Prednosti i nedostaci korištenja CSS predprocesora (Sass, Less i Stylus) u odnosu na čist CSS

Dražić, Dorotea

Sveučilište Josipa Jurja Strossmayera u Osijeku

Filozofski fakultet Osijek

Preddiplomski studij informatologije

Dorotea Dražić

Prednosti i nedostaci korištenja CSS predprocesora (Sass, Less i Stylus) u odnosu na čist CSS

Završni rad

Mentor: doc. dr. sc. Tomislav Jakopec

Osijek, 2023.

Sveučilište Josipa Jurja Strossmayera u Osijeku

Filozofski fakultet Osijek

Odsjek za informacijske znanosti

Preddiplomski studij informatologije

Dorotea Dražić

Prednosti i nedostaci korištenja CSS predprocesora (Sass, Less i Stylus) u odnosu na čist CSS

Završni rad

Društvene znanosti, informacijske i komunikacijske znanosti,

informacijski sustavi i informatologija

Mentor: doc. dr. sc. Tomislav Jakopec

Osijek, 2023.

Prilog: Izjava o akademskoj čestitosti i o suglasnosti za javno objavljivanje

Obveza je studenta da donju Izjavu vlastoručno potpiše i umetne kao treću stranicu završnoga, odnosno diplomskog rada.

IZJAVA

Izjavljujem s punom materijalnom i moralnom odgovornošću da sam ovaj rad samostalno napisala te da u njemu nema kopiranih ili prepisanih dijelova teksta tuđih radova, a da nisu označeni kao citati s navođenjem izvora odakle su preneseni.

Svojim vlastoručnim potpisom potvrđujem da sam suglasna da Filozofski fakultet u Osijeku trajno pohrani i javno objavi ovaj moj rad u internetskoj bazi završnih i diplomskih radova knjižnice Filozofskog fakulteta u Osijeku, knjižnice Sveučilišta Josipa Jurja Strossmayera u Osijeku i Nacionalne i sveučilišne knjižnice u Zagrebu.

U Osijeku, 23. kolovoza, 2023.

Drazić Dorotea, 0122236419

Ime i prezime studenta, JMBAG

SAŽETAK

CSS predprocesori, kao što su Sass, Less i Stylus, pružaju značajne prednosti i istovremeno nose neke nedostatke u odnosu na klasičan čisti CSS. Ovi alati postaju sve popularniji u razvoju mrežnih stranica zbog svoje sposobnosti da olakšaju i unaprijede proces stiliziranja mrežnih stranica. Omogućuju razdvajanje CSS koda u manje datoteke, što olakšava organizaciju i održavanje projekata. Pomoću varijabli, mixina i funkcija, programeri mogu izbjegavati ponavljanje koda i smanjiti redundantnost. To također povećava čitljivost koda i pojednostavljuje promjene stilova na različitim dijelovima mrežne stranice. Dodatno, CSS predprocesori omogućuju lakše rukovanje bojama, dimenzijama i drugim svojstvima, čime se povećava fleksibilnost i skalabilnost dizajna. Programeri mogu koristiti logičke konstrukcije kao što su petlje i uvjetni izrazi, što olakšava dinamičke promjene stilova na temelju uvjeta. Međutim, postoji nekoliko nedostataka CSS predprocesora. Prvo, implementacija predprocesora zahtijeva dodatne alate i korak kompilacije, što može dodati složenost u radni proces. Također, generirani CSS može biti znatno veći od izvornog koda, što može utjecati na performanse mrežne stranice. Još jedan nedostatak je potreba za prilagodbom radnog tijeka i učenjem novih sintaksa, što može zahtijevati dodatno vrijeme i trud. Odabir između CSS predprocesora i klasičnog čistog CSS-a ovisi o potrebama i zahtjevima projekta. Prednosti fleksibilnosti CSS predprocesora mogu biti vrlo korisne u većim i složenijim projektima. Međutim, manji projekti ili oni koji zahtijevaju brže performanse mogu se i dalje osloniti na klasičan CSS. Važno je pažljivo procijeniti prednosti i nedostatke kako bi se donijela najbolja odluka.

Ključne riječi: CSS, CSS predprocesori, Sass, Less, Stylus

SADRŽAJ

1. UVOD

Cascading Style Sheets (CSS) igra ključnu ulogu u oblikovanju izgleda i stila mrežnih stranica, omogućavajući programerima i dizajnerima da kreativno oblikuju vizualni identitet svojih projekata. Međutim, s porastom složenosti i zahtjeva modernih mrežnih stranica, čisti CSS može postati neefikasan i težak za održavanje. Upravo zbog toga, CSS predprocesori, poput Sass, Less i Stylus, postaju sve popularniji alati.

U ovom završnom radu istražuju se prednosti i nedostatki CSS predprocesora u odnosu na klasičan čisti CSS. U početku će se detaljno opisati CSS kao osnovni stilski jezik za mrežnih stranice. Otkrit će se kako CSS pruža moćan način definiranja stilova, no isto tako može postati ograničavajući kada se suočimo s kompleksnim projektima.

Zatim, fokusirat će se se na tri popularna CSS predprocesora: Sass, Less i Stylus. Razmotrit će se njihove ključne značajke, sintaksa i način rada. Svaki od ovih predprocesora ima svoje prednosti i posebne karakteristike, što će nam omogućiti bolje razumijevanje kako svaki od njih pridonosi olakšavanju i unapređenju procesa stiliziranja mrežnih stranica.

Nadalje, analizirat će se prednosti i nedostatci korištenja CSS predprocesora u odnosu na čisti CSS. Prednosti će obuhvatiti upotrebu varijabli, mixinova i funkcija, te olakšavanje organizacije i održavanja koda. Također će se istražiti kako predprocesori pružaju bolju fleksibilnost u stiliziranju, čime se olakšava dinamičko prilagođavanje stilova prema različitim uvjetima. Međutim, unatoč prednostima, također će se navesti i nedostaci CSS predprocesora. Proučit će se potreba za dodatnim alatima, složenost kompilacije i povećana veličina generiranih CSS datoteka. Također će se navesti i problem izgubljenih rednih brojeva linija u generiranom CSS-u, što može otežati pronalaženje grešaka i održavanje koda.

Kroz sve ove analize, cilj je pružiti jasno razumijevanje prednosti i izazova korištenja CSS predprocesora. Rad će pomoći u donošenju informiranih odluka o korištenju CSS predprocesora u svojim projektima, uzimajući u obzir specifične potrebe i zahtjeve svake mrežne stranice. Cilj rada je prikazati glavne aspekte ovih alata i pružiti smjernice za njihovu optimalnu upotrebu u razvoju modernih mrežnih stranica.

2. CASCADING STYLE SHEET (CSS)

Cascading Style Sheets (CSS) stilski je jezik koji se koristi za opisivanje prezentacije dokumenata napisanih u HTML-u ili XML-u. CSS opisuje kako bi elementi trebali izgledati na ekranu, papiru, govoru ili drugim medijima.[1] Razvijanjem mrežnog mjesta, elementi su u početku umetnuti u HTML za definiranje prezentacije (kao što je oznaka <font>), ali ubrzo je uočeno da je potreban stilski jezik koji oslobađa HTML potrebe za prikazom sadržaja i njegovo oblikovanje.[2] Kao što je naglašeno u nazivu, kaskadno označava srž CSS-a. To znači da se primjenjuje zadnje dodijeljeno svojstvo, odnosno svojstva se redefiniraju kaskadno te se ne mijenjaju u originalnoj CSS datoteci.[3] CSS se koristi za stiliziranje i raspored mrežnih stranica, za promjenu fonta, boje, veličine i razmaka sadržaja, za njegovu podjelu u stupce ili za dodavanje animacije i drugih ukrasnih značajki.[4]

Postoji vanjski CSS koji je u zasebnoj datoteci s nastavkom .css. Ovo je najčešći i najkorisniji način prijenosa CSS-a u dokumente. Možete povezati jednu CSS datoteku s više mrežnih stranica i stilizirati ih istom CSS datotekom. U Prvim koracima s CSS-om, poveže se vanjska datoteka s mrežnom stranicom. Na vanjsku CSS datoteku referira se pomoću HTML 'link' elementa. HTML 'link' element se može koristiti bilo gdje u HTML dokumentu, međutim standardno se definira u 'head' elementu:[5]

[1] MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (2023-06-02)
[2] Web Tech. CSS. URL: https://www.webtech.com.hr/css.php (2023-06-02)
[3] CSS-tricks. The “C” in CSS: The Cascade. URL: https://css-tricks.com/the-c-in-css-the-cascade/ (2023-06-02)
[4] MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (2023-06-03)
[5] Isto

```html
<!DOCTYPE html>
<html lang="hr">
  <head>
    <meta charset="utf-8" />
    <title>CSS</title>
    <link rel="stylesheet" href="stil.css" />
  </head>
  <body>
    <h1>Pozdrav svijetu</h1>
    <p>Moj CSS primjer</p>
  </body>
</html>
```

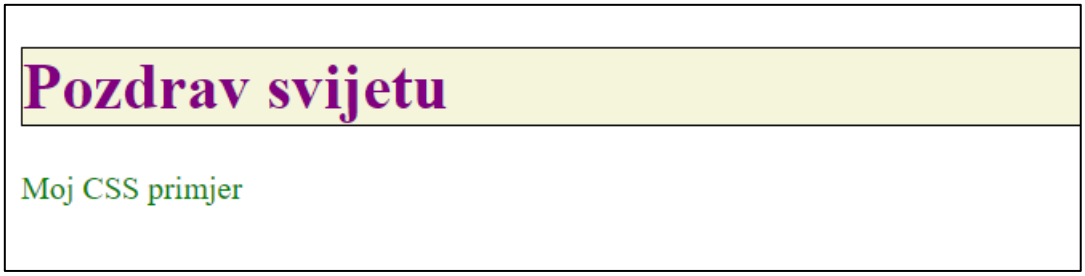
Slika 1. Vanjski CSS

CSS datoteka na koju smo se referirali može izgledati ovako:

```css
h1 {
    color: purple;
    background-color: beige;
    border: 1px solid black;
  }

  p {
    color: green;
  }
```

Slika 2. CSS datoteka

Ovo je izgled u internet pregledniku:



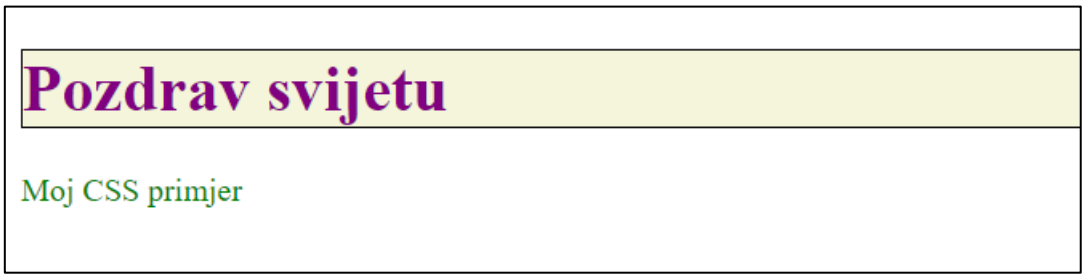
Slika 3. Prikaz u pregledniku

Unutarnji CSS nalazi se unutar HTML dokumenta. Potrebno je postaviti CSS unutar 'style' elementa koji se nalazi unutar HTML 'head' elementa:[6]

```
<!DOCTYPE html>
<html lang="hr">
  <head>
    <meta charset="utf-8" />
    <title>CSS</title>
    <style>
      h1 {
        color: purple;
        background-color: beige;
        border: 1px solid black;
      }

      p {
        color: green;
      }
    </style>
  </head>
  <body>
    <h1>Pozdrav svijetu</h1>
    <p>Moj CSS primjer</p>
  </body>
</html>
```

Slika 4. Unutarnji CSS

Ovo je izgled u internet pregledniku:



Slika 5. Prikaz u pregledniku

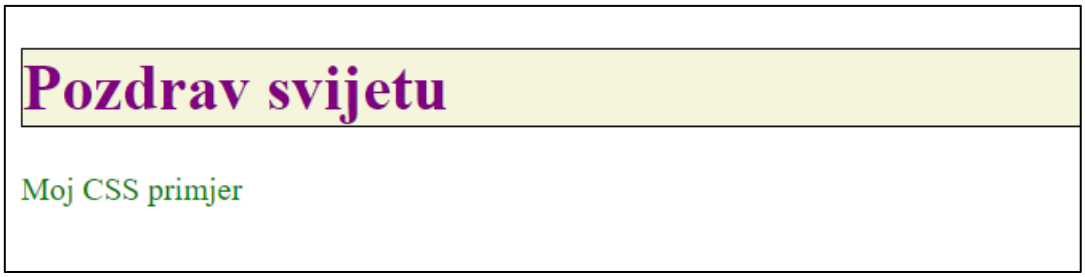
[6] MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (2023-06-03)

Također, postoji i inline (umetnuti/ugrađeni) CSS. To su CSS deklaracije koje utječu na pojedinačne HTML elemente, sadržane unutar atributa 'style':[7]

```
<!DOCTYPE html>
<html lang="hr">
  <head>
    <meta charset="utf-8" />
    <title>CSS</title>
  </head>
  <body>
    <h1 style="color: purple; background-color: beige; border: 1px solid black;">
      Pozdrav svijetu
    </h1>
    <p style="color: green;">Moj CSS primjer</p>
  </body>
</html>
```

Slika 6. Umetnuti CSS

Isto tako, izgled u internet pregledniku:



Slika 7. Prikaz u pregledniku

Ovaj način korištenja CSS-a nije preporučljiv. To je u suprotnosti s najboljom praksom, loše je za održavati, a jedna promjena stila može zahtijevati više uređivanja na jednoj mrežnoj stranici. Osim toga, ugrađeni CSS također miješa prezentacijski kod s HTML-om i sadržajem, zbog čega je teško za čitanje i razumijevanje.[8] CSS je osmišljen za mrežne dizajnere s ograničenim programerskim iskustvom, što znači da mu nedostaju određene programerske konstrukcije poput varijabli, uvjetnih i ponavljajućih blokova te funkcija. Zbog toga je otežana ponovna upotreba koda i održavanje stilskog koda. Ovaj nedostatak je bio glavni pokretač razvoja predprocesorskih alata kao što su Sass, Less i Stylus. Ti alati omogućuju programerima korištenje varijabli, petlji, funkcija i mixina unutar CSS-a. To znači da se CSS razvijanje gotovo

[7] MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (2023-06-03)
[8] Isto

približava programiranju s proširenom funkcionalnošću, što značajno olakšava rad i poboljšava ponovnu upotrebu koda.[9]

3. CSS PREDPROCESORI

Kada kompleksnost mrežne stranice raste, često primjećujemo da se CSS datoteke pune s mnogo pravila i ponavljajućim dijelovima koda. Kako bismo uštedjeli vrijeme i olakšali održavanje tih stilova na fleksibilniji način, koristimo CSS predprocesore.[10] CSS predprocesor je alat koji se koristi za proširenje osnovne funkcionalnosti zadanog čistog CSS-a putem vlastitog skriptnog jezika.[11] Većina CSS predprocesora ima značajke koje ne postoje u čistom CSS-u, kao što su mixin, selektor ugniježđenja (nesting selector), selektor nasljeđivanja (inheritance selector) i slično. Ove značajke čine CSS strukturu čitljivijom i lakšom za održavanje.[12]

Preglednici razumiju samo CSS, što ponekad nije dovoljno za pisanje čistih pravila za višekratnu upotrebu. Korištenjem samo CSS-a, dizajner ili programer ne može ponovno upotrijebiti zbirku pravila u više selektora koji su imali nejasne dijelove podataka u CSS-u. Kako bi se prevladala većina ovih ograničenja, stvoren je koncept predprocesora. Omogućuje napredan način pisanja CSS-a koji proširuje osnovnu funkcionalnost. Ovaj kod visoke razine kasnije se kompilira u normalan CSS kod koji preglednici razumiju.[13] Sada postoji nekoliko CSS predprocesora, osim Sass-a, Less-a i Stylusa, čija je upotreba postala sve popularniji trend u industriji. Na primjer, anketa provedena 2012. godine među više od 13 000 mrežnih programera pokazala je da 54% ispitanika koristi CSS predprocesore u svom radu.[14]

Jedna od ključnih značajki CSS predprocesora je mogućnost korištenja varijabli. Ova mogućnost pruža značajne pogodnosti jer omogućuje programerima CSS-a da definiraju osnovne vrijednosti, kao što su boje ili veličine fontova, kao varijable i ponovno ih koriste u

[9] CSS Preprocessing: Tools and Automation Techniques. URL: https://doi.org/10.3390/info9010017 (2023-06-03)
[10] Isto
[11] Lambdatest. Comprehensive CSS Preprocessor Comparison: Sass vs LESS vs Stylus. URL: https://www.lambdatest.com/blog/css-preprocessors-sass-vs-less-vs-stylus-with-examples/ (2023-06-04)
[12] MDN Web Docs. CSS preprocessors URL: https://developer.mozilla.org/en-US/docs/Glossary/CSS_Preprocessor (2023-06-05)
[13] Shero. What is a CSS preprocessor? URL: https://sherocommerce.com/what-is-a-css-preprocessors-why-use-them/ (2023-06-05)
[14] Coyier, C. Popularity of CSS Preprocessors. URL: https://css-tricks.com/poll-results-popularity-of-css-preprocessors/ (2023-06-05)

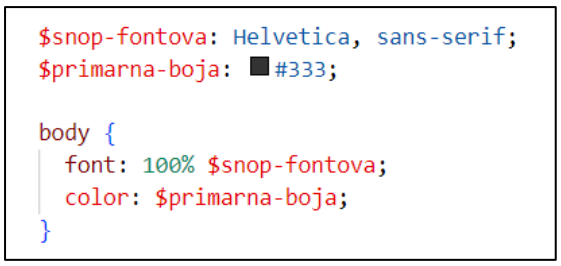
cijeloj CSS datoteci. Tradicionalni pristup zahtijeva ponavljanje istih vrijednosti u svakom pravilu, što može biti nepraktično i sklono greškama, posebno u velikim i složenim mrežnim aplikacijama. Upotrebom CSS predprocesora, kada se želi promijeniti vrijednost neke varijable, potrebno je samo izmijeniti njenu inicijalizaciju na jednom mjestu. Na taj način, promjene će se odraziti na sve dijelove CSS-a gdje je ta varijabla korištena, bez potrebe za ručnim izmjenama svakog pojedinog pravila. Varijable u CSS predprocesorima rade slično kao u bilo kojem programskom jeziku i uključuju koncepte za tipove podataka i opsega. Svaki predprocesor ima svoju sintaksu za deklariranje i inicijalizaciju varijabli. Na primjer, u Sass-u varijable počinju s "$", u Less-u s "@", dok u Stylusu nema prefiksa. Vrijednosti se obično dodjeljuju pomoću dvotočke (:) u Sass-u i Less-u, a s jednakosti (=) u Stylusu. Primjerice, deklariramo i inicijaliziramo varijablu u CSS predprocesoru i zatim je koristimo kao vrijednost svojstva za dva pravila. Konačni CSS će automatski koristiti vrijednost varijable umjesto njenih referenci u cijeloj datoteci.[15]

3.1. Sass: Syntactically Awesome Style Sheets

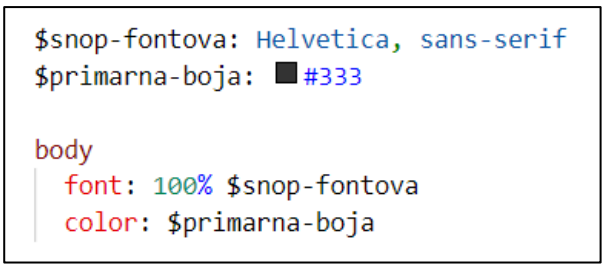
Osim što je najpopularniji CSS predprocesor, jedan je i od najstarijih, pokrenut 2006. od strane Hampton Catlina, a kasnije ga razvija Natalie Weizenbaum. Napisan je u jeziku Ruby, ali omogućeno je raščlanjivanje Sass-a na drugim jezicima i njegovo odvajanje od Rubyja korištenjem predkompilatora LibSass. Zahvaljujući svojoj stabilnosti i logičkoj snazi, Sass se postavio na čelo CSS predprocesora.[16]

Sass ima dvije sintakse. Najčešće se koristi SCSS sintaksa (.scss). To je nadskup CSS-a, što znači da je sav valjani CSS također i valjani SCSS. Uvučena sintaksa (.sass) je manje uobičajena, a koristi uvlačenje, umjesto vitičastih zagrada, za ugniježđivanje izjava i nove retke umjesto točke-zareza za njihovo razdvajanje.[17]

[15] CSS Preprocessing: Tools and Automation Techniques. URL: https://doi.org/10.3390/info9010017 (2023-06-05)
[16] Lambdatest. Comprehensive CSS Preprocessor Comparison: Sass vs LESS vs Stylus. Sass – Syntactically Awesome Style Sheets. URL: https://www.lambdatest.com/blog/css-preprocessors-sass-vs-less-vs-stylus-with-examples/ (2023-06-10)
[17] Sass. Sass Basics. URL: https://sass-lang.com/guide (2023-06-10)



Slika 8. SCSS sintaksa

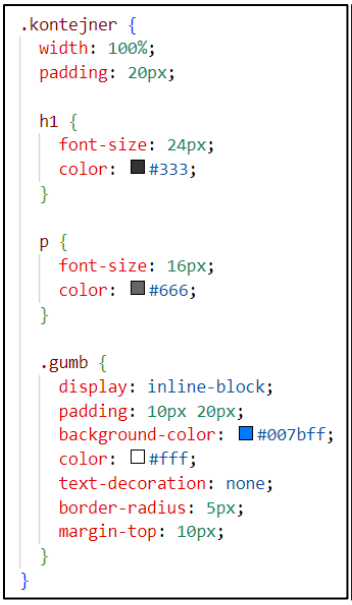


Slika 9. Sass sintaksa

U Sassu se mogu stvoriti manji fragmenti CSS-a koji se mogu umetnuti u druge Sass datoteke. Ovo je odličan način za organizaciju CSS-a i olakšavanje održavanja. Takav fragment naziva se "parcijal" i obično ima naziv s podvlakom na početku, poput "_partial.scss". Time Sass zna da je datoteka samo parcijal i da se ne bi trebala generirati u zasebnu CSS datoteku. Da bi se koristili Sass parcijali, potrebno je upotrijebiti pravilo "@use".[18] Također, uvodi koncept rezerviranih mjesta selektora, koji predstavljaju ponovno upotrebljive blokove stilova. Ovi rezervirani selektori mogu se proširiti i naslijediti od strane drugih selektora, što smanjuje potrebu za dupliciranjem koda i promovira bazu koda koja je lakša za održavanje. Ova značajka omogućuje efikasno organiziranje i ponovno korištenje stilova, što pridonosi boljoj modularnosti i čitljivosti CSS koda u projektu.[19]

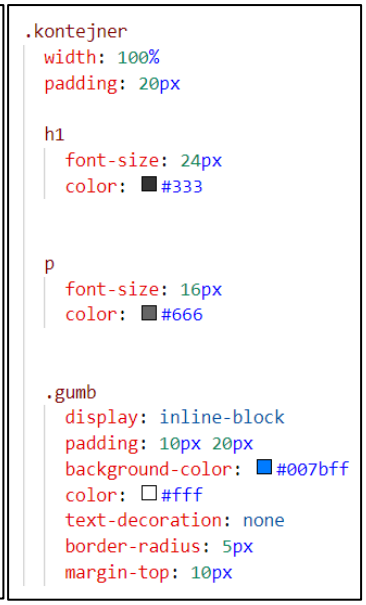
Sass omogućuje ugniježđivanje na način koji prati vizualnu hijerarhiju HTML-a. Međutim, važno je biti oprezan jer previše ugniježđenih pravila može rezultirati pretjerano kvalificiranim CSS-om koji bi mogao biti težak za održavanje i općenito se smatra lošom praksom.[20]

[18] Isto
[19] Medium. The Power of CSS Preprocessors: Less vs. Sass vs. Stylus. URL: https://medium.com/@taimoorshoaib141/the-power-of-css-preprocessors-less-vs-sass-vs-stylus-83e1c3c41c44 (2023-06-10)
[20] Sass. Sass Basics. URL: https://sass-lang.com/guide (2023-06-10)



```scss
.kontejner {
  width: 100%;
  padding: 20px;

  h1 {
    font-size: 24px;
    color: #333;
  }

  p {
    font-size: 16px;
    color: #666;
  }

  .gumb {
    display: inline-block;
    padding: 10px 20px;
    background-color: #007bff;
    color: #fff;
    text-decoration: none;
    border-radius: 5px;
    margin-top: 10px;
  }
}
```

Slika 10. SCSS sintaksa



```sass
.kontejner
  width: 100%
  padding: 20px

  h1
    font-size: 24px
    color: #333


  p
    font-size: 16px
    color: #666


  .gumb
    display: inline-block
    padding: 10px 20px
    background-color: #007bff
    color: #fff
    text-decoration: none
    border-radius: 5px
    margin-top: 10px
```

Slika 11. Sass sintaksa

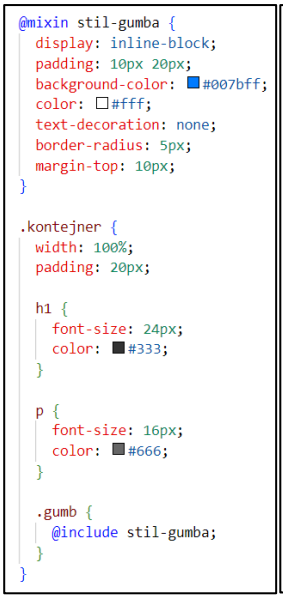
```css
.kontejner {
  width: 100%;
  padding: 20px;
}

.kontejner h1 {
  font-size: 24px;
  color: #333;
}

.kontejner p {
  font-size: 16px;
  color: #666;
}

.kontejner .gumb {
  display: inline-block;
  padding: 10px 20px;
  background-color: #007bff;
  color: #fff;
  text-decoration: none;
  border-radius: 5px;
  margin-top: 10px;
}
```

Slika 12. CSS sintaksa

Dakle, u Sass-u se koristi ugniježđivanje kako bi se unutar '.kontejner' pravila, definirala pravila za 'h1', 'p', i '.gumb'. Ovo olakšava čitanje i pisanje koda, posebno kod složenije strukture i dublje hijerarhije. U standardnom CSS-u, moraju se ručno napisati dulji selektori kako bi se postigao isti rezultat. To može dovesti do ponavljanja koda, što može povećati vjerojatnost grešaka i otežati održavanje koda.

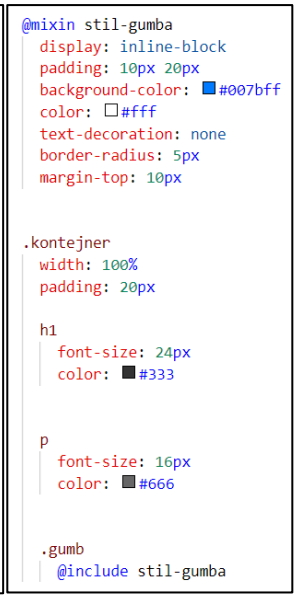
Mixin omogućuje kreiranje grupa CSS deklaracija koje se mogu ponovno koristiti na cijeloj mrežnoj stranici. Time pomaže u održavanju DRY programiranja (Don't Repeat Yourself) Sass koda. Dodatno, mixin se može prilagoditi proslijeđivanjem različitih vrijednosti, što ga čini iznimno fleksibilnim.[21]

[21] Sass. Sass Basics. URL: https://sass-lang.com/guide (2023-06-10)



```scss
@mixin stil-gumba {
  display: inline-block;
  padding: 10px 20px;
  background-color: #007bff;
  color: #fff;
  text-decoration: none;
  border-radius: 5px;
  margin-top: 10px;
}

.kontejner {
  width: 100%;
  padding: 20px;

  h1 {
    font-size: 24px;
    color: #333;
  }

  p {
    font-size: 16px;
    color: #666;
  }

  .gumb {
    @include stil-gumba;
  }
}
```

Slika 13. SCSS sintaksa



```sass
@mixin stil-gumba
  display: inline-block
  padding: 10px 20px
  background-color: #007bff
  color: #fff
  text-decoration: none
  border-radius: 5px
  margin-top: 10px


.kontejner
  width: 100%
  padding: 20px

  h1
    font-size: 24px
    color: #333


  p
    font-size: 16px
    color: #666


  .gumb
    @include stil-gumba
```

Slika 14. Sass sintaksa

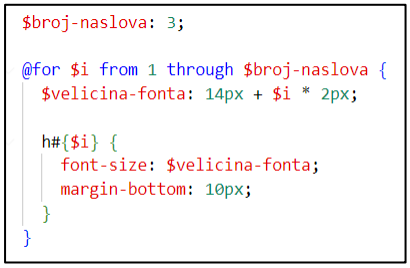
```css
.kontejner {
  width: 100%;
  padding: 20px;
}

.kontejner h1 {
  font-size: 24px;
  color: #333;
}

.kontejner p {
  font-size: 16px;
  color: #666;
}

.kontejner .gumb {
  display: inline-block;
  padding: 10px 20px;
  background-color: #007bff;
  color: #fff;
  text-decoration: none;
  border-radius: 5px;
  margin-top: 10px;
}
```

Slika 15. CSS sintaksa

U primjeru mixin-a u Sass-u, definira se '@mixin stil-gumba', koji sadrži stilove za '.button' element. Unutar '.kontejner .gumb' selektora koristi se '@include stil-gumba' kako bi se umetnuli stilovi. Time se postigne ponovno korištenje koda i smanjenjenje ponavljanja. U standardnom CSS-u, bez mixin-a, moralo bi se ručno napisati stilove za '.gumb' svaki put kad se žele primijeniti isti stilovi na taj element. To može dovesti do ponavljanja koda i otežati održavanje, pogotovo ako se želi promijeniti izgled gumba na više mjesta u kodu.

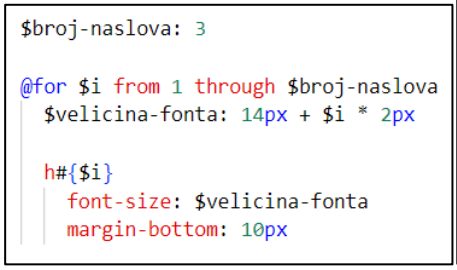
Mixin u predprocesorima kao što je Sass, omogućuje stvaranje ponovno upotrebljivih dijelova koda, čime olakšava održavanje, smanjuje količinu koda i povećava čitljivost CSS-a. To je posebno korisno za stilove koji se često koriste na različitim elementima u projektu.

Zajedno s ostalim CSS predprocesorima, Sass omogućuje korištenje naprednih značajki kao što su varijable, petlje i matematičke operacije:



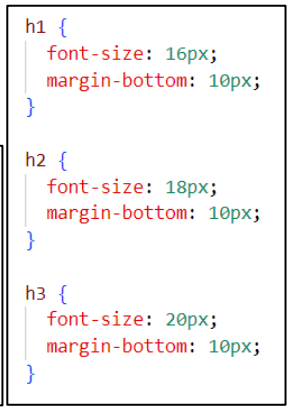
```scss
$broj-naslova: 3;

@for $i from 1 through $broj-naslova {
  $velicina-fonta: 14px + $i * 2px;

  h#{$i} {
    font-size: $velicina-fonta;
    margin-bottom: 10px;
  }
}
```

Slika 16. SCSS sintaksa



```sass
$broj-naslova: 3

@for $i from 1 through $broj-naslova
  $velicina-fonta: 14px + $i * 2px

  h#{$i}
    font-size: $velicina-fonta
    margin-bottom: 10px
```

Slika 17. Sass sintaksa



```css
h1 {
  font-size: 16px;
  margin-bottom: 10px;
}

h2 {
  font-size: 18px;
  margin-bottom: 10px;
}

h3 {
  font-size: 20px;
  margin-bottom: 10px;
}
```

Slika 18. CSS sintaksa

U ovom primjeru, koristi se varijabla '$broj-naslova' kako bi se kontrolirao broj naslova. Korištenjem petlje '@for', generiramo stilove za naslove s povećanim veličinama fonta. Matematička operacija '$velicina-fonta: 14px + $i * 2px;' osigurava da svaki naslov ima veći font od prethodnog.

Primjer prikazuje kako se pomoću Sass predprocesora može efikasno generirati više stilova s različitim svojstvima koristeći varijable, petlje i matematičke operacije, što olakšava održavanje i skaliranje stilova u većim projektima.

Sass pruža ograničene mogućnosti proceduralnog programiranja putem svojih kontrolnih direktiva i izraza. Moguće je primijeniti stilizaciju na temelju određenih uvjeta ili primijeniti isti stil više puta s različitim varijacijama. Kontrolne direktive u Sass-u omogućuju integraciju uvjetne logike u Sass kod. Ove direktive omogućuju da se kod razgranava u različite smjerove na temelju raznolikih uvjeta te da se prolazi kroz kod sve dok specifični uvjeti ne budu zadovoljeni. Uključivanjem kontrolnih direktiva kao što su '@if', '@each', '@for' i '@while', omogućujemo Sass-u da preuzme veći dio posla koji bi inače zahtijevao ponavljanje stilskih deklaracija i uvjetne logike. Kontrolne direktive obično se primjenjuju u mixin-ima. Sass pruža nekoliko at-pravila koja pomažu kontrolirati emitiranje stilova ili njihovu emisiju više puta s manjim varijacijama. Ta pravila koriste se u mixin-ima i funkcijama kako bi olakšala pisanje manjih algoritama za olakšano kreiranje Sass koda. Sass podržava četiri osnovna pravila za kontrolu toka.[22]

Direktiva '@if' zajedno s povezanim '@else if' i '@else' omogućuje da se uključi određeni Sass kod u izlazni CSS samo ako su zadovoljeni specifični uvjeti.

Direktiva '@each' će obraditi elemente unutar liste ili mape. Iako je ova funkcionalnost veoma moćna, njena sintaksa je zapravo prilično jednostavna.

Direktiva @for koristi se za izvođenje grupe naredbi određeni broj puta.

Na kraju, ukoliko želite ponoviti izvršenje naredbi više puta, dok i dalje zadržavate kontrolu na temelju uvjeta, možete koristiti @while direktivu. Kao što samo ime sugerira, @while direktiva će nastaviti generirati CSS koji proizvodi izjave dok uvjet ostaje istinit.[23]

[22] Coding Ninjas. Control Directives and expressions in Sass. URL: https://www.codingninjas.com/studio/library/control-directives-and-expressions-in-sass (2023-08-25)
[23] Tutorials teacher. Sass - Control Directives .URL: https://www.tutorialsteacher.com/sass/sass-control-directives#google_vignette (2023-08-25)

3.2. Less: Leaner Stylesheets

LESS koristi osnovnu CSS sintaksu, ali s dodatkom ekstenzije .less. To znači da će osoba koja je već upoznata s CSS-om lako naučiti Less. Međutim, Less nudi dodatne mogućnosti koje nisu dostupne u čistom CSS-u, kao što je uporaba @ znaka za definiranje varijabli. To omogućuje veću fleksibilnost u definiranju i ponovnoj uporabi vrijednosti na više mjesta u stilovima, što je nešto što se ne može postići u standardnom CSS-u.[24] Alexis Sellier pokrenio je Less 2009. godine, što je bilo tri godine nakon prvotnog lansiranja Sassa 2006. Less je JavaScript biblioteka koja proširuje mogućnosti osnovnog CSS-a, uključujući mixine, varijable, gniježđenje i petlje. Ima službenu dokumentaciju koja opisuje jezik i Less.js, JavaScript alat koji omogućuje pretvaranje Less stilova u CSS stilove. Također, dodaje nekoliko korisnih dodataka u odnosu na čisti CSS, što ga čini praktičnijim. Zbog tih malih, ali moćnih dodataka, učenje Lessa je brz i jednostavan proces.[25] U Less-u je moguće koristiti matematičke operacije unutar CSS-a, što omogućuje obavljanje izračuna na varijablama. Ova mogućnost je iznimno korisna za stvaranje responzivnog dizajna ili dinamičnih stilova koji se temelje na korisničkom unosu. Tako omogućuje jednostavno mijenjanje stilova ovisno o različitim uvjetima, što je posebno korisno za postizanje prilagodljivih i interaktivnih mrežnih stranica.[26]
Primjer sintakse Less predprocesora izgleda ovako:

```
@snop-fontova: Helvetica, sans-serif;
@primarna-boja: #333;

body {
  font: 100% @snop-fontova;
  color: @primarna-boja;
}
```

Slika 19. Less sintaksa

Less, isto kao i Sass, pruža mogućnost korištenja ugniježđivnja umjesto kaskadnog slaganja ili u kombinaciji s njim.[27]

[24] Simplilearn. Learn all about CSS Preprocessors with Examples. URL: https://www.simplilearn.com/tutorials/css-tutorial/css-preprocessors (2023-07-05)
[25] Less CSS. URL: https://lesscss.org/ (2023-07-06)
[26] Medium. The Power of CSS Preprocessors: Less vs. Sass vs. Stylus. URL: https://medium.com/@taimoorshoaib141/the-power-of-css-preprocessors-less-vs-sass-vs-stylus-83e1c3c41c44 (2023-07-01)
[27] Less CSS. URL: https://lesscss.org/ (2023-07-06)

```less
.kontejner {
  width: 100%;
  padding: 20px;

  h1 {
    font-size: 24px;
    color: #333;
  }

  p {
    font-size: 16px;
    color: #666;
  }

  .gumb {
    display: inline-block;
    padding: 10px 20px;
    background-color: #007bff;
    color: #fff;
    text-decoration: none;
    border-radius: 5px;
    margin-top: 10px;
  }
}
```

Slika 20. Less sintaksa

```css
.kontejner {
  width: 100%;
  padding: 20px;
}

.kontejner h1 {
  font-size: 24px;
  color: #333;
}

.kontejner p {
  font-size: 16px;
  color: #666;
}

.kontejner .gumb {
  display: inline-block;
  padding: 10px 20px;
  background-color: #007bff;
  color: #fff;
  text-decoration: none;
  border-radius: 5px;
  margin-top: 10px;
}
```

Slika 21. CSS sintaksa

Kao što je prikazano, sintaksa ugniježđivanja u Less-u izgleda gotovo identično kao u Sass-u. Ima istu strukturu i mogućnost grupiranja pravila unutar roditeljskih selektora. U standardnom CSS-u morali bismo koristiti duge selektore kako bismo postigli isti rezultat. Ugniježđivanje u predprocesorima poput Less-a pomaže u organizaciji koda i smanjenju ponavljanja, što čini CSS kod čišćim i lakšim za upravljanje.

Mixin u Less-u omogućuje definiranje ponovno upotrebljivih blokova koda koji se mogu umetnuti na različita mjesta u CSS-u.[28] Kao i sa Sass-om, usporedit će se primjer mixin-a u Less-u s primjerom standardnog CSS-a kako bi se vidjelo na koji način mixin olakšava ponovno korištenje koda.

[28] Less CSS. URL: https://lesscss.org/ (2023-07-10)

```less
.stil-gumba() {
  display: inline-block;
  padding: 10px 20px;
  background-color: #007bff;
  color: #fff;
  text-decoration: none;
  border-radius: 5px;
  margin-top: 10px;
}

.kontejner {
  width: 100%;
  padding: 20px;

  h1 {
    font-size: 24px;
    color: #333;
  }

  p {
    font-size: 16px;
    color: #666;
  }

  .gumb {
    .stil-gumba();
  }
}
```

Slika 22. Less sintaksa

```css
.kontejner {
  width: 100%;
  padding: 20px;
}

.kontejner h1 {
  font-size: 24px;
  color: #333;
}

.kontejner p {
  font-size: 16px;
  color: #666;
}

.kontejner .gumb {
  display: inline-block;
  padding: 10px 20px;
  background-color: #007bff;
  color: #fff;
  text-decoration: none;
  border-radius: 5px;
  margin-top: 10px;
}
```

Slika 23. CSS sintaksa

U primjeru mixin-a u Less-u, definira se '.stil-gumba' mixin, koji sadrži stilove za '.gumb' element. Unutar '.kontejner .gumb' selektora koristi se '.stil-gumba()' kako bi se umetnuli stilovi. Time se postiže ponovno korištenje koda i smanjenje ponavljanja. U standardnom CSS-u, bez mixin-a, moraju se ručno napisati stilovi za '.gumb' svaki put kada se žele primijeniti isti stilovi na taj element. To dovodi do ponavljanja koda i otežava održavanje, pogotovo ukoliko se želi promijeniti izgled gumba na više mjesta u kodu.
Mixin u predprocesorima kao što je Less omogućuje stvaranje ponovno upotrebljivih dijelova koda, olakšava održavanje, smanjuje količinu koda i povećava čitljivost CSS-a.

3.3. Stylus

Stylus je CSS predprocesor koji omogućuje generiranje CSS-a na efikasniji i dinamičniji način. Nudi logičku sintaksu za definiranje stilskih blokova CSS-a te sadrži programski jezik koji čini proces generiranja CSS-a vrlo dinamičnim.[29] Stylus je predprocesor s fleksibilnom sintaksom koja omogućuje izostavljanje zagrade, točke-zareza i dvotočke. Također, Stylus podržava korištenje varijabli, mixina, funkcija i drugih značajki.[30] Bivši programer za Node.js, TJ Holowaychuk, lansirao je Stylus 2010. Stylus je napisan kao Node.js aplikacija, što ga čini

[29] O'reilly. URL: https://www.oreilly.com/library/view/express-web-application/9781849696548/ch06.html (2023-07-10)
[30] Linked In. URL: https://www.linkedin.com/advice/0/what-benefits-drawbacks-using-sass-less-stylus (2023-07-10)

savršenim dodatkom za JavaScript (JS) okolinu. Inspiriran je logičkim sposobnostima Sass-a i jednostavnošću Less-a. Iako Stylus i dalje ima popularnost među Node.js programerima, nije uspio postići značajan udio na tržištu. Jedna od njegovih prednosti u odnosu na Sass ili Less je bogatstvo ugrađenih funkcija i sposobnost da se nosi s zahtjevnim računalstvom.[31] Stylus pruža programerima mogućnost proširivanja jezika stvaranjem vlastitih funkcija i dodataka. Ova fleksibilnost omogućuje stvaranje prilagođenih rješenja i lako integriranje Stylusa s drugim alatima. Na taj način, programeri imaju mogućnost prilagoditi Stylus prema svojim potrebama, što otvara širok spektar mogućnosti za razvoj i kreativnost pri oblikovanju stilova na mrežnim stranicama ili aplikacijama.[32]

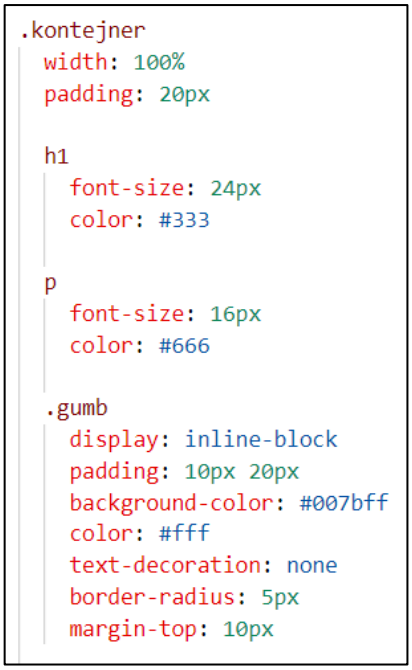
Primjer sintakse Stylus predprocesora izgleda ovako:

```
font-stack = Helvetica, sans-serif
primary-color = #333

body
  font: 100% font-stack
  color: primary-color
```

Slika 24. Stylus sintaksa

Jednako kao i u prva dva predprocesora, ugniježđivanje u Stylus-u omogućuje organiziranje CSS pravila hijerarhijski, na sličan način kao u Sass-u i Less-u. Usporedit će se primjer ugniježđivanja u Stylus-u s primjerom standardnog CSS-a.

[31] Lambdatest. URL: https://www.lambdatest.com/blog/css-preprocessors-sass-vs-less-vs-stylus-with-examples/ (2023-07-10)
[32] Medium. The Power of CSS Preprocessors: Less vs. Sass vs. Stylus. URL: https://medium.com/@taimoorshoaib141/the-power-of-css-preprocessors-less-vs-sass-vs-stylus-83e1c3c41c44 (2023-07-01)

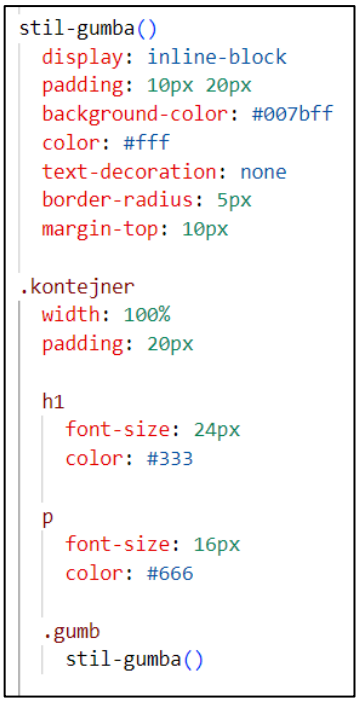


Slika 25. Stylus sintaksa

Slika 26. CSS sintaksa

U primjeru ugniježđivanja u Stylus-u može se primijetiti da se stilovi za 'h1', 'p', i '.gumb' nalaze uvučeni (bez vitičastih zagrada) ispod '.kontejner'. Stilovi unutar roditeljskih selektora uvjetuju se uvlačenjem, što rezultira čišćim i kraćim kodom. U usporedbi s time, u standardnom CSS-u morali bi se koristiti dulji selektori, kao što je prikazano u primjeru, kako bi se postigao isti rezultat. Ovo može povećati količinu koda i učiniti ga teže čitljivim, posebno za složenije strukture. Dakle, Stylus, kao i drugi CSS predprocesori, omogućuje organizaciju koda kroz ugniježđivanje, što čini stilizaciju lakšom i olakšava održavanje.

Za prikaz i objašnjenje mixin-a u Stylus-u, usporedit će se s primjerom standardnog CSS-a.



```
stil-gumba()
  display: inline-block
  padding: 10px 20px
  background-color: #007bff
  color: #fff
  text-decoration: none
  border-radius: 5px
  margin-top: 10px

.kontejner
  width: 100%
  padding: 20px

  h1
    font-size: 24px
    color: #333

  p
    font-size: 16px
    color: #666

  .gumb
    stil-gumba()
```

Slika 27. Stylus sintaksa

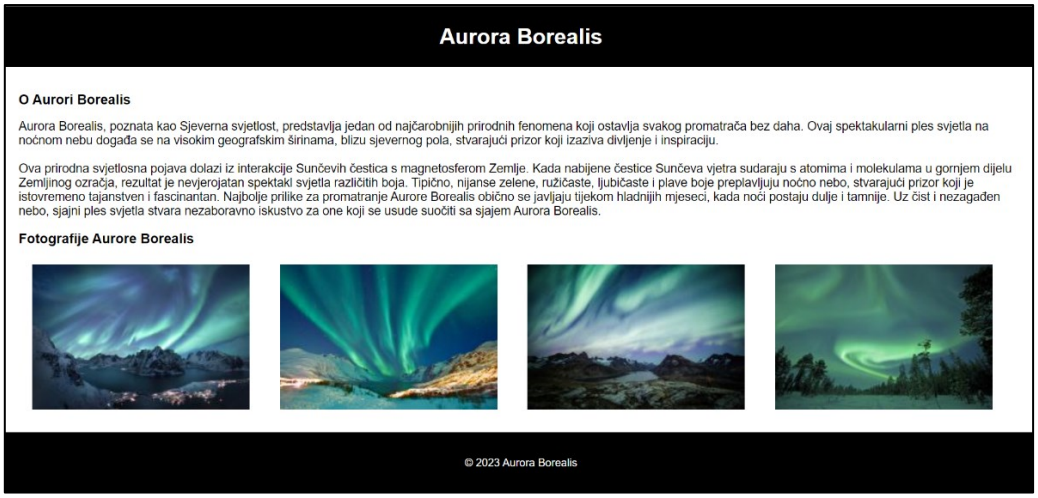
```
.kontejner {
  width: 100%;
  padding: 20px;
}

.kontejner h1 {
  font-size: 24px;
  color: #333;
}

.kontejner p {
  font-size: 16px;
  color: #666;
}

.kontejner .gumb {
  display: inline-block;
  padding: 10px 20px;
  background-color: #007bff;
  color: #fff;
  text-decoration: none;
  border-radius: 5px;
  margin-top: 10px;
}
```

Slika 28. CSS sintaksa

Gotovo identično kao i za Less, u primjeru mixin-a u Stylus-u, definira se 'stil-gumba' mixin, koji sadrži stilove za '.gumb' element. Tada se unutar '.kontejner .gumb' selektora koristi 'stil-gumba()' kako bi se umetnuli stilovi. Time se postiže ponovno korištenje koda i smanjenje ponavljanja. U standardnom CSS-u, bez mixin-a, moraju se ručno napisati stilove za '.gumb' svaki put kad želimo primijeniti iste stilove na taj element.

4. PRIMJER MREŽNE STRANICE

U nastavku prikazat će se stranica koja je stilizirala prvobitno sa čistim CSS-om, a zatim sa sve 3 varijante CSS predprocesora.



Slika 29. Primjer stranice

U nastavku je prikazan HTML i čist CSS:

```html
<!DOCTYPE html>
<html lang="hr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="../stilovi/glavni.css">
  <title>Aurora Borealis</title>
</head>
<body>
  <header>
    <h1>Aurora Borealis</h1>
  </header>
  <main>
    <section class="o-aurori">
      <h2>O Aurori Borealis</h2>
      <p>Aurora Borealis, poznata kao Sjeverna svjetlost, predstavlja jedan
        <br>
        <br>
        Ova prirodna svjetlosna pojava dolazi iz interakcije Sunčevih čest
        Najbolje prilike za promatranje Aurore Borealis obično se javljaju
      </p>
    </section>
    <section class="fotografije">
      <h2>Fotografije Aurore Borealis</h2>
      <img src="../slike/auro1.jpg" class="fotografija">
      <img src="../slike/auro.jpg" class="fotografija">
      <img src="../slike/auro2.jpg" class="fotografija">
      <img src="../slike/auro3.jpg" class="fotografija">
    </section>
  </main>
  <footer>
    <p>&copy; 2023 Aurora Borealis</p>
  </footer>
</body>
</html>
```

Slika 30. HTML

```css
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
}
header {
  background-color: #000;
  color: #fff;
  text-align: center;
  padding: 5px;
}
main {
  padding: 20px;
}
.o-aurori h2 {
  font-size: 20px;
}
.o-aurori p {
  font-size: 18px;
}
.fotografije h2 {
  font-size: 20px;
}
.fotografija {
  width: 320px;
  height: 320px/1.5;
  background-color: #3498db;
  margin: 10px 20px;
  display: inline-block;
}
footer {
  background-color: #000;
  color: #fff;
  text-align: center;
  padding: 20px;
}
```

Slika 31. CSS sintaksa

Zatim sve 3 varijante predprocesora:

```scss
$osnovni-font: Arial, sans-serif;
$primarna-boja: #3498db;
$pozadinska-boja: #000;
$tekst-boja: #fff;

$broj-fotografija: 4;

$osnovna-velicina-fonta: 16px;
$korak-velicine-fonta: 2px;

$velicina-fotografije: 320px;
$margina-fotografije: 10px 20px;

body {
  font-family: $osnovni-font;
  margin: 0;
  padding: 0;
}

header {
  background-color: $pozadinska-boja;
  color: $tekst-boja;
  text-align: center;
  padding: 5px;
}
```

Slika 32. SCSS sintaksa

```scss
main {
  padding: 20px;
}

.o-aurori h2 {
  font-size: $osnovna-velicina-fonta + $korak-velicine-fonta * 2;
}

.o-aurori p {
  font-size: $osnovna-velicina-fonta + $korak-velicine-fonta;
}

.fotografije h2 {
  font-size: $osnovna-velicina-fonta + $korak-velicine-fonta * 2;
}

.fotografija {
  width: $velicina-fotografije;
  height: $velicina-fotografije / 1.5;
  background-color: $primarna-boja;
  margin: $margina-fotografije;
  display: inline-block;
}

footer {
  background-color: $pozadinska-boja;
  color: $tekst-boja;
  text-align: center;
  padding: 20px;
}
```

Slika 33. SCSS sintaksa

```less
@osnovni-font: Arial, sans-serif;
@primarna-boja: #3498db;
@pozadinska-boja: #000;
@tekst-boja: #fff;

@broj-fotografija: 4;

@osnovna-velicina-fonta: 16px;
@korak-velicine-fonta: 2px;

@velicina-fotografije: 320px;
@margina-fotografije: 10px 20px;

body {
  font-family: @osnovni-font;
  margin: 0;
  padding: 0;
}

header {
  background-color: @pozadinska-boja;
  color: @tekst-boja;
  text-align: center;
  padding: 5px;
}
```

Slika 34. Less sintaksa

```less
main {
  padding: 20px;
}

.o-aurori h2 {
  font-size: @osnovna-velicina-fonta + @korak-velicine-fonta * 2;
}

.o-aurori p {
  font-size: @osnovna-velicina-fonta + @korak-velicine-fonta;
}

.fotografije h2 {
  font-size: @osnovna-velicina-fonta + @korak-velicine-fonta * 2;
}

.fotografija {
  width: @velicina-fotografije;
  height: @velicina-fotografije / 1.5;
  background-color: @primarna-boja;
  margin: @margina-fotografije;
  display: inline-block;
}

footer {
  background-color: @pozadinska-boja;
  color: @tekst-boja;
  text-align: center;
  padding: 20px;
}
```

Slika 35. Less sintaksa

```
osnovni-font = Arial, sans-serif
primarna-boja = #3498db
pozadinska-boja = #000
tekst-boja = #fff

broj-fotografija = 4

osnovna-velicina-fonta = 16px
korak-velicine-fonta = 2px

velicina-fotografije = 320px
margina-fotografije = 10px 20px

body
  font-family: osnovni-font
  margin: 0
  padding: 0

header
  background-color: pozadinska-boja
  color: tekst-boja
  text-align: center
  padding: 5px
```

Slika 36. Stylus sintaksa

```
main
  padding: 20px

.o-aurori h2
  font-size: osnovna-velicina-fonta + korak-velicine-fonta * 2

.o-aurori p
  font-size: osnovna-velicina-fonta + korak-velicine-fonta

.fotografije h2
  font-size: osnovna-velicina-fonta + korak-velicine-fonta * 2

.fotografija
  width: velicina-fotografije
  height: velicina-fotografije / 1.5
  background-color: primarna-boja
  margin: margina-fotografije
  display: inline-block

footer
  background-color: pozadinska-boja
  color: tekst-boja
  text-align: center
  padding: 60px
```

Slika 37. Stylus sintaksa

5. PREDNOSTI I NEDOSTACI CSS PREDPROCESORA

Navedeni predprocesori donose napredne značajke programskih jezika visoke razine u CSS, kao što su varijable, petlje, matematičke operacije te složenije funkcije. Predprocesori također omogućuju korištenje mixina, funkcije dizajnirane za smanjenje dupliciranog koda. Ove funkcije pridonose principu DRY programiranja (Don't Repeat Yourself) jer smanjuju ponavljanje istog koda. Osim što olakšavaju rad, mixini omogućuju brzo i jednostavno mijenjanje veličine fonta ili boje, što u klasičnom CSS-u može zahtijevati puno više truda. Iako su prednosti ovih predprocesora očigledne, postoji nekoliko nedostataka. Upotreba mixina može rezultirati ponavljanjem različitih svojstava u kodu, što može učiniti kod vizualno čitljivijim, ali istovremeno povećava veličinu datoteke CSS-a. U velikim projektima, ovo može postati problem ili nedostatak jer povećana veličina datoteke može usporiti vrijeme učitavanja stranice.[33]

Korištenjem varijabli, vrijednosti koje se ponavljaju na više mjesta mogu se definirati na jednom mjestu, a to olakšava promjene i održavanje stilova. Upotreba petlji omogućava generiranje velikog broja stilova temeljenih na istom uzorku. Ovo je posebno korisno kod stvaranja npr. tablica, lista ili galerija. U većim projektima, upotreba petlji i varijabli

[33] IvyPanda. URL: https://ivypanda.com/essays/pros-and-cons-of-using-css-preprocessors/ (2023-07-28)

omogućava lako skaliranje stilova bez potrebe za ručnim ponavljanjem kodova. Matematičke operacije omogućavaju dinamičke izračune, kao što je prilagodba veličina, udaljenosti i boja na temelju nekog parametra. Također bitna prednost je mogućnost dodavanja varijabli, mixina i funkcija. Ako se usporedi čist CSS s CSS predprocesorom, može se vidjeti da čist CSS pruža izravnu, ali ograničenu fleksibilnost. S druge strane, upotreba CSS predprocesora omogućuje dodavanje varijabli i funkcija, što donosi novu dimenziju i proširuje mogućnosti CSS-a, omogućavajući tako lakši i učinkovitiji razvoj. Osim toga, CSS predprocesor čini kod organiziranijim i čišćim, što je velika prednost za održavanje i nadogradnju stilova. Zatim, CSS predprocesori omogućuju posebnu funkcionalnost spajanja više stilskih listova u jedan. Mogu se stvoriti različite datoteke za stilove koji se primjenjuju na različite zaslone ili stranice, a zatim sve te datoteke je moguće uvesti u glavnu CSS datoteku. Na taj način, na mrežno mjesto uvede se samo glavna datoteka, dok će interne datoteke biti automatski učitane s poslužitelja. Ovo omogućuje bolju organizaciju i upravljanje stilovima te smanjuje opterećenje prilikom učitavanja mrežne stranice. Osnovni CSS ima veliki nedostatak jer za svaki element morate pisati stilove iznova i iznova. Ovo je suprotno principu programskih jezika koji teže smanjenju ponavljanja koda. Upravo ovdje CSS predprocesori dolaze u pomoć, jer omogućuju uštedu vremena. Možete jednom napisati zajedničke stilove koji se mogu uvesti umjesto ponovnog pisanja za svaki element. To značajno pojednostavljuje stiliziranje i pomaže vam da održavate čist i organiziran kod. [34]

[34] Techaffinity. Advantages of CSS Preprocessors. URL: https://techaffinity.com/blog/advantages-of-css-preprocessors/ (2023-07-28)

```css
.glavnidiv{
  margin:5px;
  padding:2px;
  border:1px double;
  }
.div1{
  margin:5px;
  padding:2px;
  border:2px double;
  }
.div2{
  margin:5px;
  padding:2px;
  border:3px double;
  }
```

Slika 38. Kod u CSS-u

```
.glavnidiv{
  margin:5px;
  padding:2px;
  border:1px double;
  }
.div1{
  .glavnidiv;
  border:2px;
  }
.div2{
  .glavnidiv;
  border:3px;
  }
```

Slika 39. Kod pomoću CSS predprocesora

Korištenjem CSS predprocesora može se izbjeći ponavljanje koda te organizirati svoj kod na efikasan način. Ova kombinacija uštedi mnogo vremena tijekom kodiranja. Može se napisati ista količinu stilova u puno kraćem vremenskom razdoblju, što će također olakšati skaliranje projekta. Upotreba ugniježđivanja klasa u CSS-u olakšava ciljanje određenih DOM (Document Object Model) elemenata i štedi vrijeme te značajno olakšava uređivanje CSS datoteka. Također, CSS predprocesori sadrže mnoge korisne ugrađene funkcije, a jedna od njih je funkcija za potamnjivanje i posvjetljivanje boja. Može se jednostavno primijeniti tamnjenje na određenu boju pomoću koda boje i postotka. Ovo nije bilo moguće koristiti s osnovnim CSS-om.[35]

CSS predprocesori donose brojne prednosti, no kao i većina alata, imaju i nekoliko nedostataka koji će se navesti u nastavku.

Predprocesori imaju postupak kompilacije koji čini brojeve redaka CSS-a nebitnima kada se pokušavaju ispraviti pogreške u kodu. No, uklanjanje grešaka može biti dvostruko teže od samog programiranja, što predstavlja veliki nedostatak. Karte izvora pružaju rješenje, ali zahtijevaju postavljanje i nisu podržane u svim preglednicima, posebno u onima u kojima se često javljaju pogreške. Bez njih, programeri su prepušteni traženju pravila, nadajući se da će pronaći ono što traže. Zatim, jedan od nedostataka je da vrijeme kompilacije može biti znatno

[35] Techaffinity. Advantages of CSS Preprocessors. URL: https://techaffinity.com/blog/advantages-of-css-preprocessors/ (2023-07-28)

sporo, čak i uz korištenje najboljih alata na najbržem računalu. Korištenjem CSS predprocesora može doći do stvaranja iznimno velikih CSS datoteka. Iako izvorne datoteke mogu biti male, generirani CSS može biti ogroman, što je važan aspekt koji treba uzeti u obzir. Bitno je biti svjestan da upotrebom CSS predprocesora gubimo dio kontrole nad konačnim izlazom CSS-a. Korištenje CSS predprocesora zahtijeva dodatni alat. Autori koda ne bi trebali biti ograničeni na određeni uređivač samo da bi mogli koristiti taj alat. Također, dodavanje ovih dodatnih elemenata uvodi složenost u proces. Potrebno je razumjeti, nadograđivati i održavati te dodatne komponente, što sve zajedno povećava troškove i stvara veći rizik od mogućih problema. Implementiranje CSS predprocesora u tehnološki skup može biti jednostavno, ali uklanjanje istog jednostavnom naredbom nije tako lako ako odlučimo odustati od njih.[36] Iako su se CSS predprocesori poput Sass-a, Less-a ili Stylus-a pokazali iznimno korisnima tijekom rada na front-end razvoju i imaju mnoge prednosti, postoje situacije u kojima se ne moraju nužno koristiti ovi predprocesori. Ako se namjera korištenja Sass-a ili drugih CSS predprocesora ograničava na upotrebu varijabli, tada čist CSS može vrlo dobro zadovoljiti sve potrebe. Prilagođena svojstva CSS-a su korisnički definirane varijable koje se stvaraju pomoću oznake '-' i mogu se pristupiti pomoću funkcije var(). Ova prilagođena svojstva mogu se definirati unutar bilo kojeg elementa, iako je popularna konvencija da se varijable prilagođenih svojstava definiraju globalno unutar ':root' selektora.[37] Na nama je odgovornost temeljito razmotriti njihov utjecaj na naš radni proces prije nego donesemo odluku da ih instaliramo i koristimo. Potrebno je pažljivo procijeniti kako će njihova upotreba utjecati na produktivnost i održavanje sustava kako bismo donijeli informiranu odluku. [38]

[36] Adam Silver. The disadvantages of CSS preprocessors. URL: https://adamsilver.io/blog/the-disadvantages-of-css-preprocessors/ (2023-07-29)
[37] Lambdatest. URL: https://www.lambdatest.com/blog/css-preprocessors-sass-vs-less-vs-stylus-with-examples/ (2023-07-10)
[38] Adam Silver. The disadvantages of CSS preprocessors. URL: https://adamsilver.io/blog/the-disadvantages-of-css-preprocessors/ (2023-07-29)

6. ZAKLJUČAK

CSS predprocesori predstavljaju moćan alat koji omogućuje programerima i dizajnerima lakši i efikasniji način upravljanja stilskim kodom u mrežnom dizajnu. Prilikom odabira CSS predprocesora treba uzeti u obzir nekoliko čimbenika. Less ima poznatu sintaksu sličnu CSS-u, što ga čini odličnim za programere koji poznaju CSS. Sass, iako složeniji, nudi napredne značajke i bogat ekosustav. Stylus ima minimalistički i fleksibilan pristup s jednostavnom sintaksom što ga čini privlačnim za one koji cijene jednostavnost. Ključna prednost ovih predprocesora je korištenje varijabli, petlji, funkcija i mixin-a, što znatno olakšava ponovnu upotrebu koda i održavanje. Mogućnost definiranja varijabli za boje, fontove, veličine i druge vrijednosti omogućuje konzistentnost i brži razvoj. Međutim, postoje i određeni nedostaci CSS predprocesora. Najvažniji nedostatak je potreba za dodatnim korakom u procesu razvoja, što uključuje kompilaciju predprocesorskog koda u klasičan CSS kako bi ga mrežni preglednik mogao interpretirati. To može usporiti razvojni proces i povećati kompleksnost projekta. Također, zbog dodatnih koncepta i sintakse koju predprocesori donose, postojeći timovi koji su već upoznati samo s čistim CSS-om mogli bi zahtijevati dodatno vrijeme za prilagodbu i učenje novih tehnika.

U konačnici, izbor između korištenja CSS predprocesora i čistog CSS-a ovisi o složenosti projekta, timskim vještinama i ciljevima razvoja. Ako se radi o većem projektu s mnogo stilskog koda i potrebom za ponovnom upotrebom, prednosti CSS predprocesora znatno prevladavaju nad njihovim nedostacima. S druge strane, za manje projekte s jednostavnijim stilskim zahtjevima, čisti CSS može biti adekvatna opcija. U budućnosti, s razvojem tehnologija i novih pristupa mrežnom dizajnu, moguće je da će CSS predprocesori i dalje evoluirati i nuditi još naprednije funkcionalnosti, što će učiniti njihov izbor još zanimljivijim i važnijim u mrežnom razvoju.

7. LITERATURA

1. Adam Silver. The disadvantages of CSS preprocessors. URL: https://adamsilver.io/blog/the-disadvantages-of-css-preprocessors/ (2023-07-29)

2. Coding Ninjas. Control Directives and expressions in Sass. URL: https://www.codingninjas.com/studio/library/control-directives-and-expressions-in-sass (2023-08-25)

3. Coyier, C. Popularity of CSS Preprocessors. URL: https://css-tricks.com/poll-results-popularity-of-css-preprocessors/ (2023-06-05)

4. CSS-tricks. The “C” in CSS: The Cascade. URL: https://css-tricks.com/the-c-in-css-the-cascade/ (2023-06-02)

5. CSS Preprocessing: Tools and Automation Techniques. URL: https://doi.org/10.3390/info9010017 (2023-06-03)

6. Github. Primjer. URL: https://github.com/ddrazic/Primjer

7. IvyPanda. URL: https://ivypanda.com/essays/pros-and-cons-of-using-css-preprocessors/ (2023-07-28)

8. Lambdatest. Comprehensive CSS Preprocessor Comparison: Sass vs LESS vs Stylus. URL: https://www.lambdatest.com/blog/css-preprocessors-sass-vs-less-vs-stylus-with-examples/ (2023-06-04)

9. Less CSS. URL: https://lesscss.org/ (2023-07-06)

10. Linked In. URL: https://www.linkedin.com/advice/0/what-benefits-drawbacks-using-sass-less-stylus (2023-07-12)

11. MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (2023-06-02)

12. Medium. The Power of CSS Preprocessors: Less vs. Sass vs. Stylus. URL: https://medium.com/@taimoorshoaib141/the-power-of-css-preprocessors-less-vs-sass-vs-stylus-83e1c3c41c44 (2023-06-10)

13. O'reilly. URL: https://www.oreilly.com/library/view/express-web-application/9781849696548/ch06.html (2023-07-10)

14. Sass. Sass Basics. URL: https://sass-lang.com/guide (2023-06-10)

15. Shero. What is a CSS preprocessor? URL: https://sherocommerce.com/what-is-a-css-preprocessors-why-use-them/ (2023-06-05)

16. Simplilearn. Learn all about CSS Preprocessors with Examples. URL: https://www.simplilearn.com/tutorials/css-tutorial/css-preprocessors (2023-07-05)

17. Techaffinity. Advantages of CSS Preprocessors. URL: https://techaffinity.com/blog/advantages-of-css-preprocessors/ (2023-07-28)

18. Tutorials teacher. Sass - Control Directives .URL: https://www.tutorialsteacher.com/sass/sass-control-directives#google_vignette (2023-08-25)

19. Web Tech. CSS. URL: https://www.webtech.com.hr/css.php (2023-06-02)