Front-end ohjelmointi Magentopohjaisissa projekteissa

Samu Tapanen

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2016

Tiivistelmä

Päiväys

Tekijä(t)
Tapanen Samu Santeri

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Front-end ohjelmointi Magentopohjaisissa projekteissa

Sivu- ja liite-sivumäärä
46

Opinnäytetyön otsikko englanniksi
Front-end programming in Magento based projects

Tämä opinnäytetyö käsittelee nimensä mukaisesti front-end ohjelmointia Magentopohjaisissa projekteissa. Työssä on tarkoituksena selvittää omaa henkilökohtaista kehittymistä kymmenen viikon ajan. Seurantajakso sijoittuu ajalle 22.08-11.11.

Päivittäiset merkinnät selvittävät päivän keskeisiä tehtäviä. Päivän alussa asetetaan tavoitteet ja työpäivän lopussa tarkastella saavutettiinko ne.

Työ suoritettiin Lamia Oy:ssa, joka on suomen johtavia Magentopohjaisten verkkokauppojen tarjoajia. Yrityksen konttori sijaitsee Helsingin keskustassa ja sillä on noin 20 työntekijää.

Asiasanat
Magento, cms, css, javascript, scrum, moduuli

Sisällys

1

1 Johdanto

Opinnäytetyön aikaväli on 22.08 – 27.10, jonka aikana kävin yhtä koulukurssia, joten työpäiviä per viikko tuli 4-5.

Työskentelen it-alan yrityksessä Lamia Oy:ssa ja työtehtäväni on kehittää valmiiseen cms-järjestelmään perustuvien verkkokauppojen front-end ominaisuuksia. Yrityksessä työskentelee noin 20 henkilöä ja sen erikoisosaamisiin kuuluu muun muassa konsultointi, auditointi, verkkokaupparatkaisut, käyttöliittymäsuunnittelu, integraatiot, käyttäjähallinta, laadunvarmistus ja webkehitys.

Lamia Oy:n verkkokaupparatkaisut perustuvat valmiiseen cms järjestelmään. Cms (engl. Content Management System) on sisällönhallinta järjestelmä, jonka kautta sivuston hallitsija voi syöttää dataa sivustolle. Esimerkiksi wordpress on avoimen lähdekoodin sisällönhallintajärjestelmä, jota yrityksemme käyttää erilaisten blogien sisällön tuottamiseen. Wordpressin ideana on tarjota alusta, jossa voi luoda blogiartikkeleille julkaisuvirran ja sivun, jossa niitä voi tarkastella. Magento sen sijaan on cms-järjestelmä, joka on PHP-ohjelmointikielellä ohjelmoitu valmis verkkokauppapohja. Magento perustuu avoimeen lähdekoodiin, mikä mahdollistaa sille laajan kehittämisympäristön. Vapaan lähdekoodin ideologiaan kuuluu ohjelman lähdekoodin rajoittamattomuus, eli kuka vain voi ladata ohjelman ja sen lähdekoodin, muokata sekä kehittää sitä ja tehdä sillä voittoa. Magento itsessään on pohja valmiille verkkokaupalle. Siinä on valmiina verkkokaupalle tyypillisiä ominaisuuksia kuten asiakastilisivut, kassa, kategoriasivut ja toivelista. Magentoon sisältyy sekä ilmaisia, että valmiiksi koodattuja ulkoasuteemoja. Yrityksessä, jossa työskentelen, kuitenkin ohjelmoimme jokaisen projektin front-end ja back-end ominaisuudet alusta loppuun itse. Kun ominaisuudet ovat yrityksen sisällä olevien työntekijöiden ohjelmoituja, on niiden jatkokehitys ja toiminta mahdollisissa vikatilanteissa paljon helpompaa, sillä tiedetään, mitä tekniikoita ominaisuuksiin ollaan käytetty, joten selvitystyöhön ei kulu aikaa.

1.2 Käsitteet

HTML = Hypermerkintä kieli, jossa tunnisteiden ja attribuuttien avulla kohdennetaan tekstin eri osioita

CSS = Porrastettu tyyliarkki, jossa HTML tekstin tunnisteille pystytään antamaan ulkonäöllisiä arvoja. Esimerkiksi korkeus, leveys ja taustaväri.

Luokka-attribuutti = HTML kielessä tunnisteelle annetta attribuutti, jonka avulla se pystytään tarkasti kohdentamaan CSS kielessä.

Javascript = Ohjelmointikieli jonka avulla pystytään lisäämään dynaamisia toiminnallisuuksia verkkosivuille.

Magento = Avoimen lähdekoodin PHP:lla kirjoitettu verkkokauppapohja, jossa tulee suuri joukko valmiita asetuksia ja ominaisuuksia.

PHP = Ohjelmointikieli, jonka avulla pystytään rakentamaan HTML elementtejä, tallentamaan dataa tietokantaan ja muokkaamaan HTML rakennetta.

Moduuli = Ohjelmiston osa, joka ohjelmoidaan erillisesti ja liitetään ohjelmiston päärunkoon.

QA = Laaduntarkistusprosessi, jossa projektin kaikki pienimmätkin virheet korjataan ja projekti tehdään julkaisukelpoiseksi.

1.3 Tietoperusta

Tietoperustaan liittyviä teoksia:

- Lindholm R. 2016. Kognitiivinen kuorma ohjelmoinnin oppimisessa. – Teoksessa käydään lävitse ohjelmoinnin oppimisessa syntyvän kuorman hallintaa ja eritellään keinoja soveltaa tietoa käytäntöihin.

- Nousiainen H. Ajax-pohjaiseen web-sovelluksen toteuttaminen. – Teoksen tarkoituksena on selvittää ja toteuttaa selain-pohjainen muistiinpanosovellus, joka noudattaa nykyajan vaatimuksia ja standardeja sekä hyödyntää AJAX:ia.

- Sundström T. 2014. Tapaustutkimus IT-projektien haasteista ja ongelmista. – Tutkimuksessa selvitettiin tietyn yrityksen projektityön haasteita. Mittaus suoritettiin sähköisellä kyselylomakkeella.

2 Lähtötilanteen kuvaus

2.1 Oman nykyisen työn analyysi

Magento on alun perin suunniteltu ostoskoriratkaisuksi, jonka päälle muut ohjelmoijat rakentavat moduulinsa. Siitä sitä on kuitenkin kehitetty eteenpäin omaksi itsenäiseksi verkkokauppapohjaksi.

Työtehtävissäni on tärkeää tutustua Magenton erilaisiin ominaisuuksin. Vaikka pääpaino onkin front-end kehittämisessä, on tärkeää ymmärtää koko paketin toimintaperiaate. Työtehtävät vaativat perusymmärryksen oliopohjaisesta ohjelmoinnista, kyvyn soveltaa front-end kielien tekniikoita ja ymmärtää että yhdelle asialle on useita ratkaisuita. Magenton sivun rakennus tapahtuu XML-tiedoston kautta, jossa määritellään sivustolle ladattava javascript, CSS ja PHTML tiedostot, sekä magenton hallinnasta tulevat kappaleet ja niiden sisältö. XML-tiedosto määrittelee myös sen, mille sivulle tiedot ladataan, miltä sivulta ne poistetaan ja onko käyttäjä kirjautunut sisään vai ei.

Ohjelmointikielistä täytyy pystyä soveltamaan XML, javascript, CSS, SCSS, HTML ja PHP kieliä. Koska työskentelen front-end ratkaisujen parissa, työtehtäviini kuuluu toteuttaa sivuston ulkonäkö ja siihen kuuluvat toiminnallisuudet, kuten javascriptilla toteutetut animaatiot ja PHP muokkaukset suoraan pohja tiedostoihin. Nämä pohja (eng. template) tiedostot sisältävät yhden elementin sivulle tulostettavat tiedot. Hyvänä esimerkkinä on jokaiselle nettisivulle tuleva navigaatio, jonka kautta sivustolla pystyy navigoimaan.

Arvioisin itseni aloittelevaan toimijaa. Magento ei ole ehkä kaikkein yksinkertaisin systeemi aloittelevalle koodarille. Taustatyötä saa tehdä paljon ja jo pelkästään Magento kansiorakenteen ymmärtämiseen voi kulua yllättävän paljon aikaa.

Yrityksessämme on hyvin paljon muokattuja ominaisuuksia Magentoon, mikä tarkoittaa sitä, että tietoa voi hakea vain yrityksen sisältä. Osassa moduuleista ja kirjastoista on hyvät dokumentaatiot ja niiden avulla pystyykin työskentelemään itsenäisesti, mutta suurin osa on vielä kehitysvaiheessa ja niiden kanssa työskenteleminen vaatii apua työtovereilta. Työskenteleminen ohjelmointiyrityksessä vaatii oma-aloitteisuutta ja itsenäisyyttä. Olen tällä hetkellä ollut vasta 6kk yrityksessä töissä ja harjoittelu statukseni on vaihtunut junior kehittäjään. Vaikka harjoittelu on takana, on silti jatkossa tärkeä panostaa Magenton opiskeluun, sillä uusissa versioissa tulee muutoksia ja korjauksia, joihin projektit täytyy mukauttaa.

2.2 Sidosryhmät työpaikalla

Sisäiset sidosryhmät: työntekijät, esimiehet, omistajat
Ulkoiset: Asiakkaat, kolmas osapuoli¨

Minun työntekooni eniten vaikuttavat sisäiset sidosryhmät. Kollegojen kanssa tapahtuva projektin työstäminen ja teknisten ratkaisuiden kehittäminen ja esimiesten kuten projektipäällikön vetämät SCRUM palaveri ja muut projektin hallitsemiseen liittyvät tehtävät ja tapaamiset.
Asiakkaiden kanssa tulee projektin aikana pidettyä erilaisia projektiin liittyviä tapaamisia, joissa käydään olemassa ja kehitteillä olevia ominaisuuksia läpi. Kuitenkin tällä hetkellä asiakastapaamiset ovat enemmän suunnittelijoiden ja projektipäällikön tehtävälistassa ja heidän kauttaan tehtävät tulevat epäsuorasti minulle.

2.3 Vuorovaikutustaidot työpaikalla

Koodin kehittämisen kannalta on tärkeää kommunikoida työtovereitten kanssa. Nykymaailmassa tekniikka kehittyy nopeasti ja uusia asioita täytyy jokaisen opiskella nopeaa tahtia. Yrityksessä pidämme viikoittain front-ja back-end palaverit, joissa käydään läpi, jokaisen työntekijän viikon aikana tekemiä asioita ja listataan ongelmia ja pyritään ratkaisemaan niitä. Palavereissa käydään läpi myös kehitysvaiheessa olevia työkaluja ja suunnitellaan käyttöön tulevia tekniikoita.

Asiakastapaamisissa ohjelmoijien puolelta tehtäviin kuuluu asiakkaan opettaminen käyttämään systeemiä ja käymään läpi dokumentaatio. Myös mahdolliset jatkokehityspalaverit kuuluvat ohjelmoijille.

On kyse asiakas tai sisäinen palaveri, on vuorovaikutuksen kannalta tärkeää valmistautua palaveriin. Asiakaspalavereissa haasteeksi muodostuu asiakkaan tarpeiden esiin tuominen ja niiden sopeuttaminen mahdollisimman tehokkaasti yrityksen toiminta malleihin ja tekniikkoihin. Tämän avulla työnteon tehokkuus maksimoidaan.

3 Päiväkirjaraportointi

Seurantaviikko 34

Maanantai 22.08.2016

Luoda verkkokaupan tuotekategoria sivulle tuotelistauspohja, joka tallentaa tuotteet välimuistiin ensimmäisen latauskerran jälkeen. Tämä tehtyä sivun latausnopeus kasvaa huomattavasti.

Loin Magenton hallintapaneelista testi tuotteitta 15 kappaletta koska tarkoituksena on luoda ruudukko, johon tulee viisi tuotetta per rivi isoimmassa näyttökoossa. 15 tuotteella saan kolme riviä, mikä jatkossa helpottaa responsiivista testaamista. Tuotelistaukseen annoin standardeja porrastetun tyyliarkin arvoja, kuten fonttikoon, fonttivärin, yhden tuotteen säiliön koon ja fontin tyylin.

Tuotteiden luonnin jälkeen loin uuden kappaleen Magenton tuotenäkymälle, jota hyödynnetään helper-metodin avulla. Tämän avulla saadaan tuotteet tehokkaammin välimuistiin. Kun tuote ladataan välimuistista, se latautuu sivulle nopeammin koska sen HTML-pohjaa ei tarvitse generoida uudelleen vaan välimuistista löytyvää vastaavaa näkymää voidaan hyödyntää.

Tiistai 23.08.2016

Css ohjelmointikieltä käyttäen tyylitellä yksittäisen tuotteen tarkat tyylivaatimukset ja aloittaa kategoria sivun tuoteruudukon tyylittäminen.

Yksittäisen tuotekortin phtml-tiedoston muokkaamiseen kului aikaa paljon koska useita elementtejä täytyi liikutella, jotta tekstit ja napit saatiin oikeille paikoille. Tuotekortin tyylittämisessä pääsin alkuun, mutta tämän jälkeen edellisen projektin laatutarkistus kierros tuli yllättäen edelle.

Laatutarkistuksessa tarkistettiin kaikki yleiset marginaalit värit ja toimivat sekä toimimattomat ratkaisut. Niitä oli tärkeä hioa tänään koska torstaina kaupan täytyy olla testikunnossa.

Kauppa myy auton venttiileitä ja painesensoreita joita on useita erikokoja ja malleja. Tänään korjasimme magenton ryhmätuotteeseen asetettavia attribuutteja. Tässä tapauksessa yhdelle venttiilille pitää pystyä asettamaan useita arvoja kuten halkaisija pakkauskoko hinta ja niin edelleen. PHP helper metodin avulla kutsutaan hallintapaneelista tulevat arvot phtml tiedostossa, joka rakentaa html rakenteen sivulle.

Työpäivä oli täysin erilainen kuin mitä suunnittelin, eikä aamulla asetetut tavoitteet täyttyneet. Koin että ymmärsin paremmin magentoa, koska pääsin tekemään back-end ominaisuuksia. Mielestäni nämä kehittävät front-end ohjelmointia koska ymmärtää ohjelman toimintaperiaatetta paremmin.

Keskiviikko 24.08.2016

Tavoitteet:
Viimeistellä edellisen kaupan/projektin QA ja jatkaa kategoriasivun tuotelistausnäkymän tyylittelyä. Päivän aikana on tarkoitus pitää palaveri projektissa työskentelevän työkaverin kanssa aikataulupalaveri uudesta viikko sitten alkaneesta projektista. Loppuaika käyttää uudenprojektin kategoriasivun tuotelistauksen tyylittämiseen

QA:n viimeistelyssä meni odotettua kauemmin koska taulukon ulkoasua hiottiin hieman. Tämä siirsi toiseen projektiin siirtymistä parilla tunnilla. Kun ulkoasu oli hiottu, pidimme ensimmäisen sprintin suunnittelun palaverin työkaverin kanssa, mitä huomenna jatkamme projektin hallinnoijan kanssa. Loppupäivä kului uuden projekti kategoriasivun tuotelistausta tehdessä.

Lisäsin viime projektin lopulla käyttöön otetun ECMA 6 javascript kielen uuteen projektiimme. Hieman muuttuneen syntaksiin opettelu vie aina hetken aikaa mutta tuo haastetta työntekoon.
Tavoitteisiin päästiin tänään hyvin ja loppuviikko näyttäisi siltä, että pysytään sprintin mukaisissa aikatauluissa.

Torstai 25.08.2016

Tavoitteet:
Viimeistellä kategoriasivun tuotelistausnäkymä ja asettaa tuotteelle ominaisuuksia magentossa, jotta käyttäjä voi suodattaa tuotteita. Tälle päivälle on suunniteltu yksi tapaaminen, jossa käydään läpi sprintin suunnitteluun liittyviä asioita kaikkien projektiin osallistujien kanssa.

Tuotelistaukseen käytin css flex-box ominaisuutta, jonka avulla listauksesta sai varsin skaalautuvan. Edellisessä projektissa tuli varsin paljon kyseistä ominaisuutta käytettyä, jonka takia työ sujui varsin jouhevasti.
Sprintin suunnittelu tapaamisessa ilmeni projektin laskutukseen liittyvä ongelma, joka menee johtoportaalle ratkaistavaksi eikä sinällään vaikuta ohjelmoijien työhön.

Tuotteiden ominaisuuksien asettaminen oli minulle uutta ja tuotti hankaluuksia. Itse hallintapaneelipuoli on selkeä ja helppo ymmärtää, mutta tehtäviin kuuluu luoda asennus scripti tiedostot, jotta jokainen kehittäjä joka lataa projektin saa suoraan tuotteiden ominaisuudet, ilman että hänen tarvitsee asentaa niitä erikseen hallintapaneelista.

Työpäivän tavoitteet saavutettiin lukuun ottamatta kategorioiden suodattimia. Tähän täytyy käyttää enemmän aikaa, jotta pystyy opetella asennus scriptien tekemisen alusta loppuun. Tänään silti kehittyi pohja uuden asian oppimiselle.

Perjantai 26.08.2016

Tavoitteet:
Luoda data-scirpt tiedostot jotka syöttävät tuotteille ominaisuustietoja ja aloittaa kategoriasuodattimien tyylittely.

Data-script tiedostojen tekemisessä kului paljon aikaa, koska halusin ymmärtää data syötön pohjimmaisen idean. Opetteluvaiheen jälkeen tein itse data scriptin, joka syötti kaikkie tarvittavat tiedot tietokantaan. Loppuaika päivästä kului javascriptin tekemisessä suodattimille.

Tein javascript elementin, joka laskee elementin korkeuden, jonka avulla se pystytään animoimaan nollasta pikselistä sen tarkkaan korkeuteen.

Tänään opin taas uutta magenton back-end ominaisuuksista ja kykenen itsenäisesti tekemään data-ja asennus script tiedostoja.

Viikkoanalyysi

Viikko oli työtehtäviltään monimuotoinen ja sopivan haastava. Uusia asioita tuli opeteltua paljon. Vaikka olen ollut yrityksessä töissä nyt puoli vuotta ja tehnyt ensimmäisen verkkokauppani, ei kyseessä ollut sooloprojekti, mikä tarkoittaa sitä, että kaikkia työtehtäviä en

itse päässyt tekemään. Nyt onkin ollut tärkeää, että teen itse niitä tehtäviä mitä edellisessä verkkokauppaprojektissa parini teki.

Tästä hyvänä esimerkkinä tuotteiden välimuistitallennus ja data-scriptien luonti. Molemmat edeltä mainituista asioista ovat magenton peruskauraa ja niiden opetteleminen onkin välttämätöntä, koska aina kun Magentoon asennettu moduuli sisältää vuorovaikutusta tietokannan kanssa, löytyy upgrade tai asennus scripti moduulin koodihakemistosta, jota ajetaan, kun tietyssä URL osoitteessa. (Dodia 2014). Suurimmat ongelmat joita joudin selvittämään viikon aikana, johtuivat juuri edellä mainituista asioista.

Aluksi syötin tietoa tietokantaan magenton data scriptien kautta asennus scripien sijaan. Varsinaista ongelmaa tässä ei ole, sillä data scriptit lisäävät tietokantaan sisältöä esimerkiksi lisäävät tietokantaan rivejä, kun taas asennus scriptit muokkaavat tämän sisältöä esimerkiksi lisäävät sarakkeen. Tuotteiden ominaisuudet ovat niin sanotulla harmaalla alueella koska ne ovat sekä rivejä tietokannassa, mutta toisaalta ne voivat muuttaa myös rakennetta, sillä jokainen tuotteen ominaisuus on pystyrivi tietuetiedostossa.

Näistä ongelmista esimerkkinä toimii metodit, joita käytin tuotteen välimuistiin viemisessä. Tässä tapauksessa ajoin usean metodin uudestaan tiedostossa, joka tallentaa välimuistiin tuotteen. Kyseiset metodit löytyivät jo projektista ja niitä olisi voinut uusiokäyttää eri tiedostoissa. Käyttämällä helper metodia koodia syntyy huomattavasti vähemmän eikä koodin toistoa synny. Tämä tarkoittaa sitä, ettei turhaa koodia tulee paljon koska kyseiset koodipätkät löytyvät jo projektista. Hyvä ohjelmoija ei toista itseään ja ylimääräistä koodia syntyi todella paljon koska en osannut hyödyntää helper metodia oikeassa paikassa.

Magenton back-end puolta pyöritellessä oppii hiljalleen ymmärtämään paremmin ohjelman toimintaperiaatetta. Tärkeää on osata muokata tietoa, jota kaupalle annetaan ja pystyä manipuloimaan front-end rakennetta mahdollisimman kevyeksi ja selkeäksi. Oma osaamiseni karttui juuri edeltä mainitussa asiassa, sillä pääsin aloittamaan uuden projektin puhtaalta pöydältä. Tällaisessa tilanteessa pyrkii välttämään edellisessä projektissa tulleet sudenkuopat ja siirtyä kestäviin toimintamalleihin.

Seurantaviikko 35

Maanantai 29.08.2016

Tavoitteet:
Aamulla on etäpalaveri asiakkaan kanssa, missä on tarkoituksena käydä läpi kaupan toimin nallisuuksia. Kyseessä on tapaaminen asiakkaan kanssa projektista, joka on tuotteintegraatiota vaille valmis. Tämän jälkeen olisi tarkoitus jatkaa kategoriasivun suodattimien työstämistä.

Palaverissa tuli opittua hyvin asiakkaan opastusta, mutta samalla huomasimme toisen ohjelmoijan kanssa, että viimeisin versionhallinan kommitointi oli hajottanut ryhmätuotenäkymän. Tämän korjaamiseen kului usea tunti.

Loppupäivästä pääsin tekemään kategoriasuodattimia. Projektin toinen koodari oli koodannut javascript ohjelmapätkän, joka laskee sivun navigaatio elementin korkeuden sivun ylälaidasta ja kun sivun ylälaita osuu navigaatio elementtiin, siihen lisätään position:fixed; arvo css kielellä. Eli tehdään niin sanottu sticky eli tarttuva elementti. Minun oli tarkoitus jalostaa tätä javascript pätkää pitemmälle. Tein tiedostosta sellaisen, että sivulle lisättävälle elementille pystyy määrittämään, onko elementti tarttuva vai ei.

Päivän aikana tavoitteisiin ei päästy, koska palaverissa huomattu ongelma vei suuren osan päivästä.

keskiviikko 31.08.2016

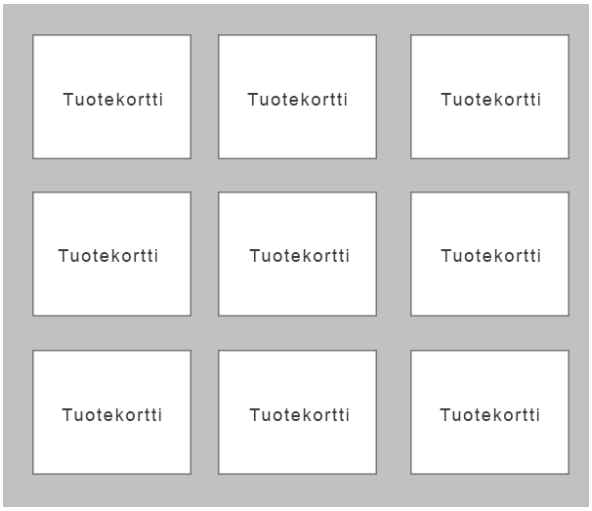
Tavoitteet:
Kategoria sivun nopean näkymän javascriptit, Tyylitellä suodattimet, arvioida versionhallintaan ladattu kommitointi

Aamu ja aamupäivä menivät versionhallinnan parissa. Käytössämme on pull-request review työskentelytapa. Tämä tarkoittaa sitä, kun ohjelman tietty osa on valmis, se ladataan erillisessä oksassaan (branch) pull-request tilaan odottamaan arviointia, jonka jälkeen arvioija jättää kommentit koodista ja päättää ladataanko se kehittämispalvelimelle, johon asiakkaalle saattaa olla pääsy.

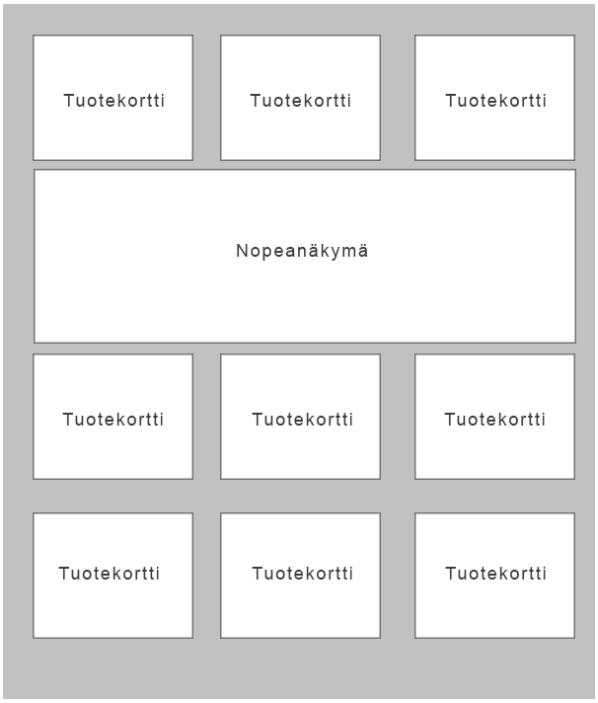
Tämän jälkeen aloin kartoittaa kategoriasivun niin sanotun nopean näkymän javascriptiä. Nopea näkymä tarkoittaa tässä tapauksessa tuotteen näkymää jossa tuotetietoja pystyy tarkastella ilman siirtymistä uudelle sivulle.

Nopeaa näkymän suunnitteleminen oli haastavaa, sillä tuotelistaus muodostui HTML list elementeistä, jotka oltiin identifioitu samalla luokka-attribuutilla. Tuotteet rivittyivät taitto-kohtien mukaan kolmessa eri kohdassa. Mobiilissa kaksi, tabletissa kolme ja työpöytä-koossa 5 tuotetta per rivi. Nopea näkymä tuli näkymään aina valitun tuotteen alapuolelle päärivien väliin.

Ratkaisuksi muodostui uuden HTML-elementin luominen jokaisen tuotteen sisälle. Tälle elementille annetaan identifioiva luokka-attribuutti, jolle määritellään CSS-kielellä posi-tion:absolute; ja visibility:hidden; . Tällöin elementti on piilotettu näkyvistä.
Tämän jälkeen tarkistetaan javascriptilla mitä nopean näkymän nappia ollaan klikattu. Kli-katun napin kanssa samassa säiliössä olevaa nopean näkymän luokka-attribuuttia kohdis-tetaan ja sille annetaan korkeus, leveys ja taustaväri.



Kuva 1 tuotenäkymä



Kuva 2 Nopeanäkymä

Ratkaisu vaatii, että tuotteen säiliölle annetaan ala marginaali arvoksi 500px (varattu tila nopealle näkymälle) ja että koko listauksen isäelementillä on position arvoksi säädetty relative koska nopeanäkymän positio on absoluuttinen.

Ratkaisun suunnittelemiseen kului aikaa, mutta teoriassa mahdollinen ratkaisu on nyt paperilla ja sitä aloitan ohjelmoimaan huomenna torstaina 1.9.2016. Päivällä tavoitteita jäi paljon saavuttamatta mutta huomisen työtehtävät ovat hyvin suunnitellut, mikä tarkoittaa, ettei niiden suorittamiseen pitäisi kulua paljon aikaa.

Torstai 1.09.2016

Tavoitteet:
Nopean näkymän javascriptin ohjelmoinnin ja tyylittelyn aloittaminen

Javascriptin toteuttaminen nopeassa näkymässä suunnittelusta huolimatta oli haastavaa. Jokaiselle tuotekortille täytyi antaa suurimman sisällön omaavan kortin korkeus. Tämän ylimääräisen javascript ohjelmapätkän olin tehnyt jo edelliseen projektiin mutta ongelmaksi muodostui ohjelmapätkän käyttöalue nykyisessä projektissa.

Javascript koodi, jonka kirjoitin, tekee muuttujan, joka on tyhjä jono. Tämän jälkeen se etsii tuotelistauksesta kaikki lista elementit, joilla on luokka-attribuutti list-product.

Ensiksi otetaan kaikkien listassa olevien elementtien korkeudet ylös offsetHeight ominaisuudella, minkä jälkee ne työnnetään tyhjään jonoon. MathMax:in avulla katsotaan mikä jonoon tulleista luvuista on isoin ja tämä asetetaan kaikille listaelementeille.

Ongelmaksi muodostuikin tuotteiden kuvantamisaika. Kun porrastettu tyyliarkki kuvantaa elementtien korkeuden ennen kuin tuotelista on latautunut kokonaan, ei tuotekortti saa oikeaa korkeutta, vaan osan siitä.

Ongelma ratkesi, kun funktiolle annettiin 200 millisekunnin viive, jotta se odottaa, kunnes kaikki tuotekortit ovat kuvantuneet.

Päivän tavoitteet täyttyivät osittain. Eilisessä suunnittelussa jäi paljon asioita ilman huomiota esimerkiksi juuri tuotteiden kuvantamisaika. Huolellisemmalla suunnitteluprosessilla tämän päivän sudenkuopista olisi selvinnyt. Nyt aikaa kului ongelmanratkaisuun liian kauan.

Perjantai 02.09.2016

Tavoitteet:
Nopean näkymän javascriptin optimointi ja responsiivisuuden varmistaminen.

Ensimmäinen työtehtäväni oli viimeistellä edellisen projektin viimeiset QA-tehtävät. Tehtävät olivat marginaalin vaihdoista värimuutoksiin, joten niiden tekemiseen ei kulunut paljoa aikaa.

Nopean näkymän toiminta responsiivisessa ikkunassa muodostui ongelmaksi. Kirjoittamani javascripti lisäsi korkeutta tietyille hypermerkintäkieli elementille. Tila joka tuotekorttien keskeltä varattiin, tehdään marginaalin lisäyksellä klikatun tuotekortin pohjalle, tarkoittaa tämä sitä, että koko tuotekortti marginaalin lisäksi korkeuteensa.

Kun laskettiin funktiota, joka ajetaan näytön kokoa muutettaessa, hajoaa koko näkymä koska jokainen tuotekortti saa saman korkeuden, vaikka vain yhdellä on olemassa pohjamarginaalia.

Ratkaisu tähän löytyi, kun haetaan javascriptin getComputedStyle ja getProperty value funktioiden avulla tuotteelle asetettu pohja marginaali ja vähennetään se kokonaiskorkeudesta silloin kun ruudun kokoa muutetaan.

Työpäivän tavoitteet olivat realistiset ja vielä ensi viikko olisi aikaa toteuttaa kategoriasivua ja sen toiminnallisuuksia. Projektin aikataulu on asiakkaasta johtuen kelluva, eikä projektin kanssa tarvitse kiirehtiä.

Viikkoanalyysi

Työviikko koostui pääasiassa javascript-kielen ohjelmoinnista. Kategorian tuotelistaus sivulle tuli yhteensä neljä erillistä javascriptillä toimivaa elementtiä. Ensimmäinen oli tarttuva elementti, joka laski etäisyyttä sivun yläreunaan ja tämän ollessa nolla, javascript muutti elementille CSS:n position arvoksi fixed. Seuraava oli nopea näkymä. Siinä javascript kuuntelee nappielementtiä ja kun sitä klikataan hiirellä, se avaa uuden HTML-säiliön. Mikä teki nopeasta näkymästä haastavan, oli se, että säiliön avaaminen täytyi suorittaa ruudukon sisällä. Muita javascript elementtejä oli päänavigaation tarttuva ominaisuus, jonka täytyy toimia yhdessä muitten elementtien kanssa ja suodattimien haitarinavigaatio.

Suoritin nopean näkymän javascriptin, mutta en onnistunut animoimaan kyseistä ominaisuutta. Lähestyin ongelmaa siten, että tein luokka attribuutin, joka annettiin elementille siinä vaiheessa, kun nopean näkymän nappia oltiin klikattu. Tätä tapahtumaa ennen, nopealla näkymällä oli näkyvyys asetettu nollaksi opacity ja visibility arvon avulla. Luokka-attribuutilla sen sijaan yli kirjoitettiin kyseiset arvot. Näin päässään tilanteeseen, jossa nappia painamalla tulee näkyviin javascriptillä tarkennettu kohde, mutta siinä ei ole mitään animaatiota.

Animaatiolla tarkoitetaan elementin tyylin muuttamista toiseen viiveellä. Ihmisen silmä hakeutuu liikkeeseen, joten animaatiolla pystytään kohdistamaan käyttäjän huomio tiettyyn kohtaan. (Makkonen 2015, 5). Hyvä animaatio on myös sisältöä rikastava elementti, eikä se saa häiritä käytettävyyttä tai keskittymistä. (Makkonen 2015, 5). Aluksi suunnitelmissani oli elementin pohjamarginaalin animointi, mutta tämä osoittautui äärimmäisen raskaaksi, eikä animaatiosta tullut sulavaa. Tarkemmin asiaa tutkittuani sain selville, ettei ole parhaimman tavan mukaista käyttää animoinnissa marginaaleja animoinnin tyylin muuttamisen kohteena. Jäljelle jäisi elementin korkeuden animointi, mutta tässä tapauksessa se ei ole mahdollista koska tuotelistan tuotekorteilla on olemassa javascript ominaisuus, joka antaa kaikille ruudukon elementeille eniten sisältöä omaavan kortin korkeuden.

Muu varteenotettava vaihtoehto on avainkehysten käyttäminen. Tässä animaatio nimetään omaksi funktioksi ja sille annetaan jokaista avainkehystä kohden tyyli-ja animaation

nopeusarvot. Avainkehysten tekeminen on yleisesti ajateltuna monimutkaisten, automaattisesti käynnistyvien ja silmukassa olevien animaatioihin käytetty tekniikka. Kun taas CSS-transition on yksinkertaisen käynnistettävien tapahtumien animoimista varten käytetty tekniikka.

Seurantaviikko 36

Maanantai 6.9.2016

Tavoitteet:
Kategoriasivun suodattimien haitarinavigaation animointi javascriptin avulla. Yleinen kategoriasivun javascriptin hiominen. Aikataulusuunnittelua projektikaverin kanssa.

Päivän työtehtävät olivat yksinkertaisia, mutta samalla myös aikaa vieviä. Kategorian tuotesivulla on paljon javascriptelementtejä, ja niiden toimivuuteen on panostettava, jotta sivusto toimisi ja latautuisi mahdollisimman kevyesti.

Aikataulusuunnittelua tarvitsi käydä työkaverin kanssa lävitse sen vuoksi, koska tämän läsnäolo työpaikalla jää kahden viikon aikana kuuteen päivään. Palaverissa suunnittelimme, mitä työtehtäviä siirtyy minulle häneltä, jotta pystymme pitämään projektiaikataulusta kiinni.

Tiistai 06.09.2016

Tavoitteet:
Kategoriasivun suodattimien haitarinavigaation javascriptin viimeistely

Haitarinavigaation tekniikan toteutin funktiolla, joka hakee jokaisen haitarinavigaation elementin, jota klikattaessa HTML-säiliö aukeaa. Tämän jälkeen liikutaan DOM puussa ylöspäin ja haetaan jokaisen yksittäisen elementin vanhempi. Nyt voimme käyttää querySelector funktiota ja haetaan kohde, joka halutaan avata ja tälle lisätään porrastetun tyyliarkin luokka-attribuutti, jolle voidaan antaa CSS-arvoja, kun luokka on näkyvissä.

Päivän tavoitteeseen päästiin, mutta kokonaisen javascript toiminniallisuuden kirjoittaminen vaatii aina varsin paljon hiomista, jotta ratkaisu olisi mahdollisimman joustava, eikä koodi toista itseään.

Keskiviikko 07.09.2017

Tavoitteet:

Viimeistellä ja hioa koko tuotekategorialistauksen javascriptit.

Kategoriasivun mobiili näkymään tulevat nopeannäkymän napit tulevat näkyviin klikkaamalla, kun taas tabletti ja työpöytäkoossa ne tulevat hover-efektillä. Tähän aloitin luomaan javascriptiä joka hyödyntää event.preventDefault funkiota, jonka tarkoituksena on estää alkuperäisen linkin laukaisu klikatessa.

En päässyt asiassa kuin alkuun sillä edellisestä projektista tuli paljon hiomista vaativia tehtäviä, jotka menivät nykyisen projektin edelle,

Torstai 0.8.09.2107

Tavoitteet:

Alustaa nopean näkymän tuotelatauksen AJAX funktio, viimeisteille edellisen kaupan QA tehtäviä.

Aamu ja aamupäivä kuluivat edellisen kaupan QA tehtäviä suorittaessa. Olin jaotellut päivän niin, että ensimmäiset kolme tuntia tein kyseisiä QA tehtäviä ja loput 4.5 tuntia olin varannut AJAX-pyynnön tekemiseen.

AJAX-pyynnön ideana oli kategoriasivun tuotekortin pikanäkymän säiliön sisälle ladattavan tuotteen tuominen. AJAX-pyyntö on tähän optimaalinen ratkaisu, koska jokaisen pikanäkymän tuotteen piilottaminen ja esiin tuominen sivulle lisäisi lataus aikaa merkittävästi, koska jokainen tuote pitäisi kuvantaa kaksi kertaa. Nyt kun pikanäkymä avataan, ladataan AJAX-pyynnön avulla phtml tuote pohjasta sisältö.

Päivä opetti AJAX-pyynnön alustamisen. Loppuun asti en asiassa päässyt, koska tarvitsin yrityksen johtavan kehittäjän apua arkkitehtuurin kanssa.

Perjantai 0.9.09.2107

Tavoitteet:

Korjata tarttuvan elementin muodostava javascript tiedosto. Siistiä kategoriasivun javascript tiedostoa.

Javascript-tiedosto, joka muodostaa tarttuvan elementin täytyi korjata, koska nykyisessä versiossa oli virhe siinä kohdassa, jossa kolmas tarttuva elementti otetaan huomioon. Tämä korjattiin tarkentamalla valitsimia jotka valitsevat luokka-attribuutteja DOM-puusta.

Kategoriasivun javascript-tiedosto oli muodossa, jossa jokaiselle tuotteen nopean näkymän nappulalle annettiin luokka-attribuuttti, jota kuunneltiin. Ja kun painallus tapahtui, kirjoitettiin heti jokaiselle luokalle oma funktio, jossa lisättiin luokka is-open, jolle oli määritelty tyylit nopean näkymän avaamiseen.

Nyt kaikki html elementit, joille lisätään uusi luokka-attribuutti, silmuikoidaan läpi ja jokaiselle lisätään silmukoinnin päätteeksi is-open luokka-attribuutti. Näin koodia tulee huomattavasti vähemmän ja vielä kun käytetään Ecma6 javascriptin nuolifunktiota, tulee koodista vielä lyhyempää.
Sulkeminen tapahtuu samalla funktiolla mutta lisäämisen sijaan luokat poistetaan.

Tänään opin käyttämään ES6 uusia ominaisuuksia, joista en ennen tiennyt.

Viikkoanalyysi

Projekteissa usein koodi elää läpi projektin elin kaaren, ja harvemmin se pysyy alkuperäisessä muodossaan julkaisuun asti. Kun kategoriasivun javascript elementit olivat alustettu, täytyi ne hioa siihen pisteeseen, missä niiden eteenpäin kehittäminen on riippuvainen sivun muista elementeistä.

Tein virhearvion nopean näkymän animoinnissa. AJAX-pyyntö, joka hakee sisällön nopean näkymän säilöön, ei tule silmänräpäyksessä vaan useamman millisekunnin viivellä. Tämä on pakko ottaa huomioon animoinnissa, joten animoinnin joutui ajastamaan uudelleen. Täysin turhaa eivät edellisen viikon animoinnit olleet, mutta mikäli haluttaisin saavuttaa optimaalinen työnkulku, olisi järkevintä suorittaa animointi viimeisimpänä.

Viikon pääteemana oli kuitenkin Ajax-pyynnön käyttäminen projektissa ja suurin osa ajasta kuluikin sen opettelemiseen. Nimitys AJAX tulee sanoista Asynchronous Javascript and XML. Vapaasti suomennettuna: epäsynkroninen Javascript ja XML. Kyseessä ei ole itsenäinen teknologia vaan termi, joka viittaa useiden teknologioiden yhteistyöhön.

Ajaxin avulla web sovelluksista saadan vuorovaikutteisempia, sillä ajax hyödyntää useita ohjelmointityökaluja kuten javascript, HTML, DHTML, XML, CSS, DOM jne. Ajax toimii välittäjänä käyttäjän selaimen ja palvelimen. Sen käytössä oleva moottori pystyy suorittamaan useita pyyntöjä ja tuomaan tietoa sivulle palvelimilta. Itse AJAX-termillä tarkoitetaan toimintatapaa, jossa selainohjelma ja palvelin vaihtavat tietoa taustalla siten, ettei käyttäjän tehdessä muutoksia tarvitse ladata sivua uudelleen. (Nousiainen 2014, 8-9).
Kun käyttäjä pyytää tietoja, javascript lähettää pyynnön AJAX moottorille, joka XML:n avulla pyytää tietoja palvelimelta samalla kun se päivittää sivua. Google maps on tunnettu sen käyttöliittymästä, jossa käyttäjä pystyy etsimään kartasta tietoa, ilman että sivua tarvitsee ladata jatkuvasti uudelleen. Kuten projektissa, myös mapsissa sivun tietojen päivittämisen hoitaa Ajax.

Seurantaviikko 37

Maanantai 10.09.2107

Tavoitteet:
Aloittaa tuotesivun tyylittäminen.

Tuotesivun tyylittämisen aloittaminen alkoi perinteisellä porrastetun tyyliarkin tiedoston lataamisen määrittelemisellä pää XML-tiedostoon. XML- tiedostoon lisättiin uusi viite, jolle määriteltiin mihin kohtaan HTML sivua tiedosto ladataan, tiedostomuoto, joka oli tässä tapauksessa CSS ja tiedoston nimi.

Kun XML-tiedostoon oli syötetty rivit, ajettiin gulp-ohjelma, joka tiivistää kaikki tiedostot yhdeksi tiedostoksi. Tämän jälkeen ajettiin yrityksessä ohjelmoituja python ohjelmointikielellä koodattuja ohjelmia jotka ensiksi indeksoivat kaikkien tiedostojen uudet polut ja otti ne käyttöön projektiin.

Päivä oli vanhan kertausta, mutta tiedostojen lisäystä pää XML-tiedostoon tulee yleensä vain projektin alussa, joten täysin automaattista tämä ei itselleni vielä ole.

Tiistai 11.09.2017

Tavoitteet:
Korjata AJAX-pyyntö. Jatkaa tuotesivun tyylittämistä. Aloittaa nopean näkymän tyylittäminen.

Päivä oli jaoteltu niin, että aamulla neljätuntia oli varattu AJAX-pyynnön korjaamiseen ja loppupäivä tuotesivun ja nopeannäkymän tyylittelyyn.

Nopea näkymä ja tuotesivun tyylit tässä tapauksessa kannatti tehdä samaan tiedostoon ja ladata kyseinen porrastetun tyyliarkin tiedosto, sekä kategoriasivulla, jossa nopea näkymä on ja myös tuote sivulla. Tämä tapahtuu pää XML-tiedoston kautta.

Työpäivän tavoitteisiin päästiin. Ajax pyyntö lataa sen ohjaimeen määritellyn phtml-kappale elementin. Näin ohjaimeen pystytään määrittelemään html-elementtejä, joita halutaan nopeaan näkymään.

Keskiviikko 14.09.2016

Tavoitteet:
Jatkaa nopean näkymän tuotekortin tyylittämistä ja tehdä javascriptiin korjaus joka mahdollistaa nopean näkymän mobiilikäytön.

Nopean näkymän mobiilinäkymä oli suunniteltu niin että, nopea näkymä aukeaa ponnahdusikkunamaisesti ja peittää koko sivun paitsi headerin. Ongelmaksi muodostuu ikkunan sijoittaminen.
Ikkuna piilotetaan aluksi antamalla sille left arvoksi 100% Kun elementtiä, joka laukaisee nopean näkymän, on painettu hiirellä, annetaan ikkunalle left arvoksi 0. Näin ollen ikkuna piiloutuu ja tulee näkyviin, kun nappia painetaan. Jotta ikkuna saadaan aina aukeamaan oikeaan kohtaan näyttöä, on elementin position ominaisuudeksi annettava fixed. Tämä tarkoittaa sitä, että ikkuna on aina samassa kohdassa riippumatta siitä rullattaanko sivua alaspäin. Mobiililaitteiden näytöt ovat kuiteinkin korkeudeltaan matalia verrattaen selaimeen jota käytetään tietokoneelta, joten tämä on otettava huomioon. InnerScreenHeight ominaisuus hakee selaimen näytön korkeuden ja tämän funktion avulla annetaan ikkunalle oikea korkeus oikeassa näytössä.

On erittäin tärkeää, että elementille saadaan staattinen korkeus. Kun nopea näkymä saa is-open luokka-attribuutin, ajetaan javascriptissä funktio joka poistaa body merkiltä rullauksen pois käytöstä ja lisää sen ponnahdus ikkunalle. Tällöin käyttäjä rullaa ponnahdusikkunaa eikä sivua.

Päivän tavoitteista jäätiin jälkeen, koska nopean näkymän ponnahdusikkunan logiikkaan kului aikaa paljon. Vaikka aikaa kului paljon, koen vastaisuudessa osaavani tehdä samanlaista systeemiä nopeammin kuin tänään.

Torstai 15.9.2016

Tavoitteet:
Luoda ja tyylittää tuotesivun kuvaselauslogiikka.

Tuotesivulle tulee yksi iso tuotekuva, jonka alle tulee loput tuotekuvat koossa 50x50px. Kun pientä tuotekuvaa klikataan, tulee se näkyviin ison tuotekuvan tilalle ja iso tuotekuva siirtyy pienten tuotekuvien sekaan uuden ison tuotekuvan tilalle.

Tämä toteutetaan niin, että ensiksi silmukoidaan kaikkien alarivissä olevien kuvien läpi, jonka jälkeen lisätään tapahtuma kuuntelija (event.listener), joka katsoo mitä elementtiä klikataan. Kun klikkaus tapahtuu, verrataan klikatun elementin kuva tunnisteen url-osoitetta ison tuotekuvan vastaavaan. Jos ne ovat erisuuria suoritetaan vaihto.

Seurantaviikko 38

Maanantai 19.9.2016

Tavoitteet:
Luoda tuotesivulle tarvittavia elementtejä ja tyylitellä ne valmistumisen mukaan.

Edellisellä viikolla loin tuotteen kuville kuva selaus systeemin. Tänään aloitin luomalla tunnisteita, tyylittelemällä otsikot, lisää ostoskoriin ja toivelistaan nappulat.

Tunnisteet ovat asiakkaan tuotteelle määriteltyjä attribuutteja. Tässä tapauksessa niitä on uusi, suositeltu ja tarjous tunniste. Jokainen tunniste määritellään magenton hallintapaneelista ja niiden paikka lopullisessa asetelmassa määritellään tuotteen mallinnus phtml tiedostossa, jossa tunnisteita kutsutaan.

Loppupäivä oli tuotesivun peruselementtien tyylittelyä.

Päivä oli tasoltaan helppolaatuinen ja tavoitteet täyttyivät, vaikka työskentelinkin ensimmäistä kertaa Magenton tunnisteiden kanssa.

Tiistai 20.9 ja keskiviikko 21.9
Tavoitteet:
Luoda haitarinavigaatio tuotesivulle. Navigaation tulee sisältää tuotteen pitkä kuvaus, liitetiedostot ja käyttäjän lisäämä kuva.

Magentossa on erikseen tuotekuvat, joita käyttäjä voi määritellä useita. Tarkoituksena on kuitenkin luoda käyttäjälle pohja jolla hän voi lisätä tuotteeseen kuvan värikartasta. Kun on kyse muusta kuin tuotekuvasta, ei tätä voi lisätä tuotekuvien joukkoon, koska se jouduttaisiin aina tuotekohtaisesti eristää näytettävistä kuvista, eikä tämä ole kestävää kehittämistä koodin kannalta.

Luomme oman tuoteattribuutin halutulle kuvalle ja annamme attribuutille arvoja, jotka todellisuudessa ovat halutut kuvat. Jokaiselle kuvalle määritellään tunniste, esimerkiksi #134. Kun asiakas valitsee hallintapaneelista värikartta pudotusikkunan kohdalta #134, palauttaa magenton tuotesivulle #134 tunnistetta vastaavan kuvan.

Päivässä oli todella paljon haastetta ja lisähaastetta toi haitarinavigaation luominen magenton omalla tabs menetelmällä. Tässä tapauksessa jokaista navigaation sisällytettyä välilehteä pystytään hallitsemaan pää xml tiedostosta, jossa niiden pohjatiedostojen sijainnit määritellään. Pohjatiedostot kutsutaan erilliseen tabs.phtml tiedostoon, joka rakentaa itse navigaation. Tämän jälkeen tabs tiedostoa voidaan kutsua getChildHtml metodilla tuotteen mallinnus phtml-tiedostossa.

Torstai 22.9 ja perjantai 23.9
Tavoitteet:
Viimeistellä tuotesivun haitarinavigaation javascript. Ohjelmoida sivulle taittokohdat.

Tuotesivun haitarinavigaatioon tuli lisäominaisuus. Asiakas halusi, että kun osan pitkästä tuotekuvauksesta peittävää lue lisää nappia painetaan, aukeaa tuotekuvaus välilehti haitarinavigaatiosta.

Kyseisen ominaisuuteen hyödynsin jo haitarinavigaatioon tekemiäni funktioita, joita oli kaksi. Ensimmäinen ajoi setOffsetHeight funktion, joka keräsi kaikkien välilehtien sisältöjen korkeuden, minkä jälkeen 0.001 ms päästä se asetti kaikille korkeuden nollaksi. Javascriptin avulla tarkistin, onko elementillä olemassa is-open luokka-attribuuttia, jos oli, se lisäsi edellä mainitussa funktiossa talteen kerätyn korkeuden elementille, jos ei se lisäsi korkeudeksi 0 pikseliä.

Asetin tapahtumakuuntelijan lue lisää napille, minkä avulla kuuntelin klikkausta ja kun tämä tapahtui, lisäsin ID:lle, joka identifioi tuotekuvaus välilehden, is-open luokka-attribuutin ja ajoin molemmat funktion, joka varmistaa is-open luokan, perään.
Päivän tavoitteet olivat oppia käsittelemään javascriptin funktioita ja niille annettuja argumentteja. Suurin osa oli uutta ja opettelemisessa meni hetki aikaa, mutta loppupäivästä ymmärsin jo tekniikan ja osasin soveltaa sitä jo muihin javascript tiedostoihin.

Viikkoanalyysi

Viikon pääteemana oli tuotesivun rakentaminen alusta loppuun. Sivu koostui useasta elementistä, jotka täytyi järjestää näytön koon mukaisesti.

Tuotesivussa oli yksinkertaisia elementtejä, jotka olivat magenton perusominaisuuksia. Esimerkiksi tuotekuvien listaus, tuotteen pitkä ja lyhyt tuotekuvaus ja nimikkeet. Muokattavia elementtejä oli erittäin paljon sisältöön nähden. Värikoodien kuvat ja ladattavat tiedostot täytyi muokata magenton juuresta ja hakea tuotteen mallintamistiedostossa ja koska asiakas haluaa käyttää merkintä systeemiä, joka avaa merkkiä klikattaessa kategoriasivun kyseinen merkki valittuna suodattimena, ei magenton omaa merkintä tapaa voida käyttää. Magentossa valmiina ominaisuutena tuleva tags eli merkinnät toimivat niin että tuotteelle voi määritellä ominaisuuksien kautta merkintä arvoja ja niitä voi kutsua tuotteen mallintamispohjassa. Kun käyttäjä klikkaa tuotteen merkintää, aukeaa uusi sivu, jossa on

kaikki tuotteet, joilla on sama merkintä. Miksi kyseistä merkintätapaa ei voi hyödyntää nykyisessä projektissa? Tulevaisuudessa todennäköisimmin AJAX:ia tullaan käyttämään pääasiallisena tapana tehdä web-sovelluksia.(Nousiainen 2014, 6-7) Tästä syystä käytössämme on yrityksen vakiotyökalu, jota kutsumme Kategoria AJAX:siksi. Työkalu Käyttää ajax-pyyntöä ja siinä on javascriptillä luotuja asetuksia, joita voidaan ottaa käyttöön kutsumalla niitä sen ohjaustiedostossa. Magenton omassa merkintä systeemissä ongelmaksi muodostuu se, että se on ristiriidassa kategoria AJAX:in kanssa. Magentossa tulee mukana suodattimet mutta ne eivät toimi yhtä joustavasti kuin kategoria AJAX-moduuli.

Ongelma magenton omassa suodatinjärjestelmässä on se, että jokaisen valitun suodattimen jälkeen sivu päivittyy. Kuten aikaisemmin totesin AJAX-pyyntö tuo tiedot ilman sivun päivitystä, mikä tekee sivustosta huomattavasti käyttäjäystävällisempää, koska koko sivusto vastaa responsiivisesti käyttäjän pyyntöihin.

Seurantaviikko 39

Maanantai 26.9 ja tiistai 27.9

Tavoitteet:
Luoda tuotesivulle kaavake jonka asiakas voi täyttää ja lähettää kyselyä tai tarjouspyyntöä tuotetta koskien.

Magentossa on valmiina kaavake ominaisuus, joka löytyy magenton ydin tiedostoista projektin pohja kansion alta. Kyseessä on yksinkertainen lomake, jolla pystytään lähettämään tietoja järjestelmänvalvojalle. Kuitenkin se on vajavainen käyttötarkoitukseeni ja siitä täytyy muokata uusi pohja, jota hyödyntämällä saadaan tarvittavat tiedot ulos. Tuotekaavakkeessa tulee olla asiayhteytenä sitä koskevan tuotteen nimi ja tuotekoodi. Nämä haetaan niin että pohjalle annetaan tuotteelle asetettu helper-metodi, jonka avulla se pystyy käyttämään tuotteen luonti tiedostossa käytettäviä tietoja. Näin ollen pystymme hakemaan tuotteen nimen ja tuotekoodin, jotka halutaan lähettää lomakkeen mukana.

Kaavakkeen tyylittäminen oli peruslaatuista syöttökenttien ja tekstialueiden muokkaamista eikä niissä erityisemmin työnsarkaa ollut.

Myös liittyvien ja viimeksi katseltujen tuotteiden moduulit täytyi asentaa tuotesivulle. Viimeksi asennettujen tuotteiden moduuli ei toiminut kunnolla, sillä sen sessionhallinnassa oli virhe. Virhe tapahtui, kun sessio määritteli jokaisen sivun päivityksen jälkeen uuden vierailija id:n tietokantaan, mikä tarkoittaa sitä, ettei samalle id:lle tule useampaa katseltua

tuotetta, eikä näin ollen mitään tule näkyviin. Virheen korjaukseen kului usea tunti, mutta backend ohjelmoijien avulla ongelma ratkaistiin. Ongelmana oli hallinnassa tiedosto, joka määritteli valimusitiin vietäviä tietoja, ja siinä ei oltu otettu huomioon kyseistä ongelmaa.

Päivät olivat haastavia, mutta sisälsivät myös helppoja tehtäviä. Tavoitteena oli saada tuotesivua eteenpäin, mutta koska moduuliongelman ratkaisemiseen kului paljon aikaa, jäätiin päivän tavoitteista.

Keskiviikko 28.9

Tavoitteet:
Saada kaavakkeeseen liittyvät ominaisuudet viimeisteltyä

Kaavakkeeseen täytyi vielä tyylitellä syöttökenttien reunukset ja jokainen fonttikoko. Lisäksi kaavakkeen tekeminen täysin responsiiviseksi vei aikaa. Syöttökentät täytyi laittaa säiliöön, jonka avulla kaikille saatiin sama marginaali.

Kaavake vaati myös javascriptiä. Kun kaavake aukeaa ponnahdusikkunan tapaan, halutaan sen tausta himmentää. Tämä tapahtuu niin, että luodaan elementti, jolla pituus ja leveys 100%, sekä position: fixed. Tällöin se peittää sivun aina, vaikka sivua rullattaisiin mihin suuntaan tahansa. Z-index ominaisuudessa sille täytyy määrittää yhden alempi arvo kuin ikkunalle, näin se peittää kaiken muun paitsi itse ponnahdusikkunan. Kun elementti on valmis, kutsutaan sitä javascriptissä ja samassa nappulan klikkaus funktiossa lisätään sille is-visible luokka attribuutti, jossa voidaan määritellä sille näkyvyys näkyväksi.

Päivän tehtävät valmistuivat ajoissa ja pääsin palaamaan liittyvien tuotteiden pariin.

Torstai 29.9

Tavoitteet
Luoda tuotesivun kaavakkeen puhelinnumero hallittavaksi hallintapaneelista. Luoda ponnahdusikkuna, jossa on kaikki tuotekuvat slider elementissä.

Pohjustin kuvanäkymää aluksi lataamalla valmiit javascriptit slideshow elementtiin, yrityksemme moduuli kansiosta. Ominaisuuden järjestelyyn kului aikaa koska, jotta kyseinen

moduuli toimisi täytyi html css ja javascriptit olla alustettu huolella. Kun sain moduulin toimimaan, pystyin aloittamaan kuvasliderin porrastetun tyyliarkin rakentamisen.

Perjantai 30.9

Tavoitteet:
Tuotesivun kuvakaruselli ikkunan javascriptin ehostus

Tuotteen kuvalle asetettu slider täytyi saada mobiilissa koko näytölle. Haastetta tehtävään toi mobiilinäytön niin sanottu landscape näkymänä, eli sama kun puhelin käännetään poikittain. Siiderille täytyi mitata korkeus aina innerheight ominaisuuden mukaan. Eli kuinka suuri selaimen korkeus on, kun sivustoa selataan.

Tavoitteet olivat pieniä koska työpäivä oli normaalia lyhyempi.

Viikkoanalyysi:

Tuotesivu koostuu useasta osa-alueesta. Tuotenäkymä (kuva 3) ohjaa asiakkaan huomion tuotteeseen jonka sivulla asiakas on. Tuotteesta on esillä tietoja materiaalista, mahdollisia värikarttoja, koska kyseessä on design alaan liittyvä kauppa, sekä kuvia paljon kuvia. Näillä pyritään vaikuttamaan asiakkaan ostopäätökseen ja tämän vuoksi lisää ostoskoriin nappi pidetään hyvin esillä.

kuva 3 Tuotenäkymä

Kuva 4 Liittyvät näkymä

Liittyvät näkymä (kuva 4) tuotesivusta pyrkii ohjaamaan asiakkaan ostamaan lisää tuotteita kaupasta. Siinä näkyy viimeksi katseltujen ja tuotteeseen liittyvien tuotteiden lista.

Edesmennyt viikko keskittyi ensimmäisen osa-alueen tekemiseen. Edellisessä projektissa ei ollut niin laajaa tuotesivua kuin mitä nykyisessä. Pääsin kehittämään osaamistani javascriptin kanssa todella paljon, sillä sivu koostuu useasta javascript elementistä.

Tarkastellaan sivun javascript elementtejä.
Tuotteen kuva ja esikatselukuvat vaativat toiminnallisuuden lisäämistä, jotta niitä pystytään tarkastelemaan. Tässä tilanteessa opin hakemaan javascriptilla image eli kuva merkinnän html rakenteesta ja vaihtamaan sen sisältöä.

Myös tuotteen kuvaa painamalla täytyi saada auki uusi kuvaslider, jossa pystyi selaamaan kaikkia tuotteen kuvia. Vaikka toteutinkin kyseisen toiminnallisuuden valmiiksi koodatulla moduulilla, opin muokkaamaan ja kustomoimaan kyseistä moduulia, mikä vaatii paljon koodirakenteen ymmärtämistä

Haitarinavigaatio, joka pitää sisällään värikartat, liitetiedostot ja pitkän tuotekuvauksen, ei ollut sinällään uutta, koska olin toteuttanut edelliseen projektiin samanlaisen systeemin. Toisaalta nykyisen projektin tuotesivun haitarinavigaatioon sisältyi ominaisuus, jossa navigaation ulkopuoleinen nappi vaikuttaa itse navigaation toimintaan. tätä ominaisuutta en aikaisemmin ollut tehnyt ja jouduin soveltamaan osaamistani.

Ecmascript-6 mukaisesti ohjelmoitua javascriptiä tuli edellisessä projektissa noin 10 tunnin verran. Verrataan että nykyisessä olen tehnyt jo reilusti yli 30 tuntia töitä javascriptin kanssa.

Javascriptin osaaminen ja soveltaminen on tärkeää ohjelmoinnissa. Sen avulla pystytään lisäämään dynaamista toiminnallisuutta nettisivulle. Siksi koen itsekin tärkeänä jatkaa oman javascript osaamiseni kehittämistä.

Seurantaviikko 40

Maanantai 3.10 ja tiistai 4.10

Tavoitteet:
Tuotekuvakarusellin luominen viimeksi katseltujen tuotteiden tuotekorttilistasta. Kuvakarusellin toiminta perustui useamman tuotekortin samanaikaiseen näyttämiseen. Tämä porrastettiin taittokohdista riippuvaisiksi. Pienillä mobiililaiteitta tuotteita on näkyvissä kaksi, normaaleilla moobililaitteilla kolme, tableteilla neljä ja normaalissa työpöytänäkymässä on viisi, joka on tuotteiden maksimimäärä, mikä tarkoittaa sitä, että karuselli ominaisuus pudotetaan tässä vaiheessa pois.

Kyseessä on sama moduuli kuin mitä käytin tuotteiden kuvien kuvakaruselliin. Suoraa asetusta moduuliin ei ole koodattu, missä pystyisi asettamaan usean elementin samanaikaisesti näkyville, vaan tämä täytyi kustomoida.

Sain kustomoinnista valmiiksi noin 50%. Käyttämällä javascriptin prototyyppi objektia, pystyin ylikirjoittamaan moduulin alkuperäistä funktiota mutta samalla hyödyntämään koodin runkoa.

Keskiviikko 4.10, torstai 5.10 ja perjantai 6.10

Tavoitteet:
Jatkaa tuotesliderin työstämistä

Heti alussa huomioin, että käyttämäni swipe-slider moduulin ylikirjoittamiseen tulee kulumaan huomattava määrä aikaa. Koska javascript osaamiseni ei ole vielä tasolla, jossa pystyisin kirjoittamaan nopealla vauhdilla toimivia prototyyppi funktioita ja koska projektin

hallinnassa kyseiselle tehtävälle oli määritelty tuntimääräksi kuusi tuntia, täytyi keksiä nopeampi ja toimivampi ratkaisu.

Edellisessä projektissa kaupan blogisivun yksittäiseen artikkeliin pystyi kaupan hallinnoija lisäämään liittyviä tuotteita. Tuotteet näytettiin myös tuote slideshow elementissä, jossa virtaus oli vertikaalinen.

Keskustelin kollegani kanssa, joka oli kyseisen moduulin ohjelmoinut, ja kävi ilmi, että vertical-slider moduulin oli lisätty ominaisuus, joka mahdollisti myös horisontaalisen elementti virtauksen.

Koska kyseinen moduuli oli minulle paljon tutumpi ja osasin käyttää sitä myös paremmin, tulin siihen tulokseen, että on järkevämpi siirtyä käyttämään sitä.

Vertical-slider moduulissa tulee yksi iso javascript tiedosto, jossa sen toiminta on määritelty. Mitä moduulin käyttäjän tulee osata, on hakea asetuksia ja tutustua niiden toimintaan.

Mikä teki vertical-slider moduulista niin paljon helpomman kuin swipe-slider, oli sen fluid asetus. Vapaasti suomennettuna sulava asetus. Kun määriteltynä käyttöön, laski automaattisesti sen, montako tuotetta mahtuu yhtäaikaisesti näytölle. Tämä helpotti huomattavasti työntekoa, koska nyt yksinkertaisilla ehtolauseilla, jotka määritellään sivun taitekohdassa, voidaan kertoa moduulille, montako tuotetta halutaan näyttää.

Tavoitteisiin päästiin kolmen päivän aikana ja nyt kun sliderin javascripti tiedosto, jossa sen asetukset määritellään, voidaan ladata minne vain sivulle, voi tuote slideria käyttää nyt nopeasti missä vain

Viikkoanalyysi

Lisää näyttävyyttä verkkosivuille tuovat kuvakarusellit. Niiden tarkoituksena on näyttää paljon tietoa yhdessä säiliössä, jota asiakas pystyy itse selaamaan.

Kuvakarusellien tarkoitus on vaihteleva ne voivat toimia mainosmaisesti, näyttämällä tarjouksia ja linkittämällä niihin. Verkkokaupoissa niillä voi listata tuotteita. Urheilujoukkue sivuilla niillä pystytään, vaikka listaamaan uusimmat uutiset. Mutta olit tapaus mikä tahansa, pohjimmainen idea säilyy, eli tuodaan sivuston käyttäjälle paljon tietoa näkyviin.

Kuvakarusellit toimivat pääasiassa javascriptilla. On olemassa hyvin paljon javascript kirjastoja, joihin on rakennettu karuselli liitännäisiä.

Yrityksessä, jossa työskentelen, ohjelmoimme kaiken niin sanotulla vanilla javascriptilla. Tämä tarkoittaa sitä, että ulkoisia kirjastoja ei hyödynnetä, vaan kaikki rakennetaan alusta alkaen itse. Kaikki javascript osat mitä kauppoihin tulee, on ohjelmoitu ja dokumentoitu alusta alkaen yrityksen työntekijöiden puolesta. Tämän ansiosta on kauppojen ylläpito helpompaa, koska jotain hajotessa ei työntekijöiden tarvitse perehtyä ulkoisten javascript kirjastojen dokumentointeihin.

Koodin laadun ja toiston minimoimisen kannalta on erittäin tärkeää, että javascript komponentit pystytään rakentamaan moduuleiksi ja jakamaan. Jotta useammat työntekijät pystyvät hyödyntämään samoja komponentteja, käytämme javascript komponenttien hallintaa NPM paketinhallintaa. NPM tulee node.js runkorakenteen mukana ja sen avulla pystytään hallinnoimaan javascript moduuleja helposti. Moduuli tarvitsee ladata NPM-tiliin, jonka kautta sen pystyy lataamaan mihin projektiin, mille koneelle tahansa.

Seurantaviikko 41

Maanantai 10.10

Tavoitteet:
Määritellä tuotesivun sliderille responsiiviset säännöt ja aloittaa nopean näkymän muokkaaminen.

Tuotesivun sliderille täytyi vielä määritellä korkeudet, jotka sopivat sen sisältöön jokaisessa sivun taittokohdassa. Sisällölle täytyi asettaa css arvoksi automaattinen korkeus. Javascript funktion tehtävänä oli laskea sisällön korkeutta ja tehdä koko sliderin säiliöstä sen korkeuden kokoinen, mutta lisätä vielä tähän 50 pikseliä.

Loppupäivästä aloitin nopean näkymän muokkaamisen. Nyt kun tuotesivu on tyylitelty, voin käyttää samaa css tiedostoa ja phtml pohjaa nopeassa näkymässä. Tämä kuitenkin vaatii muokkauksia.

Tavoitteet saavutettiin hieman etuajassa ja nyt on tärkeää viimeistellä nopeanäkymä tämän viikon aikana.

Tiistai 11.10

Tavoitteet:
Jatkaa nopean näkymän viimeistelyä.

Ongelmaksi nopeassa näkymässä muodostui ajax-pyynnön hitaus. Nyt kun kaikki elementit ja tyylitiedostot ovat paikallaan, huomasin että ajax latautuu juuri 0.5 sekuntia liian myöhään, mikä tarkoittaa sitä, että animaatiot täytyy mukauttaa. Vaikka animaatiot saatiinkin mukautettua, on nopean näkymän avaus silti liian hidas. Tämä tarkoittaa sitä, että siihen on lisättävä latauskuvake per nopeanäkymä. Lisäämällä uuden pohjan pää xml-tiedoston kautta projektiin ja lisäämällä halutun ikonin kyseiseen tiedostoon, pystyn kloonaamaan sen teoriassa jokaiseen haluttuun nopeaan näkymään cloneNode funktion avulla.

Päivän tavoitteisiin ei päästy, mutta nopeanäkymä on alustettu ja tarvitsee enää latausikonin kloonit, johon huominen pitäisi riittää.

Keskiviikko 12.10

Tavoitteet:
Jatkaa latausikonien viimeistelyä, pitää palaveri etusivun sisällöstä ja aloittaa sen suunnittelu.

Kloonaamalla svg muodossa olevan latausikonin aina tuotekortin alapuolelle tekee sivun toiminnasta varsin kevyttä.
Latausikonille täytyi antaa absoluuttinen positio ja left arvoksi 50% jotta se keskittyy horisontaalisesti. Vertikaaliseen keskittämiseen pystytään vaikuttamaan antamalla eri taittokohdissa eri ylä-marginaali.

Päivän tavoitteisiin päästiin ja ensi viikolla pystyn aloittamaan etusivun sisällönhallinnan.

Viikkoanalyysi

Projektin koosta riippuen, siinä yleensä työskentelee 2-4 henkilöä. Jotta koodin laatu ja yhteisten ohjelmapalasten kokoaminen olisi helpompaa, on järkevää käyttää versionhallintaa. Git on ilmainen alun perin Linus Torvaldsin julkaisema GNU lisenssiin nojaava vapaan lähdekoodin versionhallintaohjelmisto, joka on suunniteltu toimimaan projektin koosta riippumatta hajautetusti ja mahdollisimman tehokkaasti.

Git:in perusideana on mahdollistaa usean henkilön työskentely saman projektin parissa mahdollisimman nopeasti, estää datan katoaminen ja virheellisyys. Käyttäjät voivat luoda omia ”oksiaan”, joihin voidaan tehdä kriittisiä muutoksia ilman hajottamatta ”pääoksaa”, joka on kaikkien oksien tulos. Projektinhallinnassa oksille voidaan antaa id ja tuntiarvio. Näin ollen, kun oksa on valmis, voidaan se yhdistää pääoksaan ja siihen käytetyt tunnit kirjata ylös. Oksa on yleensä määritelty jonkun tietyn scrum-tehtävän mukaan ja oksan valmistuttua kyseinen tehtävä voidaan siirtää valmiisiin tehtäviin.

Git toimii unix-periaatteiden mukaisesti työkalusarjamaisen suunnittelutyön mukaan. Vaikka useat sen komponentit ovat uudelleenkirjoitettu C-kielellä, sen alun perin suunniteltu runko on pysynyt pääosin samana.

Git on äärimmäisen hyödyllinen it-yrityksille myös valmiiden projektien hallinnassa. Kun projekti tarvitsee päivityksiä, huoltotoimenpiteitä tai jatkokehitystä, voidaan projekti kloonata sen säilytyspaikasta pilvestä. Tämän avulla niin sanottua vanhaa versiota projektista pystyy käyttämään koko sen ajan, kun toimenpiteitä tarvitsee tehdä.

Seurantaviikko 42

Maanantai 17.10

Tavoitteet:
Lisätä tuotesivulle liittyvät blogiartikkelit ja aloittaa etusivun sisällön tekeminen.

Liittyviin blogiartikkeleihin kului kaksi tuntia aikaa, joista suurin osa meni wordpressin asentamiseen ja pohjatiedostojen alustamiseen.

Blogiartikkelilistauksen tyylit sen sijaan olivat valmiina, koska kollegani oli tehnyt jo blogisivun css koodin valmiiksi.

Loppupäivästä pääsin aloittamaan etusivun sisällönhallinnan.

Tiistai 18.10 ja keskiviikko 19.10

Tavoitteet:
Saada etusivun sisällönhallinta siihen tilaan, ettei back-end ja html rakennusta tarvitse tehdä enää loppuviikosta.

Etusivun sisältöruudukkoon tulee säiliöitä, jotka ovat suorassa yhteydessä hallintapaneelin ja niitä pitää pystyä päivittämään sieltä, blogi artikkeleja, jotka latautuvat automaattisesti ilman asiakkaan toimia ja tuotekortteja, jotka latautuvat suoraan tuotteelle ladatun ominaisuuden kautta.

Back-end toiminnallisuuksien alustamiseen ja tekemiseen kului käytännössä koko tiistai ja keskiviikkona puolestaan pääsin rakentamaan listauksen html rakenteen, jossa täytyi olla erityisen tarkkana ja olinkin laatinut suunnitelman porrastetun tyyliarkin suhteen, joten tiesin mihin järjestykseen ja minkälaisiin säiliöihin halusin elementit.

Tavoitteet saavutettiin ja loppuviikosta pääsen tekemään tyylejä etusivulle. Etusivun ruudukko oli mielenkiintoista ja siinä joutui soveltamaan magenton backend toiminnallisuuksia todella paljon saadakseen oikeanlaista dataa ulos.

Torstai 20.10

Tavoitteet:
Hioa etusivun sisällönhallinnan porrastettu tyyliarkki pääpiirteittäin valmiiksi.

Sisällönhallintaa varten täytyi luoda, ruudukko joka skaalautuu näytön koon mukaan ja listattavien elementtien määrä vaihtelee.

Erityisen hankalaa ruudukon luomisesta teki se, ettei css flex-box:ia pystynyt suoranaisesti hyödyntämään. Tämä tarkoitti sitä, että ruudukko täytyi rakentaa käsin.

Tein skaalautuvat elementit niin, että itse elementillä on korkeus 0 ja padding-bottom arvo prosentuaalinen. Kun elementin leveydeksi määritellään prosentti arvo, laskee padding bottom aina pikselinsä leveydestään. Näin elementistä saadaan skaalautuva

viikkoanalyysi

Kuten jo päiväanalyyseissä totesin, teki etusivun sisällönhallinnasta hankalaa se, ettei css:n flex-box ominaisuutta pystytty käyttämään. Flex-box:in avulla olisi ruudukon tekeminen ollut joustavampaa ja nopeampaa.

Flex-box suunnittelu perustuu tehokkaaseen tapaan luoda, sijoittaa, järjestellä ja levittää elementtejä ja niiden säiliöitä dynaamisesti, vaikka niiden säiliön kokoa ei olisi tiedossa. (Coyier 2016). Sen toimintaperiaate on muokata flex-säiliön sisällä olevia elementtejä taulukkomaiseen tapaan. Flex-box:in ideologiassa flex-säiliön lapsi elementti pystyy täyttämään aina tyhjän tilan ja skaalautumaan sulavasti erikokoisille näytöille.

Flex-boxin helppous piilee nimenomaan sen muokattavuudesta. Lapsi elementit pystytään järjestämään yhdellä asetuksella horisontaalisesti tai vertikaalisesti ja helposti linjata mihin kohtaan tahansa.

Ruudukkoa rakentaessa käytin ratkaisua, jossa verrattiin säiliön pohjan pituutta prosentuaalisesti sen korkeuteen. Kun korkeuskin on prosentuaalinen, muuttuu säiliön koko jokaisessa näyttö koossa. Prosentuaalisissa korkeuksissa ongelmaksi muodostuikin jaollisuus. Mikäli halutaan erimäärä erikorkuisia elementtejä ruudukkoon, ei esimerkiksi kolmea elementtiä saa tarkalleen vierekkäin. Tämä tarkoittaa sitä, että ruudukon skaalautuessa sadasosien ero korkeudessa tai leveydessä saattaa hajottaa sen.

Flex-box laskee suoraan elementtien leveyden ja täyttää automaattisesti tyhjän tilan. Näin ollen, kun määrätään kolme elementtiä vierekkäin ja annetaan leveydeksi 33.3%, tasaa flex-box loput pikselit, eikä erinäistä korjauskoodia tarvitse kirjoittaa korjatakseen sadasosien pikseleitä, vaikka tämä css:n calc funktiolla jokseenkin onnistuu.

Flex-box työkaluna on monimutkaisiin, paljon elementtejä ja järjestelyä omaaviin projektin osiin täydellinen ratkaisu. Sen yksi erinomaisimpia ominaisuuksia on flex-order, jossa elementeille pystytään numeroinnin avulla määrittelemään paikka listauksessa. Sitä pystyy hyödyntämään myös pienenpien osioiden keskittämis-ja sijoittamisongelmissa.

Maanantai 24.10

Tavoitteet:
Aloittaa kirjautumissivun rakentaminen

Kirjautumissivulle tulevat perus ominaisuudet, kuten kirjautumiskentät ja uusien asiakkaiden rekisteröiminen olivat ensimmäisiä asioita, joita täytyi phtml-tiedostoon rakentaa. Tämän jälkeen tyylitin kyseiset elementit.

Keskiviikko 26.10

Tavoitteet:
Jatkaa kirjautumissivua, valmistella asiakastilisivu ja kloonata etusivun sisällönhallinta.

Etusivulle tuleva ruudukko täytyi kloonata ja siihen tulevat palikat laittaa peilattuun järjestykseen. Koska käytin ruudukon järjestämiseen css float ominaisuutta, jonka arvoksi oli asetettu left, eli sisältö oli kellutettu vasemmalle, pystyin nyt vaihtamaan float arvon päinvastaiseksi ja ruudukosta muodostui edeltäjänsä peilikuva

Torstai 27.10
Tavoitteet:
Viimeistellä kirjautumissivu ja asiakastilin omat tiedot sivu.

Asiakastilille tulee tietoa asiakkaan yhteystiedoista, tilaushistoriasta ja toimitusosoitteista. Tämän lisäksi asiakkaan pitää pystyä muokkaamaan omia tietojaan. Asiakastilisivut ovat tietojen puolesta hyvin standardeja paketteja kauppojen suhteen, joten niihin pystytään käyttämään modulaarista css-kirjastoa. Tätä soveltaen pystyin säästämään useita työtunteja.

Viikkoanalyysi
Projektia kehitettäessä on huomioitava sen mahdollinen jatkokehitys tai huolto/korjaus prosessi. Projektin siistimistä helpottaa modulaariset scss kirjastot. Tämä tarkoittaa, että tietty osa on ohjelmoitu jo valmiiksi omaksi moduulikseen ja pystytään liittämään osaksi

projektia. Moduuleihin määritellään perusarvoja osiolle, jonka päälle ohjelmoija, joka ohjelmoi projektia, voi rakentaa tarvittavat mukautukset.
Projektin jatkuvuuden kannalta modulaariset koodikirjastot helpottavat projektien jatkokehitystä huomattavan paljon. Kun jokaisen projektin osa-alueen koodi pystytään pitämään standardoituna, on projektin ulkopuolisten työntekijöiden helpompi tehdä huoltotehtäviä ja jatkokehitystä. Myös uusien työntekijöiden tutustuttaminen projektien toimintaan helpottuu.

Sass on front-end ohjelmointikieli, jonka toimintaperiaatteet ovat lähempänä oliopohjaista ohjelmointitapaa verrattuna esimerkiksi normaaliin css kieleen. Sass:sin syntaksilla scss:lla pystytään tallentamaan arvoja erilaisiin muuttujiin ja kutsumaan niitä koodissa. Tämän avulla front-end kehittäjien on helppo vaihtaa tiettyjen elementtien yhtenäisiä ulkoasullisia arvoja yhtenevästä tiedostorakenteesta.

Moduulien tarkoitus ei ole silti ulkonäöllisesti tehdä kaikkien projektien tietyistä ominaisuuksista samanlaisia, vaan antaa peruseväät, joiden avulla pystytään säästämään aikaa ja keskittymään asiakkaan tarpeitten räätälöimiseen.

Seurantaviikko 43

Maanantai 7.11

Tavoitteet:
Alustaa asiakastilisivuille muistilista välilehden tyylit.

Projektiin tulee muistilistaominaisuus, jossa asiakas pystyy lisäämään tuotteita eri muistilistoille ja hallitsemaan listoja. Asiakastilisivulla näkyy kaikki muistilistat ja tämän ominaisuuden tyylejä tänään ohjelmoin.

Jokainen lista oli omassa haitari osassaan. Tämä tarkoittaa sitä, että jokaisen muistilistan nimi näkyi listassa ja kun nimeä klikattiin, aukesi listan sisältö suoraan nimen alapuolelle. Olen tehnyt kyseisen ominaisuuden javascriptilla ja koska se ilmaantuu projektissa nyt niin monessa kohdassa, tein siitä universaalin scriptin projektiin. Tähän kului tunti, mutta nyt pystyn käyttämään sitä, jokaisessa haitariominaisuudessa.

Loppupäivän aikana sain tyylit valmiiksi muistilistaosuuteen, joten tavoitteet saavutettiin päivän osalta.

Tiistai 8.11

Tavoitteet:
Aloittaa lisää muistilistalle ominaisuuden tyylien rakentaminen kaikille sivuille, joissa tarvitaan tuotekortteja.

Muistilistaan pystytään lisäämään kaikilta sivuilta, joissa on tuote näkyvissä, missä tahansa muodossa. Aloitin tuotesivusta ja tyylitin tarvittavat napit ja linkit sekä lisäsin muutaman javascript ominaisuuden niihin. Tämän jälkeen kuitenkin huomasin, että muistilistalle lisättäessä sivu latautuu uudestaan. Käyttökokemuksen kannalta lisääminen täytyy saada ilman sivunlatausta. Moduulista vastuussa oleva back-end kehittäjä aloitti iltapäivästä lisäämään ominaisuutta sivulle ja sovimme että huomenna pääsen konfiguroimaan ajaxia, jossa todennäköisesti kuluu loppuviikko.

Tavoitteellisesti saavutettiin lisäyksen perustyylit joka sivulle, mutta ominaisuuden tekemisess tulee kestämään mahdollisesti loppuviikko, koska ajaxin tarve huomattiin liian myöhään.

Keskiviikko 9.11

Tavoitteet:
Aloittaa ajax-konfigurointi muistilistan lisäystoiminnassa.

Sain heti aamusta versionhallinnan kautta ajax ominaisuudet käyttööni. Ajaxin avulla pystyin nyt antamaan url-osoitteen, jossa lisäys tapahtuu, ajaxin käsiteltäväksi. Tuotteella täytyy olla ominaisuutena: lisää ja poista olemassa olevalta muistilistalta ja luo uusi muistilista.

Tänään sain ajaxin avulla kaikki ominaisuudet toimimaan. Kuitenkin tarvitsin apua kollegalta ajaxin konfiguroimisessa, jotta koodi noudattaisi ns. best practise ratkaisuita.

Torstai 10.11

Tavoitteet:
Viimeistellä muistilistan lisäys-osio.

Tällä hetkellä kaikki ominaisuudet toimivat, mutta käyttäjä ei saa tarpeeksi infoa siitä, että tuote on lisätty muistilistalle. Vaikka eilen valmistinkin ilmoituksen siitä, kun tuote on lisätty tai poistettu lista, täytyy vielä toteuttaa ominaisuuteen html:n uudelleenlataus. Tämä sen takia koska, jos tuote on lisätty jollekin listalle, tulee siihen ulkonäöllisiä muutoksia. Näitä ei pystytä tekemään, jos muistilistan lisäystoimintojen html osiota ei saada päivitettyä.

Päivän aikana hioin muistilistan loppuun ja sain sen valmiiksi. Nyt inner.html latautuu oikeassa kohdassa ja tuotteen tiedot päivittyvät.

Perjantai 11.11

Tavoitteet:
Varmistaa että kaikilla sivuilla toimii muistilista ja hioa ominaisuus loppuun.

Päivän tavoitteet eivät toteutuneet, koska aamusta sain tehtäväkseni muokata nopean näkymän tuotekuvien näyttöä. Nopeannäkymän tuotekuvalista oli suunniteltu toimimaan normaalina lista elementtinä, joka rivin muodossa skaalautuu säiliön mukaan. Nyt kuitenkin haluttiin tehdä tästä ns. ruudukko näkymästä, slider näkymä. Aikaa vievä asia operaatiossa oli se, että sliderin elementit generoituvat vasta ajax pyynnön luotua sisällön nopeaan näkymään. Hyödynsin slideriin tuoteslider pohjaa, jota käytin tuotesivulla. Sitä joutui kuitenkin muokkaamaan ajax pyynnön takia, mistä syntyi lisätunteja.

Viikkoanalyysi:

Viikko oli haastava, sillä jouduin taas palaaman AJAX:in pariin. Tällä kertaa pelkästään AJAX:in perusominaisuuksien hyödyntäminen ei riittänyt, vaan nyt jouduin soveltamaan hieman funktiota.

Edellisessä tapauksessa nopean näkymän ohjelmoimisessa riitti vain yksi funktio joka palauttaa inner.html elementin avulla tiettyyn säiliöön sen html sisällön. Tässä tapauksessa täytyi luoda kolme funktiota, jotka kaikki kutsuvat AJAX:ia päivittämään säiliönsä html rakenteen.

Ongelmat AJAX:in kanssa kuitenkin ratkesivat viikon aikana ja ohjelman osio saatiin valmiiksi, vaikka pientä hiomista jäikin vielä jäljelle.

Kuten mainitsinkin, AJAX funktiossani on kolme osiota. Ensimmäisen on tarkoitus antaa AJAX:ille url osoite, jossa tuote lisätään muistilistalle, toisessa sama, mutta poistetaan ja kolmannessa sama mutta luodaan kokonaan uusi muistilista. Jokaisessa funktiossa ajetaan html:n päivitys säiliölle, jolloin ulkonäöllisesti napit ja linkit päivittyvät.

Olen saanut painia nyt useamman kerran AJAX pyyntöjen kanssa, enkä koe vieläkään täysin omaksuneeni asiaa. Oppimistapoja on monia, mutta tällä hetkellä olen opiskellut perusteita AJAX:iin internetistä ja kollegoiden opastuksella. Kuitenkin ohjelmointi vaatii sen, että ymmärtää kirjoitetun koodin merkityksen. Tarkkojen ohjeiden perusteella kopioitu koodi ei lasketa ohjelmoinniksi (VIITE kohta 2.1). Tästä johtuen ei oppimista myöskään tapahdu. AJAX ei ole ohjelmointi kieli vaan on työkalusarja, joka koostuu useista ohjelmointikielistä. Tämän takia sen oppiminen vaatiikin tekijältä hieman enemmän kuin yksittäisen ohjelmointikielen oppiminen.

4 Pohdinta ja päätelmät

Päiväkirjaopinnäytetyö on tuonut erilaista perspektiiviä työntekooni. Olen pystynyt seuraamaan omaa työntekoani pitkän ajanjakson ajan. Tällä ajanjaksolla aloitin uuden projektin, jossa hyödynnettiin yrityksellemme tuntematonta projektityöskentelymenetelmää, jossa työryhmäni oli kyseisen menetelmän uudisraivaajat. Opinnäytetyö sattui siinä mielessä sopivaan aikaan, että pääsin analysoimaan itseäni monella osa-alueella.

Ennen kuin aloitin raportoinnin, oli projektiosaamiseni aloittelijan tasolla. Ensimmäinen projektini oli pieni projekti ja se ei noudattanut SCRUM projektin hallintaa. Uudessa projektissa suoritettiin kunnon projektisuunnitelma ja tehtävät määriteltiin huolella. Suunnittelun lopputuloksena syntyy realistinen toteutussuunnitelma, joka vastaa laatimishetken parasta tietämystä. (Sundström 2014, 14-15). Tämä helpottaa huomattavasti projektissa työskentelevien henkilöiden työn rytmittämistä ja sprinteissä toimimista. Tähän oivan lisän toi opinnäytetyön sisältö, koska jälkikäteen näki suoraan mitä on päivittäin tehnyt, missä on onnistuttu ja missä ollaan jääty jälkeen. Työmäärien arviointia on tehtävä koko projektin

elinkaaren ajan, eikä vain alussa. (Sundström 2014, 15-16). Lisäksi lisätöitä tuovat projektin määrittelyvaiheessa huomiotta jääneet seikat. Nykyisessä projektissa suurin osa näistä seikoista liittyivät teknisiin toteutuksiin enemmän kuin ulkonäöllisiin. Hyvänä esimerkkinä seurantaviikolla 43 suoritetut AJAX ominaisuudet.

Vaikka projektityöskentelytavat olivat uusia minulle, koin saavuttaneeni uuden tason projektityöskentelytaidoissani yksinkertaisesti oman työskentelyn ja tehtävien aika ja laatuarvioiden perusteella.

Työni pääpaino oli kuitenkin selkeästi ohjelmoinnissa ja sen opettelussa. Ensimmäisessä projektissani ennen opinnäytetyön ajanjaksoa olin pääasiassa työskentelemässä front-end elementtien parissa ja koskin vähäisesti back-end ominaisuuksiin. Nyt syksyllä alkaneessa projektissa tilanne oli toinen. Kahdesta projektissa työskentelevästä ohjelmoijasta, minä olin kokeneempi ja vaikka kaikista työtehtävien teknisistä toteutustavoista ei ollut tietoa, täytyi ne opetella projektin aikana. Hyödyllistä tämän tyyppisessä tekemisessä oli se, että asiat olivat pakko opetella, mikäli halusi saada projektia eteenpäin. Toisaalta ohjelmoinnilla itsessään on merkittävän suuri kognitiivinen kuorma (Lindholm 2016, 13) ja oleellisinta sen opettelussa olisikin kokonaisuuden ymmärtäminen. Tähän vedoten, mikäli uusia asioita on paljon opeteltavana lyhyeen aikaan, ei ihminen välttämättä pysty omaksumaan yhden osa-alueen tietoja.

Projektissa täytyi myös uusien tekniikoiden lisäksi kehittää jo edellä hankittua osaamista ja uskoisin että tässä otin suurimman harppaukseni. Ohjelmointi vaatii työ- muistia, keskittymiskykyä ja kielellisiä toimintoja. Se on aktiivista toimintaa, jossa pyritään ratkaisemaan tietty ongelma käyttäen ohjelmointikieltä ja koneelle tyypillistä tapaa toimia (Lindholm, 2016). Kun tiettyä ongelmaa on useamman kerran jo ratkaissut, kykenee ihminen soveltamaan luovempia ratkaisuita ongelman suhteen. Tästä johtuen koodin laatu paranee. Omassa tapauksessani erityisesti oma Javascript osaamiseni otti askeleen eteenpäin ja pystyin luomaan vähemmän toistavaa dynaamista script kieltä.

Projektia ohjelmoidessa ei ongelmakohtia esitetä suoraan kysymyksinä kuten opetusmateriaalissa, vaan ohjelmoija muodostaa ongelmasta oman ratkaisun ja aloittaa sen työstämisen. Alussa oma työskentelytapani oli aloittaa ohjelmoimaan ensimmäistä ratkaisua, jonka sain mieleeni. Ongelmatilanteen tullessa vastaan, kävin suorittamaan uutta ongelmanratkaisutilannetta ilman, että edes ajattelin muuttavani kokonaiskuvan ratkaisua. Tällöin koodi kyllä toimii oikein, muttei aina välttämättä tule olemaan parhaimpien ohjelmointikäytäntöjen mukaista. Ohjelmoinnin luontaista kokonaiskuormaa ei pystytä vähentämään,

mutta kokonaiskuroman jakaminen pienempiin osiin on mahdollista ja siten kerralla käsiteltävä kuormaa vähenee. (Lindholm, 2016). Tätä käytäntöä hyödyntäen koin saavani ohjelmointiratkaisuihin selkeämmän ja helpommin luettavan kaavan. Sen sijaan että lähdin suorittamaan ensimmäisenä mieleeni tullutta ratkaisua, listasin kaikki mahdolliset ratkaisut, niiden heikkoudet ja vahvuudet, sekä niiden universaalin hyödynnettävyyden projektissa. Tällä tavalla olen pystynyt seulomaan parhaimman ratkaisun ongelmaan.

Työn rytmittämisessä on ollut apuna erilaiset ajanhallinta työkalut, joilla pystyy tehostamaan oman työnteon ajankäyttöä. Kyseistä työkalua, jota käytin, toimi suoraan lisäosana tekstieditorissa ja aina 25 minuutin välein se laittoi ohjelman lukkoon. Toisin sanoen pakotti käyttäjän pitämään tauon. Tällaisessa rytmittämisessä oli helppoa suunnitella työpäivälle tehtävät. Ylipäätänsä ennalta suunnittelu ohjelmointitehtävissä tuo selkeyttä projektityöskentelyyn. Näin pystyy helposti yhdistelemään eri ohjelmien osia muiden projektissa työskentelevien työntekijöiden kanssa ilman konflikteja versionhallinnassa. Opinnäytetyön aikana tehdyt päivämerkinnät auttoivat merkittävästi ongelmatilanteissa. Niiden avulla pystyi palaamaan ongelman juurille ja miettimään eri ratkaisuvaihtoehtoja, mikäli tultiin tilanteeseen, että jotain ohjelman osaa täytyi muuttaa tai, että jokin ei toiminut niin kuin sen pitäisi.

Kiinnostavin opinnäytetyön aikana esiin tullut asia oli, oman järjestelmällisyyden kehittyminen ohjelmoinnissa. Projekteissa pyritään aina parhaaseen laatuun ja koodin täytyykin silloin olla automatisoitua ja helposti ymmärrettävää, vaikka työskentelisikin projektin ulkopuolella. Järjestelmällisyydellä tarkoitan jo edellä mainittua ongelmanratkaisun kaavan kehittymistä, työtehtävien ennalta suunnittelua ja työryhmän kanssa kommunikointia. Kyseisillä osa-alueilla on suorayhteys koodin laadun paranemiseen. Ongelmanratkaisun kehittyminen tuo selkeämpää koodia, kun ratkaisut ovat tarkkaan ennalta mietittyjä. Työtehtävien suunnittelu auttaa luomaan universaaleja ratkaisuita joiden avulla koodista tulee automatisoidumpaa. Työryhmän kanssa kommunikointi ehkäisee toistoa ja konflikteista johtuvia virheitä koodissa

Lähteet

Coyer C. 2016. Complete guide to flex-box. Luettavissa: https://css-tricks.com/snippets/css/a-guide-to-flexbox/ Luettu. 1.10.2016

Dodia C. 2014. Magento install and upgrade data scripts explained. Luettavissa: https://www.sitepoint.com/magento-install-upgrade-data-scripts-explained/ Luettu 8.11.2016

Garret J. Ajax: a new approach to web applications. Luettavissa:http://adaptive-path.org/ideas/ajax-new-approach-web-applications/

Lindholm R. 2016. Kognitiivinen kuorma ohjelmoinnin oppimisessa. Luettavissa: https://jyx.jyu.fi/dspace/bitstream/handle/123456789/50236/URN%3aNBN%3afi%3ajyu-201606093000.pdf?sequence=1 Luettu 1.11.2016

Makkonen S. 2015. Animoinnin merkitys Web-kehityksessä. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/102682/ont_Makkonen.pdf?sequence=1 Luettu 20.11.2016

Nousiainen H. 2014. Ajax-pohjaiseen web-sovelluksen toteuttaminen. Luettavissa: https://theseus32-kk.lib.helsinki.fi/bitstream/handle/10024/82999/HenriNousiainen.pdf?sequence=1 Luettu 5.11.2016

Rouse M. Ajax. Luettavissa: http://searchwindevelopment.techtarget.com/definition/Ajax

Sundström T. 2014. Tapaustutkimus IT-projektien haasteista ja ongelmista. Luettavissa: http://www.theseus.fi/bitstream/handle/10024/81653/Sundstrom_Tagoror.pdf?sequence=1 Luettu 5.11.2016