Elmo Mittilä

SELAINPUOLEN JAVASCRIPT-SOVELLUSKEHYSTEN DOKUMENTAATIOIDEN VERTAILU

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittelyn koulutus

2025

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä/Tekijät	Elmo Mittilä
Työn nimi	Selainpuolen JavaScript-sovelluskehysten dokumentaatioiden vertailu
Vuosi	2025
Sivut	38 sivua
Työn ohjaaja	Jukka Selin

TIIVISTELMÄ

Tämän opinnäytetyön tavoitteena oli vertailla selainpuolen JavaScript-sovelluskehyksien sovelluskehysdokumentaatioita. Selainpuolen JavaScript-sovelluskehyksiä käytetään käyttöliittymien luomiseen. Työ on pääpiirteisesti jakautunut kahteen osaan: teoriaosuus ja käytännön osuus. Työssä rakennettiin vertailutyölle teoriapohja tutustumalla ensin web-ohjelmoinnissa käytettyyn ohjelmointikieleen JavaScriptiin, jonka päälle vertailtavat sovelluskehykset on rakennettu. Teoriapohjan rakennus jatkui perehtymällä dokumentaatioon ja sovellusdokumentaatioon sekä suosittuihin sovelluskehyksiin ja niiden sovelluskehysdokumentaatioihin.

Kun teoriapohja oli saatu riittävän laajaksi, aloitettiin käytännön osuus. Teoriaosuudessa oli perehdytty viiteen eri selainpuolen teknologiaan. Käytännön osuus aloitettiin rajaamalla teknologioiden määrä kolmeen. Näiden kolmen sovelluskehyksen dokumentaatioihin paneuduttiin tarkemmin.

Teoriaosuuden aineenkeruussa löydettiin Divion-dokumentaatiojärjestelmä, johon sovelluskehysdokumentaatioita peilattiin käytännön osuudessa. Dokumentaatiojärjestelmä oli neljäosainen ja sitä käytettiin rajatun kolmen suositun sovelluskehyksen dokumentaation arviointiin. Neljä osuutta olivat tutoriaalit, how-to-oppaat, tekninen referenssi ja syventävä osuus. Arvioimalla näitä neljää osa-aluetta saatiin sovelluskehysdokumentaatioista kattava kuva, joka pystyttiin pisteyttämään.

Neljän eri osa-alueen pisteytyksen avulla saatiin kokonaisvaltainen kuva sovelluskehysten dokumentaatioista. Laskemalla yhteen kunkin sovelluskehyksen kullakin osa-alueella saamat pisteet päädyttiin tulokseen, että Angular-sovelluskehyksellä on vahvin dokumentaatio Divion-dokumentaatiojärjestelmään peilaten tarkastelun alaisista sovelluskehyksistä. Työn päätännössä arvioitiin vielä työn onnistumista, jatkokehitysehdotuksia ja tulosten vertailukelpoisuutta.

Asiasanat: dokumentointi, JavaScript, sovelluskehykset, verkko-ohjelmointi

South-Eastern Finland University of Applied Sciences

Degree title	Bachelor of Business Administration
Author (authors)	Elmo Mittilä
Thesis title	Comparison of front-end JavaScript framework documentation
Time	2025
Pages	38 pages
Supervisor	Jukka Selin

ABSTRACT

The objective of this thesis was to compare the documentation of front-end JavaScript frameworks. Front-end JavaScript frameworks are used to develop user interfaces for websites. The frameworks extend the basic techniques used in web development to enhance the development process and the end product.

The contents of this thesis can be split into two main parts. The first part consists of the conceptual basis. The main purpose of the conceptual basis is to build a proper understanding of the main subject areas necessary to perform the comparison of the documentation. The main subject areas are JavaScript, documentation and frameworks.

The second part of the thesis contains the comparison of the front-end JavaScript framework documentation. The comparison relied heavily on a documentation system model published by Divio, which was discovered during the data collection phase of the conceptual basis. Divio's system divides good documentation into four sections: tutorials, how-to guides, explanation and reference. The comparison evaluated the four sections of the documentation.

To gain a comprehensive view of the documentation quality, a numerical score was given to all of the four sections of each documentation. Based on the total scores of the studied documentation, the documentation of the framework Angular scored the highest. After the main parts, the thesis is finished off by offering thoughts on the success of the thesis and possible ideas for further research.

Keywords: documentation, framework, JavaScript, web development

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on arvioida ja vertailla suosittujen selainpuolen JavaScript-sovelluskehysten dokumentaatioita. Moni sovellus on käyttökelvoton ilman dokumentaatiota, mutta dokumentaatio ei takaa, että sovellus olisi käyttökelpoinen. Tarkoitus on siis paneutua syvemmin dokumentaatioihin ja selvittää, mitä dokumentaatio on, millaista on hyvä dokumentaatio ja millä hyvää dokumentaatiota voidaan mitata.

Idea työhön on syntynyt ajan myötä, kun vastaan on tullut paljon erilaisia sovellusdokumentaatioita. Sovelluksia on jäänyt käyttämättä huonon tai olemattoman dokumentaation takia, vaikka joskus huonoakin dokumentaatiota on tullut käytettyä. Dokumentaation laadun määrittely on tuntunut subjektiiviselta, joten näin tarpeen perehtyä asiaan tarkemmin.

Luvussa kaksi esitellään web-kehityksessä vakaan aseman saavuttaneen ohjelmointikielen JavaScriptin rooli web-kehityksessä ja kielen kehitystä. Työssä tarkasteltavat sovelluskehykset rakentuvat JavaScriptin päälle, joten sovelluskehyksiä ymmärtääkseen on tärkeä tuntea itse ohjelmointikieli. Luvussa kolme paneudutaan tarkemmin dokumentaatioon, sovelluskehyksiin ja vielä erityisesti sovelluskehysdokumentaatioihin. Kun kattava teoriapohja aihepiirille on rakennettu, siirrytään luvussa neljä arvioimaan sovelluskehyksiä tarkemmin. Lopputuloksena syntyy arvio kolmen sovelluskehyksen dokumentaation laadusta. Luvussa viisi käsitellään työn tekoon liittyneitä tuntemuksia ja havaintoja.

2 JAVASCRIPT

Luvussa esitellään webkehityksessä yleisin käytössä oleva ohjelmointikieli JavaScript. Ensiksi selvitetään mikä JavaScript on ja mikä sen funktio on. Tämän jälkeen paneudutaan ohjelmointikielen syntyyn ja suosioon. Lopuksi tutustutaan kielen kehitystä ohjaavaan standardiin ja sen kehitykseen.

2.1 JavaScript pähkinänkuoressa

JavaScript on kevyt ajonaikaisesti käännettävä ensimmäisen luokan funktioita sisältävä ohjelmointikieli, joka on parhaiten tunnettu verkkosivustojen kehityskielenä. Ensimmäisen luokan funktio tarkoittaa, että funktioita kohdellaan kuin muita muuttujia ja funktioita voidaan käyttää funktion argumenttina, palautusarvona sekä muuttujan arvona. Verkkosivujen kehitys ei ole kuitenkaan JavaScriptin ainoa käyttötarkoitus, vaan sitä hyödynnetään myös selainten ulkopuolisissa ympäristöissä. (Mozilla 2024.) Moderneihin verkkoselaimiin on sisäänrakennettu JavaScript-moottori, joka kykenee tulkitsemaan ja suorittamaan ohjelmakoodin ajonaikaisesti. Ohjelmakoodin suorittaminen muissa ympäristöissä, kuten palvelimilla, vaatii erillisen ajoympäristön, joista selvästi suosituin ja tunnetuin on Node.js.

Yksi tärkeimmistä verkkosivujen ominaisuuksista on verkkosivujen dynaamisuus. Staattinen sivu on yksinkertaista luoda hyödyntäen merkintäkieltä, kuten HTML, ja yhdistämällä siihen tyylittely esimerkiksi CSS:n avulla. Lisäämällä kehityksen teknologiapinoon JavaScriptin pystytään verkkosivun sisältöä päivittämään reaaliaikaisesti erinäisillä toiminnallisuuksilla ja onnistutaan luomaan parempi käyttökokemus. JavaScript on verkkosivukehitystä tehostava teknologia, jonka avulla voidaan luoda sitouttava, interaktiivinen ja älykäs käyttökokemus. (Goodman ym. 2010, 3.)

JavaScriptin synty

Ennen JavaScriptin olemassaoloa verkkosivujen toiminnallisuuksien täytyi kutsua verkkopalvelimilla sijaitsevia palvelinpuolen ohjelmakoodeja, mikä aiheutti suorituksessa oman viiveensä. Selainpuolen ohjelmakoodin suorittaminen nopeutti suoritusta huomattavasti, jolloin käyttökokemus kohentui. (McGrath 2020, 8.) JavaScriptin kehitti vain kymmenessä päivässä vuonna 1995 Netscapella työskennellyt Brendan Eich. Aluksi kieltä hyödynsi vain ajan suosituin verkkoselain Netscape Navigator, mutta teknologian helppokäyttöisyys ja ainutlaatuisuus ainoana selainpuolen komentokielenä ei jäänyt huomaamatta, ja muun muassa Microsoft omaksui kielen Internet Explorer -verkkoselaimeensa nopeasti. (GeeksforGeeks 2024a.)

Tuoreen kielen kasvattaessa suosiotaan nopeasti heräsi yhtiöiden välillä erimielisyyksiä lisensoinnin suhteen. Kieltä ja sen toimintoja kehitettiin useiden toimijoiden toimesta, jolloin syntyi erilaisia versioita ja pelko kielen fragmentoitumisesta. Vuonna 1997 Ecma International standardisoi kielen nimelle ECMAScript varmistaakseen kielen yhteensopivuuden säilymisen eri selaimille. (McGrath 2020, 8.) JavaScript on suosituin ECMAScript-standardin toteutus.

JavaScriptin suosio

Stack Overflow teetti vuonna 2024 kyselyn suosituimmista teknologiosta. Kyselyssä kysyttiin ohjelmointi-, komento- ja merkintäkielten laajamittaisesta käytöstä viimeisen vuoden aikana ja halusta käyttää seuraavan vuoden aikana. Vastanneet pystyivät valitsemaan useamman teknologian. Ammattia harjoittavista vastaajista (n=45 566) lähes kaksi kolmasosaa (64,6 %) valitsi kyselyssä JavaScriptin, mikä antaa hyvän kuvan siitä, kuinka suuri osa kehittäjistä on tekemisissä kielen kanssa. (Stack Overflow 2024.)

2.2 ECMAScript-standardi

Standardin kehityksestä vastaa vuonna 1996 perustettu tekninen komitea TC39. Kieli on kehittynyt standardin ohjaamana ja tuorein merkittävä versio, ECMAScript 6, julkaistiin vuonna 2015. Se saavutti täyden tuen kaikilla moderneilla verkkoselaimilla vuonna 2018. (W3Schools 2025a.)

Ensimmäinen merkittävä ECMAScriptin versio oli ECMAScript 3. Toinen versio julkaistiin lähinnä vain varmistamaan, että kieli mukaili kansainvälisiä standardeja, eikä kielen ominaisuuksiin vielä tehty muutoksia. Kolmas versio toi mukanaan muun muassa tehokkaat säännölliset lausekkeet, poikkeustilojen hallintaa, tiukemmat virheiden määrittelyt ja parempaa formatointia. Tämän version myötä ECMAScript saavutti massiivisen omaksunnan yhdessä World Wide Webin kanssa ja petasi näin asemaansa, jonka se on nykypäivänä saavuttanut. Versio julkaistiin kesäkuussa 2002. (Ecma International 2024.)

Kolmannen version onnistumisen myötä alettiin luonnollisesti kehittämään neljättä versiota. Versio ei kuitenkaan koskaan valmistunut, eikä sitä luonnollisesti julkaistu. Kehitystyö ei silti mennyt täysin hukkaan, sillä sitä pystyttiin

käyttämään myöhemmin osittain kuudennen version kehitykseen. Viides versio luokitteli selaintoteutuksissa yleistyneitä kielen tulkintoja ja lisäsi tuen kolmannen version julkaisun jälkeen ilmaantuneille ominaisuuksille. Ominaisuuksia olivat esimerkiksi paremmat taulukon käsittelyfunktiot, reflektiivinen objektien tarkastelu ja luonti sekä tiukka tila parantamaan virheiden tarkastusta ja ohjelman turvallisuutta. Viides versio hyväksyttiin kansainväliseksi standardiksi vuonna 2011. (Ecma International 2024.)

ECMAScript 6

Viimeisimmän merkittävän ECMAScript-version keskitetty kehitystyö aloitettiin vuonna 2009, kun viidettä versiota valmisteltiin julkaistavaksi. Todellisuudessa kuudennen version valmistumisen taustalla on ollut vahvasti jo viime vuosituhannen puolella aloitettu kokeellisuus ja ponnistelu kielen suunnittelun suhteen, ja valmistumisen voi nähdä myös jopa 15 vuoden vaivannäön kulminaationa. (Ecma International 2024.)

Version tavoitteena oli parempi tuki laajoille sovelluksille, kirjastojen luonti ja ECMAScriptin käyttö kääntäjänä muille kielille. Suurimpia version tuomia parannuksia olivat muun muassa luokkien esittely, moduulit, staattinen toiminnallisuuksien ulottuvuus, iteraattorit ja generaattorit sekä asynkroninen ohjelmointi lupausten avulla. ECMAScriptin sisäänrakennettua kirjastoa laajennettiin myös huomattavasti esimerkiksi tukemaan laajemmin datan abstraktiota. (Ecma International 2024.)

Standardin kehitys ECMAScript 6:n jälkeen

ECMAScript 6:n jälkeen tekninen komitea TC39 päätti aloittaa avoimen kehitystyön ja vuosittaiset versiojulkaisut. Komitea rakensi tekstimuotoisen lähdedokumentin ECMAScript 2015 -lähdedokumentista pohjaksi tulevalle GitHubissa näkyvillä olevalle kehitystyölle. (Ecma International 2024.) Vuosittaisten versioiden myötä standardin ja näin ollen kielen kehitys on jatkuvaa. Tuorein versio on julkaistu kesäkuussa 2024 nimellä ECMAScript 2024, painos 15. Painosnumero on suora jatkumo vanhanmalliseen versionumerointiin ja versio voidaan tuntea myös nimellä ECMAScript 15.

Tiiviimmän versiointitahdin myötä ECMAScript 6:n jälkeen ei ole ollut merkittävää yksittäistä versiota, mutta standardi ja ominaisuudet ovat kehittyneet vuosittain eteenpäin. Tuoreet ominaisuudet eivät tosin aina ole yhteensopivia verkkoselainten kanssa, joten verkkokehittäjän on hyvä olla tietoinen uusien ominaisuuksien yhteensopivuudesta. Tästä syystä ECMAScript 6 -versio pysyy edelleen puheenaiheena, sillä standardin noudattaminen takaa verkkosovelluksen kaikkien toiminnallisuuksien toimimisen moderneilla verkkoselaimilla.

3 DOKUMENTAATIO JA SOVELLUSKEHYKSET

Luvussa käydään läpi mitä on dokumentaatio ja millaista on hyvä dokumentaatio. Dokumentaation esittelyn jälkeen tutustutaan sovelluskehyksiin yleisellä tasolla ja tarkastellaan selainpuolen JavaScript-sovelluskehysten suosiota. Viiden suosituimman sovelluskehyksen ominaisuuksiin perehdytään tarkemmin.

3.1 Dokumentaatio

Dokumentaatiolla tarkoitetaan mitä tahansa jaettavaa materiaalia, jota käytetään kuvailemaan, selittämään tai ohjeistamaan liittyen johonkin asiaan, järjestelmään, prosessiin tai muuhun vastaavaan. Kyseessä voi olla esimerkiksi käyttö-, asennus- tai ylläpito-ohjeistusta, tai muuten asiaa kuvailevaa materiaalia. (The Linux Information Project 2006.) EU on säätänyt hyvän dokumentaatiokäytännön lääke- ja lääkinnällisten laitteiden teollisuuden alalle pitääkseen huolen laadunvarmistuksesta. Säädöksen tärkeitä tavoitteita on varmistaa dokumentaation eheys, saavutettavuus ja luettavuus. (Euroopan komission terveys- ja kuluttaja-asioiden pääosasto 2010.)

Vaikka dokumentaatio alojen välillä voi olla monin tavoin erilaista, nämä arvot soveltuvat hyvin myös IT-alan dokumentaatioihin. Dokumentaation eheys varmistaa, että tieto on paikkansapitävää ja ajantasaista. Saavutettavuudella taataan, että kehittäjälle on pääsy dokumentaatioon kokonaisuudessaan tilanteesta riippumatta. Luettavuuden myötä dokumentaatio on ymmärrettävää ja näin ollen sitä pystytään myös hyödyntämään. Parhaassa tapauksessa hyvä dokumentaatio voi säästää aikaa, rahaa sekä hermoja.

Strimlingin (2024) tekemän tekniseen dokumentaatioon liittyvän kyselytutkimuksen mukaan dokumentaation lukijat pitävät hyvälaatuisena dokumentaationa paikkansapitävää, ajantasaista, helppolukuista ja saavutettavaa dokumentaatiota. Tulokset ovat siis hyvin linjassa EU:n tavoitteleman dokumentaation tavoitteiden kanssa.

Sovellusdokumentaatio

Hyvä sovellusdokumentaatio koostuu neljästä osasta: tutoriaaleista, how-to-oppaista, teknisestä referenssistä ja syventävästä osiosta. Tutoriaalit ovat oppitunteja, jotka ohjaavat aloittelijaa kädestä pitäen koko projektin matkan. Niiden tarkoitus on näyttää aloittelijalle, mitä kyseisellä tekniikalla pystytään saavuttamaan. Tutoriaalit ovat täysin oppimiskeskeisiä. How-to-oppaiden päämääränä on ohjeistaa lukija ratkaisuun todelliseen maailmaan kohdistuvassa esimerkissä. How-to-opas olettaa lukijan jo tuntevan tekniikan perusteet ja vastaa kysymyksiin, joita aloittelija ei edes osaisi vielä muodostaa. How-to-oppaat ovat päämääräkeskeisiä. (Divio 2025.)

Tekninen referenssi on dokumentaation informaatiokeskeinen osio ja sen työ on kuvailla. Referenssi on koodipainotteinen, sillä se kuvailee tekniikan ratkaisuja, kuten luokkia, funktioita ja ominaisuuksia. Referenssimateriaali on yksinkertaista kattavaa asiasisältöä, joka vastaa kysymykseen mikä, eikä miten. Syventävä osio on ymmärryskeskeinen ja laajentaa dokumentaation kattavuutta tekniikan aiheisiin. Osio on usein sulautettuna muiden dokumentaation osa-alueiden sekaan, eikä omana osionaan. Syventyminen on luonnoltaan diskursiivista ja tarjoaa laajemman näkökulman tai jopa erilaisen perspektiivin. Syventymällä voidaan rakentaa laajempaa käsitystä aihealueeseen ja sitä kautta antaa parempi ymmärrys. (Divio 2025.)

3.2 Sovelluskehykset

Ohjelmoinnissa ohjelman rakenteen määrittää usein hyvin pitkälti käytettävä sovelluskehys. Sovelluskehys tarjoaa valmiita rakennuspalikoita, eli sen voi ajatella tarjoavan valmiita vastauksia yleisimpiin ohjelmointiongelmiin, jolloin kehittäjä pystyy keskittymään ohjelman yksilöllisiin tarpeisiin paremmin ja näin

ollen kehittää tehokkaammin (Tieturi 2024). Työssä keskitytään erityisesti selainpuolen web-sovelluskehyksiin. Web-sovelluskehyksen tarkoitus on tukea erityisesti web-sovellusten, kuten web-palveluiden, web-resurssien ja web-API:en, kehitystyötä (Goetz & Marquez 2023, 2). Shuklan mukaan (2023) JavaScript sovelluskehyksillä on ollut merkittävä rooli niin itse ohjelmointikielen, kuin myös kokonaisvaltaisesti web-kehityksen kehittymisessä. Nykypäivän responsiiviset ja interaktiiviset verkkosivustot ovat mahdollisia uudelleenkäytettävien, skaalautuvien ja helposti ylläpidettävien sovelluskehysten ansiosta. Myös käyttökokemus on parantunut huomattavasti tämän myötä. (Shukla 2023, 1–2.)

Stack Overflown 2024 teettämässä teknologiakyselyssä yhtenä tulosten kategoriana oli web-sovelluskehykset ja web-teknologiat. Vastaajilta tiedusteltiin, minkä web-sovelluskehysten tai web-tekniikoiden parissa he ovat tehneet laajamittaista kehitystyötä viimeisen vuoden aikana ja minkä parissa he haluaisivat seuraavan vuoden aikana työskennellä. Vastaajat pystyivät valitsemaan useamman vaihtoehdon. Ammattia harjoittavien vastaajien (n=38 132) viideksi suosituimmaksi selainpuolen JavaScript-teknologioiksi nousivat React (41,6 %), Angular (19,4 %), Vue.js (16,6 %), AngularJS (7,4 %) ja Svelte (5,9 %). (Stack Overflow 2024.)

3.2.1 React

React on vuonna 2013 julkaistu Metan, entinen Facebook, julkaisema ja ylläpitämä web-teknologia. React mielletään usein sovelluskehykseksi, mutta todellisuudessa kyse on kirjastosta. GeeksforGeeksin mukaan Reactin ydinkonsepti on luoda uudelleenkäytettäviä käyttöliittymäkomponentteja, joita pystytään hallitsemaan ja päivittämään. React on joustava käyttöliittymien kehityksen työkalu, joka auttaa käyttöliittymän renderöinnissä ja manipuloinnissa, mutta se ei tarjoa varsinaista rakennetta sovelluksen luomiseen. (GeeksforGeeks 2024b.)

Hunt (2013) kirjoitti blogissaan syistä, miksi React kehitettiin. Yhdeksi keskeiseksi pääkohdaksi hän nostaa sen, että perinteisesti websovellukset rakennettiin mallien päälle, mitkä asettivat rajoituksia kehitykselle. React pilkkoo käyttöliittymät komponenteiksi ja käyttää renderöintiin itse ohjelmointikieltä,

mikä mahdollistaa abstraktioiden luomisen ja helpottaa laajan sovelluksen kehitystä. Toisena pääkohtana on reaktiiviset dokumenttioliomallin päivitykset. React pitää itselleen yllä kevyttä representaatiota dokumentista. Representaation avulla voidaan kevyesti täsmäyttää käyttöliittymän tila ennen ja jälkeen muutoksen ja tehdä dokumenttioliomalliin vain tarvittavat muutokset. (Hunt 2013.)

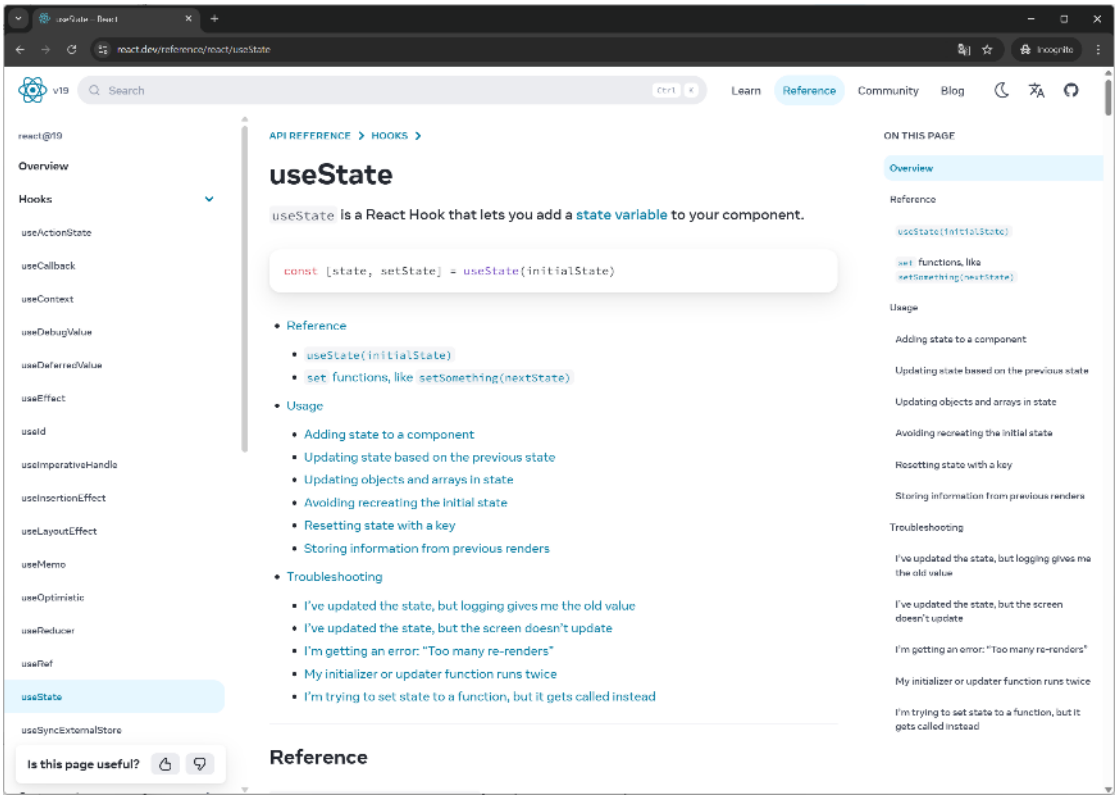
Reactia voi itsessään käyttää hyvin selainpuolen käyttöliittymäkehityksessä, vaikka se ei olekaan sovelluskehys. Reactin päälle on kuitenkin rakennettu sovelluskehyksiä, jotka tarjoavat paremman rakenteen kokonaiselle sovellukselle. Suosituin ja myös Reactin suosittelema kirjaston päälle rakennettu sovelluskehys on NextJS (React 2025).

NextJS

NextJS on React-sovelluskehys, joka on suunniteltu full stack -websovelluksille. NextJS abstraktoi ja määrittää automaattisesti Reactin vaatimat työkalut ja tarjoaa lisäominaisuuksia kehitystyöhön. Lisäominaisuuksia ovat esimerkiksi reititys, renderöinti, data fetching, tyylittely, optimointi ja paranneltu TypeScript-tuki. (NextJS 2025.)

Reactin dokumentaatio

Reactin dokumentaatio on löydettävissä Reactin virallisilta sivuilta (react.dev) ylänavigaatiosta otsikoilla Reference ja Learn. Reference-osio sisältää tarkan teknisen dokumentaation ja Learn sisältää enemmän opettavaista ja selittävää sisältöä. Rakenne molemmilla osioilla on sama (kuva 1). Sivun vasemmasta laidasta löytyy navigaatio, joka sisältää pääaiheita, jotka sisältävät ala-aiheita. Sivun oikeassa laidassa on otsikkotasolla tämänhetkisen sivun sisältö, mikä auttaa nopean yleiskuvan saamisessa aiheesta ja mahdollistaa myös suoran navigoinnin. Sivun keskiosasta löytyy aiheeseen liittyvät asiasisällöt tekstin ja koodiesimerkkien muodossa.



Kuva 1. Reactin dokumentaatiosivu

Dokumentaatioon on upotettu myös esimerkkejä, jotka ovat ladattavissa. Esimerkkejä pystyy myös muokkaamaan suoraan dokumentaatiosivulla tai koodin voi siirtää hiekkalaatikon puolelle paremman käyttökokemuksen tarpeessa. Dokumentaatiossa on käytettävissä hakutoiminto.

3.2.2 Vue

Vuen kehitti vuonna 2014 Evan You. Käytettyään aiemmin AngularJS-sovelluskehystä You näki tarpeen yksinkertaisemmalle ja helpommin opittavalle sovelluskehykselle. Tavoitteena oli kehittää kevyt, helposti opittava ja mukautuva käyttöliittymien luomisen ja yksisivuisten sovellusten sovelluskehys. (Flexiple 2025.)

Yksinkertaisuudessaan Vue on sovelluskehys käyttöliittymien rakentamiseen. Vue rakentuu perinteisten web-rakennuspalikoiden HTML, CSS ja JavaScript päälle ja tarjoaa deklaratiivisen komponenttipohjaisen ohjelmointimallin tehokkaaseen monimuotoiseen käyttöliittymäkehitykseen. Vuen ydinarvoja ovat deklaratiivinen renderöinti ja reaktiivisuus. Deklaratiivinen renderöinti onnistuu laajentamalla perinteistä HTML syntaksia, jolloin HTML-tulostus onnistutaan

perustamaan JavaScript tiloihin. Reaktiivisuus syntyy automaattisella JavaScript-tilojen seurannalla ja dokumenttioliomallin päivittämisellä. (Vue.js 2025a.)

Verkko on hyvin monimuotoinen, joten web-sovellusten muoto ja skaala vaihtelevat myös äärimmäisesti. Vue kattaa hyvin webkehityksen perustarpeita ja se on kehitetty joustavaksi ja inkrementaalisesti adoptoitavaksi. Käyttötapauksen mukaan sitä voidaan käyttää eri tavoin, kuten staattisen HTML:n tehostamiseksi, upotettuna komponenttina, yksisivuisena sovelluksena, palvelinpuolen renderöinnissä tai staattisten sivujen generoinnissa. Joustavuudesta huolimatta Vuen ydinosaaminen soveltuu kaikkiin käyttötapauksiin. Vue kuvaakin itseään progressiiviseksi sovelluskehitykseksi, joka kasvaa kehittäjän mukana ja mukautuu tarpeisiin. (Vue.js 2025a.)

Vue-komponentit rakentuvat yhden tiedoston komponentteina (kuva 2). Tiedostoon määritellään ylätason tagit <template>, <script> ja <style>, jotka jakavat tiedoston lohkoihin. Nämä lohkot toimivat komponentin päärakenteena. (Vue.js 2025b.)

```
<script setup>
import { ref } from 'vue'
const count = ref(0)
</script>

<template>
  <button @click="count++">Count is: {{ count }}</button>
</template>

<style scoped>
button {
  font-weight: bold;
}
</style>
```

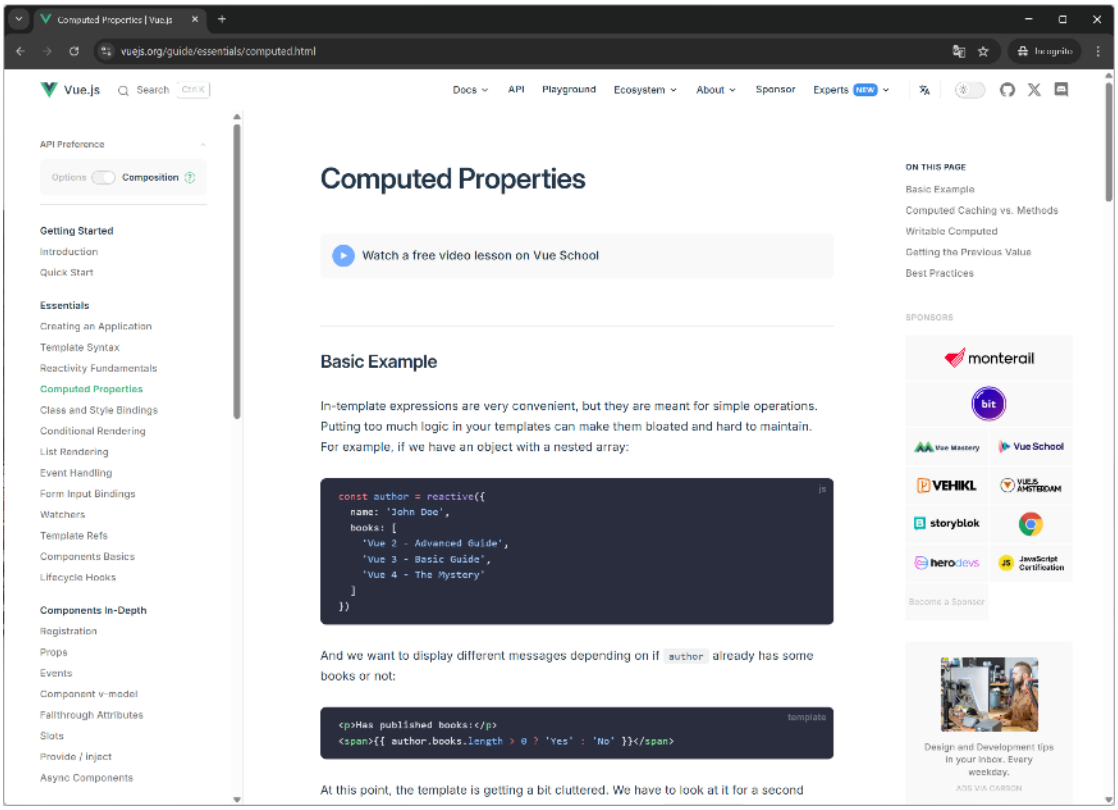
Kuva 2. Vuen yhden tiedoston komponentti

Ylätason tagien lisäksi komponenttiin on mahdollista lisätä kustomoituja tageja, esimerkiksi dokumentaatiota varten. Tagin <template>-lohkon sisältö välitetään @vue/compiler-dom:lle, valmistellaan JavaScript renderöintifunktioiksi

ja liitetään vietävän komponentin render-optioksi. Tagin <script> sisältö suoritetaan ECMAScript-moduulina. Tagiin <style> määritellään komponentin tyylit. (Vue.js 2025b.)

Vuen dokumentaatio

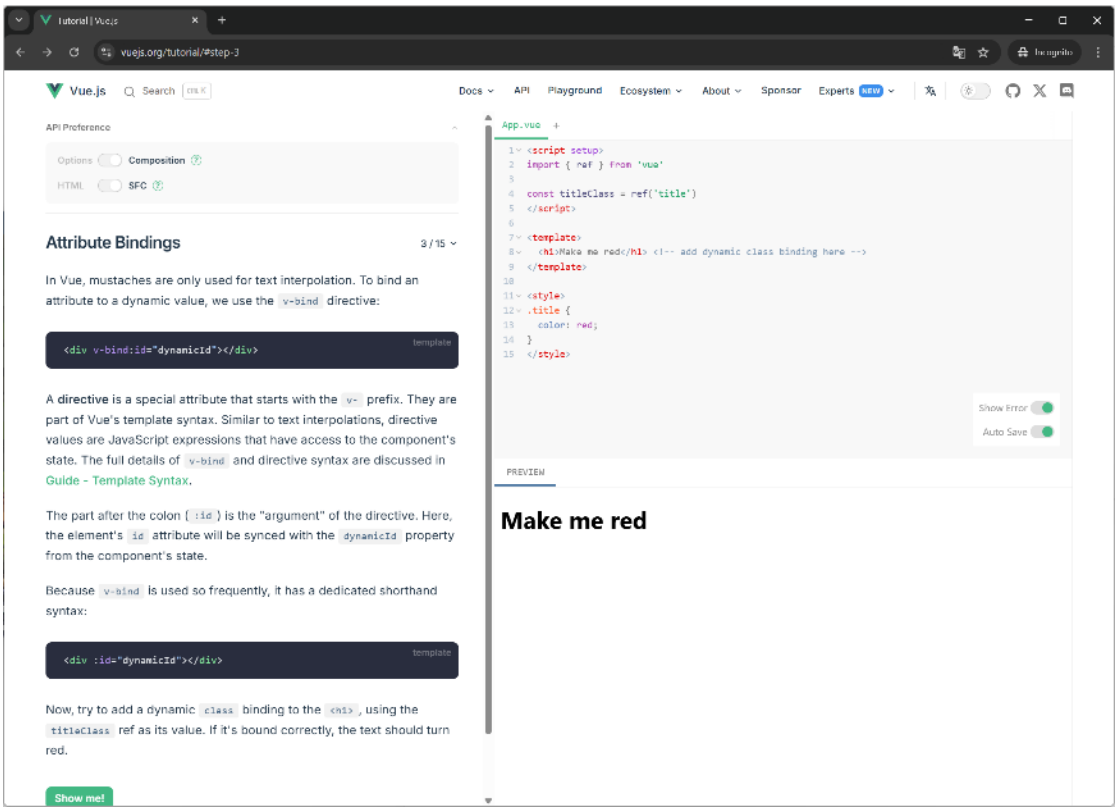
Vuen dokumentaatio on löydettävissä Vuen virallisten verkkosivujen (Vue 2025a) ylänavigaatiosta otsikolla Docs. Dokumentaatiosivu on jaettu kolmeen osaan (kuva 3). Vasemmasta reunasta löytyy navigaatio, jossa aiheita on jaettu pääotsikoihin, joiden alle on listattu aihealueen termejä. Sivun oikeasta reunasta löytyy kyseisen aihealueen sisältö, josta saa yleiskuvan sivusta ja voi navigoida sisällön eri kohtiin tarvittaessa. Keskiosasta löytyy asiasisältö, jossa selitetään aihetta tekstillä ja annetaan syntaksiesimerkkejä. Syntaksiesimerkit ovat kopioitavissa yhdellä klikkauksella.



Kuva 3. Vuen dokumentaatiosivu

Lisäksi usein tarjolla on linkki selaimeen rakennetulle leikkikentälle, jossa koodia voi testata haluamallaan tavalla. Leikkikentällä on mahdollista nähdä erillisenä myös JavaScript- ja CSS-tulosteet. Dokumentaation osana on myös tu-

toriaali, jossa sovelluskehyksen käyttöä voi harjoitella ohjeistuksen avulla. Tutoriaalisivu (kuva 4) on jaettu opetusosaan, koodieditoriin ja tulosteeseen. Opetusosassa selitetään läpikäytävä asia ja annetaan syntaksimalli. Koodieditoriin voi itse tehdä muutoksia, jotka päivittyvät reaaliajassa tulostusosioon. Ratkaisun saamiselle on myös painonappi, jos aihealue tuottaa haasteita.



Kuva 4. Vuen tutoriaalisivu

Dokumentaation osana on myös esimerkkejä, joista saa hyvää referenssiä omiin komponentteihin. Esimerkeistä näkee helposti lähdekoodin ja tulostuksen selaimessa, jolloin pieniä muutoksia voi jo testata ennen omaan koodiin liittämistä. Lisäksi tarjolla on sanasto teknisille termeille ja virhereferenssi kehitystyön helpottamiseksi. Dokumentaatiossa on käytettävissä hakutoiminto.

3.2.3 Svelte

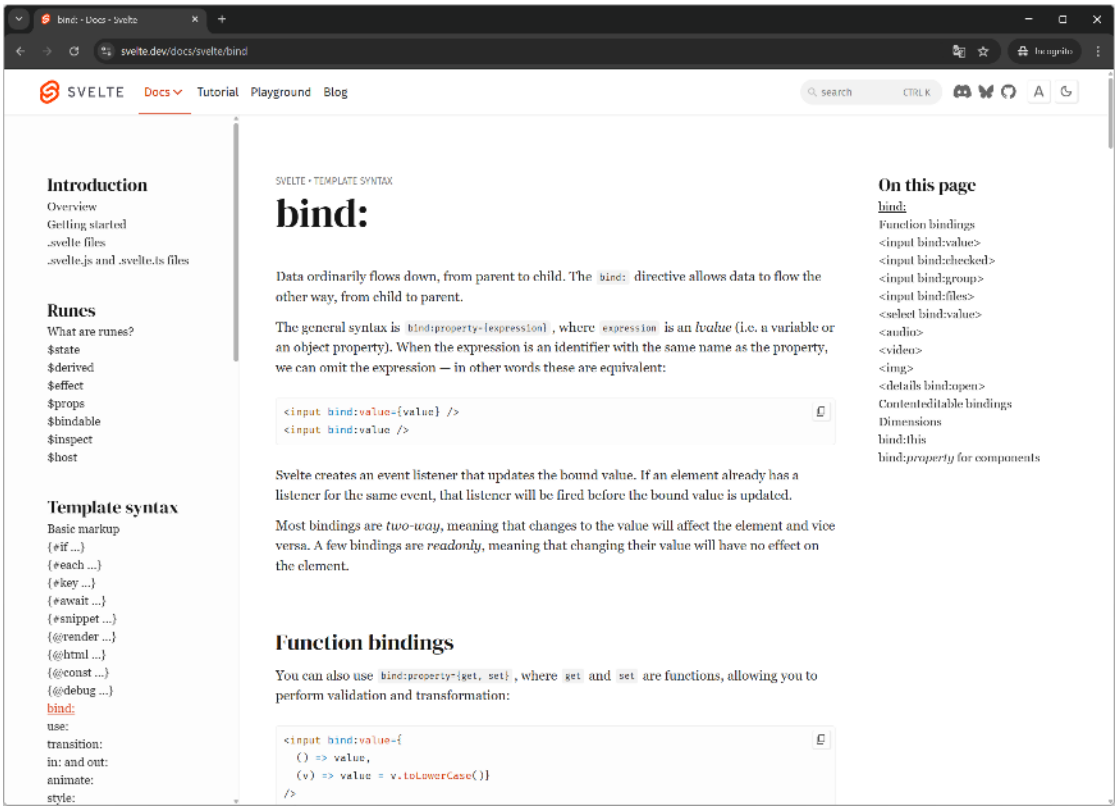
Svelte on Rich Harrisin vuonna 2016 luoma sovelluskehys, joka suunniteltiin todistamaan, ettei komponentteihin tarvitse sisällyttää paljoa koodia ja koodi voisi muistuttaa enemmän perinteistä JavaScriptia. Svelten periaatteeksi voi kuvitella, että pyörää ei tarvitse keksiä uudelleen, vaan käytetään olemassa olevia toimivaksi todettuja asioita hyväksi. Sovelluskehyksestä toimivan tekee

se, että se kääntää koodin uudelleenkäytettäviksi JavaScript-moduuleiksi ilman suorituksenaikaista sovelluskehyksen käyttöä. Tämä mahdollistaa nopean sovelluksen ilman tarpeetonta abstraktiota. (Libby 2022, 1–2.)

Keksittyyn pyörään luottamisen myötä Svelten oppiminen vaatii kehittäjältä todennäköisesti vähemmän uusien konseptien ja työkalujen opettelua, mikä tekee siitä helpon opittavan. Svelte pysyttelee perinteisten HTML-, CSS- ja JavaScript-teknologioiden lähettyvillä lisäten näihin vain muutamia laajennuksia. Suhteellisen nuoren kehyksen heikkouksia ovat rajoitettu määrä lisätyökaluja ja liitännäisiä, selvien käyttötottumusten puuttuminen ja yleinen tuen vähyys. Vakiintuneemmat ja suurten yritysten ylläpitämät sovelluskehykset pystyvät vastaavaan näihin tarpeisiin paremmin. Sveltellä on kuitenkin hyviä käyttötapauksia esimerkiksi pienitehoisten laitteiden sovelluksille, sillä kevyt toteutus ei vaadi niin paljoa prosessointitehoa tai niin hyvää nettiyhteyttä. (Mozilla 2025.)

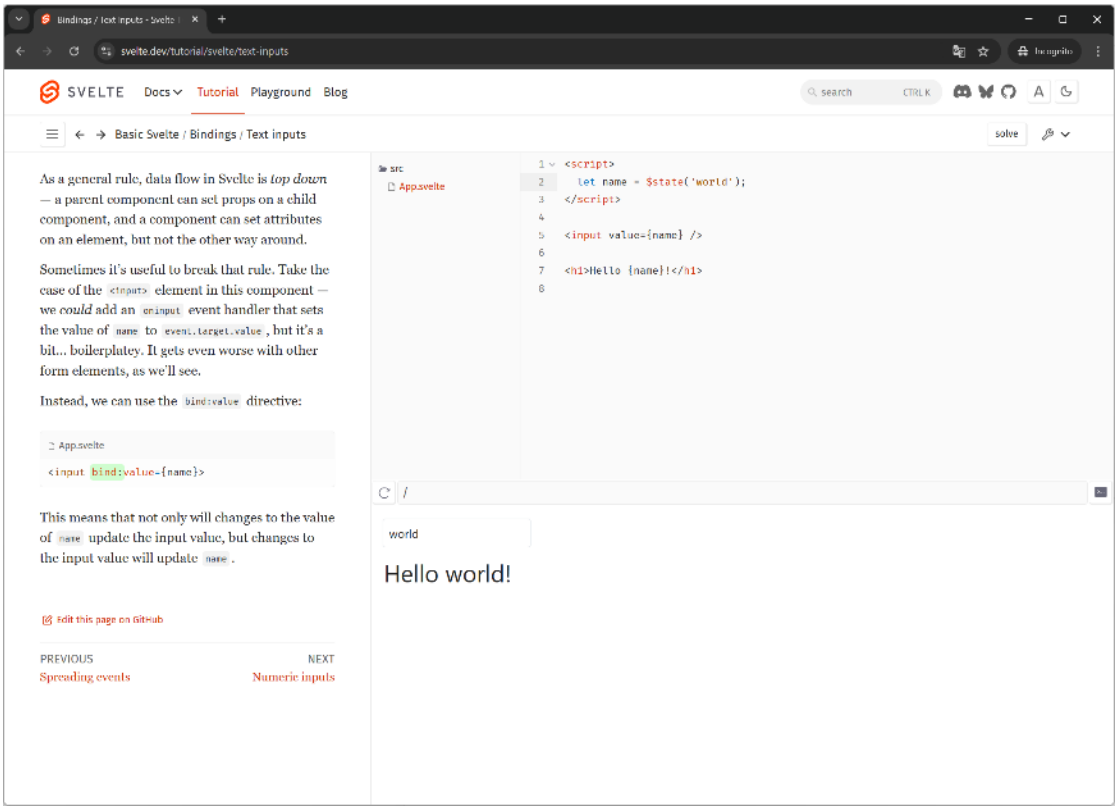
Svelten dokumentaatio

Svelten dokumentaatio on löydettävissä Svelten virallisten verkkosivujen (Svelte 2025) ylänavigaatiosta otsikolla Docs. Dokumentaatio on jaettu kolmeen osaan: Svelte, SvelteKit ja CLI. Kukin osa on rakenteeltaan samanlainen. Kuvassa 5 nähdään esimerkki dokumentaation rakenteesta. Vasemmasta reunasta löytyy rullattava navigaatio dokumentaation sisällöstä pääotsikoihin jaettuna. Pääotsikoiden alta löytyy aihealueeseen liittyvät termit, joita painamalla päästään lukemaan tarkemmin aiheesta. Aiheen avautuessa ilmestyy sivun oikeaan reunaan kyseisen aiheen sisällysluettelo, josta saa kuvan sivun sisällöstä ja pystyy suoraan navigoimaan tarkempaan aiheeseen tarvittaessa. Sivun keskelle on sijoitettu pääsisältö. Pääsisällössä avataan aihealuetta tekstin ja syntaksiesimerkkien avulla. Syntaksit ovat kopioitavissa yhdellä klikkauksella kopiointi-ikonista.



Kuva 5. Svelten dokumentaatiosivu

Dokumentaatio suosittelee sovelluskehykseen tutustumisen interaktiivisen tutoriaalin avulla. Tutoriaalisivun (kuva 6) vasemmasta yläkulmasta löytyy hampurilaisikoni, josta pystytään navigoimaan tutoriaalin sisällössä vapaasti. Sivun vasen puoli koostuu aihetta selittävästä tekstistä ja ohjeistuksesta. Sivun oikea puoli on jaettu kahteen osaan: yläosa sisältää koodieditorin ja alaosa ohjelman tulostuksen. Koodieditorista pystytään muuttamaan lähdekoodia ja tulostus näkyy reaaliaikaisena. Sivun oikeasta yläreunasta löytyy myös painonappi, jolla pystyy ratkaisemaan ongelman, mikäli siihen ei itse kykene ohjeistuksen avulla.



Kuva 6. Svelten interaktiivinen tutoriaali

Dokumentaatio kannustaa myös käyttämään selaimeen rakennettua leikkikenttää, missä sovelluskehystä voi huoletta testata. Leikkikenttä on jaettu koodieditoriin ja reaaliaikaiseen tulostukseen. Leikkikentällä on mahdollista nähdä myös eriteltynä JavaScript- ja CSS-tulostukset. Leikkikentällä on myös laaja valikoima valmiita komponentteja, joita voi käyttää referenssinä. Sivujen yläreunassa on käytettävissä hakutoiminto, jolla dokumentaatiossa pystyy navigoimaan.

3.2.4 AngularJS

AngularJS on alun perin Misko Heveryn ja Adam Abronsin vuonna 2009 kehittämä ja Googlen ylläpitämä avoimen lähdekoodin JavaScript-sovelluskehys. Sovelluskehyksen pääkäyttönä on yksisivuiset sovellukset, joissa sisällön päivitys onnistuu saumattomasti samalla sivulla. AngularJS hyödyntää Model-View-Controlleria, eli MVC-arkkitehtuuria, jonka vuoksi sovelluskehys tukee hyvin ”huolenaiheiden erottamisen”-suunnitteluperiaatetta. Arkkitehtuuri parantaa koodin ylläpidettävyyttä ja skaalautuvuutta, mistä on hyötyä eritoten

monimutkaisissa sovelluksissa. Googlen ylläpito on taannut sovelluskehykselle myös laajan kehittäjäkunnan, mikä on tarjonnut kattavan yhteisötuen. (GeeksforGeeks 2024c.)

MVC-arkkitehtuurin lisäksi AngularJS:llä on neljä muuta ydinkonseptia: kaksisuuntainen tiedon sidonta, direktiivit, palvelut ja riippuvuusinjektointi. Kaksisuuntainen tiedon sidonta päivittää automaattisesti tiedon muutokset tietomallin ja käyttöliittymän välillä. Direktiivit ovat DOM:iin asetettuja merkkejä, jotka ohjaavat sovelluskehystä liittämään tietyt toiminnot DOM-elementteihin tai muuttamaan DOM:in rakennetta. Palvelut ovat sovellukseen sisäänrakennettuja yleisiä toimintoja, kuten HTTP-pyynnöt ja reititys. Riippuvuusinjektointi on suunnittelumalli, joka sallii komponenttien yhdistämisen, mikä tehostaa modulaarisuutta ja yksinkertaistaa testausta. (GeeksforGeeks 2024c.)

Vaikka AngularJS keräsi kiitettävän suosion ja on edelleen laajasti käytössä, toi Google esiin suunnitelmansa tekniikan päivien päättämisestä tammikuussa 2018. Lopulta sovelluskehyksen pitkäaikainen tuki lakkautettiin vuoden 2022 alussa. Syynä tuen lakkautukselle oli AngularJS:n perillinen Angular, johon AngularJS:n kehittäjät suosittelivat kehittäjiä siirtymään. (Thompson 2022.)

3.2.5 Angular

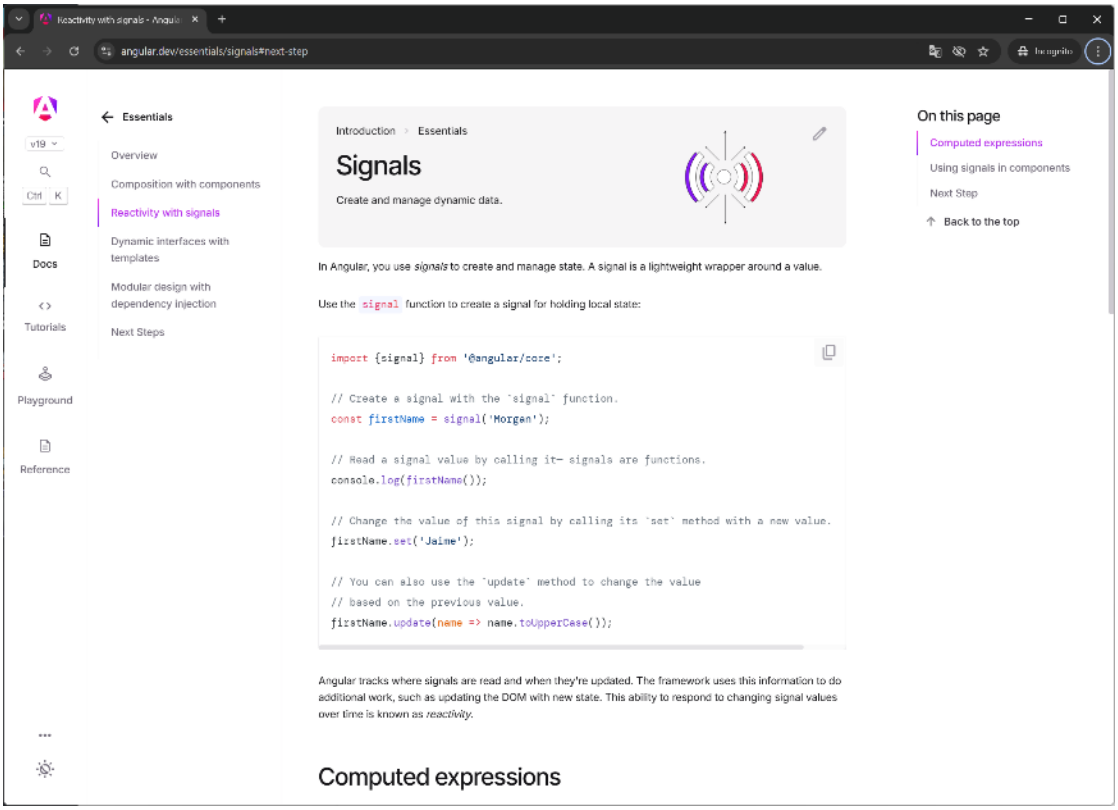
Angular on TypeScript-pohjainen kehitysalusta, johon sisältyy komponenttipohjainen sovelluskehys, kokoelma laajan tarvekirjon kattavia kirjastoja ja sarja kehittäjätyökaluja koodin kehitykseen, rakentamiseen, testaukseen ja päivittämiseen (Angular 2023). TypeScript on JavaScriptin ylijoukko, joka lisää JavaScriptiin staattisen tyypityksen. JavaScript ei normaalisti sisällä tietoa parametrien ja muuttujien tyypeistä, jolloin kehittäjien täytyy turvautua dokumentaatioon tai esittää valistuneita arvauksia tietotyypeistä. Tyypityksen avulla pystytään nostamaan virhetiloja ennen ohjelmakoodin suorittamista. (W3Schools 2025b.) Kehitysvaiheessa nostettavat paikannetut virheet ohjelmakoodissa ovat huomattavasti helpommin korjattavissa, mikä vähentää resurssien tarvetta ja jouduttaa kehitystyötä.

Angular on täysin uudelleenkirjoitettu versio edeltäjästään AngularJS:tä, joka alkoi kohdata haasteita kehittyvän webkehityksen suorituskyvyn ja joustavuuden tarpeissa. Angular keskittyi modulaarisuuteen ja suorituskykyyn. Sittemmin Angularin kehitystä on jatkettu säännöllisin päivityksin ja parannuksin vastaamaan modernin webkehityksen tarpeita. (GeeksforGeeks 2024d.)

Sovelluskehyksenä Angular pyrkii vastaamaan moniin raa'an JavaScriptin webkehityksessä kohtaamiin ongelmiin. Kustomoidut komponentit mahdollistavat toiminnallisuuksien ja renderöintilogiikan uudelleenkäytön. Tiedon sidonta puolestaan huolehtii vaivattomasta tiedon siirrosta koodista näkymään ja käyttäjän tapahtumiin reagoinnin. Riippuvuusinjektoinnin avulla pystytään luomaan modulaarisia palveluita ja injektoida ne tarvittaviin paikkoihin helpottaen palveluiden testausta ja uudelleenkäyttöä. Angular on rakennettu testattavuus mielessä pitäen ja kaikki sovelluksen osat ovat testattavissa. Sovelluskehys on kokonaisvaltainen ja sisältää sisäänrakennettuja toimintoja esimerkiksi palvelinkommunikaatioon ja reititykseen. Myös yhteensopivuus on huomioitu niin selaimien kuin käyttöjärjestelmienkin suhteen. (GeeksforGeeks 2024d.)

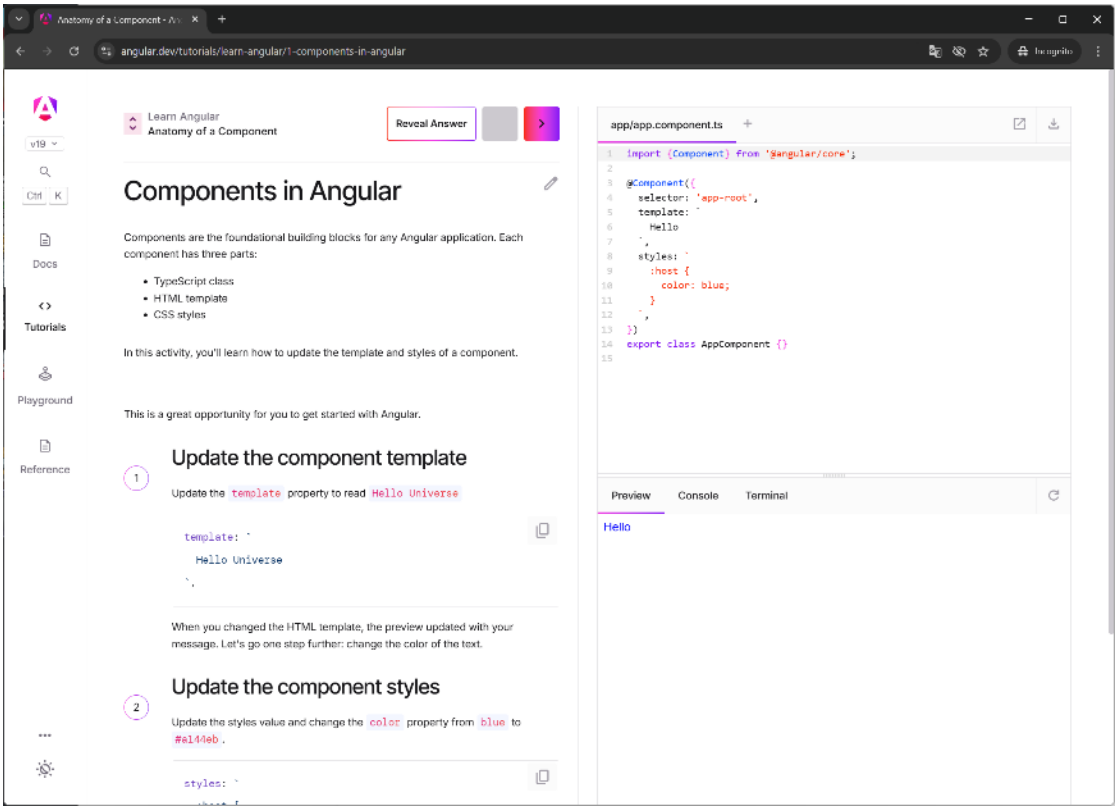
Angularin dokumentaatio

Angularin dokumentaatio on löydettävissä Angularin virallisten verkkosivujen (Angular 2025) sivunavigaatiosta otsikolla Docs. Dokumenttisivu (kuva 7) on jaettu kolmeen osaan. Vasemmassa reunassa on pääotsikoita, joiden alla on sovelluskehykseen liittyviä aihealueita. Aihealueen valitsemalla avautuu navigaatioon lisää vaihtoehtoja aihealueeseen liittyen. Sivun oikeassa reunassa on tämänhetkisen sivun otsikot, josta saadaan yleiskuva sivun sisällöstä ja voidaan navigoida tarkemmin tarvittaessa. Keskiosasta löytyy asiasisältö, joka koostuu selittävästä tekstistä ja koodiesimerkeistä. Koodiesimerkit ovat kopioitavissa yhdellä klikkauksella.



Kuva 7. Angularin dokumentaatiosivu

Dokumentaation tueksi on myös tutoriaali, jolla sovelluskehyksen käyttöä voi harjoitella selaimessa. Tutoriaalisivu (kuva 8) on jaettu osiin. Sivun vasen puoli sisältää aihealuetta selittävää tekstiä ja ohjeistusta, kuten syntaksiesimerkkejä. Oikealla puolella sivua on koodieditori sekä alalaidassa reaaliaikainen tulostusalue. Tutoriaalissa pystyy navigoimaan vapaasti painamalla vasemmassa yläreunassa olevaa sen hetkisen aiheen otsikkoa, jota kautta pystyy hyppäämään muihin otsikoihin.



Kuva 8. Angularin tutoriaali

Selaimeen on rakennettu myös leikkikenttä, jossa sovelluskehystä voi testata vapaasti. Leikkikentältä löytyy muutama valmis malli, joista voi hakea referenssiä kehitystyöhön. Dokumentaatiossa on käytettävissä hakutoiminto.

4 SOVELLUUSKEHYSDOKUMENTAATIOIDEN VERTAILU

Tässä luvussa suoritetaan selainpuolen JavaScript web-sovelluskehysten dokumentaatiopohjainen vertailu. Suosituimmista sovelluskehyksistä on kerätty riittävä yleistason ymmärrys, jotta vertailua pystytään suorittamaan. Viidestä sovelluskehyksestä tarkempaa tarkastelua varten valitaan kolme. Näiden kolmen sovelluskehyksen dokumentaatioiden ominaisuuksia vertaillaan pohjautuen Divion julkaisemaan sovellusdokumentaatiojärjestelmään, jonka mukaan hyvä sovellusdokumentaatio sisältää neljä osa-aluetta: tutoriaalit, how-to-oppaat, teknisen referenssin ja syventävän osion.

4.1 Tarkasteltavien teknologioiden rajaus

Stack Overflown kyselyyn perustuen käytetyimmiksi selainpuolen JavaScript teknologioiksi ammattia harjoittavien keskuudessa nousivat React, Angular,

Vue, AngularJS ja Svelte. React ja Angular ovat saavuttaneet laajimman käyttöasteen, mikä ei sinänsä ole ihme, sillä teknologioiden taustalla toimivat isot yhtiöt Meta ja Google. Vue ja Svelte taas ovat yksittäisten kehittäjien aikaansaannoksia, mutta toki niidenkin ylläpidosta vastaa dedikoitu tiimi.

AngularJS kantaa lippua edelleen yllättävän korkealla, vaikka pitkäaikainen tuki lakkautettiin vuoden 2022 alussa. Sovelluskehyksen suosion suunta on kuitenkin arvatenkin laskussa tuen lakkauttamisen myötä. Kuolevaan teknologiaan ei ole syytä paneutua syvemmin, eritoten kun teknologialle on selvä seuraaja Angular. Näin ollen rajataan AngularJS ulos niiden kolmen sovelluskehyksen joukosta, joihon syvennytään tarkemmin.

Metan ylläpitämä React on mielenkiintoisessa asemassa web-teknologioiden joukossa. Reactin käyttäjämäärä on yli kaksinkertainen toiseksi käytetyimpään selainpuolen web-teknologiaan verrattuna. Teknologia on siis hyvin juurtunut web-kehityksen maailmaan, mutta vaikka React mielletään usein sovelluskehykseksi, on kyseessä todellisuudessa kirjasto. Kirjastona React ei tarjoa yhtä jykevää rakennetta sovelluksille kuin sovelluskehykset. Reactin päälle on rakennettu sovelluskehyksiä, joista suosituimpina ovat Reactin suosittelemat NextJS ja Remix. NextJS on kerännyt laajemman suosion ja Remix pienemmän. Molemmat ovat full-stack-kehitykseen tarkoitettuja sovelluskehyksiä, vaikka toki pelkän selainpuolen toteutus on mahdollista. Työssä keskitytään eritoten selainpuolen JavaScript-sovelluskehyksiin, joten Reactia tai sen päälle rakennettuja sovelluskehyksiä ei tarkastella pidemmälle.

Jäljelle jäävät sovelluskehykset Angular, Vue ja Svelte. Kaikki kolme sovelluskehystä sopivat tarkemman tarkastelun kohteiksi, sillä ne on tarkoitettu selainpuolen web-kehitykseen ja ovat aktiivisesti ylläpidettyjä. Sovelluskehyksillä on myös julkisesti saatavissa oleva dokumentaatio, jota voidaan tarkastella.

4.2 Sovellusdokumentaation vertailu

Luvussa 3 saatiin jo yleiskuva sovelluskehysdokumentaatioiden rakenteesta. Ulkoisesti dokumentaatiot muistuttavat paljolti toisiaan, joten on aika paneutua dokumentaatioiden sisältöön. Vertailussa dokumentaatioita peilataan Divion

sovellusdokumentaatiojärjestelmään, joka määrittää neljä osa-aluetta, joilla on jokaisella oma tehtävänsä hyvässä dokumentaatiossa (taulukko 1).

Taulukko 1. Divion sovellusdokumentaatiojärjestelmän osa-alueet

	Tutoriaalit	How-to-oppaat	Tekninen referenssi	Syventävä osio
Keskeisyys	Oppiminen	Tavoite	Tieto	Ymmärrys
Päämäärä	Aloittelijan alkuun saattaminen	Spesifin ongelman ratkaisu	Teknologian kuvailu	Selittäminen
Muoto	Oppitunti	Sarja vaiheita	Tekninen selitys	Diskursiivinen selitys

Osa-alueiden ymmärtämiseksi käytetään usein kokkauksen analogiaa. Tutoriaalissa opetetaan aloittelevaa kokkia, how-to-opas on resepti kokkikirjassa, tekninen referenssi on artikkeli raaka-aineesta tietosanakirjassa ja syventävä osio rikastuttaa tietämystä kertomalla annoksen historiasta. Sovelluskehyksen kohdalla analogia voisi olla esimerkiksi, että tutoriaali opettaa kehittäjälle tekniikan alkeet, how-to-opas näyttää kuinka rakentaa to-do-sovellus, tekninen referenssi selittää teknologialle tärkeän metodin teknisen toteutuksen ja syventävä osio kertoo laajemmin, miten to-do-sovelluksen toteutuksessa käytetyt elementit ovat tärkeä osa web-kehitystä.

4.2.1 Sovelluskehysdokumentaatioiden tutoriaalit

Tutoriaalit on suunnattu sovelluskehyksen kanssa aloittelijaksi luokitetulle kehittäjälle, mutta se ei tarkoita, että sovelluskehyksen käytön aloitus ei vaatisi perustietoa web-kehityksestä. Sovelluskehystä käyttävällä kehittäjällä tulee olla hallussa web-kehityksen perustekniikat JavaScript, HTML ja CSS. Sovelluskehykset toimivat laajentavana jatkeena perustaville tekniikoille.

Tutoriaalin tarkoitus on näyttää, mitä teknologialla pystytään toteuttamaan. Tärkeintä ei ole päästä lopputulokseen käyttäen parhaita menetelmiä, vaan niin, että aloittelija pystyy seuraamaan ohjeita ja ymmärtämään mitä tapahtuu. Lopputuloksen tulee olla onnistuminen, ei kasa virheitä, jotta se motivoi kehittäjää jatkamaan teknologian parissa. Kädestä pitäen ohjaaminen on jopa suotavaa, jotta mahdollisilta yksinkertaisilta virheiltä vältytään. Tutoriaalin suoritettuaan kehittäjä pystyy ymmärtämään muuta dokumentaatiota ja teknologiaa

itsenäisesti. Tärkeää tutoriaalille on tekeminen, ymmärtäminen ja onnistuminen.

Angular

Tarkastellaan ensimmäiseksi Angularin tutoriaalia. Tutoriaali koostuu yhteensä 26 erillisestä oppitunnista, joista jokainen edustaa jotain teknologian omaa konseptia. Oppitunnin alussa esitellään lyhyesti ongelma tai konsepti ja kerrotaan, mitä kehittäjä oppitunnilla tulee oppimaan. Esittely sisältää useimmiten koodiesimerkin oppitunnin konseptista. Esittelyn jälkeen tulee numeroidut askeleet lopputulokseen pääsemiseksi. Askeleiden avulla annettuun lähdekoodiin tehdään muutoksia. Muutokset ovat käytännössä samoja, mitä esittelyn koodiesimerkissä on ollut, joten suurempia ponnisteluja askeleiden suorittaminen ei aiheuta. Oppitunti päättyy aina seuraavan oppitunnin sisällön kertomiseen ja kannustavaan viestiin.

Yhden oppitunnin tekemiseen ja ymmärtämiseen kului keskimäärin viitisen minuuttia. Osa konsepteista oli nopeampia, osa hitaampia. Oppituntien järjestys oli hyvässä jatkumossa ja pääosin perättäiset oppitunnit käsittelevät samaa konseptia. Konseptien selitys oli suppeaa, mutta riittävää, eli missään vaiheessa ei tuntunut tulevan liikaa informaatiota yhteen kasaan. Tutoriaali oli hyvin ymmärrettävä ja tekemiseen annettiin selvät askeleet, mikä johti onnistumisiin.

Selaintutoriaalin käyttökokemuksessa tein muutaman parannusta kaipaavan huomion. Oppituntia alas rullatessa lähdekoodi pääsee karkaamaan ylöspäin, jolloin osa koodista päätyy piiloon ja tiedostojen välillä ei pysty navigoimaan ilman ylös rullaamista, jolloin taas suoritettava askel voi mennä piiloon. Vastauksen näyttävä nappi ei päivitä ohjelmaa, vaan koodiin pitää tehdä manuaalisesti muutos päivitystä varten. Edellisiin oppitunteihin nopea referointi hankaloituu tästä syystä. Lisäksi havaitsin vastauskoodin menneen kahden eri riippuvuusinjektio-oppitunnin välillä ristiin, mikä aiheutti päänvaivaa ja hämmennystä.

Kokonaisuudessa Angular onnistui hyvin Divion määrittelemien tutoriaalivaateiden suhteen. Valitettavasti saavutettavuuden ja luotettavuuden kanssa ilmeni pieniä ongelmia. Vaikka kyse on suhteellisen pienistä havainnoista, herättää se aavistuksen luottamuspulaa dokumentaation ylläpidon suhteen.

Vue

Vuen tutoriaali on päällepäin suppeampi sisältäen vain 15 oppituntia. Tutoriaalin esittely tekee hyvin selväksi, ettei se ole kaiken kattava eikä kaikkea tarvitse ymmärtää eteenpäin mennäkseen. Esittelyssä suositellaan myös paneutumaan tutoriaalin suorittamisen jälkeen oppaaseen, joka sisältää yksityiskohtaisemmat selitykset. Lähes kaikilla oppitunneilla viitataan myös oppaaseen yksityiskohtaista selitystä varten.

Vuella on hieman erilainen tyyli tutoriaalin toteutuksessa. Oppitunnit ovat pääosin konsepteja esittelevää asiasisältöä ja lopussa on maininta, mitä annetun lähdekoodin kanssa olisi hyvä koittaa. Kehittäjälle ei anneta selviä askeleita tehtävän suorittamiseen ja kaikkien tehtävien ratkaisu ei edes vaadi mitään itse teknologiaan liittyvää, vaan yleistä webkehityksen osaamista.

Oppitunneilla ei tunnu olevan selvää punaista lankaa, ja tutoriaalin suorittaminen tuntuu hieman pitkin poikin poukkoilulta. Asiasisältö on itsessään pääosin hyvää ja ymmärrettävää ja Vuen konsepteja tulee kyllä jollain tasolla tutuksi, mutta tutoriaalin suorittaminen vaikuttaa vaativan liikaa vaivaa teknologian aloittelijalta. Tekeminen tuntui haastavalta ja onnistumiset olivat vähäisiä, mikä hieman haittasi myös ymmärtämistä. Teknisesti tutoriaali oli erinomaisesti toteutettu eikä käyttökokemuksessa ollut mitään valittamista.

Svelte

Svelte on näistä kolmesta sovelluskehyksestä tutoriaalin suhteen selvästi laajin. Basic Svelte koostuu 46 oppitunnista, jotka on jaettu yhdeksään pääaiheeseen ja Advanced Svelte 30 oppitunnista, niin ikään yhdeksään pääaiheeseen jaettuna. Oppitunnit ovat muutaman minuutin kestoisia, eli konseptit on pilkottu hyvin pieniin osiin.

Oppituntien sisällössä lähdekoodiin tehtävät muutokset on annettu suoraan selittävään tekstiin upotettuna, eikä vaadi tutoriaalin tekijältä ajatustyötä. Koodin kopiointi on lähtökohtaisesti estetty ja kehittäjää kannustetaan kirjottamaan koodi itse oppimisen tehostamiseksi, mutta toki kopioinnineston voi purkaa. Tutoriaalit olivat helposti seurattavia ja ymmärrettäviä, johtuen varmaan siitä, että omaa ajatustyötä ei koodin tuottamiseksi joutunut tekemään.

Tutoriaalin osien pääaihealueisiin jako auttoi seuraamaan edistymistä, ja myös tauottamaan opiskelua. Selvä tieto siitä, mihin aihealueeseen yksittäinen oppitunti liittyy, auttoi ymmärtämään käsiteltävää aihetta. Ymmärrettävyys oli korkealla tasolla ja tekemisestä tehtiin helppoa. Onnistumisen tunteet eivät tosin olleet yhtä vahvoja, kun oman panoksen lopputuotteeseen tiedosti olevan pieni. Tutoriaalin käyttökokemus oli erinomainen.

4.2.2 Sovelluskehysdokumentaatioiden how-to-oppaat

Tavoitekeskeiset how-to-oppaat ohjaavat kehittäjän laajemman ongelman selvittämiseen käytännössä. Kun tutoriaalissa saatetaan käydä läpi lomakkeen yksittäisiä elementtejä, niin how-to-oppaassa kerrotaan, kuinka rakennetaan täysin toimiva lomake. Opas olettaa kehittäjän ymmärtävän teknologian perusteet. Kaikkea ei selitetä yksityiskohtaisesti, vaan tarvittaessa lisätietoa ohjataan hakemaan esimerkiksi linkin takaa. Oppaat ovat myös joustavia, jolloin kehittäjä pystyy soveltamaan niitä erilaisiin käytännön tarkoituksiin.

Tarkastelluista sovelluskehyksistä ainoastaan Angularilla on selvästi how-to-oppaita. Vuen dokumentaatiossa ainoa how-to-oppaaksi luokiteltavissa oleva osio on teknologian käyttöönotto. Svelten dokumentaatiosta puuttuu täysin how-to-oppaat.

Angularin oppaat löytyvät dokumentaation In-depth Guides -osiosta. Oppaita on kattava kirjo, lähes sata kappaletta, jotka vastaavat erinäisiin web-kehityksen käytännön haasteisiin. Suurin osa Angularin oppaista ei tosin vastaa Divion määritelmää hyvästä how-to-oppaasta. Oppaiden sisältöön on upotettu paljon sellaista syventävää tietoa, mikä ei ole välttämätöntä asian ratkaisemi-

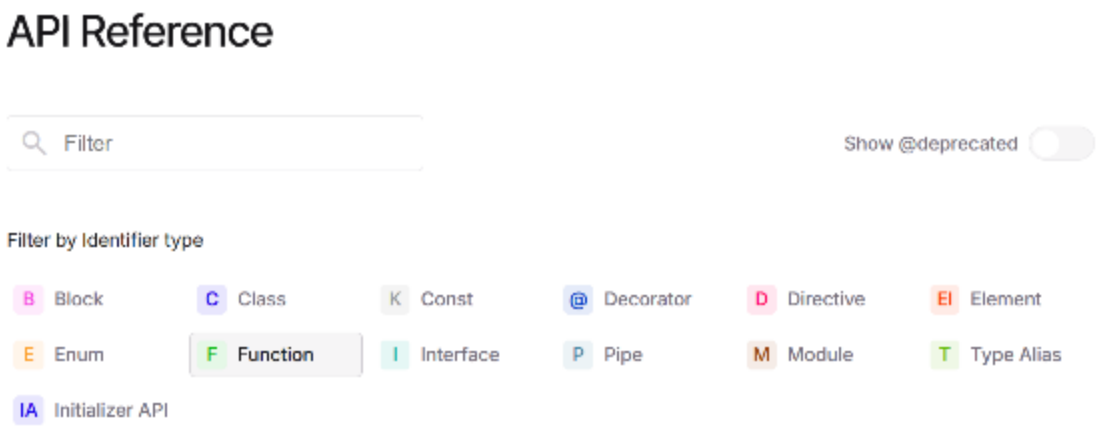
seen. Oppaiden käytännön käytettävyys on tärkeämpää kuin konseptin kokonaisvaltainen ymmärtäminen, joten liialliset selitykset olisi hyvä ohjata lukemaan muualta. Angularin oppaat saavuttavat tavoitteen spesifin ongelman ratkaisusta, joten selvästi ne ovat onnistuneita, vaikka eivät täydellisiä.

4.2.3 Sovelluskehysdokumentaatioiden tekninen referenssi

Tietokeskeinen tekninen referenssi on osa-alue, joka on usein tuotettu hyvin. Teknologian kehittäjät tuntevat sovelluksen jo hyvin, joten heille voi olla vaikeaa edes ajatella, että sovelluksen käyttö vaatisi muuta kuin teknisen referenssin. Teknisen referenssin tulisi olla yksinkertaista ja kuvailla suoraan teknologiaa, eikä niinkään kertoa miten teknologiaa käytetään. Esimerkkejä teknologian käytöstä toki saa olla, mutta yksinkertaisia konsepteja teknologian käytöstä ei selitetä. Tämä osio löytyykin hyvin kaikista kolmesta tarkastelta-vasta sovelluskehyksestä.

Angular

Angularin tekninen referenssi on erittäin laaja. Referenssissä on käytössä hyvät suodattimet (kuva 9), joiden avulla pystyy rajaamaan aihealuetta, mikäli ei aivan tarkalleen muista mitä on etsimässä. Tarkkaan suodattamiseen on tietenkin käytössä hakutoiminto.



Kuva 9. Angularin teknisen referenssin suodattimet

Referenssin sisältö on erittäin asiapitoista, eli se ei sisällä ylimääräisiä selityksiä. Tarvittaessa referenssi sisältää esimerkin. Referenssisivut ovat rakenteellisesti ja tyylillisesti yhtäläisiä, joten referenssi on helppolukuista. Sisältö on

täysin tietokeskeistä, yksinkertaista ja asiassa pysyvää. API-referenssin lisäksi saatavilla on myös muun muassa tietosanakirja virheille ja CLI-referenssi. Division määritelmiin peilaten ei Angularin osalta ole mitään huomautettavaa tässä osiossa.

Vue

Vuen dokumentaatiossa on saatavilla referenssi virheille, joka sisältää vain virhenumeron ja virhetekstin. Lisäksi on tekninen sanasto yleisille sovelluskehykseen liittyville termeille. Pääosin tekninen referenssi muodostuu kuitenkin API-referenssistä. Vue viittasi tutoriaaleissa paljon omin sanoinensa oppaaseen, mutta kuten how-to-osioita tarkasteltaessa huomattiin, ei Vuen dokumentaatio oikeastaan how-to-oppaita sisältänyt. Vuen oppaan sisältö luokittuu enemmän tekniseksi referenssiksi Division järjestelmään peilaten.

Vuen oppaiden puolella oleva tekninen referenssi sisältää yksinkertaisten asioiden selittämistä. API-referenssin puolella dokumentaation rakenne on selvästi enemmän teknisen referenssin omainen: asia, ominaisuudet ja esimerkki. API-referenssi on helppolukuista ja yhtäläistä, joten se on ymmärrettävää. Hakutoiminto toimii hyvin, mutta suodattimia sisältöön ei ole saatavilla. Dokumentaatio on hieman suppeampi ja oletettavasti kaiken kattava, joten ei suodattimet aivan välttämättömiä olekaan. Sisältö on jaoteltu hyvin pää- ja alaotsikoita käyttäen.

Svelte

Svelten dokumentaation ainoa eroteltu osio oli tutoriaali. Kun how-to-oppaita ei ollut, jää jäljelle tekninen referenssi. Svelte on nostanut dokumentaatioon omaksi pääotsikokseen referenssin, mutta Division järjestelmän mukaisesti luokittelen koko muun dokumentaation tekniseksi referenssiksi.

Svelten itse nostaman referenssin ulkopuolelle jäävät osiot noudattavat suhteellisen hyvin teknisen referenssin määritelmiä. Sisältö on hyvin tietokeskeistä, mutta jonkun verran yksinkertaisten konseptien selittämistä löytyy. Tarkastellaan sitten dokumentaation ”Reference”-otsikon alta löytyvää sisältöä. Osiosta löytyy virhe- ja varoitusreferenssit muun referenssin lisäksi. Sisältö on

yksinkertaista ja yhdenmukaista, joten luettavuus on hyvä. Esimerkkejä on saatavilla paikoittain. Suodattimia ei ole tarjolla, mutta hakutoiminta ja hyvä otsikointi tekee navigoinnista helppoa. Referenssi ei ole erityisen laaja, mutta oletettavasti kaiken kattava.

4.2.4 Sovelluskehysdokumentaatioiden syventävä osio

Dokumentaation ymmärryskeskeisen syventävän osion tarkoitus on käsitellä aiheita yleisemmällä tasolla tai eri perspektiivistä. Tarkoitus on rikastuttaa osaamista tai tietämystä paneutumatta suoranaisesti varsinaiseen aiheeseen. Syventävä osio on kevyttä luettavaa, eikä sisällä asioita, jotka muu dokumentaatio jo kattaa. Se voi olla upotettuna muuhun dokumentaatioon tai irrallisena osana. Svelten dokumentaatiosta ei löytynyt mitään syventävään osioon viittaavaa materiaalia.

Angular

Angularin oppaisiin on upotettu hieman syventävää osiota. Esimerkiksi lomakkeita käsittelevässä oppaassa kerrotaan aluksi, mihin lomakkeita voidaan käyttää, ja reitityksessä on avattu, miksi reititystä tarvitaan. Oppaissa syventävä osio on erittäin suppeaa, mutta tuo kuitenkin hieman laajempaa näkökulmaa aihealueeseen. Syventävää osiota löytyy myös parhaiden käytäntöjen alta. Tietoa löytyy muun muassa web-pohjaisten sovellusten saavutettavuudesta ja tietoturvan puolelta cross-site scriptingista ja syötteen sanitoinnista. Vaikka syventävän osion osuus on vähäistä, on sitä havaittavissa.

Vue

Vuen dokumentaatiossa on listattuna Extra topics, joka sisältää syventävän osion kaltaista materiaalia. Hyvänä esimerkkinä on osio reaktiivisuudesta, jossa ensin kerrotaan mitä reaktiivisuus ohjelmoinnissa on ja sen jälkeen esitetään, kuinka se on Vuella toteutettu. Syventävä tieto ei ole täysin välttämätöntä kehittäjälle, joten sitä ei tarvitse käsitellä muilla dokumentaation osa-alueilla, mutta halutessaan kehittäjällä on mahdollisuus syventää ymmärrystään. Muuhun dokumentaatioon upotettua syventävää materiaalia ei Vuesta löytynyt.

4.3 Sovelluskehysdokumentaatioiden vertailun tulokset

Kolmen valitun selainpuolen Javascript-sovelluskehyksen dokumentaatioihin on nyt paneuduttu Divion dokumentaatiojärjestelmän neljän osa-alueen kautta. Ensimmäisenä paneuduttiin dokumentaatioiden tutoriaaleihin. Angularin tutoriaali todettiin kokonaisvaltaisesti hyväksi, mutta pienet haasteet tutoriaalin suorituksen käyttökokemuksessa häiritsivät hieman. Vuen tutoriaali ei ollut täysin aloittelevalle kehittäjälle suunniteltu ja osoittautui tästä syystä yllättävän työlääksi, vaikka olikin tutoriaaleista selvästi lyhin. Svelten tutoriaali oli selvästi laajin ja aloittelija mielessä pitäen toteutettu, mutta onnistumisen tunnetta ei päässyt kokemaan niin vahvana, kun tekeminen oli niin vahvasti kädestä pitäen näytettyä.

How-to-oppaita löytyi vain Angularin dokumentaatiosta. Oppaita oli siellä kattava kirjo monenlaisiin kehittäjän kohtaamiin haasteisiin. Divion dokumentaatiojärjestelmään peilaten oppaat eivät kuitenkaan ollut laadullisesti erityisen hyviä. Tärkeimpään oppaat kuitenkin vastasivat, eli ne onnistuneesti ratkaisivat spesifin ongelman.

Tekninen referenssi on osa-alueena useimmiten onnistunut, joten odotusarvot sovelluskehysten osalta olivat korkealla. Angular vastasi haasteeseen erinomaisesti tarjoamalla todella laajan referenssin hyvillä suodattimilla. Sisältö oli sitä mitä sen pitääkin, eli täyttä asiaa. Vuen ja Svelten osalta tekninen referenssi on hieman kinkkinen asia. Molemmilta löytyy selvästi hyvälaatuinen varsinainen tekninen referenssi, mutta suuri osa dokumentaatioita luokittuu myös teknisen referenssin alle, vaikka sitä ei olisi siinä mielessä ja siihen tarkoitukseen kirjoitettu.

Syventävä osio oli unohtunut Svelten pelikirjasta kokonaan. Vuen dokumentaatiosta löytyi muutama laadukas syventävä osio. Myös Angular tarjosi muutaman laajemman syventävän osion ja oppaisiin oli upotettu ajoittain yleispätevää informaatiota aihealueeseen liittyen. Selvää kuitenkin oli, että mikään sovelluskehys ei ollut tähän osioon erityisesti panostanut.

Jotta sovelluskehysten dokumentaatiot saadaan laitettua jonkinlaiseen paremmuusjärjestykseen, luodaan pisteytysjärjestelmä, jolla voidaan arvioida dokumentaatioiden onnistumista. Taulukossa 2 on selitettynä, mitä eri pistearvot edustavat Divion dokumentaatiojärjestelmään nähden.

Taulukko 2. Divion sovellusdokumentaatiojärjestelmän osa-alueet

	Pistemäärän selitys
5	Osa-alue vastasi järjestelmää täydellisesti
4	Osa-alueen sisältö vastasi lähes täydellisesti järjestelmää
3	Osa-alueen sisältö sisälsi useamman järjestelmän ominaisuuden
2	Osa-alueen sisältö sisälsi järjestelmän ominaisuuden
1	Osa-alue on tunnistettavissa
0	Osa-aluetta ei ollut

Pisteytysjärjestelmän esittelyn jälkeen suoritetaan sovelluskehysdokumentaatioiden pisteytys. Taulukko 3 sisältää tarkastellussa olleen kolmen sovelluskehyksen dokumentaation saamat pisteet.

Taulukko 3. Sovelluskehysdokumentaatioiden pisteytys

	Angular	Vue	Svelte
Tutoriaalit	4	2	5
How-to-oppaat	3	1	0
Tekninen referenssi	5	3	3
Syventävä osio	2	2	0
Summa	**14**	**8**	**8**

Pisteytyksessä palkintopallin korkeimmalle paikalla astelee selvällä erolla Angular 14 pisteellä. Angularin sovelluskehysdokumentaatio oli tarkastelluista sovelluskehyksistä selvästi kokonaisvaltaisin. Näin ollen dokumentaatioon pohjautuen, selainpuolen sovelluskehitykseen voin suositella sovelluskehykseksi Angularia.

5 PÄÄTÄNTÖ

Työn tavoitteena oli arvioida ja vertailla selainpuolen JavaScript-sovelluskehysten dokumentaatioita. Työ eteni mielestäni järkevästi niin, että ensin raken-

netaan arviointia ja vertailua varten teoriapohja JavaScriptista, dokumentaatiosta ja sovelluskehyksistä. Teorian rakentamisessa hieman haasteita tuotti sovelluskehyksiin liittyvän aineiston kerääminen, sillä tuntui, että pääosin sovelluskehyksiin liittyvä materiaali liittyi tekniseen tekemiseen. Tarkoitukseni ei kuitenkaan ollut esitellä laajasti teknisellä tasolla sovelluskehysten ominaisuuksia tai toteuttaa laajaa sovellusta, joten suurimmasta osasta materiaalista en saanut paljoa irti.

Yllättävän paljon haasteita tuotti myös dokumentaatioon liittyvän aineiston kerääminen. Halusin pitää työn pääpainon dokumentaatioon pohjautuvassa vertailussa, joten tavoitteeni oli kerätä mahdollisimman kattavasti tietoa dokumentaatiosta ja sen vertailusta, mutta lähteet tuntuivat olevan kiven alla. Teoriaosuus oli jo oikeastaan valmis, kun törmäsin Divion dokumentaatiojärjestelmään, ja olin jo kerennyt miettimään työn kääntämistä enemmän ohjelmointiin painottuvaksi. Divio kuitenkin tarjosi selvät raamit, joiden avulla sovelluskehysten dokumentaatioita voisi vertailla, joten tartuin siihen kiinni.

Divion dokumentaatiojärjestelmä on mielestäni erinomainen. Se, soveltuuko se sovelluskehysten dokumentaatioon, onkin toinen kysymys. Uskon, että soveltuu. Divion dokumentaatiojärjestelmä on käytössä ainakin Djangolla, joka on Python-pohjainen web-sovelluskehys. Dokumentaatioihin perehtyessä huomasin, että olin aika yhtä mieltä Divion järjestelmän kanssa dokumentaatioiden laadusta. Se, ovatko kaikki Divion määrittelemät osa-alueet välttämättömiä sovelluskehykselle, on mielenkiintoinen kysymys. Tästä voisi mahdollisesti jonkinlaista jatkotutkimusta tehdä. Jatkotutkimuksena voisi myös olla enemmän sovelluksen toteutukseen keskittyvä dokumentaatioon peilaava työ esimerkiksi Djangolla, jonka dokumentaatio pitäisi olla laadukasta.

Palataan vielä työn tavoitteeseen, eli sovelluskehysten dokumentaatioiden arviointiin ja vertailuun. Mielestäni tämä onnistui hyvin. Vaikka tuloksissa olisin toivonut, että suositut teknologiat keräisivät hyvät pisteet, rehellisesti pisteytettynä ei hirveän korkealle päästy. Angular nappasi selvän paalupaikan, mikä ei sinänsä ollut yllättävää, sillä onhan sen taustalla valtava yhtiö. Yhtiöllä on varmasti vakiintuneita käytänteitä, mitä pienemmän taustajoukon omaavilla sovelluskehyksillä ei ole, mistä syystä dokumentaation laatukin on parempaa. Vaikka se ei ollut Divion mittareilla erinomaista, menestyi Angular silti hyvin.

Dokumentaatio on vain yksi osa sovelluskehyksen valintaa, vaikkakin tärkeä. Itselleni Vue tuntui erittäin sekavalta, liekö syy sitten ollut huonossa tutoriaalissa, enkä välttämättä lähtisi sitä suosittelemaan eteenpäin. Angularista ja Sveltestä itselleni jäi hyvä kuva ja pidän molempia mahdollisena vaihtoehtona tuleviin projekteihin.

Työn tehtyä sain hyvän uuden perspektiivin dokumentaation arviointiin, josta on varmasti hyötyä jatkossa. Toivottavasti sinäkin lukijana sait jotain irti. Mikäli dokumentaation laatu kiinnostaa, suosittelen tutustumaan Divion dokumentaatiojärjestelmään tarkemmin. Järjestelmä ei ole erityisen laaja, mutta tässä työssä sen osa-alueista tuli esiin kuitenkin vain pieni osa.

Aloitin opinnäytetyön marraskuun lopussa 2024 ja lopetan nyt huhtikuun lopussa 2025, eli kyseessä oli noin viiden kuukauden prosessi. Eniten aikaa kului suunnitteluun ja teorian tekemiseen. Tiesin näiden osa-alueiden jo alusta alkaen olevan itselleni hankalimmat, joten asia ei tullut yllätyksenä. Kokonaisvaltaisesti prosessi oli kuitenkin jouheva ja silmiä avaava.

LÄHTEET

Angular. 2023. What is Angular?. WWW-dokumentti. Päivitetty 15.8.2023. Saatavissa: https://v17.angular.io/guide/what-is-angular [viitattu 2.3.2025]

Angular. 2025. Angular. WWW-dokumentti. Saatavissa: https://angular.dev/ [viitattu 6.4.2025]

Divio. 2025. The Documentation System. WWW-dokumentti. Saatavissa: https://docs.divio.com/documentation-system/ [viitattu 1.4.2025]

Ecma International. 2025. EcmaScript 2025 Language Specification. WWW-dokumentti. Päivitetty 23.1.2025. Saatavissa: https://tc39.es/ecma262/ [viitattu 8.2.2025]

Euroopan komission terveys- ja kuluttaja-asioiden pääosasto. EudraLex, The Rules Governing Medicinal Products in the Europe Union, Good Manufacturing Practice. Brussels: European Commission, 2010. SANCO/C8/AM/S1/ares(2010)1064587.

Flexiple. 2025. All You Need to Know About VueJS. WWW-dokumentti. Saatavissa: https://flexiple.com/vue/deep-dive [viitattu 2.3.2025]

GeeksforGeeks. 2024a. History of JavaScript. WWW-dokumentti. Päivitetty 9.6.2024. Saatavissa: https://www.geeksforgeeks.org/history-of-javascript/ [viitattu 19.1.2025]

GeeksforGeeks. 2024b. Why is React considered a library and not a framework?. WWW-dokumentti. Päivitetty 6.2.2024. Saatavissa: https://www.geeksforgeeks.org/why-is-react-considered-a-library-and-not-a-framework/ [viitattu 23.3.2025]

GeeksforGeeks. 2024c. AngularJS Tutorial. WWW-dokumentti. Päivitetty 1.7.2024. Saatavissa: https://www.geeksforgeeks.org/angularjs/#what-is-angularjs [viitattu 2.3.2025]

GeeksforGeeks. 2024d. What is Angular ?. WWW-dokumentti. Päivitetty 12.4.2024. Saatavissa: https://www.geeksforgeeks.org/what-is-angular/ [viitattu 3.2.2025]

Goetz, J. & Marquez, A. F. 2023. Web Framework. *International Journal on Engineering, Science & Technology (IJonEST)* 4. PDF-dokumentti. Saatavissa: https://www.researchgate.net/profile/Jozef-Goetz/publication/374948299_Web_Framework/links/6538c8945d51a8012b6db544/Web-Framework.pdf [viitattu 15.12.2024]

Goodman, D., Morrison, M., Novitski, P. & Schupak, B. 2010. JavaScript bible. 7. painos. Indianapolis: Wiley Publishing, Inc. E-kirja. Saatavissa: https://kaakkuri.finna.fi/Record/nelli29_mamk.2580000000005493?sid=4908316035 [viitattu 19.1.2025]

Hunt, P. 2013. Why did we build React?. WWW-dokumentti. Päivitetty 5.6.2013. Saatavissa: https://legacy.reactjs.org/blog/2013/06/05/why-react.html [viitattu 23.3.2025]

Libby, A. 2022. Practical Svelte. New York: Apress. E-kirja. Saatavissa: https://ibookspdf.com/?p=view_product&id=8edd72158ccd2a879f79cb2538568fdc [viitattu 23.3.2025]

McGrath, M. 2020. Javascript in easy steps, 6th edition. Warwickshire: In Easy Steps Limited. E-kirja. Saatavissa: https://kaakkuri.finna.fi/Record/nelli29_mamk.5410000000415863?sid=4908277986 [viitattu 19.1.2025]

Mozilla. 2024. JavaScript. WWW-dokumentti. Päivitetty 5.5.2024. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/JavaScript [viitattu 15.12.2024]
NextJS. 2025. Introduction. WWW-dokumentti. Saatavissa: https://nextjs.org/docs [viitattu 23.3.2025]

Mozilla. 2025. Getting started with Svelte. WWW-dokumentti. Päivitetty 19.12.2024. Saatavissa: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries/Svelte_getting_started [viitattu 23.3.2025]

React. 2025. React. WWW-dokumentti. Saatavissa: https://react.dev/ [viitattu 23.3.2025]

Shukla, A. 2023. Modern JavaScript Frameworks and JavaScript's Future as a Full-Stack Programming Language. *Journal of Artificial Intelligence & Cloud Computing* 4. PDF-dokumentti. Saatavissa: https://www.researchgate.net/profile/Abhishek-Shukla-41/publication/377629693_Modern_JavaScript_Frameworks_and_JavaScript's_Future_as_a_Full-Stack_Programming_Language/links/65bbf3fd1bed776ae31ccd6c/Modern-JavaScript-Frameworks-and-JavaScripts-Future-as-a-Full-Stack-Programming-Language.pdf [viitattu 15.12.2024]

Stack Overflow. 2024. Technology. WWW-dokumentti. Saatavissa: https://survey.stackoverflow.co/2024/technology [viitattu 15.12.2024]

Strimling, Y. 2024. Delightful Documentation? Improving Documentation Quality with the Kano Model. In *2024 IEEE International Professional Communication Conference (ProComm)* 34-35. PDF-dokumentti. Saatavissa: https://www.researchgate.net/profile/Yoel-Strimling/publication/382528317_Extended_Abstract_Delightful_Documentation_Improving_Documentation_Quality_with_the_Kano_Model/links/66a1dc56705af536449553c4/Extended-Abstract-Delightful-Documentation-Improving-Documentation-Quality-with-the-Kano-Model.pdf [viitattu 15.12.2024]

Svelte. 2025. Documentation. WWW-dokumentti. Saatavissa: https://svelte.dev/docs [viitattu 6.4.2025]

The Linux Information Project, 2005. Documentation Definition. WWW-dokumentti. Päivitetty 23.2.2006. Saatavissa: https://www.linfo.org/documentation.html [viitattu 15.12.2024]

Thompson, M. 2022. Discontinued Long Term Support for AngularJS. WWW-dokumentti. Päivitetty 12.1.2022. Saatavissa: https://blog.angular.dev/discontinued-long-term-support-for-angularjs-cc066b82e65a [viitattu 2.3.2025]

Tieturi. 2024. Ohjelmoinnin viitekehykset. WWW-dokumentti. Saatavissa: https://www.tieturi.fi/koulutusala/ohjelmistokehitys/ohjelmoinnin-viitekehykset/ [viitattu 15.12.2024]

Vue.js. 2025a. Introduction. WWW-dokumentti. Saatavissa: https://vuejs.org/guide/introduction.html [viitattu 2.3.2025]

Vue.js. 2025b. SFC Syntax Specification. WWW-dokumentti. Saatavissa: https://vuejs.org/api/sfc-spec.html [viitattu 2.3.2025]

W3Schools. 2025a. JavaScript History. WWW-dokumentti. Saatavissa: https://www.w3schools.com/js/js_history.asp [viitattu 19.1.2025]

W3Schools. 2025b. TypeScript Introduction. WWW-dokumentti. Saatavissa: https://www.w3schools.com/typescript/typescript_intro.php [viitattu 3.2.2025]