Lappeenrannan teknillinen yliopisto
School of Engineering Science
Tietotekniikan koulutusohjelma

Kandidaatintyö

Teemu Juura

JavaScript-kehysten vertailu – Systemaattinen kirjallisuuskatsaus

Työn tarkastaja: Apulaisprofessori Antti Knutas

Työn ohjaaja: Apulaisprofessori Antti Knutas

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
School of Engineering Science
Tietotekniikan koulutusohjelma

Teemu Juura

JavaScript-kehysten vertailu – Systemaattinen kirjallisuuskatsaus

Kandidaatintyö

2019

32 sivua, 4 kuvaa, 4 taulukkoa, 2 liitettä

Työn tarkastaja: Apulaisprofessori Antti Knutas

Hakusanat: JavaScript, kehys, vertailu, systemaattinen kirjallisuuskatsaus
Keywords: JavaScript, framework, comparison, systematic mapping study

JavaScript-kehyksiä on olemassa tuhansia, eikä niiden vertailuun ole yleisiä metodeja. Tässä työssä selvitetään systemaattisen kirjallisuuskatsauksen avulla miten JavaScript-kehyksiä on erilaisissa tutkimuksissa vertailtu. Määritellään hakukysely ja valintakriteerit, joiden avulla valitaan analysoitavat tutkimukset. Kirjallisuudessa JavaScript-kehyksiä on vertailtu niiden ominaisuuksien ja käytännön testien perusteella. Tutkimukset kategorisoidaan vertailutavan ja tutkimusnäkökulman mukaan. Vertailukriteereitä tunnistetaan 20. Analysoiduista tutkimuksista suurin osa ehdottaa tapaa vertailla JavaScript kehyksiä. Johtopäätöksenä voidaan todeta, että systemaattiselle kehysten vertailumetodille olisi käyttöä. Analysoitujen tutkimusten määrä oli melko pieni verrattuna hakutulosten kokonaismäärään, joten tutkimustulokset ovat korkeintaan suuntaa antavia.

ABSTRACT

Lappeenranta University of Technology
School of Engineering Science
Degree Program in Computer Science

Teemu Juura

JavaScript framework comparison – A systematic mapping study

Bachelor's Thesis

32 pages, 4 figures, 4 tables, 2 appendices

Examiner: Assistant professor Antti Knutas

Keywords: JavaScript, framework, comparison, systematic mapping study

There are thousands of JavaScript frameworks and there exists no common method to compare them with. In this thesis a systematic mapping study is conducted in order to find out how and with what criteria JavaScript frameworks have been compared in literature. A search string and study selection criteria are defined based on which the studies to be analyzed are selected. Two ways of comparing JavaScript frameworks are identified. The studies are categorised based on the method of comparison and research type. 20 different comparison criteria are identified. Most of the analyzed studies were classified as solution proposals, which suggests that a need for a systematic method of comparing JavaScript frameworks exists. However the number of analyzed studies was rather small compared to the total number of search results, which affects the validity of the findings.

SISÄLLYSLUETTELO

LIITE1 arviointikriteerit ja mittarit

LIITE2 Aineiston kategorisointi

SYMBOLI- JA LYHENNELUETTELO

ACM	Association for Computing Machinery
Ajax	Asynchronous JavaScript and XML
ASAAM	Aspectual Software Architecture Analysis Method
CSS	Cascading Style Sheets
DoSAM	Domain-specific software architecture comparison model
HTML	Hypertext Markup Language
IEEE	Institute of Electrical and Electronics Engineers
JSON	JavaScript Object Notation
MPA	Multi Page Application
SPA	Single Page Application
SAAM	Software Architecture Analysis Method
XML	Extensible Markup Language

1 JOHDANTO

1.1 Tausta

Web-sovellusten suosio on kasvanut huomattavasti viimeisen kahdenkymmenen vuoden aikana. Web-kehityksen perustyökalujen, HTML:n, CSS:n ja JavaScript:n, rinnalle on kehitetty monia sovelluskehyksiä ja kirjastoja. JavaScript-kehys tai kirjasto on kokoelma apuohjelmia ja funktiota, jotka helpottavat selainyhteensopivan JavaScript-koodin tuottamista (Lennon, 2010).

Ongelmana on kirjastojen ja kehysten suuri määrä, minkä vuoksi voi olla ongelmallista valita, mitä kirjastoa käytetään. Jster.net -sivun mukaan JavaScript-kirjastoja on tarjolla tuhansia moniin eri tarkoituksiin (JSter). Verkkosovelluksia kehitettäessä on tärkeää valita oikea kehys käytettäväksi (Gizas et al., 2012). Väärän kehyksen valinta voi lisätä kehityskustannuksia sekä projektin kestoa (Ignacio Fernández-Villamor et al., 2008).

JavaScript-kehyksiä on vertailtu melko laajasti erilaisissa teknisissä blogeissa ja alan verkkosivuilla, mutta kirjallisuudessa kehyksien vertailua on tehty verrattain vähän (Pano et al., 2018). Useita tapoja vertailla sekä JavaScript, että muita sovelluskehyksiä on esitetty, mutta vakiintunutta käytäntöä tai mallia kehysten vertailuun ei näytä olevan. Sovelluskehysten vertailua lähinnä oleva tieteenala on sovellusarkkitehtuurien vertailu (Ignacio Fernández-Villamor et al., 2008). Sovellusarkkitehtuurien vertailemiseksi on kehitetty monia metodeja ja käytäntöjä (Kazman et al., 1998, 1994; Tekinerdogan, 2004).

1.2 Tavoitteet ja rajaukset

Työn tavoitteena on selvittää, miten JavaScript-kehyksiä voidaan vertailla toisiinsa. Tutkimuskysymykset ovat seuraavat:

1) Millä menetelmillä JavaScript-kehyksiä on vertailtu toisiinsa kirjallisuudessa?
2) Millä kriteereillä JavaScript-kehyksiä on vertailtu toisiinsa kirjallisuudessa?

Työssä luodaan kokonaiskuva sovelluskehyksien vertailusta. Kirjallisuuskatsauksella kartoitetaan kehysten vertailussa yleisesti käytetyt vertailukriteerit ja vertailutavat.

1.3 Työn rakenne

Luvussa 2 esitellään JavaScript-kehyksen perusominaisuudet. Lisäksi käydään läpi aiempaa tutkimusta sovelluskehyksien ja arkkitehtuurien vertailusta. Luvussa 3 käydään läpi tutkimusmenetelmä ja kuvaillaan sen vaiheet. Luvussa 4 esitetään saadut hakutulokset. Luvussa 5 vastataan tutkimuskysymyksiin ja käydään läpi tuloksien merkitys ja vastataan tutkimuskysymyksiin. Luvussa 6 kootaan tutkimustulokset.

2 AIEMPAA TUTKIMUSTA

Tässä luvussa esitellään aiheeseen liittyvää aiempaa tutkimusta. Ensin määritellään JavaScript-kehys ja esitellään sen yleiset ominaisuudet. Seuraavaksi esitellään aiempaa tutkimusta sovelluskehysten ja sovellusarkkitehtuurien vertailusta.

2.1 JavaScript-kehyksen yleiset ominaisuudet

Shanin ja Huan mukaan kehys on uudelleenkäytettävä sovellusrunko, joka sisältää palveluiden ja komponenttien rakennuspalikat (Shan & Hua, 2006). Lennonin mukaan JavaScript kehykset ja kirjastot sisältävät seuraavat perusominaisuudet:

- Valitsijat (eng.selectors)
- DOM -läpikäynti (eng. DOM - traversal)
- DOM -muokkaus (eng. DOM -manipulation)
- Apufunktiot (eng.Utility functions)
- Tapahtumien hallinta (eng.Event handling)
- AJAX

Valitsijat ja DOM:in läpikäynnin apuohjelmat helpottavat haluttujen elementtien valintaa. DOM:in muokkauksella tarkoitetaan näkymän muuttamista ja päivittämistä. JavaScript kehykset sisältävät apufunktioita, jotka nopeuttavat ohjelmistonkehitystä. Yleisiä apufunktioita ovat mm. listan läpikäynti. Eri selaimet reagoivat tapahtumiin eri tavalla, mikä on ongelmallista, kun halutaan tuottaa JavaScript koodia, joka toimii usealla selaimella. JavaScript-kehykset tarjoavat tavan käsitellä tapahtumia siten, että sovellus toimii samalla tavalla käytetystä selaimesta riippumatta. Ajax (Asynchronous JavaScript and XML) on tekniikka, jolla voidaan tehdä HTTP-pyyntöjä asynkronisesti JavaScriptin avulla. Ajax pyyntö palauttaa joko XML -tai JSON -muotoisen vastauksen. JavaScript-kehykset tarjoavat tavan tehdä Ajax pyyntöjä. (Lennon, 2010) Yleisesti JavaScript-kehykset helpottavat perusasioiden toteuttamista, jolloin kehittäjä voi keskittyä suurempiin kokonaisuuksiin sovelluskehityksessä.

2.2 Kehystenvertailu

Verrattain tuoreessa tutkimuksessa, Pano et al. (2018) selvittivät mitkä seikat vaikuttavat JavaScript-kehyksen valintaan. Tutkimuksessa tunnistettiin seuraavat kehyksen valintaan vaikuttavat ominaisuudet: käytettävyys, opittavuus, ymmärrettävyys, hinta, suorituskyky, koko ja toiminnallisuus, johon sisältyy mm. automatisointi ja laajennettavuus (Pano et al., 2018). Tutkimus ei varsinaisesti vertaillut erilaisia JavaScript-kehyksiä toisiinsa, mutta se esitteli ominaisuuksia, joiden perusteella kehyksen valinta usein tehdään.

Vuoden 2013 tutkimuksessa Heitkötter et al. vertailivat mobiilisovellus -kehyksiä toisiinsa. Tutkimuksessa lueteltiin kehysten vertailukriteereitä. Heitkötter et al. tunnistivat seuraavat kriteerit kehittäjän kannalta: lisenssi ja hinta, pitkän aikavälin soveltuvuus, dokumentaatio ja tuki, oppimisen helppous, kehityksen haastavuus, laajennettavuus ja ylläpidettävyys. Käyttäjän kannalta tärkeäksi havaittiin: käyttöliittymä, ulkoasu ja tuntuma, latausaika ja suorituskyky. (Heitkötter et al., 2013)

Gizas et al. vertailivat JavaScript-kehyksiä sovellusmittarien avulla. Tutkimuksessa suoritettiin laatu, validointi -ja suorituskykymittauksia kuudelle eri kehykselle. (Gizas et al., 2012) Tutkimuksessa esiteltiin tapa arvioida JavaScript-kirjastoja erilaisten laatumittarien avulla.

Graziotin & Abrahamsson suorittivat tutkimuksen perustuen Gizas et al. tutkimukseen. Tutkimuksessa selvitettiin mitä kehittäjät haluavat JavaScript-kehykseltä. Tutkimuksessa haastateltiin neljää front-end kehittäjää. Kehittäjiltä kysyttiin miten he valitsevat käytettävän JavaScript-kehyksen. Tutkimuksessa selvisi, että kehittäjät arvostavat dokumentaationlaatua ja yhteisöä. Tutkimuksessa ehdotetaan kaksikerroksista arviointimenetelmää, joka koostuu teoreettisesta kerroksesta perustuen Gizas et al.:n ehdottamiin mittareihin, sekä käytännön kerrokseen perustuen sovelluskehittäjien määrittelemiin kriteereihin. (Graziotin & Abrahamsson, 2013)

Vuonna 2018 Kaluža et al. vertailivat Front-End kehyksiä toisiinsa. Tutkimuksessa analysoitiin kolmen JavaScript-kehyksen soveltuvuutta sekä SPA (single-page-application)

että MPA (multi-page-application) sovellusten kehitykseen. Kehyksiä analysoitiin laatimalla joukko kysymyksiä perustuen SPA ja MPA sovelluksen vaatimuksiin. MPA arviointikriteerit olivat seuraavat: riittääkö pelkkä JavaScript sisällytys aloittamiseen, kehyksen suorituskyky, onko kehyksessä tietty työnkulku, palvelin renderöinti. SPA arviointikriteerit: suuren koodimäärän huollettavuus, riippuvuus kolmannen osapuolen paketeista, sovelluksen pakkaus, minimointi ja data-tilanhallinta. (Kaluža et al., 2018)

Ignacio Fernandez-Villamor et al. vertailivat ketteriä web-kehyksiä vuoden 2008 tutkimuksessa. Arviointitapa määriteltiin jotakuinkin DoSAM:in (Domain-Specific Software Architechture Comparison Model) mukaan. Tutkimuksessa määriteltiin abstraktio web-kehykselle. Arviointikriteerit määriteltiin tunnistamalla kehyksen perusominaisuudet. Perusominaisuudet selitettiin ja niistä muodostettiin kysymyksiä. Valitut kehykset arvioitiin sen perusteella, miten hyvin kunkin kehyksen ominaisuudet vastaavat määriteltyihin kysymyksiin. Arviointikategoriat olivat seuraavat: toiminta-alue, komponenttien käyttö, esitystapa, palvelupainotteisuus, käytettävyys, tietoturva, omaksuminen ja testattavuus. (Ignacio Fernández-Villamor et al., 2008)

2.3 Arkkitehtuurivertailu

Sovellusarkkitehtuuri vertailumetodeista tunnetuin lienee SAAM (software architecture analysis method) jonka esittivät Kazman et al. vuonna 1994. SAAM:issa arkkitehtuuria arvioidaan erilaisten skenaarioiden avulla. Skenaario on kuvaus vuorovaikutuksesta systeemin kanssa. (Kazman et al., 1994)

SAAM:n pohjalta on kehitetty ASAAM (Aspectual Software Architecture Analysis Method). ASAAM perustuu SAAM:iin, mutta koska arkkitehtuurit koostuvat useammista rakenteista, jotka keskittyvät eri alueisiin ne on otettava huomioon arkkitehtuureja analysoidessa. ASAAM tarkastelee arkkitehtuuria useammasta näkökulmasta. (Tekinerdogan, 2004)

Eräs toinen arkkitehtuurivertailu metodi on ATAM (Architecture Tradeoff Analysis Method). ATAM:in tavoitteena on vertailla kilpailevia arkkitehtuureja keskittymällä

arkkitehtuuri valinnan kompromisseihin ja niiden vaikutuksiin. (Kazman et al., 1998)

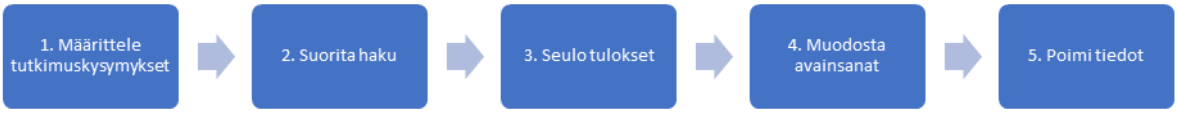
3 TUTKIMUSMENETELMÄ

Tässä luvussa esitellään työn tutkimusmenetelmä. Tiedonhakustrategia kuvaillaan. Lisäksi tiedon kategorisointi ja analysointi esitellään.

3.1 Systemaattinen kirjallisuuskatsaus

Tiedonkeräämisen metodina käytetiin systemaattista kirjallisuuskatsausta. Systemaattisen kirjallisuuskatsauksen tarkoituksena on antaa yleiskuva tutkimusalasta, sekä olemassa olevien tutkimusten ja niiden tuloksien määrästä ja laadusta (Petersen et al., 2008). Systemaattisessa kirjallisuuskatsauksessa on suoritettava seuraavat vaiheet: määritellään tutkimuskysymykset, joihin pyritään vastaamaan, tiedonhaussa käytettävät hakusanat, lähteiden valintakriteerit, tiedon kerääminen, tiedon analysointi ja luokittelu (Petersen et al., 2015).

Tutkimuskysymykset määritellään, jotta tiedetään mitä tietoa kirjallisuuskatsauksella haetaan. Lisäksi tehdään testihakuja tietokantoihin kysymysten validoimiseksi.
Prosessin vaiheet esitetään kuvassa 1.



Kuva 1 Prosessin vaiheet (Petersen et al. 2008)

3.2 Tiedonhaku ja valintaperusteet

Kirjallisuuskatsauksen tavoitteena oli muodostaa yleiskuva siitä, miten JavaScript-kehyksiä on kirjallisuudessa vertailtu. Kirjallisuuskatsauksen avulla kartoitetaan

yleisimmät sovelluskehysten vertailutavat ja vertailussa käytetyt kriteerit. Tiedonhaussa on syytä määritellä lähteiden valinta- ja hylkäyskriteerit (Petersen et al., 2008). Lähteiksi pyrittiin valitsemaan akateemisia lähteitä, kuten konferenssijulkaisuja ja tutkimusraportteja. JavaScript-kehyksiä on vertailtu paljon erilaisissa blogeissa ja muissa verkkolähteissä. Verkkolähteiden ongelmana on, ettei vertailukriteereiden valintaperusteita ole usein perusteltu. Verkkolähteiden käyttöä pyrittiin pääsääntöisesti välttämään.

Työn kannalta tarvittavaa tietoa haettiin julkisista akateemisista tietokannoista, kuten IEEE:sta (Institute of Electrical and Electronics Engineers), ACM:sta (Association for Computing Machinery), Google Scholarista ja Lappeenrannan tiedekirjaston Finna-palvelusta. Lisäksi lähteiden on oltava saatavilla maksutta digitaalisessa muodossa, joko suomeksi tai englanniksi. Valintakriteerit esitetään taulukossa 1.

Taulukko 1 Aineiston valintakriteerit

Hyväksytään	**Hylätään**
• Vertaillaan tai arvioidaan jotakin kehystä tai kehyksiä • Englannin- tai suomenkielisiä • Lähteeseen on viitattu vähintään kahdesti • Pääsy artikkeliin	• Muut kuin englannin- tai suomenkieliset • Kandidaatintyötä vastaavat opinnäytetyöt • Ei vapaasti saatavilla

Työn kannalta kiinnostavaa tietoa olivat vertailukriteerit, niiden valintaperusteet ja se, miten kyseisiin kriteereihin on vastattu. Tutkimusaiheen kannalta oleellisia hakutermejä ovat *JavaScript*, *Framework tai library* ja *comparison.* Valittuihin tietokantoihin tehtiin aluksi testihakuja hakutulosten laajuuden selvittämiseksi. Testihakuja tehtiin usein erilaisin hakusanayhdistelmin, mutta jokaisessa haussa esiintyivät ainakin *JavaScript* ja *Framework*. Testihakujen tuloksia käytiin suurpiirteisesti läpi, jotta saatiin selville, tuottiko haku tutkimusaiheen kannalta oleellisia tuloksia.

Testihakuja tehtäessä huomattiin, että hakutermi: *(”JavaScript” OR ”architecture”) AND*

(”Framework” OR ”library”) AND (”Comparison” OR ”evaluation”) tuotti aivan liian paljon tuloksia Google Scholar: 4 140 000, ACM: 4039, IEEE: 5026.

Hakuehtoja täytyi tarkentaa. Hakuehtoja tarkennettiin pudottamalla *”architechture”* pois hakukyselystä, sillä se ei suoranaisesti liity aiheeseen. Tällöin hakutermit olivat *(”JavaScript”) AND (”Framework” OR ”library”) AND (”Comparison” OR ”evaluation”).* Tällöin Google scholar tuotti 396 000 tulosta, ACM tuotti 115 ja IEEE 64 tulosta. Google Scholaria luukun ottamatta tulosten määrä oli sopiva käsiteltäväksi. Google Scholarin tapauksessa hakutuloksia pyrittiin rajaamaan lisäämällä hakukyselyyn sanoja, kuten *software* ja *web development* sekä rajaamalla haun aikaväliä. Yrityksistä huolimatta Google Scholarin tuloksia ei saatu rajattua tarpeeksi. Pienin määrä hakutuloksia oli noin 7 000, joka oli silti aivan liikaa, eikä kyseinen haku tuottanut sopivia tuloksia. Google Scholarin hakutuloksia ei saada kylliksi rajattua, joten näistä otettiin huomioon 200 ensimmäistä.

Taulukko 2 Valitut tulokset tietokannoittain

Tietokanta	**Tulokset (Kaikki/valitut)**
ACM	115/4
IEEE	64/4
Google Scholar	200/26
Yhteensä	**379/31**

Tiedonhaun apuna käytettiin Zotero-työkalua. Aineiston kategorisoinnissa ja tiedon poiminnassa hyödynnettiin excel-taulukoita.

3.3 Aineiston kategorisointi

Tutkimukset kategorisoitiin tiivistelmien perusteella. Tiivistelmien ollessa riittämätön tutkimuksen kategorisointiin luettiin lisäksi johdanto sekä johtopäätökset ja arviointikriteerit. Tutkimustenkategorioiksi muodostuivat teoreettiset, ominaisuuksiin perustuvat kehysten arvioinnit, käytännön arviot ja tutkimusnäkökulmat. Lisäksi vertailukriteerit olivat yksi kategoria.

Teoreettinen vertailu määriteltiin siten, että tutkimuksessa käsiteltäviä kehyksiä vertailtiin niiden ominaisuuksien perusteella. Tämän kategorian tutkimuksissa ei käytetty prototyyppiä eikä suoritettu minkäänlaisia käytännön testejä. Käytännön arviot määritettiin seuraavasti: jos tutkimuksessa toteutettiin jokin ohjelmademo tai käytettiin olemassa olevaa demoa arvioinnissa, luettiin se käytännön arvioiksi.

Vertailukriteereillä tarkoitetaan ominaisuuksia ja mittareita, joiden perusteella kehyksiä vertailtiin valitussa aineistossa. Vertailukriteereitä tunnistettiin 20 kappaletta. Vertailukriteerit esitetään liitteessä 2. Vertailukriteereissä esiintyy hieman päällekkäisyyksiä. Tutkimuksissa tietyt kriteerit esiintyivät hieman eri nimillä, esimerkiksi van der Geest & Ettema (2011) ottivat kriteeriksi yhteisön tuen, kun taas muissa tutkimuksissa se oli usein määritelty yhteisöksi.

Wieringa:n et al. mukaan tutkimusten näkökulmat voidaan luokitella seuraaviin kuuteen ryhmään: validaatiotutkimus, arvioiva tutkimus, ratkaisuehdotus, filosofinen tutkimus, kokemukseen perustuva tutkimus ja mielipiteeseen perustuva tutkimus. (Wieringa et al., 2006) Tutkimusnäkökulmat on esitetty tarkemmin taulukossa 3.

Taulukko 3 Tutkimusnäkökulmat ja niiden kuvaukset (Wieringa et al., 2006)

Näkökulma	**Kuvaus**
Arvioiva tutkimus	Tutkii ongelmaa käyttäen olemassa olevaa, menetelmää tai työkalua.
Validaatiotutkimus	Tutkii ratkaisuehdotusta, jota ei ole vielä käytetty. Tutkimusmetodina esimerkiksi: koe, simulaatio, prototyyppi tai matemaattinen analyysi.
Ratkaisuehdotus	Tutkimuksessa esitetään tapa ratkaista ongelma. Saattaa sisältää jonkinlaisen esimerkin ratkaisusta.
Filosofinen tutkimus	Esittää uuden näkökulman tai esittelee työkalun.
Mielipiteeseen perustuva tutkimus	Tutkimus perustuu kirjoittajan mielipiteeseen.
Kokemukseen perustuva tutkimus	Tutkimus perustuu kirjoittajan kokemukseen.

4 KIRJALLISUUSKATSAUKSEN TULOSTEN KARTOITUS

Tässä luvussa käydään läpi kirjallisuuskatsauksen tulokset. Kategorisoidut tulokset, niiden määrä ja niiden merkitys kuvataan. Taulukossa 4 on esitetty systemaattinen kartta aineistosta. Taulukossa vaaka-akselilla on tutkimuksen kategoria ja pystyakselilla tutkimusnäkökulma. Seuraavissa aliluvuissa käydään kategoriat tarkemmin läpi

Taulukko 4 Systemaattinen kartta

	Käytännön vertailut (11)	Teoreettiset vertailut (18)	Kategoriattomat (2)
Ratkaisuehdotus (21)	(Delcev & Draskovic, 2018; Dhillon & Mahmoud, 2015; Gizas et al., 2012; Malmström, 2014; Molin, 2016; Prokofyeva & Boltunova, 2017; Raigoza & Thakkar, 2016)	(Graziotin & Abrahamsson, 2013; Heitkötter et al., 2013; Ignacio Fernández-Villamor et al., 2008; Kähkönen, 2014; Kaluža et al., 2018; Majchrzak et al., 2017; Mora & Nadi, 2018; Pano et al., 2018; Razak & Ghazali, 2011; Salas-Zárate et al., 2015; Samkari & Joukhadar, 2008; Sommer & Krusche, 2013; Sultan, 2017; van der Geest & Ettema, 2011)	
Validaatiotutkimus (3)	(Ferreira, 2018; Misra & Cafer, 2012; Ocariza et al., 2015)		
Arvioiva tutkimus (1)	(Mariano, 2017)		
Filosofinen tutkimus (6)		(Nitze & Schmietendorf, 2014; Ramos et al., 2018, 2016; Shan & Hua, 2006)	(Jain et al., 2014; Laine et al., 2011)

4.1 Käytännön vertailut

Käytännön vertailuiksi kategorisoiduissa tutkimuksissa kehyksiä vertailtiin ja arvioitiin hyödyntäen jotakin käytännön testiä tai prototyyppiä. Lähes kaikissa käytännönarvioiksi kategorisoiduissa tutkimuksissa kehyksiä vertailtiin myös teoreettisesti. Ainoastaan (Gizas et al., 2012; Misra & Cafer, 2012; Raigoza & Thakkar, 2016) suorittamissa tutkimuksissa arviointi tehtiin pelkästään käytännöntestien perusteella.

Käytännön vertailuiksi kategorisoiduista tutkimuksista suurin osa oli ratkaisuehdotuksia. Ratkaisuehdotukset voidaan jakaa edelleen yleisiin vertailuihin, joissa kehysten hyviä ja huonoja puolia käytiin läpi (Delcev and Draskovic, 2018), ja vertailukehyksen esittämisiin, joissa pyrittiin luomaan toimintatapa kehysten vertailuun (Dhillon & Mahmoud, 2015; Gizas et al., 2012; Malmström, 2014; Molin, 2016; Prokofyeva & Boltunova, 2017; Raigoza & Thakkar, 2016).

Validaatiotutkimuksia aineistossa oli yhteensä kolme. (Misra & Cafer, 2012) esittivät ja validoivat tavan arvioida JavaScript-koodin laatua, (Ferreira, 2018) käytti yleisiä tehokkuuden mittausmenetelmiä. (Ocariza et al., 2015) toteuttivat ja testasivat työkalua JavaScript-ohjelmien epäkohtien löytämiseen.

4.2 Teoreettiset vertailut

Teoreettisiksi vertailuiksi kategorisoiduissa tutkimuksissa sovelluskehyksiä vertailtiin teoreettisesti ja seikkaperäisesti eikä käytännön testejä tai prototyyppejä käytetty. Ratkaisuehdotus oli tässäkin kategoriassa yleisin tutkimusnäkökulma.

Tutkimukset jakautuivat yleisten arviointitapojen esityksiin (Graziotin and Abrahamsson, 2013; Ignacio Fernández-Villamor et al., 2008; Pano et al., 2018; van der Geest and Ettema, 2011) ja tiettyyn käyttöön kohdistettuun arviointiin (Heitkötter et al., 2013; Kähkönen, 2014; Kaluža et al., 2018; Majchrzak et al., 2017; Mora & Nadi, 2018; Razak & Ghazali, 2011; Salas-Zárate et al., 2015; Samkari & Joukhadar, 2008; Sommer & Krusche, 2013; Sultan, 2017). Kohdistetut vertailut keskittyivät mm. arvioimaan kehysten

soveltuvuutta mobiilikehitykseen (Heitkötter et al., 2013) MVC -mallin hyödyntämiseen (Kähkönen, 2014) ja Front-End kehitykseen (Kaluža et al., 2018). Arviointi perustettiin usein laatuominaisuuksiin, ja kehittäjien toiveisiin (Graziotin & Abrahamsson, 2013; Pano et al., 2018).

Kahdessa tutkimuksessa vertailumetodi perustettiin osittain johonkin arkkitehtuurivertailun metodiin (Ignacio Fernández-Villamor et al., 2008; Molin, 2016). Yksi tutkimus perusti arviointinsa osittain jonkin ISO-standardiin (Kähkönen, 2014). Muissa tutkimuksissa arviointi tehtiin vertailtavien kehysten ominaisuuksien perusteella.

Filosofisia tutkimuksia oli 4. Filosofisia tutkimuksia olivat Java web kehysten taksonomia (Shan & Hua, 2006), modulaarisuuden esittely (Nitze & Schmietendorf, 2014) sekä kaksi selvitystä kehyksen ominaisuuksista kyselytutkimuksilla (Ramos et al., 2018, 2016).

4.3 Kategoriattomat tutkimukset

Osaa tutkimuksista ei voitu luokitella teoreettisiksi tai käytännön vertailuiksi. Näissä tutkimuksissa ei suoritettu vertailua lainkaan, vaan esiteltiin ja arvioitiin yksittäistä kehystä tai sen käyttöä. Molemmat kategoriattomat tutkimukset olivat filosofisia tutkimuksia, joissa esiteltiin, jokin kehys ja sen ominaisuuksia (Jain et al., 2014; Laine et al., 2011).

4.4 Vertailukriteerit ja mittarit

Tehokkuutta mitattiin yleisesti ohjelmademolla ja erilaisilla työkaluilla (Dhillon & Mahmoud, 2015; Ferreira, 2018; Gizas et al., 2012; Graziotin & Abrahamsson, 2013; Malmström, 2014; Mariano, 2017; Molin, 2016; Raigoza & Thakkar, 2016). Tehokkuuden mittaamiseen käytettiin toisinaan myös JavaScript-paketin kokoa (Kähkönen, 2014; Pano et al., 2018; Sommer & Krusche, 2013). Myös kehyksen ominaisuuksia, kuten välimuisti (eng. caching) tukea käytettiin tehokkuuden arviointiin (Malmström, 2014; Molin, 2016).

Yhteisön suosiota ja tukea mitattiin yleisesti StackOverflow, Google ja GitHub statistiikan avulla (Ferreira, 2018; Graziotin & Abrahamsson, 2013; Jain et al., 2014; Majchrzak et al., 2017; Ramos et al., 2018; Sultan, 2017; van der Geest & Ettema, 2011). Opittavuutta mitattiin oppimisen nopeuden (Heitkötter et al., 2013; Pano et al., 2018; Samkari &

Joukhadar, 2008) ja helppouden (Molin, 2016) avulla. Siirrettävyyttä arvioitiin kehyksen osien sidoksellisuuden (Kähkönen, 2014), alusta- ja selaintuen (Laine et al., 2011; Malmström, 2014; Molin, 2016) ja kehyksen vaihtamisen helppouden (Mora & Nadi, 2018) avulla.

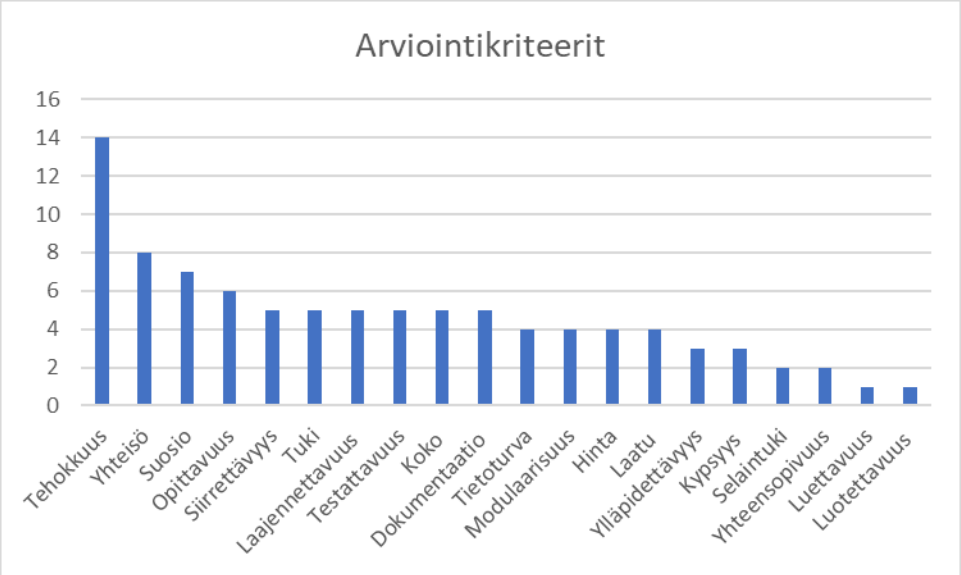
Tukea arvioitiin kehyksen kehittäjän (Heitkötter et al., 2013; Majchrzak et al., 2017) ja päivitystiheyden mukaan (Heitkötter et al., 2013; Mora & Nadi, 2018; Pano et al., 2018; Prokofyeva & Boltunova, 2017). Laajennettavuutta arvioitiin uusien ominaisuuksien lisäyksen helppouden perusteella (Heitkötter et al., 2013; Kähkönen, 2014; Pano et al., 2018; Salas-Zárate et al., 2015). Testattavuutta arvioitiin kehyksen tarjoamien testaustyökalujen mukaan (Ignacio Fernández-Villamor et al., 2008; Malmström, 2014; Molin, 2016; Salas-Zárate et al., 2015; Sommer & Krusche, 2013). Kehyksen koko oli toisinaan erillisenä arviointikategoriana tehokkuudesta. Kokoa mitattiin kehyksen JavaScript-tiedoston koon avulla (Delcev & Draskovic, 2018; Ferreira, 2018; Gizas et al., 2012; Heitkötter et al., 2013; Kähkönen, 2014; Kaluža et al., 2018; Laine et al., 2011; Pano et al., 2018; Sommer & Krusche, 2013; van der Geest & Ettema, 2011).

Dokumentaatiota arvioitiin sen laajuuden (Delcev & Draskovic, 2018; Graziotin & Abrahamsson, 2013; Heitkötter et al., 2013; Malmström, 2014; Sommer & Krusche, 2013) ja pelkän olemassaolon avulla (Molin, 2016; Salas-Zárate et al., 2015). Tietoturvaa mitattiin erilaisien testien (Delcev & Draskovic, 2018; Ocariza et al., 2015) ja kehyksen ominaisuuksien, kuten sivustojen välisen skriptauksen (eng. cross site scripting) estämisen avulla (Ignacio Fernández-Villamor et al., 2008; Laine et al., 2011; Molin, 2016; Salas-Zárate et al., 2015; Samkari & Joukhadar, 2008). Modulaarisuutta mitattiin sen mukaan, miten kehys jakoi toiminnallisuuden (eng. seperation of concens) (Heitkötter et al., 2013; Malmström, 2014; Molin, 2016; Nitze & Schmietendorf, 2014; Pano et al., 2018).

Kehyksiä arvioitiin myös niiden hinnan, ja lisenssin tyypin avulla (Heitkötter et al., 2013; Pano et al., 2018; Sommer & Krusche, 2013). Kehyksien laatua mitattiin erilaisin laatumittarien avulla (Gizas et al., 2012; Graziotin & Abrahamsson, 2013; Misra & Cafer, 2012). Ylläpidettävyyttä arvioitiin laatumittareiden (Gizas et al., 2012; Molin, 2016), koodin ymmärrettävyyden (Heitkötter et al., 2013), sovellusarkkitehtuurin (Kähkönen,

2014) sekä kehittäjän ja käyttäjämäärän mukaan (Molin, 2016). Kypsyyttä mitattiin käyttäjämäärän (Ignacio Fernández-Villamor et al., 2008) ja yrityskäyttäjien mukaan (Molin, 2016). Selaintukea mitattiin tuettujen selainversioiden mukaan (Delcev & Draskovic, 2018; Malmström, 2014; Molin, 2016; van der Geest & Ettema, 2011).

Kuvassa 2 on eritelty löydetyt arviointikriteerit ja niiden määrä.



Kuva 2 Kriteerien esiintyvyys

4.5 Vertailukriteerit kategorian mukaan

Kuvassa 3 on esitetty kriteerien esiintyminen tutkimuskategorian mukaan.

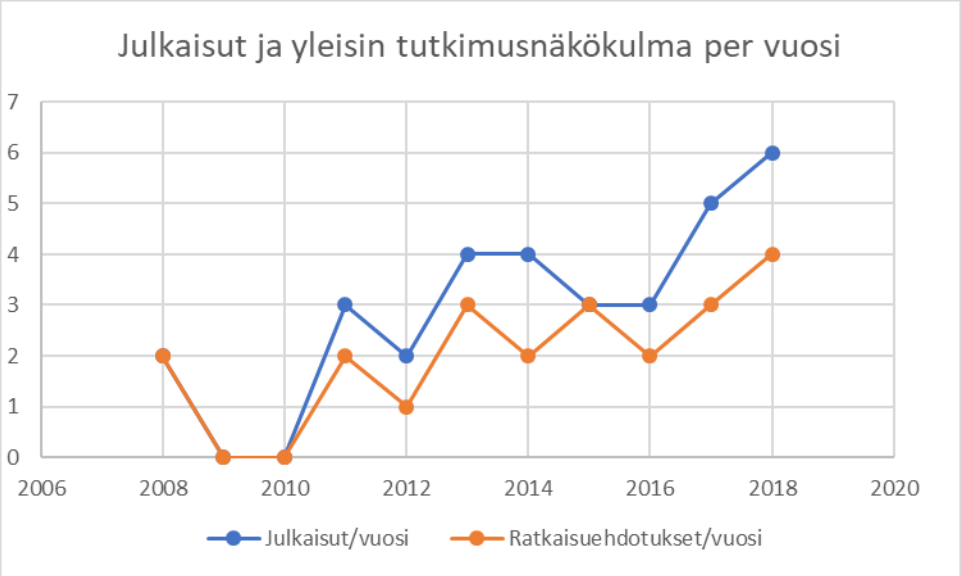
	Käytännön	Teoreettine	Kategoriaton
Tehokkuus	7	7	0
Yhteisö	1	6	1
Suosio	3	4	0
Opittavuus	2	4	0
Siirrettävyys	2	2	1
Tuki	1	3	1
Laajennettavu	0	5	0
Testattavuus	2	3	0
Koko	1	4	0
Dokumentaati	2	3	0
Tietoturva	2	2	1
Modulaarisuus	2	2	0
Hinta	1	3	0
Laatu	3	1	0
Ylläpidettävyy	1	2	0
Kypsyys	2	1	0
Selaintuki	1	1	0
Yhteensopivuu	1	0	1
Luettavuus	0	1	0
Luotettavuus	0	0	1

Kuva 3 Vertailukriteeri tutkimus kategorian mukaan

Teoreettisissa vertailuissa yhteisöä ja suosiota mitattiin enemmän kuin käytännön vertailuissa. Käytännön vertailuissa tehokkuus ja kehyksen laatu olivat suosion ohella käytetyimmät kriteerit. Laatua mitattiin usein erilaisin koodianalyysein ja mittarein, joten ei ole yllättävää, että niitä arvoitiin ohjelmademojen avulla.

4.6 Julkaisujen määrä

Kuvassa 4 on havainnollistettu valitut tutkimukset julkaisuvuotta kohden. Aikaisimmat valitut tutkimukset ovat vuodelta 2008. Seuraavat tutkimukset ovat vuodelta 2011. Kuvajasta nähdään, että tutkimusten määrä pysyi tasaisena välillä 2014 – 2016. Tutkimusten määrä näyttää nousseen vuodesta 2016 lukien.



Kuva 4 Julkaisut vuosittain

5 POHDINTA, TULEVAISUUS JA RISKIT

Tässä luvussa esitetään työn tulokset. Lisäksi vastataan luvussa 1 asetettuihin tutkimuskysymyksiin. Myös tutkimuksen riskit esitetään.

5.1 Vertailutapa

Kirjallisuuden kartoituksen perusteella selvisi, että JavaScript-kehyksiä on vertailtu sekä teoreettisesti ominaisuuksiin perustuen, että käytännön testein. Teoreettisessa vertailussa kehyksiä vertailtiin joko painotetuin, tai binäärisin kriteerein. Käytännön vertailussa kehyksiä arvioitiin erilaisin käytännön testein ja koodin laatumittarein. Lähes kaikki käytännön testejä sisältävät tutkimukset sisälsivät myös teoreettista ominaisuuksien vertailua. Hieman yllättävää oli, että teoreettisissa vertailuissa tehokkuus esiintyi yhtä usein kuin käytännön arvioinneissa. Tämä oli yllättävää, sillä tehokkuutta voisi olettaa olevan helpompi arvioida ohjelmademon avulla. Kehyksen ominaisuuksien perusteella saadaan vain suuntaa antava käsitys sen suorituskyvystä. Toisaalta suorituskyky, tai sen puute huomataan sovellusta käytettäessä helposti, minkä vuoksi sen mittaaminen ja arviointi koetaan tärkeäksi. Käytännössä kaikissa tutkimuksissa muutamaa lukuun ottamatta kehyksien ominaisuudet otettiin huomioon arvioinnissa.

5.2 Vertailukriteerit ja mittarit

Yleisimpiä vertailukriteereitä olivat seuraavat: tehokkuus, yhteisö, suosio, opittavuus, siirrettävyys, tuki, laajennettavuus, testattavuus, koko, dokumentaatio ja tietoturva. Käytännön testejä sisältävissä vertailuissa yleisimpiä kriteereitä olivat tehokkuus ja laatu. Teoreettisissa vertailuissa yleisimpiä kriteereitä olivat, tehokkuus, opittavuus, yhteisö, suosio, dokumentaatio, laajennettavuus, testattavuus ja koko.

Tunnistetuissa kriteereissä oli useita päällekkäisyyksiä. JavaScript-kehyksen koko esiintyi usein tehokkuuden mittarina, mutta toisinaan se oli erotettu omaksi kategoriakseen. Myös tuki, yhteisö ja suosio sisälsivät samoja mittareita. Yleisin suosion mittari olivat erilaiset verkkohakujen, kuten YouTube, Google ja StackOverflow, tuloksien määrä.

Verkkohakujen tulokset luettiin toisinaan myös tuen ja dokumentaation mittareiksi (Heitkötter et al., 2013). Suosiolla, tuella ja yhteisöllä tarkoitettiin yleisesti sellaisia asioita, jotka helpottavat kehyksen käyttöä pitkällä aikavälillä. Kehyksen suosio vaikuttaa siihen, miten helposti ongelmatilanteissa löydetään apua niin kehittäjältä kuin kehyksen käyttäjäkunnalta. Myös kehyksen kehitystiimin koko ja käyttäjämäärä vaikuttavat osaltaan ylläpidettävyyteen. Mikäli suuret yritykset käyttävät kehystä jossakin projektissaan, voidaan olettaa, että kehystä tuetaan pidempään kuin vähäisesti käytettyä kehystä.
Kriteerit voidaan ryhmitellä karkeasti tukeen ja yhteisöön, tehokkuuteen ja ylläpidettävyyteen.

5.3 Tulevaisuus

Monissa tutkimuksista tuotiin esille se seikka, että JavaScript kehyksien vertailua on tutkittu kirjallisuudessa vähän. Vähäisen tutkimuksen määrää tukee se, että suurin osa tutkimuksista luokiteltiin ratkaisuehdotuksiksi. Ehdotuksissa kehyksiä vertailtiin hyvin samankaltaisten ominaisuuksien perusteella. Kehyksiä vertailukriteerit valittiin muutamassa tutkimuksessa kyselytutkimuksen avulla (Graziotin & Abrahamsson, 2013; Molin, 2016; Pano et al., 2018; Ramos et al., 2018, 2016). Validaatiotutkimuksissa ei esitetty tapaa vertailla JavaScript-kehyksiä. Validaatiotutkimukset keskittyvät erilaisiin JavaScript-koodin laadun mittaamismetodeihin. Yleisesti näyttää olevan selvää mitä ominaisuuksia JavaScript-kehyksissä arvostetaan. Toisaalta konteksti, jossa kehystä arvioidaan vaikuttaa painotettuihin ominaisuuksiin. Tieteellisessä tutkimuksessa arvostetaan erilaisia ominaisuuksia kuin alalla (Graziotin & Abrahamsson, 2013). Kehyksiä vertaillessa lienee syytä ottaa huomioon mihin tarkoitukseen kehystä halutaan käyttää. Tutkimusten julkaisutiheydestä huomataan, että kehyksien vertailututkimusten määrä on kasvanut vuodesta 2016 lukien. Kiinnostus kehystenvertailuun selittynee sillä, että nykyisin suurin osa web-sovelluksista on toteutettu kokonaan tai ainakin osittain jotakin JavaScript-kehystä käyttäen.

Tunnistetuissa vertailukriteereissä oli huomattavasti päällekkäisyyksiä. Samoja asioita oli vertailtu erinimisillä kriteereillä usein. Mielekkäämpää olisi, jos kehysten vertailuun olisi olemassa standardoidut kriteerit. Aineistoa läpikäytäessä huomattiin, että JavaScript-kehyksellä ja JavaScript-kirjastolla tarkoitettiin usein samaa asiaa. Tarkalle kehyksen ja

kirjaston määritelmälle voisi myös olla tarvetta.

Suuri ratkaisuehdotusten määrä verrattuna validaatiotutkimuksiin ja arvioiviin tutkimuksiin tarkoittanee, ettei JavaScript-kehysten vertailemiseksi ole kehitetty yleisiä käytäntöä ja metodeja toisin kuin sovellusarkkitehtuurien vertailuun. Tutkimusnäkökulmien nojalla voitanee olettaa, että tällä tutkimusalalla olisi tarvetta yleiselle kehystenvertailumetodille. Tällaisen metodin tarpeellisuuden selvittämiseksi vaadittaisiin kuitenkin kattavampi kirjallisuudenkartoitustutkimus. Esitettyjä vertailumetodeja olisi syytä validoida ja arvioida. Toisaalta kuten (Ignacio Fernández-Villamor et al., 2008) tutkimuksessa kävi ilmi, voidaan arkkitehtuurivertailun malleja ja metodeja hieman muunneltuna hyödyntää myös sovelluskehysten vertailuun. Sovelluskehykset ja arkkitehtuurit kuitenkin eroavat toisistaan jonkin verran, mutta niiden valinta- ja arviointiperusteet ovat melko samanlaisia.

5.4 Tutkimuksen riskit

Hakutulosten määrän rajaus tuotti merkittävästi ongelmia. Varsinkin Google Scholarin tuloksista käsiteltiin vain pieni osa, minkä vuoksi paljon oleellista tutkimusta jäi varmasti huomiotta. Hakusanat tuottivat myös merkittävän määrän tutkimusta, joka ei suoraan liittynyt JavaScript-kehysten vertailuun. ACM:n ja IEEE:n tuloksista hyvin pieni osa olivat oleellisia, mistä voidaan päätellä, että muodostettu hakukysely oli riittämätön tai että oleellista tutkimusta ei ole indeksoitu riittävän tarkasti. Toisaalta Google Scholar tuotti eniten aiheeseen liittyviä tuloksia, joista moni kuitenkin lopulta oli peräisin joko ACM:stä tai IEEE:stä. Toisaalta epäoleelliset hakutulokset voivat johtua myös JavaScript-kehysten vertailuun liittyvän tutkimuksen vähäisyydestä.

Tulosten valintakriteerit pyrittiin määrittelemään siten, että aineisto olisi mahdollisimman korkealaatuista. On myös mahdollista, että valintaprosessin aikana joitakin oleellisia tutkimuksia ei ole valittu. Tulokset kategorisoitiin melko suurpiirteisesti tässä työssä. Lisäksi on mahdollista, että osa tutkimuksista on kategorisoitu väärin riittämättömän lukusyvyyden vuoksi. Kategorisointiin olisi ollut hyvä olla enemmän aikaa. Tuloksien merkittävyyteen vaikuttaa myös pieni otoskoko verrattuna hakutulosten kokonaismäärään. Otoskoon oli oltava tarpeeksi pieni, jotta tutkimusprosessi saataisiin ajoissa valmiiksi.

Parempien tulosten saamiseksi olisi vaadittu joko enemmän aikaa tai useampi tekijä.

6 YHTEENVETO

Työn tavoitteena oli selvittää, miten JavaScript-kehyksiä on vertailtu toisiinsa. Lisäksi selvitettiin mitkä olivat yleisimmät vertailukriteerit kehyksiä vertailtaessa. Työssä tehtiin systemaattinen kirjallisuuskatsaus. Kirjallisuuskatsauksella luotiin yleiskuva siitä, miten JavaScript-kehyksiä on vertailtu. Hakukyselyksi muodostui *(”JavaScript”) AND (”Framework” OR ”library”) AND (”Comparison” OR ”evaluation”).* Hakutuloksista valittiin yhteensä 31 kolmesta eri tietokannasta, jotka olivat Google Scholar, ACM ja IEEE. Tulokset kategorisoitiin vertailun tyypin, vertailukriteerien ja tutkimusnäkökulman mukaan.

Yleisimmät vertailukriteerit olivat seuraavat: tehokkuus, yhteisö, suosio, opittavuus, siirrettävyys, tuki, laajennettavuus, testattavuus, koko, dokumentaatio ja tietoturva. Kehyksiä arvioitiin niiden ominaisuuksien ja käytännön testien avulla. Lähes jokaisessa tutkimuksessa kehyksiä vertailtiin niiden ominaisuuksien avulla.

Yleisin tutkimusnäkökulma oli ratkaisuehdotus, joita oli yhteensä 19. Filosofisia tutkimuksia tunnistettiin seitsemän, validaatiotutkimuksia kolme ja arvioivia tutkimuksia yksi. Tutkimusnäkökulmien nojalla voitanee olettaa, että yleiselle kehystenvertailumetodille voisi olla tarvetta. Analysoitujen tutkimusten määrä oli melko pieni verrattuna hakutulosten kokonaismäärään, joten tutkimustulokset ovat korkeintaan suuntaa antavia, eikä niistä voida tehdä vahvoja johtopäätöksiä.

LÄHTEET

Delcev, S., Draskovic, D., 2018. Modern JavaScript frameworks: A Survey Study, in: 2018 Zooming Innovation in Consumer Technologies Conference (ZINC). Presented at the 2018 Zooming Innovation in Consumer Technologies Conference (ZINC), pp. 106–109.

Dhillon, S., Mahmoud, Q.H., 2015. An evaluation framework for cross-platform mobile application development tools. Softw. Pract. Exp. 45, 1331–1357.

Ferreira, J., 2018. A JavaScript Framework Comparison Based on Benchmarking Software Metrics and Environment Configuration 108.

Gizas, A., Christodoulou, S., Papatheodorou, T., 2012. Comparative evaluation of javascript frameworks, in: Proceedings of the 21st International Conference Companion on World Wide Web - WWW '12 Companion. Presented at the the 21st international conference companion, ACM Press, Lyon, France, p. 513.

Graziotin, D., Abrahamsson, P., 2013. Making Sense out of a Jungle of JavaScript Frameworks: towards a Practitioner-friendly Comparative Analysis. ResearchGate.

Heitkötter, H., Majchrzak, T.A., Ruland, B., Weber, T., 2013. Evaluating Frameworks for Creating Mobile Web Apps 13.

Ignacio Fernández-Villamor, J., Díaz-Casillas, L., Iglesias, C.Á., 2008. A Comparison Model for Agile Web Frameworks, in: Proceedings of the 2008 Euro American Conference on Telematics and Information Systems, EATIS '08. ACM, New York, NY, USA, pp. 14:1–14:8.

Jain, N., Mangal, P., Mehta, D., 2014. AngularJS: A Modern MVC Framework in JavaScript 7.

Javascript Territory - JSter Javascript Catalog [WWW- dokumentti]. [Viitattu 20.3.2019] Saatavilla: http://jster.net/

Kähkönen, J., 2014. MVC-malliin perustuvien Javascript-sovelluskehysten vertailua. Pro gradu -tutkielma

Kaluža, M., Troskot, K., Vukelić, B., 2018. COMPARISON OF FRONT-END FRAMEWORKS FOR WEB APPLICATIONS DEVELOPMENT. Zb. Veleuč. U Rijeci 6, 261–282.

Kazman, R., Bass, L., Abowd, G., Webb, M., 1994. SAAM: a method for analyzing the properties of software architectures, in: Proceedings of 16th International Conference on Software Engineering. Presented at the Proceedings of 16th International Conference on Software Engineering, pp. 81–90.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., Carriere, J., 1998. The architecture tradeoff analysis method, in: Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No.98EX193). Presented at the Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No.98EX193), pp. 68–78.

Laine, M., Shestakov, D., Litvinova, E., Vuorimaa, P., 2011. Toward Unified Web Application Development. IT Prof. 13, 30–36.

Lennon, J., n.d. Compare JavaScript frameworks. 2010 18.

Majchrzak, T.A., Biørn-Hansen, A., Grønli, T.-M., 2017. Comprehensive Analysis of Innovative Cross-Platform App Development Frameworks 10.

Malmström, T.J., 2014. Structuring modern web applications A study of how to structure web clients to achieve modular, maintainable and long lived applications. Diplomityö

Mariano, C.L., 2017. Benchmarking JavaScript Frameworks. Dublin Institute of Technology. Diplomityö

Misra, S., Cafer, F., 2012. Estimating Quality of JavaScript | Request PDF. ResearchGate.

Molin, E., 2016. Comparison of Single-Page Application Frameworks: A method of how to compare Single-Page Application frameworks written in JavaScript. Diplomityö

Mora, F.L. de la, Nadi, S., 2018. Which Library Should I Use?: A Metric-Based Comparison of Software Libraries, in: 2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER). Presented at the 2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER), pp. 37–40.

Nitze, A., Schmietendorf, A., 2014. MODULARITY OF JAVASCRIPT LIBRARIES AND FRAMEWORKS IN MODERN WEB APPLICATIONS 4.

Ocariza, F.S., Pattabiraman, K., Mesbah, A., 2015. Detecting Inconsistencies in JavaScript MVC Applications, in: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. Presented at the 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE), IEEE, Florence, Italy, pp. 325–335.

Pano, A., Graziotin, D., Abrahamsson, P., 2018. Factors and actors leading to the adoption of a JavaScript framework. Empir. Softw. Eng. 23, 3503–3534.

Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic Mapping Studies in Software Engineering, in: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, EASE'08. BCS Learning & Development Ltd., Swindon, UK, pp. 68–77.

Petersen, K., Vakkalanka, S., Kuzniarz, L., 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. Inf. Softw. Technol. 64, 1–18.

Prokofyeva, N., Boltunova, V., 2017. Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems. Procedia Comput. Sci. 104, 51–56.

Raigoza, J., Thakkar, R., 2016. Browser Performance of JavaScript Framework, SAPUI5 amp;amp; jQuery, in: 2016 International Conference on Computational Science and Computational Intelligence (CSCI). Presented at the 2016 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 1420–1421.

Ramos, M., Valente, M.T., Terra, R., 2018. AngularJS Performance: A Survey Study. IEEE Softw. 35, 72–79.

Ramos, M., Valente, M.T., Terra, R., Santos, G., 2016. AngularJS in the Wild: A Survey with 460 Developers, in: Proceedings of the 7th International Workshop on Evaluation and Usability of Programming Languages and Tools, PLATEAU 2016. ACM, New York, NY, USA, pp. 9–16.

Razak, N.A., Ghazali, M., 2011. Usability in software development: Frameworks comparison between IKnowU and user behavior analysis framework (UBAF), in: 2011 Malaysian Conference in Software Engineering. Presented at the 2011 Malaysian Conference in Software Engineering, pp. 330–335.

Salas-Zárate, M. del P., Alor-Hernández, G., Valencia-García, R., Rodríguez-Mazahua, L., Rodríguez-González, A., López Cuadrado, J.L., 2015. Analyzing best practices on Web development frameworks: The lift approach. Sci. Comput. Program. 102, 1–19.

Samkari, K., Joukhadar, A., 2008. Comparison Matrix for Web HCI, in: 2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications. Presented at the 2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications, pp. 1–5.

Shan, T.C., Hua, W.W., 2006. Taxonomy of Java Web Application Frameworks, in: 2006 IEEE International Conference on E-Business Engineering (ICEBE'06). Presented at the 2006 IEEE International Conference on e-Business Engineering (ICEBE'06), pp. 378–385.

Sommer, A., Krusche, S., 2013. Evaluation of cross-platform frameworks for mobile applications 14.

Sultan, M., 2017. Angular and the Trending Frameworks of Mobile and Web-based Platform Technologies: A Comparative Analysis 9.

Tekinerdogan, B., 2004. ASAAM: aspectual software architecture analysis method, in: Proceedings. Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA 2004). Presented at the Proceedings. Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA 2004), pp. 5–14.

van der Geest, J., Ettema, M., 2011. Comparison of JavaScript libraries. Rijksuniversiteit Groningen. Universiteitsbibliotheek.

Wieringa, R., Maiden, N., Mead, N., Rolland, C., 2006. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requir. Eng. 11, 102–107.

LIITE 1. Kriteerit ja arviointitavat

Kriteeri	Mittari
Tehokkuus	tiedoston koko (Kähkönen, 2014; Pano et al., 2018; Sommer & Krusche, 2013) Tarkkailijoiden toteutus (Kähkönen, 2014), Caching (Malmström, 2014; Molin, 2016), Benchmark (Dhillon & Mahmoud, 2015; Ferreira, 2018; Gizas et al., 2012; Graziotin & Abrahamsson, 2013; Malmström, 2014; Mariano, 2017; Molin, 2016; Raigoza & Thakkar, 2016)
Yhteisö	GitHub yms. statistiikka (Ferreira, 2018; Graziotin & Abrahamsson, 2013; Jain et al., 2014; Majchrzak et al., 2017; Sultan, 2017; van der Geest & Ettema, 2011)
Suosio	Sama kuin yhteisö
Opittavuus	Oppimisen nopeus (Heitkötter et al., 2013; Pano et al., 2018; Samkari & Joukhadar, 2008), Helppous (Malmström, 2014)
Siirrettävyys	Sidoksellisuus (Kähkönen, 2014), Alustatuki (Laine et al., 2011; Molin, 2016),Selaintuki(Molin, Malmström), JavaScriptin versio (Molin, 2016) , Kuinka helposti kehyksen voi vaihtaa toiseen(Mora & Nadi, 2018)
Tuki	Kehittäjä (Heitkötter et al., 2013; Majchrzak et al., 2017), Päivitystiheys (Heitkötter et al., 2013; Mora & Nadi, 2018; Pano et al., 2018; Prokofyeva & Boltunova, 2017)
Laajennettavuus	Kuinka helposti uusia ominaisuuksia lisätään (Heitkötter et al., 2013; Kähkönen, 2014; Pano et al., 2018; Salas-Zárate et al., 2015).
Testattavuus	Testaustuki/työkalut (Ignacio Fernández-Villamor et al., 2008; Malmström, 2014; Molin, 2016; Salas-Zárate et al., 2015; Sommer & Krusche, 2013)

(Jatkuu)

Liite 1 (Jatkoa)

Koko	Kehyksen pakkaus (Delcev & Draskovic, 2018; Ferreira, 2018; Gizas et al., 2012; Heitkötter et al., 2013; Kähkönen, 2014; Kaluža et al., 2018; Laine et al., 2011; Pano et al., 2018; Sommer & Krusche, 2013; van der Geest & Ettema, 2011).
Dokumentaatio	Dokumentaation laajuus (Delcev & Draskovic, 2018; Graziotin & Abrahamsson, 2013; Heitkötter et al., 2013; Malmström, 2014; Sommer & Krusche, 2013), Dokumentaation olemassa olo (Molin, 2016; Salas-Zárate et al., 2015)
Tietoturva	Testit (Delcev & Draskovic, 2018; Ocariza et al., 2015), Ominaisuudet (Ignacio Fernández-Villamor et al., 2008; Laine et al., 2011; Molin, 2016; Salas-Zárate et al., 2015; Samkari & Joukhadar, 2008)
Modulaarisuus	Seperation of concerns (Heitkötter et al., 2013; Malmström, 2014; Molin, 2016; Nitze & Schmietendorf, 2014; Pano et al., 2018).
Hinta	Hinta/lisenssi (Heitkötter et al., 2013; Pano et al., 2018; Sommer & Krusche, 2013)
Laatu	Erilaiset laatumittarit esim. LOC avulla (Gizas et al., 2012; Graziotin & Abrahamsson, 2013; Misra & Cafer, 2012)
Ylläpidettävyys	Laatumittarit (Gizas et al., 2012; Molin, 2016), Koodin ymmärrettävyys(Heitkötter et al., 2013), Sovellusarkkitehtuuri (Kähkönen, 2014) Kehittäjä ja käyttäjät (Molin, 2016)
Kypsyys	Käyttäjä määrä (Ignacio Fernández-Villamor et al., 2008), kuka käyttää suuryritykset? (Molin, 2016)

(Jatkuu)

Liite 1 (Jatkoa)

Selaintuki	Mitä selaimia kehys tukee (Delcev & Draskovic, 2018; Malmström, 2014; Molin, 2016; van der Geest & Ettema, 2011).
Yhteensopivuus	yhteensopivuus vanhojen versioiden kanssa (Mora & Nadi, 2018)
Luotettavuus	vakaus ja koko (Sommer & Krusche, 2013)
Luettavuus	Koodin luettavuus (Kähkönen, 2014)

Liite 2 Aineiston kategorisointi

Seuraavassa taulukossa on esitetty työtä varten kootut julkaisut ja niiden kategorisointi. Taulukko sisältää myös tarkasteluvaiheessa hylätyt lähteet.

Lähde	**Vuosi**	**Julkaisu**	**Näkökulma**	**Tyyppi**	**Kriteerit**
Delcev & Draskovic	2018	2018 Zooming Innovation in Consumer Technologies Conference (ZINC)	Ratkaisu	Käytäntö	Koko, Suosio, Dokumentaatio, Opittavuus, Selaintuki
Dhillon & Mahmound	2015	Software: Practice and Experience	Ratkaisu	Käytäntö	Tehokkuus
Ferreira	2018	-	Validaatio	Käytäntö	Tehokkuus, Yhteisö
Gizas et al.	2012	Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion	Ratkaisu	Käytäntö	Laatu, Validointi, Tehokkuus

(Jatkuu)

Liite 2 (Jatkoa)

Graziotin, & Abrahamsson	2013	Proceedings of the 14th International Conference on Product-Focused Software Process Improvement (PROFES 2013), LNCS 7983, Springer-Verlag, pp. 334-337, 2013	Ratkaisu	Teoreettinen	Validointi, Laatu, Tehokkuus, Dokumentaatio, Yhteisö, Käytäntö
Heitkötter et al.	2013	Proceedings of the 9th International Conference on Web Information Systems and Technologies. 209-221.	Ratkaisu	Teoreettinen	Hinta, Suosio, Opittavuus, Laajennettavuus, Ylläpidettävyys, Tehokkuus

(Jatkuu)

Liite 2 (Jatkoa)

Ignacio-Fernandez-Villamor	2008	Proceedings of the 2008 Euro American Conference on Telematics and Information Systems	Ratkaisu	Teoreettinen	Tietoturva, Testattavuus, Yhteisö
Jain et al	2014	Journal of Global Research in Computer Science	Filo	NAN	Yhteisö
Kaluza et al.	2018	Zbornik Veleučilišta u Rijeci	Ratkaisu	Teoreettinen	Ominaisuudet
Kähkönen	2014	-	Ratkaisu	Teoreettinen	Sidoksellisuus, Luettavuus, Tehokkuus, Siirrettävyys, Ylläpidettävyys, Laajennettavuus, Koko

(Jatkuu)

Liite 2 (Jatkoa)

Laine et al.	2011	IT Professional	Filo	NAN	Luotettavuus, Tietoturva, Yhteensopivuus, Siirrettävyys, Tuki
Madsen et al	2013	Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2013	-Arvio	-Käytäntö	
Majchrzak et al	2017	Proceedings of the 50th Hawaii International Conference on System Sciences | 2017	ratkaisu	Teoreettinen	Yhteisö, suosio
Malmström	2014	-	Ratkaisu	Käytäntö	Kypsyys, Tehokkuus, Siirrettävyys, Testattavuus, Modulaarisuus, Suosio, Opittavuus, Dokumentaatio

(Jatkuu)

Liite 2 (Jatkoa)

Mariano	2017	-	Arvio	Käytäntö	Laatu
Misra, Cafer	2012	International Arab Journal of Information Technology 9(6):535-543 · November 2012	Validaatio	Käytäntö	Laatu, JCCM
Molin	2016	-	Ratkaisu	Käytäntö	Tietoturva, Modulaarisuus, Suosio, Kypsyys, Käytettävyys, Siirrettävyys, Yhteensopivuus, Tehokkuus, Testattavuus, Ylläpidettävyys

(Jatkuu)

Liite 2 (Jatkoa)

Mora, Nadi	2018	2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER)	Ratkaisu	Teoreettinen	Suosio, Siirrettävyys, Tietoturva, Tehokkuus
Nitze, Schmietendorf	2014		Filo	Teoreettinen	Modulaarisuus
Ocariza et al.	2015	2015 IEEE/ACM 37th IEEE International Conference on Software Engineering	Validaatio	Käytäntö	Tietoturva
Pano et al.	2018	Empirical Software Engineering	Ratkaisu	Teoreettinen	Tehokkuus, Koko, Opittavuus, Ymmärrettävyys, Yhteisö, Hinta
Prokofyeva, Boltunova	2017	Procedia Computer Science	Ratkaisu	Käytäntö	Hinta, Tehokkuus

(Jatkuu)

Liite 2 (Jatkoa)

Raigoza, Thakkar	2016	2016 International Conference on Computational Science and Computational Intelligence (CSCI)	Ratkaisu	Käytäntö	Tehokkuus
Ramos et al (2018)	2018	IEEE Software (Volume: 35 , Issue: 2 , March/April 2018)	Filo	Teoreettinen	Suosio
Ramos et al (2016)	2016	Proceedings of the 7th International Workshop on Evaluation and Usability of Programming Languages and Tools	Filo	Teoreettinen	Tuki
Razak, Ghazali	2011	2011 Malaysian Conference in Software Engineering	Ratkaisu	Teoreettinen	Käytettävyys

(Jatkuu)

Liite 2 (Jatkoa)

Salas-Zarate et al.	2015	Science of Computer Programming	Ratkaisu	Teoreettinen	Dokumentaatio, Laajennettavuus, Testattavuus
Samkari, Joukhadar	2008	2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications	Ratkaisu	Teoreettinen	Opittavuus, Turvallisuus, Kypsyys, Laajennettavuus
Shan, Hua	2006	2006 IEEE International Conference on e-Business Engineering (ICEBE'06)	Filo	Teoreettinen	Tuki
Sommer, Krusche	2013	Proceedings - Series of the Gesellschaft fur Informatik (GI).	Ratkaisu	Teoreettinen	Käytettävyys Dokumentaatio, Opittavuus, Koko, Tehokkuus, Hinta

(Jatkuu)

Liite 2 (Jatkoa)

Sultan	2017	Future Technologies Conference (FTC) 2017 29-30 November 2017| Vancouver, Canada	Ratkaisu	Teoreettinen	Laajennettavuus, Testattavuus, Modulaarisuus, Yhteisö
Sun, Ryu	2017	ACM Computing Surveys	-Filo	-AN	
van der Geest, Ettema	2011	Rijksuniversiteit Groningen. Universiteitsbibliotheek	Ratkaisu	Teoreettinen	Selaintuki, Yhteisö, Tehokkuus, Koko