

Tallinna Ülikool  
Informaatika Instituut

# Veebirakenduste kasutajaliideste kohandatud elementide loomine Polymer projekt vahenditega

Bakalaureusetöö

Autor: Kristina Õim

Juhendaja: Andrus Rinde

Autor: ..... ,, ..... ,, 2015

Juhendaja: ..... ,, ..... ,, 2015

Instituudi direktor: ..... ,, ..... ,, 2015

Tallinn 2015

## Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.....

(autor)

# **Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks**

Mina \_\_\_\_\_ (sünnikuupäev: \_\_\_\_\_)

*(autori nimi)*

1. annan Tallinna Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

---

---

---

*(lõputöö pealkiri)*

mille juhendaja on \_\_\_\_\_,

*(juhendaja nimi)*

säilitamiseks ja üldsusele kättesaadavaks tegemiseks Tallinna Ülikooli Akadeemilise Raamatukogu repositooriumis.

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.

Tallinnas/Haapsalus/Rakveres/Helsingis, \_\_\_\_\_

*(digitaalne) allkiri ja kuupäev*

# Sisukord

|   |           |
|---|-----------|
| <b>SÕNASTIK</b> .....   | <b>5</b>  |
| <b>SISSEJUHATUS</b> .....                                     | <b>6</b>  |
| <b>1 PROJEKT „POLYMER“</b> .....                              | <b>7</b>  |
| 1.1 KOHANDATUD ELEMENDID ( <i>CUSTOM ELEMENTS</i> ) .....     | 8         |
| 1.2 POLYMER ELEMENTIDE ESITLEMINE .....                       | 11        |
| 1.3 POLYMERI KOHANDATUD ELEMENTIDE LISAMINE VEEBILEHELE ..... | 12        |
| 1.4 ELEMENTIDE TÜÜBID.....                                    | 12        |
| 1.5 VARI, LIHTSUSTATUD JA ÜHENDATUD DOM .....                 | 14        |
| 1.6 VEEBIKOMPONENTIDE POLYFILL´ID.....                        | 15        |
| <b>2 DISAIN POLYMERIGA</b> .....                              | <b>17</b> |
| 2.1 KUJUNDUSELEMENTIDE KASUTAMINE .....                       | 17        |
| 2.2 VEEBIANIMATSIOONID ELEMENTIDE RIKASTAMISEKS .....         | 18        |
| 2.3 IKOONID.....  | 20        |
| 2.4 AKNAD.....  | 22        |
| 2.5 NUPUD .....   | 23        |
| 2.6 TÖÖRIIST „DESIGNER“ .....                                 | 26        |
| <b>3 POLYMER PROJECT VAHENDITE KATSETAMINE</b> .....          | <b>28</b> |
| 3.1 DIALOGIAKNA LOOMINE .....                                 | 28        |
| 3.2 TEATEAKNA LOOMINE .....                                   | 29        |
| 3.3 FIREBASE ANDMEBAASI LIIDESE LOOMINE.....                  | 30        |
| 3.4 NUPU LOOMINE.....   | 31        |
| 3.5 MÄRKERUUDU LOOMINE .....                                  | 32        |
| <b>KASUTATUD KIRJANDUS</b> .....                              | <b>35</b> |

## Sõnastik

|  |   |
|--|---|
| API<br><i>(Application Programming Interface )</i> | Rakendusliides, arvuti operatsioonisüsteemiga või rakendusprogrammiga määratud reeglistik mille alusel rakendusprogramm kasutab operatsioonisüsteemi või teiste rakenduprogrammide teenuseid. |
| Core element                                       | Põhielement. Nähtavad (ikoonid jms) ja peidetud (varustamine) praktilised elemendid.<br><br>Näiteks <code>&lt;core-header-panel&gt;</code>  |
| Paper element                                      | Paberelement. Materjalikujunduse töövahendid – luuakse nuppudele kujundust (varju, värvi) kui ka nuppe endeid<br><br>Näiteks <code>&lt;paper-shadow&gt;</code>                                |
| UI (User Interface)                                | Kasutajaliides. Arvutiprogrammi ja kasutaja vaheline ühenduslüli.   |
| DOM ( <i>Document Object Model</i> )               | Dokumendi objekti mudel. Eeskiri kuidas objekte (tekst, pildid, pealkirjad, lingid jm) veebilehel esitada.  |
| Shadow DOM   | Vari DOM on DOM'i võime kaasata alampuu DOM elementidest dokumendi visualisualiseerimisse, kuid mitte põhilise dokumendi DOM puusse.  |
| Light DOM  | Lihtsustatud DOM, kasutab vari DOM elementi uues võtmes ehk võtab vari DOM-ist ainult osakese kasutusele.   |
| SVG Animations                                     | Skaleeritava vektorgraafika animatsioonid   |

## Sissejuhatus

Tänapäevases väga kiiresti arenevas veebis tekivad pidevalt uued võimalused ja lahendused veebielementide loomiseks, ehitamiseks ja kasutamiseks. Kõike püütakse teha lihtsamaks, et hoida kokku aega ja selle tulemusel ka raha. Rakendused valmivad suure koodi kirjutamise, rakenduse kujundamise ning katsetamise tagajärjel. Et olla kaasaegne püütakse meetodeid teha lihtsamaks ning loovamaks. Eesmärgiks luua üha uusi omanäolisi kujundusi ning standardsed kasutajaliidese elemendid ei rahulda enam disainereid. Soovitakse luua erilisi, kohandatud elemente, mis suudaksid rohkem kui varasemalt harjutud on.

Autor valis selle bakalaureusetöö teema, kuna on ise kokku puutunud säärase vajadusega luua kohandatud elemente, lisaks tundus vaadeldav tehnoloogia esmasel tutvumisel huvitav ning põnev. Esimest korda saabus informatsioon Polymeri kohta Google I/O konverentsilt. Google arendajad on ise väga vaimustuses Polymer project'ist ja usuvad selle tulevikku kui nad esimest korda sellest 2014.aastal just selle konverentsi käigus rääkisid. Tegemist on väga uue ning jätkuvalt töös oleva tehnoloogiaga ning oma suure potentsiaali tõttu vajab kindlasti tutvustamist.

Käesoleva bakalaureusetöö eesmärgiks on tutvustada Polymeri olemust, kohandatud kasutajaliidese elementide loomist ja katsetada, kas tehnoloogia pakub tõesti seda mida lubab – luua lihtsalt, mugavalt, töökindlalt elemente ning nende toimimist. Töö võiks pakkuda huvi veebirakenduste disaineritele, kes soovivad kasutada omanäolisi kasutajaliidese elemente ja arendajatele, kes soovivad lihtsustada elementidega ümberkäimist. Samuti neile, kes tunnevad teema enda jaoks huvitav olevat.

Autor tutvub tehnoloogia spetsifikatsiooniga ning katsetab praktikas olulisemaid võimalusi. Üritab välja selgitada tehnoloogia tugevusi ja probleeme, kui neid leidub ning annab omapoolse hinnangu selle kasutatavuse kohta.

# 1 Projekt „Polymer”

Veebilehtedele interaktiivsuse lisamiseks ja veebirakenduste kasutajaliidese loomiseks on traditsiooniliselt kasutatud põhilisi elemente. Nendeks on näiteks `<select>`, `<div>`, `<img>`, `<body>`, `<input>`. Mõned neist on päris võimekad, kuid enamus ei tee väga midagi. Üheks võimekaks võib lugeda `<select>` elementi, alates tema funkionaalsusest lõpetades paindlikusega. Teda saab kaasata enamus teistesse elementidesse ning tema käitumine võib vastavalt sellele muutuda.

Kuna veebirakendused on muutunud nii keerukaks, ei piisa enam ainult nendest põhilistest elementidest. Seda puudujääki on siiani parandatud skript-idega. Skript on mingis programmeerimiskeeles kirja pandud käsujada, mida täidetakse ilma kasutajapoolse vahelesegamiseta (vallaste, 2007). Kui kujutada ette veebilehel kuupi siis skriptiga on võimalik panna see kuup lõputult erinevates suundades pöörlema. Ta vahetab juhuslikult tahku mingi kindla aja tagant. Polymeri loojate arvamuse kohaselt ei ole see lahenduseks, kirjutada suurtes kogustes skripti vaid pigem luua selle asemel võimsamaid elemente. Uus tehnoloogia veebielemendid, teevad selle töö võimalikuks. Idee põhineb mõttel põimuda väiksemaid elemente kokku suuremateks ja võimsamateks elementideks. Nende elementide kasutamisel tuleb samuti kasutada skripti, kuid tunduvalt vähem kui seda on tehtud siiani (polymer, 2015).

Polymer muudab elementide loomise väga lihtsaks, kerge on teha nuppe ning koguni terveid rakendusi, mis töötavad erinevatel platvormidel. Kokkuvõtlikult öeldes on Polymer „isevalmistatud” koostalitusvõimeliste elementide kogu, mis laiendab HTML – i (*Hyper Text Markup Language*) (polymer, 2015). Loodud on ta Google arendajate poolt ning Polymeri tutvustati esimest korda Google I/O konverentsil 2014.aastal, üritusel mis toimub igal aastal, paar päeva. Siis tutvustatakse uuemaid arendusi ning prototüüpe. Korraga viibib üritusel 5000 kuulajat ning veebivahendusel kantakse seda ka otse üle. Konverents on mõeldud tarkvaraarendajatele. Käesolev töö on valminud Polymer 0.5 vahenditega, mis oli ainuke väljalase Polymerist antud bakalaureusetöö kirjutamise alustamishetkel.

Bakalaureusetöö valmimise hetkeks on Polymer välja lasknud uue versiooni oma teegile – 0.8. See ei ole mõeldud veel hetkel asendada 0.5te, ta on olnud väga lühikest aega püsiv ning kõiki elemente ei ole sinna veel üle kantud. Küll aga on uus versioon mõeldud baasina

edasiseks tööks (polymer, 2015). Hetkel oodatakse tagasisidet ning mõtteid kasutajatelt, et arendada Polymeri veel rohkem edasi ning anda välja versioon 1.0.

Polymer jaotab oma vahendid mõtteliselt kihtidesse, mille alusel tehnoloogiast paremini aru saada. Kihte on kolm:

- Veebikomponendid (*Web Components*) – tavalistele HTML komponentidele üles ehitatud kohandatud elemendid. Kuna kõik veebilehitsejad ei toeta neid veel siis eraldi kiht komponente tegeleb sellega, täites Javascript'i API'sid (Application Programming Interface) ehk rakendusliidest (vallaste, 2007). Valitakse kiireim tee kas omaloodud või Javascript.
- Polymeri teek (*Polymer library*) – annab deklaratiivse süntaksi, millega on lihtsam defineerida omaloodud elemente. Lisab võimalusi ehitamiseks võimekamaid ja korduskasutavamaid kasutajaliidese elemente kui varem.
- Elemendid (*Elements*) – põhi- (*Core elements*) ja paberelementikogud (*Paper elements*) moodustavad laiaulatuslikud kasutajaliidese (*UI*) ja kasutajaliidesevälised (*non-UI*) elemendid, mida saab kasutada tavapärasest erinevalt. Need küll sõltuvad Polymeri teegist, kuid on siiski eraldiseisvad. Saab kasutada ka ilma Polymerita või Polymeri kasutades luua täiesti enda elemendid ning mitte siduda neid põhi- ja paberelementitega. Põhi – ja paberelemente saab ka kokku sobitada teiste elementidega seal hulgas kohandatud elementidega (polymer,2015).

## 1.1 Kohandatud elemendid (*Custom Elements*)

Polymeril on oma filosoofia: „kõik on elemendid”. Kohandatud elemendid järgivad seda mõtet. Nad lasevad defineerida oma elemendi tüübid kohandatud elemendi (*tag*) nimedega. Javascript'i koodi saab liita kohandatud elemente ning neid kasutatakse nagu tavalisi elementegi. Näiteks võib luua uue elemendi nupu kasutajaliidesele nimega „minu-loodud-nupp”, sellisel juhul kasutatakse seda järgnevalt (Koodinäide 1).

```
<minu-loodud-button></minu-loodud-nupp>
```

**Koodinäide 1 – element**



NB! Kohandatud elementide nimed peavad alati sisaldama sidekriipsu (-).

Seda elementi saab veebilehele luua ka Javascript'is nagu tavalisi HTML elemente (Koodinäide 2).

```
var s = document.createElement('minu-loodud-nupp');  
s.innerHTML = "Olen äge!";
```

### Koodinäide 2 – elemendi deklareerimine

Kohandatud elementidel saab kasutada standart DOM (Document Object Model) meetodeid, panna juurde sündmuste kuulajaid (*event listeners*), omada ligipääsu omadustele ja kujundada neid CSS – iga (polymer, 2015).

DOM on programmeerimisliides HTML-ile, XML-ile ja SVG (*Scalable Vector Graphics*) dokumentidele. Annab struktureeritud vaate dokumendist (puu) ning defineerib kõik dokumendi elementide objektid ja omadused. Samuti ka meetodid nende kättesaamiseks, et teha vajalikud muudatused dokumendi struktuuris, stiilis ja sisus (MDN, 2015).

Polymeri elemendi defineerimisel ei kutsuta otse `registerElement` nagu tavaliselt seda tehakse vaid kasutatakse `<polymer-element>` elementi nagu allpool olevas näites välja toodud (Koodinäide 3). Niimoodi luuakse elemente Polymeris. Fraasi `registerElement` kasutatakse, et registreerida uut kohandatud elementi veebilehitsejas, mis seepeale tekitab uue elemendi (loob uuele elemendile konstruktori) (MDN, 2015).

```
<polymer-element name="tervitus">  
  <template>  
    <div> Tere </div>  
  </template>  
  <script>  
    Polymer(...);  
  </script>  
</polymer-element>
```

### Koodinäide 3 – polymeri elemendi loomine

`<template>` element defineerib kasutajaliidese elemendi ja `Polymer()` skripti sees registreerib elemendi.

Kohandatud elementidel on võrreldes tavaliste elementidega mitmeid eelised:

- Vähendab koodi kogust, mida kirjutama peab – suure hulga skripti asemel on võimsamad elemendid, mis hoiavad koodi koguse väiksemana.
- Väljendab koodi funktsionaalsust – nupud, millega saab teha rohkem kui varem aga väiksema vaevaga.
- Kapseldab sisemisi detaile – andmed ja funktsioonid pakitakse kokku ühte komponenti. Element elemendi sees.
- Täidab API´sid (*Application Program Interface*) vastavalt elemendi tüübile – rakendusliidest täidetakse vastava elemendi juurde kuuluvate meetodite ja omadustega
- Suurendab produktiivsust, lastes elemente korduvkasutada – pole vaja midagi uuesti luua, saab kasutada ka teiste loodud elemente ning neid ka laiendada. Kord loodud elemente saab üha uutes projektides kasutada.
- Kasutab pärimist (*inheritance*), et luua elemendi märgendeid põhinedes teistele märgenditele (polymer, 2015) – uued märgendid on arusaadavamad, ei teki segadust funktsionaalsuses.

Polymer lubab luua kohandatud elemente deklaratiivselt ning varustab spetsiaalsete funktsioonidega nagu näiteks:

- kahesuunaline andmete - sidumine, mis tähendab seda, et kui omadused (*properties*) muutuvad andmetes siis muutub ka UI (*user interface*) ehk kasutajaliides. Ning samamoodi kui UI elemente muudetakse siis muutused paljundatakse ka andmetesse (stackoverflow, 2012).
- deklaratiivne sündmuse käsitlemine – sisaldab sündmuste tõlgendamist, millele järgnevad välised toimingud või muudatused programmi olekus (Microsoft Research, 2015). Tavaliselt käsitleb sündmusi, millele järgnevad lisa toimingud või muudatused programmi.
- deklaratiivne pärandamine – vanemelemendi omadusi ja meetodeid omistatakse ka tema tütarelementidele

- ja palju muud

Kohandatud elementide spetsifikatsioon defineerib programmilise viisi, et defineerida kohandatud elemente kasutades `document.registerElement` (Koodinäide 4). See võimaldab elementi kohe kasutama hakata.

```
var minu-element = document.registerElement('minu-element');
```

**Koodinäide 4 – elemendi registreerimine**

## 1.2 Polymer elementide esitlemine

Kui element on registreeritud siis saab loodud elemente veebilehele lisada kas käsikval viisil kasutades `document.createElement` või deklaratiivselt (polymer, 2015) (Koodinäide 5). Täpselt nagu HTML elementegi.

```
<minu-element></minu-element>
```

**Koodinäide 5 - element**

Polymer element saab laiendada teisi elemente kui kasutada definitsioonis `extends` atribuuti. Laiendamise all peetakse silmas, et koodi funktsionaalsus kasvab, ta suudab teha rohkem kui enne. Vanemelemendi (*parent*) omadused ja meetodid pärandatakse tütarelemendile (*child*) ja andmetele (*data-bound*). Samamoodi saab laiendada tavalisi DOM elemente nagu ka kohandatud elemente, näiteks `div` ja `button` (Koodinäide 6) .

```
<polymer-element name="loendaja-nupp" extends="button"
  on-click="increment">
  <template>
    Increment:
  </template>

  <script>
    Polymer({
      counter: 0,
      increment: function(e, detail, sender) { this.counter++ }
    })
  </script>
</polymer-element>
```

**Koodinäide 6 – elemendi laiendamine (polymer, 2015)**

Koodinäites laiendatakse tavapärasest sisendväljaga elementi button ehk nuppu. Lisaks võetakse kasutusele nupuvajutusi loendav loendur. Et kasutada Polymer elementi tuleb see importida oma faili kasutades *HTML Import* (Koodinäide 7).

```
<link rel="import"
      href="bower_components/paper-button/paper-button.html">
```

**Koodinäide 7 – HTML faili importimine (polymer, 2015)**

### 1.3 Polymeri kohandatud elementide lisamine veebilehele

Polymeri abil loodud kohandatud elemendid pannakse kirja eraldi loodud HTML faili. See fail tuleb elementide töötamiseks importida tegelikule veebilehele ning seejärel seal välja kutsuda. Selleks kasutatakse tavalist *link* elementi (Koodinäide 8). Meeles tuleks pidada, et ainult ühte elemendi faili saab importida, see tähendab et kui importida mitu faili siis välja kutsutavat elementi ei kuvata, sest leheküljel ei otsi imporditud failide seast õiget faili õige elemendiga.

HTML importimine laseb taaskasutada HTML dokumente teistes HTML dokumentides. Samamoodi nagu `<script>` element laseb kaasata välist Javascript'i. Import lubab kaasata kohandatud elementide definitsioone väliste URL ehk internetiaadressite kaudu (polymer,2015).

```
<link rel="import" href="minu-kohandatud-element.html">
```

**Koodinäide 8 – HTML importimine**

### 1.4 Elementide tüübid

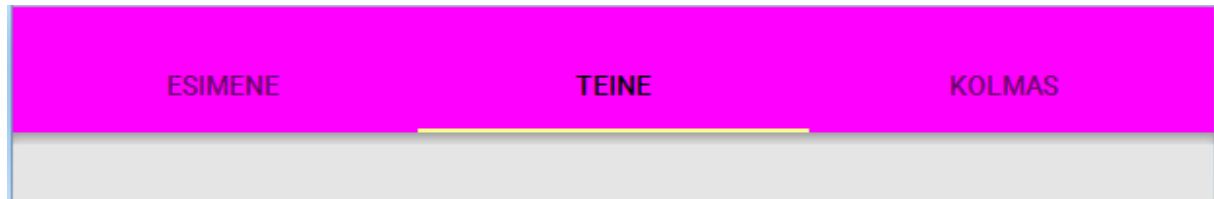
Elementide käitumise ja kasutuse alusel saab neid jagada kaheks. Kasutajaliidese (*User Interface*) elemendid, mis visualiseerivad kasutajaliidese ekraanile ja kasutajaliidesevälised (*non - UI*) elemendid, mis annavad teisi teenuseid kasutajaliidest visualiseerimata.

Sarnaselt HTML-ile on kasutajaliidese elementideks `<select>` ja `<core-selector>`, nad visualiseerivad kasutajaliidese (*UI*) ja on nähtavad veebilehel. Nendeks elementideks on veel `<core-collapse>`, `<core-toolbar>` ja `<paper-tabs>` (polymer, 2015). Viimastega saab

luua oma veebilehele vahelehed (Joonis 1) (Koodinäide 9).

```
<paper-tabs selected="0">
  <paper-tab>Esimene</paper-tab>
  <paper-tab>Teine</paper-tab>
  <paper-tab>Kolmas</paper-tab>
</paper-tabs>
```

#### Koodinäide 9 - paper-tabs loomine



#### Joonis 1 – vahelehed paper-tabs elemendiga

Kasutajaliidesevälised elemendid ei visualiseeri ekraanile midagi. Neil elementidel on oma eesmärk ning nende täitmiseks ei visualiseerita kasutajaliidest. Näidetena `<script>` ja `<style>` elemendid. Nemad tegelevad teenustega nii öelda ekraani taga. Näiteks `<core-ajax>` märgend laseb teha AJAX päringu märgistusest (Koodinäide 10). Sellised elemendid vähendavad stereotüüpset koodi kirjutamist. Varjavad keerulist API-it nagu näiteks XMLHttpRequest (polymer, 2015).

```
<core-ajax url="http://gdata.youtube.com/feeds/api/videos/" auto
  params='{ "alt": "json", "q": "chrome" }' handleAs="json"></core-ajax>
<script>
  var ajax = document.querySelector('core-ajax');
  ajax.addEventListener('core-response', function(e) {
    console.log(this.response);
  });
</script>
```

#### Koodinäide 10 – AJAX päring (polymer, 2015)

Teiseks elementide liigitamise võimaluseks on Polymeri kaks elementide kogu: paber-elementid (*paper elements*) ja põhielemendid (*core elements*). Paberelementide kollektsioon sisaldab endas elemente materjali disainimiseks veebikeskkonnas. Sinna alla kuuluvad nii veebilehe paigutus, sirvimismeetodid kui ka elementide kujundus. Põhielementideks on visuaalsed ja mitte-visuaalsed üldkasutatavad elemendid. Näiteks kasutajate sisendid ning

valikud (polymer, 2015). Erinevatest elementidest tuleb juttu järgnevates peatükkides.

## 1.5 Vari, lihtsustatud ja ühendatud DOM

Veebilehe DOM sisaldab endas kõiki elemente, nüüd lisaks ka kohandatud elemente. Kohandatud elementide tegelik kirjeldus ja ülesehitus on lisatud eraldi failidesse, nii on kood ilusam ning ka arendajatel kergem jälgida. Veebilehe konsooli vaadates on Polymeri elemendid kuvatud eraldi alampuuna ehk vari DOM-is (*shadow DOM*). Vari DOM on veebilehitseja omadus kaasata alampuu DOM elementidest dokumendi visualiseerimisse, kuid mitte põhilise dokumendi DOM puusse (Glazkov. D, 2011). Lahutab sisu esitlemisest sealjuures välistades nime segadust ja parendades koodi ilmet.

Vari DOM spetsifikatsioonid varustavad programmilise meetodiga `createShadowRoot`, lisamaks vari DOM alampuud elemendile. Kui defineerida Polymer elementi siis elementide vari DOM sisu saab määratleda kasutades `<template>` elementi (polymer, 2015). Kõik mis jääb sinna vahele kuulub vari DOM-i, see on lõppkasutaja eest varjatud (Koodinäide 11). Ligipääs on arendajatel või administraatoril, kes sisu muudab.

```
<polymer-element name="minu-kohandatud-element" noscript>
  <template>
    <span>Inimesed: </span>
    <content select="q"></content>
    <footer>mõnikord</footer>
  </template>
</polymer-element>
```

### Koodinäide 11 – kohandatud elemendi vari DOM

Lihtsustatud DOM (*light DOM*) kasutab `<template>` elemendi sees olevat sisu osa, mis on määratletud `<select>` elemendiga ehk siis lihtsustatud DOM kasutab vari DOM elementi uues võtmes. Ühe sõlmega (*node*) on võimalik väljendada kõiki kolme alampuud – lihtsustatud DOM, vari DOM ja ühendatud (*composed*) DOM. Lihtsustatud DOM ja vari DOM koos moodustavad loogilise (*logical*) DOM-i, millega suhtlevad arendajad. Veebilehitseja kasutab ühendatud DOM- i, et piksleid ekraanile visualiseerida. Kohandatud elemendi kasutaja määrab lihtsustatud DOM – i, see on nähtav lõppkasutajale. Vari DOM on elemendi sisene ning seetõttu lõppkasutaja varjatud eest. Ühendatud DOM – i renderdab ehk visualiseerib veebilehitseja. Visualiseerimiseks jagatakse lihtsustatud DOM vari DOM-iga, et moodustada ühendatud DOM (polymer, 2015). Lihtsustatud ja vari DOM-is näitavad sõlmed

vanemelemendi ja tütarelemendi vahelisi suhteid vastava puu struktuurile.

## 1.6 Veebikomponentide polyfill'id

Kõikide elementide toimimiseks peavad olema veebilehitsejal olemas vahendid, mis neid visualiseerivad. Kuna kõikidel neid veel ei ole, on selleks olemas polyfillid. Polyfill on tükike koodi või plugin, mis tagab tehnoloogia, mida veebilehitseja peaks omama algupäraselt ehk teisiti öeldes tagab funktsionaalsuse töötamiseks elementidel ka vanemates veebilehitsejates (Sharp, R, 2010). Niimoodi on Polymer taganud tagasiühilduvuse veebilehitsejate vanemate versioonidega. Polymer ehitab alles loodavate veebikomponentide tehnoloogiate peale, mis pole veel saadaval kõigis veebilehitsejates. Sellegipoolest saab Polymeri kasutada igas lehitsejas, kasutades *Web Components* kasutajatuge – `webcomponents.js`. See on teek polyfillidest uute veebitehnoloogiate jaoks, mida pole veel kõigis lehitsejates kasutusel (polymer, 2015). Need teegid muudavad võimalikuks arendajatel kasutada neid standardeid üle kogu veebi moodsates veebilehitsejates. Allalaadides Polymeri algfailid on `webcomponents.js` seal kohe olemas, kuid seda faili võib allalaadida ka üksikult nagu kõiki teisi elementide faile. See fail tuleb samamoodi oma lehele importida, et tagada toimimine ka vanemates veebilehtedes. Muidu ei tööta näiteks veebilehe transformatsioonid üldse või nii nagu peaks.

Polymer on võtnud eesmärgiks töötada kõikides uuemates browseri versioonides, tundub ka kui „*evergreen*” veebilehitsejates. Mõiste „*evergreen*” tähistab antud juhul veebilehitseja võimet ennast ise värskendada uuendustega ilma et kasutaja peaks selleks midagi tegema (Dale, T, 2013). Veebilehitsejad IE11 +, Chrome, Chrome Androidile, Firefox, Safari 7 + ja mobiilne Safari on suutelised seda tegema (GitHub, 2015). Kõikide need veebilehitsejate uuemad versioonid toetavad Polymeri polyfille. Seega on tagatud toimimine kasutajale meelepärasel veebilehitsejas.

Varem säilitati polyfillide teeki Polymeri loojate poolt ning väljastati `platform.js` nime alt. Hetkel on nad ülekantud `WebComponents.org` ja ümbernimetatud `webcomponents.js` nime alla. Vanematel veebilehitsejatel hoitakse tagasiühilduvuse tarbeks ka `platform.js` koopiana `webcomponents.js`’ist.

Polyfill `webcomponents.min.js` teek sisaldab endas järgmist:

## 1. veebikomponendid

1. *Shadow DOM* (*vari DOM*) – DOM alampuude kapseldamine ning nendega seotud CSS
2. *HTML Imports* – laeb elemendi definitsioonid ja teised ressursid deklaratiivselt
3. *Custom Elements* (kohandatud elemendid) – defineerib uusi elemente HTML'is

## 2. Veebikomponentide polyfillide toimimiseks vajalikud

1. *WeakMap* – kollektsoon võtmetest ja väärtustest, kus võtmed on objektid ja väärtused võivad olla omavolilised väärtused (MDN, 2015)
2. *Mutation Observers* – jälgivad muutusi DOM – is, arendajatel on võimalus reageerida vastavalt (MDN, 2015)

Kokkupakitud Polymeri algafile allalaadides on `webcomponents.js` juba seal sees olemas aga seda saab ka üksikult allalaadida. Niimoodi on olemas kõik vahendid, et elemendid töötaksid ka vanemate veebilehitsejate peal. Siis saab kasutada *HTML Imports*, *Custom Elements*, *Shadow DOM* ja teisi standardeid oma rakendustes (polymer, 2015).

Edasi tuleb kasutada elementi nagu iga tavalist elementi. Kõikvõimalikud vajalikud failid on vaja eraldi importida lehtedesse allalaaditud kaustadest, vastasel juhul elemendid lihtsalt ei toimi.

Polymeri on võrdlemisi lihtne kasutada kui on olemas oskused ja teadmised HTML-ist ning CSS-ist. Põhjuseks süntaksite sarnasus. Tuleb kasuks ka Javascript'i oskus, kuna vähesel määral kasutatakse skripte erinevate elementide toimimiseks.



## 2 Disain Polymeriga

Disaini all peetakse silmas nii visuaalset disaini kui ka elementide osade liigutamist. Polymeri paberelementide (*paper-elements*) kolleksioon sisaldab disanielemente veebile. Põhielementide kolleksioon pakub elemente, mida saab kasutada et kujundada rakenduse paigutust lehel, üleminekuid ja sirvimisvõimalusi. Paigutuse puhul on võimalik luua tööriistaribasid (*toolbars*), vahelehti (*tabs*) ja kõrvalisi navigatsiooniribasid (*side nav*) (polymer, 2015).

Paber - ja põhielementide kohta saab pikemalt ning põhjalikumalt lugeda Polymeri kodulehelt (<https://www.polymer-project.org/0.5/docs/elements/> valikust „elements”). Seal on kõikide elementide kohta välja toodud näited, demod, seletused, põhjendused. Autori tehtud näiteid on võimalik vaadata veebileheküljelt <http://oimk.getmeteoric.com/> .

### 2.1 Kujunduselementide kasutamine

Polymeril on omad elementide kogud - põhielemendid ja paberelemendid. Need sisaldavad hulgaliselt elemente, mida kasutada erineva kujunduse saavutamiseks rakendustel. Järgnev loetelu toob välja mõned põhielemendid:

- `<core-header-panel>` - lihtne mahuti lehepea ja sisu osaga. Pea saab liikuda veebilehe liikumisega kaasa kui ka jääda paigale veebilehe ülemisse ossa. Sellisel juhul pole ta vaadeldav kui sirvija jõuab veebilehe lõppu.
- `<core-toolbar>` - saab kasutada rakenduse tööriistaribana või tööriistaribana väiksemale kasutajaliidese komponendile. Tööriistariba võib olla ka mahuti nuppudele ja veebilehekülgedele.
- `<core-drawer-panel>` - muutustele reageeriv (*responsive*) mahuti, mis kombineerib lehekülje navigatsiooni ja peamise sisu ala.
- `<core-scaffold>` - element rakenduse kasutajaliidese struktureerimiseks, tagab kiire muutustele reageeriva (*responsive*) kujunduse, mis sisaldab endas navigatsiooni, põhilist rakenduse tööriistariba ja sisu osa, mis on teostatud kasutades `core-drawer-panel`, `core-header-panel` ja `core-toolbar` (polymer, 2015).

## 2.2 Veeb animatsioonid elementide rikastamiseks

Veeb animatsioone kasutatakse, et lisada kohandatud elementidele ilmekust. Selleks kasutatakse samu tehnoloogiaid, mida tavaliste HTML failide puhulgi kasutada saab. Valiku tegemisel tuleb lähtuda kasutaja vajadustest ning tehnoloogiate tugevusestest ja nõrkustest. Veeb animatsioonid defineerivad API – sid (rakendusliidest), et ühtida veeb animatsiooni mudeleid keeruliste skriptitud (*scriptable*) animatsioonidega. Veebiplatvormidel eksisteerib hetkel 4 animatsiooni tehnoloogiat – sarnast spetsifikatsiooni: *CSS Transitions*, *CSS Animations*, *SVG Animations* / *SMIL* ja `requestAnimationFrame()`.

- *CSS Transitions* / *CSS Animations* ei ole väga väljendusrikkad. Animatsioone ei saa koostada järjestada ega isegi usaldusväärselt paralleelselt jooksutada. Ning neid ei saa skriptist välja võtta. Samas kõige lihtsam ning tuttavam viis kuidas kujundada elemente, rahuldab liidesed (polymer, 2015).
- *SVG Animations* on väga väljendusrikkad aga ka väga keerulised – sisaldavad suurt hulka koodi. Neid ei saa rakendada HTML sisule. Animatsioonideks kasutatakse animatsiooni elemente, mis olid algselt defineeritud *SMIL* animatsiooni spetsifikatsioonis (*Animation specification*) (CSS-Tricks, 2015). *SMIL (Synchronized Multimedia Integration Language)* võimaldab veebiarendajatel kuvada multimeediumsisu eraldiseisvate failide ja voogudena (audio, video, tekst, pildid), saata kasutaja arvutisse individuaalselt ning kuvada neid kui terviklikku multimeediumivoogu (vallaste, 2007).
- `requestAnimationFrame()` ei ole deklaratiivne lähenemine, veebilehitsejale antakse käsk käivitamiseks spetsiaalse funktsiooni, et animatsiooni uuendada ekraanil enne uuesti loomist (MDN, 2015)

Tulemas on uus spetsifikatsioon *Web Animations*, seda arendatakse kui W3C spetsifikatsiooni, olles osa CSS ja *SVG working groups*. Selle mõtteks on sihtida eelneva kirjeldatud tehnoloogia puuduseid, samuti asendada *CSS Transitions*, *CSS Animations* ja *SVG Animations* aluseid. Põhjused on järgnevad:

- vähendamaks töömahtu koodi hooldusel ehk vähendab arenduskulusid

- tagada animatsioonide koostöövõimelisus
- oleks üks platvorm, kus katsetada uusi animatsioonide uuendusi veebi parendamiseks (polymer, 2015)

*Web Animations* mudel on tehnoloogia kirjeldus veebis animatsioonide sisuks. Tehnoloogia on võimekas toetama ka *CSS Transitions*, *CSS Animations* ja *SVG Animations* mudeleid (W3C, 2014). *Web Animations* API avaldab liideseid ka Javascripti mudelile. Mõned tähtsamad neist on: *Animations*, *AnimationEffects*, *TimingDictionaries*, *TimingGroups* ja *AnimationPlayers* (polymer, 2015).

*Animations*, mis defineerib ühe ainsa animatsiooni efekti ühe elemendi puhul (W3C, 2014). Elemendil võib olla ka mitu efekti. Järgnevas näites elemendi „left” liigub sujuvalt 0 pikslilt üle 100 pikslini 2 sekundi jooksul (Koodinäide 12).

```
var animation = new Animation(targetElement,
  [{left: '0px'}, {left: '100px'}], 2000);
```

#### **Koodinäide 12 - animatsiooni liikumise defineerimine**

*AnimationEffects*, mis omab kontrolli, millised CSS omadused ja SVG atribuudid muudetakse animatsiooni poolt ning milliste väärtuste vahel need omadused ja atribuudid on.

*TimingDictionaries*, mida kasutatakse, et kontrollida animatsiooni mudelit. Sealt saab välja tuua mitmeid omadusi.

- *duration* (kestvus) – animatsiooni üks kordus
- *iterations* (iteratsioon) - mitu korda animatsiooni mängitakse
- *iterationStart* (iteratsiooni algus) – esimese korduse start
- *fill* – kas elemendil on juba enne animatsioon sellised omadused nagu animatsiooni alustades, kas talle jäävad samad omadused pärast animatsiooni lõppu
- *delay* – viivitus enne animatsiooni käivitumist
- *playbackRate* - animatsiooni kaadrisagedus

- *direction* – suund, milles kordused tagasi mängivad
- *easing* – kontrollib kuidas väline aeg mõjutab animatsiooni kogu animatsiooni toimimise vältel (polymer, 2015)

Mõeldud on ka vajadusele animatsioone sünkroniseerida ja järjestada, selle lahendab *TimingGroups*. Neid on kahte tüüpi: *AnimationGroup*, võimaldab mitu olemust siduda üheks transformeeritavaks tervikuks (formz, 2015), ja *AnimationSequence*, mis võimaldab mängida ühe animatsiooni kaadri korraga, neid kaadreid järjest tagasimängides saab luua sujuva animatsiooni (*Unreal Engine*, 2015).






*AnimationPlayer*, esindab ühte animatsiooni mängijat, annab täieliku kontrolli animatsiooni alguse ja tagasimängimise üle. Kasutaja ei saa muuta animatsiooni sisemisi detaile (MDN, 2015).

Järgnevalt on välja toodud kõige enam kasutatavad elemendid ning nende loomised. Suurt kasutatavust põhjendab autori läbivaadatud näidiste ja harjutustega (*tutorials*), kus nendele pannakse suurt rõhku. Samuti on nendele tähelepanu pööratud Google I/O konverentsil ning Google ametlikes õpetustes.

## 2.3 Ikoonid

Väga palju kasutatakse ikoone. Polymer kasutab selleks `<core-icon>` elementi, selleks tuleb ta elemendifaili importida. Importitud HTML fail sisaldab endas elemendi loomiseks vajalikke vahendeid ja üle 150 erineva levinud ikooni. Olemas on ka terve kogu erinevaid ikoone, mille praktiseerimiseks on mõistlik kasutada `<core-iconset>` elementi. Lisaks võib seda elementi kasutades luua päris enda ikoonidest koosneva kollektsiooni (polymer, 2015). Autor on välja toonud mõned näited valmisikoonidest Polymeri kollektsioonist (Tabel 1). Vaikimisi on ikooni suuruseks 24px, näidetes on parema vaadeldavuse huvides ikoonid suurusega 35px.



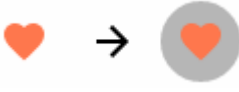



**Tabel 1 – valik ikoone**

|  |   |
|--|---|
| Prügikasti ikoon - <code>&lt;core-icon class="delete" icon="star"&gt;&lt;/core-icon&gt;</code> |  |
| Tähekese ikoon - <code>&lt;core-icon class="suur" icon="star"&gt;&lt;/core-icon&gt;</code>     |  |
| Äratuskella ikoon - <code>&lt;core-icon class="suur" icon="alarm"&gt;&lt;/core-icon&gt;</code> |  |
| Android - <code>&lt;core-icon class="suur" icon="android"&gt;&lt;/core-icon&gt;</code>         |  |
| Asukoha ikoon - <code>&lt;core-icon class="suur" icon="room"&gt;&lt;/core-icon&gt;</code>      |  |

Paberelementide kollektsioon sisaldab endas mitmeid kontrolleri elemente (polymer, 2015). Nende hulka kuuluvad tavapärased surunupud (*pushbutton*), märkeruudud (*checkbox*), raadionupud (*radio button*), pöörd – ja liugnupud (*spin button*, *slider*) ning ikoonid (*icon*) aga ka mõningaid uusi nuputüüpe, näiteks spetsiaalselt puutekraanidele mõeldud „hõljuvad” nupud (*paper-fab*). Järgnevalt mõned näited nuppudest (Tabel 2).

Nende kontrolleri hulka kuuluvad veel näiteks sisestusväljad (*input*).

**Tabel 2 - kontrolleriid**

|  |  |
|--|--|
| <p>Lülitusnupp</p> <p><code>&lt;paper-toggle-button&gt;</code></p>   |     |
| <p>Raadionupud</p> <p><code>&lt;paper-radio-group&gt;</code><br/><code>&lt;paper-radio-button&gt;</code></p> |     |
| <p>Icoon nupp</p> <p><code>&lt;paper-icon-button&gt;</code></p>  |    |
| <p>„Hõljuv nupp”</p> <p><code>&lt;paper-fab&gt;</code></p>   |    |
| <p>Liugnupp</p> <p><code>&lt;paper-slider&gt;</code></p>   |  |
| <p>Märkeruut</p> <p><code>&lt;paper-checkbox&gt;</code></p>  |  |

## 2.4 Aknad

Polymeriga saab luua mitmeid erinevaid infoaknaid. Dialoogid, väikesed teateaknad (*snackbars*, *toast*) ilmuvad eraldi akendena lehele, jättes algse veebilehe tahaplaanile. Paberelementide kolleksiooni kuuluvad kaks: *dialog* ja *toast/snackbar* (polymer, 2015).

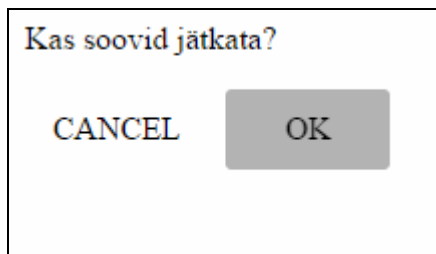
Dialoogi välja kutsudes tuleb kasutada `<paper-dialog>` elementi. Sellest ilmub aknake, mis võib sisaldada pealkirja, teksti, aktiivseid nuppe ja teisi kontrollereid. Pealkirja saab kirjutada *heading* omadusse (Koodinäide 13).

Aktiivseteks nuppudeks on võimalik lisada näiteks kaks nuppu, ühega saab toiminguid edasi teha ning teisega katkestatakse protsess. Kinnitava nupu vajutusega (nt OK, Salvesta)

minnakse protsessiga edasi, asetseb paremal pool. Vastupidist funktsiooni väljendav nupp viib eelmisele ekraanile tagasi, asetseb vasakul pool (Joonis 2).

```
<paper-dialog id="dialog" heading="Käivitan?"  
  transition="paper-dialog-transition-bottom">  
  <p>Kas soovid jätkata?.</p>  
  <paper-button dismissive>Cancel</paper-button>  
  <paper-button affirmative default>OK</paper-button>  
</paper-dialog>
```

### Koodinäide 13 – dialoogi akna loomine



### Joonis 2 – paper-dialog nuppudega

`<paper-toast>` element ilmub arvutil veebilehe alumisse äärde, mobiilsel platvormil vasakule poole alla. Teksti näitamiseks kasutatakse `text` atribuuti (Koodinäide 14). Nii dialoog kui ka `toast` on vaikimisi peidetud ning tuleb eraldi välja kutsuda, selleks kasutatakse `toggle` atribuuti. Mõlemaid elemente kasutatakse peamiselt lühiinformatsiooni kiireks edasiandmiseks.

```
<paper-toast id="toast" text="Sinu dokument on salvestatud ."></paper-toast>
```

### Koodinäide 14 – teateakna loomine

## 2.5 Nupud

Visuaalseid efekte saab ka lisada enda loodud kohandatud elementidele ning üldistele HTML elementidele nagu näiteks `<div>` -idele. Selleks saab kasutada `<paper-ripple>` ja `<paper-shadow>` elemente. Elemendi `<paper-ripple>` puhul tekitab nupp sisendi saades valguse

laiali valgumise efekti. Animatsioon, mis liigub väljapoole kuni nupu servadeni.

Et kasutada `<paper-ripple>` elementi tuleb ta defineerida efekti elemendi tütarelemendiks. Tuleb määrata absoluutne positsioon (`position: absolute`) ning mõõtmeteks vanemelemendi mõõtmed. Efekti saab käima panna ka ringikujuliselt, see tähendab et efekt liigub ringjalt väljapoole ning värvi lisamiseks tuleb kasutada CSS-i (Koodinäide 15).

```
paper-ripple {  
  color: green;  
}
```

#### Koodinäide 15 – värvi efekt nupu vajutusel

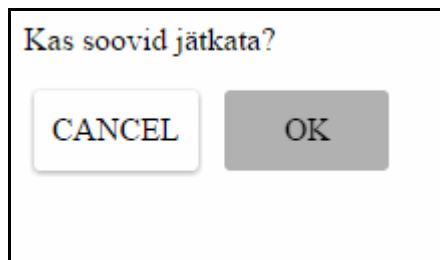
Kujundades paberelementi tuleb leida CSS selektor elemendi API dokumendist. Efekt asub vari DOM-is.

```
<p>Kas soovid jätkata?</p>  
  
<paper-button raised dismissive>Cancel</paper-button>  
  
<paper-button raised affirmative default>OK</paper-button>
```

#### Koodinäide 16 – paper-button loomine

Samuti saab lisada elemendile ka varju (z-telje suunas). Polymeri puhul võib varju suurus varieeruda 0st 5ni. Loetakse täisarve, väärtusega 5 on vari kõige laiem. Paberelementidel on varjuefekt sisseehitatud, näiteks kui `<paper-button>` deklareerida atribuudiga *raised* paistab see asetsevat kõrgemal kui ülejäänud veebileht (polymer, 2015). Nupule tekib automaatselt vari ning ka piirjooned, mis tekitavad tunde, et nupp asetseb muu taustaga võrreldes kõrgemal (Koodinäide 16). Ka varju värvi on võimalik muuta, mis on vaikimisi hall (Koodinäide 17). Järgneval joonisel on „Cancel” nupp varju efektiga ning „OK” nupp laialivalguva (*ripple*) efektiga. „OK” on värvitud, kuna animatsioon toimib kuni hiirevajutus kestab (Joonis 3).





**Joonis 3 – paper-button**

```
paper-button::shadow paper-ripple {  
  color: green;  
}
```

**Koodinäide 17 – nupu varju värvuse lisamine**

Omaloodud elementidel saab sama tulemuse saavutamiseks kasutada `<paper-shadow>` elementi (Koodinäide 18). Varju ulatust saab määrata andes z-teljel olevale ühikule väärtuse, 0 ehk pole varju kuni 5, mille puhul moodustub suurim võimalik vari (Joonis 4).

```
<template is="auto-binding">  
  <paper-shadow class="ruut" z="0">Z=0</paper-shadow>  
  ...  
  <paper-shadow class="ruut" z="5">Z=5</paper-shadow>  
</template>
```

**Koodinäide 18 – varju loomine elemendile**



**Joonis 4 – vari elementidel**

Autori näide on kättesaadav ka veebilehel <http://oimk.getmeteoric.com/vari.html> .

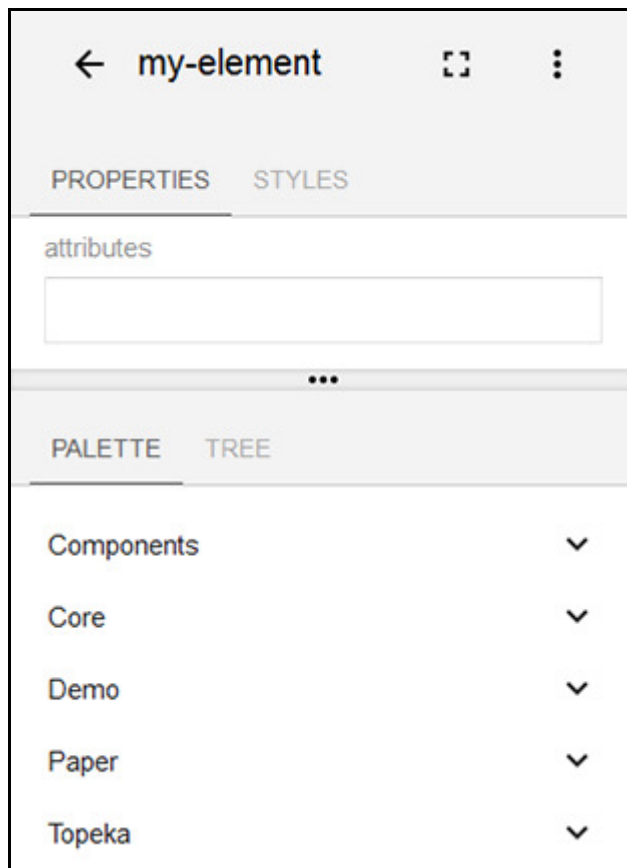
Transformatsioone kasutatakse elementidele ilmekuse andmiseks. Nende toeks kasutatakse `<core-animated-pages>` elementi, mis näitab ühte tütarelementi korraga ning pakub tuge sujuvateks transformatsioonideks tütarelementide või lehtede vahel. Alamlehtede vahetamise ajaks saab määrata mitu animatsiooni, näiteks alamlehelt nr.1 minnes alamlehele nr.2 liigub menüü üles aga vastupidi toimides liigub menüü paremale (Joonis 1). `<core-scroll-header-panel>` element laseb määrata tööriistariba käitumise kui muuta teksti suurust, liikuda taustade vahel.

## 2.6 Tööriist „Designer”

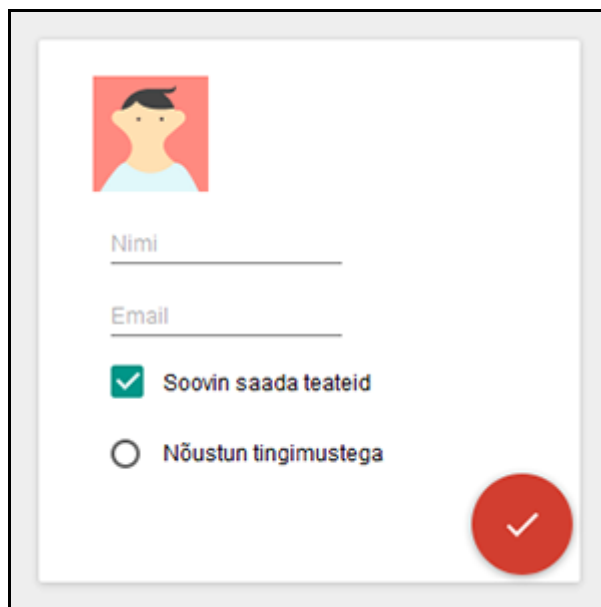
Polymeri arendajad on loonud ka tööriista nimega Designer, muutes nii elementidega tutvumise ja ka töötamise lihtsamaks.

Mõte seisneb selles, et ilma koodi kirjutama võib luua vajaliku nupu või miks ka mitte väiksema kogumi omavahel suhtlevatest elementidest. Tegemist on tüüpilise WYSIWYG (*What You See is What You Get*) lähenemisega. Kõik elemendid on olemas, need tuleb lihtsalt ekraanile tõmmata ning kokku sobitada vastavalt soovile (Joonis 5). Samuti saab kohandada omadusi ning elementide stiili. Kõigest kokku sobitatust genereeritakse kood, mis töötab ka eraldi veebilehel.

See on hea võimalus saada aimu, kuidas elemendid suhtlevad ning kuidas nad on pesastatud, kui tegemist on mitme aknavaatega elemendi sees. Muidugi lisandub ka katsetamisvõimalus ning kohe saab pildi asja visuaalsest küljest (Joonis 6). Autori kogemuse põhjal võib öelda, et suuremaid projekte antud tööriistaga on üpris keeruline korda saata. Väga suurt tähelepanu vajab erinevate elementide ühendamine, et kõik töötaks nii nagu peab.



**Joonis 5 - Designer tööriist**



**Joonis 6 - Designer tööriista näide**

Autori näide on vaadeldav ka veebilehel <http://oimk.getmeteoric.com/designer.html> .

### 3 Polymer project vahendite katsetamine

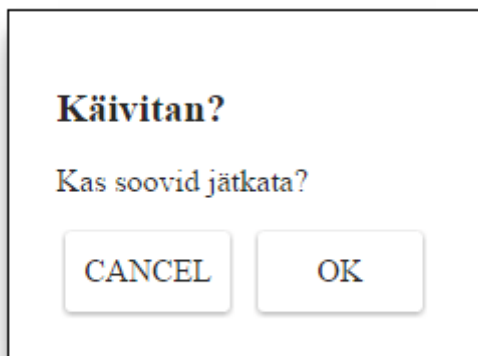
Autor on testinud erinevate kohandatud elementide loomist veebirakenduse kasutajaliidese jaoks. Käsitletakse kõige levinumaid elementide loomist ning näiteid nendest. Kõige levinumad on nad just oma suure kasutatavuse põhjal õpetustes ning tutvustustes. Autor käsitles elementide loomisel järgmisi: `<paper-dialog>`, `<paper-toast>`, `<paper-button>`, `<paper-checkbox>` ja `firebase` liidest.

Polymeri kodulehel on olemas põhielementide ja paberelementide kirjeldused, demod ning seletused. Osadel on juures ka koodinäited, kuid autorile tundusid need õppimisfaasis liiga pealiskaudsed. Allikatega tutvudes on selgeks saanud, et elementide töötamiseks on vaja importida kindlaid HTML faile loodud lehtedele, mis sisaldavad elementide toimimiseks vajalike vahendeid. Elementide juurde toodud koodinäidetes pole neist juttu. Mõnede puhul on lihtne välja mõelda, mis faili parajasti importida on vaja või siis aitab veebilehitseja konsool sellega aga viiteid kui selliseid, milliseid täpselt, ei ole. Muidugi on vaja importida kõikide nende elementide failid, mis on kirjutatud enda loodud faili. Lisaks veel kindlasti võimalike probleemide vältimiseks `polyfill webcomponents.js`, edasi aitab küsimuste korral internet.

#### 3.1 Dialoogiakna loomine

Dialoog või teateaken on enamuse rakenduste kasutajaliidese tarvis ning sellise elemendi kohandamine võimaldab muuta oma rakendust pisut väljapaistvamaks. Dialoogi kast ilmub nähtavale nupu vajutuse peale, ilmudes olemasoleva lehe ette, kõik mida kasutaja enne nägi on nüüd tagaplaanil (Joonis 7). Vaikimisi on ta peidetud. Välja kutsumiseks võib kasutada *toggle* funktsiooni, mis ta uuesti peidab või toob nähtavale. Dialoogi aknaga katsetades tuli välja asjaolu, et `onclick` ja `on-click` on kaks eraldi sündmust. Esimene on Javascript'is element ning Polymer teda ei tunnista, kuigi element ilmus ekraanile Firefox'is kuid mitte Chromes. Vahetades välja see `on-click` vastu töötas element mõlemas veebilehitsejas.

Autori näide on kättesaadav ka veebilehel <http://oimk.getmeteoric.com/dialog.html>.



Joonis 7 - paper-dialog aken

### 3.2 Teateakna loomine

Kaasaegses tarkvaras on levinud väikeste teadete kuvamine, näiteks annab arvuti ekraani all paremas nurgas teada kui traadita interneti leviala ei leitud, samamoodi kui aku tühjeneb peale mingit piiri. Veebileheküljel on selleks teateaknaks *toast*. *Toast* ilmub vaikimisi ekraani alla vasakusse serva, teda on võimalik ka mujal paigutada. Ta näitab samuti mingit teadet, mingi hetke vältel või kuni kasutaja vajutab kuskil mujal ekraanil. Loomulikult saab määrata teate ekraanile jäämise aega (*duration*). Võimalikud on erinevad kujundused ning teatele saab lisada ka nupu (Joonis 8). *Toast* ilmub erkaanile nupuvajutuse tagajärjel, vaikimisi on ta peidetud.



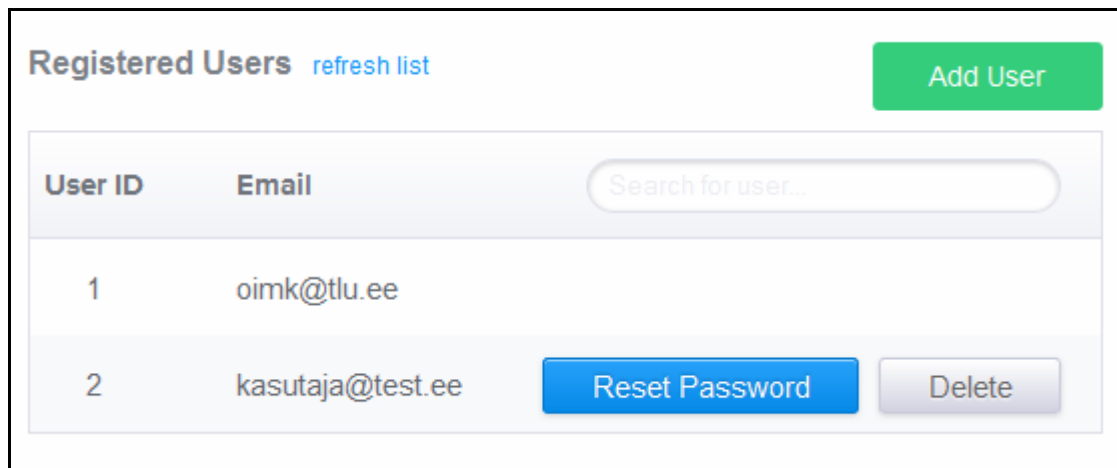
Joonis 8 - erinevad kujundusmeetodid teateakendele

Autori näide on kättesaadav ka veebilehel <http://oimk.getmeteoric.com/toast.html>.

### 3.3 Firebase andmebaasi liidese loomine

Olukordades, kus on tarvis andmebaasi näiteks, süsteemi kasutajate tuvastamisel, sisselogimisel, pakub Polymer Firebase andmebaasi liidest.

Autor kasutaski Firebase elementi süsteemi sisselogimise näite loomisel. Selleks tuli teha Firebase kodulehel kasutaja ning ühendada sinna külge omakorda loodud veebilehe kasutajad (Joonis 9). Konto loomisega saab oma isikliku andmebaasi aadressi, mis tuleb loodud koodi kopeerida. Sinna talletuvad veebilehe andmed, mida on võimalik ka Firebase kodulehel vaadata ning muuta. Nii on kohe olemas ka andmebaas, mida pole vaja ise luua, mis teeb töö kiiremaks ning lihtsamaks. Element kontrollib, kas kasutaja on olemas Firebase andmebaasist.



Joonis 9 - kasutajate lisamine Firebase kodulehel

Samuti on see juba kellegi teise poolt koostatud element, mis tõestab kohandatud elementide laiaulatuslikku kasutamisevõimalust, millest ka eespool juttu tehti. Kindlasti ei pea kasutama just Firebase elementi ning soovi korral saab andmebaasi ka ise luua. Autor soovis välja tuua näite juba teiste poolt loodud elementide sidumisest enda tööga. See on ka üks aspekt, mida Polymer välja reklaamib ning teha julgustab.

Antud juhul on sisselogimisvahendid seotud `<paper-dialog>` elementi kasutades. Nupu vajutuse peale avaneb sisselogimisaken sisestusväljadega. Nupude tegemisel kasutas autor `<paper-button>` meetodit. Sisestusväli muutub kasutamisel aktiivseks, mis kujundusliku poole pealt mõjub silmatorkavalt (Joonis 10).

Autori tehtud näide on kättesaadav ka veebilehel <http://oimk.getmeteoric.com/memo.html> .

Sisselogimise katsetamiseks saab kasutada järgmisi tunnuseid:

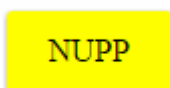
- Email – [kasutaja@test.ee](mailto:kasutaja@test.ee)
- Password – test123



Joonis 10 - sisselogimisaken aktiivse väljaga

### 3.4 Nupu loomine

Nupud on kasutajaliidese kõige tavalisemad elemendid, millega tehakse valikuid, määratakse väärtuseid ning sisendeid. Nuppudega on asi lihtne, lood elemendi ning ta on olemas. Tavapärasest eriliseks teeb nad asjaolu, et vajutuse peale toimub vajutuskohast värvi laiali valgumine kogu nupu ulatuses (Joonis 12) ning vajutuse lõppedes taastab oma algse kujunduse (Joonis 11). Laialivalguv värv on vaikumisi hall, kuid seda saab muuta meelepärasemaks kasutades *ripple* efekti. See on pärit Polymeri kujundusmeetodite valikust.



Joonis 11 – passiivne seisund



Joonis 12 - aktiivne seisund

### 3.5 Märkeruudu loomine

Valikute tegemisel on kasutajaliideses tavalised märkeruudud (*checkbox*). Märkeruute on mugav kasutada ja kasutataksegi kui kasutaja võib valida mitu üksteist mitte-välistavat varianti antud hulgast ehk valikuid saab teha mitu. Märkeruutel on kolm seisundit - märgitav, vaikimisi juba valitud ning väljalülitatud olek ehk ruutu ei saa soovi korral märkida (Joonis 13). Ka siin saab samamoodi muuta valitud ruudu värvimuutust.

- WiFi**  
Teavita mind WiFi levialast
- Vaikimisi valitud**
- Ei saa valida**

Joonis 13 - paper-checkbox

Autori näide on kättesaadav ka veebilehel <http://oimk.getmeteoric.com/kujundus.html> .



## Kokkuvõte

Käesoleva bakalaureusetöö lähtus veebirakenduste loojate vajadusest muuta oma rakendusi omanäolisteks ning luua selle jaoks üha huvitavamaid nuppe, dialoogiaknaid ja teisi kasutajaliidese elemente. Aastast 2014 eksisteerib spetsiaalselt sellel otstarbel loodud tehnoloogia Polymer project.

Bakalaureusetöö eesmärgiks oli tutvustada uue tehnoloogia Polymer project olemust, elementide loomist ning katsetada kas tehnoloogia pakub lubatud - lihtsaid, mugavaid ning töökindlaid kohandatud elemente.

Polymer'i põhimõte on uue tehnoloogia alusel luua endale meelepäraseid kohandatud elemente, mida saab kokku sobitada ja kasutada teiste elementidega. Kuna süntaks sarnaneb HTML ja CSS'ile ei vaja teemast arusaamine liigseid pingutusi, pigem nõuab harjumist pidev elementfailide importimine. Polymer project tehnoloogia on toetatud kõigi tuntumate veebilehitsejate uusimates versioonides. Tänu polyfillidele on tagatud ka tagasiühilduvus vanemate veebilehitsejatega.

Autor tutvus Polymer project spetsifikatsiooniga ning katsetas olulisemaid võimalusi. Tuleb tõdeda, et tõesti oli mugav ning koguni lihtne. Eriti kasulik oli luua nuppe, lihtsa liigutusega saab lisada värvi ja varju kohe nupule juurde. See kõik otse elementi, ilma CSS failita. Mõne elemendi juures tuli üllatuseks kui ainuke viga miks veebileht ei näidanud soovitud, oli see, et õige import fail oli puudu. Tutvudes Polymeri kodulehega ning vaadeldes antud näiteid siis peaks seda eraldi meeles pidama, sest elementide juures seda rohkem välja toodud pole. Kui tehnoloogiaga alustada on allalaaditud failide seas olemas kõik vajalik erinevatele elementidele. Puudujääke saab juurde GitHub'ist. Muidugi tuleb ka meeles pidada, et lisaks elementide failidele on vaja omaloodud elemendi fail veebilehele importida. Väiksemateks ettevõtmisteks saab elemente disainida kasutades Designer tööriista. Autor kasutas näidetes enam kasutatud elemente, töös kasutatud näiteid saab vaadelda veebilehel <http://oimk.getmeteoric.com/>.

Bakalaureusetöö eesmärgid said täidetud. Autor tutvus tehnoloogiaga ning selle võimalustega. Elementide loomine on lihtne, mugav ja ka üpriski põnev ettevõtmine.

Kasu võiks bakalaureusetööst olla arendajatel ja disaineritel, kes soovivad luua enda kohandatud elemente ning ka kõigil, kes leiavad Polymeri huvitava enda jaoks.

Tehnoloogial on kindlasti juba praegu potentsiaali. Bakalreusetöö valmimise hetkel väljas avalikustati versioon 0.8, mida täiendatakse vastavalt kasutajate tagasisidele ning arvatavasti kasvab potentsiaal tuleva versiooniga 1.0 veelgi. Kahjuks võivad mitmed siin töös toodud koodinäited (versioon 0.5) uuema versiooniga mitte toimida.

## Kasutatud kirjandus

e-teatmik. (kuupäev puudub). *API*. Loetud aadressil: <http://vallaste.ee/> külastatud 15.03.2015

Polymer. (kuupäev puudub). *Understanding Polymer*. Loetud aadressil: <https://www.polymer-project.org/0.5/docs/start/everything.html> külastatud 15.03.2015

Polymer. (kuupäev puudub). *About custom elements*. Loetud aadressil: <https://www.polymer-project.org/0.5/platform/custom-elements.html> külastatud 15.03.2015

stackoverflow. (2012). *What is two way binding?* Loetud aadressil: <http://stackoverflow.com/questions/13504906/what-is-two-way-binding> külastatud 16.03.2015

Microsoft Research. (1998). *Declarative Event-Oriented Programming*. Loetud aadressil: <http://research.microsoft.com/apps/pubs/default.aspx?id=69665> külastatud 16.03.2015

Polymer. (kuupäev puudub). *About shadow DOM*. Loetud aadressil: <https://www.polymer-project.org/0.5/platform/shadow-dom.html> külastatud 24.03.2015

Polymer. (kuupäev puudub). *Layout elements*. Loetud aadressil: <https://www.polymer-project.org/0.5/docs/elements/layout-elements.html> külastatud 05.04.2015

Polymer. (kuupäev puudub). *Material design with Polymer*. Loetud aadressil: <https://www.polymer-project.org/0.5/docs/elements/material.html> külastatud 05.04.2015

Mozilla Developer Network. (kuupäev puudub). *Document Object Model (DOM)*. Loetud aadressil: [https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model) külastatud 10.04.2015

GitHub. (kuupäev puudub). *Browser Support*. Loetud aadressil: <https://github.com/WebComponents/webcomponentsjs#browser-support> külastatud 17.04.2015

CSS – Tricks. (kuupäev puudub). *A guide to SVG Animations (SMIL)*. Loetud aadressil: <https://css-tricks.com/guide-svg-animations-smil/> külastatud 28.04.2015

e-teatmik. (kuupäev puudub). *Script*. Loetud aadressil: <http://vallaste.ee/> külastatud

28.04.2015

Glazkov, D. (2011, 14.jaanuar). *What the Heck is Shadow DOM?* [ajaveebi postitus]. Loetud aadressil: <http://glazkov.com/2011/01/14/what-the-heck-is-shadow-dom/> külastatud 29.04.2015

Sharp, R. (2010, 8.oktoober). *What is a Polyfill?* [ajaveebi postitus]. Loetud aadressil: <https://remysharp.com/2010/10/08/what-is-a-polyfill> külastatud 29.04.2015

Mozilla Developer Network. (kuupäev puudub). *MutationObserver*. Loetud aadressil: <https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver> külastatud 29.04.2015

Mozilla Developer Network. (kuupäev puudub). *WeakMap*. Loetud aadressil: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/WeakMap](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/WeakMap) külastatud 29.04.2015

Unreal Engine. (kuupäev puudub). *Animation Sequence*. Loetud aadressil: <https://docs.unrealengine.com/latest/INT/Engine/Animation/Sequences/index.html> külastatud 29.04.2015

Formz. (kuupäev puudub). *Animation Group*. Loetud aadressil: [http://www.formz.com/manuals/animationmanual/!SSL!/WebHelp/21240\\_Animation\\_Group\\_Tool.html](http://www.formz.com/manuals/animationmanual/!SSL!/WebHelp/21240_Animation_Group_Tool.html) külastatud 29.04.2015

Mozilla Developer Network. (kuupäev puudub). *AnimationPlayer*. Loetud aadressil: <https://developer.mozilla.org/en-US/docs/Web/API/AnimationPlayer> külastatud 29.04.2015

Mozilla Developer Network. (kuupäev puudub). *Window.requestAnimationFrame()*. Loetud aadressil: <https://developer.mozilla.org/enUS/docs/Web/API/window/requestAnimationFrame> külastatud 29.04.2015

e-teatmik. (kuupäev puudub). *SMIL*. Loetud aadressil: <http://vallaste.ee/> külastatud 29.04.2015

polymer. (kuupäev puudub). *Using core icons*. Loetud aadressil: <https://www.polymer-project.org/0.5/docs/elements/icons.html> külastatud 29.04.2015

Mozilla Developer Network. (kuupäev puudub). *Document.registerElement()*. Loetud

aadressil: <https://developer.mozilla.org/en-US/docs/Web/API/Document/registerElement>  
külastatud 29.04.2015

Polymer. (kuupäev puudub). *About this release*. Loetud aadressil: <https://www.polymer-project.org/0.8/> külastatud 03.05.2015

Dale, T. (2013, 24.mai). *Evergreen Browsers [ajaveebi postitus]*. Loetud aadressil: <http://tomdale.net/2013/05/evergreen-browsers/> külastatud 03.05.2015

W3C. (2014, juuni). *Web Animations 1.0*. Loetud aadressil: <http://www.w3.org/TR/web-animations/#introduction> külastatud 03.05.2015

W3C. (2014, juuni). *Animations*. Loetud aadressil: <http://www.w3.org/TR/web-animations/#animations> külastatud 03.05.2015

Õim, K. (kuupäev puudub). *Näited*. Loetud aadressil: <http://oimk.getmeteoric.com/> külastatud 03.05.2015

## Summary

Title: Creating Customized Elements for Web Application User Interfaces Using Tools of Polymer Project

This Bachelor Thesis focused on the problem for the web developers need to create interesting user interface elements such as buttons, toast, dialog window and others. Since 2014 the problem is being solved with Polymer project which is created for that very reason.

The purpose of this Bachelor Thesis was to introduce new technology in custom elements and web design – the Polymer project. Also to figure out does the technology truly provide us with simple, easy to use custom elements.

Polymer project technology is based on building powerful and reliable custom elements. You can use the elements with Polymer or with other HTML elements. It's not difficult to adjust because the syntax is very similar to HTML and CSS. Polymer is determined to work on all the browsers, also known as evergreen browsers. For backwards compatibility in older browsers they have polyfills to do the job. User has to download a package from homepage and is ready to get started. Polymer is created by Google developers, who are very eager about the subject. It was introduced to the public at the Google I/O software developer focused conference in 2014. It is held every year to introduce the latest upgrades in software development.

To analyze the technologies first release version 0.5, the author acquainted with documentation and practiced the most common elements first hand. It was easy to create buttons and at the same time add shade and coloring all in one HTML file. No CSS file needed. Surprisingly a lot of problems were caused by not importing the right HTML file to the page or forgetting to import it at all. It is easy to forget because there is no mention of this in element tutorials on the Polymer page, it's found on "Getting started with Polymer" (polymer, 2015). Otherwise all the files were in the starting package, the one user downloads to get all the resources to start creating with Polymer. For smaller projects it is useful to design elements using the Designer Tool, which can be found on the homepage. Although Designer Tool is not a great choice for a beginner to do a sizable project. Linking the properties to each other may be difficult. The examples made by the author are displayed on <http://oimk.getmeteoric.com/>.

The purpose was fulfilled. The author agrees with the developers, it is simple and handy to use these elements. To make them work for yourself and your website. The trick is not to forget the simple rules how they work.

Paper in hand should be useful to all web designers and developers who are interested in making their own custom elements to improve their websites or people who just find Polymer interesting for themselves. At the moment the Google developers are waiting feedback from users about the version 0.8 to upgrade it to 1.0. The technology truly has potential and with the coming version 1.0 it will definitely grow. Sadly some of the code examples in here may not work with the new version 0.8.