SASS

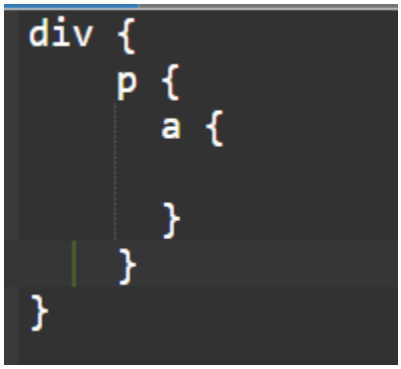
Introduction

Till now we have learned to write a lot of CSS, but now we will learn how to write CSS more efficiently. To write CSS more efficiently and easily, we shall use SASS

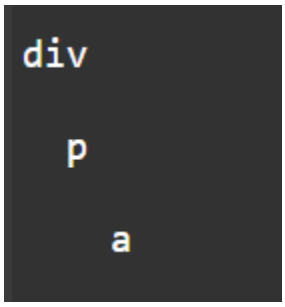
SCSS and SASS are two different ways of writing CSS.

SCSS or SASS?

- SCSS is the most commonly used and SASS is called the indented syntax.
- SCSS stands for (**Sassy Cascading Style Sheets**) and it's an extension of CSS which adds nested rules, variables, mix in selectors, inheritance, and a lot more features.
- Sass (**Syntactically Awesome Style Sheets)** is an extension of CSS that enables you to use things like variables, nested rules, inline imports, and more. It also helps to keep things organized and allows you to create stylesheets faster.
- At the end of it, both SASS and SCSS are converted to look like CSS because browsers only understand CSS.
- In SCSS, we need brackets and semicolons for the styling properties but in SASS we don't need brackets and semicolons.
- SCSS looks like this -



- SASS looks like this -



- ***EXTRA*** **-** You can refer to the following link for a further read

  https://sass-lang.com/documentation

Setting Up SCSS

- Now it is time to dive into the code and add SCSS in the project. We will be converting the SCSS to CSS later.
- We will be using **Node SASS middleware** which is an npm package.
- We will write our code in SCSS. Post that, whenever the server starts, the middleware or the npm package will convert SCSS into CSS.
- Whenever the views reference a stylesheet they will be referring to CSS because the CSS thing will be served to the browser.
- Install using the command **{ npm install node-sass }**.
- After installing, require it in the file **{ index.js }**.
- We have to create a new folder in assets named **{ SCSS }**.

- We need to put the SASS middleware just before the server is starting because we need those files to be precompiled before the server starts. Whenever the templates ask for CSS, these precompiled files will be given back.

  The properties inside the **app.use** would be -

- **Source** - Path from where we pick up the SCSS files to convert them into CSS.
- **Destination** - Path where we put the CSS files.
- **Debug Mode** - Whatever information we see while the server is running over the terminal, do we want to display some error that is there in the compilation of files while conversion. If yes then set it to true. {If it is in production mode then we have to set it to false }.
- **OutputStyle** - Do we want everything to be in a single line or do we want it in multiple lines. If we want in multiple lines then use “extended” otherwise “compressed”.
- **Prefix** - It is basically the location where the server should look for the CSS files.

```
const sassMiddleware =require('node-sass-middleware');

app.use(sassMiddleware({
    src: './assets/scss',
    dest: './assets/css',
    debug: true,
    outputStyle: 'extended',
    prefix: '/css'
}));
```

EXTRA **-** You can refer to the following link for a further read

https://github.com/sass/node-sass#:~:text=Node-sass%20is%20a%20library%20that%20provides%20binding%20for,incredible%20speed%20and%20automatically%20via%20a%20connect%20middleware.

Using SCSS + Mini Assignment

- The files do not get compiled at the time of starting the server.
- Server compiles the files whenever we load the page it is required on, which makes it very slow.
- When we are in production mode, we have to send all the files beforehand.
- The assignment is to create the header and footer styles.
- Create two new files in the SCSS folder **{ header.scss & footer.scss }.**
- In the **{layout.ejs }** file, link the header.css and footer.css files.
- After reloading, the SCSS files will get converted into CSS files from { header.scss & footer.scss }.

Assignment- You can replicate the design by yourself as is mentioned in the video. {Write the styles in terms of SCSS }.

Summarizing

- We have used **node-sass-middleware:** a package to use SCSS.
- Two ways of writing the style are SCSS and SASS. Although we learnt how to work with SCSS in the lecture.
- Whenever we run the server and initialise a page, the **node-sass-middleware** converts the SCSS to CSS and links it to the views.