TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

Tarkvaraarendaja vahendid veebibrauserites

bakalaureusetöö

Üliõpilane: Marek Mänd
Üliõpilaskood: IAPB970835
Juhendaja: Enn Õunapuu

Tallinn
2014

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

....

(*kuupäev*) (*allkiri*)

Annotatsioon

Bakalaureusetöö eesmärgiks on koostada ajakohane eestikeelne hetkeülevaade kaasaja veebibrauseritesse integreeritud arendusvahenditest, mis on suunatud programmeerijatele veebirakenduste silumiseks ning veebirakenduste esitluskihi ehk kasutajaliidese disainimiseks.

Bakalaureusetöö seab eesmärgiks töötada välja soovitus uuritavate arendusvahendite hulgast sobivaima vahendi valikuks.

Töös uuritakse veebibrauseritesse integreeritud arendusvahendite erisusi ja samasusi, püstitatakse kriteeriumid, mille järgi hinnata arendusvahendi kasutamise sobivust ja otstarbekust.

Töö tulemuseks on eestikeelne avalikult kättesaadav teemasse sisse juhatav juhendmaterjal, mida saab kasutada abivahendina arendusvahenditega iseseisvalt tutvumiseks ning esitatud hinnangute põhjal soovitus, millist arendusvahendit uuritute hulgast kasutada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 3 peatükki, 9 joonist, 4 tabelit.

Abstract

The goal of this bachelor thesis is to produce in Estonian an overview of current developer tools built into web browsers that are targeted for programmers with the aim to debug web applications and to design the presentation layer – user interface.

The goal of this bachelor thesis is also to work out a recommendation for the optimal development tool selection.

The work researches the differences and similarities of the developer tools built into web browsers and estabilishes the criteria, by which the developer tools will be compared and judged by effectiveness and their ease of use.

As a result of this work an openly accessible guiding material in Estonian will be presented for a wider audience, which can be used for initial autodidactical schooling to acquaint the readers with the basics of these tools. Also a recommendation is made based on the assessment of criteria which development tool among those researched is the optimal one to be used in the selected programming environment.

The thesis is in Estonian and contains 33 pages of text, 3 chapters, 9 figures, 4 tables.

Lühendite ja mõistete sõnastik

Chrome	***Google Chrome***. Veebibrauser Google Chrome
CSS	***Cascading Style Sheets*** Kaskaadlaadistik. Deklaratiivne keel, millega kirjeldatakse veebilehtede sisu vormindus ning sisu paigutus
DOM	***Document Object Model*** Dokumentobjektmudel on arvutiplatvormist ja programmeerimiskeelest sõltumatu liides, mis sätestab kuidas programmid ja skriptid saavad lugeda ning ka muuta dokumendi struktuuri ja stiili
FF	***Firefox*** Veebibrauser Mozilla Firefox
HTML	***Hypertext Markup Language*** Märgendkeel veebilehtede sisu ja struktuuri üles märkimiseks
JavaScript	Programmeerimiskeel mida kasutatakse veebibrauserites veebilehe HTML sisu ning CSS stiilide muutmiseks DOM vahendusel
MSIE	***Microsoft Internet Explorer*** Veebibrauser Microsoft Internet Explorer
MSIE11	***Microsoft Internet Explorer 11*** Veebibrauser Microsoft Internet Explorer versioon 11
XPath	***XML Path Language*** Xpath defineerib metakeele, mis kirjeldab kuidas adresseerida XML (aga ka XHTML) dokumendis DOM sõlmi

Jooniste nimekiri

Tabelite nimekiri

Sisukord

1. Sissejuhatus

Veebitehnoloogiate arenedes oleme tänapäeval jõudnud ajastusse kus veebibrauseris kuvatav sisu ei ole lihtsalt staatiline triviaalne veebileht tekstide ja piltidega, milles tuleks tähelepanu pöörata ainult õigekirjavigadele. Tavakasutuses olevad arvutiprogrammid, mis on olnud tavapäraselt desktop-versioonidena kasutusel, on läbi teinud kontseptuaalse muudatuse – nad on kasutatavad ka veebibrauseri vahendusel. Tõuke selleks on andnud programmeerimiskeele JavaScript areng ning standardimine ECMAScript nimelisena, HTML märgendkeele uue standardi HTML5 välja töötamine, CSS spetsifikatsiooni täiustused ning veebibrausereid tootvate tarkvarafirmade jõupingutused välja töötatud standardeid enda toodetes kasutusele võtta. Veebilehe roll lihtsa infomaterjalina on ajas muutunud infomaterjalist veebirakenduse staatiliseks vaateks ning sealt edasi dünaamilise sisuga programmiks, millel on oma toimeloogika ning võimekus suhelda teiste veebilehtedega. Veebirakenduste kasutamise eeliseks võrreldes tavapäraste traditsiooniliste klient-server rakendustega infrastruktuurilisest vaatepunktist on nende versiooniuuendamise lihtsus. Uuendades rakenduse programmkoodi keskses serveris, saavad veebirakenduse kasutajad kasutada automaatselt uut programmiversiooni ilma, et oleks vaja läbi viia klientarvutitel töömahukaid ning rahakulukaid tarkvara paigaldamise operatsioone süsteemihaldurite poolt. Seetõttu on veebirakendused järjest enam populaarsust võitmas.

1.1 Taust ja probleem

Kuivõrd üha rohkem ja rohkem rakendusprogramme realiseeritakse veebibrauseris käitatavate programmidena, on kvaliteetsete rakenduste tootmiseks vajalik tunda seda liiki programmide arendus- ja silumisvahendeid. Eesti keeles informatsiooni nende arendusvahendite kohta aga napib ning teadlikkus on kesine. Samas aga vahendite keerukus ja võimekus ajas järjest kasvab, mille tõttu on nendega tutvumine järjest keerulisem. Autorile ei ole ka teada eesti keelset publikatsiooni, mis kõrvutaks ning võrdleks eri marki veebibrauseritesse integreeritud arendusvahendeid omavahel. Seda lünka püüab käesolev bakalaureusetöö täita.

1.2 Ülesande püstitus

Käesolev bakalaureusetöö seab endale eesmärgiks koostada eesti keelse sissejuhatava õppevahendina kasutatava juhendmaterjali nende töövahenditega iseseisvalt tutvumiseks ning ka nende vahendite põgusa võrdluse kasutusomaduste ning –võimaluste küljest.

Ülesande sisuks on veebibrauseritesse integreeritud arendusvahendite poolt pakutava funktsionaalsuse võrdlemise tulemusel püüda leida soovitus, millist arendusvahendit oleks kaasajal sobilik kasutada.

Iga arendusvahend jaguneb seesmiselt mitmeks alamaks mooduliks, millega on võimalik teha spetsiifilisi operatsioone. Võrdluste teostamiseks tuleb sama tööoperatsioone teostada võimaldavate moodulite kaupa välja töödata ja püstitada kriteeriumid, mille põhjal hinnata nende moodulite võimekust eri marki veebibrauserite lõikes.

1.3 Metoodika

Ülesandes püstitatud eesmärkide saavutamiseks analüüsin empiiriliselt hetkel enamlevinud kasutuses olevate brauserite uusimates versioonides sisalduvaid integreeritud arendusvahendeid. Subjektiivsete kriteeriumite puhul kasutan enda arvamust, olen olnud valdkonnas praktikuna tegev 2001. aastast; viimased 3,5 aastat olen tegelenud eKooli infosüsteemi arendamisega ning kasutanud brauseritesse integreeritud arendusvahendeid igapäevaselt.

Vaatluse alla võtan hetkel enamlevinud kasutuses olevate veebibrauserite uusimad versioonid:

1. Google Chrome 35.0.1916.114 m
2. Microsoft Internet Explorer 11.0.7
3. Mozilla Firefox 30.0

Lisaks vaadeldakse veel ülaltoodud Mozilla Firefoxi versiooni koos laiendusprogrammiga Firebug 2. Seega on võrdluses 4 erinevat tarkvarapaketti.

Vaatluse alt jääb välja Opera veebibrauser endale omaseks olnud Dragonfly nimelise arendusvahendiga kuivõrd Opera veebibrauserit arendav tarkvaraettevõte Operasoftware on loobunud oma firmaspetsiifilise brauserikomponendi arendamisest (1) ning on üle läinud

Blink nimelise brauserikomponendi kasutamisele – olles seetõttu samaväärne Chrome veebibrauseriga. Vaatluse alt jääb välja ka Safari veebibrauser enda arendusvahendiga kuivõrd tavakasutus on vähene jäädes 5% piiresse (2).

Võrdluste illustreerimiseks kasutan kuvatõmmiseid eri marki arendusvahenditest, näitan nende samasusi ja erinevusi, võrlden milline on võimekam ning arendajat abistavam.

1.4 Ülevaade tööst

Töö jaguneb kaheks osaks, millest esimeses (2. peatükk) uuritakse alapeatükkide kaupa sama sihtotstarbega arendusvahendeid veebibrauserisse integreeritud arendusvahendite paketis. Iga alapeatükk jaguneb alapeatükkideks, milles tutvustatakse vahendi kasutamise otstarvet, kirjeldatakse vahendi kasutajaliidest ning püstitatakse kriteeriumid vahendi sobivuse ja otstarbekuse hindamiseks. Igale vahendile antakse hinnang erinevat marki veebibrauserite lõikes.

Teises osas (Tulemused ja kokkuvõte) võetakse saadud tulemused kokku ning esitatakse tulemuste põhjal soovitus, millist marki brauserit on uuritavate brauserite hulgast arendusvahendina sobivaim kasutada.

2. Tarkvaraarendaja vahendite analüüs ning soovitused

Erinevat marki brauserid nimetavad endas käitatavate veebirakenduste silumisvahendeid erineva üldnimetusega, peamiselt kasutatakse inglise keeles mõistet „Developer Tools“ ehk „Tarkvaraarendaja vahendid“. *De facto* mõistena on argises kõnepruugis üldistavalt sünonüümina käibel ka Firebug, vaatamata sellele, et Firebug on Mozilla Firefox brauserisse paigaldatava laiendusprogrammi nimetus. Selle põhjuseks on Firebugi kui toote võimekus ja pikaealisus – Firebug loodi programmeerija Joe Hewitt'i poolt 2006. aasta jaanuaris ning oli üks esimesi selle laadsetest arendusvahenditest brauseris (3).

Üldiseks tavaks on kujunenud, et silumisvahenditele saab arendaja ligi vajutades aktiivses brauseri aknas klaviatuuril sõrmisele F12 või valides veebilehel mingi elemendi ning hiire teisesele (ehk tavaliselt paremale) klahvile vajutades kontekstimenüüst valikul „*Inspect element*“ või „*Inspect element with Firebug*“ (viimasel juhul kasutatakse Mozilla Firefoxi laiendusprogrammi Firebug).

Arendusvahendite põhifunktsionaalsusesse kuuluvad võimalused:

1. veebilehe käitamisel tekkinud vigade, hoiatuste ning info ja silumisteadete kuvamine
2. programmkoodi dünaamiliselt uute objektide, funktsioonide ja muutujate tekitamine ning juba olemasolevate ümberdefineerimine; olemasolevate meetodite ning funktsioonide väljakutsumine
3. veebilehe HTML struktuuri ning selle põhjal koostatud DOM-puu uurimine ning muutmine
4. veebilehte välimust kujundavate CSS-stiilideklaratsioonide uurimine ning muutmine
5. veebilehe välimust kujundavate CSS-stiilifailide sisu struktureeritud viisil kuvamine ning muutmine; stiilifailide toime kehtivuse ükshaaval välja lülitamise võimalus
6. veebibrauserist veebiserveritele tehtud päringute ning nendele päringutele saadud vastuste loetelu koos päringute tegemiseks kulunud ajainfoga ning mahuga baitides

7. veebilehe käitumist määratleva JavaScripti programmkoodi struktureeritud viisil kuvamine, programmkoodi silumine, täitmine reahaaval, katkestuspunktid (*breakpoints*), muutujate väärtuste jälgimine (*watches*)

Eri marki veebibrauserites on sama otstarbelistelt kasutatavate vahendite nimetused erinevad. Tabel 1 koondab endasse üle kõigi uuritavate veebibrauseri markide funktsionaalsuse lühikirjeldustele vastava mooduli nimetuse arendusvahendis.

Tabel 1

Samamõisteliste vahendite nimelised erisused veebibrauserites

	FF30 Firebug	**FF30**	**Chrome 35**	**MSIE11**
Teadete vaatamise ja käskude sisestamise vahend	Console	Web Console	JavaScript Console	Console
HTML struktuuri uurimise ja muutmise vahend	HTML section	Inspector	Elements panel	DOM Explorer
Stiilifaili tervikuna muutmise vahend	CSS section	Style Editor	Sources panel	vahend puudub
Võrguliikluse jälgimise vahend	Net section	Network	Network panel	Network
JavaScripti koodi silumise vahend	Script section	Debugger	Sources panel	Debugger

2.1 Arendusvahend: konsool

2.1.1 Kasutusotstarve

Konsooli esmaseks kasutusotstarbeks on võimaldada arendajal näha vea-, tõrke- ning hoiatusteateid, mis võivad tekkida mitte kvaliteetse veebilehe koodi käitamisel ja kuvamisel veebibrauseris.

Konsool koosneb kahest loogilisest osast, mille moodustavad:

1. veateadete, hoiatuste ja muude tulemuste kuvamise logiaken
2. konsooli käsurida – käskluste ning JavaScripti koodi sisestamise tekstisisestuselement

Konsooli teiseseks kasutusotstarbeks on võimaldada arendajal iseseisvalt kuvada ning logida diagnostilist informatsiooni mis tavakasutajale huvi ei paku. See hõlmab kunstlikult tekitatud veateateid, hoiatusi aga ka lihtsalt tehnilist infot, mida arendaja soovib programmeeritavast veebirakendusest endale teatavaks teha. Tüüpiliselt on selleks JavaScripti koodis muutujate väärtuste väljatrükk konsoolile või arendaja poolt seatud märgendteksti-tunnuse kuvamine konsoolil eesmärgiga teada saada, milliseid programmkoodi harusid tingimuslausetes kehtinud muutujate väärtuste puhul läbiti ja milliseid mitte. Arendaja jaoks on need konsoolile kirjutavad JavaScripti funktsioonid defineeritud *Console API* (4) raamistikus, peamiseks kasutatavaks logimisfunktsiooniks on *console.log()* meetod (5).

Konsooli kolmandaks kasutusotstarbeks on konsooli käsurida kasutades arendajal võimaldada käitada ja dünaamiliselt lisada veebilehele JavaScripti koodi, mida seal algselt ei olnud (6). See võimaldab arendajal:

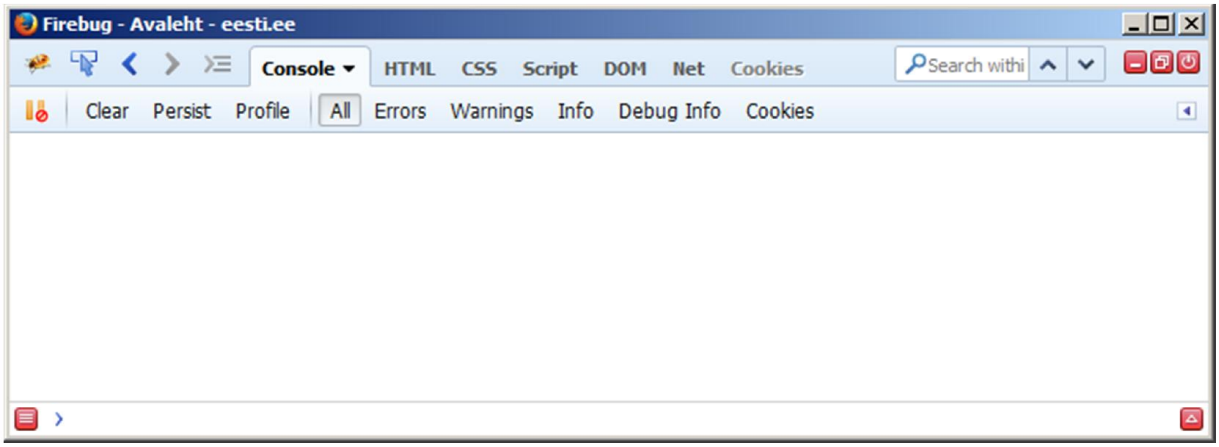
1. luua uusi muutujaid, funktsioone, objekte
2. muuta muutujate väärtusi, k.a. kustutada objektidest defineeritud andmevälju
3. defineerida ümber funktsioonide realisatsioone deklareerides uuesti samanimelise funktsiooni konsooli käsureal teistsuguse realisatsiooniga
4. kutsuda välja funktsioone ja meetodeid

Kolmas kasutusotstarve tagab arendajale mugava kiire võimaluse katsetada väikeseid programmijupikesi paindlikult ja koheselt koos parandusideedega, ilma et peaks arendatavat koodi kirjutama täielikult tekstiredaktorisse, faili salvestama, brauseri aknas veebilehe uuesti laadima. Eriti mugav on see keerulisemate veebirakenduste korral, mille puhul rakendusserveril juurutamiseks ei piisaks lihtsalt muudetava faili salvestamisest ja brauseriakna uuesti laadimisest, vaid mis eeldavad ka mitmeid eelnevaid vahesamme erinevate eelprotsessorite, kompilaatorite jms rakendamise näol.

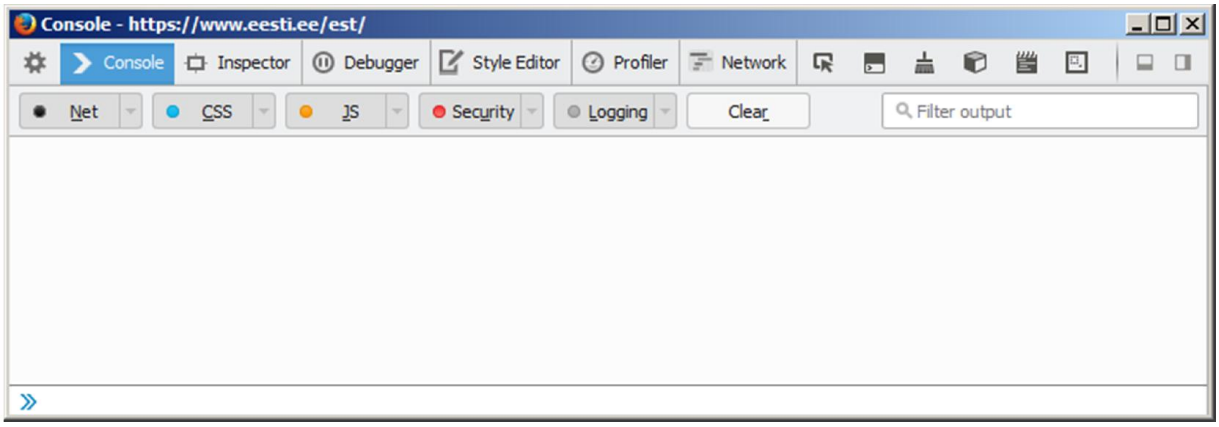
Konsooli neljandaks kasutusotstarbeks on võimalus konsooli käsurealt sisestada käsklusi, millega saab juhtida veebibrauserisse integreeritud arendusvahendi teiste moodulite tööd ja sooritada arendusvahendis tavapäraseid operatsioone, milleks elementide valik ja uurimine DOMist, profileerija ning siluri käivitamine või peatamine, DOMis tekkivate ja asetleidvate sündmuste jälgimine. Arendajale võimaldatavad käsud on kirjeldatud *Command Line APIs* (7) (8).

2.1.2 Tutvumine vahendi kasutajaliidesega ekraanipiltide vahendusel

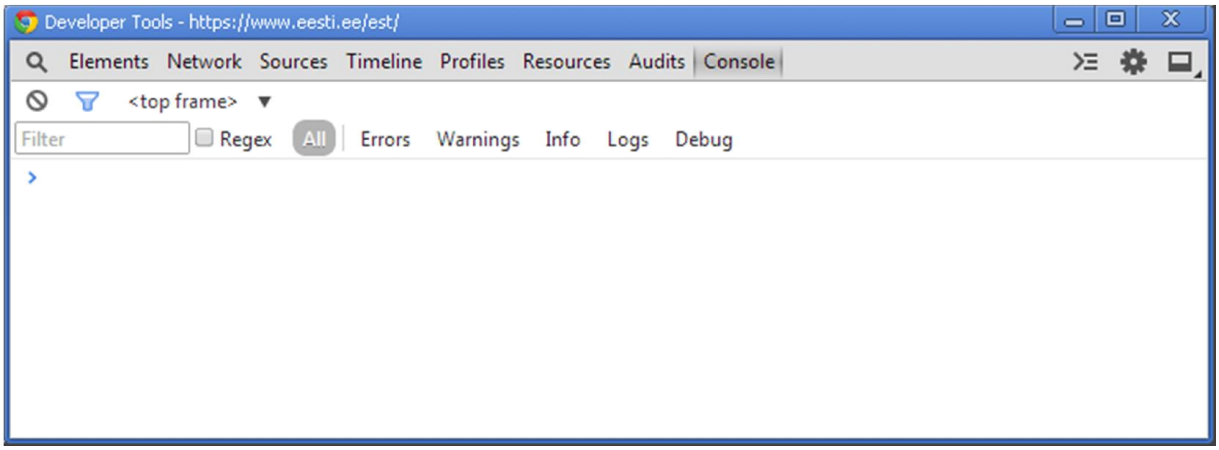
Alapeatükis 2.1.2 kujutatakse joonisel 1, joonisel 2, joonisel 3 ning joonisel 4 pildiliselt konsooli ekraanivorme eri marki veebibrauserites.



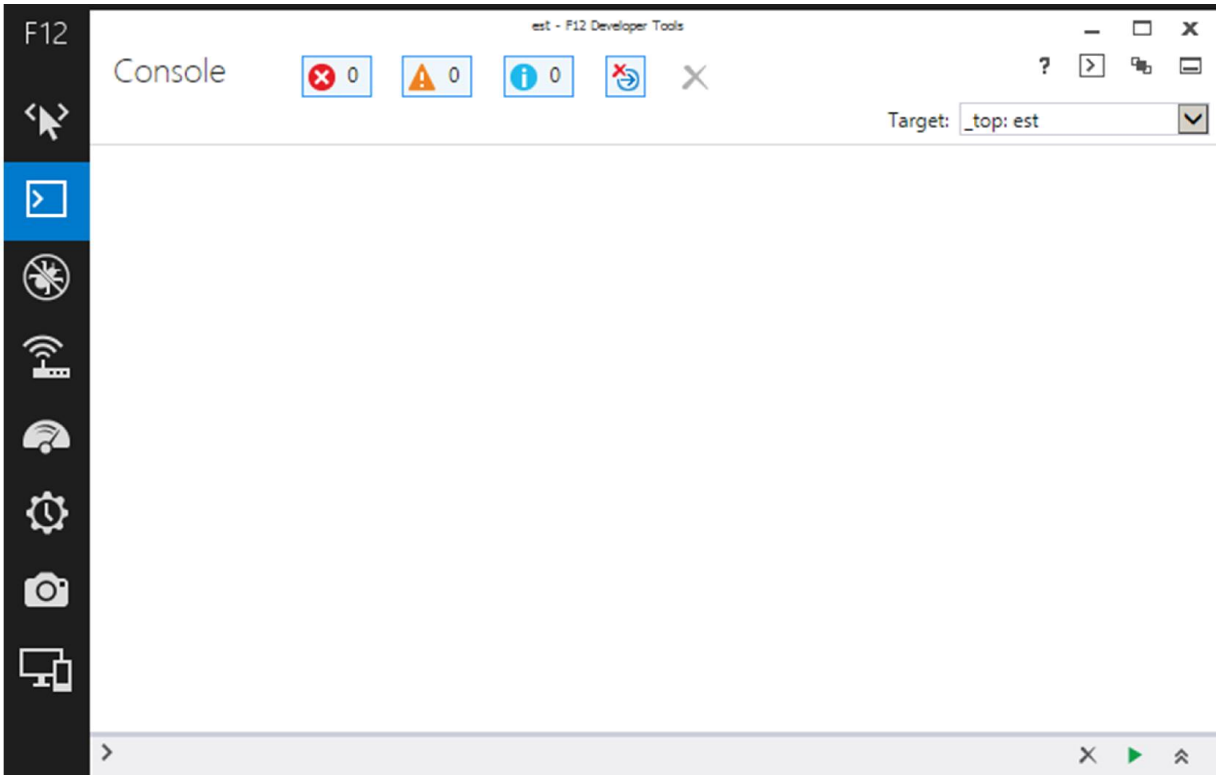
Joonis 1. Konsool veebibrauseris Mozilla Firefox koos lisaprogrammiga Firebug



Joonis 2. Konsool veebibrauseris Mozilla Firefox



Joonis 3. Konsool veebibrauseris Chrome



Joonis 4. Konsool veebibrauseris Internet Explorer 11

2.1.3 Konsoolile kirjutamiseks mõeldud funktsioonidega tutvumine näiteprogrammi abil

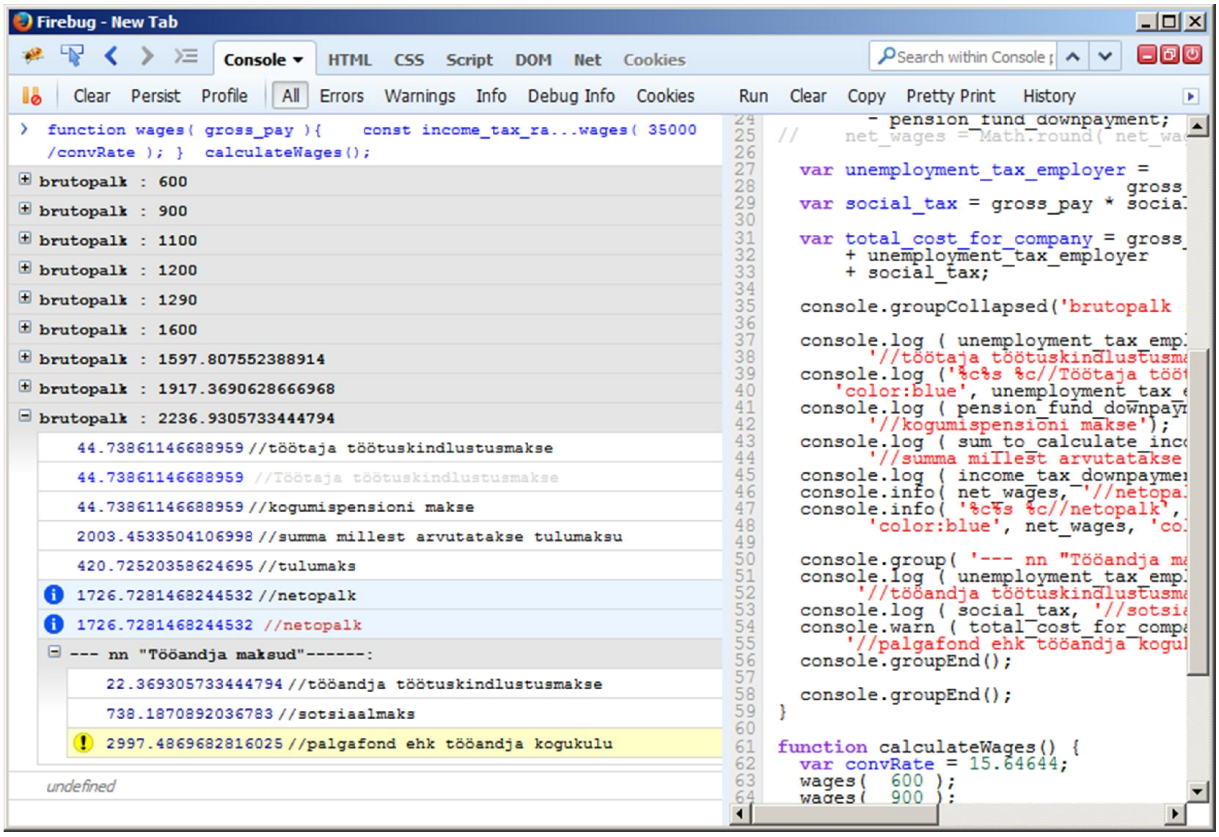
Konsool on ka praktiliseks vahendiks lihtsamate arvutuslike skriptide käitamiseks, mida muidu tavapäraselt kasutataks tabelarvutusprogrammides. Konsooli eelis seisnebki asjaolus, et veebibrauser on kaasajal paigaldatud enamasti iga tavakasutaja arvutisse, tabelarvutusprogramm nõuab aga eraldi programmipaketi paigaldamist. Konsooli väljundiga tutvumiseks kasutame ise loodud lihtsat JavaScripti näiteprogrammi, mis arvutab etteantud brutopalga põhjal netopalga, tulumaksu jms osised, võimaldades leida palju palka peaks töötaja küsima, et saada kätte töötajat rahuldav rahasumma. Kopeerides koodinäite 1 konsooli käsureale ning vajutades klaviatuuril Enterile (MSIE11s mitmerealises sisestusrežiimis tuleb vajutada CTRL+Enter) käivitatakse käsureale sisestatu.

```
function wages( gross_pay ){
    const income_tax_rate = 0.21;
    const unemployment_tax_employee_rate = 0.02; // 0.028
    const pension_fund_downpayment_rate = 0.02;
    const monthly_incometax_deductible_sum_for_employee = 144;
    const unemployment_tax_employer_rate = 0.01; // 0.014
    const social_tax_rate = 0.33;

    var unemployment_tax_employee =
                                  gross_pay * unemployment_tax_employee_rate;
    var pension_fund_downpayment  =
                                  gross_pay *  pension_fund_downpayment_rate;

    var sum_to_calculate_incometax_from = gross_pay -
                      unemployment_tax_employee - pension_fund_downpayment;
    sum_to_calculate_incometax_from -=
                               monthly_incometax_deductible_sum_for_employee;

    var income_tax_downpayment =
                          sum_to_calculate_incometax_from * income_tax_rate;
    var net_wages = gross_pay
                  - income_tax_downpayment
                  - unemployment_tax_employee
                  - pension_fund_downpayment;

//    net_wages = Math.round( net_wages );


    var unemployment_tax_employer =
                                  gross_pay * unemployment_tax_employer_rate;
    var social_tax = gross_pay * social_tax_rate;

    var total_cost_for_company = gross_pay
            + unemployment_tax_employer
            + social_tax;

    console.groupCollapsed('brutopalk : ' + gross_pay);

    console.log ( unemployment_tax_employee,
                  '//töötaja töötuskindlustusmakse');
    console.log ('%c%s %c//Töötaja töötuskindlustusmakse',
       'color:blue', unemployment_tax_employee, 'color:lightgray' );
    console.log ( pension_fund_downpayment,
                  '//kogumispensioni makse');
    console.log ( sum_to_calculate_incometax_from ,
                  '//summa millest arvutatakse tulumaksu');
    console.log ( income_tax_downpayment, '//tulumaks');
    console.info( net_wages, '//netopalk');
    console.info( '%c%s %c//netopalk',
                  'color:blue', net_wages, 'color:#ca312d');

    console.group( '--- nn "Tööandja maksud"------:');
    console.log ( unemployment_tax_employer,
             '//tööandja töötuskindlustusmakse');
    console.log ( social_tax, '//sotsiaalmaks');
    console.warn ( total_cost_for_company,
            '//palgafond ehk tööandja kogukulu');
    console.groupEnd();

    console.groupEnd();
}
```

```
function calculateWages() {
    var convRate = 15.64644;
    wages(  600 );
    wages(  900 );
    wages( 1100 );
    wages( 1200 );
    wages( 1290 );
    wages( 1600 );
    wages( 25000  /convRate);
    wages( 30000  /convRate );
    wages( 35000  /convRate );
}

calculateWages();
```

Koodinäide 1

Koodinäide 1 demonstreeris enamlevinud logimisfunktsioonide kasutamist. Programmi väljund on näidatud alljärgneval joonisel 5 ning kasutatud funktsioonide selgitused tuuakse ära tabelis 2.



Joonis 5. Koodinäite 1 käitamisel tekkinud väljund Firebugis

Tabel 2

Koodinäites 1 kasutatud *console* objekti meetodid

Meetodi signatuur	Meetodi funktsionaalsus
console.log(määramata arv eritüübilisi argumente)	kirjutab teate konsoolile
console.info(määramata arv eritüübilisi argumente)	kirjutab teate konsoolile ning tähistab selle informatiivse teate tüübilisena. Tähis paikneb tüüpiliselt ikoonina teate suhtes vasakul ning kujutab sinist ringi, mille sees on valkjas i-täht.
console.warn(määramata arv eritüübilisi argumente)	kirjutab teate konsoolile ning tähistab selle hoiatuse tüübilisena. Tähis paikneb tüüpiliselt ikoonina teate suhtes vasakul ning kujutab kollast kolmnurka, mille sees on hüüumärk.
console.group(*stringGrupiNimetus*)	teateid graafiliselt puukujuliseks struktuuriks kujundada võimaldav mugavusfunktsioon, mis aitab konsoolil kokku hoida ruumi ja tagab parema loetavusselguse. Meetodi väljakutse loob konsoolile uue grupeeriva alamtaseme, millesse paigutatakse edaspidine konsoolile suunatud väljundvoog. Alamtasemetel olevaid teateid saab peita ja nähtavale tuua ühekorraga, alamtaseme pealkirjaks kuvatakse konsoolil argumendi *stringGrupiNimetus* väärtus

Meetodi signatuur	Meetodi funktsionaalsus
console.groupCollapsed(*stringGrupiNimetus*)	sama, mis console.group() ainult loodava uue alamtaseme elemendid on algselt peidetud. Arendaja saab peidetud kanded hiireklõpsuga grupiviisiliselt nähtavale tuua
console.groupEnd(*stringGrupiNimetus*)	sulgeb loogiliselt grupi nimetusega *stringGrupiNimetus*. Järgmised konsoolile väljastavad laused console.log jt kuvavad enda teateid konsoolil puustruktuuris visuaalselt tase kõrgemal

2.1.4 Konsooliga töötamisel hinnatavad praktilised kriteeriumid

Enne brauserisse konsooli juurutamist ei olnud ajalooliselt arendajal palju valikuid silumisel huvi pakkuvate väärtuste kuvamiseks, seda tehti tavaliselt modaalse dialoogikastikesega (JavaScript *alert()* funktsiooniga, mille sulgemiseks tuleb vajutada dialoogivormil OK nuppu), mis peatas muuhulgas ka JavaScripti edasise täitmise kuniks dialoogiaken oli suletud (9). Ka JavaScripti vigadest teavitati kasutajat modaalse infoaknaga. See sobis praktikas seni kuni käitatavad skriptid ning sündmusprotseduurid olid sedavõrd lihtsad, et ei mõjutanud veebirakenduse toimeloogikat.

Konsool võimaldab aga testida rakendust vähem pealetükkival ja protsessi segaval viisil, enam ei ole vaja kasutada *alert()* funktsiooni, sest veateateid on võimalik suunata otse konsoolile. See hoiab palju aega kokku ning võimaldab testida veebirakendust sujuvamal viisil (10).

Samas kuigi konsool erinevat marki veebibrauserites näeb põhimõtteliselt sarnane välja, eristub selgelt konsooli kasutamise mugavus ja praktilisus arendaja vaatepunktist.

Peamiseks hindamiskriteeriumiks arendaja seisukohalt vaadatuna tuleb pidada konsooli käsurea mitmerealise sisestamise režiimi olemasolu ning võimekust. Mitmerealine sisestusrežiim võimaldab sisestada silmale loetavat trepitud blokkstruktuuriga JavaScripti koodi. Üherealise sisestusrežiimi eesmärgipärane kasutus piirdub siiski üldjuhul üksiku(te) lihtsate omistustehete või funktsioonide väljakutsetega, sest keerulisema struktuuriga ning liigendusega koodi tõlgendamine ning mõistmine oleks ühel tekstireal selle lugejale raskendatud.

Teiseseks hindamiskriteeriumiks arendaja seisukohalt vaadatuna on võimalus konsooli käsureale sisestatud programmkoodi käivitada vaid osalises mahus – ainult seda osa, mis on tekstikursori valikuga esile tõstetud. See võimaldab arendajal vältida koodis nende osade välja kommenteerimist kommentaari sümbolitega (/* */), milliseid koodiosi antud ajahetkel testimisel ei soovita käivitada. Selline võimekus hoiab kokku arendaja aega ning vähendab ka vigade tekkimise võimalust kuivõrd reaalselt kommentaaride sisestamise ning eemaldamisega koodi ei muudeta ning arendaja vaid tekstikursori valikuga juhib millised programmlaused koodis käivitatakse.

Tabelis 3 püstitan arendusvahendist sõltuvad kasutusomaduslikud kriteeriumid, mis mõjutavad arendaja efektiivsust vahendiga töötamisel.

Tabel 3

Konsooli omaduste võrdlus püstitavate kriteeriumite kaupa

Kriteerium	FF30 Firebug	FF30	Chrome 35	MSIE 11
Veebilehe laadimisel või veebilehelt teisele navigeerimisel võimalus valida kas konsoolil olevad kanded säilivad või kustutatakse	JAH (Persist nupp tööriistareal)	JAH (seadetest „Enable Persistent Logs“)	JAH (seadetest „Preserve log upon navigation“)	JAH („Clear on navigate enabled“ nupp tööriistareal)
Konsooli käsurealt mitmerealiselt JavaScripti koodi koostamise võimalus	JAH (vaja eraldi menüüst aktiveerida, aktiveerub automaatselt mitmerealise teksti kleepimisel lõikelaualt)	JAH (automaatne SHIFT+Enter)	JAH (automaatne SHIFT+Enter)	JAH (automaatne SHIFT +Enter)
Konsooli käsurealt ainult valitud tekstiosa käivitamine	JAH	EI	EI	EI
Konsooli väljundi filtreerimine sisestatava otsisõne põhjal	JAH (Search within Console panel)	JAH (Filter output)	JAH (Filter, toetab ka regulaaravaldisi)	EI
Konsooli käsureale sisestatud käskluste ajalugu ja menüüst uuesti välja kutsumise võimalus	JAH	JAH	JAH	JAH
Automaattekst s.t. konsooli käsureale osaliselt sisestatud fraasile olemasolevate objektide ja nende väljade nimetuste soovitamise funktsionaalsus	JAH	JAH	JAH	JAH
JavaScripti süntaksi esiletõstmine visuaalselt käsureal	JAH	EI	EI	EI
Command Line API	JAH	JAH	JAH	EI
XMLHttpRequest päringute logimine konsoolile	JAH (Console menüüst „Show XMLHttp Requests“)	JAH (Console tööriistaribalt nupp „Net“)	JAH (seadetest „Log XMLHttp Requests“)	EI

Vastavalt tabelis 3 leitud tulemustele järjestan veebibrauserite margid vastavalt nende arendusvahendis oleva konsooli võimekusele pingeritta:

1. Mozilla Firefox 30 Firebugiga
2. Google Chrome 35
3. Mozilla Firefox 30
4. Internet Explorer 11

Mozilla Firefoxi Firebugi esimesele kohale asetamine tulenes just konsooli mitmerealise sisestusrežiimi võimekuse ning konsooli sisestuselemendis oleva koodi valikulist täitmist toetavate funktsioonide ainukordsuse ees võrreldes teiste brauserite arendusvahenditega. Firebug on arendajat siinkohal toetavaim, pakkudes ainukesena uuritud arendusvahendite hulgast ka konsoolil JavaScripti koodi süntaksi esiletõstmist kasutades selleks erinevaid värve (*syntax highlighting*), mis on nähtav joonisel 5 parempoolses osas.

Google Chrome 35 teisele kohale asetamine tulenes temale järgnevate arendusvahendite konsoolide suhtes võimekast väljundi filtreerimise funktsionaalsusest. Uuritud veebibrauserite arendusvahendite hulgas oli Chrome ainukene, mis võimaldas konsoolil olevat väljundit filtreerida kasutades selleks vahendisse kasutaja poolt sisestatud regulaaravaldist Muud marki veebibrauserites oli võimalik teostada vaid stringiotsingut ning MSIE11s puudus otsing üldse.

Viimasele kohale asetasin MSIE11, sest selle veebibrauseri arendusvahend oli püstitatud kriteeriumite lõikes kõige nõrgema tulemusega.

Lähtuvalt MSIE11 viimasest kohast ja Chrome teisest kohast asetasin kolmandale kohale uuritutest seni veel positsioneerimata veebibrauseri – Mozilla Firefoxi.

2.2 Arendusvahend DOM Inspector/Explorer

2.2.1 Kasutusotstarve

Arendusvahend „DOM Inspector/Explorer“ (edaspidi lihtsamalt Inspector) on veebiarendaja poolt brauseris kõige enam kasutatud arendusvahend, sest selle vahendi abil on võimalik tutvuda veebilehe HTML lähtekoodiga ning seda muuta viisil, et tehtavad muudatused kajastuvad nähtavalt samaaegselt veebibrauseri aknas, mis võimaldab veebilehe infoarhitektuuri prototüüpida.

Inspector koosneb kahest loogilisest osast, mille moodustavad:

1. arendusvahendi kasutajaliideses kuvatav hierarhiline puustruktuur (DOM-vaade), mis kujutab endast HTML lähtekoodi põhjal genereeritud DOMi visuaalset esitust

2. infoala, milles kuvatakse teavet (atribuutide/omaduste väärtusi), mille sisu sõltub arendusvahendi kasutajapoolt hiirekursoriga tehtavast valikust puustrukuuri kujutavas kasutajaliidese elemendis

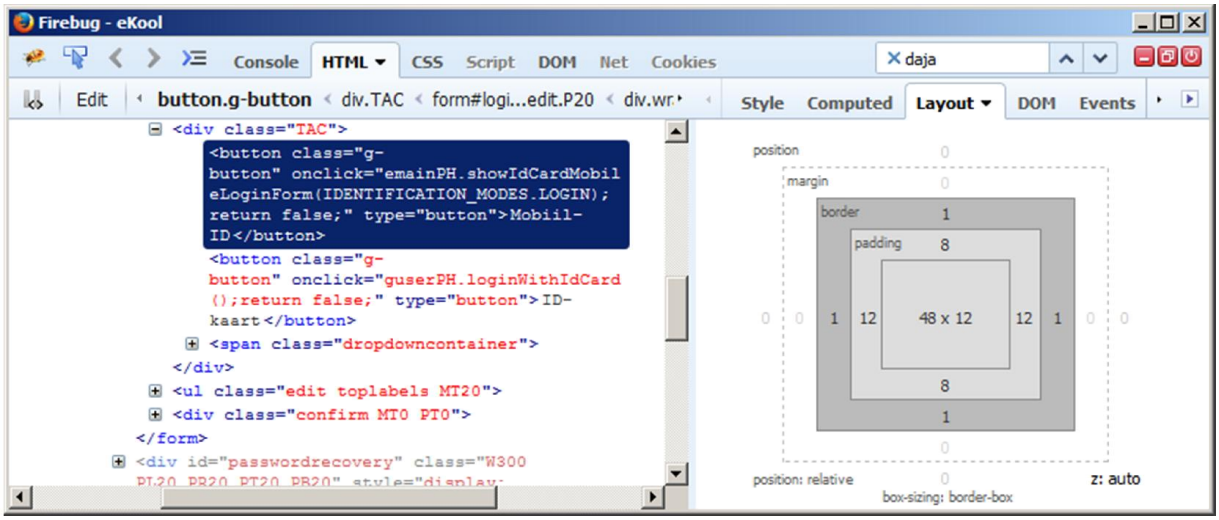
Selle kirjelduse avab paremini jooniste 6, 7, 8, 9 uurimine.

DOM-vaates on võimalik HTML elementide atribuutide väärtusi muuta, kustutada ning uusi atribuute luua. Olemasolevaid HTML elemente on võimalik kustutada, muuta HTML elemendi järjestust teise HTML elemendi suhtes, kopeerida HTML elementi sisu eesmärgiga seda HTML elementi DOMis paljundada. Kõige enam kasutatud funktsioon on aga HTML-elemendi redigeerimisrežiimis HTML elemendi sees olevate HTML märgendite muutmine.

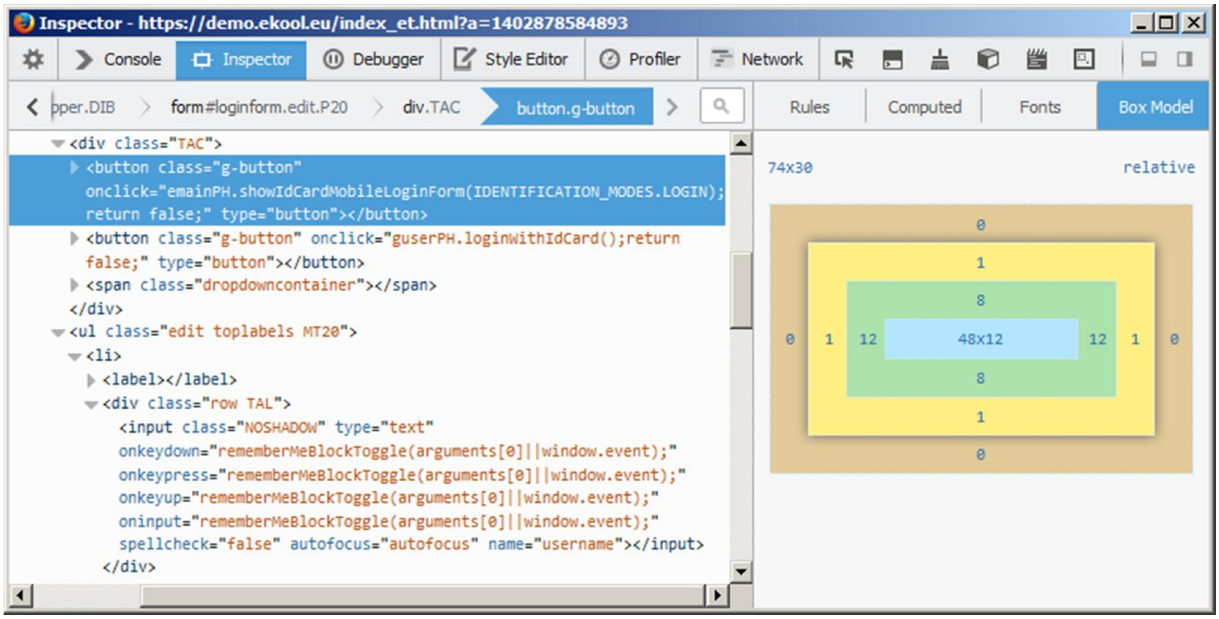
Inspectoris on võimalik prototüüpida ka veebilehe välimust ja elementide paigutust. Valides hiirekursoriga DOM puustruktuuri kujutavast kasutajaliidese komponendist HTML elemendi, on võimalik muuta Inspectori infoalas elemendi välimust (sh suurust, positsiooni, värvust, nähtavust) CSS stiilideklaratsioonide kustutamise, muutmise või lisamise teel. Ajutiselt saab arendusvahendis muuta kehtetuks ka CSS deklaratsioonilausetes väärtuste kehtivust, tehes seda mugaval viisil CSS lausetes CSS atribuutide ees märkeruudu funktsionaalsusega elemendil klõpsates.

2.2.2 Tutvumine vahendi kasutajaliidesega ekraanipiltide vahendusel

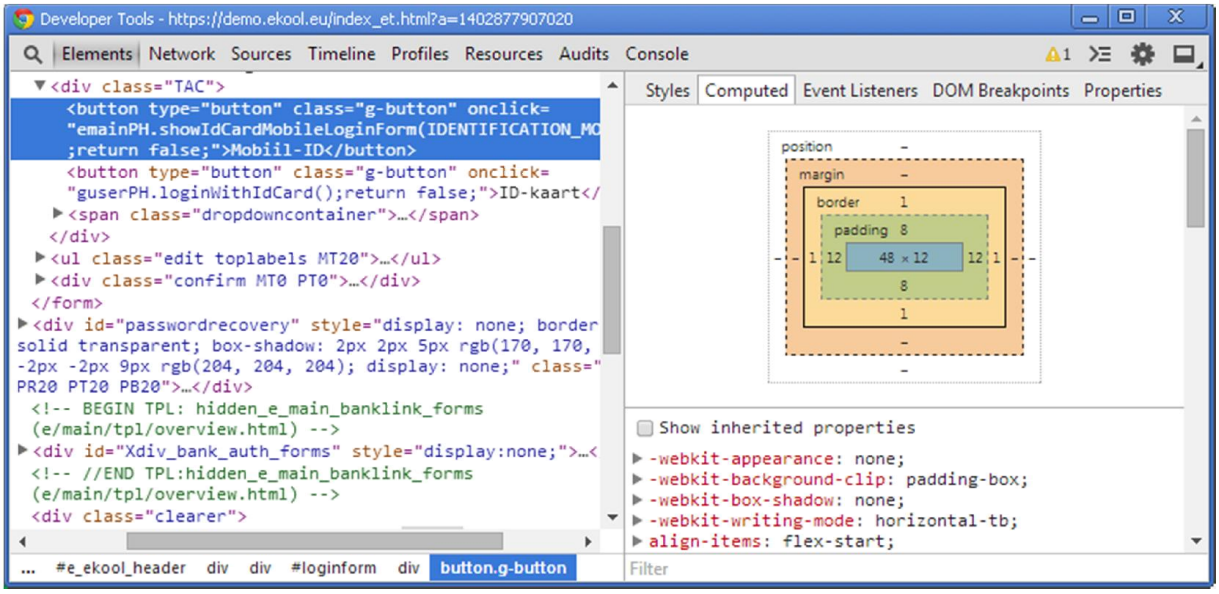
Alapeatükis 2.2.2 kujutatakse joonisel 6, joonisel 7, joonisel 8 ning joonisel 9 pildiliselt Inspectori ekraanivorme eri marki veebibrauserites.



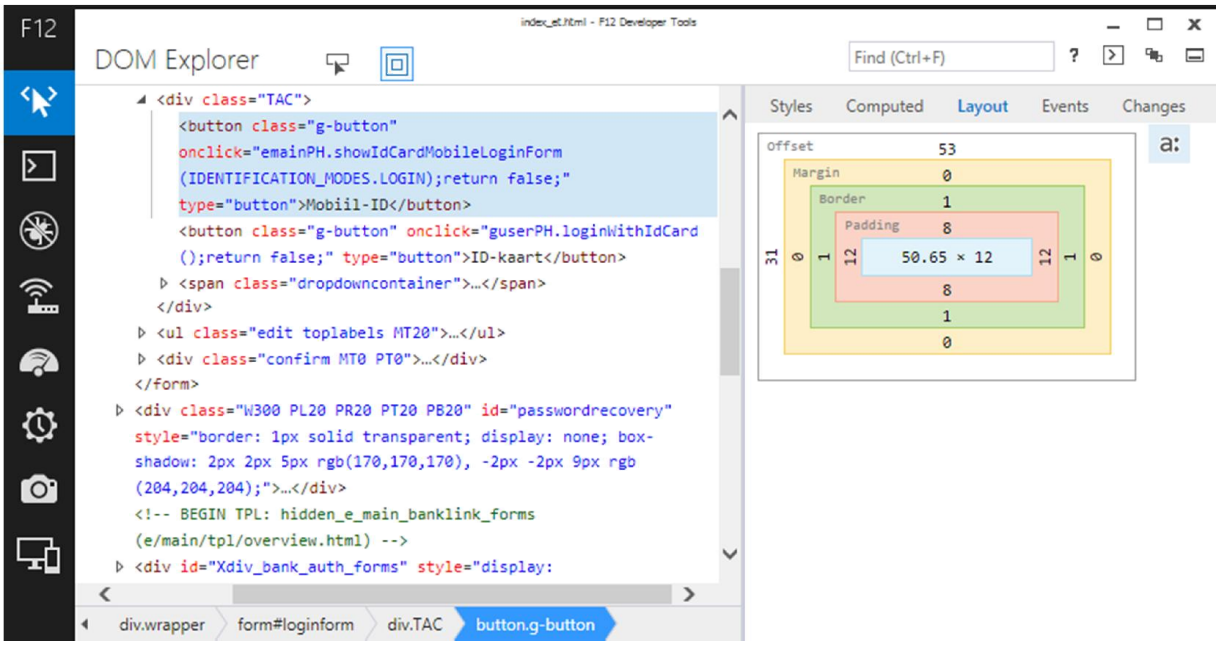
Joonis 6. Inspector veebibrauseris Mozilla Firefox koos lisaprogrammiga Firebug elemendi mõõtmete alamvaates (paremal)



Joonis 7. Inspector veebibrauseris Mozilla Firefox elemendi mõõtmete alamvaates



Joonis 8. Inspector veebibrauseris Chrome elemendi mõõtmete alamvaates



Joonis 9. Inspector veebibrauseris Microsoft Internet Explorer 11 elemendi mõõtmete alamvaates

2.2.3 DOM Inspector/Explorer'ga töötamisel hinnatavad praktilised kriteeriumid

Tabel 4

Inspectori omaduste võrdlus püstitavate kriteeriumite kaupa

	FF30 Firebug	**Firefox 30**	**Chrome**	**MSIE 11**
DOMis sõlmede ümberjärjestamise võimalus hiirega DOM-puus	EI	EI	JAH	JAH
Võimalus sisestada valitud HTML-elemendile CSS-atribuute *margin*, *border*, *padding* ning elemendi dimensioone ja paiknevust mõõtmete alamvaates (vt joonised 6,7,8,9)	JAH	EI	JAH	JAH (kuid ebamugav – tekstisisestus-lahter muudab enda asukohta kui saab fookuse)
Atribuutide arvväärtuste muutmine nooleklahvidega 1 ja 10 võrra	JAH	JAH	JAH	JAH
Atribuutide arvväärtuste muutmine nooleklahvidega 0.1 võrra	JAH (CTRL+nooleklahv)	JAH (ALT+nooleklahv)	JAH (ALT+nooleklahv)	EI
DOMis-sõlme kopeerimine ja kopeeritud sõlme DOMi tagasi kleepimine	JAH (replace node, paste before , paste after, paste as first child, paste as last-child)	EI	EI	JAH (paste as child, paste before)
CSS-stiilideklaratsioonis atribuudi väärtuse üledefineerimine viisil, et arendusvahendis jääks ka vana väärtus nähtavalt alles	EI (uue atribuudi lisades, milline juba varem eksisteeris kirjutatakse olemas olnud atribuut uue väärtusega üle)	JAH	JAH	JAH
Aruanne brauseriakna viimase laadimise hetkest praeguseni hetkeni arendaja poolt vahendis muudetud CSS-stiilideklaratsioonidest	EI	EI	EI	JAH

	FF30 Firebug	**Firefox 30**	**Chrome**	**MSIE 11**
CSS-tee leidmise ja lõikelauale kopeerimise võimalus DOM puus valitud HTML-elemendile	JAH (Copy CSS path)	JAH (Copy unique selector)	JAH (Copy CSS path)	EI
XPath-tee leidmise ja lõikelauale kopeerimise võimalus DOM puus valitud HTML-elemendile	JAH (Copy XPath)	EI	JAH (Copy XPath)	EI
DOM-puus valitud HTML elemendi HTML lähtekoodi redigeerimine mitte popupis vaid suures mugavas tekstiredaktori alas	JAH	EI	EI	EI

Vastavalt tabelis 3 leitud tulemustele järjestan veebibrauserite margid vastavalt nende arendusvahendis oleva konsooli võimekusele pingeritta:

1. Mozilla Firefox 30 Firebugiga
2. Google Chrome 35
3. Internet Explorer 11
4. Mozilla Firefox 30

3. Tulemused ja kokkuvõte

Töö eesmärgiks oli koostada eestikeelne sissejuhatav õppevahendina kasutatav juhendmaterjal veebibrauseritesse integreeritud arendusvahenditega iseseisvalt tutvumiseks ning uurides arendusvahendite omadusi anda soovitus uuritute hulgast sobivaima ja võimekaima vahendi valikuks.

Uuritud vahendite hulgast osutus sobivaimaks arendusvahendiks Mozilla Firefox kombinatsioonis koos lisaprogrammiga Firebug. Teisele kohale paigutus Google Chrome brauseris olevad arendusvahendid. Kolmandale kohale tuli Microsoft Internet Explorer 11. Viimaseks jäi Mozilla Firefoxi enda sisene arendusvahend. Kokkuvõtvate tulemuste koostamisel arvestasin, et vahendi Inspector iseärasusi tuleb arvestada suurema kaaluga kui konsooli omi, sest Inspector on brauserisse integreeritud arendusvahenditest veebiarendajale põhiliseks tööriistaks.

Töö eesmärk saavutati osaliselt, sest algselt oli plaanis tutvustada rohkem arendusvahendeid kui töös hetkel käsitletakse, kuid ajalised piirangud ning bakalaureusetöö tavapärane maht seadsid sellele piirid.

Täpsema soovituse koostamiseks sobivaima arendusvahendi leidmisel tuleb mitte piirduda selles töös uuritud vahendite hulgaga, vaid valimit laiendada.

JavaScripti ja HTML5ga seonduv valdkond on kiiresti arenev, nii samuti ka arendusvahendid, mille tõttu problemaatika ei kaota oma aktuaalsust. Käesoleva kirjatööde laadset vajadust rõhutab tüüpilise tarkvaraarendaja võrdkuju, kes on harjunud töötama enda harjumuspäraste kätteõpitud vahenditega ning vaevu ületama enda mugavustsooni, et kulutada aega paremate vahendite leidmiseks kui need tekivad.

Käesolev töö on elektroonilisel kujul avalikult kätte saadav TTÜ raamatukogus, on terviklik ega sisalda tööst lahus olevaid osasid, mille kasutus on litsentside või muuga piiratud. See tagab võimaluse, et seda tööd kasutatakse ja arendatakse edasi täpsuse ja ajakohasuse suunas.

Summary

The goal of this bachelor thesis was to produce in Estonian a didactic guide to developer tools integrated into web browsers and by investigating the features of these tools to provide a recommendation for the optimal tool selection.

Among the researched tools the best development tool came out to be *Mozilla Firefox* with *Firebug* add-on program. *Google Chrome* with its integrated tools placed second in the evaluation, third was *Microsoft Internet Explorer* 11. The last was *Mozilla Firefox* with its integrated development tools. By summarizing the results of this research and creating the ranking order I interpreted the features of *Inspector-tool* with more weight than the features of *Console-tool* for the sole reason, that the *Inspector* is the main tool used by the web developer.

The objective of this work was fulfilled only partly, as time constraints and the usual volume of bachelor thesis did set their limits – by the initial plan I had in mind I wanted to introduce and investigate more tools as I did by now.

In order to provide a more exact recommendation for the selection of the optimal developer tool, one should not restrict the research to the tools researched by this thesis, but the selection of the tools should be widened.

The JavaScript and HTML5 related subject field is fast evolving, so are the developer tools, and therefore the subject touched by this thesis will not lose actuality. The need for similar papers such as this thesis is emphasized by the typical software developer, who is used to work with a fixed tools known to him and does not step out of his comfort zone to invest time to find better tools when such might become available on the market.

The current work is openly available as an electronic document in the library of Tallinn Technical University, in its entirety and does not contain any separate parts that usage is restricted by licenses and such. That ensures the opportunity that this work can and will be used by others to develop it more further towards exactness and timeliness.

Kasutatud kirjandus

1. Opera gears up at 300 million users. [Võrgumaterjal] 2013. veebruar 13. a. [Vaadatud: 26. mai 2014. a.] http://www.operasoftware.com/press/releases/general/opera-gears-up-at-300-million-users.

2. Top 5 Desktop Browsers from Dec 2013 to May 2014. *StatCounter Global Stats.* [Võrgumaterjal] [Vaadatud: 2. juuni 2014. a.] http://gs.statcounter.com/#desktop-browser-ww-monthly-201312-201405-bar.

3. **Hewitt, Joe.** Introducing Firebug 1.0. [Võrgumaterjal] [Vaadatud: 26. mai 2014. a.] http://joehewitt.com/2006/11/15/introducing-firebug-10.

4. Console API. [Võrgumaterjal] [Vaadatud: 26. mai 2014. a.] https://getfirebug.com/wiki/index.php/Console_API.

5. Using the Console. *Chrome Developer Tools.* [Võrgumaterjal] [Vaadatud: 5. juuni 2014. a.] https://developer.chrome.com/devtools/docs/console#using-the-console.

6. Using the Console to view errors and debug. *Microsoft Developer Network.* [Võrgumaterjal] [Vaadatud: 6. juuni 2014. a.] http://msdn.microsoft.com/en-US/library/ie/dn255006(v=vs.85)#mainSection.

7. Command Line API. [Võrgumaterjal] [Vaadatud: 26. mai 2014. a.] https://getfirebug.com/wiki/index.php/Command_Line_API.

8. Command Line API reference. [Võrgumaterjal] [Vaadatud: 26. mai 2014. a.] https://developer.chrome.com/devtools/docs/commandline-api.

9. **Seddon, Ryan.** The Remote Debugging Landscape. [Võrgumaterjal] [Vaadatud: 5. juuni 2014. a.] http://thecssninja.com/talks/remote_debugging/#alert4.

10. **Liang, Yuxian Eugene.** Javascript Testing. *Javascript Testing.* lk 200. : Packt Publishing, 2010.

Lisa 1

Töös käsitletud rakendusliidestega saab iseseisvalt tutvuda järgnevatel veebiaadressitel:

Console API

- Console API
  https://getfirebug.com/wiki/index.php/Console_API
- Console API Reference
  https://developer.chrome.com/devtools/docs/console-api
- Console Debugging API
  http://msdn.microsoft.com/en-US/library/ie/hh772173(v=vs.85)
- Console
  https://developer.mozilla.org/en-US/docs/Web/API/console

Command Line API

- Command Line API
  https://getfirebug.com/wiki/index.php/Command_Line_API
- Command Line API Reference
  https://developer.chrome.com/devtools/docs/commandline-api
- Helper commands
  https://developer.mozilla.org/en-US/docs/Tools/Web_Console#Helper_commands