Sass

U lekciji koja je pred vama, po prvi put izlazimo iz okvira standardnih frontend tehnologija koje web pregledači direktno razumeju. Naravno, u takve osnovne tehnologije ubrajaju se kompjuterski jezici HTML, CSS i JavaScript. Ovakve jezike nazivamo osnovnim, zato što su web pregledači u stanju da direktno razumeju njihov kod.

S obzirom na to da je primarna tema ovog kursa napredno stilizovanje web sajtova, ova i naredna lekcija biće posvećene dvama jezicima koji se takođe mogu koristiti za formulisanje stilizacije. Takvi jezici se drugačije nazivaju pretprocesorski jezici za stilizovanje.

Pojam pretprocesora

Pretprocesori su specijalni alati (kompjuterski programi) koji obavljaju prevođenje izvornog koda jednog jezika u neki drugi. U slučaju jezika za stilizovanje web sajtova, pretprocesori omogućavaju prevođenje koda nekog drugog jezika u CSS.

U ovom trenutku se sa pravom možete zapitati – *zbog čega su nam pretprocesori uopšte i potrebni?* Odgovor na ovo pitanje veoma je jednostavan. Pretprocesori, odnosno pretprocesorski jezici za stilizovanje, omogućavaju korišćenje različitih naprednih funkcionalnosti koje izvorno ne postoje u CSS jeziku. Sve ovo na kraju omogućava mnogo jednostavnije i produktivnije pisanje koda za stilizovanje web sajtova.

Šta je Sass?

Jedan od najpoznatijih pretprocesorskih jezika za stilizovanje jeste Sass. Naziv *Sass* zapravo je akronim pojma ***S**yntactically **A**wesome **S**tyle**s**heet*.

Slika 24.1. Sass logo

Sass je nastao 2006. godine i njegovi tvorci su Hampton Catlin, Natalie Weizenbaum i Chris Eppstein. Tokom godina, prošao je kroz brojne iteracije unapređivanja, pa iako nije jedini, sa pravom se može reći da je najpopularniji pretprocesorski jezik za stilizovanje. Tome u prilog ide i činjenica da je Sass danas sastavni deo nekoliko veoma popularnih frontend frameworka, od kojih je svakako najpopularniji Bootstrap.

Koji su preduslovi za korišćenje Sassa?

Sass je jezik za stilizovanje koji je neophodno prevesti u CSS kod kako bi bio upotrebljiv. Takvo prevođenje se može obaviti na nekoliko načina:

- korišćenjem nekog od specijalizovanih programa sa grafičkim korisničkim okruženjem (Koala, CodeKit, Hammer...);
- korišćenjem konzole (Command Prompt na Windows, Terminal na macOS operativnom sistemu);
- korišćenjem node.js-a.

Najbazičniji način za prevođenje Sass koda u CSS podrazumeva upotrebu konzolnog Sass programa. Takav program dostupan je za najpopularnije operativne sisteme današnjice (Windows, Linux, macOS) i može se preuzeti iz sledećeg GitHub repozitorijuma:

`https://github.com/sass/dart-sass/releases/`

Nakon preuzimanja paketa za odgovarajući operativni sistem, preuzetu arhivu je potrebno raspakovati. Za najudobnije korišćenje, raspakovani folder je potrebno smestiti unutar sistemske Path promenljive.

Dodavanje putanje u sistemsku Path promenljivu

Svi najpopularniji operativni sistemi poznaju pojam sistemske Path promenljive. Ipak, procedura za smeštanje putanja unutar Path promenljive razlikuje se u zavisnosti od operativnog sistema.

Na **Windows** operativnom sistemu, prozor za rukovanje sistemskim promenljivama naziva se *Environment Variables*. Do njega je moguće doći kucanjem *Advanced System Settings* u polju za pretragu. Otvara se *System Properties* prozor i *Advanced* tab, na čijem dnu se nalazi opcija *Environment Variables*. Unutar *Environment Variables* prozora potrebno je pronaći `Path` promenljivu i izvršiti dupli klik, čime će se otvoriti prozor za unos nove putanje unutar `Path` promenljive.

Na **macOS** operativnom sistemu procedura za unos putanje unutar Path promenljive podrazumeva kucanje sledeće komande unutar *Terminal* aplikacije: **sudo nano /etc/paths**. Nakon upućivanja ove komande potrebno je uneti korisničku lozinku. Nakon unosa ispravne lozinke, na dno fajla je potrebno dodati novu putanju i sačuvati izmene.

Na većini **Linux** distribucija, dodavanje putanje unutar Path promenljive može se obaviti editovanjem fajla `~/.bashrc.in`. U ovaj fajl je potrebno dodati novu liniju sledeće strukture: `export PATH=$PATH:/directory-name`. U ovom slučaju, putanja koja se dodaje je: `/directory-name`.

Provera postojanja Sass prevodioca

Nakon preuzimanja Sass paketa i smeštanja putanje na kojoj se on nalazi unutar Path promenljive, iz konzole ili terminala je moguće uputiti i prvu komandu Sass prevodiocu. Kako bismo se uverili da on uistinu i postoji na sistemu, dovoljno je u konzoli napisati sledeće:

```
sass –version
```

Ukoliko je Sass prevodilac dostupan na sistemu, nakon upućivanja prikazane komande, u novoj liniji konzole ispisaće se njegova verzija.

Sass sintaksa

Sass poseduje dve različite sintakse za pisanje koda. Originalna sintaksa veoma često se naziva *The indented syntax*, zato što se za organizaciju koda koriste uvlačenja, prelasci u novi red i prazna mesta, a ne tradicionalni CSS elementi za obavljanje takvog posla – vitičaste zagrade i karakter tačka sa zapetom.

Novija sintaksa naziva se *Sassy CSS*, odnosno **SCSS**. Iz samog naziva se može zaključiti da je dosta sličnija CSS-u, što podrazumeva upotrebu tradicionalnih CSS elemenata za organizaciju koda.

Osnovne razlike između dve upravo opisane Sass sintakse ilustrovane su slikom 24.2.

Slika 24.2. Dve različite Sass sintakse

Razlog zbog kojeg su tvorci Sassa naknadno kreirali novu, SCSS sintaksu, vrlo je jednostavan – **SCSS je superset jezika CSS**. Ovo praktično znači da je svaki CSS kod ujedno i kod koji zadovoljava sintaksna pravila SCSS-a. Takva odlučujuća prednost *Sassy CSS* sintakse omogućava veoma laku adaptaciju postojećih CSS fajlova i njihovo pretvaranje u SCSS kod. Takođe, upravo opisana osobina SCSS-a omogućava da se samo neki delovi stilizacije unutar jednog dokumenta formulišu korišćenjem Sassa, i to upravo oni za koje postoji opravdana potreba. Sav ostali kod moguće je pisati pravilima CSS jezika. Upravo zbog ovoga, u nastavku ove lekcije isključivo će biti razmatrana novija SCSS sintaksa Sass jezika.

Kreiranje Sass dokumenta

U zavisnosti od sintakse koja se koristi, Sass kod se piše unutar dokumenata sa ekstenzijom `.sass` ili `.scss`. Originalna sintaksa podrazumeva korišćenje `.sass`, a nova sintaksa `.scss` fajlova.

Fajlove sa ovakvim ekstenzijama moderni tekstualni editori automatski prepoznaju kao fajlove sa Sass kodom i u većini slučajeva im dodeljuju specijalne ikonice, kako bi se razlikovali od .html, .js ili .css fajlova.

Uzimajući u obzir sve što je rečeno, prvi praktičan korak u ovoj lekciji podrazumevaće kreiranje praznog fajla sa ekstenzijom `.scss`.

Prevođenje Sass koda u CSS

S obzirom na to da je svaki CSS kod ujedno i validan SCSS kod, za početak, unutar praznog Sass fajla može se napisati:

```
body {
  font-family: sans-serif;
}
```

Na ovaj način je definisana familija fonta na `body` elementu. Iako je reč o običnom CSS kodu, ovakav kod ujedno je i validan Sass kod. Postupak kojim se obavlja prevođenje Sass koda biće ilustrovan na upravo prikazanom primeru.

Prevođenje Sass koda u CSS obavlja se upućivanjem sledeće komande korišćenjem konzole (terminala):

```
sass style.scss style.css
```

Komanda koja se poziva jeste `sass`, nakon čega se navode putanje do SCSS i CSS fajlova. Potpuno razumljivo, unutar SCSS fajla nalazi se Sass kod koji je potrebno prevesti i smestiti unutar navedenog CSS fajla.

Napomena

Za uspešno izvršavanje prikazane komande za prevođenje, neophodno je prethodno se pozicionirati unutar foldera u kome se nalaze .scss i .css fajlovi.

Nakon izvršavanja prikazane komande, unutar CSS fajla naći će se sledeći sadržaj:

```
body {
  font-family: sans-serif;
}
```

Može se videti da Sass pretprocesor nije imao baš mnogo posla prilikom prevođenja Sass koda u CSS. S obzirom na to da prilikom pisanja Sass koda nismo iskoristili nikakvu specijalnu funkcionalnost tog jezika, Sass i CSS kod izgledaju identično.

Šta je .map fajl?

Prilikom prevođenja Sass koda u CSS, pretprocesor automatski kreira i jedan dodatni fajl sa ekstenzijom `.map`. Reč je o fajlu JSON formata, koji sadrži informacije o samom prevođenju dokumenta. Korišćenjem `.map` fajla, moguće je napraviti vezu između svake konstrukcije unutar Sass fajla i odgovarajućeg CSS koda koji nastaje kao proizvod prevođenja takvog Sass koda.

S obzirom na to da je pronalazak grešaka u Sass kodu veoma otežan zato što se unutar browsera ne vidi Sass već CSS, web pregledači `.map` fajl mogu da koriste kako bi na osnovu CSS-a koji je dobijen prevođenjem Sass-a došli do izvornog koda napisanog Sass jezikom.

Automatsko prevođenje Sass koda

U prethodnim redovima prikazan je osnovni pristup za prevođenje Sass koda u CSS. Ubrzo nakon početka praktičnog rada sa Sassom, postaje jasno da ručno prevođenje Sass koda u CSS može biti i više nego zamorno, pogotovu ako je nakon svake manje izmene potrebno videti efekat unutar web pregledača. Upravo zbog takvih situacija, Sass omogućava i opciju automatskog prevođenja. Automatsko prevođenje se zasniva na tome da Sass samostalno prati stanje fajla i prilikom detekcije promene na takvom fajlu automatski pokreće prevođenje.

Automatsko prevođenje se može aktivirati na sledeći način:

```
sass --watch style.scss style.css
```

Pitanje

Sassy CSS karakteriše se fajlovima sa ekstenzijom:

- **.scss**
- .css
- .sss
- .sass

Objašnjenje:

Sass poseduje dve različite sintakse, koje diktiraju i ekstenziju fajlova koji se koriste za pisanje koda. Originalna sintaksa poznaje .sass ekstenziju, dok se nova Sassy CSS sintaksa piše unutar fajlova sa ekstenzijom .scss.

Varijable

Prva specijalna funkcionalnost Sass jezika koja će biti ilustrovana u ovoj lekciji jesu Sass varijable.

U uvodnoj lekciji ovog modula ilustrovano je korišćenje proizvoljnih CSS svojstava, odnosno CSS varijabli. Tada je prikazano kako se one mogu koristiti za poboljšanje čitljivosti i preglednosti i smanjenje ponavljanja koda. Sass odlazi i korak dalje, pa tako omogućava da se varijable kreiraju i upotrebljavaju na još jednostavniji način:

```
$variablename: value;
```

Sass varijable karakterišu se karakterom `$` koji se navodi na početku njihovog naziva. Sve nakon ovog karaktera predstavlja proizvoljni naziv varijable. Tako jedna Sass varijabla može da izgleda ovako:

```
$primary-dark-color: #A2687E;
```

Upravo je definisana Sass varijabla sa nazivom `$primary-dark-color` i vrednošću `#A2687E`. Ovakva varijabla se može upotrebiti na sledeći način:

```
body {
  font-family: sans-serif;
  color: $primary-dark-color;
}
```

Sada je za definisanje boje teksta na body elementu upotrebljena Sass varijabla. Efekat prevođenja ovakvog Sass koda u CSS ilustrovan je slikom 24.3.

Slika 24.3. Primer upotrebe Sass varijabli

Iz upravo prikazanog primera se može zaključiti da su pojmovi CSS varijabli i Sass varijabli dva potpuno odvojena, zasebna pojma. Konkretno, Sass varijable se ne prevode u CSS varijable, već se njihove vrednosti direktno postavljaju kao vrednosti svojstava na kojima su upotrebljene (u primeru je to svojstvo color).

Oblast važenja Sass varijabli

Upravo prikazani primer podrazumevao je kreiranje globalne Sass varijable, zato što je ona bila definisana izvan bilo kakvog bloka. Takva varijabla se može koristiti unutar svih CSS opisa. Ipak, Sass varijablu je moguće definisati i unutar nekog CSS opisa i tada je ona vidljiva samo unutar takvog bloka. Primer ovakve situacije ilustrovan je slikom 24.4.

Slika 24.4. Primer oblasti važenja Sass varijabli

Sa slike 24.4. može se videti da u prevedenom CSS kodu h1 naslovi poseduju crvenu boju teksta, iako je unutar bloka body vrednost Sass varijable postavljena na plavu boju.

Definisanje globalnih varijabli iz bloka

Sass varijablu koja je definisana unutar nekog bloka moguće je proglasiti i globalnom, ukoliko se za tako nešto javi potreba. To se postiže upotrebom ključne reči !global:

Slika 24.5. Primer definisanja globalne varijable u Sassu

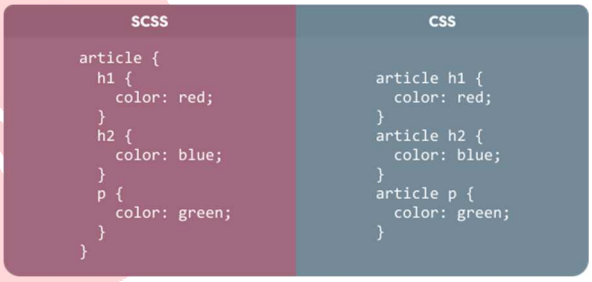
Sada je nakon vrednosti Sass promenljive unutar `body` bloka postavljena ključna reč `!global`. Zbog ove male izmene, unutar prevedenog CSS koda, `h1` naslovi imaju plavu boju teksta.

Napomena

Najbolja praksa podrazumeva definisanje varijabli globalno, izvan bilo kakvih blokova. Veoma česta praksa podrazumeva i smeštanje svih varijabli unutar jednog zasebnog fajla, koji se zatim uključuje u fajl sa glavnom stilizacijom. Takva praksa biće ilustrovana nešto kasnije, kada bude reči o modularizaciji stilizacije korišćenjem Sassa.

Gnežđenje

Osnovu HTML dokumenata čini struktura koju HTML elementi međusobno grade. Takva pojava, koja omogućava smeštanje jednog HTML elementa unutar nekog drugog, drugačije se naziva gnežđenje. Ipak, ovaj pojam ne postoji u CSS jeziku, odnosno nije moguće jedan CSS opis smestiti unutar drugog. Sass ispravlja ovakav nedostatak CSS jezika, te dozvoljava da se gradi razgranata struktura CSS opisa (slika 24.6).



Slika 24.6. Primer gnežđenja u Sassu

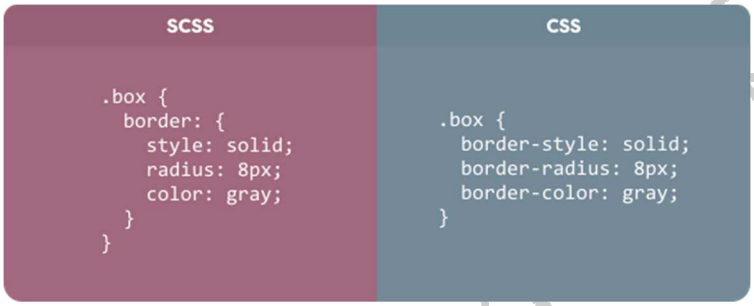
Iz upravo prikazanog primera može se videti na koji način funkcioniše gnežđenje u Sassu. Nekoliko blokova Sass koda (koji inače započinju otvorenom, a završavaju zatvorenom vitičastom zagradom) smešteno je unutar jednog bloka koda. Na ovaj način biće stilizovani svi `h1`, `h2` i `p` elementi koji se nalaze unutar `article` elemenata.

Na desnoj polovini slike 24.6, može se videti kako se realizuje Sass gnežđenje kada se kod prevede u CSS. Jasno je da se za realizaciju Sass gnežđenja koriste selektori zasnovani na relacijama.

Zajednički prefiksi

Sass omogućava da se na veoma lak način definišu vrednosti nekoliko svojstava koja poseduju zajedničke prefikse. U CSS jeziku postoji nekoliko grupa takvih svojstava, koja uglavnom poseduju srodne osobine. Na primer: `font-family`, `font-size`, `font-weight`, `font-style`... Zatim `text-decoration`, `text-align`, `text-transform`, `text-overflow`... Takođe, `border-style`, `border-radius`, `border-color`, `border-width`, `border-image`... Ovo su samo neki od primera svojstava koja poseduju zajedničke prefikse.

Svojstva sa zajedničkim prefiksima veoma često se navode u grupama. Sass olakšava njihovo pisanje (slika 24.7).



Slika 24.7. Primer Sass sintakse za definisanje svojstava sa zajedničkim prefiksom

Slika 24.7. ilustruje primer definisanja vrednosti tri svojstva koja započinju identičnim prefiksom. Umesto navođenja prefiksa na svakom od svojstava, kreiran je blok koda koji je identifikovan zajedničkim prefiksom. Zatim su pojedinačna svojstva navedena unutar bloka, bez prefiksa.

Mixini

Ponekad se prilikom stilizovanja web sajtova može javiti potreba za ponavljanjem jedne iste stilizacije na više mesta. Takva situacija se veoma često može javiti prilikom pisanja CSS svojstava koja se za različite pregledače formulišu korišćenjem specifičnih prefiksa. Jedno od takvih svojstava je, na primer, `transform`. Ovo svojstvo se može javiti u različitim oblicima:

- `-webkit-transform`
- `-moz-transform`
- `-ms-transform`
- `-o-transform`

Primer definisanja jedne transformacije navođenjem svih oblika za različite web pregledače može da izgleda ovako:

```
.box {
  -webkit-transform: translateX(20px);
  -moz-transform: translateX(20px);
  -ms-transform: translateX(20px);
  -o-transform: translateX(20px);
  transform: translateX(20px);
}
```

Ukoliko je prilikom stilizacije jednog web sajta potrebno obaviti ovakvu ili sličnu transformaciju na nekoliko različitih elemenata, jasno je da je uvek iznova neophodno ponavljati prikazane linije, koje predstavljaju oblike sa prefiksima specifičnim za različite pregledače. Olakšica za obavljanje takvog posla može doći u obliku Sass mixina (slika 24.8).

Slika 24.8. Primer Sass mixina

Slika 24.8. ilustruje primer jednog Sass mixina. Mixini započinju specifičnom Sass direktivnom – `@mixin`, nakon koje sledi naziv mixina. Naziv mixina u prikazanom primeru je `transform`.

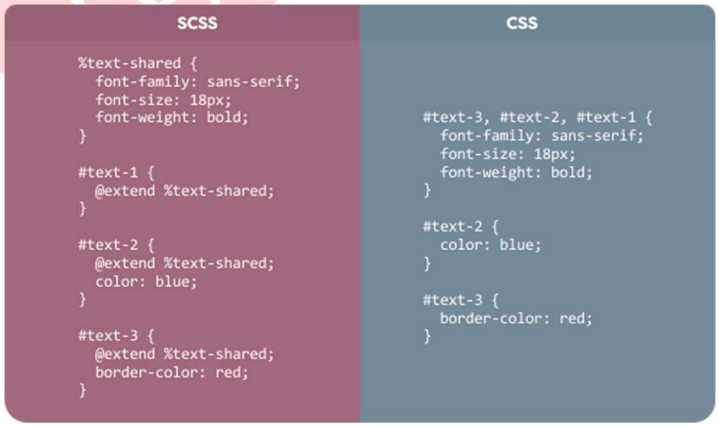
Mixin iz primera prihvata i jedan parametar koji se definiše korišćenjem Sass varijable. Naziv takvog parametra je `$property` i on nam omogućava da mixinu prosledimo bilo koju vrednost, koja će biti postavljena za vrednosti svojstava unutar mixina.

Jednom kreiran mixin moguće je iskoristiti korišćenjem CSS deklaracije `@include`. Nakon ove deklaracije navodi se naziv mixin-a, a u zagradama se prosleđuje vrednost Sass varijable koja će biti postavljena za vrednosti pojedinačnih svojstava.

Naravno, moguće je kreirati mixin i bez parametara, ali i sa većim brojim različitih parametara.

Proširivanje (nasleđivanje)

Kako bi se olakšao posao frontend programera kada se identična stilizacija ponavlja unutar većeg broja CSS opisa, Sass poznaje pojam proširivanja, odnosno nasleđivanja. Proširivanje, odnosno nasleđivanje, oslobađa nas potrebe za ponavljanjem jedne iste stilizacije na većem broju mesta, pa se na taj način olakšava održavanje i poboljšava preglednost. Primer Sass nasleđivanja ilustrovan je slikom 24.9.



Slika 24.9. Primer nasleđivanja u Sassu

Unutar upravo prikazanog primera, na početku SCSS koda naveden je jedan specifičan CSS opis, koji je izgrađen korišćenjem jednog posebnog SCSS selektora. Može se videti da takav selektor započinje karakterom `%`, a reč je zapravo o pojmu koji se u Sassu naziva **placeholder selektor**.

Opis koji se kreira placeholder selektorom ne prevodi se u CSS, što se može i videti na desnoj polovi slike 24.9. S obzirom na to da se ne prevodi u CSS, sa pravom se može postaviti pitanje koja je namena placeholder selektora? Osnovna namena placeholder selektora jeste definisanje stilizacije koja će biti nasleđena, odnosno proširena.

U prikazanom primeru, stilizacija koja je definisana opisom sa placeholder selektorom nasleđuje se, odnosno proširuje, unutar tri naredna CSS opisa. Nasleđivanje/proširivanje se u Sassu obavlja korišćenjem ključne reči **@extend**, nakon koje se navodi selektor sa deklaracijama koje se proširuju. Sve ovo praktično znači da će opisi `#text-1`, `#text-2` i `#text-3` posedovati tri deklaracije koje su navedene unutar CSS opisa kreiranog placeholder selektorom.

Prilikom nasleđivanja stilizacije, Sass omogućava da se nasleđena stilizacija i proširi, dodavanjem nekih specifičnih deklaracija. Tako nešto se može videti na primeru `#text-2` i `#text-3` opisa, koji poseduju po jednu dodatnu CSS deklaraciju.

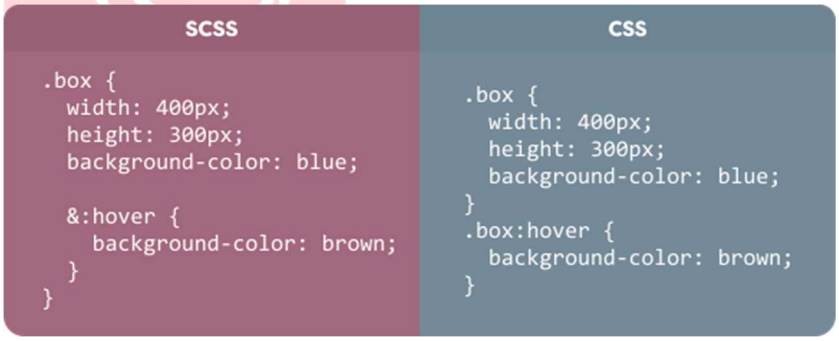
Napomena

Sass nasleđivanje/proširivanje može se postići i bez placeholder selektora. Tako je moguće naslediti/proširiti i CSS opis definisan bilo kojim drugim selektorom.

Roditeljski selektor

U prethodnim redovima prikazan je jedan specijalan Sass selektor – *placeholder selektor*. Sass poseduje još jedan specijalni selektor koji je moguće koristiti prilikom gneždenja. Reč je o selektoru koji upućuje na roditeljski element i obeležava sa karakterom ampersand (`&`).

Roditeljski selektor posebno je koristan prilikom definisanja pseudoklasa ili pseudoelemenata (slika 24.10).



Slika 24.10. Primer roditeljskog selektora u Sassu

U prikazanom primeru, korišćenjem karaktera `&`, selektovan je roditeljski element, odnosno element sa klasom `box`.

Operatori

U uvodnoj lekciji ovog modula ilustrovan je osnovni pristup CSS jezika koji omogućava definisanje izraza prilikom formulisanja vrednosti CSS svojstava. Naravno, reč je o `calc()` funkciji, koja dozvoljava korišćenje osnovnih računskih operacija. Sass odlazi korak dalje, pa tako omogućava upotrebu većeg broja operatora i to na mnogo jednostavniji način.

Prilikom pisanja Sass koda, nije potrebno koristiti CSS funkciju `calc()` kako bi se definisao neki izraz. Izraze je moguće pisati direktno kao vrednosti svojstava (slika 24.11).

Slika 24.11. Primer upotrebe operatora u Sassu

Primer sa slike 24.11. ilustruje korišćenje operatora i to direktno prilikom navođenja vrednosti svojstva `width`. Sass će za nas izračunati vrednost ovakvog izraza, pa će nakon prevođenja svojstvo `width` imati proračunatu vrednost, što se može videti na desnoj polovini slike 24.11.

Modularizacija stilizacije

Sass poseduje odličan mehanizam za modularizaciju koda stilizacije, što podrazumeva mogućnost pisanja koda u većem broju zasebnih fajlova koji se zatim spajaju u jedan.

Osnovni CSS pristup za importovanje

Sličan sistem za modularizaciju dostupan je i unutar CSS jezika i on podrazumeva korišćenje direktive `@import`:

```
@import 'common.css';
```

Upravo prikazana linija CSS koda može se smestiti unutar nekog CSS fajla, čime se u takav fajl uključuju svi CSS opisi definisani unutar fajla `common.css`.

Direktivu `@import` neophodno je postaviti na početak nekog CSS fajla pre bilo kakvih drugih CSS opisa.

Najveća mana `@import` direktive ogleda se u načinu na koji se ona obrađuje od strane browsera. Naime, izvršavanjem svake `@import` direktive, web browser upućuje novi HTTP zahtev, što značajno utiče na performanse web sajta.

Sass unapređuje osobine izvorne `@import` direktive i ispravlja njen najveći nedostatak koji se odnosi na upućivanje pojedinačnih HTTP zahteva. Tako Sass već prilikom prevođenja sve importovane fajlove sa stilizacijom spaja u jedan.

Direktiva `@import` u Sassu se koristi na sledeći način:

```
@import "common";
```

Bitno je primetiti da sada nije potrebno definisati ekstenziju, već samo naziv SCSS fajla.

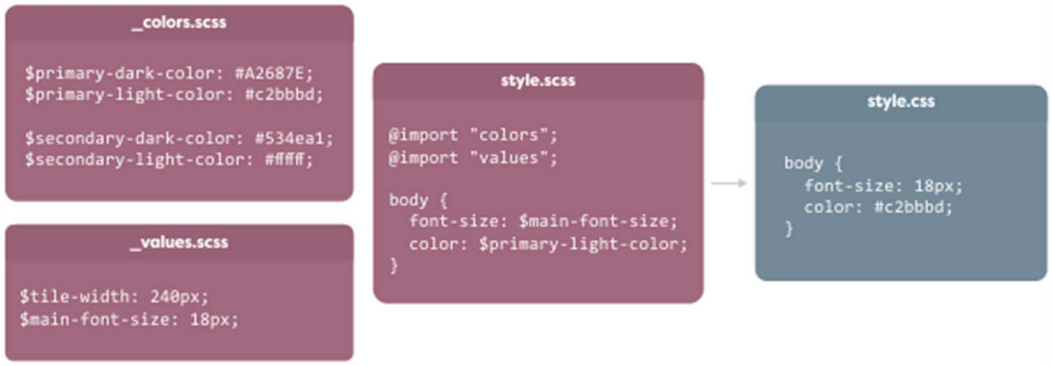
Sass parcijalni fajlovi

Prilikom sprovođenja modularizacije korišćenjem Sassa, veoma često se koriste i specijalni fajlovi sa stilizacijom koji se nazivaju parcijalni. Reč je o SCSS fajlovima koji se ne prevode u CSS, već su isključivo namenjeni uključivanju u neke druge fajlove.

Parcijalni Sass fajlovi se karakterišu donjom crtom kojom započinje njihov naziv. Na primer:

```
_colors.scss
_values.scss
```

Primer modularizacije korišćenjem Sassa ilustrovan je slikom 24.12.



Slika 24.12. Primer modularizacije korišćenjem Sassa

Na slici 24.12. mogu se videti dva parcijalna SCSS fajla (`_colors.scss` i `_values.scss`) u kojima su definisane različite Sass varijable. Ovakva dva parcijalna fajla uključena su u glavni SCSS fajl sa stilizacijom korišćenjem `@import` direktive. U glavnom fajlu sa stilizacijom, iskorišćene su vrednosti Sass varijabli koje su definisane u parcijalnim fajlovima. Na kraju, na slici 24.12, može se videti kako izgleda CSS kod nakon prevođenja. Iako je za postizanje stilizacije korišćeno nekoliko Sass fajlova, na kraju je dobijen samo jedan CSS fajl.

Direktiva @use

Pored `@import` direktive, za postizanje modularizacije u Sassu je moguće koristiti i direktivu `@use`. Reč je o novom pristupu, koji u potpunosti zaobilazi korišćenje izvorne `@import` direktive. Stoga se za nove projekte preporučuje upotreba `@use` direktive, čije korišćenje je istovetno korišćenju upravo prikazane `@import` direktive.

Rezime

- Pretprocesori su specijalni alati (kompjuterski programi) koji obavljaju prevođenje izvornog koda jednog jezika u neki drugi.
- Pretprocesorski jezici za stilizovanje omogućavaju korišćenje različitih naprednih funkcionalnosti koje izvorno ne postoje u CSS jeziku, što na kraju omogućava mnogo jednostavnije i produktivnije pisanje koda za stilizovanje web sajtova.
- Jedan od najpoznatijih pretprocesorskih jezika za stilizovanje jeste Sass – ***S****yntactically* ***A****wesome* ***S****tyle****s****heet*.
- Baš kao i svaki drugi pretprocesorski jezik za stilizovanje, Sass je neophodno prevesti u CSS kod kako bi bio razumljiv web pregledaču.
- Najjednostavniji način za prevođenje Sass koda u CSS podrazumeva upotrebu konzolnog Sass programa, koji je dostupan za najpopularnije operativne sisteme.
- Provera postojanja Sassa može se obaviti korišćenjem komande `sass –version`.
- Sass poseduje dve različite sintakse za pisanje koda: *The indented syntax* i *Sassy CSS*.
- *The indented syntax* je originalna sintaksa i kod nje se za organizaciju koda koriste uvlačenja, prelasci u novi red i prazna mesta.
- *Sassy CSS*, odnosno SCSS je novija sintaksa, koja podrazumeva upotrebu tradicionalnih CSS elemenata za organizaciju koda.
- SCSS je superset jezika CSS.
- Originalna sintaksa podrazumeva korišćenje `.sass`, a nova sintaksa `.scss` fajlova.
- Sass varijable karakterišu se karakterom `$` koji se navodi na početku njihovog naziva.
- Sass varijablu je moguće definisati globalno, izvan svih blokova, ali i unutar nekog CSS opisa; globalna varijabla je vidljiva svuda, a varijabla unutar CSS opisa samo unutar takvog bloka.
- Sass varijablu koja je definisana unutar nekog bloka moguće je proglasiti globalnom upotrebom ključne reči `!global`.
- Sass omogućava gnežđenje stilizacije, odnosno smeštanje jednog CSS opisa u neki drugi.
- Proizvoljan broj CSS deklaracija koje se često koriste kao grupa moguće je objediniti unutar mixina.
- Mixini započinju specifičnom Sass direktivnom – `@mixin`, nakon koje sledi naziv mixina.
- Sass poznaje specijalan selektor koji se naziva placeholder selektor i karakteriše se karakterom `%`; CSS opis sa takvim selektorom ne prevodi se direktno u CSS.
- Nasleđivanje/proširivanje se u Sassu obavlja korišćenjem ključne reči `@extend`, nakon koje se navodi selektor sa deklaracijama koje se proširuju.
- Sass poseduje i selektor koji upućuje na roditeljski element i obeležava se karakterom ampersend (`&`).
- Prilikom pisanja Sass koda, nije potrebno koristiti CSS funkciju `calc()` kako bi se definisao neki izraz, već je operatore moguće koristiti prilikom definisanja vrednosti CSS svojstava ili Sass varijabli.
- Parcijalni Sass fajlovi se ne prevode u CSS, već su isključivo namenjeni uključivanju u neke druge fajlove; njihov naziv započinje karakterom donja crta (_).
- dodavanje jednog Sass fajla nekom drugom može se obaviti korišćenjem direktiva `@import` i `@use`.