednost je crna boja.

	Nacrtati objekat dimenzija 150x150 kao na slici. Debljina okvira je 30px. 

Iz primera možete da vidite da se radi o složenom objektu, odnosno nije ga moguće realizovati samo jednom naredbom. Za ovaj primer iskoristićemo kombinaciju metoda *fillRect* i *clearRect*. Rešenje zadatka dato je kôdom:

```
9.    <canvas id="mojkanvas" width="150" height="150"></canvas>
10.   <script>
11.       var canvas = document.getElementById("mojkanvas");
12.       var ctx = canvas.getContext("2d");
13.       ctx.fillRect(0,0,150,150);
14.       ctx.clearRect(30,30,90,90);
15.   </script>
```

Prvi korak je da se kreira popunjeni pravougaonik dimenzija 150x150. Drugi korak je kreirati unutrašnji pravougaonik. Ako pogledate sliku 1. videćete da je gornji levi ugao zadat koordinatama 0,0. Ako je debljina okvira 30px, to znači da od vrha treba pomeriti tačku y za trideset i sa leva tačku x za 30px. Metoda sada ima oblik clearRect(30,30,?,?). Preostaje da se odredi širina i visina pravougaonika. Imamo na raspolaganju 120px za širinu, od kojih 30px pripada desnom delu okvira, odnosno maksimalna širina je 90px. Kako crtamo kvadrat širina i visina su iste. Metoda clearRect sada je definisana sa (30,30,90,90) i na ovaj način dobijamo objekat zadat na slici.

Crtanje putanja

Drugi način crtanja objekata je korišćenje putanja. Bilo koji složeni objekat moguće je iscrtati korišćenjem metode spajanja tačaka. Da bi se putanja iscrtala moraju se implementirati sledeći koraci:

1. Prvi korak je započeti iscrtavanje putanje. Metoda koja se poziva nad kanvas objektom je ***beginPath***.
2. Drugi korak je korišćenje neke od metoda za iscrtavanje željenih oblika putanje.
3. Treći korak je opcionalno zatvaranje putanje. Ukoliko zatvorite putanju u tom slučaju će se poslednja i prva tačka spojiti. Metoda koja se poziva je ***closePath.***
4. Četvrti korak je iscrtavanje putanje na samom kanvasu. Može se pozvati metoda ***fill*** ili ***stroke*** u zavisnosti da li treba popuniti prostor unutar objekta ili samo prikazati ivice.

Prve funkcije za crtanje linije koje će biti predstavljene su ***lineTo*** i ***moveTo***. *LineTo(x,y)* je metoda koja od trenutne pozicije (podrazumevana je poslednja tačka koja je iscrtana) iscrtava liniju do zadatih koordinata x i y. *MoveTo(x,y)* pomera „olovku" na sledeću poziciju od koje će se iscrtati, bez da iscrtava koordinate.

Nacrtati objekat dat na slici korišćenjem putanja.

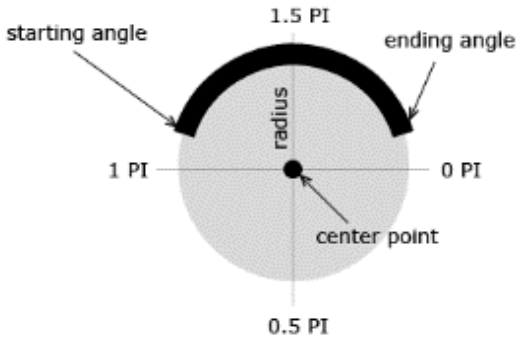
Dimenzije kanvasa su 250x250.

Objekat koji dat u primeru se sastoji od dva različita objekta – trougla. Za prvi objekat korišćeno je stroke svojstvo za iscrtavanje dok kod drugog objekta je korišćeno fill svojstvo. Rešenje je predstavljeno sledećim kôdom.

```
1.    <canvas id="mojkanvas" width="300" height="300"></canvas>
2.    <script>
3.        var canvas = document.getElementById("mojkanvas");
4.        var ctx = canvas.getContext("2d");
5.        ctx.beginPath();
6.        ctx.lineTo(0,100);
7.        ctx.lineTo(100,0);
8.        ctx.lineTo(100,100);
9.        ctx.closePath();
10.       ctx.stroke();
11.
12.       ctx.beginPath();
13.       ctx.moveTo(0,101);
14.       ctx.lineTo(101,101);
15.       ctx.lineTo(101,201);
16.       ctx.lineTo(0,101);
17.       ctx.closePath();
18.       ctx.fill();
19.   </script>
```

Ono što možete da primetite u kôdu je da za iscrtavanje drugog objekta je korišćen 1px više nego za drugi. Ovo je zato što stroke metoda dodaje 1px na originalnu veličinu za svaku ivicu. Sa ovim smo se već sretali kada smo pričali o modelu kutije za stilove objekata. Kako fill metoda ne dodaje ivice potrebno je dodati po 1px za svaku ivicu.

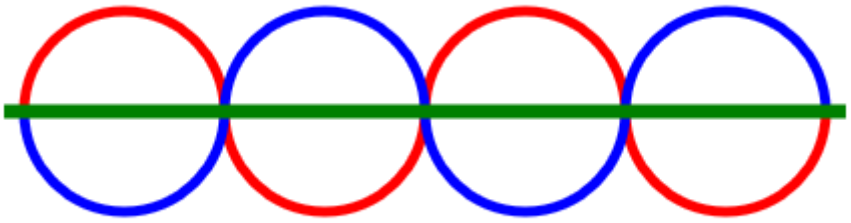
Sledeće metode za iscratavanje su ***arc*** i ***arcTo***. Nareba *arc* za zadate parametre iscrtava luk oko tačke za dati prečnik, početni i krajnji ugao. Poslednji argument je da li se luk iscrtava u smeru kazaljke na satu ili ne – boolean vrednost. Uglovi se zadaju u radijanima. Da biste dobili ugao u radijanima koristite sledeću formulu $radijan = \frac{\pi}{180} \cdot stepen$. Grafički prikaz kako arc funkcija radi dat je na slici 2.



Slika 2. Prikaz iscrtavanja luka korišćenjem *arc* metode

ArcTo metoda iscrtava luk između kontrolnih tačaka za zadati radius. Ugao luka se automatski formira bez uticaja programera.

Nacrtati objekat dat na slici korišćenjem putanja.



Dimenzije kanvasa su 450x300.

Pre rešenja zadatka biće objašnjena dva svojstva koja su potrebna za realizaciju zadatka. Prvo svojstvo je ***strokeStyle*** za tip iscrtavanja putanje korišćenjem boje, gradijenta ili šablona. Drugo svojstvo je ***lineWidth*** za debljinu linije.

Jedan način za rešavanje ovog zadatka dat je sledećim kôdom:

```
1.    <canvas id="sinusoida" width="450" height="300"></canvas>
2.        <script>
3.            var canvas = document.getElementById("sinusoida");
4.            var ctx = canvas.getContext("2d");
5.            ctx.beginPath();
6.            ctx.strokeStyle = "red";
7.            ctx.lineWidth = 5;
8.            ctx.arc(60,100,50,Math.PI,2*Math.PI,false);
9.            ctx.arc(160,100,50,Math.PI,0,true);
10.           ctx.arc(260,100,50,Math.PI,2*Math.PI,false);
11.           ctx.arc(360,100,50,Math.PI,0,true);
```

```
12.           ctx.stroke();
13.
14.           ctx.moveTo(50,200);
15.           ctx.beginPath();
16.           ctx.strokeStyle = "blue";
17.           ctx.lineWidth = 5;
18.           ctx.arc(60,100,50,Math.PI,0,true);
19.           ctx.arc(160,100,50,Math.PI,2*Math.PI,false);
20.           ctx.arc(260,100,50,Math.PI,0,true);
21.           ctx.arc(360,100,50,Math.PI,2*Math.PI,false);
22.           ctx.stroke();
23.
24.
25.           ctx.beginPath();
26.           ctx.strokeStyle = "green";
27.           ctx.lineWidth = 7;
28.           ctx.moveTo(0,100);
29.           ctx.lineTo(420,100);
30.           ctx.stroke();
31.       </script>
```

Iz datog kôda možete da vidite da smo iscrtavali tri različita objekta. Ovaj zadatak je moguće rešiti i sa dosta manje linija kôda korišćenjem funkcionalnosti Javascript-a.

Pored strokeStyle postoji i **fillStyle** koji će popuniti bojom objekte koji se iscrtavaju. I strokeStyle i fillStyle rade sa predstavljanjem boja kao rgb i rgba vrednost.

Za transparencije koja se koristi se svojstvo **globalAlpha**. Vrednost može da bude između 0 i 1. Menjanjem vrednosti svo dalje iscrtavanje koristi tu transparentnost. Kada ne želite da ta transparentost bude primenjena morate je vratiti na vrednost 1.

Za crtanje putanja bitna svojstva su i svojstva vezana za liniju. Prvo svojstvo je **lineWidth** koje definiše debljinu linije. Sledeće svojstvo je **lineCap** koje definiše kakvi su krajevi linije (zaobljeni, okrugli ili kockasti). Za zaobljene delove vrednost je *butt*, za okrugle vrednost je *rounded* a za kockaste vrednost je *square*.

Literatura

[1] Mozilla Canvas API - https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
[2] Canvas HTML5 Tutorial - http://www.html5canvastutorials.com/
[3] W3Schools Canvas Element - http://www.w3schools.com/tags/ref_canvas.asp
[4]

SVG – Scalable Vector Graphic

Вежба 3

SVG *(Scalable Vector Graphic)* je jezik za opisivanje dvodimenzionalne grafike. Nastao je 2001.godine i omogućava 3 vrste grafičkih objekata: vektorske grafičke oblike, slike (rasterska grafika) i tekst. SVG podržava implementaciju statične i animirane grafike. Modifikacije u vremenu se mogu opisati u *SMIL*, ili programirati u nekom jeziku kao što je npr. *Javaskript.*

SVG sadrži sledeći set osnovnih oblika:

- правоугаонике (укључујући опционе заобљене углове), створене са елементом "**rect**"
- кругови, креирани елементом "**circle**",
- елипси, креирани елементом "**ellipse**"
- праве линије, креиране елементом "**line**"
- полилини, створени елементом "**polyline**" ,
- полигоне, створене са елементом " **polygon** " .

Сви облици се могу приказати са попуном (fill) и/или ивичном линијом (stroke).

Основни елемент унутар кога се реализују исцртавања је "**svg".**

```
<svg x="0" y="0" width="500" height="500"></svg>
```

Елемент "<rect>"

Елемент "<rect>" дефинише правоугаоник који је поравнат са координатним почетком. Заобљене ивице се могу постићи додавањем атрибута „rx" и „ry".

Дефиниције атрибута

- **x** означава координату елемента по x оси. Уколико се не наведе, подразумевана вредност је 0.
- **y** **о**значава координату елемента по y оси. . Уколико се не наведе, подразумевана вредност је 0.
- **width** означава ширину елемента. Уколико се за ширину елемента постави вредност 0, елемент се неће приказати.
- **height** означава висину елемента. Уколико се за ширину елемента постави вредност 0, елемент се неће приказати.
- **rx** означава ниво заобљења ивица по x оси. Атрибут није обавезан. Уколико се не наведе, ивице ће бити оштре.
- **ry** означава ниво заобљења ивица по x оси. Атрибут није обавезан. Уколико се не наведе, ивице ће бити оштре.

У случају да је дефинисана вредност за само један од rx,ry елемената, недефинисани елемент преузима исту вредност.

Пример:

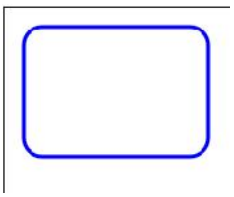
```
<rect x="10" y="10" width="100" height="70"
      fill="red" stroke="blue" stroke-width="2"/>
```

У овом примеру дефинисан је један правоугаоник који је позициониран од тачке (x,y)=>(10,10), димензије 100x70. Правоугаоник има дефинисане атрибуте fill=”red” који попуњава елемент црвеном бојом, затим атрибут „stroke=“blue” који исцртава ивичну линију плаве боје, и “stroke-width=2” који дефинише дебљину ивичне линије.

Пример:

```
<rect x="10" y="10" width="100" height="70"
      rx="10" fill="transparent" stroke="blue" stroke-width="2"/>
```

У овом примеру дефинисан је један правоугаоник који је позициониран од тачке (x,y)=>(10,10), димензије 100x70. Правоугаоник има дефинисан атрибут „rx=10“ који дефинише заобљење ивица. Такође, елемент “fill” има вредност „transparent” који дефинише да ће правоугаоник бити провидан. У случају да атрибут “fill” није дефинисан, подразумевана боја без је црна без обзира на тип елемента.



Елемент „<circle>”

Елемент <circle> дефинише круг. Дефинисање круга се заснива на центру и полупречнику.

Дефиниције атрибута

- **cx** – x координата центра круга. Подразумевана вредност је 0.
- **cy** – y координата центра круга. Подразумевана вредност је 0.
- **r** – полупречник круга. Уколико није дефинисан или је вредност 0, елемент се неће приказати.

Пример:

```
<circle cx="50" cy="50" r="30" fill="yellow" stroke="green"></circle>
```

У овом примеру дефинисан је круг са центром у тачки (cx,cy) => (50,50). Полупречник круга „r” је 30. Боја попуне „fill” круга је жута, а боја ивичне линије „stroke” је зелена.

Елемент <ellipse>

<ellipse> дефинише елипсу која је поравната са координатним почетком и одређена са средишњом тачком и два полупречника.

Дефиниције атрибута

- **cx** – x координата центра круга. Подразумевана вредност је 0.
- **cy** – y координата центра круга. Подразумевана вредност је 0.
- **rx** – полупречник по x оси.
- **ry** – полупречник по y оси.

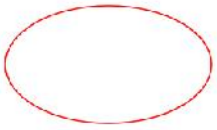
Уколико нису дефинисана оба полупречника, елемент се неће приказати.

Обратите пажњу! Ако оба полупречника имају исту вредност добићемо круг.

Пример:

```
<ellipse rx="70" ry="40" cx="100" cy="100" stroke="red"
fill="none"></ellipse>
```

У овом примеру дефинисана је елипса са центром у тачки (rx,ry) => (100,100) са полупречником по x оси 70, и полупречником по y оси 40. Ивична линија је црвене боје, попуна не постоји.

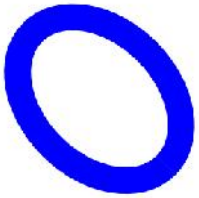


Пример:

```
<ellipse transform="rotate(45 105 105 ) "
         rx="70" ry="50" cx="100" cy="100"
         fill="none" stroke="blue" stroke-width="20"  >

</ellipse>
```

Атрибут „**transform**” омогућава извршавање неке од следећих трансформација: translate, rotate, scale итд.
У овом примеру „rotate(45 105,105) означава да ће се елипса ротирати за 45 степени, и да је центар ротације тачка (105,105). Ова тачка је изабрана из разлога што представља центар елипсе. С’обзиром да је елипса на позицији (100,100) и да има полупречник по x оси 70, а по y оси 50, центар елипсе је у тачки (105,105). Из тога следи да се елписа ротира за 45 степени око своје осе. Центар ротације може бити било која
друга тачка.



Елемент „<line”

<line> омогућава дефинисање линије која почиње у једној тачки и завршава се у другој.

Дефиниције атрибута

- **x1** дефинише почетну тачку по x оси
- **y1** дефинише почетну тачку по y оси
- **x2** дефинише крајњу тачку по x оси
- **y2** дефинише крајњу тачку по y оси

Линија се може исцртати коришћењем атрибута „stroke”.

Пример:

```
<line x1="100" y1="300" x2="300" y2="100"
      stroke-width="5"  />
```

Почетна тачка линије из овог примера је (100,300), а крајња тачка је (300,100). Дебљина линије је постављена на 5.

Елемент „<polyline“

<polyline> елемент дефинише скуп повезаних линијских сегмената. Обично се користи за дефинисање отворених облика.

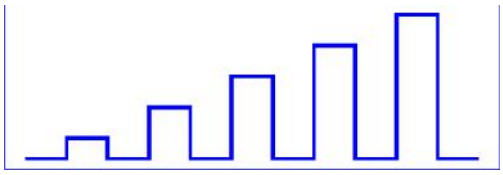
Дефиниције атрибута

- **points** дефинише преломне тачке линије

Пример:

```
<polyline fill="none" stroke="blue" stroke-width="10"
points="50,375
150,375 150,325 250,325 250,375
350,375 350,250 450,250 450,375
550,375 550,175 650,175 650,375
750,375 750,100 850,100 850,375
950,375 950,25 1050,25 1050,375
1150,375" />
```

Унутар атрибута „points” дефинисане су преломне тачке у паровима x,y одвојени размаком.



Елемент <polygon>

<polygon> елемент образује затворени облик који се састоји од низа повезаних линијских сегмената.

Дефиниције атрибута

- **points** дефинише преломне тачке линије

Пример:

```
<polygon fill="red" stroke="blue" stroke-width="10"
         points="350,75  379,161 469,161 397,215
                 423,301 350,250 277,301 303,215
                 231,161 321,161" />
<polygon fill="lime" stroke="blue" stroke-width="10"
         points="850,75  958,137.5 958,262.5
                 850,325 742,262.6 742,137.5" />
```

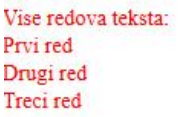
Пример дефинише два објекта, звезду и шестоугао као на слици.

Текст

Елемент <text> дефинише графички елемент који садржи текст. Овај елемент исто као и сви остали има исте могућности приказивања и трансформације.

```
<text x="0" y="20" fill="red" text-anchor="start" font-family="Verdana"
font-size="25">Vise redova teksta:
    <tspan x="0" y="40">Prvi red</tspan>
    <tspan x="0" y="60">Drugi red</tspan>
    <tspan x="0" y="80">Treci red</tspan>
</text>
```

Референтна тачка текста је у доњем левом углу. То значи да уколико текст испишемо на координати (x,y) => (0,0) текст ће се позиционирати изнад горње ивице и неће бити видљив. У примеру је текст по у оси померен за 20 што даје резултат као на слици испод.



Text-anchor својство може имати следеће вредности : “start”, “end”, “middle”. Означава да ли ће текст почети, или ће се завршити или ће средина текста бити у дефинисаној тачки (x,y).

Таг <tspan> омогућује исписивање текста у више редова. Битно је водити рачуна о размацима између редова јер они нису унапред одређени као у Text editor-има, већ је потребно рачунати позицију сваког реда у односу на (x,y).

Методе за трансформацију

Постоје следеће методе за трансформацију:

-**translate(x,y) –** премешта елемент са подразумеване тачке (0,0) или неке друге претходно дефинисане.

-rotate(alfa) – ротира елемент за угао алфа. Могуће је задати и координате тачке која ће бити центар ротације.

-scale(x,y) – скалира елемент

У претходним примерима већ смо поменули методу за ротирање елемента.

Пример:

```
<text x="20" y="20" fill="red" transform=" translate(100,100) rotate(90
50,50)" text-anchor="start">Vise redova teksta:
    <tspan x="0" y="40">Prvi red</tspan>
    <tspan x="0" y="60">Drugi red</tspan>
    <tspan x="0" y="80">Treci red</tspan>
</text>
```

У овом примеру атрибут „transform” је постављен на:

```
transform=" translate(100,100) rotate(90 50,50)”
```

Прво ће се извршити транслирање елемента, односно премештање елемента са његове тренутне позиције за 100 по x и 100 по y оси. У конкретном случају, текст се са тачке (20,20) помера на тачку (120,120) Затим ће се елемент на новој позицији ротирати за 90 степени око тачке (50,50) која се рачуна у односу на тачку (120,120) која је дефинисана као нови координатни почетак.

Градијент

Да би неки елемент могао бити обојен градијентом, потребно је претходно дефинисати изглед градијента, а потом поставити тај градијент као вредност атрибута за попуну елемента (fill).

```
<defs>
    <linearGradient id="grad1" x1="0" y1="0" x2="100" y2="0">
        <stop offset="0%" stop-color="red"></stop>
        <stop offset="100%" stop-color="yellow"></stop>
    </linearGradient>
</defs>
<rect x="0" y="0" width="200" height="150" fill="url(#grad1)"></rect>
```

Унутар <defs> тага дефинишемо градијент.
<linearGradient> означава да се ради о линеарном градијенту, додељен му је id=“grad1“ kako бисмо касније могли да га применимо.
Тачке (x1,y1) и (x2,y2) представљају смер простирања градијента.
Потребно је дефинисати бар два <stop> елемента. Сваки од њих има обавезан “*offset*” задат у процентиме који дефинише однос две боје на градијенту.

Након дефинисања градијента, потребно је исти применити на жељени елемент. Елемент <rect> са позиције (0,0) димензије 200x150 се боји градијентом тако што се за вредност својства попуне “*fill*” додели „id” претходно дефинисаног градијента.

Анимације

Пошто је Web динамички медијум, SVG подржава могућност промене векторске графике током времена. Неки од основних елемената за анимацију су: animate, animateTransform.

Пример: ротирање правоугаоника око своје осе

```
<rect x="150" y="150" width="150" height="150" fill="blue" stroke="black"
stroke-width="5" transform="rotation">

    <animateTransform
            attributeName="transform"
            begin="0s"
            dur="20s"
            type="rotate"
    from="0 225 225"
    to="360 225 225"
    repeatCount="indefinite"
    />
</rect>
```

Пример: ротирање звезде око своје осе и мењање својства *fill* у току времена

```
<svg x="0" y="0" width="10000" height="10000">
        <polygon points="200,110 140,298 290,178 110,178 260,298"
                style="fill:lime;stroke:purple;stroke-width:5;fill-
rule:nonzero;" >
            <animateTransform attributeName="transform"
attributeType="XML" type="rotate" from="0 200 200" to="360 200 200"
dur="5s" begin="0s" repeatCount="indefinite"></animateTransform>
            <animate attributeName="fill" values="lime;red;blue;orange"
attributeType="CSS" dur="5s" begin="0s"
repeatCount="indefinite"></animate>
        </polygon>

</svg>
```

Пример: померање и повећавање круга по хоризонтали

```
<svg x="0" y="0" width="10000" height="10000">

    <circle cx="0" cy="150" r="15" fill="blue" stroke="black" stroke-
```

```
width="1">
        <animate attributeName="cx" from="0" to="500" dur="5s"
repeatCount="indefinite" />
        <animate attributeName="r" to="100" dur="5s"
repeatCount="indefinite"></animate>
    </circle>

</svg>
```

CSS – Cascading Style Sheets

Вежба 4

CSS (Cascading Style Sheets) је језик за дефинисање стилова који одређују изглед *HTML* странице.

У почетку је једино *HTML* дефинисао комплетну структуру, изглед и садржај интернет странице. У следећим верзијама *HTML* језика почело се са додавањем тагова који ближе одређују изглед елемената. То је проузроковало потешкоће за web програмере јер су *HTML* странице постајале све веће и било је тешко сналазити се у њима. Да би се такав *HTML* документ упростио, од верзије *HTML 4*.0 уведен је *CSS*.

Након увођења ***CSS***-а, документи се деле на :

- *HTML* део који дефинише садржај странице
- *CSS* део који дефинише изглед и структуру странице

Синтакса CSS језика је крајње једноставна. Сваки CSS опис састоји се од дефинисања циљних елемената, својства и вредности.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Vežba4</title>
    <style>
        h1{
            color:red;
        }
    </style>
</head>
<body>
<h1>Naslov1</h1>
</body>
</html>
```

CSS стилови дефинишу се у формату:

Selektor { svojstvo: vrednost; }

CSS стил се повезује са HTML документом на следеће начине:

- Повезивањем на спољни документ
- Увожењем спољног документа
- Уметањем у заглавље документа
- Додавањем у линији атрибутом style

Повезивање на спољни документ изводи се тагом <link> који се додаје у заглавље HTML документа унутар тага <head>.

Битни атрибути тага <link>:

- **Rel** – дефинише однос између тага и одредишног документа (за повезивање CSS-а има вредност „stylesheet”)
- **Type** – дефинише тип одредишног документа (у овом случају тип документа је „text/css”).
- **Href** – URI (Uniform Resource Identifier) документа који се повезује.

Увожење спољног документа обавља се кључном речи @import која се уписује унутар <style> тага.

@import ulr(„css/stil.css“);

Такође, CSS се може дефинисати у заглављу документа унутар тага <style>. Могуће је и

комбиновати увоз и локалне дефиниције.

CSS се може дефинисати и атрибутом *style* унутар самог HTML тага.

Селектор идентификује елемент или део HTML странице.

Најчешће коришћени типови селектора:

- једноставни селектори
- Класни селектори
- ID селектори
- контекстни селектори
- псеудокласе

Једноставни селектори

Једноставни селектори одговарају називу HTML тага и примењују се на сваки истоврсни таг у документу. Коришћењ ових селектора не захтева интервенције у HTML коду. Селектор body је у самом врху хијерархије па ће сви елементи који се налазе испод њега наследити његова својства.

```
body{
    color:blue;
    text-align: center;
}
```

Класни селектори

Класе имају произвољне називе, и наводе се са тачком (.) испред назива. Класе нам омогућавају да исти стил применимо на више елемената истовремено, односно на све елементе који имају дату класу.

```
    <style>
        h1{
            color:red;
        }

        p.tekst{
            color:blue;
            text-align:center;
        }

    </style>
</head>
<body>
<h1>Naslov1</h1>
<p class="tekst"> Lorem ipsumemo provident sunt totacimus est necessit.
Dolore ducimus est necessitatibus quaerat quas
    quia rerum sit erum sit ullam vel! Ab alias autem enim in intotam!</p>
<p> Lorem ipsuas autem enim in inventore minima nemo provident vident sunt
totaat quas
    quia rerum sit ullam vel! Ab alias autem enim in inventore minima nemo
provident sunt totam!</p>
<div class="tekst"> Ovo je kontejner koji ima klasu tekst</div>
```

У примеру изнад постоје 2 параграфа од којих 1 има класу „tekst”, и један који такође има наведену класу. Такође, постоји и <h1> елемент за који је дефинисан стил који ће се применити на све елементе типа <h1>.

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

Овако дефинисан CSS стил односи се на све елементе типа параграф који имају класу „tekst”. Елемент такође има класу „tekst”, али се на њега овај стил не примењује. Уколико бисмо желели да се стил примени на све елементе са класом „tekst” без обзира на тип елемента, навели би само назив класе са тачком.

ID селектори

ID селектори се дефинишу коришћењем префикса #.

ID elementa служе за definisanje jedinstvenih objekata u HTML dokumentu. Sve deklaracije unutar klasa, ID ili elemenata su striktno definisane reči kao što je font-size, color, width, height... Svaka od deklaracija se zatvara tačka-zarezom (;).

ID селектор је везан само за један јединствени елемент.

```
    <style>

      #id{
          color:orange;

      }

    </style>
</head>
<body>
<h1>Naslov1</h1>
<p id="tekst"> Lorem ipsumemo provident sunt totacimus est necessit.
Dolore ducimus est necessitatibus quaerat quas
    quia rerum sit erum sit ullam vel! Ab alias autem enim in intotam!</p>
<p> Lorem ipsuas autem enim in inventore minima nemo provident vident sunt
totaat quas
    quia rerum sit ullam vel! Ab alias autem enim in inventore minima nemo
provident sunt totam!</p>

</body>
```

Контекстни селектори

Контекстни селектори се користе када желимо да применимо одређени стил на елемент који се налази унутар неког другог елемента.

```
    <style>

      p b{
            color:red
      }

    </style>
</head>
<body>
<h1>Naslov1</h1>
<p> <b>ovo je boldovani tekst</b>Lorem ipsuas autem enim in inventore
minima nemo provident vident sunt totaat quas
    quia rerum sit ullam vel! Ab alias autem enim in inventore minima nemo
provident sunt totam!</p>
</body>
```

Псеудокласе

Стања линкова

a: link {} - izgled neposećenog linka
a:visited{} - izgled posećenog linka
a:hover{} - izgled linka kada se preko njega pređe mišem
a:active{} – izgled linka od momenta pritiska mišem do otpuštanja klika

```
/* unvisited link */
a:link {
    color: #FF0000;
}

/* visited link */
a:visited {
    color: #00FF00;
}

/* mouse over link */
a:hover {
    color: #FF00FF;
}

/* selected link */
a:active {
    color: #0000FF;
}
```

Može se koristiti i kombinacija klasnih selektora i pseudoklasa: **a.vrsta1:link{}**

p:first-child{} - prvi potomak bilo kog elementa. Primeniće se samo na paragraph kada je on prvi potomak roditelja.

p i:first-child{} - primeniće se na <i> element koji je prvi potomak elementa <p>.

p:first-child i {} - primeniće se na svaki <i> element koji se nalazi unutar paragrafa koji je prvi potomak roditelja.

input:checked - primeniće se na sve input elemente koji imaju vrednost atributa checked=”checked”.

Primer:
Definisati formu sa 2 input polja tipa “checkbox” i atributom checked.

```
<form>
    <input type="checkbox" name="Prvi"  value="Apple" id="prvi"
checked="checked">
```

```
    <input type="checkbox" name="Drugi" value="Car" id="drugi">
</form>
```

input:valid {}
input:invalid {} - stilovi će se primeniti u zavisnosti od sadržaja input polja. Ukoliko je tip polja „email“ i nije unet znak @ primeniće se stilovi za invalid selector sve dok se ne unese ispravna adresa.

input:required {} - stilovi će se primeniti na input polja koja imaju oznaku “required” - obavezna polja. Ukoliko želimo da se stil primeni na sva polja koja nisu obavezna koristimo:
input:read-only {} -primeniće se na input polja koja imaju “readonly” oznaku.
input:read-write {} – primeniće se na input polja koja nemaju “readonly” oznaku

input:out-of-range{} - primeniće se na input elemente tipa number koji imaju definisane min i max attribute i čija je vrednost izvan definisanog opsega.
input:in-range{} - suprotno od prethodnog elementa, primenjuje se ukoliko je vrednost unutar definisanog opsega .

```
<form>
    <input type="number" name="Prvi"  min="5" max="10" value="5" id="prvi"
>
</form>
```

input:focus {} - primenjuje se kada se klikne na input polje
input:disabled - primenjuje se na input polje koje ima atribut disabled
input:enabled – primenjuje se na sva input polja koja nisu onemogućena.

p:only-child{} - primenjuje se na paragraph koji je jedino dete roditelja.

p:only-of-type{} - jedino dete datog tipa

```
    <style>
p:only-of-type{
    color:red;
}

    </style>
</head>
<body>
    <div><p>Paragraf1</p><span>dodatno</span></div>
    <div><p>Paragraf 2</p> <p>Paragraf 3</p></div>
</body>
```

U prvom div-u je <p> jedino dete tipa paragraph, u drugom divu postoje dva deteta tipa paragraph.

p:nth-of-type{n} - n-ti element tipa paragraph. Umesto n parametra moguće je upisati odd ili even što se odnosi na parno i neparno pojavljivanje selektora.

```
p:nth-of-type(4){
    color:red;
```

```
}

    </style>
</head>
<body>
<h1>Naslov1</h1>
<p>1 paragraf</p>
<p>2 paragraf</p>
<p>3 paragraf</p>
<p>4 paragraf</p>
<p>5 paragraf</p>
</body>
```

:not(p) - svaki element koji nije paragraph.

input[type=text]:disabled - odnosi se na sva input polja tipa tekst koja su disabled (onemogućena).

Pseudo elementi

Псеудо елементи се додају на селектор и односе се на посебне аспекте HTML документа.

```
    <style>
        p::before{
            content:"Read this...";
            color:red;

        }

        p::after{
            content:"Remember this...";
            color:red;

        }
    </style>
</head>
<body>
<p>My name is Donald.</p>
</body>
```

::before – садржај својства Content се додаје испред садржаја елемента
::after – садржај својства Content се додаје иза садржаја елемента
::first-letter - примењује се на прво слово текста унутар елемента.
::first-line – примењује се на прву линију текста унутар елемента
::selection – примењује се на селектовани текст унутар елемента

Када више селектора дели одређена својства, могуће је груписати селекторе тако што се одвајају запетама. Тада се дефинисана својства примењују на све селекторе.

```
<style>
   h1,h2,p {
       color:blue;
   }
</style>
```

Element <**span**> је линијски генерички контејнер који се користи када желимо да гурпишемо елементе да бismo na njih primenili neko stilizovanje. Npr. naglašavanje teksta.

Каскаде

Деловање селектора по приоритету:

- стил додељен атрибутом style
- ID селектори
- класни селектори
- контекстни селектори
- једноставни селектори

```
    <style>
        #naslov{color:red;}
        .plavo{color:blue;}
        h1{color:blue;}
    </style>
</head>
<body>
<h1 id="naslov">Naslov1</h1>
<p class="plavo" style="color:red">Lorem ipsum dolor sit amet, consectetur
adipisicing elit. Alias consectetur</p>
</body>
```

Текст *Naslov1* је црвене боје јер је id селектор „*#naslov*“ већег приоритета од једноставног селектора *h1*.
Текст параграфа је дефинисан атрибутом *style*, који је већег приоритета од класног селектора “*.plavo*” па је текст црвене боје.

CSS елементи се могу поделити на:

- блок елементе (block level elements)
- линијске елементе (inline elements)
- листе (lists)

Block елементи у документу одвојени су од осталих елемената што их окружују. Предефинисано се понашају тако да следећи елемент започиње у новој линији (npr. <p>, <h1>,)

Линијски елементи за разлику од блок елемената долазе у низу један за другим (<a>, <img>)

Листе су блок елементи који осим блок својстава имају и графичку или бројчану ознаку испред текста.

Својства текста

-font-family - дефинише фонт
-font-weight – дефинише дебљину слова (lighter,normal,bold)…
-font-size – дефиние величину фонта (може бити задато у px, pt, %, em)… 1em=16px;

Својство font-size се наслеђује:

```
body {  font-size:10pt;}
p    {  font-size:150%;}
```

U ovom primeru paragraf nasleđuje veličinu definisanu selektorom body, pa bi tekst u paragrafu bio prikazan fontom veličine 15pt.

-letter-spacing - definiše razmak između karaktera
- line-spacing – definiše razmak između linija teksta

-text-decoration – Описује декорације које се додају тексту. Може имати вредности: none, underline, overline, line-through.

-text-transform - Претвара слова у велика или мала. Могуће вредности су : capitalize, uppercase, lowercasse. Capitalize поставља почетно слово у свкаој речи на велико слово.

-text-align - хоризонтално поравнавање текста које се примењује само на блок елементе, а могуће вредности су : left, right, center, justify.

Задаци:

1. Kreirati jedan element sa proizvoljnim tekstom, i obezbediti da na prelazak miša preko , promeni se pozadinska boja elementa.

2. Kreirati HTML stranicu kao što sledi:

```
<body>
<div>Hover over me to show the p element
  <p>Tada! Here I am!</p>
</div>
</body>
```

   Obezbediti da kada korisnik pređe mišem preko da se pojavi element <p>.

3. Kreirati formu sa sledećim poljima vodeći računa o tipu polja : Ime i prezime, Email, Username, Password i obezbediti:
   -sva polja su obavezna
   -kada je polje u fokusu obojiti ivičnu liniju u zelenu boju
   -kada uneta vrednost nije validna, obojiti ivičnu liniju u crvenu boju.

4. Kreirati HTML kao što sledi:

```
<h2>Naslov</h2>
<div>Lorem ipsum dolor sit amet,<p> ipsum dolor </p> consed rerum
sint totam voluptas! </div>
<p>Lorem ipsum dolor sit amet, consed rerum sint totam voluptas!</p>
<p>Lorem ipsum dolor sit amet,  <i>Iskošen tekst </i> consed rerum
sint totam voluptas! <i>Iskošen tekst </i> rerum sint totam
volup</p>
<p>Lorem ipsum dolor sit amet,  <i>Iskošen tekst </i> consed rerum
sint totam voluptas! </p>
<div>Lorem ipsum dolor sit amet,  <i>Iskošen tekst </i> consed rerum
sint totam voluptas! </div>
<div>Lorem ipsum dolor sit amet,  <p>Paragraf123 </p> consed rerum
sint totam voluptas! </div>
```

   Obojiti u crveno samo <i> elemente koji su prvi potomak roditelja korišćenjem pseudo klasa.

5. Korišćenjem istog HTML dokumenta kao u prethodnom primeru, obojiti pozadinu u plavo svim paragrafima koji su potomci elementa.

1.

<style>

div {

background-color: green;

color: white;

padding: 25px;

text-align: center;

}

div:hover { background-color: blue; }

</style>

</head>

<body>

Mouse over the div element below to change its background color:

Mouse Over Me

</body>

2.

```
p {
    display: none;
    background-color: yellow;
    padding: 20px;
}

div:hover p {
    display: block;
}
</style>
</head>
<body>
<div>Hover over me to show the p element
  <p>Tada! Here I am!</p>
</div>
```

3.

```
    <style>
        input:invalid {
            border:1px solid red;
        }
         input:focus {
             border:1px solid green;
         }
    </style>
</head>
<body>

<form>
    <label>Ime i prezime:</label> <br>
    <input type="text" required  /><br>

    <label>Email:</label> <br>
    <input type="Email"  required  /> <br>

    <label>Username:</label> <br>
    <input type="text"  required /> <br>

    <label>Password:</label> <br>
    <input type="password" required  />
</form>
</body>
```

4.

```
p i:first-child{

    color:red
}
```

5.

```
div p {

     background: blue;
}
```

Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) je opisni jezik koji suži da browser-u opiše kako treba vizuelno da izgleda HTML dokument?

Često kod početnika postoji zabuna kod terminologije, koji jezik ima koju funkcionalnost kod klijentskog dela web aplikacije. Pod klijentskim delom misli se na sam prikaz bilo koje stranice sajta. Da ne bi bilo zabune između HTML-a i CSS-a kada se koristi termin „opisuje“, bolje je paralelno dati definicije čemu služi jedan a čemu drugi jezik u prikazu jedne web stranice:

- HTML je jezik koji opisuje strukturu dokumenta. On govori browseru koji su elementi dati na stranici kako bi ih browser znao interpretirati.
- CSS je jezik koji opisuje dizajn HTML elemenata u dokumentu. On govori browseru kako treba da izgleda svaki element na stranici.

U CSS-u definišete koja je boja slova paragrafa, koja je veličina slova, boju pozadine stranice itd. On nije vezan za strukturu stranice, samo sa vizuelnim načinom prikaza njenih elemenata.

U CSS-u glavni deo sintakse selektori (selector) i svojstva (property). **Selektor** je šablon za izbor elemenata nad kojima će biti primenjena određena svojstva. Postoji nekoliko načina da se izaberu elementi:

- Po tagu – Izbor elemenata po tagu definiše da svi elementi koji su datog taga implementiraju određena svojstva
- Po identifikatoru – Izbor jedinstvenog elementa na stranici. Samo jedan element može da ima dati indetifikator
- Po klasi – Izbor grupe elemenata koji imaju istu klasu.
- Po atributu – Izbor elemenata koji imaju specifičan atribut (a da nije identifikator ili klasa). Može biti name, href, src ili neki drugi atribut.
- Složeni selektori – Izbor elemenata korišćenjem složenog šablona. Jedan primer bi bio da se izaberu svi elementi koji se nalaze unutar trećeg div elementa na stranici.

Svojstvo je naziv određenog pravila kako treba neki deo elementa vizuelno prikazati. Na primer – svojstva tekstualnog prikaza na nekoj stranici su veličina fonta, tip fonta, boja fonta, debljina slova itd. Svako svojstvo se može pojedinačno modifikovati. Odnosno svako svojstvo je vizuelni deo tog elementa. Svojstva imaju svoje vrednosti. Vrednost zavisi od tipa svojstva.

CSS sintaksa ima opšti oblik:

```
selektor { svojstvo : vrednost; }
```

Jedan selektor može imati više svojstava:

```
selektor { svojstvo1: vrednost1; svojstvo2: vrednost2; ... }
```

Definicije za jedan selektor mogu se ponoviti više puta u jednom stilu. Na primer:

```
1.   h1 { color: red; }
2.   h1 { font-size: 24px; }
```

Ukoliko postoji ponovljena definicija sa različitim vrednostima kao:

```
1.   h1 { color: red; }
2.   h1 { color: green; }
```

gleda se druga definicija.

Da bi mogli da se koristite CSS stilovi, postoje tri načina da implementacije:

Inline – Moguće je definisati u samom tagu elementa CSS pravila za element. Da bi se implementirao CSS potrebno je koristiti atribut style.

```
<p style="color: red; font-size: 12px;">Ovo je paragraf</p>
```

Primetite da ovde nije korišćen selektor. To je zato što ovakav tip definisanja je vezan samo za element u kome je svojstvo definisano. Nije praksa koristiti ovakav način stilova, kod može postati nepregledan i teško čitljiv ostalim programerima. Koristiti samo ako je stil jako kratak, ili testirate direktno u datom delu aplikacije različite vrste prikaza.

Style tag – U okviru <head> taga moguće je definisati style tag sa različitim svojstvima.

```
1.   <style>
2.       p {
3.           color: red;
4.           font-size: 12px;
5.       }
6.       h1 {
7.           background-color: beige;
8.       }
9.   </style>
```

U ovom slučaju pravila se odnose na ceo dokument. Ovakav tip stilova koristi se kada na stranici ima malo elemenata koje treba stilizovati. Veći kod je poželjno držati u spoljnim dokumentima a zatim ih uvoziti na stranicu.

External page – Na stranicu se korišćenjem link taga ubacuje spoljni dokument. Za razliku od prva dva slučaja, eksterne stranice imaju prednost jer se mogu ponovo koristiti i na drugim stranicama. Link za stranicu se poziva na sledeći način:

```
<link rel="stylesheet" href="style.css">
```

Rel atribut je obavezan i govori kakav je tip dokumenta koji se uvozi. U nekim browserima stilovi neće biti učitani bez ovog atributa. Href je takođe obavezni atribut koji diktira lokaciju na kojoj se nalazi dokument.

Svojstva

U prethodnom delu objašnjeno je šta je CSS, kako se definišu pravila i kako se implementiraju u okviru web stranice. U narednom delu biće reči o nekim grupama i često korištenim svojstava i kakav uticaj imaju na stranice.

Svojstva koja se odnose na boju slova i prozirnost

Svojstvo	Opis / Vrednosti	Primer

color	Definiše boji slova teksta. Neki od tipova vrednosti za koje prima su:
	Tekstualne: red, blue, green, beige...
	Heksadecimalne: #fafabc, #333 (skraćeni oblik od #333333)
		`p { color: red }`
		`p { color: #444; }`
		`p { color: rgb(100,107,86); }`

	RGB: rgb(255,0,0), rgb(0,215,200)	
opacity	Definiše neprozirnost elementa. Može da prihvata vrednosti od 0 do 1.	`p { opacity: 0.5 }`

Svojstva vezana koja se odnose na karakteristike teksta

Svojstvo	Opis / Vrednosti	Primer
font-family	Definiše tip fonta, odnosno porodicu fontova koje prikazuju tekst nekog elementa. Može da se stavi više vrednosti. Ukoliko ime fonta ima više od jedne reči potrebno je ime staviti u navodnike. Redosled imena fonta je redosled fallback opcije. Ukoliko ne postoji dati font na računaru korisnika, uzima sledeći itd. Sans-serif, serif, monospaced su porodice fontova i stavljaju se kao zadnji argument i govore browseru da uzme bilo koji font koji pripada toj porodici ukoliko fontovi nisu dostupni.	`body { font-family: "Open Sans", Arial, sans-serif; }` `h1 { font-family: serif; }`

font-size	Definiše veličinu fonta. Veličina koristi neke od jedinica kao što su pikseli (px), procenti (%), points (pt) i drugi.
	Procenti se skaliraju u odnosu na veličinu fonta definisanu browserom za HTML tag. Jedinice kao procenti, em units, rem units su skalabilne jedinice, odnosno zavisne su od okruženja u kom se prikazuje stranica, dok su pikseli i points jedinice fiksne veličine.
		`p { font-size: 12px; }`
		`p { font-size: 125%; }`
		`p { font-size: 11pt; }`

font-weight	Definiše debljinu slova teksta. Može da koristi vrednosti od 100 do 900 ili normal, bold i bolder za debljinu. Ne podržavaju svi brovseri brojevne veličine pa je preporuka da se koriste tekstualne.	`h1 { font-weight: 800; }` `h1 { font-weight: 900; }`
font-style	Definiše da li je font regularan ili nakošen. Vrednosti koje atribut dobija su norma, italic ili oblique.	`p { font-style: italic }` `p { font-style: normal }`
text-decoration	Definiše da li tekst ima liniju ispod teksta, iznad teksta ili kroz tekst ili nema nikakve dekorativne linije. Vrednosti su: underline, overline, line-through, none.	`a { text-decoration: none; }` `p { text-decoration: overline; }`
text-align	Definiše poravnanje teksta u odnosu na kontejner (stranica ili element koji je ograničen dimenzijama). Vrednosti su: left, right, center, justify	`p { text-align: justify; }` `p { text-align: right; }`

Ovo su česta svojstva vezana za tekst, za ostale posetite stranicu: http://www.w3schools.com/css/css_text.asp

Svojstva koja se odnose na karakteristike pozadine

Svojstvo	Opis / Vrednosti	Primer

background-color	Definiše boju pozadine elementa. Neki od tipova vrednosti za koje prima su:
	- Tekstualne: red, blue, green, beige...
	- Heksadecimalne: #fafabc, #333 (skraćeni oblik od #333333)
	- RGB: rgb(255,0,0), rgb(0,215,200)
		`div {`
		`  width: 200px; height: 200px;`
		`background-color: #bada55; }`

background-image	Definiše sliku koju pozadina može da ima. Vrednost je url('*link ka slici*'). Kada se	`div {`

	koristi slika kao pozadinski element, potrebno je definisati dodatna podešavanja kao background-position i background-repeat kako bi se jasno definisalo kako slika treba da se prikaže.	width: 200px; height: 200px;
		background-image: url('images/flowers.jpg');
		}

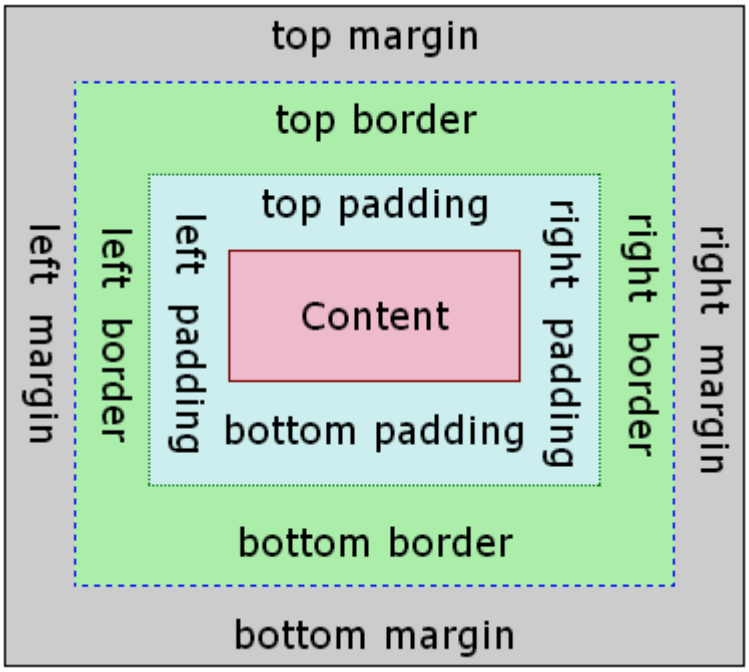
background-repeat	Definiše da li se slika ponavlja i u kom pravcu. Ukoliko slika nije dovoljno velika kao element za čiju se pozadinu prikazuje, browser podrazumevano ponavlja prikaz slike kako bi popunio prozor. Ovo dovodi do isecanja ponovljene slike ako je slika veća od prostora koji ispunjava.
	Vrednosti su: repeat, repeat-x, repeat-y, no-repeat
		div {
		width: 200px; height: 200px;
		background-image: url('images/flowers.jpg');
		background-repeat: repeat-x;
		}

background-position	Definiše poziciju slike pozadine. Za vrednosti se koriste vrednosti top, left, right, bottom, center kao pojedinačne vrednosti ili kao parovi vrednosti koji mogu imati dodatni offset. Primeri za vrednosti su:
	Background-position: top (slika se pozicionira uz gornju ivicu elementa i nalazi se u centru elementa)
	background-position: top left (slika se pozicionira uz gornji levi ugao)
	background-position: top 30px left 20px (slika se pozicionira uz gornji levi ugao sa 30px udaljenosti od gornje ivice i 20px udaljenosti od donje ivice)
		div {
		width: 200px; height: 200px;
		background-image: url('images/flowers.jpg');
		background-repeat: no-repeat;
		background-position: bottom 20px right 30px;
		}

Ovo su česta svojstva vezana pozadinu, za ostala dostupna svojstva posetiti stranicu http://www.w3schools.com/cssref/#background

Svojstva koja se odnose na veličinu elemenata

Box Model je koncept koje se odnosi na veličinu elementa (slika 1) i definiše da element osim veličina definisanih svojstvima za visinu i širinu, u svoju veličinu uračunava i razmak sadržaja od ivica elementa (padding), debljinu ivica (border) i razmak elementa od roditelja (margin). Margine ne utiču direktno na dimenzije elementa ali ih treba računati u slučaju da element se nalazi u drugom elementu gde bi njegova udaljenost od elementa uticala na dimenzije tog elementa.



Slika 1. Prikaz koncepta Box Model

	Kolike su dimenzije div element koji ima širina od 600px, visina od 400px udaljenost sadržaja sa svih strana od 20px, debljinu svih ivica 5px i margine sa svih strana od 10px.

Treba izračunati ukupnu dužinu i širinu elementa. Krenimo od dužine.

600px je širina elementa, na koju zbog udaljenost sa leve i desne ivice dodajemo po 20px koliko je udaljenost sadržaja od ivica. Zatim se dodaje i debljina ivica sa leve i desne strane po 5px. Kako element ne sadrži drugi element koji ima margine, margine ne ulaze u račun. Margine elementa ne utiču na njegovu veličinu.

Konačni račun za širinu je: 600px + 2 * 20px + 2 * 5px = 650px je dužina elementa.

Za visinu je isti princip računanja kao za širinu osim što se gledaju gornja i donja ivica.

Konačni račun za visinu je: 400px + 2 * 20px + 2 * 5px = 450px.

Dimenzije elementa su vizuelno 650 x 450 px i zbog poštovanja ovog pravila box model zadaje probleme osobama koje su prvi put susreću sa CSS stilovima.

Svojstvo	Opis / Vrednosti	Primer
width	Definiše širinu elementa. Vrednosti mogu da budu izražene u pixelima, points, procentima i drugim metričkim jedinicama.	`div { width: 50px; height: 50px; }`
height	Definiše visinu elementa. Vrednosti mogu da budu izražene u pixelima, points, procentima i drugim metričkim jedinicama.	`div { width: 50px; height: 50px; }`
border	Definiše svojstva svih ivica elementa. Za vrednosti uzima debljinu ivica, tip ivice (puna linija, isprekidana linija...) i boju ivice. Vrednosti deblije ivice mogu da budu izražene u pixelima, points, procentima i drugim metričkim jedinicama, tip ivice koristi	`div { width: 50px; height: 50px; border: 1px solid #333; }` `div { width: 50px; height: 50px; border-left: 1px dashed #654; }`

	predefinisane stilove (solid – puna linija, dashed – isprekidana, dotted – istačkana ...) Moguće je definisati svojstva svake ivice pojedinačno kada se umesto border koriste border-left, border-right, border-top, border-bottom.	

margin	Definiše razmak elementa sa svih strana od drugih elemenata ili ivica body elementa.
	Za vrednosti moguće koristiti sledeća tri formata:
	margin: vrednost – definiše jednaku udaljenost elementa sa svih strana
	margin: vrednost1 vrednost2 – vrednost1 je udaljenost elementa od njegovog vrha i dana, a vrednost2 od leve i desne strane
	margin: vrednost1 vrednost2 vrednost3 - vrednost1 je udaljenost elementa od njegovog vrha, od leve i desne strane, a vrednost3 od dna.
	margin: vrednost1 vrednost2 vrednost3 vrednost4 – vrednost1 je udaljenost elementa od njegovog vrha, vrednost2 udaljenost od desne strane, vrednost3 udaljenost od dna, vrednost4 udaljenost od leve strane
	Vrednosti mogu da budu izražene u pixelima, points, procentima i drugim metričkim jedinicama. Moguće je individualno menjati ivice korišćenjem svojstava margin-left, margin-right, margin-top, margin-bottom
		`div { width: 50px; height: 50px; margin: 10px; }`
		`div { width: 50px; height: 50px; margin: 10px 20px; }`
		`div { width: 50px; height: 50px; margin: 10px 20px 10px; }`
		`div { width: 50px; height: 50px; margin: 5px 10px 5px 10px; }`
		`div { width: 50px; height: 50px; margin-left: 10px; }`
		`div { width: 50px; height: 50px; margin-right: 10px; }`

padding	Definiše udaljenost sadržaja elementa od njegovih ivica. Razlikovati od margina, margine su vezane za sam element, ne za njegov sadržaj (tekst).
	Za vrednosti moguće koristiti sledeća tri formata:
	padding: vrednost – definiše jednaku udaljenost elementa sa svih strana
	padding: vrednost1 vrednost2 – vrednost1 je udaljenost elementa od njegovog vrha i dana, a vrednost2 od leve i desne strane
	padding: vrednost1 vrednost2 vrednost3 - vrednost1 je udaljenost elementa od njegovog vrha, od leve i desne strane, a vrednost3 od dna.
	padding: vrednost1 vrednost2 vrednost3 vrednost4 – vrednost1 je udaljenost elementa od njegovog vrha, vrednost2 udaljenost od desne strane, vrednost3 udaljenost od dna, vrednost4 udaljenost od leve strane
	Vrednosti mogu da budu izražene u pixelima, points, procentima i drugim metričkim jedinicama. Moguće je individualno menjati udaljenosti korišćenjem svojstava padding-left, padding-right, padding-top, padding-bottom
		`div { width: 50px; height: 50px; padding: 10px; }`
		`div { width: 50px; height: 50px; padding: 10px 20px; }`
		`div { width: 50px; height: 50px; padding: 10px 20px 10px; }`
		`div { width: 50px; height: 50px; padding: 5px 10px 5px 10px; }`
		`div { width: 50px; height: 50px; padding-right: 10px; }`

Identifikatori i klasni selektori

U prethodnom delu primeri su odnosili na sve elemente na stranici. Ukoliko napišete sledeću liniju koda

p { font-size: 18px; }

svim paragrafima na datoj stranici biće izmenjena veličina teksta na 18 pixela. Šta se dešava ako želimo da veličina samo prvog paragrafa bude 18px? Ili možda da drugi, četvrti i peti paragraf imaju tekst žute boje? Na neki način moramo da izdvojimo date elemente. To možemo na tri načina:

- definišemo jedinstveni identifikator koji se odnosi samo na jedan element na stranici
- definišemo klasni selektor koji se odnosi na grupu elemenata
- uzmemo neki atribut elementa ili grupe

Fokus će biti na prva dva načina izbora elemenata.

Identifikator

Predstavlja atribut koji daje jedinstveni naziv elementu kom želimo da menjamo svojstva. Pravilo je da na stranici mora da bude samo jedan element sa jedinstvenim nazivom. Iako tehnički niste sprečeni da dajete više elemenata isti naziv sa identifikatorom, to može izazvati probleme kod određenih browsera kao i sam Javascript koji neće pokretati skripte ukoliko postoje sintaksne greške u HTML kodu (iako će sam HTML kod biti prikazan). Sintaksa za identifikator izgleda:

```
<p id="nekiJedinstveniNaziv">
```

Pravila kod imenovanja su da ID ne treba da počinje sa brojem, koriste se mala i velika slova, brojevi, crtica i donja crta za imenovanje. Razmak se ne sme koristiti. Razmakom se navodi više različitih ID-jeva. Tako bi kod

```
<p id="neki Jedinstveni Naziv">
```

Značilo da p element ima tri identifikatora – „neki“, „Jedinstveni“ i „Naziv“.

Da bi se mogao stilizovati element sa navedenim ID-jem u CSS su takvi elementi biraju korišćenjem znaka **#** praćenog imenom dodeljenom sa ID-ijem.

```
#nekiJedinstveniNaziv { svojstvo1: vrednost1; ... }
```

	Data su tri paragrafa sa proizvoljnim tekstom, od kojih treći paragraf ima identifikator „treci“. Datom paragrafu promeniti boju fonta u crvenu i veličinu fonta na 24px.

HTML kôd:

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.      <meta charset="UTF-8">
5.      <title>Tri paragrafa</title>
6.      <style>
7.          #treci {  color: red;  font-size: 24px;  }
8.      </style>
```

```
</head>
<body>
<p>Paragraf #1</p>
<p>Paragraf #2</p>
<p id="treci">Paragraf #3</p>
</body>
</html>
```

Klasni selektor

Kada treba primeniti stil na određenoj grupi elemenata koristi se klasni selektor. Klasni selektor je atribut koji grupiše HTML elemente. Jedan primer kako se može koristiti klasni selektor je da se obeleže svi studenti nisu odslušali 80% laboratorijskih vežbi. Rezervisana reč je class a zatim se navodi ime za grupisanje. Primer o kom smo pričali napisan je ispod.

```
<ul>
    <li class="nemauslov">Pera Perić - 6/10</li>
    <li>Mika Mikić - 8/10</li>
    <li>Žika Žikić - 10/10</li>
    <li class="nemauslov">Milena Milenković - 4/10</li>
</ul>
```

U CSS-u biramo elemente korišćenjem klasnog selektora sa sledećom sintaksom:

```
.nazivKlase { svojstvo1: vrednost1; ... }
```

	U CSS-u definisati pravilo koje za klasni selektor „nemauslov“ postavlja boju fonta kao crvenu a debljinu slova postavlja na vrednost bold. Za testiranje koristit primer studenata koji nemaju 80% uslova za laboratorijske vežbe.

HTML kôd:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Tri paragrafa</title>
    <style>
        .nemauslov {  color: red;  font-weight: bold;  }
    </style>
</head>
<body>
<ul>
    <li class="nemauslov">Pera Perić - 6/10</li>
    <li>Mika Mikić - 8/10</li>
    <li>Žika Žikić - 10/10</li>
    <li class="nemauslov">Milena Milenković - 4/10</li>
</ul>
</body>
</html>
```

	Korišćenjem CSS-a i div elemenata kreirati kutije za informacije, upozorenja i greške. Kutija za informacije treba da ima svetloplavu pozadinu i tamnoplavi tekst. Kutija za upozorenja treba da ima svetlu nijansu žute i tamno narandžasti tekst. Kutija za greške treba da ima svetlu nijansu crvene i crveni tekst. Stilove definisati u zasebnom dokumentu – stilovi.css

Za početak potrebno je imenovati klasne selektore. Zašto ne identifikatore? Postoji mogućnost da aplikacija prikazuje više obaveštenja u različitim segmentima sajta. Ukoliko se dva elementa nalaze sa istim ID-ijem već je bilo reči da to može dovesti do greške u funkcionisanju elemenata.

HTML kôd:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="stilovi.css">
</head>
<body>
<div class="info">Kutija za obaveštenje</div>
<div class="warning">Kutija za upozorenje</div>
<div class="error">Kutija za greške</div>
</body>
</html>
```

CSS kôd (stlovi.css):

```
.error {
    background-color: #ff6f6b;
    color: darkred;
    padding: 5px;
}
.info {
    background-color: #94b3e1;
    color: #3b4cba;
    padding: 5px;
}
.warning {
    background-color: #feffc9;
    color: darkorange;
    padding: 5px;
}
```

Za prikaz podazadine i teksta korišćena su respektivno svojstva background-color i color, dok je radi malo estetskog prikaza korišćen padding da odvoji tekst od samih ivica elemenata.

Prikaz i pozicioniranje elemenata

Prikaz elemenata

Svi elementi na HTML stranici imaju način kako se pozicioniraju. Kreirajte stranicu sa jednom manjom slikom (npr. 120x120) i jednim dugmetom. Kod za elemente je:

```
1.   <img src="slika120x120.png" alt="slika 120x120">
2.   <button>Klikni me!</button>
```

Izgled ove stranice dat je na slici 4.

Slika 4. Izgled stranice sa slikom 120x120 i dugmetom „Klikni me!“

Dodajte paragraf između slike i dugmeta. Rezultat je dat na slici 5. Šta se dogodilo?

Slika 5. Izgled stranice sa slikom 120x120, paragrafom i dugmetom „Klikni me!“

Reč je o načinu kako se elementi prikazuju na stranici. U zavisnosti od elementa postoji podrezumevano svojstvo kako ga treba prikazati. U CSS-u moguće je manipulisati načinom prikaza elemenata. Za prikaz elemenata koristi se svojstvo display koje ima različite načine prikaza od kojih će za potrebe ovog kursa biti pomenuta četiri:

- none – Element se ne prikazuje na stranice. Često se koristi kada postoji interakcija kroz Javascript događaje.
- inline – Element se prikazuje tako što zauzima tačno onoliko prostora koliko je potrebno njegovom sadržaju. Moguće je za druge elemente da se prikazuju pored elementa sa ovakvim svojstvom. Elementima sa ovakvim načinom prikaza nije moguće menjati visinu i širinu. Elementi sa ovim svojstvom su <a>,,<button>...
- block – Element zauzima celu širinu browsera. Sa leve i desne strane elementa ne mogu biti drugi elementi. Moguće je podesiti širinu i visinu elemenata, ali i sa smanjenim dimenzijama ne oslobađaja se prostor za druge elemente. Elementi sa ovim svojstvom su <p> ,<h1>...<h6>,<table>...
- inline-block – Element zauzima onoliko prostora koliko zauzima njihov sadržaj ili kolike dimenzije mu se definišu sa svojstvima za veličinu. Moguće je za druge

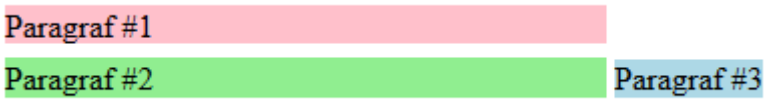
elemente da se prikazuju pored elementa sa ovakvim svojstvom. Element sa ovakvim svojstvom je <img>

	Na stranici je potrebno kreirati tri paragrafa. Prvi paragraf treba da ima širinu 300px, rozu boju pozadine i da se prikazuje kao block element. Drugi paragraf treba da ima širinu 300px, svetlozelenu boju pozadine i da se prikazuje kao inline-block element. Treći paragraf ima širinu 500px, svetloplavu boju pozadine i treba da se prikazuje kao inline element.

Za demonstraciju primera biće korišćen inline način definisanja CSS-a.

```
1.  <p style="width: 300px; background-color: pink; display: block">Paragraf
    #1</p>
2.  <p style="width: 300px; background-color: lightgreen;
              display: inline-block;">Paragraf #2</p>
3.  <p style="width: 500px; background-color: lightblue; display:
    inline;">Paragraf #3</p>
```

Vizuelni prikaz je dat na slici 6.



Slika 6. Rezultat primera prikaza tri paragrafa

Paragraf #1 se prikazuje kao block element (to je i podrazumevani način za prikaz paragrafa) i zbog svojstva prikaza iako zauzima 600px u dimenzijama drugi elementi ne mogu da budu sa njegove leve i desne strane tako da se Paragraf #2 prikazuje u sledećem redu.

Paragraf #2 se prikazuje kao inline-block element i kako je predefinisana veličina 300px moguće je posle njega da se prikaže i Paragraf #3 koji ima svojstvo koje mu dozvoljava da se pozicionira uz Paragraf #2. Paragraf #3 je inline element i veličina od 500px na njega nema uticaj, jer inline elementi samo zauzimaju veličinu njihovog sadržaja. U ovom slučaju to će biti prostor koji zauzima tekst „Paragraf #3“.

Pozicioniranje elemenata

Pored načina kako prikazujemo elemente moguće je pozicionirati ih na stranici. Za pozicioniranje elemenata koristi se svojstvo **position** koje ima sledeće vrednosti:

- static – podrazumevano svojstvo. Elementi se prikazuju u redosledu u kom su napisani u kodu.
- absolute – Elementi se pozicioniraju na određenu poziciju u odnosu na web stranicu ili na roditelja koji ima poziciju sa vrednošću relative.
- relative – Elementi se pozicioniraju na novu poziciju u odnosu na poziciju na kojoj su se nalazili na stranici pre promene položaja.
- fixed – Elementi se fiksno pozicioniraju na određenu poziciju u odnosu na web stranicu, ali za razliku od absolute elementi se pomeraju sa stranicom ukoliko sadržja na stranici postoji.

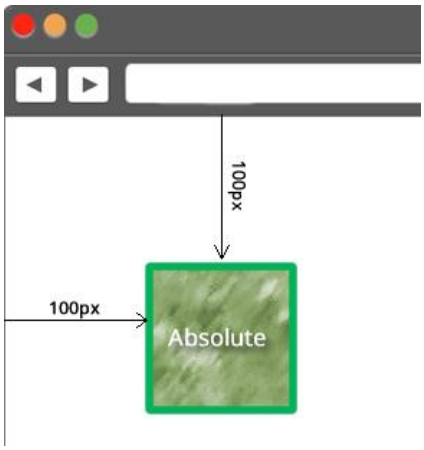
Pozicije elemenata koji imaju za svojstvo position manipulišu se sa svojstvima **top,left,right** i **bottom**.

Element sa „absolute“ / „fixed“ position vrednošću

Definisan je element sa apsolutnom pozicijom koji ima dimenzije 100x100 i nalazi se od leve ivice stranice za 100px i od vrha stranice 100px. Kod za primer je dat ispod:

```
1.   div {
2.       width: 100px; height: 100px;
3.       position: absolute; top: 100px; left: 100px;
4.   }
```

Demonstracija rezultata data je na slici 7.



Slika 7. Pozicioniranje elementa sa apsolutnom pozicijom u odnosu na stranicu

Drugi slučaj je kada imamo element koji je pozicioniran relativno i sadrži element koji je pozicioniran apsolutno.

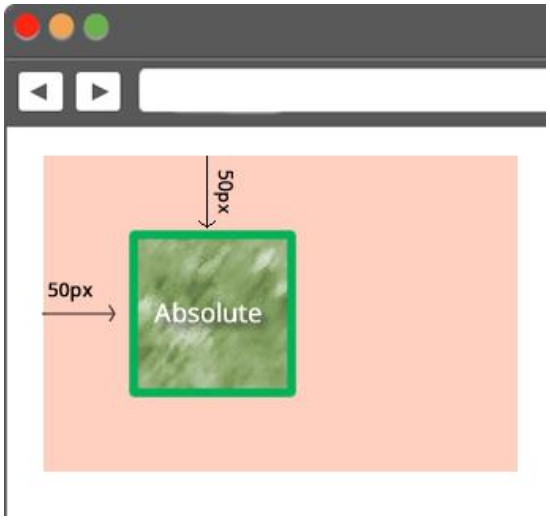
HTML kôd:

```
1.   <div id="roditelj">
2.      <div id="dete"></div>
3.   </div>
```

CSS kôd:

```
1.   #roditelj { position: relative; width: 200px; height: 200px; }
2.   #dete {
3.       width: 100px; height: 100px;
4.       position: absolute; top: 50px; left: 50px;
5.   }
```

Demonstracija rezultata data je na slici 8.



Slika 8. Pozicioniranje elementa sa apsolutnom pozicijom u odnosu na roditeljski element sa relativnom pozicijom

Elementi sa „fixed“ position vrednošću imaju iste osobe kao i absolute elementi, ali su uvek vidljivi na datoj poziciji (ako se scroll-uje stranica element ostaje na istoj poziciji) i uvek su u relaciji sa stranicom (ukoliko se nalaze u drugom elementu, i dalje se pozicioniraju u odnosu na stranicu).

Element sa „relative“ position vrednošću

Za razliku od apsolutnog pozicioniranja relativno pozicioniranje pomera element od svoje stare pozicije na novu. Kod za primer dat je ispod:

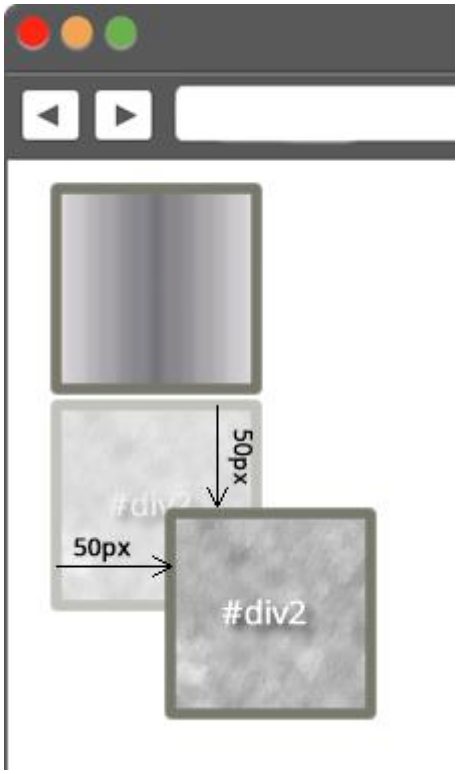
HTML kôd:

```
1.   <div style="width:100px;height:100px;"></div>
2.   <div id=="div2"></div>
```

CSS kôd:

```
1.   #div2 {
2.   width:100px;height:100px;
3.   position:relative;  top: 50px;  left: 50px;
4.   }
```

Demonstracioni prikaz dat je na slici 9.



Slika 9. Pomeranje div elementa za 50px od stare pozicije

	Kreirati stranicu sa četiri div elementa koji imaju identifikatore topDiv1, topDiv2, topDiv3 i topDiv4 respektivno.topDiv1 sadrži topDiv2, a topDiv3 sadrži topDiv4. topDiv1 je pozicioniran relativno i udaljen je 100px ulevo i 60px od vrha od stare pozicije. Širine je 150px a visine 100px i ima plavu boju pozadine. topDiv2 je pozicioniran apsolutno i nalazi se 10px od dna i 15px udesno od ivica roditelja. topDiv3 je pozicioniran relativno i udaljen je 100px ulevo i 100px od vrha od stare pozicije. Širine je 150px a visine 80px i ima zelenu boju pozadine topDiv4 pozicioniran fiksno 45px od leve ivice i 45px od vrha. Dimenzija je 30x30 i ima crvenu boju.

HTML kôd:

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.      <meta charset="UTF-8">
5.      <title>Hello World!</title>
6.  </head>
7.  <body>
8.      <div id="topDiv1">
9.          <div id="topDiv2"></div>
10.     </div>
11.     <div id="topDiv3">
12.         <div id="topDiv4"></div>
```

```
    </div>
</body>
</html>
```

CSS kôd:

```
#topDiv1 {
    position: relative;
    top: 60px;
    left: 100px;
    width: 150px;
    height: 100px;
    background-color: #94b3e1;
}

#topDiv3 {
    position: relative;
    top: 100px;
    left: 100px;
    background-color: #fffa47;
    width: 150px;
    height: 80px;
}

#topDiv2 {
    position: absolute;
    bottom: 10px;
    right: 15px;
    background-color: #3e8e41;
    width: 40px;
    height: 40px;
}

#topDiv4 {
    position: fixed;
    top: 45px;
    left: 45px;
    width: 30px;
    height: 30px;
    background-color: red;
}
```

CSS Pseudo-elementi i pseudo-klase

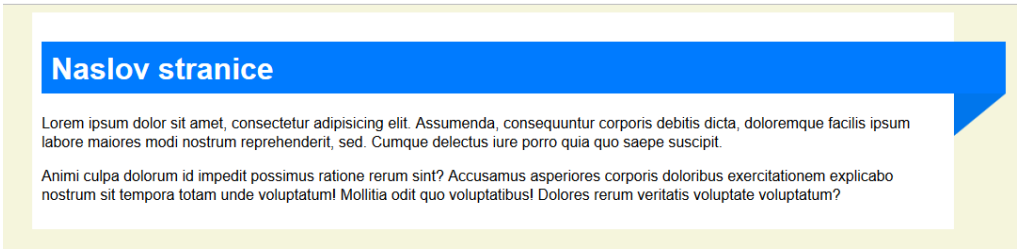
Pseudo-elementi

Pseudo-elementi su elementi koji dozvoljavju vam izaberete specifičan deo dokumenta i da ga izmenite.

before i after

Dve moćne pseudo-elementi su **before** i **after** koje nam omogućavaju da stilizujemo sadržaj pre ili posle sadržaja elementa. Njihova najveća moć je da korišćenjem svojstva ***content*** dodamo sadržaj na stranicu bez izmene postojeće HTML strukture.

Kreirati dizajn kao na slici.



Na slici je data stranica koja ima zaglavlje čiji deo izlazi iz okvira date stranice. Jedan od načina za postizanje ovog elementa bi bilo da imamo element koji je roditelj dva blok elementa i onda na osnovu apsolutnog pozicioniranja bi drugi element bio tako da se dobije vizuelni efekat. Isto rešenje se može dobiti sa „after“ pseudo-elementom.

HTML kôd:

```
<div class="stranica">
<h1>Naslov stranice</h1>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Assumenda, consequuntur corporis debitis dicta,doloremque facilis ipsum labore maiores modi nostrum reprehenderit, sed. Cumque delectus iure porro quia quo saepe suscipit.</p>
<p>Animi culpa dolorum id impedit possimus ratione rerum sint? Accusamus asperiores corporis doloribus exercitationem explicabo nostrum sit tempora totam unde voluptatum! Mollitia odit quo voluptatibus! Dolores rerum veritatis voluptate voluptatum?</p>
</div>
```

U zasebnoj CSS datoteci stilizovaćemo element sa klasom „stranica“ i njegovu decu. Pre korišćenja pseudo-elementa napisaćemo neki osnovni kôd kao osnovni šablon od kog ćemo krenuti da detaljnije stilizujemo stranicu.

CSS kôd:

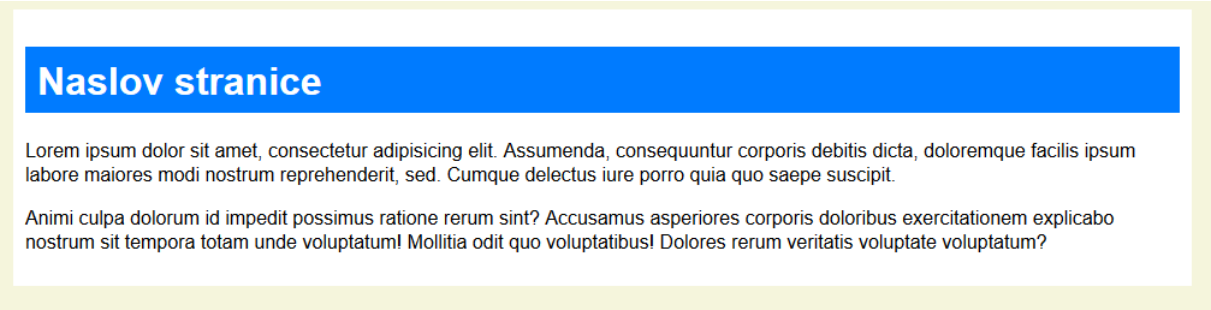
```
body {
    background-color: beige;
    font-family: Arial, sans-serif;
}

.stranica {
    background-color: white;
    padding: 10px;
    width: 960px;
    margin: 0 auto;
}

.stranica h1 {
    color: white;
    height: 35px;
```

```
16.       padding: 10px;
17.       background-color: #007bff;
18.   }
```

Trebalo bi da dobijete dizajn kao na slici 1.



Slika 1. Početni dizajn stranice za primer x.

Kada smo dobili osnovni dizajn prva stvar koju vidimo je da zbog padding svojstva elementa klase „stranica“ h1 element nema punu širinu na desnoj strani. Za rešenje ovog problema potrebno je izmeniti dva svojstva – position i width. Poziciju elementa stavićemo da je relativna, a za širinu moramo da pogledamo dati dizajn. Na slici se jasno vidi da element treba da probije roditeljski kontejner za neku proizvoljnu širinu. Usvojićemo da širinu elementa treba proširiti za 45px. Kako ne znamo kolika je tačno širina elementa, osim ako je sami ne definišemo što nije dobra praksa, koristićemo svojstvo „calc“. „Calc“ svojstvo uzima matematički izraz i adaptira ga u zavisnosti od veličine stranice. Naš izraz je da na 100% širine elementa kolika god ona bila dodajemo 45px. Potrebno je da izmenite selektor u vašoj datoteci sa sledećim kôdom:

CSS kôd:

```
1.  .stranica h1 {
2.      position: relative;
3.      color: white;
4.      width: calc(100% + 45px);
5.      height: 35px;
6.      padding: 10px;
7.      background-color: #007bff;
8.  }
```

Sada smo izvršili sve izmene nad h1 elementom. Sledeći korak je da sa pseudo-elementom „after“ dodamo sadržaj. Pre nego što prikažemo kôd za rešenje treba pojasniti neke stvari. Prva je sama pozicija pseudo-elementa. Njegova pozicija će definitivno biti zadata korišćenjem svojstava „right“ i „bottom“. Element sigurno treba fiksirati uz desnu ivicu tako da „right“ svojstvo imaće vrednost 0. Što se tiče tiče bottom pozicije ona će imati negativnu vrednost i biće jednaka visine elementa. Što se tiče pozicioniranja elementa, stavljamo apsolutnu poziciju (kako je h1 ima relativnu poziciju, element će se pozicionirati u odnosu na njega). Druga stvar je kreiranje oblika elementa. Znamo da su svi elementi na stranici pravougaonog oblika, pa se postavlja pitanje kako je dođeno do trougaonog. Mogli smo da kreiramo „stripe“ i da ga iskoristimo kao pozadinu, ali postoji i drugi način koji je ekspoitacija CSS3 svojstava. Korišćenjem ivica možemo odseći elementu deo i kreirati trougao. Kao što je „border-radius“ svojstvo daje nam mogućnost kreiranja elipsi i krugova,

tako i „transparent“ vrednost za ivice može da iseče određeni deo elementa. U nastavku je kôd za pseudo-element koji će biti detaljnije razjašnjen:

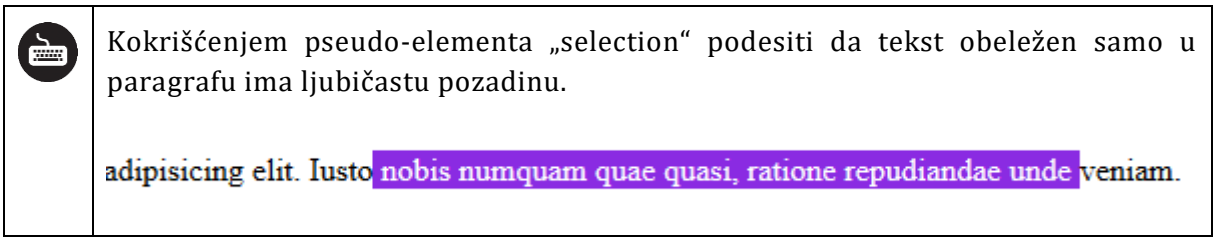
CSS kôd:

```
1.   .stranica h1:after {
2.       content: '';
3.       position: absolute;
4.       right: 0;
5.       bottom: -45px;
6.       display: block;
7.       width: 0;
8.       height: 0;
9.       border-style: solid;
10.      border-width: 45px 55px 0 0;
11.      border-color: #0076ec transparent transparent transparent;
12.  }
```

Na liniji 2 svojstvu content dodeljen je prazan sadržaj. Pseudo-element ne može da postoji a da nema neki sadržaj, makar sadržaj bio prazan. Na lijima 7 – 11 dat je kod za formiranje trougaonog oblika. „border“ svojstvo je podeljeno na delove. „border-style“ ima vrednost „solid“ odnosno ivice su pune linije za sve četiri ivice. Onda je sa „border-width“ data širina ivica – top, right, bottom, left. Vidimo da donja i leva ivica imaju vrednost 0. „border-color“ govori da samo gornja ivica ima boju dok ostale nemaju. Kako je desna ivica transparenta a ima debljinu ta transparentnost se prelama sa gornjom ivicom i zbog toga dobijamo izgled troula. Kao što je ranije rečeno ovo je ekspoitacija koja je posledica kako se „border“ svojstvo ponaša, i otkrivena je kao rad ljudi koji imaju dugogodišnje iskustvo u frontend razvoju.

Selection

Pseudo-element selection dozvoljava da postoji proizvoljni izgled za teksta teksta koji je obeležio korisnik. Svojstva koja se jedino mogu menjati su color, background, cursor, i outline. Mozilla Firefox ne podržava standard za selection i mora se koristiti –moz- prefiks.



Kokrišćenjem pseudo-elementa „selection“ podesiti da tekst obeležen samo u paragrafu ima ljubičastu pozadinu.

adipisicing elit. Iusto nobis numquam quae quasi, ratione repudiandae unde veniam.

CSS kôd:

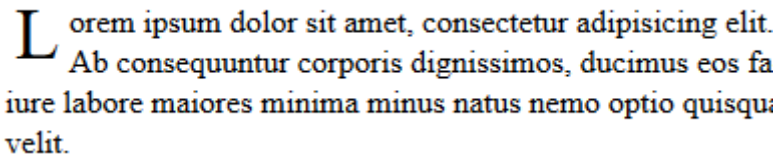
```
1.   p::-moz-selection {  background-color: #8a2be2;  color: white;  }
2.   p::selection {  background-color: #8a2be2;  color: white;  }
```

Na liniji 1 napisan je kôd za Firefox, dok druga linija je namenjena za standard koji podržavaju ostali pretraživači.

First-line i first-letter

Pseudo-elementi „first-line“ i „first-letter“ kao što im naziv kaže dozvoljavaju modifikaciju nad tekstom elementa. „first-line“ menja prvu liniju teksta. Linija koja se menja zavisi od

različitih faktora kao što su dužina elementa, veličina ekrana, veličina teksta. U zavisnosti od ovih faktora, korisnici koji posećuju vaš sajt mogu imati različito iskustvo. „first-letter“ menja samo prvo slovo teksta.

	Korišćenjem pseudo-elementa formatirati prvo slovo svakog paragrafa da ima dizajn prikazan na slici. 

CSS kôd:

```
1.   p::first-letter {
2.       font-size: 36px;
3.       padding: 5px;
4.       float: left;
5.   }
```

Da bi ostali tekst, bez obzira na dužinu, popunio prostor oko slova korišćeno je svojstvo float koje će izvaditi element iz normalnog toka. Tekst ima specifičnu karakteristiku da umesto da budu ispod lebdeće elementa, oni obavijaju dati element. „padding“ svojstvo je stavljeno kako bi se napravio razmak između slova i teksta koji ga obavija.

Pseudo-klase

Pseudo-klase su slične selektorima klasa, osim što mogu biti i rezultat akcije korisnika. CSS 2 standardom bile su definisane prve pseudo-klase vezane za korisnikovu interakciju sa linkovima – visited, active i link. Sa CSS 3 standardom lista novih pseudo-klasa značajno je proširena sa novim selektorima koji dozvoljavaju manipulaciju nad sadržajem koji se ranije mogao jedino postići korišćenjem složenog HTML kôda ili Javascript-a.

Pseudo-klase za linkove

Uvod u pseudo-klase počećemo sa najstarijim selektorima. Već smo pomenuli tri stanja linka – linkovi koji nisu posećeni (link), aktivni link kada korisnik klikne na link (active) i link koji je korisnik već posetio (visited).

	Korišćenjem pseudo-klasa za link, definisati link element koji vodi ka sajtu vIŠER-a i podesiti da je neposećeni link crven, da je aktivni link zelen a posećeni link narandžast.

HTML kôd:

```
<a href="http://www.viser.edu.rs">VIŠER</a>
```

Dok bi kod za CSS pseudo-klase bio:

```
1.   a:link { color: red; }
2.   a:active { color: green; }
3.   a:visited { color: orange; }
```

Pseudo-klase za input elemente

CSS 3 standard definisao je nekoliko pseudo-klasnih selektora sa ciljem za manipulaciju input elemenata (ali neki nisu eksplicitno pisani samo za njih). U nastavku biće demonstrirana neka od njih:

- **checked** – selektor za input elemente koji su tipa radio button ili checkbox, odnosno elementi kojima se korisničkom interakcijom menja stanje – true/false. Primenjuje se kada je element izabran (štikliran, izabrano radio dugme).

Korišćenjem pseudo-klasa za promenu stanja elementa, promeniti veličinu elementima na 40px po širini i visini. Od elemenata data su dva dugmeta u istoj grupi i jedno dugme za štikliranje.

HTML kôd:

```
1.    <input type="radio" name="rb"> Dugme 1
2.    <input type="radio" name="rb"> Dugme 2
3.    <input type="checkbox"> Checkbox 1
```

CSS kôd:

```
input:checked {  width: 40px;  height: 40px;  }
```

- **disabled** – selektor za elemente koji imaju atribut disabled.

Korišćenjem pseudo-klase za izbor elemenata koji imaju atribut disabled, promeniti im boju pozadine.

Za demonstracioni primer uzećemo jedan input element i jednu padajuću listu koje imaju atribut „disabled“.

HTML kôd:

```
1. <input type="text" disabled>
2. <select disabled>
3.     <option value="1">1</option>
4.     <option value="2">2</option>
5.     <option value="3">3</option>
6. </select>
```

CSS kôd:

```
*:disabled { background: #ff8f77; }
```

- **enabled** – selektor za elemente u koje korisnik može uneti vrednost.

Korišćenjem pseudo-klase za izbor elemenata koji imaju atribut enabled, promeniti im boju pozadine.

Za demonstracioni primer biće korišćeni jedan input element u koji može da se unese tekst i jedan u koji ne može.

HTML kôd:

```
1.    <input type="text">
2.    <input type="text" disabled>
```

CSS kôd:

```
input:enabled { background-color: yellow; }
```

- **focus** – selektor za elemente koji se nalaze u fokusu.

	Korišćenjem pseudo-klase za izbor elemenata koji imaju atribut focus, promeniti im veličinu sa 100px na 250px koristeći proizvoljnu tranziciju.

HTML kôd:

```
<input type="text">
```

CSS kôd:

```
1.   input {  width: 100px;  transition: width .5s; }
2.   input:focus {  width: 250px; }
```

- **invalid** – selektor za input elemente koji zahtevaju da vrednost bude u određenim granicama ili format za unos nije dobar (email, date, pattern).

	Korišćenjem pseudo-klase za izbor elemenata koji imaju atribut invalid, promeniti im stil u stil za upozorenje.

HTML kôd:

```
<input type="number" min="1" max="10" value="5">
```

CSS kôd:

```
input:invalid { color: #ff244e; background-color: #fff2d1; }
```

U tekstualno polje unesite vrednost izvan opsega, npr. 25 kako biste videli promenu.

- **optional** – selektor za sve elemente koji nemaju „required“ atribut.

	Korišćenjem pseudo-klase za izbor elemenata koji nemaju atribut required, promeniti im boju okvira u blago naranžastu.

Za demonstraciju primera jedan element neće imati „required“ atribut dok drugi hoće.

HTML kôd:

```
1.   <input type="text">
2.   <input type="text" required>
```

CSS kôd:

```
input:optional { border: 1px solid #ffc9aa; }
```

- **required** – selektor za sve elemente koji imaju „required“ atribut.

	Korišćenjem pseudo-klase za izbor elemenata koji imaju atribut required, promeniti im boju okvira u neku nijansu plave.

Za demonstraciju primera jedan element neće imati „required“ atribut dok drugi hoće.

HTML kôd:

```
1.   <input type="text">
2.   <input type="text" required>
```

CSS kôd:

```
input:required { border: 1px solid #6aafff; }
```

- **valid** – selektor za input elemente koji zahtevaju da vrednost bude u određenim granicama ili format za unos je dobar (email, date, pattern).

Korišćenjem pseudo-klase za izbor elemenata kojima je unos dobar, promeniti im boju okvira u stil odobrenja.

Za demonstraciju primera biće korišćen input element za unos e-mail adrese.

HTML kôd:

```
<input type="email">
```

CSS kôd:

```
input:valid { background-color: #d0ffcf; border: 1px solid #389959; }
```

Pseudo-klase za izbor dece elemenata

Možda i najvažniji selektori su selektori pseudo-klasa za izbor dece elemenata. Ranije problem izbora dece elemenata svodio sa na ispis klasa uz svaki element, što je usložnjavalo HTML kôd, a u nekim slučajevima zahtevalo od programera korekcije kao što će biti demonstrirano u narednom primeru. Recimo da trebate da održavate statičku stranicu na kojoj se nalazi cenovnik usluga. Od dizajnera ste dobili tabelarni izgled gde je prva vrsta zaglavlje tamno plave boje, dok se vrste u cenovniku smenjuju između nijanse svetle sive i bele boje. Jedno rešenje je da se kreiraju pravila klase „siva“ i „bela“ a zaglavlje kako je jedinstveno može da ima i identifikator. Problem je šta se desi ako neko obriše neki red? U tom slučaju mora se modifikovati cela HTML struktura te tabele. U praksi ovo se rešava tako što se struktura elementa dinamički generiše u zavisnosti od sadržaja. Ovaj problem je prepoznat pa su sa CSS3 standardom uvedene specifikacije za nove pseudo-klase:

- first-child
- first-of-type
- last-child
- last-of-type
- not
- nth-last-child
- nth-last-of-type
- nth-of-type
- only-of-type
- only-child