JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 225–229
Received: 21 June 2021
Accepted: 4 July 201

# Comparison of the compilation speed of the SCSS and LESS preprocessors

# Porównanie szybkości kompilowania kodów preprocesorów SCSS oraz LESS

Andrii Berkovskyy*, Kostiantyn Voskoboinik, Marcin Badurowicz

*Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland*

**Abstract**
This article compares the compilation speed of the SCSS and LESS preprocessor codes. Each preprocessor has its own syntax, which is transpiled into the CSS stylesheet language in the further development of the web page. These technologies serve the same purpose - to simplify and speed up the writing of page views, but are based on different programming languages - Ruby (SCSS) and JavaScript (LESS). As part of the research, the compilation times of the codes in the SCSS and LESS format with sizes of 10, 50, 100 and 200 kB were measured, the obtained results clearly showed that LESS turned out to be a faster tool for processing the code into CSS syntax. The size of the CSS result code was also taken into account in the analyzes.

*Keywords*: CSS preprocessors; compilation speed comparison; SCSS; LESS

**Streszczenie**
Ten artykuł jest poświęcony porównaniu szybkości kompilowania kodów preprocesorów SCSS oraz LESS. Każdy preprocesor ma swoją składnię, która w ciągu dalszym tworzenia strony webowej jest transpilowana do składni języka arkuszy stylów CSS. Technologie te służą temu samemu celowi – uproszczeniu i przyśpieszeniu pisania widoków stron, ale bazują się na różnych językach programowania – języku Ruby (SCSS) oraz JavaScript (LESS). W ramach przeprowadzonych badań dokonano pomiarów czasów kompilacji kodów w formacie SCSS i LESS o rozmiarach 10, 50, 100 i 200 kB, otrzymane wyniki jednoznacznie pokazały, że szybszym narzędziem do przetwarzania kodu na składnię CSS okazał się LESS. W analizach pod uwagę również wzięto wielkość kodu wynikowego CSS.

*Słowa kluczowe*: preprocesory CSS; porównanie szybkości kompilowania; SCSS; LESS

*Corresponding author
*Email address*:



## 1. Wstęp

Coraz więcej ludzi korzysta z usług webowych, a zaprogramowanie stron internetowych jest jednym z popularniejszych rodzajów działalności programistów. Tworzenie efektywnych widoków stron może odbywać się na kilka sposobów, np., wykorzystywanie konstruktorów stron webowych z gotowymi szkicami, które twórca tylko wypełnia treścią, lub pisanie unikalnych kaskadowych arkuszy stylów CSS (Cascading Style Sheets) co zapewni stronie unikalność w porównaniu do innych.

Żeby ułatwić pracę programistom i wnieść do programowania widoków stron cechy właściwe językom obiektowym zostały wymyślone preprocesory CSS. Transpilator kodu korzystając z preprocesora przetwarza kod napisany w składni preprocesora na kod CSS. Korzystanie z tych technologii zaoszczędza czas, ponieważ główną zaletą preprocesorów jest wykorzystywanie zmiennych z możliwością przypisania do nich cech potrzebnych do realizacji widoku. Później napisany kod zostanie transpilowany do zwykłego CSS, który jest zrozumiały dla przeglądarek internetowych.

Aby skrócić czas programowania w tych technologiach jeszcze bardziej, można posłużyć się preprocesorem, który szybciej przetwarza kod na składnie CSS. Autorzy niniejszej pracy pragną znaleźć odpowiedź pytanie dotyczące wydajności tego typu narzędzi.

Do realizacji badań dotyczących sprawdzenia szybkości przetwarzania kodów źródłowych został stworzony internetowy transpilator rozróżniający gramatykę obu preprocesorów SCSS (Sassy CSS) oraz LESS (Leaner CSS), pozwalający mierzyć czas transpilacji oraz pobierać pliki końcowe w formacie CSS. Jako wyniki pracy preprocesorów pozwalające wskazać bardziej efektywne narzędzie, będą brane pod uwagę czasy transpilacji. Preprocesor, który szybciej przetworzy kod źródłowy na kod CSS, będzie określany szybszym.

## 2. Przegląd prac o podobnej tematyce

Istniejące prace na czas obecny przedstawiają sobą ogólny opis badanych preprocesorów, ich zalety i funkcjonalności. Żadna z przejrzanych przez autorów prac nie przedstawia w sposób eksperymentalny wyników do wskazania preprocesora, który szybciej przetwarza kod swojej składni na składnie CSS. Do takich prac należą książki oraz prace naukowe [1-3].

Wymienione wyżej prace nie przedstawiają eksperymentów nad czasem kompilacji, które będą wykonane przez autorów w ciągu dalszym danej pracy.

Istnieją również artykuły i książki naukowe prezentujące dane eksperymentalne biorące pod uwagę funkcjonalności badanych preprocesorów, a nie czasu ich pracy. Można tu wymienić pracę D. Mazinanian & N. Tsantalis pod tytułem "An empirical study on the use of CSS preprocessors." [4], w której można znaleźć tabele i wykresy danych dotyczących zbiorów funkcjonalności preprocesorów SCSS i LESS. Tu mogą być znalezione dane statystyczne wykorzystywania mixinów (fragmenty kodu, które mogą być wywołane wielokrotnie) w tych preprocesorach lub np., o ile w tym czy innym preprocesorze częściej jest stosowane zagnieżdżanie. Ale jak i w przypadku wyżej wymienionych prac, nie ma tu eksperymentów i otrzymanych danych opisujących czasy kompilacji.

Prace, w których były przedstawione czasy kompilacji - „A survey on CSS preprocessors." [5] oraz „CSS preprocessing: Tools and automation techniques" [6] autorstwa Queirós, R., gdzie autor wykorzystuje tabele pokazującą czas kompilacji preprocesorów w zależności od rozmiaru pliku wejściowego bez żadnych opisów zawartości treści kodów oraz bez przedstawienia dodatkowych informacji o plikach wejściowych. Pozostała część treści wyżej wymienionych artykułów danego autora również jest poświęcona opisowi możliwości, funkcjonalności i wykorzystaniu danych preprocesorów.

## 3. Technologie

Niniejszy artykuł opisuje badania na preprocesorach CSS – SCSS oraz LESS. Są to dwie popularniejsze technologie na rynku w czasie obecnym do wykorzystania w swojej dziedzinie. Każdy z nich posiada własną składnię do napisania kodów źródłowych, które w przyszłości są transpilowane na składnie CSS. Kod CSS jest produktem końcowym działania obu preprocesorów.

### 3.1. Język arkuszy stylów CSS

CSS jest językiem arkuszy stylów opisującym widok strony internetowej. Język ten jest stosowany do definicji kolorów, czcionki, stylów, pozycjonowania oddzielnych bloków oraz innych elementów prezentacji widoku stron webowych. Głównym celem rozwoju CSS było oddzielenie opisu struktury strony webowej (za pomocą HTML lub innych języków) od opisu widoku tej strony, który teraz jest realizowany za pomocą arkuszy stylów CSS [7]. Takie rozdzielenie zwiększa dostępność dokumentu oraz przedstawia dużą elastyczność oraz możliwość sterowania widokiem.

Oprócz tego, CSS pozwala wyświetlać ten sam dokument w różnych stylach lub metodach wyświetlania, takich jak prezentacja ekranowa, prezentacja do druku, czytanie głosowe lub prezentacja na urządzeniach wykorzystujących Alfabet Braille'a.

### 3.2. Preprocesor SCSS

SCSS – dialekt języka SASS ze składnią zbliżoną do składni języka CSS przeznaczony do uproszczenia tworzenia kodu CSS. Składnia języka jest bardzo elastyczna, biorąca pod uwagę wiele drobiazgów, które są tak pożądane w CSS. Język ten posiada logikę (*@if*, *@each*), matematykę (można dodawać liczby, wiersze i kolory). SCSS jest łatwym do zrozumienia programistom pracującym ze zwykłym CSS i jest lepszym wyborem w przejściu na programowanie widoków stron z wykorzystaniem preprocesorów [8].

### 3.3. Preprocesor LESS

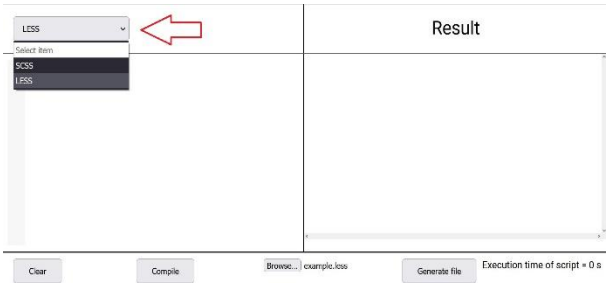
LESS również jest preprocesorem arkuszy stylów CSS. On jest nowszym preprocesorem w odniesieniu do SCSS i jest często z nim porównywany. Jak i SCSS ma on korzenie w języku Ruby, ale w czasie obecnym bazuje się na JavaScript. Wyróżniającą cechą tego preprocesora jest kompilacja kodu na kod CSS przez przeglądarkę w czasie rzeczywistym.

Składnia LESS pozwala na korzystanie z następujących elementów preprocesora [9]:

- zmienne – jednostki z przypisanymi wartościami;
- mixins – wielokrotnie wywoływany fragment kodu;
- zagnieżdżanie – hierarchia wierszy;
- operacje matematyczne.

## 4. Środowisko badawcze

W celu przeprowadzenia serii testów został stworzony transpilator kodów badanych preprocesorów w postaci aplikacji webowej, dostępnej z przeglądarki internetowej.




Rysunek 1: Interfejs aplikacji testowej.

Na rysunku 1 przedstawiony jest interfejs aplikacji testowej. Można tu zobaczyć dwa główne pola tekstowe. Pole w lewej części służy do wstawiania kodu poprzez opcję wklejania lub za pomocą przycisku „Browse…". Do wyboru preprocesora służy pole w lewej górnej części z etykietą „Select", w którym są dwie opcje – „SCSS" oraz „LESS". Pole tekstowe w prawej części interfejsu wyświetla wynik kompilacji w postaci kodu formatu CSS. Dolną część aplikacji zajmują przyciski „Clear" – do czyszczenia kodu preprocesora, przycisk „Compile" – do rozpoczęcia kompilacji, „Browse" – do wyboru pliku z kodem źródłowym preprocesora oraz „Generate file" – do wygenerowania pliku formatu CSS z kodem wyniku. W tym miejscu również znajduje się pole z wyświetlającym się czasem bieżącej kompilacji.

Do kompilacji kodów preprocesorów napisane oddzielne metody z wykorzystaniem gotowych bibliotek ze źródeł otwartych do transpilacji kodów preprocesorów SCSS [10] oraz LESS [11]. Podczas wyboru preprocesora jednocześnie jest wybierana metoda z odpowiednią biblioteką do transpilacji kodu wybranego preprocesora.

Do funkcji transpilatora należą:

- wybór preprocesora do transpilacji;
- wczytanie fragmentów kodu poprzez wstawianie kodu w odpowiednie pole lub poprzez plik wejściowy z danym kodem;
- wyświetlenie wyników transpilacji kodu preprocesora w formacie CSS;
- pomiar czasu wykonywanej transpilacji;
- generowanie i pobieranie pliku wyjściowego z wynikiem transpilacji.

Funkcjonalność środowiska testowego nie jest obciążona zaawansowanym widokiem czy niewykorzystywanymi funkcjami, które mogą skutkować na niepoprawne wyniki testów. Transpilator tylko i wyłącznie otrzymuje kod od użytkownika i przetwarza ten kod wybranym preprocesorem na składnie CSS.

## 5. Scenariusze badań

Badania zostały przeprowadzone na zbiorach kodów o różnych rozmiarach plików wejściowych dla każdego z preprocesorów. Takie podejście pozwoli zachować lepsze podobieństwo danych wejściowych, w przeciwieństwu, np., liczby linijek kodu. Spowodowano to różnicą w składni preprocesorów oraz możliwością wypełnienia linijek mniej obciążoną składnią, co w konsekwencji może doprowadzić do mniej prawdopodobnych badań.

Zbiory kodów poddanych eksperymentom są przedstawione plikami formatów SCSS oraz LESS o rozmiarach 10KB, 50KB, 100KB oraz 2000KB dla każdego preprocesora. Pliki o rozmiarze 200KB będą górną granicą pozwalającą zobaczyć różnicę pracy preprocesorów z dużą objętością danych wchodzących.

W środowisku badawczym do każdego z preprocesorów po kolei zostaną wczytane pliki o przedstawionych rozmiarach. Dla uwiarygodnienia wyników, badania zostały powtórzone 5 razy, a następnie uśrednione. Według autorów pracy, ta liczba kompilacji każdego pliku jest wystarczającą, ponieważ czasy kompilacji są o dużym stopniu spójne i w pełni ilustrują zachowywanie preprocesorów z odpowiednim fragmentem kodu. Po każdym kolejnym eksperymencie jest zapisywany czas kompilacji z pola „Execution time of script" oraz wygenerowany plik wyjściowy formatu CSS do sprawdzenia jego rozmiaru.

W wyniku opisanych badań otrzymano zbiory danych do obróbki w postaci 5-ciu czasów kompilacji dla każdego rozmiaru pliku każdego preprocesora oraz pliku wyjściowego formatu CSS.

## 6. Wyniki badań

Niżej zostaną przedstawione wyniki badań w postaci zbioru tabeli i wykresów.

Przedstawiony w tabeli 1 zbiór danych pozwoli wyliczyć średni czas kompilacji każdego z preprocesorów dla każdego rozmiaru pliku.

Tabela 1: Zbiór czasów kompilacji dla każdego rozmiaru pliku wejściowego

| | | Czas kompilacji (s) | |
|---|---|---|---|
| | | SCSS | LESS |
| Rozmiary plików wejściowych (KB) | 10 | 0,12617 | 0,00990 |
| | | 0,07447 | 0,00914 |
| | | 0,07518 | 0,00887 |
| | | 0,07430 | 0,00914 |
| | | 0,07808 | 0,01156 |
| | 50 | 1,80656 | 0,04972 |
| | | 1,64693 | 0,04713 |
| | | 1,78883 | 0,04710 |
| | | 1,62705 | 0,04607 |
| | | 1,67479 | 0,04797 |
| | 100 | 6,68259 | 0,10307 |
| | | 5,88420 | 0,09813 |
| | | 6,26396 | 0,10444 |
| | | 6,54755 | 0,09945 |
| | | 6,46393 | 0,09610 |
| | 200 | 29,2196 | 0,20374 |
| | | 25,8104 | 0,22330 |
| | | 25,3238 | 0,21979 |
| | | 24,6682 | 0,21164 |
| | | 24,4384 | 0,22547 |

Tabela 2: Średnie czasy kompilacji dla każdego rozmiaru pliku wejściowego

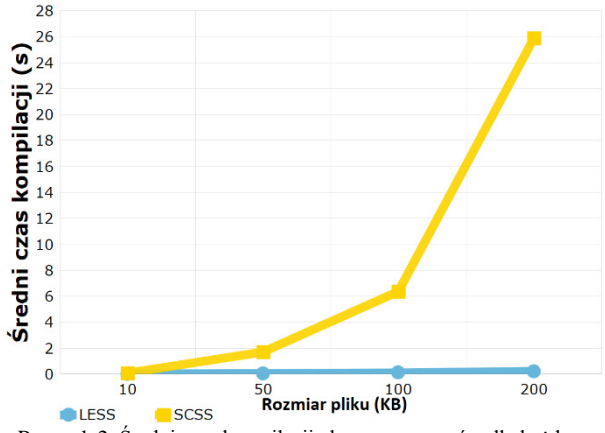
| | | Średni czas kompilacji (s) | |
|---|---|---|---|
| | | SCSS | LESS |
| Rozmiary plików wejściowych (KB) | 10 | 0,08564 | 0,00972 |
| | 50 | 1,70883 | 0,0476 |
| | 100 | 6,36845 | 0,10024 |
| | 200 | 25,89208 | 0,21679 |

Zbiór danych przedstawiony w tabeli 2 pozwala zobaczyć o ile średnie czasy kompilacji obu preprocesorów różnią się. Jednocześnie, dane przedstawione w taki sposób są łatwiejsze do obejrzenia różnicy w pracy preprocesorów niż zbiór danych składający się ze wszystkich czasów kompilacji wszystkich eksperymentów.

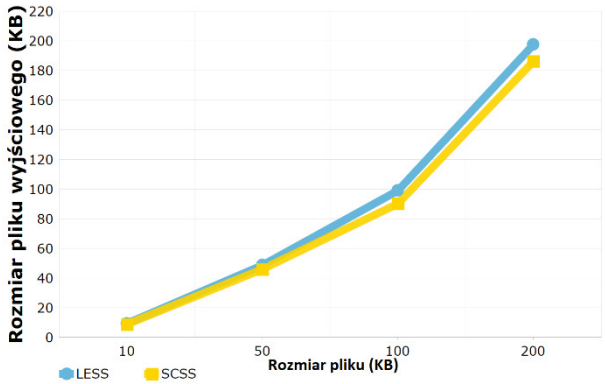
Tabela 3: Rozmiary plików wynikowych formatu CSS

| | | Rozmiary plików wyjściowych formatu CSS (KB) | |
|---|---|---|---|
| | | SCSS | LESS |
| Rozmiary plików wejściowych (KB) | 10 | 9 | 10 |
| | 50 | 45 | 48 |
| | 100 | 89 | 97 |
| | 200 | 182 | 193 |

Tabela 3 zawiera dane dotyczące rozmiarów plików wyjściowych formatu CSS, co pozwala przedstawić efektywność mechanizmów kompresji plików zawartych w preprocesorach. Dane wskazują na to, że preprocesor SCSS lepiej kompresuje rozmiar pliku wyjściowego.




Rysunek 2: Średni czas kompilacji obu preprocesorów dla każdego rozmiaru pliku wejściowego.




Rysunek 3: Rozmiary plików wyjściowych dla każdego rozmiaru pliku wejściowego

Przedstawione wyżej rysunki ilustrują wykresy, na których pokazano porównanie średnich czasów kompilacji oraz porównanie rozmiarów plików wyjściowych.

## 7. Wnioski

Porównując preprocesory SCSS oraz LESS w szybkości kompilacji kodów, można stwierdzić, że preprocesor LESS jest szybszy we wszystkich przedstawionych przypadkach badań dla wszystkich rozmiarów plików źródłowych. Różnica w czasie kompilacji jest widoczna już na etapie kompilacji plików o najmniejszym rozmiarze (10KB). Chociaż w takim przypadku ona jest najmniejsza i różni się o setną sekundy.

Wraz ze wzrostem rozmiaru (objętości) kodu źródłowego, a odpowiednio i rozmiaru pliku wejściowego, różnica w czasie kompilacji też znacznie rośnie, ponieważ preprocesor LESS kontynuuje przetwarzać kod nawet dla pliku o rozmiarze 200KB mniej niż za sekundę, w przeciwieństwie do preprocesora SCSS, któremu na taką pracę już potrzebne są od 24 do 29 sekund.

Jedynym aspektem, w którym preprocesor SCSS prezentował nieco lepsze wyniki, to jest rozmiar pliku wyjściowego formatu CSS. Ze względu na lepszą optymizację swojej składni preprocesora SCSS na składnie CSS, pliki wyjściowe mają mniejszy rozmiar niż przy wykorzystaniu preprocesora LESS, co ilustrują tabela 3 oraz rysunek 3. Również, różnice w poszczególnych przypadkach rozmiarów wynoszą około 10KB, co nie można nazwać wadą preprocesora LESS.

## 8. Podsumowanie

Ze względu na elastyczność i łatwość pracy z preprocesorami arkuszy stylów, programiści coraz częściej korzystają z tych technologii. Preprocesory wnoszą funkcjonalność zwykłych języków programowania, co ułatwia pracę i skraca czas pisania kodu w składni tych technologii.

Celem niniejszej pracy było określenie preprocesora, który szybciej skompiluje kod ze swojej składni na składnie CSS, co w przyszłości pozwoli użytkownikom tych technologii nie wybierać między dwoma, a pracować od razu na lepszej.

Z wyników eksperymentów, które były splanowane i przeprowadzone twórcami danej pracy, można powiedzieć, że preprocesor LESS jest technologią, która zapełni szybszą pracę. Preprocesor ten pokazuje czas kompilacji o wiele mniejszy niż preprocesor SCSS. Nawet dla plików w zakresie od 100 do 200 kilobajtów, czas potrzebny preprocesorowi LESS będzie mniejszy jednej sekundy.

Z drugiej strony, jeżeli twórca widoków stron zaczyna pracować z preprocesorami od „zera” i te technologie nie są mu znane dotychczas, nauczenie się składni też nie będzie decydującym czynnikiem, ponieważ trzeba będzie nauczyć się zarówno jednej tak i drugiej technologii jako całkiem nieznanej, a nie można stwierdzić, że jedna jest łatwiejsza do nauczania lub trudniejsza od drugiej.

Zatem, biorąc pod uwagę, że preprocesor LESS o wiele szybciej przetwarza kod i nie ma znaczących wad w porównaniu do preprocesora SCSS, można jego nazwać lepszą technologią do korzystania w pisaniu arkuszy widoków. Taki wniosek opiera się na eksperymentach i ich wynikach przeprowadzonych w punkcie 6 niniejszej pracy.

## Literatura

[1] A. Prabhu, Introduction to Preprocessors. In Beginning CSS Preprocessors, Apress, Berkeley, CA (2015) 1-12.

[2] M. Dowden, M. Dowden, Preprocessors. In Architecting CSS, Apress, Berkeley, CA (2020) 165-180.

[3] T. Laukkanen, CSS preprocessor-Sass eller Less. Toni (2017).

[4] D. Mazinanian, N. Tsantalis, An empirical study on the use of CSS preprocessors. In 2016 IEEE 23rd international conference on Software Analysis, Evolution, and Reengineering (SANER), IEEE, Vol. 1 (2016) 168-178.

[5] R. Queirós, A survey on CSS preprocessors. SLATE (2017).

[6] R. Queirós, CSS preprocessing: Tools and automation techniques, Information (2018) 9(1) 17.

[7] M. W. Alawar, S. S. Abu-Naser, CSS-Tutor: An intelligent tutoring system for CSS and HTML (2017).

[8] Możliwości preprocesora SCSS, https://sass-lang.com/guide , [03.05.2021].

[9] Funkcjonalność preprocesora LESS, https://lesscss.org/, [03.05.2021]

[10] Biblioteka do transpilacji kodu preprocesora SCSS: https://scssphp.github.io/scssphp/, [26.06.2021]

[11] Biblioteka do transpilacji kodu preprocesora LESS: https://leafo.net/lessphp/, [26.06.2021]