Приднестровский государственный университет им. Т.Г. Шевченко
Физико-технический институт

ОСНОВЫ ВЕБ-РАЗРАБОТКИ С ПРИМЕНЕНИЕМ ИИ-АССИСТЕНТОВ

Лабораторный практикум

Разработал:
ст. преподаватель
кафедры ИТ
Бричаг Д.В.

г. Тирасполь
2025 г.

Лабораторная работа №3

CSS-препроцессоры: LESS и SASS.

Цель работы:

− Познакомиться с концепцией CSS-препроцессоров, понять их роль в современном фронтенд-разработке.

− Научиться компилировать LESS/SASS в обычный CSS, использовать переменные, вложенность, миксины и импорты.

Теоретическая справка

Проблема обычного CSS

Когда проект становится большим, обычный CSS начинает создавать неудобства:

- трудно поддерживать и редактировать длинные файлы;
- невозможно использовать переменные, функции, вложенные селекторы и другие инструменты, привычные программистам;
- часто приходится повторять одинаковые фрагменты кода.

Чтобы сделать работу с CSS удобнее, были придуманы CSS-препроцессоры.

Что такое препроцессор

CSS-препроцессор — это надстройка над CSS, которая расширяет его возможности и делает написание стилей более логичным и компактным.

Разработчик пишет код не в обычном CSS, а в специальном языке — например, LESS или SASS, — а затем этот код компилируется (переводится) в обычный CSS-файл, понятный браузеру.

То есть браузер не понимает напрямую .less или .scss файлы — перед использованием их обязательно нужно скомпилировать в .css.

Зачем нужны препроцессоры

Использование препроцессоров позволяет:

- использовать переменные для хранения цветов, шрифтов, размеров и других значений;
- создавать вложенные селекторы, чтобы код выглядел структурированнее;
- использовать миксины — аналоги функций, чтобы не повторять одно и то же;
- писать условия и циклы в стилях;
- разделять код на отдельные файлы и подключать их по необходимости.

В результате код становится:

- короче и аккуратнее;
- легче поддерживать;
- проще масштабировать при росте проекта.

Характеристика	**LESS**	**SASS/SCSS**
Расширение	.less	.sass или .scss
Переменные	@variable	$variable
Написан на	JavaScript	Ruby (изначально), сейчас есть dart-sass
Используется в	старых проектах, Bootstrap 3	в современных, Bootstrap 5, React
Компиляция	через npm-пакет less	через npm-пакет sass
Особенности	проще, ближе к CSS	мощнее, гибче

Таблица – LESS vs SASS – краткое сравнение

Как это работает на практике

1. Разработчик пишет стили в файле styles.less или styles.scss.
2. Компилятор (например, установленный через **npm**) переводит этот файл в styles.css.

3. В HTML подключается уже готовый styles.css:

```
<link rel="stylesheet" href="styles.css">
```

Основные возможности CSS-препроцессоров

CSS-препроцессоры расширяют стандартный синтаксис CSS, добавляя в него переменные, вложенные селекторы, миксины, импорт и другие инструменты, которые делают код компактнее и удобнее для поддержки.

Ниже рассмотрены основные возможности на примерах для LESS и SASS/SCSS.Синтаксис у них похож, поэтому разобраться несложно.

1. Переменные

Проблема: в обычном CSS приходится многократно повторять одинаковые цвета, размеры и шрифты.

Решение: в препроцессорах можно объявлять переменные и использовать их в стилях.

LESS:

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

body {
  color: @main-color;
  font-size: @font-size;
}
```

SASS:

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

body {
  color: $main-color;
  font-size: $font-size;
}
```

2. Вложенные селекторы

Проблема: в обычном CSS нужно постоянно повторять родительские селекторы.

Решение: в LESS и SASS можно вкладывать селекторы друг в друга, как в HTML.

LESS:

```
nav {
  background: #eee;

  ul {
    list-style: none;

    li {
      display: inline-block;
    }
  }
}
```

SASS:

```
nav {
  background: #eee;

  ul {
    list-style: none;

    li {
      display: inline-block;
    }
  }
}
```

После компиляции это превратится в обычный CSS с полными селекторами.

3. Миксины

Миксин — это как функция: вы определяете набор свойств один раз и можете подключать его в разных местах.

LESS:

```
.rounded-corners(@radius: 5px) {
  border-radius: @radius;
}

button {
  .rounded-corners(10px);
}
```

SASS:

```
@mixin rounded-corners($radius: 5px) {
  border-radius: $radius;
}

button {
  @include rounded-corners(10px);
}
```

Импорт файлов

Можно разбивать код на несколько файлов и подключать их в главный файл стилей.

LESS:

```
@import "variables.less";
@import "buttons.less";
```

SASS/SCSS (расширение .scss можно не указывать):

```
@import "variables";
@import "buttons";
```

Вычисления и операции

LESS и SASS поддерживают математические операции прямо в коде:

LESS:

```
@base: 10px;
.container {
  padding: @base * 2;
}
```

SASS/SCSS:

```
$base: 10px;
.container {
  padding: $base * 2;
}
```

Результат компиляции

Все эти конструкции преобразуются в обычный CSS, который понимают браузеры. Например:

```
.container {
  padding: 20px;
}
button {
  border-radius: 10px;
}
```

Что такое npm

npm (Node Package Manager) — это менеджер пакетов для платформы Node.js. С его помощью можно устанавливать, обновлять и удалять различные инструменты и библиотеки, используемые при веб-разработке.

Проще говоря, npm — это как магазин программ для JavaScript: он скачивает нужные пакеты из интернета и устанавливает их на компьютер, чтобы вы могли использовать их в своих проектах.

Когда мы устанавливаем компилятор LESS или SASS через npm, мы получаем программу, которая может «переводить» файлы .less или .scss в обычный CSS, понятный браузеру.

Пример установки

Перед этим нужно установить **Node.js**, так как npm идёт в комплекте с ним. Проверить наличие Node.js и npm можно командами:

```
node -v
npm -v
```

Установить компилятор можно так:

```
npm install -g less
```

или для Sass:

```
npm install -g sass
```

После этого можно использовать команды:

```
lessc styles.less styles.css
```

или

```
sass styles.scss styles.css
```

Проверка компиляции и подключение стилей к HTML

После установки компилятора **LESS** или **SASS** (через npm) можно приступить к первому запуску.

1. Подготовка файлов

Для подготовки файлов необходимо создать папку для проекта, например lesson3, тогда файлы внутри неё будут иметь следующую структуру:

```
lesson3/
│
├── styles.less      ← исходный файл с препроцессором
├── index.html       ← веб-страница
```

(Если используется SASS, то создаётся styles.scss вместо styles.less)

2. Пример содержимого styles.less (или styles.scss)

```
@main-color: #3498db;

body {
  background: #f5f5f5;
  color: @main-color;
  font-family: Arial, sans-serif;

  h1 {
    text-align: center;
  }
}
```

3. Компиляция в CSS

Для компиляции файла с расширением .css, который понимает браузер необходимо выполнить скрипт, который сформирует из .less/.scss файла файл с расширением .css. При помощи терминала (или консоли) в папке проекта выполняется команда:

Для LESS:

```
lessc styles.less styles.css
```

Для SASS:

```
sass styles.scss styles.css
```

После выполнения появится файл styles.css — это уже готовый CSS, который понимает браузер.

4. Подключение CSS к HTML

Теперь необходимо добавить в файл index.html в тег <head> ссылку на скомпилированный CSS:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>LESS / SASS пример</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Привет, LESS и SASS!</h1>
</body>
</html>
```

5. Проверка результата

Теперь просто открываем index.html в браузере как и раньше. Если всё сделано правильно, то можно увидеть заголовок, оформленный стилями из вашего компилированного CSS.

Наблюдатели – вотчеры - watchers

Вотчеры часто используются в веб разработке для компиляции шаблонов, препроцессоров, библиотек и js-фреймворков. Если вы хотите, чтобы компилятор автоматически отслеживал изменения в исходном файле и перекомпилировал его при каждом сохранении, можно использовать режим “watch”:

LESS**:**

```
lessc --watch styles.less styles.css
```

SASS**:**

```
sass --watch styles.scss styles.css
```

Теперь достаточно просто редактировать .less или .scss — CSS будет обновляться автоматически.

Практическая часть

Сформируем простое техническое задание. Необходимо:

1. Создать папку проекта и его структуру
2. Установить и настроить компилятор SASS или LESS.
3. Создать небольшую стилизованную страницу (например, «профиль разработчика») с:
   - использованием переменных, миксинов и вложенности;
   - адаптивностью (через медиазапросы).
4. Скомпилировать препроцессорный код в CSS.

1. Подготовка

Если ещё нет — установите **Node.js** (https://nodejs.org). Проверьте установку:

```
node -v
npm -v
```

2. Создайте проект

```
mkdir lab3
cd lab3-preprocessors
npm init -y
```

3. Установите компилятор

Для SASS:

```
npm install -g sass
```

или локально:

```
npm install --save-dev sass
```

Для LESS:

```
npm install -g less
```

4. Создайте файлы

```
index.html
styles/
  ├── style.scss
  └── _variables.scss
```

5. Напишите SCSS код

В _variables.scss:

```
$main-color: #4CAF50;
$text-color: #333;
```

В style.scss:

```
@import 'variables';

body {
  font-family: Arial, sans-serif;
  color: $text-color;
  background-color: #f8f8f8;

  .card {
    border: 1px solid $main-color;
    border-radius: 8px;
    padding: 1rem;

    @media (max-width: 600px) {
      padding: 0.5rem;
    }
  }
}
```

6. Скомпилируйте SCSS → CSS

В терминале:

```
sass styles/style.scss styles/style.css
```

или с автообновлением:

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

7. Подключите CSS в index.html

```
<link rel="stylesheet" href="styles/style.css">
```

8. Проверка адаптивности

Откройте страницу в браузере → уменьшайте ширину окна → убедитесь, что медиазапросы работают.

Использование ИИ для генерации шаблона

Примеры промптов:

```
"Сгенерируй простой пример SCSS со структурой страницы профиля: аватар, имя, кнопка. Используй переменные и вложенность."
```

```
"Создай пример LESS-файла для сайта-визитки с миксином для кнопок и медиазапросами."
```

Другие запросы:

```
Покажи, как скомпилировать SASS в CSS в VS Code через терминал.
```

```
Объясни разницу между SASS и CSS простыми словами
```

```
Покажи пример использования циклов и условий в SASS
```

Редактирование вручную

− Доработайте проект, чтобы он содержал файлы для медиазапросов, миксинов и переменных.

− Создайте тёмную тему для страницы, используя переменные в препроцессоре.

− Проверьте работу сетки при изменении ширины окна браузера.

Проверка адаптивности

− Откройте сайт в браузере.

− Сожмите окно до ширины смартфона (~400px).

− Проверьте, что элементы перестраиваются корректно.

Работа с git и GitHub

1. Инициализируйте репозиторий:

```
git init
git add .
git commit -m "Lab3: initial commit"
```

2. Создайте новый репозиторий на GitHub.

3. Свяжите локальный проект с GitHub:

```
git remote add origin https://github.com/<ваш_логин>/<имя_репозитория>.git
git branch -M main
git push -u origin main
```

Отчёт (в README.md)

В README.md добавить:

- Заголовок: *Лабораторная работа №3*

- Скриншоты: версия для светлой и тёмной темы.
- Ответы на вопросы:
  - Почему выбрали этот препроцессор?
  - Какие файлы для компонентов препроцессора создавали?
  - Каким образом реализована тёмная тема страницы?
  - Какие запросы делали LLM?

Результаты работы

В итоге у вас должно быть:

1. Файлы SASS/LESS и скомпилированные стили.
2. Адаптивная страница с переключателем светлой и тёмной темы.
3. Репозиторий на GitHub с проектом.
4. README.md с отчётом и скриншотами.

Критерии оценки

- Использован препроцессор (2 балла).
- Использованы миксины, переменные, медиазапросы в разных файлах и их импорт (2 балла).
- Реализовано переключение темы (2 балла).
- Код выложен на GitHub (2 балла).
- README.md с отчётом и скриншотами (2 балла).