Tampereen ammattikorkeakoulu

JavaScript-kirjastojen vertailu web-kehityksessä

Joni Jyrinki

OPINNÄYTETYÖ
Kesäkuu 2025

Tietojenkäsittely
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Ohjelmistotuotanto

JYRINKI, JONI
JavaScript-kirjastojen vertailu web-kehityksessä

Opinnäytetyö 26 sivua, joista liitteitä 0 sivua
Kesäkuu 2025

Työn tavoitteena oli vertailla moderneja web-kehityskirjastoja. Vertailuun valittiin React ja Vue tarkoituksena selvittää miten teknologiat eroavat eri osa-alueiden, kuten komponenttirakenteen, suorituskyvyn, syntaksin ja kehittäjäkokemuksen osalta. Lisäksi työhön koottiin molemmista teknologioista yleiskatsaukset, joiden tehtävänä on helpottaa sovelluskehittäjiä valitsemaan työhönsä soveltuvampi teknologia.

Osa-alueiden vertailun tueksi kehitettiin kaksi identtistä sovellusta. Molemmista teknologioista löytyi sekä vahvuuksia ja heikkouksia. Vue soveltuu erittäin hyvin pieniin projekteihin sekä aloitteleville kehittäjille sen selkeän komponenttirakenteen ja dokumentoinnin takia, mutta koska sillä on pienempi yhteisö eri laajennukset ja ulkopuoliset työkalukirjastot jäävät vajaiksi. React puolestaan on sopiva erityisesti laajempiin ja monimutkaisempiin projekteihin laajan ekosysteeminsä ja vapaamman arkkitehtuurinsa takia, mutta käyttöönotto ja syntaksi saattavat olla haastavia etenkin uudelle React-kehittäjälle.

Molemmat teknologiat ovat erittäin hyviä vaihtoehtoja dynaamisten ja modernien web-sovellusten kehittämiseen. Lopullinen valinta teknologioiden välillä on riippuvainen pääasiassa projektin laajuudesta, tavoitteesta ja kehittäjien osaamisesta.

Asiasanat: javascript, react, vue, web-kehitys

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Software Development

JYRINKI, JONI
Comparison of JavaScript Libraries in Web Development

Bachelor's thesis 26 pages, appendices 0 pages
June 2025

The objective of thesis was to compare the React and Vue libraries and examine how they differ in areas such as component structure, performance, syntax, coding experience, and to provide an overview of technology selection for different development needs.

Two identical online store applications were created using both React and Vue. The applications were created using the same functional requirements. Vue proved to be more beneficial for beginners due to its clearer code structure and good documentation. However, its smaller community and external libraries and extensions may be more limited. React was shown to be more suitable for large-scale and complex projects due to its large ecosystem and flexible architecture, although its setup and syntax may be more challenging for beginners.

Both technologies are good choices for creating dynamic web applications, but the final choice primarily depends on the project’s objectives, size, and the developer’s level of skill.

Key words: javascript, vue, react, web development

SISÄLLYS

LYHENTEET JA TERMIT

AJAX	Asyncronous JavaScript and XML, tekniikka, jonka avulla voi keskustella taustaohjelman kanssa ilman sivun uudelleenlatausta.
API	Application Programming Interface, rajapinta, jonka kautta sovellukset keskustelevat toistensa kanssa.
CSS	Cascading Style Sheets, kieli, jolla voit määrittää verkkosivujen ulkoasun.
DOM	Document Object Model, Selaimen tapa esittää verkkosivun rakenne puumaisena mallina.
HTML	Hypertext Markup Language, verkkosivujen merkintäkieli rakenteen kuvaamiseen.
JSX	JavaScript XML, React-kirjastossa käytettävä syntaksi, joka mahdollistaa HTML:n ja JavaScriptin.
SPA	Single Page Application, toteutustapa, jossa sivu ladataan vain kerran ja tämän jälkeen vain päivitettävä tieto muuttuu.

1 JOHDANTO

Nykyajan web-kehitys painottuu pääosin JavaScriptiin, joka on keskeinen ohjelmointikieli modernien verkkosovellusten kehittämisessä. Erityisesti koodikirjastot, kuten React ja Vue, ovat muuttaneet tapaa, jolla käyttöliittymiä kehitetään. Näiden kirjastojen avulla voidaan tehdä dynaamisia, tehokkaita ja käyttäjäystävällisiä sovelluksia.

Työn tavoitteena on vertailla näitä teknologioita, ja etsiä niiden keskeiset erot, sekä tutkia mihin käyttötarkoituksiin kukin sopeutuisi parhaiten. Vertailusta on rajattu ulkopuolelle mobiilisovelluskehitys sekä muiden kirjastojen, kuten Angularin, vertailu. Aihe on rajattu näin, jotta vertailu pysyisi selkeänä ja kohdennettuna.

Työssä keskitytään vertailemaan kahta suosittua kirjastoa, Reactia ja Vueta, eri osa-alueiden kuten komponenttirakenteen, suorituskyvyn, syntaksin ja kehittäjäkokemuksen osalta. Työssä selvitetään teknologioiden keskeiset erot ja arvioidaan niiden soveltuvuus erilaisten web-sovellusten kehittämiseen.

Työssä vertaillaan myös kahta identtistä verkkosovellusta, joista toinen on toteutettu Reactilla ja toinen Vuella. Vertailu auttaa hahmottamaan eroja teknologioiden koodirakenteessa, kehityskokemuksessa ja ylläpidettävyydessä.

Lopuksi analysoidaan kehitystyön perusteella saadut tulokset ja esitetään johtopäätöksiä siitä, mihin käyttötarkoituksiin kumpikin kirjasto soveltuu parhaiten.

2 JAVASCRIPTIN HISTORIA

2.1 JavaScriptin synty ja kehitys

JavaScriptin kehityshistoria alkaa vuodesta 1995. JavaScriptin kehitti Brendan Eich, ja sen alkuperäinen tarkoitus oli antaa kehittäjille mahdollisuus tehdä sivustoista vuorovaikutteisia.

Brendan Eich kertoi dotJS-konferenssissa vuonna 2017 kirjoittaneensa ensimmäisen JavaScript prototyypin 10 päivässä. Hän tiesi jo alusta asti, että kieli joko kuolisi nopeasti tai eläisi pitkäaikaisesti (Eich, 2018).

JavaScript syntyi halusta tehdä verkkosivuista dynaamisempia, koska nettiyhteydet olivat huomattavasti hitaampia ja esimerkiksi pelkkä lomake vei aikaa koska validointi hoidettiin palvelimen puolella, joka tarkoitti sivun uudelleen lataamista, jos lomake oli väärin täytetty (Maricheva, 2023).

JavaScriptin kehitys jatkui merkittävästi, kun projekti siirtyi Mozillalle. Mozilla jatkoi JavaScriptin kehittämistä avoimen lähdekoodin periaatteella, jolloin useammat kehittäjät pystyivät osallistumaan kehitykseen. Samoihin aikoihin JavaScriptistä tuli ECMA-standardi, tämän tarkoituksena oli varmistaa toimivuus eri selaimissa, sekä yhdistää kielten eri versiot (Maricheva, 2023).

AJAXin yleistyminen oli tärkeä virstanpylväs JavaScriptin kehityspolussa, sillä se mahdollisti tiedon hakemisen ja sivuston päivittymisen asynkroonisesti, Iman verkkosivun uudelleenpäivitystä (Keith, Sambells, Drimmie, 2010, luku 1).

Projektien kasvaessa monimutkaisemmiksi alkoi esiintyä tarve hallittavammille ja tehokkaammille tavoille rakentaa käyttöliittymiä. Suora DOM-manipulointi alkoi olla hankala ylläpitää, etenkin nopeasti kasvavissa web-sovelluksissa. Tämä johti modernien web-kehityskirjastojen kuten Reactin, Vuen ja Angularin kehitykseen.

Kirjastot kuten React ja Vue antoivat kehittäjille mahdollisuuden rakentaa nettisivuja komponenttipohjaisesti, mikä paransi koodin uudelleenkäytettävyyttä, sekä

ylläpidettävyyttä. Nämä kirjastot toivat mukanaan virtuaalisen DOM:in, joka nopeutti käyttöliittymän päivityksiä ja helpotti tilanhallintaa.

2.2 Reactin kehitystausta

Reactin kehitti Facebook (nyk. Meta), ennen Reactin julkaisua sitä käytettiin sisäisenä työkaluna käyttöliittymän skaalauksessa sekä hallintaan. Tarve Reactille tehdä laajoista ja nopeasti muuttuvista käyttöliittymistä kuten Facebookin uutissyötteestä dynaamisempia (Osmani, 2023, luku 1; CultRepo, 2023).

React julkaistiin vuonna 2013. Monet kehittäjät pitivät Reactia aluksi hyvin epäselvänä, koska siinä sekoitettiin HTML:ää JavaScriptin sekaan, kun ennen nämä oli pidetty erillään (Sharp, 2016). Ajan myötä kehittäjät kumminkin huomasivat tämän edut kuten komponenttirakenne ja suorituskyky, joka auttoi Reactia nousemaan suureen suosioon.

2.3 Vuen kehitystausta

Vuen kehitti Evan You, joka halusi yhdistää Reactin ja Angularin parhaat puolet. Vue julkaistiin vuonna 2014 ja vetosi nopeasti kehittäjiin sen selkeän rakenteen ja matalan oppimiskynnyksen takia (You, 2024). Vuella ei ollut yksittäistä yritystä tukemassa kehitystä, mutta vahva kehittäjäyhteisön tuki alkoi kasvaa nopeasti julkaisun jälkeen.

Vuonna 2016 julkaistiin Vue 2.0, joka toi huomattavia muutoksia, kuten parannuksia virtuaaliseen DOM:iin, sekä työkaluista kuten Vue Router ja Vuex tuli virallisia Vue-tiimin ylläpitämiä kirjastoja. Päivityksen jälkeen Vue tuli suosituksi etenkin Kiinassa (You, 2016).

3 YLEISKATSAUS JAVASCRIPT-POHJAISEEN WEB-KEHITYKSEEN

3.1 JavaScriptin rooli nykyaikaisessa web-kehityksessä

Nykyaikaisessa web-kehityksessä JavaScript mahdollistaa toiminnallisen ja interaktiivisen käyttökokemuksen. HTML- ja CSS-koodi määrittää verkkosivun ulkoasun, mutta JavaScript tuo mukaan logiikan.

Nykyään JavaScript ei ainoastaan rajoitu web-kehitykseen, Node.js:n myötä myös palvelinpuolen kehitys on mahdollista. Tämä mahdollistaa sovellusten kehittämisen ainoastaan JavaScriptillä.

3.2 SPA

Single Page Application eli yksisivusovellus on toteutustapa, jolla verkko sovellus ladataan vain kerran ja tämän jälkeen sivunvaihdoksilla päivitetään vain tarvittavaa tietoa.

SPA:n toimintaa mahdollistaa AJAX, joka on web-kehitystekniikka, jossa verkkosovellus hakee sisältöä palvelimelta tekemällä asynkronisia HTTP-pyyntöjä ja käyttää uutta sisältöä päivittääkseen vain tarvittavat osat sivusta (MDN Web Docs, ei pvm.).

Tämä toteutustapa on erittäin suosittu, koska se nopeuttaa käyttäjien asiointia sovelluksessa huomattavasti. Käyttöliittymä reagoi nopeasti tiedon muuttumiseen, mikä parantaa myös mobiililaitteilla asiointia.

Siihen liittyy myös uhkia, kuten mahdolliset tietoturvariskit. Lisäksi hakukoneoptimointi voi mahdollisesti heikentyä, jos ei ole perehtynyt asiaan.

3.3 Komponenttipohjainen kehitys

Komponenttipohjainen kehitys on kehitystapa, jolla sovellus on jaoteltu uudelleenkäytettäviin komponentteihin, joita voi koota tarvittaessa kuin LEGO-palikoita. Tämä kehitystapa on oleellinen osa moderneissa web-kehitys kirjastoissa kuten React ja Vue (Macrae, 2018, luku B.; Abramov, 2024).

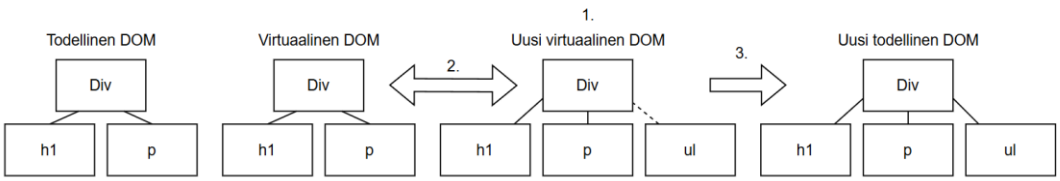
Komponenttipohjainen kehitys mahdollistaa sovelluksen jakamisen pienempiin uudelleenkäytettäviin osiin, joka parantaa koodin rakennetta, ylläpidettävyyttä, uudelleenkäytettävyyttä sekä helpottaa testaamista (Macrae, 2018, luku 2.).

3.4 Virtuaalinen DOM

Document Object Model (DOM) on verkkosivun rakennepuu, joka pitää sisällään verkkosivun HTML-koodin selaimelle ymmärrettävässä muodossa. (MDN Web Docs, ei pvm.) Ennen virtuaalista DOM:ia muutokset tehtiin suoraan oikeaan DOM:iin, mikä oli erittäin hidasta ja työlästä laajoissa sovelluksissa.

Virtuaalinen DOM on teknologia, joka tekee käyttöliittymän päivittämisestä mahdollisimman nopeaa. Kyseiset teknologiat eivät suoraan päivitä oikeaa DOM:ia. Sen sijaan ne (Kuvio 1):

1. Luo uuden virtuaalisen DOM:in
2. Vertailee uutta ja vanhaa Virtuaalista DOM:ia
3. Tekee tarvittavat muutokset



Kuvio 1. Virtuaalisen DOM:in päivitysprosessi.

3.5 Modernit työkalut ja kehitysympäristöt

Modernissa web-kehityksessä on monia työkaluja, jotka tukevat koodin laatua, kehitystyön sujuvuutta ja sovellusten suorituskykyä. Tässä osiossa tarkastellaan muutamia tärkeitä työkaluja.

ESLint on koodikirjasto, joka auttaa kehittäjää haivaitsemaan eri virheitä syntaksissa. ESLint pystyy myös ehdottamaan kehittäjälle parhaita käytäntöjä, jotta koodi olisi mahdollisimman luettavaa laajemmissa tiimeissä ja projekteissa.

Vite on Vuen kehittäjän Evan Youn luoma build-työkalu, joka helpottaa kehityksessä nopealla kehitysympäristön käynnistyksellä ja automaattisilla koodinpäivityksillä. Viten avulla myös tuotantoversion pakkaaminen on helppoa, sillä Vite optimoi projektin valmiiksi käyttäen Rollupia, joka yhdistää ja minimoi tiedostot automaattisesti (Vite Docs, ei pvm.).

Axios on koodikirjasto, jonka avulla käyttöliittymä keskustelee taustajärjestelmän kanssa. Axios helpottaa kehittäjää API-kutsujen tekemisessä ja hallinnassa.

4 REACTIN JA VUEN EROT JA VERTAILU

4.1 Syntaksin ja komponenttirakenteen erot

React ja Vue lähestyvät erityisesti komponenttirakennetta eri tavoilla. Vue jakaa komponentin kolmeen eri osaan: template, script ja style, jotka sijoitetaan samaan tiedostoon, tätä kutsutaan yksitiedostoiseksi komponentiksi.

```
1
2   <script>
3   export default {
4     data() {
5       return {
6         title: 'EASit Chair'
7       }
8     },
9     methods: {
10      doSomething() {
11      }
12    }
13  }
14  </script>
15
16  <template>
17    <div class="container">
18      <h1>{{ title }}</h1>
19      <button @click="doSomething">Lisää ostoskoriin</button>
20    </div>
21  </template>
22
23  <style scoped>
24  .container {
25    text-align: center;
26    padding: 20px;
27  }
28
29  button {
30    padding: 10px 20px;
31    font-size: 16px;
32  }
33  </style>
```

Kuva 1. Vue komponentti esimerkki.

Reactissa käytetään JSX (JavaScript XML) -elementtejä, jossa JavaScript ja HTML yhdistyvät funktion tai luokan sisällä. Tämä mahdollistaa monen komponentin tallentamisen samaan tiedostoon, vaikka tätä ei aina suositella.

```jsx
import React, { useState } from 'react';
import '../assets/styles/styles.css';

function Test() {
  const [title, setTitle] = useState('EASit Chair');

  const doSomething = () => {};

  return (
    <div className="container">
      <h1>{title}</h1>
      <button onClick={doSomething}>Lisää ostoskoriin</button>
    </div>
  );
}
```

Kuva 2. React komponentti esimerkki.

Reactissa voi myös lisätä toiminnallisuus itse onClick funktion sisään, tavalla mikä ei ole Vuessa mahdollista.

```jsx
import React, { useState } from 'react';
import '../assets/styles/Test.css';

function Test() {
  const [title, setTitle] = useState(0);

  return (
    <div>
      <h1>{title}</h1>
      <button
        onClick={() => {
          //doSomething
        }}
      >
        Lisää ostoskoriin
      </button>
    </div>
  );
}
```

Kuva 3. React yhdistetty kokonaisuus.

Kuten kuvasta huomaa, Reactin logiikka on huomattavasti tiiviimpi verrattuna Vueen, koska sen komponentit yhdistävät HTML- ja JavaScript-koodin yhdeksi kokonaisuudeksi, sekä CSS-koodi on erillään komponentin logiikasta ja tuodaan import-komennolla.

4.2 Tilanhallinnan ja reitityksen ratkaisut

Tilanhallinnalla tarkoitetaan käyttöliittymässä käytettävän tiedonhallintaa kuten käyttäjän lisäämää tekstiä tai napin tilaa. Paikallisen tilanhallinta löytyy molemmista Reactista sekä Vuesta, mutta globaali tila missä tiedot ovat sovelluksen eri osissa, vaatii Reactissa kolmannen osapuolen kirjaston.

```jsx
import React, { useState } from 'react';

function Testi() {
  const [count, setCount] = useState(0);

  const increase = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <p>Arvo: {count}</p>
      <button onClick={increase}>Kasvata</button>
    </div>
  );
}
```

Kuva 4. Paikallinen tilanhallinta Reactissa.

Kuvassa näkyy, miten Reactissa paikallinen tilanhallinta toteutetaan useState -hookin avulla. Napin painallus kutsuu increase-funktiota, joka kasvattaa count-muuttujan arvoa yhdellä. Vuessa sama toteutetaan ref-functiolla, joka luo reaktiivisen tilan, ja näkymä päivittyy nappia painettaessa.

```vue
<script setup>
import { ref } from 'vue'

const count = ref(0)

function increase() {
  count.value++
}
</script>

<template>
  <div>
    <p>Arvo: {{ count }}</p>
    <button @click="increase">Kasvata</button>
  </div>
</template>
```

Kuva 5. Paikallinen tilanhallinta Vuessa.

Reititys tarkoittaa sovelluksen sisällä tapahtuvaa navigointia eri sivujen välillä. Reititys on hyvin tärkeä yksisivu sovellusten kanssa sillä tämä mahdollistaa navigoinnin ilman sivujen kokonaista uudelleenlataamista.

Koska React on pohjiltaan vain käyttöliittymäkirjasto, eikä kokonainen ohjelmistokehys kuin Vue. React vaatii valintoja globaalin tilanhallinnan sekä reitityksen kanssa. React yhteisön sekä yritysten kehittämiä ratkaisuja löytyy monta, kuten esimerkiksi Redux tai Recoil tilanhallinta kirjastot ja React Router tai Next.js reititys kirjastot.

4.3 Kehittäjäkokemus ja oppimiskäyrä

Reactin ja Vuen kehittäjäkokemuksen erot näkyvät parhaillaan kirjastojen valinnassa sekä kehittäjän taustassa.

React tarjoaa enemmän vapautta sovelluksen kehittämisessä ja rakenteessa, mutta kehittäjän pitää tehdä valintoja eri koodikirjastojen kanssa. Kokenut kehittäjä hyödyntää tietotaitoansa ja valitsee sopivat kirjastot kehityksen alkuvaiheessa, kun taas aloittelevalla kehittäjällä voi ilmestyä ongelmia, kun yhteensopivuus tai kirjastojen käyttötavat eivät ole täysin hallinnassa, mikä johtaa kehityksen hidastumiseen.

Vue on täysi ohjelmistokehys, joka tarjoaa kaikki tarpeelliset työkalut kehittämisen aloittamiseen, sekä muistuttaa perinteistä HTML-kehitystä. Tämä tekee Vuesta hyvin aloittelijaystävällisen.

Vue ohjaa kehittäjiä käyttämään yhtenäistä rakennetta ja noudattamaan parhaita käytäntöjä sovellusten rakentamisessa. Tämä auttaa suuria tiimejä pitämään koodi yhtenäisenä. React antaa enemmän vapautta, mutta samalla asettaa kehittäjille enemmän vastuuta koodin yhtenäisyyden ja laadun säilyttämisessä.

4.4 Yhteisön ja tuen näkökulma

Reactilla on huomattavasti suurempi yhteisö kuin Vuella. Erot näkyvät esimerkiksi dokumentaation koossa, kolmannen osapuolen kirjastojen ja latausten määrässä. Suurempi yhteisö auttaa teknologian jatkuvassa kehityksessä, sekä auttavat kehittäjiä löytämään vastauksia ongelmiin nopeammin.

Kuva 6. Reactin(vas.) ja Vuen(oik.) viikoittaiset latausmäärät npmjs.com-sivuston mukaan (Kuva: npm, Inc).

Koska Vue on lähinnä yhteisön kehittämä, se on kerännyt aktiivisen yhteisön ympärilleen. Vue on panostanut aloittelijaystävällisyyteensä, joka näkyy erityisesti dokumentoinnissa, joka on järkevästi eroteltua ja alkaa perustaidoista. Vuen integroidut työkalut, kuten Vue Router ja Vuex vähentävät myös tarvetta ulkopuolisille kirjastoille, josta yhteisö pitää.

Verkosta löytyy monia eri keskusteluita, kuinka kehittäjät eivät pidä Reactista. Kritiikki painottuu pääosin siihen, että React ei ole täysi ohjelmistokehys ja vaativat ulkopuolisia kirjastoja ja kuinka kehittäjät eivät pidä JSX-syntaksista. Vuen kritiikki painottuu lähinnä vähäisten laajennusten, kirjastojen ja laajennusten olemassaoloon.

5 KIRJASTOJEN VERTAILU KEHITYSKRITEERIEN PERUSTEELLA

5.1 Skaalautuvuus ja ylläpidettävyys

Molemmat teknologiat ovat suunniteltu tukemaan komponenttipohjaista arkkitehtuuria, joka helpottaa sovellusten skaalautuvuutta ja ylläpitoa.

React projekteissa kehittäjillä on mahdollisuus itse valita ratkaisut esimerkiksi tilanhallinnalle, mikä tarjoaa joustavuutta suurten järjestelmien kehittämiseen. Tämä kuitenkin vaatii kehittäjältä enemmän suunnittelua ja yhteensopivuuksien tarkastelua, etenkin projektin alkuvaiheessa ja kirjastoja päivittäessä.

Lisäksi React on suunniteltu laajoihin ja jatkuvasti kehittyviin sovelluksiin, eikä sen syntaksi pakota kehittäjiä mihinkään tiettyyn tapaan rakentaa sovellus. Tämä mahdollistaa sovellusten optimoinnin projektikohtaisesti.

Vuella puolestaan on pienempi yhteisö, joten työkalujakin löytyy vähemmän, mutta Vue tarjoaa sisäänrakennettuja työkaluja, mikä auttaa pienien ja keskikokoisten projektien kehityksessä. Vuen komponentti rakenne huomattavasti ylläpidettävän, sillä komponentin kaikki osa-alueet löytyvät samasta tiedostosta.

Vuen sisäänrakennetut työkalut myös vähentävät kirjastojen päivittämisestä syntyviä yhteensopivuus ongelmia ja määrää. Tämä voi tehdä kirjastojen päivittämisestä huomattavasti vaivattomampaa kuin Reactissa.

5.2 Suorituskyky

Sekä Vue että React ovat molemmat hyvin optimoituja, mutta Vue kestää paremmin tilamuutoksia ja laajempia komponenttirakenteita yleensä paremmin kuin React ilman manuaalista optimointia.

React tarjoaa useita työkaluja, kuten useMemo tai useCallback, suorituskyvyn optimoinnille, mutta tämä vaatii kehittäjältä asiantuntemusta. Oikein käytettynä React toimii tehokkaasti etenkin suurissa ja monimutkaisissa sovelluksissa.

Suorituskyvyn arvioinnissa käytettiin Google Lighthouse- työkalua, joka on Googlen kehittämä avoimen lähdekoodin työkalu, sen avulla kehittäjät voivat arvioida sivujen suorituskykyä, saavutettavuutta, hakukoneystävällisyyttä ja yleistä laatua. (Google. ei pvm.) Lighthouse antaa jokaiselle osa-alueelle pisteytyksen, sekä parannusehdotuksia, joiden avulla voidaan optimoida verkkosivuja paremmin.

Lighthousella vertailtiin identtisiä sovelluksia sekä Vue- että React-versioina. Testien perusteella molemmat sovellukset suoriutuivat hyvin etenkin saavutettavuuden ja parhaiden käytäntöjen osalla (Taulukko 1). Suorituskyvyssä Vue sai hieman paremman pistemäärän ilman lisäoptimointeja kummassakaan.

Taulukko 1. Lighthouse -työkalun arvioimat pistemäärät osa-alueittain

Osa-alue	**React**	**Vue**
Suorituskyky	52	57
Saavutettavuus	100	100
Parhaat käytännöt	96	100

Myös teknologioiden tuotantorakennelmien koissa on selviä eroja. Esimerkiksi pelkän Reactin ja Vuen perusasennukset ilman ylimääräisiä kirjastoja eroavat toisistaan koon puolesta. Pakattu React vie tilaa noin 60 kilotavua, kun taas Vuen vastaava pakattu koko on noin 40 kilotavua. Kuitenkin on huomattava, että React mahdollisesti vaatisi ulkopuolisten kirjastojen lataamista kehityksessä.

5.3 Dokumentaatio

Vue tarjoaa selkeän ja aloittelijaystävällisen dokumentaation, joka on hyvin jäsenneltyä, sekä sisältää paljon koodiesimerkkejä. Tämän takia Vue on helposti lähestyttävä koodikirjasto niille, jotka eivät ole hirveästi tutustuneet JavaScript kehitykseen.

Reactin virallinen dokumentaatio on erittäin laaja, mutta myös hyvin tekninen, sekä monet projektit vaativat usein ulkopuolisten kirjastojen käyttöönoton. Tämä

vaatii perehtymistä eri kirjastojen dokumentointiin, joka johtaa siihen, että kehittäjän täytyy etsiä tietoa monista eri lähteistä.

Molemmille teknologioille löytyy kattava määrä yhteisön kehittämiä ilmaisia sekä maksullisia video oppitunteja, joiden avulla voi kehittää osaamistaan, mutta Vuelta löytyy virallinen sivusto (Vue School). Mikä sisältää kattavia kursseja kaikentasoisille kehittäjille.

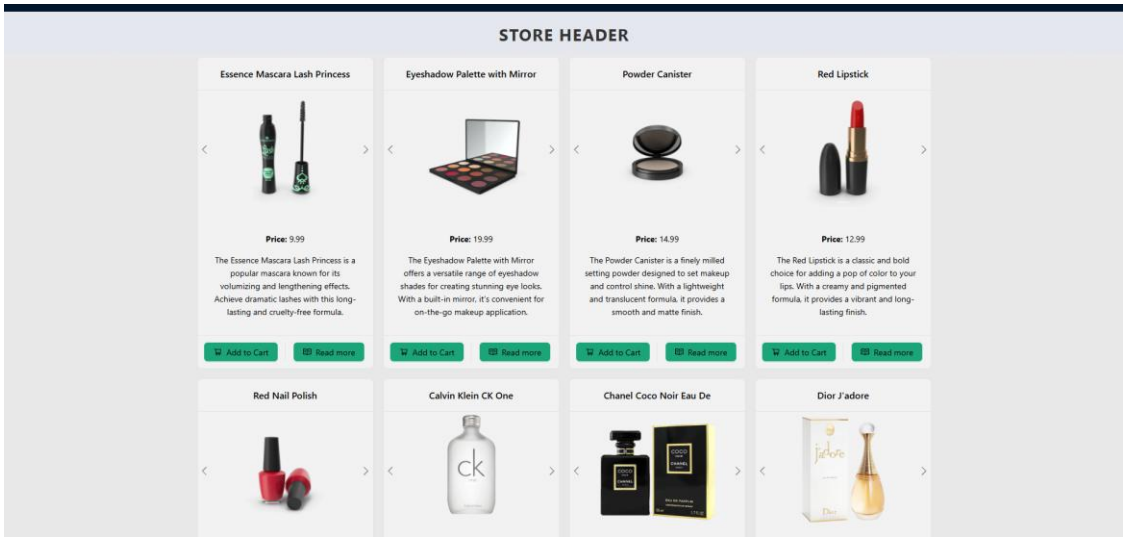
Esimerkiksi jos verrataan Vuen ja Reactin dokumentaatioiden lomakeosuuksia, voidaan huomata selviä eroja. Vue-dokumentaatio esittelee koodiesimerkkeinä kaikki yleisimmät lomaketoiminnallisuudet, sekä sivulta löytyy video aiheesta

React-dokumentaatio painottuu enemmän lomakkeen tilanhallinnan, virheiden käsittelyn ja optimistisiin päivityksiin, ja se ohjaa usein eri sivuille.

6 KÄYTÄNNÖN OSUUS

6.1 Sovellusten kuvaus ja esittely

Työssä toteutetut sovellukset ovat yksinkertaisia verkkokauppasovelluksia, joiden toiminnallisiin kriteereihin kuuluu tuotteiden selaaminen, tarkastelu ja lisääminen ostoskoriin. Sovellukset ovat kehitetty visuaalisesti identtisiksi, jotta teknologioiden vertailu olisi tasapuolista.



Kuva 7. Kuva sovelluksesta.

Sovelluksissa on käytetty Ant Design -tyylikirjastoa komponenttien luomisessa, mikä mahdollistaa yhtenäisen ja modernin ulkoasun molemmissa sovelluksissa, sekä Axios kirjastoa helpottamaan API-kutsujen hallintaa.

Reactissa reititykseen on käytetty React Router-kirjastoa, sovelluksissa määritettiin reitit, etusivu (/), tuotesivu (/product/:id) ja ostoskori (/cart), jotka liitettiin omiin komponentteihinsa. Reititys on toteutettu BrowserRouter, Routes ja Route-komponenttien avulla.

```
<>
  <ConfigProvider theme={theme}>
    <Navigation />
    <Routes>
      <Route path="/cart" element={<Cart />} />
      <Route path="/store/:id" element={<Product />} />
      <Route path="/" element={<Store />} />
      <Route path="/about" element={<About />} />
      <Route path="/contacts" element={<Contacts />} />
    </Routes>
  </ConfigProvider>
</>
```

Kuva 8. Reititys Reactissa.

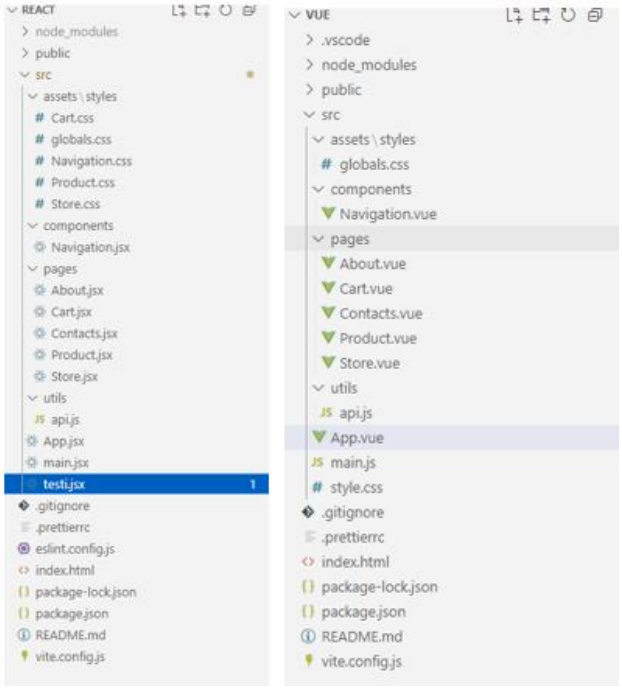
Vuessa reitityksen on käytetty Vuen virallista vue-routerin sisältämää createRouter funktiota samoilla reitityksillä.

```
const routes = [
  { path: '/', component: Store },
  { path: '/cart', component: Cart },
  { path: '/about', component: About },
  { path: '/contacts', component: Contacts },
  { path: '/store/:id', component: Product }
]

const router = createRouter({
  history: createWebHistory(),
  routes
})

export default router

createApp(App).use(router).use(Antd).mount('#app')
```

Kuva 9. Reititys Vuessa

6.2 Rakenne

Molemmat sovellukset ovat rakennettu parhaita käytäntöjä. Komponentit ovat jaettu omiin tiedostoihinsa, joka auttaa kehittäjää ylläpitämisessä sekä selkeyttää sovelluksien kokonaisrakennetta.



Kuva 10. React (vas.) ja Vue (oik.) sovelluksen rakenne

Molempien projektien tiedostorakenteet näyttävät pääosin samalta, mutta rakenteiden välillä on joitakin eroja (Taulukko 2).

Taulukko 2. React- ja Vue-projektien tiedostorakenne.

React	**Vue**
App.jsx: Sovelluksen pääkomponentti, joka pitää sisällään reititystä ja globaalia rankennetta	App.vue: pääkomponentti, johon näkymät upotetaan
Components/: sisältää yksittäiset komponentit kuten Navigation.jsx.	Components/: uudelleenkäytettävät komponentit
Pages/: näkymät eri sivuille, kuten tuotesivu tai ostoskori	Pages/: näkymät eri sivuille, kuten tuotesivu tai ostoskori sekä yhdistävät css-koodit
Utils/: API-kutsut	Utils/: API-kutsut
Assets/: sisältää sivujen CSS tiedostot	

6.3 Kehityskokemusten vertailu

Vuen selkeä tapa jäsentää tietoa tuki sovelluksen kehitystä, koska se teki tyylimuutoksista ja muusta komponenttien hallinnasta mahdollisimman helppoa, on

helppo hahmottaa, mihin mikäkin osa vaikuttaa ja missä mahdolliset ongelmat sijaitsevat. Vaikka Reactissa CSS-koodi on mahdollista sisällyttää .jsx tiedostoon, se ei kuulu Reactin parhaisiin käytäntöihin.

Reactissa puolestaan koodin selkeä ulkoasu sekä mahdollisuus yhdistää JavaScriptiä ja HTML-koodia saman komponentin sisällä osoittautuivat selkeäksi ja helposti ylläpidettäväksi. JSX-syntaksi oli selkeää ja tarjosi tehokkaan tavan rakentaa komponentteja. Tämä oli itselleni luonteva tapa kehittää, mahdollisesti aikaisemman kokemuksen ja tottumuksen vuoksi.

6.4 Koodin ylläpidettävyys ja luettavuus

Molemmissa sovelluksissa komponentit ovat jaettu omiin tiedostoihinsa, mikä parantaa koodin ylläpidettävyyttä ja luettavuutta, mutta Vuen yhden tiedoston komponentti rakenne pakottaa kehittäjän tähän ratkaisuun, mikä on tärkeää molempien teknologioiden parhaiden käytäntöjen näkökulmasta.

```
65    <a-layout-content class="cart-content">
66      <template v-if="cartItems.length > 0">
67        <a-card v-for="(product, index) in cartItems" :key="index" class="cart-item-card">
68          <a-row class="cart-item-row">
69            <a-col class="cart-item-image-col" >
70              <img :alt="product.title" :src="product.images[0]" class="cart-item-image" />
71            </a-col>
72            <a-col class="cart-item-info">
73              <a-typography-title :level="5">{{ product.title }}</a-typography-title>
74              <div>
75              <span class="bold-text">Price: ${{ product.price }}</span>
76              </div>
77              <div class="cart-item-quantity">
78              <span class="bold-text">Quantity:</span>
79              <a-input-number :min="1" :value="product.quantity" @change="(value) => updateQuantity(index, value)"
80                class="cart-item-input-number" />
81              </div>
82            </a-col>
83            <a-col class="cart-item-remove">
84              <a-button type="primary" danger @click="removeItem(index, product)" class="remove-button">
85                Remove
86              </a-button>
87            </a-col>
88          </a-row>
89        </a-card>
90      </template>
91      <template v-else>
92        <a-typography-text>Your cart is empty.</a-typography-text>
93      </template>
94    </a-layout-content>
95  </a-layout>
```

Kuva 11. Vue-komponentti rakenne ostoskorin näkymälle.

Koodiesimerkistä huomaa, kuinka luettava koodi olisi kehittäjälle, jolla on taustalla vain HTML-osaamista, sillä se muistuttaa lähemmin HTML-koodia.

Reactissa rakenne on myös selkeä, mutta tyyli tietojen erottelu eri tiedostoihin hajauttaa koodia enemmän (kuva 12). Tämä toi mukanaan enemmän siirtymistä tiedostojen välillä, mikä hidasti työskentelyä erityisesti projektin alkuvaiheessa.

```jsx
<Content className="cart-content">
  {cartItems.length > 0 ? (
    cartItems.map((product, index) => (
      <Card key={index} className="cart-item-card">
        <Row gutter={16} className="cart-item-row">
          <Col className="cart-item-image-col">
            <img alt={product.title} src={product.images[0]} className="cart-item-image" />
          </Col>
          <Col className="cart-item-info">
            <Title level={4}>{product.title}</Title>
            <span className="bold-text">Price: ${product.price}</span>
            <Space className="cart-item-quantity">
              <span className="bold-text">Quantity:</span>
              <InputNumber
                min={1}
                value={product.quantity}
                onChange={(value) => updateQuantity(index, value)}
                className="cart-item-input-number"
              />
            </Space>
          </Col>
          <Col className="cart-item-remove">
            <Button type="primary" danger onClick={() => removeItem(index, product)} className="remove-button">
              Remove
            </Button>
          </Col>
        </Row>
      </Card>
    ))
  ) : (
    <Text>Your cart is empty.</Text>
  )}
</Content>
```

Kuva 12. JSX-Rakenne ostoskorin näkymälle.

Molemmat lähestymistavat tukevat ylläpidettävää ja luettavaa kehitystä, mutta Vue tarjoaa selkeämmän ja luonnollisemman kokonaisrakenteen.

7 TULOSTEN ANALYYSI

7.1 Toteutuksen perusteella tehdyt havainnot

Toteutuksen aikana havainnoitiin, että molemmat teknologiat tarjoavat ajanmukaiset työkalut ja tukevat komponenttipohjaista kehittämistä, jotka ovat keskeisiä osia modernissa web-kehityksessä.

Reactin kehitystyö vaati alussa enemmän aikaa erityisesti erillisten kirjastojen valitsemisessa, koska React ei tarjoa sisäänrakennettuja ratkaisuja esimerkiksi reititykseen, kehittäjän tehtävänä on valita sopivat työkalut puuttuville osille kattavasta kirjasto- ja työkalu tarjonnasta. Tämä tuo esiin Reactin joustavuuden, mutta edellyttää samalla kehittäjältä osaamista ja aikaa vaihtoehtojen valintaan.

Koska Vue tarjoaa valmiiksi sisäänrakennettuja ratkaisuja tarpeisiin kuten reititykseen. Tämä vähensi tarvetta ulkopuolisten kirjastojen etsimiselle ja mahdollisti nopean siirtymisen itse sovelluksen kehitykseen.

7.2 Vahvuudet ja heikkoudet

Toteutus nosti esiin vahvuuksia ja heikkouksia molemmissa teknologioissa. Keskeiset huomiot kehitystyössä on koottu taulukkoon 3.

Taulukko 3. Reactin ja Vuen vahvuudet ja heikkoudet.

Vue	**Vahvuudet**	**Heikkoudet**
Hyvä ja helposti lähestyttävä dokumentointi sekä selkeät virheilmoitukset.		Pienempi yhteisö ja kirjastoekosysteemi verrattuna Reactiin.
Ei vaadi suuria määriä ulkoisia kirjastoriippuvuuksia.		Vähemmän joustavuutta monimutkaisissa projekteissa.
Matalampi oppimiskynnys		
React	**Vahvuudet**	**Heikkoudet**
Suuri yhteisö ja laaja kirjastoekosysteemi.		Vaatii usein kolmannen osapuolen kirjastojen käyttöönottoa.
Joustava arkkitehtuuri.		Rakenne hajautuu useisiin tiedostoihin mikä voi vaikeuttaa kehitystä ja ylläpitoa.
Soveltuvuus laajoihin projekteihin.		Korkeampi oppimiskynnys verrattuna Vueen.

8 JOHTOPÄÄTÖKSET JA POHDINTA

Molemmat teknologiat ovat merkittävässä asemassa nykyaikaisessa web-kehityksessä, sillä ne tarjoavat tehokkaita ratkaisuja modernien käyttöliittymien rakentamiseen. Näiden teknologioiden avulla kehittäjät pystyvät rakentamaan monipuolisia ja modulaarisia web-sovelluksia tehokkaasti, mikä nopeuttaa kehitysprosessia ja parantaa skaalautuvuutta.

Kehityskokemuksen perusteella Vue sopii erityisesti projekteille, joissa vanhoja HTML-sivustoja halutaan päivittää modernimpaan ja dynaamisempaan muotoon ilman tarvetta rakentaa koko sovellusta alusta asti. Vuen matala oppimiskynnys ja selkeä komponenttirakenne, jossa HTML, CSS ja JavaScript jaetaan omiin lohkoihinsa, tekee Vuesta erinomaisen valinnan tähän käyttötarkoitukseen. Vue on myös sopiva valinta kehittäjille, jotka vasta aloittelevat JavaScriptin käyttöä sovelluksissaan ja haluavat aloittaa matalalta kynnykseltä.

React on taas sopiva erityisesti laajempiin ja monimutkaisempiin projekteihin, vaikka käyttöönotto ja syntaksi saattaa olla haastava etenkin uudelle React-kehittäjälle, sen kehityksen joustavuus, ulkopuolisten kirjastojen määrä sekä suuren yhteisön tuki tekevät siitä erinomaisen valinnan.

Lopullinen valinta teknologioiden välillä riippuu pääasiassa projektin laajuudesta, tavoitteesta ja kehittäjien osaamisesta. Molemmat teknologiat ovat aktiivisesti ylläpidettyjä ja tulee olemaan pitkäikäisiä, joten valinta teknologioiden välillä ei ole niin mustavalkoinen.

Tulevaisuuden näkökulmasta on hyvä huomioida, että JavaScriptin merkitys voi laskea ja on mahdollista, että tulemme näkemään kehittäjien keskuudessa siirtymää TypeScript-pohjaisen kehityksen pariin. Tämä näkyy jo siinä, että kolmas suosittu web-kehityskehys Angular on siirtynyt täysin TypeScriptin varaan. Lisäksi molemmat React- ja Vue-yhteisöt suosittelevat TypeScriptin käyttöä kasvavassa määrässä sen tarkan ja selkeän syntaksin vuoksi.

LÄHTEET

Abramov, D. (2024). The Two Reacts. Haettu 8.5.2025. https://overreacted.io/the-two-reacts/

CultRepo. (10.2.2023). How A Small Team of Developers Created React at Facebook | React.js: The Documentary. YouTube. CultRepo (formerly Honeypot). https://www.youtube.com/watch?v=8pDqJVdNa44&ab_channel=CultRepo%28formerlyHoneypot%29

Eich, B. (6.8.2018). A Brief History of JavaScript [konferenssi]. YouTube. Tech Talk. https://www.youtube.com/watch?v=GxouWy-ZE80&ab_channel=TechTalk

Eich, B. (12.2.2021). Brendan Eich: JavaScript, Firefox, Mozilla, and Brave | Lex Fridman Podcast #160 [podcast-tallenne]. Lex Fridman. Youtube. https://www.youtube.com/watch?v=krB0enBeSiE&ab_channel=LexFridman

Google. (ei pvm.). Introduction to Lighthouse. Haettu 26.5.2025. https://developer.chrome.com/docs/lighthouse/overview

Keith, J., Sambells, J., Drimmie, R. (2010). *DOM Scripting: Web Design with JavaScript and the Document Object Model, Second Edition* (2nd ed.) *[e-kirja]*. friends of ED. https://learning.oreilly.com/library/view/dom-scripting-web/9781430233893/

Macrae, C. (2018). *Vue.js: Up and Running [e-kirja]*. O'Reilly Media, Inc. https://learning.oreilly.com/library/view/vue-js-up-and/9781491997239/

Maricheva, A. (3.10.2023). A Trip Back in Time: the History of JavaScript. Softteco. Haettu 19.5.2025. https://softteco.com/blog/history-of-javascript

MDN Web Docs. (ei pvm.) AJAX. Mozilla. Haettu 13.5.2025. https://developer.mozilla.org/en-US/docs/Glossary/AJAX

npm, Inc. (2025). React – npm package. Haettu 13.5.2025. https://www.npmjs.com/package/react

npm, Inc. (2025). Vue – npm package. Haettu 13.5.2025. https://www.npmjs.com/package/vue

Osmani, A. (2023). *Learning JavaScript Design Patterns, 2nd Edition [e-kirja]*. O'Reilly Media, Inc. https://learning.oreilly.com/library/view/learning-javascript-design/9781098139865/

React Docs. (ei pvm.) React – A JavaScript library for building user interfaces. Haettu 19.5.2025. https://react.dev/

Vite Docs. (ei pvm.) Vite Getting started. Haettu 19.5.2025. https://vite.dev/guide/

Vue.js Docs. (ei pvm.) Vue.js – The Progressive JavaScript Framework. Haettu 19.5.2025. https://vuejs.org/

Sharp, R. (15.9.2016). First impressions of react. Medium. Haettu 20.5.2025. https://medium.com/remys-blog/first-impressions-of-react-ef780fb33a60

You, E. (30.9.2016). Vue 2.0 is Here!. Medium. Haettu 20.5.2025. https://medium.com/the-vue-point/vue-2-0-is-here-ef1f26acf4b8

You, E. (7.2.2024). Evan You, Creator of Vue.js & Vite | Full Interview/Conversation | Recorded LIVE in Singapore. CS Dojo. YouTube. https://www.youtube.com/watch?v=nydRe2_qD4k&ab_channel=CSDojo