VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Berit Alasmäki

ESTEETTÖMÄN WEB-SIVUSTON TEKEMINEN ANGULAR-TEKNIIKALLA

Liiketalous
2018

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Berit Alasmäki
Opinnäytetyön nimi	Esteettömän web-sivuston tekeminen Angular-tekniikalla
Vuosi	2018
Kieli	suomi
Sivumäärä	67
Ohjaaja	Klaus Salonen

Opinnäytetyöideani sai alkuunsa työharjoittelun aikana, jolloin yhtenä projektina oli tarkoitus luoda esteetön web-sivusto AngularJS-tekniikalla. Esteettömien web-palveluiden kysyntä kasvaa vuosittain, ja nykyisten sivustojen esteettömyyttä pyritään parantamaan jatkuvasti. Tarkoitukseni on tuoda esille opinnäytetyössäni, miten esteettömät sivut voidaan käytännössä toteuttaa ja jota asioita on tärkeää ottaa huomioon esteettömiä sivustoja suunnitellessa.

Käytin sivuston luomisessa nykyistä Angular 5.2.7 ohjelmistokehystä ja sen tukena Node.JS 8.9.4 palvelintyökalua ja npm paketinhallintasovellusta Angular-lisäosien lataamista varten. Tekstieditorina käytin Atomin Windows-versiota 1.24.1.

Lopputuloksena sain suunnitelmieni mukaisesti luotua yksinkertaisen ja esteettömän web-sovelluksen, soveltaen kirjoittamaani teoriaa. Sivusto vastaa nykyistä WCAG-standardia ja toimii sekä web- että mobiilisovelluksena.

Avainsanat esteettömyys, ES6, ES2015, Angular, web-sivusto, WCAG

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Informational Technology

ABSTRACT

Author	Berit Alasmäki
Title	Creating an Accessible Website with Angular Technology
Year	2018
Language	Finnish
Pages	67
Name of Supervisor	Klaus Salonen

I became inspired to write my thesis about Web Accessibility because of my earlier experiences in my practical training period. There I learned for the first time about Angular technology and web accessibility. The demand for accessible web-services are growing rapidly every year while the present websites are updated to more accessible user interfaces. The objective of the thesis was to research the methods of accessible website development and the kind of features important to take a note of when designing an accessible website.

I used the current Angular 5.2.7 framework with the support of Node.JS 8.9.4 server tool to develop the website example for this thesis. As an editor software I used Atom for Windows version 1.24.1.

As a result, I was able to develop a simple and accessible web service that matched my original design and plan. The website matches the current WCAG-standard and works well as a web application or as a mobile application.

Keywords accessibility, ES6, ES2015, Angular, web-site, WCAG

SISÄLLYS

KUVA- JA TAULUKKOLUETTELO

1. JOHDANTO

Vuoden 2016 joulukuussa otin ensimmäistä kertaa yhteyttä tulevaan työharjoittelupaikkaani Leadiniin Helsingissä. Yritys on erikoistunut web-sivustojen suunnitteluun ja toteuttamiseen ja sen käyttämät tekniikat ovat nykyajan sivusto suunnittelussa ja toteutuksessa tärkeällä sijalla. Kuultuani pääseväni yritykseen työharjoittelemaan, olin varma, että tulisin oppimaan paljon uusimmista nettisivuston tekniikoista. Ensimmäisen projektin aloitettuani, pääsin tutustumaan AngularJS ohjelmistokehykseen, johon kuului myös Git-versiohallintajärjestelmän, että Bitbucket-palvelun käyttöä. Selitän näiden tehtävän ja tarkoituksen paremmin seuraavassa luvussa.

Esteettömyys tai toiselta nimeltään helppokäyttöisyys tietokoneissa ja nettisivustoissa, tuli minulle ensimmäistä kertaa ajankohtaiseksi ensimmäisessä työharjoitteluprojektissani. Projekti kesti neljä kuukautta ja tehtäväni oli suunnitella ja toteuttaa esteetön online-raportointityökalu jota myös yrityksen sokea työntekijä pystyisi käyttämään sujuvasti. Tämän lisäksi yritys itsessään oli erikoistunut web-sivustojen esteettömyyden tarkistamiseen. He siis tarkistivat asiakkaidensa sivustoja esteettömyyden suhteen ja kokosivat näistä havainnoista raportin. Raportti tehtiin aikaisemmin Word-tiedostoon, mutta he halusivat tehdä raportoinnin sujuvammaksi ja nopeammaksi online-palvelun avulla. Projekti onnistui hyvin, mikä antoikin minulle motivaation kirjoittaa opinnäytetyöni perustuen tähän aihealueeseen.

Opinnäytetyöni tarkoitus onkin esitellä lukijalle nykyisen Angular ohjelmistokehyksen (aikaisemmin käyttämäni oli AngularJS) ja siihen pohjautuvan Angular Materialin sekä opettaa lukijalle esteettömän nettisivuston suunnittelun ja toteuttamisen eri vaiheet. Aihe on ajankohtainen, sillä Euroopan parlamentti hyväksyi 26.10.2016 direktiivin julkisen sektorin verkkopalveluiden saavutettavuudesta, mikä tarkoittaa sitä, että kaikkien julkisen sektorin verkkopalve-

luiden pitää olla saavutettavat uusissa sivustoissa syyskuussa 2019 ja vanhoissa sivustoissa syyskuussa 2020 (European Commission 2016 & Celia 2017)

Odotan lukijan olevan aloitteleva ohjelmoija (esimerkiksi toisen vuoden opiskelija), jolla on jo jonkinlaista ymmärrystä web-sivustojen rakenteesta ja toteutuksesta (HTML, CSS ja JavaScript). Haluan opinnäytetyöni olevan selkeä opas uuden tekniikan oppimiselle.

2. TERMIEN JA KÄSITTEIDEN SELITYS

Useimmiten suomenkielistä materiaalia löytyy suppeammin kuin englanninkielistä, joten päätin suomenkielisten termien lisäksi lisätä useimpiin myös englanninkieliset käännökset. Tällä tavoin lukija voi halutessaan hankkia monipuolisemmin lisätietoa kiinnostavasta termistä tai aihe-alueesta.

2.2 Termejä lyhyesti

Asynkronisuus tarkoittaa ei-reaaliaikaista kommunikointia eli kommunikointitapaa, jossa kommunikoinnin osapuolet eivät ole ajallisesti toisistaan riippuvaisia eli ovat ajasta (ja paikasta) riippumattomia. Tietokone maailmassa hyvinä esimerkkeinä voitaisiin pitää esimerkiksi sähköpostia, blogia tai uutisryhmiä. Tässä tapauksessa asynkronisuus tarkoittaa web-teknologiaa, missä käyttäjän näkemä sivusto muuttuu dynaamisen järjestelmän tilan muuttuessa, ilman että käyttäjän tarvitsee olla vuorovaikutuksessa käyttöliittymän kanssa (esimerkiksi lataamalla sivuston uudestaan). Hyviä esimerkkejä tällaisesta toiminnasta on Twitterin uutissyöte, joka päivittyy automaattisesti ilman, että käyttäjän pitää päivittää sivustoa. (Maryka 2009)

BitBucket-online työkalu on hajautettu versionhallintajärjestelmä, jonka avulla yhteistyö tiimin kanssa onnistuu helposti ja vaivattomasti. BitBucketia käytetään gitin avulla. (Atlassian 2018)

Dynaaminen sivusto on kokonainen websivusto tai yksittäin web-sivu joka sisältää muuttuvaa dataa, riippuen esimerkiksi käyttäjän sijainnista tai aikavyöhykkeestä. Dynaamisen sivuston vastakohtana on staattinen sivusto, jonka data ei muutu vaan pysyy samana jokaisella käyttäjällä riippumatta heidän sijainnistaan tai aikavyöhykkeestään. (Computer Hope 2017)

Git versionhallintaohjelmisto, joka on suunniteltu toimimaan hajautetusti ja mahdollisimman tehokkaasti. Git toimii parhaiten UNIX ja iOS käyttöjärjestelmillä, mutta sitä on mahdollista käyttää myös Windowsin koneella. Gitiä käytetään mm. BitBucketin ja GitHubin tyylisissä webpalveluissa. (Atlassian)

Komponentti on Angular sovelluksen tavallisempia käyttöliittymän rakennuspalikoita. Komponentteihin on myös asetettu oletusulkoasut. (Angular)

Skriptaus (*scripting*) tarkoittaa menetelmää, jonka avulla web-sivustoon voidaan rakentaa toiminnallisia rakenteita ja näin luoda sivustosta dynaamisemman. (Lappalainen 2010)

UI-suunnittelu (User Interface) tarkoittaa käyttöliittymän visuaalista suunnittelua.

UX-suunnittelu (User Experience) tarkoittaa käyttöliittymän toiminnallista suunnittelua eli käyttäjäkokemuksen suunnittelua.

Esteettömyys nettisivustossa

”Esteettömyydessä on kyse ihmisten moninaisuuden huomioonottamisesta rakennetun ympäristön suunnittelussa ja toteuttamisessa. Se merkitsee palvelujen saavutettavuutta, välineiden käytettävyyttä, tiedon ymmärrettävyyttä ja mahdollisuutta osallistua itseä koskevaan päätöksentekoon” (Invaliidiliitto)

Puhuessamme esteettömyydestä, emme puhu ainoastaan liikkumisen esteettömyydestä fyysisessä ympäristössä vaan myös esimerkiksi näkemisen, kuulemisen, kommunikaation ja sähköisen viestinnän esteettömyydestä. Esteettömyys merkitsee palvelujen saatavuutta, välineiden käytettävyyttä, tiedon ymmärrettävyyttä ja mahdollisuutta osallistua itseä koskevaan päätöksentekoon. Käytännössä esteet voidaan jakaa fyysisiin, psyykkisiin, sosiaalisiin ja taloudellisiin. (Söderholm 2003)

Esteettömät nettisivustot tarkoittavat sitä, että kaikenlaisilla käyttäjillä on mahdollisuus käyttää web-palvelua sujuvasti ja itsenäisesti. Tämä ei helpota ainoastaan käyttäjän vaan myös palveluntarjoajien toimintaa huomattavasti. Aikaa kuluu vähemmän tiedon etsimiseen ja hakemiseen sekä virheellisten toimintojen korjaamiseen.

Ihmisiä on erilaisia mikä tarkoittaa myös tarpeiden vaihtelevan suunnasta toiseen. Joku tarvitsee tukea fyysisessä toiminnassa esimerkiksi käden tai pään liikkeissä,

kun taas jollekin toiselle näkeminen tai kuuleminen on hankalaa tai lähes mahdotonta. Väestön ikääntyessä ihmisten eritystarpeet kasvavat entisestään, jolloin myös palveluntarjoajien pitää olla valmis muuttumaan. Suomessa on 5.5 miljoonaa ihmistä, joista esimerkiksi näkövammaisia on noin 60 000. Monet heistä joutuvat tukeutumaan toisen ihmisen apuun jolloin myös itsenäinen päätöksenteko voi joutua vaaraan. Tämän vuoksi onkin tärkeää, että web-palvelut luodaan tavalla, mikä edistää itsenäisyyttä ja yksilön valinnan vapautta. (Näkövammaisten liitto ry 2016)

Esteetön suunnittelu tarkoittaa

- verkkopalveluita ja -ympäristöjä, joiden toteuttamisessa on otettu huomioon kaikki käyttäjäryhmät riippumatta käyttäjän iästä, toimintakyvystä, vammoista tai muista yksilöllisistä ominaisuuksista (Design for all).
- verkkoympäristöjen toteutuksessa käytettyjä tekniikoita, jotka ottavat huomioon apuvälineiden ominaisuudet.

Esteettömän verkkosivuston pitää olla

- helposti mukautuva eri aisteille ja apuvälineille sopivaksi.
- rakenteeltaan helposti navigoitava ja selkeä.
- kieliasultaan yksiselitteistä ja ytimekästä.

Hyvä verkkosuunnittelu lähtee siitä, että sivu voidaan kerralla suunnitella kaikille sopivaksi. Kun otetaan huomioon erilaisia käyttäjäryhmiä erilaisine tarpeineen, estämättä minkään ryhmän pääsyä verkkoon, edistää se verkon käytettävyyttä kaikkien kannalta. (Söderholm 2003)

2.3 Käyttäjät ja tarpeet

Tietokoneen käyttäjät ovat kuin ruokakaupan asiakkaita. Tarpeita on erilaisia: liikkuminen kaupan käytävillä ja tavaroiden nostaminen kärrylle, tuoteselosteiden ja

tuotemerkin näkeminen, kuulutusten kuuleminen sekä kassalla maksaminen. Asiakkailla on erilaisia esteitä ruokavalion, liikkumisen, kuulemisen ja näkemisen suhteen. Sama pätee myös tietokoneen käyttäjiin. Jota paremmin käyttäjien erityistarpeet ja esteet otetaan huomioon, sitä arvostetumpi web-palvelu on. Web-palvelun tarjoavat ovat kuin kauppiaita. On heidän vastuullaan, että asiakkaat saavat sekä kohteliasta ja hyvää asiakaspalvelua että ympäristö on sopiva liikkumista, näkemistä ja kuulemista varten. Tämä ei hyödytä pelkästään eritystarpeellisia vaan myös ns. tavallisten asiakkaiden kauppakokemusta. Se viestittää asiakkaille, että kauppa on vastuullinen ja oikeasti kiinnostunut asiakkaidensa tarpeista. Esteetön ympäristö toimii yleensä myös fiksummin ja selkeämmin. Kun asiat pidetään yksinkertaisena, toimii myös ihmisten välinen toiminta ja kommunikointi sujuvammin.

Tietämys eri ryhmien kannalta esteettömistä verkkoympäristöistä sekä heidän käyttämistään apuvälineistä auttaa ottamaan huomioon erilaisten käyttäjäryhmien tarpeet. Lisäksi tietämys auttaa kiinnittämään huomion sellaisiin esteisiin, jotka ovat ratkaistavissa nimenomaan verkkoympäristöjen suunnittelulla.

Kun puhutaan erityistarpeista, on yleensä kyse ihmisen aisteihin liittyvästä tarpeesta. Ihmisen viisi perusaistia – näkö-, haju-, maku-, kuulo- ja tuntoaisti – auttavat havainnoimaan ympäristöä. Tämän lisäksi ihmisellä on myös tasapaino- ja asentoaisti, jotka kertovat kehon sijainnista ympäristössään. (Tieteen kuvalehti 2010)

2.4 Miten aistihäiriöt vaikuttavat eri tietokoneenkäyttäjiin

2.4.1 Visuaalisuus

"Näkövammaiseksi määritellään henkilö, jonka paremman silmän laseilla korjattu nääntarkkuus on heikompi kuin 0,3, ja sokeaksi jos paremman silmän laseilla korjattu nääntarkkuus on alle 0.05 tai näkökenttä supistunut halkaisijaltaan alle 20 asteeseen, tai jos toiminnallinen näkö on jostain muusta syystä vastaavalla tavalla heikentynyt." (Näkövammaisten liitto ry.)

Tavallisesti näkeville ihmisille tietokone on visuaalinen laite, koska sen tuoma data on asetettu näyttönäkymään. Kaikki perustuu näkemiseen. Ilman näkemistä, tietokonetta olisi mahdotonta käyttää. Vai olisiko? Tietokone on nopeuttanut ja helpottanut monen ihmisen arkea. Useimmat terveydenhuollon, pankin, vakuutusyhtiön ja kunnanpalvelut ovat siirtyneet perinteisestä toimistosta web-palveluksi. Asiakkaiden ei tarvitse enää käydä paikan päällä, kun monet yksinkertaiset asiat pystytään hoitamaan netin välityksellä. Tämä helpottaa myös heikosti näkevien ja sokeiden arkea, koska ulos ei tarvitse lähteä vaan kaiken voi hoitaa kotoa käsin. Miten tietokonetta voidaan siis käyttää, kun ihmisen näköaisti on heikko tai puuttuu jopa kokonaan?

Heikosti näkevät

Heikkonäköinen ihminen ei näe joko kauas tai lähelle. Hän saattaa siis pärjätä hyvin liikkuessaan ulkona, kun taas lukeminen saattaa olla vaikeaa. Hän voi nähdä tarkasti muutaman senttimetrin kokoisen alueen ja muun taas pelkkänä sumuna. Useasti henkilö voi kärsiä myös heijastuksista ja nähdä ympäristönsä ikään kuin keltaisten linssien lävitse. Tämän lisäksi heikosti näkevillä ihmisillä voi olla hankaluuksia nähdä värejä. (Söderholm 2006)

Heikosti näkeville on siis ensisijaista, että tekstiä, kuvioita ja kuvia on mahdollista suurentaa tai pienentää. Visuaalinen ilme ja värimaailma pitää ottaa tarkasti huomioon. Varsinkin värisokeille on tärkeää, että saman tummuus- tai vaaleusasteen värejä ei sekoitettaisi keskenään. Kontrasti pitää olla tarpeeksi selkeä, jotta pää- ja taustavärin voidaan erottaa toisistaan. Jos esimerkiksi tekstin taustaväri lähentelee samaa sävy ja vaaleusastetta kuin itse tekstin väri, voi varsinkin värisokeille olla hankalaa erottaa tekstiä taustasta. Jos varmuutta värien esteettömyydestä nettisivustolla ei ole, voi netissä olevia värikontrasti suhteen analysaattori työkaluja käyttää apuna. WebAim on yksi käytetyimmistä ja luotettavimmista värikontrastin tarkistamistyökaluista.

Sokeat

Sokeakaan henkilö ei aina ole täysin sokea. Hän saattaa nähdä esimerkiksi valoja ja varjoja. Kun sokeille henkilöille suunnitellaan verkkosivuja, hyvä suunnittelu lähtee siitä, että näköä ei edellytetä mutta jäljellä olevan näkökyvyn käyttöä tuetaan sivun suunnittelun keinoin. Samat periaatteet jotka pätevät heikkonäköisille, pätevät myös sokeille käyttäjille: värejä voidaan muuttaa, kirjaimien kokoa voidaan vaihtaa tai ellei näiden muuttaminen ole mahdollista, ainakin värit muodostavat keskenään selkeän kontrastin. (Söderholm 2006)

Sokeat käyttävät samoja graafisia selaimia kuin näkevätkin henkilöt. Osa heistä käyttää edelleen myös tekstipohjaista Lynx-selainta. Jotta sokea käyttäjä voisi käyttää verkkoa sujuvasti, pitää hänen käyttää helppokäyttöisen selaimen lisäksi myös muita apuvälineitä. Sokeat käyttävät puheselaimia, ruudunlukuohjelmia ja ”pistenäyttöä”. Monesti myös erittäin heikkonäköiset henkilöt käyttävät ruudunlukuohjelmia.

Sokeiden henkilöiden käyttämät apuvälineet ymmärtävät verkkosivulta vain tekstimuodossa olevan informaation. Apuvälineohjelmat muuntavat verkkosivulla olevan tekstin puheeksi tai pistekirjoitukseksi. Tärkein ohje jota sivujen tekijöiden olisi hyvä muistaa on, että kaikille kuville, kuten valokuville, animaatioille ja kuvamuotoisille painikkeille, tulisi kirjoittaa alt-teksti. Hyvä alt-teksti kertoo esimerkiksi valokuvan merkityksen sivulla tai kuvana olevan toimintopainikkeen toiminnon ytimekkäästi. Vain aivan poikkeustapauksissa tulisi luoda erillinen tekstiversio sivustosta.

Verkkosivujen visuaalisuus on seikka, joka asettaa eniten haasteita sokeille ja vaikeasti heikkonäköisille henkilöille ja siten myös sivujen suunnittelijoille. Nykyisin meidän onneksemme on kuitenkin kehitetty web-tekniikoita, kuten Angular, joiden avulla visuaalisesti rajoittuneiden käyttäjien tarpeet voidaan ottaa huomioon sivustojen toteuttamisessa sujuvammin ja tehokkaammin. (Söderholm 2006)

2.4.2 Toiminnallisuus

Ihmisen toiminta perustuu fyysisiin, kognitiivisiin ja neurologisiin toimintoihin.

Fyysisesti rajoittuneet

Fyysisesti rajoittuneet henkilöt ovat voineet menettää jonkin kehonsa osan toiminnan esimerkiksi vanhenemisen (sairaudet, lihasten ja luuston heikentyminen) tai onnettomuuden johdosta. Motorisia taitoja heikentäviä sairauksia ovat esimerkiksi niveltulehdukset (nivelreuma) joiden aiheuttajina usein toimivat bakteerit ja virukset. Nivelsairaudet ovat tietokoneen käyttäjän kannalta erittäin esteellisiä, koska ne voivat heikentää tai poistaa jopa kokonaan käyttäjän hiiren tai näppäimistön käytön mahdollisuuden. Myös onnettomuuden aiheuttamat fyysiset rajoitteet esim. halvaantuminen voi aiheuttaa tietokoneen käyttäjälle monenlaisia haasteita. Kaikista radikaalimmat rajoitteet ovat, kun käyttäjä kykenee käyttämään vain päätänsä tai pelkästään silmiänsä saadakseen luotua tietokoneen näytönohjaimelle liikeradan. Uutta teknologiaa kehitetään tämän saralla jatkuvasti esimerkiksi silmänliikkeen tunnistavia kameroita, jolloin käyttäjä voi liikuttaa hiirtä pelkästään katseensa avulla. (WebAim 2012; Lumio 2016)

Yleisesti ottaen, kun fyysisesti rajoittuneiden käyttäjien tarpeita otetaan huomioon, olisi erittäin oleellista kiinnittää huomiota hiiren käyttöön. Useimmiten samaa näppäimistöteknologiaa (aria-label) jota sokeille käyttäjille käytetään, voidaan hyödyntää myös fyysisesti rajoittuneiden käyttäjien tukemisessa.

On olemassa myös käyttäjiä, joilla on amputoitu toinen tai jopa molemmat kädet. Tällöin sekä hiiren että näppäimistön käyttö vaikeutuu. Yksikätisille henkilöille on olemassa yhden käden näppäimistöjä, kun taas kädettömien henkilöiden on mahdollista käyttää näppäimistöä esimerkiksi pitämällä suussa tai varpaissa apuvälinettä (nk. suutikkua) joka helpottaa näppäimistön käyttöä. Tällainen käytäntö on kuitenkin usein hidasta ja aikaa vievää. Nopeampia apuvälineitä ovat esimerkiksi pääpannat jotka reagoivat pään liikkeisiin tai äänentunnistuslaitteet, jolloin käyttäjä voi käyttää tietokonetta äänen avulla. Useimmiten suuriosa motoristisia rajoitteita varten tehdyt apulaitteet toimivat samalla tavoin kuin näppäimistö.

Parkinsonin kaltaiset sairaudet ovat haasteellisia, koska käyttäjän käden tärinä voi vaikeuttaa tämän hiiren ja näppäimistön käyttöä. Parkinsonin tauti voi olla siinä

mielessä hankala, jos henkilö ei pysty käden tärinän vuoksi käyttämään kumpaakaan hiirtä tai näppäimistöä eikä äänentunnistusohjelmat kykene tunnistamaan käyttäjän ääntä, jos äänikin värisee liiakseen. Tällaisissa tapauksissa ei ole paljonkaan vaihtoehtoja tarjolla tällä hetkellä. On kuitenkin mielenkiintoista seurata, miten tautia saadaan hillittyä teknologian avulla esimerkiksi ns. aivoihin kohdistuvan sähköstimulaation avulla (The Michael J.Fox Foundation).

Kognitiivisesti rajoittuneet

Kognitiivisista vammoista ja oppimishäiriöistä kärsivät ovat yhteiskunnan suurin esteellinen käyttäjäryhmä. Silti tämä käyttäjäryhmä usein unohdetaan web-sivuston esteettömyyttä suunniteltaessa.

Kognitiivinen vammaisuus on itsessään jo niin laaja käsite, että usein sitä on vaikea määritellä. Yksinkertaisesti sanottuna kognitiivisesti vammautuneella henkilöllä on keskimääräistä vaikeampaa suorittaa yhden tai useamman mentaalisen tehtävän. On kuitenkin hyvä muistaa, että kognitiivisia vammoja on lukuisia, eikä niitä voi määritellä ainoastaan yhdellä lauseella. Suuriosa kognitiivisista vammoista perustuu joko biologisiin tai psykologisiin seikkoihin. Henkilön biologian ja henkisen prosessin keskinäinen yhteys tulee usein parhaiten esillä traumaattisissa aivovammoissa ja geneettisissä häiriöissä.

Henkilö jolla on vaikea kognitiivinen vamma saattaa tarvita apua melkein kaikessa päivittäisessä toiminnassa, kun taas henkilö jolla on lieviä oppimisvaikeuksia, kykenee usein toimimaan sujuvasti vammastaan huolimatta, jopa niin hyvin, että kyseistä vammaa ei tulla koskaan löytämään tai diagnosoimaan.

Kognitiivisista vammoista puhutaan usein kliinisinä vammoina. Lievänä kognitiivisena vammana pidetään esim. oppimisvaikeudet tai tarkkaavaisuus- ja ylivilkkaushäiriöt (ADD/ADHD). Haasteellisemmat kognitiiviset vammat ovat esimerkiksi autismi, Down Syndrooma tai dementia. Web-esteettömyyden näkökul-

masta kliininen näkökulma ei ole tarpeeksi selkeä ja siitä on vaikea kehittää mitään selkeää kokonaisuutta. Senpä vuoksi WebAim sivusto ehdottaakin kognitiivisten vammojen jakamisen toiminnallisiin piirteisiin eikä kliinisiin.

Yleisimmät toiminnalliset kognitiiviset piirteet voisivat olla WebAimin mukaan seuraavanlaiset:

- muisti
- ongelmaratkaisu
- huomiointikyky
- lukeminen
- kirjallinen ja suullinen ilmaisutaito
- matemaattinen ilmaisutaito
- visuaalinen ilmaistutaito.

Kognitiiviset vammat ovat sen verran haasteellisia huomioida esteettömyyden suunnitellussa sen laajuuden vuoksi, että usein se jätetään vähemmälle huomiolle. Web-sivustojen suunnittelijoiden ja sisällöntuottajien on kuitenkin mahdollista kehittää sivustoa paremmaksi tekemällä yhteistyötä alan ammattilaisten, esimerkiksi erikoistuneiden psykologien kanssa. Sivustosta voidaan tehdä visuaalisesti kiinnostavampi (aisteja herättävän) ja sisällöstä yksinkertaisen (esimerkiksi tekstit voidaan luoda yksinkertaisemmaksi). Tämä vaatii kuitenkin perusteellisen tutkimuksen, jotta sivusto toimisi toivotulla tavalla.

Sisällöntuottajien ja suunnittelijoiden tulisi pyrkiä yksinkertaisuudessaan helposti ymmärrettävään ja käytettävään ulkoasuun ja sisältöön. Tämä sääntö ei päde pelkästään huomioidessa kognitiivisesti rajoittuneet käyttäjät vaan myös muutkin käyttäjäryhmät. (WebAim 2013)

Neurologisesti rajoittuneet

Neurologiset vammat voivat aiheuttaa ongelmia aisteissa (aistiharhat), mentaalisessa prosessissa (esim. kun päätöksiä tehdään tai tunteita käsitellään) ja motorisia toimintoja. Tyypillisimmät oireet voivat usein olla esimerkiksi halvaantuminen, vapiseminen, muistin menetys ja kognitiiviset toimintahäiriöt. Neurologiset ongelmat voivat johtua geneettisistä häiriöistä jotka vaikuttavat aivojen ja hermoston toimintaan kuten esimerkiksi lihasdystrofia joka rajoittaa ihmisen liikkumista merkittävästi, rappeutumista aiheuttavat taudit, kuten Alzheimer, ja kouristuskohtaukset, kuten epilepsia. (Indiana University & Rintahaka 2016)

2.4.3 Ääni

Kuurot

Kuulovammaisuus kuvaa noin joka kahdeksannen suomalaisen tilannetta. Kuulovammaisia on monenlaisia ja vamman vaikutukset vaihtelevat yksilöstä toiseen. Esimerkiksi lievästi huonokuuloinen ei tarvitse välttämättä kuulolaitetta, kun taas kuurot käyttävät täysin omaa kieltä viittoen. Yksilöllisyys asettaa hyvin erilaisia vaatimuksia kuulovammaisten tarpeiden täyttämiselle. Tiedonsaannin mahdollisuus ja vuorovaikutus ovat kuitenkin tasavertaisen osallistumisen ehdoton edellytys. Tarjolla ei ole pelkästään yhtä patenttiratkaisua, vaan useampia vaihtoehtoja, joista käyttäjä voi valita itselleen sopivimman. Suuri osa sisältöjen saavutettavuutta parantavista ratkaisuista hyödyttää kuitenkin kaikkia käyttäjiä.

Kuulovammaisuus ei tarkoita ainoastaan, että ihminen kuulee heikommin. On myös tärkeää ymmärtää se, että vaikka kuulovammainen käyttäisikin kuulolaitetta, se ei palauta kuuloa normaaliksi. Kuulovammaisuus vaikeuttaa esimerkiksi ryhmäkeskusteluihin osallistumiseen, koska puheesta ja muista tarpeellisista äänistä sekä niiden suunnasta on vaikea saada selvää erityisesti meluisassa tai kaikuisessa ympäristössä. Jos taas valaistus on riittämätön tai häikäisevä saattaa huulilta lukeminen ja viittominen koitua hankalaksi. Nämä esimerkit tuovatkin hyvin esille sen, kuinka haasteellista olisi, jos informaation saaminen perustuisi vain kuulemiseen. Tekstimuotoisen ja visuaalisen informaation tarjoamisen lisäksi on tärkeää kiinnittää huomiota myös kokoontumis-, opetus-, ja palvelutilojen kuunteluolosuhteisiin.

Verkkosivuja suunnitellessa on tärkeää varmistaa, että minkäänlainen informaatio ei ole ainoastaan riippuvainen kuulosta. Tämä ominaisuus ei hyödytä ainoastaan kuulovammaisia vaan myös normaalikuuloisia, jotka käyttävät päätelaitetta ilman kaiuttimia. (Salomaa 2005)

2.4.4 Mykät

Mykkyys voidaan yhdistää myös aikaisemmin esille tuotuun kognitiiviseen toiminnallisuuteen, koska usein mykkyys voi johtua aivosairaudesta, aivotoiminnan synnynnäisistä puutteellisuuksista tai myöhemmällä iällä aivotautien vaikutuksesta syntyneestä afasiasta. Mykkyys on puhevamma ja tarkoittaa kyvyttömyyttä puhua. (Aggarwal, Sharma, Kumar & Sharma 2010)

Mykkyys ei sinänsä haittaa tietokoneen käyttäjän toimintaa, ellei omaa ääntä tarvitse jollakin tapaa käyttää. Mykät ihmiset kuulevat kyllä, mutta eivät pysty itse vastaamaan äänellä. Internet on itseasiassa helpottanut mykkien ihmisten päivittäistä toimintaa, koska tietokoneen avulla he voivat kommunikoida paljon tehokkaammin, kuin esimerkiksi käymällä paikan päällä. Monet asiat voidaan hoitaa kirjoittamalla. Mykille käyttäjille soveltuukin hyvin muiden, aikaisemmin mainittujen käyttäjäryhmien esteettömien web-sivustojen piirteet. (Bigby 2018)

2.5 Tekniikat esteettömän nettisivuston toteutuksessa

RIA ja ARIA (WAI-ARIA)

Rich Internet Application **eli RIA** tarkoittaa internetissä olevaa sovellusta, mikä muistuttaa hyvin paljon tietokoneen työpöytäsovelluksia laajuutensa vuoksi. Laajuudella tarkoitetaan sovelluksen käyttöominaisuuksia ja käyttötapoja eli mitä sovelluksella pystytään tekemään. Hyviä esimerkkejä nykyisistä RIA-teknologioista on esimerkiksi AJAX, Adobe Flex ja Microsoft Silverlight teknologiat. Tunnettu videopalvelu Youtube on esimerkiksi toteutettu Adobe Flex-teknologialla.
Vuonna 2007 Jyväskylän yliopiston opiskelija Mikko Matias Poikonen teki kandidaattitutkielmansa tästä aiheesta ja kuvasi RIA-sovelluksia seuraavalla tavalla:

”--- rikkaat Internet-sovellukset ovat Web-sovelluksia, joissa sovelluksen esityslogiikan lisäksi myös ainakin osa liiketoimintalogiikasta on toteutettu myös asiakkaan puolella. Rikkaissa Internet-sovelluksissa sovelluksen latauksen jälkeen asiakas ja palvelin kommunikoivat pääasiallisesti asynkronisesti. Käyttöliittymän toteutuksessa on käytetty korkeaa interaktiotasoa tarjoavia komponentteja. Asiakkaan puoleinen osa sovelluksesta voidaan toteuttaa joko Web-selaimen päällä toimivilla tekniikoilla tai Web- toimitusta hyödyntävillä tekniikoilla.” (Poikonen 2007)

ARIA-teknologian avulla taas erityskäyttäjät, varsinkin sokeat käyttäjät, pystyvät lukemaan RIA-pohjaisia websivustoja. Esimerkiksi ARIAn avulla näytönlukijaohjelmat pystyvät lukemaan websivustoa loogisesti, kunhan web-sivusto on tehty semanttiseksi. ARIA on lyhenne sanasta **A**ccessibility of **R**ich **I**nternet **A**pplications mikä tarkoittaa karkeasti suomennettuna “Laajojen/Rikkaiden Internet sovellusten helppokäyttöisyys/esteettömyys”.

ARIA (WAI-ARIA) on periaatteessa HTML-koodin asetettuja ”selitekoodeja”, joiden tarkoitus on parantaa web-sivuston komponenttien toimivuutta ja esteetöntä ”lukuelämystä”. Esimerkiksi kun sokean käyttäjän pitää täyttää lomakkeen netissä pitää lomake-mallissa käyttää tekstilaatikoita, joihin käyttäjä voi syöttää dataa mm. oman nimen, osoitteen ja puhelinnumeron. Oletusarvona näytönlukija lukee nämä tekstilaatikot vain tekstilaatikoina *(input)*, ilman sen tarkempaa kuvausta. Kun ARIA selite asetetaan ja sen arvoksi annetaan esimerkiksi tekstilaatikkoa kuvaavan nimen ”Nimi” tai ”Puhelinnumero”, lukee näytönlukija ”input” tekstin lisäksi myös ARIAN avulla asetetut selitteet. Tällä tavoin käyttäjä pystyy täyttämään lomakkeen sujuvasti.

Hyvin monet web-sivustot hyödyntävät yhä enemmän RIA-teknologiaa, jonka vuoksi ARIAn merkitys esteettömien web-sivustojen suunnittelussa ja toteutuksessa paranee vuosi vuodelta.

Jota ongelmia, RIA-teknologia voi oikeastaan aiheuttaa esteettömyyden silmissä? Tiivistettynä RIAssa on seuraavat haasteet:

- Kyvyttömyys tarjota semanttisuutta web-sivuissa ja sen toiminnoissa (esimerkiksi navigointi, pääsisältö, haku jne.)
- Esteellisen teknologian käyttö, missä dynaaminen sisältö vaihtuu sivustossa jatkuvasti (esimerkiksi AJAXin tekemät sisällönpäivitykset)
- Kyvyttömyys vaihtaa näppäimistön fokusta (*focus*) sivun elementeissä (esimerkiksi vaihtamalla fokus sivuston virheilmoitukseen, kun sellainen tulee esille)
- Monimutkaisten aputyövälineiden (*widgets*) ja navigointi elementtien vaikea käyttö näppäimistöllä ja näytönlukija sovelluksella (esimerkiksi liukukuvagalleriat (*sliders*), valikkonäkymä (*menu tree*) jne.)

ARIAn avulla monia tällaisia ongelmia pystytään ratkomaan.

ARIAssa on kolme pääkomponenttia roolit (*roles*), ominaisuudet (*properties*) ja tilat (*states*).

Roolit (*Roles*)

ARIA roolit määrittelevät mikä elementti on ja jota se tekee. Useimmilla HTML-elementtien rooleilla on oletusarvo joka tulee esille avustavassa teknologiassa. Esimerkiksi, painikkeen roolilla (*button*) on oletusarvona ”button” ja lomakkeen roolilla on oletusarvona ”form”. ARIAn avulla voit itse määritellä rooleja jotka eivät ole saatavilla HTML:ssä. HTML-koodin oletusarvot voidaan myös korvata omilla arvoilla. Tämän avulla voidaan määritellä tarkemmat tiedot näytönlukijalle, mikä takaa sujuvamman käyttökokemuksen. Hyvänä esimerkkinä ARIA roolista on *<form role=”search”>* mikä kuvaa kyseisen lomakkeen olevan hakemista (*search*) varten. HTML-kielessä kaikissa lomakkeissa on samanlainen semanttisuus. Mutta ARIAn avulla voit lisätä tietyn lomakkeen oletusrakenteeseen (semanttiin) määritelläksesi sen hauksi.

Ominaisuudet (*Properties*)

ARIA ominaisuudet kuvaavat elementtien tarkoitusta. HTML-kielen alkuperäistä semanttisuutta voidaan laajentaa määrittelemällä elementeille ominaisuuksia jotka

eivät ole sallittuja standardissa HTML-kielessä. Esimerkiksi <input aria-required=”true”>. Tämä ominaisuus varmistaa sen, että käyttäjän on pakko lukea tämä kohta ruudunlukijan kanssa, päästäkseen seuraavaan kohtaan.

Tilat (*States*)

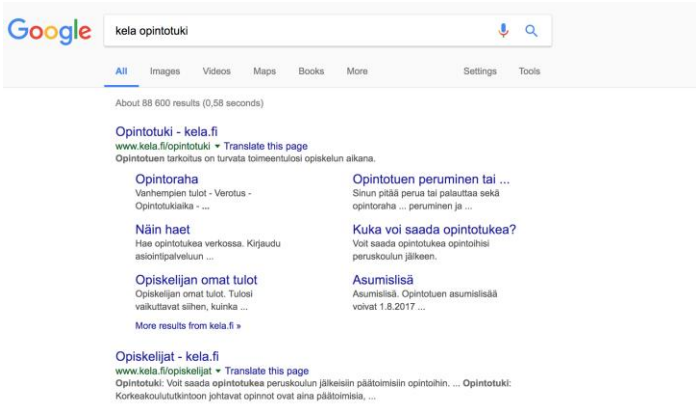
ARIA tilat ovat ominaisuuksia jotka määrittelevät elementin nykyisen tilan. Nämä yleensä vaihtuvat riippuen käyttäjän toimintaan tai joihin tiettyihin dynaamisiin muuttujiin (variables). Esimerkiksi <input aria-disabled=”true”>. Tässä tapauksessa ruudunlukija jättää tämän syötteen (*input*) lukematta. Tämän on kuitenkin yleensä helppoa muuttaa ”*false*” tilaan riippuen käyttäjän syöttämästä tiedosta.

Pikalinkit (*shortcut links*)

Koska HTML-kielessä ei ole olemassa mekanismia, jonka avulla käyttäjä pääsee automaattisesti pääsisällön pariin, on tärkeää luoda pikalinkkejä, jotka auttavat näytönlukija käyttäjää sivuttamaan kaikki muut sivuston kohdat esim. päävalikon ja sivuvalikon linkit. Tällä tavoin käyttäjän ei tarvitse navigoida läpi sivuston jokaista kohtaa, vaan pystyy siirtymään suoraan haluamansa sisällön pariin. Tämä nopeuttaa käyttökokemusta huomattavasti.

Hyvänä esimerkkinä on tilanne, jossa käyttäjä etsii Kela.fi-sivustolta ohjetta opintuen hakemiseen.

Parhaiten tiedon varmasti löytää hakukoneen esimerkiksi Googlen avulla (Kuva 1) asettamalla sopivat hakusanat. Yleisimmin käyttäjä valitsee Google hakutuloksista ensimmäisen kohdan.



Kuva 1: Käyttäjä etsii tietoa hakukoneen avulla ja klikkaa auki ensimmäisen hakutuloksen. (17.11.2017 Google.fi)

Avatessaan sivuston auki, ilman pikalinkkiä (Kuva 2), tarvitsisi näytönlukijalla toimivan käyttäjän käydä joka ikinen sivuston linkki ja kohta läpi. Pikalinkin avulla käyttäjä siirtyy automaattisesti pääsisällön, tässä tapauksessa opintotukeen liittyvän tiedon pariin.

Kuva 2: Ilman pikalinkkiä, käyttäjä joutuisi käymään läpi sivuston joka ikisen kohdan päästäkseen käsiksi pääsisältöön. Tällainen toiminta on hidasta ja aikaa vievää. (17.11.2017 Kela.fi)

Pikalinkkejä voidaan tukea ARIAn avulla asettamalla elementteihin roolit. Tällä hetkellä saatavilla olevia rooleja on:

banner

Sivustoon sisältöön liittyvä rooli mikä ilmaisee esimerkiksi web-sivuston nimeä, sivun otsikkoa tai logoa.

navigation

Alue joka kattaa sivuston tai tiedoston navigointilinkit

main

Sivuston tai tiedoston pääsisältö

search

Sivuston hakutoiminto

article

Erittelee pääsisällön pienempiin osiin varsinkin, jos kyseessä on blogiteksti, kommentti blogissa tai keskustelupalstan postaus jne.

complementary

Pääsisältöä täydentävä/tukeva sisältö

contentinfo

Sisältöön liittyvää tietoa antava alisisältö esimerkiksi alaviitteet, tekijänoikeudet, linkit lähteisiin jne.

Useimpiin sivustoihin voidaan lisätä ainakin muutama rooli helpottamaan käyttäjän toimintaa. Esimerkiksi <ul role=”navigation”>, ja <form=”search”> elementit. Tämä toiminto ei ole ainoastaan rajoitettu *rikkaisiin sovelluksiin* mutta niitä voidaan käyttää melkein missä tahansa nykyisessä web-sivustossa

Avustavat teknologiat tarjoavat pikavalintanäppäimiä joiden avulla käyttäjä voi navigoida erilaisten elementtien läpi, hypätä tiettyyn rakenteelliseen elementtiin

(esimerkiksi käyttämällä S-näppäintä hakukoneen (*search*) löytämistä varten) tai tarjota listan sivuston eri elementeistä. Tämän lisäksi, ARIA tarjoaa rooleja erityylisille elementeille esimerkiksi role=”presentation” jota käytetään sivuston taulukoissa. ARIAn avulla käyttäjän on mahdollista tunnistaa lomakkeiden vaadittuja elementtejä, tarjota parempia merkintä- ja kuvaustapoja sekä mahdollistaa välittömän palautteen antamisen näytönlukija käyttäjille.

Tabindexin käyttö

HTML-kielessä vain lomakkeilla ja linkeillä on näppäimistön fokus-tila. Tämä tarkoittaa, että tab-näppäimellä liikkuvien käyttäjien näytönlukija lukee vain linkkien ja lomakkeiden tiedot ja jättää esimerkiksi otsikot lukematta. Tällainen oletuskäytäntö heikentää näytönlukija käyttäjien käyttökokemusta ja ymmärtävyyttä. Yhtenä hyvänä esimerkkinä näytönlukijan ongelmallisuudesta fokus-tilan kanssa on virheilmoitukset. Tavalliset käyttäjät näkevät virheilmoitukset normaalisti, kun taas näytönlukija jättää virheilmoitukset usein lukematta. Tällaisessa tapauksessa virheilmoitus tarvitsisi fokus-tilan käyttöönottoa. Miten sellainen pystytään sitten asettamaan?

ARIA tarjoaa ratkaisun **tabindex**-ominaisuuden avulla. ARIAn avulla tabindexiä voidaan hyödyntää sivuston kaikissa elementeissä. Asettamalla elementin tabindexin nollaan (tabindex=”0”), elementti siirtyy ruudunlukijan tab-näppäimen lukulistalle. Tämä tarkoittaa, että selain pysähtyy ja asettaa fokuksen elementtiin HTML-tiedoston navigointijärjestyksen mukaisesti. Tämä mahdollistaa lisätoimintojen asettamisen elementtiin, kuten esimerkiksi käynnistämistoimintoja (*triggering*) elementtien vastaanottaessa näppäimistö fokuksen tai kun käyttäjä painaa näppäintä elementin ollessa fokus-tilassa.

Tabindex arvo -1 mahdollistaa elementin vastaanottamaan fokuksen, mutta ainoastaan kun fokus on asetettu ohjelmallisesti – tarkoittaen että käyttäjän aktivoidessa (klikkaamalla) elementin linkin tai kun fokus on asetettu manuaalisesti koodaamalla. (WebAim 2013)

Esteettömyys pienoisohjelmissa

Monille web-sivustojen toteuttajille englanninkielinen sana *widget* saattanee olla tuttu jo entuudestaan. Suomenkielellä suoraan käännettynä puhutaan vempaimesta, mutta harvakseen *vempain* sanalla löytyy oikeanlaisia hakutuloksia. Sanastokeskus TSK ry:n mukaan *widget* sanan parempi suomenkielinen käännös olisi *pienoisohjelma*.

Pienoisohjelma on ryhmän mukaan *”Graafiseen käyttöliittymään erillisen lisätoiminnon tuova yksinkertainen ohjelma.”* (Sanastokeskus TSK ry)

Pienoisohjelmaa voitaisiin kuvata interaktiivisena elementtinä, joka luodaan ja ohjataan skriptauksen (esim. JavaScript) avulla. Hyviä esimerkkejä ovat liukukuvagalleriat (*sliders*), pudotusvalikot (*drop-down menus*), sivulle liukuvat valikot (*fly-out menus*), sivustokartat (*tree systems*), veto- ja pudotustyökalut (*drag&drop controls*) sekä automaattisesti täydentyvät tekstilaatikot (*auto-completing text boxes*). Tällaisten pienoisohjelmien esteettömyys ei tapahdu luontaisesti, jonka lisäksi HTML-kielelle on hyvin rajallinen mahdollisuus asettaa esteettömyyttä lisääviä merkintöjä pienoisohjelmiin. Lyhyesti sanottuna, pienoisohjelmiin on mahdollista koodata esteettömyyselementtejä, mutta usein ne ovat monimutkaisia ja aikaa vieviä. ARIA helpottaa pienoisohjelmien esteettömyyttä huomattavalla tavalla, asettamalla niihin aikaisemmin mainitsemani *roolin*, *tilan* tai *ominaisuuden*. Näin web-sivuston toteuttajan ei tarvitse erikseen kirjoittaa pienoisohjelmien koodiin esteettömyysominaisuutta.

Yllämainittujen esimerkkien avulla voidaan todeta ARIAn olevan monipuolinen työväline. ARIA ei poista kaikkia esteettömyyttä haittaavia tekijöitä, mutta sen rooli on merkittävä esteettömien web-sivustojen toteutuksessa. ARIA-teknologia esiintyy monissa yleisimmissä web-kirjastoissa, kuten myös Angular Material-kirjastossa.

2.6 Ohjelmistokehykset ja Angular

Googlen kehittämä Angular (aikaisemmin AngularJS) on yksi suosituimmista ohjelmistokehyksistä, jota ovat käyttäneet lukuisat yritykset web-sovelluksissaan.

Angular on *MCV-tyylinen* ohjelmistokehys, joka toimii parhaiten monimutkaisissa front-end sovelluksissa, jossa halutaan käyttää vain yhtä vakioitua ohjelmistokehystä. Angular tarjoaa hyvän ratkaisun järjestelemään HTML, JavaScript ja CSS-tiedostot pitäen front-end koodin siistinä. (Pollack 2016)

2.6.1 JavaScript

JavaScript on yksi yleisimmin käytetyistä koodikielistä front-end toteutuksessa. JavaScript (usein lyhennettynä JS) on kevyt, objektiivinen kieli, jolla on hyvin kehittyneitä toimintoja (*first-class functions*). Se tunnetaan parhaiten verkkosivujen komentosarjakielenä, mutta sitä käytetään usein myös selaimen ulkopuolella. JavaScript on prototyyppipohjainen, monen paradigman skriptauskieli, joka on dynaaminen ja tukee objektiivisia, imperatiivisia ja funktionaalisia ohjelmointityylejä. JavaScript toimii verkossa asiakkaan puolella, jota voidaan käyttää suunnittelemaan ja ohjelmoimaan, miten verkkosivut käyttäytyvät tapahtuman tapahtuessa. JavaScript kieltä on helppo oppia ja se on myös tehokas komentosarjakieli, jota käytetään laajasti verkkosivujen toiminnallisuuden hallintaan. (Mozilla MDN 2018)

2.6.2 Ohjelmistokehysten kriitikot

Monet nk. JavaScript fundamentalistit korostavat usein sitä, kuinka tärkeää on kirjoittaa omaa JavaScript kieltä, mieluummin kuin käyttäisi minkäänlaista kirjastoa tai ohjelmistokehystä. (Walsh 2007)

Heidän mukaansa:

1. JavaScript koodin tekemistä ei voi oppia syvällisesti, kun ohjelmistokehyksien tarjoamia pikafunktioita/toimintoja käytetään.
2. JavaScript ohjelmistokehykset ovat ylipaisuneita ja sisältävät paljon ylimääräistä koodia, jota et tule todennäköisesti koskaan käyttämään.
3. Käyttäjät joutuvat lataamaan paljon ylimääräisiä tiedostoja, jota he eivät tarvitse.
4. Toisten tekemiin koodeihin ei voi täysin luottaa.

2.6.3 Ohjelmistokehysten puolestapuhujat

Yllämainitut väittämät ovat hieman harhaanjohtavia. On hyvin tärkeää, että opit ymmärtämään JavaScript kielen perusteet jollakin tavalla ja osaat kirjoittaa omaa koodia, mutta ohjelmistokehysten avulla nopeutat oppimistasi ja parannat ajankäyttöäsi tulevaisuudessa, koska ohjelmistokehykset tarjoavat paljon käytännön ratkaisuja nopeasti. (Walsh 2007)

Esimerkiksi

1. Koodaajan ei tarvitse kirjoittaa kaikkea koodia jatkuvasti uudestaan. Tavallisimmat koodinrakenteet löytyvät nopeinten ohjelmistokehyksistä, joiden avulla voit keskittyä paremmin ratkaisemaan oleellisimpia koodaushaasteita.
2. Ohjelmistokehykset tarjoavat hyviä koodausratkaisuja, missä koodi on lyhyttä ja selkeää. Monet, varsinkin aloittelevat koodaajat – minä mukaan lukien – kirjoittavat helposti pitkiä koodinpätkiä, jotka voitaisiin kirjoittaa lyhyemmin. Tämä selkeyttää koodin rakennetta ja on helpommin luettavissa, jota tarvitaan varsinkin, kun yritetään korjata bugeja. Tämän lisäksi se myös parantaa sivuston latautumisprosessia.
3. Säästää aikaa. Tämä on erityisen tärkeää varsinkin asiakastilausprojekteissa, missä noudatetaan tiukkaa aikataulua.
4. On myös hyvä muistaa, että JavaScript ohjelmistokehyksien kehittäjät ovat todennäköisesti sinua taitavampia ja kokeneempia JavaScript koodaajia. Tämä taas tarkoittaa sitä, että heidän kirjoittamassa koodissa on vähemmän virheitä, mitkä aiheuttaisivat bugeja. Tämän lisäksi monilla ohjelmistokehyksillä, kuten Angular, on laaja kannattajakunta, joilta voi aina tarvittaessa kysyä apua tai saada tukea ikävissä koodausongelmissa.
5. Ohjelmistokehyksen avulla pyörivät sivustot toimivat usein myös nopeammin, koska usein ohjelmistokehyksen kehittämisessä on pohja-ajatuksena luoda työväline, mikä helpottaa massakävijä sivustojen tekemistä (esimerkiksi Huuto.net tai Tori.fi-tyyliset sivustot).

2.6.4 Miksi valita Angular ohjelmistokehys?

Jos käytät JavaScriptiä luodaksesi dynaamisen web-sivuston, Angular on oiva valinta ohjelmistokehykseksi. Miksi?

1. Angular organisoi JavaScript koodin puolestasi
2. Angular on luo sivustot responsiivisiksi, mikä mahdollistaa sivuston käytön eri laitteilla.
    a. Responsiivisuus mahdollistaa myös sujuvan kommunikoinnin back-end sovellusten kanssa.
3. Angular on helppo testata, jolloin pystyt rakentamaan helposti ylläpidettävän sivuston.

2.6.5 AngularJS ja Angular – erot toisiinsa

Kuten mainitsin opinnäytetyöni alussa, käytin ensimmäisessä Angular-projektissani AngularJS enkä nykyistä Angularia. Minun oli alun perin tarkoitus tehdä opinnäytetyöni AngularJS-tekniikasta, mutta koska se on vanhempi versio nykyisestä Angularista, sen käyttö tulevaisuudessa tulee vähenemään entisestään. Tämän lisäksi Google tulee todennäköisesti jossain vaiheessa lopettamaan AngularJS tuen, joten Angular on siinä mielessä parempi vaihtoehto opeteltavaksi. Miten AngularJS ja Angular sitten eroavat toisistaan?

Ensinäkin, AngularJS tarkoittaa kaikkia 2.0 vanhempia versioita Angularista, kun taas Angular kuvaa nykyistä ja kaikkia 2.0 eteenpäin olevia versioita. Muita huomattavia eroja ovat esim.

1. Nopeus – Angular on nopeampi.
2. Komponentit – Angular käyttää komponentteja ohjainten (*controllers*) sijasta.
3. Yksinkertaisemmat toimintaohjeet (*directives*) – Omien toimintaohjeiden tekeminen on paljon helpompaa.
4. Komponenttien tiedonsiirto sivustolle on sujuvampaa (*Intuitive Data Binding)*

Muitakin eroja löytyy, mutta nämä ovat ehkä mainittavimpia eroavaisuuksia Angularin ja AngularJS välillä.

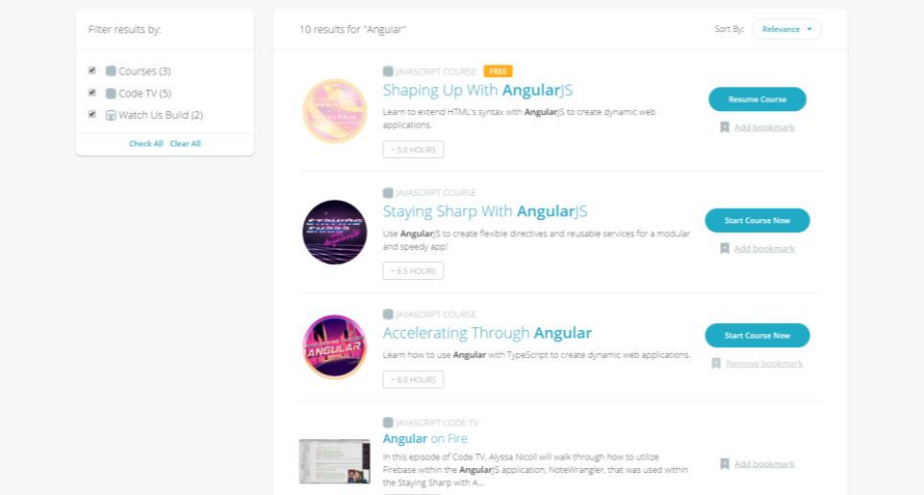
2.7 Opetusmateriaalia netissä

Ohjelmointityössä pitää jatkuvasti olla ajan tasalla uusimpien tekniikoiden ja päivitysten osalta. Esimerkiksi vuonna 2015 JavaScriptiin julkaistiin päivitetty versio, missä monet tavallisimmatkin koodinpätkät saivat uudenlaisen ilmeen. Kyseistä päivitystä kutsutaan nimellä ES2015 ja sitä käytetään yhä useammin moderneissa selaimissa.

Kysyin pientä listaa vanhalta mentoriltani, joka on itse toiminut ohjelmoijan työssä jo useamman vuoden. Hän pyrkii jatkuvasti kehittämään taitojaan eteenpäin sekä pysymään ajanvirrassa mukana lukemalla viimeisimmät uutiset ohjelmointimaailmasta.

Uusien ohjelmointitekniikoiden, työkalujen ja kielin opettelemiseen hän suositteli seuraavia sivustoja:

- Code Academy (ilmainen ja monipuolinen)
- CodeSchool.com (tarjoaa sekä ilmaisia että maksullisia kursseja (kuva 3). Edullinen kuukausihinta ja selkeät videotutoriaalit sekä harjoitukset. Suosittelen tätä varsinkin aloitteleville koodareille)



Kuva 3: Angular kursseja CodeSchool.com-sivustolla. "Shaping Up With AngularJS" on ilmainen alkeiskurssi sivustolla ja hyödyllinen, jos haluaa tutustua tekniikkaan videoluentojen avulla.

Sitten taas ajankohtaisimmat uutiset voit lukea seuraavilta sivustoilta:

- The Verge (ehkä suosituimpia IT-uutisten nettisivustoja)
- Slash Dot (yksi parhaimmista sivustoista löytämään ajankohtaisimmat ohjelmointi uutiset nopeasti)
- DZone (yli miljoonan ohjelmoijan verkosto sisältäen myös sisäpiiriuutisia)

3. PROJEKTIN VALMISTELU JA TOTEUTUS

Ennen projektin aloittamista on tärkeää tehdä pienimuotoisia valmisteluja. Tätä voitaisiin verrata esimerkiksi ruuanlaittoon. Projektia voitaisiin myös omalla tavallaan kutsua reseptiksi. Reseptiin kuuluu usein

- valmistelu
- toteutus
- lopputuloksen julkaisu.

Valmistelussa otetaan huomioon raaka-aineet ja niiden hinnat, käytettävät työkalut, reseptin eri vaiheet ja kokonaisaika. Ennen kuin aterian valmistus voi alkaa, pitää kokin käydä kaupassa ostamassa tarvittavat raaka-aineet ja työkalut. Ruuanlaittoon kuuluu siis huolellinen suunnittelu, mikä voi aina aloittelijoista tuntua kovalta työltä, mutta huolellinen valmistelu ja toteutus takaa herkullisen lopputuloksen. Sama pätee myös ohjelmointiprojektin suhteen. Useimmiten saatamme saada työksemme projektin (reseptin), joka eroaa huomattavasti aikaisemmin tekemistämme projekteista. Vaiheet eivät välttämättä poikkea toisistaan, mutta maku kyllläkin.

Mikä on siis ohjelmistoprojektin reseptin valmistelussa oleellisinta?

- **Raaka-aineet ja niiden hinta** = Jota asiakas haluaa, millä aikataululla ja mihin hintaan (budjetti?)
- **Työkalut** = Millä tekniikoilla ja työkaluilla voimme toteuttaa kyseisen projektin parhaiten (*front-end* ja *back-end* ratkaisut)?
- **Vaiheet** = Jota eri vaiheita pitää olla, että pääsemme haluttuun lopputulokseen (kick-off tapaaminen asiakkaan ja tiimiläisten kesken, käyttäjätutkimukset, UX, UI ja esimerkiksi back-end tietokanta rakenteiden suunnitelmat sekä testaus, toteutus, julkaisu).
- **Kokonaisaika** = Eli aikataulu. Sujuvin ja ehkäpä tällä hetkellä tehokkain aikataulutustekniikka on Scrum, missä eri vaiheet on tehty lyhyiksi sprinteiksi. Scrumin avulla eri osapuolet kommunikoivat jatkuvasti keskenään, mikä on tärkeää varsinkin testaus- ja kehityspuolten välisessä yhteistyössä.

Vaikkakin opinnäytetyössäni oleva projekti on toteutettu harjoitusmielessä, oli minusta tärkeää tuoda esille myös todellisen projektin eri vaiheet. Asiakastilaus projekteissa aikataulu ja laatu ovat ensisijaisia, joten sujuvan ja fiksun ajankäytön, suunnittelun ja työskentelytekniikan harjoitteleminen on hyvä aloittaa jo koulunpenkillä.

3.2 Valmistelu

Kokosin pienen muistilistan asioista, joihin kannattaa kiinnittää huomiota, kun aloittaa uutta harjoitusprojektia:

- Työkalut - Jota ohjelmia/työkaluja tarvitsen projektin toteuttamista varten?
- Aikaisempi tietopohja - Jota tietolähteitä voisin hyödyntää projektin toteutuksessa (esim. tutkimustulokset)?
- Uuden tiedon hankinta – Miten voin oppia uutta ja mistä löydän luotettavaa opiskelumateriaalia?
- Ongelmatilanteet - Mistä/keneltä kysyn apua, jos tarvitsen sitä (esim. mentorit tai yhteistyö yritykset)?

Tein aikaisemman Angular projektini Apple Macbookin kannettavalla. Macbook eroaa koodaustyökalujensa osalta Windowsista, vaikkakin työskentelytapa on pitkälti samanlaista. Syy miksi halusin toteuttaa opinnäytetyöni Windowsin koneella, lähti pitkälti siitä ajatuksesta, että hyvin monet opetukseen käytettävät koneet ovat vielä tänäkin päivänä Windows-vetoisia. Jota tulee työelämään, varsinkin IT-alan yrityksiin, Mac on nousemassa käytetymmäksi koneeksi, kuin Windows. Yksi syy tälle saattaa olla Macin nopea toimivuus sekä turvallisuus. Mac on rakennettu Unix-pohjaan mikä parantaa tietokoneen suojaa huomattavasti. Tämän lisäksi Macbookin kannettavat pystyvät pitämään pystyssä useampaa raskasvetoista ohjelmaa (esim. Adoben ohjelmat), kun taas monet Windowsin käyttöjärjestelmällä toimivat kannettavat eivät välttämättä kykene suorittamaan edes yhtäkään ilman häiriöitä. (Haslam 2018)

Kuitenkin oman kokemukseni pohjalta sanoisin, että Windows on aloittelijoiden kannalta hyvä vaihtoehto, varsinkin jos henkilöllä ei ole aikaisempaa kokemusta Applen koneista. Kun henkilö on saanut varmuutta uuteen tekniikkaan voi hän tulevaisuudessa testata tämän käyttöä paremmin myös Applen tietokoneella.

Kokosin listan tarvittavista Windows 10 käyttöjärjestelmässä käytettävistä työkaluista ja niiden mahdollisesta Macbook vaihtoehdosta. Koska toteutin työni itsenäisesti, en ole listannut mukaan sellaisia työkaluja jota käytettäisiin tiimityöskentelyssä (esim. BitBucket). Selitin BitBucketin käyttötarkoituksen lyhyesti kolmannen kappaleen termeissä (2.2). BitBucket on tärkeää tietää ainakin nimeltä, koska se tulee usein esille ohjelmointityöskentelyssä.

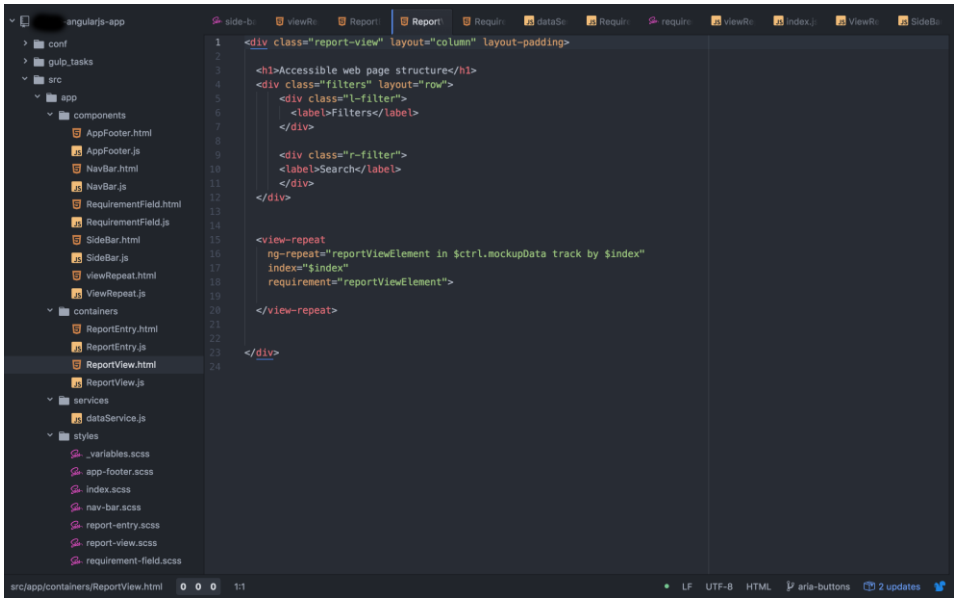
Huomioi, että monia työkaluja voi käyttää molemmilla käyttöjärjestelmillä.

Taulukko 1: Angular-projektia tukevia työkaluja/sovelluksia

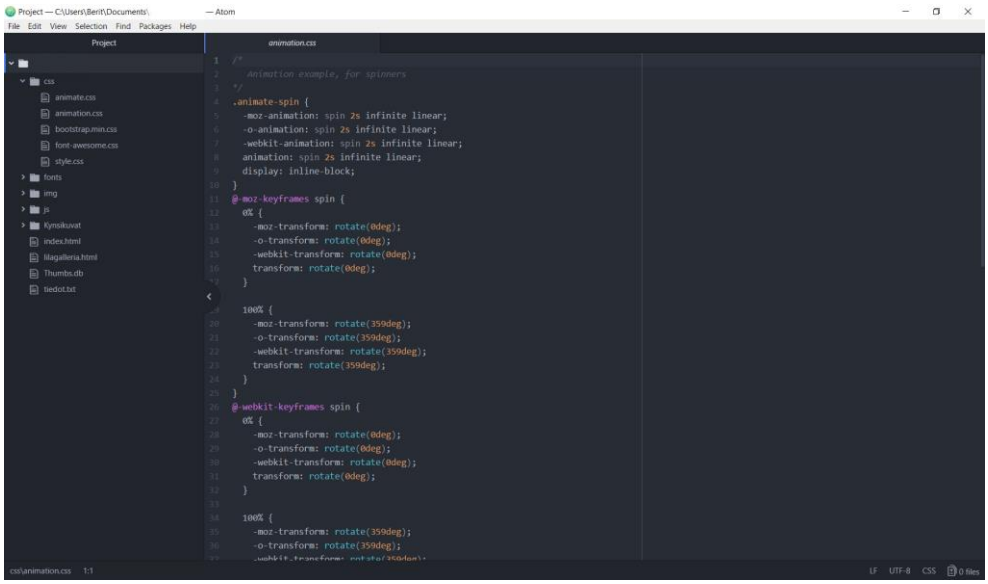
Työkalun käyttötarkoitus	**Windows**	**MacOS**	**Syy valinnalle**	**Lataus vai online?**
Projektin tehtävien listaus	Trello.com	-||-	Helppokäyttöinen ja selkeä online-työkalu	Online
Projektin aikataulutus (taulukointiohjelma)	Google Docs	-||-	Helppokäyttöinen, nopea ja selkeä	Online
Koodaustyö	Atom (vaihtoehtoisesti Notepad++)	Atom	Helppokäyttöinen ja selkeä sekä Windowsissa että Macissä.	**Lataus**

Koodin virhetestaus	Command Prompt (sisäänrakennettu)	Terminal (sisäänrakennettu)	Nopea virheiden/bugien löytäminen sekä työkalujen/ohjelmien lataus	-
Oletusselain testaukseen ja sivuston katseluun	Google Chrome	-||-	Selain on usein ajantasaisin front-end koodauksen puolesta.	Lataus
Luonnostelu ja visuaalinen suunnitelma (UX&UI)	Mockflow.com (vaihtoehtoisesti Adobe Illustrator/Photoshop)	-||-	Nopea ja helppokäyttöinen Wireframien tekoon.	Online

Käytin aikoinani Notepad++ koodaustyöhön Windowsissa, mutta opeteltuani Atomin käytön, en näe syytä palata Notepad++ pariin. Atom on siitä mainio, että se järjestelee projektin selkeästi kansioihin sekä siihen on mahdollista ladata erilaisia lisäosia parantamaan koodauksen sujuvuutta.



Kuva 4: Havaintokuva Atomin sisällä olevasta projektista Macbookin koneessa (ladattujen lisäosien kanssa)



Kuva 5: Atomin näkymä Windowsissa (ilman ladattuja lisäosia)

Atom ei juuri eroa Windowsin (kuva 5) ja MacOSin (kuva 4) versioissa. Tämän vuoksi suosittelenkin Atomin opettelemista, koska esimerkiksi Notepad++ ei ole saatavana Mac-versiona.

Jota taas tulee UX ja UI suunnitelmiin, käyttäisin asiakastilausprojekteissa Adoben Illustratoria luodakseni *wireframet* ja sivuston visuaalisen suunnitelman.

Opinnäytetyön esimerkkityössä päätinkin kuitenkin käyttää yksinkertaistettua vaihtoehtoa, eli *Mockflow.comin* online-työkalua, jonka avulla pystyn rakentamaan nopeasti yksinkertaisen UX- ja UI-suunnitelman. Tämä on myös oivallinen työkalu aloitteleville suunnittelijoille.

Hyvin harvoin ohjelmoijan itsessään tarvitsee suunnitella käyttöliittymän visuaalista tai toiminnallista puolta, vaan yrityksissä on usein omat UX ja UI-suunnittelijat tukemassa ohjelmoijien työtä. *Ohjelmoijat siis vain toteuttavat suunnitelmia*. Itse näen silti eduksi sen, että ohjelmoija osaa tehdä hieman myös visuaalista suunnittelua, koska hyvin harvoin näin on. Se voi myös parantaa työnsaantimahdollisuuksia, koska erottuu joukosta.

3.2.1 Aikataulutus

Aikataulutus on myös hyvin oleellinen osa harjoitusprojektia, koska sen avulla työskentelet tehokkaammin ja olet selkeämmin perillä projektisi tilanteesta. Aikataulussa kiinni pysyminen vaati itsekuria ja päättäväisyyttä eikä se ole mitenkään helppoa. On kuitenkin tärkeää, että opettelee itsekuria jo tässä vaiheessa, koska työelämässä se tulee erottamaan jyvät akanoista. Yksinkertaisin tapa on koota taulukolle (*Google Docs*) lista tehtävistä asioista ja aikatauluttaa ne eri viikoille. Viikossa voit kuluttaa esimerkiksi 6-12 tuntia harjoitusprojektin tekemiseen jonka voitaisiin jakaa useammalle päivälle tai esimerkiksi vapaapäiville. Yhtenä hyvänä sääntönä voitaisiin pitää seuraavaa: ”*Sivu päivässä/Tehtävä päivässä*”. Pääasia on, että teet jotain ja saat tavoittelemasi työtehtävät tehdyksi.

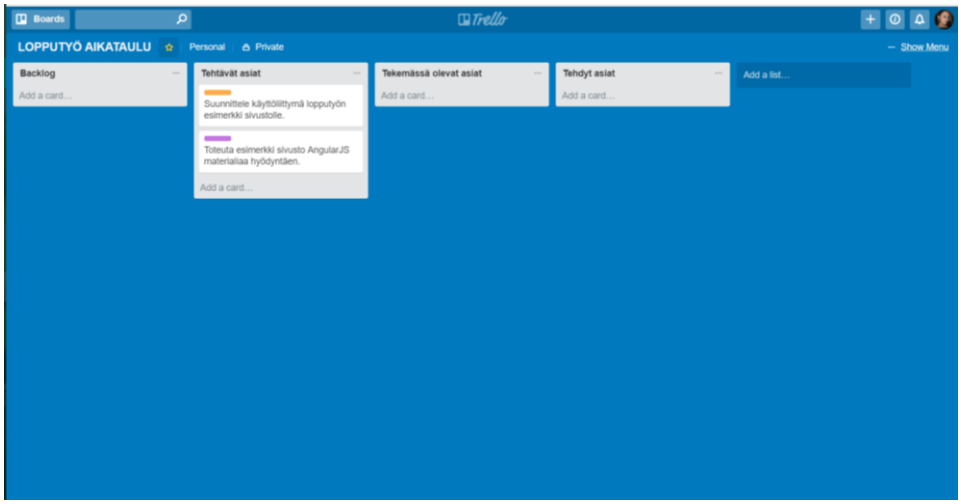
Tärkein ja ehkä oleellisen päivämäärä on työn *deadline*, koska se motivoi sinua pääsemään haluttuun tavoitteeseen tietyn ajan kuluessa. Harjoitusprojekteihin olisi ihan hyvä antaa itselleen pari tai kolme kuukautta aikaa, jotta ehdit tutustumaan ja opettelemaan uusien tekniikoiden käyttöä tarpeeksi mieleenpainuvasti.

3.2.2 Tehtävien listaus

Tehtävien listausta voi selkeyttää entisestään niin sanotun online-korttitaulun avulla (*Trello.com*). Sivustossa luodaan siis uusi taulu (projekti) mihin lisätään

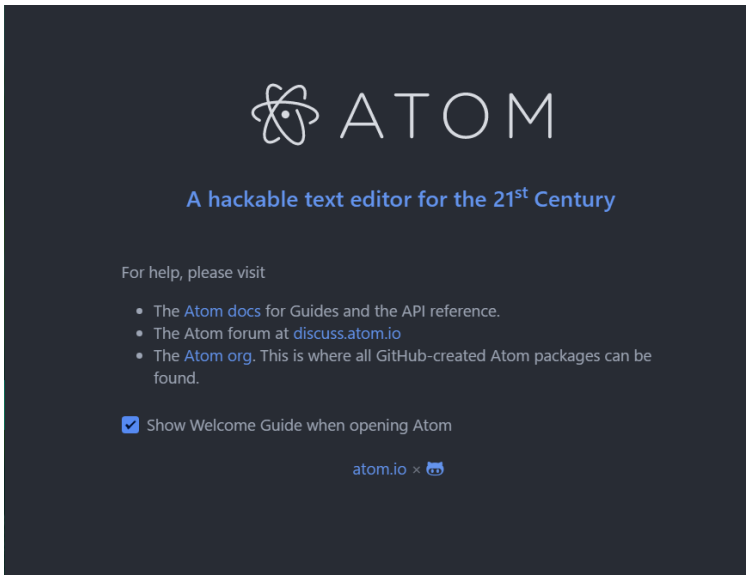
erilaisia tehtäviä aihealueen tai kiireellisyysluokan perusteella (kuva 6). Useimmiten asiakastilaus projekteissa tiimit käyttävät ns. kiireellisyysluokkalajittelua jolloin luodaan neljä taulua:

- **Backlog** – korttiin listataan kaikki mahdolliset tehtävät (myös ne jota ei välttämättä ehditä toteuttaa)
- **To do** – korttiin listataan kaikki tärkeimmät ja oleellisimmat tehtävät mitkä on PAKKO toteuttaa, jotta projekti saadaan kunnolla tehtyä.
- **Doing** – korttiin listataan taas ne tehtävät, joita olet jo alkanut tekemään/tekemässä tällä hetkellä.
- **Done** – korttiin listataan tehtävät, jotka olet saanut jo tehtyä.



Kuva 6: Esimerkki Trellon käyttämisestä. Korttien sisälle listataan asioita, jotka voidaan merkata eri värin kuvaamaan työnluonnetta. Tässä tapauksessa oranssi väri tarkoittaa UI-suunnittelua ja lila väri taas koodausta.

3.2.3 Atomin lisäosat

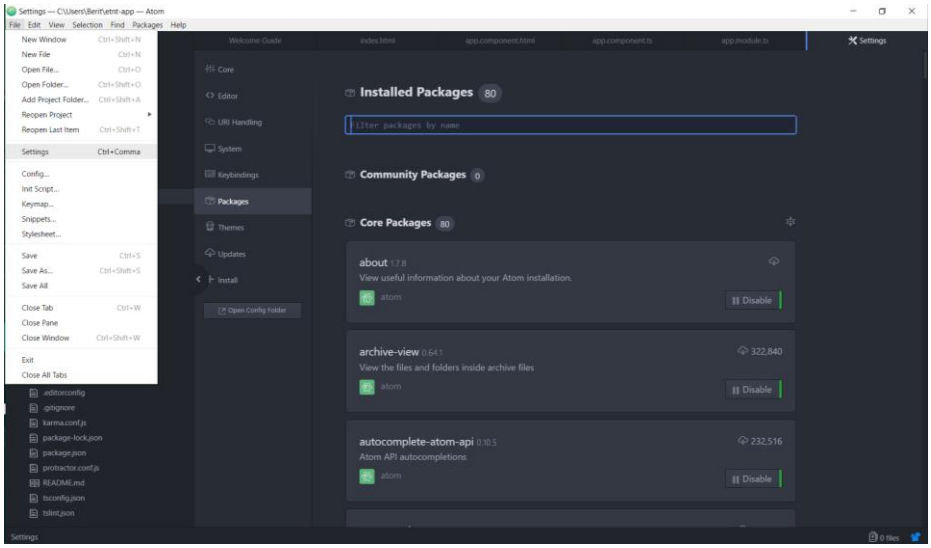


Kuva 7: Atomin aloitusikkuna (versiossa 1.23.3)

Atomi (kuva 7) on siinä mielessä mainio ohjelma, että sen sisältämien lisäosien avulla pystyt tekemään koodauksesta huomattavasti sujuvampaa ja tehokkaampaa. Lisäosat helpottavat esimerkiksi koodauksen rakenteen siistimistä, eri tiedostojen erottamista toisistaan (esim. .css .js .html) sekä löytämään pienet virheelliset koodinpätkät, ilman, että tarvitsee vaivata päätään Command Promptin puolella. Tietysti isompien bugien suhteen Command Prompt on luotettavampi virheentarkistaja kuin Atomin lisäosat.

Atomin lisäosat (kuva 9) jaetaan neljään kategoriaan: yhteisön julkaisemat, sisäänrakennetut, kehitystyön lisäosat ja git lisäosat. Tavallisesti ladatut lisäosat, kuten alla mainitsemani, löytyvät yleensä yhteisön julkaisujen kategoriasta.

Atomiin lisäosiin pääsee Windowsissa *File/Settings* kautta (Kuva 8) ja taas macOS-tietokoneilla *Atom/Preferences* kautta.



Kuva 8: Atomin lisäosien sijainti Windowsin koneella.

Kokosin listan kaikista hyödyllisistä lisäosista:

atom-beautify → järjestää koodin siistiksi automaattisesti.

atom-runner → ajaa koodia Atomin sisällä

autoclose-html → sulkee html-koodin sisällä olevat tägit automaattisesti esimerkiksi

busy-signal → näyttää sinulle APIn välityksellä, kun lisäosa suorittaa tehtävää (eli näet, jos lisäosa toimii tai ei toimi)

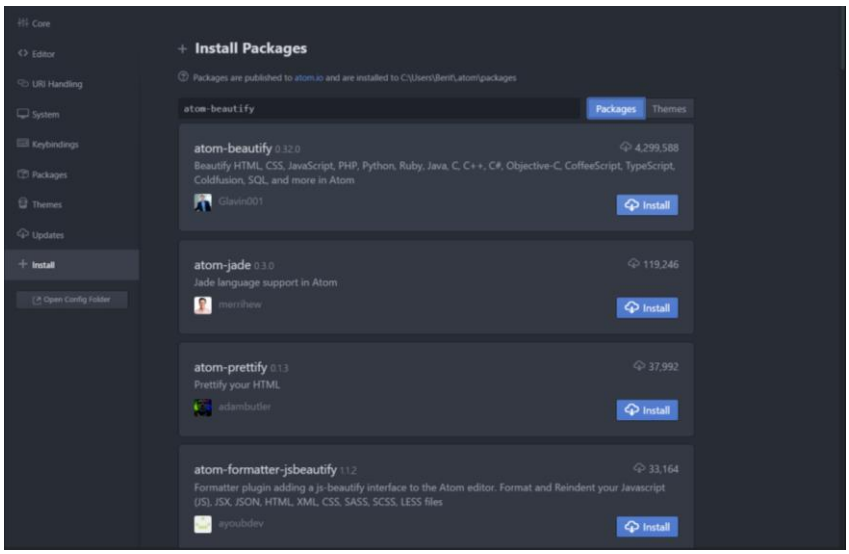
file-icons → asentaa kaikille tiedostoille kuvakkeet, jolloin eri tiedostomuotojen erottaminen helpottuu.

intentions → lisäosa mikä helpottaa koodaustyötä esimerkiksi korostamalla html-tägien alku ja loppusulut.

linter → havaitsee esimerkiksi JavaScriptin sisällä tapahtuvat virheet ja näyttää ne Atomin sisällä

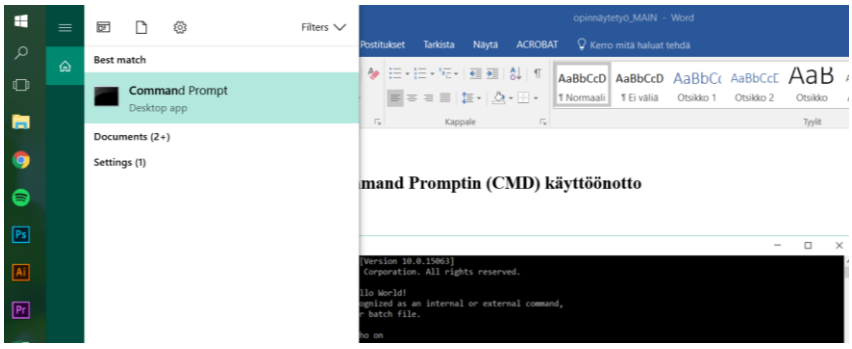
linter-eslint → JavaScriptille omistettu virheidentunnistaja

merge-conflicts → Tätä lisäosaa ei periaatteessa tarvita, jos gitiä ei käytetä, mutta hyödyllinen lisäosa tulevaisuutta ajattelen. Helpottaa huomattavasti ryhmätyöskentelyä, jolloin kahden koodarin väliseen koodiin ilmestyneet konfliktit voidaan yhdistää ja selvittää nopeammin.



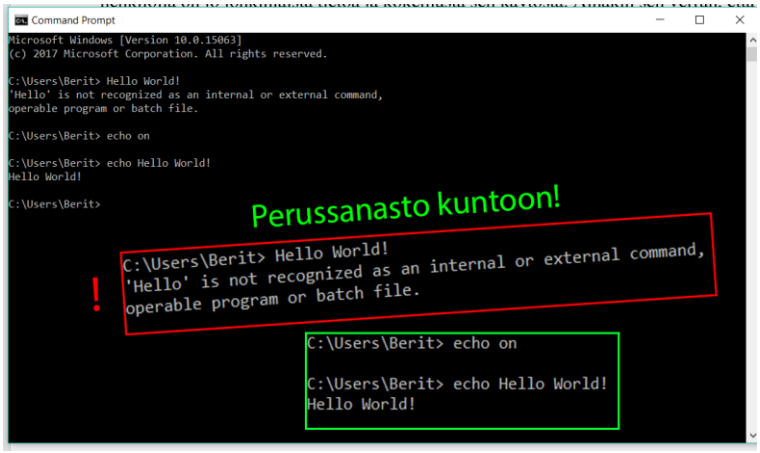
Kuva 9: Atom lisäosien latausikkuna

3.2.4 Command Promptin (CMD) käyttöönotto



Kuva 10: Command Promptin löytää helposti Windowsin hakukoneella.

Nyt päästäänkin siihen ehkä toiseksi jännittävämpään kohtaan, Command Promptin (kuva 10) käyttöön. Ennen Command Promptin käyttämistä tällaisessa projektissa, olisi tärkeää, että henkilöllä on jo jonkinlaista tietoa ja kokemusta sen käytöstä, ainakin sen verran, että hän ymmärtää CMDn perussanastoa (kuva 11).



Kuva 11: Command Prompt näkymä

Miksi Command Prompt on tärkeä hyvän koodaustyön osalta? Kuten jo aikaisemmin mainitsin, se on erinomainen työkalu tarkistamaan bugeja, mutta myös pitämään sivuston ajan tasalla ilman, että pitää itse olla päivittämässä selaimen sivustoa manuaalisesti. Command Promptin saadaan synkronoitua hyvin Angular palvelimen kanssa yhteen niin, että sivusto päivittyy automaattisesti itse. Tämän lisäksi voisin myös tuoda esille tekniikan, jota kutsutaan nimellä Git.

Git on erinomainen tapa tallentaa projektin eri versioita nk. haaroihin (*branches*), jolloin koko projekti ei ole pelkästään vain yhden version varassa. Tämän lisäksi voit sirpaloida sivuston eri komponentit (top-navigation, sidebar jne.) omiin versioihin ja myöhemmin yhdistää (*merge*) ne projektin pääversioon. Edistykselliset koodaajat käyttävät usein tällaista komponenttien sirpalointimenetelmää, jolloin bugit eivät voi vahingoittaa kuin vain yhtä pientä osaa koko projektissa. Tällöin myös bugien korjaaminen käy nopeammin ilman ylimääräistä päänvaivaa.

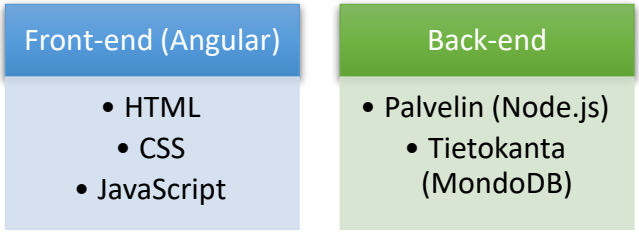
Koska toteutan itse hyvinkin yksinkertaisen harjoitustyön, päätän jättää sirpaloinnin tulevaisuutta varten. Tämä on kuitenkin tärkeä työmenetelmä monissa ohjelmointiyrityksessä, jota ei kannattaa täysin unohtaa.

3.2.5 Angular kehitysympäristön lataus

Ennen kehitystyön aloittamista, on lähdettävä kysymyksestä:

”Jota tarvitsemme kehittäessämme uutta web-sovellusta?”

Oleellisimmat asiat (kuva 12): front-end (HTML, CSS ja JavaScript) ja back-end (palvelin ja tietokanta**).** Tässä tapauksessa front-end kehitys tapahtuu Angularin avulla, kun taas palvelinta ja tietokantaa varten tarvitsisimme Angularia tukevan liitännän. Node.js ja MongoDB ovat mainio ratkaisu tähän. En kuitenkaan tule käyttämään esimerkkityössäni, kuin Node.js palvelinta. (Angular 2018)



Kuva 12: Web-sovelluksen anatomia

Ennen Angularin lataamista, on tärkeää, että lataat koneellesi **Node.JS** ja **npm**.

Node.JS

Node.js on asynkroninen JavaScriptin ajoaika ympäristö ja avoimessa lähdekoodissa pyörivä palvelin (back-end) ohjelmistokehys. Node.js mahdollistaa JavaScriptin käyttämisen omassa tietokoneessa, jonka avulla voit päästä suoraan käsiksi esimerkiksi omiin tiedostoihin, tietokantoihin ja tarkastella tietokoneen verkkoliikennettä. Aikaisemmin tämä ei ollut mahdollista, koska JavaScript toimi lähinnä nettiselaimissa front-endin tukena. Eli siis jota aikaisemmin pystyit tekemään PHPn ja Rubyn avulla, pystyt nyt myös tekemään JavaScriptin kanssa. (NodeJS)

npm

Npm (Node Package Manager) on oletus paketinhallintajärjestelmä Node.js:lle. Npm koostuu verkossa olevasta web-alustasta jossa käyttäjät voivat jakaa JavaScript koodilla kirjoitettuja työkaluja jota voidaan hyödyntää front-end ja back-end puolella, että Command Promptissa. Tämän lisäksi npm käytetään Command Promptissa työkaluna, kun halutaan olla yhteydessä verkossa olevaan npm-alustaan ja ladata uusia työkaluja sivuston kehittämistä varten. Npm työkaluja käytetään sitten Node.js käytön puolella. Node.js sekä npm voidaan ladata selaimen kautta.

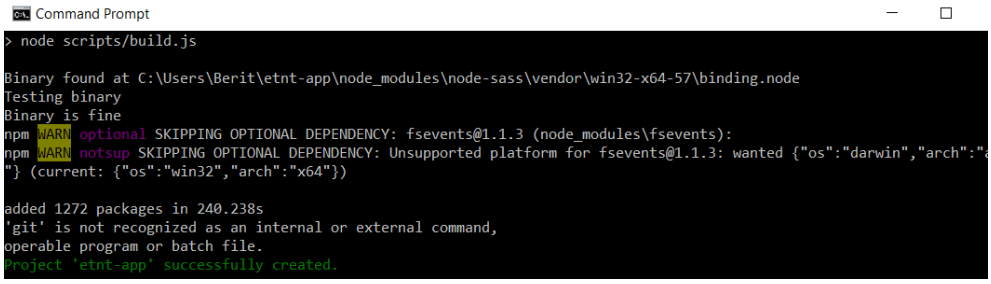
Node.js ja npm latauksen jälkeen ladataan vielä Angular CLI liitäntä, jonka avulla voidaan luoda uusia projekteja, tiedostoja ja suorittaa useita kehitystyövaiheita esimerkiksi testausta, tiedostojen yhteen niputtamista ja käyttöönottoa Command Promptin sisällä. Nämä kaikki kolme Node.js, npm ja Angular CLI ovat hyvin oleellisia ja työskentelyä helpottavia työkaluja Angular projektin tekemisessä. Angular CLI voidaan ladata Command Promptissa, kun npm on ladattuna koneelle:

```
npm install -g @angular/cli
```

Tämän jälkeen voidaan aloittaa ensimmäinen Angular projekti. Tämäkin voidaan tehdä Command Promptissa (kuva 13). On tärkeää kuitenkin huomioida, että projektisi ladataan haluamaasi kansioon. Oletuksena Command Promptia avatessa on C:\Users\kayttaja-nimi. Voit kuitenkin halutessasi luoda uuden kansion `mkdir kansion-nimi` ja siirtyä siihen `cd ..` komennolla.

Uusi Angular sovellus saadaan luotua seuraavaa koodia käyttäessä:

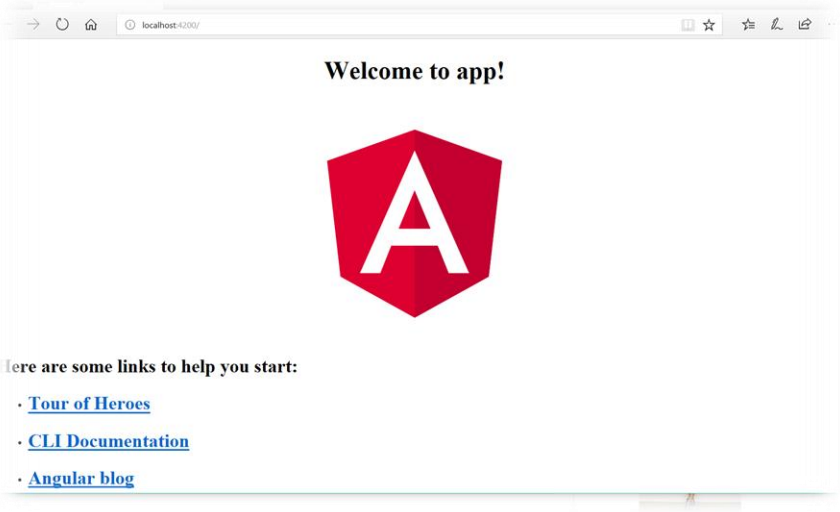
```
ng new sovelluksen-nimi
```



Kuva 13: Havaintokuva Command Promptin sisältä, kun uusi projekti on saatu luoduksi Angular CLIn avulla.

Kun projekti on ladattu kohdekansioon, Node.JS palvelin voidaan käynnistää. Näin sinun ei tarvitse jatkuvasti ladata selaimesi ikkunaa uudestaan tehdessäsi muutoksia sivustoon, vaan Node.JS tekee sen puolestasi. Tällä tavoin pystyt työskentelemään nopeammin ja joustavammin.

Palvelimen aktivoinnin jälkeen pitäisi avautua uusi selainikkuna (kuva 14). Oletuksena palvelin pyörii yleensä osoitteessa **localhost:4200**.



Kuva 14: Palvelimen aktivoinnin jälkeen pitäisi avautua uusi selainikkuna.

3.2.6 Angular Materiaaliin tutustuminen

Kun projekti ja palvelin ovat toimintakunnossa, voidaan siirtyä Angular Materiaalin lataukseen. Angular Material on siitä oivallinen työkalu, että sen valmiiden komponenttien avulla voidaan rakentaa nopealla aikataululla moderneja ja siistejä

sivustoja. Tavallisimmat komponenttiratkaisut ovat Material kirjastossa, ja sieltä poimittavissa. Materialin ulkoasua voidaan myös muokata joustavasti.

Angular Material ladataan CLIn tavoin Command Promptin sisällä. Älä kuitenkaan suorita komentoa samassa ikkunassa, mihin avasit palvelimen yhteyden, vaan avaa uusi Command Prompt ikkuna ja siirry projektikansioon. Tällä tavoin palvelin pysyy auki Angular Material latauksen aikanakin, ja sinun ei tarvitse avata yhteyttä enää uudelleen. Näin pääset projektissasi nopeammin alkuun ilman turhaa työmäärää. Voit myös käynnistää palvelimen vasta Angular Materialin latauksen jälkeen.

Angular Material ladataan aina projektikansioon, jotta sen tiedot voidaan latauksen aikana tallentaa Angular-projektin sisällä olevaan package.json tiedostoon. Lataus tapahtuu seuraavalla koodilla:

```
npm install –save @angular/material @angular/animations @angular/cdk
```

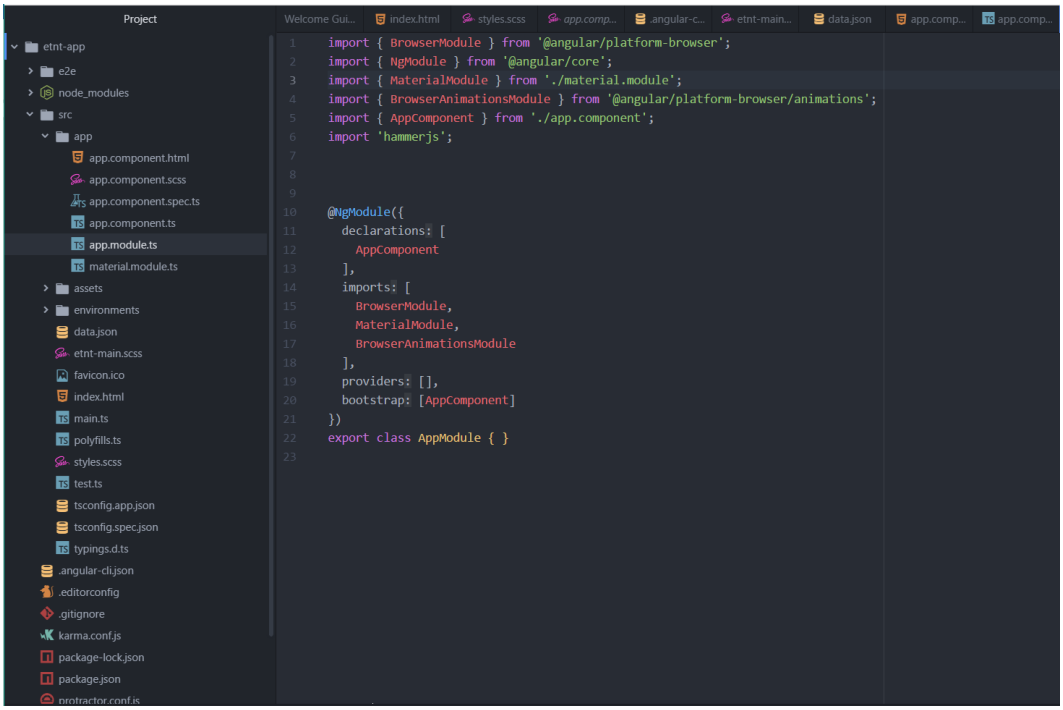
Tämän jälkeen avataan tekstieditori (Atom) ja siirrytään projektikansiossa sijaintiin *src/app.* Tähän kansioon luodaan uusi tiedosto nimeltä *material-modules.ts*. Aikaisemmin kyseistä vaihetta ei tarvinnut tehdä, kun MaterialModule voitiin ladata suoraan sijainnista @angular/material, mutta koska nykyisessä versioissa sitä on muutettu, pitää meidän manuaalisesti linkittää MaterialModule projektiin. Material-modules-tiedoston sisälle kirjoitetaan koodi, johon on lueteltu kaikki ne komponentit, jota projektissa halutaan käyttää (Alligator.io 2018):

```
import { NgModule } from '@angular/core';

import {
  MatButtonModule,
  MatMenuModule,
  MatToolbarModule,
  MatIconModule,
  MatCardModule
} from '@angular/material';

@NgModule({
  imports: [
    MatButtonModule,
    MatMenuModule,
    MatToolbarModule,
    MatIconModule,
    MatCardModule
  ],
  exports: [
    MatButtonModule,
    MatMenuModule,
    MatToolbarModule,
    MatIconModule,
    MatCardModule
  ]
})
export class MaterialModule {}
```

Tämän jälkeen avataan tiedosto *src/app/app.module.ts* mihin on tarkoitus tuoda Angular Material moduulin (kuva 15) material-module.ts tiedoston app.module.ts tiedostoon, mihin on ladattu myös muut moduulit:

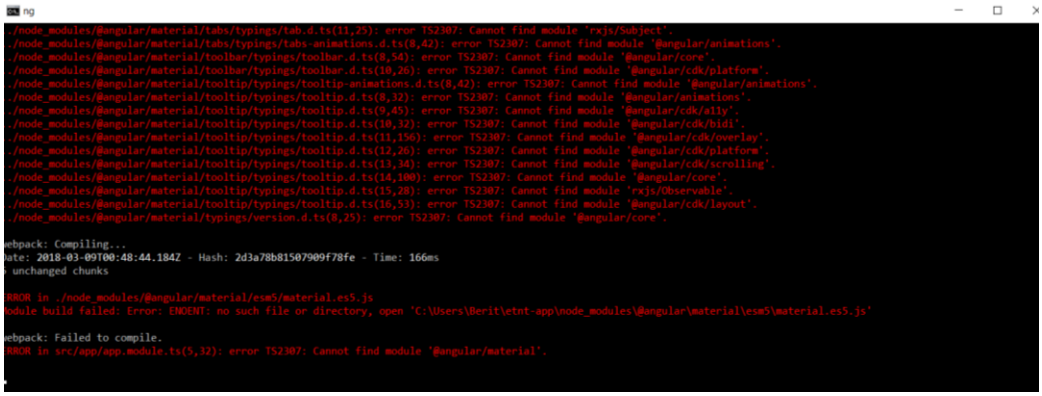
```
import { MaterialModule } from './material.module';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
```

```
MaterialModule,
BrowserAnimationsModule
```



Kuva 15: Angular Material moduulin lataus tekstieditorissa

Angular Material kansion lisäksi meidän pitää myös ladata Angular CDK, jotta tiedostojen siirto koneelle onnistuu. Muussa tapauksessa yllä mainittu JavaScript-koodi ei toimi app.module.ts tiedostossa ja saat virheilmoituksen Command Promptiin (kuva 16). Kohtasin itse kyseisen ongelman, ja kesti jonkin aikaa ennen kuin sain ongelman ratkottua. Olin seurannut erästä Angular videotutoriaalia, missä CDKn latauksesta ei puhuttu, enkä ollut huomannut tarkistaa Angular Materialin viralliselta sivustolta, miten ensimmäinen lataus pitää suorittaa (Angular Material).



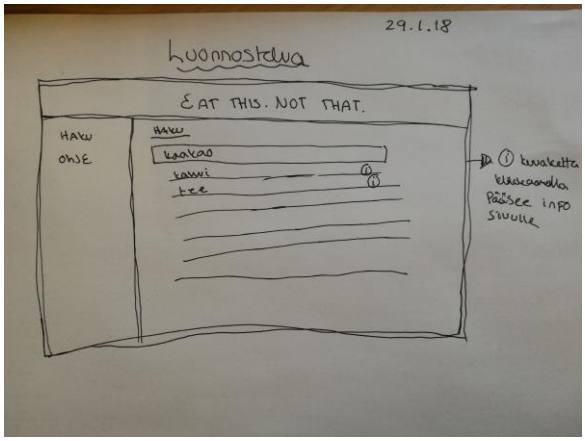
Kuva 16: Command Prompt näyttää virheilmoitukset palvelimen ollessa käynnissä (ng serve)

3.3 Toteutus

3.3.1 Projektin ideasta lyhyesti

Opinnäytetyöni perustuu kahteen keväällä 2018 käymääni kurssiin, joissa oli tarkoitus toteuttaa projektiluontainen web-sovellus. Päätin toteuttaa web-palvelun, missä käyttäjät voivat hakea terveellisiä korvikkeita epäterveellisille ruuille. Sovelluksen nimeksi annoin ”Eat this. Not that”. Ajattelin, että tällaisen harjoitustyön avulla tuon esille Angularin käyttöliittymä suunnittelun ja datakäsittelyn front-end puolella.

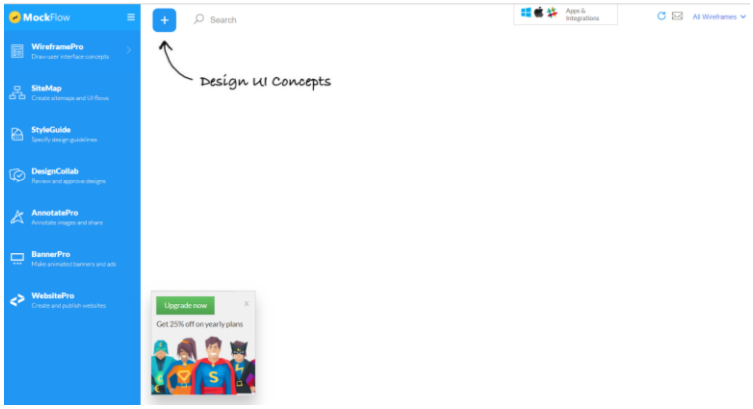
3.3.2 Wireframien teko (yksinkertaistettu UX- ja UI)



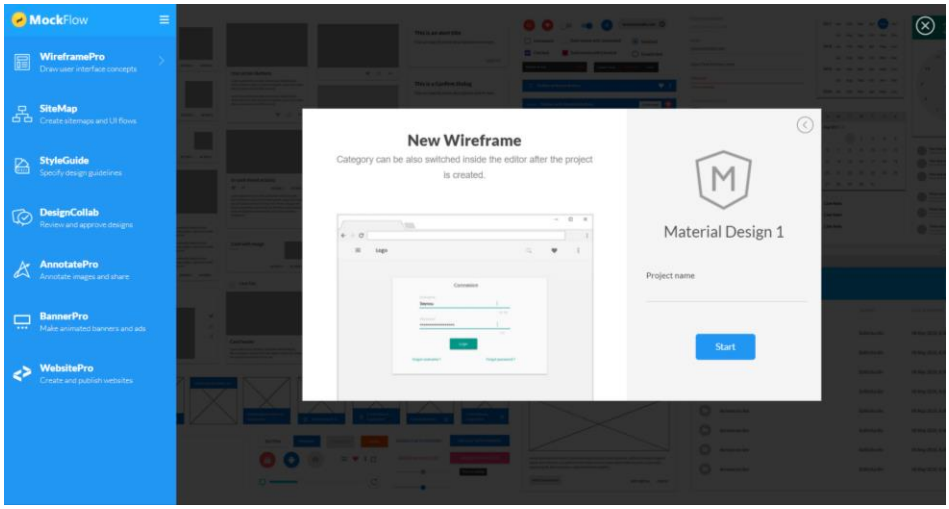
Kuva 17: Nettisivun UI-suunnitelmaa voi luonnostella aluksi paperille.

Esteetöntä sivustoa suunnitellessa (kuva 17), on tärkeää pohtia esimerkiksi näppäimistökäyttäjien toimintaa ja miten sen eroaa tavallisesta käyttäjästä. Tämän lisäksi pitää ottaa huomioon erilaisten komponenttien sopivuus näppäimistökäytössä ja värimaailmaa. Värimaailma on erityisen oleellinen niille henkilöille, jotka eivät erota tiettyjä värejä taustasta vai ovat kokonaan värisokeita. Taustan ja tekstin kontrastin pitää olla tarpeeksi syvä, että näkyvyys toimii.

Käytin tällä kertaa apunani Mockflow web-palvelua (kuva 18), missä saa luotua nopeasti yksinkertaisia wireframeja esimerkiksi kun halutaan rakentaa demo-version jostakin web-sivustosta.

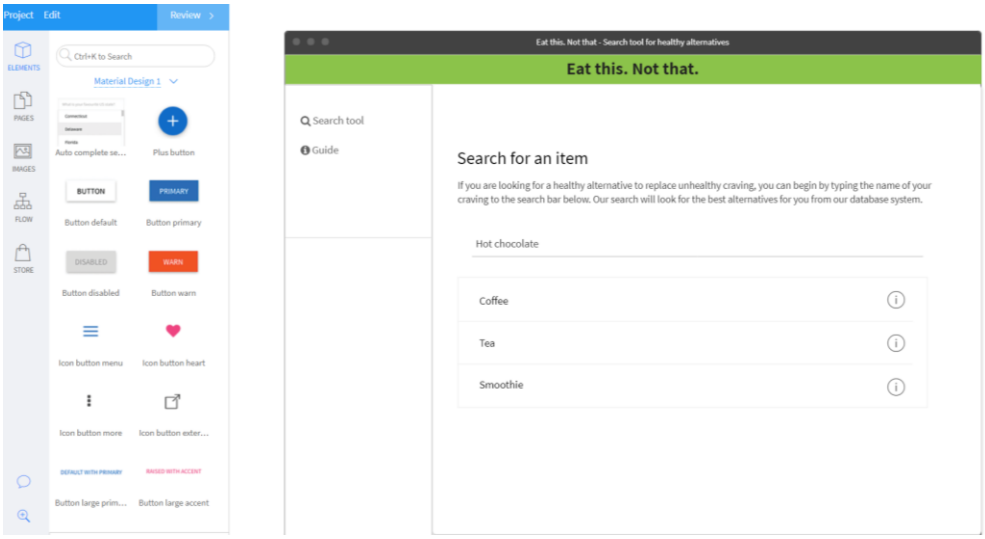


Kuva 18: Mockflown aloitusnäkymä



Kuva 19: Angular materiaaliin perustuen, voidaan käyttää Material Design 1 UI-wireframe pakettia.

Mockflow tarjoaa Wireframe vaihtoehtona Bootstrapin lisäksi myös Material Design paketin (kuva 19), jossa kaikki komponentit ja värit perustuvat Angular Materialiin.

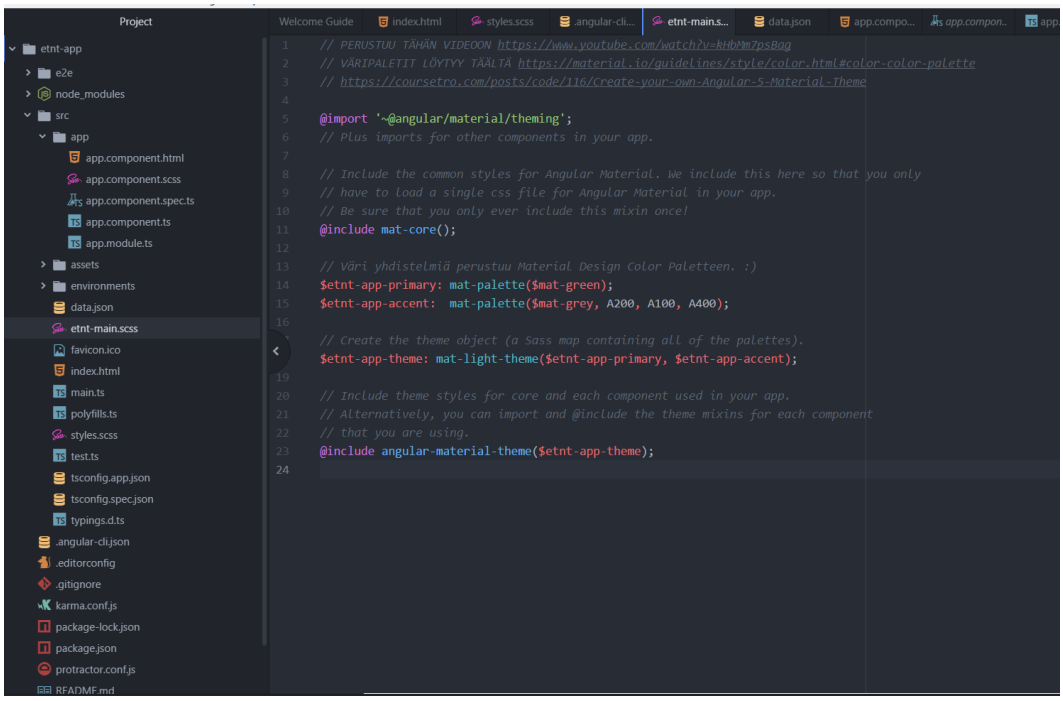


Kuva 20: Esimerkkityön käyttöliittymä suunniteltuna

Päätin luoda yksinkertaisen käyttöliittymäsuunnitelman (kuva 20), missä kuitenkin tulee esille erilaisia sivuelementtejä. Samat komponentit löytyvät myös Angular Material kirjastosta.

3.3.3 Käyttöliittymän rakentaminen

Ensimmäiseksi halusin määritellä käyttöliittymän värimaailman. Muutin css-tiedostot scss-tiedostoiksi, jolloin koodaustyöstä tulee siistimpää ja yksinkertaisempaa ilman pitkiä koodinpätkiä. Tämän jälkeen loin uuden scss-tiedoston nimeltä ”etnt-main.scss” johon asetin Angular Materialiin pohjautuvan väripaletin, joka sopii myös käyttöliittymäsuunnitelmani värimaailmaan.



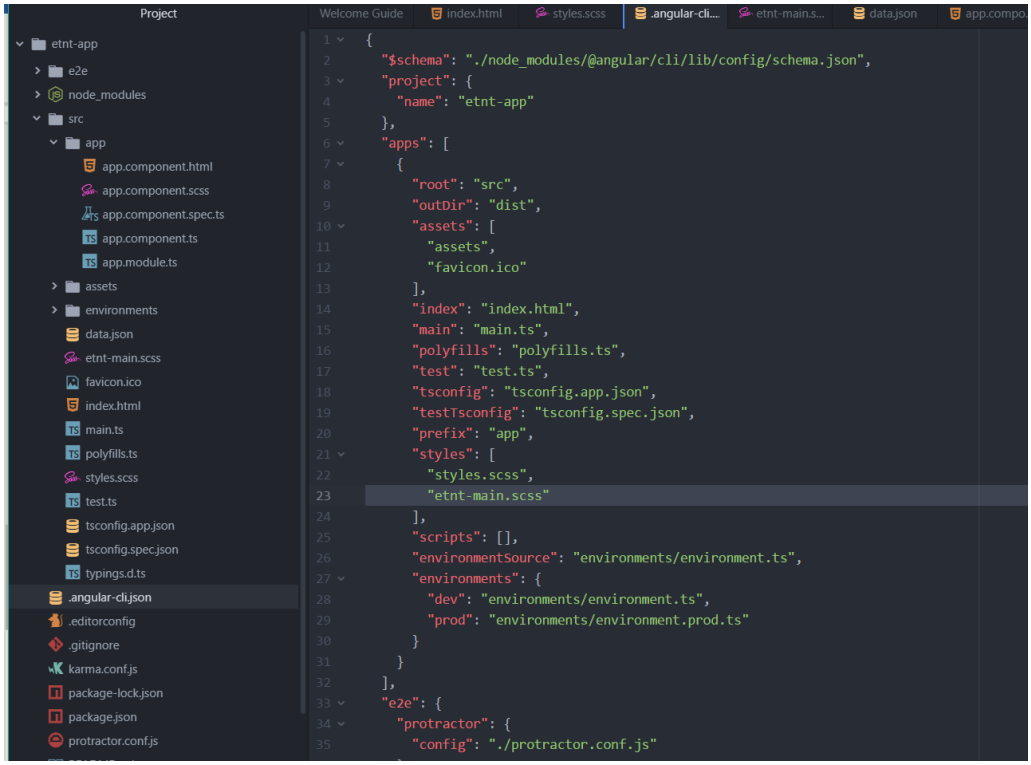
Kuva 21: Sivuston värimaailman luominen uuteen scss-tiedostoon

Kuten kuvasta 21 tulee esille, väriteemaa määritellessä luodaan uusia muuttujia (variables), joita käytetään komponenttien tyyliä määritellessä html-tiedostoissa.

Kun projektin oletuksena on css eikä scss, pitää asetuksia hiukan muuttaa Command Promptissa ja projektin sisällä. Command Promptissa määritellään oletustyyliksi scss seuraavalla koodilla:

```
ng set defaults.styleExt=scss
```

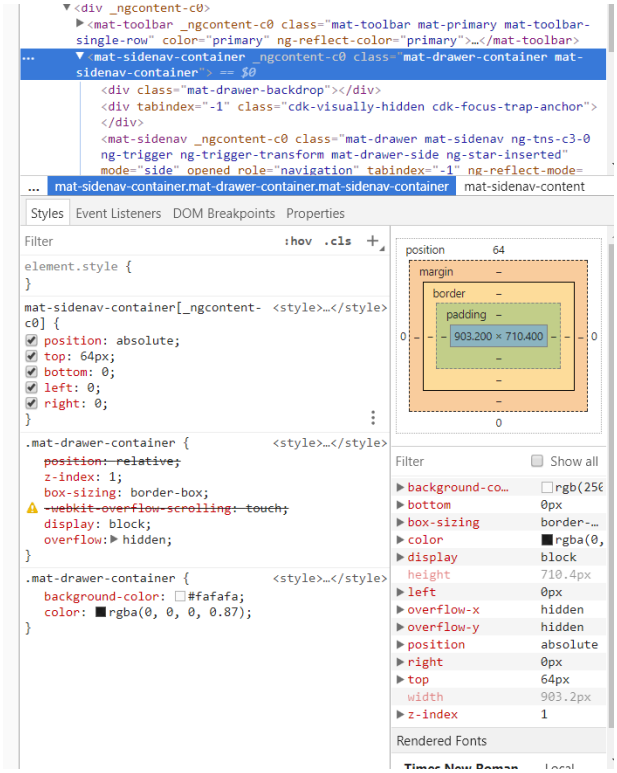
Tässäkin on tärkeää, että sijaintisi on projektikansiossa, jotta angular-cli.json tiedosto päivittyy ajan tasalle (kuva 22). Tämän jälkeen lisäsin vielä luomani uuden scss-tiedoston tiedot json-tiedostoon. Tällä tavoin tyylitiedosto ulottuu kaikkiin uusiin html-sivuihin.



Kuva 22: Uuden scss-tiedoston linkittäminen projektiin angular-cli.json tiedostossa.

On myös tärkeää muistaa muokata projektin alkuperäiset css-tiedostot scss-muotoon (esimerkiksi styles.css ja app.component.css).

Työtä tehdessäni, kohtasin muutamia ongelmia komponenttien css-sääntöjen vuoksi. Kaikissa Angular Material komponenteissa on oletusarvoiset css-määritellyt, mikä voi aiheuttaa ongelmia oman css-koodin kanssa. Esimerkiksi jos haluat asettaa tietylle komponentille css-luokan saattaa oletusarvot Angular Materialissa estää muutokset. Tämän vuoksi on tärkeää, että osaat lukea selaimen Console-ikkunassa olevat Angular Materialin oletusmäärittelyt (kuva 23). Sen lisäksi suurin osa css-koodista olisi hyvä tehdä app-kansiossa olevan oletus css-tiedoston sisälle, koska se korvaa oletusmääritellyt automaattisesti.



Kuva 23: Google Chrome selaimen Console-näkymästä näkee Angular Materialin CSS oletusmäärittelyt.

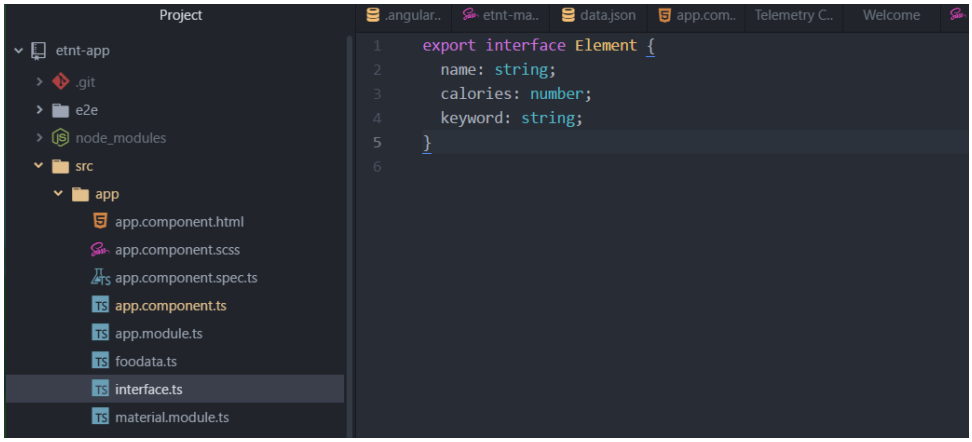
HTML ja CSS-koodausta tehdessä, on hyvä asettaa samalla tarvittavat esteettömyys ominaisuudet komponentteihin ja luokkiin. Tällä tarkoitan esimerkiksi HTML-tägeihin laitettavat role-attribuutit ja css-koodeihin tulevat pseudo-luokat esimerkiksi :focus mikä näyttää/kertoo käyttäjän sijainnin sivustolla.

Hakukoneen toiminnallinen osuus (TypeScript)

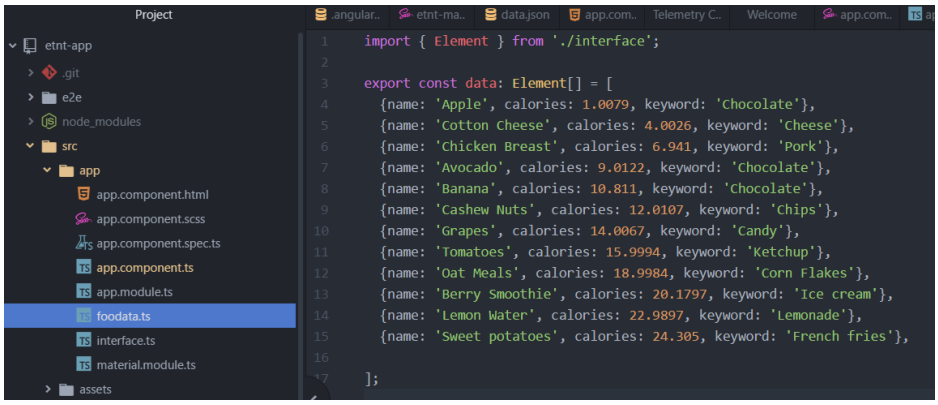
Hakukoneen tekeminen uusimmalla ES2015 JavaScriptillä oli minulle kyllä todellinen haaste. Kesti jonkin aikaa etsiä ja kysellä tietoa, miten kyseinen hakukone kannattaisi rakentaa. Loppujen lopuksi kyseessä oli melko yksinkertainenkin ratkaisu, varsinkin kun en itse käytä tässä työssä erillistä tietokantaa.

Ensimmäiseksi piti luoda kaksi uutta TypeScript-tiedostoa, mihin toisessa on taulukon ulkoasun sarakkeiden otsikot (kuva 20: interface.ts) ja toisessa sarakkeiden rivitiedot eli siis taulukon tiedot (kuva 21: foodata.ts). Foodata.ts tiedoston tarkoitus on korvata tietokannan. On kuitenkin muistettava, että yleisesti hakukoneisiin

käytetään aina jotain tietokantaa, koska yksittäinen tiedosto ei yleensä pysty kattamaan useampia hakusanoja kuormittamatta sivustoa.



Kuva 24: Interface.ts (eli sarakkeiden otsikot)



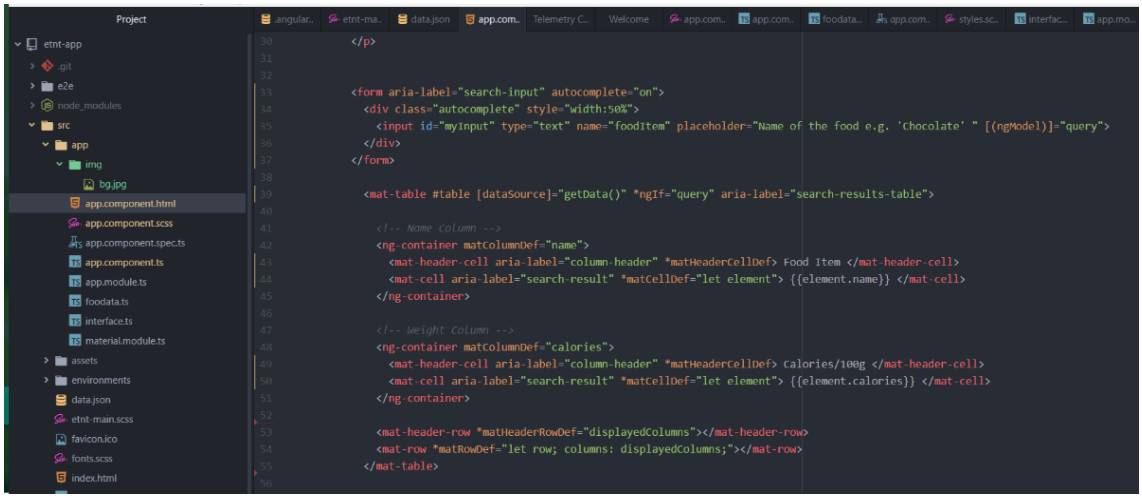
Kuva 25: foodata.ts (eli taulukon listattavat tiedot). Tietokannan korvaava tiedosto.

Sarakkeiden otsikoihin on lisättävä myös otsikon tietotyyppi esimerkiksi numero (*number*) tai merkkijono (*string*) (kuva 24). Merkkijono siis tarkoittaa sanaa tai lausetta, johon voi laittaa myös symboleja esimerkiksi huutomerkkejä tai kysymysmerkkejä (tätä ei kuitenkaan suositella).

Hakukoneessa *string*-merkkijonon käyttämistä varten tarvitsee asettaa myös sanaan liittyviä muotoiluasetuksia (kuva 26). Eli käyttäjän hakusanat toimivat vaikka hakusana ei olisi täysin merkkijonoon asetetun muodon mukainen (alkaen esim. isolla alkukirjaimella). Ilman näitä asetuksia hakukone vaatisi käyttäjän aina

kirjoittamaan kyseisen hakusanan isolla alkukirjaimella, jotta hakutulokset voitaisiin esittää. Yleisesti suositellaan esimerkiksi http://fusejs.io/ käyttämistä, koska sen avulla syötteeseen asetetun merkkijonon ei tarvitse täysin vastata string-merkkijonoon asetettua dataa. Fusejs sivuttaa siis ns. pilkun viilauksen. Itse taas päätin käyttää itse tekemässäni getData()-komennossa koodia ”toLowerCase”, mikä siis auttaa hakukonetta löytämään hakutuloksen sekä pienellä että isolla alkukirjaimella.

TypeScript-tiedostojen luomisen jälkeen lisäsin tarvittavat koodit html-, css- ja komponentti TypeScript-tiedostoihin. HTML-tiedostossa käytin mm. html-standardi tägejä esimerkiksi <form> sekä Angularin omia tägejä esimerkiksi <mat-table>. Tässä on tärkeää myös huomioida esteettömyysominaisuudet. Näissä käytin aria-label attribuuttia, joka auttaa ruudunlukijaa selkeyttämään kyseisen elementin tehtävän. Esimerkiksi <mat-table> sisälle kirjoitin aria-labelin sisälle ”search-results-table” mikä kertoo käyttäjälle hakutulosten sijainnin.

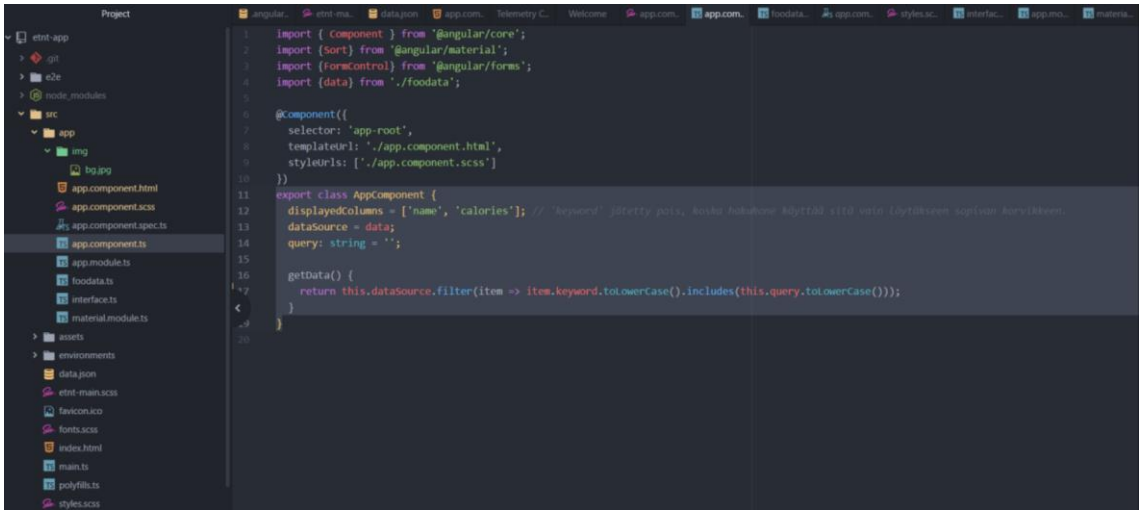


Kuva 26: Hakukoneen asettaminen html-tiedostoon.

CSS-tiedostossa oleellista oli ottaa huomioon esteettömyys hakukoneen käytössä (kuva 27). Eli pseudo-luokat :focus ja :hover olivat oleellisessa osassa input-tägin muotoiluasetuksia tehdessä.

Kuva 27: Hakukoneen muotoiluasetusten tekeminen ja esteettömyyden huomioiminen

HTML ja CSS-tiedostojen muokkauksen jälkeen lisäsin vielä app.component.ts tiedostoon hakukoneen toiminnallisuudesta vastaavan koodinpätkän (kuva 28).



Kuva 28: App.component.ts tiedostoon asetetaan hakukoneen toiminnallisuudesta vastaavan koodin mihin kuuluu yhdistäminen taulukon tietoihin foodata.ts ja html-tiedostoon.

Uusien sivujen luominen

Uusien sivujen luominen Angularissa ei olekaan niin yksinkertaista, kuin tavallisessa HTML-sivustojen koodauksessa opitaan. Ei siis pelkästään riitä, että luo uudet html-tiedostot ja linkittää ne <a> tägeillä sivustoon mukaan, vaan ne pitää myös reitittää komponentteina projektiin sisälle.

Jotta reititys tapahtuu sujuvasti pitää meidän luoda myös uusi pääkomponentti app.componentin tilalle. Uusien Angular komponenttien luominen tapahtuu Command Promptin komentorivillä seuraavanlaisesti (kuva 29):

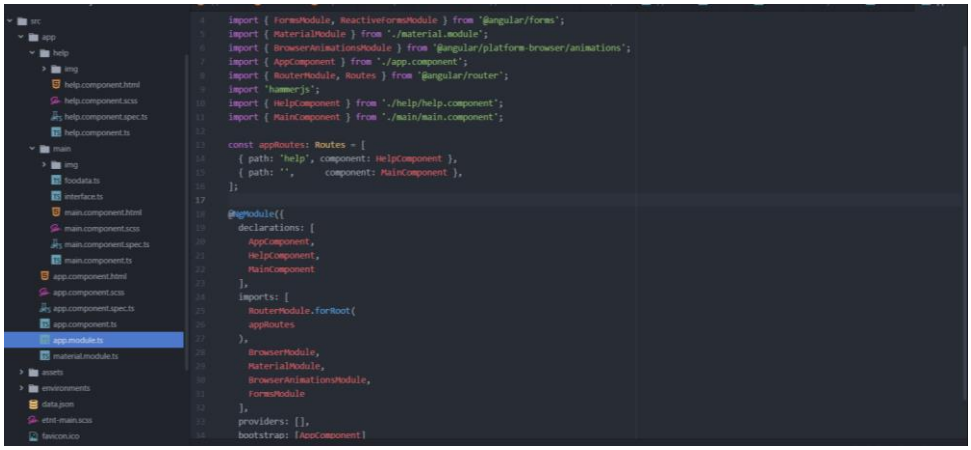
```
ng generate component komponentin_nimi --module app.module.ts
```

Kyseinen komento siis luo uudet komponentit ja siirtää niiden tiedot automaattisesti app.module.ts tiedostoon. Tämän lisäksi app.module.ts tiedostoon pitää lisätä seuraavat koodit:

```
import { RouterModule, Routes } from '@angular/router';
```

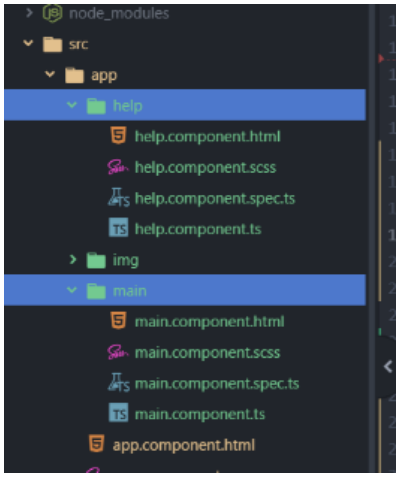
Importin lisäksi komponentit pitää reitittää seuraavalla tavalla:

```
const appRoutes: Routes = [
  { path: 'komponentin_nimi', component: nimiComponent },
  { path: '',      component: MainComponent },
];

@NgModule({
  imports: [
    RouterModule.forRoot(
      appRoutes
    )
  ],
  ...
})
```



Kuva 29: Havaintokuva app.module.ts sisältä, kun uusien komponenttien reititys on valmis.

Tässä tapauksessa loin komponentit *help* ja *main* missä *main* toimii app.component:in korvaavana komponenttina (kuva 30).



Kuva 30: Uudet Angular komponentin osat luodaan omiin kansioihinsa.

Tämän jälkeen, kun komponentit on saatu luotua, pitää meidän vielä lisätä app.component tiedostojen tiedot main.component tiedostoihin (eli siis html, css ja ts tiedostot korvaaviin tiedostoihin). Tämän jälkeen app.component.html poistetaan vanha koodi ja lisätään reititin tägi **<router-outlet></router-outlet>.** Tämän jälkeen app.component tiedostoihin ei tehdä muutoksia vaan kaikki muutokset tehdään main.component tiedostoihin.

Tämän jälkeen esimerkiksi navigointi-menun linkitykset tehdään seuraavalla tavalla (kuva 31):

```
  <ul>
    <li><a routerLink="/" routerLinkActive="active"> Search Tool </a></li>
    <li><a routerLink="/help" routerLinkActive="active"> Help </a></li>
</ul>
```

Huomioi, että main.componentiin viitatessa *routerLink=”/”* jätetään tyhjäksi.

Kuva 31: Linkitys uusien komponenttien luomisen jälkeen

3.3.4 Esteettömyyden testaus

Esteettömyyttä testatessa oleellisinta on huomioida ruudunlukija-käyttäjien ja heikosti näkevien henkilöiden tarpeet. Itsessään sivuston ulkoasun pitää olla kunnossa seuraavissa asioissa:

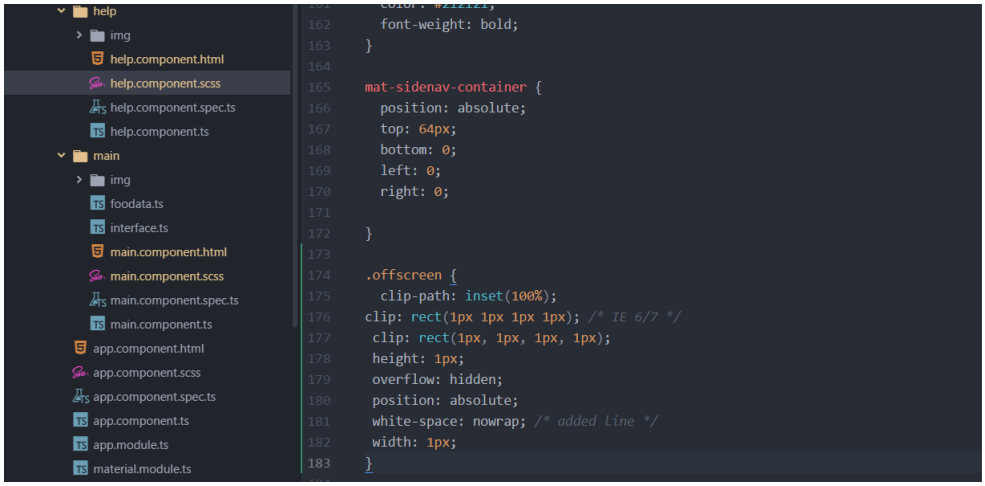
- **Värimaailma** (kontrasti-suhteet selkeät, jotta esimerkiksi värisokeat pystyvät erottamaan tekstin taustasta).
- **Sivuston selkeä rakenne** (navigointi on selkeää sekä tavallisille hiiren käyttäjille että näppäimistön käyttäjille → ei liikaa klikattavaa tai monimutkaisia linkki listauksia)
- **Fontin koko** (heikosti näkeville, varsinkin iäkkäille käyttäjille, fontin koko pitää tavallisessa leipätekstissä olla 14 pt ja otsikoissa vähintään 18 pt).

- **Tab-näppäimen käyttö** (Tab-näppäimen fokus siirtyy oikeassa järjestyksessä, länsimaalaisen lukutavan mukaan, ylhäältä alas vasemmalta ja oikealle)

Ruudunlukijan toimintaa voi testata Windowsin koneella esimerkiksi NVDA ruudunlukijalla. NVDA on ilmainen ruudunlukija, mutta Windowsille on tarjolla myös lukuisia maksullisia ruudunlukijoita. Käytin kuitenkin itse testaamisessa NVDAta. Sen avulla voidaan varmistaa, että näppäimistön käyttö (lähinnä Tab-näppäimen käyttö tässä projektissa) toimii moitteettomasti näin mahdollistaen sujuvan sivuston käytön myös sokeille käyttäjille.

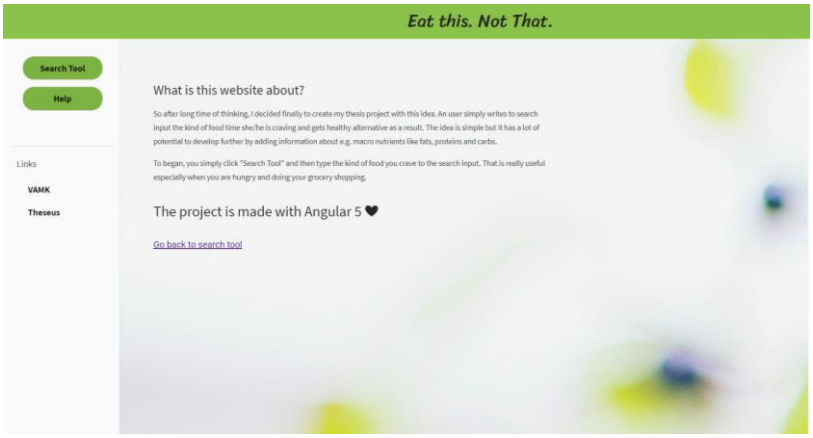
3.3.5 Offscreen-tekniikan käyttö

Lisäsin *Help*-sivulla pikalinkin takaisin hakukoneeseen ruudunlukijoita varten. Linkki on siis sellainen, jota ei voi visuaalisesti nähdä, mutta jonka ruudunlukija voi lukea. Kyseistä tekniikkaa kutsutaan Offscreen-tekniikaksi ja se toteutetaan CSS:n kautta (kuva 32).



Kuva 32: Offscreen-tekniikan tekeminen CSS-tiedostossa.

Offscreen on kätevä niissä tapauksissa, joissa halutaan helpottaa ruudunlukijoiden navigointia sivustolla, kuitenkaan vaikuttamatta sivuston visuaaliseen ilmeeseen. Ilman offscreen-luokan asettamista haluamaasi HTML-tägiin, pikalinkki tulisi näkyviin myös näkeville käyttäjille (kuva 33).



Kuva 33: Ilman Offscreen-tekniikkaa, ruudunlukijoille tarkoitetut pikalinkit tulevat näkyviin myös näkeville käyttäjille.

3.4 Lopputulos

Lopputuloksena sain luotua suunnitelmien mukaisen (ja hiukan omia odotuksia ylittävän) siistin Angular-sovelluksen. Saavutin kaikki asettamani tavoitteet sivuston suhteen. Haasteelliseksi projektin teki sen, että en ollut aikaisemmin tehnyt sivustoa uusimmalla Angular-versiolla. Tämän lisäksi minun piti myös opetella jonkin verran myös uusinta JavaScriptiä ES2015 ja TypeScriptin käyttöä. Sain onnekseni kuitenkin apua työharjoittelupaikkani kollegoilta, jotka ovat itse yrityksen johtavia Angular-asiantuntijoita.

Tulevaisuudessa harjoitustyötäni voitaisiin kehittää eteenpäin liittämällä hakukoneen datan tietokantaan, ja antamalla käyttäjille laajempaa tietoa eri ruokalajeista sekä miten niitä voidaan käyttää ruuanlaitossa (tarjoamalla esimerkiksi reseptejä).

4. POHDINNAT JA KEHITYSIDEAT

On mielenkiintoista nähdä miten Angularin tapaiset ohjelmakehykset kehittyvät tulevaisuudessa. Ohjelmakehysten käyttö on helpottanut paljon sovelluskehittäjien ja web-suunnittelijoiden työtä sekä mahdollistanut yritysten digitalisoitumisen aivan uudella tavalla. Tämän lisäksi Angular tukee nykyistä käyttäjäkokemukseen perustuvaa sivustonkehitysmallia, missä myös sivuston esteettömyys tulee vahvasti esille.

Mielestäni olisi erittäin tärkeää, että myös koulutuslaitokset ottaisivat kyseiset ohjelmakehysteknologiat paremmin esille, koska niiden käyttö lisääntyy jatkuvasti digitaalisten palveluiden suunnittelun ja kehittämisen asiantuntijayrityksissä. On myös autettava opiskelijoita löytämään luotettavia lähdesivustoja, jossa voi seurata uusien teknologioiden syntyä ja niiden käyttöönottoa. Tällä tavoin oppilaille syntyy selkeämpi pohja omaa asiantuntevuuttaan kehittäessä. Kun pohja on rakennettu hyvin, on taitoja helpompaa kehittää tulevaisuudessa eteenpäin, jos vaikka esimerkiksi haluaa ruveta kehittämään IoT-sovelluksia tai pilvipalveluita.

Esteettömyyden opettaminen tulisi olla tietojenkäsittelykoulutuksessa yhtenä tärkeimmistä opeteltavista asioista, koska sen merkitys tulee esille sekä suunnitteluettä toteutuspuolella. Jota useampi käyttäjä kykenee käyttämään web-palvelua tai sovellusta, sitä enemmän se hyödyttää sekä sivuston ylläpitäjää että eri käyttäjäryhmiä. Ihan tavallisellekin käyttäjälle esteettömät sivut tuovat paljon etuja. Sivustot ovat tällä tavoin toteutettuna usein selkeämmät ja helpommin navigoitavat.

Esteettömyyden kehittäminen tulevaisuudessa auttaa myös taloudellisen kehityksen puolella. Maailmassa on miljoonia ihmisiä, jotka olisivat valmiita tekemään töitä, mutta eivät pysty esteidensä takia. Digitaalinen esteettömyys luo mahdollisuuksia myös heille, joilla mahdollisuutta ei ole aikaisemmin ollut. Verorahoja voidaan kerätä yhä useammalta henkilöltä ja ihmisten yleinen hyvinvointi lisääntyy. Nykyään voimme tavata monenlaisia digitaalisten palveluiden asiantuntijoita, jotka saattavat olla esimerkiksi sokeita. He voivat tästä huolimatta luoda yhtä sujuvasti tai jopa paremmin hyvää työnlaatua, kuin näkevät työkaverinsa.

LÄHTEET

Aggarwal.A, Sharma.D.D, Kumar.R & Sharma.R.C. 2010. ”Mutism as the Presenting Symptom: Three Case Reports and Selective Review of Literature”. Viitattu: 13.11.2017. Indian Journal of Psychological Medicine. Vol 32. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3137816/

Alligator. 14.1.2018. ”Getting Started With Angular Material 2”. Viitattu: 23.1.2018. https://alligator.io/angular/angular-material-2/

Angular Material. ”Getting Started”. Viitattu: 23.1.2018. https://material.angular.io/guide/getting-started

Angular. “Component”. Viitattu: 1.4.2018. https://angular.io/api/core/Component

Angular. 2018. “QuickStart”. Viitattu: 23.1.2018. Google. https://angular.io/guide/quickstart

Atlassian. ”What is Git?”. Viitattu: 1.4.2018. https://www.atlassian.com/git/tutorials/what-is-git

Atlassian. 2018. “Bitbucket: Koodaa, hallitse ja tee yhteistyötä”. Viitattu: 1.4.2018. https://fi.atlassian.com/software/bitbucket

BabelJs. ”ES2015 modules to CommonJS transform”. Viitattu: 23.1.2018. Verkkojulkaisu. https://babeljs.io/docs/plugins/transform-es2015-modules-commonjs/

Bigby.G. 25.1.2018. ”How Website Accessibility Affect Persons with Disabilities”. Viitattu: 1.2.2018. https://dynomapper.com/blog/27-accessibility-testing/447-how-website-accessibility-issues-affect-persons-with-disabilities

Computer Hope. 4.10.2017. ”Dynamic website”. Viitattu: 1.4.2018. https://www.computerhope.com/jargon/d/dynasite.htm

European Commission. 26.10.2016. ”The Adoption of a Directive on the Accessibility of the Sector Bodies’ Websites and Mobile Apps”. Viitattu: 8.11.2017. Lehdistötiedote. https://ec.europa.eu/digital-single-market/en/news/adoption-directive-accessibility-sector-bodies-websites-and-mobile-apps

Haslam.K.. 12.3.2018. ”Do Macs need antivirus software?”. Viitattu: 18.1.2018. Macworld. Web-artikkeli. https://www.macworld.co.uk/how-to/mac-software/can-macs-get-viruses-3454926/

Indiana University. ”Types of Disabilities: The ability to recognize different types of disabilities helps make proper accommodations”. Viitattu: 13.11.2017. https://accessibility.iu.edu/understanding-accessibility/types-of-disabilities.html#neurological_6

Invalidiliitto. 2017. “Esteettömyys”. Viitattu: 8.11.2017. https://www.invalidiliitto.fi/tietoa/liikkumisen-tuen-palvelut/esteettomyys

Invaliidiliitto. “Esteeton.fi”. Viitattu: 8.11.2017. https://www.invalidiliitto.fi/esteetonfi

Lappalainen.J. 12.3.2010. ”Skriptaus: Perusteita skriptauksesta”. Viitattu: 1.4.2018. Jyväskylän yliopisto. Verkkojulkaisu. https://wiki.jyu.fi/display/opentvt/Skriptaus

Lumio.J. 27.10.2016 ”Niveltulehdus (artriitti)”. Viitattu: 8.11.2017. Lääkärikirja DuoDecim. Verkkojulkaisu. http://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=dlk00589

Maryka.S. 1.4.2009. ”What is the Asynchronous Web, and How is it Revolutionary?”. Viitattu: 8.11.2017. The Server Side. http://www.theserverside.com/news/1363576/What-is-the-Asynchronous-Web-and-How-is-it-Revolutionary

Mozilla MDN. 3.2.2018 ”What is JavaScript?”. Viitattu: 14.2.2018. Web-artikkeli. https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

NodeJS. 2018. “About Node.js”. Viitattu 23.1.2018. https://nodejs.org/en/about/

Näkövammaisten liitto ry. ”Näkövammaisuuden määritys”. Viitattu: 8.11.2017. http://www.nkl.fi/fi/etusivu/nakeminen/maaritys

Näkövammaistenliitto ry. 2016. ”Näkövammaisten vuosikirja 2016 Osa 1 kpl 1.3: Arviot näkövammaisten lukumäärästä Suomessa”. Viitattu: 8.11.2017. Verkkojulkaisu. http://www.nkl.fi/fi/etusivu/nakeminen/julkaisu/nvrek_vuosikirja/1_3_arviot_nv_lukumaarasta

Poikonen.M.M. 21.5.2007 ”Rikkaat Internet-sovellukset”. Viitattu: 16.11.2017. Tietojärjestelmätieteen kandidaatintutkielma. Jyväskylän yliopisto. Tietojenkäsittelytieteiden laitos. Jyväskylä. http://users.jyu.fi/~jorma/kandi/2007/Kandi_Poikonen.pdf

Pollack.G. 2016. ”Accelerating Through Angular” Viitattu: 23.11.2017. Verkkokurssi. Codeschool LLC. https://www.codeschool.com/courses/accelerating-through-angular

Rintahaka.J. 8.3.2016. ”Duchennen lihasdystrofia”. Viitattu: 13.11.2017. Rinnekoti-Säätiö. http://www.kvtietopankki.fi/oireyhtymat/d/duchennen-lihasdystrofia

Salomaa.A. 18.1.2005. ” Kuulovammaisille esteettömät verkkosivut”. Viitattu: 13.11.2017. Essityöryhmä. Web-artikkeli. http://appro.mit.jyu.fi/essikurssi/kuulovammaisuus/t2/

Sanastokeskus TSK ry. 5.12.2014. “Pienoisohjelma” Viitattu: 23.11.2017. http://www.tsk.fi/tsk/termitalkoot/hakemistot-267.html?page=get_id&id=ID193&vocabulary_code=TSKTT

Sanastokeskus TSK. 12.5.2014. “Tietotekniikan termitalkoot: Pienoisohjelma”. Viitattu: 23.11.2017. Hakemistot. http://www.tsk.fi/tsk/termitalkoot/hakemistot-267.html?page=get_id&id=ID193&vocabulary_code=TSKTT

Semantic Web. Viitattu: 1.4.2018. http://semanticweb.org/wiki/Main_Page.html

Söderholm.H. 8.11.2006. ”Näkövammaisille esteettömät verkkosivut”. Viitattu: 8.11.2017. Essityöryhmä. Web-artikkeli. http://appro.mit.jyu.fi/essikurssi/nako-vammaisuus/t2/

Söderholm.M. 3.11.2003. ”Esteettömyys”. Viitattu: 8.11.2017. Essityöryhmä. Web-artikkeli. http://appro.mit.jyu.fi/essikurssi/nakovammaisuus/t1/

The Michael.J.Fox Foundation. ”Living with Parkinson’s: Deep Brain Stimula-tion”. Viitattu: 8.11.2017. Web-artikkeli. https://www.michaeljfox.org/understan-ding-parkinsons/living-with-pd/topic.php?deep-brain-stimulation

Tieteen kuvalehti. 28.4.2010. ”Tiedon Maailma -kirjasarjasta: Ihmisen seitsemän aistia”. Viitattu 8.11.2017. Verkkojulkaisu. http://tieku.fi/ihminen/elimisto/ihmi-sen-seitseman-aistia

Walsh.D. 6.9.2007. ”6 Reasons To Use JavaScript Libraries & Frameworks”. Vii-tattu: 14.2.2018. Web-artikkeli. https://davidwalsh.name/6-reasons-to-use-ja-vascript-libraries-frameworks

WebAim. 12.10.2012. ”Motor Disabilities”. Viitattu: 8.11.2017. Web-artikkeli. https://webaim.org/articles/motor/assistive

WebAim. 28.8.2013. “Accessibility of Rich Internet Applications”. Viitattu: 16.11.2017. Web-artikkeli. https://webaim.org/techniques/aria/

WebAim. 9.8.2013 ”Cognitive”. Viitattu: 13.11.2017. Web-artikkeli. https://webaim.org/articles/cognitive/

Ylänne.K. 18.5.2017. ”Mikä ihmeen saavutettavuusdirektiivi?”. Viitattu: 8.11.2017. Celia. https://www.celia.fi/blog/mika-ihmeen-saavutettavuusdirektiivi/