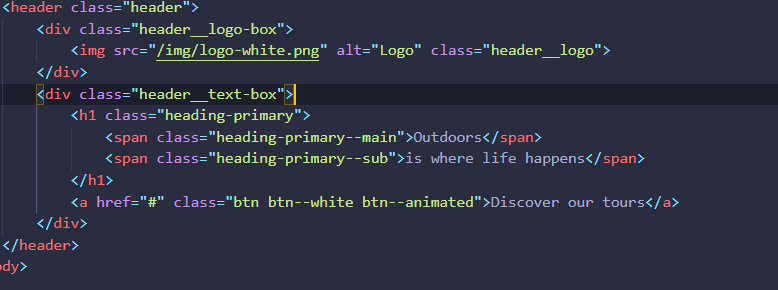
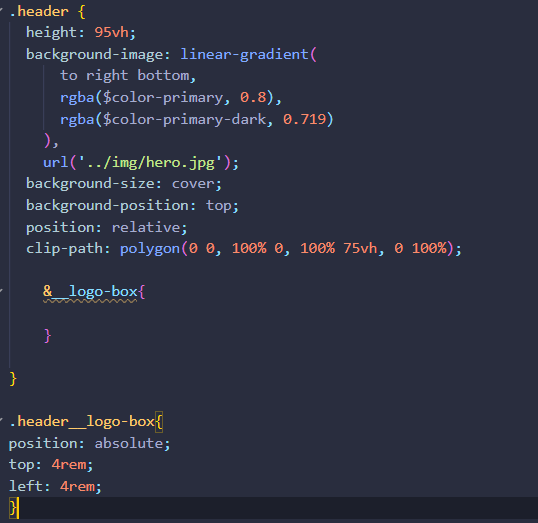
Nesting

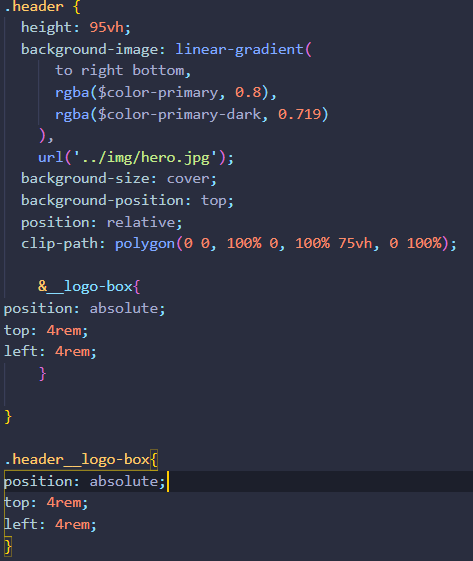
It’s possible to nest sub-classes into main classes and this is how you do it:



In this block of code you can see a header with the class “header”. Inside the header class there are sub classes, like: “header\_\_logo-box” , “header\_\_text-box”. These subclasses are nestable inside the Sass stylesheet inside the header styling.

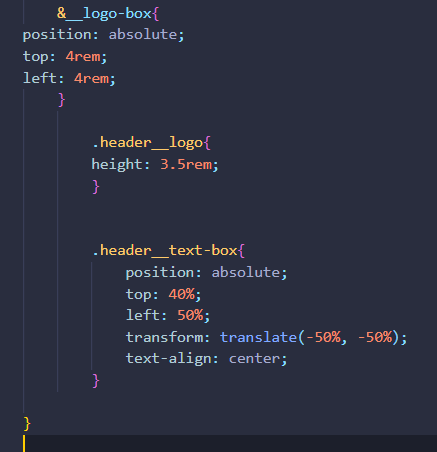
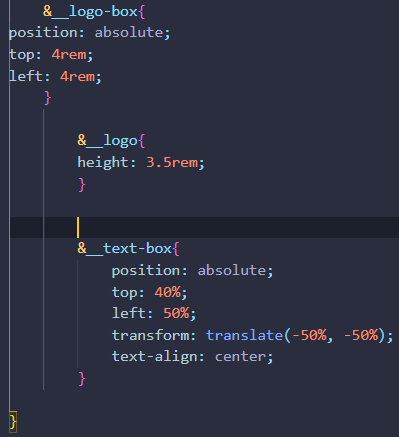
 Below the “.header” block there is an sub class named “.header\_\_logo-box”. This sub class is nestable inside the .header block because it is in the header block in the HTML file.

To INSERT the subclass just type an ampersap (&) and without a space type the sub class of the html element inside the parent class.

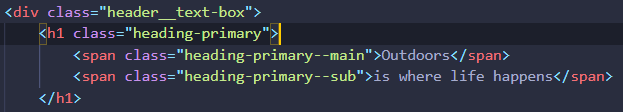


Afterwards you can delete the “.header\_\_logo-box” because you nested it succefully into the block of the parent class. This practice can also be made for other subclasses of the same parent class.

Here is an example of typing nesting other classes.

 Can be changed to : 

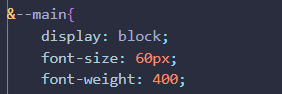
If the child classes have the same name as the parent class the child classes ‘s names should be removed, otherwise the code wont work, here’s an example:

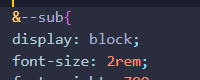


The child classes in this instance are “heading-primary--main” and “heading-primary--sub”.

In the Sass code the “heading-primary” part should be removed if nesting. Which only results in :

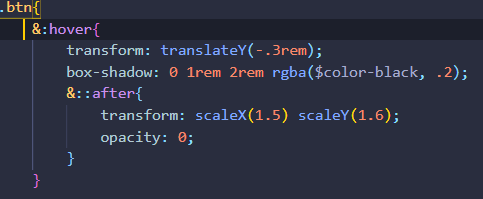
&--main and &--sub.

 Should be: 

 Should be: 

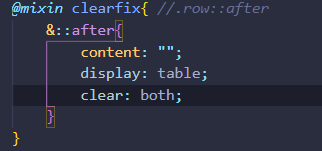
PSEUDO CLASSES

Pseudo classes can also be nested, here is an example:



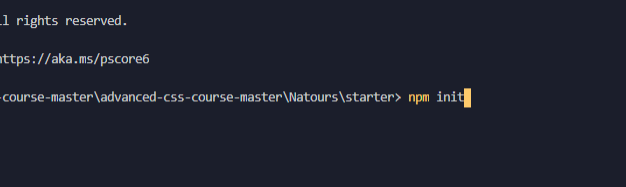
The actual meaning of &:hover is “.btn:hover” and the meaning of “&::after” is “.btn:hover::after”.  
As you can see the ::after pseudo class can be nested inside the hover block using only 1 ampersamp.

For pseudo elements to work, you always need to include the “content: “ ”;” line, otherwise it wont work. The content: “” basically means to show something on the page, if you leave it blank, it wont show something on the website.

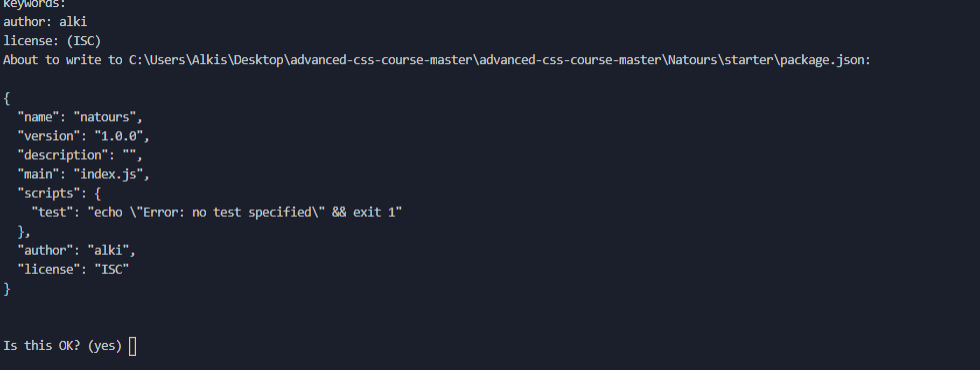
 => 

To start off firstly open the terminal within your code editor to install package.json

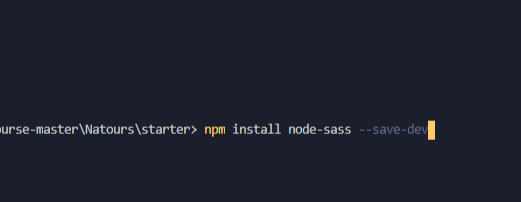
Type “ npm init ” in the terminal and press enter



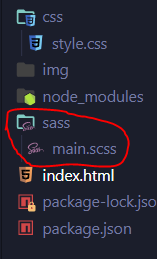
Type the package information for the package.json file it looks something like this:



After that type “npm install node-sass –save-dev-” to install the sass node package

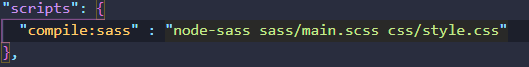


Create a new sass folder with the sass file in it.



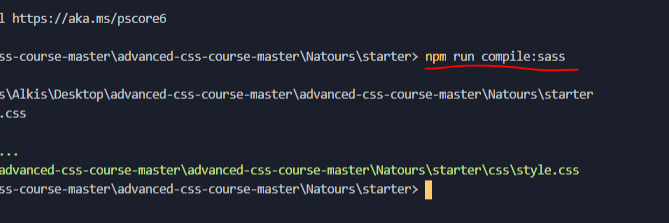
If you already have put code in to a different css style sheet, just copy the code from that stylesheet and paste it into the main.scss file.

Then go to package.json and change the “script” to



1. “ node-sass” is the package name
2. sass/main.scss is the sass file /input file
3. css/style.css is the output file (compiler)
4. “compile:sass” is declared to call the name in the run command

First locate to the sass folder and then run the command, type “npm run compile:sass”



To automatically update after the main.scss file has been edited add “ -w” in the script in the package.json file and run the command line in the sass folder with the same command again (“npm run compile:sass”).

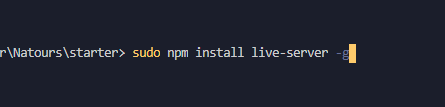


You can install live server globally by typing : “ npm install live-server -g ”. You shouldn’t be running into any errors while doing this.  
If you are on a MAC, after trying installing live-server and It gives you an error, try typing “sudo” before the npm install live server. So:

Windows:



Mac:



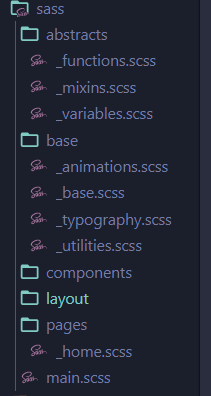
After installation is complete you can activate the live server by typing:

Live-server in the command line/ terminal. But make sure you are on the main project folder and not any sub folder.

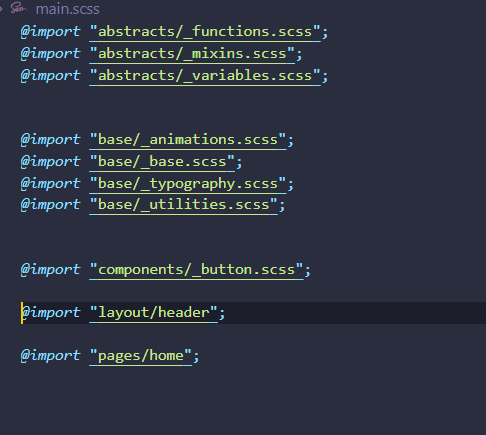


Folders buildup

When working on a website that is rather large, organization would be the key. Organizing you sass files is one of the key components for a successful website. That way you can locate every function of stylesheet when you need to change something. Here is an example of how a folder structure should look like when including multiple files.



Open your main.scss stylesheet and include the files like the following:



Basic responsive design principles

Fluid Grids and Layouts

To allow content to easily adapt to the current viewport width user to browse the website. U

Uses % rather than px for all layout related lengths

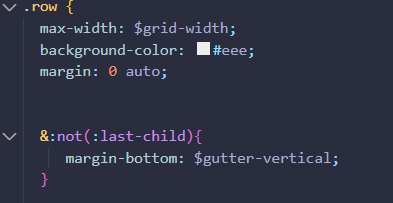
Flexible/ responsive images

Images behave differently than text content, and so we need to ensure that they also adapt nicely to the current viewport

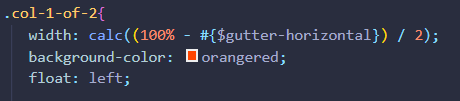
Media queries

To change styles on certain viewport widths (breakpoints), allowing us to create different version of our website for different widths.

There is a very practical way of how you can select some child elements of a class.



The &:not(:last-child) represents .row:not() in the original css stylesheet but remember, this is sass, not css.   
You can also calculate sizes with the calc() method in sass



The calc() method follows the same rules as normal mathematical equations.

To include a variable in the calc method, use #{‘$variable\_name’} , otherwise the desired outcome won’t be showed.