Lappeenranta-Lahti University of Technology

LUT School of Engineering Science

Sofware Development Skills

Jani Heinikoski, 541122

LEARNING DIARY, FRONT END MODULE

**LEARNING DIARY**

**02.03.2021**

I read the information presented in the course Moodle -page and learnt that this course is arranged very similarly compared to the Mobile module (which I have already passed). I have very little experience in the front-end stack at this moment; I am only familiar with HTML and basics of CSS. I checked out the course’s primary book on HTML5 and CSS3 (I read the first ten pages) and learnt a bit about the history of HTML and CSS.

I started the coursework by watching the first video (introduction to workflow and sass). I learnt about the “coursework project” (the portfolio website which is done during the module task list). I already had VS Code, git (and GitHub Desktop) and Node.js preinstalled so the environment setup was straight forward (only had to install the VS Code extensions and sass with the node package manager).

I had previously used the live server and prettier extensions (on another machine) for the VS Code -editor so they were nothing new/surprising. I learnt that the package.json file contains the node dependencies and other project related metadata handled by the node package manager NPM. Furthermore about the Node-sass library, I learnt that it is a preprocessor (written in C) that is able to compile .scss -files into CSS with great performance.

03.03.2021

I started the second part of the course work task list. I copied the part\_1 content to folder part\_2 (easier for the grader of my course work to check different parts). I followed the first five minutes basically copying the modifications to the index.html -file. I did not learn anything new there since I am already familiar with common HTML5 tags. Next up I learned about the Font Awesome website and how to use their styling services (fonts, icons etc.).

I learned how to select all elements in CSS with the asterisk symbol \* and used it to set all elements’ box-sizing attribute to border-box. I had not heard from the box-sizing attribute before, so I read about it from the MDN Web Docs:

“The box-sizing CSS property sets how the total width and height of an element is calculated.” (MDN Web Docs – box sizing, 2021)

“border-box tells the browser to account for any border and padding in the values you specify for an element's width and height. If you set an element's width to 100 pixels, that 100 pixels will include any border or padding you added, and the content box will shrink to absorb that extra width. This typically makes it much easier to size elements.” (MDN Web Docs – box sizing, 2021)

I learnt how you can use nesting in sass to modify multiple elements with same classes:



In the above code snippet, with the ampersand symbol & I can refer to all h1, h2 and h3 elements which have a class “lg-heading” and change their font-size to 6 times the HTML defined font size.

I learnt about the sass lighten -function which can be used to create a lighter background compared to the current background color. I also learnt that the rgba -function can make the background transparent/opaque by adjusting the opacity argument. I learnt that you could use the text-decoration CSS -attribute to apply decorations (e.g., underline, line-through etc.) to text.

I learnt that the position property can be set to fixed to position an element relative to the browser window (in this case the header stays always at the top). Z-index property can be used to “elevate” elements (i.e., tells how close to the viewer the element is positioned).

I learnt that you can define properties for certain events such as the hover event with <tag>:<event> {//attrs} (e.g., a:hover {color: #fff;}). In sass you may also use the ampersand again for the nesting.

I learnt about transitions and mixins in CSS/Sass which kind of work the same as animations in native Android development. Also, I learnt a new measurement unit called viewport height (vh) and how to divide scss files into multiple ones (main and partials).

I copied all the part 2 content into the part 3 folder, committed it and started watching the video of the part 3.

During the first 14 minutes of the part 3 video, I learnt how to reference DOM -elements from JavaScript using the document class querySelector -function and how to add event listeners using addEventListener -function. Furthermore, I checked out the MDN Web Docs about the NodeList class where I found out the difference of a static and a live NodeList object (querySelectorAll returns a static NodeList i.e., DOM updates do not update automatically). (MDN Web Docs – NodeList, 2021)

The remainder of the video focused on creating the transformation effect where the hamburger menu icon turns into an X which was fairly trivial using the nth-child pseudo class and translate & rotate CSS-functions.

Once again, I copied the part 3 content to part 4 folder and proceeded to watch the fourth video Menu Overlay & Responsiveness. First, I learnt that you can refer to all classes that end with -branding by using again the ampersand symbol (e.g., &-branding matches to <anything>-branding). I learnt that the display property in CSS specifies how an element should be displayed.

I learnt about the inline-block display property and relative position property. The first one is basically a block without newlines and latter means that the position of the element respects the top, left, right and bottom properties to position itself relative to its normal static position.

I learnt how to do a for loop using sass and how to use the translate3d function to move elements. Finally, I learnt how to create a responsive website using the media query mixins with sass to have customized styling based on the screen size.

04.03.2021

I (once again) copied the part 4 content to part 5. I noticed that the fifth part covers the CSS Grid component which (based on the name) sounds like a good component to create a 2D-matrix. I read the “A Complete Guide to Grid” -article about it first because I wanted to get a more comprehensive understanding about it for the project (C. House, 2020). I thought about creating a visual sudoku solver for the project. For the sudoku matrix CSS Grid seems to be the best option.

After reading the article, I learnt that indeed CSS Grid is used to make a two-dimensional grid layout. It requires a container element on which the display: grid is applied. Grid items are the container’s children, grid lines are the dividing lines which make up the structure of the grid, grid cell is the space between two adjacent grid row and column lines, grid track is the space between two adjacent grid lines and finally grid area is the space between four grid lines. (C. House, 2020)

I proceeded to watch the fifth part video material. First, I learnt how to create functions using SCSS (with the @function pre-processor directive). Next, I learnt how to create a sticky footer using the calc -function. The fifth video did not teach much new things since I read the article about CSS Grid beforehand. I proceeded by copying the part five content to part six folder and started watching the sixth video.

In the sixth video, the first 15 minutes focused on the work-page and making it responsive. During the first 15 minutes, I learnt that inheriting properties in SCSS is possible with the @extend <class name> pre-processor directive. Finally, I learnt how to use the flexbox display property which in this case wraps the children if they do not fit adjacently horizontally in the viewport.

After watching the seventh (and last) part of the video, I learnt how to publish my websites using the GitHub Pages. This was not as easy to do as it was shown in the part 7 video in my case (due to Windows 10 being my OS). The deploy script did not work because git was not in my PATH -environment variable. I found the git.exe from the following path: “C:\Users\Jani\AppData\Local\GitHubDesktop\app-2.6.4\resources\app\git\cmd”. After adding it to my PATH -environment variable, the deploy script worked.

**Note,** that I have moved the part\_6 content to a whole new repository called modern\_portfolio because I did **not** want the URL to be <https://jani-heinikoski.github.io/sds_front_end/>. I did a few updates including the installation of the gh-pages Node package and updated the package.json file in the new repository (i.e., completed the seventh part in the other repository). Then I copied back all the modern\_portfolio content back to this repository in the coursework/part\_7 folder so all of the parts can be inspected in this repository alone. The website can be accessed from the link: <https://jani-heinikoski.github.io/modern_portfolio/>. Repository where the part 7 (same as coursework/part\_7 content) can be accessed from <https://github.com/jani-heinikoski/modern_portfolio>.

04.04.2021

I learned that you could set the min-width – CSS-property to fit-content to prevent child elements from overflowing. I used it to solve a problem where the side menu’s a-elements were overflowing when reducing the viewport width.

After watching Kevin Powell’s video on YouTube about Flexbox design patterns, I learnt to better utilize flexbox in the project. I used to create a quick access link at the homepage to the focus of the project (the sudoku solver itself). (Kevin Powell, 2020)

12.06.2021

I learned how to disable arrows on hover from the input HTML-elements (W3Schools, 2021). I used the input elements for the project’s sudoku grid and I did not want increment/decrement arrows in the elements.

For some reason, I could not get the CSS ID selector with # to work with the input elements in the sudoku game grid. After intensive Googling, I found out that it is equivalent to use #my\_id and [id=”my\_id”] from which the latter worked fine. (MDN Web Docs – ID selectors, 2021)

28.06.2021

I found out that there is some weird scoping behaviour in JavaScript through the hard way. After debugging my sudoku solver for half an hour, I recalled that I have had seen somewhere a quote: “JavaScript has some serious scoping issues”. I declared a for-loop inside a function as follows . Inside the for-loop I called another function which also had a similar for-loop with the same index variable . What I did not know is that JavaScript hoists variable declarations at the top of the document if they are declared as mentioned before. After watching few videos, I learned that you need to use the let keyword inside the for-loop brackets if you want the index variable to live inside the for-loop’s block scope. (Dani Krossing, 2018)

**Reference list**

* MDN Web Docs. 2021. Resources for developers, by developers. [Cited 03.03.2021]. Available: <https://developer.mozilla.org/en-US/>
* Chris House. Published 2016. Updated 2020. A Complete Guide to Grid. CSS-Tricks. [Cited 04.03.2021]. Available: <https://css-tricks.com/snippets/css/complete-guide-grid/>
* Kevin Powell. Published 2020. Flexbox design patterns you can use in your projects. YouTube. [Cited 04.03.2021]. Available: <https://youtu.be/vQAvjof1oe4>
* W3Schools. 2021. How TO - Hide Arrows From Input Number. [Cited 12.06.2021]. Available: <https://www.w3schools.com/howto/howto_css_hide_arrow_number.asp>
* Dani Krossing. Published 2018. 16: JavaScript Scopes | Local Scope and Global Scope in JavaScript | JavaScript Tutorial | mmtuts. [Cited 28.06.2021]. Available: <https://www.youtube.com/watch?v=hTU1OSbnov8>
* Dani Krossing. Published 2018. 17: Let and Const Variables | JavaScript Block Scope and Function Scope | JavaScript Tutorial. [Cited 28.06.2021]. Available: https://www.youtube.com/watch?v=aK\_nuUAdr8E
* Dani Krossing. Published 2018. 18: Hoisting in JavaScript Explained | What is Hoisting in JavaScript | JavaScript Tutorial. [Cited 28.06.2021]. Available: <https://www.youtube.com/watch?v=ppMlvGMT2qE>