Lappeenrannan teknillinen yliopisto
School of Engineering Scienc

Sofware Development Skills

Pyry Santahuhta, 0545254

LEARNING DIARY, FRONT-END MODULE

LEARNING DIARY

You can check each part's status from when it was concluded in the courseworks/part # folder in github. The Project folder contains the most up to date version of the portfolio.

PART 1

07.05.2021

I started the course today. I started by going through the course Moodle page and grasping the core contents of the course. The course seems very intriguing since I've already dabbled a bit in HTML and CSS by creating my own homepage for my browser.

I started the coursework by creating the requisite folders on my computer and by creating a git repository for the course and testing it out with some test commits. Then I formatted the course materials into the project and part sections. I also chose VS Code since I've already used it beforehand as my editor and tried pushing straight from there worked well. VS Code just pushes to "master" branch instead of the default "main" on the repository, but these can just be merged and isn't much of a hassle. I started watching the first tutorial video and also downloaded node.js. During the tutorial I was amazed by all the #test or ! tab snippet completions and thought that these will be very handy in the project. Setting up the live server was really easy as well and a really nice way to see your code updates in real time.

I also created the package.json with npm init and downloaded node-sass to get my sass workflow started. Creating a sass script for compiling the scss files into css files was pretty simple and I got it working after one hiccup with the destination folder. The power of sass will probably come in later when the stylesheet gets more and more complex, so it is more easily maintained in the scss file. Ultimately this concludes the part 1 video, and I understood the usability of the sass workflow well. I've also used responsive design in my own things with @ media tags, but I'm intrigued to see how it is implemented in this course's tutorial since it seemed very usable.

PART 2

10.05.2021

Today I did part 2 of the tutorial. The tutorial consisted of creating the homepage for the portfolio. During the designing of the homepage, I learned about nesting in Sass. The act of making css feel more like code with nesting feels very intuitive and it makes the logic much easier to understand. I also learnt to use rem values in font sizes and, how to use external css styles for different purposes, here to get the logos from fontawesome.com.
Variable usage will take a while to get used to since the declaration is a bit weird always using the $ symbol, but I already understood how it is very good for keeping the sites style intact and not having to redefine the same thing many times, when it is stored in a variable. In understanding the basic principles of sass, this site was very useful: https://sass-lang.com/guide.
I also moved my mixin's and variable declarations to a 'Partial' file to make the scss files even more compact. I created a js file for future js work, and added images to the site. In adding the background image and stylizing the page, rgba() function was very useful in changing the opacity of different things. I also learned a bit about css "Animations" with transitioning on hover.
I diverged from the tutorial portfolio a bit already and I think that is part of the fun. Creating your own twists and signature look in the homepage is essential to me. For creating my own color scheme I used Coolors.co to create a nice colour palette.
After part 2, I am really into the project and want to continue, so I think I will work on part 3 today as well.

PART 3

10.05.2021

In tutorial three, work on the overlay begins. I started out by creating the necessary Javascript to add the classes to the menu divs. It was pretty straight forward, just getting the DOM items into variables with queryselector, listen for click with the menubutton and finally when the button is pressed add or remove the overlay class from the DOM items.
After that I created an scss menu file, where I created the menu button, which is just three lines to resemble a hamburger menu. These three lines are "animated" with pure css to turn into a cross when the overlay is opened to signal a close button instead. I learnt that you can specify a unique div of the same type with nth-child. This was needed to make the hamburger menu animation. Next up in the tutorial will probably be creating rest of the overlay which seems easy enough to do from this point with the Javascript already ready.

PART 4

11.5.2021

Tutorial 4 consisted of adding the menu overlay and changing the homepage and overlay to be responsive.
The menu overlay was created into the menu.scss file. I added the required css for the brand and navigation menus respectively, which was simple to implement. For both area's background I used primary color, which was darkened for the nav side for a clean look. To create the "animation", I used translate3d to move the panels from the bottom up and top to bottom respectively. On the branding side I added a portrait photo of myself with a simple background url change, and made it into a circle with outlines using border-radius: 50% and border color. The individual navigation items were also animated to go from right to left, which was done similarly only this time using certain pixel amounts instead of percentages. In order to slide them one by one a scss for loop was used. It worked similarly to js/c so it was already clear to me.

While creating the animations I fiddled around with the translate3d a bit and it was intuitive to use. With translate3d or translate x/y and different transition styles I'm confident I could create a lot of different animations for different purposes. I also added a different styling to the current page in the navigation by changing it's color. This was done with adding a class 'current' to the index.html. I imagine with different pages I will just add 'current' to the desired page. A hover effect was also added to improve user experience.

Speaking of user experience, this tutorial also contained the media query responsiveness. As I've already said, I have used @media tags before in my css hobbyprojects but with scss it was really elegant to do. I replaced the @media tags with scss mixins, I created four different sizes to style the site with. These were added to the config files. I created a partial to keep the responsiveness css away from the main scss in order to keep it clean. In the partial I just changed a few font sizes and image sizes to fit in smaller devices/monitors. For the medium to small I also changed the overlay to be vertical instead of side by side. I changed the ratio from 50/50 to 25/75 so the navigation items have more room.
This was the point where I really got the power of using a preprocessor for css. Media tags always added so much bloat to the stylesheet and this was much clearer to understand.

PART 5

14.5.2021
I started the day off with commenting some of my old code since I noticed I had not explained everything. This is something I need to work on in my mindset to always comment on pieces of code after they are done. In the part 5 tutorial the first thing that I did was changing the text color to be responsive to the background color. This was done with a simple scss function where I check the background color's lightness with an if statement. I also changed the hamburger menu and the social media icons to work this way as well.
Next up I started working on the second actual page of the portfolio, the about me page. I started off by copying the index.html because all the pages have a lot in

common as the menu and the overlay is the same everywhere. I then added the necessary html and some lorem ipsum to substitute for actual content. The html just consisted of some paragraphs and headings. Next up is the css to make this look nicer. Grid is used for the layout of this page. With grid we can use grid-template-areas to set the layout just as we want it very intuitively, since the code matches the shape of the actual page's layout. To size the grids columns, a fr or a "fraction" unit is used. This is nice since it automatically cuts the remaining area into that many fractions, if for example I used one bigger column with a specific size and multiple smaller columns I could cut them automatically into same sized pieces with fr.
Then I just had to specify every sections grid-area, I also added a little background color and left border style to the three text paragraphs. In order to get the footer, stick to the bottom of the page, I gave it a specific height, and changed the main height to be min height instead, calculated with calc 100vh (The whole page) minus the specific height of the footer, so the main area is still 100% just minus the footer. Now all that is left is the responsiveness of this page. This was very simple with the grid-templat-areas, since I could just add a new grid template into the mobile file where I stacked all of the grid items on top of each other. This concluded the tutorial 5.

PART 6

17.5.2021

Today I started the part 6 of the tutorial. First I began working on the projects.html page. On this page I collected some of my own projects into a grid like structure just like on the about page. I added my own hover effect to the images of the projects, so that when the images are hovered, a short description comes in and explains about the project. This was done by wrapping the image and the description in the same div, and then using sass to create the hover effect by nesting the different effects under the wraps hover.
I also added buttons which had similar icons to the main page and hover effects to change in color. All of these features were not new so It was just recap on what I've already learned. I felt much more confident using sass this exercise since I've gotten accustomed to it, and using it comes more naturally now.
Styling the responsiveness for the projects page was simple with grid structure, I just added how many columns there should be for each size, 1 for mobile and 4 for xl screens for example.
Next up was creating the contact page, which holds the least amount of information. For this page, the flexbox structure is chosen. Working with flexbox is easy since it scales automatically depending on the website size, so almost no responsiveness is needed for this page aside from a few font size adjustments. I just created three divs containing different info and added an icon and a hover effect to them, nothing ground-breaking. Flex-wrap and justify-content are used to determine how the divs collapse on one another and how far they are from eachother. While training my flexbox skills I discovered a great site to learn from: https://flexboxfroggy.com, here you just adjust the flexbox properties until you align the frogs on the lily pads. That concluded the work on part 6, next up is deployment, which will be entirely new to me.

PART 7

18.5.2021

Today I did part 7 of the tutorial. This consisted of deploying the site online using Github pages, which was a lot simpler than what I had imagined. As I had already created the repository, I did not have to work with Github on that end. I downloaded gh-pages with npm so I could deploy the site straight from vscode with a script. After that I created a script just like I did for sass which just called gh-pages with the -d flag and the folder which it had to deploy, which for me was project/dist. And that's it! My site was online.
After seeing the site online I decided to check it out on mobile as well to check the responsiveness. I found a few font size issues which I also fixed after watching the tutorial to the end. I was also happy with the domain so I didn't bother buying my own.
That concludes the tutorials, I might work on the about me page for the project a bit but overall I'm happy with the portfolio!