Tomi Salmi, Jesse Veijalainen

Vue.js-kurssi Moodleen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka, monimuoto

Insinöörityö

17.4.2023

Tiivistelmä

Tekijä:	Tomi Salmi
Tekijä:	Jesse Veijalainen
Otsikko:	Vue.js-kurssi Moodleen
Sivumäärä:	41 sivua
Aika:	17.4.2023

Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Osaamisaluepäällikkö Janne Salonen

Projektin tilasi Metropolia Ammattikorkeakoulu, joka on Suomen suurin ammattikorkeakoulu. Insinöörityön tavoitteena oli luoda Vue.js:n perusteet -verkkokurssi Metropolia Ammattikorkeakoulun Moodle-ympäristöön. Tarkoituksena oli, että kurssi toimisi monipuolisesti avoimen AMK:n, monimuoto- sekä päiväopiskelijoiden vapaasti valittaviin opintokokonaisuuksiin. Tavoitteena oli, että opiskelija saisi perustason ymmärryksen JavaScript-kieleen ja Vue.js-kirjastoon.

Työssä tutkitaan avoimen lähdekoodin projekteja, jotka ovat vaatimuksena kurssin totuttamiseen sekä eri avoimen lähdekoodin lisenssejä ymmärtämään näiden vaikutus eri komponenttien käyttöön. Teoriaosuudessa syvennymme eri ohjelmointikieliin sekä ohjelmistopaketteihin, jotka ovat kriittinen osuus nykyaikaisia verkkosovelluksia sekä vertailemme näitä kilpaileviin teknologioihin. Tähän sisältyvät sekä Moodlen että Vue.js:n eri riippuvuudet.

Lopputuloksena muodostui Moodle-ympäristön toteutettu kurssi, jossa opiskelija ymmärtää perusteet Vue.js-ohjelmoinnista ja osaa hyödyntää ymmärrystä tulevaisuudessa omissa projekteissaan sekä ymmärtää vaadittavat työkalut, joita vaaditaan kehitysympäristössä.

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

Abstract

Author:	Tomi Salmi
Author:	Jesse Veijalainen
Title:	Vue.js course to Moodle
Number of Pages:	41 pages
Date:	17 April 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Software Engineering
Supervisors:	Janne Salonen, Head of school

The project was commissioned by Metropolia University of Applied Sciences, which is Finland's largest university of applied sciences. The goal of the bachelor thesis was to create a Vue.js fundamentals online course in Metropolia Moodle environment. The purpose was that the course would work flexibly for the freely chosen study units of the open UAS, blended learning and day-time students. The goal was for the course percipients to gain a basic understanding of the JavaScript language and the Vue.js library.

The work examines open-source projects, which are required for the course, as well as different open-source licenses to understand their impact on the use of different components. In the theoretical part, we delve into different programming languages and software packages, which are a critical element of modern web applications, and we compare these to competing technologies. This includes the various dependencies of both Moodle and Vue.js.

The result was a course implemented in the Moodle environment, where the student understands the basics of Vue.js programming and can use the understanding in their own projects in the future, as well as understands the tools that are required in the development environment.

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

Sisällys

Lyhenteet

CSS: Cascading Style Sheets. Porrastetut tyyliohjeet, jotka määrittävät verkkosivun tyyliä.

LMS: Learning Management System. Digitaalinen oppimisympäristö.

LAMP: Linux, Apache, MySQL, PHP. Yleinen kokonaisuus eri ohjelmistoja verkkosivujen tarjoamiseen.

MYSQL: Tietokantamoottori.

PHP: Hypertext Preprocessor. Palvelinpohjainen ohjelmointikieli.

JS: JavaScript sekä JavaScriptillä tehty tiedoston pääte.

VUE: Visual User Enviroment. JavaScriptin ohjelmointikirjasto.

VDOM: Virtuaalinen dokumenttiobjektimalli.

DOM: Dokumenttiobjektimalli.

NPM: Node Packet Management. Noden pakettien hallinta

I/O: Input/Output.

HTML: Hypertext Markup Language. Verkkosivujen kehittämiseen kieli

ECMA: European Computer Manufacturers Association. JavaScript ylläpito.

1 Johdanto

Insinöörityön tavoitteena oli luoda Metropolia Ammattikorkeakoulun Moodle-ympäristöön Vue.js-kurssi harjoituksineen ja materiaaleineen. Tarkoituksena on, että kurssi toimisi avoimen AMK:n, monimuoto- sekä päiväopiskelijoiden vapaasti valittaviin opintokokonaisuuksiin.

Luodun kurssin tavoitteena on saada opiskelijalle perustason osaaminen ja ymmärrys Vue.js-ohjelmointikielen perusteisiin, miten luodaan omia komponentteja, ymmärrystä JavaScriptin sekä TypeScriptin rakenteeseen ja opiskelija pystyy hyödyntämään oppeja omissa projekteissaan. Projektin tilaajana toimii Metropolian ammattikorkeakoulu.

Metropolia Ammattikorkeakoulu on opiskelijamäärältään Suomen suurin AMK. Metropolia on monialainen ja kansainvälinen ammattikorkeakoulu, joka aloitti toimintansa 1. elokuuta 2008, kun Helsingin Ammattikorkeakoulu Stadia ja EVTEK-ammattikorkeakoulu yhdistyivät uudeksi ammattikorkeakouluksi. Metropolia tarjoaa tutkintoja neljältä eri koulutusalalta, joita ovat kulttuuri, liiketalous, sosiaali- ja terveysala sekä tekniikka. Opetusta tarjotaan myös englanniksi.

Työssä käytävät aiheet ovat kaikki avoimeen lähdekoodiin perustuvia. Avoin lähdekoodi on yleistynyt paljon internetin myötä ja tarjoaa monia hyötyjä kehittäjille mutta luo myös omat haasteet esimerkiksi eri lisenssien muodossa.

Opinnäytetyön teoriaosuudessa käydään läpi yleisesti Vue.js:n historiaa, Moodle-ympäristöä sekä Vue.js-kehitysympäristöä. Lisäksi käydään läpi Vue.js-perusteita lyhyesti. Opinnäytetyön kohteena on ohjelmoinnista kiinnostuneet tekniikan alan opiskelijat, joilla on jo perustieto HTML- ja JavaScript-ohjelmoinnista.

Opinnäytetyön toisessa osassa käydään läpi enemmän Vue.js-kirjastoa sekä tutustutaan yleiseen rakenteeseen ja toimintaperiaatteeseen. Moodlen opetusmateriaali koostuu teoriasta ja käytännön harjoitteluista, joka auttaa opiskelijaa

hahmottamaan yleistä ymmärrystä web-ohjelmoinnista sekä tiedostojen rakenteesta ja omien komponenttien tekemisestä. Tavoitteena on, että opiskelija ymmärtää mitä, tarkoittavat eri komponentit, tyylitiedostojen muokkaaminen ja miten saadaan liitettyä sovellukseen eri komponentteja.

Lopputuloksena muodostui Moodle-ympäristön kurssi, jossa opiskelija ymmärtää perusteet Vue.js-ohjelmoinnista ja osaa hyödyntää ymmärrystä tulevaisuudessa omissa projekteissaan.

2 Avoin lähdekoodi

2.1 Yleistä

Avoimen lähdekoodin yhteisö on kasvanut nopeasti viimeisiten vuosikymmenten aikana. Internetin ilmaantumisen ja tekniikan kehittymisen myötä avoimesta lähdekoodista on tullut kehittäjien suosima tapa tehdä yhteistyötä ja jakaa työnsä. Yksi avoimen lähdekoodin yhteisön kasvun tärkeimmistä syistä on kehitysprosessin yhteistoiminnallinen luonne. Kehittäjät kaikkialta maailmasta voivat osallistua projektiin, mikä tekee siitä kestävämmän ja luotettavamman. Tämä yhteistyöhön perustuva lähestymistapa on mahdollistanut ohjelmiston luomisen, joka on vapaasti käytettävissä ja helposti saatavilla. Lisäksi avoimen lähdekoodin yhteisö on luonut läpinäkyvyyden ja vastuullisuuden kulttuurin. Koska koodi on avoin kaikille, virheet ja tietoturvaongelmat voidaan tunnistaa ja korjata nopeasti. Tämä on johtanut kehitettävien ohjelmistojen laadun yleiseen paranemiseen. Lisäksi avoimen lähdekoodin yhteisö on auttanut teknologian demokratisoinnissa. Asettamalla ohjelmistot vapaasti saataville henkilöt ja organisaatiot, joilla ei ehkä ole ollut pääsyä kalliisiin patentoituihin ohjelmistoihin, voivat nyt käyttää ja hyötyä siitä. Kaiken kaikkiaan avoimen lähdekoodin yhteisön kasvulla on ollut merkittävä vaikutus ohjelmistokehitysteollisuuteen ja, se on johtanut laadukkaiden, helppokäyttöisten ohjelmistojen luomiseen.

2.2 Avoimen lähdekoodin lisensointi

Avoimen lähdekoodin lisensointi viittaa oikeudelliseen kehykseen, joka säätelee yleisön saataville asetettujen ohjelmistojen käyttöä, jakelua ja muokkaamista. Avoimen lähdekoodin lisensoinnin avulla käyttäjät voivat käyttää lähdekoodia, joka on ohjelman taustalla oleva ohjelmistokoodi, ja tehdä muutoksia tai parannuksia kyseiseen koodiin. Tätä vapautta käytetään usein uusien ohjelmistojen luomiseen, jotka on räätälöity tiettyjen tarpeiden tai mieltymysten mukaan. Yleisimpiä avoimen lähdekoodin lisenssejä ovat General Public License (GPL), Berkeley Software Distribution (BSD) -lisenssi ja Apache License. Jokainen näistä lisensseistä tarjoaa eritasoisia vapauksia ja rajoituksia käyttäjän tai kehittäjän tarpeitten mukaan. Viime kädessä avoimen lähdekoodin lisensoinnin

tarkoituksena on edistää yhteistyötä ja innovaatioita ohjelmistokehityksessä antamalla käyttäjien vapaasti käyttää ja muokata ohjelmistoja.

Type	Permissive	Permissive	Permissive	Copyleft	Copyleft	Copyleft
Provides copyright protection	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE
Can be used in commercial applications	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE
Provides an explicit patent license	✓ TRUE	✗ FALSE	✗ FALSE	✗ FALSE	✗ FALSE	✗ FALSE
Can be used in proprietary (closed source) projects	✓ TRUE	✓ TRUE	✓ TRUE	✗ FALSE	✗ FALSE partially	✗ FALSE for web
Popular open-source and free projects	Kubernetes Swift Firebase	Django React Flutter	Angular.js JQuery, .NET Core Laravel	Joomla Notepad++ MySQL	Qt SharpDevelop	SugarCRM Launchpad

Kuva 1 Taulukko avoimen lähdekoodin lisensseistä (Todavchich: 2020)

2.2.1 BSD-lisenssi

BSD-lisenssi on suosittu avoimen lähdekoodin ohjelmistolisenssi, jonka useat organisaatiot ja yksityishenkilöt ovat ottaneet laajalti käyttöön. Se on salliva lisenssi, jonka avulla käyttäjät voivat muokata ja jakaa ohjelmistoja uudelleen ilman rajoituksia. BSD-lisenssi on jaettu kahteen pääluokkaan, joita ovat alkuperäinen BSD-lisenssi, jonka on kehittänyt Kalifornian yliopisto, Berkeley, ja muokattu BSD-lisenssi, joka on luotu FreeBSD-projektissa. Alkuperäinen BSD-lisenssi sisältää lausekkeen, joka edellyttää, että kaikki mainosmateriaalit tunnustavat lisensoidun ohjelmiston käytön, kun taas muokattu BSD-lisenssi ei sisällä tätä vaatimusta. Yksi BSD-lisenssin tärkeimmistä eduista on, että se mahdollistaa lisensoidun ohjelmiston sisällyttämisen omistusoikeuteen ilman, että tuotettu ohjelmisto on julkaistava samalla lisenssillä. Tämä tekee BSD-lisenssistä houkuttelevan vaihtoehdon yrityksille, jotka haluavat käyttää avoimen

lähdekoodin ohjelmistoja tuotteissaan ilman, että heidän tarvitsee julkaista ohjelmistonsa lähdekoodia. Copyleft-säännösten puuttuminen BSD-lisenssistä tarkoittaa kuitenkin sitä, että kaikki lisensoituun ohjelmistoon tehdyt muutokset voidaan sisällyttää omistusoikeuteen ilman samoja avoimen lähdekoodin vaatimuksia. Siksi, vaikka BSD-lisenssi tarjoaa huomattavaa joustavuutta, se ei ehkä ole paras valinta kehittäjille, jotka haluavat varmistaa, että heidän tekemänsä muutokset pysyvät avoimen lähdekoodin.

2.2.2 GPL-lisenssi

General Public License tai GPL on laajalti käytetty avoimen lähdekoodin ohjelmistolisenssi, joka säätelee ohjelmistojen käyttöä, jakelua ja muokkaamista. Sen loi Free Software Foundation (FSF) vuonna 1989, ja siitä on sittemmin tullut yksi suosituimmista vapaiden ohjelmistojen jakelun lisensseistä. GPL-lisenssi on suunniteltu tarjoamaan käyttäjille vapaus käyttää, muokata ja jakaa ohjelmistoja ja varmistaa samalla, että nämä vapaudet siirretään tuleville käyttäjille. GPL on copyleft-lisenssi, mikä tarkoittaa, että sen avulla käyttäjät voivat vapaasti käyttää, muokata ja jakaa ohjelmistoja, kunhan kaikki muutokset tai johdannaisteokset ovat myös GPL-lisenssillä. Tämä varmistaa, että ohjelmisto pysyy ilmaisena ja avoimena lähdekoodina, vaikka muut muuttavat ja jakelivat sitä. GPL sisältää myös säännöksiä, jotka vaativat käyttäjiä antamaan pääsyn ohjelmiston muokattujen versioiden lähdekoodiin, mikä edistää läpinäkyvyyttä ja yhteistyötä avoimen lähdekoodin yhteisössä. Lisäksi GPL sisältää lausekkeita, jotka kieltävät ohjelmiston käytön patentoiduissa tuotteissa tai palveluissa, mikä auttaa estämään avoimen lähdekoodin ohjelmistojen hyödyntämisen kaupallisissa tahoissa. Kaiken kaikkiaan GPL on tärkeä työkalu avoimen lähdekoodin ohjelmistojen kehittämisen ja jakelun edistämisessä, ja sillä on ollut merkittävä rooli nykyaikaisen ohjelmistoteollisuuden muotoilussa.

2.2.3 Apache-lisenssi

Apache-lisenssi on avoimen lähdekoodin lisenssi, jota käytetään laajasti ohjelmistokehitysteollisuudessa. Apache-lisenssin tärkein ominaisuus on sen salliva luonne, jonka ansiosta kehittäjät voivat käyttää, muokata ja jakaa ohjelmistoa

ilman rajoituksia. Tämä salliva lähestymistapa helpottaa kehittäjien sisällyttämistä avoimen lähdekoodin ohjelmistoihinsa ja projekteihinsa ilman, että heidän tarvitsee huolehtia juridisista ongelmista tai lisenssimaksuista. Kuten Sinclair (2010) huomauttaa, Apache-lisenssi on yksi suosituimmista avoimen lähdekoodin lisensseistä, koska se mahdollistaa maksimaalisen joustavuuden ja rohkaisee kehittäjien välistä yhteistyötä. Lisäksi Apache License on yhteensopiva muiden avoimen lähdekoodin lisenssien kanssa, mikä helpottaa kehittäjien yhdistämistä eri avoimen lähdekoodin projekteihin ja uusien ohjelmistojen luomiseen. Tämä yhteensopivuus varmistaa myös sen, että Apache-lisenssiä voidaan käyttää monissa yhteyksissä pienistä henkilökohtaisista projekteista suuriin kaupallisiin sovelluksiin. Kaiken kaikkiaan Apache-lisenssin avoimen lähdekoodin, sallivat ja yhteensopivat ominaisuudet tekevät siitä arvokkaan työkalun kehittäjille, jotka haluavat luoda innovatiivisia ohjelmistoja ja edistää samalla yhteistyötä ja yhteisölähtöistä kehitystä.

3 Moodle LMS

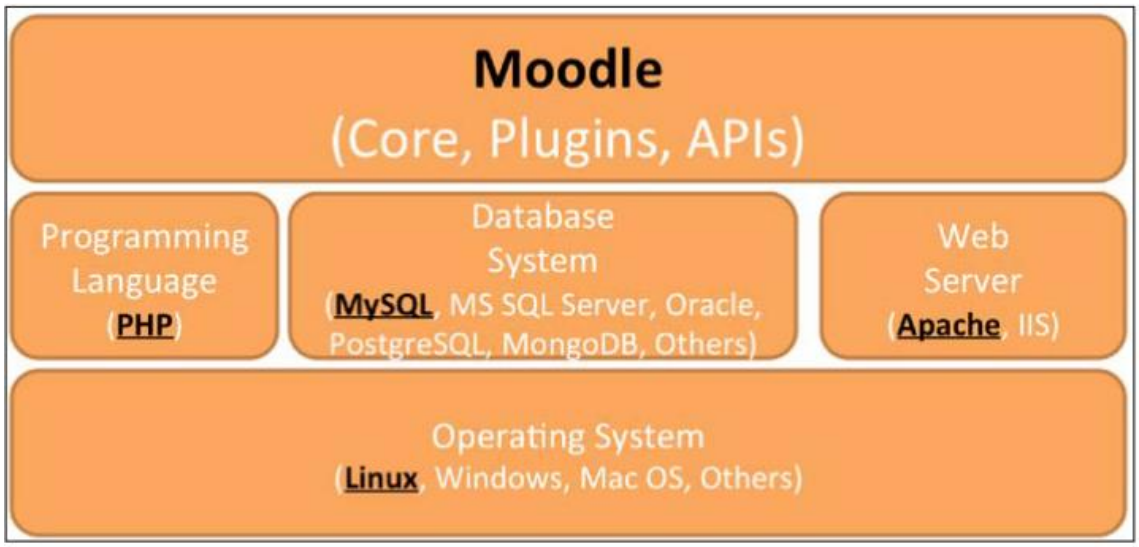
3.1 Moodle-oppimisympäristö

Moodle, joka tulee sanoista Modular Object-Oriented Dynamic Learning Environment, on avoimen lähdekoodin oppimisen hallintajärjestelmä, jonka Martin Dougiamas julkaisi vuonna 2002. Se on verkkopohjainen alusta, jonka avulla voidaan luoda verkkokursseja ja muita oppimistoimintoja oppilaitoksille ja yrityksille. Moodle on suunniteltu tarjoamaan opettajille ja ohjaajille joustava tapa luoda kursseja ja aktiviteetteja opiskelijoilleen sekä seurata heidän edistymistään ja antaa palautetta. Moodle tarjoaa opettajille erilaisia työkaluja verkko-oppimisympäristön luomiseen ja ylläpitämiseen. Sen avulla he voivat myös tarjota kurssinsa opiskelijoiden saataville intuitiivisella ja mukaansatempaavalla tavalla. Lisäksi Moodle mahdollistaa vuorovaikutuksen ja yhteistyön opettajien, opiskelijoiden ja muiden oppimisyhteisön jäsenten välillä.

Moodle voidaan räätälöidä vastaamaan kouluttajan erityistarpeita. Muokkausvaihtoehtojen avulla käyttäjät voivat mukauttaa sovelluksen asettelua ja toimintoja vastaamaan opiskelijoiden ja kurssin tarpeita. Opettaja voi esimerkiksi näyttää kurssimateriaalit tietyssä järjestyksessä tai lisätä tiettyjä widgetejä kurssisivulleen, jotta opiskelijat löytävät materiaalit ja pääsevät niihin helpommin. Lisäksi Moodle antaa käyttäjille mahdollisuuden lisätä laajennuksia tai lisäosia kurssien ulkoasun ja toiminnan mukauttamiseksi laajalti. Näitä laajennuksia voidaan käyttää kurssisivun ulkoasun mukauttamiseen, lisäominaisuuksien lisäämiseen ja jopa oman mukautetun koodin lisäämiseen. Hyödyntämällä Moodlen räätälöintivaihtoehtoja, opettaja voi tehokkaasti räätälöidä Moodlen vastaamaan erityistarpeita ja luoda opiskelijoilleen henkilökohtaisemman oppimiskokemuksen.

3.2 Moodlen käyttämä alusta

Moodle on yleensä asennettu perinteisen LAMP-pinon päälle mutta tukee myös nimensä mukaisesti modulaarisesti tästä poikkeavaa alustaa (kuva 2). Lyhenne LAMP tulee sanoista Linux, Apache, MySQL/MariaDB, PHP.



Kuva 2. Moodle-alustan rakenne (Büchner 2016: 34)

Kuten näkyy, alustan osalta joustavuutta löytyy laajalti. Kunhan taustalla on joku monista tuetuista tietokannoista, PHP tuki sekä web-palvelimen tarjoilemaa sivustoa. Tämä sallii nopean käyttöönoton niillä resursseilla, joita on saatavilla ympäristössä ja varmasti yksi syy Moodlen suosioon koulutusalustana.

3.2.1 Linux

Linux on vapaa ja avoimen lähdekoodin käyttöjärjestelmä, jonka Linus Torvalds kehitti ensimmäisen kerran vuonna 1991 Helsingin yliopistossa. Järjestelmä kehitettiin vastauksena kaupallisiin käyttöjärjestelmiin tuolloin liittyneisiin rajoituksiin ja korkeisiin kustannuksiin. Linuxin varhaiseen kehitykseen vaikutti voimakkaasti Unix-käyttöjärjestelmä, ja Torvalds käytti alun perin Unix-tyyppistä Minix-ydintä projektia lähtökohtana, josta Torvalds lähti toteuttamaan täysin uudelleen kirjoitettua Linux-ydintä yhteisön palautteen perusteella. Projekti julkistettiin vuonna 1992, ja se sai nopeasti seuraajia avoimen lähdekoodin yhteisössä. Tämä yhteisö osallistui järjestelmän kehittämiseen tarjoamalla virheenkorjauksia, parannuksia ja uusia ominaisuuksia. Linuxin varhaiselle kehitykselle oli ominaista myös useiden käyttöjärjestelmän jakelujen tai versioiden luominen, mukaan lukien Slackware ja Debian. Näiden jakelujen kehittäminen mahdollisti Linuxin räätälöinnin tiettyihin käyttötapauksiin ja auttoi järjestelmän

popularisoinnissa. Nykyään Linuxia käytetään monenlaisissa sovelluksissa pöytätietokoneista palvelimiin ja sulautettuihin järjestelmiin. Sen suosio johtuu osittain sen avoimen lähdekoodin luonteesta, joka mahdollistaa kehittäjien välisen yhteistyön ja innovaation.

Linux on suosittu käyttöjärjestelmän ydin, jota käytetään erilaisissa ympäristöissä. Se tarjoaa useita etuja, jotka tekevät siitä sopivan valinnan moniin käyttötarkoituksiin. Ensinnäkin sen avoimen lähdekoodin luonne tekee siitä kustannustehokkaan ratkaisun yrityksille ja yksityishenkilöille. Toisin kuin monet kaupalliset ohjelmistot Linux on vapaasti käytettävä, ja sen lähdekoodi on avoimesti saatavilla muokkaukseen ja parantamiseen. Tämä tekee siitä erinomaisen työkalun kehittäjille ja järjestelmänvalvojille järjestelmän mukauttamiseksi omiin tarpeisiinsa. Toiseksi Linux on erittäin turvallinen. Sen ydinkomponentit on suunniteltu kestämään ja suojaamaan käyttäjätietoja haitallisilta hyökkäyksiltä. Kolmanneksi Linux on erittäin luotettava. Se on suunniteltu vakaaksi ja pystyy toimimaan pitkiä aikoja ilman ongelmia. Tämä tekee siitä ihanteellisen palvelimille ja verkkosovelluksille, joiden on oltava käytettävissä 24/7. Lopuksi Linux on yhteensopiva useiden laitteistojen ja ohjelmistojen kanssa, joten se on ihanteellinen valinta mihin tahansa ympäristöön.

3.2.2 Apache

Apache-verkkopalvelin on Apache Software Foundationin vuodesta 1995 lähtien kehittämä avoimen lähdekoodin projekti. Apache-verkkopalvelin on suosittu verkkopalvelinohjelmisto, jota käytetään verkkosivustojen ja verkkosovellusten isännöintiin. Apache-verkkopalvelin on erittäin tehokas ja monipuolinen palvelin, ja sitä voidaan käyttää nykyaikaisten verkkosovellusten sekä staattisen sisällön palvelemiseen. Se on konfiguroitavissa ja sitä voidaan käyttää useiden verkkosovellusten tarjoamiseen, ja tietyissä kokoonpanoissa on käytettävissä kuormantasaus. Se toimii myös vikasietoisesti. Apache-verkkopalvelin pystyy palvelemaan verkkosivuja sekä muun tyyppistä sisältöä, kuten kuvia, videoita ja äänitiedostoja. Se pystyy myös tarjoamaan suojattuja yhteyksiä verkkosovelluksille käyttämällä SSL/TLS-protokollia salaukseen. Apache-verkkopalvelin on myös erittäin laajennettava ja sitä voidaan käyttää mukautettujen verkkosovellusten ja

-palveluiden luomiseen. Apache-verkkopalvelin pystyy myös luomaan dynaamista sisältöä, mikä mahdollistaa dynaamisten verkkosivujen tai sovellusten luomisen. Lisäksi Apache-verkkopalvelin pystyy tarjoamaan skaalautuvuutta ja suorituskykyä sekä erilaisia välimuistivaihtoehtoja.

Apache-verkkopalvelin on tärkeä komponentti verkkosivujen tarjoamisessa, ja sitä käytetään laajalti sen luotettavuuden ja joustavuuden ansiosta. Apache-verkkopalvelin on avoimen lähdekoodin ohjelmisto, joka on vapaasti yleisön saatavilla ja jota kehittäjäyhteisö päivittää jatkuvasti. Yksi syistä, miksi Apache on niin suosittu, on se, että se on joustavasti konfiguroitavissa, mikä tarkoittaa, että se voidaan mukauttaa eri verkkosivustojen tarpeisiin. Apache tunnetaan myös vakaudestaan, ja se pystyy käsittelemään suuren määrän pyyntöjä kaatumatta. Lisäksi Apache-verkkopalvelin on yhteensopiva useiden käyttöjärjestelmien kanssa, mukaan lukien Windows, Unix ja Linux. Kaiken kaikkiaan Apache-verkkopalvelin on luotettava ja monipuolinen työkalu, joka on välttämätön verkkosivujen tarjoajille, ja sen suosio on osoitus sen laadusta ja tehokkuudesta.

Yksi Apachen käytön tärkeimmistä eduista on sen joustavuus ja skaalautuvuus. Apache voi toimia useissa käyttöjärjestelmissä, mukaan lukien Windows, Linux ja macOS, mikä tekee siitä monipuolisen vaihtoehdon verkkokehittäjille. Lisäksi Apache mahdollistaa useiden ohjelmointikielien, kuten PHP:n, Pythonin ja Perlin käytön, mikä antaa kehittäjille vapauden luoda verkkosovelluksia haluamallaan kielellä. Toinen Apachen käytön etu on sen vahvat suojausominaisuudet. Apachella on modulaarinen arkkitehtuuri, jonka avulla käyttäjät voivat lisätä tai poistaa moduuleja tietoturvatarpeidensa perusteella. Esimerkiksi Apachen mod_security-moduuli tarjoaa tavan suojautua yleisiltä verkkohyökkäyksiltä, kuten SQL-injektio ja cross-site scripting (XSS). Lisäksi Apachen pääsynhallinta-moduulit antavat järjestelmänvalvojille mahdollisuuden rajoittaa pääsyä tiettyihin verkkosivustonsa osiin käyttäjien roolien ja käyttöoikeuksien perusteella. Lopuksi Apache tarjoaa erinomaisen suorituskyvyn ja nopeuden. Apachen MPM (multi-processing module) mahdollistaa suuren määrän samanaikaisia pyyntöjä tehokkaasti. Lisäksi Apachen välimuistimoduulit voivat nopeuttaa verkkosivustojen latausaikoja tallentamalla välimuistiin usein käytettyä sisältöä. Lopuksi

Apache on tehokas verkkopalvelinohjelmisto, joka tarjoaa käyttäjilleen joustavuutta, turvallisuutta ja suorituskykyä.

3.2.3 MariaDB

MariaDB on avoimen lähdekoodin relaatiotietokannan hallintajärjestelmä, joka perustuu MySQL:ään. Se on kirjoitettu C- ja C++-kielellä tavoitteena auttaa käyttäjiä saamaan maksimaalisen suorituskyvyn, skaalautuvuuden ja luotettavuuden samalla, kun he voivat siirtyä kalliista kaupallisista tietokannoista avoimeen lähdekoodiin. MariaDB on suosittu kehittäjien ja organisaatioiden keskuudessa, koska se tarjoaa kaikki MySQL:n ominaisuudet, mutta lisää myös edistyneitä ominaisuuksia, parannettua tietoturvaa ja laajennettavuutta.

Vaikka MariaDB:llä on monia etuja, sillä on myös joitain haittoja. Yksi MariaDB:n käytön etu on, että siinä on laaja valikoima ominaisuuksia ja se on suhteellisen helppokäyttöinen. Tämä tarkoittaa sitä, että kehittäjät ja tietokannan ylläpitäjät voivat nopeasti ja helposti luoda ja hallita tietokantoja. Lisäksi MariaDB on erittäin laajennettavissa ja sitä voidaan skaalata ylös tai alas sovelluksen tarpeiden mukaan. Tämä tekee siitä ihanteellisen sovelluksille, joissa käyttöaste vaihtelee. Toisaalta yksi MariaDB:n käytön haittapuoli on, että sen käyttöönotto ja ylläpito voi olla vaikeaa. Lisäksi MariaDB ei ole yhtä turvallinen kuin jotkin muut tietokannat, ja MariaDB-tietokannassa on raportoitu tietojen korruptoitumista. Kaiken kaikkiaan vaikka MariaDB:llä on monia etuja, on tärkeää punnita edut ja haitat ennen kuin päätetään, onko se oikea valinta sovellukseen.

3.2.4 MariaDB vs. MySQL

MariaDB ja MySQL ovat molemmat avoimen lähdekoodin relaatiotietokannan hallintajärjestelmiä (RDBMS), joita käytetään laajasti verkkosovelluksissa. MySQL:n loi vuonna 1995 Michael Widenius, kun taas MariaDB:n perusti vuonna 2009 sama luoja hänen erottuaan MySQL-kehitystiimistä. Molemmat järjestelmät käyttävät Structured Query Language (SQL) -kieltä ja ovat yhteensopivia eri käyttöjärjestelmien kanssa. Suorituskyvyn suhteen molemmat järjestelmät ovat erittäin skaalautuvia ja tarjoavat nopean tiedonkäsittelyn. MariaDB

on kuitenkin osoittanut parempaa suorituskykyä joissakin vertailuissa, varsinkin kun on kyse suurten tietojoukkojen käsittelystä. MariaDB tarjoaa myös enemmän ominaisuuksia ja parannuksia esimerkiksi tietoturvan ja replikoinnin aloilla, mikä tekee siitä suositun valinnan yritystason sovelluksille. Toisaalta MySQL:tä pidetään edelleen suositumpana vaihtoehtona joissakin sovelluksissa, erityisesti sellaisissa, jotka edellyttävät yhteensopivuutta kolmannen osapuolen ohjelmistojen kanssa. Molemmissa järjestelmissä on suuret ja aktiiviset yhteisöt, jotka tarjoavat tukea, dokumentaatiota ja päivityksiä. Kaiken kaikkiaan valinta MariaDB:n ja MySQL:n välillä riippuu sovelluksen erityistarpeista ja vaatimuksista.

MariaDB on MySQL:n forkki, joka luotiin vastauksena siihen, että Oracle osti Sun Microsystemsin, joka omisti tuolloin MySQL:n. Yksi merkittävimmistä eroista MariaDB:n ja MySQL:n välillä on lisensointimalli. MariaDB julkaistaan GNU General Public License (GPL) -lisenssillä, kun taas MySQL on julkaistu patentoitujen ja avoimen lähdekoodin lisenssien yhdistelmällä. Toinen ero on se, että MariaDB on ottanut käyttöön joitakin ominaisuuksia, joita MySQL:ssä ei ole, kuten virtuaaliset sarakkeet, joiden avulla käyttäjät voivat luoda sarakkeita, joita ei tallenneta tietokantaan mutta jotka lasketaan lennossa muiden sarakkeiden perusteella. MariaDB sisältää myös suorituskyvyn parannuksia, kuten säikeenvaraajan laajennuksen, joka parantaa skaalautuvuutta korkean samanaikaisuuden työkuormilla. On kuitenkin syytä huomata, että MySQL:llä on suurempi käyttäjäkunta ja laajempi työkalujen ja lisäosien ekosysteemi. Yhteenvetona voidaan todeta, että MariaDB:llä ja MySQL:llä on joitain tärkeitä eroja lisensoinnin, ominaisuuksien ja suorituskyvyn parannusten suhteen, mutta molemmat ovat edelleen suosittuja valintoja relaatiotietokannan hallintajärjestelmissä.

3.2.5 PHP

PHP on suosittu ohjelmointikieli, jota verkkokehittäjät käyttävät dynaamisten, interaktiivisten verkkosivustojen ja verkkosovellusten luomiseen. Se tunnetaan vakaana ja luotettavana kielenä, ja verkossa on runsaasti resursseja niille, jotka haluavat oppia lisää. PHP on myös erittäin laajennettava, joten kehittäjät voivat lisätä toimintoja kirjastojen, kehysten ja laajennusten avulla.

Eri PHP-kehykset ovat tulleet yhä suositummiksi viime vuosina, koska ne pystyvät yksinkertaistamaan verkkosovellusten ja verkkosivustojen kehitysprosessia. Laazirin ym. (2019) mukaan, PHP-kehysten käyttö on hyödyllistä kehittäjille ja suunnittelijoille, koska se tarjoaa joukon työkaluja, kirjastoja ja komponentteja, joita voidaan helposti käyttää uudelleen, mikä tekee kehitysprosessista nopeamman ja vähemmän alttiiksi virheille. Lisäksi nämä puitteet tarjoavat myös rakenteen koodin järjestämiseen ja hallintaan, jolloin kehittäjät voivat helposti muokata ja ylläpitää koodia. Lisäksi PHP-kehykset tarjoavat myös tietoturvaominaisuuksia, kuten syötteen vahvistuksen, todennusta ja salausta, jotka ovat ratkaisevan tärkeitä verkkosovellusten suojauksessa. Nämä puitteet myös tarjoavat skaalautuvuutta ja joustavuutta, minkä ansiosta kehittäjät voivat helposti muokata ja lisätä ominaisuuksia olemassa olevaan verkkosovellukseen. Tämä helpottaa kehittäjien pysymistä muuttuvan tekniikan ja käyttäjien tarpeiden mukana. Lisäksi PHP-kehysten suosio piilee niiden kyvyssä antaa kehittäjille mahdollisuuden integroida helposti muihin verkkoteknologioihin. Tämä helpottaa verkkosovellusten kehittämistä erilaisilla ominaisuuksilla. Yhteenvetona voidaan todeta, että PHP-kehysten suosio johtuu niiden kyvystä yksinkertaistaa verkkosovellusten kehitysprosessia, tarjota rakenne koodin järjestämiseen ja hallintaan sekä tarjota suojausominaisuuksia ja antaa kehittäjille mahdollisuuden integroida helposti muihin verkkoteknologioihin.

4 Vue.js

4.1 Yleistä Vue.js:stä

Vue.js on suosittu progressiivinen JavaScript-kehys, joka on aiheuttanut aaltoja verkkokehitysyhteisössä julkaisustaan vuonna 2014 lähtien. Se on avoimen lähdekoodin kehys, jonka avulla kehittäjät voivat rakentaa käyttöliittymiä helposti ja joustavasti. Tutkitaan hieman Vue.js:n ominaisuuksia ja etuja sekä verrataan sitä muihin JavaScript-kehikkoihin ja tarkastellaan Vue.js:n todellisia sovelluksia eri aloilla. Sukelletaan siis sisään ja tutkitaan Vue.js:n maailmaa. Vue.js:ssä päivitetään data dynaamisesti HTML-elementteihin, jotka ovat sidottuina eri Vue-objekteihin. Tällöin voidaan tarjota loppukäyttäjälle interaktiivisen lopputuloksen, joka ei vaadi sivun päivittämistä, vaan sivu päivitetään automaattisesti hyödyntäen JavaScriptin tapahtumia, joka päivittää automaattisesti sivuston HTML- ja CSS-koodia.

Vue.js on suunniteltu käyttöliittymien rakentamiseen. Se esiteltiin ensimmäisen kerran vuonna 2014, ja se on saavuttanut laajan suosion vankkojen ominaisuuksiensa ja lukuisten etujensa ansiosta. Rojasin (2019) mukaan yksi Vue.js:n tärkeimmistä ominaisuuksista on sen yksinkertaisuus, jonka ansiosta kehittäjien on helppo ymmärtää ja käyttää Vue.js-kehystä. Toinen Vue.js:n tärkeä ominaisuus on sen reaktiivisuusjärjestelmä, jonka avulla kehittäjät voivat rakentaa dynaamisia käyttöliittymiä, jotka päivittyvät automaattisesti tietojen muutosten perusteella. Lisäksi Vue.js on erittäin joustava ja, se voidaan helposti integroida muihin kirjastoihin ja kehyksiin. Tämä tekee siitä ihanteellisen valinnan kehittäjille, joiden on rakennettava monimutkaisia sovelluksia monipuolisilla toiminnoilla. Vue.js:n käytöllä on lukuisia etuja, mukaan lukien nopeammat kehitysajat, parempi suorituskyky ja parempi skaalautuvuus. Vue.js:llä on myös suuri ja aktiivinen yhteisö, joka tarjoaa kehittäjille pääsyn runsaisiin resursseihin ja tukea yhteisöltä. Kaiken kaikkiaan Vue.js on tehokas ja monipuolinen kehys, josta on tullut yhä suositumpi kehittäjien keskuudessa sen helppokäyttöisyyden, joustavuuden ja lukuisten etujen ansiosta.

Verrattuna muihin JavaScript-kehyksiin, kuten Reactiin ja Angulariin, Vue.js:n oppimiskäyrä on yksinkertaisempi sen kevyen luonteen ja helppokäyttöisyyden vuoksi. Vue.js tunnetaan myös erinomaisesta dokumentaatiostaan, mikä helpottaa kehittäjien oppimista ja käyttöä. Novacin ym. (2021) mukaan Vue.js:llä on selkeä etu suorituskyvyn suhteen verrattuna muihin JavaScript-kehyksiin. Vue.js:n tiedostokoko on pienempi, mikä nopeuttaa latausaikoja ja parantaa suorituskykyä. Lisäksi Vue.js:ssä on reaktiivisuusjärjestelmä, jonka avulla kehittäjät voivat helposti päivittää tiedot ja näkymän ilman monimutkaista tilanhallintaa. Yksi Vue.js:n haitoista on kuitenkin sen rajallinen yhteisön tuki verrattuna Reactiin ja Angulariin. Kuten Novac ym. (2021) huomauttaa, Reactilla ja Angularilla on suuremmat yhteisöt, mikä tarkoittaa, että kehittäjät voivat käyttää enemmän resursseja, kirjastoja ja työkaluja. Lopuksi Vue.js on tehokas JavaScript-kehys, jolla on vahvuutensa ja heikkoutensa. Vaikka sillä ei välttämättä ole samantasoista yhteisön tukea kuin Reactilla ja Angularilla, Vue.js tarjoaa erinomaisen suorituskyvyn ja helppokäyttöisyyden, mikä tekee siitä varteenotettavan vaihtoehdon kehittäjille. (Novac ym. 2021.)

Yksi tärkeimmistä syistä, miksi Vue.js on saamassa suosiota, johtuu sen todellisista käyttökohteista eri aloilla. Filipovan (2016) mukaan Vue.js:ää on käytetty menestyneissä projekteissa, kuten Alibabassa, Xiaomissa ja WeChatssä. Alibaba, johtava Kiinan verkkokauppayritys, käytti Vue.js:ää mobiilisovelluksensa rakentamiseen, mikä johti parempaan suorituskykyyn ja nopeampaan latausaikaan. Kiinalainen elektroniikkayritys Xiaomi käytti Vue.js:ää Mi Home -sovelluksessaan, jolla ohjataan kodin älylaitteita. Kehys auttoi heitä saavuttamaan sujuvan käyttökokemuksen ja paransi sovelluksensa yleistä suorituskykyä. WeChat, suosittu viestintäsovellus Kiinassa, käytti myös Vue.js:ää verkkoversionsa rakentamiseen, mikä johti parempaan suorituskykyyn ja nopeampiin latausaikaan. Nämä menestystarinat osoittavat Vue.js:n monipuolisuuden ja tehokkuuden todellisissa sovelluksissa.

4.1.1 Vue.js vs. Angular

Vue.js ja Angular ovat kaksi suosittua JavaScript-kehystä, joita käytetään dynaamisten verkkosovellusten rakentamiseen. Vue.js on progressiivinen kehys,

joka on suunniteltu asteittain käyttöönotettavaksi ja joka voidaan integroida olemassa oleviin projekteihin. Toisaalta Angular on kattava kehys, jota käytetään laajasti suurten yrityssovellusten rakentamiseen. Novacin ym. (2021) mukaan Vue.js tunnetaan yksinkertaisuudestaan ja helppokäyttöisyydestään, minkä vuoksi se on suosittu valinta pienten ja keskisuurten sovellusten rakentamiseen. Kehys tarjoaa yksinkertaisen ja intuitiivisen API:n, jonka avulla on helppo luoda uudelleenkäytettäviä komponentteja. Angular puolestaan on monimutkaisempi kehys, joka vaatii jyrkemmän oppimiskäyrän, mutta tarjoaa enemmän ominaisuuksia ja toimintoja. Se tarjoaa vankan sarjan työkaluja suurten sovellusten rakentamiseen, kuten riippuvuusinjektioon, reaktiiviseen ohjelmointiin ja tehokaseen mallimoottoriiin. Molemmilla puitteilla on vahvuutensa ja heikkoutensa, ja ne sopivat erilaisiin projekteihin. Kehittäjien tulee valita tarpeisiinsa parhaiten sopiva viitekehys projektin vaatimusten, kehityskokemuksen ja tiimin osaamisen perusteella.

Molemmilla on vahvuutensa ja heikkoutensa, ja kehittäjät keskustelevat usein siitä, kumpaa he käyttävät tiettyyn projektiin. Vue.js on kevyt kehys, joka keskittyy yksinkertaisuuteen ja helppokäyttöisyyteen. Sen avulla kehittäjät voivat luoda interaktiivisia käyttöliittymiä nopeasti ja tehokkaasti. Toisaalta Angular on monimutkaisempi kehys, joka tarjoaa laajan valikoiman ominaisuuksia ja työkaluja. Se on suosittu suurissa yrityssovelluksissa, jotka vaativat monimutkaista tiedonhallintaa ja korkeaa suorituskykyä. Novacin ym. (2021) mukaan "Vue.js on helpompi oppia ja käyttää pienissä ja keskisuurissa projekteissa, kun taas Angular sopii paremmin suuriin projekteihin, joissa on monimutkaisia data- ja logiikkavaatimuksia." Vue.js vaatii vähemmän asennusta ja määritystä, joten se on ihanteellinen nopeaan prototyyppien luomiseen ja pieniin projekteihin. Angularilla sen sijaan on jyrkempi oppimiskäyrä ja, se vaatii enemmän asennusta ja konfigurointia, mutta se tarjoaa vankemman ja skaalautuvamman kehyksen suurille sovelluksille. Yhteenvetona voidaan todeta, että valinta Vue.js:n ja Angularin välillä riippuu käsillä olevan projektin erityistarpeista. Pienet ja keskisuuret projektit voivat hyötyä Vue.js:n yksinkertaisuudesta ja helppoudesta, kun taas suuret yrityssovellukset voivat vaatia Angularin kestävyyttä ja skaalautuvuutta.

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

@Component ({
   selector: 'my-app',
   template: `<h1>Hello {{name}}</h1>`,
})
export class AppComponent  { name = 'World'; }
```

Esimerkkikoodi 1 Angular Hello world

4.1.2 Vue.js vs. React

React on käyttöliittymien rakentamiseen tarkoitettu JavaScript-kirjasto, jonka Facebook loi vuonna 2011. React tunnetaan korkeasta suorituskyvystään ja joustavuudestaan, minkä vuoksi se on suosittu valinta suuriin projekteihin. Sekä Vue.js että React käyttävät virtuaalista DOM:ia, mikä mahdollistaa käyttöliittymän muutosten nopeamman renderöinnin. Vue.js käyttää kuitenkin mallipohjaista syntaksia, kun taas React käyttää JavaScript-pohjaista syntaksia. Filipovan (2016:50-52) mukaan Vue.js:ää verrataan usein Angular and Reactiin, ja se erottuu yksinkertaisuudellaan ja helppokäyttöisyydellään. React sen sijaan valitaan usein sen korkean suorituskyvyn ja kyvyn rakentaa monimutkaisia sovelluksia vuoksi. Viime kädessä valinta Vue.js:n ja Reactin välillä riippuu kehittäjän ja kyseessä olevan projektin erityistarpeista ja mieltymyksistä.

Molemmilla viitekehyksellä on joitain yhtäläisyyksiä, mutta eroavat tietyiltä osin. Vue.js on progressiivinen ja vähitellen käyttöön otettava kehys, joka keskittyy näkymäkerrokseen, kun taas React on kirjasto käyttöliittymien rakentamiseen, joka korostaa komponenttipohjaista lähestymistapaa. Yksi Vue.js:n eduista on, että sillä on alhainen oppimiskäyrä ja se on helppo ymmärtää, mikä tekee siitä hyvän vaihtoehdon aloittelijoille. Lisäksi Vuen syntaksi on tiiviimpi ja intuitiivisempi, mikä helpottaa lukemista ja kirjoittamista. Sitä vastoin Reactilla on jyrkempi oppimiskäyrä ja, se vaatii kattavampaa ymmärrystä JavaScriptistä ja sen syntaksista. Reactin komponenttipohjaisen lähestymistavan ansiosta kehittäjät voivat kuitenkin käyttää koodia uudelleen ja luoda monimutkaisia käyttöliittymiä helpommin. Lisäksi Reactin virtuaalinen DOM mahdollistaa tehokkaan renderöinnin ja parantaa suorituskykyä. Näistä eduista huolimatta Reactilla on joitain haittoja, kuten lisäkirjastojen tarve tilanhallinnan käsittelyyn ja datan

reititykseen. Toisaalta Vue.js:ssä on sisäänrakennetut tilanhallinta- ja reititysominaisuudet, jotka tekevät sen käytöstä mukavampaa. Vue.js:n yhteisö ei kuitenkaan ole yhtä laaja kuin Reactin, mikä saattaa rajoittaa sen resurssien ja tuen saatavuutta. Yhteenvetona voidaan todeta, että sekä Vue.js:llä että Reactilla on hyvät ja huonot puolensa, ja kehyksen valinta riippuu viime kädessä kehittäjän mieltymyksistä ja sekä projektin vaatimuksista.

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

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

export default App;
```

Esimerkkikoodi 2 React Hello world

Taulukko 1 Angularin, Reactin ja Vuen ominaisuudet

Ominaisuus	**Vue.js**	**Angular**	**React**
Komponenttipohjainen	X	X	X
VDOM	X		X
Natiivi TypeScript tuki	X	X	X
Direktiivit	X	X	
Kaksisuuntainen tiedon sitominen	X	X	

4.2 Node

Node.js on avoimen lähdekoodin monialustainen JavaScript-ympäristö ja kirjasto. Node.js:n on kehittänyt Ryan Dahl vuonna 2009 ja viimeisin julkaistu versio on 19.8.1 maaliskuussa 2023. Kehittäjät käyttävät Node.js:ää palvelinpuolen verkkosovellusten luomiseen ja, siksi se on täydellinen dataintensiivisiin sovelluksiin, koska siinä käytetään asynkronista tapahtumapohjaista mallia.

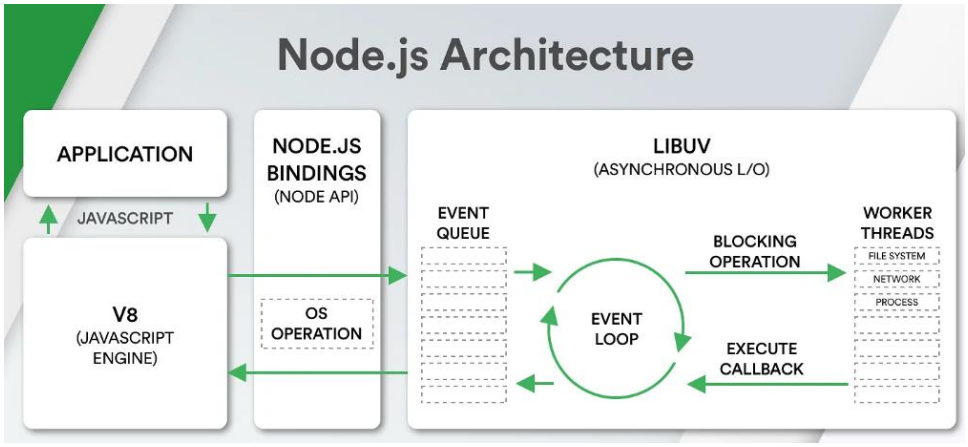
Node.js:n käyttöön on monta syytä, joita käytämme eri sovelluksien luomiseen palvelinpuolelle:

- Rakennettu Google Chromen V8-moottorille ja tämän avulla sovelluksista saadaan nopeita suoritukseltaan sekä toiminnaltaan.

- Noden pakettihallinnassa (NPM) on saatavilla yli 50 000 pakettia, jonka avulla kehittäjät voivat tuoda sovellukseen minkä tahansa toiminnallisuuden, koska vain ja jolloin säästetään kehittäjien aikaa.

- Videon ja äänen latausaika on pienempi, koska loppukäyttäjän ja palvelimen välillä koodin synkronointi on parempaa.

- JavaScript-kehys, joka tarjoaa avoimen lähdekoodin hyödyt.



Kuva 3 Node.js:n rakenne (Kaneriya: 2022)

Node.js-arkkitehtuuri toimii yhdellä säikeellä, jolloin voidaan käsitellä tuhansia samanaikaisia tapahtumakutsuja.

Noden käyttöä voidaan pitää helppona, jos on JavaScript-ohjelmointitaustaa entuudestaan. Node.js:ää ei kannata käyttää

- kun kyseessä on erittäin raskas prosessi/raskas sovellus
- kun sovelluksessa on raskasta laskentaa
- käyttäessään pelkästään HTML-sovelluksen kanssa, joka toimittaa tiedot suoraan palvelimelta ja kun ei ole mitään API:a välissä.

Yhtä prosessia käytettäessä Node.js hyödyntää tapahtumapohjaista, estävää I/O-mallia ja tästä syystä kaikki intensiiviset suorittimen käsittelytoiminnot estävät saapuvan pyynnön. Forbesin mukaan Node.js-kehittäjät ovat hyvin kysyttyjä ympäri maailmaa, koska JavaScriptillä on laaja kirjasto. Vuonna 2020 node.js ladattiin 98,9 miljoonaa kertaa. Node.js:n käyttö tuotannossa on vuoden 2010 julkaisun jälkeen lisääntynyt dramaattisesti. Palkallisesti kehittäjät saavat parempaa keskipalkkaa kuin muut verkkoteknologian kehittäjät, esimerkiksi Intiassa kehittäjät tienasivat keskimäärin 900 000 Intian rupiaa ja Yhdysvalloissa 115 000 dollaria vuodessa. Netflix, PayPal ja Uber käyttävät Node.js:ää ja, tämän myötä käyttö on kasvanut räjähdysmäisesti.

Node Package Manager (NPM) on tehokas työkalu, joka yksinkertaistaa Node.js-pakettien asennusta, päivitystä ja hallintaa. Se tarjoaa keskitetyn arkiston uudelleenkäytettäville koodimoduuleille, joita kehittäjät voivat helposti jakaa ja käyttää maailmanlaajuisesti. Se on komentorivikäyttöliittymä, jonka avulla kehittäjät voivat asentaa, päivittää ja poistaa helposti paketteja projekteistaan. NPM on rakennettu Node.js:n päälle, joka on suosittu JavaScript-ajoympäristö. Kuten Goswami ym. (2020:1) kertovat "NPM tarjoaa keskitetyn pakettien arkiston, joka voidaan helposti asentaa ja päivittää". NPM:n rekisterissä on yli miljoona pakettia, ja se kasvaa jatkuvasti. Kehittäjät voivat myös luoda omia pakettejaan ja jakaa ne yhteisön kanssa. NPM:stä on tullut olennainen työkalu JavaScript-ekosysteemissä sen helppokäyttöisyyden ja saatavilla olevien pakettien suuren määrän ansiosta. Sen suosio on johtanut muiden paketinhallintaohjelmien, kuten Facebookin kehittämän Yarnin, kehittämiseen. NPM on kuitenkin edelleen eniten käytetty paketinhallinta JavaScript-kehittäjien keskuudessa. Kaiken kaikkiaan Node Package Manager on Node.js-kehittäjille tärkeä työkalu, joka auttaa virtaviivaistamaan kehitysprosessia ja parantamaan koodin laatua.

3.3 JavaScript

JavaScript on ohjelmointikieli, jonka avulla voidaan toteuttaa dynaamisia verkkosivuja. Tarkoituksena tehdä on elementtejä, jotka parantavat sivuston kävijöiden käyttökokemusta eri selaimen ja laitteen välillä. JavaScriptillä voidaan tehdä esimerkiksi avattavat valikot, animaatioita ja dynaamisia taustavärejä. Alun perin JavaScript oli kehitetty vain sisäiseen käyttöön, kunnes NetScape ja Brendan Eich veivät kielen vuonna 1995 kansainväliseen ECMA-standardijärjestöön ja samalla perustettiin tekninen komitea kehittämään kieltä, TC39. Ensimmäinen versio julkaistiin 1.6.1997, joka nimettiin ECMA-262:ksi tai ES1:ksi, ja ensimmäinen selain, minkä tuki ES1:lle oli IE4.

JavaScript on jatkanut kehitystään uusien selainten tultua kuten Mozilla Firefoxin ja Google Chromen. Tästä Google Chrome on alkanut kehittämään modernimpaa ja kehittyneempää JavaScript-moottoria V8. Tämä kääntää bittikoodin natiiviksi konekoodiksi. Alun perin JavaScript toimi vain asiakaspuolella, mutta Node.js:n kehityksen myötä palvelinpuolelle tehtävät prosessit ovat yleistyneet, kuten Google Chromen V8-moottorin ympärille rakennettu monikäyttöinen palvelinympäristö, eikä ole rajattu pelkästään selaimeen. Kieli on kehitetty enimmäkseen verkkopohjaisten sovelluksien tekemiseen, mutta JavaScriptillä voidaan tehdä myös muutakin, esimerkiksi verkko- ja puhelinsovelluksia, verkkopalvelimia sekä verkkosovelluksia ja pelejä.

JavaScriptissä on monia hyviä puolia, mikä tekee siitä paremman kielen kuin muut:

- Yksinkertaisuus – Yksinkertainen rakenne, jonka ansiosta helppo oppia ja ottaa käyttöön sekä myös nopeampi kuin muut kielet. Virheet ovat myös helppo havaita ja korjata.

- Nopeus – Komentosarjojen suoritus suoraan verkkoselaimessa muodostamatta ensin mitään yhteyttä palvelimeen tai tarvitsematta mitään kääntäjää välissä.

- Monipuolisuus – Yhteensopiva muiden kielten kuten PHP, Perl ja Javan kanssa.

- Suosio – Saatavilla reilusti eri oppimismateriaalia ja foorumeita kielen opetteluun ja aloitukseen niille aloittelijoille, joilla ei ole hyvää teknistä osaamista tai muuten tietoa kielestä.

- Palvelimen kuormitus - Vähentää palvelimelle lähetettäviä pyyntöjä ja tietojen validointi tehdään suoraan selaimen kautta ja sovelluksen päivitykset koskevat vain tiettyjä osia.

- Päivitykset - JavaScript kehitystiimi ja ECMA päivittävät ja luovat jatkuvasti uusia kirjastoja varmistaen niiden tehokkuuden.

JavaScriptissä on myös heikkouksia kuten kaikissa muissakin ohjelmointikielissä:

- Selainten yhteensopivuus – Jokainen selain tulkitsee koodia eri tavalla, mikä aiheuttaa ongelmia. Tämän takia koodi, minkä on tehnyt, pitäisi testata myös vanhemmissa selaimen versioissa.

- Suojaus – Asiakaspuolella toimiva koodi on alttiina hakkereille.

- Virheenkorjaus – Vaikka jotkin HTML-editorit tukevatkin virheenkorjausta, niin silti ne ovat vähemmän tehokkaita kuin perinteiset editorit, koska selaimet eivät välttämättä näytä suoraan, missä virhe on, joka tekee tästä haastavaa.

JavaScriptiä siis käytetään useimmiten HTML:n ja CSS:n kanssa rinnakkain lisäämällä sivuille interaktiivisuutta. JavaScriptissä on myös monia ulkopuolisia kirjastoja, joita kehittäjä voi hyödyntää, jotta säästetään aikaa eikä aina tarvitse kehittää alusta lähtien.

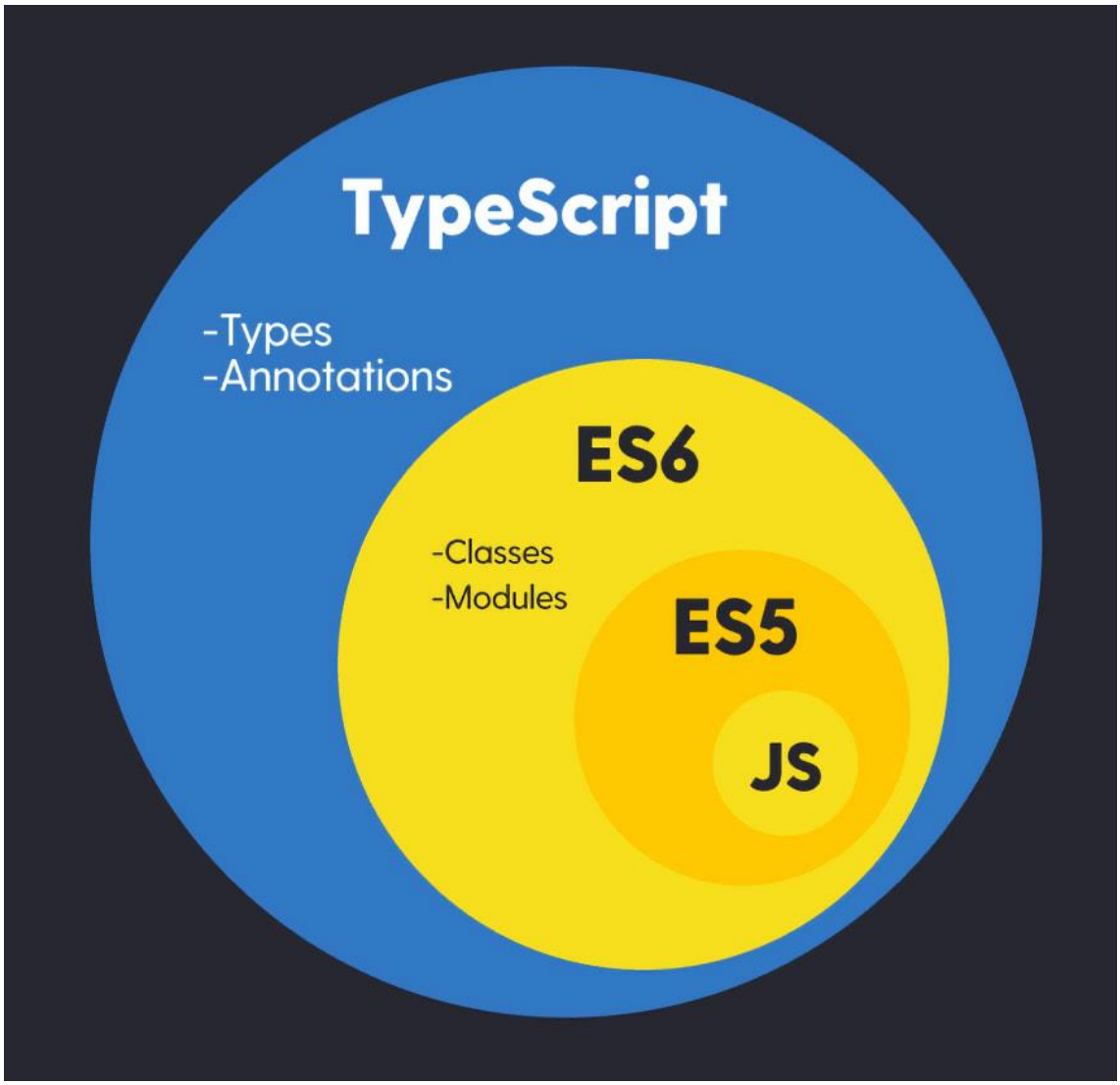
4.3 TypeScript

TypeScript on tehokas ja laajalti käytetty ohjelmointikieli, joka on saanut suosiota ohjelmistokehittäjien keskuudessa. Projekti alkoi Microsoftin sisäisenä hankkeena, jonka ne toivat julkisesti saataville 2012. TypeScript on pohjimmiltaan JavaScriptin muunnos, mikä tarkoittaa, että se lisää uusia ominaisuuksia ja toimintoja jo vakiintuneeseen JavaScript-kieleen. TypeScript on staattisesti kirjoitettu JavaScriptin muunnos, jonka tarkoituksena on parantaa laajamittaisten sovellusten kehitystä. Yksi TypeScriptin tärkeimmistä ominaisuuksista on sen tuki tyyppimerkinnöille, joiden avulla kehittäjät voivat määrittää muuttujien tietotyypit, funktioparametrit ja palautusarvot. Tämä auttaa havaitsemaan virheet käännösvaiheessa eikä vastaa suorituksen aikana, mikä johtaa

luotettavampaan koodiin. Toinen TypeScriptin tärkeä etu on sen kyky tarjota parempaa työkalutukea ja ingraatiota IDE:een, kuten koodin viimeistelyyn, uudelleenmuodostukseen ja navigointiin, joka voi parantaa kehittäjien tuottavuutta. TypeScript sisältää myös ominaisuuksia, kuten luokat, rajapinnat ja moduulit, jotka helpottavat koodin kirjoittamista ja järjestämistä modulaarisella sekä skaalautuvalla tavalla. Lisäksi TypeScriptiä voidaan käyttää suosittujen JavaScript-kehysten, kuten Vue.js:n, Reactin ja Angularin kanssa, mikä tarjoaa saumattoman integraation olemassa oleviin koodikantoihin. Kaiken kaikkiaan TypeScript tarjoaa joukon etuja nykyaikaiseen verkkokehitykseen, kuten parannetun tyyppiturvallisuuden, paremman työkalutuen ja lisääntyneen tuottavuuden verrattuna JavaScriptiin.

JavaScript ja TypeScript ovat kaksi ohjelmointikieltä, joita käytetään yleisesti verkkokehityksessä. TypeScript on suunniteltu korjaamaan joitain JavaScriptin rajoituksia, mukaan lukien staattisen tyyppijärjestelmän puute ja heikko tuki suurille sovelluksille. TypeScript ei korvaa JavaScriptiä vaan täydentää sitä. JavaScript on edelleen laajimmin käytetty ohjelmointikieli verkkokehityksessä, ja TypeScript on yksinkertaisesti tapa parantaa sen ominaisuuksia. JavaScript on olio-ohjelmointikieli, jota käytetään interaktiivisten verkkosivujen ja verkkosovellusten luomiseen. TypeScript puolestaan on JavaScriptin muunnelma, joka ottaa käyttöön staattisen kirjoittamisen ja muita ominaisuuksia. Yksi tärkeimmistä eroista näiden kahden kielen välillä on niiden lähestymistapa tyyppitarkistukseen. JavaScriptissä tyyppitarkistus tehdään dynaamisesti ajon aikana, mikä voi johtaa virheisiin, joita on vaikea havaita. TypeScript puolestaan suorittaa staattisen tyyppitarkistuksen käännöshetkellä, mikä voi auttaa havaitsemaan virheet ennen kuin koodi on edes suoritettu (Bogner ja Merkel: 2020). Toinen ero näiden kahden kielen välillä on niiden syntaksi. Vaikka JavaScript käyttää syntaksia, joka on samanlainen kuin C, TypeScript käyttää syntaksia, joka on lähempänä Javaa tai C#:a. TypeScript sisältää myös ominaisuuksia, kuten rajapintoja, luokkia ja moduuleja, joita ei ole JavaScriptissä. Nämä ominaisuudet voivat auttaa tekemään koodista modulaarisemman ja helpomman ylläpitää. Lopuksi TypeScript voidaan pitää kehittäjäystävällisempänä kielenä, koska se pystyy tarjoamaan paremman koodin valmistumisen ja virheiden havaitsemisen kehityksen aikana. On kuitenkin tärkeää huomata, että TypeScript vaatii lisäasetuksia

ja määrityksiä JavaScriptiin verrattuna. Kaiken kaikkiaan molemmilla kielillä on vahvuutensa ja heikkoutensa, mutta näiden kahden välisten erojen ymmärtäminen voi auttaa kehittäjiä valitsemaan kielen, joka parhaiten sopii heidän tarpeisiinsa (Bogner ja Merkel: 2020).



Kuva 4 Kuvaus TypeScriptin ja JavaScriptin ominaisuuserosta

4.4 ESLint

ESLint on staattinen koodianalyysityökalu, jota käytetään koodin epäjohdonmukaisuuksien etsimiseen ja korjaamiseen JavaScript- ja TypeScript-ohjelmissa. Se on avoimen lähdekoodin projekti, joka käynnistettiin vuonna 2013 ja jota ylläpitää kehittäjäyhteisö. ESLintin ensisijainen tarkoitus on parantaa JavaScript- ja

TypeScript-ohjelmien koodin laatua pakottamalla johdonmukaisia koodaustyylejä ja havaitsemalla mahdollisia koodissa olevia vikoja ja virheitä. Tämä saavutetaan analysoimalla koodin syntaksi, tunnistamalla yleisiin virheisiin liittyvät mallit ja antamalla palautetta kehittäjälle. Työkalu voidaan mukauttaa vastaamaan projektin erityistarpeita määrittämällä sen säännöt ja laajennukset. Konfigurointi voidaan tehdä projektikohtaisessa konfiguraatiotiedostossa tai komentoriviliittymän kautta. ESLintistä on tullut suosittu työkalu JavaScript-yhteisössä sen joustavuuden ja helppokäyttöisyyden ansiosta.

4.5 HTML

HTML tulee sanoista HyperText Markup Language, joka on yleisimmin käytetty kuvauskieli verkkosivujen luomiseen. Tämä mahdollistaa osien, linkkien ja kappaleiden luomisen ja rakentamisen käyttämällä HTML-elementtejä, kuten tunnisteita ja attribuutteja. HTML ei ole ohjelmointikieli vaan enemmänkin verkkostandardi, koska sillä ei voi tehdä esimerkiksi dynaamisia toimintoja. Keskimäärin jokainen verkkosivusto sisältää useita HTML-sivuja, esimerkiksi yhteydenotto, info, etusivu olisi kaikki eri HTML-tiedostot.

```
<!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>
    <img src="images/firefox-icon.png" alt="kuvateksti" />
  </body>
</html>
```

Esimerkkikoodi 3 Perus-HTML-koodi

HTML on ollut käytössä jo 30 vuotta, ja nykyään käytössä on HTML5-versio, joka toi natiivin tuen mm. videon toistolle. HTML on kehittynyt vuosien varrella HTML 1.0:sta HTML 5:een vastaamaan verkkokehityksen kasvaviin vaatimuksiin. HTML 1.0 oli ensimmäinen versio HTML:stä, joka julkaistiin vuonna 1993.

Se oli hyvin peruskieli, jossa oli vain muutama tunniste ja attribuutti. Vuonna 1995 julkaistu HTML 2.0 esitteli lisää tunnisteita ja attribuutteja, mukaan lukien tuki taulukoille, kuvakartoille ja lomakkeille. HTML 3.2, joka julkaistiin vuonna 1997, oli merkittävä päivitys HTML:ään, ja se sisälsi monia uusia ominaisuuksia, kuten tuen tyylisivuille, taulukoita edistyneemmillä muotoiluvaihtoehdoilla ja parannettuja lomake-elementtejä. HTML 4.01, joka julkaistiin vuonna 1999, esitteli entistä enemmän ominaisuuksia, kuten kehyksien, iframe-kehysten ja multimedian tuen, ja sisälsi tiukemmat säännöt koodausstandardeille. HTML 5, joka julkaistiin vuonna 2014, on HTML:n uusin versio, ja se on suunniteltu tehokkaammaksi, joustavammaksi sekä monipuolisemmiksi nykyaikaiseen verkkokehitykseen. Se sisältää uusia tunnisteita ja attribuutteja, multimediaa ja lomake-elementtejä varten sekä tukea mobiililaitteille ja offline-sovelluksille. HTML:n kehitys on antanut verkkokehittäjille mahdollisuuden luoda edistyneempiä ja kehittyneempiä verkkosivustoja, jotka ovat nopeampia, helppokäyttöisempiä ja interaktiivisempia. Pitkän historian takia HTML on myös hyvin yleinen perinteisten verkkosivujen ulkopuolella. Esimerkiksi sähköpostit ovat usein HTML-pohjaisia sekä osa muistiinpanosovelluksista käyttää taustalla HTML-rakennetta.

Kuva 5 HTML:n historia

HTML-tiedoston pääte on .html tai .htm. Verkkoselain lukee HTML-tiedoston ja renderöi sen niin, että jokainen sivustolla kävijä voi tarkastella sitä. Jokaisella

HTML-sivulla on HTML-elementtejä, jotka koostuvat attribuuteista ja tageista. HTML-elementit ovat siis verkkosivun rakennuspalikoita, jonka avulla tunniste kertoo verkkoselaimelle, mistä tunniste alkaa ja mihin se loppuu, kun taas attribuutti kuvaa elementin ominaisuuksia. Toinen kriittinen osa HTML-elementtiä on attribuutti, jossa on kaksi osaa: nimi ja attribuutin arvo. Nimi identifioi käyttäjän lisätiedot, kuten taas attribuutin arvo antaa lisätietoja.

HTML-class, eli luokka, on myös hyödyllinen ja tärkein attribuutti ohjelmoinnin kannalta. Class-attribuutti lisää tyylitietoja, joita voidaan käyttää myös muissa elementeissä, kunhan on sama luokka-arvo. Useimmissa elementeissä on aloitus- ja lopetustunnisteet, mutta kaikki eivät tarvitse sitä toimiakseen, esimerkiksi tyhjät elementit, eivät käytä niitä sen takia, koska niillä ei ole sisältöä.

```
<img src=”/” alt=”Image”>
```

Esimerkkikoodi 4 Tällä tunnisteella on kaksi attribuuttia, src-attribuutti, joka on kuvan polku ja alt-attribuutti, joka kertoo kuvasta, jos sitä ei pystytä renderöimään tai jos käytössä näytönlukuohjelma. Tässä ei kuitenkaan ole lopputunnistetta tai sisältöä.

HTML on verkkosivujen ensisijainen merkintäkieli ja jokaisella HTML-sivulla on monta elementtiä, jotka luovat verkkosivuston tai sovelluksen rakenteen. Kieli on myös hyvin aloittelijaystävällinen ja tähän löytyy paljon ohjeita eri foorumeilta sekä internetistä ja tätä käytetään yleisesti staattisen verkkosivun luomiseen. Parhaiten voidaan hyödyntää CSS:n ja JavaScriptin kanssa.

4.6 CSS

CSS on toinen asia, jonka voi oppia heti, kun on ymmärtänyt HTML:n. CSS tarkoittaa CSS-tyylisivuja, jotka Hakon Wium Lie loi vuonna 1994. Hakon Wium Lietä pidetään CSS:n keksijänä. Hän työskenteli HTML:n kehittäjän Tim Berners-Leen kanssa CERNissä.

CSS:ää tarjottiin web-muotoilukieleksi, jotta se olisi houkutteleva. Se oli ratkaisu, jota suurin osa HTML:n käyttäjistä tuolloin katsoi. Kun se julkaistiin, käyttäjät saattoivat käyttää HTML 4.01:tä ja CSS:ää yhdessä tehdäkseen verkkosivuistaan houkuttelevampia ja visuaalisesti kauniimpia.

HTML:n ohella CSS:n standardoi myös W3C, joka hallitsee verkkostandardeja. CSS on vapaasti käytettävä, riippumaton, ja se on myös avoimen lähdekoodin standardi. Käyttäjät voivat käyttää HTML:ää ja CSS:ää yhdessä W3C:n kanssa. Jos katsomme CSS:n historiaa, huomaamme, että se on nähnyt pääasiassa kolme päivitystä. CSS-taso 1 julkaistiin vuonna 1996, ja se tuli uudelleen vuonna 1999 muutamilla parannuksilla. CSS-taso 2 tai CSS2 tuli kuitenkin vuonna 1998 mediatuella. Tässä versiossa käyttäjä voi hyödyntää muita mediaelementtejä. Lopuksi sen uusin versio CSS-taso 3 julkaistiin, ja tällä hetkellä käytämme vain tätä versiota.

```
body {
     background-color: lightgray;
}
 h1 {
     color: green;
     text-align: center;
}
 p {
     font-family: sans-serif;
     font-size: 16px;
}
```

Esimerkkikoodi 5 CSS-koodi

CSS1

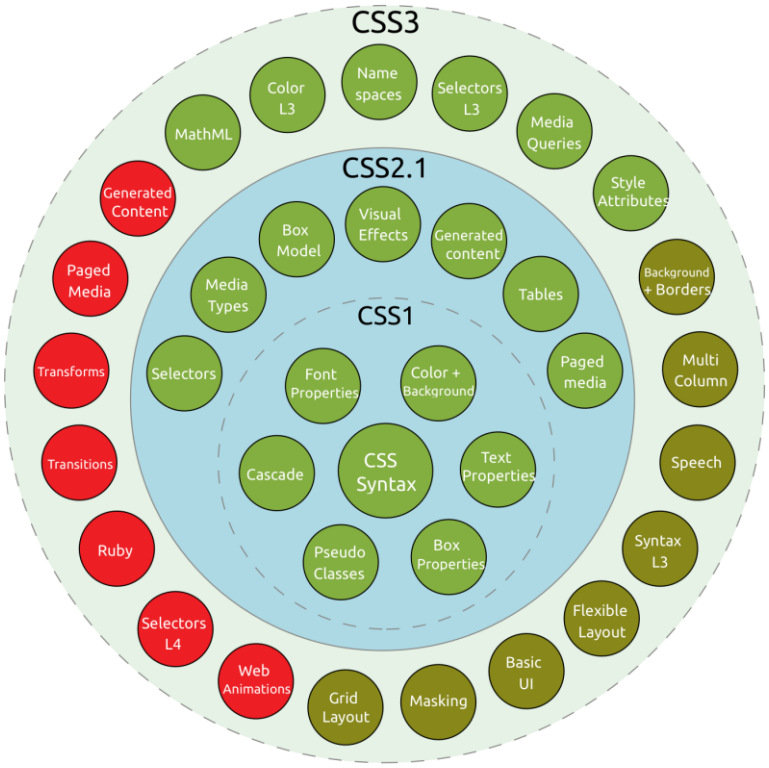
CSS 1 on ensimmäinen versio CSS-tyylitaulukosta ja W3C:n suosituksesta. Se lanseerattiin vuonna 1996 fonttiominaisuuksilla. Sitä käytetään myös värin lisäämiseen taustalle ja tekstipuolelle. CSS 1:ssä oli tekstin tasaustoimintoja. Siinä on myös pehmuste-, sijoittelu- ja yleisluokitteluominaisuudet. Nyt tämä versio on vanhentunut, eikä W3C ole enää ylläpitänyt sitä.

CSS2

W3C kehitti CSS:n seuraavan version ja antoi sille nimen CSS2, jonka se julkaisi vuonna 1998. Siinä on enemmän ominaisuuksia ja toimintoja kuin edellisessä versiossa. Nyt käyttäjät voivat käyttää uusia ominaisuuksia, kuten suhteellista, absoluuttista ja kiinteää paikannusta. Mediatyyppejä oli, ja siellä oli myös kaksisuuntaisia tekstiominaisuuksia.

CSS3

CSS3 on W3C:n virallisesti viimeisin versio CSS:stä, joka julkaistiin vuonna 1999. Siinä on laaja valikoima kirjasintyyppejä, ja voi käyttää mitä tahansa kirjasintyyppiä Googlesta ja Typecastista. Lisäksi tämä versio on jaettu useisiin moduuleihin, jotka helpottavat sen käsittelyä ja säästävät myös aikaa verkkosivujen muotoilussa. Tällä hetkellä useimmat yritykset ja organisaatiot käyttävät CSS3:a ja HTML5:tä verkkokehitykseen ja -suunnitteluun.



Kuva 6 CSS:n kirjasintyyppien kuvaus

CSS:n tulevaisuus

Aiemmin käytimme vain ensisijaista verkkosivustoa, jolla ei ole visuaalista tehostetta. Mutta CSS:n julkaisun jälkeen web-suunnittelun asiantuntijat ovat vieneet tämän tekniikan uudelle tasolle. Näin ollen ilman CSS-tyyliä verkkosivustoa ei näyttäisi houkuttelevalta ja mielenkiintoiselta käyttäjille. Olemme myös aiemmin huomanneet monia päivityksiä ja muutoksia samassa. Näin ollen myös tulevaisuudessa voi olla mielenkiintoisia päivityksiä. Mutta tällä hetkellä CSS3 on uusin versio.

5 Kurssin luonti

Työn tilaajana toimi Metropolia Ammattikorkeakoulu ja tavoitteena oli luoda Vue.js:n perusteet -kurssi. Lähdimme toteuttamaan tätä Metropolian olemassa olevaan Moodle-ympäristöön. Kurssi lähtee alkuun teorialla, josta sukelletaan suoraan koodaustehtäviin. Koska kyseessä on frontend-kirjasto, asetimme lähtövaatimukseksi perustason osaamisen HTML- ja CSS-kielistä. Koska Vue.js 2 -tuki päättyy 31.12.2023, valitsimme kurssia varten Vue.js 3 -version. Tämä tuotti pieniä haasteita, koska uuden version dokumentaatio ei ole samalla tasolla, kuin vuonna 2016 julkaistun 2. version.

Koska kyseessä on non-stop-toteutus pyydettiin kurssin osaksi myös 40 kysymyksen monivalintakoe. Tässä valitsimme laajasti kysymyksiä eri moduuleista sekä opiskelijoille annenetuista linkeistä, mistä heidän piti etsiä lisätietoa.

5.1 Moduuli 1

Tässä ensimmäisessä luennossa käsitellään yleisesti Vue.js viitekehykstä. Vuen historiaa, sekä Vuen kehittäjiä, mitä tarkoitusta varten Vue on kehitetty, kuka kehittänyt ja koska kehittänyt. Nykypäivänä Vue.js on johtava moderni web-sovellusviitekehys. Tässä ensimmäisessä luennossa käsitellään myös Reactin, Noden ja Angularin erilaisuuksia sekä yhtäläisyyksiä sekä verrataan eri käyttötarpeita ja tekniikoita.

Tällä luennolla syvennytään enemmän itse Vue.js:n tekniikkaan ja itse viitekehykseen sekä siihen, mitä arkkitehtuuria Vue.js käyttää ja mitä kirjastoja viitekehys käyttää. Opiskelijan on tarkoitus oppia Vuen perusteet. Tarkoituksena on myös, että opiskelija osaa asentaa Node.js:n sekä tarvittavat kehittäjän työkalut Opitaan myös, miten luodaan ensimmäinen projekti, jota tullaan tarvitsemaan jatkossa. Opiskelija ymmärtää myös tarvittavat hyödylliset lisäosat ja kerrotaan myös muista IDE:istä. IDE:ksi valitsimme Microsoftin julkaiseman Visual Studio Coden tämän avoimen lähdekoodin sekä laajan ominaisuustuen vuoksi.

5.1.1 Visual Studio Code

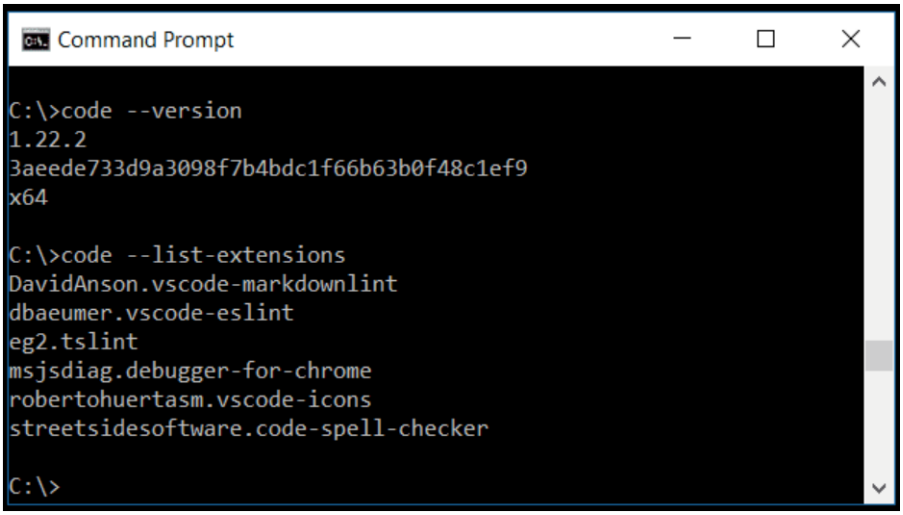
Visual Studio Code eli VS Code on lähdekoodieditori ja yksi maailman suosituimmasta kehittäjätyökaluista ohjelmoinnissa. Visual Studio Code julkaistiin 29. huhtikuuta 2015 Microsoftin 2015 Build-konferenssissa. VS Code on yhteensopiva lähes kaikkien työpöytäkäyttöjärjestelmien kanssa ja tarjoaa valmiin binäärin MacOS-, Linux- sekä Windows-käyttöjärjestelmiin ja sitä ylläpidetään aktiivisesti suuren suosion takia. Nykypäivänä sillä on päivittäisessä käytössä jo miljoonia käyttäjiä. Marraskuun 18. päivänä vuonna 2015 Visual Studio Coden lähde julkaistiin MIT-lisenssin alaisena saataville GitHubiin. Nykyään VS Coden lähdekoodi löytyy GitHubista MIT-lisenssillä.

VS Code on alun perin ohjelmoitu TypeScriptin ja JavaScriptin sekoituksena, nykyään ainoastaan TypeScriptillä. Myöhemmässä vaiheessa Coden liitettiin parannettu työkalutuki, kuten staattinen tyyppitarkastus ja koodin refaktorointi. VS Code tukee useampaa ohjelmointikieltä, muun muassa C++, C#, CSS, Dart, Dockerfile, F#, Go, HTML, Java, JavaScript, JSON, Julia, Less, Markdown, PHP, PowerShell, Python, R, Ruby, Rust, SCSS, T-SQL ja TypeSript. Lista tuettuja kieliä kasvaa jatkuvasti. Arkkitehtuurisesti Visual Studio Code yhdistää parhaat verkko-, natiivi- ja kielikohtaiset tekniikat. Electronin avulla VS Code yhdistää verkkoteknologiat kuten JavaScriptin ja Node.js:n natiivisovellusten nopeuteen. Visual Studio Codessa on myös erinomaiset debugging-työkalut, ja helpon GUI:n ansiosta ohjelmointi on vaivatonta.

5.1.2 Visual Studio Coden ominaisuuksia

Visual Studio Codessa on lukuisia hyödyllisiä ominaisuuksia. Esittelen niistä muutaman: komentorivin, tilapalkin, debugging-ominaisuudet eli ohjelmistovirheiden jäljitysen ja git-integraation.

Visual Studio Codessa on tehokas komentorivikäyttöliittymä, jonka avulla pystytään hallitsemaan editorin käynnistystä, avata erilaisia tiedostoja, asentaa laajennuksia ja jopa vaihtaa näyttökieltä käynnistyksen yhteydessä sekä tulostaa diagnostiikkaa. Komentorivin avulla voidaan myös tehdä etäkäyttötunneli.



Kuva 7 Visual Studio Coden käyttö komentorivillä

Tilapalkki Visual Studio Codessa sijaitsee työpöydän alaosassa, joka näyttää tietoja ja toimintoja, jotka liittyvät projektiin. Kohteet on jaettu kahteen eri ryhmään: ensisijainen sijaitsee vasemmalla ja toissijainen oikealla. Koko projektiin liittyvät varoitukset ja ongelmat näkyvät vasemmalla puolella ja kontekstuaaliset kohteet näkyvät oikealla: kieli, välit, palautteet.

Kuva 8 VSCode-tilapalkki

Kolmantena esittelen hyödyllisen git-integraatio ominaisuuden. Visual Studio Codessa on sisäänrakennettu git-tuki, mutta tämän käyttämiseen tarvitaan 2.0.0 tai uudempi git-versio, jotta integraatiota voidaan käyttää. Tärkeimpiä hyötyjä tästä ominaisuudessa ovat git-repositoryn alustaminen, repositoryn kloonaaminen, tunnisteiden luominen, muutoksien tekeminen, koodin pushaaminen, lataaminen ja synkronointi repoon sekä ratkaista yhdistämisongelmat. Lisäksi oikean repon näkee tilapalkista.

Viimeisenä, mutta ei vähäisimpänä ominaisuutena esittelen hyödyllisen debuggaus-ominaisuuden. Debuggaus, eli virheenkorjaus tarkoittaa virheiden poistamista koodista. Käytännössä debuggaus skannaa koodin ja etsii esimerkiksi

kirjoitusvirheitä tai syntaksivirheitä. Tämä työkalu on tehokas apu ohjelmoinnissa koodin virheiden etsimiseen ja korjaamiseen. Tämä työkalu on myös helppokäyttöinen.

Kuva 9 Visual Studio Coden Debug-palkki

5.2 Moduuli 2

Toisessa luennossa opiskelija aloittaa perinteisestä ”hello worldistä”, joka on ensimmäinen askel Vue.js-viitekehykseen. Tällä luennolla aloitetaan myös koko kurssin projekti. Opiskelijalla on tarkoitus tällä luennolla oppia komponentin luominen sekä miten se liitetään projektiin. Lisäksi tässä kerrataan HTML-kielen perusteita sekä aloitetaan JavaScriptin perusteet.

Luennon tarkoituksena on käsitellä listausta ja propsien käyttöä, mitä propsi tarkoittaa ja miten sitä voidaan hyödyntää projektissa. Projektissa käytetään myös juoksevaa uniikkia ID:tä ja miten se lisätään projektiin. Tämän luennon jälkeen opiskelijalla on perusymmärrys Vue.js-viitekehyksen propseihin, listaan ja uniikin ID:n luomiseen sekä komponenttimalleihin.

5.3 Moduuli 3

Edellisessä luennossa käsiteltiin yhtä komponenttia. Tässä kolmannessa luennossa käsitellään useampaa komponenttia ja sitä, miten niitä lisätään sovellukseen ja miten ne saadaan näkyviin loppukäyttäjälle Vue.app-sovellukseen. Lisäksi tässä luennossa opiskelija ymmärtää sen, mitä tarkoitetaan renderöinnillä.

Vuessa on mahdollisuus tehdä silmukka ja tarkoituksena on, että opiskelija ymmärtää silmukan hyödyntämistä sovelluksessa ja listan taulukossa sekä miten lisätään dataa listaukseen useamman eri komponentin avulla. Ideana on, että

se näkyisi erillisenä komponenttina. Opiskelija ymmärtää myös Vue-sovelluksen arkkitehtuuria ja rakennetta. Opiskelija ymmärtää myös avain-attribuutin tarkoituksen sovelluksessa ja mitä sillä voi tehdä. Luennon viimeisellä oppitunnilla käsitellään koodin refaktorointia eli koodin siistimistä.

5.4 Moduuli 4

Tällä luennolla käsitellään sitä, miten voidaan listaan lisätä omia tehtäviä, miten lista lähetetään, käsitellään ja hallitaan. Tällä hetkellä sovelluksessa ei voi päivittää itse tehtäviä koskematta koodiin. Tällä luennolla käsitellään myös lisää komponentteja ja niiden hallintaa, opitaan uusi menetelmä ja muistellaan HTML-listan tekoa. Opiskelija ymmärtää myös, miten sivusto lähetetään palvelimelle takaisin.

Opiskelijan tulee myös ymmärtää, miten lisätään uusi metodi tiettyyn tapahtumaan ja edellisellä oppitunnilla opittiin uniikkien ID:n lisääminen. Opitaan myös uusi tapahtumien syntaksi. Kerrotaan myös datan käsittelystä ja miten se sidotaan tietyllä komennolla sovellukseen sekä mitä tarkoitetaan v-mallilla. Luennolla myös käsitellään modifiointia ja sen menetelmiä, mikä se on ja missä tilanteessa sitä voidaan käyttää.

Luennolla käsitellään myös alikomponentteja ja miten niitä voidaan hyödyntää sovelluksessa. Kerrotaan myös v-on-menetelmästä HTML-lomakkeen avulla. Lisäksi kerrotaan console.log-menetelmästä: mitä se tekee ja mitä hyötyä kehittäjälle siitä on.

5.5 Moduuli 5

Tässä moduulissa opiskelija oppii CSS:n käyttöä sovelluksessa. Lisäksi opitaan se, miten webpack käsittelee eri tiedostoja ja miten käytetään CSS-esiprosessoreita ja jälkiprosessoreita. Tässä moduulissa myös käsitellään sitä, miten CSS-tiedosto lisätään olemassa olevaan sovellukseen.

Opiskelija oppii myös globaalien tunnisteiden käytön ja ymmärtää, mitä se tarkoittaa. Lisäksi tässä moduulissa käsitellään myös laajennettua CSS-tiedostoa ja sen muokkaamista ja kenttien grafiikka muokataan loppukäyttäjälle mieluisaksi.

5.6 Moduuli 6

Tämän moduulin tavoitteena on, että opiskelija ymmärtää laskennallisia ominaisuuksia ja laskurin käytön Vue-sovelluksessa, miten ne toimivat ja suoriutuvat. Tarkoituksena on oppia myös se, miten laskuria hyödynnetään sovelluksessa ja mitä ongelmia se aiheuttaa, jos kyseessä on suurempi sovellus.

Kuva 10 Sovelluksen laskurin käyttöä

Opiskellaan, miten sivu renderöityy laskurin muuttuessa ja hyödyntäen konsolin lokitusta. Opitaan myös mallien eli templaten muokkaaminen laskuria hyödyntäen. Tässä oli myös tavoitteena, että opiskelija saisi käsityksen, miten luodaan eri tapahtumia ja miten ne saadaan päivittymään sivuille, kun käyttäjä valitsee tehtävän tehdyksi ja v-mallin hyödyntämistä toisella tapaa, miten aiemmin on käsitelty eli miten seurataan tilaa.

5.7 Moduuli 7

Tässä moduulissa on tarkoitus käsitellä Vuen ehdollisia renderöintiominaisuuksia ja hyödyntää niitä sovelluksessa, sekä saada perusymmärrys, mitä ne ovat ja miten niitä käytetään eli v-if- ja v-else-ominaisuudet. Lisäksi tarkastellaan eri toimintojen hyödyntämistä erilaisissa tilanteissa ja miten niitä olisi järkevää käyttää. Lisäksi käsittelemme myös arvojen muuttamista.

Käsittelemme tässä moduulissa myös komponenttien ehdollista näyttämistä sivuilla v-if:n ja v-elsen avulla ja hyödynnetään enemmän lohkojen renderöintiä.

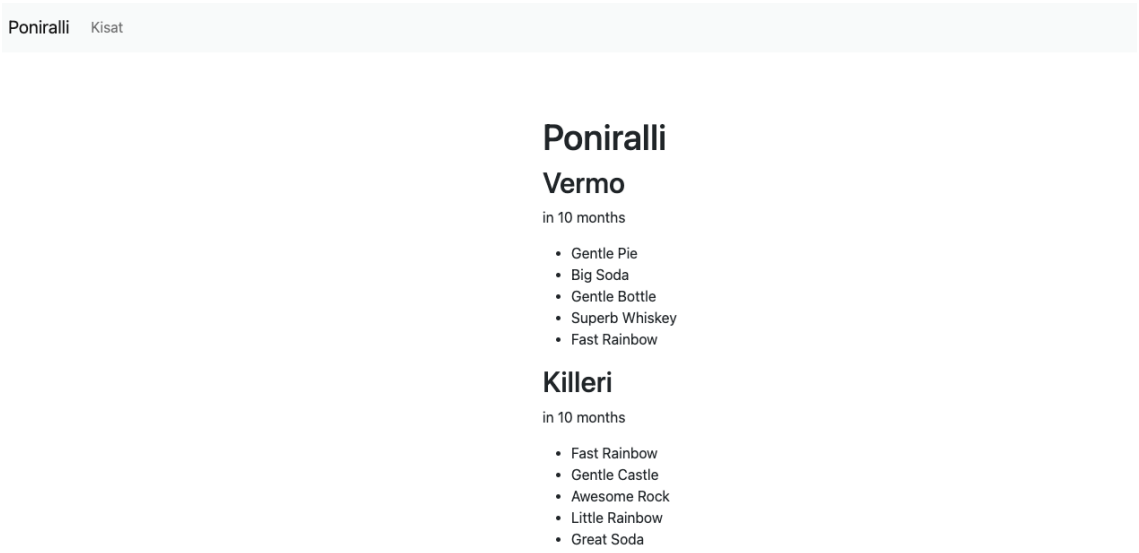
Tapahtumakäsittelijöiden lisäämistä ja olemassa olevien tapahtumien muokkaamista. Perehdytään myös tarkemmin eri alkioiden laskemiseen ja niiden ymmärtämiseen sovelluksessa sekä siihen, miten niitä käytetään. Lisäksi tavoitteena on, että opiskelija ymmärtää myös, mitä tarkoittaa natiivi tapahtumaobjekti ja missä sitä on järkevää käyttää.

5.8 Heppaseikkailu

Viimeisessä moduulissa tavoitteemme oli pintapuolisesti käydä läpi TypeScriptiä. Koska Vue.js voi kirjoittaa myös sillä JavaScriptin sijaan, on hyvä käydä läpi perusteita opiskellessa myös se läpi. Koska edellisissä moduuleissa on käyty laajalti läpi oman koodin kirjoittamista ja teoriaa, painottuu TypeScript-osuus enemmän valmiiseen validoitavaan koodiin. Tämän takia myös tehtäviin rakennettiin unit-testaus uutena oppina. Otimme tähän mukaan ESLint-kirjaston eli staattisen koodianalyysin lisäoppina, miten erilaisia työkaluja voidaan hyödyntää.

Opetus käynnistyy aivan perusteita luomalla uusi projekti tietyillä lisäominaisuuksilla ja asentamalla tämän riippuvuudet, josta tarjosimme valmiin listan. Käymme myös heti alkuun läpi Bootstrap-kirjaston ja miten se helpottaa sovelluksen graafisen ilmeen luontia. Tästä siirrymme sujuvasti luomaan ensimmäistä komponenttia käyttämällä TypeScript-kieltä sekä hyödyntämällä Bootstrap-kirjaston ominaisuuksia luoda reaktiivisia komponentteja helposti.

Seuraavaksi syvennytään datamalleihin ja näkymien luontiin. Tämä on ensimmäinen kunnon kosketus TypeScriptin kirjoittamiseen sekä miten eri tiedostoja voidaan tuoda import-toiminnolla osaksi sovellusta tai vaikka toista datamallia. Lisäksi käymme läpi, miten käytetään laskennallisia arvoja osana datamallia. Tästä saadaan lopputuloksena (kuva 11) hyvä kokonaisuus perusteita, joista on helppo jatkaa omiin sovelluksiin tai jatkokehittämään esimerkkisovellusta pidemmälle.



Kuva 11 Valmis sovellus

Tämän moduulin luonti oli aika työläs johtuen unit-testauksen lisäämisestä varmistamaan, että harjoitukset tulee tehtyä oikein. Myös TypeScrptin kirjoittaminen oli uutta, mutta positiivinen kokemus verrattuna puhtaaseen JavaScriptiin. Päätimme, että perusteiden kurssissa annetaan opiskelijoille paljon komponentteja valmiiksi kirjoitettuna, jotta työkuorma on järkeen käyvä ja ei sukelleta liian syvälle, miten esimerkiksi kirjoitetaan omat testaustiedostot tai miten konfiguroidaan ESLint toimimaan.

6 Yhteenveto

Opinnäytetyössä suunniteltiin ja kehitettiin Vue.js-kurssi Moodle-ympäristöön Metropolian opiskelijoita varten. Tarkoituksena on, että kurssi julkaistaisiin monimuoto-opintoihin, vapaasti valittaviin opintoihin, avoimiin AMK-opintoihin ja väylä-opintoihin. Kurssi on todettu toimivaksi ja lähtönä Vue.js-viitekehyksen perusteisiin. Käytimme tässä apuna kahta Metropolian opiskelijaa tarkistamaan sisällön sekä suorittamaan loppukokeen.

Kurssilla käsitellään Vue.js-perusteita sekä omien komponenttien tekemistä ja hyödyntämistä omassa sovelluksessa. Kurssin opiskelijalta toivotaan pientä ohjelmointitaustaa jo entuudestaan, koska kurssilla ei käydä läpi esimerkiksi HTML:n perusteita vaan oletetaan, että opiskelijalla on perusteet hallussa. Kurssin suorittamiseen riittää siis perusohjelmointiosaaminen. Työ on suunniteltu opettamaan perusteet Vue-ohjelmointiin ja eri ohjelmointikielien kertausta.

Opiskelijan näkökulmasta kurssi on toteutettu askel kerrallaan-menetelmällä ja helposti seurattavaksi moduuli kerrallaan. Teoria-osuus on tehty helpoksi ja ymmärrettäväksi myös unit-testien osalta.

Työn teoria oli laajalti jo työelämän ja harrastuneisuuden kautta tuttua. Uuttakin oppia tuli paljon ja oman tiedon tarkistamista luotettavista lähteitä tarvitsi tehdä. Käytännön toteutuksen osalta Moodle-kurssin suunnitteleminen ja toteuttaminen tulivat uutena.

Lähteet

Bogner, J. ja Merkel, M. 2022. To Type or Not to Type? A Systematic Comparson of the Software Quality of JavaScript and TypeScript Applications on GitHub. arXiv.

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

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

Goswami P, Gupta S, Meng S. 2020. Investigating the reproducibility of npm packages. IEEE.

Kaneriya, Tejas. 2022. What is Node.js? Where, When & How To Use It. Verkkoaineisto https://www.simform.com/blog/what-is-node-js/#section2 Luettu 02.04.2023.

Laaziri, Majida ym. 2019. A Comparative study of PHP frameworks performance. Science Direct.

Novac, Ovidiu Constantin ym. 2021 Comparative study of some applications made in the Angular and Vue.js frameworks. IEEE.

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

Sinclair, Andrew. 2010. Licence Profile: Apache License, Version 2.0. DOI:10.5033/ifosslr.v2i2.42.

Todavchich, Slava. 2020. Understanding open-source and free software licensing. Moqud.