TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Erik Räni

JavaScripti raamistike võrdlus

Bakalaureusetöö (9 EAP)

Juhendaja: Vambola Leping, MSc

Tartu 2016

JavaScripti raamistike võrdlus

Lühikokkuvõte:

Käesoleva bakalaureusetöö eesmärk on luua süstemaatiline võrdlev käsitlus JavaScripti raamistikest AngularJS, Backbone.js, React, Ember.js, KnockoutJS ja Dojo. Autor hindab neid vastavalt töös toodud reeglitele. Üks oluline osa võrdlusprotsessis on raamistikega tutvumine, TodoMVC projektist pärit testrakenduste lähtekoodi mõistmine ning kirjalikul kujul selgitamine. Töö tulemusena valmivad hinnangutel põhinev edetabel ning tekstimaterjal, mis võimaldab lugejal mõista kaasatud raamistikega seonduvaid probleeme ja teha enda jaoks sobivaim valik.

Võtmesõnad:

JavaScript, MVC, MV*, veebirakendus

CERCS:

P175 (Informaatika, süsteemiteooria)

Comparison of JavaScript frameworks

Abstract:

The aim of this Bachelor's thesis is to research JavaScript frameworks AngularJS, Backbone.js, React, Ember.js, KnockoutJS and Dojo in a systematic and comparative way. The author evaluates these frameworks according to the rules defined in this thesis. One important part of the evaluation process is understanding and explaining the source code of applications from project TodoMVC. In conclusion, this work offers the reader an opportunity to understand potential problems of given frameworks and helps to make the best choice.

Keywords:

JavaScript, MVC, MV*, web application

CERCS:

P175 (Informatics, systems theory)

Sisukord

Sissejuhatus

2015. aasta suvel küsis bakalaureusetöö autor Eesti mobiili- ja veebiarendusega tegeleva tarkvarafirma Mooncascade OÜ personalijuhilt veebiarendaja rolli ning firma veebiarenduses kasutatavate tehnoloogiate kohta. Katkend vastusest oli järgmine: „Veebirakenduste (ning neid toetavate teenuste) arendamine toimub vastavalt ülesandepüstitusele kasutades olukorras sobivaid tööriistu. Tööpõld on suhteliselt lai, kuna meil on üpriski erinev tehnoloogia *stack*, kuid enamjaolt on märksõnadeks Angular, React, Backbone, PHP, Laravel, MySQL.“

Aktiivselt kasvava tarkvaraettevõtte veebiarendust iseloomustavast kuuest märksõnast olid pooled JavaScripti raamistike või teekide nimed. Veebirakenduse loomisel aitavad raamistikud luua struktuuri ja suurendada tõenäosust, et kood on ka tulevikus kergemini hallatav [1]. Võimaluste arv kasvab [2]. Arendajad peavad aru saama, mis nende vajadusi parimal viisil rahuldab. Otsuse tegemise abistamiseks loodi TodoMVC projekt, milles on kindla funktsionaalsusega tegevuste haldamise testrakendus implementeeritud erinevate JavaScripti MV* raamistike abil[1] [2].

Käesoleva bakalaureusetöö eesmärk on luua süstemaatiline võrdlev käsitlus JavaScripti raamistikest AngularJS, Backbone.js, React, Ember.js, KnockoutJS ja Dojo. Nende hindamine toimub vastavalt kindlatele reeglitele. Autor teadvustab, et taolist tegevust võib nimetada mingil määral subjektiivseks. Selle tõttu esitatakse enne hinnangute andmist ka põhjendused, mis võimaldavad lugejal enda seisukohta kujundada. Üks oluline osa võrdlusprotsessis on raamistikega tutvumine, TodoMVC projektist pärit testrakenduste lähtekoodi mõistmine ning kirjalikul kujul selgitamine [2]. Töö tulemusena valmivad hinnangutel põhinev edetabel ning tekstimaterjal, mis võimaldab lugejal mõista kaasatud raamistikega seonduvaid probleeme ja teha enda jaoks sobivaim valik.

Esimese peatüki alguses esitatakse taustinformatsioon JavaScripti ning raamistike kohta. Järgnevalt fikseeritakse hindamiskriteeriumid: startimishinnang, kogukond ja dokumentatsioon, arendustööriistade tugi, testimistugi, litsents äri vaatepunktist, keele ja vormingu tugi, TodoMVC testrakendusel põhinev keerukus, TodoMVC testrakendusel põhinev õppimine ja õppimiskiirus ning TodoMVC testrakendusel põhinev jõudlus.

Teises peatükis sooritatakse võrdlus koos hinnangute andmisega. Lõpuks tehakse kokkuvõte. Lisades esitatakse keerukuse pingeread, lühimaterjal ühe testrakenduse juures katsetatud serveri ja Deployd API kohta, TodoMVC projekti repositooriumi sisu kasutamise litsents ning lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks.

Hindamiskriteeriumite fikseerimise osas viidatakse ideedele, mis on saadud mitmete teiste autorite poolt varem läbi viidud võrdlustest. Käesolev töö on uudne testrakenduse projekti huvitava kasutusviisi ja põhjaliku lähenemise poolest.

[1] Peatükis 1.1.1 selgitatakse, mida tähendavad MVC ja MV*.

1. Ülevaade

Tähelepanelik lugeja märkab, et töö pealkiri on „JavaScripti raamistike võrdlus“, kuid mõnda edaspidi võrreldavat tarkvara nimetatakse isegi selle ametlikul kodulehel teegiks (*library*), mitte raamistikuks (*framework*). Neid termineid on tarkvaraarenduse teemal raamatuid avaldanud [3] Martin Fowler defineerinud järgnevalt toodud viisil [4]:

- teek on hulk funktsioone, mida saab vajadusel välja kutsuda;
- raamistik on abstraktse disaini kehastus, millesse sisestatakse konkreetsetesse kohtadesse raamistiku koodi poolt kasutatavat arendaja loodud koodi.

TodoMVC projekti tutvustussõnad on „Aidates Teil valida MV* raamistikku“ ehk mainitud terminoloogiat ei rakendata uurimisobjektidele rangel viisil [2]. Samast mitteranguse põhimõttest lähtuti ka bakalaureusetöö pealkirja ning sisu loomisel.

1.1 JavaScript[2] ja raamistikud

1995. aastal lõi Brendan Eich JavaScripti keele, mis tehti avalikuks aastal 1996 [5]. Selle alusstandardiks on tänapäeval Ecma International standardite organisatsiooni poolt standardiseeritav JavaScriptist hiljem loodud skriptimiskeel ECMAScript [6] [7]. 2015. aasta juunis ilmus ECMAScripti kuues versioon. Analoogiliselt Mozilla materjalidele kasutab autor sarnasuse tõttu edaspidi ECMAScriptist või JavaScriptist kirjutades nime JavaScript [5]. JavaScript toetab programmeerimist objektorienteeritult, imperatiivselt ja protseduraalselt [8]. Bakalaureusetöö autor ei asu käesolevat keelt kirjeldama, vaid eeldab, et lugejal on olemas kogemus JavaScripti või mõne muu kodeerimiskeelega.

Käesolevas lõigus tutvustatakse James Sugrue raamatu materjali põhjal oluliste sündmuste ahelat, mis viis MVC ja MV* arhitektuurimustri kasutamise vajaduseni JavaScripti rakendustes [9, lk 2–3]. Ajax ehk asünkroonne JavaScript ja XML muutus populaarseks 2005. aastal. Seda kasutatakse tavaliselt serveriga suhtlemise abil veebirakenduse lehe alamosa dünaamiliseks uuendamiseks ehk tervet lehte pole tarvis uuesti laadida. Ajax-päringud tehakse RESTful teenuste abil, milles lühend REST (*Representational State Transfer*) tähistab üle HTTP toimuva kliendi ja serveri vahelise suhtluse arhitektuuri. 2006. aastast anti välja raamistik jQuery, mis võimaldab arendajatel dokumendiobjektide mudeliga (DOM - *Document Object Model*) efektiivsemalt manipuleerida ning Ajaxit kasutavate rakenduste koodi kergemini luua. Lihtsustus üheleheveebirakenduste (*single-page web applications*, ka SPA) implementeerimine. SPA puhul kuvatakse kasutajale esitluskiht veebilehitsejas ainult ühe laadimiskorraga ning edaspidi pole uuestilaadimist tarvis: vajalikud lisaelemendid tuuakse sinna dünaamiliselt. Sellised tarkvaralahendused toovad kliendi poolele eesrakenduses (*front end*) palju koodi ning loogikat, mille efektiivseks haldamiseks läheb vaja korralikku andmemudelit koos disainimustriga (*design pattern*). Probleemi lahendamiseks on JavaScripti maailma jõudnud MVC ja MV* arhitektuurimustrid ja mitmed nende mustrite põhimõtete rakendamist parendavad raamistikud. Hea üheleheveebirakenduse näide on Gmail.

1.1.1 MVC ja MV*

Mudel-vaade-kontroller (*Model-View-Controller* ehk MVC) termin pärineb 1970. aastatest graafiliste kasutajaliidestega rakendustega tegelenud Smalltalki projektist [10, lk 47]. Sealt liikus käesoleva mõiste taga olev printsiip esmalt tagarakenduste arendamisse (*back end*

[2] JavaScript® on firma Oracle kaubamärk või registreeritud kaubamärk USA-s ja teistes riikides [8].

või *server-side end development*) ning on jõudnud viimaste aastatega keerulisemaks muutuvate eesrakenduste arendamisse (*front end* või *client-side end development*) [10, lk 47]. MVC raamistikud jagavad rakenduse kolmeks funktsionaalseks osaks: mudeliks, vaateks ja kontrolleriks, kus mudeliks võivad olla kas ainult andmed, mida vaates esitatakse, või andmed koos loogikaga, näiteks oma operatsioonidega või manipuleerimisreeglitega [10, lk 49]. Mudeli objektidest võib mõelda kui päris maailma olemite andmeesitusest, kus vaade on mudeli esitluskiht, mis on valmis end muutma mudelis toimuvate muutuste tagajärjel [9, lk 4]. Vaade mudelit ise ei modifitseeri, selleks on olemas kontroller, mis kasutab vaadet nägeva kasutaja poolt algatatud mudeli muutmise sisendit ning uuendab vastavalt mudeli seisundit [9, lk 4].

MV* tähistab mustrit, milles on võib-olla muudetud kontrolleri printsiipi. Kaks olulist ideed on mudel-vaade-esitleja (*Model-View-Presenter* ehk MVP) ja mudel-vaade-vaatemudel (*Model-View-ViewModel* ehk MVVM) [11]. Esimese puhul saavutatakse suurem osade eraldatus: vaatega kommunikeerimine toimub ainult läbi liidese (*interface*), suhtlust vaate ja mudeli vahel juhib esitleja (*presenter*) [9, lk 5–6]. Testimine ja komponentite teineteisest sõltumatu arendus muutub kergemaks [9, lk 6]. Teisel juhul vastutab vaatemudel vaatesse jõudvate andmete eest: vaated seovad end mingit mudeli alamosa paljastavate vaatemudeli atribuutidega [9, lk 6] [11].

1.1.2 Võrdlusest mujal

Bakalaureusetööle eelneval ajal ning ka selle tegemise ajal on võimalik leida palju veebimaterjale, kus võrreldakse JavaScripti raamistikke. Google'i otsing andis 2016. aasta veebruaris fraasile „*javascript framework comparison*" vasteks umbes 16 600 000 tulemust. Mitmed allikad kujutavad endast postitust blogisse või veebilehele, leidub ka slaidimaterjale ning muus vormis allikaid.

Iisraeli Tehnoloogiainstituudiga seotud Uri Shaked võrdles raamistikke AngularJS, Backbone.js ning Ember.js [12]. Ta jõudis tulemuseni, et Ember.js esindab hästi MVC mustrit, Backbone mõjub positiivselt minimalistlikult ja Angular suudab innovatiivselt laiendada HMTL-koodi ning pakub tuge oma suure kogukonnaga [12]. Microsofti sertifitseeritud lahenduste arendaja (*Microsoft Certified Solutions Developer* ehk MCSD) Craig McKeachie on kirjutanud raamatu „*JavaScript Framework Guide AngularJS, Backbone, Ember*", milles võrdleb raamistikke kindlate omaduste põhjal, näiteks marsruutimine (*routing*) ja andmesidumine (*data-binding*) koos koodinäidetega [13] [14].

Firma Crytek vanem veebiarendaja Dmitry Sheiko väljendas oma ettekandes „*JavaScript MV* Framework - Making the Right Choice*" arvamust, et kolm olulist raamistikku on AngularJS, Backbone ja Ember.js [15]. Ta juhtis slaididel tähelepanu MV* mustri realiseerimisele, kogukonna aktiivsusele ja keerukusele ning jõudis tõdemuseni, et tema meelest on Backbone parim lahendus.

Üle 13 aasta tarkvaraarenduses töötanud James Sugrue kirjutas oma raamatu „*Beginning Backbone.js*" sissejuhatavas osas raamistikest Knockout.js, AngularJS ja Ember [9, lk 7–10]. Ta tõi välja, et Knockout.js MVVM mustri kasutamise abil saab teha automaatseid kasutajaliidese uuendusi [9, lk 7–8]. Keerulisemate andmemudelite puhul eelistab Sugrue aga Backbone'i poolt pakutavaid võimalusi RESTful teenustega suhtluseks [9, lk 8]. AngularJS võib arendajatele suruda peale oma stiili [9, lk 9]. Emberil on sarnasusi Backbone'iga [9, lk 10].

Veebiarhitektuurikonsultant Matt Raible võrdles 2015. aasta sügisel lihtsal kujul slaididel Angulari, Emberit ja Reacti järgmistes Yevgeniy Brikmani poolt välja mõeldud kategooriates: õppimine, arendamine, testimine, turvalisus, ehitusprotsess (*build*), paigaldus

(*deploy*), silumine (*debug*), skaleeritavus, hooldatavus (*maintainability*) ning jagamine (*share*) [16]. Käesoleva lõigu ülejäänud tekst põhineb Matt Raible'i slaididel [16]. Arendamise kategoorias vaadeldi näiteks marsruutimist (*routing*), mallide (*templates*) kasutamise võimalust, andmetesidumisvõimalust (*data-binding*), TodoMVC projekti testrakenduse loogiliste ridade ja failide arvu. Ehitusprotsessi kategoorias vaadeldi milliseid tööriistu kasutada saab: Angulari ja Emberi puhul näiteks töötavad Bower ja raamistiku Node.js paketihalduri npm, Reacti puhul Babel [17]. Jagamise kategooria all vaadeldi ka tööpakkumisi ja tööle kandideerijate arvu raamistike kaupa keskkonnas LinkedIn: mõlemal juhul oli Angular ülejäänud kahest vaadeldavast võrdlusobjektist suurema arvulise näitajaga. Lõpuks võitis Ember ühepunktilise eduga Angulari ja neljapunktilise vahega Reacti ees. Järgmisel slaidil aga tõmmati tulemuste peale punane mahakriipsutamist tähistav rist põhjendusega, et see kõik oli slaidide autori arvamus.

2013. aastal võrdles Mati Kärner oma bakalaureusetöös „Avatud lähtekoodiga veebipõhiste kasutajaliideste raamistike võrdlus" raamistikke Bootstrap, Dojo, YUI, SproutCore, Qooxdoo ja Enyo [18, lk 6]. Hindamiskriteeriumid olid õppimiskiirus, veebilehitseja standarditele orienteeritus, lokaliseerimine, sisendi valideerimine, integreeritud arenduskeskkond (IDE), kasutajatoe elemendid, kogukonna aktiivsus, testimisvahendid, litsentsi tingimused ja jõudlus [18, lk 7–8]. Parima hinnangu ehk 97,5/100 saavutas Dojo, mis kuulub ka käesolevas bakalaureusetöös võrreldavate raamistike hulka [18, lk 36]. Kärner kulutas Dojo testrakenduse loomisele optimaalseks peetud viie tunni asemel üheksa tundi ning kinnitas suurepärast sobivust veebilehitsejate ja standarditega [18, lk 17–18]. Lokaliseerimisega, sisendi valideerimisega, IDE-dega, kasutajatoe elementidega, kogukonna aktiivsusega, testimisvahendide leidumisega ja litsentsiga seotud hinnangud olid maksimaalsed [18, lk 18–20]. Jõudluse hinne oli maksimumilähedane [18, lk 20]. Käesoleva bakalaureusetöö mõned hindamiskriteeriumite ideed pärinevad Kärneri tööst. Iga kriteeriumi juurde on lisatud viited ideede allikatele. Osa informatsiooni Dojo kohta, mis sai esitatud juba bakalaureusetöös „Avatud lähtekoodiga veebipõhiste kasutajaliideste raamistike võrdlus" võib taasesineda ka „JavaScripti raamistike võrdluse" vastavas peatükis. Kärner keskendus kasutajaliideste raamistikele [18, lk 4], käesoleva töö autor keskendub aga MV* raamistikele.

Mitmetes allikates on tehtud raamistike ja võrdlemiskriteeriumite puhul mingi valik. Slaidide ja blogide puhul on valitud pigem mõned tuntud raamistikud ja vähese tekstiga idee edasi antud. Käesolev bakalaureusetöö on autori poolt muudetud kasulikuks kuue raamistiku valikuga, põhjaliku tekstilise osaga ja testrakenduse projekti huvitava kasutusviisiga. Peatükis 1.4 on viidatud materjalidele, millest pärinevad hindamiskriteeriumite ideed.

1.2 TodoMVC testrakendus

TodoMVC projekti MIT litsentsis on kirjas eestvedajate nimed: Addy Osmani, Sindre Sorhus, Pascal Hartig ja Stephen Sawchuk (lisa „III. TodoMVC projekti repositooriumi sisu kasutamise litsents") [19].

Iga loodav tegevuste haldamise testrakendus peaks järgima võimalikult suurel määral juhendmaterjali [20]. Rakenduse välisstruktuuri moodustavad failid index.html, package.json ja readme.md ning kaustad `css` ja `js`. Eelistatud on raamistiku Node.js paketihalduri npm kasutamine ning sõltuvuste (*dependencies*) kirjutamine faili package.json [17]. Uus rakendus tuleb üles ehitada etteantud mallrakenduse (*template application*) põhjal ning sisaldab seega kindlaid faile, näiteks base.css ja app.css [20].

Alltoodud loetelus on kirjas juhendmaterjalil tuginev lühiülevaade testrakenduse funktsionaalsusest, kust on jäetud välja näiteks CSS-klasside lisamise või eemaldamise nõuete info [20].

- Kasutaja saab sisestada ekraani ülaosas paiknevasse tekstikasti uue tegevuse pealkirja.
    - Klahvi Enter vajutamisel lisatakse tekstikastist saadud pealkirjaga objekt tegevuste loetellu ja puhastatakse tekstikast tekstist.
- Tegevuste loetelu ja jalus on nähtavad ainult siis, kui tegevusi eksisteerib.
- Iga tegevuse pealkirja juures peab olema kustutamisnupp, mis on nähtav siis, kui hiirekursor asetseb konkreetse pealkirja piirkonnas.
- Tegevuse pealkirja peab olema võimalik topeltklõpsamise järel redigeerida.
    - Redigeerimise käigus tegevuse pealkirja kustutamine tähendab ka tegevuse loetelust kustutamist.
    - Redigeerimise käigus klahvi Escape vajutamine taastab tegevuse pealkirja redigeerimisele eelnenud kujul.
- Iga tegevuse pealkirja juures peab olema märgendkast, millele klõpsamisega saab tegevust märkida tehtuks või mittetehtuks.
- Kasutajal peab olema võimalik märkida ühest kindlast märgendkastist korraga kõik tegevused tehtuks või mittetehtuks.
    - Kui kasutaja on iga tegevuse pealkirja juurest paikneva märgendkasti ära märkinud, siis peab iseenesest märgituks muutuma ka kõikide tegevuste märkimise märgendkast.
- Rakendus peab veebilehitseja andmehoius (*localstorage*) salvestama tegevuste objekte, millel on atribuutideks identifikaator, pealkiri ja tõeväärtusväli tehtuks või mittetehtuks märkimise kohta [21].
- Rakenduses peab toimima URL-haldus, mis lubab kasutajal filtreerida ja kuvada järgnevate tegevuste loetelusid: tehtuks märgituid, mittetehtuks märgituid või kõiki tegevusi. Filtreerimisnupud asuvad jaluses.
- Kasutaja peab saama eemaldada kõiki tehtuks märgitud tegevusi spetsiaalsest nupust jaluses.
    - Kõiki tehtuks märgitud tegevusi kustutada võimaldav nupp peab olema nähtav vaid selliste tegevuste eksisteerimisel.
- Koos ingliskeelse sõnaga „asi“ (*item*) või „asjad“ (*items*) peab kasutajale olema näha loendur, mis näitab mitu mittetehtuks märgitud tegevust eksisteerib. Rakendus peab valima sõltuvalt loendurist korrektse sõnavormi.

Testrakenduse ekraanivaade on toodud joonisel 1.

Joonis 1. TodoMVC projekti testrakendus

1.3 Võrdlusesse kuuluvad raamistikud

Testrakendustes on kasutatud järgnevaid raamistikke: AngularJS 1.4.9, Backbone.js 1.2.2, React 0.13.3, Ember.js 1.10.0-beta.3, KnockoutJS 3.2.0, Dojo 1.10.2 ning Knockback.js 1.0.0[3]. Käesoleva töö autor otsustas 15 TodoMVC testrakenduse keerukust hinnata ja kindlaks määrata Angulari, Backbone'i, Reacti, Emberi, Knockouti ja Dojo testrakenduste asukohad keerukuse pingeridades (lisa „I. Keerukuse pingeread"). Testrakendustest, milles teostati ainult keerukuse hindamine, on kasutatud järgnevaid raamistikke: CanJS, Polymer, Mithril, Ampersand, Flight, Vue.js, MarionetteJS (Backbone.Marionette), TroopJS ning RequireJS[4]. Startimishinnangu kategoorias hinnangu andmisel kasutatakse raamistikest uusimat stabiilset versiooni.

1.4 Reeglid võrdluse teostamiseks

Alltoodud alapunktides antud juhiste järgi saab iga raamistik kas null, ühe, kaks või kolm punkti. Mingi punktisumma saavutamiseks peavad olema täidetud hindamiskriteerimi vastava taseme kõik miinimumnõuded. Autor kujundab hindamiskriteeriumid alltoodud stiilis.

1. Hinde kolm saamise puhul leidub puudujääke väga vähe.
2. Hinde kaks saamise puhul leidub mõningaid puudujääke.
3. Hinde üks saamise puhul leidub palju puudujääke.
4. Hinne null antakse siis, kui vaadeldav omadus on olematu või mittetoimiv.

Maksimaalselt on võimalik saada 27 punkti, minimaalselt aga 2 punkti.

1.4.1 Startimishinnang

Esimeste tegevuste hulgas uue raamistikuga tutvumisel on selle paigaldamine ametlikult kodulehelt arendaja süsteemi ja lihtsa lause väljastamine ekraanile. Oluline on ka motivatsioon, mille arendaja saab teadmisest, et leidub edukaid äriprojekte, milles on raamistikku varem kasutatud. Alltoodud loetelus on kirjas hinnete kolm kuni üks määramise kriteeriumid. Startimishinnangu kategoorias ei omistata võrdluses olevale raamistikule ühelgi juhul hinnet null, sest on teada, et paigaldamisprotsess pole puudulik.

1. Piisab oluliste failide allalaadimisest õigesse kausta ning veebilehitsejas käivitatavale failile viidete loomisest. Lisaks leidub vähemalt kolm vaadeldavat raamistikku kasutavat edukat ärifirmat.
2. Peale failide õigesse kausta allalaadimise ning veebilehitsejas käivitatavale failile viidete loomise tuleb sooritada veel mingi tegevus. Lisaks leidub vähemalt üks vaadeldavat raamistikku kasutav edukas ärifirma.
3. Peale failide õigesse kausta allalaadimise ning veebilehitsejas käivitatavale failile viidete loomise tuleb sooritada veel mitu tegevust. Puuduvad vaadeldavat raamistikku kasutavad ärifirmad.

Käesoleva kriteeriumi kasutamise idee sai bakalaureusetöö autor kahest allikast [37, slaid 9] [9, lk 11–15].

1.4.2 Kogukond ja dokumentatsioon

Kogukond ja dokumentatsioon on olulised, sest raamistiku kohta vähe infot omav arendaja võib selle kasutamisega hätta jääda. Võib-olla saab lahenduse leida dokumentatsioonist,

[3] [22] [23] [24] [25] [26] [27]
[4] [28] [29] [30] [31] [32] [33] [34] [35] [36]

kuid mõnikord on abi kogenuma inimese poole pöördumisest. Kogukond on arendaja jaoks tugivõrgustik, mille muudab tugevaks suur arv aktiivseid liikmeid. Hinnangute andmisel arvestatakse Githubist, Stack Exchange'ist ja raamistiku kodulehelt pärinevat infot. Alltoodud loetelus on kirjas hinnete kolm kuni null määramise kriteeriumid.

1. Githubis olevate vaatajate, tähega märkijate ning koodi harutajate (inglise keeles on koodi harutamine *forking*) summa on vähemalt 3000. Stack Exchange'is on mittevastatuks märgitud küsimused moodustamas kõige rohkem 40% kõigist raamistiku nime kandva sildiga küsimustest. Lisaks eksisteerib koodinäiteid sisaldav dokumentatsioon, mida arendajal on võimalik muuta või teha ettepanekuid muudatusteks.
2. Githubis olevate vaatajate, tähega märkijate ning koodi harutajate summa on vähemalt 2000. Stack Exchange'is on mittevastatuks märgitud küsimused moodustamas kõige rohkem 50% kõigist raamistiku nime kandva sildiga küsimustest. Dokumentatsioon eksisteerib, kuid seal puuduvad koodinäiteid või pole arendajal võimalik seda muuta või soovitada muudatusi.
3. Githubis olevate vaatajate, tähega märkijate ning koodi harutajate summa on vähemalt 1000. Stack Exchange'is on mittevastatuks märgitud küsimused moodustamas kõige rohkem 60% kõigist raamistiku nime kandva sildiga küsimustest. Dokumentatsioon eksisteerib, seal puuduvad koodinäiteid ning arendajal pole võimalik seda muuta või soovitada muudatusi.
4. Githubis olevate vaatajate, tähega märkijate ning koodi harutajate summa on väiksem kui 1000. Stack Exchange'is on mittevastatuks märgitud küsimused moodustamas rohkem kui 60% kõigist raamistiku nime kandva sildiga küsimustest. Dokumentatsioon puudub.

Käesoleva kriteeriumi kasutamise idee sai bakalaureusetöö autor kahest allikast [15, slaidid 7–8] [18, lk 9].

1.4.3 Arendustööriistade tugi

Arendajale on abiks integreeritud programmeerimiskeskkonnale või tekstiredaktorile leiduv vastava raamistikuga seotud lisafunktsionaalsust võimaldav laiendus või pistikprogramm. Hinnangu andmisel kasutatakses laienduste veebilehti toodetele Eclipse, Jetbrains PhpStorm/WebStorm, Sublime Text, NetBeans ja Visual Studio[5]. Jetbrainsi kahte programmeerimiskeskkonda vaadatakse ühe tervikuna. Alltoodud loetelus on kirjas hinnete kolm kuni null määramise kriteeriumid.

1. Neli arendustööriista on toetatud.
2. Vähemalt kaks arendustööriista on toetatud.
3. Toetatud on üks arendustööriist.
4. Arendustööriistade Eclipse, Jetbrains PhpStorm/ WebStorm, Sublime Text, NetBeans ja Visual Studio jaoks puudub lisafunktsionaalsust võimaldav laiendus või pistikprogramm.

Käesoleva kriteeriumi kasutamise idee sai bakalaureusetöö autor ühest allikast [18, lk 9].

1.4.4 Testimistugi

Suuremate JavaScripti rakenduste arendamise puhul on testimine ja testimisest lähtuva arenduse (*Test-Driven Development*) põhimõtete rakendamine oluline: pikemas perspektiivis võib kuluda vähem aega vigadega tegelemise peale, arendajad on sunnitud MV*

[5] [38] [39] [40] [41] [42]

paradigmast testimise hõlbustamiseks kinni pidama, uued arendajad saavad testide lugemise kaudu tarkvarast hästi aru [9, lk 151 – 153]. Hinnangute andmisel lähtutakse ühiktestide (*unit tests*) ning algaja arendaja vaatepunktist: testide kirjutamise info ning näited on alustamiseks tähtsad. Alltoodud loetelus on kirjas hinnete kolm kuni üks määramise kriteeriumid. Testimistoe kategoorias ei omistata võrdluses olevale raamistikule ühelgi juhul hinnet null, sest on teada, et testimist seletavate materjalide seisukord pole puudulik.

1. Leidub vastava raamistiku abil loodud rakenduse testimist seletav materjal koos koodinäidistega.
2. Leidub vastava raamistiku abil loodud rakenduse testimist seletav materjal, koodinäidised puuduvad.
3. Leidub vastava raamistiku abil loodud rakenduse testimisega seotud vihjetega materjal, koodinäidised puuduvad ning tekst on pinnapealne.

Käesoleva kriteeriumi kasutamise idee sai bakalaureusetöö autor kahest allikast [9, lk 151 – 153] [18, lk 10].

1.4.5 Litsents äri vaatepunktist

Raamistikud saavad litsentsidega seotud hinnangu sõltuvalt äri vaatepunktist. Hinde kolm teenimiseks peab saama raamistikuga loodud rakendust kasutada äriks ning rakenduse lähtekood võib jääda salajaseks. Selles kategoorias ei omistata võrdluses olevale raamistikule ühelgi juhul hindeid üks või kaks. Äritegevuse keelu või märkimisväärse piirangu puhul on tagajärjeks null punkti.

Käesoleva kriteeriumi kasutamise idee sai bakalaureusetöö autor kahest allikast ning kriteerium on olemuselt identne Mati Kärneri bakalaureusetöös oleva hindamispunktiga „Litsentsi tingimused“ [18, lk 10] [43].

1.4.6 Keele ja vormingu tugi

Mõnikord on vaja teha rakendust rahvusvahelisele kogukonnale. Leidub eri kuupäeva, numbri ja rahaühiku formaate ning palju keeli. Hinnatakse seda, kas raamistik hõlbustab selle mitmekesisusega hakkamasaamist vastavalt rakenduse kasutaja keele või asukohaseadete põhjal. Alltoodud loetelus on kirjas hinnete kolm kuni null määramise kriteeriumid.

1. Raamistik võimaldab kasutajal rakenduse keelt muuta ja kasutada asukohale sobivaid formaate.
2. Raamistik võimaldab kasutajal rakenduse keelt muuta või kasutada asukohale sobivaid formaate.
3. Raamistik võimaldab kasutajal rakenduse keelt muuta ja/või kasutada asukohale sobivaid formaate, kuid selleks tuleb raamistikust eraldiseisvaid lisatarkvara kasutada. Samas need lisatööriistad on mõeldud just vaadeldavale raamistikule.
4. Konkreetselt vaadeldavale raamistikule mõeldes loodud keele ja vormingu tuge pakkuv tarkvara puudub või on raskesti leitav.

Käesoleva kriteeriumi kasutamise idee sai bakalaureusetöö autor ühest allikast [18, lk 8].

1.4.7 TodoMVC testrakendusel põhinev keerukus

Testrakenduse põhjal saadav keerukushinnangu saab autor projekti *compexity-report* analüüsiprogrammi kasutades [44]. Idee käesoleva kriteeriumi kasutamiseks tuli Ramiro Gómezi TodoMVC projekti rakenduste keerukuse mõõtmise postitusest 2013. aastal [45]. Kuna on möödunud mitu aastat ja TodoMVC projekt on edasi arenenud, siis käesoleva töö

autor otsustas sarnasel viisil 15 TodoMVC testrakenduse keerukust hinnata ja kindlaks määrata Angulari, Backbone'i, Reacti, Emberi, Knockouti ja Dojo testrakenduste asukohad keerukuse pingeridades (lisa „I. Keerukuse pingeread"). Võrdlusesse lähevad JavaScripti failid, mis kuuluvad testrakenduse implementatsiooni ning mis pole raamistiku installimise failid. Allpool on toodud keerukuse hindamise kategooriad, mille põhjal koostatakse pingeread, mis paiknevad lisas „I. Keerukuse pingeread".

1. Lähtekoodi ridade arvu kategoorias liidetakse kokku loogiliste ridade arv.
2. McCabe'i tsüklomaatiline keerukuse (*Cyclomatic complexity*) keskmise kategoorias liidetakse kokku iga vaadeldava JavaScripti faili tsüklomaatilise keerukuse arv ning jagatakse vaadeldavate failide arvuga. Mida madalam väärtus, seda parem.
3. Iga vaadeldava JavaScripti faili funktsiooni põhjal leitakse Halstead'i jõupingutuste (*effort*) keskmine funktsioonide põhjal (*Mean per-function Halstead effort*). Halstead'i jõupingutus aitab hinnata vaimse pingutuse taset, mida läheb tarkvara arendamiseks või ülalpidamiseks vaja [46]. See saadakse Halstead'i raskusastet ja mahuastet korrutades. Mida suurem on jõupingutuse number, seda rohkem peab arendaja vaimselt pingutama.
4. Hooldatavuse (*maintainability*) indeksi kategoorias liidetakse kokku iga vaadeldava JavaScripti faili indeks ja jagatakse see testrakenduse raames vaadeldavate JavaScripti failide arvuga. Mida suurem on hooldatavuse number, seda paremini hooldatav on testrakendus.

Taolisel viisil moodustub testrakendustele vastavatest raamistikest neli pingerida. Järgnevas loetelus on kirjas nelja pingerea põhjal hinnete kolm kuni null määramise kriteeriumid. Raamistiku abil loodud testrakendusel on järgnev asukoht pingeridades:

1) vähemalt neljas pingereas esiviisikus;
2) kahes pingereas esiviisikus või neljas pingereas esimese kaheksa hulgas;
3) ühes pingereas esiviisikus või neljas pingereas esikümnes;
4) pingeridade lõpus.

1.4.8 TodoMVC testrakendusel põhinev õppimine ja õppimiskiirus

Bakalaureusetöö autor laadib testrakenduse alla ja asub olemasolevate materjalide abil rakenduse ning raamistiku toimimist õppima. Selle käigus seletatakse alapeatükina lahti olulisemad komponendid TodoMVC projekti testrakenduse näitel ning mõõdetakse, kui palju aega kulus. Alltoodud loetelus on kirjas hinnete kolm kuni null määramise kriteeriumid.

1. Kulus kuni kaheksa tundi.
2. Kulus rohkem kui kaheksa tundi.
3. Kulus rohkem kui kaksteist tundi.
4. Kulus rohkem kui neliteist tundi.

Käesoleva hindamiskriteerimi alapeatükk võimaldab bakalaureusetöö lugejal aru saada, kuidas raamistikud AngularJS, Backbone.js, React, Ember.js, KnockoutJS ja Dojo toimivad.

1.4.9 TodoMVC testrakendusel põhinev jõudlus

Jõudlushinnangu saab YSlow Chrome'i pistikprogrammi abil, mis annab maksimaalselt 100 punkti. Selleks järgib YSlow järgnevaid reegleid (tõlgitud otse inglise keelest eesti keelde) [47] [48]:

1) vähendada HTTP päringuid;

2) kasutada sisuedastusvõrku (*Content Delivery Network* ehk CDN);
3) vältida tühja src-d või href-i[6];
4) lisada päis *Expires* või vahemälukontrolli päis (*Cache-Control Header*);
5) tihendada komponente gzip-meetodiga;
6) panna stiililehed faili ülaosasse;
7) panna skriptifailid faili alaossa;
8) vältida CSS-avaldisi
9) teha JavaScript ja CSS välimisteks;
10) vähendada DNS-järelevaatuseid (*DNS Lookups*);
11) minimeerida JavaScripti ja CSS-i;
12) vältida ümbersuunamisi;
13) eemaldada korduvad skriptid;
14) seadistada olemisilte (*ETags*);
15) teha AJAX puhverdatavaks (*cacheable*);
16) kasutada GET-päringut AJAX päringute jaoks;
17) vähendada DOM-elementide arvu;
18) ei 404;
19) vähendada küpsise suurust;
20) kasutada küpsisevabu domeene staatiliste komponentide jaoks;
21) vältida filtreid;
22) mitte skaleerida pilte HTML-is;
23) teha favicon.ico väike ja puhverdatav.

Hinnete kolm kuni null andmisel kasutab autor YSlow hinnangut alltoodud viisil.

1. YSlow andis 90 kuni 100 punkti.
2. YSlow andis 80 kuni 89 punkti.
3. YSlow andis 70 kuni 79 punkti.
4. YSlow andis 0 kuni 69 punkti.

Käesoleva kriteeriumi kasutamise idee sai bakalaureusetöö autor ühest allikast [18, lk 10].

[6] Vastavate atribuutide väärtused ei tohi olla tühjad. Näiteks ei tohi esineda HTML-is `<img src="">`.

2. Võrdlus

2.1 AngularJS

Misko Hevery ja Adam Abrons tegelesid 2009. aastal kõrvalprojektiga GetAngular [49][7]. Hevery töötas ka Google Feedback'i arendamisega ning kirjutas selle 17 000 koodireaga rakenduse GetAngular abiga 1500-realiseks. Misko Hevery ning tema Google'i ülemus Brad Green muutsid projekti GetAngular projektiks AngularJS, millega tegeleb sellest ajast peale Google.

AngularJS on dünaamiliste CRUD (*Create, Read, Update, Delete*) veebirakenduste loomist parendav raamistik [50]. See püüab arendajat vabastada madalatasemelisest DOM-iga manipuleerimisest. Iisraeli Tehnoloogiainstituudiga seotud Uri Shaked kirjutab enda võrdluses „AngularJS vs. Backbone.js vs. Ember.js", et tema meelest on Angulari olulised võimalused kahesuunaline andmete sidumine (*two-way data binding*), sõltuvuse süstimine (*dependency injection*), kergesti testitav kood ning HTML-i laiendamine direktiividega (*directives*) [12].

2.1.1 Startimishinnang

Ettevõtetest kasutavad Angulari oma kodulehtede koodis näiteks valuutaülekannete firma Transferwise, Ameerika Ühendriikide televisioonifirma MSNBC ja lennufirma Virgin America[8].

Angulari kodulehelt saab peale allalaadimisnupule vajutamist valida raamistiku versiooni ning seda, kas allalaaditav tarkvara on minimeeritud, pakkimata või pakitud. Pakutud on ka sisuedastusvõrgu link, paketihaldurite installimiskäsud ja lisamoodulite kasutamise võimalus.

Bakalaureusetöö autor laadis alla faili angular.min.js ning asetas selle samasse kausta failiga teremaailm.html-iga, mille sisuks on HTML5-kood, kus on päises `<script>`-siltide alas viide allalaaditud failile ja kehas sektsioon `<div data-ng-app="">Tere {{2+2*2}}</div>`, mis kasutab Angulari arvutustehte sooritamiseks. Avades veebilehitsejas Chrome käesoleva HTML5-faili, kuvatakse teksti „Tere 6".

Eeltoodu vastab hinde kolm saamise kriteeriumitele. Angulari puhul piisab oluliste failide allalaadimisest õigesse kausta ning veebilehitsejas käivitatavale failile viidete loomisest. Lisaks leidub vähemalt kolm vaadeldavat raamistikku kasutavat edukat ärifirmat. Need on Virgin America, MSNBC ja Transferwise. Autor annab startimishinnangu kategoorias kolm punkti.

2.1.2 Kogukond ja dokumentatsioon

2015. aasta augustikuu seisuga on raamistiku Githubi lehel 3878 vaatajat, 41803 tähega märkijat ja 18469 koodi harutajat [54]. Stack Exchange'i lehelt otsides leiab 115 479 AngularJS-sildiga küsimust, millest vastamata on 40 792 [55] [56].

Raamistiku kodulehel on menüüs valik „*Develop*", kust saab edasi liikuda näidisrakenduse loomise õpetuseni (*Tutorial*), olulisi põhimõtteid kirjeldava arendaja teejuhini (*Developer Guide*) või üksikasjaliku API-informatsioonini (*API Reference*) [57] [58] [59] [60]. Nende kolme olulise abimaterjali juures asetseb nupp „*Improve this Doc*", mille abil saab soovi-

[7] Terve ülejäänud lõik põhineb allikal [49].
[8] [51] [52] [53]

tada muudatusi. Lisaks leidub palju materjalist arusaamist hõlbustavaid koodinäiteid. Dokumentatsioon on litsentsiga CC BY 4.0 [61].

Eeltoodu vastab hinde kolm saamise kriteeriumitele. Githubis olevate vaatajate, tähega märkijate ning koodi harutajate (inglise keeles on koodi harutamine *forking*) summa on 64150. Stack Exchange'is moodustavad mittevastatuks märgitud küsimused 35% kõigist raamistiku nime kandva sildiga küsimustest. Lisaks eksisteerib koodinäiteid sisaldav dokumentatsioon, milles arendaja saab teha ettepanekuid muudatusteks. Autor annab kogukonna ja dokumentatsiooni kategoorias kolm punkti.

2.1.3 Arendustööriistade tugi

Integreeritud programmeerimiskeskkonnale Eclipse leidub pistikprogramm nimega AngularJS Eclipse, mis pakub raamistiku funktsionaalsusega seotud koodi lõpule viimist (*code completion*) nii HTML-is kui ka JavaScriptis [62] [63] [64]. Täiendavalt annab spetsiaalne vaade „*Angular Explorer View*" projekti kausta puuvaates (*tree view*) parema ülevaate raamistikuga seotud moodulitest ja kontrolleritest [62] [65].

Tekstiredaktorile Sublime leidub mitmeid pistikprogramme [66]. Üks populaarne valik on nimega AngularJS, mis pakub näiteks raamistiku koodi lõpule viimist (*code completion*), koodi väljavõtteid (*snippets*) ja definitsioonideni liikumist (*go to definition*) [67]. Jetbrainsi toodetele on loodud sama nimega pistikprogramm John Lindquisti, Dennis Ushakovi ja Alexander Zolotovi poolt [68] [69].

Programmeerimiskeskkonnale Netbeans eksisteerib pistikprogramm nimega AngularJS Tools, mis pakub raamistiku koodi lõpule viimist (*code completion*) HTML-failides [70] [71]. Viimane uuendus on tehtud aga 2011. aastal. Selle info põhjal võib Netbeansi toe Angularile lugeda nõrgaks.

Visual Studiol on sissehitatud Angulari tugi, mis toetab näiteks sõltuvuse süstimise süsteemi ja HTML-koodi Angulari stiilis [72].

Autor annab arendustööriistade toe eest kolm punkti, sest toetatud on vähemalt neli vaadeldavat arendustööriista.

2.1.4 Testimistugi

Angulari kodulehel olevas arendaja teejuhis leidub peatükk ühiktestimisest (*unit testing*), kus tutvustatakse testimistööriistu Karma ning Jasmine [73]. Seejärel tuuakse koodinäiteid, kuidas testimisega algust teha.

Angulariga loodud rakenduse testimist seletav materjal koos koodinäidistega ekisteerib. Autor annab testimistoe kriteeriumi eest kolm punkti.

2.1.5 Litsents äri vaatepunktist

Raamistiku AngularJS puhul kehtib MIT (Massachusetts Institute of Technology) litsents [57]. Angulari autoriõiguse hoidja Google ei vastuta selle kasutamisega kaasneda võiva kahju eest [74]. Litsentsis paiknevad autoriõiguse märkus ja litsentsi teadaanne peavad eksisteerima kõigis koopiates või olulises mahus Angulariga loodavas tarkvaras [75].

Vastava raamistiku abil loodud rakendust lubatakse äriks kasutada, rakenduse lähtekood võib jääda salajaseks. AngularJS saab litsentsi eest kolm punkti.

2.1.6 Keele ja vormingu tugi

Angulari dokumentatsiooni arendaja teejuhis on eraldi peatükk internatsionaliseerimise (i18n) ja lokaliseerimise (l10n) kohta [76]. Kuupäeva, numbri ja valuuta formaatide jaoks eksisteerivad spetsiaalsed filtrid. Githubis saab vaadata toetatud lokaate (*locales*) [77].

Rakenduse kuvakeele muutmiseks ei leidnud bakalaureusetöö autor dokumentatsioonist piisavat infot. Lisavõimaluste uurimise järel pakub soovitavats funktsionaalsust moodul `angular-translate` [78].

Tuleb siiski märkida, et `angular-translate` pole Angulari kodulehel pakutavate lisamoodulite hulgas, vaid eraldi Angulari jaoks mõeldud tarkvara. Autor annab keele ja vormingu toe kriteeriumi eest kaks punkti, sest raamistik võimaldab kasutada asukohale sobivaid formaate.

2.1.7 TodoMVC testrakendusel põhinev keerukus

Vastavalt üldkriteeriumitele sai hinnatud testrakenduse JavaScripti faile, mis on esindatud alapunktis 2.1.8 [22]. Tulemused on toodud allpool.

1. Lähtekoodi loogiliste ridade arv kokku liidetult paikneb tabelis 1.

Tabel 1. Angulari testrakenduse ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
app.js	12	167
todoCtrl.js	63	
todoEscape.js	10	
todoFocus.js	7	
todoStorage.js	75	

2. McCabe'i tsüklomaatilise keerukuse (*Cyclomatic complexity*) keskmine (liidetakse kokku iga vaadeldava JavaScripti faili tsüklomaatilise keerukuse arv ning jagatakse vaadeldavate failide arvuga) paikneb tabelis 2.

Tabel 2. Angulari testrakenduse tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
app.js	1	4
todoCtrl.js	11	
todoEscape.js	2	
todoFocus.js	2	
todoStorage.js	4	

3. Halstead'i jõupingutuste (*effort*) keskmine funktsioonide põhjal (*Mean per-function Halstead effort*) oli 285, 8976894724569.
4. Hooldatavuse indeks (liidetakse kokku iga vaadeldava JavaScripti faili indeks ja jagatakse see vaadeldavate failide arvuga) paikneb tabelis 3.

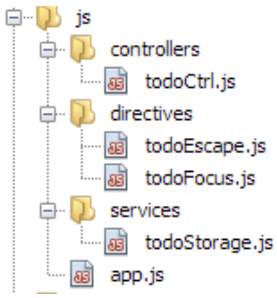
Tabel 3. Angulari testrakenduse hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**
app.js	129, 86594898240182	137, 37
todoCtrl.js	131, 10181106386105	
todoEscape.js	142, 4916344635128	
todoFocus.js	151, 9798973691816	
todoStorage.js	131, 425444907299	

Angulariga loodud testrakendus kuulub loogiliste ridade arvu kategoorias 8.–9. kohale (lisa „I. Keerukuse pingeread", tabel 20), tsüklomaatilise keerukuse kategoorias 8. kohale (lisa „I. Keerukuse pingeread", tabel 21), hooldatavuse indeksi kategoorias 2. kohale (lisa „I. Keerukuse pingeread", tabel 22) ning funktsioonide põhjal saadud Halstead'i jõupingutuste (*effort*) keskmise kategoorias 3. kohale (lisa „I. Keerukuse pingeread", tabel 23). AngularJS saab testrakendusel põhineva keerukuse kategoorias kaks punkti, sest tulemuseks on kahes pingereas esiviisikusse kuulumine.

2.1.8 TodoMVC testrakendusel põhinev õppimine ja õppimiskiirus

Järgneva MIT litsentsiga TodoMVC projekti testrakenduse autorid on Christoph Burgdorf, Eric Bidelman, Jacob Mumm ja Igor Minar (lisa „III. TodoMVC projekti repositooriumi sisu kasutamise litsents") [22]. Rakenduse struktuuri välimine fail on index.html. Sellega samas kaustas on kaust `js`, mille sisu on toodud joonisel 2.



Joonis 2. Kausta `js` sisu [22]

Failis app.js on defineeritud Angulari moodul `todomvc`, mis kasutab sõltuvusena lisamooduleid `ngRoute` ja `ngResource` (joonis 3) [79] [80]. Käesolevat moodulit seadistatakse Angulari muutujaga `$routeProvider`, mis jälgib, kas URL-teeks on kaldkriips või kaldkriips koos mingi lisainfoga [81]. Mainitud kahel juhul suunatakse kasutaja muutujas `routeConfig` defineeritud asukohta. Vastasel juhul seatakse URL-teeks kaldkriips. Muutujas `routeConfig` hoitakse objekti, mis kasutab kontrollerit `TodoCtrl` ning vaate malli `todomvc-index.html`, mis asub failis index.html. Atribuudiga `re-`

`solve` tekitatakse lisasõltuvus, mis süstitakse kontrollerisse [81]. Tänu sellele tegevusele töötab rakendus tagarakenduse (*back end*) API või veebilehitseja andmehoiu (HTML5 Local Storage) abil, mille tõttu saab rakenduses kasutatavaid andmeid salvestada [21].

```
/**
 * The main TodoMVC app module
 *
 * @type {angular.Module}
 */
angular.module('todomvc', ['ngRoute', 'ngResource'])
    .config(function ($routeProvider) {
        'use strict';
        var routeConfig = {
            controller: 'TodoCtrl',
            templateUrl: 'todomvc-index.html',
            resolve: {
                store: function (todoStorage) {
                    // Get the correct module (API or localStorage).
                    return todoStorage.then(function (module) {
                        module.get(); // Fetch the todo records in the background.
                        return module;
                    });
                }
            }
        };
        $routeProvider
            .when('/', routeConfig)
            .when('/:status', routeConfig)
            .otherwise({
                redirectTo: '/'
            });
    });
```

Joonis 3. Fail app.js [22]

Faili todoCtrl.js alguses on näha, et moodulile `todomvc` luuakse vastav kontroller `TodoCtrl`, mis kasutab Angulari argumente `$scope`, `$routeParams` ja `$filter` (joonis 4). Lisaks esineb kontrolleri definitsioonis argumendina ka `store`, mis on seotud andmete salvestamise funktsionaalsusega ja failiga todoStorage.js. Objekt `$scope` on Angularis mudeli rollis: selles paiknevat infot on võimalik esitada HTML-vaates ning sellega saavad suhelda kontrollerid (*controller*) [82] [83, lk 13].

```
/*global angular */

/**
 * The main controller for the app. The controller:
 * - retrieves and persists the model via the todoStorage service
 * - exposes the model to the template and provides event handlers
 */
angular.module('todomvc')
    .controller('TodoCtrl', function TodoCtrl($scope, $routeParams, $filter, store) {
        'use strict';

/* Alltoodud märgistus märgistab siin ja edaspidi, et kood päris failis jätkub.*/
/* ... */ või //...//
```

Joonis 4. Fail todoCtrl.js [22]

Argumendi `store` kaudu saadud info tegevusobjektide kohta (*todos*) hoitakse muutujas `todos` ning atribuudis `$scope.todos` (joonis 5). Atribuutidele `$scope.newTodo` ja `$scope.editedTodo` omistatakse vastavalt tühi sõne ning väärtus `null`. Seejärel hakatakse jälgima muutujat `todos` meetodiga `$watch` [84]. Muutuse korral käivitatakse meetodi `$watch` sees paiknev tagasikutse funktsioon (*callback*). Filtriga `$filter` salvestatakse atribuuti `$scope.remainingCount` selliste tegevusobjektide arv, mis pole märgitud tehtuks ehk mille valmisoleku atribuudi `completed` väärtus on väär (*false*) [85]. Lõpetatud tegevuste arv salvestatakse atribuuti `$scope.completedCount` ning

tõeväärtus kõigi tegevuste tehtuks märkimise oleku kohta omistatakse atribuudile `$scope.allChecked`. Sündmus `$routeChangeSuccess` käivitub, kui URL-tee muutus on edukalt toimunud [86]. Meetod `$on` kutsub sündmuse `$routeChangeSuccess` korral välja enda teises argumendis defineeritud funktsiooni, mis kasutab URL-tee parameetrite lugemiseks teenust `$routeParams` [87] [88]. Atribuuti `$scope.statusFilter` salvestatakse URL-prarameetritest pärit atribuudi `status` põhjal objekt tõeväärtusatribuudiga `completed` või tühi objekt, kui `status` on defineerimata. Eelneva tegevuse eesmärk on võimaldada failis index.html atribuuti `statusFilter` kasutada filtrina, et vastavalt rakenduse kasutaja soovile kuvada kas kõiki, tehtuks märgitud või mittetehtuks märgitud tegevuste objekte [89].

```
/* ... */
        var todos = $scope.todos = store.todos;

        $scope.newTodo = '';
        $scope.editedTodo = null;
        $scope.$watch('todos', function () {
            $scope.remainingCount = $filter('filter')(todos, { completed: false }).length;
            $scope.completedCount = todos.length - $scope.remainingCount;
            $scope.allChecked = !$scope.remainingCount;
        }, true);
        // Monitor the current route for changes and adjust the filter accordingly.
        $scope.$on('$routeChangeSuccess', function () {
            var status = $scope.status = $routeParams.status || '';
            $scope.statusFilter = (status === 'active') ?
                { completed: false } : (status === 'completed') ?
                { completed: true } : {};
        });
/* ... */
```

Joonis 5. todoCtrl.js [22]

Tegevuse objekti lisamise funktsiooni `addTodo` implementatsioon on toodud joonisel 6. Failis index.html direktiivi `ngModel` kujul `ng-model="newTodo"` kasutavast tekstialast saadud üleliigsete tühikuteta tekst omistatakse objekti `newTodo` pealkirja atribuudi `title` väärtuseks ning kontrollitakse, et tegu poleks tühja sõnega [90] [91]. Seejärel omistatakse atribuudile `$scope.saving` tõene tõeväärtus ning sisestatakse andmetesalvestuskohta `store` funktsiooniga `insert` vastloodud tegevuse objekt. Õnnestumise korral lähtestatakse `$scope.newTodo` tühja sõnega. Lõpuks muudetakse funktsioonis `insert` atribuudi `$scope.saving` tõeväärtus vääraks. Senikaua, kuni sisestamisprotsess funktsioonis `insert` kestab, on kasutaja jaoks sisendi andmine võimalus peatatud, sest failis index.html on atribuut `$scope.saving` seotud direktiiviga `ngDisabled` [92].

```
/* ... */
        $scope.addTodo = function () {
            var newTodo = {
                title: $scope.newTodo.trim(),
                completed: false
            };
            if (!newTodo.title) {
                return;
            }
            $scope.saving = true;
            store.insert(newTodo)
                .then(function success() {
                    $scope.newTodo = '';
                })
                .finally(function () {
                    $scope.saving = false;
                });
        };
/* ... */
```

Joonis 6. todoCtrl.js ning `addTodo` [22]

Failis index.html on loodud võimalus juba sisestatud tegevuse objekti redigeerida. Tegevuse nime muutmiseks on sinna lisatud `<label>`-sildiga alasse `ng-dblclick="editTodo(todo)"`. Järelikult, kui kasutaja teeb tekstil topeltklõpsu, käivitub funktsioon `$scope.editTodo` (joonis 7) [93]. See võtab argumendiks sündmusega seotud tegevuse objekti ja omistab väärtused atribuutidele `$scope.editedTodo` ning `$scope.originalTodo`. Viimane mainitud atribuut kujutab endast funktsiooniga `angular.extend` loodud objekti koopiat argumendist `todo` [94].

```
/* ... */
        $scope.editTodo = function (todo) {
            $scope.editedTodo = todo;
            // Clone the original todo to restore it on demand.
            $scope.originalTodo = angular.extend({}, todo);
        };
/* ... */
```

Joonis 7. todoCtrl.js ning `editTodo` [22]

Failis index.html on vormi `<form>`-sildiga alas kasutatud direktiivi `ngSubmit` kujul `ng-submit="saveEdits(todo, 'submit')"`, millega saadetakse vormist pärit info kontrolleri `TodoCtrl` funktsioonile `$scope.saveEdits` (joonis 8) [95]. Failis index.html esineb `saveEdits` ka eelnevalt mainitud vormi sees paiknevas `<input>`-siltidega alas kujul `ng-blur="saveEdits(todo, 'blur')"`. Järelikult kutsutakse direktiiviga `ngSubmit` välja kohandatud tegevus hägustumissündmusele (*blur event*), mis ilmneb, kui mingi element kaotab fookuse [95] [96]. Funktsiooni `$scope.saveEdits` koodi alguses hoolitsetakse selle eest, et kasutaja tehtud muudatused õigesti salvestuksid ning välditakse mitmekordset salvestamist. Atribuudi `$scope.reverted` tõeväärtuse põhjal kontrollitakse, kas kasutaja katkestas muutmisprotsessi. Muudatusi ei salvestata muutmisprotsessi katkestamise korral või uue pealkirja võrdumise korral eelnevalt tehtud tegevuse objekti koopia `$scope.originalTodo` pealkirjaga. Kui mitmed eelnevalt mainitud tingimuskontrollid on läbitud, siis muudetakse andmete salvestamise kohas `store` tegevuse objekti pealkirja failis todoStorage.js defineeritud funktsiooniga `put`. Tegevuse objekt eemaldatakse kohast `store` funktsiooniga `delete`, kui tegevuse objekti pealkiri kustutatakse. Muutmise või kustutamise protsessi ebaõnnestumise korral taastatakse funktsioonis `error` tegevuse objekti pealkiri vastavalt redigeerimisele eelnenud olukorrale koopiast `$scope.originalTodo`.

```
/* ... */
        $scope.saveEdits = function (todo, event) {
            // Blur events are automatically triggered after the form submit event.
            // This does some unfortunate logic handling to prevent saving twice.
            if (event === 'blur' && $scope.saveEvent === 'submit') {
                $scope.saveEvent = null;
                return;
            }
            $scope.saveEvent = event;

            if ($scope.reverted) {
                // Todo edits were reverted-- don't save.
                $scope.reverted = null;
                return;
            }
            todo.title = todo.title.trim();

            if (todo.title === $scope.originalTodo.title) {
                $scope.editedTodo = null;
                return;
            }
```

```
            store[todo.title ? 'put' : 'delete'](todo)
                .then(function success() {}, function error() {
                    todo.title = $scope.originalTodo.title;
                })
                .finally(function () {
                    $scope.editedTodo = null;
                });
        };
/* ... */
```

Joonis 8. todoCtrl.js ning `saveEdits` [22]

Funktsioonile `$scope.revertEdits` viidatakse faili index.html eelnevas lõigus mainitud vormi `<input>`-siltide alas kujul `todo-escape="revertEdits(todo)"`. Seega kasutatakse failis todoEscape.js (joonis 10) defineeritud direktiivi koos failis todoCtrl.js paikneva funktsiooniga `revertEdits` (joonis 9). Funktsiooni `$scope.revertEdits` töö tulemusena taastatakse argumendiks saadud tegevuse objekt redigeerimisele eelnevalt tehtud koopia `$scope.originalTodo` abil, lähtestatakse atribuudid `$scope.editTodo` ja `$scope.originalTodo` väärtusega `null` ning omistatakse atribuudile `$scope.reverted` tõene tõeväärtus.

```
/* ... */
        $scope.revertEdits = function (todo) {
            todos[todos.indexOf(todo)] = $scope.originalTodo;
            $scope.editedTodo = null;
            $scope.originalTodo = null;
            $scope.reverted = true;
        };
/* ... */
```

Joonis 9. todoCtrl.js ning `revertEdits` [22]

Funktsioon `revertEdits` on seotud direktiiviga failis todoEscape.js, mis on defineeritud meetodiga `Module.directive` [10, lk 383]. Direktiivi nimi `todoEscape` on meetodi esimene argument ning tehasefunktsioon (*factory function*) on teine argument [10, lk 383]. HTML-koodis tähistavad direktiivi nimes esinevad suurtähed omaette sõnu, mis eraldatakse sidekriipsuga ning kirjutatakse läbivalt väikeste tähtedega, näiteks `todoEscape` ja `todo-escape` [10, lk 383–384]. Tehasefunktsiooni sees asetseb viitfunktsioon (*link function*), mis ühendab omavahel direktiivi, HTML-faili ning programmi skoobi (*scope*) [10, lk 385]. Viitfunktsioon võtab argumendid `scope`, `elem` ja `attrs`. Seega saadakse selles funktsioonis ligipääs direktiiviga seotud skoobile, HTML-elemendile ja viimase atribuutidele. Joonisel 10 kujutatud koodi põhjal käivitab klahvi Escape alla vajutamine elemendiga `elem` seotud atribuudi `todo-escape` kasutamise [97]. Seega rakendatakse meetodiga `$apply` failis index.html atribuudiga `todo-escape` seotud funktsiooni `revertEdits` [10, lk 319, 345]. Direktiivi lõpus paiknev `scope.$on('$destroy'…` on hea tava: direktiivid peavad enda järel koristama, et vältida näiteks mälulekkeid [98].

```
/**
 * Directive that executes an expression when the element it is applied to gets
 * an `escape` keydown event.
 */
angular.module('todomvc')
    .directive('todoEscape', function () {
        'use strict';

        var ESCAPE_KEY = 27;
        return function (scope, elem, attrs) {
            elem.bind('keydown', function (event) {
                if (event.keyCode === ESCAPE_KEY) {
                    scope.$apply(attrs.todoEscape);
                }
            });
```

```
            scope.$on('$destroy', function () {
                elem.unbind('keydown');
            });
        };
    });
```

Joonis 10. todoEscape.js [22]

Ka failis todoFocus.js on defineeritud direktiiv (joonis 11). Sealse tehasefunkstiooni viitfunktsioon kasutab meetodit `$watch`, et elemendiga `elem` seotud atribuudi `todo-focus` tõeväärtuse muutuse korral läbi teenuse `$timeout` rakendada elemendile `elem` fookusesse tõstmise meetodit `focus` [84] [10, lk 319] [99] [100]. Vastavalt oma kolmele argumendile kutsub `$timeout` fookuse tekitaja välja viivitamatult [99].

```
/**
 * Directive that places focus on the element it is applied to when the
 * expression it binds to evaluates to true
 */
angular.module('todomvc')
    .directive('todoFocus', function todoFocus($timeout) {
        'use strict';
        return function (scope, elem, attrs) {
            scope.$watch(attrs.todoFocus, function (newVal) {
                if (newVal) {
                    $timeout(function () {
                        elem[0].focus();
                    }, 0, false);
                }
            });
        };
    });
```

Joonis 11. todoFocus.js [22]

Failis todoCtrl.js defineeritud funktsioonid `$scope.removeTodo` ja `$scope.saveTodo` võimaldavad andmete salvestamise asukohast `store` argumendiks saadud tegevuse objekti vastavalt kustutada funktsiooniga `delete` või lisada funktsiooniga `put` (joonis 12).

```
/* ... */

        $scope.removeTodo = function (todo) {
            store.delete(todo);
        };

        $scope.saveTodo = function (todo) {
            store.put(todo);
        };

/* ... */
```

Joonis 12. todoCtrl.js ning `removeTodo` ja `saveTodo` [22]

Failis index.html on igale tegevusobjektide loetelus esinevale tegevuse pealkirjale lisatud kõrvale oma märgistuskast (*checkbox*) `<input>`-siltide alasse koos direktiividega kujul `ng-model="todo.completed" ng-change="toggleCompleted(todo)"`. Järelikult jälgib mooduli `ng` direktiiv `ngChange` olukorda, kus kasutaja muudab märgistuskasti olekut ja käivitab siis funktsiooni `$scope.toggleCompleted` [101]. Direktiiv `ngModel` seob märgistuskasti väärtuse vastava tegevuse objekti atribuudi `completed` väärtusega [90]. Funktsioonis `$scope.toggleCompleted` omistatakse argumendi `completed` tõeväärtus argumendiks saadud tegevuse objekti `todo` atribuudi `completed` väärtuseks ning salvestatakse tehtud muudatus funktsiooniga `put` andmete salvestamise kohta `store` (joonis 13). Kui selle ülesande sooritamine ebaõnnestub, siis käivitatakse funktsioon `error`. Funktsioon `$scope.markAll` on HTML-failis kõikide tegevuste korraga märkimiseks mõeldud märgistuskastiga seotud direktiivide abil kujul `ng-model="allChecked" ng-click="markAll(allChecked)"`. Kui kasuta-

ja klõpsab seda märgistuskasti, siis käivitub funktsioon `$scope.markAll`, mis läbib tsükliga kõikide tegevuste kogumi ning muudab vajadusel funktsiooniga `$scope.toggleCompleted` vaadeldava objekti atribuudi `completed` väärtuse argumendiks saadud väärtusega võrdseks. Direktiiv `ngModel` seob märgistuskasti väärtuse atribuudi `$scope.allChecked` väärtusega [90]. Failis index.html on nupp, millele klõpsamine käivitab funktsiooni `$scope.clearCompletedTodos`, mille ainuke ülesanne on välja kutsuda funktsioon `store.clearCompleted`, mis on defineeritud failis todoStorage.js.

```
/* ... */
        $scope.toggleCompleted = function (todo, completed) {
            if (angular.isDefined(completed)) {
                todo.completed = completed;
            }
            store.put(todo, todos.indexOf(todo))
                .then(function success() {}, function error() {
                    todo.completed = !todo.completed;
                });
        };
        $scope.clearCompletedTodos = function () {
            store.clearCompleted();
        };
        $scope.markAll = function (completed) {
            todos.forEach(function (todo) {
                if (todo.completed !== completed) {
                    $scope.toggleCompleted(todo, completed);
                }
            });
        };
    });
```

Joonis 13. todoCtrl.js ning `toggleCompleted`, `clearCompletedTodos` ja `markAll` [22]

Failis todoStorage.js on loodud teenus (*service*) `todoStorage` meetodiga `Module.factory`, milles esimene argument on teenuse nimi ning teine argument on tehasefunktsioon (*factory function*) (joonis 14) [10, lk 480]. HTTP-serveriga suhtlemiseks mõeldud Angulari teenuse `$http` kaudu kontrollitakse, kas on võimalik kasutada tagarakenduse API-it. Bakalaureusetöö autor tegi käesoleval juhul TodoMVC testrakenduse koodis natuke muudatusi, et katsetada lokaalselt Deployd API Engine teenust, mida on võimalik arvutisse paigaldada lehelt http://deployd.com/ [103]. Nimelt andis autor meetodile `$http.get` argumendi `'http://localhost:2403/todos'`. Selleks et katsetada tagarakenduse funktsionaalsust, käivitas autor Node.js-põhise serveri faili server.js abil, lõi Deployd haldamislehel vajaliku struktuuri `todos` ja kohandas failis todoStorage.js esinevaid viiteid vastavalt tagarakenduse teenusele. Lisas II on toodud faili server.js kood ning Deployd haldamislehe ekraanivaade. Autor nimetab Deployd lihtsustatult edaspidises tekstis andmebaasiks. Teenus `todoStorage` tagastab andmebaasi olemasolu korral meetodiga `$injector.get` teenuse `todos` isendi, muul juhul aga teenuse `localStorage` isendi [104].

```
/**
 * Services that persists and retrieves todos from localStorage or a backend API
 * if available.
 *
 * They both follow the same API, returning promises for all changes to the
 * model.
 */
angular.module('todomvc')
    .factory('todoStorage', function ($http, $injector) {
        'use strict';
        // Detect if an API backend is present. If so, return the API module, else
        // hand off the localStorage adapter
        return $http.get('http://localhost:2403/todos')
            .then(function () {
                return $injector.get('todos');
            }, function () {
                return $injector.get('localStorage');
            });
    }
/* ... */
```

Joonis 14. todoStorage.js ning andmebaasi valimine [22]

Teenuse `todos` tehasefunktsioon kasutab argumendina moodulisse `ngResource` kuuluvat teenust `$resource`, mille abil saab serveriga suhelda (joonis 15) [105][9]. Objekti `store` atribuudi `api` väärtuseks saab teenuse `$resource` abil defineeritud andmebaasiga suhtlust võimaldav objekt. Esimene argument on `$resource`'il URL ligipääsuga andmebaasi tegevusobjektide andmestruktuurile koos parameetriga `id`. Defineeritud on meetod `update`, mis võimaldab HTTP PUT-meetodiga uuendusi teha.

```
    .factory('todos', function ($resource) {
        'use strict';
        var store = {

            todos: [],

            api: $resource('http://localhost:2403/todos/:id', null,
                {
                    update: { method:'PUT' }
                }
            ),

/* ... */
```

Joonis 15. todoStorage.js ning teenus `todos` [22]

Objekti `store` sees on kirja pandud mitu funktsiooni. TodoMVC testrakenduse funktsioon `clearCompleted` ei toiminud käesoleva andmebaasiga korrektselt. Seetõttu kirjutas bakalaureusetöö autor selle funktsiooni ümber (joonis 16) ning tegi TodoMVC projekti autoritele ettepaneku (*pull request*) seda samuti muuta projekti repositooriumis. Funktsioonis `clearCompleted` läbitakse tsükliga kõigi tegevusobjektide kogum ning kustutatakse andmebaasist tehtuks märgitud tegevused.

```
/* ... */
            clearCompleted: function () {
                var completeTodos = store.todos.filter(function (todo) {
                    return todo.completed;
                });

                for (var i = 0; i < completeTodos.length; i++) {
                    store.delete(completeTodos[i]);
                };
            },
/* ... */
```

Joonis 16. todoStorage.js ning teenuse `todos` funktsioon `store.clearCompleted`

[9] Ülejäänud lõik põhineb samuti allikal [105].

Taastamisvõimaluse tagamiseks tehakse funktsioonis `store.delete` enne kustutamisprotsessi muutujasse `originalTodo` JavaScripti meetodiga `slice` loetelurivi tüüpi (*array*) koopia loetelurivi tüüpi atribuudis `store.todo` paiknevatest tegevusobjektidest (joonis 17) [106]. Seejärel kasutatakse JavaScripti meetodit `splice`, et kustutada argumendiks saadud tegevuse objekt loetelurivist `store.todos` [107]. Andmebaasist kindla tegevuse objekti kustutamiseks kasutatakse objekti `api` kustutamismeetodi `delete` argumendis unikaalset identifikaatorit `id` [103]. Ebaõnnestumise korral käivitatakse funktsioonis `error` kopeerimisfunktsioon `angular.copy`, et taastada ebaõnnestunud kustutamisele eelnenud tegevusobjektide loetelurivi [108].

```
/* ... */

            delete: function (todo) {

                var originalTodos = store.todos.slice(0);

                store.todos.splice(store.todos.indexOf(todo), 1);

                return store.api.delete({ id: todo.id },
                    function () {
                    }, function error() {
                        angular.copy(originalTodos, store.todos);
                    });
            },

/* ... */
```

Joonis 17. todoStorage.js ning teenuse `todos` funktsioon `store.delete` [22]

Funktsiooni `get` ülesanne on andmebaasist ammutada salvestatud info tegevusobjektide kohta (joonis 18). Selle sees paiknev meetod `store.api.query` kasutab HTTP GET-meetodit, et andmebaasist vajalik info küsida ning salvestada vastus kopeerimisfunktsiooniga `angular.copy` atribuuti `store.todos` [105] [108].

```
/* ... */
            get: function () {
                return store.api.query(function (resp) {
                    angular.copy(resp, store.todos);
                });
            },
/* ... */
```

Joonis 18. todoStorage.js ning teenuse `todos` funktsioon `store.get` [22]

Tegevusobjektide andmebaasi sisestamise funktsioonis `insert` kasutatakse eelnevalt selgitatud koopiapõhist taastamistehnikat juhuks, kui midagi läheb valesti (joonis 19). Käesolevas funktsioonis saadab meetod `store.api.save` HTTP POST-meetodiga andmed andmebaasi [105]. Eduka andmebaasisisestuse järel tulnud vastusest pärit andmebaasi poolt määratud unikaalne `id` omistatakse tegevuse objektile `todo`, mis lisatakse JavaScripti meetodiga `push` loetelurivvi `store.todos` [109]. Funktsioonis `insert` tagastatakse käivitatud meetodi `store.api.save` atribuut `$promise`, mis on ressursiisendi (*Resource instance*) lisatribuut: tegemist on asünkroonse funktsiooni toimumise õnnestumise indikaatoriga [105] [110].

```
/* ... */
            insert: function (todo) {
                var originalTodos = store.todos.slice(0);

                return store.api.save(todo,
                    function success(resp) {
                        todo.id = resp.id;
                        store.todos.push(todo);
                    }, function error() {
                        angular.copy(originalTodos, store.todos);
                    })
                    .$promise;
            },
/* ... */
```

Joonis 19. todoStorage.js ning teenuse `todos` funktsioon `store.insert` [22]

Objektis `store` on viimasena defineeritud funktsioon `put`. See uuendab argumendiks saadud tegevuse objekti `todo` atribuudile `id` vastavat andmebaasiobjekti HTTP PUT-meetodit kasutava meetodiga `store.api.update` (joonis 20). Funktsioon `put` tagastab atribuudi `$promise` [105]. Teenus `todos` tagastab oma definitsiooni lõpus objekti `store`.

```
/* ... */
            put: function (todo) {
                return store.api.update({ id: todo.id }, todo)
                    .$promise;
            }
        };

        return store;
    })
/* ... */
```

Joonis 20. todoStorage.js ning teenuse `todos` funktsioon `store.put` [22]

Failis todoStorage.js on tagarakenduse API puudumise korral andmete salvestamiseks kasutusel teenuses `localStorage` veebilehitseja andmehoid ehk HTML5 Local Storage (joonis 21) [21]. Muutuja `STORAGE_ID` põhjal saab funktsioonidega `getItem` ja `setItem` abifunktsioonides `_getFromLocalStorage` ja `_saveToLocalStorage` andmeid andmehoiust ammutada ning sinna sisestada [111] [112]. Andmete JSON-kujul töötlemiseks kasutatakse JavaScripti funktsioone `JSON.stringify()` ning `JSON.parse()` [113] [114].

```
/* ... */
    .factory('localStorage', function ($q) {
        'use strict';

        var STORAGE_ID = 'todos-angularjs';

        var store = {
            todos: [],
            _getFromLocalStorage: function () {
                return JSON.parse(localStorage.getItem(STORAGE_ID) || '[]');
            },
            _saveToLocalStorage: function (todos) {
                localStorage.setItem(STORAGE_ID, JSON.stringify(todos));
            },
/* ... */
```

Joonis 21. todoStorage.js ning teenus `localStorage` [22]

Veebilehitseja andmehoiust kõiki tehtuks märgitud tegevusobjekte kustutav funktsioon `clearCompleted` tekitab esmalt objekti `deferred`, mis kujutab endast Angulari Deferred API isendit (joonis 22) [115]. Taolise objekti eesmärk on asünkroonsete väljakutsete puhul võimaldada aru saada, kas väljakutse on edukalt täidetud, ebaedukalt täidetud või mis on väljakutse staatus [115]. Funktsioonis `clearCompleted` filtreeritakse välja mittetehtuks märgitud tegevusobjektid ning kopeeritakse need loetelurivvi `store.todos` ja

veebilehitseja andmehoidu. Siis lahendatakse meetodiga `resolve` objektiga `deferred` seotud lubadus (*promise*) ehk lõpetatakse alustatud ülesanne ära [115] [116]. Lõpuks tagastatakse objekti `deferred` asünkroonse väljakutse õnnestumise indikaatoriks olev atribuut `promise` [115] [116].

```
/* ... */
            clearCompleted: function () {
                var deferred = $q.defer();

                var incompleteTodos = store.todos.filter(function (todo) {
                    return !todo.completed;
                });
                angular.copy(incompleteTodos, store.todos);

                store._saveToLocalStorage(store.todos);
                deferred.resolve(store.todos);

                return deferred.promise;
            },
/* ... */
```

Joonis 22. todoStorage.js ning teenuse `localStorage` funktsioon `store.clearCompleted` [22]

Ülejäänud funktsioonid `delete`, `get`, `insert` ja `put` on analoogsed eespool esitatud funktsioonidega. Seega autor nendel neid ei käsitle. Teenus `localStorage` tagastab lõpuks objekti `store`.

Faili index.html päises viidatakse kasutatud CSS-failidele (joonis 23). Lisaks kasutatakse direktiivi `ngCloak` eesmärgiga mitte näidata rakenduse kasutajale poolikut sisu enne Angulari poolt tehtava töötluse lõppu [117].

```
<!doctype html>
<html lang="en" data-framework="angularjs">
    <head>
        <meta charset="utf-8">
        <title>AngularJS • TodoMVC</title>
        <link rel="stylesheet" href="node_modules/todomvc-common/base.css">
        <link rel="stylesheet" href="node_modules/todomvc-app-css/index.css">
        <style>[ng-cloak] { display: none; }</style>
    </head>
/* ... */
```

Joonis 23. index.html ning päis [22]

Ülejäänud oluline osa HTML-failis asub `<body>`-siltide alas (joonis 24). Direktiiviga `ngApp` määratakse rakenduse juurelement, mis kasutab moodulit `todomvc` kui oma juurmoodulit [118]. HTML-i kehas on esimesel real direktiivi `ngView` silt [119]. Selles asukohas esitatakse HTML-malli (*template*), millele viitab `$routeProvider` [119]. Järgnevates lõikudes keskendutakse ülejäänud koodile `<body>`-siltide alas.

```
/* ... */
    <body ng-app="todomvc">
        <ng-view />
                /* Kood, mis praegu välja jäetud. */
    </body>
</html>
```

Joonis 24. index.html ning keha [22]

Mallis identifikaatoriga `todomvc-index.html` asub `<section>`-siltide alas kolm järgnevate siltidega piirkonda: `<header>`, `<section>` ja `<footer>` (joonis 25). Neist esimeses on vorm direktiiviga `ngSubmit`, mis on seotud funtksiooniga `addTodo` [95]. Teine ja kolmas ala on nähtavad ainult tegevusobjektide eksiteerimise korral. Direktiiv `ngShow` kasutab CSS-peiteklassi `.ng-hide` lisamist ja kustutamist vastavalt sellega seotud atribuudi `$scope.todos.length` tõeväärtusele [120].

```
/* ... */
  <script type="text/ng-template" id="todomvc-index.html">
      <section id="todoapp">
          <header id="header">
              <h1>todos</h1>
              <form id="todo-form" ng-submit="addTodo()">
                /* ... */
              </form>
          </header>
          <section id="main" ng-show="todos.length" ng-cloak>
                /* ... */
          </section>
          <footer id="footer" ng-show="todos.length" ng-cloak>
                /* ... */
          </footer>
      </section>
      <footer id="info">
         /* Kood, mis esitab staatilist infot, ei kuulu selgitamisele. */
      </footer>
   </script>
  /* Vajalikud JavaScripti failide viited, ei kuulu selgitamisele. */
  /* ... */
```

Joonis 25. index.html [22]. Lisatud on kommentaare.

Malli `todomvc-index.html` sildi `<section>` alamsildi `<header>` alas asub tekstikast, kuhu kasutaja saab sisestada uue tegevuse objekti pealkirja (joonis 26). See on seotud direktiivi `ngModel` kaudu atribuudiga `$scope.newTodo` [90]. Lisaks on tekstikastile lisatud HTML5 automaatse fookuse atribuut ja direktiiv `ngDisabled`, mille kohta kirjutas bakalaureusetöö autor eelnevalt funktsiooni `addTodo` kohta käivas lõigus [99] [121].

```
/* ... */
<input id="new-todo" placeholder="What needs to be done?" ng-model="newTodo" ng-disabled="saving" autofocus>
/* ... */
```

Joonis 26. index.html ning uue tegevuse objekti sisestuskoht [22]

Malli `todomvc-index.html` sildi `<section>` alamsildi `<section>` alas paikneb märgistuskast, mis on direktiivi `ngModel` kaudu seotud atribuudiga `$scope.allChecked` (joonis 27) [90]. Märgistuskastile klõpsamine käivitab funktsiooni `markAll`. Elemendis identifikaatoriga `todo-list` esitatakse tsüklipõhimõttega direktiiviga `ngRepeat` atribuudi `statusFilter` järgi filtreeritud tegevusobjektide loetelu [122]. Iga tegevuse pealkirja juures olevast märgistuskastist saab vastava tegevuse tehtuks märkida või hoopis selle juures olevast kustutamisnupust ära kustutada. Kahekordsete loogeliste sulgude vahel on võimalik kuvada objekti `$scope` atribuutide väärtuseid kasutajale. Nimede redigeerimise vormi ja sellega seotud HTML-koodi sai selgitatud juba peatüki varasemas osas. Direktiiviga `ngClass` saab lisada või eemaldada CSS-klasse [123].

```
/* ... */
<input id="toggle-all" type="checkbox" ng-model="allChecked" ng-click="markAll(allChecked)">
<label for="toggle-all">Mark all as complete</label>
<ul id="todo-list">
  <li ng-repeat="todo in todos | filter:statusFilter track by $index" ng-class="{completed:
todo.completed, editing: todo == editedTodo}">
    <div class="view">
      <input class="toggle" type="checkbox" ng-model="todo.completed" ng-
change="toggleCompleted(todo)">
      <label ng-dblclick="editTodo(todo)">{{todo.title}}</label>
      <button class="destroy" ng-click="removeTodo(todo)"></button>
    </div>
    <form ng-submit="saveEdits(todo, 'submit')">
      <input class="edit" ng-trim="false" ng-model="todo.title" todo-
escape="revertEdits(todo)" ng-blur="saveEdits(todo, 'blur')" todo-focus="todo == editedTodo">
    </form>
  </li>
</ul>
/* ... */
```

Joonis 27. index.html ning loetelu [22]

Malli `todomvc-index.html` sildi `<section>` alamsildi `<footer>` alas kuvatakse kasutajale tegevusobjektide arvu (joonis 28). Selle arvu kõrval esitatakse direktiiviga `ngPluralize` ingliskeelse sõna „asi“ (*item*) või „asjad“ (*items*) korrektne vorm [124]. Loeteluna eksisteerivad lingid lubavad kasutajal filtreerida ja kuvada järgnevate tegevuste loetelusid: tehtuks märgituid, mittetehtuks märgituid või kõiki tegevusi. Kõikide tehtuks märgitud tegevuste kustutamise nupp on direktiivi `ngShow` tõttu nähtaval ainult siis, kui leidub selliseid tegevusi [120].

```
/* ... */
<span id="todo-count"><strong>{{remainingCount}}</strong>
                        <ng-pluralize count="remainingCount" when="{ one: 'item left', other:
'items left' }"></ng-pluralize>
</span>
<ul id="filters">
  <li>
    <a ng-class="{selected: status == ''} " href="#/">All</a>
  </li>
  <li>
    <a ng-class="{selected: status == 'active'}" href="#/active">Active</a>
  </li>
  <li>
    <a ng-class="{selected: status == 'completed'}" href="#/completed">Completed</a>
  </li>
  </ul>
  <button id="clear-completed" ng-click="clearCompletedTodos()" ng-show="completedCount">Clear
completed</button>
/* ... */
```

Joonis 28. index.html ning jalus [22]

Autor annab raamistikule AngularJS õppimise kategoorias null punkti, sest testrakenduse mõistmisele ning kirjalikul kujul selgitamisele kulus ligikaudu 15 tundi.

2.1.9 TodoMVC testrakendusel põhinev jõudlus

Lokaalses masinas käivitati testrakendus veebilehitsejas Chrome ning lasti YSlow pistikprogrammil hinnata jõudlust. Üldine hinne jõudluse eest oli 94 punkti 100 võimalikust punktist. YSlow soovitas kasutada sisuedastusvõrku, kokku pakkida gzip-meetodiga failid base.css ja index.css ning lisada Expires-päised eelmainitud CSS-failidele.

Autor annab TodoMVC testrakendusel põhineva jõudluse eest kolm punkti.

2.2 Backbone.js (teek)

Jeremy Ashkenase poolt loodud Backbone'i esimene väljalase (*release*) toimus 2010. aastal, kuid selle areng sai alguse juba varem rakenduse DocumentCloud osana [9, lk 1]. Backbone'i eesmärk on pakkuda arendajatele võimalust suurepäraselt struktureerida ees-

rakenduse (*client-side web app* või *front end*) andmemudelit [9, lk 1]. Backbone sõltub teegist Underscore.js ning osaliselt ka teegist jQuery [125].

2.2.1 Startimishinnang

Äridest kasutavad Backbone'i näiteks muusikajagamisplatform SoundCloud ning kaubandusfirma Walmart oma mobiilile loodud veebirakendustes, õppimisrakendus Code School, koodihaldussüsteem Atlassian Bitbucket ja koostöörakendus Trello [126].

Lehelt backbonejs.org saab alla laadida inimloetava arendusversiooni (*development version*), kompaktse tootmisversiooni (*production version*) või veel pooliku arendatava versiooni (*edge version*). Autor laadis alla inimloetavad arendusversioonid teekidest „Backbone.js 1.2.3", „Underscore.js 1.8.3" ja „jQuery JavaScript Library v2.2.0". Faili teremaailm.html (joonis 29) loomiseks sai kasutatud idee ja näitena Githubi kasutaja yanlum näitefaili [127]. Avades veebilehitsejas Chrome faili teremaailm.html, kuvatakse teksti „Tere 6" identifikaatorit `sisu` omava HTML-elemendi alas. Selleks läks vaja esmalt defineerida ning seejärel luua isend Backbone'i vaatest `Tekst`, mis kasutab mallina teegi Underscore.js funktsiooni `_.template` argumenti ja kus funktsioon `render` esitab lõpptulemuse, milles sisaldub ka arvutustehte vastus (joonis 29) [128] [129] [130].

```
<!DOCTYPE html>
<html>
    <head><title>Tere</title></head>
    <body>
      <script src="jquery.js" type="text/javascript"></script>
      <script src="underscore.js" type="text/javascript"></script>
      <script src="backbone.js" type="text/javascript"></script>
      <div id="sisu"></div>
      <script type="text/javascript">
        var Tekst = Backbone.View.extend({
            el: '#sisu',
            template: _.template("<div>Tere <%= arvutus %></div>"),
            initialize: function(){
                this.render();
            },
            render: function(){
                this.$el.html(this.template({arvutus: 2+2*2}));
            }
        });
        new Tekst();
      </script>
    </body>
</html>
```

Joonis 29. Backbone'i teremaailm.html allika [127] põhjal

Eeltoodu vastab hinde kaks saamise kriteeriumitele. Backbone'i puhul piisab oluliste failide (backbone.js, underscore.js ja jquery.js) allalaadimisest õigesse kausta ning veebilehitsejas käivitatavale failile viidete loomisest, kuid lihtsa lause väljastamise katsetamiseks on tarvis kulutada aega vaate `Backbone.View` toimimispõhimõtetega tutvumiseks. Seega tuleb teha üks lisategevus. Leidub vähemalt kolm vaadeldavat teeki kasutavat edukat ärifirmat. Need on Walmart, Trello ja Atlassian. Autor annab startimishinnangu kategoorias kaks punkti.

2.2.2 Kogukond ja dokumentatsioon

2016. aasta veebruarikuu seisuga on teegi Githubi lehel 1686 vaatajat, 24049 tähega märkijat ja 5377 koodi harutajat [131]. Stack Exchange'i lehelt otsides leiab 19338 Backbone.js-sildiga küsimust, millest vastamata on 4889 [132] [133].

Backbone'i kodulehel on vasakul ääres lingid, mis võimaldavad sirvida dokumentatsiooni [125]. Dokumentatsiooni sees eksisteerivad ka koodinäited.

Eeltoodu vastab hinde kaks saamise kriteeriumitele. Githubis olevate vaatajate, tähega märkijate ning koodi harutajate summa on 31112. Stack Exchange'is moodustavad mittevastatuks märgitud küsimused 25% kõigist raamistiku nime kandva sildiga küsimustest. Lisaks eksisteerib koodinäiteid sisaldav dokumentatsioon, kuid arendajal pole võimalik seda muuta või soovitada muudatusi. Autor annab kogukonna ja dokumentatsiooni kategoorias kaks punkti.

2.2.3 Arendustööriistade tugi

DaVinci Studio oli Eclipse'i laienduste otsingus ainuke vaste märksõnale „Backbone". See toetab raamistikke nagu jQuery Mobile, KnockOut, Underscore, Handlebars ja Backbone [134]. Pakutakse head projektihalduri (*project manager*) ja koodiredaktori funktsionaalsust. Viimane uuendus on 2016. aasta veebruarikuu seisuga tehtud 2014. aasta juunis. Pole täpselt kirjas, missugusel viisil DaVinci Studio just Backbone'i rakenduse arendamist paremaks muudaks.

Sublime Text 2 paketihaldurist võib leida pistikprogrammid nimedega Backbone.js ja Lazy Backbone.js [135] [136]. Mõlemaid on viimati uuendatud 3 aastat tagasi. Mõlemad pakuvad TAB-iga lõpule viimist (*tab completion*) ja koodiväljavõtteid (*snippets*). Esimesena nimetatud pistikprogrammi on viis korda rohkem installitud kui teisena nimetatut [135] [136].

Visual Studiol on sissehitatud mitmete eesrakenduste raamistike tugi [137]. Backbone'i nimi esineb toetatavate hulgas.

Autor annab arendustööriistade toe eest kaks punkti, sest toetatud on vähemalt kaks, kuid mitte vähemalt neli vaadeldavat arendustööriista. Jetbrainsi toodetele ja NetBeansile mõeldud Backbone'i pistikprogramm puudub.

2.2.4 Testimistugi

TodoMVC projekti üks eestvedaja Addy Osmani on oma Backbone'i kohta kirjutatud tasuta veebipõhisesse raamatusse kaasanud ühiktestimise peatüki [138]. Seal on põhjalikud selgitused koos paljude koodinäidetega testimisraamistike Jasmine, QUnit ja SinonJS kohta.

Backbone'iga loodud rakenduse testimist seletav materjal koos koodinäidistega ekisteerib. Autor annab testimistoe kriteeriumi eest kolm punkti.

2.2.5 Litsents äri vaatepunktist

Teegi Backbone.js puhul kehtib MIT litsents [125]. Backbone'i autoriõiguse hoidjad Jeremy Ashkenas ning DocumentCloud ei vastuta raamistiku kasutamisega kaasneda võiva kahju eest [139]. Litsentsis oleva autoriõiguse märkus ja litsentsi teadaanne peab eksisteerima kõigis koopiates või olulises mahus teegiga Backbone.js loodavas tarkvaras [75].

Vastava teegi abil loodud rakendust lubatakse äriks kasutada, rakenduse lähtekood võib jääda salajaseks. Backbone.js saab litsentsi eest äri vaatepunktist kolm punkti.

2.2.6 Keele ja vormingu tugi

Backbone'i dokumentatsioonis puuduvad juhised keelte toe ning kuupäeva, numbri ja valuuta formaatide toe kohta. Firma Airbnb on loonud teegi Polyglot.js, mis võimaldab tagada mitme keele toe tõlgitud fraaside vastavusse seadmise funktsionaalsusega [102]. Airbnb on kasutanud Polygloti loomisel enda i18n funktsionaalsuse lisamise kogemust Backbone'i rakendustesse [102]. Bakalaureusetöö autor leidis, et JavaScripti programmide lokali-

seerimist võimaldab teek LocalePlanet [140]. Lisaks eksisteerib internatsionaliseerimise ning lokaliseerimise jaoks loodud JavaScripti teek globalize [141]. JavaScripti kogukonna poolt on loodud ka globaliseerimisfarm ehk erinevate mitme keele ja formaadi tuge pakkuvate raamisitke/teekide võrdlus [142].

Airbnb küll mainis Polygloti seotust Backbone'iga, kuid konkreetselt vaadeldavale teegile loodud keele ja vormingu tuge pakkuv tarkvara puudub või on raskesti leitav. Autor annab keele ja vormingu toe kriteeriumi eest null punkti.

2.2.7 TodoMVC testrakendusel põhinev keerukus

Vastavalt üldkriteeriumitele sai hinnatud testrakenduse JavaScripti faile, mis on esindatud alapunktis 2.2.8 [23]. Tulemused on toodud allpool.

1. Lähtekoodi loogiliste ridade arv kokku liidetult paikneb tabelis 4.

Tabel 4. Backbone'i testrakenduse ridade arvu kategooria.

Fail	Loogiliste ridade arv (*Logical LOC*)	Summa
app.js	6	167
todos.js	17	
todo.js	11	
router.js	12	
app-view.js	66	
todo-view.js	55	

2. McCabe'i tsüklomaatiline keerukuse (*Cyclomatic complexity*) keskmine (liidetakse kokku iga vaadeldava JavaScripti faili tsüklomaatilise keerukuse arv ning jagatakse vaadeldavate failide arvuga) paikneb tabelis 5.

Tabel 5. Backbone'i testrakenduse tsüklomaatilise keerukuse kategooria.

Fail	Tsüklomaatiline keerukus (*Cyclomatic complexity*)	Keskmine
app.js	2	3, 83
todos.js	3	
todo.js	2	
router.js	3	
app-view.js	5	
todo-view.js	8	

3. Halstead'i jõupingutuste (*effort*) keskmine funktsioonide põhjal (*Mean per-function Halstead effort*) oli 462, 4561440259143.
4. Hooldatavuse indeks (liidetakse kokku iga vaadeldava JavaScripti faili indeks ja jagatakse see vaadeldavate failide arvuga) paikneb tabelis 6.

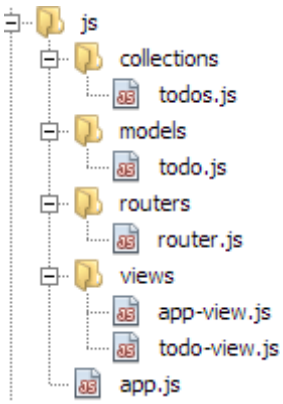
Tabel 6. Backbone'i testrakenduse hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**
app.js	151, 38573714199626	130, 14
todos.js	130, 43273052428242	
todo.js	129, 40635189784538	
router.js	125, 3388728119731	
app-view.js	120, 24330743580023	
todo-view.js	124, 05643762498195	

Backbone'iga loodud testrakendus kuulub testrakenduse ridade arvu kategoorias 8.–9. kohale (lisa „I. Keerukuse pingeread", tabel 20), tsüklomaatilise keerukuse kategoorias 7. kohale (lisa „I. Keerukuse pingeread", tabel 21), hooldatavuse indeksi kategoorias 9. kohale (lisa „I. Keerukuse pingeread", tabel 22) ning funktsioonide põhjal saadud Halstead'i jõupingutuste (*effort*) keskmise kategoorias 7. kohale (lisa „I. Keerukuse pingeread", tabel 23). Backbone.js saab testrakendusel põhineva keerukuse kategoorias ühe punkti, sest tulemuseks on neljas pingereas esikümnesse kuulumine.

2.2.8 TodoMVC testrakendusel põhinev õppimine ja õppimiskiirus

Järgneva MIT litsentsiga TodoMVC projekti testrakenduse autor on Addy Osmani (lisa „III. TodoMVC projekti repositooriumi sisu kasutamise litsents") [23]. Failiga index.html asub samas kaustas kaust nimega `js`, mille sisu on toodud joonisel 30.



Joonis 30. Kausta `js` sisu [23]

Failis app.js kasutatakse jQuery teegi `ready()` meetodit, mis ootab, kuni DOM on laaditud, et seejärel luua failis app-view.js defineeritud vaade (joonis 31) [143]. Lisaks juhib autor tähelepanu joonisel 31 olevale reale `var app = app ||{}`, mis loob sama ni-

meruumi kõigile seda omavatele JavaScripti failidele ja lubab neid faile suvalises järjekorras laadida [144].

```
/*global $ */
/*jshint unused:false */
var app = app || {};
var ENTER_KEY = 13;
var ESC_KEY = 27;

$(function () {
    'use strict';
    // kick things off by creating the `App`
    new app.AppView();
});
```

Joonis 31. app.js [23]

Testrakenduse kaks vaadet on defineeritud failides app-view.js ning todo-view.js. Tegevuse (*todo*) objekti mudel paikneb failis todo.js (joonis 32). Seal luuakse uus mudeli klass `Todo` meetodiga `extend` laiendades mudelit `Backbone.Model` [145] [146]. Iga käesoleva mudeli isend saab objektliteraali (*object literal*) `defaults` kaudu endale vaikimisi tühja sõne väärtusega pealkirjaatribuudi ja tõeväärtusatribuudi, mis tähistab tegevuse mittetehtud olekut [147] [9, lk 40]. Mudeli sees on defineeritud lisaks funktsioon `toggle`, mis muudab atribuudi `completed` tõeväärtuse vastupidiseks. Muudatus salvestub meetodiga `save`, mis üldjuhul rakendab omakorda funktsiooni `Backbone.sync` [148] [149]. Käesolev testrakendus kasutab salvestuskohana veebilehitseja andmehoidu (HTML5 Local Storage), sest projekti sõltuvuste failis package.json kirja pandud adapteri `backbone.localstorage` funktsionaalsus asendab funktsiooni `Backbone.sync` [21] [150].

```
/*global Backbone */
var app = app || {};

(function () {
    'use strict';

    // Todo Model
    // ----------
    // Our basic **Todo** model has `title`, `order`, and `completed` attributes.
    app.Todo = Backbone.Model.extend({
        // Default attributes for the todo
        // and ensure that each todo created has `title` and `completed` keys.
        defaults: {
            title: '',
            completed: false
        },
        // Toggle the `completed` state of this todo item.
        toggle: function () {
            this.save({
                completed: !this.get('completed')
            });
        }
    });
})();
```

Joonis 32. todo.js [23]

Faili todos.js (joonis 33) koodis on loodud mudeli `Todo` põhjal kollektsioon `Backbone.Collection` ehk järjestatud mudelite hulk [151]. Atribuudi `model` kaudu defineeritakse mudeli `Todo` sisaldumine kollektsioonis `Todos` [152]. Atribuut `localstorage` on adapteri `backbone.localstorage` isend andmete salvestamiseks veebilehitseja andmehoidu [150].

```
/*global Backbone */
var app = app || {};

(function () {
    'use strict';

    // Todo Collection
    // ---------------
    // The collection of todos is backed by *localStorage* instead of a remote
    // server.
    var Todos = Backbone.Collection.extend({
        // Reference to this collection's model.
        model: app.Todo,

        // Save all of the todo items under this example's namespace.
        localStorage: new Backbone.LocalStorage('todos-backbone'),
/* ... */
```

Joonis 33. todos.js ning kollektsiooni `Todos` defineerimine [23]

Funktsioonid `completed` ja `remaining` sisaldavad mõlemad meetodi `where` kollektsioonile rakendamist [153]. Esimene tagastab JavaScripti loetelurivi (*array*) kõikidest mudelitest, mille puhul tegevus on märgitud tehtuks. Teine funktsioon tagastab JavaScripti loetelurivi kõikidest mudelitest, mille puhul tegevus on märgitud mittetehtuks.

```
/* ... */
        // Filter down the list of all todo items that are finished.
        completed: function () {
            return this.where({completed: true});
        },
        // Filter down the list to only todo items that are still not finished.
        remaining: function () {
            return this.where({completed: false});
        },
/* ... */
```

Joonis 34. todos.js ning funktsioonid `completed` ja `remaining` [23]

Kui kollektsioonis ei eksisteeri tegevusi, siis tagastab funktsioon `nextOrder` (joonis 35) järjekorranumbri 1. Vastasel juhul tagastatakse number, mis on kollektsiooni viimase elemendi järjekorranumbrist ühe võrra suurem. Mudeli `Todo` definitsioonis pole kirjas midagi atribuudi `order` kohta. See lisatakse mudeli objektile failis app-view.js defineeritud funktsiooniga `newAttributes`. Faili todos.js lõpus (joonis 35) kehtestatakse kollektsiooni sorteerimine atribuudi `order` järgi `comparator`-konstruktsiooniga ning luuakse globaalselt uus kollektsiooni isend käsuga `new Todos()` [154].

```
/* ... */
        // We keep the Todos in sequential order, despite being saved by unordered
        // GUID in the database. This generates the next order number for new items.
        nextOrder: function () {
            return this.length ? this.last().get('order') + 1 : 1;
        },
        // Todos are sorted by their original insertion order.
        comparator: 'order'
    });
    // Create our global collection of **Todos**.
    app.todos = new Todos();
})();
```

Joonis 35. todos.js ning kollektsiooniisendi loomine [23]

Failis router.js (joonis 36) luuakse klassi `Backbone.Router` laiendamisega meetodiga `extend` URL-haldaja klass `TodoRouter` [155]. Vastavused seatakse atribuudile `routes` vastavas objektis: tärniga tähistatud `*filters` tähendab, et mistahes URL-komponendi puhul kujuga `…#/` kutsutakse välja funktsioon `setFilter` pärast seda URL-kuju kirjutatud argumentsõnega [156]. Käesoleva funktsiooni sees seatakse kasutatav filtri `app.TodoFilter` väärtuseks argumendi väärtus või tühi sõne ja käivitatakse

tagasikutse (*callback*) sündmustemooduli `Backbone.Events` meetodiga `trigger` sündmus `filter` [157]. Faili router.js lõpus luuakse käsuga `new TodoRouter()` URL-haldaja isend ning alustatakse URL-i muutuste jälgimist kutsega `Backbone.history.start()` [158].

```
/*global Backbone */
var app = app || {};
(function () {
    'use strict';
    var TodoRouter = Backbone.Router.extend({
        routes: {
            '*filter': 'setFilter'
        },
        setFilter: function (param) {
            // Set the current filter to be used
            app.TodoFilter = param || '';

            // Trigger a collection filter event, causing hiding/unhiding
            // of Todo view items
            app.todos.trigger('filter');
        }
    });
    app.TodoRouter = new TodoRouter();
    Backbone.history.start();
})();
```

Joonis 36. router.js [23]

Failides router.js, todos.js ja todo.js asub kood joonisel 37 esitatud JavaScripti moodulimustri (*JavaScript Modul Pattern*) anonüümse sulundi (*closure*) konstruktsiooni sees [159]. Selle eesmärgiks on rakenduse eluea (*application lifetime*) jooksul anonüümse funktsiooni sees pakkuda koodile privaatsust ja oma olekut (*state*).

```
var app = app || {};

(function () {
    'use strict';
        /* ... */
        //..Kood.//
        /* ... */
})();
```

Joonis 37. router.js, todos.js, todo.js ning moodulimustri anonüümse sulundi konstruktsioon [23]

Failides app-view.js ja todo-view.js asuvad vaadete `AppView` ja `TodoView` koodilõigud joonisel 38 esitatud JavaScripti moodulimustri globaalse impordi (*global import*) konstruktsiooni sees [159]. See võimaldab globaalset jQueryt kasutada anonüümse funktsiooni sees viite `$` kaudu: globaalsete muutujate kasutus muutub koodi lugeja jaoks lihtsamini mõistetavaks.

```
/*global Backbone, jQuery, _, ENTER_KEY */
var app = app || {};

(function ($) {
    'use strict';
        /* ... */
        // KOOD, MIDA AUTOR VAATAB HILJEM
        /* ... */

})(jQuery);
```

Joonis 38. app-view.js, todo-view.js ning moodulimustri globaalse impordi konstruktsioon [23]

Vaade `AppView` luuakse funtksiooniga `Backbone.View.extend()` ning seotakse atribuudi `el` kaudu failis index.html oleva HTML-elemendiga, mille klass on `todoapp` (joonis 39) [128] [160]. Atribuudiga `statsTemplate` võetakse kasutusele teegi Underscore.js mall, mille identifikaator `id` failis index.html on `stats-template` [129].

Vaate sündmuste atribuut `events` seob dokumendiobjektide mudeli sündmused kindlate meetoditega läbi jQuery funktsiooni `on` kasutava funktsiooni `delegateEvents` [161] [162]. Kui failis index.html oleval HTML-elemendil klassiga `new-todo` esineb klahvi alla vajutamise sündmus, siis kutsutakse välja funktsioon `createOnEnter`. Kui aga klõpsatakse HTML-elementi klassiga `clear-completed` või `toggle-all`, siis käivitatakse vastavalt funktsioonid `clearCompleted` ja `toggleAllComplete` [163].

```
/* ... */
    // The Application
    // ---------------
    // Our overall **AppView** is the top-level piece of UI.
    app.AppView = Backbone.View.extend({
        // Instead of generating a new element, bind to the existing skeleton of
        // the App already present in the HTML.
        el: '.todoapp',
        // Our template for the line of statistics at the bottom of the app.
        statsTemplate: _.template($('#stats-template').html()),
        // Delegated events for creating new items, and clearing completed ones.
        events: {
            'keypress .new-todo': 'createOnEnter',
            'click .clear-completed': 'clearCompleted',

            'click .toggle-all': 'toggleAllComplete'
        },
/* ... */
```

Joonis 39. app-view.js ning `el`, `statsTemplate` ja `events` [23]

Vaate funktsioon `initialize` kutsutakse välja vaate loomise ajal [164]. Selle sees salvestatakse edaspidise mugava viitamise eesmärgil vahemälu jQuery objektid (*cached jQuery objects*), mis saadakse jQuery kaudu vaate elemendist `el` lähtuvatelt HTML-elementidelt klassidega `toggle-all`, `new-todo`, `footer`, `main` ja `todo-list` [165] [166]. Seejärel pannakse funktsiooniga `listenTo` käesolev vaade jälgima mudelite kollektsiooni `app.todos` sündmust (*event*) `add` [167]. Kui sündmus toimub, siis tehakse tagasikutse vaate kontekstis vaates defineeritud funktsiooniga `addOne` [167]. Analoogiliselt seotakse vaadeldav kollektsioon järgmistel ridadel veel mitmete sündmustega. Sündmuse `all` sidumise puhul kasutatakse teegi Underscore.js funktsiooni `debounce`, millega saab parendada rakenduse jõudlust pideva taasesitamise (*re-render*) mõju vähendamisega [168] [169]. Joonisel 40 esitatud koodi lõpus ammutatakse salvestatud tegevuste objektid funktsiooniga `fetch` ning pannakse kollektsiooni `todos`, mida argumendi `{reset: true}` tõttu uuendatakse täies ulatuses ning välditakse vaate taasesitamist iga üksiku mudeli puhul [170] [171].

```
        // At initialization we bind to the relevant events on the `Todos`
        // collection, when items are added or changed. Kick things off by
        // loading any preexisting todos that might be saved in *localStorage*.
        initialize: function () {
            this.allCheckbox = this.$('.toggle-all')[0];
            this.$input = this.$('.new-todo');
            this.$footer = this.$('.footer');
            this.$main = this.$('.main');
            this.$list = $('.todo-list');
            this.listenTo(app.todos, 'add', this.addOne);
            this.listenTo(app.todos, 'reset', this.addAll);
            this.listenTo(app.todos, 'change:completed', this.filterOne);
            this.listenTo(app.todos, 'filter', this.filterAll);
            this.listenTo(app.todos, 'all', _.debounce(this.render, 0));
            // Suppresses 'add' events with {reset: true} and prevents the app view
            // from being re-rendered for every model. Only renders when the 'reset'
            // event is triggered at the end of the fetch.
            app.todos.fetch({reset: true});
        },
/* ... */
```

Joonis 40. app-view.js ning `initialize` [23]

Backbone'i vaate funktsiooni `render` ülekirjutamine (joonis 41) tagab, et muudatuste korral uueneks kasutajale nähtav vaade [130]. Juhul, kui kollektsioonis on mudeleid, muu-

detakse nähtavaks `$main` ja `$footer` poolt viidatavad elemendid jQuery meetodiga `show` [172]. Lisaks saavad väärtustatud käesolevale vaatele vastavas failis index.html asuvas mallis atribuudid `completed` ja `remaining` ja uus HTML-kood esitatakse jQuery meetodi `html` abil [130] [173]. Klassi `filters` omava HTML-loetelu `<a>`-sildiga elementidelt eemaldatakse klass `selected` ning rakendatakse filteerimise eesmärgiga failis router.js defineeritud filtrit `app.TodoFilter` ja siis jQuery filtrit ning lisatakse klass `selected` [174] [175] [176]. Juhul, kui kollektsioonis pole mudeleid, peidetakse `$main` ja `$footer` poolt viidatavad elemendid jQuery meetodiga `hide` [177]. Funktsiooni `render` lõpus uuendatakse atribuudi `allCheckbox` väärtust.

```
/* ... */
        // Re-rendering the App just means refreshing the statistics -- the rest
        // of the app doesn't change.
        render: function () {
            var completed = app.todos.completed().length;
            var remaining = app.todos.remaining().length;
            if (app.todos.length) {
                this.$main.show();
                this.$footer.show();
                this.$footer.html(this.statsTemplate({
                    completed: completed,
                    remaining: remaining
                }));
                this.$('.filters li a')
                    .removeClass('selected')
                    .filter('[href="#/' + (app.TodoFilter || '') + '"]')
                    .addClass('selected');
            } else {
                this.$main.hide();
                this.$footer.hide();
            }
            this.allCheckbox.checked = !remaining;
        },
/* ... */
```

Joonis 41. app-view.js ning `render` [23]

Funktsioon `addOne` (joonis 42) loob uue isendi vaatest `TodoView`, mille definitsioon on failis todo-view.js. Vaate `TodoView` funktsiooniga `render` saadud HTML-kuju lisatakse vaates `AppView` viite `$list` poolt viidatavale elemendile. Samal joonisel funktsioon `addAll` tühjendab `$list` poolt viidatava HTML-elemendi sisu ning rakendab funktsiooni `addOne` kõigile kollektsiooni `app.todos` mudeliobjektidele [178].

```
/* ... */
        // Add a single todo item to the list by creating a view for it, and
        // appending its element to the `<ul>`.
        addOne: function (todo) {
            var view = new app.TodoView({ model: todo });
            this.$list.append(view.render().el);
        },
        // Add all items in the **Todos** collection at once.
        addAll: function () {
            this.$list.html('');
            app.todos.each(this.addOne, this);
        },
```

Joonis 42. app-view.js ning `addOne` ja `addAll` [23]

Funktsioon `filterOne` käivitab argumendiks saadud mudeliga seotud sündmuse `visible` (joonis 43) [157]. Samal joonisel `filterAll` rakendab funktsiooni `filterOne` kõigile kollektsiooni `app.todos` mudelitele [178].

```
/* ... */
        filterOne: function (todo) {
            todo.trigger('visible');
        },
        filterAll: function () {
            app.todos.each(this.filterOne, this);
        },
/* ... */
```

Joonis 43. app-view.js ning `filterOne` ja `filterAll` [23]

Funktsioon `newAttributes` tagastab objekti kolme atribuudiga (joonis 44). Atribuudi `title` väärtus on `$input` poolt viidatava elemendi väärtus: sõne, millelt on JavaScripti meetodiga `trim` eemaldatud mõlemast äärest tühikud [91]. Atribuudile `order` annab väärtuse kollektsioonile `app.todos` rakendatud failis todos.js defineeritud funktsioon `nextOrder`. Atribuut `completed` on väära tõeväärtusega.

```
/* ... */
        // Generate the attributes for a new Todo item.
        newAttributes: function () {
            return {
                title: this.$input.val().trim(),
                order: app.todos.nextOrder(),
                completed: false
            };
        },
/* ... */
```

Joonis 44. app-view.js ning `newAttributes` [23]

Joonisel 45 esitatud funktsioonis `createOnEnter` kontrollitakse, kas argumendiks saadud sündmus oli klahvi Enter vajutamine ning kas `$input` poolt viidatava elemendi üleliigsetest tühikutest puhastatud sõne pikkus on suurem nullist [161]. Kui eelnev tingimuskontroll on positiivselt täidetud, siis luuakse kollektsiooni `app.todos` uus tegevuse mudeliobjekt funktsiooniga `create`, mille argumendiks on funktsiooni `newAttributes` poolt tagastatud kolme atribuudiga objekt. Lisaks omistatakse `$input` poolt viidatavale elemendile väärtuseks tühi sõne ehk viidatavast tekstikastist kustutatakse tekst.

```
/* ... */
        // If you hit return in the main input field, create new **Todo** model,
        // persisting it to *localStorage*.
        createOnEnter: function (e) {
            if (e.which === ENTER_KEY && this.$input.val().trim()) {
                app.todos.create(this.newAttributes());
                this.$input.val('');
            }
        },
/* ... */
```

Joonis 45. app-view.js ning `createOnEnter` [23]

Funktsioon `clearCompleted` kustutab ära tegevuste mudeliobjektid, mille atribuudi `completed` väärtus on tõene ehk mis on märgitud tehtuks (joonis 46). Selleks kasutatakse teegi Underscore.js funktsiooni `invoke`, mis rakendab igale tingimust rahuldavale mudelile funktsiooni `destroy`, mis kustutab mudeli [179] [180].

```
/* ... */
        // Clear all completed todo items, destroying their models.
        clearCompleted: function () {
            _.invoke(app.todos.completed(), 'destroy');
            return false;
        },
/* ... */
```

Joonis 46. app-view.js ning `clearCompleted` [23]

Funktsioon `toggleAllComplete` kasutab funktsiooni `save` mudeliobjekti atribuudi väärtuse muudatuse salvestamiseks (joonis 46) [148]. Funktsiooni töö tulemusena saavad kõik kollektsioonis olevad mudelid atribuudi `completed` väärtuseks korraga väära või

tõese tõeväärtuse: indikaatorina kasutatakse tõeväärtust `allCheckbox.checked`, mille abil saab teada, kas kõigi kollektsioonis olevate mudelite atribuut `completed` on tõene.

```
/* ... */
        toggleAllComplete: function () {
            var completed = this.allCheckbox.checked;
            app.todos.each(function (todo) {
                todo.save({
                    completed: completed
                });
            });
        }
    });
/* ... */
```

Joonis 47. app-view.js ning `toggleAllComplete` [23]

Eelnevalt (joonis 42) loodi vaate `AppView` funktsioonis `addOne` vaate `TodoView` isend. Vaade `TodoView` defineeritakse failis todo-view.js funktsiooniga `Backbone.View.extend` (joonis 48). Atribuudiga `tagName` antakse instruktsioon vaatele vastav HTML-kood paigutada loendielemendi `<li>`-siltide vahele [9, lk 213] [128] [160]. Atribuudiga `template` võetakse kasutusele teegi Underscore.js mall, mille identifikaator `id` failis index.html on `item-template` [129] . Vaate sündmuste atribuut `events` seob dokumendiobjektide mudeli sündmused kindlate meetoditega [161]. Vastavusse seadmisel jälgitakse HTML-koodis esinevaid elemente klassidega `toggle`, `destroy` ja `edit` ning elementi sildiga `label`. Reageeritakse klõpsamise (`click`), topeltklõpsamise (`dblclick`), klahvivajutuse (`keypress`), klahvi alla vajutamise (`keydown`) ja elemendi fookuse kaotamise (`blur`) peale[10]. Vastavusse seatud funktsioonid on `toggleCompleted`, `edit`, `clear`, `updateOnEnter`, `revertOnEscape` ja `close`.

```
/* ... */
    // Todo Item View
    // --------------
    // The DOM element for a todo item...
    app.TodoView = Backbone.View.extend({
        //... is a list tag.
        tagName:  'li',
        // Cache the template function for a single item.
        template: _.template($('#item-template').html()),
        // The DOM events specific to an item.
        events: {
            'click .toggle': 'toggleCompleted',
            'dblclick label': 'edit',
            'click .destroy': 'clear',
            'keypress .edit': 'updateOnEnter',
            'keydown .edit': 'revertOnEscape',
            'blur .edit': 'close'
        },
/* ... */
```

Joonis 48. todo-view.js ning `tagName`, `template` ja `events` [23]

Vaate loomise ajal välja kutsutavas funktsioonis `initialize` pannakse vaade funktsiooniga `listenTo` jälgima mudlei `Todo` sündmusi `change`, `destroy` ja `visible` (joonis 49) [164] [167]. Vaate kontekstis tehakse vastavalt tagasikutsed `render`, `remove` ja `toggleVisible`: neist teine eemaldab vaate koos oma juurelemendiga DOM-ist ning lõpetab mudeli puhul sündmuste jälgimise funktsiooniga `stopListening` [182] [183].

[10] [96] [97] [163] [181]

```
/* ... */
        // The TodoView listens for changes to its model, re-rendering. Since
        // there's a one-to-one correspondence between a **Todo** and a
        // **TodoView** in this app, we set a direct reference on the model for
        // convenience.
        initialize: function () {
            this.listenTo(this.model, 'change', this.render);
            this.listenTo(this.model, 'destroy', this.remove);
            this.listenTo(this.model, 'visible', this.toggleVisible);
        },
/* ... */
```

Joonis 49. todo-view.js ning `initialize` [23]

Funktsiooni `render` alguses lahendatakse üleliigne vaate esitamise (*render*) olukord tingimuskontrolliga (joonis 50). Koodis asuvas kommentaaris viidatakse mainitud tingimuskontrolli vajalikkusele, sest Backbone ja veebilehitseja andmehoid teevad halvasti koostööd seoses atribuudi `id` muutustega. [184]

```
/* ... */
        // Re-render the titles of the todo item.
        render: function () {
            // Backbone LocalStorage is adding `id` attribute instantly after
            // creating a model.  This causes our TodoView to render twice. Once
            // after creating a model and once on `id` change.  We want to
            // filter out the second redundant render, which is caused by this
            // `id` change.  It's known Backbone LocalStorage bug, therefore
            // we've to create a workaround.
            // https://github.com/tastejs/todomvc/issues/469
            if (this.model.changed.id !== undefined) {
                return;
            }
/* ... */
```

Joonis 50. todo-view.js ning `render` [23]

Funktsiooni `render` kood jätkub joonisel 51. Seal kasutatakse mudeli atribuute, et läbi malli esitada vaates mudelist pärineva infoga HTML-koodi. Teegi jQuery abil lisatakse või eemaldatakse HTML-elemendilt klass `completed` vastavalt sellele, kas mudeli atribuut `completed` on tõese või väära tõeväärtusega [185]. Liskas kutsutakse välja funktsioon `toggleVisible` ning jäetakse meelde jQuery viide HTML-elemendile `edit` [165] [166].

```
/* ... */
            this.$el.html(this.template(this.model.toJSON()));
            this.$el.toggleClass('completed', this.model.get('completed'));
            this.toggleVisible();
            this.$input = this.$('.edit');
            return this;
        },
/* ... */
```

Joonis 51. todo-view.js ning lõpp funktsioonist `render` [23]

Joonisel 52 esitatud funktsiooni `isHidden` kaudu saadakse tõeväärtus, mida kasutab funktsioon `toggleVisible`, mis lisab või eemaldab HTML-koodis käesoleva vaate `<li>`-sildiga juurelemendi puhul klassi `hidden` [185]. Funktsioon `toggleCompleted` kutsub välja mudelis `Todo` defineeritud funktsiooni `toggle`.

```
/* ... */
        toggleVisible: function () {
            this.$el.toggleClass('hidden', this.isHidden());
        },
        isHidden: function () {
            return this.model.get('completed') ?
                app.TodoFilter === 'active' :
                app.TodoFilter === 'completed';
        },
```

```
        // Toggle the `"completed"` state of the model.
        toggleCompleted: function () {
            this.model.toggle();
        },
/* ... */
```

Joonis 52. todo-view.js ning `toggleVisible`, `isHidden` ja `toggleCompleted` [23]

Kui `$input` poolt viidatav HTML-element on kaotanud fookuse, siis funktsioon `close` võtab sellele elemendile vastavast sisust sõne, kust on eemaldatud üleliigsed tühikud ja kontrollib klassi `editing` olemasolu käesoleva vaate `<li>`-sildiga elemendil jQuery funktsiooniga `hasClass` [186]. Kui vastav klass eksiteerib ning sõne pikkus on suurem nullist, siis salvestatakse tekst mudeliobjekti pealkirjaks ehk atribuudi `title` väärtuseks. Tühja sõne puhul rakendatakse vaatele funktsiooni `clear`, millest autor kirjutab edaspidises tekstis. Lõpuks eemaldatakse klass `editing` vaate `<li>`-sildiga elemendilt.

```
/* ... */
        // Close the `"editing"` mode, saving changes to the todo.
        close: function () {
            var value = this.$input.val();
            var trimmedValue = value.trim();
            // We don't want to handle blur events from an item that is no
            // longer being edited. Relying on the CSS class here has the
            // benefit of us not having to maintain state in the DOM and the
            // JavaScript logic.
            if (!this.$el.hasClass('editing')) {
                return;
            }
            if (trimmedValue) {
                this.model.save({ title: trimmedValue });
            } else {
                this.clear();
            }
            this.$el.removeClass('editing');
        },
/* ... */
```

Joonis 53. todo-view.js ning `close` [23]

Funktsioon `updateOnEnter` kontrollib, kas kasutaja vajutas klahvi Enter (joonis 54). Selle tingimuse täitmise järel rakendatakse eelnevas lõigus käsitletud funktsiooni `close`.

```
/* ... */
        // If you hit `enter`, we're through editing the item.
        updateOnEnter: function (e) {
            if (e.which === ENTER_KEY) {
                this.close();
            }
        },
/* ... */
```

Joonis 54. todo-view.js ning `updateOnEnter` [23]

Funktsioon `revertOnEscape` kontrollib, kas kasutaja vajutas klahvi Escape (joonis 55). Selle tingimuse täitmise järel eemaldatakse vaate `<li>`-sildiga elemendilt klass `editing` ja taastatakse mudeliobjekti pealkirja atribuudi väärtuse järgi kasutajale redigeerimisel kuvatav tekst `$input` poolt viidataval HTML-elemendil.

```
/* ... */
        // If you're pressing `escape` we revert your change by simply leaving
        // the `editing` state.
        revertOnEscape: function (e) {
            if (e.which === ESC_KEY) {
                this.$el.removeClass('editing');
                // Also reset the hidden input back to the original value.
                this.$input.val(this.model.get('title'));
            }
        },
/* ... */
```

Joonis 55. todo-view.js ning `revertOnEscape` [23]

Topeltklõpsu toimumise tõttu käivituv funktsioon `edit` lisab vaate `<li>`-sildiga elemendile klassi `editing` ning toob `$input` poolt viidatavale HTML-elemendile fookuse meetodiga `focus` (joonis 56) [99]. Funktsioon `clear` rakendab mudelile funktsiooni `destroy`, mis kustutab selle mudeli [180].

```
/* ... */
        // Switch this view into `"editing"` mode, displaying the input field.
        edit: function () {
            this.$el.addClass('editing');
            this.$input.focus();
        },
        // Remove the item, destroy the model from *localStorage* and delete its view.
        clear: function () {
            this.model.destroy();
        }
/* ... */
```

Joonis 56. todo-view.js ning `edit` ja `clear` [23]

Faili index.html alguses viidatakse stiililehtedele (joonis 57). Sama faili kehas ehk `<body>`-siltide ala lõpuosas viidatakse kasutatavatele JavaScripti failidele. Lisaks leidub seal ka jalus tekstilise infoga testrakenduse autori Addy Osmani kohta ning järgnevates lõikudes käsitletav kood.

```
<!doctype html>
<html lang="en" data-framework="backbonejs">
    <head>
        <meta charset="utf-8">
        <title>Backbone.js • TodoMVC</title>
        <link rel="stylesheet" href="node_modules/todomvc-common/base.css">
        <link rel="stylesheet" href="node_modules/todomvc-app-css/index.css">
    </head>
/* ... */
```

Joonis 57. index.html ning päis [23]

Joonisel 58 esitatud HTML-koodis on olemas komponendid, millele failis app-view.js viidati. Backbone koos teegi Underscore.js mallidega võimaldas vaadeldava osa kirja panna lihtsat HTML-märgendkeelt kasutades. Olemas on koht, kuhu kasutaja saab sisestada tegevuse pealkirja. Lisaks eksisteerivad jalus, loetelu `<ul>`−siltide ala ja märgendkast kõigi tegevuste märgistamiseks.

```
/* ... */
        <section class="todoapp">
            <header class="header">
                <h1>todos</h1>
                <input class="new-todo" placeholder="What needs to be done?" autofocus>
            </header>
            <section class="main">
                <input class="toggle-all" id="toggle-all" type="checkbox">
                <label for="toggle-all">Mark all as complete</label>
                <ul class="todo-list"></ul>
            </section>
            <footer class="footer"></footer>
        </section>
/* ... */
```

Joonis 58. index.html HTML-kood klassiga `todoapp` [23]

Vaates `AppView` kasutati malli `stats-template` ning vaates `TodoView` kasutati malli `item-template` (joonis 59). Mallide HTML-failis esitatud kujudes on näha, kuidas märkide `<% %>` vahel saab hoida JavaScripti koodi abil väärtuse saavaid muutujaid ning tingimuskontrolle. Ülejäänud kood kujutab endast tavalist HTML-koodi.

```
/* ... */
   <script type="text/template" id="item-template">
      <div class="view">
        <input class="toggle" type="checkbox" <%= completed ? 'checked' : '' %>>
        <label><%- title %></label>
        <button class="destroy"></button>
      </div>
      <input class="edit" value="<%- title %>">
    </script>
    <script type="text/template" id="stats-template">
      <span class="todo-count"><strong><%= remaining %></strong> <%= remaining === 1 ?
'item' : 'items' %> left</span>
      <ul class="filters">
        <li>
          <a class="selected" href="#/">All</a>
        </li>
        <li>
          <a href="#/active">Active</a>
        </li>
        <li>
          <a href="#/completed">Completed</a>
        </li>
      </ul>
      <% if (completed) { %>
      <button class="clear-completed">Clear completed</button>
      <% } %>
    </script>
/* ... */
```

Joonis 59. index.html ning kaks malli [23]

Autor annab Backbone'ile õppimise kategoorias null punkti, sest testrakenduse mõistmisele ning kirjalikul kujul selgitamisele kulus ligikaudu 16 tundi.

2.2.9 TodoMVC testrakendusel põhinev jõudlus

Lokaalses masinas käivitati testrakendus veebilehitsejas Chrome ning lasti YSlow pistikprogrammil hinnata jõudlust. Üldine hinne jõudluse eest oli 77 punkti 100 võimalikust punktist. YSlow soovitas kasutada sisuedastusvõrku, kokku pakkida gzip-meetodiga failid base.css, index.css, base.js, underscore.js, todo.js, backbone.js, todos.js, todo-view.js, backbone.localStorage.js, app-view.js ja jquery.js, lisada Expires-päiseid eelmainitud mitmetele failidele ning teha vähem HTTP-päringuid ehk kombineerida JavaScripti faile kokku vähemaks arvuks välisteks JavaScripti failideks.

Autor annab TodoMVC testrakendusel põhineva jõudluse eest ühe punkti.

2.3 React (teek)

Reacti esitletakse kui JavaScripti teeki kasutajaliideste ehitamiseks [187]. Selle arendamist alustasid ja jätkavad senini keerukate kasutajaliideste ning dünaamiliste muudatustega seotud probleeme lahendavad Facebooki arendajad [188, lk 1]. React avaldati 2013. aastal ning tegemist pole tavalise MV* raamistikegrupi liikmega, vaid pigem lähedasem vaate osale mustrist „Mudel-vaade-kontroller" [188, lk 1] [188, lk 4]. Teek kasutab kiiret JavaScripti objektidel põhinevat virtuaalset dokumendiobjektide mudelit (*virtual DOM*), mida kasutatakse rakenduses toimuvate muutuste järel hinnangu andmiseks küsimusele, kas aeglasemalt toimiva päris DOM-i uuendus osutub vajalikus. Eelneva vajaduse ilmnemisel rakendab React tõelisele DOM-ile võimalikult vähe muudatusi ehk sooritab lepituse (*reconciliation*) [189] [190]. Käesoleva teegiga töötamisel soovitatakse kasutada XML-

märgendkeelele sarnast JavaScripti süntaksi laiendust JSX, mida saab muuta tavaliseks JavaScripti koodiks semantikat kaotamata [191].

2.3.1 Startimishinnang

Reacti Githubi Wiki alamleheküljel „Sites Using React“ on esitatud rida ettevõtteid, mis käesolevat teeki kasutavad [192]. Mitmete näidete seas asuvad muusikateenus Deezer, failimajutusteenused Dropbox ja Box, sotsiaalvõrgustikud Facebook ning Instagram ja sõidujagamisteenus Uber.

Reacti kodulehe keskel on kaks suurt nuppu: esimene viitab alustamisjuhendile ja teine allalaadimislehele [187]. Kasutades teist viita, saab ühe pakutud võimalusena alla laadida alustamiskomplekti (*starter kit*), mis sisaldab Reacti ning näidisrakendusi. Autor uuris alustamiskomplekti näidet `basic-click-counter`. Selle põhjal sai kirjutatud faili teremaailm.html sisu koos arvutusega 2+2*2 (joonis 60). Avades veebilehitsejas Chrome käesoleva HTML5-faili, kuvatakse teksti „Tere 6“. Samasse kausta koos HTML-failiga asetati failid react.js ning react-dom.js. Kolmas vajaminev Babeli JavaScripti kompileerija fail browser.min.js saabub sisuedastusvõrgu kaudu ja võimaldab veebilehitsejas esitada JSX-sisu [193]. Näide oli piisav, et lisasammudeta joonisel 60 olev kood luua.

```
<!DOCTYPE html>
<html>
  <head><title>Tere</title></head>
  <body>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/babel-
core/5.8.24/browser.min.js"></script>
    <div id="sisu"></div>
    <script type="text/babel">
      ReactDOM.render(
        <div>Tere {2+2*2}</div>,
        document.getElementById('sisu')
      );
    </script>
  </body>
</html>
```

Joonis 60. Reacti teremaailm.html alustamiskomplekti näite `basic-click-counter` põhjal

Eeltoodu vastab hinde kolm saamise kriteeriumitele. Reacti puhul piisab oluliste failide (react.js ja react-dom.js) allalaadimisest õigesse kausta ning veebilehitsejas käivitatavale failile viidete loomisest. Lisaks leidub vähemalt kolm vaadeldavat teeki kasutavat edukat ärifirmat. Need on Deezer, Dropbox ja Facebook. Autor annab startimishinnangu kategoorias kolm punkti.

2.3.2 Kogukond ja dokumentatsioon

2016. aasta veebruarikuu seisuga on teegi Githubi lehel 2674 vaatajat, 36442 tähega märkijat ja 5974 koodi harutajat [194]. Stack Exchange’i lehelt otsides leiab 10614 reactjs-sildiga küsimust, millest vastamata on 3679 [195] [196].

Reacti kodulehe alamlehel paikneb dokumentatsioon [197]. Kõigi lehekülgede puhul on arendajal võimalik soovitada muudatusi lingi „*Edit on GitHub*“ kaudu. Teksti toetab suur kogus koodinäiteid.

Eeltoodu vastab hinde kolm saamise kriteeriumitele. Githubis olevate vaatajate, tähega märkijate ning koodi harutajate (inglise keeles on koodi harutamine *forking*) summa on 45090. Stack Exchange’is moodustavad mittevastatuks märgitud küsimused ligikaudu 35% kõigist teegi nime kandva sildiga küsimustest. Lisaks eksisteerib koodinäiteid sisal-

dav dokumentatsioon, milles arendaja saab teha ettepanekuid muudatusteks. Autor annab kogukonna ja dokumentatsiooni kategoorias kolm punkti.

2.3.3 Arendustööriistade tugi

Reacti Githubi Wiki alamleheküljel on toodud mitmeid antud bakalaureusetööst välja jäävaid vaadeldava teegiga töötamist parendavaid arendustööriistade nimesid koos viitadega [198]. Arenduskeskkonnale Eclipse puuduvad Reacti pistikprogrammid. Samuti on ka Netbeansiga: inimesed on esitanud ametliku teate, milles soovitakse tuge JSX-formaadile [199]. Geertjan Wielenga tegeleb Reacti pistikprogrammi loomisega Netbeansile [200].

Jetbrains WebStorm on välja tulnud vastavalt tutvustusele paremaks muudetud Reacti toega [201]. Mitmete võimaluste seast tuuakse välja, et toimib koodi lõpule viimine (*code completion*) teegile spetsiifiliste atribuutide jaoks. Lisaks on võimalik paigaldada nii WebStormile kui ka PhpStormile Reacti mallide kasutamist lihtsustavat pistikprogrammi React-Templates [202].

Paketihaldurist tekstiredaktorile Sublime Text leiab mitmeid teegi React jaoks mõeldud abipakette, kuid üks populaarne valik on babel-sublime, mis töötab Sublime Text 3-ga [203]. See sisaldab keeledefinitsioone ECMAScript 6+ standardi JavaScriptile ning React JSX süntaksile ja oskab seega õigesti värvida koodis olevaid võtmesõnu. Lisaks babel-sublime paketile võib paigaldada ka paketi babel-sublime-snippets, mille abil saab lasta tihti kasutust leidvaid koodiväljavõtteid (*snippets*) kiirvalikuga genereerida [204].

Visual Studiol on sisse ehitatud täielik JavaScripti ja JSX süntaksi redigeerimistugi [205]. Lisaks saab alla laadida koodiväljavõttete (*snippets*) paki React Snippet Pack, mille abil on võimalik tihti kasutust leidvaid koodiväljavõtteid (*snippets*) kiirvalikuga genereerida [206].

Autor annab arendustööriistade toe eest kaks punkti, sest toetatud on vähemalt kaks vaadeldavat arenduskeskkonda: Jetbrainsi tooted, Sublime Text ja Visual Studio.

2.3.4 Testimistugi

Reacti dokumentatsioonis on eraldi lehekülg pühendatud testimisele [207]. Eksisteerib lisapakett React TestUtils, mis võimaldab testida vaadeldava teegi komponente arendaja valitud testimisraamistiku abil [207]. Facebook kasutab sellel otstarbel testimisraamistikuna ühiktestimise raamistikku Jest [208]. Nii TestUtilsi kui ka Jesti kasutamise põhimõtete kohta eksisteerivad lisaks selgitustele arusaamist parendavad koodinäited [207] [208].

Leidub Reactiga loodud rakenduse testimist seletav materjal koos koodinäidistega. Autor annab testimistoe kriteeriumi eest kolm punkti.

2.3.5 Litsents äri vaatepunktist

Teegi React puhul kehtib BSD 3 klausli litsents *(The BSD 3-Clause License*) [209]. Seega peab Reacti muudetud või muutmata kujul kasutav tarkvara lähtekood või binaarne vorm sisaldama litsentsis toodud autoriõiguse märkust ning litsentsi täisteksti [210]. Facebooki ja selle kaastöötajate nimesid on keelatud eelneva kirjaliku loata Reacti põhjal loodud tarkvara edu suurendamise eesmärgil kasutada. Autoriõiguse omanik koos kaastöötajatega pole vastutav kahjude eest, mida Reacti kasutamine kaasa võib tuua.

Selleks, et arendajad Facebooki avalikke projekte saaks kindlustundega kasutada, lisatakse neile ka patent [211]. Reacti pantendi järgi on teeki kasutav äritegevus lubatud. Patenditekstis antud litsents lõppeb automaatselt ja teadaandeta tekstis toodud kolmel juhul, mis

on seotud patendi väitmistegevusega (*Patent Assertion*) seal defineeritud moel Facebooki või mistahes osapoole vastu vaadeldava patendi tekstis toodud viisil [212].

Vastava teegi abil loodud rakendust lubatakse äriks kasutada, rakenduse lähtekood võib jääda salajaseks. React saab litsentsi eest kolm punkti.

2.3.6 Keele ja vormingu tugi

Internatsionaliseerimiseks mõeldud JavaScripti teekide kogusse FormatJS kuuluv teek React Intl pakub keele ning vormingu toe funktsionaalsust [213]. See võimaldab React teeki kasutatavates rakendustes vormindada numbreid komponendiga `FormattedNumber` ja kuupäevi `FormattedDate` abil, kindlaid või suhtelisi ajamääranguid vastavalt komponendiga `FormattedTime` või `FormattedRelative`. Tõlgitud ICU sõnumi formaadis sõnekujul sõnumi vormindamiseks tuleb kasutada komponenti `FormattedMessage` või `FormattedHTMLMessage` [214]. Lokaadiga (*locale*) seotud funktsionaalsuse jaoks tuleb kasutada React Intli Locale Data API-t [215].

React Intl hõlbustab mitmekesisusega hakkamasaamist: võimaldab kasutajal rakenduse keelt muuta ja kasutada tema asukohale sobivaid formaate. Kuna konkreetselt teegil React loodud keele ja formaatide tugi põhineb raamistikust eraldiseisva lisatarkvara võimalustel, siis autor annab keele ja vormingu toe eest ühe punkti.

2.3.7 TodoMVC testrakendusel põhinev keerukus

Vastavalt üldkriteeriumitele sai hinnatud testrakenduse JavaScripti faile või jsx-formaadist saadud JavaScripti faile, mis on esindatud punktis 2.3.8 [24]. Kuna testrakenduses kasutatakse jsx-formaadis faile, siis keerukuse arvutamiseks muudeti need js-formaadis JavaScripti failideks JavaScripti kompileerijaga Babel [216]. Tulemused on toodud allpool.

1. Lähtekoodi loogiliste ridade arv kokku liidetult paikneb tabelis 7.

Tabel 7. Reacti testrakenduse ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
BabeligaSaadud_app.js (failist app.jsx)	112	278
BabeligaSaadud_footer.js (failist footer.jsx)	28	
BabeligaSaadud_todoItem.js (failist todoItem.jsx)	62	
todoModel.js	44	
utils.js	32	

2. McCabe'i tsüklomaatilise keerukuse (*Cyclomatic complexity*) keskmine (liidetakse kokku iga vaadeldava JavaScripti faili tsüklomaatilise keerukuse arv ning jagatakse vaadeldavate failide arvuga) paikneb tabelis 8.

Tabel 8. Reacti testrakenduse tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
BabeligaSaadud_app.js (failist app.jsx)	10	8
BabeligaSaadud_footer.js (failist footer.jsx)	3	
BabeligaSaadud_todoItem.js (failist todoItem.jsx)	9	
todoModel.js	4	
utils.js	14	

3. Halstead'i jõupingutuste (*effort*) keskmine funktsioonide põhjal (*Mean per-function Halstead effort*) oli 2479, 73395828517.
4. Hooldatavuse indeks (liidetakse kokku iga vaadeldava JavaScripti faili indeks ja jagatakse see vaadeldavate failide arvuga) paikneb tabelis 9.

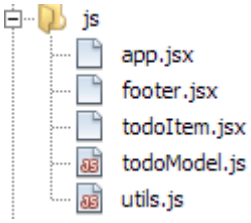
Tabel 9. Reacti testrakenduse hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**
BabeligaSaadud_app.js (failist app.jsx)	115, 9208201534499	116, 96
BabeligaSaadud_footer.js (failist footer.jsx)	100, 193440128084	
BabeligaSaadud_todoItem.js (failist todoItem.jsx)	116, 89418193843146	
todoModel.js	135, 7288867495176	
utils.js	116, 05270167430497	

Reactiga loodud testrakendus kuulub loogiliste ridade arvu kategoorias 12. kohale (lisa „I. Keerukuse pingeread", tabel 20), tsüklomaatilise keerukuse kategoorias 13. kohale (lisa „I. Keerukuse pingeread", tabel 21), hooldatavuse indeksi kategoorias 15. kohale (lisa „I. Keerukuse pingeread", tabel 22) ning funktsioonide põhjal saadud Halstead'i jõupingutuste (*effort*) keskmise kategoorias 14. kohale (lisa „I. Keerukuse pingeread", tabel 23). React saab testrakendusel põhineva keerukuse kategoorias null punkti, sest tulemuseks on pingeridade lõppu kuulumine.

2.3.8 TodoMVC testrakendusel põhinev õppimine ja õppimiskiirus

Järgneva MIT litsentsiga TodoMVC projekti testrakenduse autor on Pete Hunt (lisa „III. TodoMVC projekti repositooriumi sisu kasutamise litsents“) [24]. Failiga index.html asub samas kaustas kaust `js`, mille sisu on toodud joonisel 85.



Joonis 61. Kausta `js` sisu [24]

Faili index.html päises `<head>`-siltide vahel on viited kasutatavatele TodoMVC projekti stiililehtedele ning kehas `<body>`-siltide vahel kaks olulist piirkonda. Elemendis klassiga `todoapp` paikneb rakenduse põhifunktsionaalsuse osa (joonis 86). Jaluses kuvatakse staatilist infot Pete Hunti kohta.

```
//...//
        <section class="todoapp"></section>
        <footer class="info">
               <p>Double-click to edit a todo</p>
               <p>Created by <a href="http://github.com/petehunt/">petehunt</a></p>
               <p>Part of <a href="http://todomvc.com">TodoMVC</a></p>
        </footer>
//...//
```

Joonis 62. index.html ning `todoapp` ja `info` [24]

Teise olulise piirkonna moodustab vajalike JavaScripti või JSX-failidele viitamine. Selle alumises osas (joonis 63) kasutatakse kõiki kausta `js` faile ning kommenteeritakse, et failid utils.js ja todoModel.js pole Reactiga seotud ning seega implementeeritud tavalises JavaScripti koodis. Kolm ülejäänud faili aga kasutavad ära JSX-süntaksit, mida kompileeritakse töötava programmi saamiseks JSX-kompileerijaga JavaScripti kujule [188, lk 43]. Selleks on kaustast `node_modules` kasutusse võetud lisaks failidele base.js, react-with-addons.js, index.js, director.js ka fail JSXTransformer.js. Alates 2015. aasta suvest aga JSXTransormerit enam ei kasutata: selle asemel peab uute rakenduste loomisel töötama kompileerijaga Babel [193] [217].

```
//...//
        <script src="js/utils.js"></script>
        <script src="js/todoModel.js"></script>
        <!-- jsx is an optional syntactic sugar that transforms methods in React's
        `render` into an HTML-looking format. Since the two models above are
        unrelated to React, we didn't need those transforms. -->
        <script type="text/jsx" src="js/todoItem.jsx"></script>
        <script type="text/jsx" src="js/footer.jsx"></script>
        <script type="text/jsx" src="js/app.jsx"></script>
//...//
```

Joonis 63. index.html ning viiteid failidele [24]

Kõigi järgnevate failide esimeseks koodireaks on `var app = app || {};`, mis loob sama nimeruumi kõigile seda omavatele JavaScripti ning JSX-failidele ja lubab neid faile suvalises järjekorras rakenduse tarvis laadida [144]. Nende failide kood asub JavaScripti moodulimustri (*JavaScript Modul Pattern*) anonüümse sulundi (*closure*) konstruktsiooni sees (joonis 37) [159]. JavaScript toetab programmeerimist objektorienteeritult, imperatiivselt ja protseduraalselt [8]. Failis todoModel.js on kasutatud tegevuse (*todo*) mudeli defineerimiseks objektorienteeritud viisi [218]. Kõigepealt luuakse muutuja `Utils` globaalsest nimeruumist pärineva failis utils.js defineeritud `app.Utils` põhjal. Seejärel

defineeritakse klass `TodoModel` atribuutidega `key`, `todos` ja `onChanges`. Atribuut `todos` on seotud failis utils.js defineeritud veebilehitseja andmehoidu haldava funktsiooniga `store`.

```
//.../
var Utils = app.Utils;
  // Generic "model" object. You can use whatever
  // framework you want. For this application it
  // may not even be worth separating this logic
  // out, but we do this to demonstrate one way to
  // separate out parts of your application.
  app.TodoModel = function (key) {
    this.key = key;
    this.todos = Utils.store(key);
    this.onChanges = [];
  };
//.../
```

Joonis 64. todoModel.js ning konstruktor [24]

Prototüübiatribuudi kaudu defineeritakse klassile `TodoModel` edaspidi failis todoModel.js mitmeid meetodeid (joonis 65) [219]. Meetod `subscribe` kasutab JavaScripti meetodit `push` selleks, et lisada argumendiks saadu vaadeldava klassi loetelurivi tüüpi atribuudi `onChanges` lõppu [109]. Meetod `inform` kutsub välja failis utils.js defineeritud funktsiooni `store` klassi `TodoModel` kahe atribuudiga. Lisaks läbitakse JavaScripti meetodiga `forEach` atribuut `onChanges` ja kutsutakse selle elemente välja funktsioonidena – nimi `cb` on inspireeritud tagasikutse (*callback*) inglisekeelsest sõnast. Meetodit `inform` hakatakse rakendama kõigis faili todoModel.js edaspidi esinevates meetodites, et realiseerida nende poolt tehtud muudatused veebilehitseja andmehoius.

```
//.../
  app.TodoModel.prototype.subscribe = function (onChange) {
    this.onChanges.push(onChange);
  };

  app.TodoModel.prototype.inform = function () {
    Utils.store(this.key, this.todos);
    this.onChanges.forEach(function (cb) { cb(); });
  };
//.../
```

Joonis 65. todoModel.js ning `subscribe` ja `inform` [24]

Kui kasutaja lisab tegevuste haldamise rakenduses uue tegevuse, siis selle pealkiri läheb meetodi `addTodo` argumendiks, mis lisab klassi `TodoModel` atribuudi `todos` elemendiks objekti kolme atribuudiga (joonis 66). Nendeks on failis utils.js defineeritud funktsiooniga `uuid` saadav identifikaator, pealkiri ning tõeväärtusväli tegevuse tehtuks märkimise kohta. JavaScripti meetod `concat` tekitab uue ühise loetelurivi oma argumendist ja olemasolevast loetelurivist `todos` [220].

```
//.../
  app.TodoModel.prototype.addTodo = function (title) {
    this.todos = this.todos.concat({
      id: Utils.uuid(),
      title: title,
      completed: false
   });

    this.inform();
  };
//.../
```

Joonis 66. todoModel.js ning `addTodo` [24]

Meetod `toggleAll` on mõeldud kõigi tegevuste atribuudis tegevuste korraga tehtuks või mittetehtuks märkimise jaoks (joonis 67). See võtab argumendiks iga tegevuse objekti klassi `TodoModel` atribuudist `todos` ja rakendab siis failis utils.js defineeritud meetodit

`extend`. Iga elemendini jõutakse JavaScripti meetodiga `map`, mis rakendab mainitud funktsiooni igale loetelurivi elemendile ning loob saadud tulemustest uue loetelurivi [221].

```
//...//
  app.TodoModel.prototype.toggleAll = function (checked) {
    // Note: it's usually better to use immutable data structures since they're
    // easier to reason about and React works very well with them. That's why
    // we use map() and filter() everywhere instead of mutating the array or
    // todo items themselves.
    this.todos = this.todos.map(function (todo) {
      return Utils.extend({}, todo, {completed: checked});
   });

    this.inform();
  };
//...//
```

Joonis 67. todoModel.js ning `toggleAll` [24]

Meetodi `toggle` ülesanne on muuta tegevuse objekti tehtuks märkimist väljendavat tõeväärtust vastupidiseks (joonis 68). Selleks kasutatakse analoogseid võtteid nagu `toggleAll` puhul. Meetod `destroy` aga jätab JavaScripti meetodiga `filter` atribuuti `todos` alles kõik sellised tegevusobjektid, mis pole võrdsed argumendiks saadud tegevuse objektiga ehk argumendiks saadu kustub [222].

```
//...//
  app.TodoModel.prototype.toggle = function (todoToToggle) {
    this.todos = this.todos.map(function (todo) {
      return todo !== todoToToggle ?
        todo :
        Utils.extend({}, todo, {completed: !todo.completed});
    });
   this.inform();
  };
  app.TodoModel.prototype.destroy = function (todo) {
    this.todos = this.todos.filter(function (candidate) {
      return candidate !== todo;
    });

    this.inform();
  };
//...//
```

Joonis 68. todoModel.js ning `toggle` ja `destroy` [24]

Meetodi `save` ülesanne on salvestada kasutaja poolt algatatud tegevuse objekti pealkirja muudatus (joonis 69). Meetod `clearCompleted` kasutab JavaScripti funktsiooni `filter`, et välja filtreerida kõik need tegevuste objektid, mis pole märgitud tehtuks – selle tagajärjel tehtuks märgitud tegevused kustuvad.

```
//...//
  app.TodoModel.prototype.save = function (todoToSave, text) {
    this.todos = this.todos.map(function (todo) {
      return todo !== todoToSave ? todo : Utils.extend({}, todo, {title: text});
    });
   this.inform();
  };
  app.TodoModel.prototype.clearCompleted = function () {
    this.todos = this.todos.filter(function (todo) {
      return !todo.completed;
    });

    this.inform();
  };
//...//
```

Joonis 69. todoModel.js ning `save` ja `clearCompleted` [24]

Failis utils.js on defineeritud objekt `Utils`, milles leiduvad funktsioonid `uuid`, `pluralize`, `store` ja `extend`. Neist esimese ülesandeks on genereerida tegevuste objektidele juhuslikke identifikaatoreid, näiteks `d849fa91-2757-4cac-85ed-0994083b818c`

(joonis 70). Selle jaoks kasutatakse JavaScripti funktsiooni `Math.random`, mis tagastab ujukomaarvu vahemikus [0, 1) [223].

```
//...//
app.Utils = {
    uuid: function () {
      /*jshint bitwise:false */
      var i, random;
      var uuid = '';
      for (i = 0; i < 32; i++) {
        random = Math.random() * 16 | 0;
        if (i === 8 || i === 12 || i === 16 || i === 20) {
         uuid += '-';
        }
        uuid += (i === 12 ? 4 : (i === 16 ? (random & 3 | 8) : random))
          .toString(16);
      }
     return uuid;
    },
//...//
```

Joonis 70. utils.js ning `uuid` [24]

Failis footer.jsx kasutust leidev funktsioon `pluralize` tagab, et inglise keeles kasutajale näidatava arvu juures paikneva sõna puhul oleks mitmus ja ainsus keeleliselt korrektselt väljendatud (joonis 71). Veebilehitseja andmehoiga manipuleeriv funktsioon `store` salvestab kahe argumendi olemasolu korral andmed andmehoidu, kuid ühe argumendi puhul tagastab andmehoius paineva sisu. Selle käigus viiakse enne veebilehitseja andmehoidu salvestamist andmed JSON-sõne kujule meetodiga `JSON.stringify` ja taastatakse nende originaalne kuju meetodiga `JSON.parse` [113] [114].

```
//...//
   pluralize: function (count, word) {
      return count === 1 ? word : word + 's';
    },
   store: function (namespace, data) {
      if (data) {
        return localStorage.setItem(namespace, JSON.stringify(data));
      }
      var store = localStorage.getItem(namespace);
      return (store && JSON.parse(store)) || [];
    },
//...//
```

Joonis 71. utils.js ning `pluralize` ja `store` [24]

Faili utils.js funktsioon `extend` (joonis 72) leiab kasutust failis todoModel.js defineeritud meetodites, kus läheb vaja muuta tegevuse objekti mingi atribuudi väärtust. Funktsioon `extend` loob uue objekti, vaatab läbi kõik funktsiooni argumendid JavaScripti funktsioonidele omase objekti `arguments` kaudu ja teostab uue objekti atribuutidele väärtuste omistamise pärast meetodi `hasOwnProperty` tõest tulemust [224] [225].

```
//...//
    extend: function () {
      var newObj = {};
      for (var i = 0; i < arguments.length; i++) {
        var obj = arguments[i];
        for (var key in obj) {
          if (obj.hasOwnProperty(key)) {
            newObj[key] = obj[key];
          }
        }
      }
      return newObj;
    }
  };
//...//
```

Joonis 72. utils.js ning `extend` [24]

Failis footer.jsx on loodud Reacti komponendiklass (*component class*) `TodoFooter` (joonis 73). Reacti meetodit `React.createClass` kasutatakse uute komponentklasside defineerimiseks [226]. Nende sees peab olema implementeeritud meetod `render` [227]. Komponentidel on hulk atribuute, millele pääsetakse ligi viida `this.props` kaudu [188, lk 17]. Joonisel 73 kasutatakse atribuuti `count` ja failist utils.js pärinevat funktsiooni `pluralize`, et muutujasse `activeTodoWord` salvestada sõna „*item*“ kas mitmuses või ainsuses. Kui `TodoFooter`-i atribuut `completedCount` on arvust null suurem, siis saab muutuja `clearButton` JSX-süntaksit kasutades nupu elemendiks. Selle kasutajale kuvatav tekst on „*Clear completed*“ ja sellele klõpsates atribuuti `onClearCompleted` kasutades failis app.jsx defineeritud viisil kutsutakse välja failis todoModel.js defineeritud tehtuks märgitud tegevuste objekte kustutav funktsioon `clearCompleted`.

```
//...//
    app.TodoFooter = React.createClass({
       render: function () {
          var activeTodoWord = app.Utils.pluralize(this.props.count, 'item');
          var clearButton = null;
          if (this.props.completedCount > 0) {
             clearButton = (
                <button
                   className="clear-completed"
                   onClick={this.props.onClearCompleted}>
                   Clear completed
                </button>
             );
          }
//...//
```

Joonis 73. footer.jsx ning meetodi `render` algus [24]

Muutuja `nowShowing` väärtuseks saab vaadeldava komponendiklassi atribuudi `nowShowing` väärtus (joonis 74). Lõpuks tagastatakse JSX-süntaksit kasutades element, mis on jaluse (*footer*) tüüpi. Sildi `<span>` alas näidatakse kasutajale loendurit ja selle kõrval teksti muutujast `activeTodoWord`. Seejärel järgneb loetelu `<ul>`-siltide alas.

```
//...//
          var nowShowing = this.props.nowShowing;
          return (
             <footer className="footer">
                  <span className="todo-count">
                     <strong>{this.props.count}</strong> {activeTodoWord} left
                  </span>
                  <ul className="filters">
//...//
```

Joonis 74. footer.jsx ning meetodi `render` keskmine osa [24]

Eelmise lõigu lõpus mainitud loetelu on lühendatud kujul toodud joonisel 75. Seal on näha, et `<li>`-sildiga elementide järel on kirjas {' '}, mis kuvatakse kasutajale tühikuna, mis eraldab vaates neid elemente teineteisest. Pärast `<ul>`-sildi piirkonna lõppu ning enne `<footer>`-sildi piirkonna lõppu on kirjas `{clearButton}`, mis kujutab endast eelnevalt joonisel 73 defineeritud nupu elementi, mis muutub nähtavaks ainult siis, kui `TodoFooter`-i atribuut `completedCount` on arvust null suurem. Jooniselt 75 loetelu seest välja jäetud iga koodijupp asub `<a>`-sildi sees ehk kasutaja jaoks on tegu hüperlinkidega [228]. Esimesel on neist URL-viide `href="#/"` ja tekst „*All*“, teisel `href="#/active"` ja tekst „*Active*“ ning kolmandal `href="#/completed"` koos tekstiga „*Completed*“. Tegemist on kasutajale tegevuste filtreerimiseks mõeldud nuppudega. Filtreerimisprotsess toimub kõigil kolmel elemendil atribuudi `classname` kaudu: sellele on omistatud `{classNames({selected: …})}`, kus kolme punkti asemel on muutuja `nowShowing` võrduse kontroll vastavalt `app.ALL_TODOS`-ga,

`app.ACTIVE_TODOS`-ga ning `app.COMPLETED_TODOS`-ga. Funktsioon `classNames` on käesoleva rakenduse lisasõltuvus projektist Classnames, mis võimaldab tingimuskontrolli sooritades elementidele klasse lisada või eemaldada [229].

```
//...//
                    <li>
                        //...//
                    </li>
                    {' '}
                    <li>
                        //...//
                    </li>
                    {' '}
                    <li>
                        //...//
                    </li>
                </ul>
                {clearButton}
            </footer>
//...//
```

Joonis 75. footer.jsx ning meetodi `render` alumine osa [24]

Failis todoItem.jsx on defineeritud komponendiklass `TodoItem` (joonis 76). Komponentidel eksisteerib oma olek (*state*), mille sisule saab ligi viidaga `this.state` [188, lk 18]. Funktsioon `handleSubmit` on mõeldud selleks, et kontrollida kasutaja poolt sisestatud teksti: kui see eksisteerib, siis kasutada failis app.jsx defineeritud funktsiooni `onSave` ja omistada see tekst `editText`-i uueks väärtuseks meetodiga `setState` [230]. Vastasel juhul rakendatakse failis app.jsx defineeritud funktsiooni `onDestroy`. Oleku muutusel toimub alati komponendi taasesitumine (`re-rendering`) [231].

```
//...//
    var ESCAPE_KEY = 27;
    var ENTER_KEY = 13;

    app.TodoItem = React.createClass({
       handleSubmit: function (event) {
          var val = this.state.editText.trim();
          if (val) {
             this.props.onSave(val);
             this.setState({editText: val});
          } else {
             this.props.onDestroy();
          }
       },
//...//
```

Joonis 76. todoItem.jsx ning `handleSubmit` [24]

Funktsioon `handleEdit` kutsub välja failis app.jsx defineeritud funktsiooni `onEdit` ning annab atribuudile `editText` väärtuseks atribuudi `todo` pealkirja atribuudi (joonis 77) [230]. Argumendiks sündmuse saanud funktsioon `handleKeyDown` kontrollib, kas sündmuseks oli klahvi Escape või klahvi Enter vajutamine. Esimesel juhul antakse atribuudile `editText` väärtuseks tegevuse pealkirja atribuut ja kutsutakse välja failis app.jsx defineeritud funktsiooni `onCancel`, teisel juhul käivitatakse eelneval joonisel esitatud `handleSubmit`.

```
//...//
        handleEdit: function () {
           this.props.onEdit();
           this.setState({editText: this.props.todo.title});
        },

        handleKeyDown: function (event) {
           if (event.which === ESCAPE_KEY) {
              this.setState({editText: this.props.todo.title});
```

```
            this.props.onCancel(event);
          } else if (event.which === ENTER_KEY) {
            this.handleSubmit(event);
          }
        },
//...//
```

Joonis 77. todoItem.jsx ning `handleEdit` ja `handleKeyDown` [24]

Funktsioon `handleChange` kasutab sündmuse sihtmärgi (*event target*) atribuuti: kui atribuut `editing` läbib tingimuskontrolli, siis omistatakse atribuudile `editText` selle elemendi väärtus, millega seoses sündmus toimus (joonis 78) [232]. Komponendi elutsükli alguses kasutatav funktsioon `getInitialState` tagastab objekti, millel on atribuut `editText`, mille väärtus on atribuudi `todo` atribuudi `title` väärtus ehk tegevuse objekti kasutajale kuvatava pealkirja väärtus [188, lk 31].

```
//...//
        handleChange: function (event) {
          if (this.props.editing) {
            this.setState({editText: event.target.value});
          }
        },

        getInitialState: function () {
          return {editText: this.props.todo.title};
        },
//...//
```

Joonis 78. todoItem.jsx ning `handleChange` ja `getInitialState` [24]

Failis todoItem.jsx on kasutusele võetud ka jõudlust parendav komponendi elutsükli funktsioon (*component lifecycle function*) `shouldComponentUpdate` (joonis 79) [233]. Seda kutsutakse välja enne komponendi taasesitamist (*re-rendering*): kui tagastatav tõeväärtus on väär, siis React ei käivita funktsiooni `render`. Vaadeldaval juhul kontrollitakse enne uude olekusse või uute atribuutide väärtusteni jõudmist üle komponendi atribuudid `todo` ja `editing` ning `editText` [234].

```
//...//
          /**
           * This is a completely optional performance enhancement that you can
           * implement on any React component. If you were to delete this method
           * the app would still work correctly (and still be very performant!), we
           * just use it as an example of how little code it takes to get an order
           * of magnitude performance improvement.
           */
          shouldComponentUpdate: function (nextProps, nextState) {
            return (
              nextProps.todo !== this.props.todo ||
              nextProps.editing !== this.props.editing ||
              nextState.editText !== this.state.editText
            );
          },
//...//
```

Joonis 79. todoItem.jsx ning `shouldComponentUpdate` [24]

Funktsiooni `componentDidUpdate` kutsutakse välja, kui komponenti uuendused on kajastatud DOM-is. Selle argumendid on tavaliselt uuendusele eelnenud atribuudid ja olek [188, lk 36] [235]. Käesolevas rakenduses viiakse funktsiooniga `componentDidUpdate` kasutaja poolt redigeeritavale tekstisisestusvälja DOM-elemendile fookus (joonis 80) [99].

```
//...//
        /**
         * Safely manipulate the DOM after updating the state when invoking
         * `this.props.onEdit()` in the `handleEdit` method above.
         * For more info refer to notes at https://facebook.github.io/react/docs/component-api.html#setstate
         * and https://facebook.github.io/react/docs/component-specs.html#updating-componentdidupdate
         */
         componentDidUpdate: function (prevProps) {
            if (!prevProps.editing && this.props.editing) {
               var node = React.findDOMNode(this.refs.editField);
               node.focus();
               node.setSelectionRange(node.value.length, node.value.length);
            }
         },
//...//
```

Joonis 80. todoItem.jsx ning `componentDidUpdate` [24]

Faili todoItem.jsx lõpus on defineeritud komponendi esitamise funktsioon `render`. Jooniselt 81 on välja jäetud joonistel 82 ja 83 kujutatud osa. Joonisel 81 on näha, et tagastatakse loetelu element. Rakenduse lisasõltuvusena kasutatava funktsiooniga `classNames` lisatakse või eemaldatakse vaadeldava loetelu elemendi klassid `completed` ja `editing` vastavalt atribuutide `todo.completed` või `editing` tõeväärtusele [229].

```
//...//
         render: function () {
           return (
              <li className={classNames({
                 completed: this.props.todo.completed,
                 editing: this.props.editing
              })}>
                //...//
              </li>
           );
         }
      });
//...//
```

Joonis 81. todoItem.jsx ning lühendatud `render` [24]

Loetelu elemendi ühe olulise osa moodustab joonisel 82 esitatud `<div>`-sildiga element. Selle sees on kasutajale mõeldud märgendkast: klõpsamine muudab komponendi atribuudi `todo.completed` tõeväärtust. Muutusi jälgitakse tagasikutse (*callback*) seadmisega Reacti atribuudile `onChange` [236]. Märgendkastile järgneb sildi `<label>` ala, millel topeltklõpsamine on seotud funktsiooniga `handleEdit` ning milles võimaldatatakse näidata kasutajale tegevuse objekti pealkirja ehk atribuudi `todo.title` väärtust. Nupule klassiga `destroy` klõpsates käivitatakse failis app.jsx defineeritud atribuutfunktsioon `onDestroy`, mille tagajärjel tegevuse objekt kustutatakse.

```
//...//
               <div className="view">
                  <input
                     className="toggle"
                     type="checkbox"
                     checked={this.props.todo.completed}
                     onChange={this.props.onToggle}
                  />
                  <label onDoubleClick={this.handleEdit}>
                     {this.props.todo.title}
                  </label>
                  <button className="destroy" onClick={this.props.onDestroy} />
               </div>
//...//
```

Joonis 82. todoItem.jsx ning loetelu sisu algus [24]

Tegevuse objekti redigeerimiseks loodud sisestuskastielement joonisel 83 omab Reacti atribuuti `ref`, mis võimaldab elemendile viidata funktsiooni `componentDidUpdate` sees [237]. Muutusi jälgitakse tagasikutsete (*callback*) seadmisega Reacti atribuutidele `onChange`, `onBlur` ja `onKeyDown` [236] [238] [239].

```
//...//
                    <input
                       ref="editField"
                       className="edit"
                       value={this.state.editText}
                       onBlur={this.handleSubmit}
                       onChange={this.handleChange}
                       onKeyDown={this.handleKeyDown}
                    />
//...//
```

Joonis 83. todoItem.jsx loetelu sisu lõpp [24]

Faili app.jsx alguses seotakse lokaalsed muutujad `TodoFooter` ja `TodoItem` neile vastavate globaalse nimeruumi komponentidega (joonis 84). Globaalsesse nimeruumi luuakse muutujad `ALL_TODOS`, `ACTIVE_TODOS` ja `COMPLETED_TODOS`, millele omistatakse väärtuseks sõned, mida hiljem saab URL-haldamise juures kasutada.

```
//...//
    app.ALL_TODOS = 'all';
    app.ACTIVE_TODOS = 'active';
    app.COMPLETED_TODOS = 'completed';
    var TodoFooter = app.TodoFooter;
    var TodoItem = app.TodoItem;
    var ENTER_KEY = 13;
//...//
```

Joonis 84. app.jsx ning muutujad [24]

Komponendiklassis `TodoApp` esimese funktsioonina defineeritud komponendi elutsükli alguses kasutatav funktsioon `getInitialState` tagastab kolme atribuudiga objekti (joonis 85). Kuna atribuudi `nowShowing` väärtus on `ALL_TODOS`, siis alguses näidatakse kasutajale nii tehtuks kui mittetehtuks märgitud tegevusi.

```
//...//
    var TodoApp = React.createClass({
       getInitialState: function () {
          return {
             nowShowing: app.ALL_TODOS,
             editing: null,
             newTodo: ''
          };
       },
//...//
```

Joonis 85. app.jsx ning `getInitialState` [24]

Reacti komponendi algesitamise (*initial render*) jooksul kutsutakse välja järgnevad funktsioonid järgnevas järjekorras: `getInitialProps`, `getInitialState`, `componentWillMount`, `render` ja `componentDidMount` [188, lk 37, joonis 2-1] [240]. Joonisel 86 on esitatud komponendiklassi `TodoApp` funktsioon `componentDidMount`. Selle sees seotakse võtmesõnaga `this` kasutades JavaScripti meetodiga `bind` vaadeldav komponendiklass ja komponendiklassi meetod `setState`, millega saab muuta argumentobjekti atribuudi `nowShowing` väärtust [230] [241]. URL-haldamine seatakse sisse, kui eelnevalt mainitud sidumised ühendatakse testrakenduse sõltuvuse Flatiron Director projekti URL-haldajaga `Router` [242]. Väljakutsega `router.init('/')` avaneb kasutajal rakenduse alglaadimise järel URL-tee lõppu `'/#/'` [243].

```
//...//
        componentDidMount: function () {
           var setState = this.setState;
           var router = Router({
              '/': setState.bind(this, {nowShowing: app.ALL_TODOS}),
              '/active': setState.bind(this, {nowShowing: app.ACTIVE_TODOS}),
```

```
            '/completed': setState.bind(this, {nowShowing: app.COMPLETED_TODOS})
         });
         router.init('/');
      },
//...//
```

Joonis 86. app.jsx ning `componentDidMount` [24]

Funktsiooni `handleChange` sees omistatakse meetodiga `setState` atribuudile `newTodo` selle elemendi väärtus, millega seoses sündmus toimus (joonis 87) [230] [232]. Funktsioonis `handleNewTodoKeyDown` listakse klahvi Enter vajutamise järel kasutaja poolt sisestatud pealkirjaga tegevuse objekt tegevusobjektide hulka ning algväärtustatakse atribuut `newTodo` tühja sõnega.

```
//...//
      handleChange: function (event) {
         this.setState({newTodo: event.target.value});
      },
      handleNewTodoKeyDown: function (event) {
         if (event.keyCode !== ENTER_KEY) {
            return;
         }
         event.preventDefault();
         var val = this.state.newTodo.trim();
         if (val) {
            this.props.model.addTodo(val);
            this.setState({newTodo: ''});
         }
      },
//...//
```

Joonis 87. app.jsx ning `handleChange` ja `handleNewTodoKeyDown` [24]

Lühikese kirjapildiga funktsioonid joonisel 88 kasutavad seni käsitletud funktsioonidega sarnaseid põhimõtteid. Nendest bakalaureusetöö autor ei kirjuta.

```
//...//
      toggleAll: function (event) {
         var checked = event.target.checked;
         this.props.model.toggleAll(checked);
      },
      toggle: function (todoToToggle) {
         this.props.model.toggle(todoToToggle);
      },
      destroy: function (todo) {
         this.props.model.destroy(todo);
      },
      edit: function (todo) {
         this.setState({editing: todo.id});
      },
      save: function (todoToSave, text) {
         this.props.model.save(todoToSave, text);
         this.setState({editing: null});
      },
      cancel: function () {
         this.setState({editing: null});
      },
      clearCompleted: function () {
         this.props.model.clearCompleted();
      },
//...//
```

Joonis 88. app.jsx ning seitse funktsiooni [24]

Komponendiklassis `TodoApp` leiduvad veel lisaks eelnevatele funktsioonidele ka funktsioon `render`, mille sisu on jooniselt 89 välja jäetud ja kujutatud joonistel 90 kuni 94. Pärast komponendiklassi `TodoApp` definitsiooni omistatakse muutuja `model` väärtuseks failis todoModel.js defineeritud mudeli isend võtmeargumendiga `react-todos`. Järgnevalt defineeritud funktsioon `render` kutsub välja funktsiooni `React.render`. See esitab failist index.html meetodiga `getElementsByClassName` leitud elemendis klas-

siga `todoapp` komponendi `TodoApp`, mis kasutab muutujas `model` olevat objekti [188, lk 27]. Joonise 89 viimasel real käivitatakse rakendus väljakutsega `render`.

```
//...//
    render: function () {
        //...//
   }
});
    var model = new app.TodoModel('react-todos');

    function render() {
       React.render(
          <TodoApp model={model}/>,
          document.getElementsByClassName('todoapp')[0]
       );
    }

    model.subscribe(render);
    render();
//...//
```

Joonis 89. app.jsx faili lõpp [24]

Komponendiklassi `TodoApp` funktsiooni `render` koodi sees omistatakse muutujale `todos` atribuudi `model.todos` väärtus (joonis 90). Muutujas `shownTodos` filtreeritakse vastavalt olekuatribuudile `nowShowing` välja tingimust rahuldavad tegevusobjektid [222].

```
//...//
            var footer;
            var main;
            var todos = this.props.model.todos;

            var shownTodos = todos.filter(function (todo) {
               switch (this.state.nowShowing) {
               case app.ACTIVE_TODOS:
                  return !todo.completed;
               case app.COMPLETED_TODOS:
                  return todo.completed;
               default:
                  return true;
               }
            }, this);
//...//
```

Joonis 90. app.jsx ning `renderi` `shownTodos` [24]

Muutuja `todoItems` väärtuseks saab meetodi `map` poolt tagastatud loetelurivi muutujas `showTodos` olevate tegevusobjektidele vastavatest komponendi `TodoItem` elementidest (joonis 91) [221]. Seejuures teostatakse mitmeid sidumisi JavaScripti meetodiga `bind` [241].

```
//...//
            var todoItems = shownTodos.map(function (todo) {
               return (
                  <TodoItem
                     key={todo.id}
                     todo={todo}
                     onToggle={this.toggle.bind(this, todo)}
                     onDestroy={this.destroy.bind(this, todo)}
                     onEdit={this.edit.bind(this, todo)}
                     editing={this.state.editing === todo.id}
                     onSave={this.save.bind(this, todo)}
                     onCancel={this.cancel}
                  />
               );
            }, this);
//...//
```

Joonis 91. app.jsx ning `renderi` `todoItems` [24]

Muutujad `activeTodoCount` ja `completedCount` sisaldavad infot mittetehtud ja tehtud tegevusobjektide arvu kohta (joonis 92). Neist esimese puhul on kasutatud JavaScripti meetodit `reduce`, mis töötleb loetelurivis olevad tegevusobjekte ning salvestab tulemuseks saadud arvu akumulaatorisse `accum` [244]. Kui aktiivsete või lõpetatud tegevuste objektide arvu muutuja väärtus on suurem nullist, siis luuakse jalus komponendiga `TodoFooter`.

```
//...//
        var activeTodoCount = todos.reduce(function (accum, todo) {
           return todo.completed ? accum : accum + 1;
        }, 0);
        var completedCount = todos.length - activeTodoCount;

        if (activeTodoCount || completedCount) {
           footer =
              <TodoFooter
                 count={activeTodoCount}
                 completedCount={completedCount}
                 nowShowing={this.state.nowShowing}
                 onClearCompleted={this.clearCompleted}
              />;
        }
//...//
```

Joonis 92. app.jsx ning `render`i jaluskomponendi loomine [24]

Kui tegevusobjekte on rohkem kui null, siis luuakse märgendkasti element, millega saab korraga märkida tehtuks või mittetehtuks kõik tegevusobjektid. Need kuvatakse loendina loodavas elemendis klassiga `todo-list`, kasutades eelnevalt defineeritud muutujat `todoItems` (joonis 93).

```
//...//
            if (todos.length) {
               main = (
                  <section className="main">
                     <input
                        className="toggle-all"
                        type="checkbox"
                        onChange={this.toggleAll}
                        checked={activeTodoCount === 0}
                     />
                     <ul className="todo-list">
                        {todoItems}
                     </ul>
                  </section>
               );
            }
//...//
```

Joonis 93. app.jsx ning `render`i märgendkast ja loetelu [24]

Komponendiklassi `TodoApp` funktsiooni `render` lõpus tagastatakse `<div>`-sildiga element, mille päises on sisend uute tegevusobjektide pealkirjade sisestamiseks koos eelnevalt defineeritud sündmuste halduse funktsionaalsusega. Pärast päist kuvatakse kasutajale muutujates `main` ja `footer` olevat sisu, kui nende kuvamise tingimuskontrollid on edukalt läbitud (joonis 94).

```
//...//
    return (
      <div>
         <header className="header">
            <h1>todos</h1>
            <input
               className="new-todo"
               placeholder="What needs to be done?"
               value={this.state.newTodo}
               onKeyDown={this.handleNewTodoKeyDown}
               onChange={this.handleChange}
```

```
                autoFocus={true}
              />
            </header>
            {main}
            {footer}
          </div>
      );
  }
  //...//
```

Joonis 94. app.jsx ning `<div>` [24]

Autor annab Reactile õppimise kategoorias kaks punkti, sest testrakenduse mõistmisele ning kirjalikul kujul selgitamisele kulus ligikaudu kümme tundi ehk rohkem kui kaheksa ning vähem kui kaksteist tundi.

2.3.9 TodoMVC testrakendusel põhinev jõudlus

Lokaalses masinas käivitati testrakendus veebilehitsejas Chrome ning lasti YSlow pistikprogrammil hinnata jõudlust. Üldine hinne jõudluse eest oli 78 punkti 100 võimalikust punktist. YSlow soovitas kasutada sisuedastusvõrku, kokku pakkida gzip-meetodiga failid base.css, index.css, director.js, utils.js, todoModel.js, base.js, index.js, JSXTransformer.js ja react-with-addons.js, lisada Expires-päiseid mitmetele failidele, teha vähem HTTP-päringuid ehk kombineerida JavaScripti faile kokku vähemaks arvuks välisteks JavaScripti failideks ning minimeerida faile utils.js ja JSXTransforme.js.

Autor annab TodoMVC testrakendusel põhineva jõudluse eest ühe punkti.

2.4 Ember.js

2011. aastal tutvustati MVC mustrit kasutavat raamistikku Ember.js, mis oli SproutCore 2.0 raamistiku ümbernimetamise tulemus [9, lk 9]. Ember pakub mudeli ja vaate ning mudelitevahelist andmesidumist (*data binding*), sisseehitatud URL-tee haldamisvõimalust ning vaates teegi Handlebars mallide kasutusvõimalust [9, lk 10] [245, lk 8].

2.4.1 Startimishinnang

Raamistiku kodulehe alamlehel on esitatud Emberit kasutavad ettevõtted [246]. Nende hulka kuuluvad näiteks tehnoloogiafirmad Microsoft ning Yahoo, veebikaubandusega tegelev Groupon ja klienditeenindusplatform Zendesk.

Emberi kodulehel suunatakse arendaja kasutama raamistiku Node.js paketihaldurit npm [17] [247]. Pärast selle installeerimist, tuleb paigaldada npmi kaudu käsurea tööriist ember-cli käsuga `npm install -g ember-cli` ning luua mitmeid kaustasid sisaldav baasrakendus käsuga `ember new my-app` [247] [248]. Tulemuse käivitamiseks peab vastavast juurkaustast käivitama serveri käsuga `ember server` [248]. Selleks, et vältida Windowsi operatsioonisüsteemis esinevat vigast reaalajas taaslaadimist (*live reload*), on üks võimalus avada juurkasutas fail .ember-cli ja lisada õigesse kohta rida `"liveReload": false` [249] [250]. Kaustas `app\templates` asuva teegi Handlebars malli sisu muutis autor selliseks nagu joonisel 95.

```
<h2 id="title">Tere {{arvutus}}</h2>

{{outlet}}
```

Joonis 95. app\templates\application.hbs

Eelnevas mallis muutuja `arvutus` jaoks läks vaja luua kontroller failis applications.js (joonis 96) [251].

```
/*global Ember */
export default Ember.Controller.extend({
  arvutus: 2+2*2
});
```
Joonis 96. app\controllers\application.js

Joonisel 97 esitatud HTML-kood kasutab vaadeldud malli koos kontrolleriga ning veebilehitsejas Chrome kuvatakse teksti „Tere 6“. Eelnevalt pidi autor käivitama serveri käsuga `ember server` ja avama Chrome'is kohaliku serveri lehe pordil 4200.

```
<!DOCTYPE html>
<html>
  <head><title>Tere</title>
    {{content-for "head"}}
    {{content-for "head-footer"}}
  </head>
  <body>
    <script src="assets/vendor.js"></script>
    <script src="assets/my-app.js"></script>
    {{content-for "body"}}
    {{content-for "body-footer"}}
  </body>
</html>
```
Joonis 97. app\index.html

Eeltoodu vastab hinde üks saamise kriteeriumitele. Emberi failide allalaadimine soovituslikul viisil eeldab npmi installeerimist. Arendajal tuleb raamistiku lihtsaks kasutamiseks teha mitmeid samme. Bakalaureusetöö autor pidi tutvuma kontrolleri loomisega, baaskrakenduse mahuka struktuuriga ja ember-cli tööpõhimõttega. Leidub vähemalt kolm käesolevat raamistikku kasutavat edukat ärifirmat. Need on Yahoo, Groupon ja Zendesk. Autor annab startimishinnangu kategoorias ühe punkti.

2.4.2 Kogukond ja dokumentatsioon

2016. aasta veebruarikuu seisuga on raamistiku Githubi lehel 1172 vaatajat, 15694 tähega märkijat ja 3379 koodi harutajat [252]. Stack Exchange'i lehelt otsides leiab 18042 ember.js-sildiga küsimust, millest vastamata on 5124 [253] [254].

Raamistikul dokumentatsioonis leidub palju koodinäiteid ja selgitusi [255]. Iga alateema juures asub link, mis laseb arendajal muudatusi soovitada.

Eeltoodu vastab hinde kolm saamise kriteeriumitele. Githubis olevate vaatajate, tähega märkijate ning koodi harutajate summa on 20245. Stack Exchange'is moodustavad mittevastatuks märgitud küsimused 28,4% kõigist raamistiku nime kandva sildiga küsimustest. Lisaks eksisteerib koodinäiteid sisaldav dokumentatsioon, milles arendaja saab teha ettepanekuid muudatusteks. Autor annab kogukonna ja dokumentatsiooni kategoorias kolm punkti.

2.4.3 Arendustööriistade tugi

Eclipse IDE-le Eclipse Marketplace'i lehel märksõnale „Ember“ pistikprogrammi ei eksisteeri. Emberi poolt kasutatava malli Handlebars jaoks aga leidub märksõnale „Handlebars“ vastav DaVinci Studio, mis toetab HTML5 veebirakenduse arenduse parendamiseks avatud koodiga raamistikke nagu jQuery Mobile, KnockOut, Underscore, Handlebars ja Backbone [134].

Jetbrainsi toodetele WebStorm ja PhpStorm leidub pistikprogramm nimega Ember.js, mis pakub antud raamistiku kasutamiseks baastuge [256]. Arenduskeskkonnale Netbeans pistikprogrammid või muu Emberi tugi puudub: on esitatud ametlik teade, milles soovitakse tuge Ember.js raamistikule [257].

Tekstiredaktori Sublime Text jaoks eksisteerib Ember.js rakendustes navigeerimist lihtsutav pistikprogramm „Simple Ember.js Navigator“ ja koodilõigete kiirvaliku pakkuja „Ember.js snippets for Sublime Text 2“ [258] [259].

Visual Studiol on sissehitatud mitmete eesrakenduste raamistike tugi [137]. Ember on mainitud toetatavate hulgas.

Autor annab arendustööriistade toe eest kaks punkti, sest toetatud on vähemalt kaks, kuid mitte vähemalt neli vaadeldavat arendustööriista. Eclipse ning NetBeans ei paku konkreetseid lahendusi.

2.4.4 Testimistugi

Emberi dokumentatsioonis juhiste (*Guides*) alamlehel leidub eraldi peatükk ühiktestimise baasinfo kohta [260]. Koodinäidiste ja selgitustega saab arendaja ülevaate arvutatud atribuutide (*computed properties*), objektide meetodite ning jälgijate (*observers*) testimisest. Ühe koodinäite põhjal on selge, et ühiktestimisel on soovituslik kasutada QUnit testimisraamistikku [260]. Läbi kolmandate osapoolte lisade on toetatud ka teised testimisraamistikud [261].

Leidub Emberiga loodud rakenduse testimist seletav materjal koos koodinäidistega. Autor annab testimistoe kriteeriumi eest kolm punkti.

2.4.5 Litsents äri vaatepunktist

Raamistiku Ember.js puhul kehtib MIT litsents [262]. Emberi autoriõiguse hoidjad Yehuda Katz, Tom Dale ning Ember.js kaastöötajad ei vastuta raamistiku kasutamisega kaasneda võiva kahju eest. Litsentsis paiknevad autoriõiguse märkus ja litsentsi teadaanne peab eksisteerima kõigis koopiates või olulises mahus Emberiga loodavas tarkvaras [75].

Vastava raamistiku abil loodud rakendust lubatakse äriks kasutada, rakenduse lähtekood võib jääda salajaseks. Ember.js saab litsentsi eest kolm punkti.

2.4.6 Keele ja vormingu tugi

Emberi jaoks leidub internatsionaliseerimispakett Ember-i18n, mis pakub võimalust defineerida erinevates keeletes tõlkeid ja seejärel rakenduses neid kasutada [263]. Ember-i18n tõlkefailide mugavamaks haldamiseks eksisteerib Ember i18n Editor [264].

React teegi keele ja vormingu toe juures mainitud FormatJS kollektsiooni kuuluvad ka Emberile ja sellega seotud Handlebarsi mallidele loodud teegid Ember Intl ning Handlebars Intl. Seega saab Emberi rakendustes vormindada numbreid ja kuupäevi, kindlaid või suhtelisi ajamääranguid ning on toetatud mitmesse keelde tõlgitud ICU sõnumi formaadis sõnekujul vormindamine [214] [265] [266].

Kuna konkreetselt raamistikule Ember.js loodud keele ja formaatide tugi põhineb raamistikust eraldiseisva lisatarkvara võimalustel, siis autor annab keele ja vormingu toe eest ühe punkti.

2.4.7 TodoMVC testrakendusel põhinev keerukus

Vastavalt üldkriteeriumitele sai hinnatud testrakenduse JavaScripti faile, mis on esindatud punktis 2.4.8. Tulemused on toodud allpool.

1. Lähtekoodi loogiliste ridade arv kokku liidetult paikneb tabelis 10.

Tabel 10. Emberi testrakenduse ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
app.js	3	119
router.js	22	
todo_controller.js	31	
todos_controller.js	31	
todos_list_controller.js	12	
pluralize.js	6	
todo.js	6	
todo_input_component.js	8	

2. McCabe'i tsüklomaatiline keerukuse (*Cyclomatic complexity*) keskmine (liidetakse kokku iga vaadeldava JavaScripti faili tsüklomaatilise keerukuse arv ning jagatakse vaadeldavate failide arvuga) paikneb tabelis 11.

Tabel 11. Emberi testrakenduse tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
app.js	1	1,5
router.js	1	
todo_controller.js	2	
todos_controller.js	3	
todos_list_controller.js	1	
pluralize.js	2	
todo.js	1	
todo_input_component.js	1	

3. Halstead'i jõupingutuste (*effort*) keskmine funktsioonide põhjal (*Mean per-function Halstead effort*) oli 440, 9969907187362.
4. Hooldatavuse indeks (liidetakse kokku iga vaadeldava JavaScripti faili indeks ja jagatakse see vaadeldavate failide arvuga) paikneb tabelis 12.

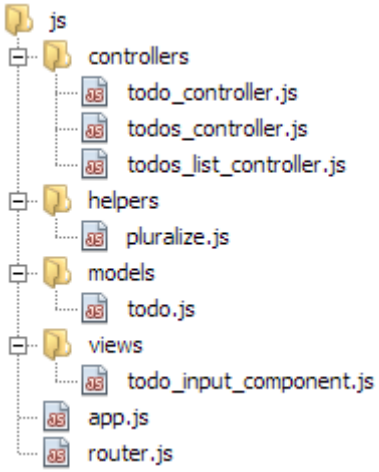
Tabel 12. Emberi testrakenduse hooldatavuse indeksi kategooria.

Fail	Hooldatavuse indeks (*Maintainability index*)	Keskmine
app.js	136, 18187152480502	130, 26
router.js	136, 18187152480502	
todo_controller.js	127, 43526506859993	
todos_controller.js	114, 78474360551247	
todos_list_controller.js	123, 21697156311129	
pluralize.js	140, 61270815647657	
todo.js	128, 5583024164997	
todo_input_component.js	135, 08788765970158	

Emberiga loodud testrakendus kuulub testrakenduse ridade arvu kategoorias 6. kohale (lisa „I. Keerukuse pingeread“, tabel 20), tsüklomaatilise keerukuse kategoorias 1. kohale (lisa „I. Keerukuse pingeread“, tabel 21), hooldatavuse indeksi kategoorias 8. kohale (lisa „I. Keerukuse pingeread“, tabel 22) ning funktsioonide põhjal saadud Halstead'i jõupingutuste (*effort*) keskmise kategoorias 6. kohale (lisa „I. Keerukuse pingeread“, tabel 23). Ember.js saab testrakendusel põhineva keerukuse kategoorias kaks punkti, sest tulemuseks on neljas pingereas esimese kaheksa hulka kuulumine.

2.4.8 TodoMVC testrakendusel põhinev õppimine ja õppimiskiirus

Järgneva MIT litsentsiga TodoMVC projekti testrakenduse autorid on Tom Dale, Addy Osmani ja Cory Forsyth (lisa „III. TodoMVC projekti repositooriumi sisu kasutamise litsents“) [25]. Failiga index.html asub samas kaustas kaust `js`, mille sisu on toodud joonisel 123.



Joonis 98. Kausta `js` sisu [25]

Failis app.js on loodud objekt `Ember.Application`, mis on kogu Emberi testrakenduse alguskoht (joonis 99). Nii tekkinud globaalsesse nimeruumi `Todos` defineeritakse edaspidi ülejäänud klassid. Selleks, et veebilehitseja andmehoiuga andmevahetust luua, kasutatakse projekti sõltuvusena eksisteerivat spetsiaalset veebilehitseja andmehoiu adapterit (*local storage adapter*), milles on nimeruumivõtme atribuut väärtusega `todo-emberjs` [267] [268]. Uue Emberi alamklassi defineerimiseks kasutatakse meetodit `extend` [269].

```
/*global Ember, DS, Todos:true */
window.Todos = Ember.Application.create();

Todos.ApplicationAdapter = DS.LSAdapter.extend({
   namespace: 'todos-emberjs'
});
```

Joonis 99. app.js ning `Todos` ja `ApplicationAdapter` [25]

Tegevuse objekti mudel defineeritakse failis todo.js klassi `DS.Model` abil (joonis 100) [270]. Mudeli `Todo` pealkirja atribuut on sõne tüüpi ja tegevuse tehtud või mittetehtud olekut tähistav atribuut tõeväärtuse tüüpi [271]. Failis todo.js ning järgnevates JavaScripti failides asub vaadeldav kood JavaScripti moodulimustri (*JavaScript Modul Pattern*) anonüümse sulundi (*closure*) konstruktsiooni sees (joonis 37) [159].

```
//...//
   Todos.Todo = DS.Model.extend({
       title: DS.attr('string'),
       isCompleted: DS.attr('boolean')
   });
//...//
```

Joonis 100. todo.js [25]

Tekstisisestusele järgneva fookuse tarvis on defineeritud klassi `Ember.Text` alamklassina `TodoInputComponent` (joonis 101) [272]. Selle sees olev funktsioon `focusOnInsert` ootab Ember.js komponendi DOM-i lisamisel käivituvat sündmust `didInsertElement`. Vaadeldava sündmuse toimumisel rakendatakse tekstisisestuskohale jQuery abil meetodit `focus`.

```
//...//
   Todos.TodoInputComponent = Ember.TextField.extend({
      focusOnInsert: function () {
         // Re-set input value to get rid of a redundant text selection
         this.$().val(this.$().val());
         this.$().focus();
      }.on('didInsertElement')
   });
//...//
```

Joonis 101. todo_input_component.js [25]

Failis pluralize.js on loodud abistaja (*helper*) `pluralize`, mis võtab argumendiks sõne ning ühe numbri (joonis 102) [273]. Klassist `Ember.Inflector` kasutatakse funktsionaalsust, mis muudab vajadusel ainsuse vormis oleva sõne mitmuse vormiks [274]. Seda abistajat rakendatakse failis index.html kujul `{{pluralize 'item' remaining.length}}`. Tulemusena näeb rakenduse kasutaja sõna *item* ainsuse või mitmuse vormi vastavalt `remaining.length` arvulisele väärtusele.

```
//...//
   Ember.Handlebars.helper('pluralize', function (singular, count) {
      /* From Ember-Data */
      var inflector = Ember.Inflector.inflector;

      return count === 1 ? singular : inflector.pluralize(singular);
   });
//...//
```

Joonis 102. pluralize.js [25]

Kontroller `TodosListController` on objektide loetelurivi haldamiseks loodud klassi `ArrayController` alamklass (joonis 103) [275]. Alates Ember 2.0-ist taolist konstruktsiooni enam ei kasutatata: selle asemel peab kasutama `Ember.Controller.extend`-i [276]. Atribuudiga `needs` luuakse ligipääsuseos kontrolleriga `TodosController`: seega saab `TodoListController` ligi `TodosController`'ile [277] [278]. Alates Ember 2.0-ist taolist konstruktsiooni enam ei kasutatata: selle asemel peab kasutama `todos: Ember.inject.controller()`-it [279]. Atribuudile `allTodos` vastav `Ember.computed.alias` loob võimaluse selle argumendis toodud atribuudile `controllers.todos` suunatud manipuleerimiskäske `get` ja `set` välja kutsuda hoopis atribuudil nimega `allTodos` [280]. Atribuut `itemController` määrab, et iga loetelurivis oleva mudeli puhul kasutatakse sellele vastavat kontrollerit `TodoController` [281]. Kirjutades funktsiooni `canToggle` lõppu `.property()`, saab seda funktsiooni vaadelda kui atribuuti [282]. Atribuudina koheldav `canToggle` sõltub aga atribuutidest `allTodos.length` ning `@each.isEditing`, seetõttu kirjutatakse need `.property()` argumentideks [282]. Eriline võti `@each` paneb Emberi jälgima loetelurivis mudelite `isEditing` atribuuti ja automaatselt uuendama loetelurivi muudatuste korral atribuudina koheldava funktsiooni väärtust [283]. Funktsiooni `canToggle` sees kasutatav meetodi `isAny` tagastusväärtus on tõene, kui mingi loetelus vaatluse all olevatest atribuutidest `isEditing` on tõene [284].

```
//...//
   Todos.TodosListController = Ember.ArrayController.extend({
      needs: ['todos'],
      allTodos: Ember.computed.alias('controllers.todos'),
      itemController: 'todo',
      canToggle: function () {
         var anyTodos = this.get('allTodos.length');
         var isEditing = this.isAny('isEditing');

         return anyTodos && !isEditing;
      }.property('allTodos.length', '@each.isEditing')
   });
//...//
```

Joonis 103. todos_list_controller.js [25]

Kui `ArrayController` oli vajalik objektide loetelurivi jaoks, siis üheainsa objekti kontolleri `TodoController` defineerimisel kasutatakse `ObjectController`'i laiendamist [285]. Alates Ember 2.0-ist taolist konstruktsiooni enam ei kasutatata: selle asemel peab laiendama klassi `Ember.Controller` [286]. Käesoleva kontrolleri sees luuakse atribuudid `isEditing` ning `bufferedTitle` (joonis 104). Viimase definitsioonis luuakse meetodiga `oneWay` argumendis toodud atribuudiga `title` koopialaadne viide atribuudile `bufferedTitle`, millesse saab salvestada väärtusi atribuuti `title` mõjutamata [287]. Abistaja (*helper*) `action` väli kontrolleri sees sisaldab neid funktsioone, mida kasutaja kutsub välja failis index.html väljaga `action` elementidele klõpsates [288]. Funktsioon `editTodo` muudab atribuudi `isEditing` tõeväärtuse tõeseks.

```
//...//
   'use strict';
   Todos.TodoController = Ember.ObjectController.extend({
      isEditing: false,

      // We use the bufferedTitle to store the original value of
      // the model's title so that we can roll it back later in the
      // `cancelEditing` action.
      bufferedTitle: Ember.computed.oneWay('title'),

      actions: {
```

```
        editTodo: function () {
            this.set('isEditing', true);
        },
//...//
```

Joonis 104. todo_controller.js ning `bufferedTitle` ja `actions` algus [25]

Käesoleva kontrolleri atribuudi `actions` objekti sees paiknevas funktsioonis `doneEditing` välditakse üleliigset funktsiooni täitmiskorda meetodi `debounce` rakendamisega: ebasoovitav tulemus delegeeritakse funktsioonile `removeTodo` (joonis 105) [289]. Funktsioon `doneEditing` salvestab kasutaja redigeerimise tulemuse mudelis `Todo` kasutaja poolt sisestatud pealkirjaga. Manipuleeritava mudelini jõudmiseks kastutatakse meetodit `get`, mudeli pealkirja atribuudile muutuja `bufferedTitle` väärtuse omistamiseks meetodit `set` ning muudatuste salvestamiseks meetodit `save` [290].

```
//...//
        doneEditing: function () {
            var bufferedTitle = this.get('bufferedTitle').trim();

            if (Ember.isEmpty(bufferedTitle)) {
                // The `doneEditing` action gets sent twice when the user hits
                // enter (once via 'insert-newline' and once via 'focus-out').
                // We debounce our call to 'removeTodo' so that it only gets
                // made once.
                Ember.run.debounce(this, 'removeTodo', 0);
            } else {
                var todo = this.get('model');
                todo.set('title', bufferedTitle);
                todo.save();
            }

            // Re-set our newly edited title to persist its trimmed version
            this.set('bufferedTitle', bufferedTitle);
            this.set('isEditing', false);
        },
//...//
```

Joonis 105. todo_controller.js ning `actions`i `doneEditing` [25]

Kontrolleri `TodoController` atribuudi `actions` sees paiknevad lisaks funktsioonid `cancelEditing` ning `removeTodo` (joonis 106). Esimene taastab redigeerimisele eelnenud tegevuse objekti pealkirja, teine kutsub välja funktsiooni `removeTodo`, mis on defineeritud käesolevas kontrolleris väljaspool atribuuti `actions`.

```
//...//
        cancelEditing: function () {
            this.set('bufferedTitle', this.get('title'));
            this.set('isEditing', false);
        },

        removeTodo: function () {
            this.removeTodo();
        }
    },
//...//
```

Joonis 106. todo_controller.js ning `actions`i `cancelEditing` ja `removeTodo` [25]

Faili todo_controller.js kaks viimasena defineeritud funktsiooni on `removeTodo` ja `saveWhenCompleted` (joonis 107). Esimene kustutab tegevuse objekti klassi `DS.Model` alamklasside kustutamiseks mõeldud meetodiga `deleteRecord` ning muudab tehtu püsivaks salvestusmeetodiga `save` [291]. Funktsioon `saveWhenCompleted` jälgib meetodiga `observes` atribuuti `isCompleted` ning jälgitava atribuudi muutuste korral salvestab vastava tegevuse objekti mudeli [292].

```
//...//
      removeTodo: function () {
         var todo = this.get('model');

         todo.deleteRecord();
         todo.save();
      },

      saveWhenCompleted: function () {
         this.get('model').save();
      }.observes('isCompleted')
   });
//...//
```

Joonis 107. todo_controller.js ning `removeTodo` ja `saveWhenCompleted` [25]

Kontroller `TodosController` on objektide loetelurivi haldamiseks mõeldud klassi `ArrayController` alamklass [275]. Kontrolleri `TodosController` atribuudi `actions` objekti kuuluvad funktsioonid `createTodo` ning `clearCompleted` (joonis 108). Funktsioon `createTodo` kasutab failis index.html väljaga `action` seotud väärtust muutujast `newTitle`, millesse salvestub kasutaja sisestatud tegevuse pealkiri. Uus tegevuse objekt luuakse meetodiga `createRecord` salvestuskohta `store`, seejärel tühjendatakse pealkirjasisestuskoht [293].

```
//...//
   Todos.TodosController = Ember.ArrayController.extend({
      actions: {
         createTodo: function () {
            var title, todo;
            // Get the todo title set by the "New Todo" text field
            title = this.get('newTitle').trim();
            if (!title) {
               return;
            }
            // Create the new Todo model
            todo = this.store.createRecord('todo', {
               title: title,
               isCompleted: false
            });
            todo.save();
            this.set('newTitle', ''); // Clear the "New Todo" text field
         },
//...//
```

Joonis 108. todos_controller.js ning `actions`i `createTodo` [25]

Funktsioon `clearCompleted` kasutab käesoleva kontrolleri sees defineeritud atribuuti `completed`, et kutsuda meetodiga `invoke` välja nende tegevusobjektide kustutamine, mis on märgitud tehtuks (joonis 109) [294]. Atribuutide `remaining` ja `completed` loetelurivi kujul väärtused saadakse meetodiga `filterBy` kõikide mudeli `Todo` objektide atribuutide `isCompleted` tõeväärtuste põhjal [295].

```
//...//
         clearCompleted: function () {
            var completed = this.get('completed');
            completed.invoke('deleteRecord');
            completed.invoke('save');
         }
       },
      /* properties */

      remaining: Ember.computed.filterBy('model', 'isCompleted', false),
      completed: Ember.computed.filterBy('model', 'isCompleted', true),
//...//
```

Joonis 109. todos_controller.js ning `actions`i `clearCompleted` [25]

Faili todos_controller.js lõpus on defineeritud funktsioon `allAreDone` (joonis 110), mida kasutatakse failis index.html aliase `allTodos` atribuudina [282]. Atribuudina ko-

heldav `allAreDone` sõltub atribuutidest `length` ning `completed.length`: seetõttu kirjutatakse need `.property()` argumentideks [282]. Vaadeldava funktsioonatribuudi eesmärk on võimaldada kasutajal korraga märkida kõik tegevused märgistuskastis klõpsates tehtuks või mittetehtuks.

```
//...//
      allAreDone: function (key, value) {
         if (value !== undefined) {
            this.setEach('isCompleted', value);
            return value;
         } else {
            var length = this.get('length');
            var completedLength = this.get('completed.length');
            return length > 0 && length === completedLength;
         }
      }.property('length', 'completed.length')
   });
//...//
```

Joonis 110. todos_controller.js ning `allAreDone` [25]

Failis router.js on defineeritud ressursi `todos` URL-haldaja grupp, mis loob URL-teed `'/'`, `'/active'` ja `'/completed'` (joonis 111) [296]. Klassi `Ember.Route` alamklassi `TodosRoute` funktsioon `model` tagastab kõik mudeli `Todo` salvestatud objektid [297].

```
//...//
   Todos.Router.map(function () {
      this.resource('todos', { path: '/' }, function () {
         this.route('active');
         this.route('completed');
      });
   });

   Todos.TodosRoute = Ember.Route.extend({
      model: function () {
         return this.store.find('todo');
      }
   });
//...//
```

Joonis 111. router.js ning `map` ja `TodosRoute` [25]

Klass `TodoIndexRoute` on klassi `TodosRoute` alamklass. Selles alamklassis fikseeritakse malli `todo-list` ning kontrolleri `TodosListController` kasutamine (joonis 112). Klassid `TodosActiveRoute` ja `TodosCompletedRoute` on omakorda klassi `TodoIndexRoute` alamklassid. Mõlemad tagastavad enda meetodi `model` kaudu kõik mudeli `Todo` salvestatud objektid, mis on vastavalt märgitud mittetehtuks või tehtuks. Mõlemad kasutavad enda meetodi `model` sees meetodit `filter`. Failis router.js defineeritu töötab koos failis index.html olevate abistajatega (*helpers*) `{{link-to}}`: kui kasutaja abistajaga elementidele vajutab, toimub URL-haldus ning näidatakse kas kõiki, tehtud või mittetehtud tegevusi [298].

```
//...//
   Todos.TodosIndexRoute = Todos.TodosRoute.extend({
      templateName: 'todo-list',
      controllerName: 'todos-list'
   });

   Todos.TodosActiveRoute = Todos.TodosIndexRoute.extend({
      model: function () {
         return this.store.filter('todo', function (todo) {
            return !todo.get('isCompleted');
         });
      }
   });

   Todos.TodosCompletedRoute = Todos.TodosIndexRoute.extend({
```

```
      model: function () {
         return this.store.filter('todo', function (todo) {
            return todo.get('isCompleted');
         });
      }
   });
//...//
```

Joonis 112. router.js [25]

Failis index.html on mitu piirkonda: päis, erinevatele failidele viitamised, testrakenduse autorite nimede info ja kehas paiknevad Handlebarsi mallid `todo-list` ning `todos` (joonis 113). Järgnevalt keskendub autor kahele mallile kui faili index.html põhilistele piirkondadele.

```
//...//
  <body>
    <script type="text/x-handlebars" data-template-name="todo-list">
      //...//
    </script>
    <script type="text/x-handlebars" data-template-name="todos">
      //...//
    </script>
    //...//
  </body>
//...//
```

Joonis 113. index.html ning kahe malli asukoht [25]

Handlebars malli `todo-list` sisu asub Handlebarsi abistaja `{{#if}}` mõjuala sees (joonis 114) [299]. Kui leidub tegevuste objekte, mida näidata, siis kuvatakse mallis tegevuste pealkirjad ja märgendkast kõikide tegevuste korraga tehtuks märkimiseks. Loetelurivi iga elemendi esitamiseks kasutatakse tsükli põhimõttel toimivat abistajat `{{each}}` [300]. Tegevused on seotud eelnevalt JavaScripti failides defineeritud funktsioonidega. Kui kasutaja näiteks redigeerib tegevuse objekti, siis klahvi Escape vajutamine käivitab funktsiooni `cancelEditing` või klahvi Enter vajutamine käivitab funktsiooni `doneEditing` [301]. Igal loetelu elemendiga on seotud abistaja `{{bind-attr}}` kaudu CSS-klassid `completed` ja `editing` vastavalt atribuutide `isCompleted` ja `isEditing` tõeväärtustele [302]. Kui kasutaja ei redigeeri mingi tegevuse pealkirja, siis saab seda tegevust kustutada funktsiooniga `removeTodo` seotud nupule klõpsamisega, märkida tegevus atribuudiga `isCompleted` seotud märgendkastist tehtuks või asuda topelklõpsuga nime peal pealkirja redigeerima.

```
//...//
      {{#if length}}
        <section id="main">
          {{#if canToggle}}
            {{input type="checkbox" id="toggle-all" checked=allTodos.allAreDone}}
          {{/if}}
          <ul id="todo-list">
            {{#each}}
              <li {{bind-attr class="isCompleted:completed isEditing:editing"}}>
                {{#if isEditing}}
                  {{todo-input type="text" class="edit" value=bufferedTitle focus-out="doneEdi-
ting" insert-newline="doneEditing" escape-press="cancelEditing"}}
                {{else}}
                  {{input type="checkbox" class="toggle" checked=isCompleted}}
                  <label {{action "editTodo" on="doubleClick"}}>{{title}}</label>
                  <button {{action "removeTodo"}} class="destroy"></button>
                {{/if}}
                </li>
            {{/each}}
          </ul>
        </section>
      {{/if}}
//...//
```

Joonis 114. index.html ning malli `todo-list` sisu [25]

Handlebarsi mall `todos` võimaldab päises kasutajal sisestada uue tegevuse pealkirja ning seejärel luua uue tegevuse objekti (joonis 115). Kohal, kus asub `{{outlet}}`, esitatakse malli `todo-list` sisu [303]. Kui leidub tegevuste objekte, mida näidata, siis kuvatakse ka jalus, milles kasutaja saab valida tehtud, mittetehtud või kõigi tegevuste esitamise. Kui leidub tehtud tegevusi, siis kuvatakse jaluses ka funktsiooniga `clearCompleted` seotud nupp.

```
//...//
      <section id="todoapp">
        <header id="header">
          <h1>todos</h1>
          {{todo-input id="new-todo" type="text" value=newTitle action="createTodo" placehol-
der="What needs to be done?"}}
        </header>
          {{outlet}}
          {{#if length}}
            <footer id="footer">
              <span id="todo-count"><strong>{{remaining.length}}</strong> {{pluralize 'item' re-
maining.length}} left</span>
              <ul id="filters">
                <li>
                  {{#link-to "todos.index" activeClass="selected"}}All{{/link-to}}
                </li>
                <li>
                  {{#link-to "todos.active" activeClass="selected"}}Active{{/link-to}}
                </li>
                <li>
                  {{#link-to "todos.completed" activeClass="selected"}}Completed{{/link-to}}
                </li>
              </ul>
              {{#if completed.length}}
                <button id="clear-completed" {{action "clearCompleted"}}>Clear completed</button>
              {{/if}}
            </footer>
          {{/if}}
      </section>
//...//
```

Joonis 115. index.html ning malli `todos` sisu [25]

Autor annab Emberile õppimise kategoorias kolm punkti, sest testrakenduse mõistmisele ning kirjalikul kujul selgitamisele kulus ligikaudu 7 tundi.

2.4.9 TodoMVC testrakendusel põhinev jõudlus

Lokaalses masinas käivitati testrakendus veebilehitsejas Chrome ning lasti YSlow pistikprogrammil hinnata jõudlust. Üldine hinne jõudluse eest oli 76 punkti 100 võimalikust punktist. YSlow soovitas kasutada sisuedastusvõrku, kokku pakkida gzip-meetodiga failid base.css, index.css, localstorage_adapter.js, handlebars.js, base.js, ember-data.js, ember.js, router.js, todos_controller.js, todo_controller.js ja jquery.js, lisada Expires-päiseid paljudele failidele, teha vähem HTTP-päringuid ehk kombineerida JavaScripti faile kokku vähemaks arvuks välisteks JavaScripti failideks ning minimeerida faile localstorage_adapter.js, ember-data.js ja ember.js.

Autor annab TodoMVC testrakendusel põhineva jõudluse eest ühe punkti.

2.5 KnockoutJS (teek)

Avatud lähtekoodiga (*open source*) teek KnockoutJS avaldati 2010. aastal [9, lk 7]. Knockout võimaldab mudel-vaade-vaatemudel (*Model-View-ViewModel* ehk MVVM) mustri realiseerimist, mille abil saab teha automaatseid kasutajaliidese uuendusi [9, lk 7–8]. Tegemist on kerge kaaluga (*lightweight*) teegiga, mille maht on minimeeritud ehk kompakteks muudetud kujul (*minified*) 54 kb [304].

2.5.1 Startimishinnang

Knockouti kodulehel puudub info ärifirmadest, mis seda teeki kasutavad [304]. Kasutades veebilehitsejas Chrome arendaja tööriistade (*developer tools*) funktsionaalsust allikate (*resource*) vahelehe skriptide (*scripts*) osas, on võimalik näha, milliseid faile avatud veebileht või rakendus kasutab. Bakalaureusetöö autor tuvastas sellisel viisil, et teegi KnockoutJS faile läheb vaja autotootja BMW Põhja-Ameerika kodulehel, sündmuste ja piletite agenditeenust pakkuva firma Eventim UK rakenduses ja veebipõhise aktsiatega kauplemise pakkuja AmeriTrade'i rakenduses[11].

Knockouti kodulehe alamlehelt sai alla laadida teegi faili knockout-3.4.0.js [308]. Autor uuris alamlehelt „*Live Examples*" kõige esimest näidet [309]. Selle põhjal sai kirjutatud käesolevat teeki kasutades fail teremaailm.html, milles esitatakse sõne koos arvutuse 2+2*2 tulemusega (joonis 116). Avades veebilehitsejas Chrome selle HTML5-faili, kuvatakse teksti „Tere 6". Samasse kausta HTML-failiga asetati knockout-3.4.0.js. Näide oli piisav, et lisasammudeta joonisel 116 olev kood luua.

```
<!DOCTYPE html>
<html>
  <head><title>Tere</title></head>
  <body>
    <script type='text/javascript' src='knockout-3.4.0.js'></script>
    <h1>Tere <span data-bind="text: arvutus"></span></h1>
    <script>
      var viewModel = function() {
        this.arvutus = 2+2*2;
      };
      ko.applyBindings(viewModel);
    </script>
  </body>
</html>
```

Joonis 116. teremaailm.html allika [309] esimese näite põhjal

Eeltoodu vastab hinde kolm saamise kriteeriumitele. Knockouti puhul piisab oluliste failide (knockout-3.4.0.js) allalaadimisest õigesse kausta ning veebilehitsejas käivitatavale failile viidete loomisest. Lisaks leidub vähemalt kolm edukat ärifirmat, mis kasutavad teeki KnockoutJS. Need on BMW, Eventim UK ja AmeriTrade. Autor annab startimishinnangu kategoorias kolm punkti.

2.5.2 Kogukond ja dokumentatsioon

2016. aasta märtsi seisuga on teegi Githubi lehel 600 vaatajat, 7239 tähega märkijat ja 1240 koodi harutajat [310]. Stack Exchange lehelt otsides leiab 15667 knockout.js-sildiga küsimust, millest vastamata on 3243 [311] [312].

Knockouti kodulehe alamlehel paikneb dokumentatsioon [313]. Arendajal pole võimalik seda muuta või soovitada muudatusi. Dokumentatsiooni sees leidub koodinäiteid.

Eeltoodu vastab hinde kaks saamise kriteeriumitele. Githubis olevate vaatajate, tähega märkijate ning koodi harutajate summa on 9079. Stack Exchange'is on mittevastatuks märgitud küsimused moodustamas 20,70% kõigist raamistiku nime kandva sildiga küsimustest. Lisaks eksisteerib dokumentatsioon, milles leidub koodinäiteid, kuid arendajal pole võimalik seda muuta või soovitada muudatusi. Autor annab kogukonna ja dokumentatsiooni kategoorias kaks punkti.

[11] [305] [306] [307]

2.5.3 Arendustööriistade tugi

Ainuke pistikprogramm Eclipse Marketplace'i lehe otsingutulemustest märksõnale „Backbone" oli DaVinci Studio. Sama tulemus oli ka märksõnaga „knockoutjs", sest DaVinci tutvustuses on kirjas, et see toetab HTML5 veebirakenduse arenduse parendamiseks avatud koodiga raamistikke nagu Knockout ja Backbone [134]. Jetbrainsi toodetele Knockouti pistikprogramm puudub.

Sublime Texti paketihaldurist on leitav pakett Sublime-KnockoutJS-Snippets, mida on viimati uuendatud 4 aastat tagasi [314]. See pakub KnockoutJS teegiga arendamist lihtsustavaid koodiväljavõtteid (*snippets*).

NetBeansile on loodud RESTful veebiteenuste põhimõttele vastava baaskoodi generaator, mis kasutab teeki Knockout ning loob koos vajalike failidega kaustade hierarhia [315] [316].

Visual Studiol on sissehitatud Knockouti tugi, mis pakub täielikku IntelliSense funktsionaalsust vaate mudelitele (*view models*) HTML-elementides atribuudiga `data-bind` [317]. Intellisense tähendab näiteks koodi lõpule viimist või mingi objekti kohta abistava info kuvamist [318].

Autor annab arendustööriistade toe eest kolm punkti, sest toetatud on vähemalt neli vaadeldavat arendustööriista.

2.5.4 Testimistugi

Roberto Messora on kirjutanud raamatu „Web App Testing Using Knockout.JS" [319]. Teoses antakse põhjalik abistavate koodinäidetega ülevaade Knockoutiga tehtud rakenduse ühiktestimisest testimisraamistikuga Jasmine. Lisaks leidub tasuta materjale veebis. Näiteks Jimmy Breck-Mckye blogis õpetatakse koodinäidete toel Knockouti rakendusi Jasmine'i testraamistikuga testima [320].

Knockoutiga loodud rakenduse testimist seletav materjal koos koodinäidistega eksisteerib. Autor annab testimistoe kriteeriumi eest kolm punkti.

2.5.5 Litsents äri vaatepunktist

Teegi KnockoutJS puhul kehtib MIT litsents [321]. Knockouti autoriõiguse hoidjad Steven Sanderson, Knockouti meeskond ning teised kaastöötajad ei vastuta raamistiku kasutamisega kaasneda võiva kahju eest. Litsentsis oleva autoriõiguse märkus ja litsentsi teadaanne peab eksisteerima kõigis koopiates või olulises mahus KnockoutJS teegiga loodavas tarkvaras [75].

Vastava raamistiku abil loodud rakendust lubatakse äriks kasutada, rakenduse lähtekood võib jääda salajaseks. KnockoutJS saab litsentsi eest kolm punkti.

2.5.6 Keele ja vormingu tugi

KnockoutJSi jaoks on eraldiseisva teegina loodud teegil i18next põhinev teek i18next-ko [322]. See võimaldab kasutada erinevaid tõlkeid. Kuvakeele või jälgitava muutuja väärtuse muutumisel toimub tõlke automaatne uuendus. Kuupäeva, numbri ja rahaühiku formaatide jaoks bakalaureusetöö autor ei leidnud spetsiaalselt Knockoutile loodud tarkvara. Asukohale vastavate kuupäevade formaatide jaoks saab näiteks kasutada JavaScripti rakendustele loodud teeki Moment.js [323].

Kuna konkreetselt teegile Knockout.js loodud keeletugi põhineb raamistikust eraldiseisva lisatarkvara võimalustel, siis autor annab keele ja vormingu toe eest ühe punkti.

2.5.7 TodoMVC testrakendusel põhinev keerukus

Vastavalt üldkriteeriumitele sai hinnatud testrakenduse JavaScripti faile, mis on esindatud punktis 2.5.8. Tulemused on toodud allpool.

1. Lähtekoodi loogiliste ridade arv kokku liidetult paikneb tabelis 13.

Tabel 13. Knockouti testrakenduse ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
app.js	107	107

2. McCabe'i tsüklomaatiline keerukuse (*Cyclomatic complexity*) keskmine (liidetakse kokku iga vaadeldava JavaScripti faili tsüklomaatilise keerukuse arv ning jagatakse vaadeldavate failide arvuga) paikneb tabelis 14.

Tabel 14. Knockouti testrakenduse tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
app.js	9	9

3. Halstead'i jõupingutuste (*effort*) keskmine funktsioonide põhjal (*Mean per-function Halstead effort*) oli 637, 7864270818525.
4. Hooldatavuse indeks (liidetakse kokku iga vaadeldava JavaScripti faili indeks ja jagatakse see vaadeldavate failide arvuga) paikneb tabelis 15.

Tabel 15. Knockouti testrakenduse hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**
app.js	128, 56449861200912	128, 56

Knockoutiga loodud testrakendus kuulub testrakenduse ridade arvu kategoorias 4. kohale (lisa „I. Keerukuse pingeread", tabel 20), tsüklomaatilise keerukuse kategoorias 14. kohale (lisa „I. Keerukuse pingeread", tabel 21), hooldatavuse indeksi kategoorias 10. kohale (lisa „I. Keerukuse pingeread", tabel 22) ning funktsioonide põhjal saadud Halstead'i jõupingutuste (*effort*) keskmise kategoorias 12. kohale (lisa „I. Keerukuse pingeread", tabel 23). KnockoutJS saab testrakendusel põhineva keerukuse kategoorias ühe punkti, sest tulemuseks on ühes pingereas esiviisikusse kuulumine.

2.5.8 TodoMVC testrakendusel põhinev õppimine ja õppimiskiirus

Järgneva MIT litsentsiga TodoMVC projekti testrakenduse autorid on Ashish Sharma ja Ryan Niemeyer (lisa „III. TodoMVC projekti repositooriumi sisu kasutamise litsents") [26]. Failiga index.html asub samas kaustas kaust `js`, mis sisaldab ainult faili app.js. Selles olev kood asub JavaScripti moodulimustri (*JavaScript Modul Pattern*) anonüümse sulundi (*closure*) konstruktsiooni sees (joonis 37) [159].

Knockouti kasutajaliidese osa ja selle poolt kasutatavad andmed on seotud läbi sidumissüsteemi (*bindings*) [324][12]. HTML-failide elementidele tuleb seega lisada atribuut `data-bind`, millele omistatakse seose nimi koos seose väärtusega. Nii saab näiteks siduda elemendile klõpsamise mingi funktsiooni väljakutsega. Faili app.js alguses on defineeritud abifunktsioon `keyhandlerBindingFactory`, mille ülesanne on lihtsustada kohandatud (*custom*) klahvivajutuse sidumise (*binding*) loomist (joonis 117). Abifunktsiooni sees paiknevat kohandatud sidujale omast tagasikutset (*callback*) `init` kutsutakse välja elemendile sidumise rakendamisel. Funktsioonis `wrappedHandler` kontrollitakse, missugust klahvi vajutati, ning tingimuskontrolli läbimisel seotakse JavaScripti funktsiooni `valueAccessor` kaudu saadud atribuut andmete ja sündmuse argumentidega. Funktsioon `newValueAccessor` seab `wrappedHandler`i vastavusse klahvivajutuse lõpetamisega ning asub joonise 117 alumisel real `ko.bindingHandlers`i väljakutse `init` argumentides `valueAccessor`i asemele [325].

```
//...//
  var ENTER_KEY = 13;
  var ESCAPE_KEY = 27;
  // A factory function we can use to create binding handlers for specific keycodes.
  function keyhandlerBindingFactory(keyCode) {
    return {
      init: function (element, valueAccessor, allBindingsAccessor, data, bindingContext) {
        var wrappedHandler, newValueAccessor;
        // wrap the handler with a check for the enter key
        wrappedHandler = function (data, event) {
          if (event.keyCode === keyCode) {
            valueAccessor().call(this, data, event);
          }
        };
        // create a valueAccessor with the options that we would
        // want to pass to the event binding
        newValueAccessor = function () {
          return {
            keyup: wrappedHandler
          };
        };
        // call the real event binding's init function
        ko.bindingHandlers.event.init(element, newValueAccessor, allBindingsAccessor, data, bindingContext);
      }
    };
  }
//...//
```

Joonis 117. app.js ning `keyhandlerBindingFactory` [26]

Kohandatud sidujaid registreeritakse `ko.bindingHandlers`i alamatribuutidena [324]. Joonisel 118 registreeritakse kohandatud sidujad `enterKey` ja `escapeKey` abifunktsiooniga `keyhandlerBindingFactory` klahvide Enter ning Escape sündmusi haldama.

```
//...//
  // a custom binding to handle the enter key
  ko.bindingHandlers.enterKey = keyhandlerBindingFactory(ENTER_KEY);

  // another custom binding, this time to handle the escape key
  ko.bindingHandlers.escapeKey = keyhandlerBindingFactory(ESCAPE_KEY);
//...//
```

Joonis 118. app.js ning `enterKey` ja `escapeKey` [26]

Failis app.js on defineeritud ka kohandatud siduja `selectAndFocus`, milles eksisteerivad tagasikutsed `init` ja `Update` (joonis 119). Neist viimast kutsutakse välja elemendile sidumise rakendamisel ning sidujaga seotud objektide muutuste korral [324]. Vaadeldava siduja ülesanne on kasutaja poolt redigeeritavale tegevuse objekti elemendile fookus tuua.

[12] Ülejäänud laused käesolevas lõigus kasutavad samuti viidet [324]

Fookuse tekitamiseks luuakse fookuse sündmuse haldaja dokumenteerimata funktsiooniga `ko.utils.registerEventHandler` [326].

```
//...//
  // wrapper to hasFocus that also selects text and applies focus async
  ko.bindingHandlers.selectAndFocus = {
    init: function (element, valueAccessor, allBindingsAccessor, bindingContext) {
      ko.bindingHandlers.hasFocus.init(element, valueAccessor, allBindingsAccessor, bindingContext);
      ko.utils.registerEventHandler(element, 'focus', function () {
        element.focus();
      });
    },
    update: function (element, valueAccessor) {
      ko.utils.unwrapObservable(valueAccessor()); // for dependency
      // ensure that element is visible before trying to focus
      setTimeout(function () {
        ko.bindingHandlers.hasFocus.update(element, valueAccessor);
      }, 0);
    }
  };
//...//
```

Joonis 119. app.js ning `selectAndFocus` [26]

Knockouti rakenduses on tegevuse objekti mudel `Todo` defineeritud joonisel 120 toodud viisil. Mudeli atribuudid on deklareeritud jälgitavateks (*observables*) [327]. Kui tuvastatakse atribuudiväärtustes muudatusi, siis uuendatakse automaatselt vaadet.

```
//...//
  // represent a single todo item
  var Todo = function (title, completed) {
    this.title = ko.observable(title);
    this.completed = ko.observable(completed);
    this.editing = ko.observable(false);
  };
//...//
```

Joonis 120. app.js ning `Todo` [26]

Rakenduses kasutatakse vaatemudelit `ViewModel`, mis võtab argumendiks tegevusobjektide loetelurivi (joonis 121) [327]. Vaatemudeli muutujasse `todos` salvestatakse jälgitav loetelurivi konstruktsiooniga `ko.observableArray` [328]. Seega uuendatakse loetelurivi elementides toimuvate väärtuste muudatuste korral automaatselt vaadet. JavaScripti meetodiga `map` rakendatakse iga vaatemudeli argumendiks olnud loetelurivi elemendile operaatorit `new` koos mudeli `Todo` konstruktoriga, et vaatemudeli `ViewModel` loetelurivi tüüpi muutuja `todos` iga element oleks `Todo` tüüpi [221] [329]. Muutujad `current` ja `showMode` luuakse jälgitavatena. Vaatemudelis olev funktsioon `filteredTodos` on aga arvutatud jälgitav (*computed observable*) ehk jälgitav, mis sõltub ühest või enamast jälgitavast [330]. Lisaks on funktsioon `filteredTodos` JavaScripti meetodi `bind` tõttu seotud otse vaatemudeli skoobiga ehk märksõna `this` funktsiooni sees viitab vaatemudeli sees märksõnaga `this` poolt viidatavale [241]. Taolist sidumist kasutavad veel mitmed failis app.js olevad funktsioonid. Arvutatud jälgitava `filteredTodos` ülesanne on filtreerida välja meetodiga `filter` tehtud, mittetehtud või kõik tegevuste objektid, mida tagastada [222].

```
//...//
  // our main view model
  var ViewModel = function (todos) {
    // map array of passed in todos to an observableArray of Todo objects
    this.todos = ko.observableArray(todos.map(function (todo) {
      return new Todo(todo.title, todo.completed);
    }));
    // store the new todo value being entered
    this.current = ko.observable();
    this.showMode = ko.observable('all');
    this.filteredTodos = ko.computed(function () {
      switch (this.showMode()) {
      case 'active':
        return this.todos().filter(function (todo) {
          return !todo.completed();
        });
      case 'completed':
        return this.todos().filter(function (todo) {
          return todo.completed();
        });
      default:
        return this.todos();
      }
    }.bind(this));
//...//
```

Joonis 121. app.js ning `ViewModel`i koodi algus [26]

Funktsioonid `add`, `remove` ja `removeCompleted` vastavalt lisavad tegevuse objekti, eemaldavad tegevuse objekti või eemaldavad kõik tehtuks märgitud tegevusobjektid (joonis 122). Selleks kasutatakse JavaScripti poolt pakutavaid tavalisi võimalusi. Lisamise funktsioonis toimub lisaks kasutaja sisendi olemasolu kontroll ning lisamise järel sisendimuutuja algväärtustamine tühja sõnega.

```
//...//
    // add a new todo, when enter key is pressed
    this.add = function () {
      var current = this.current().trim();
      if (current) {
        this.todos.push(new Todo(current));
        this.current('');
      }
    }.bind(this);

    // remove a single todo
    this.remove = function (todo) {
      this.todos.remove(todo);
    }.bind(this);

    // remove all completed todos
    this.removeCompleted = function () {
      this.todos.remove(function (todo) {
        return todo.completed();
      });
    }.bind(this);
//...//
```

Joonis 122. app.js ning `ViewModel`i `add`, `remove` ja `removeCompleted` [26]

Tegevuse redigeerimiseks on funktsioon `editItem` (joonis 123). See seab oma tegevuse objektist argumendi `item` jälgitavaks deklareeritud atribuudi `editing` tõeväärtuse tõeseks. Atribuudi `previousTitle` väärtuseks omistatakse argumendi jälgitavaks deklareeritud atribuudi `title` väärtus.

```
//...//
   // edit an item
    this.editItem = function (item) {
      item.editing(true);
      item.previousTitle = item.title();
    }.bind(this);
//...//
```

Joonis 123. app.js ning `Viewmodel`i `editItem` [26]

Funktsiooni `saveEditing` ülesandeks on salvestada kasutaja poolt redigeeritud tegevuse objekti pealkirja muudatus või eemaldada vaadeldav tegevuse objekt, kui pealkiri kustutati (joonis 124). Ebavajalike tühikute eemaldamiseks kasutatakse JavaScripti meetodit `trim` [91].

```
//...//
   // stop editing an item.  Remove the item, if it is now empty
    this.saveEditing = function (item) {
      item.editing(false);

      var title = item.title();
      var trimmedTitle = title.trim();

      // Observable value changes are not triggered if they're consisting of whitespaces only
      // Therefore we've to compare untrimmed version with a trimmed one to chech
      // whether anything changed
      // And if yes, we've to set the new value manually
      if (title !== trimmedTitle) {
        item.title(trimmedTitle);
      }

      if (!trimmedTitle) {
        this.remove(item);
      }
    }.bind(this);
//...//
```

Joonis 124. app.js ning `ViewModel`i `saveEditting` [26]

Funktsioon `cancelEditing` seab oma tegevuse objektist argumendi `item` atribuudi `editing` tõeväärtuse vääraks ning omistab atribuudi `title` väärtuseks atribuudi `previousTitle` väärtuse ehk taastab redigeerimisele eelnenud pealkirja (joonis 125). Arvutatud jälgitav funktsioon `completedCount` filtreerib JavaScripti meetodiga `filter` välja tehtuks märgitud tegevused ning tagastab nende arvu. Arvutatud jälgitav funktsioon `remainingCount` tagastab mittetehtud tegevuste arvu.

```
//...//
   // cancel editing an item and revert to the previous content
    this.cancelEditing = function (item) {
      item.editing(false);
      item.title(item.previousTitle);
    }.bind(this);

    // count of all completed todos
    this.completedCount = ko.computed(function () {
      return this.todos().filter(function (todo) {
        return todo.completed();
      }).length;
    }.bind(this));

    // count of todos that are not complete
    this.remainingCount = ko.computed(function () {
      return this.todos().length - this.completedCount();
    }.bind(this));
//...//
```

Joonis 125. app.js ning `ViewModel`i kolm funktsiooni [26]

Kirjutatav arvutatud jälgitav (*writable computed* observable) `allCompleted` tagastab muudatuste korral tõese tõeväärtuse, kui kõik tegevused on märgitud tehtuks, vastasel juhul väära tõeväärtuse (joonis 126) [331]. Funktsioonis kasutatav tagasikutse `write` võimaldab muuta korraga kõik vaatemudeli loetelurivis `todos` olevad tegevuste objektid tehtuks või mittetehtuks [331].

```
//.../// 
   // writeable computed observable to handle marking all complete/incomplete
    this.allCompleted = ko.computed({
      //always return true/false based on the done flag of all todos
      read: function () {
        return !this.remainingCount();
      }.bind(this),
      // set all todos to the written value (true/false)
      write: function (newValue) {
        this.todos().forEach(function (todo) {
          // set even if value is the same, as subscribers are not notified in that case
          todo.completed(newValue);
        });
      }.bind(this)
    });
//.../// 
```

Joonis 126. app.js ning `ViewModel`i kirjutatav arvutatud jälgitav [26]

Failis app.js defineeritud funktsiooni `getLabel` kasutatakse failis index.html selleks, et kuvada inglisekeelset sõna *item* mitmuses või ainsuses vastavalt arvulisele argumendile `count` (joonis 127). Funktsioonist `getLabel` allpool on defineeritud arvutatud jälgitav funktsioon, mida kutsutakse välja vaatemudeli muutujas `todos` toimuvate muudatuste korral. Siis salvestatakse funktsiooniga `ko.toJSON` muutuja `todos` andmed JSON-kujul veebilehitseja andmehoidu võtmega `todos-knockoutjs` [332]. Funktsioonile `ko.computed` on lisatud laiendaja (*extender*) `rateLimit`, mis ei lase salvestamist teostada rohkem kui kaks korda sekundis [333].

```
//.../// 
   // helper function to keep expressions out of markup
    this.getLabel = function (count) {
      return ko.utils.unwrapObservable(count) === 1 ? 'item' : 'items';
    }.bind(this);

    // internal computed observable that fires whenever anything changes in our todos
    ko.computed(function () {
      // store a clean copy to local storage, which also creates a dependency on
      // the observableArray and all observables in each item
      localStorage.setItem('todos-knockoutjs', ko.toJSON(this.todos));
    }.bind(this)).extend({
      rateLimit: { timeout: 500, method: 'notifyWhenChangesStop' }
    }); // save at most twice per second
  };
//.../// 
```

Joonis 127. app.js ning `ViewModel`i `getLabel` ja alati valmis arvutatud jälgitav [26]

Joonisel 128 toodud kood asub vaatemudeli ViewModel definitsioonist väljaspool. Muutujasse `todos` salvestatakse veebilehitseja andmehoiust pärit tegevuste objektid, JSON-kujust teisendatakse need vajalikule kujule funktsiooniga `ko.utils.parseJson`. Muutujale `viewModel` omistatakse operaatoriga `new` ja konstruktoriga `ViewModel` loodud vaatemudeli isend [329]. Knockouti rakenduse aktiveerimiseks kutsutakse välja `ko.applyBindings` argumendiga `viewModel` [327]. URL-haldamine seatakse sisse testrakenduse sõltuvuse Flatiron Director projekti URL-haldajaga `Router`, mis kasutab vaatemudelis olevat jälgitavat atribuuti `showMode` [242].

```
//.../// 
  // check local storage for todos
  var todos = ko.utils.parseJson(localStorage.getItem('todos-knockoutjs'));
  // bind a new instance of our view model to the page
  var viewModel = new ViewModel(todos || []);
  ko.applyBindings(viewModel);
  // set up filter routing
  /*jshint newcap:false */
  Router({ '/:filter': viewModel.showMode }).init();
//.../// 
```

Joonis 128. app.js ning `todos`, `viewModel` ja `Router` [26]

Failis index.html `<body>`-siltide alas (joonis 129) on päises kasutaja jaoks tegevuse objekti pealkirja sisestamise koht, mis on seotud jälgitava atribuudiga `current` ning mis klahvi Enter vajutamise järel käivitab funktsiooni `add`. Rakenduse funktsionaalsuse aspektist on olulised alad identifikaatoritega `main` ning `footer`. Esimene neist on kasutajale nähtav, kui tegevuse objekte on vähemalt üks, ning teine on nähtav, kui tehtuks märgitud tegevusi või mittetehtuks märgitud tegevusi on vähemalt üks [334]. Nende alade sisse jääv kood on kujutatud joonistel 130 ja 131.

```
//...//
    <section id="todoapp">
      <header id="header">
        <h1>todos</h1>
        <input id="new-todo" data-bind="value: current, valueUpdate: 'afterkeydown', enterKey:
add" placeholder="What needs to be done?" autofocus>
      </header>
      <section id="main" data-bind="visible: todos().length">
        //...//
      </section>
      <footer id="footer" data-bind="visible: completedCount() || remainingCount()">
        //...//
      </footer>
    </section>
    <footer id="info">
      //...//
    </footer>
//...//
```

Joonis 129. index.html ning oluline struktuur [26]

Ala `main` sisse jäävas koodis (joonis 130) on defineeritud märgendkast kõigi tegevuste tehtuks või mittetehtuks märkimiseks funktsiooniga `allCompleted`. Seejärel on loodud loetelu, milles esitatakse jälgitava funktsiooni `filteredTodos` poolt tagastatud loetelurivi iga element sidumise `foreach` kaudu [335]. Loetelu iga elemendile rakendatakse sidujat `css`, mis vastavalt atribuutide `completed` ja `editing` tõeväärtustele lisab või eemaldab sama nimega CSS-klasse [336]. Vaates sidujate kasutamine tähendab, et Knockout kasutab sidumiskonteksti (*binding context*), mis on sidujate hierarhiline viidete objekt [337]. Kuna loetelus kasutatav `foreach` loob alluvusseose (*child binding*) ja rakendus kasutab Knockouti rakenduse aktiveerimisel kasutatud vaatemudeli konteksti edasi, siis on vajalik HTML-failis olevate atribuutides `data-bind` viidata sidumiskonteksti juurkontekstile (*root context*) viidaga `$root` [337]. Iga loeteluelementi saab märgendkastiga märkida tehtuks, topeltklõpsuga asuda redigeerima ja nupule klõpsamisega kustutada. Redigeeritav väärtus on jälgitav pealkirja atribuut, mis salvestatakse klahvi Enter vajutamise järel ning ennistatakse klahvi Escape vajutamisel. Lisaks hallatakse fookuse tekitamist redigeeritavale elemendile.

```
//...//
       <input id="toggle-all" data-bind="checked: allCompleted" type="checkbox">
        <label for="toggle-all">Mark all as complete</label>
        <ul id="todo-list" data-bind="foreach: filteredTodos">
          <li data-bind="css: { completed: completed, editing: editing }">
            <div class="view">
              <input class="toggle" data-bind="checked: completed" type="checkbox">
              <label data-bind="text: title, event: { dblclick: $root.editItem }"></label>
              <button class="destroy" data-bind="click: $root.remove"></button>s
            </div>
            <input class="edit" data-bind="value: title, valueUpdate: 'afterkeydown', enterKey:
$root.saveEditing, escapeKey: $root.cancelEditing, selectAndFocus: editing, event: { blur:
$root.saveEditing }">
          </li>
        </ul>
//...//
```

Joonis 130. index.html ning `main` [26]

Identifikaatoriga `footer` alas (joonis 131) kuvatakse infot sellest, mitu mittetehtud tegevust veel eksisteerib. Lisaks saab kasutaja viidetele klõpsates filtreerida välja tehtud, mittetehtud või kõik tegevused. Siduja `css` lisab valitud viite elemendile klassi `selected` [336]. Kõikide tehtuks märgitud tegevuste kustutamise nupp on nähtav, kui eksisteerib selliseid tegevusi [334]. Vaadeldavale nupule klõpsamine eemaldab tehtud tegevused funktsiooniga `clearCompleted`.

```
//...//
       <span id="todo-count">
          <strong data-bind="text: remainingCount">0</strong>
          <span data-bind="text: getLabel(remainingCount)"></span> left
        </span>
        <ul id="filters">
          <li>
            <a data-bind="css: { selected: showMode() == 'all' }" href="#/all">All</a>
          </li>
          <li>
            <a data-bind="css: { selected: showMode() == 'active' }" href="#/active">Active</a>
          </li>
          <li>
            <a data-bind="css: { selected: showMode() == 'completed' }" href="#/completed">Comp-
leted</a>
          </li>
        </ul>
        <button id="clear-completed" data-bind="visible: completedCount, click: removeComple-
ted">Clear completed</button>
//...//
```

Joonis 131. index.html ning `footer` [26]

Autor annab teegile KnockoutJS õppimise kategoorias kolm punkti, sest testrakenduse mõistmisele ning kirjalikul kujul selgitamisele kulus ligikaudu 7 tundi.

2.5.9 TodoMVC testrakendusel põhinev jõudlus

Lokaalses masinas käivitati testrakendus veebilehitsejas Chrome ning lasti YSlow pistikprogrammil hinnata jõudlust. Üldine hinne jõudluse eest oli 85 punkti 100 võimalikust punktist. YSlow soovitas kasutada sisuedastusvõrku, kokku pakkida gzip-meetodiga failid base.css, index.css, base.js, director.js, app.js ja knockout-latest.js ning lisada Expires-päiseid mitmetele failidele.

Autor annab TodoMVC testrakendusel põhineva jõudluse eest kaks punkti.

2.6 Dojo (teek)

Dojo kodulehel nimetatakse seda teeki JavaScripti tööriistakomplektiks (*toolkit*), mis pakub keelevahendeid (*language utilities*), kasutajaliidese komponente (*UI components*) ja muid arenduse abivahendeid [338]. Dojole panid aluse Alex Russell ja Dylan Schiemann koos abilistega 2004. ja 2005. aastal [339]. Vaadeldav teek on jagatud mitmeks põhipaketiks [340]. Põhipakett `dojo` on oluline osa Dojo teegist: see sisaldab näiteks funktsionaalsust DOM-i manipuleerimiseks ja AJAX-i rakendamiseks. Lisaks eksisteerivad veel põhipaketid `dijit`, `dojox` ja `util`.

2.6.1 Startimishinnang

Dojo kodulehel on kirjas, et see teek on usaldatud (*trusted by*) näiteks tehnoloogiafirmade Cisco ja IBM, finantsteenuste firma JPMorgan ning hotellifirma Marriot poolt. Järelikult need firmad kasutavad Dojot.

Autor laadis teegi Dojo kasutamiseks vajalikud failid alla lehe download.dojotoolkit.org kaudu ning uuris kodulehe alamlehel olevat õpetust „*Hello Dojo*!“ [341] [342]. Selle põhjal sai kirjutatud käesolevat teeki kasutades fail teremaailm.html, milles esitatakse sõne

koos arvutuse 2+2*2 tulemusega (joonis 132). Avades selle HTML5-faili veebilehitsejas Chrome, kuvatakse teksti „Tere 6“. Samasse kausta HTML-failiga asetati alla laaditud failide kaustad `dojo`, `dijit` ja `dojox`. Õpetuse esimene ning teine koodinäide olid piisavad, et lisasammudeta alltoodud joonisel olev kood luua.

```
<!DOCTYPE html>
<html>
  <head><title>Tere</title></head>
  <body>
    <script src="dojo/dojo.js" data-dojo-config="async: true"></script>
    <h1 id="sisu">Tere </h1>
    <script>
      require([
          'dojo/dom',
          'dojo/dom-construct'
      ], function (dom, domConstruct) {
          var arvutus = 2+2*2
          var greetingNode = dom.byId('sisu');
          domConstruct.place('<var>'+arvutus+'</var>', greetingNode);
      });
    </script>
  </body>
</html>
```

Joonis 132. Dojo teremaailm.html allika [342] põhjal

Eeltoodu vastab hinde kolm saamise kriteeriumitele. Dojo teegi puhul piisab oluliste failide allalaadimisest õigesse kausta ning veebilehitsejas käivitatavale failile viidete loomisest. Lisaks leidub vähemalt kolm edukat ärifirmat, mis kasutavad vaadeldavat teeki. Need on Cisco, IBM ja JPMorgan. Autor annab startimishinnangu kategoorias kolm punkti.

2.6.2 Kogukond ja dokumentatsioon

2016. aasta märtsi seisuga on teegi peamise põhipaketi `dojo` Githubi lehel 175 vaatajat, 792 tähega märkijat ja 389 koodi harutajat [343]. Stack Exchange'i lehelt otsides leiab 8392 dojo-sildiga küsimust, millest vastamata on 2686 [344] [345].

Dojo kodulehe alamlehel „Docs“ paiknevad viited vähemdetailsele dokumentatsioonile „Dojo *Toolkit Reference Guide*“ ja detailsemale dokumentatsioonile „*The* Dojo *Toolkit API*“ [340] [346]. Mõlemad sisaldavad koodinäiteid. Dokumentatsiooni täiendamise loa saamiseks saab sõlmida organisatsiooniga The Dojo Foundation kaastöötaja litsentsi nõusoleku (*Contributor License Agreement, CLA*) [347]. Võib saata ka tagasisidet läbi ankeedi Google Forms, mille link on leitav nii vähemdetailse kui ka detailse dokumentatsiooni lehekülgede lõpus.

Eeltoodu vastab hinde üks saamise kriteeriumitele. Githubis olevate teegi peamise põhipaketi `dojo` vaatajate, tähega märkijate ning koodi harutajate summa on 1356. Stack Exchange'is on mittevastatuks märgitud küsimused moodustamas 32,01% kõigist teegi nime kandva sildiga küsimustest. Lisaks eksisteerib koodinäiteid sisaldav dokumentatsioon, milles arendaja saab teha ettepanekuid muudatusteks. Autor annab kogukonna ja dokumentatsiooni kategoorias ühe punkti, sest Githubis olevate vaatajate, tähega märkijate ning koodi harutajate summa on vähemalt 1000, kuid mitte vähemalt 2000.

2.6.3 Arendustööriistade tugi

Lehel Eclipse Marketplace märksõnale „dojo js“ leidub üle kümne vaste. Neist kõige esimene on pistikprogramm Tern Eclipse IDE, mis toetab mitmete JavaScripti raamistike seas ka Dojot [348]. Pärast installeerimist ja olemasoleva projekti Terni projektiks muutmist abistab arendajaid Dojo koodi lõpule viimise (*code completion*) funktsionaalsus [349].

Jetbrainsi toodetele leidub pistikprogramm Needs More Dojo, mis hõlbustab Dojo AMD (*Asynchronous Module Definition*) moodulite importimishaldust, refaktoreerimist ning teisi Dojole spetsiifiliste omaduste kasutamist [350] [351].

Sublime Text paketihaldurist ei leidu vasteid märksõnale „dojo“ ehk vaadeldavale teegile loodud lisapaketid puuduvad. Netbeansi arenduskeskkonnas kasutatavaid pistikprogramme pole Dojole loodud. Visual Studiol on sissehitatud mitmete eesrakenduste raamistike tugi, kuid Dojo kohta sellekohane info puudub. Ka Visual Studio pistikprogrammide otsing märksõnaga „dojo“ ei andnud kasulikke vasteid.

Autor annab arendustööriistade toe eest kaks punkti, sest toetatud on vähemalt kaks vaadeldavat arendustööriista.

2.6.4 Testimistugi

Dojo kodulehe alamlehel „D.O.H.: Dojo Objective Harness“ on õpetus Dojo Toolkit Community poolt arendatud ühiktestimise raamistiku D.O.H kasutamise kohta [352]. Vaadeldav abitööriist töötab nii veebilehitsejas kui ka keskkondades nagu Rhino ja node.js [352].

Leidub vastava raamistiku abil loodud rakenduse testimist seletav materjal koos koodinäidistega. Autor annab testimistoe kriteeriumi eest kolm punkti.

2.6.5 Litsents äri vaatepunktist

Teegi Dojo puhul kehtib kasutaja enda valiku järgi kas BSD 3 klausli litsents (*The BSD 3-Clause License*) või tastuta akadeemiline litsentsi versioon 2.1 (*Academic Free License version* 2.1) [353].

Järgnev BSD litsentsi kohta käiv lõik põhineb Dojo kodulehe alamlehe „Dojo Toolkit License“ ja Fossa Inc. vastavat litsentsi tutvustaval infol [210] [353]. BSD 3 klausli litsentsi puhul peab Dojo muudetud või muutmata kujul kasutav tarkvara lähtekood või binaarne vorm sisaldama litsentsis toodud autoriõiguse märkust ning litsentsi täisteksti. Dojo Foundationi ja selle kaastöötajate nimesid on keelatud eelneva kirjaliku loata Dojo põhjal saadud tarkvara edu suurendamise eesmärgil kasutada. Autoriõiguse omanik koos kaastöötajatega pole vastutav kahjude eest, mida Dojo kasutamine kaasa võib tuua.

Järgnev AFL litsentsi kohta käiv lõik põhineb Dojo kodulehe alamlehe „Dojo Toolkit License“ infol [353]. AFL litsents sisaldab 15 alapunkti, mida Dojo kasutaja peab järgima. Oluline on fakt, et äritegevus on lubatud. Samas litsentsiaari ja selle kaastöötajate nimesid ning kaubamärke on keelatud eelneva kirjaliku loata Dojo põhjal saadud tarkvara edu suurendamise eesmärgil kasutada. Lisaks peab säilitama või juurde lisama käesoleva teegiga seotud materjalides kõik autoriõiguste, patentide ja kaubamärkide teadaanded koos omistamismärkusega (*Attribution Notice*) ja teadaandega, et Dojo originaalmaterjale on kasutatud.

Vastava teegi abil loodud rakendust lubatakse äriks kasutada, rakenduse lähtekood võib jääda salajaseks. Dojo saab litsentsi eest kolm punkti.

2.6.6 Keele ja vormingu tugi

Dojo toetab erinevate keelte tõlgete kasutamist ja asukohapõhiste kuupäevade, arvude ning rahaühikute formaatide rakendamist [354]. Tõlgete tööriist on põhipaketis `dojo` paiknev i18n moodul [355]. Asukohale (*locale*) vastavad tekstid defineeritakse arendajate poolt ressursikimbu (*resource bundle*) failides ehk failides, mis sisaldavad mingile asukohale vastavate tekstide tõlgete JavaScripti objekti literaale (*object literal*) [355]. Kuupäevafor-

maatide jaoks saab kasutada põhipaketi `dojo` kaustas `date` olevat moodulit `locale` [356]. Numbri- ning rahaühikuformaatide tarvis leiduvad samas põhipaketis moodulid `number` ja `currency`.

Autor annab keele ja vormingu toe kriteeriumi eest kolm punkti.

2.6.7 TodoMVC testrakendusel põhinev keerukus

Vastavalt üldkriteeriumitele sai hinnatud testrakenduse JavaScripti faile, mis on esindatud alapunktis 2.6.8. Tulemused on toodud allpool.

1. Lähtekoodi loogiliste ridade arv kokku liidetult paikneb tabelis 16.

Tabel 16. Dojo testrakenduse ridade arvu kategooria.

Fail	Loogiliste ridade arv (*Logical LOC*)	Summa
main.js	22	357
computed.js	78	
empty.js	1	
TodoLocalStorage.js	53	
CSSToggleWidget.js	9	
Todo.js	51	
TodoEnter.js	8	
TodoEscape.js	8	
TodoFocus.js	6	
Todos.js	121	

2. McCabe'i tsüklomaatiline keerukuse (*Cyclomatic complexity*) keskmine (liidetakse kokku iga vaadeldava JavaScripti faili tsüklomaatilise keerukuse arv ning jagatakse vaadeldavate failide arvuga) paikneb tabelis 17.

Tabel 17. Dojo testrakenduse tsüklomaatilise keerukuse kategooria.

Fail	Tsüklomaatiline keerukus (*Cyclomatic complexity*)	Keskmine
main.js	1	7
computed.js	21	
empty.js	1	
TodoLocalStorage.js	11	

CSSToggleWidget.js	6	
Todo.js	10	
TodoEnter.js	2	
TodoEscape.js	2	
TodoFocus.js	3	
Todos.js	13	

3. Halstead'i jõupingutuste (*effort*) keskmine funktsioonide põhjal (*Mean per-function Halstead effort*) oli 615, 3148062854182.
4. Hooldatavuse indeks (liidetakse kokku iga vaadeldava JavaScripti faili indeks ja jagatakse see vaadeldavate failide arvuga) paikneb tabelis 18.

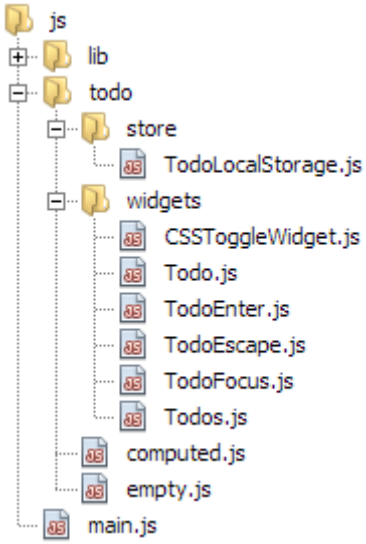
Tabel 18. Dojo testrakenduse hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**
main.js	110, 49950953452024	133, 48
computed.js	123, 23155668038147	
empty.js	171	
TodoLocalStorage.js	127, 10565617285711	
CSSToggleWidget.js	137, 8854556749669	
Todo.js	126, 4266381451266	
TodoEnter.js	139, 53162953707772	
TodoEscape.js	139, 53162953707772	
TodoFocus.js	135, 27887088751513	
Todos.js	124, 27986555917202	

Dojoga loodud testrakendus kuulub testrakenduse ridade arvu kategoorias 14. kohale (lisa „I. Keerukuse pingeread", tabel 20), tsüklomaatilise keerukuse kategoorias 11. kohale (lisa „I. Keerukuse pingeread", tabel 21), hooldatavuse indeksi kategoorias 6. kohale (lisa „I. Keerukuse pingeread", tabel 22) ning funktsioonide põhjal saadud Halstead'i jõupingutuste (*effort*) keskmise kategoorias 11. kohale (lisa „I. Keerukuse pingeread", tabel 23). Dojo saab testrakendusel põhineva keerukuse kategoorias null punkti, sest tulemuseks on pingeridade lõppu kuulumine.

2.6.8 TodoMVC testrakendusel põhinev õppimine ja õppimiskiirus

Järgneva MIT litsentsiga TodoMVC projekti testrakenduse autorid on James Thomas, Ed Chatelain ja Akira Sudoh (lisa „III. TodoMVC projekti repositooriumi sisu kasutamise litsents“) [27]. Failiga index.html asub samas kaustas kaust `js`, mille sisu struktuur on toodud joonisel 133. Dojo rakendustes luuakse tihti klassil `dijit._WidgetBase` põhinevaid vidinklasse (*widgets*) [357]. Bakalaureusetöö autor nimetab edaspidi neid lihtsalt klassideks.



Joonis 133. Kausta `js` sisu [27]

Failis main.js paiknev kood asub JavaScripti moodulimustri globaalse impordi (*global import*) konstruktsiooni sees (joonis 134) [159]. Käesoleval juhul võimaldatakse veebilehitseja akna objektile (*Window Object*) [358] viitavat globaalset võtmesõna `this` kasutada anonüümse funktsiooni sees muutuja `global` kaudu: globaalsete muutujate kasutus muutub koodi lugeja jaoks lihtsamini mõistetavaks [159].

```
(function (global) {
  //...//
})(this);
```

Joonis 134. main.js ning globaalse impordi konstruktsioon [27]

Testrakenduse seadistus (*configuration*) määratakse failis main.js globaalse funktsiooniga `require` (joonis 135) [359]. Konfiguratsioonimuutuja (*configuration variable*) `async` tõene tõeväärtus tagab failide AMD (*Asynchronous Module Definition*) asünkroonse laadimise võimaluse [360]. Sõnena esinev punkt `baseUrl`i väärtusena määrab faili index.html sisaldava kausta juurkaustaks, millest lähtuvalt saab ülejäänud ressursside asukohale viidata [361]. Funktsiooniga `callback` kaasatakse rakendusse lisamoodul Dojo parser, mis võimaldab vajadusel DOM-elemente muuta kasutajaliidese teegi Dojo Dijit komponentideks või muudeks objektideks [362] [363]. Konfiguratsioonimuutuja `deps` määratakse selleks, et kohe pärast Dojo laadimist laaditaks Dojo parser ressursiviitega `dojo/parser` [364]. Dojo parseri eduka laadimise järel käivitatakse funktsioonis `callback` Dojo parser [364]. Kasutatavate pakettide nimed ning asukohad on kirjas objektidena loetelurivis `packages`. Jooniselt 135 on lühiduse saavutamiseks välja jäetud kasutatavatest pakettidest lisaks paketile `dojo` kirjas olevad `dijix`, `dojox` ja `todo` koos oma asukohaviidetega. Viimane neist on juurkaustast lähtuvale asukohale viitava sõnega `'./js/todo'`. Välja `map` kasutatakse HTML-mallide DOM-puudeks muutmi-

seks mõeldud mooduli `dijit/_TemplatedMixin` identifikaatoris viida `dojo/cache` suunamiseks viidale `todo/empty` [364] [365]. Vastavalt dokumentatsioonis olevatele soovitustele on atribuudi `parseOnLoad` väärtuseks väär tõeväärtus [364]. Seega jäetakse rakenduse lehekülje parsimise ülesande haldamine Dojo parseriga väljadega `deps` ning `callback` seotud eelnevalt selgitatud mehhanismi hooleks.

```
//...//
  global.require = {
    async: true,
    baseUrl: '.',
    callback: function (parser) {
      parser.parse();
    },
    deps: ['dojo/parser'],
    packages: [
      {
        name: 'dojo',
        location: './node_modules/dojo'
      },
      //...//
      }
    ],
    map: {
      // TodoMVC application does not use template from file system
      'dijit/_TemplatedMixin': {
        'dojo/cache': 'todo/empty'
      }
    },
    parseOnLoad: false
  };
//...//
```

Joonis 135. main.js [27]

Failis main.js suunati viit `dojo/cache` kaustas `todo` olevale failile empty.js (joonis 136). Viimases olev kommentaar selgitab, et ühereal koodi kasutatakse paketis `dojo` oleva faili cache.js laadimise vältimiseks. Välditakse mooduli ja URL-argumentidega seotud sõnede vahemälus säilitamist [366]. Kood kujutab endast Dojo globaalse funktsiooniga `define` defineeritud moodulit, milles on tühi funktsioon [351].

```
// Used for avoiding dojo/cache from being loaded
define(function () {});
```

Joonis 136. empty.js [27]

Globaalse funktsiooniga `define` on failis TodoEnter.js defineeritud moodul, mis kasutab sõltuvusi failidest declare.js, lang.js ning _WidgetBase.js (joonis 137) [351]. Funktsiooniga `declare` luuakse tagastatav klass, mille ülemklassiks on `_WidgetBase` [367]. Mainitud klassi sees paiknev funktsioon `postCreate` kutsub käsuga `this.inherited(arguments)` välja ülemklassi `_WidgetBase` samanimelist funktsiooni ning reageerib alati originaalkontekstis klahvi alla vajutamise sündmusele `lang.hitch` rakendamise tõttu [97] [367] [368] [369]. Klahvi Enter vajutamise puhul käivitatakse meetodiga `emit` sündmus `change` [370].

```
define([
  'dojo/_base/declare',
  'dojo/_base/lang',
  'dijit/_WidgetBase'
], function (declare, lang, _WidgetBase) {
  // To use Dojo's super call method, inherited()
  /*jshint strict:false*/

  var ENTER_KEY = 13;

  /**
   * Widget that emits `change` event when the element it is applied to gets an keydown event of enter key.
   * Primary for IE which does not fire `change` event on `<input>` upon parent form's submission.
   * @class TodoEnter
   */
  return declare(_WidgetBase, {
    postCreate: function () {
      this.inherited(arguments);
      this.on('keydown', lang.hitch(this, function (event) {
        if (event.keyCode === ENTER_KEY) {
          this.emit('change');
        }
      }));
    }
  });
});
```

Joonis 137. TodoEnter.js [27]

Failis TodoEscape.js (joonis 138) on kasutatud samasid võtteid nagu failis TodoEnter (joonis 137). Erinevusena kontrollitakse klahvi Enter vajutamise asemel klahvi Escape vajutamist ning sündmuse `change` asemel käivitatakse sündmus `escape`.

```
define([
  'dojo/_base/declare',
  'dojo/_base/lang',
  'dijit/_WidgetBase'
], function (declare, lang, _WidgetBase) {
  // To use Dojo's super call method, inherited()
  /*jshint strict:false*/

  var ESCAPE_KEY = 27;

  /**
   * Widget that emits `escape` custom event when the element it is applied to gets an keydown event of escape
key.
   * @class TodoEscape
   */
  return declare(_WidgetBase, {
    postCreate: function () {
      this.inherited(arguments);
      this.on('keydown', lang.hitch(this, function (event) {
        if (event.keyCode === ESCAPE_KEY) {
          this.emit('escape');
        }
      }));
    }
  });
});
```

Joonis 138. TodoEscape.js [27]

Globaalse funktsiooniga `define` on failis TodoFocus.js defineeritud moodul, mis kasutab sõltuvusi failidest declare.js, _WidgetBase.js ja FocusMixin.js (joonis 139) [351]. Funktsiooniga `declare` luuakse tagastatav klass, mille ülemklassideks on `_WidgetBase` ja `_FocusMixin` [371]. Käesoleva klassi sees paiknev funktsioon `_setShouldGetFocusAttr` on seotud failis index.html klassiidentifikaatorit `edit` omava HTML-elemendiga, millele viitavad `this.domNode` ning `this.focusNode`. Kui selle HTML-elemendi Dojo abil defineeritud atribuut `shouldGetFocus` on tõese tõeväärtusega, siis rakendatakse elemendile fookust [99].

```
define([
  'dojo/_base/declare',
  'dijit/_WidgetBase',
  'dijit/_FocusMixin' // For blur event support in data-dojo-attach-event
], function (declare, _WidgetBase, _FocusMixin) {
  'use strict';

  /**
   * Widget that places focus on the element it is applied to when `shouldGetFocus` property
becomes `true`.
   * @class TodoFocus
   */
  return declare([_WidgetBase, _FocusMixin], {
    _setShouldGetFocusAttr: function (value) {
      if (value) {
        (this.focusNode && this.focusNode.nodeType === Node.ELEMENT_NODE ? this.focusNode :
this.domNode).focus();
      }
    }
  });
});
```

Joonis 139. TodoFocus.js [27]

Failis Todo.js on defineeritud funktsiooni `define` sees moodul (joonis 140), mis kasutab sõltuvusena viidetele `'dojo/_base/declare'`, `'dojo/_base/lang'`, `'dojo/when'`, `'dojo/Deferred'`, `'dojo/Stateful'`, `'dijit/registry'`, `'dijit/_WidgetBase'`, `'dijit/_TemplatedMixin'`, `'dijit/_WidgetsInTemplateMixin'`, `'dojox/mvc/_InlineTemplateMixin'`, `'dojox/mvc/at'`, `'../computed'`, `'dojox/mvc/Element'`, `'./TodoEscape'` ja `'./TodoFocus'` vastavaid faile [351]. Dojo laadija poolt sõltuvuste loetelurivile järgneval moodulina salvestataval funktsioonil on argumendid `declare`, `lang`, `when`, `Deferred`, `Stateful`, `registry`, `_WidgetBase`, `_TemplatedMixin`, `_WidgetsInTemplateMixin`, `_InlineTemplateMixin`, `at` ja `computed` [351]. Funktsiooniga `declare` luuakse tagastatav klass, mille ülemklassideks on `_WidgetBase`, `_TemplatedMixin`, `_WidgetsInTemplateMixin` ja `_InlineTemplatedMixin` [371]. Käesoleva klassi sees paiknevas funktsioonis `startup` kutsutakse käsuga `this.inherited(arguments)` välja ülemklassi `_WidgetsInTemplateMixin` samanimelist funktsiooni: see on kõige viimasena päritud käesoleva nimega funktsiooni sisaldav ülemklass [369] [371] [372]. Seejärel kontrollitakse, kas failis index.html defineeritud atribuut `todosWidget` eksisteerib ning lisatakse klassile töötleja (*handler*) mälulekkeid vältida aitava meetodiga `own` [373]. Töötlejaks on funktsioon `computed`, mis on defineeritud failis computed.js ning mille tööpõhimõtet selgitavad selles failis olevad joonistelt välja jäetud kommentaarid [374]. Tööleja tagastab tegevuse objekti kohta tõeväärtuse, kas `todosWidget`i atribuut `editedTodo` ja atribuut `this.target` on võrdsed [232] [374].

```
define([
  'dojo/_base/declare',
  // ... //
  './TodoFocus'
], function (declare,
  lang,
  // ... //
  computed) {
  // ... //
  return declare([_WidgetBase, _TemplatedMixin, _WidgetsInTemplateMixin, _InlineTemplateMixin], {
    startup: function () {
     this.inherited(arguments);
     if (!this.todosWidget) {
       throw new Error('this.todosWidget property should be there before this widgets starts up: ' + this);
```

```
      }
      this.own(computed(this, 'isEditing', function (editedTodo) {
        return editedTodo === this.target;
      }, at(this.todosWidget, 'editedTodo')));
    },
//...//
```

Joonis 140. Todo.js algus [27]

Tegevuse objekti pealkirja redigeerimiseks loodud funktsioon `editTodo` (joonis 141) kasutab ressursiviidale `dojo/Stateful` vastavat liidest [375]. Järelikult saab kasutada atribuutidega manipuleerimiseks meetodeid `get` ning `set`. Redigeerimistegevusele suunatud tegevuse objekti pealkiri salvestatakse võtmesõnale `this` vastava objekti argumenti `originalTitle` [232] [375]. Failis index.html defineeritud atribuudi `todosWidget` atribuuti `editedTodo` salvestatakse redigeerimisele suunatud tegevuse objekt [232] [375].

```
//...//
   editTodo: function () {
      this.set('originalTitle', this.target.get('title'));
      this.todosWidget.set('editedTodo', this.target);
    },
//...//
```

Joonis 141. Todo.js ning `editTodo` [27]

Funktsioon `invokeSaveEdits` (joonis 142) kutsutakse välja vastavalt failis index.html koodilõigule `data-dojo-attach-event="blur: invokeSaveEdits` siis, kui selle koodilõiguga seotud HTML-element kaotab fookuse [96]. Käesoleva funktsiooni sees luuakse uus asünkroonsete tagasikutsete halduri klassi `Deferred` objekt `dfd`, mille töö lõpetatakse kutsega `resolve`, kui tagasikutse halduris `when` toimunud funktsiooni `saveEdits` väljakutse on edukalt lõppenud [376] [377]. Probleemide korral lõpetatakse objekti `dfd` töö kutsega `reject`. Lõpuks tagastatakse objekti `dfd` ressursiviitega `dojo/promise/Promise` seotud atribuut `promise`, mis kujutab endast JavaScripti lubaduse (*promise*) Dojo implementatsiooni [376] [378] [379].

```
//...//
   invokeSaveEdits: function () {
      // For handling input's blur event, make sure change event has been fired
      var dfd = new Deferred();
      setTimeout(lang.hitch(this, function () {
        when(this.saveEdits(), function (data) {
          dfd.resolve(data);
        }, function (e) {
          dfd.reject(e);
        });
      }), 0);
      return dfd.promise;
    },
//...//
```

Joonis 142. Todo.js ning `invokeSaveEdits` [27]

Funktsioon `saveEdits` on mõeldud redigeerimisprotsessis tehtud muudatuste jäädvustamiseks (joonis 143). Redigeerimisele eelnenud pealkirja ning üleliigsete tühikuteta redigeeritud pealkirja väärtusi sisaldavate muutujate `originalTitle` ja `newTitle` erinevus tähendab, et tegevuse objekti pealkirja on muudetud [91]. Rakenduse kasutamisel ilmnes, et tingimuskontrollides kasutatav muutuja `blur` on tõese tõeväärtusega, kui pole tegemist sündmusega `submit`, sest sellisel juhul on argumendina kasutatav muutuja `event` väärtusega `undefined`, millest loogilise eituse operaatoriga `!` saadu on tõene tõeväärtus. Tingimuskontrollides kasutatav muutuja `goAhead` on tõese tõeväärtusega juhul, kui atribuut `isEditing` on tõene ning tegevuse objekti pealkirja muudeti. Taoline olukord esineb testrakenduse kasutamisel siis, kui kasutaja on muutnud pealkirja ning va-

jutab siis hiirega mõne teise tegevuse objekti pealkirjale, uue tegevuse sisestamise väljale või tegevuste filtreerimise linkidele. Redigeerimistegevusele suunatud tegevuse objekti pealkirja atribuudis `title` uuendatakse, kui vähemalt üks muutujatest `blur` või `goAhead` on tõene [232]. Järgneva tingimuskontrolli postitiivse läbimise korral salvestatakse muutuja `goAhead` tõese väärtuse ning muudetud pealkirja eksisteerimise korral muutujasse `progress` failis Todos.js defineeritud tegevuse objekti salvestamise funktsiooni `saveTodo` tagastusväärtus. Kui muudetud pealkirja pole, siis salvestatakse muutujasse `progress` failis Todo.js defineeritud tegevuse objekti eemaldamise funktsiooni `removeTodo` tagastusväärtus.

```
//...//
   saveEdits: function (event) {
      var progress;
      var originalTitle = this.get('originalTitle');
      var newTitle = lang.trim(this.target.get('title'));
      var goAhead = this.get('isEditing') && newTitle !== originalTitle;
      var blur = !event;
      if (blur || goAhead) {
        this.target.set('title', newTitle);
      }
      if (goAhead) {
        if (newTitle) {
          progress = this.todosWidget.saveTodo(this.target, originalTitle, this.target.get('completed'));
        } else {
          progress = this.removeTodo();
        }
      }
//...//
```

Joonis 143. Todo.js ning `saveEdits` [27]

Eelnevas lõigus käsitletud funktsiooni kood jätkub joonisel 144. Muutujasse `progress` salvestatakse tagasikutse halduris `when` toimuva atribuudile `editedTodo` väärtuse `null` omistamise õnnestumise tulemus, kui vähemalt üks muutujatest `blur` või `goAhead` on tõene [377]. Võtmesõnale `this` vajalikku konteksti säilitatakse konstruktsiooni `lang.hitch` rakendamisega [368]. JavaScripti meetodiga `preventDefault` ei lubata käesolevas funktsioonis sündmuse `submit` korral sellele sündmusele vastaval vaikimisi käivitataval sündmusteahelal alata [380]. Lõpuks tagastatakse muutuja `progress`.

```
//...//
     if (blur || goAhead) {
        progress = when(progress, lang.hitch(this, function () {
          this.todosWidget.set('editedTodo', null);
        }), lang.hitch(this, function (e) {
          this.todosWidget.set('editedTodo', null);
          throw e;
        }));
      }
      if (event && event.type === 'submit') {
        event.preventDefault();
      }
      return progress;
    },
//...//
```

Joonis 144. Todo.js ning `saveEdits`i jätkumine [27]

Failis index.html on redigeeritava elemendiga seotud klahvi Escape vajutamisel käivitatav failis Todo.js defineeritud funktsioon `revertEdits` (joonis 145). Selle sees kontrollitakse, kas atribuudi `isEditing` tõeväärtus on tõene. Positiivsel juhul seatakse atribuudi `todosWidget` atribuudi `editedTodo` väärtuseks `null` ja taastatakse failis Todos.js kirjas oleva funktsiooniga `replaceTodo` redigeerimisele eelnenud pealkiri. Taastatud pealkirjaga objekt omab liidese `Stateful` poolt pakutavaid manipuleerimisvõimalusi [375]. Viimaks eemaldatakse võtmesõnaga `this` seotud klass meetodiga `dest-`

royRecursive: tegemist on Dojo dokumentatsioonis soovitatud viisiga vidinklasside hävitamiseks [370].

```
//...//
   revertEdits: function () {
      if (this.get('isEditing')) {
        this.todosWidget.set('editedTodo', null);
        this.todosWidget.replaceTodo(this.target, new Stateful({
          id: this.target.get('id'),
          title: this.get('originalTitle'),
          completed: this.target.get('completed')
        }));
        this.destroyRecursive();
      }
    },
//...//
```

Joonis 145. Todo.js ning revertEdits [27]

Funktsioonide toggleCompleted ja removeTodo sees kasutatakse vastavalt failis Todos.js defineeritud tegevuse objekti salvestamise funktsiooni saveTodo ning tegevuse objekti eemaldamise funktsiooni removeTodo (joonis 146). Esimene neist muudab atribuudi completed tõeväärtuse vastupidiseks. Teine asub tagasikutse halduri when sees ning selle väljakutse õnnestumise korral kasutatakse tegevuse objekti hävitamiseks meetodit destroyRecursive [370] [377] [381].

```
//...//
    toggleCompleted: function () {
     this.todosWidget.saveTodo(this.target, this.target.get('title'), !this.target.get('completed'));
    },

    removeTodo: function () {
     return when(this.todosWidget.removeTodo(this.target), lang.hitch(this, this.destroyRecursive));
    }
  });
});
//...//
```

Joonis 146. Todo.js ning toggleCompleted ja removeTodo [27]

Failis Todos.js on defineeritud funktsiooni define sees moodul (joonis 147), mis kasutab sõltuvusena viidetele 'dojo/_base/declare', 'dojo/_base/array', 'dojo/_base/lang', 'dojo/router', 'dojo/when', 'dojo/Stateful', 'dijit/_WidgetBase', 'dijit/_TemplatedMixin', 'dijit/_WidgetsInTemplateMixin', 'dojox/mvc/_InlineTemplateMixin', 'dojox/mvc/at', 'dojox/mvc/getStateful', 'dojox/mvc/StatefulArray', 'dojox/mvc/StoreRefController', '../computed', '../store/TodoLocalStorage', 'dojox/mvc/Element', 'dojox/mvc/WidgetList', './CSSToggleWidget'ja './TodoEnter' vastavaid faile [351]. Dojo laadija poolt sõltuvuste loetelurivile järgneval moodulina salvestataval funktsioonil on argumendid array, lang, router, when, Stateful, _WidgetBase, _TemplatedMixin, _WidgetsInTemplateMixin, _InlineTemplateMixin, at, getStateful, StatefulArray, StoreRefController, computed ja TodoLocalStorage [351]. Funktsiooniga declare luuakse tagastatav klass, mille ülemklassideks on _WidgetBase, _TemplatedMixin, _WidgetsInTemplateMixin, _InlineTemplatedMixin ja StoreRefController [371]. Käesoleva klassi sees defineeritakse hulk atribuute, objekte ning funktsioone.

```
define([
  'dojo/_base/declare',
  //...//
  './TodoEnter'
], function (declare,
  array,
  //...//
  TodoLocalStorage) {
  //...//
  return declare([_WidgetBase, _TemplatedMixin, _WidgetsInTemplateMixin, _InlineTemplateMixin,
StoreRefController],
    /** @lends Todos# */ {
    store: null,
    newTodo: '',
    status: '',
    saving: false,
    filteredTodos: null,
    remainingCount: 0,
    completedCount: 0,
    areAllChecked: false,
//...//
```

Joonis 147. Todos.js algus [27]

Funktsiooni `getStatefulOptions` sees hallatakse funktsioonis `getStatefulStoreData` veebilehitseja andmehoiust pärit tegevuste objektide loetelurivi (joonis 148). Selleks kasutatakse Dojo funktsiooni `getStateful`, mis argumendiks saadud objekti teisendab manipuleerimist hõlbustava liidese `Stateful` kasutamist võimaldavale kujule [375] [382].

```
//...//
   getStatefulOptions: {
      getType: function () {
        return 'storeData';
      },

      getStatefulStoreData: function (data) {
        return getStateful({todos: data});
      }
    },
//...//
```

Joonis 148. Todos.js ning `getStatefulOptions` [27]

Objekt `emptyConverter` (joonis 149) on kasutusel failis index.html, kus see tagastab funktsiooniga `at` jälgitavate atribuutide numbriliste väärtuste põhjal funktsiooni `transform` sees tõeväärtuse [383]. Sellega määratakse HTML-atribuudi `hidden` rakendamine ehk otsus, kas seotud HTML-elemente peidetakse või kuvatakse [384]. Funktsiooni `transform` sees paiknev teisendamist juhtiv objekt peab sisaldama nii ühes kui ka teises suunas teisendamist võimaldavaid funktsioone `format` ja `parse` [383] [385]. Objektis `emptyConverter` funktsioon `format` tagastab argumendiks saadud arvu põhjal tõeväärtuse tüüpi tõeväärtuse. Seega funktsioon `parse` peaks tagastama vastupidise tulemuse ehk arvu, kuid seda pole testrakenduses vaja ning seetõttu genereeritakse väära tõeväärtusega viga [386].

```
//...//
   emptyConverter: {
      format: function (count) {
        return count === 0;
      },
      parse: function () {
        // No backward conversion
        throw false;
      }
    },
//...//
```

Joonis 149. Todos.js ning `emptyConverter` [27]

Sarnaselt eelmises lõigus käsitletud objektiga kasutatakse ka objekte `pluralizeConverter` ja `statusConverter` HTML-faili index.html elementide atribuutidega manipuleerimiseks (joonis 150). Esimene objekt määrab, kas kuvatakse elemendi atribuudi `one` väärtuseks olevat ingliskeelse teksti ainsusevormi või atribuudi `two` väärtuseks olevat teksti mitmusevormis. Teise objekti poolt tagastatav tõeväärtus määrab, millisele navigeerimislingile lisatakse CSS-klass `selected`.

```
//...//
   pluralizeConverter: {
      format: function (count) {
        return count > 1 ? this.target.other : this.target.one;
      },
      parse: function () {
        // No backward conversion
        throw false;
      }
    },

    statusConverter: {
      format: function (status) {
        return status === this.target.statusForElem;
      },
      parse: function () {
        // No backward conversion
        throw false;
      }
    },
//...//
```

Joonis 150. Todos.js ning `pluralizeConverter` ja `statusConverter` [27]

Klassile lisatud meetod `postMixinProperties` kutsutakse Dojo klassides välja enne nende esitamist (*rendering*) (joonis 151) [387]. Käesoleval juhul salvestatakse selles meetodis atribuuti `store` failis TodoLocalStorage.js defineeritud klassi isend. Seejärel kutsutakse välja meetod `queryStore`, mis käivitab veebilehitseja andmehoiu peal failist TodoLocalStorage.js pärineva funktsiooni `query`, millega ammutatakse seal salvestatud tegevuste objektide andmed [388]. Meetodiga `own` lisatakse klassile URL-haldaja `router` [373]. Meetodiga `register` määratakse ära kõik rakenduses kasutatavad URL-teed. Atribuudi `status` väärtuse määramisel kasutatav sündmuseatribuut `newPath` kujutab endast uut URL-tee osa, mille sisu kuulub järgmisena esitamisele – selle puudumisel kasutatakse tühja sõne, mis vastab kõikide tegevusobjektide loetelu esitamise URL-ile [389]. Muutuja `statusTable` kujutab endast objekti, mis sisaldab tõeväärtusega atribuute. Seda kasutatakse hiljem tehtuks märgitud tegevuste objektide filtreerimisel.

```
//...//
   postMixInProperties: function () {
      if (!this.store) {
        this.store = new TodoLocalStorage();
      }
      this.queryStore();

      // Set up router
      this.own(router.register(/.*/, lang.hitch(this, function (event) {
        var table = {
          '/active': 'active',
          '/completed': 'completed'
        };
        this.set('status', table[event.newPath] || '');
```

```
      }));
      router.startup();
     // Set up computed properties
      var statusTable = {
        active: false,
        completed: true
      };
//...//
```

Joonis 151. Todos.js ning `postMixinProperties` ja `statusTable` [27]

Käesolevale klassile Todos lisatakse meetodiga `own` (joonis 152) töötlejana (*handler*) funktsioon `computed`, mis on defineeritud failis computed.js ning mille tööpõhimõtet selgitavad selles failis olevad joonistelt välja jäetud kommentaarid [373] [374]. Tööleja tagastab kõiki tegevuse objekte sisaldava loetelurivi, kui muutuja `status` väärtuseks on tühi sõne. Vastasel juhul filtreeritakse välja ning tagastatakse vastavalt olukorrale tehtuks või mittetehtuks märgitud tegevusobjektid. Funktsiooni `computed` viimased argumendid sisaldavad Dojo funktsioone `mixin` ja `at` – vastavalt failis computed.js olevatele kommentaaridele pannakse nii paika, et töötleja sees kasutatakse iga olemasolevat tegevuse objekti ja atribuute `status` ning `completed` [374] [390] [391]. Järgmine klassile lisatud töötleja põhineb samuti funktsioonil `computed`. Töötlejafunktsiooni sees kasutatakse iga tegevuse objekti atribuuti `completed` ning tagastatakse mittetehtuks märgitud tegevuste koguarv muutuja `remainingCount` kaudu [374].

```
//...//
     this.own(computed(this, 'filteredTodos', lang.hitch(this, function (completed, status) {
        var todos = this.get('todos');
        if (!status) {
          return todos;
        } else {
          var filteredTodos = [];
          for (var i = 0; i < completed.length; i++) {
            if (completed[i] === statusTable[status]) {
              filteredTodos.push(todos[i]);
            }
          }
          return filteredTodos;
        }
      }), lang.mixin(at(this.get('todos'), 'completed'), {each: true}), at(this, 'status')));

      this.own(computed(this, 'remainingCount', function (completed) {
        var count = 0;
        for (var i = 0; i < completed.length; i++) {
          count += +!completed[i];
        }
        return count;
      }, lang.mixin(at(this.get('todos'), 'completed'), {each: true})));
//...//
```

Joonis 152. Todos.js ning `filteredTodos` ja `remainingCount` [27]

Klassi Todos viimased kaks funktsioonil `computed` põhinevat töötlejat salvestavad atribuutidesse `competedCount` ning `areAllChecked` vastavalt kõigi tehtuks märgitud tegevusobjektide arvu ning tõeväärtuse sellest, kas kõik tegevusobjektid on märgitud tehtuks või mitte (joonis 153). Funktsiooni `postMixInProperties` lõpus kutsutakse käsuga `this.inherited(arguments)` välja ülemklassi `_WidgetBase` samanimelist funktsiooni [369] [371].

```
//.../
     this.own(computed(this, 'completedCount', function (completed) {
        var count = 0;
        for (var i = 0; i < completed.length; i++) {
          count += +completed[i];
        }
        return count;
      }, lang.mixin(at(this.get('todos'), 'completed'), {each: true})));

      this.own(computed(this, 'areAllChecked', function (remainingCount) {
        return remainingCount === 0;
      }, at(this, 'remainingCount')));

      this.inherited(arguments);
    },
//.../
```

Joonis 153. Todos.js ning `completedCount` ja `areAllChecked` [27]

Funktsiooni `addTodo` (joonis 154) ülesanne on salvestada kasutaja poolt failis index.html oleva pealkirja vormi täitmise ja sündmuse `submit` algatamise järel uus tegevuse objekt soovitud pealkirjaga veebilehitseja andmehoidu. Selleks kontrollitakse, kas ebavajalikest tühikutest vaba pealkiri on vähemalt ühe sõnemärgi pikkune, luuakse objekt pealkirja ning atribuudiga `completed` ning sooritatakse andmehoidu sisestus. Muutujasse `ret` salvestatakse tagasikutse halduris `when` toimuva tegevuse õnnestumise tulemus [377]. Kui funktsioon `addStore` saab lisada probleemideta uue objekti veebilehitseja andmehoidu, kutsudes välja failis TodoLocalStorage.js asuvat abifunktsiooni `add`, siis uuendatakse vastavalt ka atribuuti `todos`, algväärtustatakse tühja sõnega atribuut `newTodo` ja omistatakse atribuudile `saving` väär tõeväärtus [381] [388]. Probleemide korral aga omistatakse atribuudile `saving` väär tõeväärtus ning genereeritakse väära tõeväärtusega viga [381] [386]. JavaScripti meetodiga `preventDefault` ei lubata käesolevas funktsioonis sündmuse `submit` korral sellele sündmusele vastaval vaikimisi käivitataval sündmusteahelal alata [380]. Lõpuks tagastatakse muutuja `ret`.

```
//.../
   addTodo: function (event) {
      var ret;
      var title = lang.trim(this.get('newTodo'));
      if (title.length > 0) {
        this.set('saving', true);
        var data = {
          title: title,
          completed: false
        };
        ret = when(this.addStore(data), lang.hitch(this, function () {
          this.get('todos').push(new Stateful(data));
          this.set('newTodo', '');
          this.set('saving', false);
        }), lang.hitch(this, function (e) {
          this.set('saving', false);
          throw e;
        }));
      }
      event.preventDefault();
      return ret;
    },
//.../
```

Joonis 154. Todos.js ning `addTodo` [27]

Funktsiooni `saveTodo` kasutatakse olemasoleva tegevuse objekti muudatuste salvestamiseks veebilehitseja andmehoidu (joonis 155). Failis Todo.js kasutasid seda funktsioonid `saveEdits` ning `toggleCompleted`. Funktsiooni `saveTodo` sees seatakse tõene tõeväärtus atribuudi `saving` väärtuseks. Seejärel luuakse argumendiks saadud tegevuse objektist `todo` madal (*shallow*) koopia muutujasse `data` konstruktsiooniga

`lang.mixin({}, todo)` ning kustutatakse sellelt koopialt JavaScripti operaatoriga `delete` atribuut `isEditing` [392]. Kui funktsioon `putStore` saab salvestada muudetava objekti veebilehitseja andmehoidu probleemideta, kutsudes välja failis TodoLocalStorage.js asuvat abifunktsiooni `put`, siis omistatakse atribuudile `saving` väär tõeväärtus [381] [388]. Probleemide korral aga omistatakse atribuudile `saving` samuti väär tõeväärtus, lisaks taastatakse tegevuse objekti muutmisele eelnenud atribuutide väärtused abimuutujatest `originalTitle` ja `originalCompleted` ning genereeritakse tekkinud viga [381] [386].

```
//...//
   saveTodo: function (todo, originalTitle, originalCompleted) {
      this.set('saving', true);
      var data = lang.mixin({}, todo);
      delete data.isEditing;
      return when(this.putStore(data), lang.hitch(this, function () {
        this.set('saving', false);
      }), lang.hitch(this, function (e) {
        this.set('saving', false);
        todo.set('title', originalTitle);
        todo.set('completed', originalCompleted);
        throw e;
      }));
    },
//...//
```

Joonis 155. Todos.js ning `saveTodo` [27]

Funktsiooni `removeTodo` kasutatakse olemasoleva tegevuse objekti kustutamiseks veebilehitseja andmehoiust (joonis 156). Failis Todo.js kasutasid seda funktsioonid `saveEdits` ning `removeTodo`. Funktsiooni `saveTodo` sees seatakse tõene tõeväärtus atribuudi `saving` väärtuseks. Kui funktsioon `removeStore` saab kustutada kustutatava objekti probleemideta veebilehitseja andmehoiust, kutsudes välja failis TodoLocalStorage.js asuvat abifunktsiooni `remove`, siis omistatakse atribuudile `saving` väär tõeväärtus ja eemaldatakse veebilehitseja andmehoiust eemaldatu ka atribuudist `todos` JavaScripti loetelurivi manipuleeriva meetodiga `splice` [107] [381] [388]. Probleemide korral aga omistatakse atribuudile `saving` samuti väär tõeväärtus, lisaks genereeritakse tekkinud viga võtmesõnaga `throw` [381] [386]. Funkstioon `replaceTodo` eemaldab esimeseks argumendiks saadud tegevuse objekti atribuudist `todos` ning lisab sinna samasse asukohta, kus asus kustutatu, teise argumendina saadud tegevuse objekti `newTodo` meetodiga `splice` [107].

```
//...//
    removeTodo: function (todo) {
      this.set('saving', true);
      return when(this.removeStore(this.store.getIdentity(todo)), lang.hitch(this, function () {
        this.set('saving', false);
        this.get('todos').splice(array.indexOf(this.get('todos'), todo), 1);
      }), lang.hitch(this, function (e) {
        this.set('saving', false);
        throw e;
      }));
    },

    replaceTodo: function (oldTodo, newTodo) {
      var index = this.get('todos').indexOf(oldTodo);
      if (index >= 0) {
        this.get('todos').splice(index, 1, newTodo);
      }
    },
//...//
```

Joonis 156. Todos.js ning `removeTodo` ja `replaceTodo` [27]

Faili Todos.js lõpus on defineeritud funktsioonid `markAll` ja `clearCompletedTodos` (joonis 157). Mõlemad kasutavad loetelurivi tüüpi atribuudi läbimiseks Dojo tsüklikonstruktsiooni `forEach` [393]. Esimeses läbitakse loetelurivi tüüpi atribuuti `todos` nii, et märgitakse kõik selle elemendid tehtuks või mittetehtuks sõltuvalt atribuudi `areAllChecked` tõeväärtusest, millest saab teada, kas kõik tegevusobjektid on juba märgitud. Teises funktsioon läbitakse loetelurivist `todos` JavaScripti meetodiga `slice` loodud madalat (*shallow*) koopiat ning eemaldatakse funktsiooniga `removeTodo` need tegevusobjektid, mis on märgitud tehtuks ehk mille atribuudi `completed` väärtus on tõene [106].

```
//...//
   markAll: function () {
      var current = this.get('areAllChecked');
      array.forEach(this.get('todos'), function (todo) {
        var old = todo.get('completed');
        if (old !== current) {
          todo.set('completed', current);
          this.saveTodo(todo, todo.get('title'), old);
        }
      }, this);
    },

    clearCompletedTodos: function () {
      array.forEach(this.get('todos').slice(), function (todo) {
        if (todo.get('completed')) {
          this.removeTodo(todo);
        }
      }, this);
    }
  });
});
```

Joonis 157. Todos.js ning `markAll` ja `clearCompletedTodos` [27]

Faili index.html kujutaval joonisel 158 on osa koodist välja jäetud. Kehas ehk `<body>`-siltide alas olevas esimese `<script>`-sildi alas määratakse Dojo funktsiooni `at` kasutamise võimalus [394]. Teises `<script>`-sildi alas identifikaatoriga `item-template` on defineeritud HTML-mall [395]. Identifikaatorit `todoapp` omava HTML-elemendi alas on defineeritud veel üks HTML-mall. Atribuut `data-dojo-type="todo/widgets/Todos"` lisamine HTML-elemendile identifikaatoriga `todoapp` tähendab, et selles elemendis käivitub rakenduse funktsionaalne osa, kui Dojo loob sinna isendi vidinklassist, mis on defineeritud failis Todos.js [396]. Jaluses identifikaatoriga `footer` paikneb info testrakenduse autorite kohta.

```
//...//
  <body>
    <script type="dojo/require">
      at: 'dojox/mvc/at'
    </script>
    <script id="item-template" type="dojox/mvc/InlineTemplate">
        //...//
    </script>
    <section id="todoapp" data-dojo-type="todo/widgets/Todos">
      <script type="dojox/mvc/InlineTemplate">
        //...//
      </script>
    </section>
```

```
    <footer id="info">
      //...//
    </footer>
    <script src="node_modules/todomvc-common/base.js"></script>
    <script src="js/main.js"></script>
    <script src="js/lib/dojo/dojo.js"></script>
    <!-- Use below instead of above line to use non-built version of Dojo components. -->
    <!-- <script src="node_modules/dojo/dojo.js"></script> -->
  </body>
//...//
```

Joonis 158. index.html ning ülevaade [27]

Joonisel 159 on kujutatud faili index.html HTML-elemendis identifikaatoriga `item-template` loeteluelemendisiltide vahel paiknev HTML-malli kood [395]. Tegevuse tehtuks märkimise märgendkast on atribuudiga `data-dojo-type` märgitud Dojo klassi `Element` isendiks [397]. Sündmus `change` on atribuudi `dojo-data-attach-event` kaudu seotud funktsiooni `toggleCompeted` käivitamisega [398]. Atribuudiga `data-dojo-props` ning selle kaudu viite tekitava funktsiooniga `at` seotakse tekstikast atribuudiga `completed` [391] [399]. Ka tegevuse objekti pealkirja kuvava sildi `<label>` ala on Dojo klassi `Element` isend [397]. Sellel toimuv topeltklõpsamise sündmus on seotud redigeerimisfunktsiooniga `editTodo` ja tekstikuvamise atribuut `innerText` on seotud atribuudiga `title` [391] [398]. Tegevuse objekti kustutamiseks mõeldud nupule klõpsates kutsutakse välja funktsioon `removeTodo`. Sildi `<form>` alas paiknev tegevuse objekti redigeerimise vorm kasutab samuti eelnevates lausetes toodud võtteid. Lisaks kasutab see atribuuti `data-dojo-mixins`, millega viitab HTML-elemendis kasutatavate funktsioonide definitsioone sisaldavate vidinklassiressursside TodoEscape.js ja TodoFocus.js asukohtadele.

```
//...//
     <li>
        <div class="view">
          <input class="toggle" type="checkbox" data-dojo-type="dojox/mvc/Element" data-dojo-
props="checked: at('rel:', 'completed')" data-dojo-attach-event="change: toggleCompleted">
          <label data-dojo-type="dojox/mvc/Element" data-dojo-props="innerText: at('rel:', 'tit-
le')" data-dojo-attach-event="dblclick: editTodo"></label>
          <button class="destroy" data-dojo-attach-event="click: removeTodo"></button>
        </div>
        <form data-dojo-attach-event="submit: saveEdits">
          <input class="edit" data-dojo-type="dojox/mvc/Element" data-dojo-mixins="todo/wid-
gets/TodoEscape,todo/widgets/TodoFocus" data-dojo-props="_setDisabledAttr: 'domNode', value:
at('rel:', 'title'), shouldGetFocus: at('rel:', 'isEditing'), disabled: at('rel:todosCtrl', 'sa-
ving')" data-dojo-attach-event="blur: invokeSaveEdits, escape: revertEdits">
        </form>
      </li>
//...//
```

Joonis 159. index.html ning `item-template` [27]

Failis index.html asuv teine mall sisaldab uue tegevuse objekti pealkirja sisestamise vormi identifikaatoriga `todo-form`, tegevusobjektide kuvamise ja redigeerimise sektsiooni identifikaatoriga `main` ning jalust identifikaatoriga `footer` (joonis 160). Kaks viimasena mainitud ala on kasutaja eest peidetud, kui tegevusobjektide nimekirjas objekte pole [384]. Nende sisu on kujutatud järgnevate tekstilõikudega seotud joonistel.

```
//...//
       <section>
          <header id="header">
            <h1>todos</h1>
            <form id="todo-form" data-dojo-attach-event="submit: addTodo">
              <input id="new-todo" placeholder="What needs to be done?" data-dojo-type="do-
jox/mvc/Element" data-dojo-mixins="todo/widgets/TodoEnter" data-dojo-props="_setDisabledAttr:
'domNode', value: at(this, 'newTodo'), disabled: at(this, 'saving')" autofocus>
            </form>
```

```
          </header>
          <section id="main" data-dojo-type="dijit/_WidgetBase" data-dojo-props="_setHiddenAttr:
'', hidden: at(this.get('todos'), 'length').transform(this.emptyConverter)">
            //...//
          </section>
          <footer id="footer" data-dojo-type="dijit/_WidgetBase" data-dojo-props="_setHiddenAttr:
'', hidden: at(this.get('todos'), 'length').transform(this.emptyConverter)">
            //...//
          </footer>
        </section>
//...//
```

Joonis 160. index.html ning teine mall [27]

Sektsioonis identifikaatoriga `main` eksisteerib märgendkast kõigi tegevuste tehtuks märkimiseks ning `dojox/mvc/WidgetList` tüüpi loetelu piirkond, et esitada loetelurivi `Todo` tüüpi tegevusobjektidest `filteredTodos` (joonis 161) [400]. Atribuut `templateString` määrab iga loeteluelemendi esitamiseks eelnevalt defineeritud malli `item-template` kasutamise. Tõese väärtusega atribuut `partialRebuild` tagab mingi tegevuse objekti muutuse korral ainult muutunud objektide uuendamise loetelus [400]. Kasutades ressursi `dijit/_WidgetBase` võimalusi, on failis CSSToggleWidget.js defineeritu põhjal faili index.html atribuutides `data-dojo-props` kasutatud CSS-klasside tingimuslikku elementidele lisamist kirjapildiga `_set…Attr`, milles kolme punkti asemel on eesmärki iseloomustav sõna [370] [401].

```
//...//
          <input id="toggle-all" type="checkbox" data-dojo-type="dojox/mvc/Element" data-dojo-
props="checked: at(this, 'areAllChecked')" data-dojo-attach-event="change: markAll">
          <label for="toggle-all">Mark all as complete</label>
          <ul id="todo-list"
            data-dojo-type="dojox/mvc/WidgetList"
            data-dojo-props="todosWidget: this, children: at(this, 'filteredTodos'), partial-
Rebuild: true, templateString: require('dojo/dom').byId('item-template').innerHTML"
            data-mvc-child-type="todo/widgets/Todo"
            data-mvc-child-mixins="todo/widgets/CSSToggleWidget"
            data-mvc-child-props="todosWidget: this.parent.todosWidget, _setCompletedAttr:
{type: 'classExists'}, _setIsEditingAttr: {type: 'classExists', className: 'editing'}, completed:
at(this.target, 'completed'), isEditing: at(this.target, 'isEditing')">
//...//
```

Joonis 161. index.html ning HTML-elemendi `id="main"` kood [27]

Sektsioonis identifikaatoriga `footer` kuvatakse kasutajale mittetehtud tegevusobjektide arvu koos korrektse mitmust või ainsust arvestava ingliskeelse tekstiga numbri kõrval, kolme tegevuste filtreerimise linki ning nuppu kõigi tehtuks märgitud tegevuste kustutamiseks (joonis 162). Filtreerimislinkidest on jooniselt välja jäetud tehtud ja mittetehtud tegevuste loeteluelemendid: nende URL-viidad on vastavalt `href="#/active"` ja `href="#/completed"`. Mainitud kustutamisnupp on nähtaval vaid tehtuks märgitud tegevuste olemasolu korral [384].

```
//...//
          <span id="todo-count"><strong data-dojo-type="dojox/mvc/Element" data-dojo-props="in-
nerText: at(this, 'remainingCount')"></strong>
            <span data-dojo-type="dojox/mvc/Element" data-dojo-props="innerText: at(this, 're-
mainingCount').transform(this.pluralizeConverter), one: 'item left', other: 'items left'"></span>
          </span>
          <ul id="filters">
            <li>
              <a data-dojo-type="todo/widgets/CSSToggleWidget" data-dojo-props="_setSe-
lectedAttr: {type: 'classExists'}, selected: at(this, 'status').transform(this.statusConverter),
statusForElem: ''" href="#/">All</a>
            </li>
            //...//
          </ul>
```

```
            <button id="clear-completed" data-dojo-type="dijit/_WidgetBase" data-dojo-attach-
event="click: clearCompletedTodos" data-dojo-props="_setHiddenAttr: '', hidden: at(this, 'comple-
tedCount').transform(this.emptyConverter)">Clear completed</button>
//...//
```

Joonis 162. index.html ning HTML-elemendi `id="footer"` kood [27]

Failis CSSToggleWidget.js on defineeritud funktsiooni `define` sees moodul (joonis 934), mis kasutab sõltuvusena viidetele `"dojo/_base/array"`, `"dojo/_base/declare"`, `"dojo/_base/lang"`, `"dojo/dom-class"` ja `"dijit/_WidgetBase"` vastavaid faile [351]. Funktsiooniga `declare` luuakse tagastatav klass, mille ülemklassiks on `_WidgetBase` [367]. Mainitud klassi sees on toodud kommentaarid. Nende põhjal saab aru, et CSSToggleWidget.js on loodud selleks, et atribuutide tõeväärtuste põhjal CSS-klasse HTML-elementidele lisada. Näiteks eelnevalt käsitletud HTML-koodis tegevusobjektide filtreerimisviitade atribuutide juures (joonis 162) paiknev väljadega `_setSelectedAttr` ja `selected` seotud kood tagab CSS-klassi `selected` lisamist või eemaldamist nendele elementidele. Funktsioonis `_attrToDom` kutsutakse käsuga `this.inherited` välja ülemklassi `_WidgetBase` samanimelist funktsiooni [369]. Selle funktsiooni sees sooritatakse tingimuskontrolle ning hoolitsetakse vajalike argumentide leidmise eest loetelurivist. DOM-mudeli elemendile klassi lisamiseks või eemaldamiseks kasutatakse lõpuks meetodit `domClass.toggle` [402].

```
define([
  "dojo/_base/array",
  //...//
  "dijit/_WidgetBase"
], function(array, declare, lang, domClass, _WidgetBase){
  return declare(_WidgetBase, {
   // summary:
    //         Widget supporting widget attributes with classExists type.
    //         classExists type allows boolean value of an attribute to reflect existence of a CSS
class in a DOM node in the widget.
     //...//
    _attrToDom: function(/*String*/ attr, /*String*/ value, /*Object?*/ commands){
      // summary:
      //    Handle widget attribute with classExists type.
      //    See dijit/_WidgetBase._attrToDom() for more details.

      var callee = arguments.callee;
      array.forEach((function(){ return lang.isArray(commands) ? commands.slice(0) :
[commands]; })(arguments.length >= 3 ? commands : this.attributeMap[attr]), function(command){
        command.type != "classExists" ?
         this.inherited("_attrToDom", lang.mixin([attr, value, command], {callee: callee})) :
         domClass.toggle(this[command.node || "domNode"], command.className || attr, value);
      }, this);
    }
  });
});
```

Joonis 163. CSSToggleWidget.js [27]

Järgneva faili TodoLocalStorage.js algusesse on testrakenduse autorid jätnud kommentaari, et see fail eemaldataks, kui Dojole millalgi peaks loodama vastav sisseehitatud funktsionaalsus veebilehitseja andmehoiuga taolisel viisil suhtlemiseks (joonis 164). Käesolevas failis on defineeritud funktsiooni `define` sees moodul, mis kasutab sõltuvusena viidetele `'dojo/_base/declare'`, `'dojo/_base/lang'`, `'dojo/has'`,`'dojo/json'`, `'dojo/store/util/QueryResults'` ja `'dojo/store/util/SimpleQueryEngine'` vastavaid faile [351]. Dojo `has.add` võimaldab kontrollida, kas JavaScripti funktsioon `findIndex` on kasutatav [403] [404]. Näiteks veebilehitsejas Internet Explorer seda funktsiooni ei saa kasutada [404]. Järgnevas sama nimega funktsioonis kontrollitakse tehtud kontrolli tulemuse tõeväärtust väljakutsega `has('array-findindex-api')`. Positiivse tulemuse korral rakendatakse tegevusobjektide loetelurivi sisaldaval argumendil `a` JavaScripti funktsiooni `findIndex` koos

tagasikutse ning objekti argumentidega. Negatiivse tulemuse korral leitakse argumendiks saadud objekti loetelurivis paiknemise indeks tsükliga.

```
// TODO: Remove this file once Dojo provides this feature out-of-the-box.
define([
  'dojo/_base/declare',
   //...//
  'dojo/store/util/SimpleQueryEngine'
], function (declare, lang, has, json, QueryResults, SimpleQueryEngine) {
  'use strict';

  has.add('array-findindex-api', typeof Array.prototype.findIndex === 'function');

  function findIndex(a, callback, thisObject) {
    if (has('array-findindex-api')) {
      return a.findIndex(callback, thisObject);
    } else {
      for (var i = 0; i < a.length; i++) {
        if (callback.call(thisObject, a[i])) {
          return i;
        }
      }
      return -1;
    }
  }
//...//
```

Joonis 164. TodoLocalStorage.js ning `findIndex` [27]

Funktsiooniga `declare` luuakse tagastatav klass, millel puudub ülemklass (joonis 165) [405]. Päringute tegemiseks määratakse atribuudi `queryEngine` väärtuseks Dojo tööriist `SimpleQueryEngine` [406]. Funktsioonis `_saveStorage` viiakse andmed enne veebilehitseja andmehoidu salvestamist JSON-sõne kujule meetodiga `JSON.stringify` ja taastatakse nende originaalne kuju meetodiga `JSON.parse` veebilehitseja andmehoiust ammutamise funktsioonis `_loadStorage` [113] [114]. Rakenduse käivitamisel välja kutsutavas funktsioonis `postscript` kasutatakse argumendi `props` ning võtmesõna `this` poolt viidatava objekti sidumiseks Dojo funktsiooni `mixin` [390] [407].

```
//...//
return declare(null, /** @lends TodoLocalStorage# */ {
    storageId: 'todos-dojo',
    idProperty: 'id',
    queryEngine: SimpleQueryEngine,

    _loadStorage: function () {
      return json.parse(localStorage.getItem(this.storageId) || '[]');
    },
    _saveStorage: function (data) {
      localStorage.setItem(this.storageId, json.stringify(data));
    },
    postscript: function (props) {
      lang.mixin(this, props);
    },
//...//
```

Joonis 165. TodoLocalStorage.js ning `_loadStorage`, `_saveStorage` ja `postscript` [27]

Selleks, et argumendi `id` põhjal veebilehitseja andmehoiust tagastada vastav tegevuse objekt, leitakse funktsioonis `get` selle asukoht (joonis 166). Objekti puudumisest saadakse aru, kui asukohaks määrati arv -1. Andmehoiuga suhtlemisel on vaja leida tegevusobjektide identifikaatoreid. Ülesande täidab argumentobjektil JavaScripti atribuudi ligipääsusüntaksit kasutav funktsioon `getIdentity` [408].

```
//.../
   get: function (id) {
      var data = this._loadStorage();
      var index = findIndex(data, function (item) {
        return this.getIdentity(item) === id;
      }, this);
      if (index >= 0) {
        return data[index];
      }
    },

    getIdentity: function (object) {
      return object[this.idProperty];
    },
//.../
```

Joonis 166. TodoLocalStorage.js ning `get` ja `getIdentity` [27]

Uute tegevusobjektide lisamisel või vanade uuendamisel rakendatakse funktsiooni `put` (joonis 167). Kasutatav identifikaator saadakse argumendist `options`, funktsiooni `getIdentity` tööst olemasoleva tegevuse objekti korral või uute objektide korral JavaScripti funktsioonist `Math.random`, mis tagastab ujukomaarvu vahemikus [0, 1) [223]. Uued objektid listakse andmehoiu andmete loetelurivi lõppu, vanad jäetakse vanale asukohale.

```
//.../
   put: function (object, options) {
      var id = options && options.id || this.getIdentity(object) || Math.random();
      var data = this._loadStorage();
      var index = findIndex(data, function (item) {
        return this.getIdentity(item) === id;
      }, this);
      data[index >= 0 ? index : data.length] = object;
      object[this.idProperty] = id;
      this._saveStorage(data);
      return id;
    },
//.../
```

Joonis 167. TodoLocalStorage.js ning `put` [27]

Funktsioon `add` lisab funktsiooniga `put` saadud objekti veebilehitseja andmehoidu, kui seesama objekt seal juba ei eksisteeri (joonis 168). Vastasel juhul genereeritakse veateade võtmesõnaga `throw` ning JavaScripti veateateobjektiga `Error` [386] [409]. Funktsioon `remove` eemaldab andmehoiu põhjal loodud loetelurivist `data` argumendiks saadud identifikaatorile vastava objekti JavaScripti meetodiga `splice`, kui see objekt eksisteerib käesolevas loetelurivis [107]. Muudatuste salvestamiseks vaabilehitseja andmehoidu kutsutakse välja funktsioon `_saveStorage`.

```
//.../
    add: function (object, options) {
      if (this.get(this.getIdentity(object))) {
        throw new Error('Object already exists.');
      }
      return this.put(object, options);
    },

    remove: function (id) {
      var data = this._loadStorage();
      var index = findIndex(data, function (item) {
        return this.getIdentity(item) === id;
      }, this);
      if (index < 0) {
        throw new Error('Object not found.');
      }
      data.splice(index, 1);
      this._saveStorage(data);
    },
//.../
```

Joonis 168. TodoLocalStorage.js ning `add` ja `remove` [27]

Failis Todos.js kutsutakse välja meetod `queryStore`, mis käivitab veebilehitseja andmehoiu peal failist TodoLocalStorage.js pärineva funktsiooni `query` (joonis 169), millega ammutatakse seal salvestatud tegevuste objektide andmed [388]. Funktsioon Query tagastab tulemused mähise (*wrapper*) `QueryResults` sees, mis tagab, et tulemusi saab kohelda kui loetelurivi [410]. `SimpleQueryEngine` tüüpi `queryEngine` teeb päringu veebilehitseja andmehoiust saadavate tulemuste põhjal [411].

```
//...//
   query: function (query, options) {
      /*jshint newcap:false*/
      return QueryResults(this.queryEngine(query, options)(this._loadStorage()));
    }
  });
});
```

Joonis 169. TodoLocalStorage.js ning `query` [27]

Faili computed.js algusesse on testrakenduse autorid jätnud kommentaari, et see fail eemaldataks, kui Dojole millalgi peaks loodama vastav sisseehitatud funktsionaalsus liidest `dojo/Stateful` kasutavate atribuutidega manipuleerimiseks (joonis 170) [375]. Käesolevas failis on defineeritud funktsiooni `define` sees moodul, mis kasutab sõltuvusena viidetele `'dojo/_base/array'` ja `'dojo/has'` vastavaid faile [351]. Dojo `has.add` võimaldab kontrollida, kas JavaScripti funktsioon `Object.is` on kasutatav [403] [412]. Näiteks veebilehitsejas Internet Explorer seda funktsiooni ei saa kasutada [412]. Muutujale `areSameValues` omistatakse funktsioon `Object.is`, kui väljakutsega `has(object-is-api')` saadakse tõene tõeväärtus [403]. Vastasel juhul defineeritakse samaks otstarbeks kahe argumendi samasuse võrdlusega tegelev funktsioon muutuja `areSameValues` väärtuseks [413]. Muutujale `arrayProto` omistatakse aga JavaScripti loetelurivi tüüpi loetelurivi (*Array*) konstruktori prototüüp [414].

```
// TODO: Remove this file once Dojo provides this feature out-of-the-box.
define([
  'dojo/_base/array',
  'dojo/has'
], function (array, has) {
  'use strict';

  var arrayProto = Array.prototype;

  has.add('object-is-api', typeof Object.is === 'function');
  var areSameValues = has('object-is-api') ? Object.is : function (lhs, rhs) {
    return lhs === rhs && (lhs !== 0 || 1 / lhs === 1 / rhs) || lhs !== lhs && rhs !== rhs;
  };
//...//
```

Joonis 170. computed.js algus [27]

Funktsioonis `watch` kontrollitakse, kas argumendiks saadud objekt `o` puhul saab `dojo/Stateful`-meetodit `watch` kasutada (joonis 171) [415]. Kui saab, siis lisatakse objekti `o` atribuudile `prop` meetodiga `watch` tagasikutse (*callback*) funktsioon, mida kutsutakse välja, kui atribuudi `prop` väärtus muutub [415]. Muutujasse `hWatch` salvestub seega töötleja (*handler*) [415]. Lõpuks tagastatakse funktsioon `remove`, millega saab eemaldada eelnevalt seotud objekti atribuudi muutuste jälgija.

```
//.../
function watch(o, prop, callback) {
    var hWatch;

    if (o && typeof o.watch === 'function') {
      hWatch = o.watch(prop, function (name, old, current) {
        if (!areSameValues(old, current)) {
          callback(current, old);
        }
      });
    } else {
      console.log('Attempt to observe non-stateful ' + o + ' with ' + prop + '. Observation not
happening.');
    }

    return {
      remove: function () {
        if (hWatch) {
          hWatch.remove();
        }
      }
    };
  }
//.../
```

Joonis 171. computed.js ning `watch` [27]

Funktsioonis `getProps` jõutakse argumendiks saadud loetelurivi tüüpi argumendi iga elemendini ressursi `dojo/_base/array` meetodiga `map`, mis rakendab oma teises argumendis paiknevat funktsiooni igale loetelurivi elemendile ning loob saadud tulemustest uue loetelurivi (joonis 172) [416]. Loodava loetelurivi sees võidakse rakendada sõltuvalt sealt sooritatavast tingimuskontrollist veel üks kord meetodit `map`. Lisaks kasutatakse JavaScripti atribuudi ligipääsusüntaksit ning sooritatakse tingimuskontrolle [408]. Funkstiooni `removeHandles` sees eemaldatakse argumendiks saadud loetelurivi algusest elemente, kuni loetelurivi tühjaks saab ning rakendatakse igale loetelurivi elemendile meetodit `remove`, mille eesmärk lähtuvalt funktsiooni nimest on töötlejate (*handlers*) eemaldamine.

```
//.../
  function getProps(list) {
    return array.map(list, function (p) {
      return p.each ?
        array.map(p.target, function (entry) {
          return entry.get ? entry.get(p.targetProp) : entry[p.targetProp];
        }) :
        p.target.get ? p.target.get(p.targetProp) :
          p.target[p.targetProp];
    });
  }

  function removeHandles(handles) {
    var h = null;
    while ((h = handles.shift())) {
      h.remove();
    }
  }
//.../
```

Joonis 172. computed.js ning `getProps` [27]

Failist computed.js tagastatakse rakenduse teistes failides kasutatav funktsioon `computed` (joonis 173), mille üldpõhimõttest on kirjutatud näiteks joonise 152 kohta käivas lõigus. Selle funktsiooni sees defineeritud funktsioonis `applyComputed` kutsutakse argumendina `compute` saadud funktsioon välja JavaScripti meetodiga `apply` objektil `target` loetelurivilaadse argumendiga `data` [417]. Juhul, kui eelmainitud tegevus õnnestub omistatakse funktsiooni rakendamise tulemus objekti `target` atribuudi `targetProp` väärtuseks.

```
//...//
return function (target, targetProp, compute) {
    function applyComputed(data) {
      var result;
      var hasResult;

      try {
        result = compute.apply(target, data);
        hasResult = true;
      } catch (e) {
        console.error('Error during computed property callback: ' + (e && e.stack || e));
      }

      if (hasResult) {
        if (typeof target.set === 'function') {
          target.set(targetProp, result);
        } else {
          target[targetProp] = result;
        }
      }
    }
//...//
```

Joonis 173. computed.js ning `applyComputed` [27]

Funkstiooni `computed` sees kontrollitakse, et objekt `target` ja atribuut `targetProp` vastaks kindlatele nõutele (joonis 174). Muutujasse `deps` salvestatakse JavaScripti funktsiooniga `slice` loetelurivi alates neljandast loetelurivi `arguments` elemendist [106]. Ressursi `dojo/_base/array` meetodiga `map` rakendab iga loetelurivi `deps` elemendile meetodi `map` teises argumendis defineeritud funktsiooni [417]. Viimase sees paikneb omakorda funktsioon `observeEntry`, mis lisab argumendiks saadud objekti atribuutidele muutuste jälgijaid.

```
//...//
    if (target === null || target === undefined) {
      throw new Error('Computed property cannot be applied to null.');
    }
    if (targetProp === '*') {
      throw new Error('Wildcard property cannot be used for computed properties.');
    }

    var deps = arrayProto.slice.call(arguments, 3);
    var hDep = array.map(deps, function (dep, index) {
      function observeEntry(entry) {
        return watch(entry, dep.targetProp, function () {
          applyComputed(getProps(deps));
        });
      }
//...//
```

Joonis 174. computed.js ning `deps` ja `hDep` [27]

Eelnevas lõigus ressursi `dojo/_base/array` meetodi `map` teises argumendis defineeritud funktsiooni kood jätkub joonisel 175. Seal lisatakse atribuutidele muutuste jälgijaid ning tagastatakse funktsioon, millega saab eemaldada eelnevalt seatud atribuudi muutuste jälgijaid.

```
//...//
     if (dep.targetProp === '*') {
        throw new Error('Wildcard property cannot be used for computed properties.');
      } else if (dep.each) {
        var hArray;
        var hEntry = array.map(dep.target, observeEntry);

        if (dep.target && typeof dep.target.watchElements === 'function') {
          hArray = dep.target.watchElements(function (idx, removals, adds) {
            removeHandles(arrayProto.splice.apply(hEntry, [idx, removals.length].concat(ar-
ray.map(adds, observeEntry))));
            applyComputed(getProps(deps));
          });
        } else {
          console.log('Attempt to observe non-stateful-array ' + dep.target + '. Observation not
happening.');
        }

        return {
          remove: function () {
            if (hArray) {
              hArray.remove();
            }
            removeHandles(hEntry);
          }
        };
      } else {
        return watch(dep.target, dep.targetProp, function (current) {
          var list = [];
          arrayProto.push.apply(list, getProps(deps.slice(0, index)));
          list.push(current);
          arrayProto.push.apply(list, getProps(deps.slice(index + 1)));
          applyComputed(list);
        });
      }
    });
//...//
```

Joonis 175. computed.js ning eelneva joonise funktsiooni jätk [27]

Funktsiooni `computed` koodi lõpus kutsutakse välja objekti atribuutide väärtustamise funktsioon `applyComputed`. Lisaks tagastatakse funktsioon, millega saab eemaldada eelnevalt seatud muutuste jälgijaid.

```
//...//
    applyComputed(getProps(deps));
    return {
      remove: function () {
        removeHandles(hDep);
      }
    };
  };
});
```

Joonis 176. computed.js ning faili lõpp [27]

Autor annab Dojole õppimise kategoorias null punkti, sest testrakenduse mõistmisele ning kirjalikul kujul selgitamisele kulus ligikaudu 33 tundi.

2.6.9 TodoMVC testrakendusel põhinev jõudlus

Lokaalses masinas käivitati testrakendus veebilehitsejas Chrome ning lasti YSlow pistikprogrammil hinnata jõudlust. Üldine hinne jõudluse eest oli 76 punkti 100 võimalikust punktist. YSlow soovitas kasutada sisuedastusvõrku, kokku pakkida gzip-meetodiga failid base.css, index.css, base.js, main.js, computed.js, TodoLocalStorage.js, CSSToggleWidget.js, Todo.js, TodoEscape.js, TodoFocus.js, Todos.js, TodoEnter.js ja dojo.js, lisada Expires-päiseid mitmetele failidele, teha vähem HTTP-päringuid ehk kombineerida JavaScripti faile kokku vähemaks arvuks välisteks JavaScripti failideks ning minimeerida faili main.js. Autor annab TodoMVC testrakendusel põhineva jõudluse eest ühe punkti.

3. Tulemused

Allpool on tabelis 19 esitatud hindamise kokkuvõtvad tulemused. Tabeliga ning kogu tööga seonduv kokkuvõttev tekst paikneb neljandas peatükis.

Tabel 19. Edetabel

Asukoht edetabelis ning nimi	Startimishinnang	Kogukond ja dokumentatsioon	Arendustööriistade tugi	Testimistugi	Litsents äri vaatepunktist	Keele ja vormingu tugi	TodoMVC testrakendusel põhinev keerukus	TodoMVC testrakendusel põhinev õppimine ja õppimiskiirus	TodoMVC testrakendusel põhinev jõudlus	Summa
1. AngularJS	3	3	3	3	3	2	2	0	3	22
2. KnockoutJS	3	2	3	3	3	1	1	3	2	21
3. Ember.js	1	3	2	3	3	1	2	3	1	19
4. React	3	3	2	3	3	1	0	2	1	18
5. Dojo	3	1	2	3	3	3	0	0	1	16
6. Backbone.js	2	2	2	3	3	0	1	0	1	14

Õppimise ja õppimiskiiruse peatükis võis bakalureusetöö autorit hakata mõjutama töö käigus järjest suurenev dokumentatsiooni, uute tehnoloogiate ning JavaScriptiga tegelemise vilumus. Seda väidet toetab vastava alapeatükiga tegelemisele kulunud aja vähenemine kõigepealt 16 tunnilt 10 tunnile ning seejärel 7 tunnile. Väite vastu on aga fakt, et viimasena käsitletud teegi Dojo mõistmisele ning kirjalikul kujul selgitamisele kulus ligikaudu 33 tundi. Bakalaureusetöö viimistlemise käigus lisandus ajakulu, mida ei liidetud hindamiseks läinud tulemustele juurde.

4. Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua süstemaatiline võrdlev käsitlus JavaScripti raamistikest AngularJS, Backbone.js, React, Ember.js, KnockoutJS ja Dojo. Autor hindas neid vastavalt töös toodud reeglitele.

Startimishinnangu kategoorias leidus iga raamistiku puhul vähemalt kolm seda raamistikku kasutavat edukat ettevõtet. Hindamisel läksid arvesse järgnevad firmad: Virgin America, MSNBC, Transferwise, Walmart, Trello, Atlassian, Deezer, Dropbox, Facebook, Yahoo, Groupon, Zendesk, BMW, Eventim UK, AmeriTrade, Cisco, IBM ja JPMorgan. Ember ja Backbone kaotasid stardihinnangus algelise rakenduse loomise juures punkte, sest nõudsid arendajalt rohkemate sammude tegemist kui teised raamistikud.

Kõigil raamistikel leidus koodinäiteid sisaldav dokumentatsioon. Knockouti puhul polnud aga arendajal võimalik materjale muuta või soovitada muudatusi. Dojo peamise põhipaketi `dojo` Githubi lehel oli vaatajate, tähega märkijate ning koodi harutajate summa 1356. Angulari, Backbone'i, Reacti, Emberi ja Knockouti vastavad arvväärtused hindamishetkel olid aga 64150, 31112, 45090, 20245 ning 9079. Ühelgi võrdlusobjektil ei moodustanud Stack Exchange'is mittevastatuks märgitud küsimused rohkem kui 40% kõigist raamistiku nime kandva sildiga küsimustest.

Pidades silmas arendustööriistu Eclipse, Jetbrains PhpStorm/WebStorm, Sublime Text, NetBeans ja Visual Studio, moodustus kaks gruppi. AngularJS ning KnockoutJS olid toetatud vähemalt nelja arendustööriista poolt, ülejäänud aga kahe või kolme poolt.

Igale raamistikule leidus ühiktestimist seletav materjal koos koodinäidetega ning litsents, mis võimaldab raamistikuga loodud rakendust kasutada äriotstarbel. MIT litsentsi kasutavad AngularJS, Backbone.js, Ember.js ja KnockoutJS. Reacti puhul kehtib BSD 3 klausli litsents koos Facebooki avalike projektide patendiga. Dojo pakub võimalust valida BSD 3 klausli litsentsi ning tasuta akadeemiline litsentsi versiooni 2.1 vahel.

Konkreetselt Backbone'ile loodud keele ja vormingu tuge pakkuv tarkvara puudub või on raskesti leitav. Dojos on aga mainitud võimalused sisse ehitatud. AngularJS võimaldab kasutada asukohale sobivaid formaate. Kuvakeele muutmiseks tuleb raamistikust eraldiseisvat lisatarkvara kasutada. KnockoutJS, Ember.js ning React võimaldavad kasutajal rakenduse keelt muuta või kasutada asukohale sobivaid formaate, kuid selleks tuleb raamistikust eraldiseisvat lisatarkvara kasutada. Samas need lisatööriistad on mõeldud just nendele raamistikele.

Keerukuse kategoorias hindas käesoleva töö autor projekti *compexity-report* analüüsiprogrammi kasutades 15 TodoMVC testrakenduse keerukust ja määras kindlaks Angulari, Backbone'i, Reacti, Emberi, Knockouti ja Dojo testrakenduste asukohad keerukuse pingeridades. Arvestati lähtekoodi loogiliste ridade arvu summat, McCabe'i tsüklomaatilist keerukust, Halstead'i jõupingutuste keskmist funktsioonide põhjal ja hooldatavuse indeksit. Võrdlusesse kuuluvatest raamistikest ei jõudnud ükski vähemalt neljas pingereas esiviisikusse. AngularJS kuulus kahes pingereas esiviisikusse, Ember.js kuulus neljas pingereas esimese kaheksa hulka: mõlemad said seega ülejäänud raamistikest rohkem punkte. Knockout ning Backbone paiknesid ühes pingereas esiviisikus või neljas pingereas esikümnes. Dojo ning Reacti koht oli keerukuse pingeridade lõpus.

Üks oluline osa võrdlusprotsessis oli raamistikega tutvumine, TodoMVC projektist pärit testrakenduste lähtekoodi mõistmine ning kirjalikul kujul selgitamine. Käesolev hindamiskriteerim võimaldas bakalaureusetöö lugejal aru saada, kuidas raamistikud AngularJS,

Backbone.js, React, Ember.js, KnockoutJS ja Dojo toimivad. Emberi ning Knockouti testrakenduse mõistmisele ning kirjalikul kujul selgitamisele kulus ligikaudu 7 tundi, Reacti testrakendusele 10 tundi. Angulari, Bacbone'i ning Dojo vastavad väärtused olid 15, 16 ja 33 tundi. Ember kui üks kahest paremast raamistikust keerukuse kategoorias kuulus õppimise ja õppimiskiiruse kategoorias kahe parima sekka. Dojo kui üks kahest halvimast raamistikust keerukuse kategoorias sai õppimise ja õppimiskiiruse kategoorias miinimumtulemuse.

YSlow soovitas testrakendusi hinnates kasutada sisuedastusvõrku, kokku pakkida faile gzip-meetodiga, lisada Expires-päiseid, teha vähem HTTP-päringuid ehk kombineerida JavaScripti faile kokku vähemaks arvuks välisteks JavaScripti failideks ning minimeerida faile. Kõige parema jõudlushinde sai Angulari testrakendus.

Valmis hinnangutel põhinev edetabel, milles raamistikud paiknevad järgnevas järjestuses: AngularJS, KnockoutJS, Ember.js, React, Dojo ning Backbone.js. Bakalaureusetöö tekst võimaldab lugejal mõista kaasatud raamistikega seonduvaid probleeme ja teha enda jaoks sobivaim valik.

5. Kasutatud allikad

[1] Osmani, A., 2012. *Journey Through The JavaScript MVC Jungle* [Võrgumaterjal]
http://www.smashingmagazine.com/2012/07/journey-through-the-javascript-mvc-jungle/
[Kasutatud 15. august 2015]

[2] Osmani, A., Sorhus, S., Hartig, P., Sawchuk, S., Eberhardt, C., Verschaeve, A., Saccone, S., 2015. *TodoMVC. Helping you select an MV* framework* [Võrgumaterjal]
http://todomvc.com/
[Kasutatud 14. august 2015]

[3] Fowler, M. *Martin Fowler* [Võrgumaterjal]
http://martinfowler.com/aboutMe.html
[Kasutatud 9. veebruar 2016]

[4] Fowler, M., 2005. *InversionOfControl* [Võrgumaterjal]
http://martinfowler.com/bliki/InversionOfControl.html
[Kasutatud 9. veebruar 2016]

[5] Mozilla Developer Network ja individuaalsed kaastöötajad, 2005-2016. *A re-introduction to JavaScript (JS tutorial)* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript
[Kasutatud 10. veebruar 2016]

[6] Mozilla Developer Network ja individuaalsed kaastöötajad, 2005-2016. *JavaScript* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript
[Kasutatud 10. veebruar 2016]

[7] Mozilla Developer Network ja individuaalsed kaastöötajad, 2005-2016. *JavaScript language resources* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources
[Kasutatud 10. veebruar 2016]

[8] Mozilla Developer Network ja individuaalsed kaastöötajad, 2005-2016. *About JavaScript* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript
[Kasutatud 10. veebruar 2016]

[9] Sugrue, J., S., 2013. *Beginning Backbone.js* New York: Apress

[10] Freeman, A., 2014. *Pro AngularJS*, New York: Apress

[11] Osmani, A., 2015. *Learning JavaScript Design Patterns. JavaScript MV* Patterns* [Võrgumaterjal]
https://addyosmani.com/resources/essentialjsdesignpatterns/book/#detailmvcmvp
[Kasutatud 10. veebruar 2016]

[12] Shaked, U., 2014. *AngularJS vs. Backbone.js vs. Ember.js* [Võrgumaterjal]
https://www.airpair.com/js/javascript-framework-comparison
[Kasutatud 11. veebruar 2016]

[13] McKeachie, C., 2014. *JavaScript Framework Guide AngularJS, Backbone, Ember. Sample* [Võrgumaterjal]
https://www.funnyant.com/javascript-framework-guide/

[Kasutatud 11. veebruar 2016]

[14] Funny Ant, 2016. *About* [Võrgumaterjal]
http://www.funnyant.com/about/
[Kasutatud 11. veebruar 2016]

[15] Sheiko, D., 2014. *JavaScript MV* Framework - Making the Right Choice* [Võrgumaterjal]
http://www.slideshare.net/dsheiko/java-script-framework-making-the-right-choice?related=2
[Kasutatud 16. august 2015]

[16] Raible, M., 2015. *Comparing Hot JavaScript Frameworks: AngularJS, Ember.js and React.js - SpringOne 2GX 2015* [Võrgumaterjal]
http://www.slideshare.net/mraible/comparing-hot-javascript-frameworks-angularjs-emberjs-and-reactjs-springone-2gx-2015http://www.funnyant.com/about/
[Kasutatud 11. veebruar 2016]

[17] npm, Inc. *npm* [Võrgumaterjal]
https://www.npmjs.com/
[Kasutatud 17. veebruar 2016]

[18] Kärner, M., 2013. *Avatud lähtekoodiga veebipõhiste kasutajaliideste raamistike võrdlus* (Tartu Ülikoolis kaitstud bakalaureusetöö) [Võrgumaterjal]
http://dspace.ut.ee/handle/10062/32832
[Kasutatud 1. mai 2016]

[19] Osmani, A., Sorhus, S., Hartig, P., Sawchuk, S., 2013. *todomvc/license.md* [Võrgumaterjal]
https://github.com/tastejs/todomvc/blob/master/license.md
[Kasutatud 11. mai 2016]

[20] GitHub Inc & tastejs/todomvc, Sorhus, S., Hartig, P., Carraway, E., Trott, R., Verschaeve, A., Pablo, C., Couch, A., 2016. *Application Specification* [Võrgumaterjal]
https://github.com/tastejs/todomvc/blob/master/app-spec.md
[Kasutatud 18. aprill 2016]

[21] W3Schools, 2015. *HTML5 Local Storage* [Võrgumaterjal]
http://www.w3schools.com/html/html5_webstorage.asp
[Kasutatud 2. veebruar 2016]

[22] Burgdorf, C., Bidelman, E., Mumm, J., Minar, I., 2015. *AngularJS TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/angularjs
[Kasutatud 3. veebruar 2016]

[23] Osmani, A., 2015. *Backbone.js TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/backbone
[Kasutatud 5. veebruar 2016]

[24] Hunt, P., 2015. *React TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/react
[Kasutatud 5. veebruar 2016]

[25] Forsyth, C., Dale, T., Osmani, A., 2015. *Ember.js TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/gh-pages/examples/emberjs

[Kasutatud 5. veebruar 2016]

[26] Sharma, A., Niemeyer, R., 2015. *Knockout.js TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/knockoutjs
[Kasutatud 5. veebruar 2016]

[27] Thomas, J., Chatelain, E., Sudoh, A., 2015. *Dojo TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/dojo
[Kasutatud 6. veebruar 2016]

[28] Malakoff, K., 2015. *Knockback.js TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/knockback
[Kasutatud 6. veebruar 2016]

[29] Bitovi, 2015. *CanJS TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/canjs
[Kasutatud 6. veebruar 2016]

[30] The Polymer Authors, 2015. *Polymer TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/polymer
[Kasutatud 6. veebruar 2016]

[31] Hakes, T., Monette, J.-P., 2015. *Mithril TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/mithril
[Kasutatud 6. veebruar 2016]

[32] Joreteg, H., Karrys L., Roberts, P., 2015. *Ampersand TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/ampersand
[Kasutatud 6. veebruar 2016]

[33] You, E., 2015. *Vue.js TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/vue
[Kasutatud 6. veebruar 2016]

[34] Overson, J., Bailey D., 2015. *Backbone.Marionette TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/backbone_marionette
[Kasutatud 6. veebruar 2016]

[35] Karon, M., 2015. *TroopJS TodoMVC* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/troopjs_require
[Kasutatud 6. veebruar 2016]

[36] Kuklis, M., 2015. *Flight TodoMVC Example* [Võrgumaterjal]
https://github.com/tastejs/todomvc/tree/master/examples/flight
[Kasutatud 6. veebruar 2016]

[37] Nath, D., S., 2014. *Javascript Frameworks Comparison - Angular, Knockout, Ember and Backbone* [Võrgumaterjal]
http://www.slideshare.net/deepusnath/javascript-frameworks-comparison-angular-knockout-ember-and-backbone?qid=0f80ad09-6db1-4023-bf8a-f59c4bf44a86&v=default&b=&from_search=1
[Kasutatud 15. august 2015]

[38] The Eclipse Foundation, 2015. *Eclipse Marketplace* [Võrgumaterjal]
https://marketplace.eclipse.org/search/site/

[Kasutatud 4. veebruar 2016]

[39] Jetbrains. *Plugins* [Võrgumaterjal]
https://plugins.jetbrains.com/
[Kasutatud 4. veebruar 2016]

[40] Bond, W., 2015 *Package Control* [Võrgumaterjal]
https://packagecontrol.io/
[Kasutatud 4. veebruar 2016]

[41] Oracle Corporation, 2015. *NetBeans Plugin Portal* [Võrgumaterjal]
http://plugins.netbeans.org/PluginPortal/
[Kasutatud 4. veebruar 2016]

[42] Microsoft, 2015. *Products and Extensions for Visual Studio* [Võrgumaterjal]
https://visualstudiogallery.msdn.microsoft.com/
[Kasutatud 4. veebruar 2016]

[43] Stack Exchange Inc, Aquatic, 2009. *Restrictions of GPL on javascript libraries [closed]* [Võrgumaterjal]
https://stackoverflow.com/questions/1239470/restrictions-of-gpl-on-javascript-libraries
[Kasutatud 18. august 2015]

[44] Booth, P., 2015. *complexity-report* [Võrgumaterjal]
https://github.com/philbooth/complexity-report
[Kasutatud 19. august 2015]

[45] Gómez, R., 2013. *How Complex are TodoMVC Implementations* [Võrgumaterjal]
http://blog.coderstats.net/todomvc-complexity/
[Kasutatud 19. august 2015]

[46] IBM, 2015 *IBM Knowledge Center. Halstead effort* [Võrgumaterjal]
http://www-01.ibm.com/support/knowledgecenter/SS3JHP_6.1.0/com.ibm.raa.analyze.doc/common/chalstd.html
[Kasutatud 22. august 2015]

[47] Duran, M. *YSlow Help* [Võrgumaterjal]
http://yslow.org/user-guide/
[Kasutatud 16. aprill 2016]

[48] Yahoo Deeloper Network. *Best Practices for Speeding Up Your Web Site* [Võrgumaterjal]
https://developer.yahoo.com/performance/rules.html
[Kasutatud 16. aprill 2016]

[49] Austin, A., 2015. *An Overview of AngularJS for Managers* [Võrgumaterjal]
http://andrewaustin.com/an-overview-of-angularjs-for-managers/
[Kasutatud 20. august 2015]

[50] Google, 2010–2015. *What Is Angular?* [Võrgumaterjal]
https://docs.angularjs.org/guide/introduction
[Kasutatud 20. august 2015]

[51] Polepeddi, L. *Made With Angular. TransferWise* [Võrgumaterjal]
https://www.madewithangular.com/#/sites/transferwise
[Kasutatud 16.aprill 2016]

[52] Polepeddi, L. *Made With Angular. MSNBC* [Võrgumaterjal]
https://www.madewithangular.com/#/sites/msnbc
[Kasutatud 16.aprill 2016]

[53] Polepeddi, L. *Made With Angular. Virgin America* [Võrgumaterjal]
https://www.madewithangular.com/#/sites/virgin-america
[Kasutatud 16.aprill 2016]

[54] GitHub Inc & Angular, 2015. *angular.js* [Võrgumaterjal]
https://github.com/angular/angular.js
[Kasutatud 21. august 2015]

[55] Stack Exchange Inc, 2015. *Tag Info. About angularjs* [Võrgumaterjal]
https://stackoverflow.com/tags/angularjs/info
[Kasutatud 21. august 2015]

[56] Stack Exchange Inc, 2015. *Tagged Questions* [Võrgumaterjal]
https://stackoverflow.com/questions/tagged/angularjs?sort=unanswered
[Kasutatud 21. august 2015]

[57] Google, 2010–2016. *AngularJS* [Võrgumaterjal]
https://angularjs.org/
[Kasutatud 17. aprill 2016]

[58] Google, 2010–2015. *PhoneCat Tutorial App* [Võrgumaterjal]
https://docs.angularjs.org/tutorial
[Kasutatud 21. august 2015]

[59] Google, 2010–2015. *Guide to AngularJS Documentation* [Võrgumaterjal]
https://docs.angularjs.org/guide
[Kasutatud 21. august 2015]

[60] Google, 2010–2015. *AngularJS API Docs* [Võrgumaterjal]
https://docs.angularjs.org/api
[Kasutatud 21. august 2015]

[61] Creative Commons, 2015. *Attribution 4.0 International (CC BY 4.0)* [Võrgumaterjal]
https://creativecommons.org/licenses/by/4.0/
[Kasutatud 17. aprill 2016]

[62] The Eclipse Foundation, 2015. *Eclipse Marketplace. AngularJS Eclipse* [Võrgumaterjal]
https://marketplace.eclipse.org/content/angularjs-eclipse
[Kasutatud 21. august 2015]

[63] GitHub Inc & Zerr, A. 2014. *HTML Features* [Võrgumaterjal]
https://github.com/angelozerr/angularjs-eclipse/wiki/HTML-Features
[Kasutatud 21. august 2015]

[64] GitHub Inc & Zerr, A. 2014. *Javascript Features* [Võrgumaterjal]
https://github.com/angelozerr/angularjs-eclipse/wiki/Javascript-Features
[Kasutatud 21. august 2015]

[65] GitHub Inc & Zerr, A. 2014. *Angular Explorer View* [Võrgumaterjal]
https://github.com/angelozerr/angularjs-eclipse/wiki/Angular-Explorer-View
[Kasutatud 21. august 2015]

[66] Bond, W., 2015. *Package Control. AngularJS* [Võrgumaterjal]

https://packagecontrol.io/search/angularjs
[Kasutatud 21. august 2015]

[67] GitHub Inc & AngularUI. 2015. *angular-ui/AngularJS-sublime-package* [Võrgumaterjal]
https://github.com/angular-ui/AngularJS-sublime-package
[Kasutatud 21. august 2015]

[68] Jetbrains. *PhpStorm Plugins. AngularJS* [Võrgumaterjal]
https://plugins.jetbrains.com/plugin/6971?pr=phpStorm
[Kasutatud 21. august 2015]

[69] GitHub Inc & Starovoyt, A. 2015. *intellij-plugins/AngularJS* [Võrgumaterjal]
https://github.com/JetBrains/intellij-plugins/tree/master/AngularJS
[Kasutatud 21. august 2015]

[70] Oracle Corporation, 2015. *AngularJS Tools - plugin detail* [Võrgumaterjal]
http://plugins.netbeans.org/plugin/40296/angularjs-tools
[Kasutatud 21. august 2015]

[71] GitHub Inc & Dolk, M., 2015. *angular-netbeans-plugin* [Võrgumaterjal]
https://github.com/mdolk/angular-netbeans-plugin
[Kasutatud 21. august 2015]

[72] Microsoft, 2015. *Client-side. Angular.js* [Võrgumaterjal]
http://webtooling.visualstudio.com/frameworks/client-side/#angular.js
[Kasutatud 21. august 2015]

[73] Google, 2010–2015. *Developer Guide / Unit Testing* [Võrgumaterjal]
https://docs.angularjs.org/guide/unit-testing
[Kasutatud 21. august 2015]

[74] GitHub Inc & angular, 2016. *angular/LICENSE* [Võrgumaterjal]
https://github.com/angular/angular/blob/master/LICENSE
[Kasutatud 1. märts 2016]

[75] FOSSA, Inc, 2012–2015. *MIT License (Expat)* [Võrgumaterjal]
https://tldrlegal.com/license/mit-license#summary
[Kasutatud 18. veebruar 2016]

[76] Google, 2010–2015. *Developer Guide / i18n and l10n* [Võrgumaterjal]
https://docs.angularjs.org/guide/i18n
[Kasutatud 22. august 2015]

[77] GitHub Inc & Angular, 2015. *angular/angular.js/src/ngLocale* [Võrgumaterjal]
https://github.com/angular/angular.js/tree/master/src/ngLocale
[Kasutatud 22. august 2015]

[78] Precht, P., Philipp, J., Princhenko, M., Valero, D., Longaeret, B., Other Github contributors, 2015. *angular translate* [Võrgumaterjal]
https://angular-translate.github.io/
[Kasutatud 22. august 2015]

[79] Google, 2010–2015. *AngularJS API Docs.ngRoute* [Võrgumaterjal]
https://docs.angularjs.org/api/ngRoute
[Kasutatud 30. jaanuar 2016]

[80] Google, 2010–2016. *AngularJS API Docs. ngResource* [Võrgumaterjal]

https://docs.angularjs.org/api/ngResource
[Kasutatud 30. jaanuar 2016]

[81] Google, 2010–2015. *AngularJS API Docs. ngRoute/provider components in ngRoute / $routeProvider* [Võrgumaterjal]
https://docs.angularjs.org/api/ngRoute/provider/$routeProvider
[Kasutatud 30. jaanuar 2016]

[82] Google, 2010–2016. */ Developer Guide / Scopes* [Võrgumaterjal]
https://docs.angularjs.org/guide/scope
[Kasutatud 4. veebruar 2016]

[83] Lerner, A., 2013. *ng-book. The Complete Book on AngularJS*, Avaldaja: Fullstack.io

[84] Google, 2010–2015. *AngularJS API Docs. ng / type components in ng / $rootScope.Scope. $watch(watchExpression, listener, [objectEquality]);* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/type/$rootScope.Scope#$watch
[Kasutatud 23. august 2015]

[85] Google, 2010–2016. *AngularJS API Docs. ng / service components in ng / $filter* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/service/$filter#$digest
[Kasutatud 19. aprill 2016]

[86] Google, 2010–2015. *AngularJS API Docs. ngRoute / service components in ngRoute / $route. $routeChangeSuccess* [Võrgumaterjal]
https://docs.angularjs.org/api/ngRoute/service/$route#$routeChangeSuccess
[Kasutatud 23. august 2015]

[87] Google, 2010–2015. *AngularJS API Docs. $on(name, listener)* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/type/$rootScope.Scope#$on
[Kasutatud 23. august 2015]

[88] Google, 2010–2015. *AngularJS API Docs. ngRoute / service components in ngRoute / $routeParams* [Võrgumaterjal]
https://docs.angularjs.org/api/ngRoute/service/$routeParams
[Kasutatud 23. august 2015]

[89] Google, 2010–2016. *AngularJS API Docs. / API Reference / ng / filter components in ng / filter* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/filter/filter
[Kasutatud 4. veebruar 2016]

[90] Google, 2010–2016. *AngularJS API Docs. ngModel* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/directive/ngModel
[Kasutatud 1. veebruar 2016]

[91] Mozilla Developer Network ja individuaalsed kaastöötajad, *String.prototype.trim()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/Trim
[Kasutatud 13. veebruar 2016]

[92] Google, 2010–2016. *AngularJS API Docs. ngDisabled* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/directive/ngDisabled
[Kasutatud 31. jaanuar 2016]

[93] Google, 2010–2016. *AngularJS API Docs. / API Reference / ng / directive components in ng / ngDblclick* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/directive/ngDblclick
[Kasutatud 19. aprill 2016]

[94] Google, 2010–2016. *AngularJS API Docs. angular.extend* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/function/angular.extend
[Kasutatud 31. jaanuar 2016]

[95] Google, 2010–2016. *AngularJS API Docs. ngSubmit* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/directive/ngSubmit
[Kasutatud 31. jaanuar 2016]

[96] Mozilla Developer Network ja individuaalsed kaastöötajad. *blur (event)* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/Events/blur
[Kasutatud 31. jaanuar 2016]

[97] Mozilla Developer Network ja individuaalsed kaastöötajad, *keydown* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/Events/keydown
[Kasutatud 14. veebruar 2016]

[98] Google, 2010–2016. *Developer Guide/Directives* [Võrgumaterjal]
https://docs.angularjs.org/guide/directive
[Kasutatud 1. veebruar 2016]

[99] Mozilla Developer Network ja individuaalsed kaastöötajad. *HTMLElement.focus()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/focus
[Kasutatud 14. veebruar 2016]

[100] Google, 2010–2016. */ API Reference / ng / service components in ng / $timeout* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/service/$timeout
[Kasutatud 1. veebruar 2016]

[101] Google, 2010–2016. *AngularJS API Docs. ngChange* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/directive/ngChange
[Kasutatud 1.veebruar 2016]

[102] Airbnb, 2014. *Polyglot.js* [Võrgumaterjal]
http://airbnb.io/polyglot.js/
[Kasutatud 5. veebruar 2016]

[103] Google, 2010–2016. *AngularJS API Docs. $http* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/service/$http
[Kasutatud 2. veebruar 2016]

[104] Google, 2010–2016. *AngularJS API Docs. $injector* [Võrgumaterjal]
https://docs.angularjs.org/api/auto/service/$injector
[Kasutatud 2. veebruar 2016]

[105] Google, 2010–2016. *AngularJS API Docs. $resource* [Võrgumaterjal]
https://docs.angularjs.org/api/ngResource/service/$resource
[Kasutatud 2. veebruar 2016]

[106] Mozilla Developer Network ja individuaalsed kaastöötajad. *Array.prototype.slice()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/slice
[Kasutatud 26. märts 2016]

[107] Mozilla Developer Network ja individuaalsed kaastöötajad. *Array.prototype.splice()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice
[Kasutatud 26. märts 2016]

[108] Google, 2010–2016. *AngularJS API Docs. angular.copy* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/function/angular.copy
[Kasutatud 2. veebruar 2016]

[109] Mozilla Developer Network ja individuaalsed kaastöötajad. *Array.prototype.push()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/push
[Kasutatud 20. aprill 2016]

[110] Google, 2010–2016. *AngularJS API Docs. $q*[Võrgumaterjal]
https://docs.angularjs.org/api/ng/service/$q
[Kasutatud 2. veebruar 2016]

[111] Mozilla Developer Network ja individuaalsed kaastöötajad. *Storage.getItem()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/API/Storage/getItem
[Kasutatud 2. veebruar 2016]

[112] Mozilla Developer Network ja individuaalsed kaastöötajad. *Storage.setItem()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/API/Storage/setItem
[Kasutatud 2. veebruar 2016]

[113] Mozilla Developer Network ja individuaalsed kaastöötajad. *JSON.stringify()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify
[Kasutatud 2. veebruar 2016]

[114] Mozilla Developer Network ja individuaalsed kaastöötajad. *JSON.parse()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse
[Kasutatud 2. veebruar 2016]

[115] Google, 2010–2016. *AngularJS API Docs. $q. The Deferred API* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/service/$q/#the-deferred-api
[Kasutatud 2. veebruar 2016]

[116] Google, 2010–2016. *The Promise API* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/service/$q/#the-promise-api
[Kasutatud 2. veebruar 2016]

[117] Google, 2010–2016. *AngularJS API Docs. ngCloak* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/directive/ngCloak
[Kasutatud 2. veebruar 2016]

[118] Google, 2010–2016. *AngularJS API Docs. ngApp* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/directive/ngApp
[Kasutatud 2. veebruar 2016]

[119] Google, 2010–2016. *AngularJS API Docs. ngView* [Võrgumaterjal]

https://docs.angularjs.org/api/ngRoute/directive/ngView
[Kasutatud 3. veebruar 2016]

[120] Google, 2010–2016. *AngularJS API Docs. ngShow* [Võrgumaterjal]
https://docs.angularjs.org/api/ng/directive/ngShow
[Kasutatud 3. veebruar 2016]

[121] W3Schools, 2015. *HTML <input> autofocus Attribute* [Võrgumaterjal]
http://www.w3schools.com/tags/att_input_autofocus.asp
[Kasutatud 3. veebruar 2016]

[122] Google, 2010–2016. *AngularJS API Docs. ngRepeat* [Võrgumaterjal]

https://docs.angularjs.org/api/ng/directive/ngRepeat
[Kasutatud 4. veebruar 2016]

[123] Google, 2010–2016. *AngularJS API Docs. ngClass* [Võrgumaterjal]

https://docs.angularjs.org/api/ng/directive/ngClass
[Kasutatud 4. veebruar 2016]

[124] Google, 2010–2016. *AngularJS API Docs. ngPluralize* [Võrgumaterjal]

https://docs.angularjs.org/api/ng/directive/ngPluralize
[Kasutatud 4. veebruar 2016]

[125] Backone.js, *Backbone.js* [Võrgumaterjal]
http://backbonejs.org/
[Kasutatud 3. veebruar 2016]

[126] Backone.js, *Examples* [Võrgumaterjal]
http://backbonejs.org/#examples
[Kasutatud 3. veebruar 2016]

[127] GitHub Inc & yanlum, 2016, *yanlum/HelloWorldBackbone* [Võrgumaterjal]
https://github.com/yanlum/HelloWorldBackbone
[Kasutatud 3. veebruar 2016]

[128] Backone.js, *Backbone.View* [Võrgumaterjal]
http://backbonejs.org/#View
[Kasutatud 3. veebruar 2016]

[129] Underscore.js, *template* [Võrgumaterjal]
http://underscorejs.org/#template
[Kasutatud 3. veebruar 2016]

[130] Backone.js, *render* [Võrgumaterjal]
http://backbonejs.org/#View-render
[Kasutatud 13. veebruar 2016]

[131] GitHub Inc & jashkenas 2016. *jashkenas/backbone* [Võrgumaterjal]
https://github.com/jashkenas/backbone
[Kasutatud 3. veebruar 2016]

[132] Stack Exchange Inc, 2016. *Tagged Questions* [Võrgumaterjal]
https://stackoverflow.com/questions/tagged/backbone.js
[Kasutatud 3. veebruar 2016]

[133] Stack Exchange Inc, 2016. *Unanswered Questions* [Võrgumaterjal]
https://stackoverflow.com/unanswered/tagged/backbone.js
[Kasutatud 3. veebruar 2016]

[134] The Eclipse Foundation, 2016. *Eclipse Marketplace. DaVinci Studio* [Võrgumaterjal]
https://marketplace.eclipse.org/content/davinci-studio
[Kasutatud 4. veebruar 2016]

[135] tomasztunik., 2013. *Backbone.js* [Võrgumaterjal]
https://packagecontrol.io/packages/Backbone.js
[Kasutatud 4. veebruar 2016]

[136] pwhisenhunt., 2013. *Lazy Backbone.js* [Võrgumaterjal]
https://packagecontrol.io/browse/authors/pwhisenhunt
[Kasutatud 4. veebruar 2016]

[137] Microsoft, 2015. *Client-side. Others* [Võrgumaterjal]
http://webtooling.visualstudio.com/frameworks/client-side/#others
[Kasutatud 19. veebruar 2015]

[138] Osmani, A., 2012–2013. *Developing Backbone.js Applications. Unit Testing* [Võrgumaterjal]
https://addyosmani.com/backbone-fundamentals/#unit-testing
[Kasutatud 5. veebruar 2016]

[139] Ashkenas, J., 2010–2016. *LICENSE* [Võrgumaterjal]
https://github.com/jashkenas/backbone/blob/master/LICENSE
[Kasutatud 18. veebruar 2016]

[140] LocalePlanet. *LocalePlanet* [Võrgumaterjal]
http://www.localeplanet.com/index.html
[Kasutatud 5. veebruar 2016]

[141] De Souza, R. X., González, S., Zaefferer, J., 2016. *globalize* [Võrgumaterjal]
https://www.npmjs.com/package/globalize
[Kasutatud 5. veebruar 2016]

[142] Github Pages & De Souza, R. X., 2015. *Javascript-globalization* [Võrgumaterjal]
https://rxaviers.github.io/javascript-globalization/
[Kasutatud 5. veebruar 2016]

[143] The jQuery Foundation, 2016. *.ready()* [Võrgumaterjal]
https://api.jquery.com/ready/
[Kasutatud 11. veebruar 2016]

[144] Stack Overflow & Alnitak, *What does "var FOO = FOO || {}" mean in Javascript?. Answered Alnitak* [Võrgumaterjal]
https://stackoverflow.com/questions/6439579/what-does-var-foo-foo-mean-in-javascript

[Kasutatud 13. veebruar 2016]

[145] Backone.js, *Backbone.Model* [Võrgumaterjal]
http://backbonejs.org/#Model
[Kasutatud 11. veebruar 2016]

[146] Backone.js, *extend* [Võrgumaterjal]
http://backbonejs.org/#Model-extend
[Kasutatud 11. veebruar 2016]

[147] Backone.js, *defaults* [Võrgumaterjal]
http://backbonejs.org/#Model-defaults
[Kasutatud 11. veebruar 2016]

[148] Backone.js, *save* [Võrgumaterjal]
http://backbonejs.org/#Model-save
[Kasutatud 11. veebruar 2016]

[149] Backone.js, *Backbone.sync* [Võrgumaterjal]
http://backbonejs.org/#Sync
[Kasutatud 12. veebruar 2016]

[150] Backone.js, *backbone.localStorage.js* [Võrgumaterjal]
http://backbonejs.org/docs/backbone.localStorage.html
[Kasutatud 12. veebruar 2016]

[151] Backone.js, *Backbone.Collection* [Võrgumaterjal]
http://backbonejs.org/#Collection
[Kasutatud 12. veebruar 2016]

[152] Backone.js, *model* [Võrgumaterjal]
http://backbonejs.org/#Collection-model
[Kasutatud 12. veebruar 2016]

[153] Backone.js, *where* [Võrgumaterjal]
http://backbonejs.org/#Collection-where
[Kasutatud 12. veebruar 2016]

[154] Backone.js, *comparator* [Võrgumaterjal]
http://backbonejs.org/#Collection-comparator
[Kasutatud 12. veebruar 2016]

[155] Backone.js, *Backbone.Router* [Võrgumaterjal]
http://backbonejs.org/#Router
[Kasutatud 12. veebruar 2016]

[156] Backone.js, *routes* [Võrgumaterjal]
http://backbonejs.org/#Router-routes
[Kasutatud 12. veebruar 2016]

[157] Backone.js, *trigger* [Võrgumaterjal]
http://backbonejs.org/#Events-trigger
[Kasutatud 3. mai 2016]

[158] Backone.js, *start* [Võrgumaterjal]
http://backbonejs.org/#History-start
[Kasutatud 13. veebruar 2016]

[159] Cherry, B., *JavaScript Module Pattern: In-Depth* [Võrgumaterjal]

http://www.adequatelygood.com/JavaScript-Module-Pattern-In-Depth.html
[Kasutatud 13. veebruar 2016]

[160] Backone.js, *extend* [Võrgumaterjal]
http://backbonejs.org/#View-extend
[Kasutatud 13. veebruar 2016]

[161] Backone.js, *events* [Võrgumaterjal]
http://backbonejs.org/#View-events
[Kasutatud 13. veebruar 2016]

[162] Backone.js, *delegateEvents* [Võrgumaterjal]
http://backbonejs.org/#View-delegateEvents
[Kasutatud 13. veebruar 2016]

[163] Mozilla Developer Network ja individuaalsed kaastöötajad, *keypress* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/Events/keypress
[Kasutatud 13. veebruar 2016]

[164] Backone.js, *constructor / initialize* [Võrgumaterjal]
http://backbonejs.org/#View-constructor
[Kasutatud 13. veebruar 2016]

[165] Backone.js, *$el* [Võrgumaterjal]
http://backbonejs.org/#View-$el
[Kasutatud 13. veebruar 2016]

[166] Backone.js, *$ (jQuery)* [Võrgumaterjal]
http://backbonejs.org/#View-dollar
[Kasutatud 13. veebruar 2016]

[167] Backone.js, *listenTo* [Võrgumaterjal]
http://backbonejs.org/#Events-listenTo
[Kasutatud 13. veebruar 2016]

[168] Underscore.js, *debounce* [Võrgumaterjal]
http://underscorejs.org/#debounce
[Kasutatud 13. veebruar 2016]

[169] Dugas, C., 2013. *Implementing live updates with Backbone.js* [Võrgumaterjal]
http://www.position-absolute.com/articles/implementing-live-updates-with-backbone-js/
[Kasutatud 13. veebruar 2016]

[170] Backone.js, *fetch* [Võrgumaterjal]
http://backbonejs.org/#Collection-fetch
[Kasutatud 13. veebruar 2016]

[171] Backone.js, *reset* [Võrgumaterjal]
http://backbonejs.org/#Collection-reset
[Kasutatud 13. veebruar 2016]

[172] The jQuery Foundation, 2016. *.show()* [Võrgumaterjal]
https://api.jquery.com/show/
[Kasutatud 13. veebruar 2016]

[173] The jQuery Foundation, 2016. *.html()* [Võrgumaterjal]
https://api.jquery.com/html/

[Kasutatud 13. veebruar 2016]

[174] The jQuery Foundation, 2016. *.removeClass()* [Võrgumaterjal]
https://api.jquery.com/removeClass/
[Kasutatud 13. veebruar 2016]

[175] The jQuery Foundation, 2016. *.filter()* [Võrgumaterjal]
https://api.jquery.com/filter/
[Kasutatud 13. veebruar 2016]

[176] The jQuery Foundation, 2016. *.addClass()* [Võrgumaterjal]
https://api.jquery.com/addClass/
[Kasutatud 13. veebruar 2016]

[177] The jQuery Foundation, 2016. *.hide()* [Võrgumaterjal]
https://api.jquery.com/hide/
[Kasutatud 13. veebruar 2016]

[178] Backone.js, *Binding "this"* [Võrgumaterjal]
http://backbonejs.org/#FAQ-this
[Kasutatud 13. veebruar 2016]

[179] Underscore.js, *invoke* [Võrgumaterjal]
http://underscorejs.org/#invoke
[Kasutatud 13. veebruar 2016]

[180] Backone.js, *destroy* [Võrgumaterjal]
http://backbonejs.org/#Model-destroy
[Kasutatud 13. veebruar 2016]

[181] Mozilla Developer Network ja individuaalsed kaastöötajad, *dblclick* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/Events/dblclick
[Kasutatud 14. veebruar 2016]

[182] Backone.js, *remove* [Võrgumaterjal]
http://backbonejs.org/#View-remove
[Kasutatud 14. veebruar 2016]x

[183] Backone.js, *stopListening* [Võrgumaterjal]
http://backbonejs.org/#Events-stopListening
[Kasutatud 14. veebruar 2016]

[184] Backone.js, *changed* [Võrgumaterjal]
http://backbonejs.org/#Model-changed
[Kasutatud 14. veebruar 2016]

[185] The jQuery Foundation, 2016. *.toggleClass()* [Võrgumaterjal]
https://api.jquery.com/toggleclass/#toggleClass-className-state
[Kasutatud 14. veebruar 2016]

[186] The jQuery Foundation, 2016. *.hasClass()* [Võrgumaterjal]
https://api.jquery.com/hasClass/
[Kasutatud 14. veebruar 2016]

[187] Facebook Inc., 2013-2016. *React* [Võrgumaterjal]
https://facebook.github.io/react/index.html
[Kasutatud 16. veebruar 2016]

[188] Gackenheimer, C., 2015. *Introduction to React*, New York: Apress

[189] Facebook Inc., 2013-2016. *Avoiding reconciling the DOM* [Võrgumaterjal]
https://facebook.github.io/react/docs/advanced-performance.html#avoiding-reconciling-the-dom
[Kasutatud 16. veebruar 2016]

[190] Facebook Inc., 2013-2016. *Reconciliation* [Võrgumaterjal]
https://facebook.github.io/react/docs/reconciliation.html
[Kasutatud 16. veebruar 2016]

[191] Facebook Inc., 2013-2016. *Why JSX?* [Võrgumaterjal]
https://facebook.github.io/react/docs/jsx-in-depth.html#why-jsx
[Kasutatud 16. veebruar 2016]

[192] Github, Inc. & Facebook Inc., 2016. *Sites Using React* [Võrgumaterjal]
https://facebook.github.io/react/docs/jsx-in-depth.html#why-jsx
[Kasutatud 16. veebruar 2016]

[193] Babel. *Babel is a JavaScript compiler.* [Võrgumaterjal]
https://babeljs.io/
[Kasutatud 16. veebruar 2016]

[194] GitHub Inc & facebook, 2016. *facebook/react* [Võrgumaterjal]
https://github.com/facebook/react
[Kasutatud 17. veebruar 2016]

[195] Stack Exchange Inc, 2016. *Tagged Questions* [Võrgumaterjal]
https://stackoverflow.com/questions/tagged/reactjs
[Kasutatud 17. veebruar 2016]

[196] Stack Exchange Inc, 2016. *Unanswered Questions* [Võrgumaterjal]
https://stackoverflow.com/unanswered/tagged/reactjs
[Kasutatud 17. veebruar 2016]

[197] Facebook Inc., 2013-2016. *Top-Level API* [Võrgumaterjal]
https://facebook.github.io/react/docs/top-level-api.html
[Kasutatud 17. veebruar 2016]

[198] GitHub Inc & facebook, 2016. *facebook/react. Complementary Tools* [Võrgumaterjal]
https://github.com/facebook/react/wiki/Complementary-Tools
[Kasutatud 18. veebruar 2016]

[199] Oracle Corporation, 2015. *Bug 250778 - Add support for JSX (ReactJS)* [Võrgumaterjal]
https://netbeans.org/bugzilla/show_bug.cgi?id=250778
[Kasutatud 18. veebruar 2016]

[200] Wielenga, G., 2016. *React.js and NetBeans IDE (Part 2)* [Võrgumaterjal]
https://blogs.oracle.com/geertjan/entry/react_js_and_netbeans_ide1
[Kasutatud 18. veebruar 2016]

[201] Prigara, E. *Working with ReactJS in WebStorm: Coding Assistance* [Võrgumaterjal]
https://blog.jetbrains.com/webstorm/2015/10/working-with-reactjs-in-webstorm-coding-assistance/
[Kasutatud 4. mai 2016]

[202] Jetbrains. *WebStorm Plugins. React-Templates* [Võrgumaterjal]

https://plugins.jetbrains.com/plugin/7648?pr=webStorm
[Kasutatud 18. veebruar 2016]

[203] babel, 2015. *Babel (babel-sublime)* [Võrgumaterjal]
https://packagecontrol.io/packages/Babel
[Kasutatud 18. veebruar 2016]

[204] babel, 2015. *Babel Snippets (babel-sublime-snippets)* [Võrgumaterjal]
https://packagecontrol.io/packages/Babel%20Snippets
[Kasutatud 18. veebruar 2016]

[205] Microsoft. *Client-side. React.js* [Võrgumaterjal]
http://webtooling.visualstudio.com/frameworks/client-side/#react.js
[Kasutatud 18. veebruar 2016]

[206] Microsoft. *React Snippet Pack* [Võrgumaterjal]
https://visualstudiogallery.msdn.microsoft.com/234d79e9-f0fd-41e1-a926-850da8e8c7d7
[Kasutatud 18. veebruar 2016]

[207] Facebook Inc., 2013-2016. *Test Utilities* [Võrgumaterjal]
https://facebook.github.io/react/docs/test-utils.html
[Kasutatud 23. veebruar 2016]

[208] Facebook Inc., 2015. *Tutorial – React* [Võrgumaterjal]
https://facebook.github.io/jest/docs/tutorial-react.html
[Kasutatud 23. veebruar 2016]

[209] GitHub Inc & facebook 2016. *facebook/react/LICENSE* [Võrgumaterjal]
https://github.com/facebook/react/blob/master/LICENSE
[Kasutatud 18. veebruar 2016]

[210] FOSSA, Inc, 2012–2015. *BSD 3-Clause License (Revised)* [Võrgumaterjal]
https://tldrlegal.com/license/bsd-3-clause-license-%28revised%29
[Kasutatud 18. veebruar 2016]

[211] Facebook, 2016. *Updating Our Open Source Patent Grant* [Võrgumaterjal]
https://code.facebook.com/posts/1639473982937255/updating-our-open-source-patent-grant/
[Kasutatud 18. veebruar 2016]

[212] GitHub Inc & facebook, 2016. *facebook/react/PATENTS* [Võrgumaterjal]
https://github.com/facebook/react/blob/master/PATENTS
[Kasutatud 18. veebruar 2016]

[213] Yahoo! Inc., 2014. *React Intl* [Võrgumaterjal]
http://formatjs.io/react/
[Kasutatud 23. veebruar 2016]

[214] IBM ja teised, 2000-2009. *ICU User Guide. Formatting Messages* [Võrgumaterjal]
http://userguide.icu-project.org/formatparse/messages
[Kasutatud 23. veebruar 2016]

[215] GitHub Inc & yahoo, Ferraiuolo, E., 2016. *Locale Data APIs* [Võrgumaterjal]
https://github.com/yahoo/react-intl/wiki/API#react-intl-api
[Kasutatud 4. mai 2016]

[216] Babel. *Try it out* [Võrgumaterjal]
https://babeljs.io/repl/

[Kasutatud 6. veebruar 2016]

[217] O'Shannessy, P., 2015. *Deprecating JSTransform and react-tools* [Võrgumaterjal]
https://facebook.github.io/react/blog/2015/06/12/deprecating-jstransform-and-react-tools.html
[Kasutatud 23. veebruar 2016]

[218] Mozilla Developer Network ja individuaalsed kaastöötajad, *Introduction to Object-Oriented JavaScript* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript
[Kasutatud 23. veebruar 2016]

[219] Mozilla Developer Network ja individuaalsed kaastöötajad, *The methods* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript#The_methods
[Kasutatud 23. veebruar 2016]

[220] Mozilla Developer Network ja individuaalsed kaastöötajad, *Array.prototype.concat()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/concat
[Kasutatud 23. veebruar 2016]

[221] Mozilla Developer Network ja individuaalsed kaastöötajad, *Array.prototype.map()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map
[Kasutatud 23. veebruar 2016]

[222] Mozilla Developer Network ja individuaalsed kaastöötajad, *Array.prototype.filter()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter
[Kasutatud 23. veebruar 2016]

[223] Mozilla Developer Network ja individuaalsed kaastöötajad, *Math.random()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random
[Kasutatud 24. veebruar 2016]

[224] Mozilla Developer Network ja individuaalsed kaastöötajad, *Arguments object* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/arguments
[Kasutatud 24. veebruar 2016]

[225] Mozilla Developer Network ja individuaalsed kaastöötajad, *Object.prototype.hasOwnProperty()*[Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/hasOwnProperty
[Kasutatud 24. veebruar 2016]

[226] Facebook Inc., 2013-2016. *React.createClass* [Võrgumaterjal]
https://facebook.github.io/react/docs/top-level-api.html#react.createclass

[Kasutatud 24. veebruar 2016]

[227] Facebook Inc., 2013-2016. *render* [Võrgumaterjal]
https://facebook.github.io/react/docs/component-specs.html#render
[Kasutatud 6. mai 2016]

[228] W3Schools, 2015. *HTML <a> Tag* [Võrgumaterjal]
http://www.w3schools.com/tags/tag_a.asp
[Kasutatud 24. veebruar 2016]

[229] Watson, J., 2015. *Classnames* [Võrgumaterjal]
https://jedwatson.github.io/classnames/
[Kasutatud 24. veebruar 2016]

[230] Facebook Inc., 2013-2016. *setState* [Võrgumaterjal]
https://facebook.github.io/react/docs/component-api.html#setstate
[Kasutatud 25. veebruar 2016]

[231] Facebook Inc., 2013-2016. *Reactive state* [Võrgumaterjal]
https://facebook.github.io/react/docs/tutorial.html#reactive-state
[Kasutatud 25. veebruar 2016]

[232] Mozilla Developer Network ja individuaalsed kaastöötajad, *Event.target* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/API/Event/target
[Kasutatud 25. veebruar 2016]

[233] Facebook Inc., 2013-2016. *Avoiding reconciling the DOM* [Võrgumaterjal]
https://facebook.github.io/react/docs/advanced-performance.html#avoiding-reconciling-the-dom
[Kasutatud 25. veebruar 2016]

[234] Facebook Inc., 2013-2016. *Updating: shouldComponentUpdate* [Võrgumaterjal]
https://facebook.github.io/react/docs/component-specs.html#updating-shouldcomponentupdate
[Kasutatud 25. veebruar 2016]

[235] Facebook Inc., 2013-2016. *Updating: componentDidUpdate* [Võrgumaterjal]
https://facebook.github.io/react/docs/component-specs.html#updating-componentdidupdate
[Kasutatud 25. veebruar 2016]

[236] Facebook Inc., 2013-2016. *Interactive Props* [Võrgumaterjal]
https://facebook.github.io/react/docs/forms.html#interactive-props
[Kasutatud 25. veebruar 2016]

[237] Facebook Inc., 2013-2016. *The ref String Attribute* [Võrgumaterjal]
https://facebook.github.io/react/docs/more-about-refs.html#the-ref-string-attribute
[Kasutatud 25. veebruar 2016]

[238] Facebook Inc., 2013-2016. *Focus Events* [Võrgumaterjal]
https://facebook.github.io/react/docs/events.html#focus-events
[Kasutatud 25. veebruar 2016]

[239] Facebook Inc., 2013-2016. *Keyboard Events* [Võrgumaterjal]
https://facebook.github.io/react/docs/events.html#keyboard-events
[Kasutatud 25. veebruar 2016]

[240] Facebook Inc., 2013-2016. *Mounting: componentDidMount* [Võrgumaterjal]
https://facebook.github.io/react/docs/component-specs.html#mounting-componentdidmount
[Kasutatud 25. veebruar 2016]

[241] Mozilla Developer Network ja individuaalsed kaastöötajad, *Function.prototype.bind()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/bind
[Kasutatud 25. veebruar 2016]

[242] GitHub Inc & flatiron. 2015. *flatiron/director* [Võrgumaterjal]
https://github.com/flatiron/director
[Kasutatud 25. veebruar 2016]

[243] GitHub Inc & flatiron. 2015. *init([redirect])* [Võrgumaterjal]
https://github.com/flatiron/director#initredirect
[Kasutatud 25. veebruar 2016]

[244] Mozilla Developer Network ja individuaalsed kaastöötajad, *Array.prototype.reduce()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce
[Kasutatud 25. veebruar 2016]

[245] Cravens, J., Brady, T., Q., 2014. *Building Web Apps with Ember.js*, Sebastopol, CA: O'Reilly Media

[246] Tilde Inc., 2016. *See Who's Using Ember.js* [Võrgumaterjal]
http://emberjs.com/ember-users/
[Kasutatud 17. veebruar 2016]

[247] Tilde Inc., 2016. *Ember.js* [Võrgumaterjal]
http://emberjs.com
[Kasutatud 17. veebruar 2016]

[248] npm, Inc. *ember-cli* [Võrgumaterjal]
https://www.npmjs.com/package/ember-cli
[Kasutatud 17. veebruar 2016]

[249] Github, Inc. & ember-cli., 2015. *v1.13.12 - Windows - The browser loading indicator keeps spinning #5123* [Võrgumaterjal]
https://github.com/ember-cli/ember-cli/issues/5123
[Kasutatud 17. veebruar 2016]

[250] Stack Exchange Inc. & Rice, D., 2014. *Disable / turn off LiveReload server in Emberjs / Ember-cli. Answered David Rice* [Võrgumaterjal]
https://stackoverflow.com/questions/25026796/disable-turn-off-livereload-server-in-emberjs-ember-cli
[Kasutatud 17. veebruar 2016]

[251] Tilde Inc., 2016. *Handlebars Basics* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/templates/handlebars-basics/
[Kasutatud 17. veebruar 2016]

[252] GitHub Inc & emberjs 2016. *emberjs/ember.js* [Võrgumaterjal]
https://github.com/emberjs/ember.js

[Kasutatud 17. veebruar 2016]

[253] Stack Exchange Inc, 2016. *Tagged Questions* [Võrgumaterjal]
https://stackoverflow.com/questions/tagged/ember.js
[Kasutatud 17. veebruar 2016]

[254] Stack Exchange Inc, 2016. *Unanswered Questions* [Võrgumaterjal]
https://stackoverflow.com/unanswered/tagged/ember.js
[Kasutatud 17. veebruar 2016]

[255] Tilde Inc., 2016. *Guides and Tutorials* [Võrgumaterjal]
https://guides.emberjs.com/v2.3.0/
[Kasutatud 17. veebruar 2016]

[256] Jetbrains. *WebStorm Plugins. Ember.js* [Võrgumaterjal]
https://plugins.jetbrains.com/plugin/8049?pr=webStorm
[Kasutatud 18. veebruar 2016]

[257] Oracle Corporation, 2015. *Bug 252281 - EmberJS support* [Võrgumaterjal]
https://netbeans.org/bugzilla/show_bug.cgi?id=252281
[Kasutatud 18. veebruar 2016]

[258] noklesta, 2012. *Simple Ember.js Navigator* [Võrgumaterjal]
https://packagecontrol.io/packages/Simple%20Ember.js%20Navigator
[Kasutatud 19. veebruar 2016]

[259] fabriciotav, 2014. *Ember.js snippets for Sublime Text 2* [Võrgumaterjal]
https://packagecontrol.io/packages/Ember.js%20Snippets
[Kasutatud 19. veebruar 2016]

[260] Tilde Inc., 2016. *Unit Testing Basics* [Võrgumaterjal]
https://guides.emberjs.com/v2.3.0/testing/unit-testing-basics/
[Kasutatud 23. veebruar 2016]

[261] Tilde Inc., 2016. *Introduction* [Võrgumaterjal]
https://guides.emberjs.com/v2.3.0/testing/
[Kasutatud 23. veebruar 2016]

[262] GitHub Inc & emberjs 2016. *emberjs/ember.js/LICENSE* [Võrgumaterjal]
https://github.com/emberjs/ember.js/blob/master/LICENSE
[Kasutatud 18. veebruar 2016]

[263] GitHub Inc & Rosen, J., A., 2016. *Example: Fetching Translations Live* [Võrgumaterjal]
https://github.com/jamesarosen/ember-i18n/wiki/Example:-Fetching-Translations-Live
[Kasutatud 23. veebruar 2016]

[264] GitHub Inc & van Mourik, J., 2016. *jcbvm/ember-i18n-editor* [Võrgumaterjal]
https://github.com/jcbvm/ember-i18n-editor
[Kasutatud 23. veebruar 2016]

[265] GitHub Inc & Mitchell, J. *Ember Intl* [Võrgumaterjal]
https://github.com/jasonmit/ember-intl#format-message
[Kasutatud 8. mai 2016]

[266] Yahoo! Inc., 2014. *Handlebars Intl* [Võrgumaterjal]
http://formatjs.io/handlebars/#formatMessage
[Kasutatud 23. veebruar 2016]

[267] Tilde Inc., 2016. *Replacing the Fixture Adapter with Another Adapter* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/getting-started/using-other-adapters/
[Kasutatud 28. veebruar 2016]

[268] GitHub Inc & locks. 2015. *Local Storage Namespace* [Võrgumaterjal]
https://github.com/locks/ember-localstorage-adapter#local-storage-namespace
[Kasutatud 28. veebruar 2016]

[269] Tilde Inc., 2016. *Classes and Instances* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/object-model/classes-and-instances/
[Kasutatud 28. veebruar 2016]

[270] Tilde Inc., 2016. *DS.Model Class* [Võrgumaterjal]
http://emberjs.com/api/data/classes/DS.Model.html
[Kasutatud 28. veebruar 2016]

[271] Tilde Inc., 2016. *attributes {Ember.Map}* [Võrgumaterjal]
http://emberjs.com/api/data/classes/DS.Model.html#property_attributes
[Kasutatud 28. veebruar 2016]

[272] Tilde Inc., 2016. *Focusing a Textfield after It's Been Inserted* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/cookbook/user_interface_and_interaction/focusing_a_textfield_after_its_been_inserted/
[Kasutatud 28. veebruar 2016]

[273] Tilde Inc., 2016. *Writing Helpers* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/templates/writing-helpers/
[Kasutatud 28. veebruar 2016]

[274] Tilde Inc., 2016. *Ember.Inflector Class* [Võrgumaterjal]
http://emberjs.com/api/data/classes/Ember.Inflector.html
[Kasutatud 28. veebruar 2016]

[275] Tilde Inc., 2016. *Representing Multiple Models with ArrayController* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/controllers/representing-multiple-models-with-arraycontroller/
[Kasutatud 28. veebruar 2016]

[276] Tilde Inc., 2016. *ArrayController* [Võrgumaterjal]
http://emberjs.com/deprecations/v1.x/#toc_arraycontroller
[Kasutatud 28. veebruar 2016]

[277] Tilde Inc., 2016. *Managing Dependencies Between Controllers* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/controllers/dependencies-between-controllers/
[Kasutatud 28. veebruar 2016]

[278] Tilde Inc., 2014. *needs Array* [Võrgumaterjal]
http://www.ember-doc.com/classes/Ember.ArrayController.html#property_needs
[Kasutatud 28. veebruar 2016]

[279] Tilde Inc., 2016. *Controller.needs* [Võrgumaterjal]
http://emberjs.com/deprecations/v1.x/#toc_controller-needs
[Kasutatud 28. veebruar 2016]

[280] Tilde Inc., 2016. *alias (dependentKey)* [Võrgumaterjal]

http://emberjs.com/api/classes/Ember.computed.html#method_alias
[Kasutatud 28. veebruar 2016]

[281] Tilde Inc., 2014. *itemController String | Ember.Controller* [Võrgumaterjal]
http://www.ember-doc.com/classes/Ember.ArrayController.html#property_itemController
[Kasutatud 28. veebruar 2016]

[282] Tilde Inc., 2016. *property* [Võrgumaterjal]
http://emberjs.com/api/classes/Function.html#method_property
[Kasutatud 28. veebruar 2016]

[283] Tilde Inc., 2016. *Computed Properties and Aggregate Data with @each* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/object-model/computed-properties-and-aggregate-data/
[Kasutatud 28. veebruar 2016]

[284] Tilde Inc., 2016. *isAny (key, value)* [Võrgumaterjal]
http://emberjs.com/api/classes/Ember.Enumerable.html#method_isAny
[Kasutatud 28. veebruar 2016]

[285] Tilde Inc., 2016. *Representing a Single Model with ObjectController* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/controllers/representing-a-single-model-with-objectcontroller/
[Kasutatud 28. veebruar 2016]

[286] Tilde Inc., 2016. *ObjectController* [Võrgumaterjal]
http://emberjs.com/deprecations/v1.x/#toc_objectcontroller
[Kasutatud 28. veebruar 2016]

[287] Tilde Inc., 2016. *oneWay (dependentKey)* [Võrgumaterjal]
http://emberjs.com/api/classes/Ember.computed.html#method_oneWay
[Kasutatud 28. veebruar 2016]

[288] Tilde Inc., 2016. *Actions* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/templates/actions/
[Kasutatud 28. veebruar 2016]

[289] Tilde Inc., 2016. *debounce (target, method, args*, wait, immediate) Array* [Võrgumaterjal]
http://emberjs.com/api/classes/Ember.run.html#method_debounce
[Kasutatud 28. veebruar 2016]

[290] Tilde Inc., 2016. *save (options)* [Võrgumaterjal]
http://emberjs.com/api/data/classes/DS.Model.html#method_save
[Kasutatud 28. veebruar 2016]

[291] Tilde Inc., 2016. *Deleting Records* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/models/creating-and-deleting-records/#toc_deleting-records
[Kasutatud 28. veebruar 2016]

[292] Tilde Inc., 2016. *Observers* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/object-model/observers/
[Kasutatud 29. veebruar 2016]

[293] Tilde Inc., 2016. *DS.Store Class* [Võrgumaterjal]
http://emberjs.com/api/data/classes/DS.Store.html
[Kasutatud 29. veebruar 2016]

[294] Tilde Inc., 2016. *invoke (methodName, args)* [Võrgumaterjal]
http://emberjs.com/api/classes/Ember.Array.html#method_invoke
[Kasutatud 29. veebruar 2016]

[295] Tilde Inc., 2016. *filterBy (dependentKey, propertyKey, value)* [Võrgumaterjal]
http://emberjs.com/api/classes/Ember.computed.html#method_filterBy
[Kasutatud 29. veebruar 2016]

[296] Tilde Inc., 2016. *Resources* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/routing/defining-your-routes/#toc_resources
[Kasutatud 29. veebruar 2016]

[297] Tilde Inc., 2016. *Displaying Model Data* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/getting-started/displaying-model-data/
[Kasutatud 29. veebruar 2016]

[298] Tilde Inc., 2016. *The {{link-to}} Helper* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/templates/links/
[Kasutatud 29. veebruar 2016]

[299] Tilde Inc., 2016. *Conditionals* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/templates/conditionals/
[Kasutatud 29. veebruar 2016]

[300] Tilde Inc., 2016. *Displaying a List of Items* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/templates/displaying-a-list-of-items/
[Kasutatud 29. veebruar 2016]

[301] Tilde Inc., 2016. *Ember.TextSupport Class* [Võrgumaterjal]
http://emberjs.com/api/classes/Ember.TextSupport.html
[Kasutatud 29. veebruar 2016]

[302] Tilde Inc., 2016. *Binding Element Attributes* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/templates/binding-element-attributes/
[Kasutatud 29. veebruar 2016]

[303] Tilde Inc., 2016. *The Application Template* [Võrgumaterjal]
https://guides.emberjs.com/v1.10.0/templates/the-application-template/
[Kasutatud 29. veebruar 2016]

[304] knockoutjs.com. *Knockout.* [Võrgumaterjal]
http://knockoutjs.com/index.html
[Kasutatud 2. märts 2016]

[305] BMW of North America. *BMW USA.* [Võrgumaterjal]
http://www.bmwusa.com/
[Kasutatud 2. märts 2016]

[306] Eventim. *Eventim. Tickets for Concerts, Festivals, Theatre, Sports & Attractions.* [Võrgumaterjal]
https://www.eventim.co.uk/
[Kasutatud 2. märts 2016]

[307] TD Ameritrade, Inc. *Ameritrade.* [Võrgumaterjal]

https://www.tdameritrade.com/home.page
[Kasutatud 2. märts 2016]

[308] knockoutjs.com. *Downloads.* [Võrgumaterjal]
http://knockoutjs.com/downloads/index.html
[Kasutatud 2. märts 2016]

[309] knockoutjs.com. *Hello World example.* [Võrgumaterjal]
http://knockoutjs.com/examples/helloWorld.html
[Kasutatud 2. märts 2016]

[310] GitHub Inc & knockout, 2016. *knockout/knockout* [Võrgumaterjal]
https://github.com/knockout/knockout
[Kasutatud 3. märts 2016]

[311] Stack Exchange Inc, 2016. *Tagged Questions* [Võrgumaterjal]
https://stackoverflow.com/questions/tagged/knockout.js
[Kasutatud 3. märts 2016]

[312] Stack Exchange Inc, 2016. *Unanswered Questions* [Võrgumaterjal]
https://stackoverflow.com/unanswered/tagged/knockout.js
[Kasutatud 3. märts 2016]

[313] knockoutjs.com. *Introduction.* [Võrgumaterjal]
http://knockoutjs.com/documentation/introduction.html
[Kasutatud 3. märts 2016]

[314] aaronpowell, 2012. *Sublime-KnockoutJS-Snippets* [Võrgumaterjal]
https://packagecontrol.io/packages/Sublime-KnockoutJS-Snippets
[Kasutatud 5. märts 2016]

[315] Oracle Corporation, 2015. *Knockout Client Generator - plugin detail* [Võrgumaterjal]
http://plugins.netbeans.org/plugin/55971/knockout-client-generator
[Kasutatud 5. märts 2016]

[316] Wielenga, G., 2014. *Knockout Client Generator for RESTful Web Services* [Võrgumaterjal]
https://blogs.oracle.com/geertjan/entry/knockout_client_generator_for_restful
[Kasutatud 5. märts 2016]

[317] Microsoft. *Client-side. Knockout.js* [Võrgumaterjal]
http://webtooling.visualstudio.com/frameworks/client-side/#knockout.js
[Kasutatud 18. veebruar 2016]

[318] Microsoft, 2016. *Using IntelliSense* [Võrgumaterjal]
https://msdn.microsoft.com/en-us/library/hcw1s69b.aspx
[Kasutatud 5. märts 2016]

[319] Messora, R., 2014. *Web App Testing Using Knockout.JS* Birmingham: Packt Publishing Ltd.

[320] Breck-McKye, J., 2015. *Testing Knockout.js Web Applications* [Võrgumaterjal]
http://www.breck-mckye.com/blog/2015/02/testing-knockout-dot-js-web-applications/
[Kasutatud 5. märts 2016]

[321] GitHub Inc & knockout 2016. *knockout/LICENSE* [Võrgumaterjal]
https://github.com/knockout/knockout/blob/master/LICENSE

[Kasutatud 4. märts 2016]

[322] npm, Inc. *i18next-ko* [Võrgumaterjal]
https://www.npmjs.com/package/i18next-ko
[Kasutatud 5. märts 2016]

[323] Moment.js. *Moment.js* [Võrgumaterjal]
http://momentjs.com/
[Kasutatud 5. märts 2016]

[324] knockoutjs.com. *Creating custom bindings* [Võrgumaterjal]
http://knockoutjs.com/documentation/custom-bindings.html
[Kasutatud 9. mai 2016]

[325] Mozilla Developer Network ja individuaalsed kaastöötajad, *keyup* [Võrgumaterjal]

https://developer.mozilla.org/en-US/docs/Web/Events/keyup
[Kasutatud 6. märts 2016]

[326] Sanderson, S. *Knockout JavaScript library v3.2.0.* [Võrgumaterjal]
http://knockoutjs.com/downloads/knockout-3.2.0.debug.js
[Kasutatud 6. märts 2016]

[327] knockoutjs.com. *Observables.* [Võrgumaterjal]
http://knockoutjs.com/documentation/observables.html
[Kasutatud 6. märts 2016]

[328] knockoutjs.com. *Observable Arrays.* [Võrgumaterjal]
http://knockoutjs.com/documentation/observableArrays.html
[Kasutatud 6. märts 2016]

[329] Mozilla Developer Network ja individuaalsed kaastöötajad, *new operator* [Võrgumaterjal]

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/new
[Kasutatud 6. märts 2016]

[330] knockoutjs.com. *Computed Observables.* [Võrgumaterjal]
http://knockoutjs.com/documentation/computedObservables.html
[Kasutatud 6. märts 2016]

[331] knockoutjs.com. *Writable computed observables.* [Võrgumaterjal]
http://knockoutjs.com/documentation/computed-writable.html
[Kasutatud 6. märts 2016]

[332] knockoutjs.com. *Loading and Saving JSON data.* [Võrgumaterjal]
http://knockoutjs.com/documentation/json-data.html
[Kasutatud 6. märts 2016]

[333] knockoutjs.com. *Rate-limiting observable notifications.* [Võrgumaterjal]
http://knockoutjs.com/documentation/rateLimit-observable.html
[Kasutatud 6. märts 2016]

[334] knockoutjs.com. *The "visible" binding.* [Võrgumaterjal]
http://knockoutjs.com/documentation/visible-binding.html
[Kasutatud 6. märts 2016]

[335] knockoutjs.com. *The "foreach" binding.* [Võrgumaterjal]
http://knockoutjs.com/documentation/foreach-binding.html

[Kasutatud 6. märts 2016]

[336] knockoutjs.com. *The "css" binding.* [Võrgumaterjal]
http://knockoutjs.com/documentation/css-binding.html
[Kasutatud 7. märts 2016]

[337] knockoutjs.com. *Binding context.* [Võrgumaterjal]
http://knockoutjs.com/documentation/binding-context.html
[Kasutatud 7. märts 2016]

[338] The Dojo Foundation. *Dojo Toolkit 1.10.* [Võrgumaterjal]
https://dojotoolkit.org/
[Kasutatud 2. märts 2016]

[339] The Dojo Foundation. *A Brief History of Dojo.* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.7/quickstart/introduction/history.html
[Kasutatud 2. märts 2016]

[340] The Dojo Foundation. *Dojo Toolkit Reference Guide.* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/
[Kasutatud 9. mai 2016]

[341] The Dojo Foundation. *Dojo Toolkit Downloads.* [Võrgumaterjal]
http://download.dojotoolkit.org/
[Kasutatud 2. märts 2016]

[342] The Dojo Foundation. *Hello Dojo!.* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/hello_dojo/
[Kasutatud 2. märts 2016]

[343] GitHub Inc & dojo, 2016. *dojo/dojo* [Võrgumaterjal]
https://github.com/dojo/dojo
[Kasutatud 3. märts 2016]

[344] Stack Exchange Inc, 2016. *Tagged Questions* [Võrgumaterjal]
https://stackoverflow.com/questions/tagged/dojo
[Kasutatud 3. märts 2016]

[345] Stack Exchange Inc, 2016. *Unanswered Questions* [Võrgumaterjal]
https://stackoverflow.com/unanswered/tagged/dojo
[Kasutatud 3. märts 2016]

[346] The Dojo Foundation. *The Dojo Toolkit API.* [Võrgumaterjal]
https://dojotoolkit.org/api/
[Kasutatud 9. mai 2016]

[347] The Dojo Foundation. *Contributor License Agreement.* [Võrgumaterjal]
http://dojofoundation.org/about/cla
[Kasutatud 3. märts 2016]

[348] The Eclipse Foundation, 2016. *Eclipse Marketplace. Tern Eclipse IDE* [Võrgumaterjal]
https://marketplace.eclipse.org/content/tern-eclipse-ide
[Kasutatud 5. märts 2016]

[349] GitHub Inc & angelozerr, 2016. *Tern & DojoToolkit support* [Võrgumaterjal]
https://github.com/angelozerr/tern.java/wiki/Tern-&-DojoToolkit-support
[Kasutatud 5. märts 2016]

[350] Jetbrains. *WebStorm Plugins. AngularJS* [Võrgumaterjal]
https://plugins.jetbrains.com/plugin/7264?pr=webStorm
[Kasutatud 5. märts 2016]

[351] The Dojo Foundation. *Introduction to AMD Modules.* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/modules/
[Kasutatud 5. märts 2016]

[352] Russell, A., Higgins, P., Machi, D., Jurkiewicx, J., Balaam, A. *D.O.H.: Dojo Objective Harness.* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.9/util/doh.html
[Kasutatud 5. märts 2016]

[353] Rosen, L., E., The Dojo Foundation. *Dojo Toolkit License.* [Võrgumaterjal]
https://dojotoolkit.org/license.html
[Kasutatud 4. märts 2016]

[354] The Dojo Foundation. *Internationalization with the Dojo Toolkit.* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/i18n/
[Kasutatud 5. märts 2016]

[355] The Dojo Foundation. *Resource Bundles.* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/i18n/#resource-bundles
[Kasutatud 5. märts 2016]

[356] The Dojo Foundation. *Dates, Numbers and Currencies in the Dojo Toolkit.* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/i18n/#dates-numbers-and-currencies-in-the-dojo-toolkit
[Kasutatud 5. märts 2016]

[357] The Dojo Foundation. *Writing Your Own Widget.* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/quickstart/writingWidgets.html
[Kasutatud 16. märts 2016]

[358] W3Schools, 2015. *HTML5 Local Storage* [Võrgumaterjal]
http://www.w3schools.com/js/js_window.asp
[Kasutatud 13. märts 2016]

[359] The Dojo Foundation. *Configuration/Feature Detection.* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/loader/amd.html#configuration-feature-detection
[Kasutatud 13. märts 2016]

[360] The Dojo Foundation. *Configuration Reference.* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/loader/amd.html#configuration-reference
[Kasutatud 13. märts 2016]

[361] The Dojo Foundation. *Module Identifiers.* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/loader/amd.html#module-identifiers
[Kasutatud 15. märts 2016]

[362] The Dojo Foundation. *dojo/parser.* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/parser.html#dojo-parser
[Kasutatud 15. märts 2016]

[363] The Dojo Foundation. *Dijit Overview.* [Võrgumaterjal]

https://dojotoolkit.org/reference-guide/1.10/loader/amd.html#module-identifiers
[Kasutatud 15. märts 2016]

[364] The Dojo Foundation. *Loader Configuration.* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/dojo_config/#loader-configuration
[Kasutatud 15. märts 2016]

[365] The Dojo Foundation. *dijit/_TemplatedMixin.* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dijit/_TemplatedMixin.html
[Kasutatud 15. märts 2016]

[366] The Dojo Foundation. *dojo/cache (version 1.10).* [Võrgumaterjal]
https://dojotoolkit.org/api/?qs=1.10/dojo/cache
[Kasutatud 15. märts 2016]

[367] The Dojo Foundation. *Class Inheriting from Another Class.* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/declare/#class-inheriting-from-another-class
[Kasutatud 27. märts 2016]

[368] The Dojo Foundation. *hitch().* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/_base/lang.html#hitch
[Kasutatud 16. märts 2016]

[369] The Dojo Foundation. *this.inherited* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/declare/#this-inherited
[Kasutatud 26. märts 2016]

[370] The Dojo Foundation. *dijit/_WidgetBase.* [Võrgumaterjal]
https://dojotoolkit.org/api/?qs=1.10/dijit/_WidgetBase
[Kasutatud 16. märts 2016]

[371] The Dojo Foundation. *Class with Multiple Inheritance.* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/declare/#class-with-multiple-inheritance
[Kasutatud 16. märts 2016]

[372] The Dojo Foundation. *startup().* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/understanding_widgetbase/#startup-
[Kasutatud 16. märts 2016]

[373] The Dojo Foundation. *Owning handles.* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/understanding_widgetbase/#owning-handles
[Kasutatud 16. märts 2016]

[374] Thomas, J., Chatelain, E., Sudoh, A., 2015. *Dojo TodoMVC Example. computed.js* [Võrgumaterjal]
https://github.com/tastejs/todomvc/blob/master/examples/dojo/js/todo/computed.js
[Kasutatud 6. veebruar 2016]

[375] The Dojo Foundation. *dojo/Stateful* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/Stateful.html
[Kasutatud 19. märts 2016]

[376] The Dojo Foundation. *dojo/Deferred* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/Deferred.html

[Kasutatud 19. märts 2016]

[377] The Dojo Foundation. *dojo/when* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/when.html
[Kasutatud 19. märts 2016]

[378] Mozilla Developer Network ja individuaalsed kaastöötajad. *Promise* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
[Kasutatud 19. märts 2016]

[379] The Dojo Foundation. *dojo/promise* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/promise.html#dojo-promise
[Kasutatud 19. märts 2016]

[380] Mozilla Developer Network ja individuaalsed kaastöötajad. *Event.preventDefault()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/API/Event/preventDefault
[Kasutatud 21. märts 2016]

[381] The Dojo Foundation. *dojo/when* [Võrgumaterjal]
https://dojotoolkit.org/api/?qs=1.10/dojo/when
[Kasutatud 26. märts 2016]

[382] The Dojo Foundation. *dojox/mvc/getStateful* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojox/mvc/getStateful.html
[Kasutatud 25. märts 2016]

[383] The Dojo Foundation. *Data converter* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojox/mvc/at.html#data-converter
[Kasutatud 25. märts 2016]

[384] W3Schools, 2015. *HTML hidden Attribute* [Võrgumaterjal]
http://www.w3schools.com/tags/att_global_hidden.asp
[Kasutatud 25. märts 2016]

[385] The Dojo Foundation. *dojo/number* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/number.html
[Kasutatud 25. märts 2016]

[386] W3Schools, 1999-2016. *JavaScript Errors - Throw and Try to Catch* [Võrgumaterjal]
http://www.w3schools.com/js/js_errors.asp
[Kasutatud 26. märts 2016]

[387] The Dojo Foundation. *Lifecycle* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dijit/_WidgetBase.html#lifecycle
[Kasutatud 25. märts 2016]

[388] The Dojo Foundation. *dojox/mvc/StoreRefController* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojox/mvc/StoreRefController.html
[Kasutatud 25. märts 2016]

[389] The Dojo Foundation. *dojo/router* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/router.html
[Kasutatud 25. märts 2016]

[390] The Dojo Foundation. *mixin()* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/_base/lang.html#mixin
[Kasutatud 25. märts 2016]

[391] The Dojo Foundation. *dojox/mvc/at* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojox/mvc/at.html
[Kasutatud 25. märts 2016]

[392] The Dojo Foundation. *dojo/_base/lang. Features. Usage* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/_base/lang.html#id2
[Kasutatud 26. märts 2016]

[393] The Dojo Foundation. *dojo/_base/array* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/_base/array.html#dojo-base-array-foreach
[Kasutatud 26. märts 2016]

[394] The Dojo Foundation. *at function syntax with widgets* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojox/mvc/at.html#at-function-syntax-with-widgets
[Kasutatud 26. märts 2016]

[395] The Dojo Foundation. *dojox/mvc/_InlineTemplateMixin* [Võrgumaterjal]
https://dojotoolkit.org/api/?qs=1.10/dojox/mvc/_InlineTemplateMixin
[Kasutatud 26. märts 2016]

[396] The Dojo Foundation. *Instantiating Objects* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/declarative/#instantiating-objects
[Kasutatud 26. märts 2016]

[397] The Dojo Foundation. *dojox/mvc/Element* [Võrgumaterjal]
https://dojotoolkit.org/api/?qs=1.10/dojox/mvc/Element
[Kasutatud 26. märts 2016]

[398] The Dojo Foundation. *Event Attachments* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/templated/#event-attachments
[Kasutatud 26. märts 2016]

[399] Mozilla Developer Network ja individuaalsed kaastöötajad. *checkbox. checked* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL/checkbox#a-checked
[Kasutatud 26. märts 2016]

[400] The Dojo Foundation. *dojox/mvc/WidgetList* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojox/mvc/WidgetList.html
[Kasutatud 27. märts 2016]

[401] Thomas, J., Chatelain, E., Sudoh, A., 2015. *Dojo TodoMVC Example. CSSToggle-Widget.js* [Võrgumaterjal]
https://github.com/tastejs/todomvc/blob/master/examples/dojo/js/todo/widgets/CSSToggleWidget.js
[Kasutatud 6. veebruar 2016]

[402] The Dojo Foundation. *dojo/dom-class* [Võrgumaterjal]
https://dojotoolkit.org/api/?qs=1.10/dojo/dom-class
[Kasutatud 27. märts 2016]

[403] The Dojo Foundation. *dojo/has* [Võrgumaterjal]
https://dojotoolkit.org/api/?qs=1.10/dojo/has
[Kasutatud 27. märts 2016]

[404] Mozilla Developer Network ja individuaalsed kaastöötajad. *Array.prototype.findIndex()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex
[Kasutatud 27. märts 2016]

[405] The Dojo Foundation. *Class with No Inheritance* [Võrgumaterjal]
https://dojotoolkit.org/documentation/tutorials/1.10/declare/#class-with-no-inheritance
[Kasutatud 27. märts 2016]

[406] The Dojo Foundation. *dojo/store/util/SimpleQueryEngine* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/store/util/SimpleQueryEngine.html
[Kasutatud 27. märts 2016]

[407] The Dojo Foundation. *dojo/_base/declare* [Võrgumaterjal]
https://dojotoolkit.org/api/?qs=1.10/dojo/_base/declare
[Kasutatud 28. märts 2016]

[408] Mozilla Developer Network ja individuaalsed kaastöötajad. *Property accessors* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Property_Accessors
[Kasutatud 28. märts 2016]

[409] Mozilla Developer Network ja individuaalsed kaastöötajad. *Error* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Error
[Kasutatud 28. märts 2016]

[410] The Dojo Foundation. *dojo/store/util/QueryResults* [Võrgumaterjal]
https://dojotoolkit.org/api/?qs=1.10/dojo/store/util/QueryResults
[Kasutatud 28. märts 2016]

[411] The Dojo Foundation. *dojo/store/util/SimpleQueryEngine* [Võrgumaterjal]
https://dojotoolkit.org/api/?qs=1.10/dojo/store/util/SimpleQueryEngine
[Kasutatud 28. märts 2016]

[412] Mozilla Developer Network ja individuaalsed kaastöötajad. *Object.is()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/is
[Kasutatud 28. märts 2016]

[413] Mozilla Developer Network ja individuaalsed kaastöötajad. *A model for understanding equality comparisons?* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Equality_comparisons_and_sameness#A_model_for_understanding_equality_comparisons
[Kasutatud 28. märts 2016]

[414] Mozilla Developer Network ja individuaalsed kaastöötajad. *Array.prototype* [Võrgumaterjal]

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/prototype
[Kasutatud 28. märts 2016]

[415] The Dojo Foundation. *watch()* [Võrgumaterjal]
https://dojotoolkit.org/reference-guide/1.10/dojo/Stateful.html#watch
[Kasutatud 28. märts 2016]

[416] The Dojo Foundation. *dojo/_base/array* [Võrgumaterjal]
https://dojotoolkit.org/api/?qs=1.10/dojo/_base/array
[Kasutatud 28. märts 2016]

[417] Mozilla Developer Network ja individuaalsed kaastöötajad. *Function.prototype.apply()* [Võrgumaterjal]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/apply
[Kasutatud 28. märts 2016]

Lisad

I. Keerukuse pingeread[13]

Tabel 20. Testrakenduse ridade arvu kategooria. Mida madalam väärtus, seda parem.

Raamistik	Loogiliste ridade arvu (*Logical LOC*) summa
1. Vue.js	89
2. CanJS	91
3. Knockback.js	100
4. KnockoutJS	107
5. TroopJS+RequireJS	118
6. Ember.js	119
7. Mithril	165
8. AngularJS	167
9. Backbone.js	167
10. Ampersand	218
11. MarionetteJS	231
12. React	278
13. Flight	339
14. Dojo	357
15. Polymer	4733

[13] Polymeri testrakenduse struktuur on teistsugune. Nimelt kasutab Polymer tööriista Polybuild, millega genereeritakse olulistest failidest üks HTML-fail ning üks JavaScripti fail elements.build.js. Selle JavaScripti faili põhjal saadi Polymeri hinnangud.

Tabel 21. Testrakenduse tsüklomaatilise keerukuse kategooria. Mida madalam väärtus, seda parem.

Raamistik	**Tsüklomaatiline keerukuse (*Cyclomatic complexity*) keskmine**
1. Ember.js	1, 5
2. TroopJS+RequireJS	2, 13
3. Ampersand	2, 57
4. Vue.js	2, 67
5. Flight	2, 92
6. MarionetteJS	3, 43
7. Backbone.js	3, 83
8. AngularJS	4
9. Knockback.js	4, 75
10. CanJS	5
11. Dojo	7
12. Mithril	7
13. React	8
14. KnockoutJS	9
15. Polymer	1058

Tabel 22. Testrakenduse hooldatavuse indeksi kategooria. Mida kõrgem väärtus, seda parem.

Raamistik	**Hooldatavuse indeks (*Maintainability index*)**
1. TroopJS+RequireJS	139, 56
2. AngularJS	137, 37
3. Ampersand	137, 30
4. Vue.js	135, 45

5. CanJS	134, 06
6. Dojo	133, 48
7. Flight	131, 60
8. Ember.js	130, 26
9. Backbone.js	130, 14
10. KnockoutJS	128, 56
11. MarionetteJS	128, 21
12. Knockback.js	126, 01
13. Mithril	122, 76
14. Polymer	120, 50
15. React	116, 96

Tabel 23. Testrakenduse Halstead'i jõupingutuste (*effort*) keskmine funktsioonide põhjal (*Mean per-function Halstead effort*). Mida madalam väärtus, seda parem.

Raamistik	**Halstead'i jõupingutuste (*effort*) keskmine funktsioonide põhjal**
1. TroopJS+RequireJS	249, 62495989552127
2. Vue.js	267, 18407272819553
3. AngularJS	285, 8976894724569
4. Ampersand	307, 8458693608369
5. CanJS	364, 33127860910196
6. Ember.js	440, 9969907187362
7. Backbone.js	462, 4561440259143
8. MarionetteJS	483, 9531462749237
9. Flight	533, 2322243945981
10. Knockback.js	574, 7877174304642
11. Dojo	615, 3148062854182

12. KnockoutJS	637, 7864270818525
13. Polymer	2392, 633471156666
14. React	2479, 73395828517
15. Mithril	2927, 3710015010024

Tabel 24. Knockback.js testrakenduse [28] ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
todos.js	7	100
models\todo.js	8	
app.js	53	
viewmodels\todo.js	32	

Tabel 25. Knockback.js testrakenduse [28] tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
todos.js	2	4, 75
models\todo.js	2	
app.js	7	
viewmodels\todo.js	8	

Tabel 26. Knockback.js testrakenduse [28] hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**
todos.js	128, 36561534424334	126, 01
models\todo.js	125, 98115363371775	
app.js	128, 04366362074163	
viewmodels\todo.js	121, 6655550502301	

Tabel 27. CanJS testrakenduse [29] ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
app.js	7	91
todo-app.js	59	
todo.js	25	

Tabel 28. CanJS testrakenduse [29] tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
app.js	1	5
todo-app.js	9	
todo.js	5	

Tabel 29. CanJS testrakenduse [29] hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**
app.js	140, 67691688822143	134, 06
todo-app.js	126, 22869454461653	
todo.js	135, 28828620539235	

Tabel 30. Polymer testrakenduse [30] ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
elements.build.js	4733	4733

Tabel 31. Polymer testrakenduse [30] tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
elements.build.js	1058	1058

Tabel 32. Polymer testrakenduse [30] hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**
elements.build.js	120, 49631488129386	120, 50

Tabel 33. Mithril testrakenduse [31] ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
app.js	8	165
controllers\todo.js	73	
storage.js	10	
models\todo.js	12	
footer-view.js	13	
main-view.js	49	

Tabel 34. Mithril testrakenduse [31] tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
app.js	2	7
controllers\todo.js	16	
storage.js	3	
models\todo.js	4	
footer-view.js	7	
main-view.js	10	

Tabel 35. Mithril testrakenduse [31] hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**

app.js	113, 31776246842512	122, 76
controllers\todo.js	121, 72768124964	
storage.js	143, 1907163034172	
models\todo.js	137, 85219657364303	
footer-view.js	100, 74509135519871	
main-view.js	119, 72646630077104	

Tabel 36. Ampersand testrakenduse [32] ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
app.js	13	218
router.js	7	
me.js	43	
models\todo.js	17	
todos.js	37	
main.js	47	
views\todo.js	54	

Tabel 37. Ampersand testrakenduse [32] tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
app.js	1	2, 57
router.js	2	
me.js	2	
models\todo.js	2	
todos.js	5	
main.js	2	

views\todo.js	4	

Tabel 38. Ampersand testrakenduse [32] hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**
app.js	119, 55001005308844	137, 30
router.js	157, 19825070922138	
me.js	130, 95832226898972	
models\todo.js	144, 40552433851875	
todos.js	138, 09372085346183	
main.js	138, 31436176438865	
views\todo.js	132, 56634922533922	

Tabel 39. Flight testrakenduse [33] ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
main.js	22	339
store.js	4	
utils.js	71	
data\stats.js	23	
todos.js	67	
app.js	12	
main_selector.js	10	
new_item.js	13	
ui\stats.js	19	
todo_list.js	70	
toggle_all.js	12	

with_filters.js	16	

Tabel 40. Flight testrakenduse [33] tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
main.js	1	2, 92
store.js	1	
utils.js	13	
data\stats.js	3	
todos.js	4	
app.js	1	
main_selector.js	1	
new_item.js	3	
ui\stats.js	1	
todo_list.js	5	
toggle_all.js	1	
with_filters.js	1	

Tabel 41. Flight testrakenduse [33] hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**
main.js	144, 2536598742244	131, 60
store.js	144, 9563904587855	
utils.js	109, 23612800805031	
data\stats.js	125, 65552694956597	
todos.js	122, 66717256030492	

app.js	126, 10376827329674	
main_selector.js	143, 6726088614988	
new_item.js	133, 6395744444554	
ui\stats.js	131, 11427409009497	
todo_list.js	119, 39049103790182	
toggle_all.js	143, 7615572963932	
with_filters.js	134, 71701152472974	

Tabel 42. Vue.js testrakenduse [34] ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
app.js	67	89
routes.js	13	
store.js	9	

Tabel 43. Vue.js testrakenduse [34] tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
app.js	5	2, 67
routes.js	1	
store.js	2	

Tabel 44. Vue.js testrakenduse [34] hooldatavuse indeksi kategooria.

Fail	**Hooldatavuse indeks (*Maintainability index*)**	**Keskmine**
app.js	130, 57041288782264	135, 45
routes.js	135, 58176584361667	
store.js	140, 20821062821972	

Tabel 45. MarionetteJS testrakenduse [35] ridade arvu kategooria.

Fail	Loogiliste ridade arv (*Logical LOC*)	Summa
TodoMVC.Application.js	10	231
TodoMVC.FilterState.js	8	
TodoMVC.Layout.js	64	
TodoMVC.Router.js	33	
TodoMVC.TodoList.Views.js	74	
TodoMVC.Todos.js	32	
TodoMVC.js	10	

Tabel 46. MarionetteJS testrakenduse [35] tsüklomaatilise keerukuse kategooria.

Fail	Tsüklomaatiline keerukus (*Cyclomatic complexity*)	Keskmine
TodoMVC.Application.js	2	3, 43
TodoMVC.FilterState.js	1	
TodoMVC.Layout.js	5	
TodoMVC.Router.js	3	
TodoMVC.TodoList.Views.js	6	
TodoMVC.Todos.js	5	
TodoMVC.js	2	

Tabel 47. MarionetteJS testrakenduse [35] hooldatavuse indeksi kategooria.

Fail	Hooldatavuse indeks (*Maintainability index*)	Keskmine
TodoMVC.Application.js	137, 9389817130193	128, 21
TodoMVC.FilterState.js	133, 42331711036684	

TodoMVC.Layout.js	118, 55499627171908	
TodoMVC.Router.js	128, 42053747295034	
TodoMVC.TodoList.Views.js	121, 16575758923356	
TodoMVC.Todos.js	129, 89100180866635	
TodoMVC.js	128, 04052956088196	

Tabel 48. TroopJS+Require.js [36] testrakenduse ridade arvu kategooria.

Fail	**Loogiliste ridade arv (*Logical LOC*)**	**Summa**
clear.js	9	118
component.js	3	
count.js	7	
create.js	12	
filter.js	7	
hide.js	5	
list.js	67	
toggle.js	8	

Tabel 49. TroopJS+Require.js [36] testrakenduse tsüklomaatilise keerukuse kategooria.

Fail	**Tsüklomaatiline keerukus (*Cyclomatic complexity*)**	**Keskmine**
clear.js	1	2, 13
component.js	1	
count.js	2	
create.js	3	
filter.js	2	
hide.js	1	

list.js	6	
toggle.js	1	

Tabel 50. TroopJS+Require.js testrakenduse [36] hooldatavuse indeksi kategooria.

Fail	Hooldatavuse indeks (*Maintainability index*)	Keskmine
clear.js	142, 5618771508886	139, 56
component.js	147, 09461000103389	
count.js	140, 14754001961006	
create.js	128, 90164184304484	
filter.js	141, 5651215897462	
hide.js	143, 18438232854987	
list.js	127, 04309389419693	
toggle.js	145, 96861885881415	

II. Node.js ja Deployd

Joonisel 177 on toodud raamatu „Pro AngularJS“ põhjal kirjutatud fail server.js, millega saab käivitada Node.js-põhist serverit [10, lk 9].

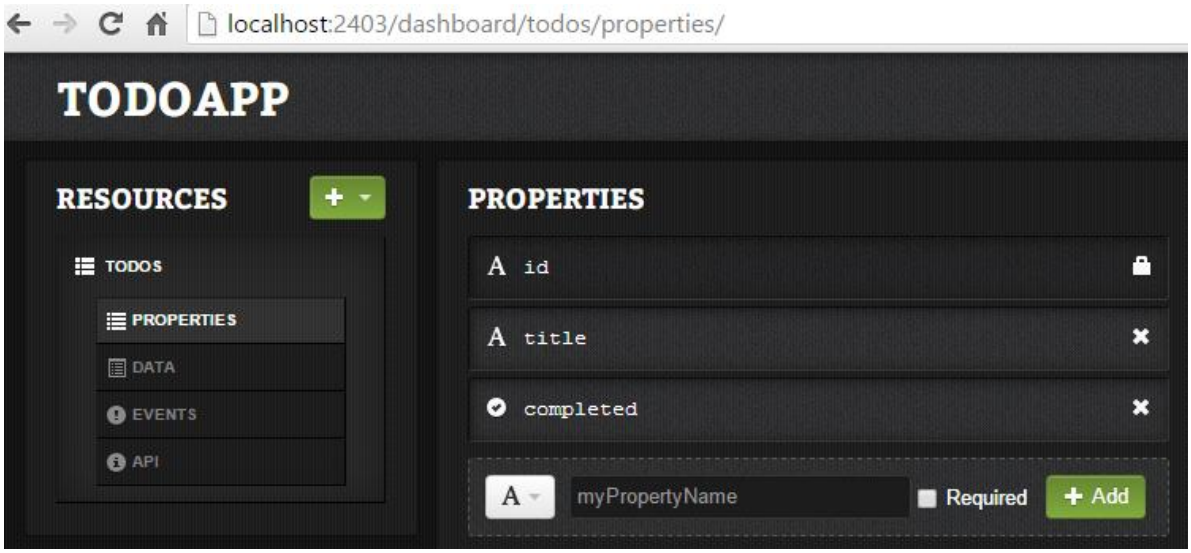
```
var connect = require('connect'),
    serveStatic = require('serve-static');

var app = connect();

app.use(serveStatic("C:/Users/Erik/Desktop/TÜ/VI semester/Bakalaureusetöö (MTAT.00.041)/8. Testrakendus/AngularJS/angularjs"));
app.listen(5000);
```

Joonis 177. server.js.

Joonisel 178 on toodud bakalaureusetöö autori poolt tehtud tegevuste andmestruktuuri haldamisleht Deployd‍is. Igal talletataval tegevusel on oma unikaalne ID, pealkiri ning tõeväärtusväli. Deployd‍i kohta andis hea ülevaate raamat „Pro AngularJS“ [10, lk 120–123].



Joonis 178. Deploydi haldamisleht.

III. TodoMVC projekti repositooriumi sisu kasutamise litsents [19]

Everything in this repo is MIT License unless otherwise specified.

Copyright (c) Addy Osmani, Sindre Sorhus, Pascal Hartig, Stephen Sawchuk.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

IV. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina **Erik Räni** (sünnikuupäev: 29.10.1993)
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
JavaScripti raamistike võrdlus,
(*lõputöö pealkiri*)

mille juhendaja on Vambola Leping,
(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **12.05.2016**