



Islam Al-Badri, Asmae Aouragh, Kawther Al Delfi

Implementing Vue.js course on Moodle Platform

Metropolia University of Applied Sciences
Bachelor of Engineering
Information and communication technology
Bachelor's Thesis
05.09.2023

Abstract

Author: Islam Al-Badri, Asmae Aouragh, Kawther Al Delfi
Title: Implementing Vue.js course on Moodle Platform
Number of Pages: 51 pages + appendices
Date: 05.09.2023

Degree: Bachelor of Engineering
Degree Programme: Information and communication technology
Professional Major: program Software production
Supervisors: Competence area manager Janne Salonen

The Metropolia University of Applied Sciences initiated a project aimed at developing an introductory online course on Vue.js within the Moodle platform.

The primary objective was to create a comprehensive learning experience for students across various study units, such as open MAK, multi-format, and day students. The overarching aim was to equip students with a foundational grasp of the JavaScript language and the Vue.js library. The project encompassed a thorough exploration of open-source projects, which served as a prerequisite for acclimatizing students to the course content.

Additionally, the endeavor involved a meticulous examination of diverse open-source licenses to comprehend their implications on the utilization of different components. The theoretical segment of the project delved into various programming languages and software packages that constitute integral components of contemporary web applications. A comparative analysis was undertaken, pitting these technologies against their competing counterparts. This encompassed an assessment of the interdependencies between Moodle and Vue.js.

The culmination of this effort was the successful creation of a Moodle-based course. The course empowers students with a foundational understanding of Vue.js programming, thereby enabling them to apply this knowledge to their future projects. Furthermore, students gain insight into essential tools requisite in the developmental milieu, fostering a holistic learning experience.

Keywords: Vue.js, Moodle, Node.js, CSS, JavaScript, TypeScript

Table of Contents

1	Introduction	1
2	Open-source code	2
2.1	General	2
2.2	Open-Source Licensing.....	3
2.2.1	BSD License	4
2.2.2	GPL License	5
2.2.3	Apache License.....	5
3	Moodle LMS.....	6
3.1	Moodle Learning Management System	6
3.2	Moodle's Platform Infrastructur	7
3.2.1	Linux: The Versatile Operating System	8
3.2.2	Apache: The Dynamic Web Server.....	10
3.2.3	MariaDB: Power and Complexity in Open-Source Databases	11
3.2.4	MariaDB vs. MySQL.....	12
3.2.5	PHP: Empowering Dynamic Web Development	13
4	Vue.js: Empowering Interactive User Interfaces	16
4.1	Overview of Vue.js	16
4.1.1	Vue.js vs. Angular: Choosing the Right Path	17
4.1.2	Vue.js vs. React: Navigating the Frontend Landscape	19
4.2	Node.js: Empowering Server-Side Application Developmen	21
4.3	JavaScript: A Dynamic Web Enabler.....	23
4.4	TypeScript: Elevating JavaScript Development	24

4.5	ESLint: Elevating Code Quality through Analysis	26
4.6	HTML: Crafting the Web Canva.....	27
4.7	Unveiling the Artistry of CSS: Crafting Visual Magic	30
5	Creating a Vue.js Basics Course	34
5.1	Module 1: Unveiling Vue.js	35
5.1.1	Visual Studio Code: A Versatile Coding Companion	35
5.1.2	Exploring Visual Studio Code's Key Features	36
5.2	Module 2: Delving into Vue.js Basics	39
5.3	Module 3: Unveiling Components and Rendering	39
5.4	Module 4: Extending Functionality and Interaction	40
5.5	Module 5: Harnessing CSS and Webpack.....	40
5.6	Module 6: Unveiling Computational Features	40
5.7	Module 7: Mastering Conditional Rendering.....	41
5.8	Horse Adventure: An Exploration of TypeScript	41
6	Summary: Development of a Vue.js Course.....	43
7	Sources.....	45

List of Abbreviations

CSS: Cascading Style Sheets. Cascading style guides that define the style of a web page.

LMS: Learning Management System. Digital learning environment.

LAMP: Linux, Apache, MySQL, PHP. A general set of different software for providing websites.

MYSQL: Database engine.

PHP: Hypertext Preprocessor. A server-based programming language.

JS: JavaScript and a file extension made with JavaScript.

TypeScript: A superset of JavaScript that adds static typing.

YEAR: Visual User Environment. JavaScript programming library.

VDOM: Virtual document object model.

DOM: Document object model.

NPM: Node Packet Management. Node package management.

I/O: Input/Output.

HTML: Hypertext Markup Language. Language for website development.

ECMA: European Computer Manufacturers Association. JavaScript maintenance..

1 Introduction

The engineering thesis aimed to develop a comprehensive Vue.js course for Metropolia University of Applied Sciences' Moodle platform. The course was designed to cater to open university, blended learning, and full-time students as an elective study module.

The primary objective of the course was to provide students with a strong foundation in Vue.js programming. This included creating custom components, understanding the JavaScript and TypeScript structures, and applying these skills to their own projects. The project was commissioned by Metropolia University of Applied Sciences, Finland's largest university of applied sciences.

Metropolia University of Applied Sciences is a prestigious Finnish institution known for its diverse range of multidisciplinary degree programs. With a focus on practical learning, state-of-the-art facilities, industry collaborations, and a vibrant community, Metropolia offers high-quality education that prepares students for successful careers. The university's commitment to academic excellence and real-world experience ensures that graduates are well-equipped to make meaningful contributions in their chosen fields.

All the topics covered in the thesis revolved around open-source technologies, which have gained significant popularity due to the internet's growth. Open source offers numerous advantages for developers, although it also presents challenges, such as different licensing requirements.

The theoretical part of the thesis provided a comprehensive overview of Vue.js's history, the Moodle learning environment, and the Vue.js development ecosystem. It also offered a concise introduction to Vue.js fundamentals. The target audience for this thesis comprised technology-oriented students who already possessed basic knowledge of HTML and JavaScript programming.

The second part of the thesis delved deeper into the Vue.js library, exploring its overall structure and operational principles. The Moodle instructional materials included both theoretical explanations and practical exercises, enabling students to develop a thorough understanding of web programming, file structure, and the creation of custom components. The goal was to ensure that students grasped the concept of different components, learned how to modify style files, and successfully integrated various components into their applications.

The final outcome was a Moodle course within the learning environment, where students gained a solid understanding of Vue.js programming fundamentals and were able to apply their knowledge effectively to their future projects. (1.)

2 Open-source code

2.1 General

The open-source community has experienced remarkable growth in recent decades. With the rise of the Internet and advancements in technology, open-source development has become the preferred way for developers to collaborate and share their work. The main driving force behind the expansion of the open-source community is the collaborative nature of the development process.

Developers from all corners of the globe can come together to contribute to a project, making it more sustainable and reliable.

This collaborative approach has resulted in the creation of software that is freely available and easily accessible to anyone. Moreover, the open-source community has cultivated a culture centered around transparency and accountability. Because the source code is open for everyone to see, issues such as errors and security vulnerabilities can be quickly identified and addressed. This has led to a notable improvement in the overall quality of the software being developed.

Furthermore, the open-source movement has had a democratizing effect on technology. By providing software free of charge, individuals and organizations that might not have had the resources to access expensive proprietary software can now benefit from and utilize these tools. As a result, the open-source community has played a significant role in making technology more inclusive and accessible.

In summary, the growth of the open-source community has had a profound impact on the software development industry. It has fostered collaboration, transparency, and accountability, resulting in the creation of high-quality software that is available for everyone to use and build upon. (2.)

2.2 Open-Source Licensing

Open-source licensing constitutes the legal framework that governs the utilization, distribution, and alteration of software accessible to the public. This licensing paradigm facilitates users in accessing the source code – the underlying software code – and effecting modifications or enhancements to that code. This liberty is frequently harnessed to craft novel software tailored to distinct requirements or preferences.

Prevalent open-source licenses encompass the General Public License (GPL), the Berkeley Software Distribution (BSD) license, and the Apache License. Each of these licenses extends varying degrees of freedom and constraints, contingent upon the user's or developer's requisites.

In essence, the crux of open-source licensing is to foster collaboration and ingenuity in software development by conferring users with the unrestricted ability to employ and refine software. (3.)












	 Free software	 Open-source software	 Freeware	 Public-domain software
Definition	"FREE" is a matter of liberty, not price	"OPEN" doesn't just mean access to the source code	"FREE" refers to price, while freedom of the use is restricted by creator	"PUBLIC DOMAIN" belongs to the public as a whole
Ground philosophy	Social movement	Development methodology	Marketing goals	Copyright disclamation
Ground rules	Four Freedoms https://www.gnu.org/philosophy/free-sw.html	Open Software initiative https://opensource.org/osd		Creative Common Organization https://creativecommons.org
Free of charge	Not necessary	Not necessary	✓ YES	✓ YES
Covered by copyright law	✓ YES	✓ YES	✓ YES	✗ NO
Examples	 	 	 	

Figure 1 Table of open-source licenses (Todavchich: 2020)

2.2.1 BSD License

The BSD license, a widely embraced open-source software license, has found favor among diverse entities and individuals. This permissive license empowers users to modify and redistribute the software without constraints. The BSD license is bifurcated into two primary categories: the original BSD license, pioneered by the University of California, Berkeley, and the adapted BSD license, formulated by the FreeBSD project.

The original BSD license incorporates a clause necessitating acknowledgment of the licensed software's usage in promotional materials. Conversely, the modified BSD license omits this stipulation. A chief benefit of the BSD license is its allowance for incorporation of the licensed software into proprietary products without obligating the publication of the produced software under the same license. This flexibility permits entities to include BSD-licensed code in their products without the need to disclose the source code.

Nonetheless, the BSD license's lack of copyleft provisions implies that modifications made to the licensed software can be integrated into proprietary products without the same open-source prerequisites. Thus, while offering significant flexibility, the BSD license might not be optimal for developers aiming to uphold open-source status for their alterations. (4.)

2.2.2 GPL License

The General Public License (GPL), a prevalent open-source software license, governs the use, distribution, and modification of software. Conceived by the Free Software Foundation (FSF) in 1989, it has emerged as a prominent choice for free software distribution. The GPL is constructed to furnish users with the liberty to employ, modify, and disseminate software, while ensuring the perpetuation of these freedoms for future users. As a copyleft license, the GPL permits users to freely manipulate and distribute software, on the condition that any changes or derivatives are also licensed under the GPL. This safeguard guarantees that the software remains open source, even when altered and distributed by others.

The GPL includes provisions mandating access to source code for modified versions, thus promoting transparency and collaboration in the open-source community. Furthermore, the GPL entails clauses prohibiting the use of software in proprietary products or services, mitigating the potential commercial exploitation of open-source software. Overall, the GPL significantly contributes to the advancement and distribution of open-source software, shaping the modern software industry. (5.)

2.2.3 Apache License

The Apache License is a widely employed open-source license within the software development domain. Noteworthy for its permissive nature, this license permits developers to freely use, modify, and distribute software without restrictions. This lenient approach simplifies the integration of open-source elements into software and projects, alleviating concerns about legal complexities or licensing fees.

Renowned for its flexibility, the Apache License fosters collaboration among developers and is compatible with various other open-source licenses. This compatibility facilitates connections between different open-source projects and encourages the creation of novel software. The Apache License's adaptability renders it applicable across diverse contexts, spanning from small personal projects to extensive commercial applications.

Ultimately, the Apache License plays a pivotal role in promoting collaboration, flexibility, and widespread adoption within the open-source landscape. (6.)

3 Moodle LMS

3.1 Moodle Learning Management System

Moodle, an acronym for Modular Object-Oriented Dynamic Learning Environment, emerged in 2002 as an open-source learning management system (LMS) developed by Martin Dougiamas. This web-based platform is harnessed for the creation of online courses and diverse learning activities for educational institutions and businesses alike. At its core, Moodle furnishes educators and instructors with a versatile avenue to construct courses and learning experiences for their students. Simultaneously, it empowers them to monitor progress, offer feedback, and establish an interactive learning environment.

Moodle equips educators with an array of tools tailored to the creation and management of online learning environments. It facilitates the presentation of courses to students in an intuitive and captivating manner, fostering a conducive learning experience. Furthermore, Moodle facilitates collaboration and interaction not only between teachers and students but also among various participants within the learning community.

Adaptability is a hallmark of Moodle, enabling tailoring to the unique requirements of educators. The platform's customization capabilities extend to layout and functionality adjustments to suit the needs of both students and the course itself. For instance, instructors can sequence course materials strategically or incorporate specific widgets to streamline access for students. Additionally, Moodle accommodates the incorporation of plugins and add-ons, allowing comprehensive customization of course aesthetics and functionality. These plugins can be employed to modify the visual aspect of the course page, introduce supplementary features, and even integrate personalized code.

By harnessing Moodle's extensive customization options, educators can create tailored and engaging learning environments that cater to the unique dynamics of their educational objectives and student preferences. (7.)

3.2 Moodle's Platform Infrastructure

Moodle is typically deployed on a foundation that aligns with the conventional LAMP stack, although its nomenclature implies versatility, allowing for alternative modular configurations (see Figure 2). The acronym LAMP signifies the amalgamation of Linux, Apache, MySQL/MariaDB, and PHP.

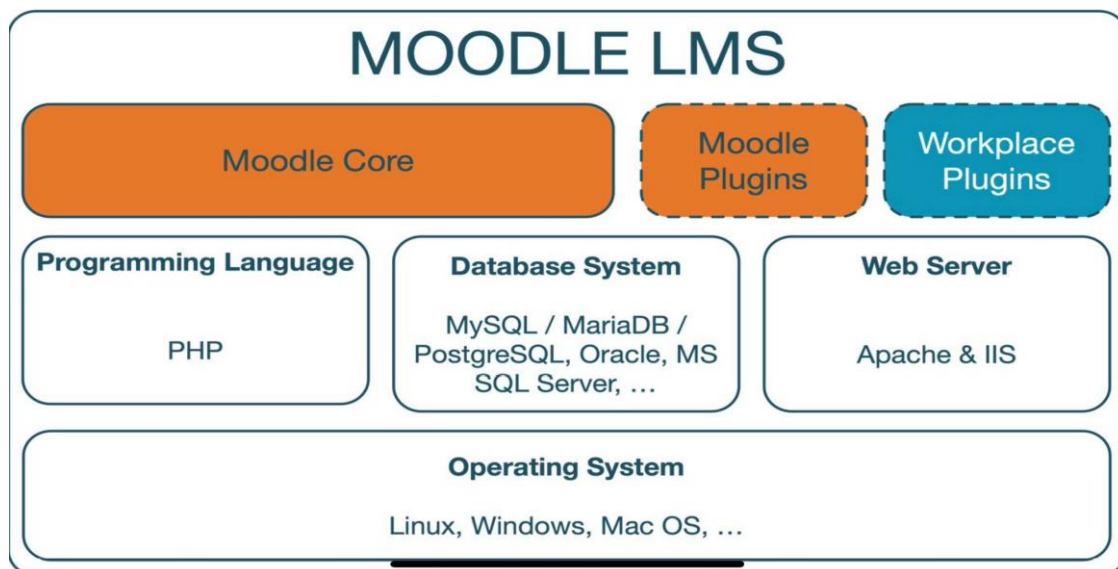


Figure 2. Structure of the Moodle platform

This structural flexibility provides a plethora of deployment possibilities. The essential prerequisites encompass a supportive database background, PHP compatibility, and a web server hosting a website. This adaptability facilitates swift deployment, leveraging existing environment resources. Unquestionably, this attribute plays a pivotal role in driving Moodle's prominence as a favored educational platform. (8.)

3.2.1 Linux: The Versatile Operating System

Linux, an open-source and free operating system, traces its origins to Linus Torvalds' pioneering work at the University of Helsinki in 1991. Born as a response to the limitations and high costs associated with commercial operating systems of that era, Linux emerged as a transformative force in the world of computing. The system's foundational principles were deeply rooted in the Unix operating system, and Torvalds commenced by utilizing the Unix-like Minix project as a foundation. Subsequently, he embarked on a comprehensive overhaul of the Linux kernel, driven by community feedback.

The unveiling of the Linux project in 1992 resonated strongly within the open-source community, propelling its rapid growth. This vibrant community actively contributed by rectifying bugs, introducing enhancements, and introducing novel features. The early phase of Linux's development also witnessed the emergence of diverse distributions or variants of the operating system, such as Slackware and Debian. This proliferation of distributions facilitated tailoring Linux to specific use cases, ultimately spurring its widespread adoption.

In the contemporary landscape, Linux's domain extends across an array of applications, spanning desktop computers, servers, and embedded systems. Its popularity can be attributed, in part, to its open-source nature, fostering collaboration and innovation among developers.

Linux, revered for its versatility, serves as a favored operating system kernel across diverse environments. Its merits render it a prudent choice for myriad applications. Firstly, its open-source nature translates into cost-effectiveness for both businesses and individuals. Linux, in stark contrast to many commercial counterparts, is freely available for use, with its source code open for adaptation and enhancement. This attribute empowers developers and system administrators to tailor the system to their specific requirements.

Secondly, Linux boasts robust security features. Its core components are intricately designed to thwart malicious attacks, safeguarding user data. Thirdly, Linux exemplifies reliability, characterized by stability and sustained operation over extended durations. This makes it an optimal choice for servers and web applications necessitating uninterrupted availability.

Lastly, Linux's compatibility with a diverse spectrum of hardware and software renders it a fitting choice for virtually any environment. Its open-source foundation continues to foster collaboration, innovation, and its enduring influence on the realm of computing. (9.)

3.2.2 Apache: The Dynamic Web Server

Since its inception in 1995, the Apache web server has been a stalwart open-source project under the purview of the Apache Software Foundation. With a robust legacy spanning decades, the Apache web server holds a preeminent position as a web server software, entrusted with the task of hosting websites and web applications.

Renowned for its potency and adaptability, the Apache web server serves as a versatile platform capable of hosting both contemporary web applications and static content. Its configurability is a prominent feature, allowing it to host multiple web applications and offer load balancing in specific configurations. Furthermore, the server's prowess extends to failover mechanisms, enhancing its reliability.

The Apache web server's capabilities encompass not only serving web pages but also diverse content formats such as images, videos, and audio files. Notably, it bolsters security by enabling secure connections to web applications through SSL/TLS protocols, bolstering encryption.

A pivotal facet is the server's capacity to generate dynamic content, pivotal for the creation of interactive web pages and applications. Additionally, scalability, performance optimization, and diverse caching options underscore its capabilities.

The Apache web server garners favor for its reliability and flexibility, attributes that fuel its widespread usage. Its open-source nature ensures accessibility to all, while its continuous updates underscore a vibrant developer community. Noteworthy is its adaptability, tailored to the distinct requirements of various websites. This resilience manifests in its stability, adeptness at handling high request volumes, and compatibility with multiple operating systems including Windows, Unix, and Linux.

The allure of Apache lies in its versatility and scalability. With cross-platform compatibility spanning Windows, Linux, and macOS, it appeals to a wide spectrum of web developers. Embracing multiple programming languages like PHP, Python, and Perl, it empowers developers to craft web applications in languages of their preference.

Strengthening its appeal are Apache's robust security mechanisms. Its modular architecture empowers users to customize security modules, exemplified by mod security which fortifies against common network attacks. Moreover, Apache's access control modules empower administrators to tailor access based on user roles and permissions.

The performance realm is where Apache excels, as witnessed in its Multi-Processing Module (MPM) adeptly handling concurrent requests. Further enhancing speed, caching modules expedite website load times by storing frequently accessed content.

In summation, Apache is an influential web server software epitomizing flexibility, security, and performance, virtues that underpin its enduring relevance in the digital landscape. (10.)

3.2.3 MariaDB: Power and Complexity in Open-Source Databases

MariaDB, an open-source relational database management system, emerges as a formidable contender based on the foundations of MySQL. Crafted in C and C++, its mission encompasses ushering users towards the apex of performance, scalability, and reliability.

At the core of MariaDB's ethos lies the seamless transition from costly commercial databases to the liberating realm of open source.

The allure of MariaDB is underscored by its resonance with both developers and organizations. By encapsulating all the attributes of MySQL, it extends a compelling proposition by integrating advanced features, fortified security, and a realm of extensibility.

A treasure trove of features characterizes MariaDB's offerings, harmoniously coupled with an intuitive user experience. This harmonious blend empowers

developers and database administrators to rapidly sculpt and oversee databases. The innate extensibility of MariaDB further cements its standing, allowing effortless scalability to accommodate diverse application demands. This elasticity renders it a perfect fit for applications with fluctuating workloads.

However, the MariaDB landscape also bears nuances that warrant consideration. The lauded array of advantages is tempered by complexities in implementation and maintenance. While affording an impressive gamut of features, there are intricacies that may pose challenges in real-world deployments. Additionally, MariaDB's security, while commendable, trails behind some other database systems. Instances of data corruption have punctuated MariaDB's history, serving as a reminder of the need for vigilance in its usage.

The verdict on MariaDB rests in a nuanced deliberation of its merits and demerits. While it stands as a testament to the power of open-source innovation, pragmatic evaluation is indispensable before embarking on its adoption journey. (11.)

3.2.4 MariaDB vs. MySQL

In the realm of open-source relational database management systems (RDBMS), MariaDB and MySQL stand as titans, both wielding substantial influence in the landscape of web applications. Conceived in 1995 by Michael Widenius, MySQL claimed its place in history, followed later by the emergence of MariaDB in 2009 under the same visionary's stewardship, post his departure from the MySQL development fold. Unified by their allegiance to Structured Query Language (SQL) and compatibility with diverse operating systems, the two-systems command attention for their distinct virtues.

In the realm of performance, both contenders exhibit commendable scalability and data processing prowess. MariaDB, however, boasts an edge in select benchmarks, particularly when confronted with the handling of voluminous datasets. This supremacy intertwines with an expansive feature set, wherein MariaDB shines bright in the domains of security fortification and replication enhancements. These attributes coalesce to cement its stature as a go-to choice for enterprise-grade

applications. Not to be outshone, MySQL maintains its standing as the favored option for specific applications, especially those that demand harmonious integration with third-party software.

Anchored by vibrant communities, both MariaDB and MySQL furnish robust support ecosystems, replete with documentation and updates, fostering an environment conducive to growth. Central to their divergence lies the genesis of MariaDB, born as a response to Oracle's acquisition of Sun Microsystems, custodian of MySQL. The licensing paradigm sets them apart - MariaDB is governed by the GNU General Public License (GPL), whereas MySQL dons a hybrid cloak of proprietary and open-source licenses.

This contrast extends to features, where MariaDB introduces innovations like virtual columns, empowering users to forge columns calculated on the fly from existing data. Performance optimization emerges as another key facet, exemplified by MariaDB's thread booker extension, a boon for managing high-concurrency workloads. Yet, MySQL wields its own aegis of strengths, characterized by an expansive user base and a flourishing ecosystem of tools and plugins.

In a narrative mosaic, MariaDB and MySQL converge and diverge, each weaving its own narrative of licensing, features, and performance strides. Their legacy endures as favored beacons in the labyrinth of relational database management systems, a testament to the diversity and vibrancy of open-source innovation. (11.)
(12.)

3.2.5 PHP: Empowering Dynamic Web Development

PHP, a venerable programming language, stands as the cornerstone of choice for web developers engrossed in crafting dynamic, interactive websites, and web applications. Renowned for its steadfast reliability, PHP's mantle is further adorned with a plethora of online resources, catering to the curious and the aspiring. Its lauded extensibility renders it a canvas for innovation, inviting developers to

embellish their creations with libraries, frameworks, and plugins. (13.)

Emerging from this fertile landscape are PHP frameworks, veritable stars that have ascended in recent years. The allure of these frameworks lies in their capacity to streamline the often-intricate process of web application development and website construction. Laazir et al. (2019) attest to their value, underscoring how these frameworks unfurl an arsenal of tools, libraries, and components, primed for facile reuse. This fortifies the development journey, engendering both swiftness and error mitigation.

A pivotal feat is the scaffolding these frameworks offer for code organization and management, bestowing developers with an avenue for facile modifications and upkeep. The security tapestry is interwoven with features like input validation, authentication, and encryption, fortifying the defense of web applications. Scalability and adaptability are steadfast companions, enabling developers to adroitly infuse new features into existing applications, deftly aligning with the ebb and flow of technology's tides.

An alluring facet is their adeptness at harmonizing with diverse web technologies, affording developers the luxury of embedding multifarious features into their creations. The crux of their popularity resides in this confluence, seamlessly enabling the realization of web applications teeming with diverse functionalities.

In summation, the ascendance of PHP frameworks is underpinned by their prowess in demystifying the labyrinthine path of web application development. These frameworks furnish a structural scaffold for code orchestration, fortify security postures, and kindle harmonious integration with an array of web technologies. Thus, the enigmatic dance of keeping pace with technology's evolution and user expectations is rendered more attainable, celebrating the marriage of innovation and pragmatism.

The Role of Open-Source Software in Moodle's Ecosystem Within the realm of Moodle's ecosystem, the bedrock of open-source software assumes a pivotal role,

underpinning its structure and functionality. Moodle, as a web-based learning management system, harnesses the power of open-source software to realize its mission of delivering flexible and accessible education.

The open-source nature of Moodle not only aligns with its ethos of collaboration and accessibility but also empowers institutions and educators with the ability to tailor the platform to their specific needs. By allowing developers worldwide to contribute to its evolution, Moodle harnesses the collective intelligence of the open-source community to continuously enhance its features and capabilities.

Open-source software's transparency and accountability shine in Moodle's realm, where errors and security vulnerabilities are swiftly identified and rectified by the global developer community. This iterative process not only ensures a safer environment for users but also cultivates a culture of continuous improvement.

Moreover, the democratization of technology, a hallmark of open-source principles, is epitomized in Moodle's deployment. Educational institutions, irrespective of their size or financial constraints, can harness the power of Moodle to offer high-quality online courses. The removal of financial barriers through open source not only fosters equitable access to education but also bolsters the global educational landscape.

In conclusion, the symbiotic relationship between Moodle and open-source software is a testament to the transformative potential of collaborative development. This synergy not only fuels the evolution of a robust learning management system but also embodies the ethos of openness, innovation, and accessibility that define the essence of both Moodle and the broader open-source movement.

4 Vue.js: Empowering Interactive User Interfaces

4.1 Overview of Vue.js

Vue.js, an innovative and progressive JavaScript framework introduced in 2014, has emerged as a frontrunner in the dynamic world of web development. This open-source framework provides developers with a flexible canvas to craft intuitive user interfaces, ushering in a new era of interactive web experiences. Let's delve into Vue.js's features, juxtapose it with its JavaScript counterparts, and explore its real-world applications across diverse domains.

At its core, Vue.js employs a reactivity system that seamlessly updates data within HTML elements bound to Vue objects. The result is a dynamic and interactive user interface, where data changes trigger automatic updates without necessitating page reloads. Simplicity reigns as one of Vue.js's defining features, allowing developers to grasp and wield its capabilities effortlessly. Furthermore, Vue.js's compatibility with other libraries and frameworks underscores its versatility, making it an ideal ally for building intricate applications laden with rich functionalities.

Vue.js's allure lies in its efficiency, performance, and scalability. The framework champions swift development cycles, fortified performance, and scalability – cornerstones of modern web development. A thriving community rallies around Vue.js, furnishing developers with a wealth of resources and peer support that enhances their proficiency.

Compared to its JavaScript counterparts, Vue.js stands distinguished by its gentle learning curve and impeccable documentation. Its slender file size translates into

quicker load times and robust performance, fortified by a reactivity system that streamlines data manipulation. However, Vue.js's community support may pale in comparison to React and Angular, which boast broader ecosystems of resources, tools, and libraries. Nonetheless, Vue.js remains a formidable contender, buoyed by its performance prowess and user-friendly nature.

Vue.js's ascendancy is exemplified by its real-world applications across diverse sectors. Majestic enterprises like Alibaba, Xiaomi, and WeChat have harnessed Vue.js's potential to elevate user experiences. Alibaba's mobile app witnessed enhanced performance and rapid load times, courtesy of Vue.js. Xiaomi's Mi Home application, a haven for controlling smart devices, reaped the rewards of Vue.js's smooth user interactions and heightened performance. The venerable WeChat, a prominent Chinese messaging platform, embraced Vue.js to bolster its web version, culminating in sprightly performance and accelerated load times.

In sum, Vue.js's meteoric rise stems from its tangible impact across industries and its ability to reshape user interfaces. Vue.js transcends the ordinary, enabling developers to orchestrate captivating and responsive user experiences, amplifying its significance in the ever-evolving landscape of web development.

4.1.1 Vue.js vs. Angular: Choosing the Right Path

In the dynamic landscape of JavaScript frameworks, Vue.js and Angular emerge as key players for crafting interactive web applications. These frameworks cater to distinct needs and project scopes, leading to discussions among developers about which one to wield. Let's delve into the realm of Vue.js and Angular, dissecting their nuances and exploring their fit for different projects.

Vue.js, hailed as a progressive framework, embraces a gradual approach to deployment, allowing seamless integration into existing projects. Angular, in contrast, embodies a comprehensive ecosystem, revered for constructing sizable enterprise applications. According to Novac et al. (2021), Vue.js shines in its

simplicity and user-friendly design, making it a go-to choice for developing small to medium-sized applications. Its intuitive API empowers developers to fashion reusable components with ease. Angular, on the other hand, demands a steeper learning curve while offering an array of advanced features. It equips developers with tools such as dependency injection, reactive programming, and a potent template engine, ideal for robust applications of grand scale.

As Novac et al. (2021) aptly observe, "Vue.js is easier to learn and use for small and medium-sized projects, while Angular is better suited for large projects with complex data and logic requirements." Vue.js boasts swift installation and minimal configuration, making it an ally for swift prototyping and compact endeavors. Conversely, Angular mandates deeper setup and configuration, affording a resilient and scalable foundation for formidable applications.

The choice between Vue.js and Angular pivots on project specifics. Vue.js, with its simplicity and agility, befits smaller ventures seeking interactive interfaces. In contrast, Angular's robustness and scalability cater to the demands of extensive enterprise applications. Developers, armed with insights into project requirements, development experience, and team expertise, can discern the framework best aligned with their goals.

To provide a glimpse, here is a simple example code snippet from Angular showcasing the timeless "Hello World" tradition:

```
``typescript
import { Component } from '@angular/core';

@Component({
  selector: 'my-app',
  template: `

# Hello {{name}}</h1>`, }) export class AppComponent { name = 'World';


```

```
}  
...
```

In conclusion, Vue.js and Angular, each armed with distinctive attributes, illuminate different avenues for web application development. The framework of choice hinges on project size, complexity, and the developer's comfort zone, driving the quest for an optimal solution in the realm of JavaScript frameworks.

4.1.2 Vue.js vs. React: Navigating the Frontend Landscape

In the realm of frontend development, Vue.js and React emerge as key players, each wielding its unique strengths. Let's embark on a journey to dissect the essence of these frameworks, highlighting their attributes and guiding developers through their distinctive paths.

React, a JavaScript library birthed by Facebook in 2011, stands as a powerhouse for constructing user interfaces. Renowned for its exceptional performance and adaptability, React reigns supreme in larger projects. A shared cornerstone of Vue.js and React lies in their utilization of a virtual DOM, expediting UI changes. However, the distinction unfolds in their syntax approach - Vue.js embraces a template-based syntax, while React leverages JavaScript-based syntax. Vue.js garners frequent comparison to Angular and React, standing out for its simplicity and user-friendliness. React, on the other hand, thrives on its prowess to sculpt intricate applications without compromising performance. The choice between Vue.js and React rests upon the developer's needs and the project's specificities.

As we delve deeper, we unearth notable divergences. Vue.js unfurls as a progressive, incremental framework, spotlighting the view layer. In contrast, React showcases its mettle as a component-centric library for crafting UIs.

Vue.js extends a friendly hand to newcomers, boasting a shallow learning curve and a concise, intuitive syntax. React, however, demands a more profound understanding of JavaScript and its syntax. Nevertheless, React entices with its component-based model that fosters code reuse and the creation of intricate user interfaces. Augmented by the virtual DOM, React amplifies rendering efficiency and elevates performance.

Though crowned with virtues, React carries some burdens - necessitating additional libraries for state management and routing. Vue.js, on the other hand, embeds these features intrinsically, ushering convenience into the development journey. Yet, Vue.js' community, while vibrant, may lack the sprawling expanse of React's, potentially affecting resource availability and support.

In culmination, Vue.js and React each hold their realm of pros and cons, leaving developers at a crossroads of preference and project prerequisites.

As a glimpse into React, here's a modest code snippet embracing the venerable "Hello World" tradition:

```
``jsx
import React from 'react';
import './App.css';




function App() {
  return (
    <h1>Hello World!</h1>
  );
}

export default App;
...

```

In the ever-evolving landscape of frontend development, Vue.js and React beckon

developers with their prowess and promise, inviting exploration and adaptation to the diverse terrains of web crafting.

	 Angular	 React	 Vue
Framework size	143k	97.5k	58.8k
Programming Lang	Typescript	Javascript	Javascript
Ui component	In-built material techstack	React UI tools	Component libraries
Architecture	component-based	component-based	component-based
Learning curve	steep	moderate	moderate
Syntax	Real DOM	Virtual DOM	Virtual DOM
Scalability	modular development structure	component-based approach	template-based syntax
Migrations	API upgrade	React codemod script	Migration helper tool

 Angular Minds

Table 1 Features of Angular, React and Vue

4.2 Node.js: Empowering Server-Side Application Development

Nestled at the heart of server-side web application development, Node.js emerges as a versatile, cross-platform JavaScript environment and library.

Conceived by Ryan Dahl in 2009, Node.js has evolved to its latest iteration, version 19.8.1, unveiled in March 2023. This dynamic ecosystem enables developers to fashion robust server-side applications, making it a prime choice for data-intensive endeavors due to its adeptness in asynchronous event-driven paradigms.

The allure of Node.js is undeniable, beckoning developers with its array of benefits for diverse server-side applications:

Harnessing the Power of V8 Engine: Enveloped within Node.js lies the foundation of Google Chrome's V8 engine. This synergy confers unparalleled performance and functionality to applications, propelling them to exude swiftness and agility.

Abundance in Package Management: The repository of Node.js packages, encapsulated in its Package Management (NPM), burgeons with over 50,000 offerings. This trove empowers developers to seamlessly infuse desired functionalities into their applications, saving precious time and effort.

Synchronized Code Execution: With Node.js orchestrating the ballet between end-users and servers, the harmonious synchronization of code ushers in reduced video and audio load times. This synchronicity augments user experiences by minimizing latency.

Gateway to Open-Source Efficacy: Seated within the realm of Node.js, the terrain of open-source beckons. Harnessing the power of a JavaScript framework in an open-source ecosystem translates to collaboration, innovation, and resource-sharing.

Intricately intertwined with the aforementioned attributes, Node.js carves a unique niche in the technology landscape, bridging creativity and efficiency, and unifying developers in the pursuit of groundbreaking server-side applications.

<https://www.infowindtech.com/what-is-node-js-and-why-you-should-use-it/>

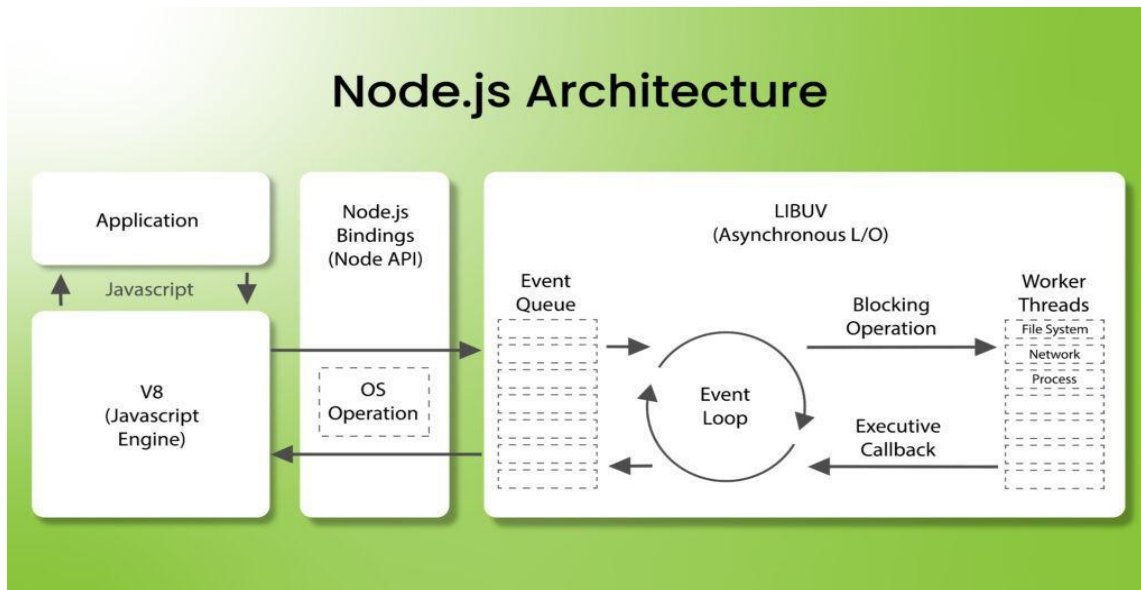


Figure 3 Structure of Node.js (Kaneriya: 2022)

Node.js Architecture and its Applicability

Node.js operates within the confines of a single thread, an architectural approach that enables the concurrent processing of a multitude of event calls. This design allows Node.js to seamlessly handle thousands of simultaneous events. (14.)

4.3 JavaScript: A Dynamic Web Enabler

JavaScript, a versatile programming language, takes center stage in crafting dynamic web pages that elevate user experiences across diverse browsers and devices. Its role extends to creating captivating elements like drop-down menus, animations, and ever-changing background colors, enhancing engagement and interaction.

The journey of JavaScript's evolution unfolds with its origin as an internal tool, later entrusted to the ECMA international standards organization in 1995 by Netscape and Brendan Eich. This marked the birth of the language's development under the TC39 technical committee. The inaugural release, named ECMA-262 or ES1, emerged on June 1, 1997, gaining support from IE4 as the first browser to adopt ES1.

JavaScript, a versatile programming language, takes center stage in crafting dynamic web pages that elevate user experiences across diverse browsers and devices. Its role extends to creating captivating elements like drop-down menus, animations, and ever-changing background colors, enhancing engagement and interaction.

4.4 TypeScript: Elevating JavaScript Development

TypeScript emerges as a formidable programming language, capturing the attention of software developers and earning its place in the web development realm. This language project, birthed within the confines of Microsoft, transitioned to the public domain in 2012, opening doors to a broader community. TypeScript stands as a refined iteration of JavaScript, enhancing its capabilities by introducing novel features and functionalities.

Nurtured as a statically typed derivative of JavaScript, TypeScript assumes a pivotal role in enhancing the creation of extensive applications. A defining attribute of TypeScript lies in its robust support for type annotations. These annotations empower developers to articulate data types, function parameters, and return values of variables. This preemptive error detection mechanism identifies issues during compile time, rather than runtime, cultivating a more dependable codebase. This quality control measure stands as a cornerstone for building resilient applications.

Equally pivotal, TypeScript's integration with Integrated Development Environments (IDEs) nurtures efficient development workflows. The realm of tools encompasses code completion, refactoring, and seamless navigation. These attributes collectively heighten developer productivity and foster an environment conducive to innovation.

TypeScript's feature repertoire extends beyond type annotations. It incorporates classes, interfaces, and modules that encourage modular and scalable coding

practices. This aligns with contemporary development paradigms, simplifying the construction and maintenance of intricate projects. Notably, TypeScript seamlessly integrates with acclaimed JavaScript frameworks such as Vue.js, React, and Angular, bridging the gap between legacy codebases and modern development.

In a broader context, TypeScript and JavaScript coexist, each bearing its unique strengths and applications. JavaScript, a cornerstone of web development, serves as a dynamic scripting language catering to interactive web pages and applications. TypeScript, however, doesn't seek to supplant JavaScript; rather, it enriches it. The key distinction lies in their approach to type checking. JavaScript employs runtime dynamic type checking, potentially resulting in challenging-to-detect errors. TypeScript, in contrast, performs static type checking during compile time, enabling preemptive error detection before code execution. Syntax nuances further delineate the two languages. JavaScript exhibits C-like syntax, while TypeScript aligns closer to Java or C#. TypeScript introduces additional features, including interfaces, classes, and modules, enriching code organization and maintainability. With its robust error detection mechanisms and developer-friendly tools, TypeScript cultivates a conducive environment for application development, empowering developers to craft dependable and efficient solutions.

In essence, TypeScript stands as a testament to the evolution of programming languages, catering to modern development challenges and aligning with a changing landscape. By recognizing the nuances that set TypeScript and JavaScript apart, developers can harness the power of both languages, creating robust, user-centric digital experiences.

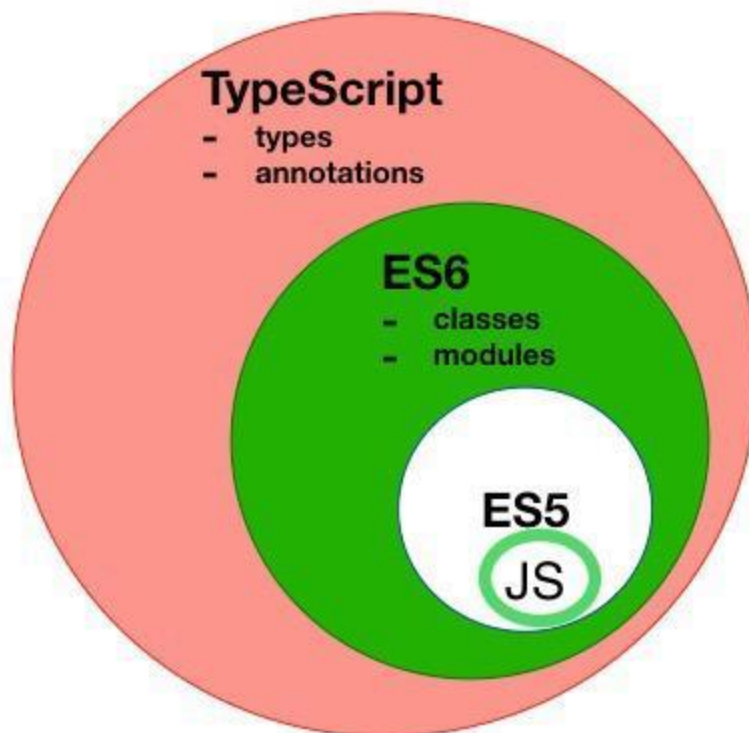


Figure 4 Description of the feature difference between TypeScript and JavaScript

4.5 ESLint: Elevating Code Quality through Analysis

ESLint emerges as a steadfast ally in the world of programming, championing the cause of code consistency and error-free scripting in JavaScript and TypeScript programs. Born from the collaborative efforts of the open-source community in 2013, this static code analysis tool assumes the role of a vigilant guardian, diligently identifying and rectifying code discrepancies.

Imbued with a mission to enhance the quality of JavaScript and TypeScript codebases, ESLint operates by imposing unwavering coding standards and unveiling latent bugs and errors. Akin to an expert detective, it scrutinizes code syntax, detecting telltale patterns linked with common coding blunders. Armed with this knowledge, it promptly furnishes developers with insightful feedback, empowering them to navigate their codebase with newfound precision.

The strength of ESLint lies in its adaptability. Through a dynamic interplay of rules and extensions, this tool can be molded to the unique contours of each project. Project-specific configurations can be seamlessly integrated through configuration files or even orchestrated via the command line interface. This adaptability grants developers the power to calibrate ESLint according to the project's specific requirements and the team's coding ethos.

ESLint has grown to prominence within the JavaScript community, owing much of its acclaim to its innate flexibility and user-friendly nature. This tool's prowess doesn't just lie in its formidable error- detection capabilities; it extends to its seamless integration into the developer's workflow. It stands as a testament to the power of collective collaboration, harnessing the strength of a diverse community of developers to fine-tune and enhance the software development experience.

In a landscape where code quality is paramount, ESLint takes on the role of an indispensable partner, ensuring that code remains robust, consistent, and free from lurking errors. As the digital landscape continues to evolve, ESLint stands firm as a symbol of the industry's commitment to excellence in coding practices, a beacon guiding developers toward the path of code perfection. (15.)

4.6 HTML: Crafting the Web Canva

HTML, the abbreviation for HyperText Markup Language, stands tall as the cornerstone of web page creation. Acting as a descriptive language, it facilitates the construction of intricate web sections, seamless links, and engaging paragraphs. HTML achieves this through the orchestration of elements like tags and attributes, which lend structure and coherence to the web's visual and functional tapestry.

It's essential to note that HTML isn't a programming language per se; rather, it takes on the role of a robust web standard. It excels at laying the groundwork for web content but lacks the dynamic prowess of traditional programming languages. It's the scaffolding upon which web pages are built—a tapestry of interconnected documents that collectively form the online experience. Each web page, whether a contact form or homepage, is molded from distinct HTML files.

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
<title>My test page</title>
</head>
<body>

</body>
</html>
```

Example code 3 Basic HTML code

HTML's journey spans three decades, evolving to meet the ever-increasing demands of web development. Today, the mantle is carried by HTML5, which brings native support for dynamic elements such as video playback. The evolution began with HTML 1.0 in 1993—a rudimentary version featuring a handful of tags and attributes. Subsequent versions built on this foundation, with HTML 2.0 in 1995 introducing more features like tables, image maps, and forms. HTML 3.2 arrived in 1997 with style sheet support, enhanced table formatting, and refined form elements.

HTML 4.01, launched in 1999, further enriched the landscape with features like frames and multimedia, alongside stringent coding standards.

The journey culminated in HTML 5 in 2014—an embodiment of efficiency and adaptability for contemporary web development. This version ushered in new multimedia and form element tags, along with enhanced support for mobile devices and offline applications.

HTML's influence isn't limited to traditional web pages alone. Its legacy is seen in HTML-based emails and even in the structural underpinnings of note-taking applications. Over the years, HTML has morphed into a multifaceted tool, empowering developers to create websites that are not just visually appealing but also interactive, intuitive, and attuned to modern web sensibilities.

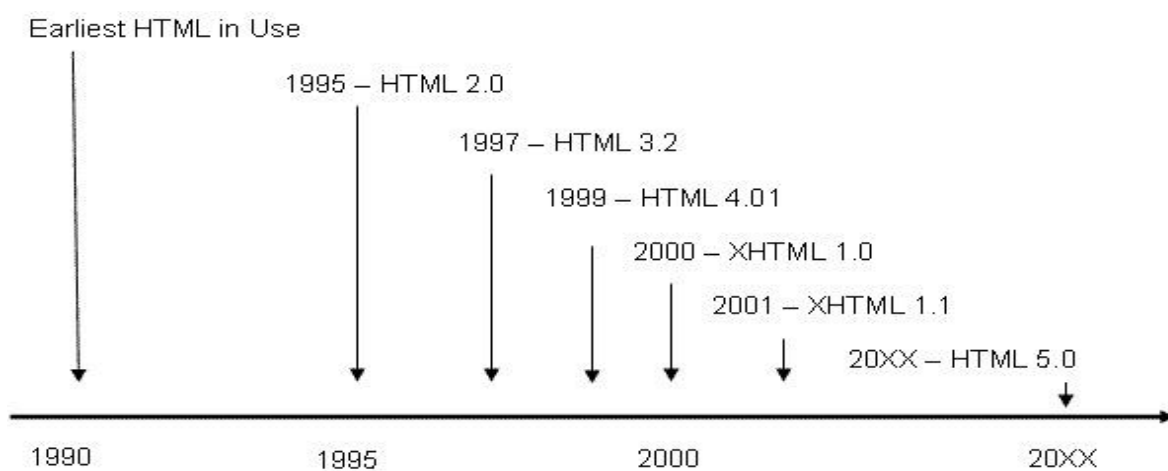


Figure 5 History of HTML

Unveiling HTML: The Framework of Web Presentation

An HTML file boasts an extension of either .html or .htm, setting the stage for web browsers to unfurl its contents for every site visitor. This universal accessibility is HTML's power—bridging the gap between creators and consumers.

At the core of an HTML page reside its elements—wielding attributes and tags as their building blocks. Tags, akin to architectural signposts, delineate where an element commences and concludes. Attributes, on the other hand, embellish elements with properties, bestowing distinct characteristics. At the heart of every attribute lies its two-fold identity: a name that beckons supplementary information and a value that imparts context.

Crucially, the class attribute commands a pivotal role in the realm of programming. Bestowing style information, it can cascade across multiple elements bearing the same class value. While most elements boast both start and end tags, not all adhere to this rule. Empty elements, for instance, shun end tags and content, thriving as self-contained entities.

```

```

Example code 4 This tag has two attributes, the src attribute, which is the path of the image, and the alt attribute, which tells about the image if it cannot be rendered or if a screen reader is used. However, there is no end tag or content here. (16.)

4.7 Unveiling the Artistry of CSS: Crafting Visual Magic

CSS is a realm waiting to be unraveled, a companion to HTML once the latter is grasped. Born from the ingenuity of Hakon Wium Lie in 1994, CSS, or Cascading Style Sheets, emerged as an artistic medium for the web. Wium Lie's collaboration with HTML luminary Tim Berners-Lee at CERN marked the genesis of this innovative venture.

Conceived to lend allure to web design, CSS offered the remedy sought by HTML enthusiasts of the time. It seamlessly harmonized with HTML 4.01, ushering in an era where web pages could be bedecked with aesthetics and visual splendor.

Sailing under the aegis of the W3C, CSS enjoys standardization akin to HTML, nurtured by the same custodian of web norms. This open-source and independent standard can be paired seamlessly with HTML, forging a formidable duo in the realm of web design. Tracing the evolution of CSS reveals a trinity of updates.

CSS Level 1, debuting in 1996, made its encore in 1999, sporting enhancements. CSS Level 2, or CSS2, stepped onto the stage in 1998, flaunting newfound media

support—ushering elements beyond the confines of the screen. The torchbearer of this lineage is CSS Level 3, reigning supreme since its release, the pinnacle of this artistic journey.

```
```css
body {
background-color: light gray;
}

h1 {
color: green;
text-align: center;
}

p {
font-family: sans-serif; font-size: 16px;
}
```
```

Example code 5 CSS code

CSS1: Laying the Foundation

Debuting in 1996, CSS1 marked the inception of this style sheet saga. It introduced font manipulation, enlivening text with colors and backgrounds. The orchestration of text alignment, padding, positioning, and classification added depth to its repertoire. Though now retired, its legacy lingers as a testament to CSS's humble beginnings.

CSS2: Expanding Horizons

CSS2 followed in 1998, a richer sequel brimming with functionality. Beyond the basics, it introduced relative, absolute, and fixed positioning—granting designers new tools to shape layouts.

Embracing media types and bi-directional text support, CSS2 heralded a more comprehensive design experience.

CSS3: Ascending Artistry

Reigning since 1999, CSS3 unfolds as a magnum opus of web adornment. Offering an array of typefaces, including the treasury of Google and Typecast, it's modularized for ease of use and efficiency. A cornerstone of modern web design, CSS3 dances in synergy with HTML5, gracing the digital landscape with unparalleled visual craftsmanship.

CSS3

Taxonomy & Status on January 20 2013

- W3C Recommendation
- Candidate Recommendation
- Last Call
- Working Draft

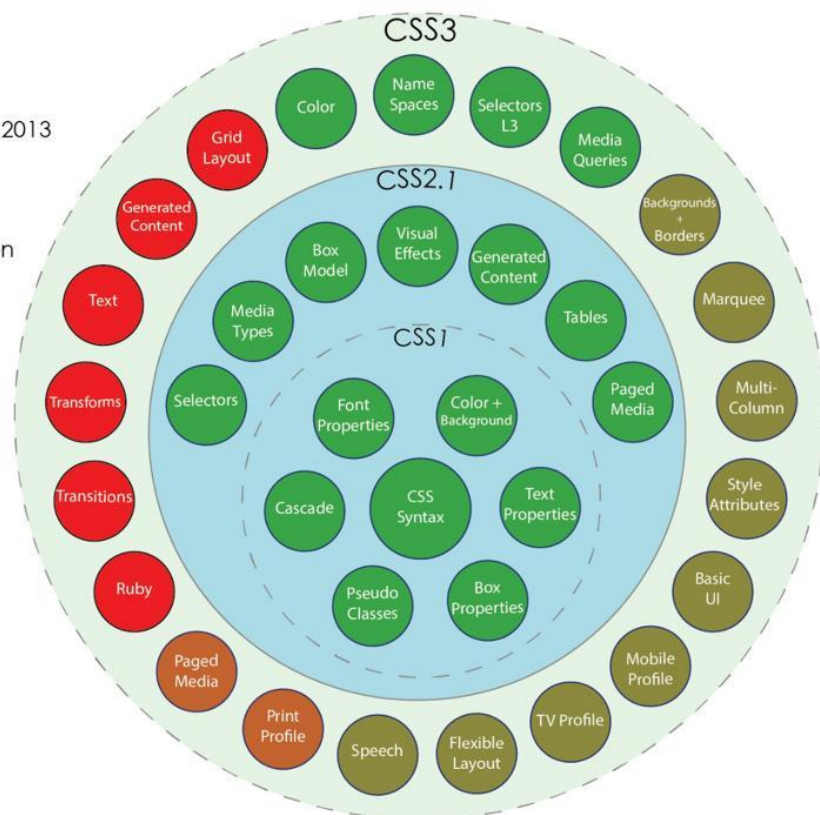


Figure 6 Description of CSS fonts

Emerging Horizons: Envisioning the Future of CSS

In yesteryears, websites stood as mere repositories of information, bereft of visual allure. However, with the advent of CSS, the realm of web design underwent a metamorphosis. This evolution has been nothing short of remarkable, transforming mundane web pages into captivating digital landscapes that beckon users with aesthetic charm.

CSS, the architect of this visual renaissance, has spurred web design experts to elevate their craft to new heights. Without the brush strokes of CSS styling, the canvas of a website would remain devoid of the colors and textures that captivate users' senses. The symbiotic partnership between HTML and CSS has bestowed upon the digital realm an artistic vitality that resonates with users.

As we reflect on the journey thus far, it's evident that CSS has not been static. Numerous updates and revisions have paved the path to its current iteration. With each stride forward, CSS has evolved, adapting to the shifting sands of technology and design trends. And just as the past was marked by transformative shifts, the future holds the promise of even more intriguing updates.

While we stand on the threshold of the unknown, it's certain that the journey of CSS is far from over. As technology advances and design sensibilities evolve, we can anticipate that CSS will continue to evolve in tandem. Each iteration will undoubtedly bring forth innovative features and functionalities, ensuring that the realm of web design remains dynamic and captivating.

In the current landscape, CSS3 reigns as the latest testament to this ongoing evolution. It stands as a snapshot of where we are today—a manifestation of the artistic finesse and technological prowess that CSS embodies. And while we embrace the marvels of CSS3, we can rest assured that the journey continues, with the canvas of CSS awaiting the strokes of ingenuity and inspiration that lie ahead.(17.)

5 Creating a Vue.js Basics Course

Commissioned by Metropolia University of Applied Sciences, our mission was to develop a comprehensive Vue.js basics course within Metropolia's Moodle environment. This engaging course takes students on a journey from theoretical understanding to hands-on coding exercises, with a particular emphasis on Vue.js 3 due to the impending end of support for Vue.js 2 on December 31, 2023.

Navigating this transition presented minor challenges, primarily due to differences in documentation levels between the two versions. However, our goal was to provide students with a seamless learning experience. To achieve this, we incorporated a 40-question multiple-choice test as an integral part of the course. This assessment covers various modules and is complemented by relevant links for students to explore and gather additional insights.

Our main source of information and reference for course material is the scientific thesis authored by Tomi Salmi and Jesse Veijalainen, titled "Vue.js course for Moodle." Their work forms the foundation of the content presented in this course, ensuring its quality and accuracy.

https://www.theseus.fi/bitstream/handle/10024/793493/Salmi_Veijalainen.pdf?sequence=2&isAllowed=y

5.1 Module 1: Unveiling Vue.js

The inaugural module serves as an introduction to the Vue.js framework. It delves into Vue's origin story, its creators, purpose, and development timeline. Today, Vue.js takes center stage as a leading contemporary web application framework. The module delves into comparative analyses of React, Node, and Angular, outlining their unique attributes, use cases, and technological nuances.

The heart of this module delves deeper into the intricacies of Vue.js. Students unravel the framework's underlying architecture, exploring the libraries it employs. The focal point is mastering the essentials of Vue.js, including setting up Node.js and essential developer tools. We guide students through their maiden project creation—a foundational skill for their journey ahead.

Furthermore, students gain insights into invaluable add-ons and alternative IDE options. Microsoft's Visual Studio Code emerged as our preferred IDE due to its open-source nature and comprehensive feature set.

5.1.1 Visual Studio Code: A Versatile Coding Companion

Visual Studio Code, often abbreviated as VS Code, stands as a paramount source code editor and an iconic tool in a developer's arsenal. Originating from Microsoft's 2015 Build conference, VS Code's prominence continues to surge. Supported across diverse operating systems, including MacOS, Linux, and Windows, its accessibility and utility are further bolstered by active maintenance and a thriving user base.

Unveiled on GitHub under the MIT license, VS Code initially merged TypeScript and JavaScript, later focusing solely on TypeScript. Evolving with enhanced tool support like static type checking and code refactoring, it accommodates an expanding roster of programming languages. Architecturally, it seamlessly blends web, native, and language-specific technologies, harnessing the power of Electron, JavaScript, and

Node.js. Its debugging prowess and user-friendly interface cement its status as a programming gem.

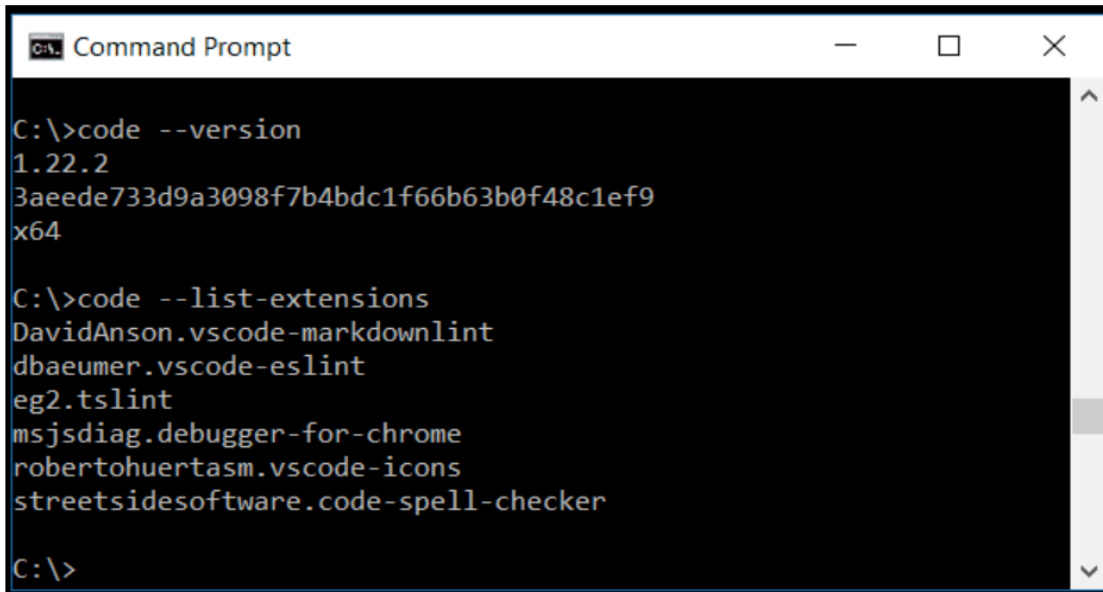
In essence, this course spearheads an immersive exploration of Vue.js while familiarizing students with invaluable tools like Visual Studio Code, ensuring a holistic and impactful learning journey.

5.1.2 Exploring Visual Studio Code's Key Features

Visual Studio Code boasts an array of invaluable features, a few of which I'll introduce here: the command line interface, the status bar, debugging capabilities, and seamless git integration.

Command Line Mastery

Embracing a robust command-line interface, Visual Studio Code empowers users to wield unparalleled control over the editor's operations. This interface facilitates diverse tasks such as launching the editor, opening various files, installing extensions, and altering display languages during startup. Moreover, it aids in diagnostics and even extends its reach to establish remote access tunnels, enhancing the editor's versatility.



```
C:\>code --version
1.22.2
3aeede733d9a3098f7b4bdc1f66b63b0f48c1ef9
x64

C:\>code --list-extensions
DavidAnson.vscode-markdownlint
dbaeumer.vscode-eslint
eg2.tslint
msjsdiag.debugger-for-chrome
robertohuertasm.vscode-icons
streetsidesoftware.code-spell-checker

C:\>
```

Figure 7 :Using Visual Studio Code on the command line

Status Bar: Your Command Center

The status bar, a pivotal component of Visual Studio Code, graces the bottom of the desktop, furnishing users with essential project-related insights and actions. Divided into two distinct factions, the primary resides on the left, showcasing project-wide warnings and issues. On the right, contextual items take the stage, encompassing language specifics, spacing, and feedback, further enriching the user experience.



Figure 8 : VSCode status bar](image-link-status-bar)

Effortless Git Integration

Built-in git support within Visual Studio Code streamlines version control, with a prerequisite of git version 2.0.0 or higher. This integration encompasses a plethora of benefits, including repository initialization, cloning, tag creation, code alterations, code pushing, seamless synchronization with repositories, and efficient resolution of merge conflicts. Additionally, the status bar provides real-time insight into the repository's state.

Debugging: Your Trusty Ally

Debugging, the art of error rectification, emboldens programmers by pinpointing and eliminating code anomalies. Ranging from minor typos to syntax errors, debugging serves as a powerful aid in code correction. Visual Studio Code's intuitive debugging tools facilitate this process, rendering error identification and resolution seamless and approachable.



Figure 9 : Visual Studio Code's Debug bar

In summary, Visual Studio Code's multifaceted toolkit caters to every facet of the coding journey, from efficient command-line control to real-time project insights, streamlined version control, and robust debugging.

5.2 Module 2: Delving into Vue.js Basics

In this module, students embark on their Vue.js journey by venturing into the traditional "hello world" project, a stepping-stone into the framework. They also kickstart the entire course project, gaining insight into creating components and integrating them into their project. The lecture revisits the fundamentals of HTML and JavaScript, setting the groundwork for what lies ahead.

Furthermore, students explore the art of listing and the utilization of props. They grasp the significance of props within a project and their role in passing data between components. The lecture equips them with the knowledge of implementing a running unique ID and its integration into the project. Armed with this newfound understanding, students acquire a fundamental grasp of Vue.js' props, list rendering, and unique ID integration.

5.3 Module 3: Unveiling Components and Rendering

Building upon the foundation established in the previous lecture, this module dives into the realm of multiple components. Students uncover the art of adding and showcasing these components within a Vue.app. The lecture introduces the concept of rendering and sheds light on the potential for creating loops within Vue, illuminating their significance in application development. Students are exposed to the architecture and structure of Vue applications, as well as the pivotal role played by the key attribute. The module culminates in the art of code refactoring, enhancing code organization.

5.4 Module 4: Extending Functionality and Interaction

Module 4 revolves around empowering students to add, manage, and manipulate tasks within a list. By comprehending the mechanics of sending, processing, and managing lists, students acquire the ability to enhance the application's functionality. The lecture navigates through the intricacies of components, offering insight into a new method. Furthermore, students gain mastery over data processing, delve into the v-model, and explore modification methods, equipping them with dynamic control over their application's behavior.

5.5 Module 5: Harnessing CSS and Webpack

This module embraces the realm of Cascading Style Sheets (CSS). Students delve into CSS application, mastering the art of leveraging webpack to manage diverse files. The lecture covers the utilization of CSS preprocessors and postprocessors, along with integrating CSS files into existing applications. Students navigate through global identifiers, augmenting their understanding of their implications. Furthermore, the module delves into extended CSS file modification, rendering graphics to suit end-user preferences.

5.6 Module 6: Unveiling Computational Features

Module 6 tackles the intricacies of computational features and counter utilization within a Vue application. The module imparts an understanding of counter functionality and its implications, along with the challenges it poses in larger applications. Students dive into the dynamics of page rendering upon counter changes, utilizing console logging to unravel the process. The module culminates in grasping model/template manipulation using the calculator and mastering the creation and management of diverse events.

2 / 3 tehtävää suoritettu

Figure 10: Using the Application Calculator

5.7 Module 7: Mastering Conditional Rendering

This module explores Vue's conditional rendering capabilities, equipping students with the power to manipulate components' display conditionally using v-if and v-else features. Students navigate the realm of functions, understanding their application and utility within varying contexts. The module deepens their understanding of value alterations, enhancing their adeptness in creating dynamic applications.

5.8 Horse Adventure: An Exploration of TypeScript

In the concluding module, our objective was to provide a superficial overview of TypeScript. Recognizing that Vue.js can also be written in TypeScript rather than JavaScript, we deemed it essential to cover its basics for a well-rounded understanding.

While the previous modules extensively addressed code composition and theoretical concepts, this segment of the course focuses more on pre-existing code for validation. As part of this initiative, unit testing was introduced as a novel practice.

Additionally, we integrated the ESLint library, fostering insights into the realm of static code analysis and the diverse tools available.

The tutorial commences with foundational steps, guiding students through the creation of a new project enriched with specific features and requisite dependencies. An introduction to the Bootstrap library enhances their ability to craft a visually appealing application interface. From this juncture, the module seamlessly transitions to constructing the initial component using TypeScript, capitalizing on Bootstrap's attributes to effortlessly generate responsive components.

Subsequently, we delve into the realm of data models and view creation, marking a significant foray into TypeScript scripting. This marks the initial encounter with importing distinct files into an application, including alternate data models, facilitated by TypeScript's import function. The exploration extends to leveraging calculated values as integral components of data models. The culmination of these endeavors results in a comprehensive foundation (as depicted in Figure 11), poised to propel students toward autonomous application development or further enhancements of the example application.

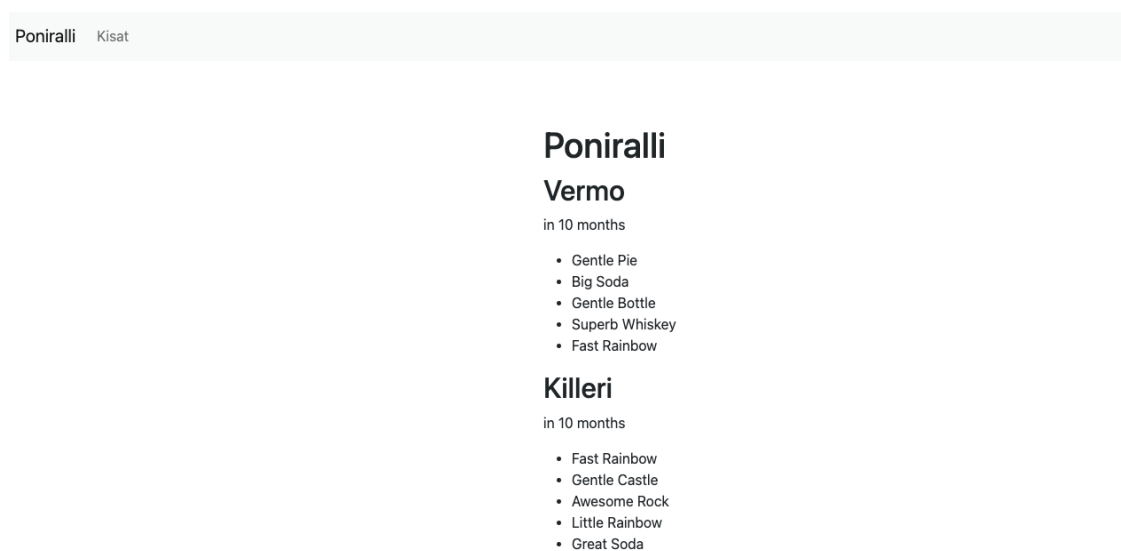


Figure 11 Completed application

The creation of this module proved labor-intensive, primarily due to the integration of unit testing, a measure vital for ensuring accurate exercise execution.

The experience of scripting in TypeScript was illuminating, serving as a constructive contrast to conventional JavaScript. For this basics-oriented course, our approach centered on equipping students with an array of pre-constructed components, striking a balance between learning and practical implementation. Aspects such as the formulation of custom test files and configuring ESLint to function were deliberately reserved for more advanced studies.

6 Summary: Development of a Vue.js Course

This thesis encompassed the design and development of a Vue.js course within the Moodle learning environment, tailored for Metropolia University students. The course's intended audience spans multi-modal studies, elective courses, open AMK studies, and pathway studies. The course's functionality was successfully achieved, anchored in Vue.js fundamentals. Validation of content and a final examination were carried out by two Metropolia students.

The course's curriculum spans Vue.js basics, elucidating the creation and utilization of custom components in personal applications. A prerequisite for course participants is a modest programming foundation, as the course doesn't provide a comprehensive introduction to HTML basics. It assumes a grasp of rudimentary programming, rendering it accessible to those equipped with fundamental programming knowledge. The curriculum strives to impart Vue programming fundamentals while also offering an overview of diverse programming languages.

From a student's perspective, the course adopts a structured, step-by-step approach that facilitates a progressive grasp of the content module by module. The theoretical components are presented in a lucid and digestible manner, extending this accessibility to unit testing materials.

While much of the theoretical groundwork was familiar from professional experiences and personal interests, the development process involved significant new learning. This entailed cross-referencing one's own knowledge against reputable sources. The practical facet of course implementation, including Moodle course planning and execution, marked new terrain.

In summary, this endeavor yielded a Vue.js course characterized by its systematic delivery and user-friendly design, catering to students with a foundation in programming. The fusion of theoretical insights and practical application exemplifies an effective pedagogical approach, providing valuable knowledge and skills for students' programming journeys.

7 Sources

Main Source:

the main source of information is the scientific thesis authored by Tomi Salmi and Jesse Veijalainen, titled "Vue.js course for Moodle." This thesis serves as the primary reference for the material provided in the text.

Salmi, Tomi, and Jesse Veijalainen. "Vue.js course for Moodle."

https://www.theseus.fi/bitstream/handle/10024/793493/Salmi_Veijalainen.pdf?sequence=2&isAllowed=y

Additional Sources:

1. Filipova, Olga. (2017). "Vue.js 2 and Bootstrap 4 Web Development." Packt Publishing. (Accessed: 07.07. 2023)
2. Raymond, Eric S. (1999). "The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary." O'Reilly Media. (Accessed: 10.07.2023).
3. Open Source Initiative. "Open Source Definition." (Accessed: 13.07.2023).
4. Open Source Initiative. "The BSD 2-Clause License (Simplified or FreeBSD License)."[\(https://opensource.org/license/bsd-2-clause/\)](https://opensource.org/license/bsd-2-clause/), (Accessed:20.07.2023).
5. Free Software Foundation. "GNU General Public License (GPL)."
<https://www.gnu.org/licenses/gpl-3.0.en.html>), (Accessed: 29.07.2023).
6. The Apache Software Foundation. "The Apache License, Version 2.0."
<https://www.apache.org/licenses/LICENSE-2.0>), (Accessed: 10.08.2023).
7. Moodle. "About Moodle." ([https://docs.moodle.org/403/en/About Moodle](https://docs.moodle.org/403/en/About_Moodle)), (Accessed: 12.08.2023).
8. Moodle. "Moodle LAMP Stack." ([https://docs.moodle.org/311/en/Step-by-step Installation Guide for Ubuntu](https://docs.moodle.org/311/en/Step-by-step_Installation_Guide_for_Ubuntu)), (Accessed: 12.08.2023).
9. Linux Foundation. "About Linux." (<https://www.linuxfoundation.org/about>) (Accessed: 20.08.2023).

10. The Apache Software Foundation. "About The Apache HTTP Server Project." (https://httpd.apache.org/ABOUT_APACHE.html) (Accessed: 29.08.2023).
11. MariaDB Foundation. "About MariaDB." (<https://mariadb.org/about/>) (Accessed: 05.09.2023)
12. MySQL. "MySQL :: MySQL Community Server." (<https://www.mysql.com/products/community/>), (Accessed: 06.09.2023).
13. Laazir, M., Zennouhi, H., & Ouazzani Touhami, S. (2019). Comparative study of PHP frameworks. Procedia Computer Science, 162, 498-505. (Accessed: 10.09.2023).
14. Title: What is Node.js and Why You Should Use It, Source: Infowind Tech (<https://www.infowindtech.com/what-is-node-js-and-why-you-should-use-it/>), (Accessed: 14.09.2023).
15. ESLint. (<https://eslint.org>) , (Accessed: 16.09.2023).
16. W3Schools. (<https://www.w3schools.com/>), (Accessed: 17.09.2023).
17. MDN Web Docs, W3C specifications, web design blogs, articles, and books on web development and CSS history. (Accessed: 20.09.2023).