| | |
|---|---|
| ДИСЦИПЛИНА | Фронтенд и бэкенд разработка |
| ИНСТИТУТ | ИПТИП |
| КАФЕДРА | Индустриального программирования |
| ВИД УЧЕБНОГО МАТЕРИАЛА | Методические указания к практическим занятиям по дисциплине |
| ПРЕПОДАВАТЕЛЬ | Загородних Николай Анатольевич<br>Краснослободцева Дарья Борисовна |
| СЕМЕСТР | 4 семестр, 2025/2026 уч. год |

# Практическое занятие №1

# CSS-препроцессоры

Рассмотрим SASS/LESS, вложенность, переменные, миксины для фронтенда. Решение практического задания осуществляется внутри соответствующей рабочей тетради, расположенной в СДО.

## Что такое препроцессор

Любой препроцессор – это программа, принимающая на входе некоторые данные, и возвращающая в результате работы некоторые данные, предназначенные для обработки другой программой.

В контексте CSS препроцессор – это программа, которой на вход дается код, написанный на языке препроцессора, а на выходе мы получаем CSS, который можем дать на вход нашему браузеру.

## Преимущества использования CSS-препроцессоров

CSS-препроцессоры позволяют расширить функционал базового синтаксиса CSS, упрощая и ускоряя работу над стилизацией веб-приложений.

Среди особенностей препроцессоров, в том числе SASS и LESS, можно выделить следующие синтаксические конструкции:

· *переменные* — можно определить цвета, размеры и другие часто используемые значения один раз и применять их по всему проекту;

· *вложенность* — возможность создавать иерархические селекторы, повторяющие структуру HTML;

· *миксины* — многократное использование блоков стилей в разных местах;

· *функции* — математические операции и манипуляции со значениями;

· *модульность* — разделение стилей на отдельные файлы с возможностью их импорта.

При стилизации веб-приложений на CSS без использования препроцессоров перед разработчиком возникает ряд проблем, таких как дублирование кода, повторяющиеся значения, громоздкие селекторы, большие монолитные файлы и другие. Такие проблемы и предназначены решать препроцессоры.

> ⚠ **Warning**
>
> Важно понимать, что препроцессоры не заменяют знание базового CSS — они надстраиваются над ним, требуя понимания основных принципов верстки сайта. При этом они делают сложные вещи проще, а рутинные задачи — быстрее.

## Сравнение SASS и LESS

В таблице представлены основные синтаксические и функциональные отличия между препроцессорами.

| **Функциональность** | **SASS** | **LESS** |
|---|---|---|
| Переменные | $variable | @variable |
| Миксины | @mixin name() {...} | .name() {...} |
| Наследование | @extend .class; | .class; |
| Вложенность | Да | Да |
| Операции с цветами | lighten(), darken(), mix() | lighten(), darken(), mix() |
| Условные выражения | @if, @else | when, and, not |
| Циклы | @for, @each, @while | Только рекурсией |
| Математические операции | Да, с единицами | Да, с ограничениями |

## Установка Sass

Sass можно установить несколькими способами:

- **Через npm (рекомендуемый метод):** `npm install -g sass`
- **Через Ruby:** `gem install sass`

> 🔥 **Tip**
>
> После установки можно компилировать Sass-файлы в CSS с помощью команды: `sass input.scss output.css`

Для автоматической компиляции при каждом сохранении файла используйте флаг --watch:
`sass --watch input.scss:output.css`

### Установка LESS

LESS также имеет несколько способов установки:

- **Через npm:** `npm install -g less`
- **Через CDN в браузере (для разработки):** `<script src="//cdnjs.cloudflare.com/ajax/libs/less.js/3.11.1/less.min.js"></script>`

> **Tip**
>
> Компиляция LESS в командной строке:
> `lessc styles.less styles.css`

## Синтаксис SASS и LESS

### Переменные

Первое заметное отличие между препроцессорами – способ объявления переменных.

Создадим переменные `primary-color` и `font-size`, которым присвоим значения произвольных цветов и шрифтов соответственно.

Пример в SASS:

```
$primary-color: #3498db;
$font-size: 16px;
```

Пример в LESS:

```
@primary-color: #3498db;
@font-size: 16px;
```

### Работа с миксинами

Как мы уже поняли, миксин - это аналог функции в языках программирования. Создадим миксин, который на вход принимает значение поворота и применяет его к свойствам `-webkit-transform`, `-ms-transform` и `transform`.

Пример в SASS:

```
@mixin transform($property) {
    -webkit-transform: $property;
    -ms-transform: $property;
    transform: $property;
}

.box {
    @include transform(rotate(30deg));
}
```

Пример в LESS:

```
.transform(@property) {
    -webkit-transform: @property;
    -ms-transform: @property;
    transform: @property;
}

.box {
    .transform(rotate(30deg));
}
```

## Условные конструкции

Создадим переменную `theme` со значением `dark`, а также условную конструкцию, в которой проверим, что переменная `theme` имеет значение `dark`. В случае, если переменная действительно имеет такое значение, зададим свойство `background-color` значением `#333`, а свойство `color` значением `white`, в ином случае зададим значения наоборот.

Пример в SASS:

```
$theme: dark;

body {
    @if $theme == dark {
        background-color: #333;
        color: white;
    } @else {
        background-color: white;
        color: #333;
    }
}
```

Пример в LESS:

```
@theme: dark;

body when (@theme = dark) {
    background-color: #333;
    color: white;
}

body when not (@theme = dark) {
    background-color: white;
    color: #333;
}
```

## Импорт файлов

Рассмотрим пример импортирования файлов в препроцессорах.

Пример в SASS:

```
// Не компилирует файл в отдельный CSS
@import "variables";

// С Sass 7+
@use "variables" as vars;
```

Пример в LESS:

```
// Просто импорт
@import "variables";

// Можно использовать опции
@import (reference) "variables"; // только для использования миксинов
```

## Примеры использования синтаксиса препроцессоров

Давайте рассмотрим использование синтаксических конструкций обоих препроцессоров на небольшом практическом примере: создадим небольшую карточку на HTML и стилизуем её, применяя полученные знания о переменных, вложенности, условиях и миксинах.

Предлагается создать контейнер (саму карточку), в который поместить название и кнопку.

### SASS

HTML

```
<div class="card card--accent">
    <h2 class="card__title">Профиль</h2>
    <a class="btn" href="#">Открыть</a>
</div>
```

SASS

```
$primary: #3498db;
$theme: dark;

// Миксин для кнопки.
// Принимает цвет фона в качестве параметра и задаёт базовый внешний вид кнопки.
@mixin button($color) {
    padding: 10px 14px;
    background: $color;
    color: white;
    border-radius: 8px;
    text-decoration: none;
}

// Вложенный элемент карточки
.card {
    padding: 16px;

    // Условный блок для тёмной темы.
    // Срабатывает только если значение переменной @theme равно dark.
    @if $theme == dark {
        background: #222;
        color: white;
    }

    &__title {
        font-size: 20px;
    }

    // Модификатор карточки с акцентной рамкой
    &--accent {
        border: 2px solid $primary;
    }
}

// Кнопка
.btn {
    // Подключаем миксин и передаём основной цвет
    @include button($primary);
}
```

CSS, который получаем на выходе

```css
.card {
    padding: 16px;
    background: #222;
    color: white;
}
.card__title {
    font-size: 20px;
}
.card--accent {
    border: 2px solid #3498db;
}

.btn {
    padding: 10px 14px;
    background: #3498db;
    color: white;
    border-radius: 8px;
    text-decoration: none;
}
```

## LESS

HTML

```html
<div class="card card--accent">
    <h2 class="card__title">Профиль</h2>
    <a class="btn" href="#">Открыть</a>
</div>
```

LESS

```less
@primary: #3498db;
@theme: dark;

// Миксин для кнопки.
// Принимает цвет фона в качестве параметра и задаёт базовый внешний вид кнопки.
.button(@color) {
    padding: 10px 14px;
    background: @color;
    color: white;
    border-radius: 8px;
    text-decoration: none;
}
```

```less
.card {
    padding: 16px;
}

// Условный блок для тёмной темы.
// Срабатывает только если значение переменной @theme равно dark.
.card when (@theme = dark) {
    background: #222;
    color: white;
}

// Вложенный элемент карточки
.card {
    &__title {
        font-size: 20px;
    }

  // Модификатор карточки с акцентной рамкой
    &--accent {
        border: 2px solid @primary;
    }
}

// Кнопка
.btn {
    // Подключаем миксин и передаём основной цвет
    .button(@primary);
}
```

CSS, который получаем на выходе

```css
.card {
    padding: 16px;
}
.card {
    background: #222;
    color: white;
}
.card__title {
    font-size: 20px;
}
.card--accent {
    border: 2px solid #3498db;
}
.btn {
    padding: 10px 14px;
    background: #3498db;
    color: white;
    border-radius: 8px;
```

```
    text-decoration: none;
}
```

## Итог

Используя разные препроцессоры и, соответственно, разный синтаксис, скомпилировали итоговые .css файлы. Сравнивая результат, можно увидеть, что содержимое получилось идентичным.

## Практическое задание

Необходимо реализовать карточку товара (содержимое: название, описание, фотография) с применением переменных (минимум 2), миксина (минимум 1) и использованием вложенной структуры селекторов на любом из двух препроцессоров (SASS или LESS).

## Формат отчета

В качестве ответа на задание необходимо прикрепить ссылку на репозиторий с реализованной практикой. Ссылка подгружается в соответствующий раздел СДО: Задания для самостоятельной работы -> Практические занятия 1-2.

## Литература


1. [Развернутое руководство по SASS] (https://tproger.ru/translations/complete-sass-guide)
2. [LESS: программируемый язык стилей] (https://habr.com/ru/articles/136525)