Tallinna Ülikool

Digitehnoloogiate Insituut

AngularJS raamistiku õppematerjal

Seminaritöö

Autor: Sander Leetus

Juhendaja: Jaagup Kippar

Tallinn 2015

Autorideklaratsioon

Deklareerin, et käesolev seminaritöö 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.

.....	.....
(kuupäev)	(autor)

Sisukord

Sissejuhatus

Käesoleva seminaritöö eesmärgiks on luua AngularJS (edasipidi Angular) veebiraamistiku õppematerjal. Angular on loodud dünaamiliste veebirakenduste jaoks, mis mallina kasutab HTMLi ning võimaldab selle süntaksit laiendada vastavalt rakenduse vajadusele.

Autor on motiveeritud Angulari teemal kirjutama, kuna puudub eesti keeles õppematerjali olemasolu ning loodab suurendada sellega Angulari populaarsust.

Õppematerjal on eelkõige loodud õpilastele, kellel puudub kokkupuude Angulari raamistikuga. Eelduseks õppematerjali läbimiseks oleksid HTML ja CSS tundmine heal tasemel ning JavaScripti tundmine rahuldaval tasemel.

Autori arvates on õppematerjalis kõige olulisemad Angulari komponendid, mida läheb vaja, et alustada Angulari rakenduse loomisega. Samuti on kõik töös olevad koodinäited ülesse pandud, autori poolt loodud veebilehele http://www.tlu.ee/~sander12/angular.

1 Angulari õpetused

Angulari kohta on mitmeid inglise keelseid õpetusi sealhulgas ka raamatuid ja kogukonna poolt modifitseerituid koodilahendusi. Õppematerjali loomise käigus kasutan peamiselt kolme veebipõhist Angulari õpetust, nendeks on Angulari oma dokumentatsioon, w3schools ja Tutorialspoint. Angulari oma dokumentatsioon on väga hästi loodud, kus on komponendid, mille kohta kõige täpsema lahenduse leiab just sealt.

1.1 Angulari dokumentatsioon

Angulari dokumentatsioon on hästi loodud. Dokumentatsioon on lahti löödud moodulitesse, mis koosnevad eri komponentidest. Komponentide kasutusvõimalused on dokumentatsioonis, koos koodinäidetega, hästi välja toodud. Dokumentatsioonis on olemas isegi õpetav juhend, kus saab paljusid komponente kasutada ja õppida selle abil. Angulari lehel on ka võimalik koodi muuta ja kohe näha, mis kood teeb „Edit in Plunker" nupu abil. Lehel on olemas ka arendaja juhend, kus asuvad olulisemad Angulari komponendid (vt joonis 1).

Joonis 1: Arendaja juhend

Angulari õpetus koonseb järgnevatest peatükitest (vt joonis 2).

Joonis 2: Angulari õpetus

Angulari juhendit kasutades võib tekkida raskusi, kuna selle kasutamiseks läheb vaja algteadmisi kuidas kasutada Git[1] versiooni süsteemi ja selle käske. Koodinäited ei pruugi ka kõige lihtsamad olla, samas on need väga hästi ära seletatud. Ei soovita koodinäidetega siit alustada, kuid Angulari komponentide kohta lugemiseks on väga hea koht.

1.2 w3schools

w3schools (edasipidi W3) on eriti hea veebipõhine õpetus, kuna peale selle, et see sisaldab näiteid Angulari kohta on seal ka „Try it yourself" nupp, kus saab koheselt koodinäite tööle panna ja modifitseerida seda kohapeal.

W3 lehel on loodud järgnevad õpetused ja nende kohta näited (vt joonis 3).

1 https://git-scm.com/

Joonis 3: W3 õpetus

W3 lehel on väga algsed ja lihtsad koodinäited. Seletused on väga lihtsustatud, ei ole kasutatud liigseid väljendeid. Iga peatüki juures on väga palju näiteid ja kõiki on võimalik muuta ja näha mis teevad (vt joonis 4). Võimalik, et antud kolmest õpetusest kõige parem leht koodinäidete jaoks.

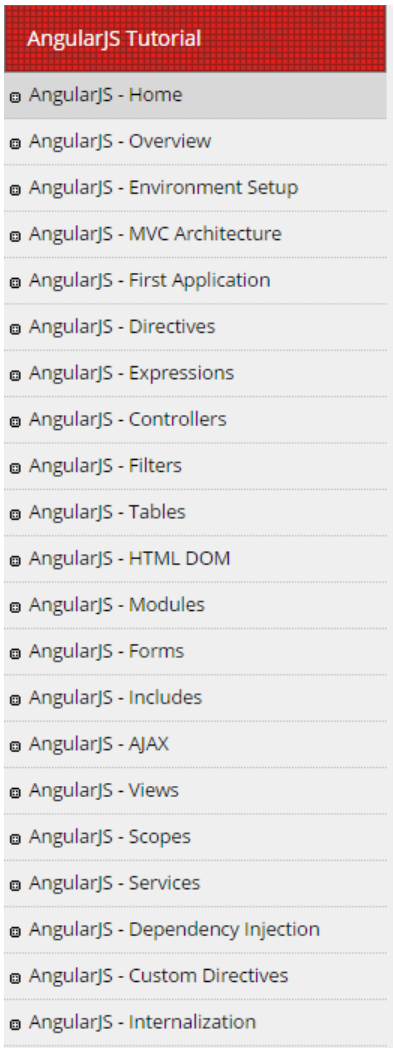
Joonis 4: W3 Koodinäite muutmise leht

Samas kuna koodinäited ja nende seletused on hästi lihtsustatud, siis jääb antud lehest arvatavasti väheseks, et iga mooduli kohta piisavalt infot saada.

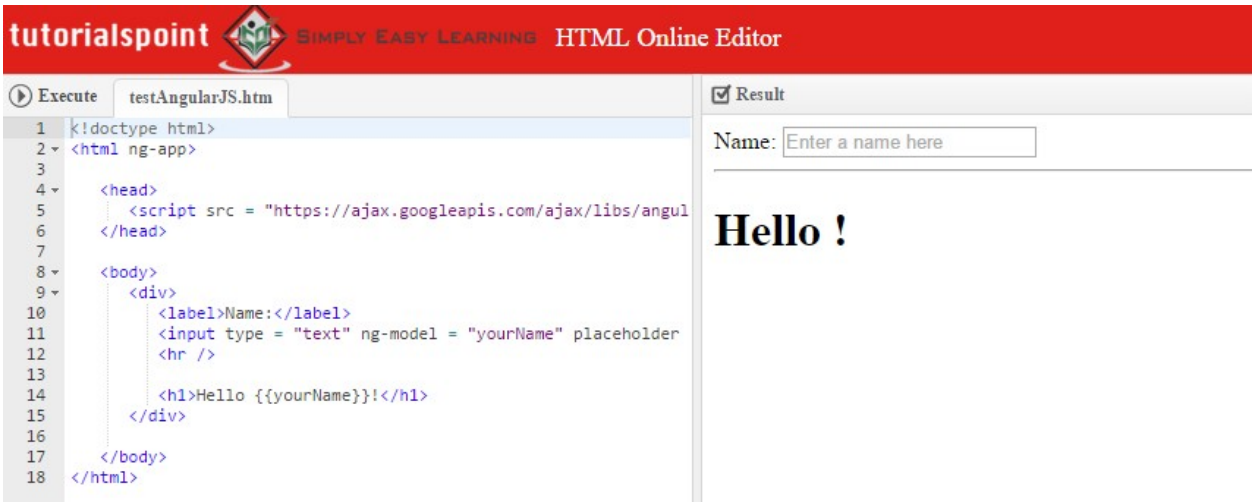
1.3 Tutorialspoint

Sarnaneb W3 õpetusega, kuid pikemad koodinäiteid ja seletusi, samas koodinäited on ka keerukamad, mis on hästi osadeks seletatud ja ülesehitus on parem. Tutorialspointis on sarnaselt W3 koodinäite redigeerimis võimalus „Try it" nupu abil (vt joonis 6). Samuti on peatüki lõpus ka koodinäite tulemus olemas, kus saab koheselt proovida mida rakendus teeb.

Võimalik, et parim õpetust antud kolmest, kuna on olemas põhjalikud koodinäited, mis on peatüki sees lahti võetud ja ära seletatud ning peatüki lõpus on olemas terve koodilõik koos rakendusega olemas. Samas lehepeal on võimalik antud kood lahti võtta ja muuta ning koheselt ka tulemust näha.



Joonis 5: Tutorialspoint õpetus



Joonis 6: Tutorialspointi koodinäite muutmise leht

2 Angular

Angular on struktuurne raamistik, mis on loodud dünaamiliste veebirakenduste jaoks. Antud raamistik võimaldab kasutada HTML malli ja laiendada HTML-i süntaksit, mis väljendab rakenduse komponente selgelt ja lühidalt. Kõik toimub veebilehitseja sees, mis teeb Angularist ideaalse partneri iga võimaliku serveritehnoloogiaga. Angulari rakenduses mallide täitmise töö infoga liigutatakse serverilt kliendile, mille tagajärel on süsteem paremini struktureeritud. All on põhifunktsioonid välja toodud, mida selleks kasutada.

Angulari loomisega alustati aastal 2009 kahe arendaja poolt, Misko Hevery ja Adam Abrons. Algselt oli antud kahe arendaja projekt nimetatud *GetAngular*, mis pidi olema vahend, et veebidisainerid saaksid suhelda samaaegselt *frontend*- ja *backend*-iga. Hevery alustas töötamist projektiga *Google Feedback* koos Googlega. Koos kahe teise arendajaga kirjutasid 6 kuu perioodiga nad 17000 rida koodi. Kuna kood oli niivõrd pikk ja tekkis raskusi testmisega ja muutmisega, siis Hevery tegi kihlveo, et suudab rakenduse ümberkirjutada, kasutades *GetAngular*, seda kahe nädalaga. Hevery küll kaotas selle kihlveo, kuna kulus tal aega 3 nädalat, kuid ta oli võimaline koodi lühendama 17000 realt 1500 reale. Tänu Hevery edule hakkas Angulari arendus ka kiirenema. (Austin)

2.1 Avaldised

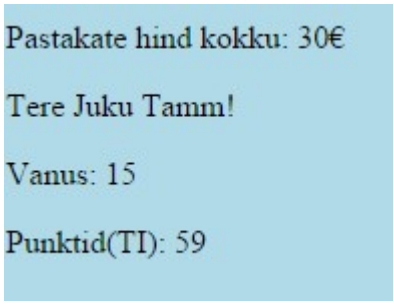
Angulari avaldised on JavaScripti sarnades koodilõigud, mis on pandud loogelistesse sulgudesse `{{ avaldis }}`. Angulari avaldised on sarnased JavaScripti omadega, järgnevate erinevustega (Google, kuupäev puudub).

- Kontekst: JavaScriptis avaldise väärtus leitakse globaalse *window* vastu. Angularis kasutatakse selleks *scope* objekti.
- Andestav: Kui JavaScript üritab hinnata defineerimata omadusi, genereerib see *ReferenceError* või *TypeError*. Angularis samas on see lihtsalt kas *undefined* või *null*.
- Puuduvad käsuvoo laused: Ei ole võimalik kasutada tingimuslauseid, silmuseid ning erindeid.
- Puuduvad funktsioonide deklareerimised: Ei ole võimalik deklareerida funktsioone

avaldistes, isegi mitte *ng-init* direktiivis.

- JavaScripti avaldised: Ei ole võimalik kasutada JavaScripti avaldisi Angulari avaldistes.
- Ei komasid ega *void* operaatoreid: Ei saa kasutada , või *void* Angulari avaldises.
- Filtrid: Filtreid on võimalik kasutada avaldiste sees, et vormistada info enne selle kuvamist.

Kui on soov jooksutada keerulist JavaScript koodi, peaks kontroller tegema selle meetodiks ja kutsuma selle vaatest. Samas kui on soov *eval()* Angulari avaldist ise, saab ka kasutada *$eval()* meetodit. (Google, kuupäev puudub)



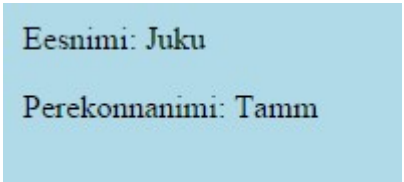
Joonis 7: Kõiki avaldisi koos kasutades

2.2 Andmete sidumine

Andmete sidumine on Angulari rakendustes info automatiseeritud sünkroniseerimine mudeli ja vaate komponentide vahel. Angulari moodus andmete sidumist rakendada laseb sul mudelit käsitleda, kui *single-source-of-truth* sinu rakenduses. Vaade on koguaeg mudeli projektsioon, kui vaade muutub, muutub ka mudel ja vastupidi. (Google, kuupäev puudub)

Enamik malle siduvad andmeid vaid ühes suunas, nad ühendavad mallide ja mudelite komponendid kokku vaatesse. Peale ühendamist ei ole mudeli muutumisi võimalik automaatselt vaates uuendada (vt joonis 8). See tähendab, et arendaja peab kirjutama koodi, mis koguaeg sünkroniseerib vaate mudeliga ja mudeli vaatega. (Google, kuupäev puudub)

Angulari mallid töötavad, aga teistmoodi. Kõigepealt mall, mis on HTML koos täiendavate direktiividega on kompileeritud veebilehitsejasse (Google, kuupäev puudub). Kõik muudatused vaates on koheselt peegeldatud mudelis ja vastupidi (vt joonis 9).

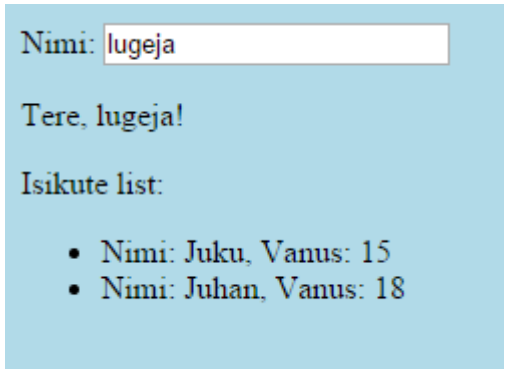


Joonis 8: Ühte pidi andmete sidumine

Joonis 9: Kahte pidi andmete sidumine

2.3 Direktiivid

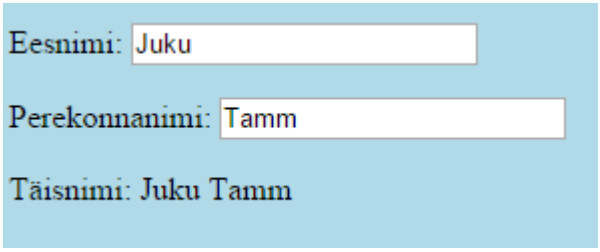
Direktiivid (*Directives*) on markerid dokumendi objekti mudeli (edaspidi DOM) elemendis, mille hulka kuuluvad atribuudid, elemendi nimed, kommentaarid või CSS klass, mis annavad Angulari HTML kompileerijale teada, et lisada ettemääratud käitumine DOM elemendile. Direktiivid lasevad Angulariil laiendada HTML-i, eesliitega *ng-*. Lähemalt on kirjas mõne direktiivi kohta allpool koodinäidete juures. (Google, kuupäev puudub)



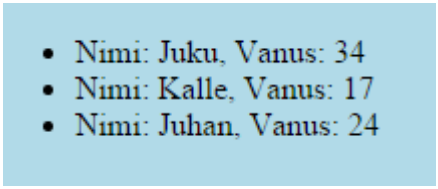
Joonis 10: Nelja direktiivi kasutus koodinäites

2.4 Kontrollerid

Angulari rakendus tugineb kontrolleritel, mis juhivad andmevoogu rakenduses. Kontrollerid defineeritakse direktiivi *ng-controller* abiga. Kontrolleril võivad olla ka meetodid, funktsioon muutujana. Kontrolleri klass sisaldab rakenduse tegevuse loogikat, mis lisab *scope* objekti funktsioonid ja väärtused. Suuremates rakendustes hoitakse kontrollereid tavaliselt eraldi failides. (w3schools, kuupäev puudub)



Joonis 11: Esimese ja teise kontrolleri koodinäide



Joonis 12: Kontrolleri faili eraldi sisse lugemine ja selle kasutamine

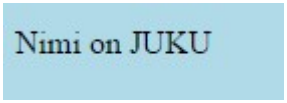
Mõlemad koodinäited (koodinäide 11 ja 12) annavad sama tulemuse, kuid koodinäites 12 on kasutatud funktsiooni täisnime saavutamisel. Koodinäites 11 on täisnimi lihtsalt avaldise abil saavutatud (Joonis 11).

2.5 Filtrid

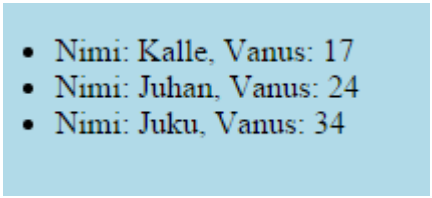
Filtrid loovad vormingu avaldise väärtusest, mis kuvatakse kasutajale. Filtreid saab kasutada avaldistes või direktiivides kasutades | (püstkriipsu) sümbolit.

Järgnevad filtrid on enim kasutatud (Tutorialspoint, kuupäev puudub).

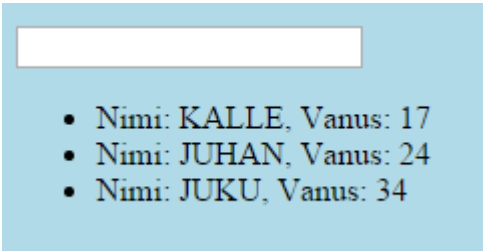
- *uppercase* – Konverdib teksti suurtähtedeks (vt joonis 14).
- *lowercase* – Konverdib(muudab) teksti väiketähtedeks.
- *currency* – Muudab teksti valuuta vormingusse.
- *filter* – Filtreerib massiivist vasted, vastavalt kriteeriumile (vt joonis 13).
- *orderBy* – Paneb massiivi järjekorda, vastavalt kriteeriumile (vt joonis 15).



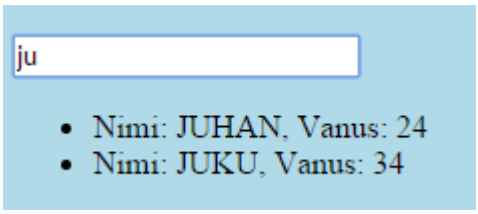
Joonis 14: Filtri uppercase kasutus



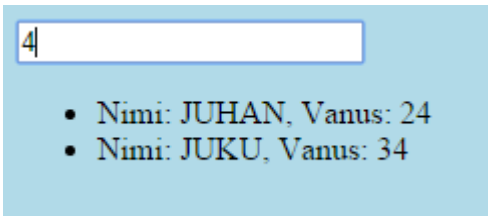
Joonis 15: Filtri orderBy kasutus



Joonis 13: Filtri filter kasutus



Joonis 16: Filtri filter kasutus



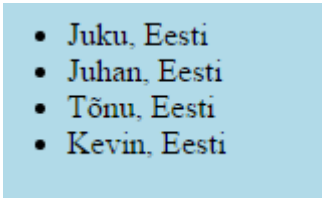
Joonis 17: Filtri filter kasutus

Midagi koodis määramata oskab rakendus filtreerida, nii nime kui ka vanust ja seda ühe ainult ühte *input* välja kasutades (vt joonis 16, 17).

2.6 Http

Angular on ka varustatud $http kontrolliga, mis töötab teenusena, et lugeda andmeid serverist. Server küsib andmebaasist vajalikud andmed. Töötamiseks peab info jõudma Angularini JSON formaadis (Tutorialspoint, kuupäev puudub). Angulari http kasutades on puudusi, kuna Angular

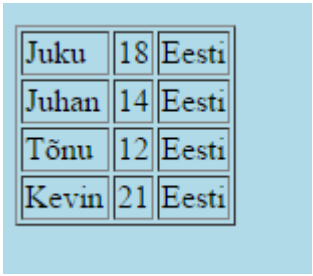
ei oska midagi sellega teha, kui info ei jõua JSON formaadis rakendusse. Seega kui on soov muus formaadis faile lugeda peab selleks *jQuery* abi kasutama.



Joonis 18: Http abil faili lugemine ja listi loomine

2.7 Tabelid

Tabelite andmed on tavaliselt korduvad, seega *ng-repeat* direktiivi kasutamine võimaldab tabeleid luua kergelt.



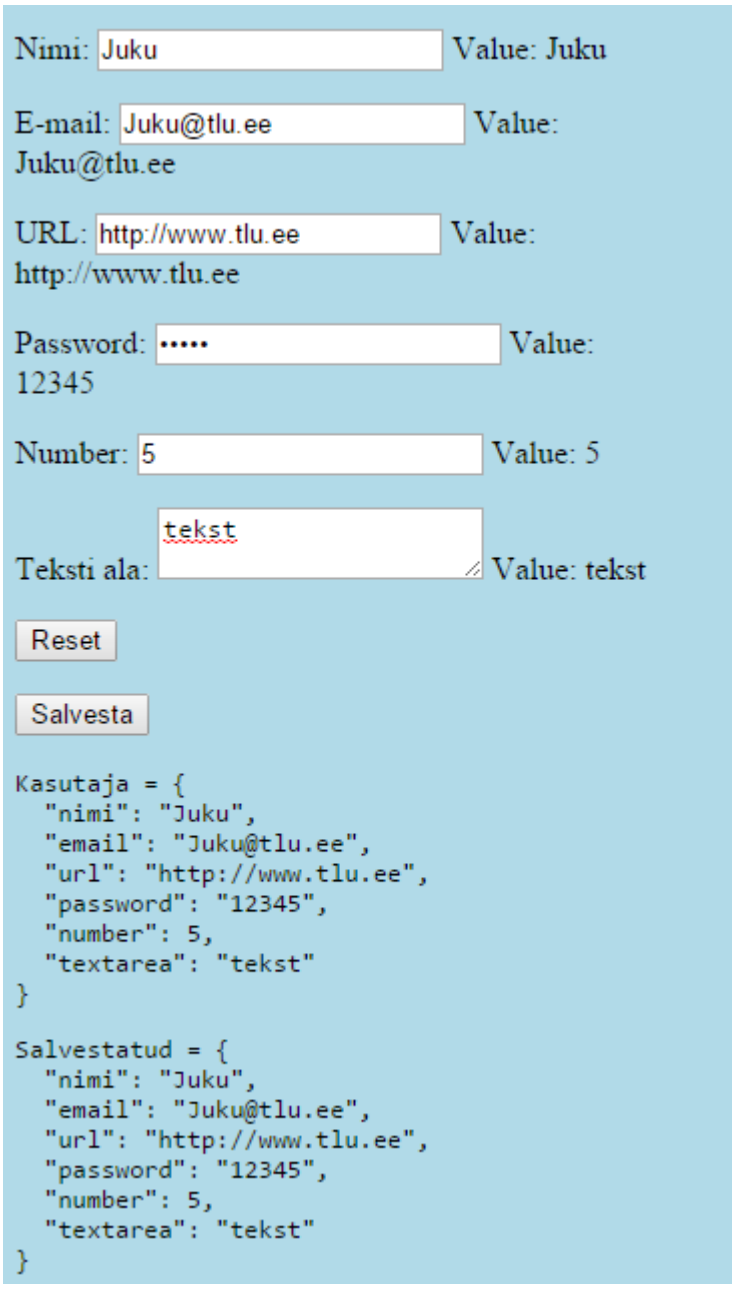
Joonis 20: Tabeli loomine

Joonis 19: Tabeli loomine ja selle filtreerimine orderBy filtriga

Filtri peatükis kasutatud *orderBy* filtrit on kasutatud ka siin (vt joonis 19).

2.8 Vormid

Vormide abil on võimalik kasutajal sisestada andmeid. Sisestatud andmeid on võimalik ka salvestada massiivi (vt joonis 21). Vormid pakuvad valideerimisteenust, mis tähendab, et kasutajale on võimalik teada anda valest sisendist (Bresolin, 2015). See pakub paremat kasutaja kogemust, kui ainult serveripoolne valideermine, kuna kasutaja saab kohest tagasisidet, kuidas viga parandada. Samas seda võidakse kergesti ära kasutada, seega serveripoolne valideerimine on siiski vajalik.

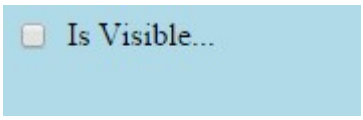


Joonis 21: Vormi loomine ja andmete salvestamine massiivi

2.9 Animatsioonid

Angular on vaikimisi animatsioonid välja lülitanud, nende kasutamiseks tuleb teek rakendusse lisada ning see ka moodulisse laadida. (ng*Animate* moodul toetab CSS-baasil animatsioone ning JavaScript-baasil animatsioone). Animatsioonid Angularis põhinevad täielikult CSS klassidel, kui veebilehel on CSS klass on HTML elemendi külge kinnitatud on sellele võimalik animatsiooni lisada.

Joonis 22: Animatsiooni abiga nähtav



Joonis 23: Animatsiooni abiga nähtamatu

3 Angulari komponentide kood

3.1 Juhend Angulari ülesseadmiseks

Kõigepealt tuleb luua uus HTML fail, kuhu kirjutatakse kogu koodiosa (vt koodinäide 1).

```
<!DOCTYPE html>
<html>
     <head>
          <meta charset='utf-8'>
          <title>Angulari õpetus</title>
     </head>
     <body>
     </body>
</html>
```

Koodinäide 1. HTML faili loomine.

Järgmiseks tuleb lisada *head* tagide sisse Angulari *library* (vt koodinäide 2).

```
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.5/angular.min.js"></script>
```

Koodinäide 2. Angulari ülesseadmine

3.2 Avaldised

```
<p>Pastakate hind kokku: {{ summa * kokku }}€</p>
```

Koodinäide 3. Avaldise abil täisarvude korrutise leidmine

```
<p>Tere {{isik.nimi + ' ' + isik.perekonnanimi}}!</p>
```

Koodinäide 4. Avaldise abil stringide kokku liitmine

```
<p>Vanus: {{isik.vanus}}</p>
```

Koodinäide 5. Avaldise abil objekti kätte saamine

```
<p>Punktid(TI): {{punktid[0]}}</p>
```

Koodinäide 6. Avaldise abil massivist tulemuse kätte saamine

```
<div ng-app='' ng-init='summa=3; kokku=10; isik={nimi: "Juku",
perekonnanimi: "Tamm", vanus: 15}; punktid=[59, 74, 82, 77]'>
     <p>Pastakate hind kokku: {{ summa * kokku }}€</p>
     <p>Tere {{isik.nimi + ' ' + isik.perekonnanimi}}!</p>
     <p>Vanus: {{isik.vanus}}</p>
     <p>Punktid(matemaatika): {{punktid[0]}}</p>
</div>
```

Koodinäide 7. Eelnevalt mainitud avaldised korraga kasutuses

3.3 Andmete sidumine

```
<div ng-app='' ng-init='isik={eesnimi: "Juku", perekonnanimi:
"Tamm"};'>
     <p>Eesnimi: {{isik.eesnimi}}</p>
     <p>Perekonnanimi: <span ng-bind='isik.perekonnanimi'>
</span></p>
</div>
```

Koodinäide 8. Ühte pidi andmete sidumine

```
<div ng-app='' ng-init="kogus=3;hind=6">
     <p>Kogus: <input type='number' ng-model='kogus'></p>
     <p>Hind: <input type='number' ng-model='hind'></p>
     <p>Kokku: {{ (kogus * hind) | currency : '€' : 3}}</p>
</div>
```

Koodinäide 9. Kahte pidi andmete sidumine

Ühte pidi andmete sidumises on andmed ette antud, kuid neid muuta mudel ei saa. Kahte pidi on samuti andmed ette antud seal, aga on võimalus ka andmeid muuta, millejärel mudel ja info uueneb automaatselt.

Andmete sidumist saab spetsifitseerida kahte eri moodi, loogeliste sulgudega `{{isik.eesnimi}}` või *ng-bind* direktiiviga (vt koodinäide 8). Teises näites olen kasutusele võtnud ka filtri *currency* ning selle 2 valikulist lisa, sümbol ja murdosa suurus (vt koodinäide 9).

3.4 Direktiivid

```
<div ng-app='' ng-init='isikud=[{nimi:"Juku", vanus:15},
{nimi:"Juhan", vanus:18}]'>
     <p>Nimi: <input type='text' ng-model='eesnimi'></p>
     <p>Tere, {{ eesnimi }}!</p>
     <p>Isikute list:</p>
     <ul>
          <li ng-repeat='isik in isikud'>
               {{'Nimi: '+ isik.nimi + ', Vanus: ' + isik.vanus}}
          </li>
     </ul>
</div>
```

Koodinäide 10. Direktiivide kasutus rakenduses

- Antud koodis *ng-app* lähtestab vaikimisi Angulari rakenduse. *ng-app* annab Angularile ka teada, et `<div>` element on rakenduse omanik.
- *ng-init* lähtestab rakenduse algandmed, antud näites kasutame JSON süntaksit, et defineerida isikute massiiv.
- *ng-model* direktiiv määrab kasutatavad mudelid või muutujad, antud juhul *input* rakenduse andmega(antud juhul on defineeritud mudel nimega „eesnimi").
- *ng-repeat* loob kordus arv HTML elemente, vastavalt sellele kui palju andmeid massiivis on.

3.5 Kontroller

- Angulari rakendus on defineeritud **`ng-app='uusApp'`**.
- Kontroller on defineeritud *ng-controller='uusCtrl'* direktiiviga. uusCtrl on *JavaScript* funktsioon.
- Angular toestab kontrolleri *$scope* objektiga. Angularis *$scope* on rakenduse objekt, mis omab rakenduse muutujaid ja funktsioone.

```
<div ng-app='uusApp' ng-controller='uusCtrl'>
     <p>Eesnimi: <input type='text' ng-model='isik.eesnimi'></p>
     <p>Perekonnanimi: <input type='text' ng-
model='isik.perenimi'></p>
     <p>Täisnimi: {{ isik.eesnimi + ' ' + isik.perenimi }}</p>
</div>
<script>
var app = angular.module('uusApp', []);
app.controller('uusCtrl', function($scope){
     $scope.isik = {
          eesnimi: 'Juku',
          perenimi: 'Tamm'
     }
});
</script>
```

Koodinäide 11. Kontrolleri loomine ja rakenduses kasutamine

Järgmises näites loome funktsiooni nimega taisnimi objekti *$scope.isik* , mille töö on tagastada täisnimi, seejärel kasutame loodud funktsiooni avaldises (vt koodinäide 12).

```
<div ng-app='uusApp' ng-controller='uusCtrl'>
     <p>Eesnimi: <input type='text' ng-model='isik.eesnimi'></p>
     <p>Perekonnanimi: <input type='text' ng-
model='isik.perenimi'></p>
     <p>Täisnimi: {{ isik.taisnimi() }}</p>
</div>
<script>
var app = angular.module('uusApp', []);
app.controller('uusCtrl', function($scope){
     $scope.isik = {
          eesnimi: 'Juku',
          perenimi: 'Tamm',
          taisnimi: function(){
               var isikObjekt;
               isikObjekt = $scope.isik;
               return isikObjekt.eesnimi + " " +
isikObjekt.perenimi;
          }
     };
});
</script>
```

Koodinäide 12. Kontrolleris funktsiooni loomine ja kasutus rakenduses

Järgmiseks loome uue kontrolleri faili, mida hiljem kasutame ka filtrite peatükis (vt koodinäide 13).

```
angular.module('minuApp', []).controller('nimedCtrl',
function($scope){
     $scope.nimed = [
          {nimi: 'Juku', vanus: 34},
          {nimi: 'Kalle', vanus: 17},
          {nimi: 'Juhan', vanus: 24}
     ];
]);
```

Koodinäide 13. Kontrolleri fail, salvestada nimega nimedKontroller.js

Kasutame tehtud kontrolleri faili rakenduses, mis siis loeb failist andmed ja kätte saadud andmete abil loob listi (vt koodinäide 14).

```
<div ng-app='minuApp' ng-controller='nimedCtrl'>
     <ul>
          <li ng-repeat='i in nimed'>
               {{ 'Nimi: ' + i.nimi + ', Vanus: ' + i.vanus }}
          </li>
     </ul>
</div>
<script src='nimedKontroller.js'></script>
```

Koodinäide 14. Kontrolleri, mis on eraldi failis, kasutus rakenduses

3.6 Filtrid

Kasutame eelmises peatükis loodud kontrollerit uusCtrl ning rakendame sellele filtri *uppercase* (vt koodinäide 15).

```
<div ng-app='uusApp' ng-controller='uusCtrl'>
     <p>Nimi on {{ isik.eesnimi | uppercase}}</p>
</div>
```

Koodinäide 15. Filtri kasutamine avaldises

Antud kood muudab uusCtrlis oleva eesnime suurteks tähtedeks, samamoodi saab ka kasutada *lowercase*. Filtrit *currency* sai kasutatud andmete sidumise peatükis (vt koodinäide 9).

Filtri kasutamine on ka võimalik direktiivides, kui eelmises peatükis sai kasutatud direktiivi *ng-repeat*, siis saab sinna juurde lisada *orderBy* filtri. Järgmise näite juures kasutame kontrollerit nimedCtrl (vt koodinäide 16).

```
<div ng-app='minuApp' ng-controller='nimedCtrl'>
<ul>
     <li ng-repeat='i in nimed | orderBy:"vanus"'>
          {{ 'Nimi: ' + i.nimi +', Vanus: ' + i.vanus }}
     </li>
</ul>
</div>
<script src='nimedKontroller.js'></script>
```

Koodinäide 16. *orderBy* filtri lisamine direktiivile

Filtrit saab ka lisada *input* abil, et saaks kuvada vastavad tulemused välja ainult (vt koodinäide 17).

```
<div ng-app='minuApp' ng-controller='nimedCtrl'>
     <p><input type='text' ng-model='filter'></p>
     <ul>
          <li ng-repeat='i in nimed | filter:filter |
orderBy:"vanus" '>
               {{ 'Nimi: '+(i.nimi | uppercase)+ ', Vanus: '
+i.vanus }}
          </li>
     </ul>
</div>
<script src='nimedKontroller.js'></script>
```

Koodinäide 17. *filter, orderBy* ja *uppercase* filtri kasutamine korraga

3.7 Http

Järgmises näites loeme *Http* abil teisest failist andmeid. Sellejaoks, aga peame uue tekstifaili looma ning andmed sinna JSON formaadis kirja panema. Loome faili nimega angular.txt (vt koodinäide 18).

```
{
"tudengid":[
    {
       "Nimi" : "Juku",
       "Vanus" : 18,
       "Riik" : "Eesti"
    },
      {
       "Nimi" : "Juhan",
       "Vanus" : 14,
       "Riik" : "Eesti"
    },
    {
       "Nimi" : "Tõnu",
       "Vanus" : 12,
       "Riik" : "Eesti"
    },
    {
       "Nimi" : "Kevin",
       "Vanus" : 21,
       "Riik" : "Eesti"
    }
]
}
```

Koodinäide 18. Angular.txt faili sisu

Peale faili loomist kasutame järgmist koodijuppi, et sealt info kätte saada. *$http.get* abil loeme faili ja õnnestumise korral kirjutatakse andmed $scope.nimed objekti. Seejärel saame nimed objektist lihtsalt andmed välja kuvada listi (vt koodinäide 19).

```
<div ng-app='minuApp' ng-controller='tudengidCtrl'>
     <ul>
          <li ng-repeat='i in nimed'>
               {{ i.Nimi + ', ' + i.Riik }}
          </li>
     </ul>
</div>

<script>
     var app = angular.module('minuApp',[]);
```

```
     app.controller('tudengidCtrl', function($scope, $http){
          $http.get('angular.txt')
               .success(function(response){
                    $scope.nimed = response.tudengid;
               }
          );
     });
</script>
```
Koodinäide 19. Http abil teisest failist andmete lugemine

3.8 Tabelid

Tabeli koostamisel kasutame *ng-repeat* direktiivi. Samuti kasutame eelmises peatükis http abil kätte saadud andmeid, mille paneme listi asemel nüüd hoopis tabelisse (vt koodinäide 20).

```
<div ng-app='minuApp' ng-controller='tudengidCtrl'>
     <table border='1'>
          <tr ng-repeat='i in nimed'>
               <td>{{ i.Nimi }}</td>
               <td>{{ i.Vanus }}</td>
               <td>{{ i.Riik }}</td>
          </tr>
     </table>
</div>

<script>
     var app = angular.module('minuApp',[]);
     app.controller('tudengidCtrl', function($scope, $http){
          $http.get('angular.txt').success(function(response){
               $scope.nimed = response.tudengid;
          });
     });
</script>
```
Koodinäide 20. Failist andmete lugemine tabelisse

Tabeleid on ka võimalik sorteerida filtri abil, kui eelmises koodinäites (koodinäide 20) *ng-repeat* ära muuta ja juurde lisada *orderBy* (vt koodinäide 21).

```
<table border='1'>
     <tr ng-repeat='i in nimed | orderBy: "Vanus"'>
          <td>{{ i.Nimi }}</td>
          <td>{{ i.Vanus }}</td>
          <td>{{ i.Riik }}</td>
     </tr>
</table>
```
Koodinäide 21. Vanuse järgi filtreeritud tabel

3.9 Vormid

Vormide koodinäites olen välja toonud mitu erinevat sisestustüüpi ja seejärel need kontrolleri abil objekti salvestanud ning *JSON* formaadis välja kuvanud, kasutades *pre* märgendeid (vt koodinäide 22).

```
<div ng-app='' ng-controller='vormiCtrl'>
     <form>
          <p>Nimi: <input type='text' ng-model='kasutaja.nimi' />
Value: {{ kasutaja.nimi }}</p>
          <p>E-mail: <input type='email' ng-
model='kasutaja.email' /> Value: {{ kasutaja.email }}</p>
          <p>URL: <input type='url' ng-model='kasutaja.url' />
Value: {{ kasutaja.url }}</p>
          <p>Password: <input type='password' ng-
model='kasutaja.password' /> Value: {{ kasutaja.password }}</p>
          <p>Number: <input type='number' ng-
model='kasutaja.number' /> Value: {{ kasutaja.number }}</p>
          <p>Teksti ala: <textarea id='textarea' ng-
model='kasutaja.textarea'></textarea> Value:
{{ kasutaja.textarea }}</p>
          <p><input type='button' ng-click='reset()'
value='Reset'/></p>
          <p><input type='submit' ng-click='uuenda(kasutaja)'
value='Salvesta'/></p>

     </form>
<pre>Kasutaja = {{ kasutaja | json }}</pre>
<pre>Salvestatud = {{ salvestatud | json }}</pre>

</div>
<script>
     var app = angular.module('uusApp', []);
     app.controller('vormiCtrl', function($scope){
          $scope.salvestatud = {};
          $scope.uuenda = function(kasutaja){
               $scope.salvestatud = angular.copy(kasutaja);
          };

          $scope.reset = function(){
               $scope.user = angular.copy($scope.salvestatud);
          };

     $scope.reset();
     });
</script>
```

Koodinäide 22. Vormi loomine ja selle andmete salvestamine objekti

3.10 Animatsioonid

Angulari animatsioonide kasutamiseks tuleb esmalt juurde lisada animatsioonide teek. Selleks tuleb järgnev koodilõik *head* tagidesse lisada (vt koodinäide 23).

```
<script
src='ajax.googleapis.com/ajax/libs/angularjs/1.4.5/angular-
animate.js'></script>
```

Koodinäide 23. Animatsioonide teeki ülesseadmine

Animatsioonide koodinäites olen loonud lihtsa *div* märgendites oleva objekti nähtamatuks muutmise, kui kastist linnuke ära võta (vt koodinäide 24).

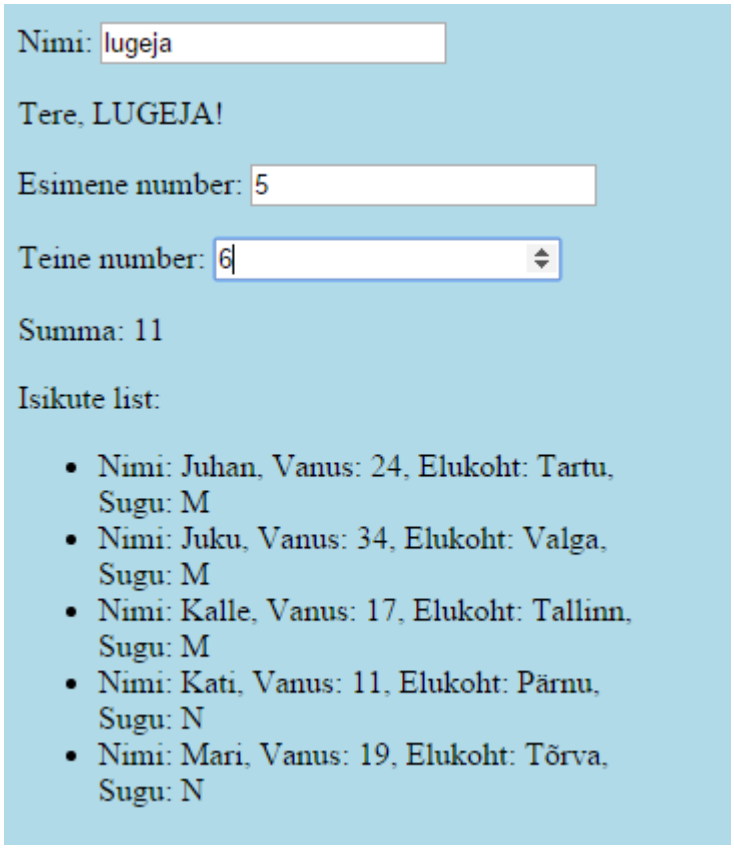
```
<div ng-app="ngAnimate" ng-init="checked=true">
     <label>
          <input type="checkbox" ng-model="checked"
style="float:left; margin-right:10px;"> Is Visible...
     </label>
     <div class="check-element sample-show-hide" ng-
show="checked" style="clear:both;">
          Visible...
     </div>
</div>
<style>
     .sample-show-hide {
     padding:10px;
     border:1px solid black;
     background:white;
     }

     .sample-show-hide {
     -webkit-transition:all linear 0.5s;
     transition:all linear 0.5s;
     }

     .sample-show-hide.ng-hide {
     opacity:0;
     }
</style>
```

Koodinäide 24. *div* elemendi ära peitmine ning nähtavaks tegemine

4 Ülesanded

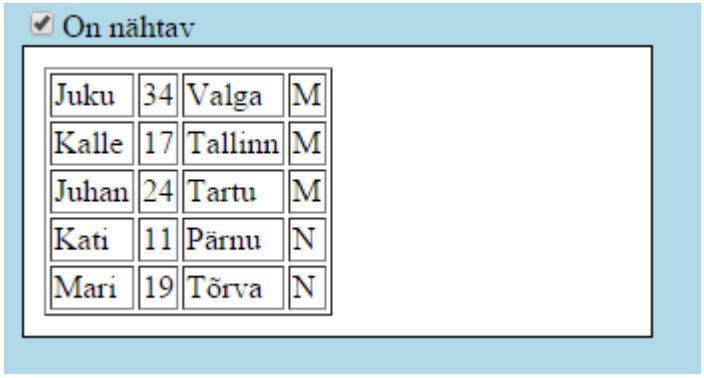
4.1 Näitamine ja arvutamine

Luua veebileht, kus on võimalik direktiivi *ng-model* abil tervitada, mida on läbi *input* välja muuta, algselt on *input* väljas kirjas „*World*”. Samuti teha, et tulem oleks suurte tähtedega. Järgmiseks lisada juurde kaks *input* välja, kuhu saab ainult numbreid kirjutada ning mis liidetakse kokku omavahel. Kolmandaks luua kontroller eraldi failina, kus on kirjas isikud ja nende nimed, vanused, elukoht ja sugu. Faili andmete põhjal luua list, mis on nimede järgi järjestatud.



4.2 Tabeli näitamine ja peitmine

Teise ülesandena kasutada eelmise ülesande kontrollerit või võib luua ka uue, uute andmetega. Selle põhjal luua tabel, mida on animatsioone kasutades võimalik nähtamatuks muuta. Nähtamatuks tegemisel saab kasutada animatsioonide peatükis olevat koodinäidet (vt koodinäide 24).



Kokkuvõte

Käesoleva seminaritöö lähtus probleemist, et puuduvad eesti keelsed Angulari õppematerjalid. Eesmärgiks oli luua õppematerjal, mis koosneb koodinäidetest ja ülesannetest, mida õpilane võiks läbida õppematerjaliga tutvumisel. Kõiki Angulari komponente õppematerjal ei sisalda, kuid autori arvates on tähtsamad komponendid olemas, mida võiks tarvis minna Angulari rakenduse loomise käigus.

Esmalt kirjutab autor vajaminevatest oskustest, mis võiks õpilasel juba omandatud olla. Seejärel on kirjas olemasolevatest õpetustest, mida on töö loomisel kasutatud ning lühidalt Angularist endast ja selle ajaloost.

Igast komponendist, mille kohta töös kirjas on juurde lisatud näidiskood ja näidiskoodi tulemus. Samuti on kõik näited, mida seminaritöös on kasutatud, üleval autori poolt loodud veebilehel http://www.tlu.ee/~sander12/angular.

Seminaritöö lõpus on ka loodud kaks ülesannet, mida on võimalik õpilasel lahendada. Hätta jäämise korral on juurde lisatud ka lõpptulemuse pildid ning koodinäide on üleval ka veebilehel.

Kasutatud kirjandus

w3schools. (kuupäev puudub). *AngularJS Tutorial.* Loetud aadressil http://www.w3schools.com/angular/default.asp

Tutorialspoint. (kuupäev puudub). *AngularJS Tutorial.* Loetud aadressil http://www.tutorialspoint.com/angularjs/index.htm

Google. (kuupäev puudub). *AngularJS*. Loetud aadressil https://angularjs.org/

Bresolin, A. (2015). *AngularJS Hub examples.* Loetud aadressil http://www.angularjshub.com/examples/

Austin, A. (kuupäev puudub). *An Overview of AngularJS for Managers.* Loetud aadressil http://andrewaustin.com/an-overview-of-angularjs-for-managers/