

04, апрель 2016

УДК 004.434

Препроцессоры CSS

Сергеев И.Л., студент
*Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»*

*Научный руководитель: Гапанюк Ю.Е., к.т.н., доцент,
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»*
gapyu@bmstu.ru

Введение

В настоящее время в области интернет-программирования существует широкий спектр технологий разработки веб приложений.

Одновременно с развитием средств разработки серверной части растёт число средств создания клиентских интерфейсов и элементов веб дизайна. Так же появляется множество новых возможностей для старых традиционных средств. Так, с появлением CSS3 и HTML5 стало возможным создание пользовательских элементов, которые раньше создать без использования JavaScript было невозможно. В спецификации CSS3 присутствуют возможности создания анимированных элементов управления, градиентов, теней и сглаживания.

Одновременно с ростом возможности инструмента растёт сложность его использования. Типичный CSS код может оказаться трудным для восприятия при поддержке или его редактировании. Это является, прежде всего, причиной того, что ему присуща особенность сильно разрастаться по мере добавления в него новых правил. Во-вторых, на трудность восприятия могут влиять такие вещи, как вложенность элементов при написании правил CSS, слабая модульность и масштабируемость кода.

Для решения проблем такого рода были придуманы CSS-Препроцессоры. Сначала хотелось бы остановиться на самом термине «препроцессор». Препроцессор – это программа, принимающая данные на входе и выдающая данные, предназначенные для входа другой программы. В данном случае происходит компиляция из языка CSS-препроцессоров, считающимся расширением традиционного CSS в сам CSS код.

Использование препроцессоров решает, прежде всего, проблему масштабируемости и модульности и добавляет вложенность, наиболее адекватную для восприятия.

Данная статья является результатом исследования двух наиболее популярных CSS – препроцессоров- SASS и LESS и выделения, по мнению автора, ключевых их возможностей, позволяющих облегчить труд веб-разработчикам. Стоит сказать, что SASS написан на языке Ruby, а LESS – на JavaScript.

Далее, в следующих разделах будут рассмотрены ключевые особенности SASS и LESS, а так же приведено сравнение этих особенностей в виде таблицы.

Переменные

Самое простое, что поддерживают SASS и LESS, – это переменные. В данном случае, это константы, к которым можно обращаться из свойств при написании правил. Здесь синтаксисы двух препроцессоров практически идентичны, существует лишь небольшая разница в том, как объявляются эти переменные.

Ниже на рисунках 1-2 приведены два фрагмента кода для демонстрации использования переменных.

SASS	LESS
<pre>\$mainColor:#ff1e00; \$font:Open Sans, sans-serif; body{ background-color: \$mainColor; font-family: \$font }</pre>	<pre>@mainColor:#ff1e00; @font:Open Sans, sans-serif; body{ background-color: @mainColor; font-family: @font }</pre>

Рис. 1. Использование переменных

Такие участки кода скомпилируются в набор стандартных CSS правил.

```
body {
  background-color: #ff1e00;
  font-family: Open Sans,sans-serif;
}
```

Рис. 2. Скомпилированный код с использованием переменных

Такая возможность придаёт лёгкость в управлении. Ведь, поменяв значение переменной один раз в нашем коде при объявлении, её значения поменяются везде при её вызове. Это можно использовать при создании множества стилей для одного и того же сайта.

Вложенность

Второй, пожалуй, самой известной возможностью, облегчающей труд дизайнерам, является вложенность. При написании стилей для страницы, имеющей сложную структуру, возникает необходимость писать стили для её элементов, которые могут быть вложены в другие, причём на разной глубине. Иногда ещё возникает необходимость группировки CSS-правил. CSS-препроцессор сделает всё необходимое за разработчиков при компиляции.

```
body {  
    #header{  
        background-color: red;  
        .title {  
            color: white;  
            font-size: 30px;  
        }  
        a {  
            display: block;  
        }  
    }  
}
```

Рис. 3. Вложенность

Такой код, представленный на рис. 3, наиболее понятен человеку, нежели традиционный код CSS, написанный для этих элементов. В данном случае ссылка и элемент с классом **title** вложены в элемент, имеющий атрибут id, равный **header**, который в свою очередь вложен в тег **body**. При компиляции такого кода создадутся следующие правила:

```

body #header {
    background-color: red;
}

body #header .title {
    color: white;
    font-size: 30px;
}

body #header a {
    display: block;
}

```

Рис. 4. Компиляция с использованием вложенности

Наследование

Встречаются случаи, когда один селектор должен иметь стили другого, дополняясь своими свойствами. Такая ситуация со свойствами селекторов подобна наследованию классов в языках программирования высокого уровня. Существует способ использовать общий селектор, наследуя от него более типизированные:

SASS	LESS
<pre> .button{ border: 1px #f00; display: block; } .warning_button{ @extend .button; background-color: red; } .success_button{ @extend .button; background-color: green; } </pre>	<pre> .button{ border: 1px #f00; display: block; } .warning_button{ &:extend(.button); background-color: red; } .success_button{ &:extend(.button); background-color: green; } </pre>

Рис. 5. Наследование

Допустим, у нас есть дизайн кнопки, селектор которой имеет класс **button**. Мы хотим получить стили для дизайна кнопок с классом **warning_button**, показываемых при требовании подтверждения от пользователя, и для обычной кнопки с классом **success_button**. Не обязательно писать все три класса, дублируя общие свойства для каждого из них, достаточно лишь сделать наследование от общего с помощью так

называемой директивы **extend**, реализующей наследование свойств классов в препроцессорах CSS. Таким образом, наши правила сгенерируются в следующий код:

```
.button, .warning_button, .success_button {  
    border: 1px none #f00;  
    display: block;  
}  
  
.warning_button {  
    background-color: red;  
}  
  
.success_button {  
    background-color: green;  
}
```

Рис. 6. Компиляция с использованием наследования

Миксины

Миксины или примеси – это куски правил, которые можно повторно вызывать в коде. В SASS, в отличие от LESS, для объявления миксина используется директива **mixin**, после неё стоит имя миксина и блок, определяющий его тело. В остальном синтаксисы SASS и LESS касательно примесей довольно похожи. В примере ниже объявляется миксин, содержащий правило для настройки шрифта, с последующим его вызовом.

SASS	LESS
<pre>@mixin large-text { font: { family: Verdana; size: 20px; weight: bold; } color: #ff0000; }</pre>	<pre>.large-text() { font-family: Verdana; font-size: 20px; font-weight: bold; color: #ff0000; }</pre>
<pre>.page-title { @include large-text; padding: 4px; margin-top: 10px; }</pre>	<pre>.page-title { .large-text(); padding: 4px; margin-top: 10px; }</pre>

Рис. 7. Миксины

```
.page-title {
  color: #ff0000;
  font-family: Verdana;
  font-size: 20px;
  font-weight: bold;
  margin-top: 10px;
  padding: 4px;
}
```

Рис. 7. Компиляция с использованием миксинов

Циклы

Очень важной особенностью CSS препроцессоров является возможность использования циклов. Использование таких выражений приближает их язык к полноценным языкам программирования высокого уровня. В SASS такие конструкции реализованы с помощью директив **for**, **each** и **while**. В LESS подобные вещи называются закольцовываниями и развиты куда более слабо. Циклы – это очень удобный инструмент при создании модульной сетки для сайта, подобной той, что предоставляет популярный CSS – Фреймворк **Bootstrap**.

SASS

```
$width : 960px;
$gutter : 20px;
$columns : 12;
$colum_width : $width/$columns;

.grid{
  display: inline;
  float: left;
  margin-right: $gutter/2;
  margin-left: $gutter/2;
}

@for $i from 1 through $columns {
  .col-#{ $i } {
    @extend .grid;
    width: $i * $colum_width - $gutter;
  }
}
```

Рис. 8. Циклы

В примере на рис. 8 показано определение классов для блоков, соответствующих 12 колонкам. Хотелось бы обратить ещё внимание на одну вещь: в теле цикла используется выражение **#{ \$i }**. Это называется интерполяцией. Значение переменной,

являющейся итератором по циклу, будет выведено на каждой итерации. Интерполяция в таком виде наследовалась SASS от языка **Ruby**.

После компиляции код преобразуется в следующий:

```
.grid, .col-1, .col-2, .col-3, .col-4, .col-5,  
.col-6, .col-7, .col-8, .col-9, .col-10, .col-11,  
.col-12 {  
  display: inline;  
  float: left;  
  margin-right: 10px;  
  margin-left: 10px; }  
  
.col-1 {  
  width: 60px; }  
  
.col-2 {  
  width: 140px; }  
  
...  
  
.col-12 {  
  width: 940px; }
```

Рис. 9. Компиляция с использованием циклов

Управляющие директивы

SASS поддерживает базовые управляющие директивы **if** и **else** для подключения стилей при определённых условиях или подключения одних и тех же стилей несколько раз с изменениями.

SASS

```
$type: monster;  
p {  
  @if $type == ocean {  
    color: blue;  
  } @else if $type == matador {  
    color: red;  
  } @else if $type == monster {  
    color: green;  
  } @else {  
    color: black;  
  }  
}
```

Рис. 10. Ветвления

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

Рис. 11. Компиляция с использованием ветвлений

Такие управляющие выражения предназначены для использования, в основном, в миксинах. Они в определённых задачах могут сильно облегчить труд веб-дизайнера, освободив его от необходимости описывать множество классов, которые может просто скомпилировать CSS – Препроцессор.

Сравнение

В таблице ниже приведено сравнение двух технологий по уже разобранным возможностям, которые принимаем в качестве критериев оценки. Кроме того, к ним добавлены встроенные функции, которые чаще всего используются в веб дизайне при работе с цветом, и возможность определения пользовательских функций. В SASS можно определять свои функции, используя при этом Ruby API.

Сравнение CSS - препроцессоров

	SASS	LESS
Переменные	Есть	Есть
Вложенность	Есть	Есть
Наследование	Есть	Есть
Миксины	Есть	Есть
Реализация Циклов	Хорошая	Слабая
Условные директивы	Есть	Нет
Встроенные функции	Есть	Нет
Возможность создания пользовательских функций	Есть	Нет

Как видно из таблицы, SASS реализует все функции, принятые как критерии оценки. Осталось сказать ещё, что для установки SASS на свой компьютер необходимо установить интерпретатор с языка Ruby, а для установки LESS необходимо всего лишь

подключить в браузере определённый файл JavaScript, являющийся «компилятором» и сам файл с расширением less. Однако необходимо заметить, что в данном случае будет «компиляция» на стороне клиента в браузере «на лету» при открытии веб-страницы, в то время как SASS будет «компилироваться» на стороне компьютера разработчика.

Заключение

В данной статье представлены два самых популярных CSS – препроцессора. Какой из них лучше, сказать нельзя, несмотря на разную реализацию их функций. Любители языка Ruby и Фреймворка для разработки веб-приложений Ruby on Rails используют SASS, а разработчики JavaScript отдают предпочтение LESS. Тем не менее, каждый из них имеет широкое сообщество разработчиков. Каждая из рассмотренных технологий позволяет веб-дизайнерам использовать её возможности для облегчения своего труда. Целью данной статьи являлось показать начинающим веб-разработчикам о существовании этих технологий, а более опытным – привести их сравнение.

Список литературы

- [1]. Дакетт Дж. HTML и CSS. Разработка и дизайн веб-сайтов: пер с англ./ под ред. Обручева В. М.: Эксмо, 2013 , 480 с. [Duckett J. HTML and CSS. Design and Build Websites: Wiley, 2011, 490 p.].
- [2]. Официальный сайт препроцессора SASS. Режим доступа: <http://sass-scss.ru> (дата обращения: 17.11.2015).
- [3]. Официальный сайт препроцессора LESS. Режим доступа: <http://lesscss.org/> (дата обращения: 17.11.2015).