Yuliya Zditovetskaya

Developing Summary Exercises for Metropolia UAS' Vue.js MOOC course

Metropolia University of Applied Sciences

Bachelor of Engineering

Information and Communication Technology

Bachelor's thesis

15 March 2024

Abstract

Author:	Yuliya Zditovetskaya
Title:	Developing Summary Exercises for Metropolia UAS' Vue.js MOOC course
Number of Pages:	29 pages
Date:	15 March 2024
Degree:	Bachelor of Engineering, UAS
Degree Programme:	Information and Communication Technology
Professional Major:	Full Stack Development
Supervisor:	Janne Salonen, Director of School (ICT)

This engineering research project, commissioned by Metropolia University of Applied Sciences, the largest institution of its kind in Finland, focuses on advancing the Vue.js MOOC online course within the Moodle environment. The primary objective is to develop comprehensive Summary Exercises tailored for Metropolia UAS students, offering an in-depth exploration of Vue.js. The intention is to extend the existing Vue.js course by integrating these exercises, enriching students' theoretical understanding acquired in previous studies.

While the current Vue.js course provides students with basic knowledge of the Vue.js library, including Summary Exercises becomes imperative for refining and expanding their comprehension. This project addresses this task.

The theoretical part of the project delves into the fundamental principles of JavaScript and Vue.js, underscoring their pivotal role in the development of contemporary web applications. Additionally, it addresses the principal concepts of Moodle environment and MOOC, highlighting their significance in online education.

The outcome of the research is the integration of Summary Exercises into the Moodle environment. Upon completion of the Vue.js course, where students acquire a foundational understanding of Vue.js programming, they are well-equipped to tackle these exercises, validating their acquired skills and knowledge.

The final goal of the Summary Exercises for the Vue.js course is to empower students to apply their newfound skills and knowledge in future projects, thereby contributing to the enhancement of their software-developing skills and enriching the overall educational experience.

Keywords: Vue.js, JavaScript, Moodle, MOOC.

Table of Contents

List of Abbreviations

CSC:	Cascading Style Sheets. A computer language for laying out and structuring web pages (HTML or XML)
HTML:	HyperText Markup Language. The standard markup language for documents designed to be displayed in a web browser.
XML:	Extensible Markup Language. A markup language and file format for storing, transmitting, and reconstructing arbitrary data.
JS:	JavaScript. A lightweight interpreted (or just-in-time compiled) programming language with first-class functions.
VDOM:	Virtual DOM. A programming concept where an ideal, or “virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as ReactDOM (has been adopted in Vue).
DOM:	The Document Object Model.
NPM:	Node package manager. A package manager and a software register.
ECMA:	ECMAScript (ES). A standard for scripting languages, including JavaScript, JScript, and ActionScript.
UI:	User Interface.
UX:	User Experience.
API:	Application Programming Interface. A set of protocols and tools that facilitates communication between two pieces of software.
DX:	Developer Experience. A term used to describe for example the experience of a developer using a product, its APIs, documentation and functionality.
Moodle:	Modular Object-Oriented Dynamic Learning Environment
MOOC:	Massive Open Online Course
UAS:	University of Applied Sciences
LMS:	Learning Management System
SPA:	Single-Page Application

1 Introduction

This engineering project was initiated at the request of Metropolia University of Applied Sciences, Finland's foremost institution of applied sciences. It is planned and designed to enhance the Vue.js MOOC online course allocated within the Moodle environment. Metropolia University is the largest institution of its kind in Finland, and it places a significant emphasis on refining and expanding the educational experience for its students.

The primary objective of this engineering project is the development of comprehensive Summary Exercises, tailored for Metropolia UAS students. These exercises serve as a substantive enhancement to the existing Vue.js course and aim to enrich students' theoretical understanding acquired in prior studies.

While the current Vue.js course establishes a foundation by imparting basic knowledge of the Vue.js library, the inclusion of Summary Exercises is necessary for the broader comprehension of students and enhancing the educational process.

The project’s theoretical part entails an examination of the core concepts of JavaScript, Vue.js, Moodle environment and MOOC. The project highlights JavaScript’s and Vue.j’s significance in developing contemporary web applications. Concurrently, the project underscores the important role of Moodle and MOOC in online education.

The main outcome of this project is the integration of Summary Exercises into the Moodle environment. As students complete the Vue.js basics course, equipped with a foundational understanding of Vue.js programming, they are entitled to complete final Summary exercises, validating their newly acquired skills and knowledge.

The main goal of the Summary Exercises for the Vue.js basics course is to empower students to apply their newfound knowledge in future projects. It contributes to improving and strengthening their software development skills, thereby elevating the overall educational experience at Metropolia UAS.

2 JavaScript and Vue.js

In contemporary web development, the importance of JavaScript as one of the foundational programming languages is evident. At the same time, Vue.js as one of the leading JavaScript frameworks gains significant importance. That’s why the studies and new courses aimed to give the students the major understanding and skills in developing modern mobile applications using JavaScript in general and Vue.js, in particular, play an important role in the educational process of many institutions.

Two decades post its initial release, JavaScript persists as the predominant programming language in website development. Surpassing 94% utilization across all websites (Paul, 2024), JavaScript stands as the linchpin for client-side programming, fostering interactive and dynamic web development. Beyond the possibility of presenting static data, JavaScript empowers developers to create web pages with sophisticated functionalities, elevating the user experience and interactivity of websites. The advent of Single-Page Applications further underscores the indispensability of JavaScript, as it orchestrates fluid page interactions and dynamic content rendering.

Vue.js, positioned as a progressive framework for constructing user interfaces (Vue.js, 2024), stands as one of the exemplars of contemporary JavaScript frameworks. Renowned for its simplicity and adaptability, Vue.js enables developers to construct modular, scalable applications. The framework's incremental adoption paradigm allows developers to seamlessly integrate Vue.js into extant projects. Characterized by its intuitive design and robust community support, Vue.js is an important tool in current web development (Tariq, 2023).

JavaScript and Vue.js foundational principles make them indispensable instruments for the contemporary web developer due to inherent to them distinctive attributes and advantages. It makes them important topics for future detailed analysis and the development of new study courses.

2.1 JavaScript

2.1.1 JavaScript: An introduction and short historical overview

JavaScript is a well-known programming language and core technology of the web development, alongside HTML and CSS. Often referred to as the language of the Web, JS exhibits good compatibility, functioning across nearly all web browsers. Its widespread presence positions it as one of the most popular programming languages globally. With a minimalist entry threshold (requiring only a text editor and a web browser) JavaScript is an advanced tool to a broad spectrum of developers. Despite its user-friendly initiation, mastering JavaScript can prove challenging due to its distinctive features. Once proficiency is attained, however, JS enables the creation of potent applications (Jones, 2016).

JavaScript operates as a high-level programming language, undergoing compilation at runtime. This necessitates an engine for interpretation and execution of programs. Predominantly, JavaScript engines are embedded within browsers like Firefox, Chrome, or Internet Explorer. JavaScript is also a dynamic language, which means that elements of a program can change while it is running (Jones, 2016).

The history of JavaScript started in the mid-1990s when Netscape Navigator, an innovative web browser, sought a means to bring dynamism to static web pages. In just ten days, Brendan Eich crafted what would become JavaScript, a scripting language initially named Mocha and later LiveScript. Its introduction in Netscape Navigator 2.0 in 1995 marked a revolutionary shift in web development.

In the late 1990s, Netscape attempted to standardize JavaScript through Ecma International, leading to the creation of the ECMAScript standard.

Despite the ups and downs in JavaScript's history, including its battle with Microsoft during the browser wars, it has stood the test of time and is now one of the most popular programming languages in the world.

JavaScript's early years were characterized by its primary role in manipulating the Document Object Model (DOM), a representation of the structure of HTML documents. This capability allowed developers to interact with and modify the structure, style, and content of web pages (Flanagan, 2020).

The early 2000s witnessed the rise of Asynchronous JavaScript and XML (AJAX), a groundbreaking technique that allowed web pages to request and receive data from servers asynchronously. This innovation, powered by JavaScript, transformed the landscape of web applications, enabling real-time updates and enhancing user interactivity.

As web development complexity increased, jQuery emerged in 2006 as a lightweight, feature-rich library designed to simplify common tasks. Widely adopted for its elegant syntax and cross-browser compatibility, jQuery became a staple for developers seeking to expedite DOM manipulation, event handling, and AJAX requests.

Nowadays JavaScript has evolved to become a foundational language for building complex, scalable web applications. Its frameworks such as Angular, React, and Vue.js has opened a new paradigm, offering structured architectures, efficient state management, and seamless component-based development.

JS has proved to have a transformative role in shaping the internet. From its origins to its current status as a dynamic and versatile language, JavaScript continues to be a driving force behind interactive, engaging web page development and high-quality user experiences (Haverbeke, 2018).

2.1.2 JavaScript: Main Concepts

As the cornerstone of modern web development, understanding JS's main concepts is paramount for developing robust, responsive, and engaging applications. At its core, JavaScript is a scripting language primarily meant to enhance the interactivity and dynamism of web pages (Flanagan, 2020). Its execution within the browser environment allows developers to manipulate the Document Object Model (DOM), "breathe life" into static HTML, and respond to user interactions. Below are the main concepts that form JavaScript's functionality.

- *Variables and Data Types: The Building Blocks*

Variables are the containers for storing data values in JavaScript. Understanding different data types, such as numbers, strings, and booleans, forms the foundation for efficient data manipulation and storage.

- *Control Flow: Navigating the Script*

Conditional statements (if, else) and loops (for, while) empower developers to control the flow of their coding programs. This enables the creation of dynamic, responsive algorithms that adapt to varying scenarios.

- *Functions: Modularizing Code*

Functions are an important tool in modular programming. They contain a set of instructions, helping to break down complex tasks into manageable units. They improve code readability and maintainability, enhance the flexibility and scalability of a program, promote reusability. JavaScript's support for higher-order functions opens possibility to powerful functional programming paradigms.

- *DOM Manipulation: "Breathing Life" into HTML*

JavaScript's ability to interact with the DOM allows developers to dynamically update and modify HTML elements. This updates users experience without the need for page reloads.

- *Asynchronous JavaScript: Promises and Callbacks*

Asynchronous programming in JavaScript is represented by so called promises and callbacks. This ensures non-blocking execution, enabling the handling of operations like fetching data from servers without freezing the user interface.

- *Object-Oriented Programming (OOP): Abstraction and Reusability*

JavaScript supports object-oriented programming principles, allowing developers to create and use objects. This promotes code organization, encapsulation, polymorphism and inheritance for efficient and scalable development.

JavaScript's main concepts make evident that its flexibility and adaptability contribute to its prominence in web development. Mastery of these fundamental concepts empowers developers to create responsive, dynamic web applications (Haverbeke, 2018).

2.1.3 JavaScript Ecosystem: Libraries and Frameworks

JavaScript is a universal language that has given rise to a big ecosystem of libraries and frameworks. Exploring the significance of JS libraries and frameworks in web development tasks is an important topic. It also includes the role of JS in the development of modern, multifunctional web applications.

JavaScript libraries, compact sets of pre-written code, were created to simplify and standardize common tasks. One of the examples is jQuery, launched in 2006, known for its concise syntax and cross-browser compatibility (Haverbeke, 2018). Developers were using jQuery as a powerful tool for DOM manipulation,

event handling, and asynchronous communication, all of these significantly reducing the complexities of web development.

Frameworks, more comprehensive than libraries, offer structured architectures and optimize the development process. AngularJS, released by Google in 2010, presented a Model-View-Controller (MVC) architecture, facilitating the creation of dynamic single-page applications (SPAs). React, developed by Facebook in 2013, introduced a declarative and component-based approach, revolutionizing user interface development. Vue.js, appearing in 2014, brought forth simplicity and flexibility, making it an accessible choice for projects of varying scales.

The decision between using a library or a framework often depends on the project's requirements and the developer's preferences. Libraries like jQuery are favored for their lightweight nature, providing specific functionalities without imposing a rigid structure (Flanagan, 2020). Frameworks, on the other hand, offer comprehensive solutions for structuring applications, managing state, and facilitating efficient development.

The use of libraries and frameworks brings several advantages to the forefront of web development. They enhance code maintainability, encourage best practices, and fasten development tasks through pre-built solutions. Frameworks, in particular, provide a cohesive structure that aids in scaling applications while ensuring consistency.

It becomes evident that JavaScript libraries and frameworks are indispensable tools in the modern web development. Whether opting for the conciseness of libraries or the comprehensive structure of frameworks, JavaScript's expansive the modern web development. Whether libraries or the frameworks, JavaScript's expansive ecosystem empowers developers to create robust, efficient, and engaging web applications (Haverbeke, 2018).

2.2 Vue.js: Introduction and Core Concepts

2.2.1 Introduction to Vue.js

Among the various JavaScript frameworks, React, Vue.js and Angular stand out as one of the most popular. The recent survey by Stack Overflow reaffirms React's dominance over Angular and Vue.js yet again this year.

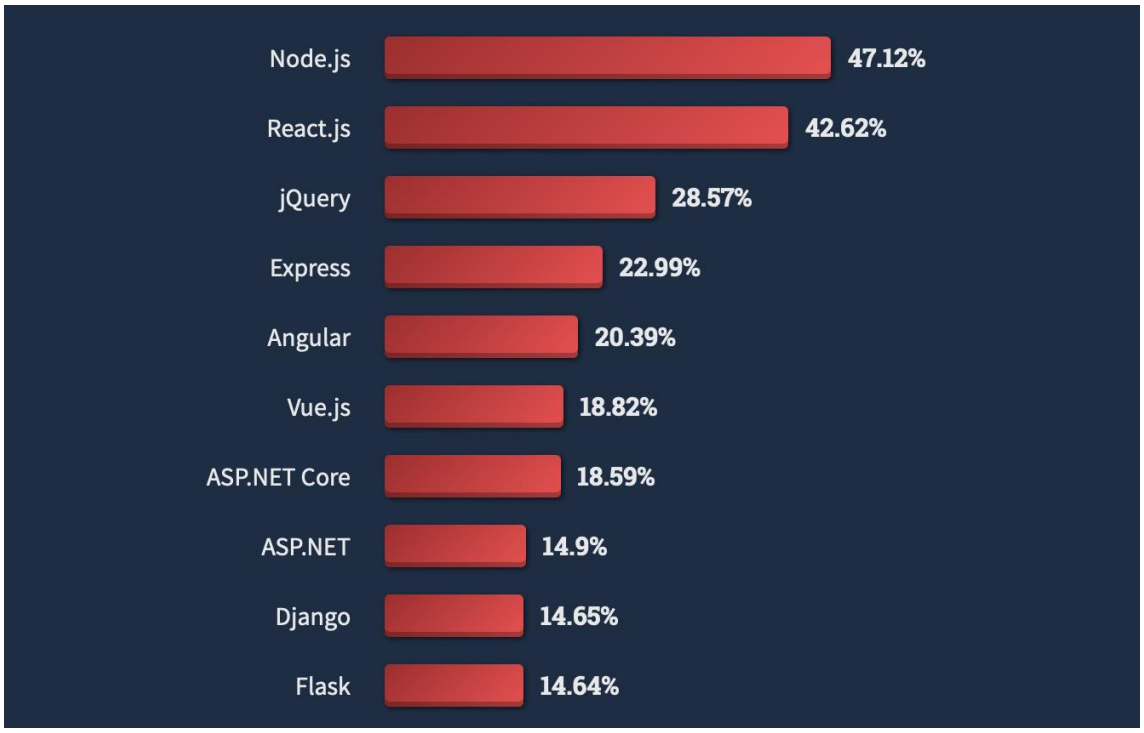

Figure 1. Top ten most popular web frameworks and technologies in 2024 (StackOverflow, 2024).

However, Vue.js also holds on the 6th position among the most popular frameworks and technologies, with its 18.82 percent. With a substantial weekly

download count of 3.4 million on NPM, Vue.js has secured a considerable share of the market.

Vue.js, introduced in 2014, has garnered substantial acclaim in the web development community as a progressive open-source JavaScript framework

(Vue.js, 2024). It facilitates the creation of interfaces with ease and flexibility. Vue.js dynamically updates data into HTML elements bound to different Vue objects. This approach ensures an interactive outcome without requiring page reloads. The automatic update is facilitated through JavaScript events, dynamically refreshing the site's HTML and CSS code.

One of Vue.js's key attributes, simplicity, facilitates developers' understanding and utilization. The reactive system is another vital feature enabling the construction of dynamic interfaces automatically updating based on data changes. Additionally, Vue.js is highly flexible and easily integrable with other libraries and frameworks, making it an ideal choice for developers working on multifaceted applications.

Vue.js offers numerous benefits, including faster development times, improved performance, and enhanced scalability. With a large and active community, developers gain access to abundant resources and support. The framework's popularity has surged due to its user-friendly nature, adaptability, and many other advantages.

Compared to other frameworks like React and Angular, Vue.js boasts a simpler learning curve (Rojas, 2019), owing to its lightweight nature and user-friendly design. Notably, Vue.js outperforms in terms of performance, with smaller file sizes accelerating load times. However, Vue.js faces a limitation in community support when juxtaposed with React and Angular.

Despite this, Vue.js stands as a powerful and versatile framework, gaining popularity among developers for its performance and user-friendliness.

Conceived by Evan You as a response to the complexities of existing frameworks, Vue is self-sustained and community-centric. Presently, Vue is under continuous maintenance by a global team comprising both full-time professionals and dedicated volunteers. Evan spearheads the project as the lead (Vue.js, 2024).

Vue.js is centered around simplicity, flexibility, and incrementality, which explains that Vue.js an approachable framework for developers at all skill levels. Mechanics of Vue.js include its reactive data-binding system and the virtual DOM, and these foundational elements contribute to the framework's efficiency and ease of use.

2.2.2 Component-Based Architecture of Vue.js

Main concepts of Vue.js include the principles of reusability, encapsulation, and maintainability. Creating and composing components, understanding and using their indispensable role in shaping the structure of modern web applications is the important part of working with Vue.js.

A component-based architecture is a software design paradigm that prioritizes constructing a system through smaller, reusable components. In the context of a Vue.js project, this entails deconstructing the user interface (UI) into manageable, scalable, and reusable components (Saafan, 2024).

Key engineering practices for implementing a component-based architecture in Vue.js projects:

- Decomposition of UI into components: first it is necessary to identify UI elements suitable for reuse across various application sections and later – to create distinct components for each;
- Maintaining small, focused components: first step is to ensure that each component has a distinct and well-defined purpose;
- Utilization of props and events for inter-component communication: using props for passing data from parent components to child components and events to emit data from child components to parent components;
- Leverage slots for flexibility: using slots to enable customization of a component's content by the parent component;
- Establishing a naming convention: adoption of a clear naming convention that reflects each component's purpose;

- Implementation of a structured component folder: components should be logically organized within a folder structure. Related components should be grouped into a "components" folder, and further categorize them into subfolders as needed;
- Using a style guide: a style guide is used to ensure uniformity in component design and development. The guide can define naming conventions, component structures, and design patterns to be consistently applied throughout the application;
- Enforcing a consistent naming convention: uniformity in naming conventions across components, fostering clarity and coherence in Vue.js project should be ensured (Saafan, 2024).

2.2.3 Data Binding and Reactivity

Vue.js uses data binding to connect data sources and consumers, ensuring automatic synchronization during retrieval. In this process, any changes in the data trigger automatic updates in the elements bound to that data, reflecting Vue.js's reactivity system.

Moreover, the concept of data binding extends to scenarios where alterations in the outer representation of data within an element prompt automatic updates in the underlying data to mirror this modification (Vue.js, 2024).

In Vue.js data binding, manipulation of an element's class list and inline styles is a requisite. Recognizing that the class list and inline style are attributes, v-bind is employed for their management. Vue.js introduces specialized features when v-bind is utilized with class and style. The expressions within this context can evaluate not only to strings but also to objects or arrays, adding a flexibility to the data binding process.

Reactivity serves as the dynamic mechanism enabling the framework to recognize alterations in the data utilized on a webpage, specifically when it undergoes mutations (Worldline, 2024). This dynamic responsiveness ensures

that the page is optimally updated to accurately reflect these changes. This mirrors the adaptive nature of control systems that dynamically adjust to changing conditions.

Reactivity is an essential feature for any web framework, aligning with the fundamental principle of real-time responsiveness and adaptability to varying inputs and conditions.

Vue.js simplifies development by automatically updating the view when data changes, eliminating the need for developers to manually trigger these updates. This aligns with principles of efficiency and automation (Worldline, 2024).

The Vue's reactivity is achieved by the utilization of a JavaScript feature known as Proxies. Proxies, introduced with the ES2015 specification, provide developers with comprehensive control over all operations involved in manipulating an object (Vue.js, 2024). This level of control is similar to adjusting the parameters and responses of a dynamic system, ensuring optimal performance and adaptability.

Vue strategically uses Proxies to intercept both read and write access to properties within views. This interception enables Vue.js to detect changes in the data and efficiently update the relevant elements impacted by these changes.

This approach reflects the precision and control required in the web development, where dynamic adjustments and real-time updates are fundamental for optimal functionality of the web pages.

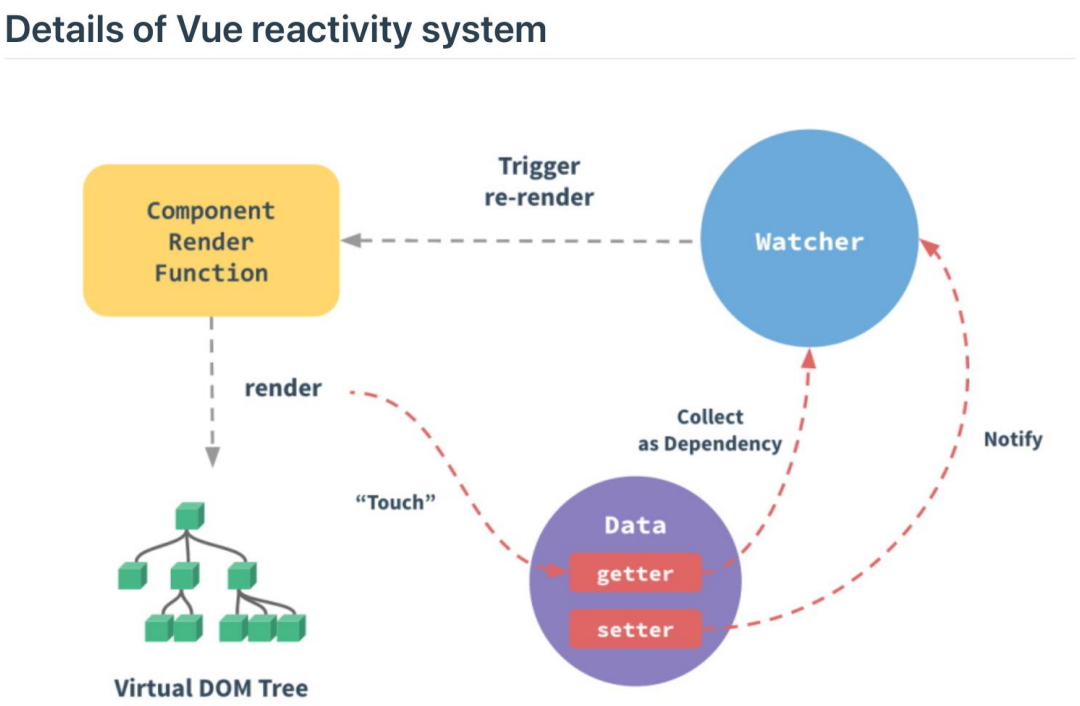

Figure 2. Details of Vue reactivity system (Wordline, 2024).

2.2.4 Vue.js Directives

Vue.js directives act as specialized HTML attributes guiding Vue on how to handle DOM elements. Prefixed with the v- symbol, and can be used to bind teh data, add event listeneres, control the rendering of the elements and more. So they are in a way dictating dynamic behaviors and presentations in the Vue.js applications. In web development, directives are akin to steering commands, orchestrating dynamic actions like toggling element visibility, addition/removal, and more (Vue.js, 2024).

Vue.js provides a set of built-in directives, each tailored for specific actions under diverse conditions or parameterers. Below are some of these directives and their counterparts:

- **v-show Directive:** Functions like a toggle, showing/hiding HTML elements conditionally—resembling an if-else condition, executing actions based on specific conditions.

- **v-bind Directive:** Aligns with one-way data binding, binding values from component logic to the HTML template—resembling the unidirectional flow of data in a control system.
- **v-model Directive:** Embodies two-way data binding, creating bidirectional connections between the script and Vue template—similar to real-time feedback mechanisms.
- **v-for Directive:** Similar to a for loop, it facilitates iteration over arrays, objects, or numeric ranges—mirroring iterative processes in functional requirements.
- **v-if, v-else, and v-else-if Directives:** Resemble conditional rendering instructions, adding or removing elements based on specific conditions—mirroring conditional statements in system responses.
- **v-text Directive:** Comparable to interpolation, it binds data variable values directly to Vue templates, following a one-way data binding approach—similar to real-time variable displays.
- **Custom Directives:** When built-in directives fall short of meeting specific functional requirements, the concept of custom directives arises. In web development, this mirrors creating bespoke control signals for unique system behaviors. The dedicated "directives" section in the Vue file's script offers a modular approach, akin to specialized control modules in engineering systems. Developers can address specific behaviors not covered by standard directives, providing a tailored engineering solution within the Vue.js framework.

  Vue.js directives form a fundamental element in dynamic and interactive web app engineering. They execute DOM manipulation, streamline data binding, and offer elegant solutions for common use cases. Mastering Vue.js directives empowers developers to engineer purposeful and efficient web applications, with each directive contributing to elegant solutions for a wide range of scenarios (Allotey, 2023).

3 Moodle and MOOC

Moodle and MOOC are two popular ways of online learning, carrying plenty of similarities and differences. Moodle, prevalent in educational institutions, provides a robust framework for educators to deliver content and assessments, and monitor student progress. MOOC is meant for individual learners, offering a diverse range of courses from various providers, and empowering students to explore different topics. Both Moodle and MOOC have an important role in the dynamic landscape of online education.

Moodle, an open-source learning management system, provides a versatile and robust platform for educators and learners alike (Moodle, 2024).The main features that make Moodle indispensable, are its flexibility in course creation, collaborative tools, and the ability to adapt to diverse learning styles. Moodle is a centralized hub for educational resources, and it emerges as a catalyst for fostering interactive and engaging learning experiences.

Massive Open Online Courses (MOOCs) have emerged as a transformative force, democratizing access to education on a global scale. MOOCs break down geographical and financial barriers, making high-quality educational content accessible to a vast audience.

Moodle's collaborative tools and MOOCs' interactive nature contribute to a shift in educational paradigms. They facilitate collaborative learning experiences, enabling students to engage in discussions, share resources, and participate in virtual classrooms.

The importance of Moodle and MOOCs in the educational landscape is evident and proven, this chapter further explores their impact on pedagogy, accessibility, and the evolving nature of learning in the digital age (Moodle, 2024).

3.1 Moodle

Moodle Learning Management System (LMS) is an excellent solution for online courses delivery, offering a spectrum of features that align with diverse educational needs.

Flexibility: Moodle's adaptability is a standout feature, catering to both small and large-scale implementations. The platform's customization options, encompassing themes, plugins, and modules, ensure tailored alignment with organizational requirements.

Ease of Use: Designed with user experience in mind, Moodle LMS ensures simplicity for both learners and instructors. Its intuitive interface facilitates easy access to course materials and seamless assignment completion. Instructors benefit from straightforward course creation, management, and comprehensive progress tracking.

Multimedia Support: Moodle LMS accommodates an extensive range of multimedia formats, including videos, audio files, and images. This versatility empowers instructors to craft engaging courses that integrate diverse media types, catering to varied learning preferences.

Assessment Tools: The LMS offers a suite of assessment tools, comprising customizable quizzes, assignments, and interactive activities. These tools enable instructors to effectively evaluate learner progress and comprehension, tailoring assessments to individual or group needs.

Community Support: Backed by a vibrant and expansive community, Moodle LMS enjoys continuous support, guidance, and regular updates from users, developers, and educators. This collective expertise contributes to the platform's evolution and enhancement.

Moodle is very versatile in course creation, content management, and communication tools, it fosters interactive learning experiences through forums,

quizzes, and collaborative activities. Moodle is adaptable in accommodating various teaching styles and pedagogical approaches.

Moodle provides an intuitive and user-friendly environment for both educators and learners. It has the layout, navigation, and accessibility features that contribute to a seamless and engaging learning experience (Moodle, 2024). It is known that a well-designed interface is important in enhancing user satisfaction and facilitating effective education delivery.

Data-driven insights of Moodle empower educators to monitor student progress, identify challenges, and tailor interventions. The significance of learning analytics in shaping adaptive learning pathways, ensuring a personalized educational journey for each student is undeniable.

Moodle is adopted in educational institutions globally. It also serves as a central hub for educational resources, supporting diverse learning models and institutional goals, it fosters collaboration among educators and facilitating continuous improvement in teaching methodologies.

In summary, Moodle LMS is an excellent choice for online course delivery due to its flexibility, user-friendly design, multimedia support, assessment tools, and community support. It is a powerful platform that can be customized to meet the needs of any organization. It is also a great choice for both small and large-scale implementations. It's easy access for learners and efficient course management for instructors make it a versatile solution for organizations of varying scales.

3.2 MOOC

A Massive Open Online Course (MOOC), or an open online course is an online course, designed for unrestricted participation and open access via the Web.

Massive Open Online Courses (MOOCs) are a transformative force in the educational processes. The main characteristics that make MOOCs unique, are

their massive scalability, open accessibility, and the potential to reach a diverse global audience.

Massive Open Online Courses (MOOCs) offer free enrolment to individuals worldwide. They present an economical and adaptable way to acquire new skills, progress in one's career, and deliver high-quality educational experiences.

Individuals across the globe use MOOCs for diverse purposes such as career development, career transitions, college readiness, supplementary learning, lifelong learning, corporate eLearning, and training (MOOC; 2024).

MOOCs have brought a significant transformation in the global learning landscape.

Popular platforms such as Coursera, edX, and Udacity are examples of MOOCs. Their unique approaches to course delivery, assessment, and learner engagement make them quite good tools in study processes. The diverse range of courses available on MOOC platforms covers various disciplines and skill levels.

MOOCs break down barriers to education, offering accessibility and inclusivity on a global scale. Thus they affect the democratization of knowledge and make an impact in providing learning opportunities to individuals who may face geographical, financial, or institutional constraints.

The interactive nature of MOOCs facilitates collaborative learning experiences. The incorporation of discussion forums, peer assessments, and collaborative projects within MOOCs illustrates how these features contribute to a dynamic and engaging online learning community.

4 Summary Exercises

As mentioned before, this project focuses on advancing the Vue.js MOOC online course within the Moodle environment by developing and adding to it Summary Exercises. These exercises are supposed to be tailored specifically for Metropolia UAS students, offering an in-depth exploration of Vue.js concepts and techniques, improving the students' knowledge and expanding their comprehension and programming skills.

While the current Vue.js course provides students with a basic understanding of the Vue.js library, integrating Summary Exercises becomes imperative for refining and expanding their comprehension. This project addresses this task by providing practical challenges that reinforce key Vue.js concepts and enhance problem-solving abilities.

Upon completing the Vue.js course, students gain a foundational understanding of Vue.js programming, preparing them to tackle Summary Exercises effectively and validate their acquired skills and knowledge.

Ultimately, the goal of the Summary Exercises for the Vue.js course is to empower students to apply their newfound skills and knowledge in future projects, thereby enhancing their software-developing capabilities and enriching their overall educational experience at Metropolia University of Applied Sciences.

4.1 Vue.js: Get Started, Methods and Directives

In the first part of Summary Exercises, the short introduction to Vue.js' main terminology and programming basics and the structure of the framework are represented. This part can help to form a comprehensive understanding of the framework's structure, syntax and provide some necessary for understanding and performing theoretical knowledge.

In this initial part of the Summary Exercises, Vue.js and its fundamental concepts are given through a series of exercises focused on getting started with the library, and some of the main methods and directives used by the library. These exercises are designed to reinforce students' understanding of Vue.js syntax and functionality. In this part, students can tackle essential Vue.js concepts through hands-on exercises. By completing these exercises, they can gain practical experience in setting up Vue.js applications, utilizing data binding, implementing methods, and working with directives. These foundational skills will serve as a solid framework as they continue to explore and build with Vue.js.

The exercises in this section cover a broad spectrum of Vue.js concepts, ranging from foundational principles to advanced techniques. By engaging in exercises focused on class binding, list rendering students immerse themselves in the intricacies of Vue.js development, honing their skills with each task completed. Also, in this part it is reminded about Vue directives shorthands, such as ':' and '@'.

Furthermore, participants will explore conditional rendering techniques such as v-if and v-show, empowering them to dynamically control the visibility of elements based on data properties. Additionally, they'll discover the power of directives like right.prevent and v-model, enabling them to prevent default behaviour and achieve two-way data binding effortlessly.

In the part dedicated to methods, participants will learn how to execute functions on events like clicks using v-on:click, while also mastering the accessing of data properties from within methods (with 'this' word). The utilization of $event allows for passing click events with the method as an argument, facilitating seamless interaction between the user interface and underlying data.

With each exercise completed, the students can empower themselves with practical experience in improving Vue.js's development skills. From setting up Vue.js applications to integrating data binding and unleashing the potential of

methods and directives, they'll navigate through the fundamental concepts of Vue.js development.

4.2 Vue.js: Computed Properties, Watchers and Scaling up

The advanced Vue.js concepts, including computed properties, watchers, and scaling up with Vue Single-File Components (SFCs) were displayed in the next part of the Summary exercises. Through them, the students are meant to deepen their understanding of these concepts and equip themselves with the tools necessary to build sophisticated Vue.js applications.

Computed properties and methods may appear similar as they are both written as functions, but they serve different purposes. Methods execute when called from HTML, whereas computed properties automatically update when a dependency changes. Computed properties function similarly to data properties but offer dynamic behavior.

Implementing Computed Properties is managed by next stages:

- Defining computed properties within a Vue component;
- Utilizing computed properties to perform calculations or transformations based on reactive data;
- Observing how computed properties update automatically when dependent data changes.

In their turn, Vue.js Watchers provide a way to perform asynchronous operations in response to changes in reactive data. They offer performed control over data changes and enable developers to execute custom logic when specific data properties change.

The scheme of utilizing of Vue.js Watchers:

- Declaring watchers within a Vue component;

- Defining custom logic to execute when watched data properties change;
- Experimenting with asynchronous operations or complex computations within watchers.

Vue Single-File Components (SFCs) revolutionize Vue.js development by encapsulating the template, logic, and styling of a component in a single file. This natural extension of HTML, CSS, and JavaScript offers numerous benefits, including modularity, colocation of concerns, and pre-compiled templates without runtime compilation cost.

While Vue Single-File Components require a build step, they offer a numerous benefits:

- Author modularized components using familiar HTML, CSS, and JavaScript syntax.
- Colocation of inherently coupled concerns for improved maintainability.
- Pre-compiled templates without runtime compilation cost for enhanced performance.
- Component-scoped CSS for better encapsulation and maintainability.
- More ergonomic syntax when working with the Composition API for improved developer experience.
- Increased compile-time optimizations by cross-analyzing template and script.
- IDE support with auto-completion and type-checking for template expressions.
- Out-of-the-box Hot-Module Replacement (HMR) support for seamless development experience.

In this part of the Summary exercises, the students have a chance to learn advanced Vue.js concepts, including computed properties, watchers, and Vue Single-File Components. By completing the summary exercises, they will expand their toolkit and deepen their understanding of Vue.js development. Armed with this knowledge, students will be well-equipped to build sophisticated Vue.js applications with efficiency.

4.3 Vue.js: Components, Slots

In this part of the Summary exercises Vue.js components are demonstrated, altogether with the usage of props for data passing between parent and child components, the $emit() method for communication from child to parent, as well as the powerful concept of slots. Through a series of summary exercises, the students can deepen their understanding of these concepts and equip themselves with the skills necessary to build versatile and reusable Vue.js components.

Props serve to pass data from parent components to child components via custom attributes on the component tag. This mechanism facilitates the communication of information down the component hierarchy, enabling parent components to supply data to their children.

While props facilitate data flow from parent to child components, the $emit() method enables communication from child to parent components. By emitting custom events, child components can notify their parent components about specific actions or changes.

Slots are a versatile feature in Vue.js that allows for flexible content distribution in components. They enable developers to define placeholder areas in a component's template where additional content can be injected from the parent component.

In scenarios where a component has multiple root elements, v-bind="$attrs" allows developers to specify on which root element a fallthrough attribute should be applied. Additionally, v-slot directives enable the passing of data from a component's slot to the parent component.

After completing these exercises, the students will learn the intricacies of Vue.js components, master the usage of props for data passing, learn how to communicate between components using $ $emit(), and the flexibility slots offer. They will expand their Vue.js toolkit and gain valuable insights into building

robust and dynamic Vue.js applications. Armed with this knowledge, they will be well-prepared to create versatile and reusable components that enhance the functionality and maintainability of Vue.js projects.

4.4 Vue.js: Refs, Lifecycle Hooks, Provide/Inject

In this part of the Summary exercises advanced Vue.js concepts, including refs, lifecycle hooks, and provide/inject are provided. These concepts offer powerful tools for interacting with the underlying DOM, managing the component lifecycle, and solving common challenges like props drilling. Through a series of summary exercises, the students can deepen their understanding of these concepts.

Refs are a mechanism for accessing specific DOM elements directly within a Vue component. While Vue's declarative rendering model abstracts away most direct DOM operations, there are scenarios where direct access to DOM elements is necessary. Refs provide a way to achieve this by assigning a reference to a DOM element or component instance.

Vue component instances go through a series of initialization steps and updates during their lifecycle. Lifecycle hooks provide developers with the opportunity to execute custom code at specific stages of a component's lifecycle. This enables tasks such as initialization, DOM manipulation, and cleanup.

Props drilling, the process of passing data through multiple layers of nested components, can lead to code complexity and decreased maintainability. Provide/Inject offers a solution by allowing a parent component to provide data that can be injected into any descendant component, regardless of its depth in the component tree.

This chapter provides the students with advanced Vue.js concepts including refs, lifecycle hooks, and provide/inject. By completing the summary exercises, they can expand their Vue.js toolkit and gain valuable insights into managing

component interactions, lifecycle events, and data flow within Vue.js applications.

4.5 Vue.js: Routing and Animations

In the last part of the Summary exercises were explained Vue.js routing and animations, high lightening the differences between client-side and server-side routing, forming understanding routing setup in Vue.js, and discovering various techniques for implementing animations within Vue.js applications.

Routing on the server side involves the server sending a response based on the URL path visited by the user. Traditionally, clicking on a link in a server-rendered web app triggers a full page reload as the browser receives a new HTML response from the server. However, in Single-Page Applications (SPAs), client-side JavaScript intercepts navigation, dynamically fetches new data, and updates the current page without full page reloads, resulting in a more responsive user experience.

Routing in Vue.js occurs on the client side, enabling fast and seamless navigation within the application without full page reloads. The routing setup is typically configured in the 'main.js' file of a Vue.js project.

Vue.js provides built-in components like <Transition> and <TransitionGroup> for handling enter/leave and list transitions. Additionally, animations can be achieved using class-based animations by dynamically adding CSS classes or interpolating values with watchers.

By completing this part of the summary exercises, the students will gain a deeper understanding of client-side routing, routing setup in Vue.js, and various animation techniques within Vue.js applications. Armed with this knowledge, they're well-equipped to create dynamic and engaging Vue.js applications that offer seamless navigation and captivating visual experiences.

5 Summary

The thesis was focusing on advancing the Vue.js MOOC online course within the Moodle environment The primary objective was to develop comprehensive Summary Exercises tailored for Metropolia UAS students, offering an in-depth exploration of Vue.js.

The theoretical part delves into the fundamental principles of JavaScript and Vue.js, underscoring their pivotal role in the development of contemporary web applications. Additionally, it addresses the principal concepts of the Moodle environment and MOOC, highlighting their significance in online education.

Through passing a series of meticulously designed exercises, students are guided into Vue.js syntax and functionality. With each exercise completed, students inched closer to realizing their potential as proficient Vue.js developers. Armed with a deep understanding of Vue.js, they are well-prepared to tackle more complex challenges and embark on ambitious projects with confidence and ease.

The integration of Summary Exercises into the Moodle environment enriches the educational experience of students completing the Vue.js course. These exercises not only refine and expand students' comprehension but also validate their acquired skills and knowledge. Upon completion of the course, students are empowered to apply their newfound skills in future projects, contributing to the enhancement of their software-developing skills.

The Summary Exercises for the Vue.js course serve as a cornerstone in the educational journey of Metropolia UAS students. By providing a comprehensive understanding of Vue.js and empowering students to apply their knowledge, this project contributes to the overall advancement of online education in the field of web development.

References

Allotey, C., 2023. Vue.js Directives: A Beginner's Guide. [Online] Available at: https://vueschool.io/articles/vuejs-tutorials/vue-js-directives-a-beginners-guide/ [Accessed 24 February 2024].

Büchner, A., 2016. Moodle 3 Administration. Packt Publishing Ltd.

Filipova, O., 2016. Learning Vue.js 2. Packt Publishing Ltd.

Flanagan, D., 2020. JavaScript: The Definitive Guide. O'Reilly Media, Inc, USA.

Haverbeke, M., 2018. Eloquent Javascript. 3rd Edition. No Starch Press, US.

Jones, D., 2016.JavaScript: Novice to Ninja. SitePoint Pty.Ltd.

MOOC, 2024. MOOC. [Online] Available at: https://www.mooc.org/ [Accessed 24 February 2024].

Moodle, 2024. Moodle for MOOCs. [Online] Available at: https://moodle.com/moodle-for-moocs/ [Accessed 24 February 2024].

Paul, C., 2024.The Role of JavaScript in Modern Web Development. [Online] Available at: https://www.linkedin.com/pulse/role-javascript-modern-web-development-charles-paul-udcbc/ [Accessed 22 January 2024].

Rojas, C., 2019. Building Progressive Web Applications with Vue.js. Apress.

Saafan, A., 2024.Good ways to Organize Large Vue.js Project. [Online] Available at: https://www.linkedin.com/pulse/good-ways-organize-large-vuejs-project-amr-saafan/[Accessed 24 February 2024].

StackOverflow, 2024. [Online] Available at: https://survey.stackoverflow.co/2022/#technology[Accessed 16 February 2024].

Tariq, M., 2023. Vue.js: The Progressive JavaScript Framework for Modern Web Apps. [Online] Available at: https://www.linkedin.com/pulse/vuejs-progressive-javascript-framework-modern-web-apps-tariq-mahmood-ggbpf/?trk=articles_directory [Accessed 22 January 2024].

Vue.js - The Progressive JavaScript Framework, 2024. [Online] Available at: https://vuejs.org/ [Accessed 16 January 2024].

Worldline, 2023. Introduction to Vue.js. Reactivity. · GitHub. [Online] Available at: https://worldline.github.io/vuejs-training/reactivity/#proxies-ecmascript-6-aka-es2015 [Accessed 24 February 2024].