UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika – praktična matematika (VSŠ)

Bojana Lazarič

Uporaba kaskadnih slogovnih predlog (CSS)

Diplomska naloga

Ljubljana, 2007

KAZALO

ZAHVALA

Iskreno se zahvaljujem mentorju mag. Matiji Lokarju za napotke in predvsem potrpežljivost pri izdelavi diplomske naloge. Prav tako se zahvaljujem družini in vsem ostalim, ki so mi stali ob strani.

PROGRAM DIPLOMSKE NALOGE

V diplomski nalogi prikažite najpomembnejše ukaze in postopke pri oblikovanju spletnih strani s pomočjo kaskadnih slogovnih predlog.

Osnovna literatura:

- Bartlett, Kynn, *Sams Teach Yourself CSS in 24 Hours*
- Kaltenekar, *Hitri vodnik – HTML, CSS in JavaScript – Oblikovanje spletnih strani*

mentor

mag. Matija Lokar

POVZETEK

V diplomski nalogi bom predstavila kako lahko s pomočjo kaskadnih slogovnih predlog oblikujemo spletne strani. Cilj diplomske naloge je na preprost in enostaven način razložiti, kako oblikovanje ločimo od same priprave vsebine spletne strani.

Diplomsko nalogo sem razdelila na osem poglavij. V prvem uvodnem poglavju so navedene prednosti uporabe kaskadnih slogovnih predlog, kratka zgodovina ter opisana orodja in testi za preverjanje kako različni pregledovalniki podpirajo kaskadne slogovne predloge. Od drugega do šestega poglavja so predstavljene osnove kaskadnih slogovnih predlog kot so zapis slogov, vstavljanje slogov v dokument HTML, pravila dedovanja in kaskadnosti, vrednosti in enote ter vrste elementov. Sedmo poglavje je namenjeno prikazu postopnega oblikovanja spletne strani, v osmem poglavju pa je prikazano še nekaj prijemov, s katerimi dopolnimo videz prej oblikovane spletne strani.

Math. Subj. Class. (2000): 68N15, 68N19

Computing Review Class. System (1998): D.1.7, D.2.0, D.2.2, D.2.3, D.2.11, D.3.2, H.5.2, I.7.2

Ključne besede: CSS, HTML, spletno oblikovanje, standardi, označevalni jeziki
Keywords: CSS, HTML, web design, standards, markup languages

1. UVOD

1.1. KAJ JE CSS

Kratica CSS je nastala iz angleškega izraza **C**ascading **S**tyle **S**heets, kar v prevodu pomeni kaskadne slogovne predloge. Označuje jezik, ki nam pomaga pri oblikovanju spletnih strani. Z njim določimo, kako naj spletni pregledovalnik prikaže posamezne elemente strani kot na primer povezave, besedilo, tabele, obrazce, ozadja, slike .... S tem jezikom torej opišemo, kakšna je oblikovna predloga različnih spletnih elementov. Izraz kaskadna označuje način, ki pove, kako se različni načini določanja izgledov kombinirajo v ustrezni prikaz.

1.2. ZAKAJ UPORABLJATI CSS?

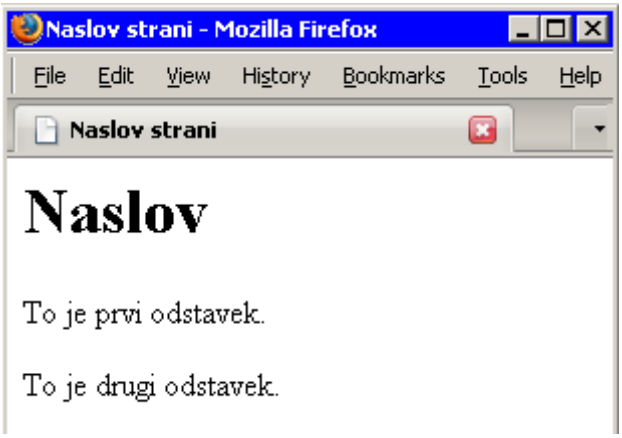
Glavni namen uporabe kaskadnih slogovnih predlog je ločiti vsebino od oblikovanja. To pomeni, da je sama vsebina internetne strani napisana na datoteki HTML, definicija izgleda internetne strani pa je navedena v datoteki CSS.
Prednost uporabe slogov je v tem, da, ko enkrat določimo izgled določene značke, bo ta na vseh straneh videti enako. Če kasneje želimo spremeniti njen izgled, ga spremenimo samo na enem mestu. S tem prihranimo na času, se izognemo morebitnim napakam pri popravljanju ter obdržimo enoten izgled vseh strani.

Poglejmo si primer preproste strani HTML brez ter z vključeno kaskadno slogovno predlogo. Razlika v kodi med posameznimi primeri je označena z rdečo barvo. Tak način pisanja bo uporabljen povsod v diplomski nalogi.

PRIMER HTML 1.2–1: Preprosta stran HTML brez slogov

stran.html

```
<html>
 <head>
   <title>Naslov strani</title>
 </head>

 <body>
   <h1>Naslov</h1>
   <p>To je prvi odstavek.</p>
   <p>To je drugi odstavek.</p>
 </body>
</html>
```



Slika 1.2–1: Izgled strani brez slogov

V dokument HTML bomo vključili kaskadno slogovno predlogo. V njej bomo nastavili, da naj bo ozadje svetlo sinje barve, glavni naslov v določenem odtenku modre barve, v odstavku pa bo za besedilo uporabljena pisava modre barve.

PRIMER HTML 1.2–2: Stran HTML z vključeno kaskadno slogovno predlogo

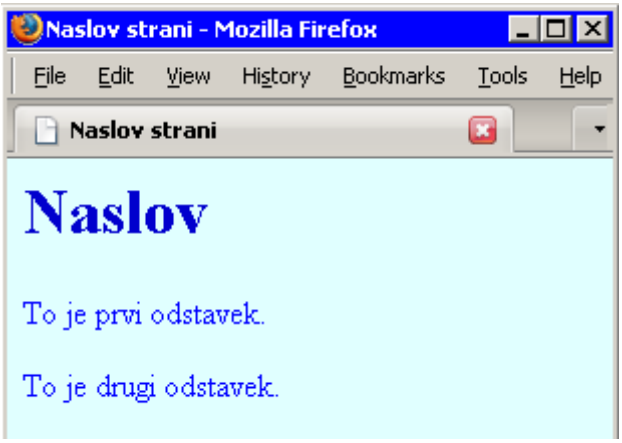
stran.html

```
<html>
 <head>
   <title>Naslov strani</title>
   <link rel="stylesheet" href="stil.css" type="text/css">
 </head>

 <body>
   <h1>Naslov</h1>
   <p>To je prvi odstavek.</p>
   <p>To je drugi odstavek.</p>
 </body>
</html>
```

PRIMER CSS 1.2–2: Kaskadna slogovna predloga

stil.css

```
body {
 background-color: #e0ffff;
}

h1 {
 color: #0000cd;
}

p {
 color: #0000ff;
}
```



Slika 1.2–2: Izgled strani z vključeno kaskadno slogovno predlogo

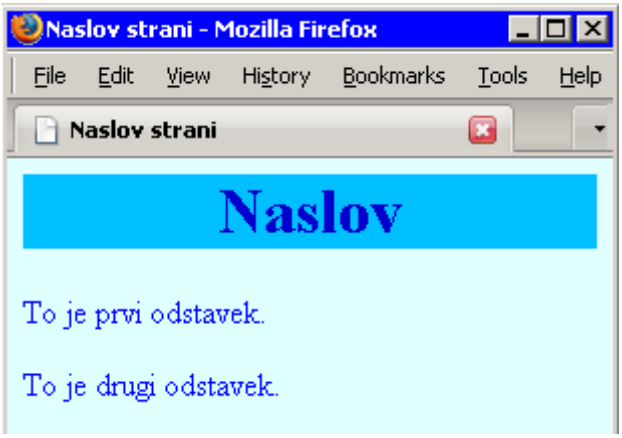
Vidimo, da z enostavno vključitvijo kaskadnih slogovnih predlog izgled same spletne strani precej spremenimo. Pomembno je, da se vsebina same strani ni spreminjala.
Če sedaj ugotovimo, da bi bolj ustrezal drugačen videz strani, je potrebno spremeniti le datoteko CSS (v našem primeru torej datoteko `stil.css`) – torej definicijo izgleda. V sami datoteki HTML ni nobene spremembe!

Spremenimo kaskadno slogovno predlogo tako, da bo glavni naslov poravnan sredinsko ter bo imel nebeško modro ozadje. Spremembe opravimo le v datoteki `stil.css`.

PRIMER CSS 1.2–3: Spremenjen izgled glavnega naslova

stil.css

```
body {
 background-color: #e0ffff;
}

h1 {
 color: #0000cd;
 text-align: center;
 background-color: #00bfff;
}

p {
 color: #0000ff;
}
```



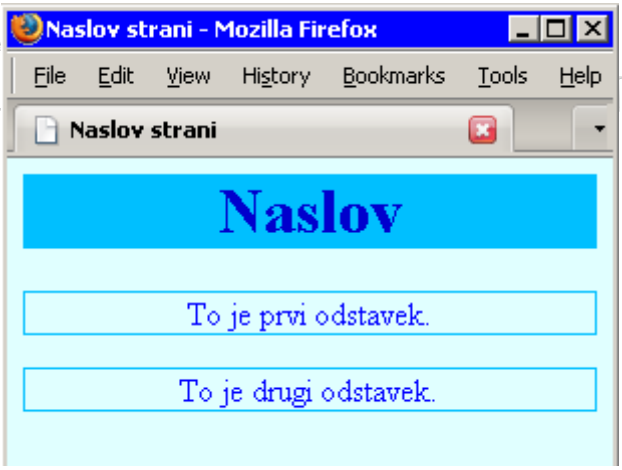
Slika 1.2–3: Stran HTML s spremenjenim izgledom glavnega naslova

Spremenimo slogovno predlogo še nekoliko bolj. Odstavkoma dodajmo nebeško modro obrobo in ju sredinsko poravnajmo.

PRIMER CSS 1.2–4: Spremenjen izgled odstavkov

stil.css

```
body {
 background-color: #e0ffff;
}

h1 {
 color: #0000cd;
 text-align: center;
 background-color: #00bfff;
}

p {
 color: #0000ff;
 text-align: center;
 border: 1px solid #00bfff;
}
```



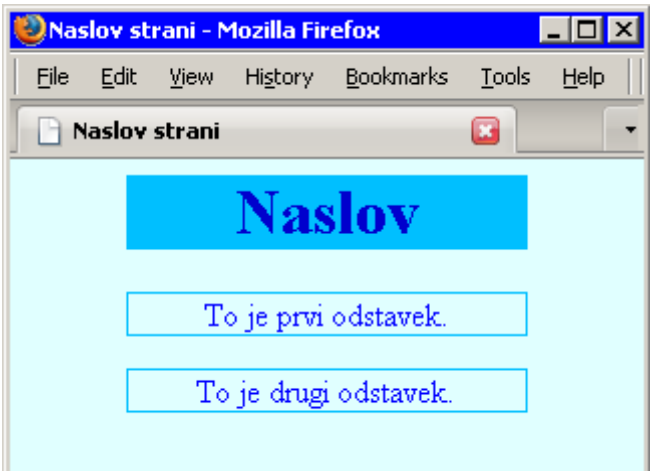
Slika 1.2–4: Stran HTML s spremenjenim izgledom odstavkov

Denimo, da ugotovimo, da bi bilo lepše, da vsebina strani zavzema sredino okna, v katerem teče pregledovalnik. Zopet so potrebne spremembe le v sami datoteki `stil.css`, v datoteki `stran.html` pa ne. Seveda se s spremembo slogovne datoteke `stil.css` spremenijo prav vse strani, ki to predlogo uporabljajo.

PRIMER CSS 1.2–5: Sredinsko poravnana vsebina spletne strani

stil.css

```
body {
 background-color: #e0ffff;
 width: 500px;
 margin-left: auto;
 margin-right: auto;
}

h1 {
 color: #0000cd;
 text-align: center;
 background-color: #00bfff;
}

p {
 color: #0000ff;
 text-align: center;
 border: 1px solid #00bfff;
}
```



Slika 1.2–5: Stran HTML s sredinsko poravnano vsebino

Videli smo torej, da z uporabo slogovnih predlog oblikovanje povsem ločimo od besedila. S spremembami predlog vplivamo na izgled spletne strani, sama vsebina datoteke HTML, kjer je zapisana vsebina strani, pa ostane nespremenjena.

V naslednjem razdelku bomo na kratko pogledali razvoj kaskadnih slogovnih predlog.

1.3. KRATKA ZGODOVINA CSS

Prvotni jezik HTML je bil enostaven. Vseboval je samo nekaj preprostih ukazov, ki se jih je bilo mogoče naučiti v nekaj urah. Namenjen je bil strukturiranju spletne strani in ne določanju njenega izgleda. Sam izgled spletne strani je bil prepuščen spletnim pregledovalnikom. Z razvojem interneta ter večjo uporabo grafičnih uporabniških vmesnikov pa je postal izgled spletne strani zelo pomemben. Pojavila se je potreba po boljših pregledovalnikih in močnejšem jeziku, ki bi omogočal tudi definiranje izgleda strani. Razvijalci so svoje pregledovalnike izgrajevali in dograjevali ter dodajali nove lastnosti jeziku HTML. Jezik HTML se je razvijal in bil opremljen z vedno močnejšimi in zmogljivejšimi ukazi, kot so na primer ukazi, s katerimi določamo lastnosti ozadja, postavljamo okvirje, določamo oblike pisav in podobno. Za izgled spletnih strani je organizacija World Wide Web Consortium[1] (v nadaljevanju W3C) želela izdelati dopolnilen označevalni sistem. Z njim naj bi na čim bolj preprost način definirali izgled spletne strani brez vpliva na samo strukturo dokumenta, zapisanega v jeziku HTML. Tako so nastale kaskadne slogovne predloge (CSS). Možnost uporabe kaskadnih slogovnih predlog je bila dodana v HTML 4.0.

Prva priporočila za CSS je izdelal W3C 17. decembra 1996 pod imenom CSS1. V priporočilih CSS1 so bile definirane slogovne lastnosti besedila (pisave, barve, ozadja) in njihove dovoljene vrednosti (vrednosti za dolžino, odstotke, barvo, naslove virov). Določen je bil način vstavljanja slogov v dokument HTML, sistem dedovanja značk in slogovnih predlog, prioriteta slogov, način pisanja komentarjev v slogovnih predlogah, določanje višine vrstice itn. Definirani so bili nepravi (pseudo) razredi (za oblikovanje videza povezave do že obiskane strani, do neobiskane strani ter aktivne povezave) in neprave (pseudo) značke (za oblikovanje prve vrstice in prve črke). Določen je bil način oblikovanja bločnih in vrstičnih ukazov[2]. Da bi lahko značkam spreminjali oziroma dodajali (nove) slogovne učinke, sta bila v jezik HTML dodana nova atributa `class` in `id`.

12. maja 1998 je ista organizacija izdelala predlog za drugo generacijo slogov, znan pod imenom CSS2. Šlo je za razširitev CSS1 z manjšimi spremembami. V tej verziji so še razširili dosedanje lastnosti oblikovanja, kot je pozicioniranje elementov (relativno, absolutno in fiksno), podporo za tiskanje (nedovisno od izgleda na zaslonu lahko določimo, kako se bo spletna stran natisnila na tiskalniku), podporo za razširitev nastavitev pisav, določevanje prikaza elementov (na primer vrstičnemu elementu lahko določimo, da se obnaša kot bločni element), določanje videza miškinega kazalca, določanje minimalne in maksimalne velikosti in širine itn.

W3C je 19. julija 2007 izdelal zadnji veljavni osnutek, poznan pod imenom CSS2.1. Vsebuje popravke številnih napak iz CSS2 in nekaj novih značilnosti. Predvsem pa so bile odstranjene določene značilnosti prejšnjih priporočil, za katere se je pokazalo, da jih večina pregledovalnikov ni podprla.

Pripravljeni pa so tudi že osnutki za CSS3. Novost v tej verzji bo razdelitev specifikacije CSS na module (na primer modul o selektorjih, modul o barvah, tekstovni modul, modul o vrednostih in enotah itn.). Priporočilo bo dopolnilo dosedanje lastnosti, dodane bodo lastnosti, ki bodo omogočale vključevanje avdia itn.

[1] World Wide Web Consortium je mednarodni inštitut, kjer člani organizacije (univerze, podjetja kot so Microsoft, Netscape, Macromedia in drugi) in javnost sodelujejo ter skupaj razvijajo standarde za splet.

[2] Razlika med bločnimi in vrstičnimi ukazi je v prikazu vsebine. Vsebina bločnega ukaza se prične na začetku nove vrstice (na primer: `div`, `h1`, `li`, `p`), medtem, ko se pri vrstičnih prikaže znotraj vrstice na mestu, kjer je bil ukaz vstavljen (na primer: `a`, `img`, `span`).

Seveda je definicija priporočil eno, drugo pa, kako so različice priporočil CSS podprte s strani pregledovalnikov. Ker so ta priporočila zelo obsežna, jih večina pregledovalnikov ne podpira v celoti. Določene stvari spustijo, druge zopet upoštevajo "po svoje". Če želimo, da se spletne strani v različnih pregledovalnikih in starejših verzijah prikazujejo tako, kot je določeno v specifikaciji CSS, moramo poznati lastnosti pregledovalnikov in v kolikšni meri specifikacijo CSS podpirajo. V naslednjem razdelku si bomo pogledali nekaj orodij za preverjanje podpore specifikacije CSS.

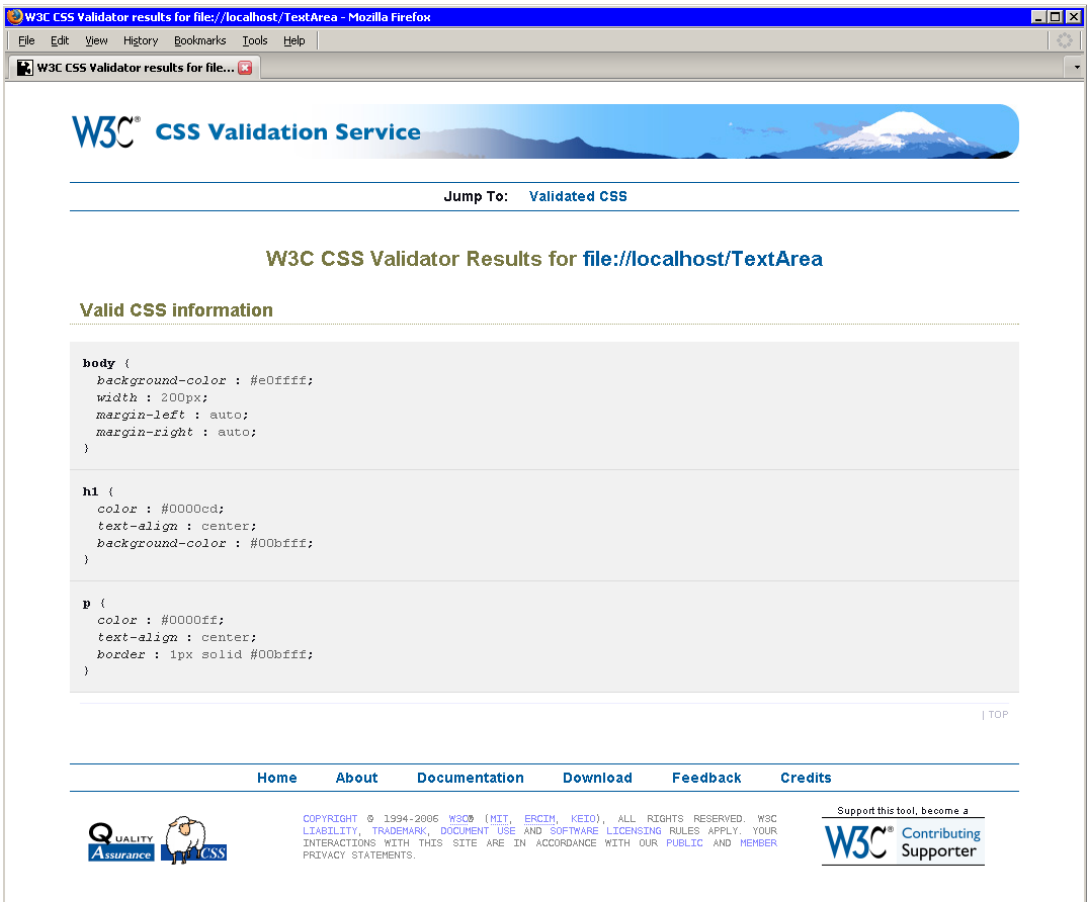
1.4. ORODJA IN TESTI ZA PREVERJANJE PODPORE SPECIFIKACIJE CSS

Na spletu je kar nekaj različnih orodij, s pomočjo katerih lahko preverimo, kako določeni pregledovalniki podpirajo posamezno različico priporočil CSS. Z njimi lahko preverimo, ali se (in v katerih pregledovalnikih) internetne strani prikažejo točno tako, kot je bilo določeno s priporočilom CSS.

Nekaj internetnih strani, kjer najdemo orodja za preverjanje CSS:

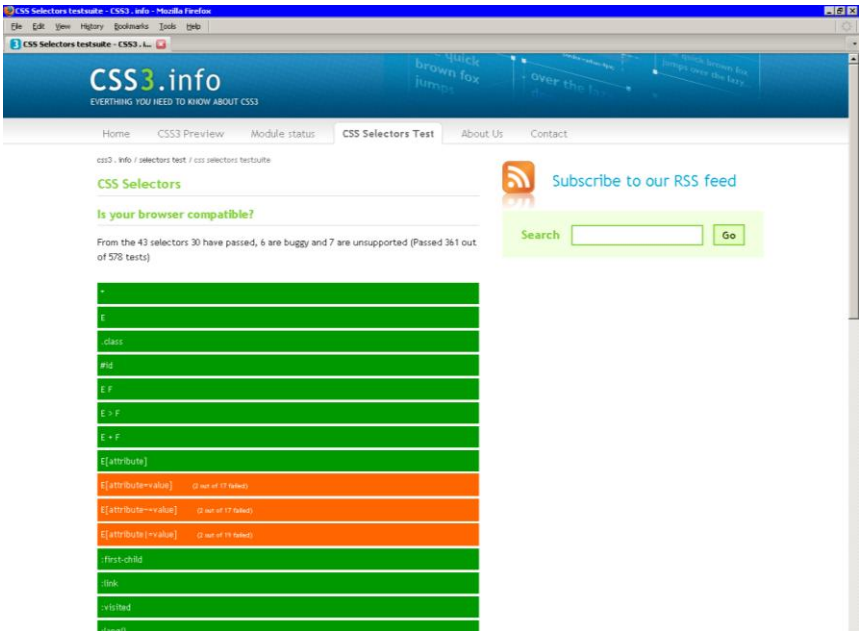
- `http://jigsaw.w3.org/css-validator`,
- `http://www.htmlhelp.com/tools/csscheck`,
- `http://webdesign.about.com/od/validators/l/bl_validation.htm#csvalidator`

Na teh straneh lahko vnesemo naslov dokumenta, ki vsebuje datoteko CSS, vnesemo vsebino kaskadne slogovne predloge ali pa podamo lokalno pot do datoteke CSS. Poženemo preverjanje in izpiše se, če je kaskadni slog pripravljen v skladu s specifikacijo.



Slika 1.4 –1: Preverjanje datoteke `stil.css` na `http://jigsaw.w3.org/css-validator`

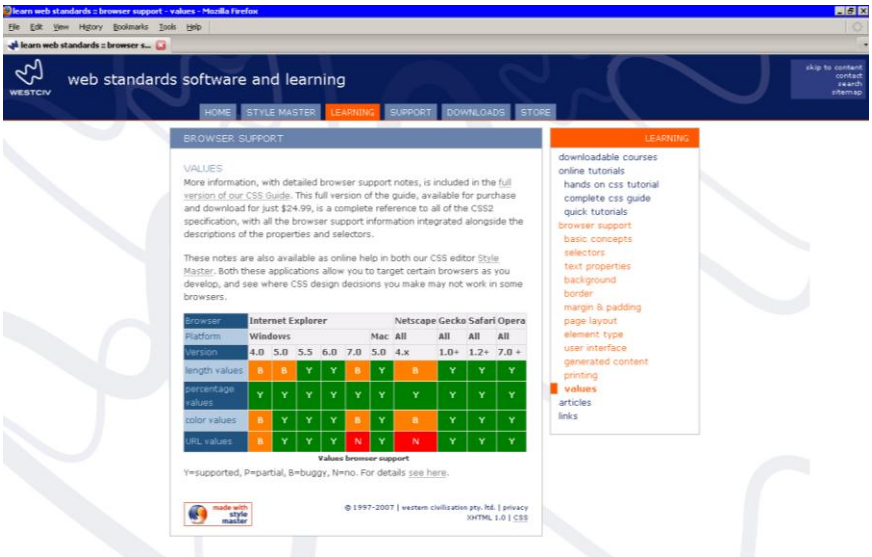
Zanimivo orodje najdemo tudi na internetni strani `http://www.css3.info/selectors-test/test.html`. Pravila oblikovanja v kaskadnih slogovnih predlogah namreč lahko zapišemo na najrazličnejše načine. Nekatere bomo predstavili v razdellku 2 ZAPIS SLOGOVNEGA PRAVILA na strani 15, ko bomo govorili o selektorjih. Na omenjeni internetni strani lahko preverimo, kako je s podporo različnim oblikam selektorjev v pregledovalniku, s katerim dostopamo do te strani.



Slika 1.4 –2: Podpora uporabe selektorjev (Firefox 2.0.0.4)

Prav tako lahko najdemo kar nekaj internetnih strani, ki vsebujejo preglednice, ki povedo katere slogovne učinke podpirajo določeni pregledovalniki:

- `http://www.w3schools.com/css/css_reference.asp`,
- `http://www.blooberry.com/indexdot/css/supportkey/syntax.htm`,
- `http://www.westciv.com/style_master/academy/browser_support/index.html`.



Slika 1.4 –3: Preglednica podpore pregledovalnikov določenim lastnostim

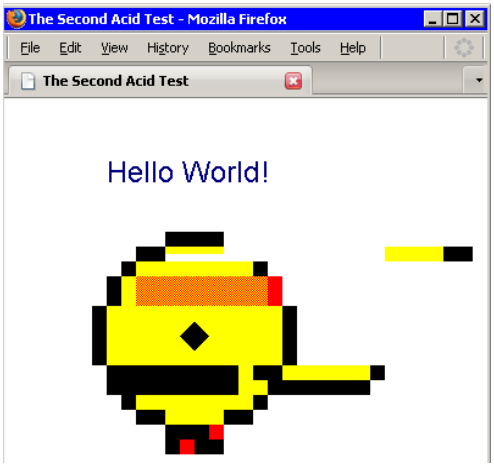
Na naslovu `http://www.webstandards.org/action/acid2` najdemo test Acid2. Z njim lahko preverimo, če pregledovalnik podpira priporočilo CSS1. Test je spletna stran, ki je narejena z

nekoliko kompleksnejšo kaskadno slogovno predlogo. Test preverja nekaj pomembnejših značilnosti oblikovanja. Če pregledovalnik test uspešno opravi, se na levi strani nariše rumen smeško. Če pa pregledovanik določenih slogov ne podpira oziroma jih ne prikazuje kot določa priporočilo CSS1, bo smeško popačen, morda prikazan v napačni barvi .... Pri testiranju pregledovalnika moramo imeti nastavljene privzete nastavitve, saj vsako spreminjanje velikosti pisave, povečevanje strani in druge nastavitve lahko vplivajo na samo izvedbo testa.

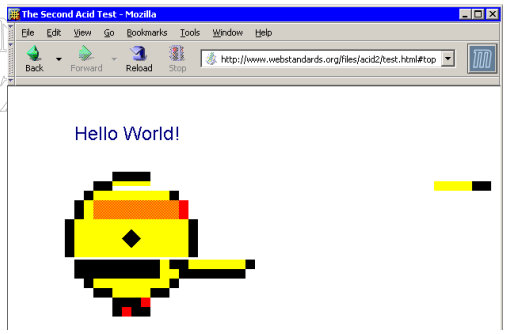
Slika 1.4 –4: Primer uspešno opravljenega testa s pregledovalnikom Opera 9.21

Na naslednjih treh slikah vidimo nekaj primerov testiranja pregledovalnikov, ki standarda CSS1 ne podpirajo v celoti.

Slika 1.4 –5: Neuspešno opravljen test (Internet Explorer 6)

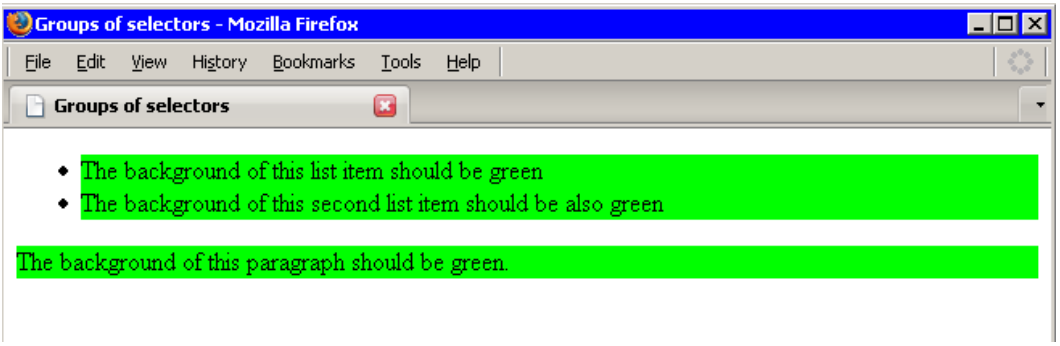


Slika 1.4 –6: Neuspešno opravljen test (Firefox 2.0.0.4)

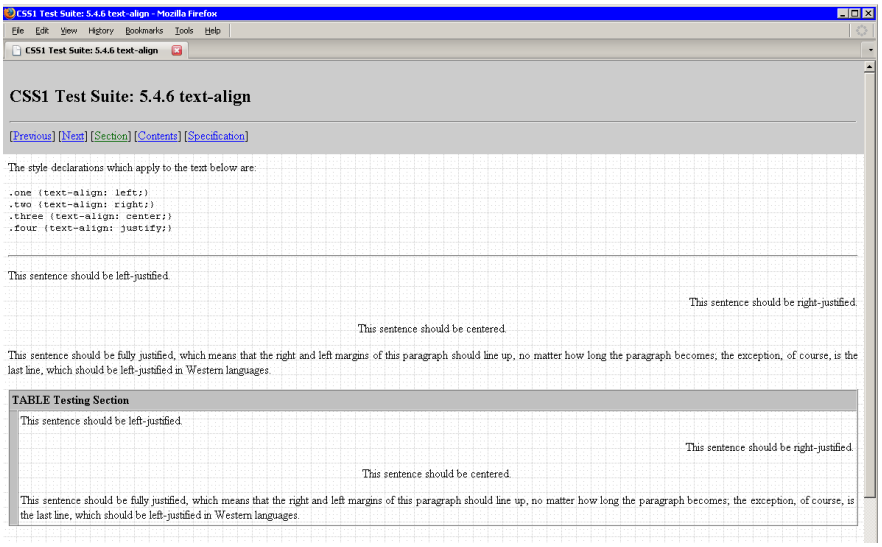


Slika 1.4 –7: Neuspešno opravljen test (Mozilla1.7.12)

Na naslovu `http://www.w3.org/Style/CSS/Test/CSS1/current/` najdemo različne teste, s katerimi lahko preverimo, v kolikšni meri naš pregledovalnik podpira priporočilo CSS1. Na naslovu `http://web5.w3.org/Style/CSS/Test/CSS2.1/current/` so testi skladnosti s priporočilom CSS2.1. Na voljo je tudi že spletna stran, ki kaže kako se pregledovalnik "ujame" s prihajajočim priporočilom CSS3. Spletno stran najdemo na naslovu `http://www.w3.org/Style/CSS/Test/CSS3/Selectors/current/html/index.html`.



Slika 1.4 –8: Primer preverjanja, če združevanje selektorjev deluje tako, kot ga predpisuje CSS3



Slika 1.4 –9: Preverjanje, ali poravnavanje besedila deluje v skladu s CSS1

V naslednjem razdelku si bomo podrobneje pogledali, kako elementom (značkam) spletne strani lahko definiramo slogovne učinke.

2. ZAPIS SLOGOVNEGA PRAVILA

Slogovno pravilo je najmanjša enota CSS, ki določa, kateremu delu spletne strani bo dodan en ali več slogovnih učinkov. Kaskadna slogovna predloga je sestavljena iz enega ali več slogovnih pravil, ki definirajo izgled posameznih elementov na internetni strani. Slogovno pravilo je sestavljeno iz dveh delov: selektorja in deklaracije. V pravilu najprej zapišemo selektor, s katerim povemo, kateremu elementu predpisujemo slogovni učinek. Znotraj zavitih oklepajev sledi deklaracija, ki določa, kako bo selektor prikazan. V deklaraciji najprej zapišemo lastnost (na primer `font-style`), nato dvopičje, ki mu sledi vrednost (na primer `italic`). Deklaracija lahko vsebuje eno ali več lastnosti, skupaj s pripadajočimi vrednostmi. Posamezne slogovne učinke (lastnosti) ločimo s podpičjem[3]. Lastnosti, ki so na voljo in njihove pripadajoče vrednosti so definirane v priporočilih CSS.

ZAPIS CSS:

```
selektor {
 lastnost1: vrednost1;
 lastnost2: vrednost2;
}
```

V kaskadni slogovni predlogi lahko tudi pišemo komentarje. Komentarje uporabljamo za lažje razumevanje kode in so namenjeni bralcu kode. Pregledovalnik jih v celoti prezre. Komentar se začne z znakoma `/*` in konča z znakoma `*/`. Komentarjev ne smemo gnezditi. Če na primer napišemo `/*To /* je */ gnezden komentar*/` bo pregledovalnik kot komentar vzel del, ki je med prvim parom kombinacije znakov `/*` in `*/`. V našem primeru bo torej komentar "`To /* je `", kar pa je narobe.

ZAPIS CSS:

```
selektor { /* To je komentar */
 lastnost1: vrednost1;
/*
  To je še en komentar,
  ki sega čez več vrstic
*/
 lastnost2: vrednost2;
}
```

V naslednjem podrazdelku si bomo podrobneje pogledali selektorje.

2.1. SELEKTORJI

Selektor je lahko katerikoli element, uporabljen v dokumentu HTML. Pogledali bomo naslednje selektorje:

- značka HTML – na primer `div` (glej razdelek 2.1.1 Značka na strani 16)
- razred – na primer `.naslov` (glej razdelek 2.1.2 Razred in atribut class na strani 16)
- značka z razredom – na primer `div.meni` (glej razdelek 2.1.2 Razred in atribut class na strani 16)
- identifikator – na primer `#natisni` (glej razdelek 2.1.3 Identifikator in atribut id na strani 18)
- značka z identifikatorjem – na primer `div#tiskane` (glej razdelek 2.1.3 Identifikator in atribut id na strani 18)
- gnezdene značke – na primer `ul li` (glej razdelek 2.1.4 Gnezdene značke na strani 20)

[3] Zadnjega slogovnega učinka ni potrebno zaključiti s podpičjem, je pa priporočljivo. Lahko se zgodi, da pri dodajanju novega učinka predhodnega pozabimo zaključiti z dvopičjem.

- psevdo razredi – na primer `a:link` (glej razdelek 7.3 OBLIKOVANJE POVEZAV na strani 48)

2.1.1 Značka

Najpreprostejši selektor je značka HTML. Navedemo jo brez oklepajev <> (na primer `div`).

ZAPIS CSS:

```
značka {
 lastnost: vrednost;
}
```

Če torej to značko uporabimo v dokumentu HTML

ZAPIS HTML:

```
<značka>tekst</značka>
```

bo vsebina značke oblikovana v skladu z navedenimi slogovnimi učinki. Oglejmo si primer.

PRIMER CSS 2.1.1 –1: Odstavki naj imajo besedilo rdeče barve

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

S tem pravilom smo določili, da bo besedilo v odstavkih (značka `p`) rdeče barve (`color: red`). Denimo, da smo ta slog uporabili za prikaz dela besedila, kot ga prikazuje naslednji primer

PRIMER HTML 2.1.1 –1: Uporaba pravila v dokumentu HTML

```
<p>tekst1</p>
<div>tekst2</div>
<p>tekst3</p>
```

Ker smo s pravilom določili, da je besedilo v odstavkih rdeče barve, bosta `tekst1` in `tekst3` rdeče barve. Ker pa za značko `div` ni določenega slogovnega učinka, bo `tekst2` privzete barve, torej črne.

S tem, da slogovne učinke določimo samo značkam iz HTML, včasih ne moremo doseči željenega slogovnega učinka ali pa bi bil zapis nepregleden in po nepotrebnem zapleten. Predpostavimo, da želimo v prvem odstavku imeti besedilo modre barve, v drugem pa zelene barve. To lahko najpreprostejše dosežemo tako, da slogovne učinke s pomočjo atributov `class` in `id` predpišemo razredom in identifikatorjem, ki jih nato uporabimo v dokumentu HTML. Kako to dosežemo, si bomo pogledali v nadaljevanju.

2.1.2 Razred in atribut class

Razredi se uporabljajo za določanje istih slogovnih učinkov večim značkam HTML hkrati. Nov razredni slog izdelamo tako, da za piko navedemo ime razreda.

ZAPIS CSS:

```
.ime_razreda {
 lastnost: vrednost;
}
```

V dokumentu HTML razred uporabimo s pomočjo atributa `class`, ki ga lahko uporabimo v vseh značkah znotraj značke `body`.

ZAPIS HTML:

```
<značka class="ime_razreda">tekst</značka>
```

Značka bo oblikovana v skladu z navedenimi slogovnimi učinki. Če določena značka določenega slogovnega učinka ne podpira, ga pregledovalnik prezre.

Oglejmo si primer.

PRIMER CSS 2.1.2–1: Izdelava razreda

```
.modro {
 color: blue;
}
```

S pravilom smo vsem elementom, ki pripadajo razredu `modro`, določili, da besedilo prikažejo v modri barvi. Uporabimo ta razred v HTML

PRIMER HTML 2.1.2–1: Uporaba pravila v dokumentu HTML

```
<p class="modro">tekst1</p>
<div class="modro">tekst2</div>
<p class="zeleno">tekst3</p>
```

S pravilom CSS smo vsem elementom v HTML, ki pripadajo razredu `modro`, določili, da besedilo prikažejo v modri barvi. To pomeni, da bosta `tekst1` in `tekst2` modre barve. Ker razredu `zeleno` nismo določili slogovnega učinka, bo `tekst3` privzete barve, torej črne. Če pa v datoteko CSS dodamo še razred `zeleno`

PRIMER CSS 2.1.2–2: Dodan razred `.zeleno`

```
.modro {
 color: blue;
}

.zeleno {
 color: green;
}
```

bodo potem imeli vsi elementi, ki pripadajo temu razredu, besedilo zelene barve. Torej bo v našem primeru `tekst3` zelene barve.

Delovanje razreda lahko omejimo samo na posamezno značko. Tej s tem lahko spremenimo obstoječ slogovni učinek. V pravilu CSS najprej zapišemo značko, nato znak `.` (pika) in ime razreda.

ZAPIS CSS:

```
značka.ime_razreda {
 lastnost: vrednost;
}
```

V dokumentu HTML razred zopet uporabimo s pomočjo atributa `class`, vendar to smemo storiti le na znački, ki smo ji določili slogovni učinek.

Oglejmo si primer.

PRIMER CSS 2.1.2–3: Razred v povezavi z značko

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

S tem pravilom smo določili, da imajo vsi odstavki, ki pripadajo razredu `modro` (selektor `p.modro`), besedilo modre barve (`color: blue`).
Torej v PRIMER HTML 2.1.2–1 bo `tekst1` modre barve, `tekst2` in `tekst3` pa črne barve. `Tekst3` je črn, saj razreda zeleno ni. Črna barva `tekst2` pa je rezultat napačne uporabe razreda `modro`. Ker smo delovanje razreda `modro` omejili le na značko p, bo `tekst2` znotraj značke `div` privzete barve.

Če v datoteko CSS dodamo še

PRIMER CSS 2.1.2–4: Dodan razred `modro` za značko `div`

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

div.modro {
 background-color: blue;
}
```

in prikažemo isto datoteko, kot je navedena v PRIMER HTML 2.1.2–1, bo `tekst1` modre barve, `tekst2` črne barve na modrem ozadju, `tekst3` pa črne barve.
Več o samem vrstnem redu, prekrivanju in združevanju slogov si bomo ogledali v razdelku 2.2 ZAPIS SLOGOV, VRSTNI RED, PREKRIVANJE IN ZDRUŽEVANJE na strani 21.

2.1.3 Identifikator in atribut id

Z identifikatorjem lahko enolično določimo element na spletni strani. Identifikator se od razreda razlikuje v tem, da ga lahko uporabimo na enem samem elementu in to samo enkrat na spletni strani. Uporaben je za določevanje slogov tistih elementov, ki se pojavijo na vsaki spletni strani le enkrat (na primer za prikaz menija, naslova, lastnosti pri tiskanju, itn.). Uporaben je tudi zato, ker ima višjo prioriteto kot razred (več o tem v razdelku 4 DEDOVANJE IN KASKADNOST na strani 31). Identifikatorje lahko uporabljamo tudi v JavaScriptu za dostopanje do točno določenega elementa.

Identifikator izdelamo tako, da za # (višajem) navedemo ime identifikatorja.

ZAPIS CSS:

```
#ime_identifikatorja {
 lastnost: vrednost;
}
```

Če identifikator uporabimo v dokumentu HTML

ZAPIS HTML:

```
<značka id="ime_identifikatorja">tekst</značka>
```

bo značka oblikovana v skladu z navedenimi slogovnimi učinki.

Oglejmo si primer.

PRIMER CSS 2.1.3–1: Izdelava identifikatorja

```
#pomembno {
 color: red;
}
```

V dokumentu HTML identifikator uporabimo s pomočjo atributa `id`, ki ga lahko uporabimo v vseh značkah znotraj značke `body`. Vendar v istem dokumentu HTML identifikator lahko uporabimo natanko enkrat.

PRIMER HTML 2.1.3–1: Uporaba identifikatorja v dokumentu HTML

```
<p id="pomembno">tekst1</p>
```

S slogovnim pravilom smo znački, ki ima identifikator `pomembno`, določili rdečo barvo besedila.

PRIMER HTML 2.1.3–2: Napačna uporaba identifikatorja

```
<p id="pomembno">tekst1</p>
<div>tekst2</div>
<p id="pomembno">tekst3</p>
```

Zgornji primer ne ustreza priporočilom CSS, ker smo v istem dokumentu HTML isti identifikator uporabili dvakrat.

Tako kot razred, lahko tudi identifikator omejimo na posamezno značko. V pravilu CSS najprej zapišemo značko, nato pa znak # (višaj), ki mu sledi ime identifikatorja.

ZAPIS CSS:

```
značka#ime_identifikatorja {
 lastnost: vrednost;
}
```

V dokumentu HTML identifikator uporabimo s pomočjo atributa `id`. Kot pri razredu lahko to storimo le na znački, ki smo ji določili slogovni učinek. Pravilo o enkratnosti uporabe ostaja, zato v samem dokumentu lahko identifikator uporabimo natanko enkrat.

ZAPIS HTML:

```
<značka id="ime_identifikatorja">tekst</značka>
```

Tudi tu si oglejmo primer.

PRIMER CSS 2.1.3–3: Izdelava značke z razredom

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

Slogovni učinek velja za značko `p`, ki ima identifikator `pomembno`.

PRIMER HTML 2.1.3–3: Uporaba pravila v dokumentu HTML


```
<p id="pomembno">tekst</p>
```

2.1.4 Gnezdene značke

Značkam lahko določimo sloge tudi glede na njihovo hierarhično strukturo, natančneje na osnovi njihovih potomcev. Potomec značke je značka, ki se nahaja znotraj te značke (na primer v `<div><h1>naslov</h1>tekst</div>` je značka `h1` potomec značke `div`).
Na naslednji strani si poglejmo, katere značke so potomci

PRIMER HTML 2.1.4–1: Potomci na strani HTML

stran.html

```
<html>
 <head>
   <title>Naslov strani</title>
 </head>

 <body>
   <h1>Naslov</h1>

   <ul>
     <li><a href="stran.html">Povezava</a></li>
   </ul>

   <img src="slika.gif">
 </body>
</html>
```

Znački `h1` in `img` nimata nobenega potomca, saj znotraj značk `h1` in `img` ni nobene druge značke. Značka `li` ima enega potomca, to je značko `a`. Značka `ul` ima dva potomca in sicer znački `li` in `a`, značka `body` pa ima pet potomcev. To so značke `h1`, `ul`, `li`, `a` in `img`.

Če želimo določiti slogovne učinke za značke (oziroma natančneje selektorje), ki so na določen način gnezdene, potem navedemo značke (oziroma selektorje) glede na njihovo hierarhijo. Selektorje ločimo s presledki.

ZAPIS CSS:

```
selektor_1 selektor_2 ... selektor_n {
 lastnost: vrednost;
}
```

PRIMER CSS 2.1.4–2: Slog gnezdene značke

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

Ta slog se nanaša samo na tiste značke `a`, ki so potomci značke p. Na primer

PRIMER HTML 2.1.4–2: Zapis slogov gnezdenim značkam

```
<p>
 <h1>naslov</h1>tekst<a href="naslov.html">povezava1</a>
  <span>tekst tekst <a href="naslov.html">povezava2</a></span>
</p>
<a href="naslov.html">povezava3</a>
```

V prvih dveh primerih bi bila barva povezave (besedilo `povezava1`, oziroma `povezava2`) zelena, barva povezave `povezava3` pa ne, ker pri tej povezavi značka `a` ni potomec značke `p`.

Ta način zapisa je uporaben, ko želimo spremeniti slogovne učinke značkam, ki imajo točno določeno hierarhično strukturo. Z njimi lahko prekrijemo tudi slogovne učinke preprostim (manj specifičnim) selektorjem. Več o tem si lahko preberemo v razdelku 4 DEDOVANJE IN KASKADNOST na strani 31.

Zapisa ne smemo zamenjati s podobnim

ZAPIS CSS:

```
selektor_1, selektor_2, ..., selektor_n {
 lastnost: vrednost;
}
```

PRIMER CSS 2.1.4–3: Slog naštetih značk

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

kjer več selektorjem hkrati določimo enake slogovne učinke (glej stran 24).

2.2. ZAPIS SLOGOV, VRSTNI RED, PREKRIVANJE IN ZDRUŽEVANJE

Sama oblika zapisa slogov ni določena. Uporabljamo lahko poljubno število presledkov in praznih vrstic. Vsi dodatni beli znaki (presledki, skoki v novo vrsto, tabulatorji) so prezrti, podobno kot v jeziku HTML.
Dobro pa je, če se pri pisanju držimo določenega stila pisanja. Sama zapisujem pravila in deklaracije drugo pod drugo, deklaracije zamaknem v desno, med pravili pa izpustim prazno vrstico.

PRIMER CSS 2.2–1: Zapis dveh pravil, ki imata enak selektor (značko)

```
p {
 color: red;
}

p {
 text-align: center;
}
```

S prvim pravilom smo določili, da je besedilo v vseh odstavkih rdeče barve, z drugim pa smo dodatno določili še da je v odstavkih uporabljena sredinska poravnava.

Ker se pravili nanašata na isti selektor (značko), lahko ti dve pravili združimo.

PRIMER CSS 2.2–1: Krajši zapis pravila

```
p {
 color: red;
 text-align: center;
}
```

S tem smo znački `p` določili isti slog, le zapis je krajši.

Če lastnosti priredimo več alternativnih vrednosti, jih ločimo z vejico. Pregledovalnik bo ob prikazu v seznamu izbral prvo, ki jo prepozna.

ZAPIS CSS:

```
selektor {
 lastnost: vrednost_1, vrednost_2, ..., vrednost_n;
}
```

PRIMER CSS 2.2–2: Zapis alternativnih vrednosti

```
p.besedilo {
 font-family: Verdana, Geneva, Arial, Helvetica, Sans-Serif;
}
```

Pregledovalnik bo torej za prikaz besedila v odstavku (selektor `p.besedilo`) poskusil uporabiti pisavo Verdana. Če je ne pozna, bo poskusil uporabiti pisavo Geneva. Če tudi te ni na računalniku, je na vrsti pisava Arial ....

Spremembe lastnosti značk veljajo, dokler jih ne prekrijemo z drugimi (več o tem v razdelku 4 DEDOVANJE IN KASKADNOST na strani 31). Če slogovna predloga vsebuje več enakih selektorjev z enakimi lastnostmi, ki pa imajo različne vrednosti, zadnja napisana prekrije vse predhodnje.

PRIMER CSS 2.2–3: Prekrivanje slogovnih učinkov

```
p {
 color: red;
 text-align: center;
}

p {
 color: green;
}
```

S prvim pravilom smo določili, da bo besedilo v vseh odstavkih rdeče barve (`color: red`). Ker pa smo z naslednjim pravilom prejšnjo vrednost prekrili, bo besedilo v odstavkih zelene barve (`color: green`). Na poravnavo teksta pa drugo pravilo ni vplivalo (saj te lastnosti ni spreminjalo), torej bodo vsi odstavki imeli sredinsko poravnano besedilo.

V kodi HTML lahko posamezno značko oblikujemo s pomočjo večih razredov. To dosežemo tako, da atributu `class` priredimo več imen razredov, ki so med seboj ločeni s presledki.

ZAPIS CSS:

```
.ime_razreda1 {
 lastnost1: vrednost1;
}

.ime_razreda2 {
 lastnost2: vrednost2;
}

.ime_razreda3 {
 lastnost3: vrednost3;
}
```

ZAPIS HTML:


```
...
<značka class="ime_razreda1 ime_razreda2 ime_razreda3">
 tekst
</značka>
...
```

Poglejmo si primer.

PRIMER CSS 2.2–4: Izdelava večih razredov

```
.rumenoBesedilo {
 color: yellow;
}

.velikostPisave {
 font-size: 12px;
}

.crnoOzadje {
 background-color: black;
}
```

PRIMER HTML 2.2–4: Uporaba večih razredov

```
...
<p class="rumenoBesedilo velikostPisave crnoOzadje">
 tekst
</p>
...
```

Besedilo odstavka bo rumene barve (`color: yellow`), pisava bo 12px (`font-size: 12px`), odstavek bo imel črno ozadje (`background-color: black`).

Če omejimo delovanje razreda na posamezno značko, lahko na isti znački uporabimo več razredov. Vendar v zapisu na posamezni znački lahko navedemo samo en razred naenkrat.

ZAPIS CSS:

```
značka.ime_razreda1 {
 lastnost1: vrednost1;
}

značka.ime_razreda2 {
 lastnost2: vrednost2;
}

značka.ime_razreda3 {
 lastnost3: vrednost3;
}
```

Če želimo isti slogovni učinek izdelati za različne značke, imamo več možnosti. Lahko navedemo vsako značko posebej in ji določimo slogovne učinke, lahko izdelamo nov razredni slog (ki ga kasneje uporabimo na značkah) ali pa pravila združimo.

PRIMER CSS 2.2–5: Isti slogovni učinek za različne značke

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

```
h1 {
 color: blue;
}
```

Ker imata selektorja isti deklaraciji, lahko pravili združimo. Več selektorjev z istimi lastnostmi in vrednostmi združimo tako, da selektorje ločimo z vejico.

ZAPIS CSS:

```
selektor_1, selektor_2, ..., selektor_n {
 lastnost_1: vrednost_1;
 lastnost_2: vrednost_2;
 ...
 lastnost_n: vrednost_n;
}
```

PRIMER CSS 2.2–6: Združeni pravili

```
p.modra, h1 {
 color: blue;
}
```

Tako smo obema selektorjema hkrati določili slog. Slogovni učinek bi lahko tudi določili z razredom in potem ta razred uporabili v obeh značkah v datoteki HTML. Vendar bi morali to seveda napisati pri vsaki znački `p` in `h1` posebej.

PRIMER CSS 2.2–7: Določevanje slogovnega učinka z razredom

```
.modra {
 color: blue;
}
```

V pravilu lahko več selektorjev, ki imajo enake deklaracije združimo ali pa jih zapišemo ločeno.

PRIMER CSS 2.2–8: Ločen zapis pravil

```
p#meni {
 color: red;
 text-align: center;
 font-family: cursive;
}

h1 {
 color: red;
 text-align: center;
}
```

Vse besedilo v odstavkih, ki imajo `id meni` (`p#meni`) je ležeče (`font-family: cursive`), rdeče barve in sredinsko poravnano. Vsi glavni naslovi (značka `h1`) imajo besedilo rdeče barve, ki je sredinsko poravnano.
Ker imata dva selektorja dva slogovna učinka (`color: red` in `text-align: center`) enaka, lahko ti deklaraciji združimo. Preostalo deklaracijo pa zapišemo posebej.

PRIMER CSS 2.2–9: Združene enake deklaracije

```
p#meni, h1 {
 color: red;
```

```
 text-align: center;
}

p#meni {
 font-family: cursive;
}
```

S prvim pravilom smo odstavkom, ki imajo `id meni` in naslovom prvega reda določili sredinsko poravnano besedilo rdeče barve. Z drugim pravilom pa smo odstavkom, ki imajo `id meni`, dodatno določili, da je besedilo ležeče. Zapis sloga je spremenjen, medtem ko je slogovni učinek ostal enak kot v prejšnjem primeru.

Če torej povzamemo:
Sloge lahko definiramo na več načinov. Pri prvem načinu enostavno spremenimo oziroma določimo slogovne učinke posamezne značke iz jezika HTML. Ti učinki potem veljajo za vse tovrstne značke v celotni datoteki HTML. Pri drugem načinu definiramo določen razred, kamor shranimo oblikovne nastavitve. Te razrede potem po potrebi uporabljamo pri različnih značkah. Pri tretjem načinu uporabimo identifikator, kamor shranimo oblikovne nastavitve. Ta identifikator lahko uporabimo natanko enkrat v istem dokumentu HTML. Sloge lahko določimo tudi na osnovi hierarhične strukture elementov, te pa nato uporabimo na točno določenih gnezdenih elementih.

3. VSTAVLJANJE SLOGOV V DOKUMENT

V dokument HTML lahko kaskadno slogovno predlogo vstavimo na tri načine. Pri prvem načinu definiramo sloge v glavi dokumenta na enega izmed predhodno opisanih načinov. Te sloge nato uporabljamo v samem telesu dokumenta HTML. Pri drugem načinu sloge definiramo sproti, znotraj značk, kot njihove slogovne učinke. Pri tretjem načinu pa definicije slogov pripravimo v samostojni zunanji datoteki. Če želimo uporabljati sloge definirane v zunanjih datotekah, moramo te datoteke uvoziti v sam dokument HTML.
Vse tri načine lahko uporabimo v istem dokumentu HTML hkrati. Vendar se zaradi številnih prednosti predvsem uporablja zadnji način, to je uvoz datoteke CSS v dokument HTML. Vseeno pa si oglejmo vse tri načine.

3.1. VSTAVLJANJE SLOGOV ZNOTRAJ ZNAČKE HTML

Slogovne učinke lahko dodelimo poljubni znački HTML kar med samo vsebino strani v HTML. Znački dodamo atribut `style`, ki ji priredimo željene slogovne vrednosti (`<p style="font-style: italic">`).
Znački lahko dodelimo eno ali več pravil, ki jih ločimo s podpičjem. Če znački dodelimo le en slogovni učinek, podpičje ni obvezno. Ta oblika se uporablja zelo redko, saj definicija sloga velja le za značko, kjer je slog definiran. Uporabimo jo le takrat, če želimo določen slogovni učinek uporabiti le na enem mestu. Pa še takrat je bolje, če definiramo ustrezen razred (v glavi ali v samostojni datoteki) in na tem mestu le uporabimo razred.
Zanimivo je, da tovrstni način zapisovanja slogov zelo uporabljajo različni programi, ki avtomatsko pretvarjajo dokumente v obliko HTML (na primer različice Worda pred 2007). Posledica tega je izjemno nepregledna koda HTML, saj ima praktično vsaka značka atribut `style` z ustrezno definicijo sloga.

ZAPIS HTML:

```
<html>
 <head>
   ...
 </head>

 <body>
   ...
   <značka style="lastnost1: vrednost; lastnost1: vrednost;...">  ...
   </značka>
 </body>
</html>
```

PRIMER HTML 3.1–1: Določitev sloga znotraj značke HTML

stran.html

```
<html>
 <head>
   ...
 </head>

 <body>
   ...
   <p style="font-style: italic">
     ...
   </p>
   ...
 </body>
</html>
```

3.2. DEFINIRANJE SLOGOV V GLAVI DOKUMENTA

Sloge lahko definiramo v glavi spletnega dokumenta (v delu znotraj značke `<head>`). Definicijo sloga napovemo z značko `style`. Ta ima lastnost `type`, ki natančneje določa tip besedila, ki ga bomo vstavili. pri definiciji slogov vedno uporabimo vrednost `text/css`, ki določa, da je vstavljeno besedilo tekstovno besedilo tipa css. Značka `style` lahko vsebuje tudi atribut `media`, vendar je neobvezen. Z njim lahko definiramo za katere izhodne naprave – medije (na primer zaslon, tiskalnik, LCD projektor, terminal, televizijo, ročno napravo itn.) bo veljala slogovna predloga. Privzeta vrednost je namenjena prikazovanju na zaslonu (`media="screen"`). Če želimo vstaviti slog, ki se bo uporabil le pri tiskanju, napišemo `media="print"`. Kadar pa želimo, da je slog uporabljen za vse medije, napišemo `media="all"`. Če naj slog velja za prikazovanje na zaslonu in za tiskanje, za ostale tipe izhodnih naprav pa ne, to dosežemo tako, da izhodne naprave zapišemo eno za drugo, ločene z vejico (na primer `media="screen, print"`). V glavi dokumenta HTML imamo lahko več značk `<style>`. V eni lahko na primer določimo sloge za prikaz na zaslonu, v drugi pa za tiskanje. Lahko bi tudi zapisali več značk `<style>` za eno in isto izhodno napravo, vendar je bolj smiselno združiti definicije znotraj ene značke `<style>`.

Nekateri starejši pregledovalniki ne znajo uporabljati slogov. Obravnavajo jih kot navadno besedilo, kar nam seveda ni všeč. Ta problem rešimo tako, da notranjost značke `style` (torej same zapise CSS) enostavno napišemo kot komentar v jeziku HTML. Definicije slogov nato zapišemo znotraj komentarja HTML (`<!- ->`). Starejši pregledovalniki, ki značke `style` ne poznajo, ta del obravnavajo kot komentar in definicije slogov preskočijo.

Tako ustvarjena kaskadna slogovna predloga pripada le tistemu dokumentu, v katerem je napisana. Zato moramo na vsaki novi strani, kjer želimo tako predlogo uporabljati, prekopirati celotno defincijo slogovne predloge. Če se kasneje odločimo za spremembo določenega sloga, moramo popravke narediti v vseh datotekah HTML, kamor smo to predlogo prenesli. Zato se tudi takega načina definiranja slogov praviloma izogibamo. Ima pa ta način določeno prednost. Vse skupaj (tako HTML kot CSS) je združeno v eni datoteki. Tako se nam ne more zgoditi, da bi pri kopiranju ali prestavljanju spletne strani kam drugam na datoteko s slogom pozabili.

ZAPIS HTML:

```
<html>
 <head>
   ...
   <style type="text/css" media="izhodna naprava">
     <!--
       selektor1 {
         lastnost1: vrednost1;
       }

       selektor2 {
         lastnost2: vrednost2;
       }
       ...
     -->
   </style>
 </head>

 <body>
   ...
 </body>
</html>
```

Oglejmo si enostaven primer.

PRIMER HTML 3.2–1: Določitev sloga v glavi dokumenta

stran.html

```
<html>
 <head>
   ...
  <style type="text/css">
   <!--
     p {
       font-style: italic;
     }
     p.modro {
       color: blue;
     }
   -->
  </style>
 </head>

 <body>
  ...
 </body>
</html>
```

Ker atribut `media` nismo uporabili, bo tekst v odstavkih ležeč le na zaslonu. Če to stran izpišemo na tiskalnik, se bodo za izpis besedila v odstavku uporabili v pregledovalniku nastavljeni slogovni učinki.

3.3. VSTAVLJANJE SLOGOV Z UVOZOM IZ ZUNANJE DATOTEKE

Sloge lahko definiramo tudi v zunanji datoteki. Tej damo končnico css (na primer `stil.css`). To datoteko potem uvozimo v dokument HTML tako, da v glavo dokumenta (znotraj značke `<head>`) zapišemo značko `link`. Ta značka ima lahko naslednje atribute:

- `rel` – določa tip povezave. Ko uvažamo datoteko s slogom, zapišemo `rel="stylesheet"`
- `href` – določa mesto, kjer je slogovna datoteka. Tako na primer napišemo `href="stil.css"`, če je datoteka `stil.css` v istem imeniku kot datoteka HTML. Pri tem lahko oprabimo poljuben zapis URL, torej je načeloma lahko datoteka tudi na drugem računalniku.
- `type` – glej razdelek 3.2 DEFINIRANJE SLOGOV V GLAVI DOKUMENTA na strani 27
- `media` – glej razdelek 3.2 DEFINIRANJE SLOGOV V GLAVI DOKUMENTA na strani 27

Oglejmo si torej shemo dokumenta v HTML, kjer uporabimo vstavljanje slogov iz zunanje datoteke.

ZAPIS HTML:

```
<html>
 <head>
   <link rel="stylesheet" href="URL_slogovne_datoteke"
         type="text/css" media="izhodna naprava">
 </head>

 <body>
   ...
 </body>
</html>
```

V sami datoteki s slogom le navedemo vse definicije slogov na že razložen način.

Prednost takega definiranja slogov je stroga ločitev oblikovanja od vsebine. Datoteka s slogovnimi predlogami, ki opisuje izgled spletne strani, je ločena od dokumenta HTML, ki določa vsebino spletne strani. Isto slogovno predlogo lahko uporabimo na večih straneh. V primeru spremembe sloga to opravimo na enem samem mestu.

Zunanjo slogovno predlogo lahko napišemo v poljubnem urejevalniku besedil (na primer s programom Notepad, WordPad, TextPad ...). Obstajajo tudi programi, ki so specializirani (ali pa vsebujejo poseben modul namenjen temu) za pisanje in preverjanje slogovnih predlog. Poznamo tako tekstovne programe, kot sta na primer programa TopStyle in Style Master, kot tudi programe, ki imajo grafični uporabniški vmesnik (na primer Microsoft FrontPage, Macromedia Dreamweaver).

Zaradi številnih prednosti, ki jih ima strogo ločevanje oblikovanja od besedila, je zelo priporočljivo uporabljati zapis slogov v zunanji datoteki. Ta način bomo uporabljali tudi v nadaljevanju diplomske naloge.

3.3.1 UVOZ SLOGOV

Datoteke z definicijami slogov lahko tudi uvažamo, bodisi v drugo slogovno datoteko, bodisi v glavo dokumenta. V ta namen uporabljamo ukaz `@import`. Ukaz za uvoz moramo napisati na samem začetku slogovne predloge, tej pa lahko nato sledijo nova slogovna pravila.
Poglejmo si, kako uvozimo datoteko v glavo dokumenta v HTML.

ZAPIS HTML:

```
<html>
 <head>
   ...
   <style type="text/css" media="izhodna naprava">
     <!--
       @import url(ime_datoteke.css);

       selektor1 {
         lastnost1: vrednost;
       }

       selektor2 {
         lastnost2: vrednost;
       }
       ...
     -->
   </style>
 </head>

 <body>
   ...
 </body>
</html>
```

Če pa uvažamo slogovno datoteko v drugo slogovno datoteko, napišemo

ZAPIS CSS:

```
@import url(ime_datoteke.css);
...
selektor1 {
 lastnost1: vrednost;
}

selektor2 {
 lastnost2: vrednost;
}
...
```

Kaskadnim slogovnim predlogam, ki jih uvažamo, lahko tudi določimo katerim izhodnim napravam (medijem) so namenjene.

ZAPIS CSS:

```
@import url(ime_datoteke.css) medij1, medij2;
...
selektor1 {
 lastnost1: vrednost;
}

selektor2 {
 lastnost2: vrednost;
}
...
```

Poglejmo si primer.

PRIMER CSS 3.3.1–1: Uvažanje kaskadne slogovne predloge

stil.css

```
@import url(nov_stil.css) print, screen;
...
p.modro {
 color: blue;
 }
 ...
```

Uvožena kaskadna predloga (`nov_stil.css`) velja v tem primeru izključno za tiskalnik (`print`) in zaslon (`screen`). Kaskadna slogovna predloga `nov_stil.css` bo uporabljena samo v primeru, če bo kaskadna slogovna predloga `stil.css` namenjena isti izhodni napravi (`print`, `screen` ali `all`), drugače ne. Recimo, da je kaskadna slogovna predloga, v katero smo uvozili `nov_stil.css`, namenjena izključno prikazu na zaslon. Potem se bodo uporabili slogovni učinki iz obeh datotek. Če pride do navzkrižja slogovnih učinkov, prevladajo zadnji napisani. Torej slogovni učinki, ki so napisani v kaskadni slogovni predlogi `stil.css` prekrijo tiste, ki so bili uvoženi. V primeru, če je kaskadna slogovna predloga `stil.css` namenjena izključno mobilnim napravam, potem se v njej zapisani slogi ne bodo uporabili pri prikazu na zaslonu. Prav tako se potem pri prikazu na zaslonu ne bodo uporabili slogovni učinki, ki so zapisani v `nov_stil.css`.

4. DEDOVANJE IN KASKADNOST

Dokument HTML je hierarhično strukturiran. Višje značke predstavljajo roditelje (starše) nižjim značkam (otrokom). Te pa so lahko roditelji drugim nižjim značkam. To je pomembno zato, ker hierarhično nižja značka (otrok) lahko podeduje določene slogovne učinke starševske značke. Kateri slogovni učinki se podedujejo, je določeno v specifikaciji CSS.

Tako lahko na primer določimo slogovne učinke znački `body`. Vsi njeni potomci (torej vse značke, saj so vse vsebovane znotraj te značke) bodo te slogovne učinke podedovali. Izjema so le učinki, ki uporabljajo lastnosti, ki se ne dedujejo. Tako se na primer lastnost ozadja elementov ne deduje. Seveda posamezne značke lahko podedovane slogovne učinke prekrijejo (nadomestijo s svojimi učinki). Podedovana lastnost torej velja le, če znački te lastnosti ne določimo posebej.

Ves čas govorimo o kaskadnih slogovnih predlogah. Kaj pa sploh pomeni kaskadnost? Kaskadnost v računalništvu običajno pomeni način tvorbe zaporedja operacij, kjer je izhod ene operacije vhodni podatek za naslednjo operacijo. Tudi pri slogih uporabljamo poseben postopek, imenovan kaskade, s katerim določamo, katera nastavitev določene oblike velja. Kaskade so množice pravil, ki pregledovalnikom povedo, kako naj združijo sloge spletne strani, pregledovalnika (privzete nastavitve) in uporabnika (zahteve).

Prvo pravilo kaskad pove, kakšna je prioriteta prej naštetih načinov vstavljanja slogov. Kot smo že omenili, vse te tri načine lahko tudi kombiniramo.
Slog, določen znotraj značke, ima najvišjo prioriteto. Sledi mu slog, določen znotraj glave dokumenta HTML in šele nato slog, določen nazunanji slogovni datoteki. Če pa značka nima definiranega sloga, pregledovalnik uporabi privzete slogovne učinke prikaza značke (kot jih je predvidel programer, ki je pisal program za pregledovalnik).

PRIMER CSS 4–1: Kaskadna slogovna predloga na zunanji datoteki

stil.css

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

PRIMER HTML 4–1: Preprosta stran HTML z različnimi vstavljenimi slogi

stran.html

```
<html>
 <head>
   <title>Naslov strani</title>
   <link rel="stylesheet" href="stil.css" type="text/css">

   <style type="text/css">
     <!--
       p {
         color: red;
       }
     -->
   </style>
 </head>

 <body>
   <h1>Naslov</h1>
   <p style="color: green">To je prvi odstavek.</p>
   <p>To je drugi odstavek.</p>

 </body>
</html>
```

Ker ima slog znotraj značke najvišjo prioriteto, bo besedilo v prvem odstavku zelene barve (`color: green`). Besedilo drugega odstavka bo rdeče, kot določa slog v glavi datoteke HTML. Sloga za odstavek, določenega na zunanji datoteki `stil.css` v tem primeru ne moremo uporabiti. Naslov (značka `h1`) bo prikazan tako, kot je predvidel programer pregledovalnika, saj ga nismo nikjer določili.

Elementu lahko slogovne učinke določimo na različne načine (glej razdelek 2 ZAPIS SLOGOVNEGA PRAVILA na strani 15). Zato lahko pride do navzkrižja slogovnih učinkov.

PRIMER CSS 4–2: Kaskadna slogovna predloga na zunanji datoteki

stil.css

```
p {
 color: blue;
}

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

PRIMER HTML 4–2: Preprosta stran HTML z različnimi vstavljenimi slogi

stran.html

```
<html>
 <head>
   <title>Naslov strani</title>
   <link rel="stylesheet" href="stil.css" type="text/css">
 </head>

 <body>
   <h1>Naslov</h1>
   <p class="zeleno">To je prvi odstavek.</p>
   <p>To je drugi odstavek.</p>

 </body>
</html>
```

V tem primeru so odstavki zelene ali modre barve. Za drugi odstavek je jasno, da bo besedilo modre barve, saj nanj vpliva le prvi slog. Kakšne barve pa bo v prvem odstavku? Načeloma se obe pravili nanašata nanj. Velja, da bolj specifični selektorji prekrijejo manj specifične selektorje. V tem primeru ima pravilo `p.zeleno` višjo prioriteto kot `p`, zato bo odstavek zelene barve.

Opredelimo pravilo o specifičnosti natančneje. Dejansko gre za posebni primer pravila, ki določa prioriteto slogov določenih elementu. Prioriteta slogov se določi na osnovi štirih združenih števil **abcd** na naslednji način:

- **vrednost a**

če je slog znotraj značke (atribut `style="stil"`) je a 1, drugače 0

- **vrednost b**

b je število identifikatorjev (#) v selektorju

- **vrednost c**

c je število razredov (.) in pseudo (nepravih) razredov v selektorju

- **vednost d**

d je število značk v selektorju

Ta štiri števila združimo in nastalo število predstavlja vrednost prioritete. Če na določenem mestu na značko vpliva več slogov, ki si nasprotujejo, uporabimo tistega z višjo prioriteto.
V PRIMER CSS 4–2 se prioriteta določi na naslednji način:

Prvi selektor (`p`) je določen iz ene značke, zato bodo vse vrednosti razen `d 0`. Vrednost `d` ustreza številu značk v selektorju. V našem primeru je samo ena, zato je `d` enak `1`. Za prvo pravilo `p` torej velja `a=0`, `b=0`, `c=0`, `d=1`. Sledi, da ima selektor `p` prioriteto 1.
Drugi selektor (`p.zeleno`) je določen iz ene značke in enega razreda, zato bosta vrednosti `c` in `d` enaki `1`. Ostali vrednosti sta `0`. Pri `p.zeleno` je torej `a=0`, `b=0`, `c=1`, `d=1`. Torej ima ta selektor prioriteto 11.
Iz napisanega je očitno, da ima drugo pravilo višjo prioriteto kot prvo. To pomeni, da ne glede na to, kako si pravili sledita, ima bolj specifično vedno višjo prioriteto.

Oglejmo si še en zgled.

PRIMER CSS 4–3: Kaskadna slogovna predloga na zunanji datoteki

stil.css

```
p {
 color: blue;
}

p.zeleno {
 color: green;
}

p#rumeno {
 color: yellow;
}
```

PRIMER HTML 4–3: Preprosta stran HTML z različnimi vstavljenimi slogi

stran.html

```
<html>
 <head>
   <title>Naslov strani</title>
   <link rel="stylesheet" href="stil.css" type="text/css">
 </head>

 <body>
   <h1>Naslov</h1>
   <p id="rumeno" class="zeleno">
     To je <span class="krepko">prvi</span> odstavek.
   </p>
   <p>To je drugi odstavek.</p>

 </body>
</html>
```

Kakšne barve bo besedilo `To je prvi odstavek.`? "Za prevlado se borijo" trije selektorji, `p`, `p.zeleno` in `p#rumeno`. Za prva dva smo že ugotovili, da imata prioriteto `1` in `11`. Kaj pa tretji? Premislek pove, da za `p#rumeno` velja `a=0`, `b=1`, `c=0`, `d=1`. Ta selektor ima torej prioriteto `101`.

V tem primeru ima višjo prioriteto razred z identifikatorjem, iz česar sledi, da bo besedilo rumene barve.

Kaj pa, če v slogovno predlogo dodamo še naslednja sloga

PRIMER CSS 4–3: Kaskadna slogovna predloga na zunanji datoteki

stil.css

```
...
p.zeleno .krepko {
 color: green;
}
```


DIPLOMSKA NALOGA:
FAKULTETA ZA MATEMATIKO IN FIZIKO

```
.krepko {
 color: yellow;
}
```


Kakšne barve bo sedaj tekst `prvi`? Tu bi za njegovo oblikovanje lahko poskrbela dva sloga. Prvi je splošni razred `.krepko`, drugi pa `.krepko` kot sin sloga `p.zeleno`. Ne pozabimo, da v tem primeru uporabljamo gnezdenje značk in ne naštevanje (glej 2.1.4 Gnezdene značke na strani 20). Določimo prioritete. `p.zeleno .krepko` ima prioriteto 201 (a=0, b=2, c=0, d=1), medtem ko ima `.krepko` prioriteto 10 (`a=0, b=2, c=1, d=0`)
Vidimo, da ima selektor `p.zeleno .krepko` višjo prioriteto, torej bo tekst `prvi` zelene barve.

Če slogovna predloga vsebuje več enakih slogov z enako prioriteto, prevladuje zadnje napisano.

Ta način določanja prioritet pa lahko spremenimo z ukazom `!important`. Definicija sloga, opremljena s tem ukazom, ima najvišjo prioriteto.

PRIMER CSS 4–5: Uporaba `!important`

stil.css

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

Ponovno si oglejmo PRIMER HTML 4–1, le da definicijo sloga v kaskadni slogovni predlogi (torej tisto, ki jo prikazuje PRIMER CSS 4–1) opremimo z ukazom `!important`. Sedaj ima za značko `p` najvišjo prioriteto slogovni učinek z ukazom `!important`. Zato bo besedilo v odstavku modre barve.

Če uporabimo več ukazov `!important` in pride do navzkrižja slogovnih učinkov, so si vse deklaracije opremljene z `!important` enakovredne in prevlada zadnja napisana.

PRIMER HTML 4–5: Uporaba večih ukazov `!important`

stran.html

```
<html>
 <head>
   <title>Naslov strani</title>
   <link rel="stylesheet" href="stil.css" type="text/css">

   <style type="text/css">
     <!--
       p {
         color: red !important;
       }

       P {
         color: yellow;
       }
     -->
   </style>
 </head>

 <body>
   <h1>Naslov</h1>
   <p style="color: green">To je prvi odstavek.</p>
   <p>To je drugi odstavek.</p>

 </body>
</html>
```

V tem primeru imamo dva slogovna učinka opremljena z ukazom `!important` (enega v datoteki `stil.css` in drugega v glavi datoteke `stran.html`). Prevladal bo tisti, ki je zadnji napisan, torej v tem primeru, bo besedilo v obeh odstavkih rdeče barve (zadnji `!important`).

Če želimo zagotoviti uporabo določenega slogovnega učinka, uporabljamo torej čim bolj specifične selektorje ter ukaz `!important`.

5. VREDNOSTI IN ENOTE

V tem razdelku si bomo pogledali na kakšen način lahko izražamo pomembnejše vrednosti, kot so na primer vrednosti za dolžino, barve, poti do datotek ....

5.1. VREDNOSTI IN MERE

Za določevanje velikosti, dolžine in širine lahko uporabljamo relativne ali absolutne vrednosti.
Relativne vrednosti so:

- `px` (točka)
- `em` (velikost trenutne pisave)
- `ex` (polovica višine črke x)
- `%` – odstotki

Absolutne vrednosti so:

- `in` (palec)
- `pt` (točka)
- `pc` (pica=12 točk)
- `mm` (milimeter)
- `cm` (centimeter).

Vrednost in enoto pišemo skupaj. Vrednost `0` lahko zapišemo brez enote.

PRIMER CSS 5.1–1: Zapis dolžine

```
p.pomembno {
 height: 5px;
}

div.besedilo {
 height: 1.2cm;
}
```

V CSS lahko uporabljamo tako cela, kot tudi decimalna števila. Pri slednjih uporabljamo decimalno piko, ne glede na lokalne nastavitve operacijskega sistema. Števila so lahko pozitivna ali negativna. Negativnih vrednosti se raje izogibajmo, saj si s tem lahko otežimo samo oblikovanje strani. Nekatere lastnosti jih tudi ne podpirajo.

PRIMER CSS 5.1–2: Zapis števila

```
p.pomembno {
 padding: 1.3px;
}

div.besedilo {
 font-size: 12;
}
```

V specifikaciji CSS je določeno, katere lastnosti se lahko izražajo v odstotkih in na osnovi katerih vrednosti se izračunajo (na primer lastnost `font-size` je odvisna od velikosti pisave višje (starševske) značke). Vrednost in znak za odstotek pišemo skupaj.

PRIMER CSS 5.1–3: Zapis odstotkov

```
p.pomembno {
 font-size: 120%;
}
```

Velikost pisave se izračuna v odvisnosti od velikosti pisave starševske značke. Če je velikost pisave v višji (starševski) znački 10px, se bo v zgornjem primeru pisava povečala za 20%. Pomeni, da bo velikosti 12px. Pri tem je še potrebno omeniti, da potomci te značke podedujejo izračunano vrednost in ne odstotkov.

5.2. BARVA

Barvo lahko določimo na več načinov:

- opisno z angleškim izrazom za barvo
  CSS 2.1 pozna 17 standardnih barv: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, read, silver, teal, white in yellow. Večino pregledovalnikov podpira tudi uporabo nekaterih nestandardnih barv kot so pink, cyan in druge. A ker se na to ne moremo zanesti, je bolje uporabljati barvni model RGB.
- z barvnim modelom RGB (red, green, blue) oziroma s šestnajstiško kodo barve.
  Barvo določimo glede na mešanje treh osnovnih barv: rdeče, zelene in modre. Za določitev barve z RGB modelom moramo vedeti koliko rdeče, zelene in modre je vsebovano v določeni barvi. Vrednosti barv lahko zapišemo v procentih, desetiški ali v šestnajstiški vrednosti.
  Zapis:
  - šestmestno število oblike #rrggbb, kjer rr, gg in bb pomenijo šestnajstiški zapis vrednosti barv,
  - tromestno število #rgb, kjer kjer r, g in b označujejo eno šestnajstiško vrednost barv. Ta način se uporablja zelo redko in ga večina knjig o CSS niti ne omenja
  - rgb(r%, g%, b%), kjer r, g in b predstavljajo vrednosti od 0% do 100% ustrezne barve
  - rgb(r, g, b), kjer r, g in b predstavljajo vrednosti od 0 do 255 ustrezne barve

Za lažje določanje barv si lahko pomagamo z barvnimi lestvicami, ki jih najdemo na svetovnem spletu. Navedimo nekaj internetnih strani z barvnimi kodami:

- `http://www.w3schools.com/css/css_colornames.asp`,
- `http://html-color-codes.com/`,
- `http://www.webmonkey.com/reference/color_codes/`.

PRIMER CSS 5.2–1: Različni zapisi za rdečo barvo

```
p {
 color: red;
}

p {
 color: #FF0000;
}

p {
 color: #F00;
}

p {
 color: rgb(100%, 0%, 0%);
}

p {
 color: rgb(255, 0, 0);
}
```

5.3. URL

Naslove spletnih strani, datotek s slikami, glasbo in drugih datotek določimo na podlagi `url` tega vira. Naslov napišemo s pomočjo funkcije `url("naslov")`. Naslov je lahko glede na datoteko, v kateri je zapisan (znotraj CSS ali HTML):

- absoluten – na primer `url("http://www.naslov.si/meni.gif")`
- relativen – na primer `url("slika.gif")`, `url("../ozadje.gif")`

Namesto dvojnih narekovajev lahko uporabimo enojne ali pa jih celo izpustimo.

PRIMER CSS 5.3–1: Zapis naslovov

```
p {
 background-image: url("slika.gif");
}
```

6. VRSTE ELEMENTOV

Na osnovi načina prikazovanja vsebine delimo značke v dve skupini: vrstične in bločne značke. Tip značke pove, kako bo pregledovalnik prikazal vsebino znotraj določene značke. Bločni element je ločen od ostale vsebine tako, da se začne in konča z novo vrstico. Bločni ukazi so: `blockquote`, `br`, `div`, `h1–h6`, `ol`, `p`, `table` .... Bločni elementi se prikazujejo eden za drugim vzdolž strani. Če niso preoblikovani s slogi oziroma drugimi značkami, se raztezajo preko celotne strani oziroma vrstice.

Vrstični element se prikaže tam, kjer je bila značka vstavljena. Vrstične značke so: `a`, `b`, `em`, `img`, `input`, `span`, ... Vrstični elementi si v vrstici sledijo eden za drugim. Ko je vrstice konec, se nadaljujejo v novi vrstici.

Znotraj vrstičnih elementov lahko gnezdimo le vrstične elemente in besedilo, znotraj bločnih pa imamo lahko tako bločne kot tudi vrstične elemente in besedilo. V naslednjem poglavju bomo opisali, kako z ustrezno izbiro bločnih in vrstičnih elementov lahko oblikujemo spletno stran, v razdelku 7.3 OBLIKOVANJE POVEZAV na strani 48 pa si bomo pogledali, na kakšen način lahko znački spremenimo način prikazovanja.

7. OBLIKOVANJE ELEMENTOV

V tem razdelku si bomo pogledali, kako z uporabo lastnosti CSS postopoma oblikujemo spletno stran. Ob tem bomo razložili, kako vplivamo na prikaz posameznih elementov, ki sestavljajo to spletno stran. Isto stran bomo oblikovali skozi vse podrazdelke tako, da bo razvidno, kako s slogi vplivamo na prikaz posameznih elementov strani.
V dokumentu HTML si bomo pri oblikovanju pomagali z značkama `span` in `div`. Razlika med njima je, da je `span` vrstični element, `div` pa bločni element. Oba elementa sta nevtralna in brez tega, da bi jima določili slogovni učinek, nimata vpliva na stran. Ker je `span` vrstični element, lahko znotraj njega gnezdimo le vrstične elemente in besedilo, znotraj `div` pa imamo lahko tako bločne kot vrstične elemente in seveda besedilo.

Kreirajmo stran HTML v katero bomo zaenkrat vključili prazno kaskadno slogovno predlogo `stil.css`. Pri tem bomo pomembnejše sestavne dele takoj označili z značkami `div`.

PRIMER HTML 7–1: Stran, ki jo bomo oblikovali

stran.html

```
<html>
 <head>
  <title>Oblikovanje strani</title>
  <meta http-equiv="Content-Type" content="text/html;
  charset=windows-1250" />
  <link rel="stylesheet" href="stil.css" type="text/css">
 </head>

 <body>
  <div>
   <h1><img src="naslov.gif"></h1>
   <div>
    <ul>
     <li><a href="stran1.html">Prva povezava</a></li>
     <li><a href="stran2.html">Druga povezava</a></li>
     <li><a href="stran3.html">Tretja povezava</a></li>
     <li><a href="stran4.html">Četrta povezava</a></li>
    </ul>
   </div>

   <div>
    <div>
     <img src="slika.gif">
    </div>

    <div>
     Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
     Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
     Tekst. Tekst. Tekst. Tekst.Tekst. Tekst. Tekst.
     Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
     Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
     Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
     Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
     Tekst. Tekst. Tekst. Tekst.
    </div>

    <table>
     <tr>
      <th><br></th>
      <th>NASLOV</th>
      <th>NASLOV</th>
      <th>NASLOV</th>
     </tr>
     <tr>
      <th>NASLOV</th>
```

```
        <td>tekst</td>
        <td><br></td>
        <td>tekst</td>
      </tr>
      <tr>
        <th>NASLOV</th>
        <td>tekst</td>
        <td><br></td>
        <td><br></td>
      </tr>
    </table>

    <form action="stran.html" method="get">
      <fieldset>
        <legend>Obrazec</legend>
        <div>
          <label>Vnesi:</label>
          <input type="text" name="vnosnoPolje">
          <label>Izberi:</label>
          <select name="padajociSeznam">
           <option>prva izbira</option>
           <option>druga izbira</option>
           <option>tretja izbira</option>
          </select>
        </div>

        <div>
          <label>Izberi vrednost:</label>
          <input type="radio" name="izbira">vrednost 1
          <input type="radio" name="izbira">vrednost 2
        </div>
        <div>
          <label>Izberi vrednost(i):</label>
          <input type="checkbox" name="polje1">vrednost 1
          <input type="checkbox" name="polje2">vrednost 2
          <input type="checkbox" name="polje3">vrednost 3
        </div>

        <div>
          <textarea name="tekst" rows="2" cols="55">
          </textarea>
        </div>

        <div>
          <input type="submit" name="poslji" value="Pošlji">
          <input type="reset" name="brisi" value="Izbriši">
        </div>
      </fieldset>
    </form>
  </div>

  <div>
    <a href="stran1.html"><img src="nazaj.gif"></a>
    <a href="stran2.html"><img src="prva.gif"></a>
    <a href="stran3.html"><img src="naprej.gif"></a>
  </div>
 </div>
</body>
</html>
```

Spletna stran brez slogovnih učinkov je videti, kot prikazuje Slika 7–1.

Slika 7–1: Izgled strani brez uporabe slogov

Naš cilj pa je, da bi bila stran videti, kot prikazuje Slika 7–2.

Slika 7–2: Izdelana spletna stran z uporabo kaskadnih slogovnih predlog

Najprej si bomo pogledali, kako lahko vplivamo na prikaz besedila spletne strani.

7.1. OBLIKOVANJE BESEDILA

V tem razdelku bomo spletni strani določili velikost pisave, vrsto pisave, poravnavo besedila in barvo pisave. Postopoma bomo nadgrajevali datoteko `stili.css`. Dodane vrednosti bomo vedno zapisali z rdečo barvo.

Najprej si bomo pogledali, kako lahko elementom spletne strani ali pa kar celotni spletni strani, spremenimo vrsto pisavo. Vrsto pisave izberemo z lastnostjo `font-family`. Vrednost nastavimo na željeno pisavo (na primer `font-family: Arial`). Če je ime pisave sestavljeno iz večih besed, jo zapišemo med narekovaji. Ker ni nujno, da ima vsak računalnik nameščeno zahtevano pisavo, je priporočljivo navesti več alternativnih pisav. Med seboj jih ločimo z vejico. Ponavadi na koncu navedemo še ustrezno družino pisave,. Pregledovalnik bo v seznamu pisav izbral prvo, ki jo prepozna (glej PRIMER CSS 7.1–1: Nastavitev pisave spletne strani).
Vsak pregledovalnik pozna pet družin pisav:

- `serif` v katero uvrščamo pisave `Times New Roman`, `MS Georgia`, `Garamond, ...`
- `sans-serif` kamor spadajo pisave `Arial`, `Helvetica`, `Futura`, `Gill Sans, ...`
- `cursive` – sem uvrščamo `Zapf-Chancery`, `Caflisch Script, ...`
- `fantasy,` kjer najdemo pisave kot so `Critter`, `Cottonwood, ...`
- `monospace` – sem uvrščamo `Courier`, `MS Courier New`, `Prestige, ...`

Za vsako družino ima pregledovalnik nastavljeno privzeto vrednost pisave (na primer Mozilla FireFox ima v operacijskem sistemu Windows Vista kot privzeto vrednost sans–serifne pisave pisavo Tahoma). Na zaslonu se najlažje berejo pisave iz družine `sans-serif`, zato bomo uporabili pisavo `Verdana` in alternativni vrednosti `Arial` ter `sans-serif`. Pregledovalnik bo uporabil prvo, ki jo bo poznal. V skrajnem primeru bo to privzeta pisava družine `sans-serif`. Ker želimo, da pisava velja na celotni spletni strani, bo to slogovni učinek značke `body`.

PRIMER CSS 7.1–1: Nastavitev pisave spletni strani

stil.css

```
body {
 font-family: Verdana, Arial, sans-serif;
}
```

Privzeto velikost pisave lahko spremenimo z lastnostjo `font-size`. To lahko podamo z relativnimi ali absolutnimi vrednostmi kot konstante ali pa številske vrednosti. Relativne se preračunajo glede na trenutno nastavljeno velikost pisave ali starševskega (višjega) elementa, medtem ko so absolutne vrednosti vnaprej definirane. Velikost lahko podamo z:

- **vnaprej definiranimi konstantami**
  - absolutne vrednosti
    - `xx-small` – najmanjša velikost
    - `x-small` – zelo majhna velikost
    - `small` – majhna velikost
    - `medium` – srednja velikost
    - `large` – velika velikost
    - `x-large` – zelo velika velikost
    - `xx-large` – največja velikost
  - relativne vrednosti
    - `larger` – večja velikost
    - `smaller` – manjša velikost
- **številskimi vrednostmi**
  Te smo predstavili v razdelku 5.1 VREDNOSTI IN MERE na strani 36.

Naši spletni strani določimo pisavo velikosti 15px. Uporabili bomo torej številsko absolutno vrednost. Če bi uporabili relativno določanje, bi se, glede na to, da značka body nima starševske značke, velikost preračunala glede na privzete vrednosti v pregledovalniku.

PRIMER CSS 7.1–2: Nastavitev velikosti pisave spletni strani

stil.css

```
body {
 font-family: Verdana, Arial, sans-serif;
 font-size: 15px;
}
```

Privzeto levo poravnavo besedila lahko spremenimo z lastnostjo `text-align`.

Besedilo naše spletne strani bomo poravnali obojestransko (`text-align: justify`), naslove prve kategorije pa sredinsko (`text-align: center`).

PRIMER CSS 7.1–3: Sredinsko poravnan naslov in obojestransko poravnano besedilo

stil.css

```
body {
 font-family: Verdana, Arial, sans-serif;
 font-size: 15px;
 text-align: justify;
}

h1 {
 text-align: center;
}
```

Če bi želeli besedilo poravnati v desno, bi uporabili vrednost `right`, za poravnavo v levo pa `left`.

Privzeta višina vrstic je odvisna od velikosti pisave. Včasih pa želimo zaradi boljše preglednosti razmik med vrsticami povečati. To dosežemo tako, da vrstici z lastnostjo `line-height` določimo višino.

PRIMER CSS 7.1–4: Nastavitev višine vrstice spletni strani

stil.css

```
body {
 font-family: Verdana, Arial, sans-serif;
 font-size: 15px;
 text-align: justify;
 line-height: 20px;
}

h1 {
 text-align: center;
}
```

Spletni strani oziroma njenim elementom lahko z lastnostjo `color` spremenimo privzeto barvo pisave. Ustrezne vrednosti te lastnosti smo spoznali v razdelku 5.2 BARVA na strani 37.

Spletni strani bomo določili, da ima besedilo rjave barve. Na eni od omenjenih spletnih strani ugotovimo, da je ustrezna koda za rjavo barvo `#A0522D`.

PRIMER CSS 7.1–5: Nastavitev besedila rjave barve

stil.css

```
body {
 font-family: Verdana, Arial, sans-serif;
 font-size: 15px;
 text-align: justify;
 line-height: 20px;
 color: #A0522D;
}

h1 {
 text-align: center;
}
```

Poglejmo si, kako je spletna stran videti potem, ko smo dodali omenjene slogovne učinke besedila.

Slika 7.1–5: Izgled spletne strani z oblikovanim besedilom

V naslednjem razdelku si bomo pogledali, kako spletni strani določimo ozadje.

7.2. OBLIKOVANJE OZADJA

Spletni strani oziroma elementom spletne strani lahko določimo barvo in/ali sliko, uporabljeno na ozadju. Barvo določimo z lastnostjo `background-color`, sliko pa z lastnostjo `background-image`.

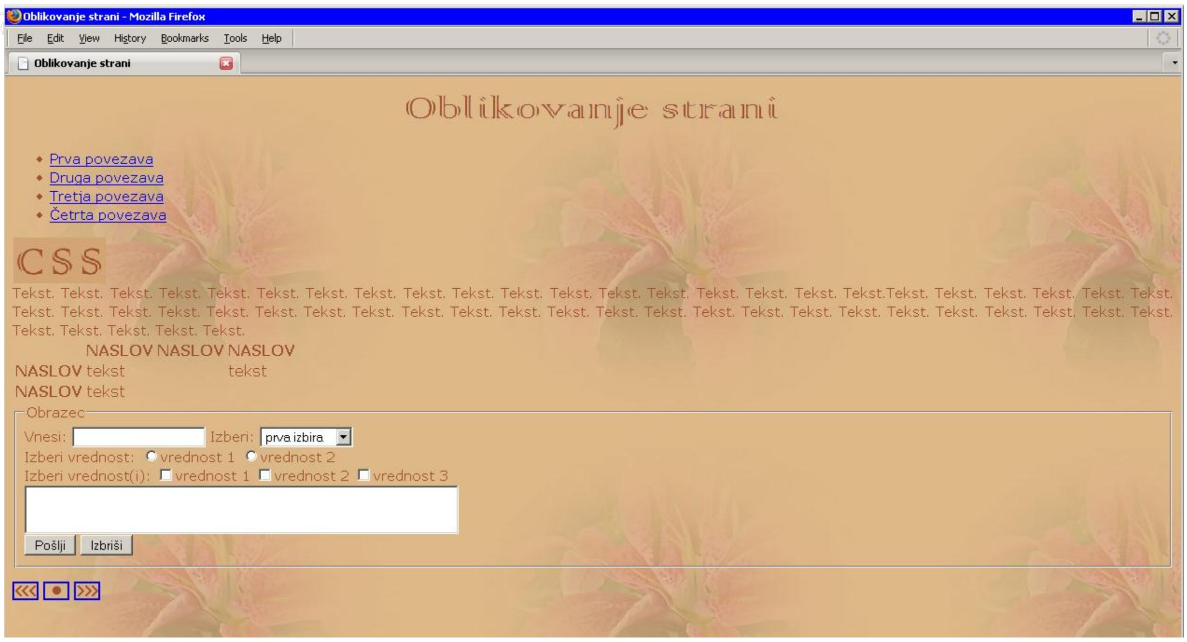
Za spletno stran bomo uporabili ozadje rjave barve ter v ozadje dodali sliko `lilija.jpg`.

PRIMER CSS 7.2–1: Nastavitev barve in slike ozadja

stil.css

```
body {
 font-size: 15px;
 font-family: Verdana, Arial, sans-serif;
 text-align: justify;
 line-height: 25px;
 color: #A0522D;
 background-color: #DEB887;
 background-image: url(lilija.jpg);
}

h1 {
 text-align: center;
}
```

Poglejmo si, kako je videti naša spletna stran z dodanim ozadjem.



Slika 7.2 – 1: Izgled spletne strani s ponavljajočo sliko

Opazimo, da se slika ponavlja. Privzeta nastavitev je namreč taka, da se slika v ozadju ponavlja v navpični in vodoravni smeri. To lahko spremenimo z lastnostjo `background-repeat`. Možne vrednosti so:

- `repeat-x` – slika se ponavlja v vodoravni smeri
- `repeat-y` – slika se ponavlja v navpični smeri
- `no-repeat` – slika se ne ponavlja
- `repeat` – slika se ponavlja v navpični in vodoravni smeri

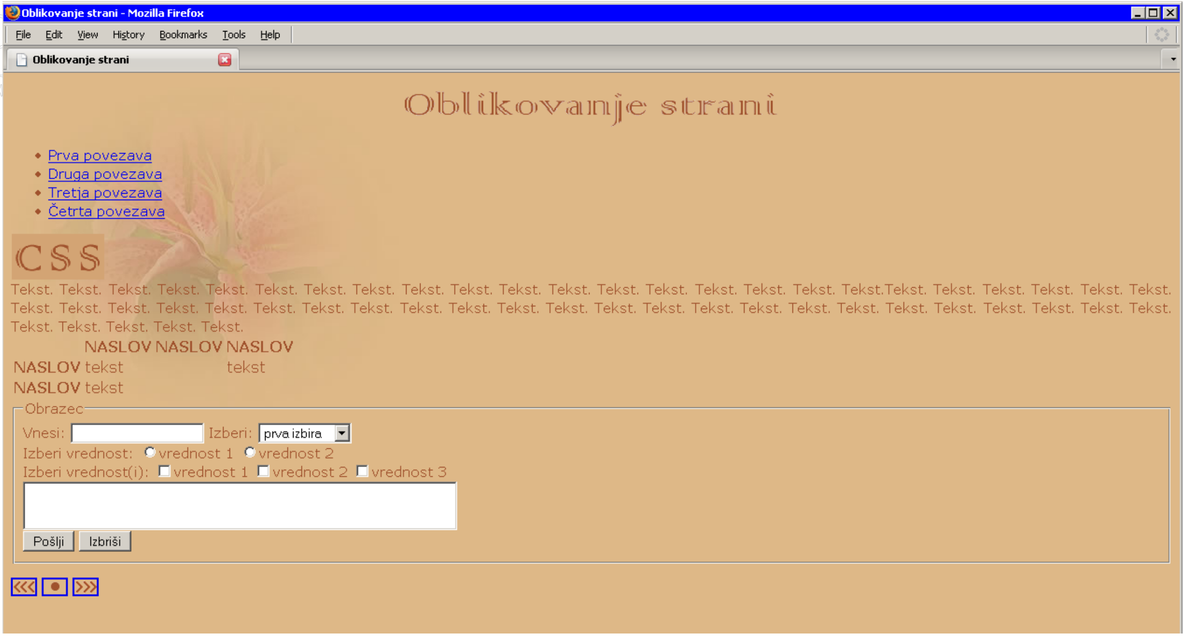
Ker ne želimo, da bi se slika ponavljala, v kaskadno slogovno predlogo dodajmo to lastnost z vrednostjo `no-repeat`.

PRIMER CSS 7.2–2: Nastavitev, da se ozadje ne ponavlja

stil.css

```
body {
 font-size: 15px;
 font-family: Verdana, Arial, sans-serif;
 text-align: justify;
 line-height: 25px;
 color: #A0522D;
 background-color: #DEB887;
 background-image: url(lilija.jpg);
 background-repeat: no-repeat;
}

h1 {
 text-align: center;
}
```

Zopet si poglejmo izgled strani.



Slika 7.2–2: Izgled strani z dodanim ozadjem

Če ne nastavimo drugače, se slika postavi v zgornji levi kot. Položaj slike spremenimo z lastnostjo `background-position`. Podajanje položaja je smiselno le v primeru, ko se slika ozadja ne ponavlja. Položaj lahko določimo z eno ali dvema vrednostima, ki sta ločeni s presledkom. Vrednosti so lahko izražene z:

- **odstotki** – Določimo odmik levega zgornjega kota slike od levega zgornjega kota okna. Če na primer uporabimo `background-position: 30% 50%` to pomeni, da je slika odmaknjena od levega roba okna za 30% in od zgornjega roba za 50% širine okna.
- **številskimi vrednostmi** – Določimo položaj levega zgornjega kota roba slike. Če na primer napišemo `background-position: 10px 40px`, pomeni, da bo slika odmaknjena od levega roba okna za `10px` in od zgornjega roba za `40px`.
- **položajem** – Namesto številskih vrednosti lahko uporabimo tudi konstante, ki določajo vodoravne in/ali navpične položaje. Vodoravni položaji so:
  - `left` – levo
  - `right` – desno
  - `center` – sredinsko

  Navpični položaji so:
  - `top` – zgoraj
  - `bottom` – spodaj
  - `center` – sredinsko

Vrednosti lahko zapišemo v poljubnem vrstnem redu. Levo oziroma desno pomeni, da smo element posatvili na levi, oziroma desni rob. Na primer `background-position: left top` pomeni isto kot `background-position: top left`. Če navedemo samo en položaj, se za drugega privzame, da je sredinski.

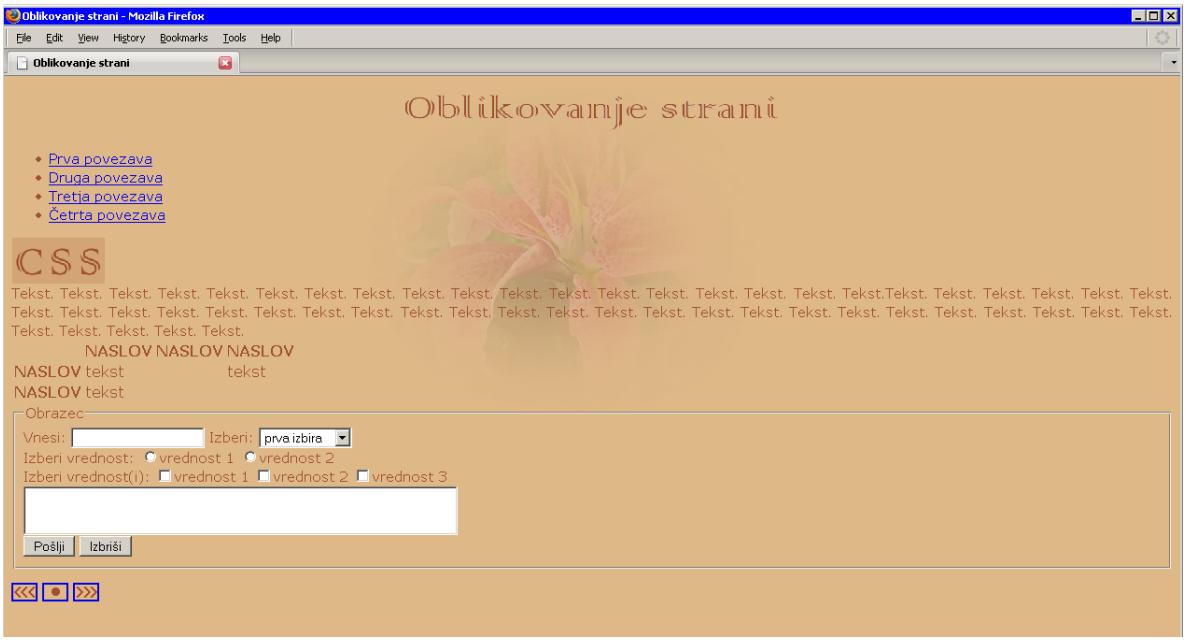
Postavimo sliko ozadja na sredino pod zgornji rob

PRIMER CSS 7.2–3: Postavitev slike ozadja zgoraj na sredino

stil.css

```
body {
 font-size: 15px;
 font-family: Verdana, Arial, sans-serif;
 text-align: justify;
 line-height: 25px;
 color: #A0522D;
 background-color: #DEB887;
 background-image: url(lilija.jpg);
 background-repeat: no-repeat;
 background-position: top center;
}

h1 {
 text-align: center;
}
```

Pa si poglejmo, kako izgleda stran z dokončno oblikovanim ozadjem.



Slika 7.2–3: Stran z dokončno oblikovanim ozadjem

Na strani opazimo, da se videz povezav ne sklada z ozadjem spletne strani. Poglejmo na kakšen način jih lahko oblikujemo.

7.3. OBLIKOVANJE POVEZAV

Tudi povezavam lahko določimo slogovne učinke. Slogovne učinke jim določimo z uporabo že vnaprej definiranih nepravih (pseudo) razredov. Ti so glede na njihova stanja štirje:

- neobiskana povezava – `a:link`
- obiskana povezava – `a:visited`
- aktivna povezava (v trenutku, ko klinemo na povezavo) – `a:active`
- ko se z miškinim kazalcem zapeljemo na povezavo – `a:hover`

Za posamezna stanja povezav bomo uporabili različne odtenke rjave barve. Slogovne učinke povezav bomo določili v novi datoteki (`povezave.css`), to pa bomo nato vključili v datoteko `stil.css`.

PRIMER CSS 7.3–1: Določitev barv povezav

povezave.css

```
a:link {
 color: #A52A2A;
}

a:active {
 color: #A0522D;
}

a:visited {
 color: #D99777;
}

a:hover {
 color:  #F5DEB3;
}
```

Povezave na spletni strani so privzeto podčrtane. Ker nam to ni všeč, bomo podčrtovanje odstranili. Podčrtava je ena od možnih tako imenovanih dekoracij besedila. Nanje vplivamo z lastnostjo `text-decoration`. Uporabimo lahko naslednje vrednosti:

- `none` – brez dekoracije
- `underline` – besedilo je podčrtano
- `overline` – besedilo je nadčrtano
- `line-through` – besedilo je prečrtano
- `blink` – besedilo utripa

Mi bomo uporabili **`none`.** Ker želimo, da to velja za poljubno stanje povezave, bomo lastnost pripisali kar znački a.

PRIMER CSS 7.3–2: Povezave niso več podčrtane

povezave.css

```
a {
 text-decoration: none;
}

a:link {
 color: #A52A2A;
}

a:active {
 color: #A0522D;
}

a:visited {
 color: #D99777;
}

a:hover {
 color:  #F5DEB3;
}
```

Ker smo povezavam odstranili dekoracijo, jih poudarimo. To dosežemo s tako imenovano težo pisave, ki jo nadzira lastnost `font-weight`. Ta ima vrednosti med vrednosti med `100` in vključno `900`. Vrednost `100` pomeni najtanjšo pisavo, vrednost `900` pa najdebelejšo pisavo. Privzeta pisava ima težo `400`. Lahko pa uporabimo tudi konstante:

- `normal` – normalna pisava (teža `400`)
- `bold` – krepka pisava (teža `700`)
- `bolder` – bolj krepka pisava (za `100` več od trenutne teže)

- `lighter` – tanjša pisava (`100` manj od trenutne teže)

Seveda v primeru, da bi `bolder` pomenilo, da ima pisava težo `1000`, nastavitev nima vpliva in pisava ostane teže `900`.

PRIMER CSS 7.3–3: Nastavitev krepke pisave povezavam

povezave.css

```
a {
 text-decoration: none;
 font-weight: bold;
}

a:link {
 color: #A52A2A;
}

a:active {
 color: #A0522D;
}

a:visited {
 color: #D99777;
}

a:hover {
 color:  #F5DEB3;
}
```

Prav tako bi radi, da seznam povezav nima oznak ter povezave postavili na sredino. Oznako seznama lahko določimo oziroma spremenimo z lastnostjo `list-style-type`. Pomembnejše vrednosti so:

- `none` – brez oznake
- `disc` – poln krogec – ●
- `circle` – prazen krogec – ○
- `square` – kvadrat – ▪
- `decimal` – številke – 1, 2, 3, ...
- `lower-roman` – male rimske številke – i, ii, iii, ...
- `lower-alpha` – male črke angleške abecede – a, b, c, ...
- `upper-latin` – velike latinske črke – A, B, C, ...

PRIMER CSS 7.3–4: Odstranitev oznak in sredinska poravnava povezav

povezave.css

```
#meni, #noga {
 text-align: center;
}

a {
 text-decoration: none;
 font-weight: bold;
}

a:link {
 color: #A52A2A;
}

a:active {
 color: #A0522D;
}

a:visited {
 color: #D99777;
}
```

```
a:hover {
 color:  #F5DEB3;
}

ul {
 list-style-type: none;
}
```

Da bomo povezave ustrezno poravnali, smo določili identifikatorja `meni` in `noga`. Prvega bomo uporabili za zgornje (besedilne) povezave in drugega za spodnje povezave (ki se "skrivajo" pod slikami). Slogovna učinka uporabimo tako, da v dokumentu HTML znački `div` dodamo ustrezni identifikator.

PRIMER HTML 7.3–4: V dokumentu HTML dodana identifikatorja `meni` in `noga`

stran.html

```
<html>
 <head>
   <title>Oblikovanje strani</title>
   <meta http-equiv="Content-Type" content="text/html;
   charset=windows-1250" />
   <link rel="stylesheet" href="stil.css" type="text/css">
 </head>

 <body>
   <div>
     <h1><img src="naslov.gif"></h1>
     <div id="meni">
       <ul>
         <li><a href="stran1.html">Prva povezava</a></li>
         <li><a href="stran2.html">Druga povezava</a></li>
         <li><a href="stran3.html">Tretja povezava</a></li>
         <li><a href="stran4.html">Četrta povezava</a></li>
       </ul>
     </div>

     ...

     <div id="noga">
       <a href="stran1.html"><img src="nazaj.gif"></a>
       <a href="stran2.html"><img src="prva.gif"></a>
       <a href="stran3.html"><img src="naprej.gif"></a>
     </div>
   </div>
 </body>
</html>
```

Lepše bi bilo, če bi bile povezave v isti vrstici. To lahko dosežemo, ne da bi posegali v samo strukturo dokumenta HTML. Elementu lahko spremenimo način prikaza z lastnostjo `display`. Pomembnejše vrednosti, ki jih lahko uporabimo:

- `none` – element bo neviden
- `block` – element bo prikazan kot bločni element
- `inline` – element bo prikazan kot vrstični element
- `inline-block` – element bo prikazan kot vrstično–bločni element
- `list-item` – element bo prikazan kot del seznama
- `inline-tabel` – element bo prikazan kot vrstična tabela
- `table - element` bo prikazan kot tabela
- `table-caption` – element bo prikazan kot naslov tabele
- `table-cell` – element bo prikazan kot celica tabele
- `table-row` – element bo prikazan kot vrstica tabele

V našem primeru bomo elemente, ki so del seznama (torej bomo določili slogovne učinke znački `li`), z uporabo `display: inline` prikazali kot vrstične elemente.

PRIMER CSS 7.3–5: Sprememba prikaza povezav

povezave.css

```
#meni, #noga {
 text-align: center;
}

a {
 text-decoration: none;
 font-weight: bold;
}

a:link {
 color: #A52A2A;
}

a:active {
 color: #A0522D;
}

a:visited {
 color: #D99777;
}

a:hover {
 color:  #F5DEB3;
}

ul {
 list-style-type: square;
}

li {
 display: inline;
}
```

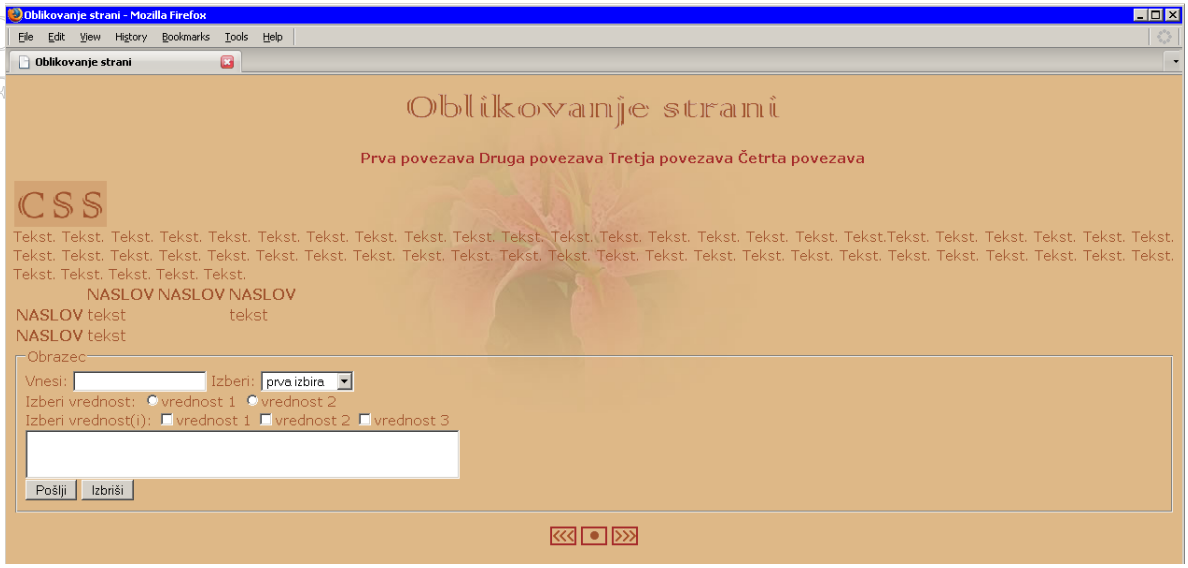
Kaskadno slogovno predlogo `povezave.css` vstavimo v predlogo `stil.css`.

PRIMER CSS 7.3–5: V predlogo `stil.css` vstavimo predlogo `povezave.css`

stil.css

```
@import url(povezave.css);

body {
 font-size: 15px;
 font-family: Verdana, Arial, sans-serif;
 text-align: justify;
 line-height: 25px;
 color: #A0522D;
 background-color: #DEB887;
 background-image: url(lilija.jpg);
 background-repeat: no-repeat;
 background-position: top center;
}

h1 {
 text-align: center;
}
```

Oglejmo si izgled strani z oblikovanimi povezavami.



Slika 7.3–5: Izgled strani z oblikovanimi povezavami

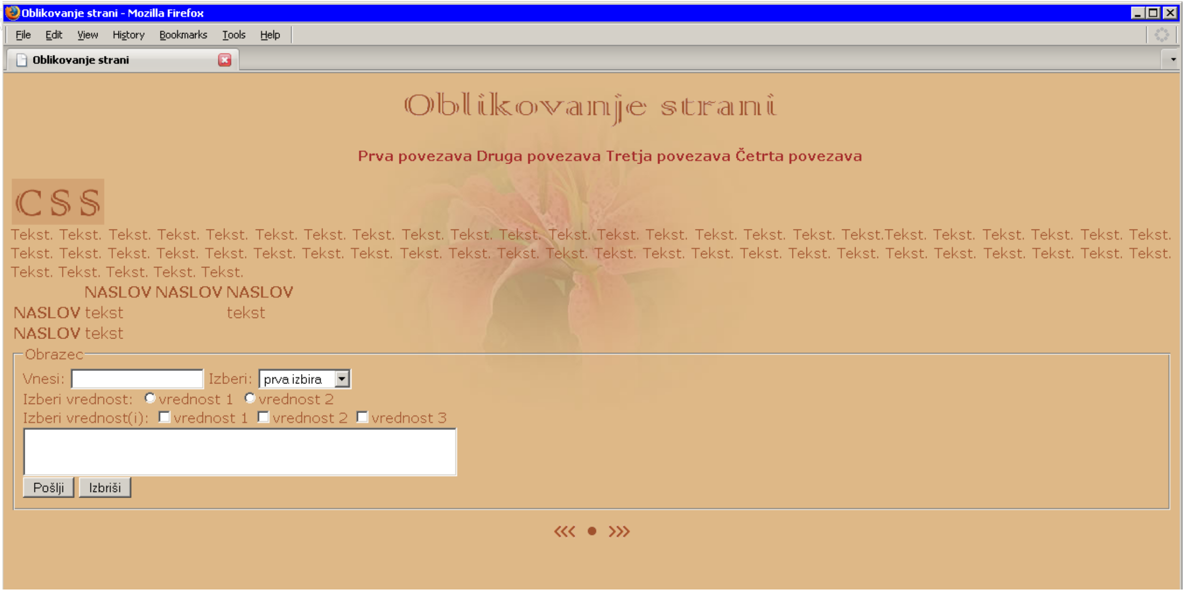
Opazimo, da imajo spodnje slike, ki so namenjene navigaciji po spletnih straneh, obrobo. Kadar sliko uporabimo za povezavo, je uporaba obrobe privzeta lastnost. Z nastavitvijo lastnosti `border` na `0` pa obrobo slikam lahko odstranimo. Več o tej lastnosti si bomo pogledali v naslednjem razdelku.

PRIMER CSS 7.3–6: Odstranitev obrob povezavam

povezave.css

```
#meni, #noga {
 text-align: center;
}

a {
 text-decoration: none;
 font-weight: bold;
}

a:link {
 color: #A52A2A;
}

a:active {
 color: #A0522D;
}

a:visited {
 color: #D99777;
}

a:hover {
 color:  #F5DEB3;
}

ul {
 list-style-type: square;
}

li {
 display: inline;
}

img {
 border: 0;
}
```

Poglejmo si izgled spletne strani.



Slika 7.3–6: Izgled strani z oblikovanimi povezavami

7.4. OBLIKOVANJE TABEL

V tem razdelku si bomo ogledali, kako bi tabeli dodali obrobo, naslovnim celicam dodali ozadje, ter določili velikost in širino celic tabele.
Tabeli bomo najprej dodali obrobo. Oglejmo si, kako to storimo.
Obrobo elementom spletnih strani določimo s tremi vrednostmi:

- **debelino obrobe** (lastnost `width`):
  - `thin` – tanka obroba
  - `medium` – srednja debela obroba
  - `thick` – debela obroba
  - številska vrednost – vrednost debeline obrobe: glej razdelek 5.1 VREDNOSTI IN MERE na strani 36
- **stilom obrobe** (lastnost `style`):
  - `none` – obrobe ni, debelina roba je 0
  - `dotted` – pikčasta obroba
  - `dashed` – črtkasta obroba
  - `solid` – neprekinjena obroba
  - `double` – dvojna neprekinjena obroba
  - `groove` – vbočena obroba
  - `ridge` – izbočena obroba
  - `inset` – vsebina znotraj obrobe je videti kot bi bila vbočena
  - `outset` – vsebina znotraj obrobe ima videz izbočenosti
- **barvo obrobe** (lastnost `color`) – za ustrezne vrednosti glej razdelek 5.2 BARVA na strani 37.

Elementu lahko določimo zgornji (`top`), desni (`right`), spodnji (`bottom`) ter levi (`left`) rob. Posamezen rob določimo na naslednji način `border-položaj-lastnost`.
Če bi na primer tabeli radi določili neprekinjen desni rob modre barve velikosti `1px`, bi to storili na naslednji način

PRIMER CSS 7.4–1: Določitev desnega roba tabele

```
table {
 border-right-width: 1px;
 border-right-color: blue;
 border-right-style: solid;
}
```

Lastnosti posamezne obrobe lahko krajše določimo tako, da uporabimo le `border-položaj` in navedemo željene lastnosti, ločene s presledkom. Prejšnji primer bi tako zapisali

PRIMER CSS 7.4–2: Krajši zapis za določitev desnega roba tabele

```
table {
 border-right: 1px solid blue;
}
```

Če želimo, da lastnost velja za vse robove v tabeli, jo določimo z `border-vrednost`. Tako na primer privzeto barvo obrobe tabele določimo z `border-color: blue`. Naslednji primer kaže, kako bi tabeli določili neprekinjeno modro obrobo velikosti `1px`.

PRIMER CSS 7.4–3: Določitev obrobe tabele

```
table {
 border-width: 1px;
 border-color: blue;
 border-style: solid;
}
```

Tudi tu lahko velikost, stil in barvo nastavimo hkrati, tako da uporabimo le lastnost `border`.

PRIMER CSS 7.4–4: Krajši zapis za določitev obrobe tabele

```
table {
 border: 1px solid blue;
}
```

Pri zapisu vrednosti lastnosti lahko uporabimo tudi več vrednosti. V tem primeru nastavitev velja za več robov. Poglejmo si podrobnosti.

Posamezno vrednost (velikost, barvo ali stil obrobe) lahko določimo z:

- eno vrednostjo. Ta potem velja za vse štiri robove, na primer `border-color: green` določi, da so vsi štirje robovi zeleni.
- dvema vrednostima. V tem primeru prva vrednost določa zgornji in spodnji rob, druga pa levega in desnega. Če torej napišemo na primer `border-color: green grey`, bosta zgornji in spodnji rob zelena, levi in desni pa siva.
- tremi vrednostmi. Tu prva vrednost določa zgornji rob, druga levi in desni rob ter tretja spodnji rob. V tabeli z nastavljeno lastnostjo `border-color: green grey blue` bo zgornji rob zelen, spodnji rob moder, levi in desni rob pa bosta siva.
- štirimi vrednostmi. Te določajo barvo vseh štirih robov. Sledijo si v smeri urinega kazalca z začetkom zgoraj. Prva vrednost torej določa zgornji rob, druga desni rob, tretja spodnji rob ter četrta levi rob. Tako na primer `border-color: green gray blue red pove`, da bo zgornji rob zelen, desni rob siv, spodnji rob moder, levi pa rdeč.

PRIMER CSS 7.4–5: Določitev obrobe tabele z dvema vrednostima

```
table {
 border-width: 1px 2px;
 border-color: blue black;
 border-style: solid dotted;
}
```

Elementu obrobo odstranimo bodisi tako, da določimo obrobo debeline `0` (glej PRIMER CSS 7.3–6) ali pa uporabimo lastnost stila none.

Sedaj opisano uporabimo na našem primeru. Določimo tabeli rjavo tanko obrobo, celicam tabele pa rjavo pikčasto obrobo. Slogovne učinke tabele bomo določili v novi datoteki `tabele.css`.

PRIMER CSS 7.4–6: Tabelo obrobimo s tanko rjavo neprekinjeno obrobo

tabele.css

```
table {
 border: thin solid #A0522D;
}

th, td {
 border: thin dotted #A0522D;
}
```

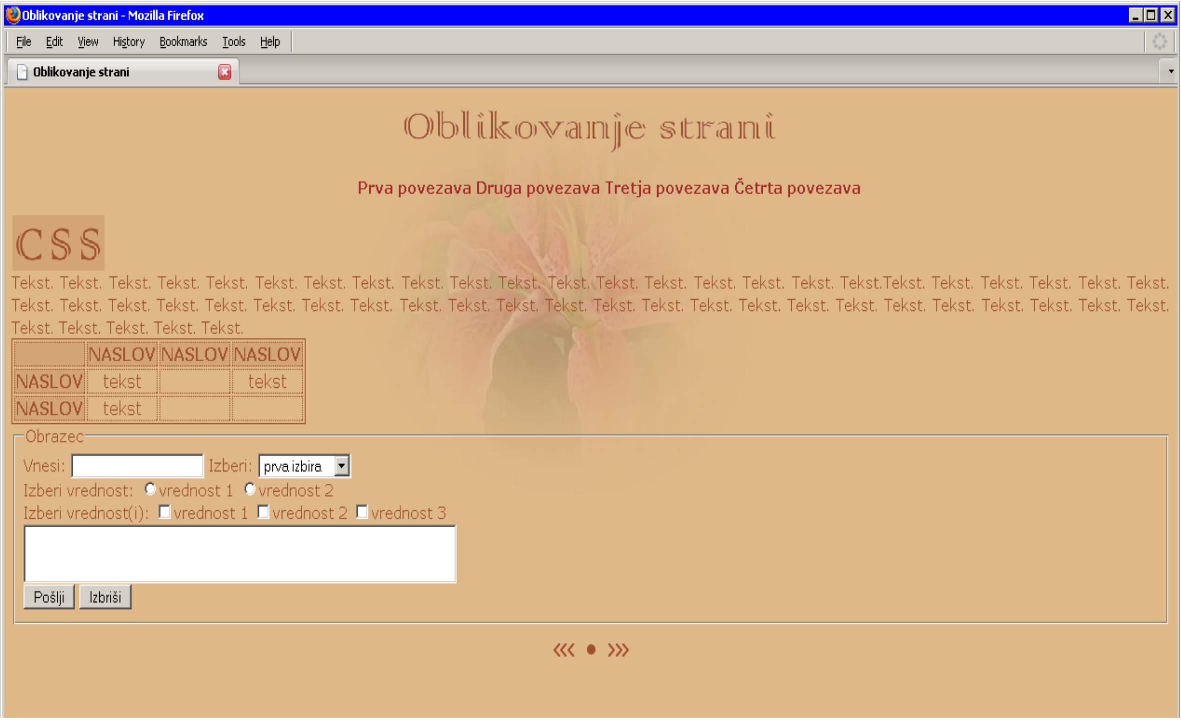
Sredinsko poravnajmo vsebino celic ter naslovnim vrsticam podamo temnejše rjavo ozadje.

PRIMER CSS 7.4–7: Sredinska poravnava celic in rjavo ozadje naslovnih vrstic tabele

tabele.css

```
table {
 border: thin solid #A0522D;
}

th, td {
 border: thin dotted #A0522D;
 text-align: center;
}

th {
 background-color: #D3A474;
}
```

Oglejmo si stran



Slika 7.4–7: Izgled strani z oblikovano tabelo

Opazimo, da so celice tabele ločene med seboj. Med njimi je tanek rob praznega prostora. Če bi želeli, bi lahko celicam spremenili način prikazovanja robov z lastnostjo `border-collapse`. Če bi se želeli omenjenega praznega prostora znebiti, torej celice združiti, bi uporabili vrednost `collapse` `(border-collapse: collapse)`, ločitev celic med seboj pa dosegli z vrednostjo `separate` (kar je tudi privzeta vrednost).

Naslovnim celicam tabele določimo še velikost z lastnostjo `width` ter višino z lastnostjo `height`.

PRIMER CSS 7.4–8: Določitev velikosti celic naslovne vrstice

```
tabele.css
table {
 border: thin solid #A0522D;
}

th, td {
 border: thin dotted #A0522D;
 text-align: center;
}

th {
 background-color: #D3A474;
 width: 90px;
 height: 30px;
}
```

Opazimo, da bi bilo lepše, če bi bila tabela in besedilo malo bolj oddaljena drug od drugega. Privzeta oddaljenost med elementi je 0. Oddaljenost med dvema elementoma lahko nastavimo v vsaki smeri posebej z:

- `margin-top` – oddaljenost od zgornjega roba elementa
- `margin-right` – oddaljenost od desnega roba elementa
- `margin-bottom` – oddaljenost od spodnjega roba elementa
- `margin-left` – oddaljenost od levega roba elementa

Tem lastnostimi podamo željeno vrednost (glej razdelek 5.1 VREDNOSTI IN MERE na strani 36), ali pa uporabimo vrednost `auto`, ko pregledovalnik samodejno določi odmik. V primeru, ko levo in desno oddaljenost nastavimo na `auto`, bo element sredinsko poravnan glede na element, v katerem se nahaja.

Vse štiri lastnosti lahko hkrati nastavimo z lastnostjo `margin`, ki ji podamo vrednosti, ki si sledijo v smeri urinega kazalca z začetkom zgoraj. Posamezne vrednosti ločimo s presledkom. Ko želimo določiti enako oddaljenost v navpični smeri (zgornja in spodnja oddaljenost) ter enako oddaljenost v vodoravni smeri (leva in desna oddaljenost), lahko oddaljenost krajše podamo le z dvema vrednostima. Prva vrednost pomeni oddaljenost v navpični smeri, druga pa v vodoravni smeri. V primeru, ko želimo določiti enako oddaljenost od levega in desnega roba, ostali dva pa poljubno določiti, uporabimo tri vrednosti. Prva vrednost pomeni oddaljenost od zgornjega roba, druga vrednost oddaljenost v vodoravni smeri (leva in desna oddaljenost), tretja vrednost pa oddaljenost od spodnjega roba. V primeru, ko želimo uporabiti enako oddaljenost v vseh štirih straneh, jo podamo le z eno vrednostjo.

Če imata dva bločna elementa oba nastavljen odmik v navpični smeri, se oddaljenosti ne seštevata kot se v vodoravni smeri, ampak se uporabi večja izmed obeh. Če ima torej prvi bločni element določeno oddaljenost `margin-bottom 10` točk in drugi pod njim `margin-top 30` točk, bo oddaljenost med tema dvema bločnima elementoma `30` točk (in ne `40` točk).

Željeno ločitev tabele in teksta bomo v našem zgledu dosegli torej tako, da povečamo odmik med besedilom in tabelo. Obenem tabelo še sredinsko poravnajmo.

PRIMER CSS 7.4–9: Poravnava tabele

tabele.css

```
table {
 border: thin solid #A0522D;
 margin-top: 15px;
 margin-left: auto;
 margin-right: auto;
}

th, td {
 border: thin dotted #A0522D;
 text-align: center;
}

th {
 background-color: #D3A474;
 width: 90px;
 height: 30px;
}
```

Kaskadno slogovno predlogo `tabele.css` z import vstavimo v datoteko `stil.css`

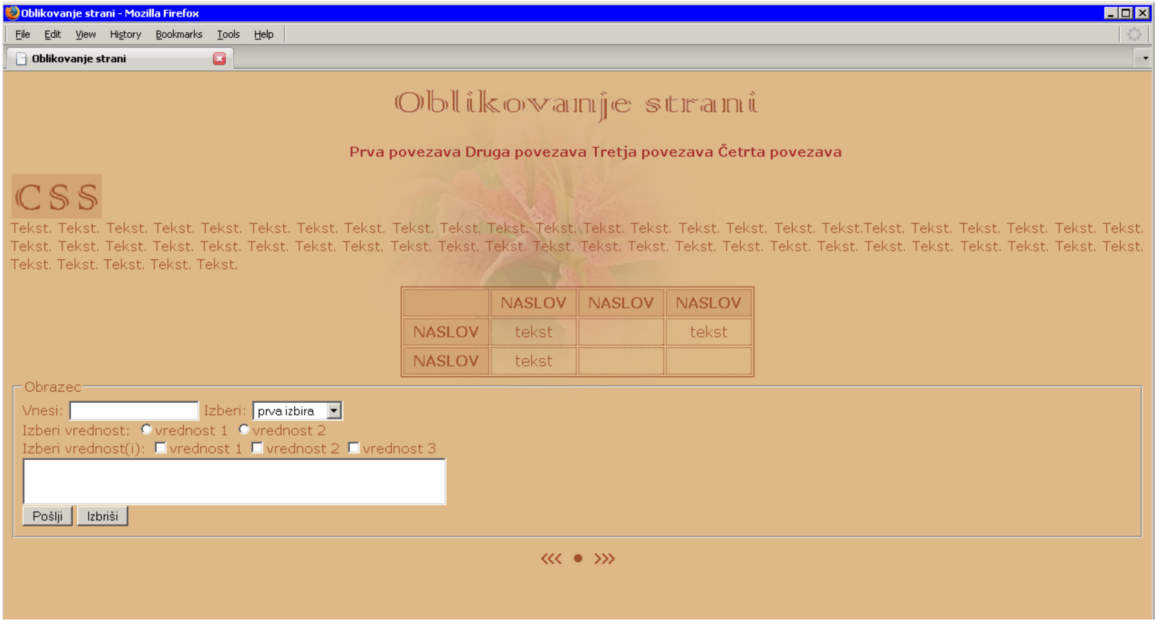
PRIMER CSS 7.4–10: V predlogo `stil.css` vstavimo predlogo `tabele.css`

stil.css

```
@import url(povezave.css);
@import url(tabele.css);

body {
 font-size: 15px;
 font-family: Verdana, Arial, sans-serif;
 text-align: justify;
 line-height: 25px;
 color: #A0522D;
 background-color: #DEB887;
 background-image: url(lilija.jpg);
```

```
 background-repeat: no-repeat;
 background-position: top center;
}

h1 {
 text-align: center;
}
```

in poglejmo, kako je sedaj videti spletna stran.



Slika 7.7–9: Izgled spletne strani z oblikovano tabelo

7.5. OBLIKOVANJE OBRAZCEV

V tem podrazdelku bomo pogledali kako se oblikujejo obrazci (vnosna polja, izbirna polja, labele, ...). Z oblikovanjem obrazcev ni priporočljivo pretiravati, saj vsi pregledovalniki ne podpirajo vseh slogovnih učinkov.

Na obrazcu bomo najprej določili rjavo barvo besedila ter rjavo ozadje vnosnih polj in gumbov ter gumbe sredinsko poravnali. Nastaviti moramo torej lastnosti značk HTML input, textarea in select.

PRIMER CSS 7.5–1: Določitev barve besedila in ozadja ter poravnava gumbov

```
obrazci.css
input, textarea, select {
 color: #A0522D;
 background-color: #D3A474;
}

.gumbi {
 text-align: center;
}
```

Za poravnavo gumbov na formi smo določili razred `gumbi`. Razred uporabimo na znački `div` znotraj katere se nahajajo gumbi forme.

PRIMER HTML 7.5–1: V dokumentu HTML uporabljen razred `gumbi`

```
stran.html
<html>
 <head>
   <title>Oblikovanje strani</title>
   <meta http-equiv="Content-Type" content="text/html;
   charset=windows-1250" />
   <link rel="stylesheet" href="stil.css" type="text/css">
 </head>

 <body>
   ...
   <div class="gumbi">
     <input type="submit" name="poslji" value="Pošlji">
     <input type="reset" name="brisi" value="Izbriši">
     </div>
   ...
 </body>
</html>
```

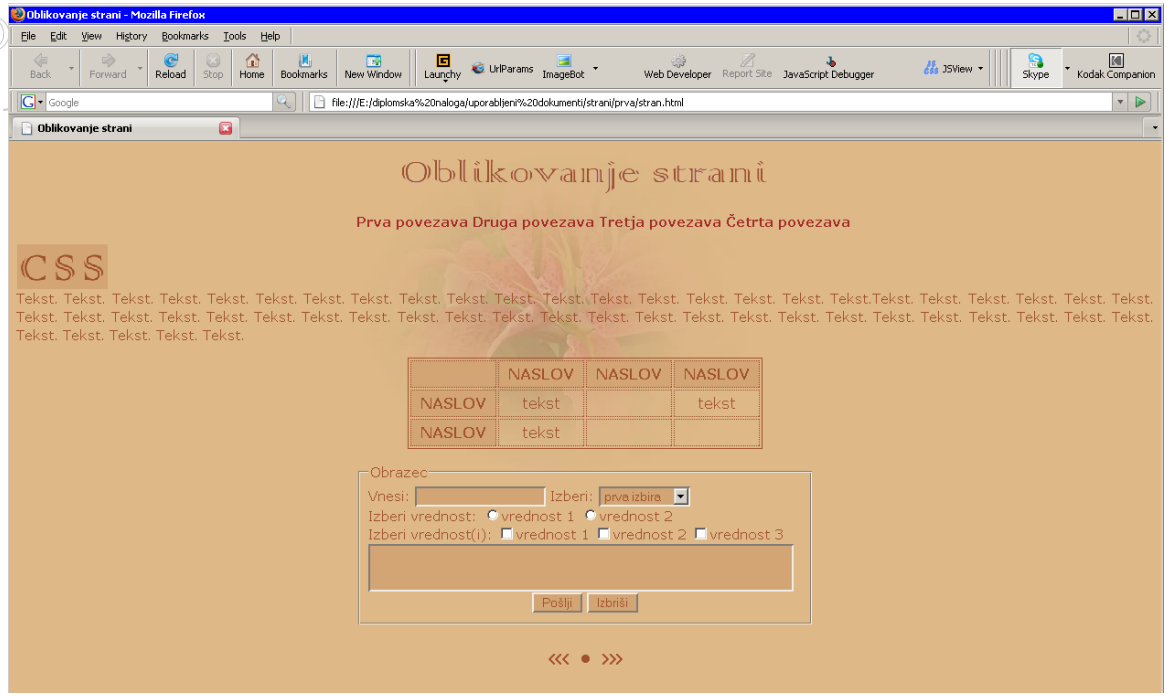
Poglejmo si stran

Slika 7.5–1: Izgled strani z dodano barvo besedila in ozadja ter poravnavo gumbov

Opazimo, da se obrazec razteza po celi strani. To nam ne ugaja, zato mu določimo ustrezno velikost. Prav tako ga še sredinsko poravnajmo ter povečajmo zgornji in spodnji odmik od obrazca.

PRIMER CSS 7.5–2: Poravnava elementov na obrazcu

```
obrazci.css
form {
 margin-top: 15px auto 30px;
 width: 500px;
}

input, textarea, select {
 color: #A0522D;
 background-color: #D3A474;
}

.gumbi {
 text-align: center;
}
```



Slika 7.5–2: Izgled strani s poravnanimi elementi

Besedilu lahko tudi določimo razmik med črkami. Kar dosežemo z lastnostjo `letter-spacing`.

V opisu obrazca, ki ga nadzira značka `legend`, bomo malce razmaknili črke in jih povdarili. Prav tako bomo opis obrobili z rjavo pikasto obrobo. Celotnemu obrazcu pa bomo določili rjavo neprekinjeno obrobo. To storimo tako, da ustrezno nastavimo lastnost značke `fieldset`.

PRIMER CSS 7.5–3: Oblikovanje opisa obrazca

obrazci.css

```
form {
 margin-top: 15px auto 30px;
 width: 500px;
}

legend {
 letter-spacing: 5px;
 font-weight: bold;
 border: thin dotted #A0522D;
}

input, textarea, select {
 color: #A0522D;
 background-color: #D3A474;
}

fieldset {
 border: thin dotted brown;
}

.gumbi {
 text-align: center;
}
```

Povečajmo še razmik med posameznimi elementi v obrazcu. V ta namen definirajmo dva razreda, `vnos` in `izberi`.

PRIMER CSS 7.5–4: Povečanje razmika med elementi

obrazci.css

```
form {
 margin-top: 15px auto 30px;
 width: 500px;
}

legend {
 letter-spacing: 5px;
 font-weight: bold;
 border: thin dotted #A0522D;
}

input, textarea, select {
 color: #A0522D;
 background-color: #D3A474;
}

fieldset {
 border: thin dotted brown;
}

.vnos {
 margin: 10px 5px;
}

.izberi {
 margin-left: 40px;
}

.gumbi {
 text-align: center;
}
```

Da bomo ta razmik uporabili, bo potreben manjši poseg v dokument HTML.

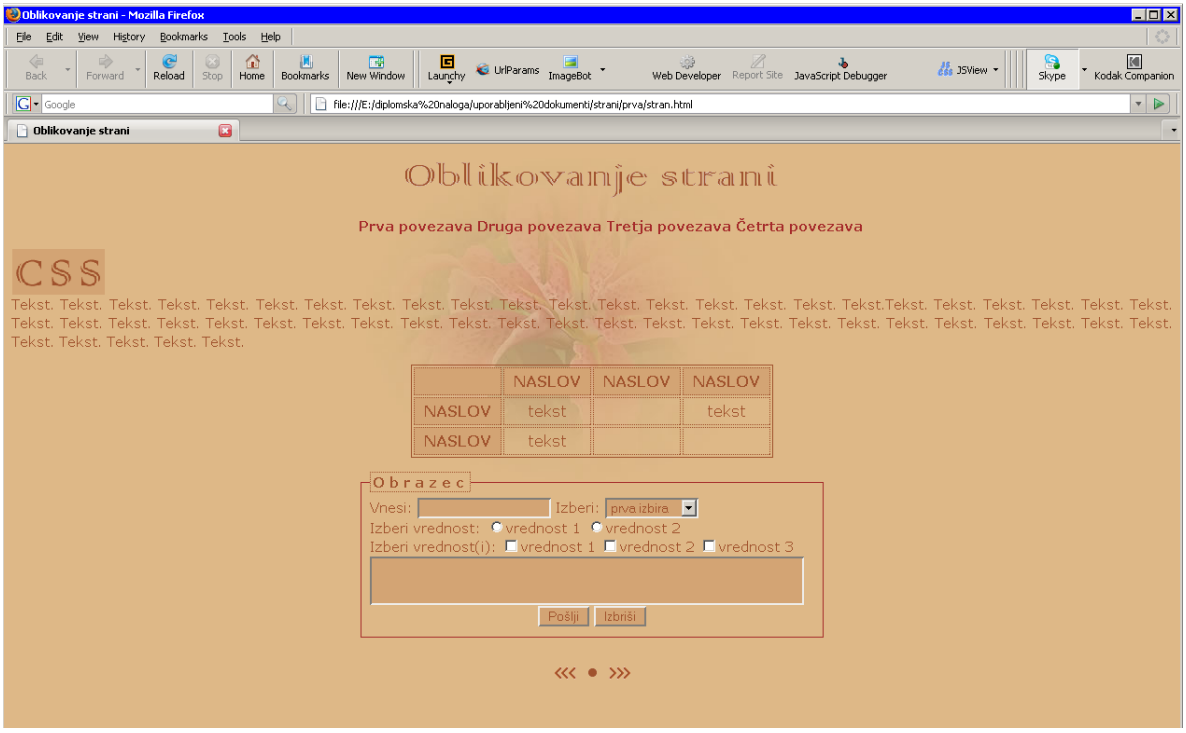
PRIMER HTML 7.5–4: V dokumentu HTML uporabljena razreda `vnos` in `izberi`

stran.html

```
<html>
 <head>
   <title>Oblikovanje strani</title>
   <meta http-equiv="Content-Type" content="text/html;
   charset=windows-1250" />
   <link rel="stylesheet" href="stil.css" type="text/css">
 </head>

 <body>
   ...
     <form action="stran.html" method="get">
       <fieldset>
         <legend>Obrazec</legend>
         <div class="vnos">
           <label>Vnesi:</label>
           <input type="text" name="vnosnoPolje">
           <label class="izberi">Izberi:</label>
           <select name="padajociSeznam">
             <option>prva izbira</option>
             <option>druga izbira</option>
             <option>tretja izbira</option>
           </select>
         </div>

         <div class="vnos">
           <label>Izberi vrednost:</label>
           <input type="radio" name="izbira">vrednost 1
```

```
        <input type="radio" name="izbira">vrednost 2
      </div>
      <div class="vnos">
        <label>Izberi vrednost(i):</label>
        <input type="checkbox" name="polje1">vrednost 1
        <input type="checkbox" name="polje2">vrednost 2
        <input type="checkbox" name="polje3">vrednost 3
      </div>

      <div class="vnos">
        <textarea name="tekst" rows="2" cols="55">
        </textarea>
      </div>

      <div class="gumbi">
        <input type="submit" name="poslji" value="Pošlji">
        <input type="reset" name="brisi" value="Izbriši">
      </div>
    </fieldset>
  </form>
  ...
</body>
</html>
```

Poglejmo si stran



Slika 7.5–4: Izgled strani s povečanim razmikom med elementi

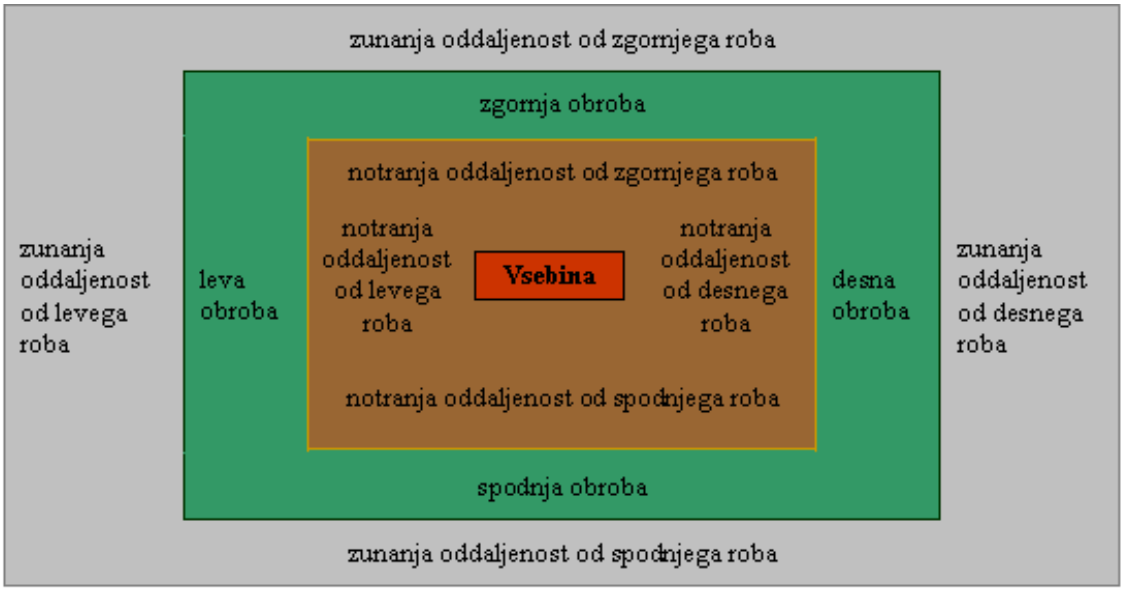
Dodajanje učinkov padajočemu seznamu ne podpirajo vsi pregledovalniki. Zaenkrat ga podpira le Firefox. Slogovnih učinkov za izbirne gumbe (`radio`) in potrditvena polja (`checkbox`) zaenkrat še noben pregledovalnik ne podpira.

7.6. OBLIKOVANJE OBROB

Vsaka prikazana vsebina (tekst, slika, obrazec itd.) vsebuje površino, ki jo obkrožajo robovi. Robove površine definirajo tri lastnosti:

- zunanja oddaljenost od roba elementa – `margin` (glej razdelek 7.4 OBLIKOVANJE TABEL na strani 54;) Na sliki Slika 7.6–1 je označena modro barvo
- obroba elementa – `border` (glej razdelek 7.4 OBLIKOVANJE TABEL na strani 54). Na sliki je označena z zeleno barvo.
- notranja oddaljenost od roba elementa – `padding`, na sliki označena z rjavo barvo. Notranja oddaljenost je prostor, ki obdaja vsebino. Določamo jo na podoben način kot lastnost `margin` (glej stran 57).

Za lažjo predstavo si poglejmo sliko



Slika 7.6–1: Prikaz robov površine elementa

Dobro je vedeti, da ko določimo elementu velikost, s tem določimo velikost površine znotraj katere je vsebina (na sliki označeno z rdečo barvo). Torej k velikosti elementa vrednosti lastnosti `margin`, `padding` in `border` niso všteti.

Spletni strani bomo določili velikost `750px`. Vsebino bomo obrobili z rjavo neprekinjeno obrobo ter celotno vsebino sredinsko poravnali. Poravnavo bomo dosegli tako, da bomo levo in desno oddaljenost od roba nastavili na `auto`.

PRIMER CSS 7.6–1: Dodana obroba spletni strani

stil.css

```
@import url(povezave.css);
@import url(tabele.css);
@import url(obrazci.css);

body {
 font-size: 15px;
 font-family: Verdana, Arial, sans-serif;
 text-align: justify;
 line-height: 25px;
 color: #A0522D;
 background-color: #DEB887;
 background-image: url(lilija.jpg);
 background-repeat: no-repeat;
 background-position: top center;
}

h1 {
```

```
 text-align: center;
}

#stran {
 width: 750px;
 margin-left: auto;
 margin-right: auto;
 border: thin solid brown;
}
```

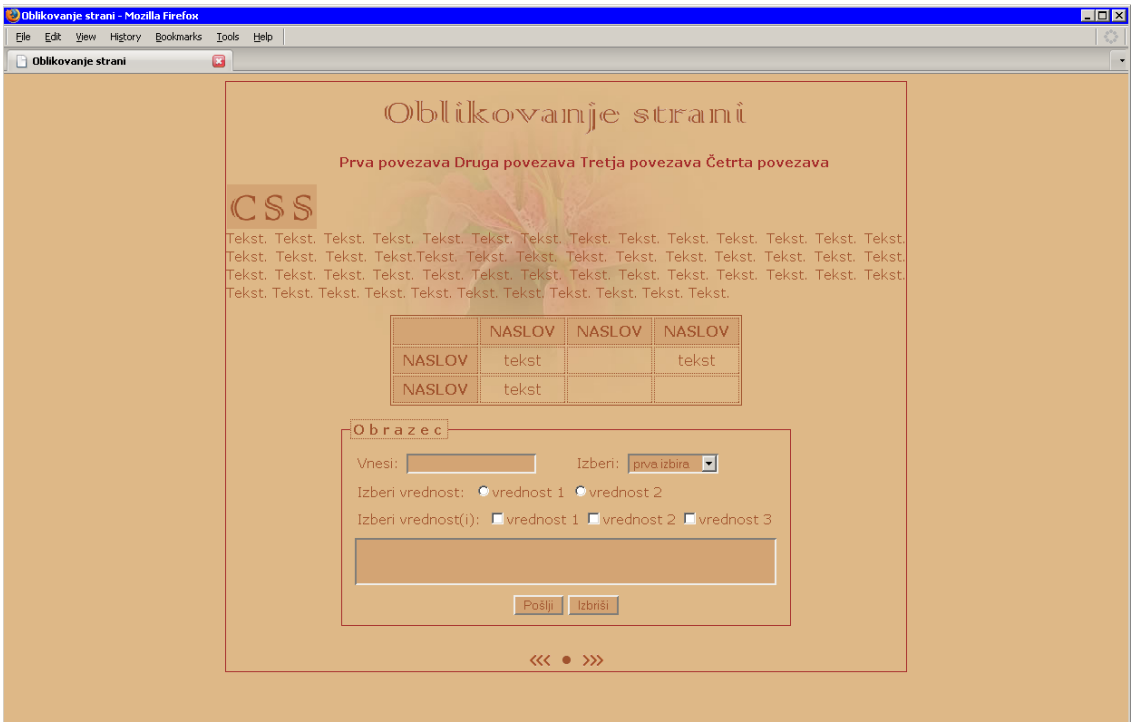
Zopet bo potreben manjši poseg v dokument HTML.

PRIMER HTML7.6–2: V dokumentu HTML uporabimo identifikator `stran`

stran.html

```
<html>
 <head>
   <title>Oblikovanje strani</title>
   <meta http-equiv="Content-Type" content="text/html;
   charset=windows-1250" />
   <link rel="stylesheet" href="stil.css" type="text/css">
 </head>

 <body>
   <div id="stran">
     ...
   </div>
 </body>
</html>
```

Poglejmo si sedaj stran



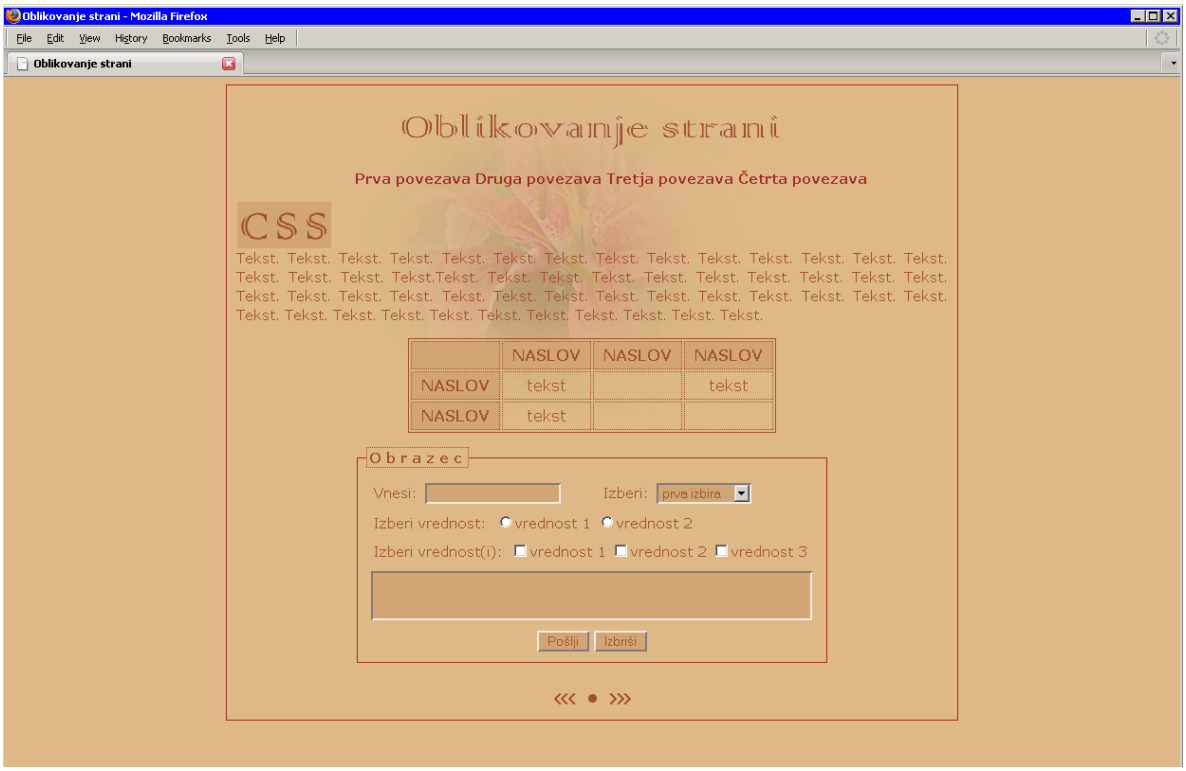
Slika 7.6–2: Izgled strani z dodano obrobo

Spletna stran bi bila lepša, če bi povečali razmik med obrobo in vsebino strani. Povečajmo oddaljenost za 10px.

PRIMER CSS 7.6–3: Povečana notranja oddaljenost vsebine strani

stil.css

```
@import url(povezave.css);
@import url(tabele.css);
@import url(obrazci.css);

body {
 font-size: 15px;
 font-family: Verdana, Arial, sans-serif;
 text-align: justify;
 line-height: 25px;
 color: #A0522D;
 background-color: #DEB887;
 background-image: url(lilija.jpg);
 background-repeat: no-repeat;
 background-position: top center;
}

h1 {
 text-align: center;
}

#stran {
 width: 750px;
 margin-left: auto;
 margin-right: auto;
 border: thin solid brown;
 padding: 10px;
}
```



Slika 7.6–3: Izgled strani z oblikovano vsebino strani

7.7. POSTAVITEV ELEMENTOV

Glede na ciljni izgled spletne strani (Slika 7–2:) moramo še ustrezno spremeniti položaj slike CSS. V ta namen si oglejmo, kako s pomočjo slogov določimo položaj elementa.

Element na spletni strani lahko postavimo na različne načine. V tem razdelku si bomo pogledali, kako položaj elementa določimo absolutno, relativno in kot lebdeči element.

Kje bo element, nadzorujejo lastnosti `top`, `right`, `left` in `bottom`, ki jim nastavimo ustrezne vrednosti. Z lastnostjo `position` povemo, ali so uporabljene vrednosti absolutne ali relativne. Če ima `position` vrednost `absolute`, so vrednosti lastnosti `top`, `right`, `left` in `bottom` mišljene absolutno glede na robove spletne strani. Če pa uporabimo vrednost `relative` se odmiki upoštevajo relativno glede na trenutno (normalno) postavitev elementa. Za lažjo predstavo bomo na različne načine določili položaj slike.

PRIMER HTML 7.7–1: Preprosta stran s sliko

```
...
<div class="tekst">
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst.  <img src="slika.gif">
</div>
...
```

Videz v pregledovalniku

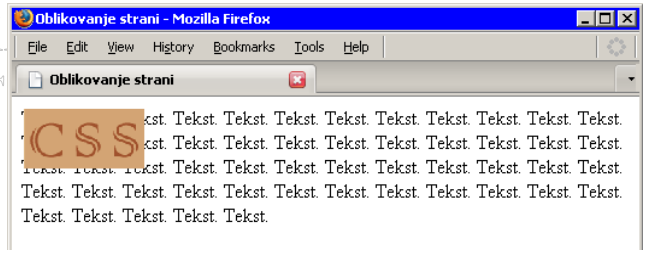
Slika 7.7–1: Izgled strani brez določenega položaja slike

Sliko bomo najprej postavili absolutno glede na spletno stran. To pomeni, da se bo slika postavila na točno določeno mesto v oknu pregledovalnika.

PRIMER CSS 7.7–2: Absolutno pozicioniranje slike

```
img {
 position: absolute;
 top: 10px;
 left: 10px;
}
```

Slika bo odmaknjena za `10 px` od levega roba okna in `10 px` od zgornjega roba okna. Če je vrednost pozitivna, se element pomakne proti notranjosti okna, drugače pa navzven.



Slika 7.7–2: Absolutno pozicioniranje slike

PRIMER CSS 7.7–3: Relativno pozicioniranje slike

```
img {
 position: relative;
 top: 10px;
 left: 10px;
}
```

Slika bo odmaknjena za `10 px` v desno in `10 px` navzdol (Slika 7.7–3) od položaja, kjer bi se drugače prikazala (glej 7.7–1).

Slika 7.7–3: Relativno pozicioniranje slike

Element spletne strani lahko tudi postavimo kot samostojni lebdeči element ob levi ali desni rob elementa, ki ta element vsebuje, preostala vsebina tega elementa pa obdaja lebdeči element. Na naši spletni strani bi lahko sliko postavili levo od teksta tako, da ga bo preostal tekst obdajal. Položaj elementa znotraj nekega drugega elementa določimo z lastnostjo `float`. Možne lastnosti so:

- `left` – položaj elementa je na levi strani elementa. Preostala vsebina tega elementa je desno od "plavajočega"elementa
- `right` – položaj elementa je na desni strani elementa. Preostala vsebina tega elementa pa na levi strani od "plavajočega" elementa
- `none` – običajen položaj elementa

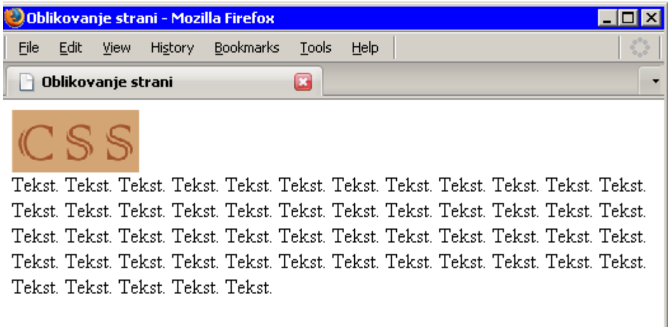
Za lažjo predstavo si bomo ogledali nekaj primerov.

PRIMER HTML 7.7–4: Spletna stran s sliko

```
...
<body>
 <div id="slika">
   <img src="slika.gif">
 </div>

 <div class="tekst">
   Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
   Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
   Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
   Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
   Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
   Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
```

```
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst.
 </div>
</body>
...
```

Izgled spletne strani

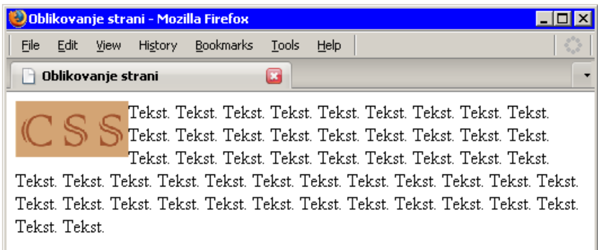


Slika 7.7–4: Izgled spletne strani

Najprej postavimo sliko (element z identifikatorjem `slika`) na levo stran spletne strani. Ker se element z identifikatorjem `slika` nahaja znotraj značke `body`, se bo slika postavila na levo stran spletne strani. Preostala vsebina pa ga bo obdajala.

PRIMER CSS 7.7–5: Postavitev slike na levo stran spletne strani

```
#slika {
 float: left;
 }
```



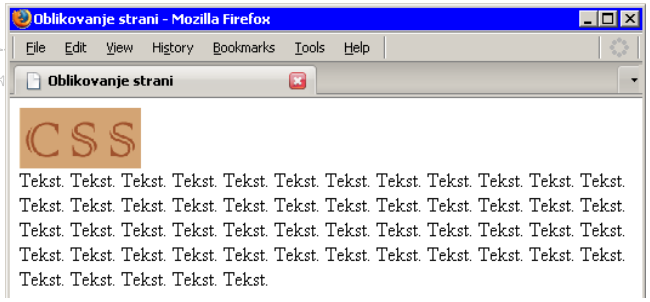
Slika 7.7–5: Postavitev slike na levo stran

Sedaj pa tekst postavimo ob sliko tako, da bo slika v enem stolpcu, tekst pa v drugem. Zopet uporabimo lastnost `float`.

PRIMER CSS 7.7–6: Postavitev teksta ob sliko

```
#slika {
 float: left;
 }

.tekst {
 float: left;
 }
```

Poglejmo kako izgleda stran



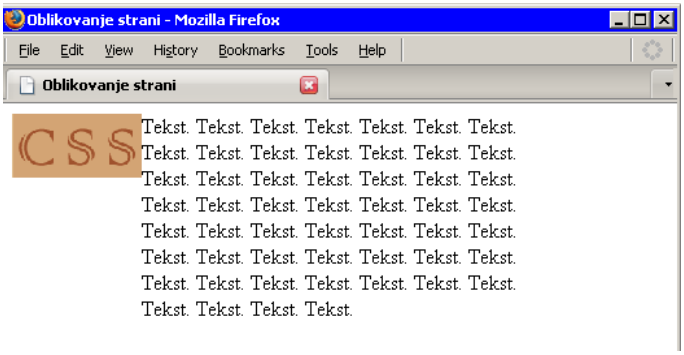
Slika 7.7–6: Izgled spletne strani s tekstom postavljenim v levo

Opazimo, da se je besedilo postavilo pod sliko. To se je zgodilo zato, ker je vsebina prevelika. Celotna vsebina se prestavi v naslednjo vrstico. Da bi dosegli željeni učinek, moramo podati velikost lebdečega elementa. Ta mora biti manjša, kot je še prostora na desni strani.

PRIMER CSS 7.7–7: Postavitev besedila na desno stran od slike

```
#slika {
 float: left;
 }

.tekst {
 float: left;
 width: 300px;
 }
```

Sedaj smo pa dobili dva stolpca enega zraven drugega.

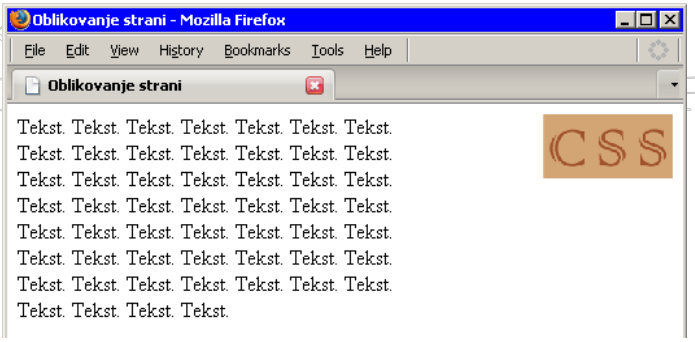


Slika 7.7–7: Postavitev besedila ob sliki

Postavimo sliko v desno.

PRIMER CSS 7.7–8: Postavitev slike v desno

```
#slika {
 float: right;
 }

.tekst {
 float: left;
 width: 300px;
}
```



Slika 7.7–8: Postavitev slike v desno

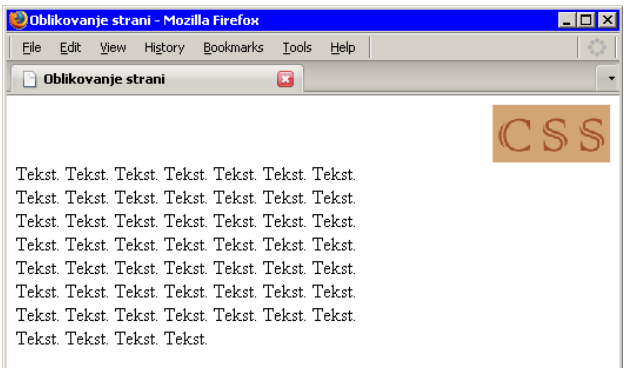
Kam se naj postavi element glede na predhodni lebdeči (`float`) element, pa lahko določimo z lastnostjo `clear`. Ta ima lahko naslednje vrednosti:

- `none` – elementi so postavljeni okoli lebdečega elementa tako, da so na levi in desni strani lebdeči elementi
- `left` – elementi so postavljeni pod lebdečim elementom tako, da na levi strani ne bo nobenih lebdečih elementov
- `right` – elementi so postavljeni pod lebdečim elementom tako, da na desni strani ne bo nobenih lebdečih elementov
- `both` – elementi so postavljeni pod lebdečim elementom tako, da na nobeni strani ne bo nobenih lebdečih elementov

Določimo besedilu lastnosti tako, da na svoji desni strani ne bo imel lebdečih elementov.

PRIMER CSS 7.7–9: Odstranitev lebdečih elementov

```
#slika {
 float: right;
}

.tekst {
 width: 300px;
 float: left;
 clear: right;
}
```



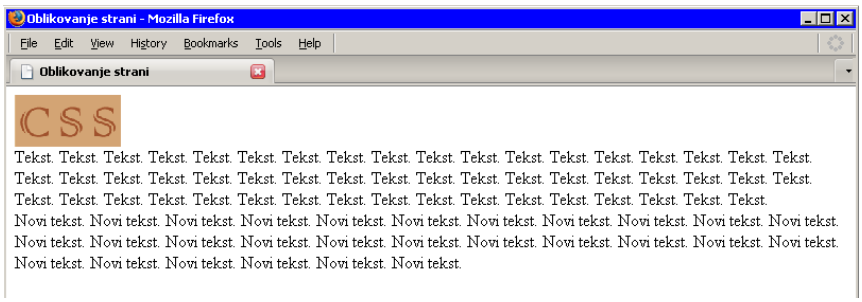
Slika 7.7–9: Odstranitev lebdečih elementov

Besedilo se je postavilo v naslednjo vrstico tako, da na svoji desni nima nobenega lebdečega elementa.

Poglejmo si še en primer. Spet začnimo od začetka – torej tako za sliko kot obe različici teksta položaja še nismo določali.

PRIMER HTML 7.7–10: Spletna stran

```
...
<div id="slika">
  <img src="slika.gif">
</div>

<div class="tekst">
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst. Tekst. Tekst. Tekst.
  Tekst. Tekst. Tekst. Tekst.
</div>

  <div class="noviTekst">
    Novi tekst. Novi tekst. Novi tekst. Novi tekst.
    Novi tekst. Novi tekst. Novi tekst. Novi tekst.
    Novi tekst. Novi tekst. Novi tekst. Novi tekst.
    Novi tekst. Novi tekst. Novi tekst. Novi tekst.
    Novi tekst. Novi tekst. Novi tekst. Novi tekst.
    Novi tekst. Novi tekst. Novi tekst. Novi tekst.
    Novi tekst. Novi tekst. Novi tekst. Novi tekst.
  </div>
...
```

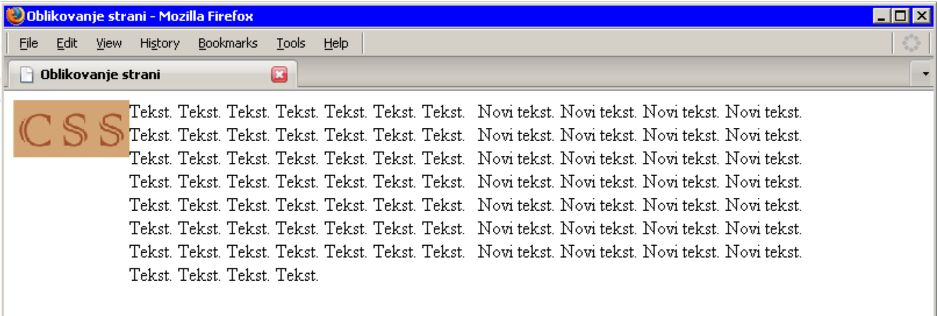


Slika 7.7–10: Izgled spletne strani

Postavimo sliko in besedilo v tri stolpce eden zraven drugega.

PRIMER CSS 7.7–11: Postavitev stolpcev

```
#slika {
 float: left;
}

.tekst {
 width: 300px;
 float: left;
}

.noviTekst {
 width: 300px;
 float: left;
}
```

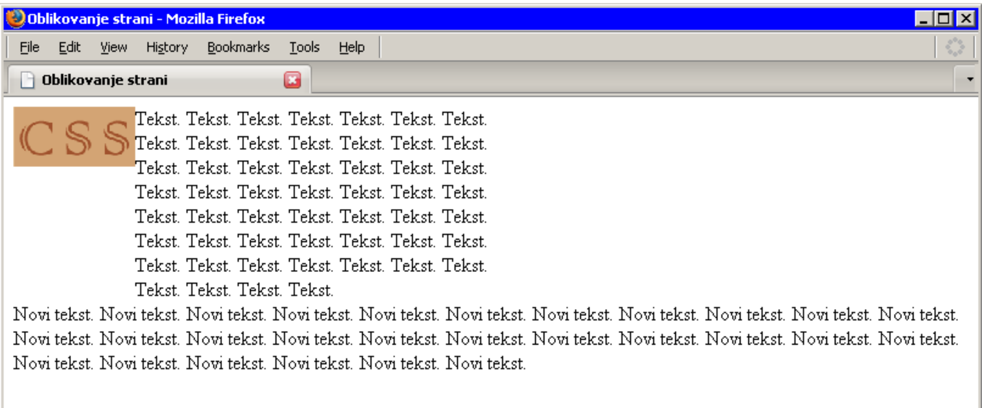


Slika 7.7–11: Trije stolpci

Sedaj si pa še poglejmo, kako bi lahko naredili dva stolpca, pod njima pa vrstico.

PRIMER CSS 7.7–12:

```
#slika {
 float: left;
}

.tekst {
 width: 300px;
 float: left;
}

.noviTekst {
 clear: left;
}
```



Slika 7.7–12: Postavitev dveh stolpcev in vrstice pod njima

Vrnimo se nazaj k oblikovanju naše spletne strani. Postavimo sliko v levo, besedilo pa naj bo izpisano okoli nje.

PRIMER CSS 7.7–13: Postavitev slike v levo

```
stil.css
@import url(povezave.css);
@import url(tabele.css);
@import url(obrazci.css);

body {
 font-size: 15px;
 font-family: Verdana, Arial, sans-serif;
 text-align: justify;
 line-height: 25px;
 color: #A0522D;
```

```
 background-color: #DEB887;
 background-image: url(lilija.jpg);
 background-repeat: no-repeat;
 background-position: top center;
}

h1 {
 text-align: center;
}

#stran {
 width: 750px;
 margin-left: auto;
 margin-right: auto;
 border: thin solid brown;
 padding: 10px;
}

.slika {
 float: left;
}
```

V dokumentu uporabimo slog tako, da sliki dodamo razred `slika`.

PRIMER HTML7.7–14: V dokumentu HTML uporabljen razred `slika`

stran.html

```
<html>
 <head>
   <title>Oblikovanje strani</title>
   <meta http-equiv="Content-Type" content="text/html;
   charset=windows-1250" />
   <link rel="stylesheet" href="stil.css" type="text/css">
 </head>

 <body>
   ...
     <div>
       <img src="slika.gif" class="slika">
     </div>
   ...
 </body>
</html>
```

Slika 7.7–14: Postavitev slike v levo

Opazimo, da sta slika in besedilo preveč skupaj. Povečajmo razmik med njima.

PRIMER CSS 7.7–15: Povečanje razmika med sliko in besedilom

stil.css

```
@import url(povezave.css);
@import url(tabele.css);
@import url(obrazci.css);

body {
 font-size: 15px;
 font-family: Verdana, Arial, sans-serif;
 text-align: justify;
 line-height: 25px;
 color: #A0522D;
 background-color: #DEB887;
 background-image: url(lilija.jpg);
 background-repeat: no-repeat;
 background-position: top center;
}

h1 {
 text-align: center;
}

#stran {
 width: 750px;
 margin-left: auto;
 margin-right: auto;
 border: thin solid brown;
 padding: 10px;
}

.slika {
 float: left;
 padding: 5px;
}
```

Poglejmo končni izgled spletne strani, ki je tak, kot smo napovedali na začetku

Slika 7.7–15: Oblikovana spletna stran

V naslednjem razdelku si bomo pogledali, na kakšen način lahko to oblikovano stran preoblikujemo.

8. DODATNI PRIMERI OBLIKOVANJA S CSS

V tem razdelku bomo nadaljevali z oblikovanjem prej pripravljene spletne strani, ki ji bomo dodali še določene elemente. Povezave, ki so sedaj tik pod naslovom, bi radi prestavili drugam in tako še bolj povdarili, da jih uporabljamo za navigacijo po spletnih straneh. Želeli bi, da so te povezave navedene na levi strani v "plavajočem" okencu. Zato bomo dopolnili datoteko `povezave.css`, kjer opisujemo izgled povezav.

Ker želimo imeti povezave vsako v svoji vrstici, je potrebno povezavam določiti, da se obnašajo kot bločni elementi. Nato jih postavimo v levo stran ter jih ustrezno poravnamo.

PRIMER CSS 8–1: Postavitev plavajočega okenca

povezave.css

```
#meni, #noga {
 text-align: center;
}

a {
 text-decoration: none;
 font-weight: bold;
}

a:link {
 color: #A52A2A;
}

a:active {
 color: #A0522D;
}

a:visited {
 color: #D99777;
}

a:hover {
 color:  #F5DEB3;
}

ul {
 list-style-type: square;
}

li {
 display: inline;
}

img {
 border: 0;
}

#meni a {
 display: block;
}

#meni {
 float: left;
 width: 150px;
 text-align: left;
 padding: 1px;
}
```

S temi spremembami je stran videti sedaj takole:

Slika 8–1: Postavitev menija na levo stran

Opazimo, da obrazec ni poravnan s preostalo vsebino. Da bo obrazec lepo poravnan z zgornjo vsebino, moramo dodati nov slogovni učinek. Imenujmo ga besedilo in ga definirajmo kot identifikator. Z njim bomo potem lahko postavili besedilo poleg menija.

PRIMER CSS 8–2: Postavitev vsebine ena zraven druge

stil.css

```
#meni, #noga {
 @import url(povezave.css);
@import url(tabele.css);
@import url(obrazci.css);

body {
 font-size: 15px;
 font-family: Verdana, Arial, sans-serif;
 text-align: justify;
 line-height: 25px;
 color: #A0522D;
 background-color: #DEB887;
 background-image: url(lilija.jpg);
 background-repeat: no-repeat;
 background-position: top center;
}

h1 {
 text-align: center;
}

#stran {
 width: 750px;
 margin-left: auto;
 margin-right: auto;
 border: thin solid brown;
 padding: 10px;
}

.slika {
```

```
 float: left;
 padding: 5px;
}

#besedilo {
 margin: 2px;
 width: 580px;
 float: left;
 padding: 1px 5px 1px 5px;
}
```

Če datoteko HTML popravimo v

PRIMER HTML 8–2: V dokumentu HTML uporabljen identifikator `besedilo`

stran.html

```
<html>
 <head>
   <title>Oblikovanje strani</title>
   <meta http-equiv="Content-Type" content="text/html;
   charset=windows-1250" />
   <link rel="stylesheet" href="stil.css" type="text/css">
 </head>

 <body>
   ...
   <div id="besedilo">
     <div>
       <img src="slika.gif" class="slika">
     </div>
   </div>
   ...
 </body>
</html>
```

Dobimo naslednji izgled spletne strani

Slika 8–2: Postavitev vsebine eno zraven druge

Na spletni strani opazimo, da se povezave raztezajo čez več vrstic, okoli njih pa je še dovolj prostora. Boljši videz bomo dosegli, če bomo ustrezno nastavili zunanjo in notranjo oddaljenost (glej razdelek 63 OBLIKOVANJE OBROB na strani 63).

PRIMER CSS 8–3: Zmanjšanje zunanje in notranje oddaljenosti pri povezavah

povezave.css

```
#meni, #noga {
 text-align: center;
}

a {
 text-decoration: none;
 font-weight: bold;
}

a:link {
 color: #A52A2A;
}

a:active {
 color: #A0522D;
}

a:visited {
 color: #D99777;
}

a:hover {
 color:  #F5DEB3;
}

ul {
 list-style-type: none;
 margin: 0;
 padding: 0;
}

li {
 display: inline;
}

img {
 border: 0;
}

#meni a {
 display: block;
}

#meni {
 float: left;
 width: 150px;
 text-align: left;
 padding: 1px;
}
```

Slika 8–3: Zmanjšana zunanja in notranja oddaljenost

Kot zadnjo spremembo pa si oglejmo še, kako bi meni prestavili z leve na desno stran. Popraviti je potrebno datoteki (popraviti smeri) `povezave.css` in `stil.css`.

PRIMER CSS 8–4: Postavitev menija na desno stran

povezave.css

```
...

#meni a {
 display: block;
}

#meni {
 float: right;
 width: 150px;
 text-align: left;
 padding: 1px;
}
```

PRIMER CSS 8–5: Postavitev besedila ob meni

stil.css

```
...

#besedilo {
 margin: 2px;
 width: 580px;
 float: right;
 padding: 1px 5px 1px 5px;
}
```

Slika 8–5: Prestavitev menija na desno stran

9. LITERATURA IN VIRI

1. Kaltenekar, Matic. Hitri vodnik – HTML, CSS in JavaScript – Oblikovanje spletnih strani. Ljubljana: Pasadena, 2006.

2. Mrhar, Peter. XHTML 1.1 in slogi CSS2. Nova Gorica: Flamingo založba, 2002

3. Bartlett, Kynn. Sams Teach Yourself CSS in 24 Hours. 2. izdaja. Indianapolis: Sams Publishing, 2006

4. Lemay, Laura in Colburn, Rafe. Sams Teach Yourself Web Publishing with HTML and CSS in One Hour a Day. Indianapolis: Sams Publishing, 2006

5. Cascading Style Sheets, level 1. 11 januar 1999. World Wide Web Consortium. http://www.w3.org/TR/1999/REC–CSS1–199901111, dostop: 3. julij 2007

6. Cascading Style Sheets, Level 2. 12 maj 1998. World Wide Web Consortium. http://www.w3.org/TR/1998/REC–CSS2–19980512, dostop: 12. november 2007

7. Cascading Style Sheets, level 2 revision 1. 19 julij 2007. World Wide Web Consortium. http://www.w3.org/TR/2007/CR–CSS21–20070719, dostop: 15. november 2007

8. http://www.westciv.com/style_master/academy/css_tutorial, dostop: 3. julij 2007