SCSS + Vite

Препроцессоры и сборка

Занятие 4

Frontend Course 2025

Что такое SCSS?

> **Вопрос:** Зачем усложнять CSS еще одним инструментом?

SCSS (Syntactically Awesome Style Sheets) — это CSS с суперспособностями:

- Переменные
- Вложенность
- Миксины
- Функции
- Импорты

Компилируется в обычный CSS

SCSS vs CSS

CSS:

```
.header {
  background: #007bff;
  padding: 20px;
}

.header h1 {
  color: white;
  font-size: 24px;
}

.header .nav {
  margin-top: 10px;
}

.header .nav a {
  color: white;
  text-decoration: none;
}
```

SCSS:

```
$primary: #007bff;

.header {
  background: $primary;
  padding: 20px;

  h1 {
    color: white;
    font-size: 24px;
  }

  .nav {
    margin-top: 10px;

    a {
      color: white;
      text-decoration: none;
    }
  }
}
```

Переменные в SCSS

> **Вопрос:** Чем SCSS переменные лучше CSS переменных?

SCSS переменные компилируются в значения, CSS переменные остаются в рантайме. SCSS — для констант, CSS — для динамических значений.

```
// SCSS переменные
$primary-color: #007bff;
$font-size-large: 24px;
$breakpoint-mobile: 768px;

// CSS переменные (для тем)
:root {
  --theme-bg: #{$primary-color};
}
```

Вложенность

> **Вопрос:** Можно ли переборщить с вложенностью?

Да! Больше 3-4 уровней вложенности = плохо читается и тормозит браузер.

```
// ❌ Слишком глубоко
.page .content .sidebar .widget .title {
}

// ✅ Хорошо
.sidebar-widget {
  .title {
  }
}
```

Родительский селектор &

> **Вопрос:** Как в SCSS сделать hover или БЭМ модификаторы?

```
.button {
  background: blue;

  &:hover {
    // .button:hover
    background: darkblue;
  }

  &--large {
    // .button--large
    padding: 20px;
  }

  &__icon {
    // .button__icon
    margin-right: 8px;
  }
}
```

Миксины

> **Вопрос:** Как не повторять CSS?

Миксины: суть

Миксины — функции для CSS.

Миксины: код

```
@mixin button($bg) {
  background: $bg;
}

@include button(blue);
```

Функции SCSS

> **Вопрос:** Можно ли в SCSS делать математику?

Конечно! SCSS умеет считать и имеет встроенные функции.

```
$base-font-size: 16px;

.title {
  font-size: $base-font-size * 1.5; // 24px
}

.small-text {
  font-size: $base-font-size * 0.875; // 14px
}

.lightened {
  background: lighten(#007bff, 20%);
}
```

Импорты и модули

> **Вопрос:** Как организовать большой проект на SCSS?

```
// _variables.scss
$primary: #007bff;
$secondary: #6c757d;

// _mixins.scss
@mixin button {
  /* ... */
}

// main.scss
@import "variables";
@import "mixins";
@import "components/button";
@import "components/card";
```

Файлы с `_` не компилируются отдельно.

Что такое Vite?

> **Вопрос:** Зачем нужен еще один инструмент сборки?

Vite — это современный сборщик:

- Мгновенный запуск (ESM)
- Быстрая пересборка (HMR)
- Простая настройка
- Поддержка всех форматов
- Оптимизированная продакшн сборка

Vite vs Webpack

> **Вопрос:** Чем Vite лучше Webpack?

Webpack:

- Сложная настройка
- Медленный старт проекта
- Бандлит всё в development
- Огромная экосистема

Vite:

- Настройка из коробки
- Мгновенный старт
- ESM в development
- Rollup для production

Vite = скорость разработки, Webpack = гибкость настройки

Создание Vite проекта

```
# Создать новый проект
npm create vite@latest my-project

# Выбрать шаблон:
# vanilla, vue, react, preact, lit, svelte

# Или сразу указать:
npm create vite@latest my-app -- --template vanilla

cd my-app
npm install
npm run dev
```

Проект готов за 30 секунд!

Структура Vite проекта

```
my-project/
├── index.html          # Входная точка
├── main.js             # JavaScript код
├── style.css           # Стили
├── public/             # Статические файлы
│   └── favicon.ico
├── package.json
└── vite.config.js      # Настройки Vite
```

`index.html` в корне — особенность Vite!

Настройка SCSS в Vite

```
# Установить SCSS
npm install -D sass
```

```
// vite.config.js
import { defineConfig } from "vite";

export default defineConfig({
  css: {
    preprocessorOptions: {
      scss: {
        additionalData: `@import "@/styles/variables.scss";`,
      },
    },
  },
});
```

Теперь можно импортировать `.scss` файлы!

SCSS в проекте

```scss
// src/styles/variables.scss
$primary: #007bff;
$secondary: #6c757d;
$spacing: 16px;

// src/styles/mixins.scss
@mixin button($bg) {
  background: $bg;
  padding: $spacing;
  border: none;
  border-radius: 8px;
}

// src/components/Button.scss
@import "../styles/variables";
@import "../styles/mixins";

.button {
  @include button($primary);

  &--secondary {
    @include button($secondary);
  }
}
```

HMR (Hot Module Replacement)

> **Вопрос:** Что такое HMR и зачем он нужен?

HMR — обновление модулей без перезагрузки страницы.
Изменили CSS → стили обновились мгновенно
Изменили JS → только этот модуль перезагрузился
Состояние приложения сохраняется!

Vite поддерживает HMR из коробки для CSS, SCSS, JS, TS.

Импорты в Vite

```
// Статические ресурсы
import logoUrl from "./logo.png";
import styles from "./component.module.scss";

// CSS/SCSS файлы
import "./global.scss";
import "./component.scss";

// JSON файлы
import config from "./config.json";

// Динамические импорты
const module = await import("./dynamic-module.js");
```

Vite понимает множество форматов!

CSS Modules

> **Вопрос:** Как избежать конфликтов CSS классов?

CSS Modules! Автоматически генерирует уникальные классы. CSS Modules — технология изоляции стилей на уровне компонентов.

```
// Button.module.scss
.button {
  background: blue;

  &:hover {
    background: darkblue;
  }
}
```

CSS Modules: как работает

```
// Button.js
import styles from "./Button.module.scss";

const button = `<button class="${styles.button}">Click</button>`;
// class="Button_button__abc123"
```

Vite автоматически добавляет хэш к классам, делая их уникальными. Конфликты CSS исчезают!

PostCSS

> **Вопрос:** Можно ли добавить фичи в обычный CSS?

PostCSS! PostCSS — инструмент для трансформации CSS.

PostCSS: пример

```
/* Пишете */
.button {
  display: flex;
}

/* Получаете */
.button {
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
}
```

Автоматически добавляет префиксы для браузеров.

Переменные окружения

```
# .env
VITE_API_URL=https://api.example.com
VITE_APP_TITLE=My App

# .env.development
VITE_API_URL=http://localhost:3000

# .env.production
VITE_API_URL=https://prod-api.example.com
```

```
// main.js
console.log(import.meta.env.VITE_API_URL);
console.log(import.meta.env.MODE); // 'development' | 'production'
console.log(import.meta.env.DEV); // true в development
```

Только переменные с `VITE_` доступны в коде!

Сборка для продакшена

```
# Сборка
npm run build

# Предпросмотр сборки
npm run preview
```

Vite создает оптимизированную сборку:

- Минификация CSS/JS
- Tree shaking (удаление неиспользуемого кода)
- Code splitting
- Кеширование ресурсов

Плагины Vite

```
// vite.config.js
import { defineConfig } from "vite";
import { resolve } from "path";

export default defineConfig({
  plugins: [
    // Ваши плагины
  ],
  resolve: {
    alias: {
      "@": resolve(__dirname, "src"),
      "@components": resolve(__dirname, "src/components"),
    },
  },
});
```

```
// Теперь можно писать:
import Button from "@components/Button.vue";
```

Vite + TypeScript

```
# Создать проект с TypeScript
npm create vite@latest my-app -- --template vanilla-ts

# Или добавить в существующий
npm install -D typescript
```

```
// main.ts
import "./style.scss";

interface Config {
  apiUrl: string;
  debug: boolean;
}

const config: Config = {
  apiUrl: import.meta.env.VITE_API_URL,
  debug: import.meta.env.DEV,
};
```

Оптимизация производительности

> **Вопрос:** Как ускорить сборку больших проектов?

- Разделяйте код на модули
- Используйте динамические импорты
- Настройте кеширование
- Оптимизируйте изображения
- Удаляйте неиспользуемый CSS

```
// Динамический импорт
const heavyModule = await import("./heavy-feature.js");
```

Отладка SCSS

> **Вопрос:** Как найти ошибку в скомпилированном CSS?

Source Maps! Vite автоматически их генерирует.

В DevTools видите исходный SCSS файл и строку, а не скомпилированный CSS.

```
// vite.config.js
export default defineConfig({
  css: {
    devSourcemap: true, // Source maps в development
  },
});
```

Лучшие практики

- Структурируйте SCSS файлы по компонентам
- Используйте переменные для всех повторяющихся значений
- Создавайте миксины для сложных паттернов
- Не вкладывайте селекторы глубже 3-4 уровней
- Используйте CSS Modules для изоляции стилей
- Настройте автоматическое форматирование (Prettier)

Домашнее задание

- Создать Vite проект с SCSS
- Настроить структуру стилей с переменными и миксинами
- Создать адаптивные компоненты с CSS Modules
- Настроить alias для импортов
- Попробовать другой препроцессор (Less/Stylus)

Следующее занятие: Адаптивность + темы

Вопросы?

SCSS — это CSS с суперспособностями.
Vite — это скорость разработки.