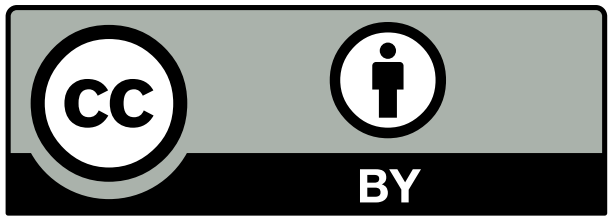
CSS Preprocessors

Péter Jeszenszky

Faculty of Informatics, University of Debrecen

November 17, 2025

Terms of Use



What is a CSS Preprocessor?

- CSS preprocessors are programs that generate CSS code from their own syntax for writing CSS.
- They extend CSS with new features such as variables, nesting, functions, mixins.
- See: MDN Web Docs Glossary – CSS preprocessor

Existing CSS Preprocessors

- Less: https://lesscss.org/
- PostCSS: https://postcss.org/
- Sass: https://sass-lang.com/
- Stylus: https://stylus-lang.com/

Examples

https://github.com/jeszy75/css-preprocessor-examples

Node.js (1)

- A JavaScript runtime environment built on the V8 JavaScript engine that is designed to build scalable network applications.
- Website: https://nodejs.org/
- Repository: https://github.com/nodejs/node
- License: MIT License
- Written in: C++, JavaScript
- Platform: Linux, macOS, Windows

Node.js (2)

- It lets the developers build applications in JavaScript that run outside a web browser.
- It can be used for client-side and server-side application development.
- Its package ecosystem, npm, is the largest ecosystem of open-source libraries in the world.

Node.js (3)

"Hello, World!" example:

```
const { createServer } = require('node:http');

const hostname = '127.0.0.1';
const port = 3000;

const server = createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, World!');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Source: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

Installing Node.js (1)

- Most implementations discussed here require Node.js.
- Installers and binaries can be downloaded from here: https://nodejs.org/en/download
- Alternatively, Node.js version managers offer a more flexible way of installation.

Installing Node.js (2)

Node.js version managers:

- Command line tools for installing Node.js without requiring system administrator privileges that allow for managing multiple versions of Node.js simultaneously.

Installing Node.js (3)

Node.js version managers:

- asdf (written in: shell; platform: Linux, macOS; license: MIT License) https://asdf-vm.com/ https://github.com/asdf-vm/asdf
- Fast Node Manager (fnm) (written in: Rust; platform: Linux, macOS, Windows; license: GPLv3) https://github.com/Schniz/fnm
- Node Version Manager (nvm) (written in: shell; platform: Linux, macOS; license: MIT License) https://github.com/nvm-sh/nvm
- Volta (written in: Rust; platform: Linux, macOS, Windows; license: Simplified BSD License) https://volta.sh/ https://github.com/volta-cli/volta

Installing Node.js (4)

Installing and using fnm:

```
$ curl -fsSL https://fnm.vercel.app/install | bash
$ fnm list-remote # Lists all remote versions
$ fnm list # Lists all locally installed versions
$ fnm install 25
$ fnm use 25
$ fnm current
$ node --version
$ fnm install 16
$ fnm use 16
$ fnm current
$ node --version
```

Common Features of CSS Preprocessors (1)

Single-line comments:

- Single-line comments start with //, and go until the end of the line.
- They don't produce any CSS.
- See: https://sass-lang.com/documentation/syntax/comments

Common Features of CSS Preprocessors (2)

`@import`:

- `@import` at-rules are treated differently by CSS preprocessors compared to web browsers.
- Plain CSS imports require the web browser to make HTTP requests as it renders the page.
- CSS preprocessors handle imports during compilation.
- See: https://sass-lang.com/documentation/at-rules/import

Nesting of Style Rules (1)

- Related style rules whose selectors share the same component can be grouped by nesting.
- Nesting organizes style rules, thus, it improves readability and maintainability.
- See: https://sass-lang.com/documentation/style-rules/#nesting

Nesting of Style Rules (3)

Example:

```
/* Without nesting: */
article {
  background-color: aliceblue;
}

article a {
  color: green;
}

article > h2 {
  text-decoration: underline;
}

article + p {
  font-size: smaller;
}
```

```
/* With nesting: */
article {
  background-color: aliceblue;
  a {
    color: green;
  }
  > h2 {
    text-decoration: underline;
  }
  + p {
    font-size: smaller;
  }
}
```

Nesting of Style Rules (3)

Example:

```
/* Without nesting: */
table.board {
  border-collapse: collapse;
}

table.board td {
  border: thin solid black;
}

table.board td, table.board th {
  height: 3em;
  text-align: center;
  vertical-align: middle;
  width: 3em;
}
```

```
/* With nesting: */
table.board {
  border-collapse: collapse;
  td {
    border: thin solid black;
  }
  td, th {
    height: 3em;
    text-align: center;
    vertical-align: middle;
    width: 3em;
  }
}
```

Nesting of Style Rules (4)

Example: parent selector (&)

```
/* Without nesting: */
p {
  text-indent: 1em;
}
h2 + p {
  text-indent: 0;
}
h2 + p::first-letter {
  font-size: 1.5em;
}
p:hover {
  background-color: gold;
}
```

```
/* With nesting: */
p {
  text-indent: 1em;
  h2 + & {
    text-indent: 0;
    &::first-letter {
      font-size: 1.5em;
    }
  }
  &:hover {
    background-color: gold;
  }
}
```

CSS Nesting Module (1)

- The following CSS specification introduces the nesting of style rules, a feature widely offered by CSS preprocessors:
  - CSS Nesting Module (W3C Working Draft, 14 February 2023)
- Browser support: https://caniuse.com/css-nesting

CSS Nesting Module (2)

- Currently, the specification does not allow the selectors of nested style rules to start with a type selector or a functional notation.
- However, in November 2025, both Firefox and Google Chrome support the invalid form in the example.

CSS Nesting Module (3)

Example:

```css
/* This is invalid nesting: */
table.board {
  td, th {
    height: 3em;
    width: 3em;
  }
}
```

```css
/* These are valid nestings: */
table.board {
  & td, & th {
    height: 3em;
    width: 3em;
  }
  :is(td, th) {
    height: 3em;
    width: 3em;
  }
}
```

Mixins (1)

- Mixins provide a means to group CSS declarations for reuse.
- Mixins can also take arguments.
- See: https://sass-lang.com/documentation/at-rules/mixin/

Mixins (2)

Example:

```
/* Sass (SCSS): */
@mixin rounded-border {
  border: thin solid black;
  border-radius: 0.25em;
}

nav {
  @include rounded-border;
  background-color: azure;
}

p {
  @include rounded-border;
  background-color: linen;
}
```

Mixins (3)

Example:

```
/* Sass (SCSS): */
@mixin set-size($size) {
  height: $size;
  width: $size;
}

table.chessboard td {
  @include set-size(2em);
}
```

Sass

- Website: https://sass-lang.com/
- Repository: https://github.com/sass/sass
- File extension: `.sass`/`.scss`

Sass: Features

See: https://sass-lang.com/guide

- Variables
- Arithmetic operations
- Built-in functions (e.g., logical, math, string, and color manipulation functions)
- Maps
- Flow control constructs (`@if`, `@each`, `@for`, `@while`)
- Extend/Inheritance
- Modules (`@use`)
- ...

Sass: Syntax (1)

Sass supports the following two syntaxes:

- SCSS: a superset of CSS syntax.
  - File extension: `.scss`
- Indented syntax: Sass's original syntax that uses indentation instead of curly braces and semicolons.
  - File extension: `.sass`

See: https://sass-lang.com/documentation/syntax

Sass: Syntax (2)

Example:

```scss
/* SCSS: */
$size: 2em;

table.chessboard {
  td {
    height: $size;
    width: $size;
  }
}
```

```sass
/* Indented syntax: */
$size: 2em

table.chessboard
  td
    height: $size
    width: $size
```

Sass: Syntax (3)

- Advantages of the indented syntax:
  - A more concise syntax compared to the SCSS syntax, which is easier to write.
  - It enforces writing clean code.
- The indented syntax comes from the Haml markup language from which Sass originates.
  - Haml: https://haml.info/

Sass: Fun Facts

- For historical reasons, Sass identifiers, such as variable, mixin, and function names, treat hyphens and underscores as identical.
  - This means that, for example, `$font-size` and `$font_size` both refer to the same variable.
  - See: https://sass-lang.com/documentation/variables/

Sass: Expressions

- Sass supports the use of expressions, each of which produces a value.
- The syntax of Sass expressions is called SassScript.
  - Further information: https://sass-lang.com/documentation/syntax/structure#expressions
- Sass supports additional data types (e.g., boolean, null) beyond those available in CSS.

Sass: Implementations

- Dart Sass (written in: Dart; license: MIT License)
  https://sass-lang.com/dart-sass https://github.com/sass/dart-sass
  - The primary implementation of Sass.
  - It is also available, compiled to JavaScript, as an npm package (`sass`).
- Ruby Sass (written: Ruby; license: MIT License)
  https://sass-lang.com/ruby-sass https://github.com/sass/ruby-sass
  - The original, now deprecated Ruby implementation of Sass.

Sass: Installation

See: https://sass-lang.com/install

- Download standalone Dart Sass:
  https://github.com/sass/dart-sass/releases/
- Install Sass with `npm`:

```
# Installing the JavaScript implementation of Sass:
npm install -g sass
```

Sass: Editors

Free and open source software:

- Visual Studio Code (platform: Linux, macOS, Windows; license: MIT License) https://code.visualstudio.com/ https://github.com/Microsoft/vscode
  - Extensions:
    - Live Sass Compiler https://marketplace.visualstudio.com/items?itemName=glenn2223.live-sass https://glenn2223.github.io/vscode-live-sass-compiler/ https://github.com/glenn2223/vscode-live-sass-compiler

See: https://code.visualstudio.com/docs/languages/css

Sass: Online Tools

- CodePen
- Sass.js

Sass: Usage

Command line use (Dart Sass):

```
sass input.scss output.css
sass input1.scss:output1.css input2.scss:output2.css
sass ./style/
sass --watch ./style/
```

Sass: Related Tools

SassDoc (license: MIT License) http://sassdoc.com/
https://github.com/SassDoc/sassdoc

- A documentation tool for Sass.
- Documentation comments in Sass are written with three slashes (///).
- The text of documentation comments is parsed as Markdown.
- Several annotations (e.g, `@author`, `@parameter`, `@return`) are available in documentation comments for providing details.

Projects Using Sass

- Many modern CSS frameworks, such as Bootstrap, Foundation, or Bulma are written is Sass.
- Developers need to use Sass for low-level customization of Bootstrap.
  - See: https://getbootstrap.com/docs/5.3/customize/sass/

Sass: Further Information

- Sass Guidelines

Mixin Libraries

- Hover (license: MIT License) https://github.com/IanLunn/Hover https://ianlunn.github.io/Hover/
- Open Color (license: MIT License) https://yeun.github.io/open-color/ https://github.com/yeun/open-color

Comparison of CSS Preprocessors

- Brian Jackson. CSS Preprocessors – Sass vs LESS. January 13, 2023.
- Jordan Irabor. Less vs Sass vs Stylus. February 20, 2022.

Source Maps (1)

- Source maps are files that tell browsers or other tools that consume CSS how generated CSS code corresponds to the files from which it was generated.
- Using source maps, web browser developer tools can show the original source from which CSS code was generated for the user.
  - This feature is helpful for debugging purposes.
- Relevant specification:
  - John Lenz, Nick Fitzgerald. Source Map Revision 3 Proposal. February 11, 2011.

Source Maps (2)

- Sass support:
  https://sass-lang.com/documentation/cli/dart-sass#source-maps
  - Dart Sass generates source maps by default for every CSS file it emits.

Source Maps (3)

Browser support:

- Firefox:
  - Style Editor – Source map support
- Chromium, Google Chrome:
  - Meggin Kearney, Paul Bakaus, Sofia Emelianova. Chrome DevTools – Debug your original code instead of deployed with source maps.