Lappeenrannan teknillinen yliopisto
School of Business and Management

Sofware Development Skills

<Davann Tem>, < x100913 >

LEARNING DIARY, Front-End MODULE

9.3.2021

1. Introduction to workflow and sass

In this 1st lesson, I have installed Visual Studio Code for code editor, GIT Bash for command line and Node Js for note packages. I also created an account in Bitbucket and a public repository there. I haven't linked it yet with my project. I tried to follow the video and create the same project with header: Hello! and grey background. Until there was a problem when it didn't run node sass (Syntactically Awesome Style Sheets). However, after reboot the app, it finally could run the project well. I find this course very interesting because I like designing pages and study this front-end allows me to be able to personalize project much better compared to using existing web creators.

15.3.2021

2. Homepage and Core Sass/CSS

Today lesson was about Homepage and Core Sass/CSS. I continued to create the simple website by following the video tutorial. I learnt how to create divisions (div) under the header. Then I learnt how to create menu and divide it into two different sections (menu-branding & menu-nav). Menu-branding is the part that I can put logo/picture while menu-nav is where I can make drop down list (ul: unorder list, li: list element, a: for href). Main

(main id) is when creating the title. I can download icons (social media) from fontawesome.com. Sometimes the page didn't update, but I found out that I have to type npm run sass in terminal to reconnect it.

16.3.2021

Today I started to work on css part. I know how to set font type, size & color, background color and line height. I know how to make the menu always stay in the same place top right corner) by *position: fixed*. To adjust the space around the font, we use *padding: 4rem*. Using *&:hover* to change the effects of icon when the cursor is on it, for example, change color & transition speed. There was a problem when the page didn't update the background picture. I found out the problem was because I didn't have *id="bg-img"* in html file. In the video the tutor forgot to mention when he added it. An easier way to blur the background image is to add *&:after{content:''}* and to adjust the level of the blur, we modify it with $background-capacity. The last thing that I learnt from the video was that I can make scss look clean and easy to review in different by cutting some parts in scss and paste it in the new created file under scss folder. To avoid the error of missing codes after moving, I have to write *@import 'file name'* in the main scss to reconnect it.

22.3.2021

3. Rotating Menu Button

In the case study, when click on the menu button, the menu shows up and if click again, the menu disappears. This is done with Js. We need a function: addEventListener('click' toggleMenu to make it performs when click. Then set function *if = false* to make the menu show up and later set as *true* to make it disappear. This has to create for each button's lines. In this case, we have 3 lines.

Following the video tutorial, we want to make the menu button rotate 180 degree. Therefore, we use function transform: rotate(180deg) to turn the whole button 180 degree. We also want after that the button turns to X when the menu shows. So, it requires to set for each line separately. For 1st line, we write:

transform: rotate(45deg) translate(5px, 5px) to make it rotate like a forward slash, 3rd line like a backward slash, and 2nd line to disappear. More modification related to the shape can be found in css transform in https://developer.mozilla.org.

From 3-lines-menu icon;

To X when click on it & menu shows

23.3.2021

4. Menu Overlay & Responsiveness

Today I worked on Menu Overlay. I modified the class menu with position, top (how far the box's top margin edge is), width, opacity (transparency), and then visibility (which is hidden, so it doesn't show in home page). Under the class menu, the class show (&.show) is set to be visible. Then for menu branding and nav are set together in one place. For this, use function

- *display: flex* (to make the columns)
- *flex-flow: column wrap* (to make the items wrap and flow vertically within the screen)
- *align-items: center & justify-content: center* (to make the items stay in the center of the element vertically & horizontally)
- *float: left* (so items float on the left side on top of the home page's center)
- *width & height* (to modify the element's size)
- *overflow: hidden* (to hide the scrolling bar)

24.3.2021

I learnt how to make the menu nav's color get darker with the function *background: darken ($primary-color, 5),* to remove the bulletins from the items with *list-style: none,* and to make the menu slide from top with *transform: translate3d (0,100%,0)* (more info can be found on translate3d in https://developer.mozilla.org. For menu branding, I used also translated3d to make it slide upward from the bottom *(0,-100%,0).* Then add the portrait, resize it with 250px with both width and height, make the portrait frame round with function *border-radius:50%,* and last make the border line thicker with the secondary color that we set using *border:solid*

3px $secondary-color. Then to make the menu nav item slide from right, we use same *translate3d* function with *(600px,0,0).* To change the color of Home link only in menu nav to secondary color, make sure a class is added into list element of Home in html. In this case, we added *current.* So, in css, *we type &.current > a {color: $secondary-color;}.* The last thing which is quite new to me was how to make the menu nav item slide in order smoothly with 0.1s delay. The syntax for it is:

```
@for $x from 1 through 4 {
.nav-item:nth-child(#{$x}) {
transition-delay: $x * 0.1s;
```

26.3.2021

In this section I learnt how to make the items conform with the different screen sizes such as on wide screens (Xlarge), laptops/desktops (large), tablets/small laptops (medium) and smartphone (small). So, in config file, add *@mixin media* for the 4 screen sizes: small (max-width: 500px), medium (max: 768px), large (min:769px-max:1170px) & Xlarge (min:1171px). Then create a new file for this and name it as mobile.scss. In mobile file, we define the format of the items. Large and Xlarge screens don't have problem or don't need to change anything because when the screen is wide the items such as texts or pictures will be fully viewed, but they still need to be included in mobile file. We can just define the small and especially medium size screen since medium size also covers the small size. In medium, we define float, width, height, font size and transformation. Because we want the menu nav slide from left and menu branding slide from right, then translated3d for menu nav is (–100%,0,0) and for menu branding is (100%,0,0). In menu branding, the photo is so big for the screen, so we need to make the photo smaller by adding class portrait under the class menu branding with this syntax:

```
background: url('../img/portrait_small.jpg');
width: 150px;
height: 150px;
```

Preview of: large & Xlarge screen size; Medium & small screen size

31.3.2021

5. Page with CSS Grid

If we change the main color which is original grey to white or any light color, the background will become white and affect the readability because the text color is also white. Therefore, to avoid this problem, we want the text to change to black color when the main color is light color. To do this, we create *@function set-text-color ($color) {@if (lightness ($color) > 40) {@return #000;} else{ @return #fff;}* . Then in main scss, we replace the color setting with set-text-color($primary color) where we want the function to apply to. In this case, we use this with the color of text, menu button line and nav link texts.

Page view when the primary color is in light color (in this case, white)

3.4.2021

A main goal of this lesson is to make pages for nav links, and on the pages, we can make grids containing texts and pictures etc. I created an html file with the name about.html for About link. Basically, I just copied the index file to about. Then I changed some places such as:

- Remove Current from list element of Home and add back to list element of About, because currently we open the About page.
- Change main id to ‘about’ then rewrite the texts in h1 & h2 an delete the logos settings there because we don’t want them anymore in About page.

To add footer: *<footer id=“main-footer”> Copyright © 2021.*

Then create class for About info:

- add the photo: *<img src=”img/portrait.jpg” alt=”John Doe” class=”bio-image”>*
- Create more classes for bio, job 1, job 2 and job 3. Then add each class the header (h3), body text (h6) and description (p).

After that go to main scss, write there at the .about-info with these settings:

display: grid; (for grid style)

grid-gap: 30px; (to make space between text boxes)

grid-template-areas:

'bioimage bio bio' (two bios here because it needs 3 columns, which will merge later)

'job 1 job2 job3';

grid-template-columns: repeat(3, 1fr); (to create 3 columns in one row)

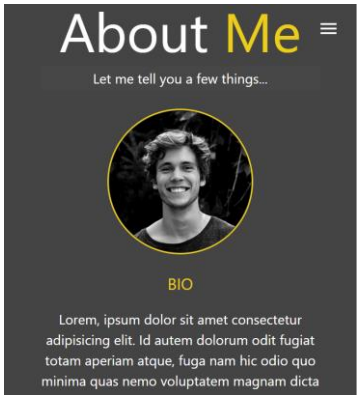
Then set each element to the right grid area and also setup the formats for each.

View of About Page

6.4.2021

If the columns of jobs are deleted, the footer of Copy Right will move up. So, to make the footer stick at the bottom, in main.scss replace *height:100%* with *min-height: calc(100vh - 60px).*

If the screen size is small, the texts in About page will not be fully shown. Hence, to make all the elements in About page show in one grid area vertically, in mobile.scss under MediaMd, we add the function: *grid-template-areas: 'bioimage' 'bio' 'job1' 'job2' 'job3'* (grid areas should be listed in or is small, the texts in About page will not be fully shown. So, to make all the elements in About page show in one grid area vertically, in mobile.scss under MediaMd, we add the function: *grid-template-areas: 'bioimage' 'bio' 'job1' 'job2' 'job3'* (grid areas should be listed in order), and then *grid-template-columns: 1fr* (to make them all listed in one column/center of the page).



About Page in medium & small size

8.4.2021

6. Work and Contact Pages

In this lesson, we learn about how to create Work and Contact pages. To create Work page, the first step is to copy all html codes from About.html, and paste it to a new created work.html file. Then modify the codes accordingly. The Work page that we want looks like below:

The final look of Work page

Therefore, to have one picture with project & github link buttons under it for each project, we add html codes for each project as following screenshot.

```
<div class="projects">
    <div class="item">
        <a href="#!">
            <img src="img/projects/project1.jpg" alt="Project">
        </a>
        <a href="#" class="btn-ligh">
            <i class="fas fa-eye"></i> Project
        </a>
        <a href="#" class="btn-dark">
            <i class="fab fa-github"></i> Github
        </a>
```

Html codes for a picture & 2 link buttons

After finish with Work.html, go to main.scss and add there the codes as following to set grid layout, pictures' formats as well as the effects when mouse is on it:

```
// Work/Projects
.projects {
    display: grid;
    grid-gap: 0.7rem;
    grid-template-columns: repeat(3, 1fr);

    img {
        width: 100%;
        border: 3px #fff solid;

        &:hover {
            opacity: 0.5;
            border-color: $secondary-color;
            @include easeOut;

        }
    }
```

Scss codes for Work page

Next is about the setting of buttons for projects and Github link buttons that are placed under each project's picture. One thing that should be remembered is the *display: block* which

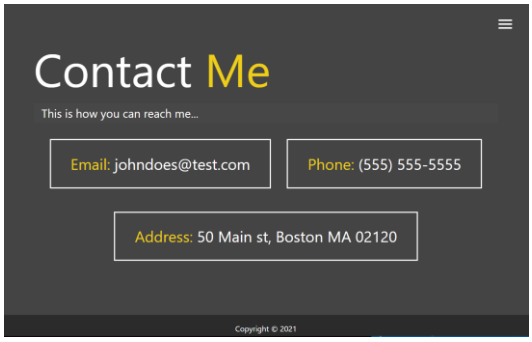
makes the buttons spread across on their own individual line. Also set the *&:hover* that changes the color of the buttons to secondary color when cursor on it. *Padding: 0.5rem 1rem* is to set the space on top-bottom and left-right of the buttons. Then create separately the setting for class btn-dark and btn-light. Each include *@extend .btn* because we want to use the btn setting in them as .btn alone is not used.

```scss
.btn {
    display: block;
    padding: 0.5rem 1rem;
    border: 0;
    margin-bottom: 0.3rem;
    @include easeOut;
    &:hover {
        background: $secondary-color;
        color: set-text-color($secondary-color);
    }

}

.btn-dark {
    @extend .btn;
    background: darken($primary-color, 50);
    color: #fff;
}

.btn-light {
    @extend .btn;
    background: lighten($primary-color, 50);
    color: #333;
}

#main-footer {
    text-align: center;
    padding: 1rem;
    background: darken($primary-color, 10);
    color: set-text-color($primary-color);
    height: 60px;
}
```

Scss codes for Project & Github buttons

We want that when the screen size is Xlarge, one grid area contains 4 columns; Large contains 3 columns, Meduim contains 2 columns and Small contain only 1 column. To do this, we go to mobile.scss file and under each screen setting, add class project with function: grid template columns. For example, Xlarge screen is: *.projects {grid-template-columns: repeat(4, 1fr);}*

What I have learnt next is how to create Contact page. Similar to previous page, an easy way is to copy all codes from about.html file, and paste it to new created contact.html. Then modify the headings and edit codes accordingly. We aim to have the contact page which contains 3 boxes for email, phone and address that looks as below:



Contact page's final look

Thus, in the contact.html, we add class 'boxes', which contain span elements for the 3-contact information.

```
<div class="boxes">
    <div>
        <span class="text-secondary">Email:
        </span> johndoes@test.com
    </div>
    <div>
        <span class="text-secondary">Phone:
        </span> (555) 555-5555
    </div>
    <div>
        <span class="text-secondary">Address:
        </span> 50 Main st, Boston MA 02120
    </div>
</div>
```

Html codes for contact information

For layout and page responsiveness, we work on the main.scss. There, we use flex display, flex wrap and space evenly for justify-content, which make the info boxes stacked well in lines. We also set *&:hove* that makes the boxes change in size when we move cursor on it.

```
// Contact Page
.boxes {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-evenly;
    align-items: center;
    margin-top: 1rem;

    div {
        font-size: 2rem;
        border: 3px #fff solid;
        padding: 1.5rem 2.5rem;
        margin-bottom: 3rem;
        @include easeOut;
        &:hover {
            padding: 0.5rem 1.5rem;
            background: $secondary-color;
            color: set-text-color($secondary-color);
            span {
                color:set-text-color($secondary-color);
            }
        }
    }
}
```

Scss codes for Contact page

9.4.2021

7. Website Deployment

I learn how to upload my project into git repository in my Github & bitbucket accounts that I have created. These are how I do it in terminal:

- Change directory to where I save my project (cd Desktop/Front-End)
- Initialize local git repository (git init)
- Add all the files to the local repository (git add --all)
- Do the commit to local repositor to save the change (git commit -m "*Write comment here*"). When run, it requires user's email address and name.
- Connect to remote repository by pasting the remote address into terminal (git remote add origin [*link address]*)
- Push the local repository to remote one (git push -u origin master). To push the update to the last remote repository, write (git push).

Finally, to deploy the website, run *npm i gh-pages* in terminal. Then go to package.json and add there url to project homepage and deploy scripts as below. Lastly, type "*npm run deploy"* in terminal.