LUT University

Software Development Skills Front-End, Online Course

Name: Niroj Acharya

Student number:1620492

**LEARNING DIARY:**

**1. Setting up VS Code Environment**

**Day 1: Introduction to VS Code**

**Date: 2024/01/04**

Today, I embarked on the journey of setting up my coding environment using Visual Studio Code(VS Code). I started by downloading and installing VS Code from the official website. The installation process was straightforward, and I appreciated the user-friendly interface. I explored the basic features and familiarized myself with the layout of the editor.

**Key Takeaways:**

* Installed VS Code.
* Explored the basic features and interface.

**Day 2: Extensions and Themes**

**Date: 2024/01/05**

I delved into the extensive world of VS Code extensions and themes today. I learned how these add-ons can enhance productivity and customize the appearance of the editor. After some research, I installed popular extensions like ”Live Server” for web development and experimented with different themes to find one that suited my preference.

**Key Takeaways:**

* Explored and installed VS Code extensions.
* Customized the editor with themes.

**Day 3: Version Control with Git**

**Date: 2024/01/06**

Understanding the importance of version control, I dedicated today to integrating Git with VS Code. I learned how to initialized a Git repository, commit changes, and push them to a remote repository. The built-in Git features in VS Code made this process smooth, and I also explored the GitLens extension for additional functionality.

**Key Takeaways:**

* Intergrated Git with VS Code.
* Explored basic version control operations.

**Day 4: Configuring Code Formatting**

**Date: 2024/01/07**

Today, I focused on maintaining clean and consistent code by configuring code formatting in VS Code. I explored settingd like indentation, line length, and code styling. The Prettier extension proved to be a valuable tool for automated code formatting, ensuring my code follows best practices.

**Key Takeaways:**

* Configured code formatting in VS Code.
* Utilized Prettier for automated code formatting.

**Day 5: Debugging and Troubleshooting**

**Date: 2024/01/08**

Understanding the importance of debugging, I dedicated today to learning how to set up and use the debugger in VS Code. I explored breakpoints, watched variables, and stepped through code to identify and fix issues. Additionally, I researched common error messages and how to troubleshoot them effectively.

**Key Takeaways:**

* Set up and used the debugger in VS Code.
* Explored common debugging techniques.

**Reflection:**

Completing the setup of my coding environment in VS Code has been a rewarding experience. I now have a well-configured editor with extensions and tools that enhance my productivity. Learning the basics of version control, code formatting, and debugging has equipped me with essential skills for efficient and effective coding. As I move forward, I look forward to exploring more advanced features and continually optimizing my development environment.

**2. Learning SASS(Syntactically Awesome Stylesheets)**

**Day 6: Introduction to SASS**

**Date: 2024/01/11**

Today marks the beginning of my journey into the world of SASS. I started by understanding the basics, learning that SASS is a preprocessor scripting language that is interpreted or complied into CSS. I grasped the concept of variables, nesting, and partials, which seem to offer a more organized and modular approach to styling.

**Key Takeaways:**

* Learned the basics of SASS as a CSS preprocessor.
* Discovered the advantages of variables, nesting, and partials.

**Day 7: Installation and Setup**

**Date: 2024/01/12**

To practically implement SASS, I needed to set it up in my project. I spent today installing SASS and configuring it for use. I realised that SASS can be complied using a command-line tool or integrated directly into my build process. I choose a method that suits my project requirements.

**Key Takeaways:**

* Installed and configured SASS for my project.
* Explored different methods for compling SASS.

**Day 8: Variables and Nesting**

**Date: 2024/01/14**

Today, I delved deeeper into the power of SASS variables and nesting. I appereciated how variables allow for easy reuse of values, enhancing maintainability. Nesting, on the other hand, enables a more structured and readable hierarchy in my stylesheets. I practiced using both features in my SASS files.

**Key Takeaways:**

* Explored the benefits of SASS variables.
* Practiced nesting styles for improved readability.

**Day 9: Mixins and Functions**

**Date: 2024/01/15**

Venturing into more advanced features of SASS, today was dedicated to understanding mixins and functions. I learned how mixins can encapsulate reusable pieces of CSS, promoting code efficiency. Additionally, I experimented with functions to create more dynamic and flexible styles.

**Key Takeaways:**

* Explored the concept of mixins in SASS.
* Experimented with creating functions for dynamic styling.

**Day 10: Importing and Partials**

**Date: 2024/01/16**

I delved into the concept of partials, understanding how they allow me to break my stylesheets into smaller, more manageable files. I explored the ’@import’ directive to in include partials in my main SASS file, making my codebase more modular and maintainable.

**Key Takeaways:**

* Utilized partials to modularize my SASS code.
* Learned how to use the ’@import’ directive for including partials.

**Reflection:**

My journey into learning SASS has been enlightening. I’ve gained a solid understanding of its core features, including variables, nesting, mixins, functions, and partials. Implementing these concepts in my stylesheets not only enhances my productivity but also contributes to writing cleaner and more maintainable code. As I move forward, I’m eager to explore more advanced features of SASS and integrate it seamlessly into my web development projects.

**3. Learning Responsive Design**

**Day 11**. **Introduction to Responsive Design**

**Date: 2024/01/20**

Today marked the commencement pf my exploration into responsive design. I delved into the concept of creating websites that adapt and respond gracefully to various devices and screen sizes. I familiarized myself with the importance of responsive design in providing a seamless user expreience across desktop, tablets, and mobile devices.

**Key Takeaways:**

* Understand the significance of responsive design.
* Grasp the concept of adaptability across different devices.

**DAY 12. Viewport and Media Queies**

**Date: 2024/01/22**

To implement responsive design, I started with the basics of the viewport meta tag. I learned how it control the initial scale and dimensions of the viewport on mobile devices. Additionally, I explored media queries, a powerful tool allowing me to apply styles based on the characteristics of the devices, such as screen width.

**Key Takeaways:**

* Implemented viewport meta tag for the better mobile responsiveness.
* Explored media queries to adapt styles based on screen characteristics.

**DAY 13. Flexible Grid Layouts**

**Date: 2024/01/24**

Today, I focused on creating flexible grid layouts using CSS. I learned about the concept of fluid grids, where the sizes of elements is specified in relative units like percentages. This allows for a layout that adapts proportionally to the screen size, ensuring a consistent experience across devices.

**Key Takeaways:**

* Created flexible grid layouts using relative units.
* Understood the importance of fluid grids in responsive design.

**DAY 14. Responsive Images**

**Date: 2024/01/26**

Understanding the impact of images on responsive design, I delved into techniques for ensuring images scale apporopriately. I explored the ’max-width: 100%’ CSS property and the ’srcset’ attribute in characteristics, contributing to faste loading times.

**Key Takeaways:**

* Implemented responsive images using CSS properties and HTML attributes.
* Explored techniques for optimizing images in a responsive design.

**DAY 15. Mobile-First Design Approach**

**Date: 2024/01/28**

Today, I embraced the mobile-first design philosophy. I learned about the advantages of starting the designs process with a focus on mobile devices and progressively enhancing it for larger screens. This approach ensures a fundamental and optimized user experience on smaller devices, with additional features for larger screens.

**Key Takeaways:**

* Embraced the mobile-first design approach.
* Understood the benefits of prioritizing mobile design in the development process.

**Reflection:**

My journey into understanding responsive design has been illminating. I’ve grasped the fundamental concepts of viewport meta tags, media queries, flexible grids, and responsive images. Embracing a mobile-first designs approach has shifted my perspective on creating websites that cater to a diverse range of devices. As I continue advanced techniques and staying updated with industry best practices to ensure my websites provide an optimal user experience across all platforms.

**DAY 16: Learning CSS transition to rotate the button into an X**

**Date: 2024/02/01**

**Objective:** To learn how to use CSS transitions effectively to rotate a button into an X shape for improved user interaction and visual feedback.

**Summary:** In this session, I focused on understanding and implementing CSS transitions to create a rotating effect for a menu button, transitioning it into an X shape. Here's a breakdown of my learning experience:

**Key Takeaways:**

1. **Understanding CSS Transitions:**
   * CSS transitions allow for smooth animations between CSS property changes.
   * By defining transition properties such as duration, timing function, and delay, we can control the animation effect.
2. **Applying Transitions to Button Rotation:**
   * I learned how to target the button element in CSS and define transition properties for the rotation effect.
   * By specifying the transform property and setting the transition duration, I enabled smooth rotation animation.
3. **Triggering Transitions with State Changes:**
   * JavaScript was used to toggle a class on the button element, triggering the rotation effect.
   * The transition effect was activated when the class responsible for rotating the button was applied or removed.
4. **Creating a Rotating Menu Button:**
   * Through the combination of CSS transitions and JavaScript, I successfully implemented a rotating menu button.
   * Clicking the button toggled its rotation state, visually transforming it into an X shape and back.

**Reflection:**

Learning to use CSS transitions for button rotation was an insightful experience. It showcased the power of CSS in creating engaging user interfaces and the importance of smooth animations for enhancing user experience. By mastering CSS transitions, I've added a valuable skill to my toolkit, enabling me to create more dynamic and interactive web elements in future projects.

**Day 17: Learning to Add Branding and Navigation to the Site**

**Date: 2024/02/03**

**Objective: To learn the process of adding branding elements and navigation to a website for improved user experience and site identity.**

**Summary:** In this session, I focused on understanding how to incorporate branding elements and navigation into a website. Here's a breakdown of my learning experience:

**Key Takeaways:**

1. **Importance of Branding:**
   * Branding elements such as logos, colors, and typography play a crucial role in defining a website's identity and creating a memorable user experience.
   * Consistent branding across all pages helps establish credibility and reinforces the site's identity.
2. **Incorporating Branding Elements:**
   * I learned how to add branding elements such as logos and taglines to the website's header section.
   * Using HTML and CSS, I positioned the branding elements appropriately and applied styling to maintain visual consistency.
3. **Implementing Navigation:**
   * Navigation menus provide users with intuitive access to different sections of the website.
   * I explored various navigation design patterns, including horizontal menus, dropdown menus, and hamburger menus for mobile devices.
4. **Ensuring Responsiveness:**
   * It's essential to ensure that the navigation menu remains accessible and user-friendly across different devices and screen sizes.
   * I leveraged media queries and responsive design techniques to adapt the navigation layout for optimal viewing on desktops, tablets, and smartphones.

**Reflection:**

Learning to add branding and navigation to the site was an enriching experience. It highlighted the significance of cohesive branding and user-friendly navigation in enhancing the overall user experience. By carefully integrating branding elements and implementing intuitive navigation, I've improved the site's usability and visual appeal, thereby contributing to its effectiveness in engaging visitors.

**Day 18: Learning to Add Sass Styling and Make the Page Responsive**

**Date: 2024/02/04**

**Objective: To learn how to leverage Sass styling techniques to enhance the design and layout of a webpage, and to ensure responsiveness across different devices using media queries.**

**Summary:** In this session, I focused on incorporating Sass styling and implementing responsiveness to a webpage. Here's a breakdown of my learning experience:

**Key Takeaways:**

1. **Benefits of Sass Styling:**
   * Sass provides powerful features such as variables, nesting, and mixins, which enhance the efficiency and maintainability of CSS code.
   * By organizing stylesheets with Sass, I can streamline the styling process and make it easier to manage and update.
2. **Implementing Sass Styling:**
   * I learned how to set up Sass in my development environment and compile Sass files into CSS.
   * Leveraging Sass features like variables and nesting, I refactored existing CSS code to improve readability and modularity.
3. **Making the Page Responsive:**
   * Responsive design ensures that the webpage adapts seamlessly to different screen sizes and devices.
   * Using media queries, I adjusted the layout and styling of elements to create a fluid and responsive design.
4. **Utilizing CSS Grid for Layout:**
   * CSS Grid offers a flexible and powerful layout system for arranging content in a grid format.
   * I applied CSS Grid techniques to create a responsive layout for the webpage, optimizing the placement of elements for various screen sizes.

**Reflection:**

**Learning to add Sass styling and make the page responsive was a rewarding experience. Sass proved to be a valuable tool for improving the organization and efficiency of my CSS code, while media queries enabled me to create a seamless user experience across different devices. By mastering these techniques, I've enhanced the visual appeal and usability of the webpage, ensuring that it remains accessible to a diverse audience.**

**Day 19: Learning to Use Flexbox**

**Date: 2024/02/07**

**Objective: To gain proficiency in using Flexbox, a powerful layout model in CSS, to create flexible and responsive web layouts.**

**Summary:** In this learning session, I focused on mastering the concepts and techniques of Flexbox layout. Here's a breakdown of my learning experience:

**Key Takeaways:**

1. **Understanding Flexbox:**
   * Flexbox is a CSS layout model that provides a more efficient way to design, align, and distribute space among items in a container.
   * I learned about the main components of Flexbox, including flex containers and flex items, as well as properties like display, flex-direction, justify-content, align-items, and flex.
2. **Creating Flexible Layouts:**
   * With Flexbox, I can easily create flexible and dynamic layouts that adapt to different screen sizes and content variations.
   * By setting the display property of a container to flex, I transformed it into a flex container, enabling the use of Flexbox properties to control the layout of its children.
3. **Aligning and Justifying Content:**
   * Flexbox offers powerful alignment and justification options, allowing me to align items along the main and cross axes of the flex container.
   * Properties like justify-content and align-items enable precise control over the positioning of items within the container.
4. **Handling Flex Item Growth and Shrinkage:**
   * Flex items can grow or shrink to fill available space within the flex container, making it easier to create responsive and adaptive layouts.
   * The flex-grow and flex-shrink properties allow me to control the flexibility of individual items based on their content and available space.

**Reflection:**

**Learning to use Flexbox has been an enlightening experience. Flexbox provides a versatile and intuitive way to create complex layouts with minimal CSS code, offering greater flexibility and control compared to traditional layout methods. By mastering Flexbox, I've expanded my toolkit for designing modern and responsive web layouts, empowering me to build more efficient and visually appealing websites.**

**Day 20: Exploring CSS Grid**

**Date: 2024/02/10**

**Objective: To deepen my understanding of CSS Grid, a powerful layout system in CSS, and learn advanced techniques for creating complex and responsive web layouts.**

**Summary:** In this learning session, I delved into the intricacies of CSS Grid and explored its capabilities for designing versatile and adaptive web layouts. Here's a breakdown of my learning experience:

**Key Takeaways:**

1. **Fundamentals of CSS Grid:**
   * CSS Grid is a two-dimensional layout system that allows for precise control over the placement and alignment of elements within a grid-based layout.
   * I reviewed the basic concepts of CSS Grid, including grid containers, grid items, grid lines, tracks, and cells.
2. **Creating Grid Layouts:**
   * With CSS Grid, I can easily create complex layouts by defining rows and columns within a grid container.
   * By setting properties such as grid-template-rows, grid-template-columns, and grid-gap, I established the structure and spacing of the grid layout.
3. **Positioning Grid Items:**
   * CSS Grid offers multiple ways to position grid items within the grid layout, including explicit grid placement and automatic grid placement.
   * I learned about the grid-row, grid-column, grid-area, and grid-auto-flow properties for controlling the placement and flow of grid items.
4. **Responsive Grid Design:**
   * CSS Grid enables responsive design by allowing grid layouts to adapt to different screen sizes and viewport dimensions.
   * Techniques such as using media queries and the fr unit for flexible track sizing help create layouts that gracefully adjust to varying screen sizes.

**Reflection:**

Exploring CSS Grid has been a rewarding journey. CSS Grid offers a powerful and intuitive approach to web layout design, providing granular control over the positioning and alignment of elements within a grid-based structure. By mastering CSS Grid, I've gained a versatile tool for creating visually stunning and responsive web layouts that adapt seamlessly to different screen sizes and devices.

**Day 21: Understanding Properties of CSS Grid and Flexbox Layouts**

**Date: 2024/02/13**

**Objective: To gain a comprehensive understanding of the properties and characteristics of both CSS Grid and Flexbox layouts, and to discern when to use each layout model based on specific design requirements.**

**Summary:** In this learning session, I focused on exploring the properties and capabilities of both CSS Grid and Flexbox layouts. Here's a breakdown of my learning experience:

**Key Takeaways:**

1. **CSS Grid Properties:**
   * **Grid Container:** A container element with display: grid property that establishes a grid formatting context.
   * **Grid Template:** Defines the structure of the grid layout using properties like grid-template-rows, grid-template-columns, and grid-template-areas.
   * **Grid Item Placement:** Grid items can be placed explicitly using grid-row and grid-column properties or automatically with grid-auto-flow.
   * **Alignment and Spacing:** Properties like justify-items, align-items, justify-content, and align-content control alignment and spacing of grid items within the grid container.
2. **Flexbox Properties:**
   * **Flex Container:** A container element with display: flex property that establishes a flex formatting context.
   * **Flex Direction:** Defines the direction in which flex items are placed within the flex container, with options like row, column, row-reverse, and column-reverse.
   * **Flex Item Alignment:** Properties like justify-content and align-items control the alignment of flex items along the main and cross axes of the flex container.
   * **Flex Item Ordering:** Flex items can be reordered using the order property to change their visual order within the flex container.
3. **Choosing Between CSS Grid and Flexbox:**
   * CSS Grid is best suited for creating complex, two-dimensional layouts with precise control over rows and columns.
   * Flexbox is ideal for creating more flexible, one-dimensional layouts, such as navigation menus, card layouts, and centering elements.

**Reflection:**

Exploring the properties of CSS Grid and Flexbox layouts has provided valuable insights into their respective strengths and applications in web design. While CSS Grid excels in creating complex, grid-based layouts, Flexbox offers more flexibility and control for one-dimensional layouts and alignment tasks. By understanding the properties of both layout models, I can leverage each effectively to create visually appealing and responsive web layouts tailored to specific design requirements.