

Software Development Skills Front-End, Online course

Author: Kim Venho | 000430285

Lappeenrannan teknillinen yliopisto

Table of Contents

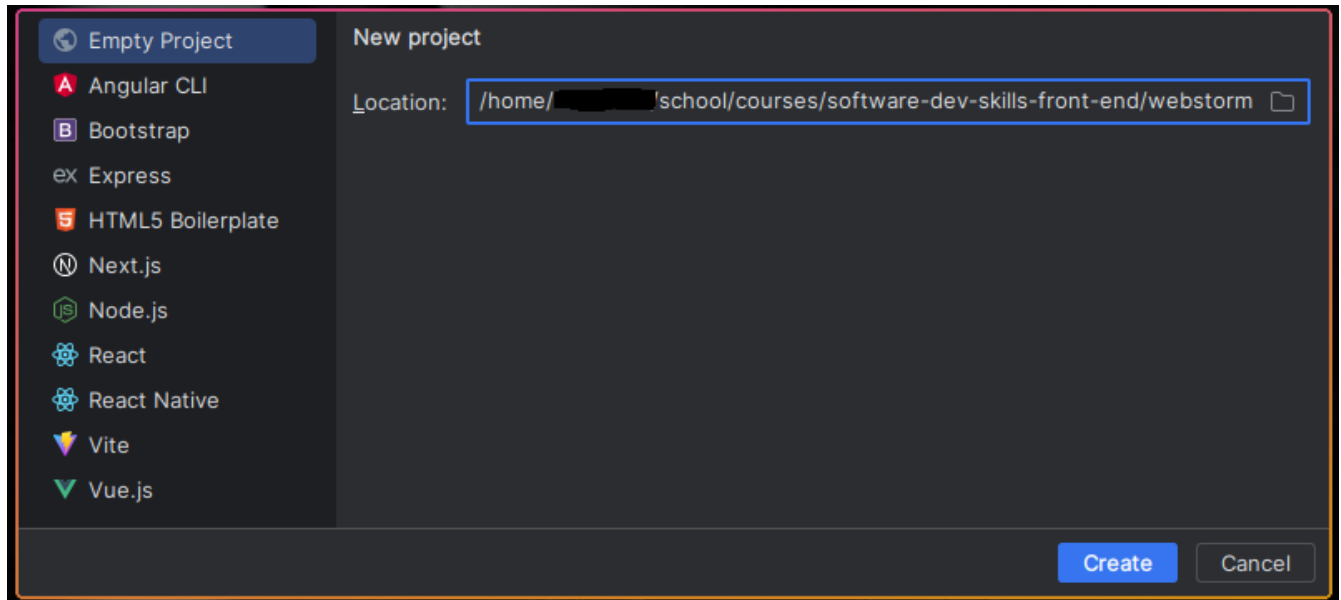
Learning Diary	1
1. Introduction to workflow and sass	1
2. Homepage and Core Sass/CSS	3
3. Rotating Menu Button	4
4. Menu Overlay & Responsiveness	9
5. Page With CSS Grid	11
6. Work and Contact Pages	13
7. Website Deployment	15
8. Final Project	16

Learning Diary

1. Introduction to workflow and sass

01.05.2023

I started by creating a new project inside WebStorm. I decided to use WebStorm because it's my favorite IDE and I happen to have a license to use every JetBrains tools they have available.



I then created index.html file, which was automatically filled by WebStorm with the html boilerplate code.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Title</title>
6  </head>
7  <body>
8
9  </body>
10 </html>
```

SCSS

Before I started to write any scss, I googled around to learn what it is and why should we use it. I found out that it is a superset for css. Any valid css is valid scss. I found out that there is also sass, which is scss — with the difference that they use indented syntax instead of curly braces. With scss you get many new features, and the .scss file at the end will be compiled to good old css which the website then will actually use.

I then created my main.scss file. A popup then asked me the following:

I have never actually needed the File Watcher, but this time it seemed to be the best solution to automatically compile the .scss file everytime I save it on the editor. I read articles from JetBrains about how the File Watcher works, and then got to work. I first installed sass npm package with: `npm install -g sass` (note: I want to install it globally since otherwise it would eat up my expensive NVMe SSD if I installed packages locally per project). I did not have rights to install to my system without using sudo, and I did not want to use sudo for this since it is security risk, so I used `npm_config_prefix=$HOME/.local` environment variable. With this the npm will save the packages to folder that it has rights without sudo.

The configuration of the File Watcher itself was pretty simple, I only changed the output path to point to the css folder:

Name: SCSS

Files to Watch

File type: SCSS style sheet ☒ Track only root files

Scope: Project Files

Tool to Run on Changes

Program: sass

Arguments: \$FileName\$: \$ProjectFileDir\$/dist/css/\$FileNameWithoutExtension\$.css

Output paths to refresh: \$ProjectFileDir\$/dist/css/\$FileNameWithoutExtension\$.css: \$ProjectFileDir\$/dist/css/\$FileNameWithoutExtension\$.css.map

Working directory: \$FileDir\$

Environment variables:

Advanced Options

☒ Auto-save edited files to trigger the watcher

☒ Trigger the watcher on external changes

☐ Trigger the watcher regardless of syntax errors

☐ Create output file from stdout

Show console: On error

Output filters:

Each line is a regular expression, available macros: \$FILE_PATH\$, \$LINE\$, \$COLUMN\$, and \$MESSAGE\$

OK Cancel

Now, I can press the debug button on the WebStorm IDE, which will create a local server and open chromium instance where this new site is running. Since I added the File Watcher, everything is updated live so the website will always show the latest revision without any restarts.

2. Homepage and Core Sass/CSS

01.05.2023

I have written some web pages on my past, so the html part was mostly just reminding me about the syntax of the language. Same goes for the css syntax (of the scss file) - but when we were using the rem sizes, I learned how the rem unit is actually calculated. 1 rem equals to size of the current scope font. So if we use 6 rem like in the upcoming image, the size will be 6 times the font that would normally be in use.

```
&.lg-heading {  
  font-size: 6rem;  
}
```

When trying to get the link to the font-awesome icon library, you needed to create an account. Instead of that, I decided to just use public CDN of older version of font-awesome.


```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
```

While we were writing the .scss file, I learned about the @mixin and @include method. It seems like very nice way to re-use a piece of css in multiple different places. Googling about it, they seem to also support arguments, which makes them really appealing option to use.

When we were setting up the background image, mine did not work. We were using the 'bg-img' id to set the image in our .scss file, but by searching for that in my index.html file, nothing was found. I checked the comments of the video and noticed that many others had the same problem. I then decided to check did I miss the part where we wrote it but no, when we created our body, at no point did we add that id to our body. The video maker most likely added it afterward and forgot to include that in the video. The fix was simple tho, just adding the 'bg-img' id to our body fixed it.

I also learned about the &:before and &:after. It allows you to add content that do not exist in the html file, via the css. Your code can be a lot cleaner with usage of these, but of course you need to be careful so that the code still stays understandable.

Finally, I learned how to split .scss files to multiple files. Those are called partials. They allow to be modular with your sass code. The partial file names should be prefixed with underscore, and then added to other scss files via @import (without underscore or filetype suffix).

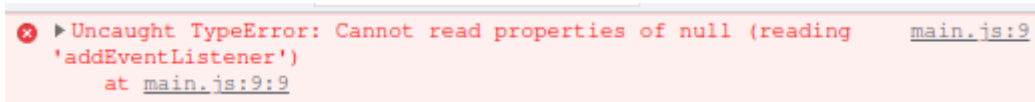
```
 _config.scss
```

```
@import 'config';
```

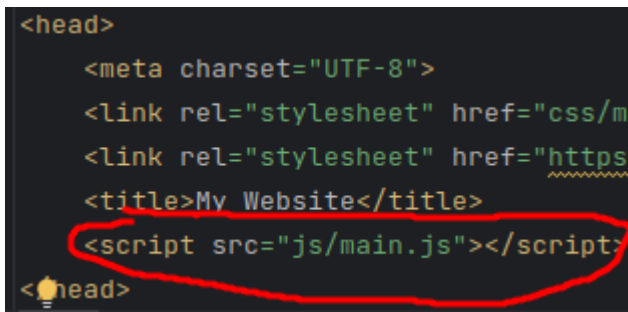
3. Rotating Menu Button

02.05.2023

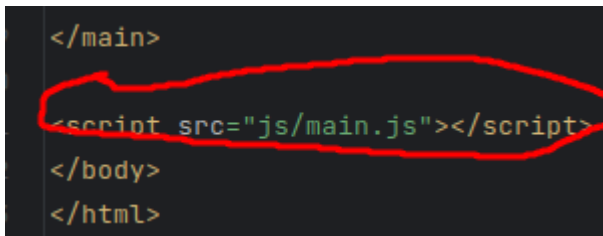
We started working on our javascript file. As with html, I have written javascript before (and a lot of java too), so this part was fairly simple for me. But I did run into a small issue. When we wrote our initial javascript that added/removed the show and close classes, I got the following error:



Just from experience I first checked that there were no typing errors. As there were none, my next debug step was to check at what point the script is run — since if we run that script before we actually have the elements in our DOM, they will not be found. I tested this by moving my script location from the start:



to the end of the file:



This fixed the issue. But this raises another issue—not an error, but preference. I don't like to declare scripts in middle of html file, since that in my opinion makes the code not very readable. You should see from glance what scripts certain files are using, without reading every single line of the code. So I put the script back to the top of the file, and modified the javascript by adding an init function that we call when the page is loaded, and declared the variables at the top of the file. Now they can be assigned in the init function, and used when needed.

```

let menuBtn;
let menu;
let menuNav;
let menuBranding;
let navItems;

let showMenu : boolean = false;

1 usage
function init() : void {
    menuBtn = document.querySelector( selectors: '.menu-btn');
    menu = document.querySelector( selectors: '.menu');
    menuNav = document.querySelector( selectors: '.menu-nav');
    menuBranding = document.querySelector( selectors: '.menu-branding');
    navItems = document.querySelectorAll( selectors: '.nav-item');

    menuBtn.addEventListener( type: 'click', toggleMenu);
}

```

```

<body id="bg-img" onload="init()">

```

I read most of the comments on the video, and found nice comment from Michael Walker replying in one comment thread:

michael Walker 3 years ago

You could be even more brief

```
// an array of all the elements except menu button;
const elements = [menu,menuNav,menuBranding,...menuItems];
let showMenu = false;

let menuToggle = ()=>{

  let toggleState = showMenu ? 'add' : 'remove'

  menuBtn.classList[toggleState]('close');
  elements.forEach(item=>item.classList[toggleState]('show'))

  showMenu = !showMenu
}
```

We take advantage of the ternary operator and the object computed property operator [] to remove the if else state all together and are instead left with clean code with little to no repeats ,

This could probably still be optimized further

Show less



Reply

So I decided to implement that myself. I added the declaration to the top:

```
// An array of all the elements except menu button;
let elements;
```

... then I assigned the elements:

```
function init() : void {
  menuBtn = document.querySelector( selectors: '.menu-btn');
  menu = document.querySelector( selectors: '.menu');
  menuNav = document.querySelector( selectors: '.menu-nav');
  menuBranding = document.querySelector( selectors: '.menu-branding');
  navItems = document.querySelectorAll( selectors: '.nav-item');

  menuBtn.addEventListener( type: 'click', toggleMenu);

  elements = [menu,menuNav,menuBranding,...navItems];
}
```

... and modified the toggleMenu function. The end result is really clean as you can see:


```
function toggleMenu() : void {
  let toggleState : string = showMenu ? 'add' : 'remove';

  menuBtn.classList[toggleState]('close');
  elements.forEach(item=>item.classList[toggleState]('show'));

  showMenu = !showMenu;
}
```

... for reference, our old toggleMenu looked like this:

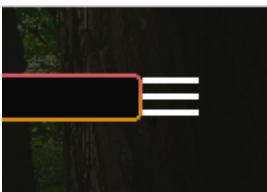
```
function toggleMenu() : void {
  if (!showMenu) {
    menuBtn.classList.add('close');
    menu.classList.add('show');
    menuNav.classList.add('show');
    menuBranding.classList.add('show');
    navItems.forEach(item => item.classList.add('show'));

    showMenu = true;
  } else {
    menuBtn.classList.remove('close');
    menu.classList.remove('show');
    menuNav.classList.remove('show');
    menuBranding.classList.remove('show');
    navItems.forEach(item => item.classList.remove('show'));

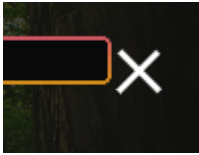
    showMenu = false;
  }
}
```

... its always fun to learn new cleaner ways to implement stuff that you might have not even thought about. As years go by, code languages gets new features — and how things should be done might change, so it's good to stay up to date with it and keep on learning.

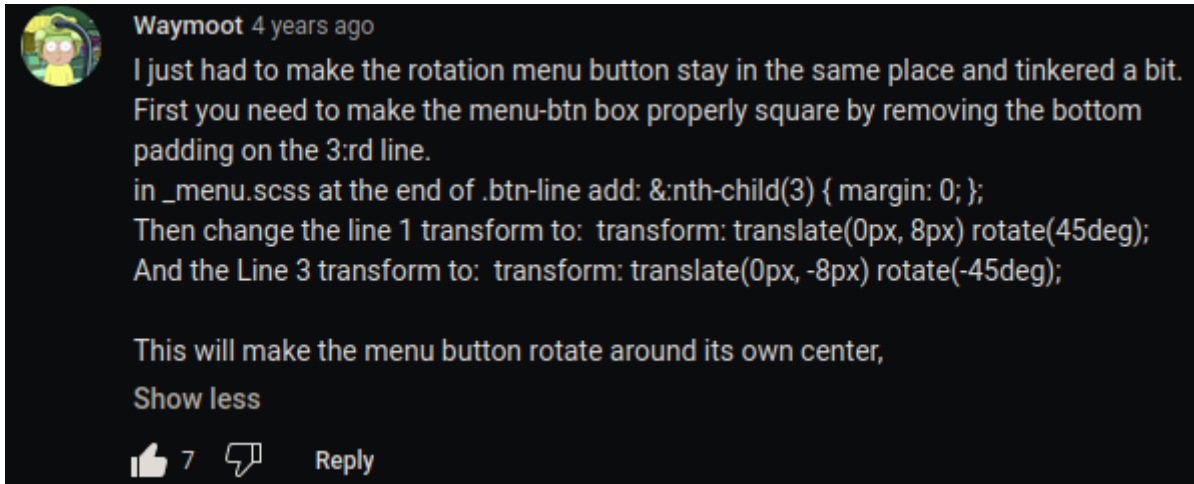
The last thing we did was creating the css transition to our hamburger menu button — we want to turn it into a cross when opened. While following the tutorial, I noticed that the instructor's animation does not rotate in place, instead it moves a bit down, which feels unpolished:



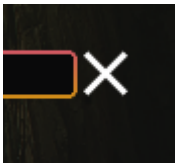
... when opened:



... I wanted to fix that, but first I decided to check the comments if someone else has fixed it already so that I don't need to reinvent the wheel. Fortunately Waymoot had the solution:



... and now the opened cross will stay in its original place:



4. Menu Overlay & Responsiveness

03.05.2023

For the first half of this tutorial, I mostly learned some neat things about sass, like how to use for loop in sass.

```
@for $x from 1 through 4 {  
  .nav-item:nth-child(#{ $x }) {  
    transition-delay: $x * 0.05s;  
  }  
}
```

... this is basic for loop that goes from 1 to 4. Note that in order to use the x inside the nth-child, we need to wrap it inside #{}. This will interpolate the value into the string. Without it, how would the sass interpreter know where your variable end when the string continues after your variable, when being connected to the variable (For example if you have variables \$chick and \$chicken, and you now use \$chickenisfood — should the interpreter paste the \$chick, \$chicken or no variables at all? What if you want to paste contents of \$chicken connected to the rest of the string, 'isfood'?).

In the second half of the tutorial we made the webpage responsive. I have in the past used bootstrap, flex and grid to make my websites responsive, but this was nice reminder on how to use css to target different sized screens.

I learned how we can add content to mixins in sass. If we declare the following:

```
@mixin mediaSm {  
  @media screen and (max-width: 500px) {  
    @content;  
  }  
}
```

... then the following will paste its contents at the previous images @content location:

```
@include mediaSm {  
  ~~~~~  
  main#home h1 {  
    margin-top: 10vh;  
  }  
}
```


... the end result on the compiled css:

```
@media screen and (max-width: 500px) {  
  main#home h1 {  
    margin-top: 10vh;  
  }  
}
```

... pretty neat! It is interesting to regularly check the compiled css, to see what magic the sass has

done behind the scenes.

As per usual - reading the comments, I fixed an issue that was present on the site:



Mike Hutchings 4 years ago

The portrait is transparent, allowing the text behind it to be visible, because the whole .menu is given transparency.

Remove this:

```
.menu {  
  opacity: 0.9;  
}
```



and change the background colors of .menu-branding and .menu-nav to use rgba instead of just the hex codes:


```
.menu {  
  &-nav {  
    background: rgba(darken($primary-color, 5), 0.9);  
  }  
  &-branding{  
    background: rgba($primary-color, 0.9);  
  }  
}
```

Now only the menu backgrounds will be transparent, and not its content.

Thanks for the awesome tutorials, Brad!

Show less

 182  Reply

 [22 replies](#)

... thanks Mike!

5. Page With CSS Grid

04.05.2023

Today we started with creating a function for our text colors. The goal was to have black text when our background is light, and vice-versa. The function syntax followed the normal '@' tags, just like in everywhere else:

```
@function setTextColor($color) {  
  @if (lightness($color) > 40) {  
    @return black;  
  } @else {  
    @return white;  
  }  
}
```

... but note the camel case that I used out of habit. I then started to look what else functions sass has, since now we used lightness function that was a new one for me. A little bit of googling later, I found out about adjust-hue, adjust-color, saturate and desaturate. The thing that I noticed was that sass seems to use dash (-) separator for the function syntax. I originally used the camel case since @mixins use those, but it appears that we have different syntax for functions, so I changed mine to use it too:

```
@function set-text-color($color) {
```

Next we started to work on grid. I've used grid before, so this was also fairly straight-forward for me. I remember learning how to use grid from <https://cssgridgarden.com/> game. It was very fun way to learn grid.

What I did learn while doing this was the calc() function. I remember having some issues back in the day on making sticky footer. The result code seemed way too complicated for how simple thing it in the end is. But calc function is very helpful function for this task.

After the tutorial was over, I wanted to go back to the set-text-color function, and change the if-else statement to be ternary. This was because I think it is more clear way for short simple if-else statement, and also I wanted to learn how to do it in sass. First I put it like this:

```
@function set-text-color($color) {  
  @return if (lightness($color) > 40, black, white);  
}
```

... and it was not working. Can you guess why? ... well apparently you cannot have space after the 'if'... who would have guessed haha — just for clarifying, here's the working version:

```
@function set-text-color($color) {  
  @return if(lightness($color) > 40, black, white);  
}
```

... the reason for this from how I understood it, is that if there's a space after the 'if', then it is a function call. If there is no space afterward, then it assigns a value. In the original version it

worked, since our if and else statements goal was to go into their respective scope depending on the \$color value. So we wanted space there. But now we want to return a value, so we cannot use it as a function — we need to actually get value back from it, that's why the space needs to be removed.

6. Work and Contact Pages

05.05.2023

We started doing our work/projects page. The first new thing for me was the `@extends` functionality. I googled around how it differs from `@include`. I came into the conclusion that it's probably better to just use `@include`, since there will be fewer errors (things compiling differently than you expected). There's also some limitations in `@extends` (which you can read from <https://sass-lang.com/documentation/at-rules/extend#limitations>). The main advantage of `@extends` seems to be smaller compiled css size, but since your css files will be anyway compressed (at least you should compress them) when hosting a site, the smaller compiled filesize advantage from `@extends` is no longer there. So I will most likely use `@include` instead of `@extends` in my own project, but I will stick with the `@extends` with this tutorial site. In fact, if you read the sass `@extends` page about the `@extends` or `@mixin`, it says: "So choose the feature that makes the most sense for your use-case, not the one that generates the least CSS!" (<https://sass-lang.com/documentation/at-rules/extend#extends-or-mixins>)

Next we did the contact me page. This time we used flex. Flex is also familiar to me since when I was learning about the grid, I also learned about flex. Similar story to grid, I learned flex from <https://flexboxfroggy.com/> browser game. Again, very fun way to learn.

After finishing the video, I did the usual - checked the comments. There was one very useful tip from idlevandal69:

idlevandal69 4 years ago

For grid-template-columns in .projects I used:

```
grid-template-columns: repeat(auto-fit, minmax(270px, 1fr));
```

That way way I didn't have to write any media queries. Thanks Brad, this is a great series.



7



Reply

... with that our code base is much cleaner and way easier to resize the boxes when so desired.

I also included the fixes that the video maker said in the comment section:



Pinned by Traversy Media

Traversy Media 4 years ago

Hey guys, Couple things to add to the "mediaSm" query in order for mobile screens to look right... in _mobile.scss. Make the .main padding to 2rem and set the font size of the .lg-heading to 5rem. Should look like this...

```
@include mediaSm {  
  main {  
    padding: 2rem;  
    #home h1 {  
      margin-top: 10vh;  
    }  
  
    .lg-heading {  
      margin-top: 1rem;  
      font-size: 5rem;  
    }  
  }  
  
  .projects {  
    grid-template-columns: 1fr;  
  }  
}
```

Show less



52



Reply

7. Website Deployment

06.05.2023

Deploying the site to GitHub pages was fairly simple. You can either use the npm package named gh-pages, or you can manually set the GitHub repository to contain the site. With the gh-pages it can be done as easily as telling the site name, and where the source folder for the site is, and running the gh-pages command. If you want to do it from the GitHub website, it is also simple. First you open the repository, then you go to the repository settings, and select the 'Pages' category. Then you just set the Source to be 'Deploy from a branch', and select which branch you want to use. Note that with this method you have to manually send the source files to the root of the git repository.

8. Final Project

10.05.2023

For my own project, I decided to create a shop page. I wanted to do this since I might want to create online shop later on my career, and I know that when you do something second time, it's always better since you know what did not work out in the first round. This will be a dummy shop, so everything that is on sale will be cats.

I wanted to make 3 different kinds of views, so I can see what different kinds of shop interfaces would look like. I decided to use grid, card and list layouts.

The grid view was fairly simple, as we used it in our example site. I of course had to modify the scss so my page can look how I envisioned it.

The card view was interesting to make. I wanted there to be a picture on the left, with product information on the right. And it of course needed to be responsive, so the information block should go under the image when viewport decreases to certain size.

List view was most simple one. As the name suggests, I wanted just to list the most important details of each product. Image on the left, name on in the middle and price on the right.

Lastly I wanted a simple view for telling our shop story. I used the hamburger menu for that.