УДК 004

Использование препроцессора SASS и синтаксиса SCSS для компиляции CSS файлов

Халиманенков Андрей Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается создание CSS файлов с помощью препроцессора SASS, который позволяет применить логику синтаксиса классических языков программирования к созданию каскадных таблиц стилей. Для этой цели используется сборщик задач Gulp с установленным модулем SASS. Итогом исследования является часть веб-приложения, отвечающая за клиентскую сторону пользовательского интерфейса, разработанная с использованием синтаксиса SCSS.
Ключевые слова: создание веб-сайтов, SCSS, SASS, CSS, интерфейс, фронтэнд.

Using the SASS preprocessor and SCSS syntax to compile CSS files

Khalimanenkov Andrey Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the creation of CSS files using the SASS preprocessor, which allows you to apply the syntax logic of classical programming languages to the creation of cascading style sheets. For this purpose, the Gulp task collection is used with the SAS module installed. The result of the study is the part of the web application responsible for the client side of the user interface, delevoped using the SCSS syntax.
Keywords: websites creating, SCSS, SASS, CSS, interface, frontend.

В начале становления веб-разработки, программисты использовали файлы CSS и его родной синтаксис для создания стилей элементов DOM. Но логика этого синтаксиса не похожа на логику классических языков программирования, что вызывало не только сложности с поиском конкретного стиля среди всего файла, но и способствовало сильному увеличению, как размера файла, так и количества строк, что в свою очередь крайне затрудняло работу при обслуживании больших сайтов. В синтаксисе CSS отсутствуют:

1. Вложенность элементов;
2. Переменные;

3. Амперсанд;
4. Арифметические операции;
5. Операторы сравнения;
6. Логические операции;
7. Функции.

Цель исследования – описать возможности SASS на примере создания CSS файла.

Вопрос разработки интерфейсов при создании сайтов волнует некоторых исследователей и специалистов: Е. В. Пантелеева [1] отразила сущность разработки сайта с использованием языка разметки HTML, особенности таблицы стилей CSS и языка программирования JavaScript. Н. Д. Лушников и А. Д. Альтерман рассмотрели основы (технические возможности) каскадных таблиц стилей CSS. Кроме того, освещены главные преимущества и принцип работы каскадных таблиц. Л. А. Надеинский [3] разработал архетип модуля каскадных таблиц стилей с LESSCSS компилятором для быстрого создания модулей для разработки каскадных таблиц стилей с возможностью подключения к WAR модулям Java. Н. О. Айдарбаев [4] раскрыл понятие адаптивного дизайна как одного из процессов веб разработки. Дал определения разновидностей фронтэнд фреймворков, используемых в веб разработке, и подробный анализ компонентов фреймворка Bootstrap. Т. В. Макарова [5] в своём учебном пособии рассмотрела теоретические вопросы: гипертекстовый способ представления информации в компьютерной сети; основные принципы визуального дизайна веб-страниц; разработка концепции, структуры, макета, сайта; анализ юзабилити.

Препроцессор Sass позволяет создавать css-свойства более наглядным и эффективным способом, используя множество подходов, реализаций которых нет в стандартном CSS.

Для начала работы нужно создать среду для работы с Sass. Для этих целей подойдёт сборщик задач Gulp [6]. SASS устанавливается в папку с проектом с помощью команды «npm install sass gulp-sass --save-dev» в терминале.

Первым преимуществом препроцессора являются переменные, которые позволяют записывать в себя стили для множественного использования их в разных частях документа. Например, цвет для выделения важного текста изменился в связи с новым дизайном веб-приложения. Вместо того, чтобы искать и заменять этот цвет по всему документу, нужно лишь заменить значение переменной. Пример показан на рисунке 1. Слева SCSS код, а справа скомпилированный препроцессором CSS файл. На всех рисунках в этой статье будет использовано такое расположение.

Рисунок 1 – демонстрация работы переменной

Второе преимущество – это вложенность элементов. Для того, чтобы применить стили ко всем заголовкам <h2> в блоке <main> в обычном Css файле применяется следующая запись **main h2{}**, в синтаксисе Scss этот селектор будет выглядеть так - **main{ h2{} }**. На одном примере это не похоже на серьёзное преимущество, но вложенность играет огромную роль в понимании структуры документа при использовании в большом проекте**.** Также такой подход позволяет копировать целый блок стилей и переносить на другой проект, т.к. внутри уже вложены стили остальных дочерних элементов. Такая запись напоминает классические языки программирования и является более простой для понимания и визуального восприятия, т.к. явно видна структура. Пример на рисунке 2.

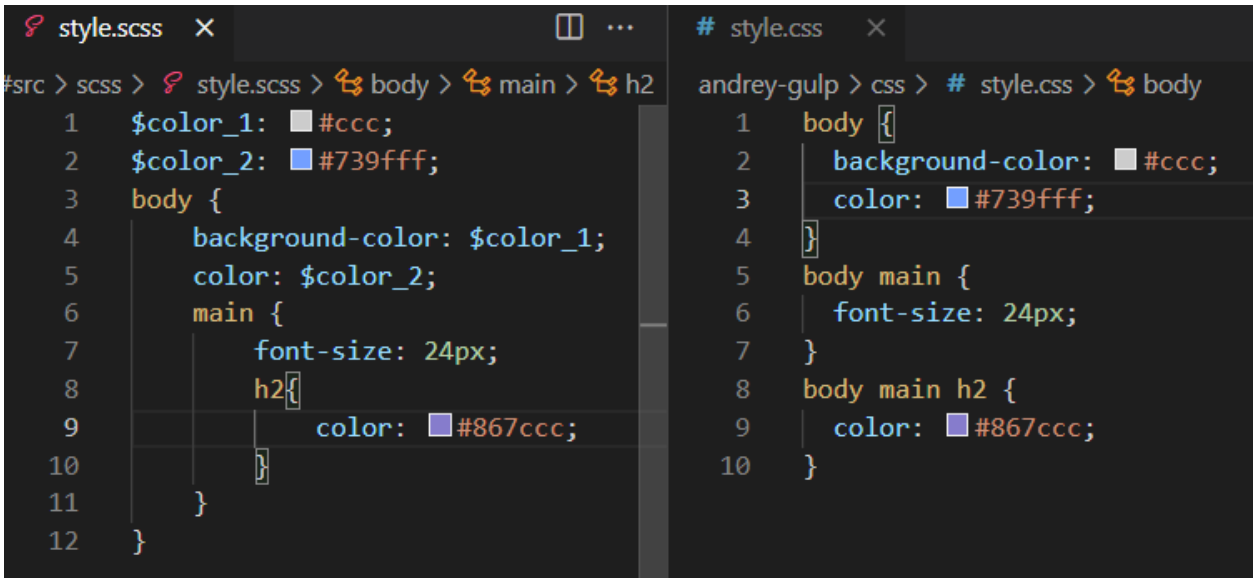

Рисунок 2 – Пример вложенности

Третьим преимуществом является амперсанд (&) – это символ, который указывает на родительский элемент при использовании вложенности. Пример на рисунке 3.

Рисунок 3 – Использование амперсанда

Четвёртое преимущество – это так называемые примеси или mixin. Это аналогия функции из классических языков программирования. Для объявления функции нужно написать следующее - @mixin *имя примеси*() {свойства;}. Пример на рисунке 4.

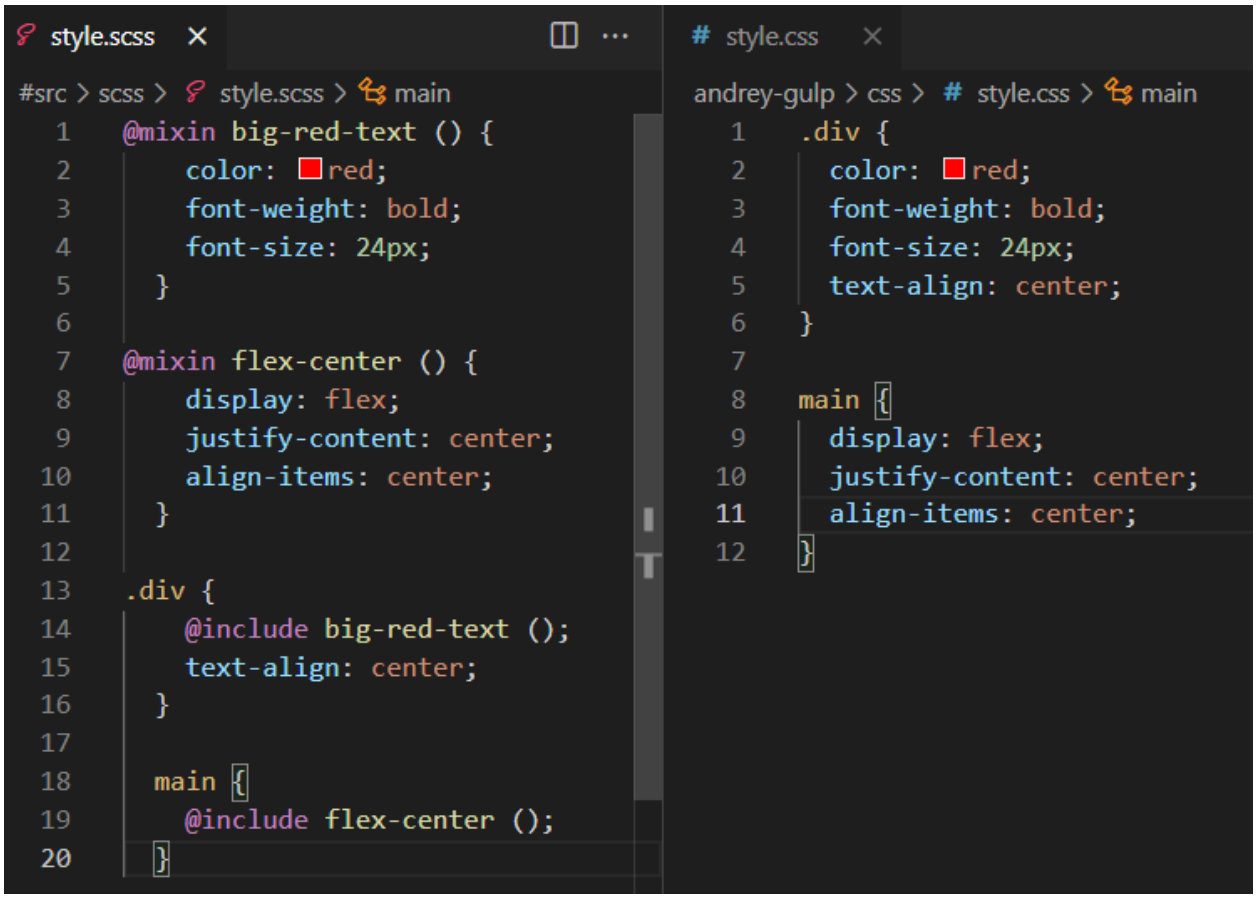

Рисунок 4 – Использование примесей, аналогов функции

Пятое преимущество – это арифметические операции. В свойстве можно использовать умножение, деление, сложение и вычитание. Примеры на рисунке 5.

style.scss

#src > scss > style.scss > div

```scss
$main_height: 20px;
div {
    padding: 1.5rem - 0.5;
    height: 10px + $main_height;
    width: 20px * 1.1;
    margin: (6px/2);
}
```

style.css

andrey-gulp > css > # style.css > d

```css
div {
  padding: 1rem;
  height: 30px;
  width: 22px;
  margin: 3px;
}
```

Рисунок 5 – Использование арифметических функций

Шестое преимущество – это операторы сравнения. Действуют также как в классических ЯП. Пример на рисунках 6 и 7, на котором определяется больший размер шрифта из двух. В первом случае больше шрифт в переменной $font-size1, а во втором $font-size2.

style.scss

#src > scss > style.scss > div

```scss
$font_size1: 18px;
$font_size2: 28px;

@mixin bigger_font() {
    @if ($font_size1 > $font_size2) {
        font-size: $font_size1;
    } @else {
        font-size: $font_size2;
    }
}

div {
    @include bigger_font();
}
```

style.css

andrey-gulp > css > # style.css > div

```css
div {
  font-size: 28px;
}
```

Рисунок 6 – Использование операторов сравнения

style.scss

#src > scss > style.scss > $font_size1

```scss
$font_size1: 38px;
$font_size2: 28px;

@mixin bigger_font() {
    @if ($font_size1 > $font_size2) {
        font-size: $font_size1;
    } @else {
        font-size: $font_size2;
    }
}

div {
    @include bigger_font();
}
```

style.css

andrey-gulp > css > style.css > div

```css
div {
    font-size: 38px;
}
```

Рисунок 7 – Использование операторов сравнения

Седьмым преимуществом являются логические операторы «и, или, не». Пример показан на рисунках 8 и 9.

style.scss

#src > scss > style.scss > button-size

```scss
@mixin button-size ($height, $width) {
    @if (($height < $width) and
    ($width >= 35px)) {
        box-shadow: none;
    } @else {
        box-shadow: 2 2 2px #ccc;
    }
}

.button {
    @include button-size(10px, 34px)
}
```

style.css

andrey-gulp > css > style.css > .button

```css
.button {
    box-shadow: 2 2 2px #ccc;
}
```

Рисунок 8 – Использование логических операторов

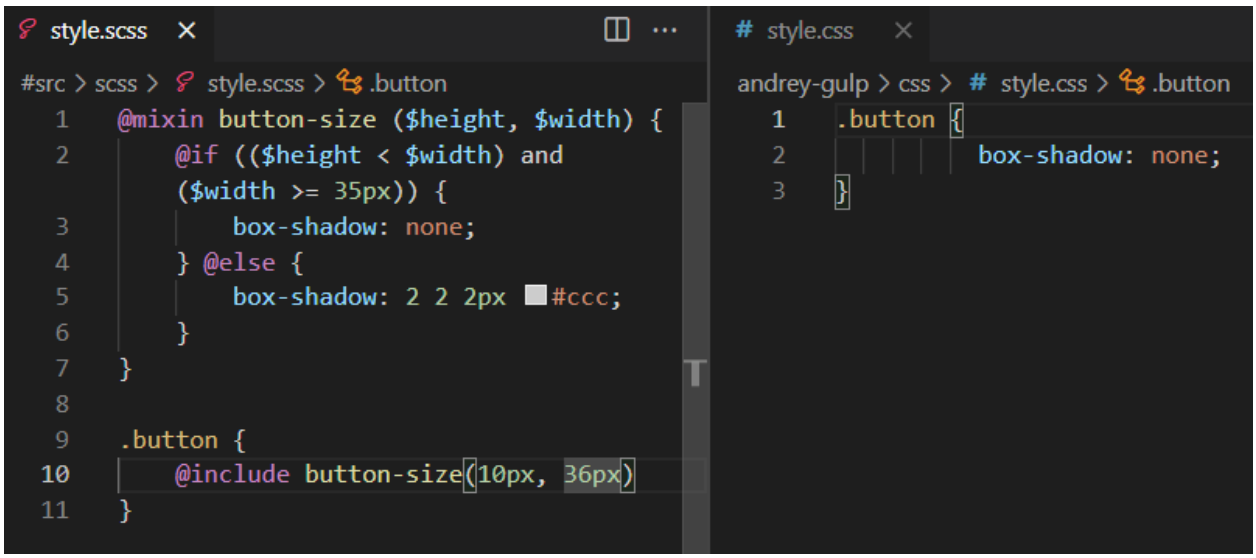

Рисунок 9 – использование логических операторов

Восьмое преимущество – возможность использования for, if, each и while. Пример показан на рисунке 10.

style.scss

#src > scss > style.scss > ...

```scss
@for $i from 1 through 3 {
    .div__block-#{$i} {height: 20px * $i}
}
/*---*/
main {
    @if 1 + 1 == 2 { color: royalblue;
    }
    @if 1 + 2 == 2 { color: red;  }
}
/*---*/
@each $content in one, two, three {
    .block-#{$content} {
        content: "#{$content}";
    }
}
/*---*/
$boats: 0;
@while $boats < 3 {
    .boat-#{$boats} { margin: 20px *
    $boats; }
    $boats: $boats + 1;
}
```

style.css

andrey-gulp > css > style.css > ...

```css
.div__block-1 {
  height: 20px;
}
.div__block-2 {
  height: 40px;
}
.div__block-3 {
  height: 60px;
}
/*---*/
main {
  color: royalblue;
}
/*---*/
.block-one {
  content: "one";
}
.block-two {
  content: "two";
}
.block-three {
  content: "three";
}
/*---*/
.boat-0 {
  margin: 0px;
}
.boat-1 {
  margin: 20px;
}
.boat-2 {
  margin: 40px;
}
```

Рисунок 10 – Пример использования функций и директив

Таким образом, наглядно видно, как сильно SASS с синтаксисом SCSS помогает облегчить жизнь фронтэнд разработчика, дав ему инструмент, который во многих моментах может заменить множество маленьких JavaScript файлов, а значит ускорить загрузку сайтов. SCSS является удачной смесью каскадных таблиц стилей и подхода к написанию кода пришедшему от классических языков программирования.

Библиографический список

1. Пантелеева Е. В. Разработка сайта с использованием языка разметки HTML, таблицы стилей CSS и языка программирования JavaScript. //

Информационные системы и технологии в образовании, науке и бизнесе. 2020. С. 94-96.
2. Лушников Н. Д., Альтерман А. Д. Основы каскадных таблиц стилей (CSS). // Наука и образование: новое время. 2019. №. 1. С. 69-72.
3. Надеинский Л. А. Архетип модуля каскадных таблиц стилей с LESSCSS компилятором. М.,2018.
4. Айдарбаев Н. О. Адаптивный дизайн веб-сайта с использованием фронтэнд-фреймворка Bootstrap // Молодой ученый. 2018. №. 21. С. 115-119.
5. Макарова Т. В. Веб-дизайн. М,2015.
6. Gulp // URL: https://gulpjs.com/ (дата обращения: 21.08.2021)
7. Node.JS // URL: https://nodejs.org/en/ (дата обращения: 21.08.2021)