Miki Helin

RESPONSIIVINEN WEB-SUUNNITELMA

Tietojenkäsittelyn koulutusohjelma

2020

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

RESPONSIIVINEN WEB-SUUNNITELMA

Helin, Miki
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Maaliskuu 2020
Sivumäärä: 35
Liitteitä:

Asiasanat: html, css, www-sivut, suunnitelma

Tämän opinnäytetyö käsittelee mitä responsiivinen web-suunnitelma sisältää. Opinnäytetyössä käydään lävitse responsiivisen web-suunnitelman menetelmiä ja käytettyjä teknologioita. Opinnäytetyö sisältää toiminnallisen osuuden, jossa suunnitellaan ja rakennetaan responsiivinen web-sivusto vaiheittain.

RESPONSIVE WEB DESIGN

Helin, Miki
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Business Information Technology
March 2020
Number of pages: 35
Appendices:

Keywords: html, css, web page, design

The purpose of this thesis was to research what responsive web design contains. This thesis goes through responsive web designs used practices and technologies. This thesis contains a practical section where a responsive web site gets built on a step-by-step basis.

TERMISTÖ

RWD	Responsive Web Design
Breakpoint	Keskeytyspiste, jolla saadaan suunniteltua ulkoasuja monen muotoisille laitteille. Kyseessä ei ole debugging breakpoint.
Container	Sivuston rakenteesta muodostuva alue verkkosisällölle.
HTML5	Hyper Text Mark Language 5
CSS	Cascade Style Sheet
XML	Extensive Markup Language
DOM	Document Object Model
SVG	Scalable Vector Graphics
W3C	World Wide Web Consortium
Card	Joustava ja laajennettava alue verkkosisällölle

SISÄLLYS

1 JOHDANTO

Verkkosivujen selaaminen mobiililaitteiden ja tablet-laitteiden kanssa on viime vuosina noussut erittäin paljon. Syynä on mobiililaitteiden suuri kasvu ja laitteistojen hintojen lasku. Yrityksen Statcounterin mukaan vuonna 2016 mobiililaitteita käytettiin ensimmäistä kertaa verkkoselaamiseen enemmän kuin tietokoneita. 51,3% Internetin käytöstä tapahtui mobiililaitteiden kautta. (Statcounter, 2016) Yritykset ovat näiden tutkimusten perusteella alkaneet kehittämään verkkosivustoja, jotka ovat helppokäyttöisiä ja jotka parantavat käyttäjäkokemusta.

Perinteisesti verkkosivut suunniteltiin kiinteään ulkoasuun, jossa käytettiin kiinteitä arvoja, rivejä ja sarakkeita. Näistä muodostui moduuleita, joihin kehittäjät pystyivät lisäämään verkkosisältöä. Kiinteiden ulkoasujen epäjoustavuus lopulta koitui ongelmalliseksi, sillä Internetiä ei enää selattu isoilla ruuduilla, esimerkiksi pöytätietokoneilla tai kannettavilla tietokoneilla, vaan käyttöön tulivat myös mobiililaitteet. Ratkaisuna tähän ongelmaan oli eritetyt verkkosivustot; puhelimen käyttäjät ohjattiin mobiilisivustolle, tablet-laitteiden käyttäjät ohjattiin tablet-sivustolle ja tietokoneen käyttäjät ohjattiin yrityksen ”oikealle” sivustolle. Tämä ratkaisu ei ole nykyään pätevä, sillä mobiililaitteillakin on erikokoisia näytön resoluutioita.

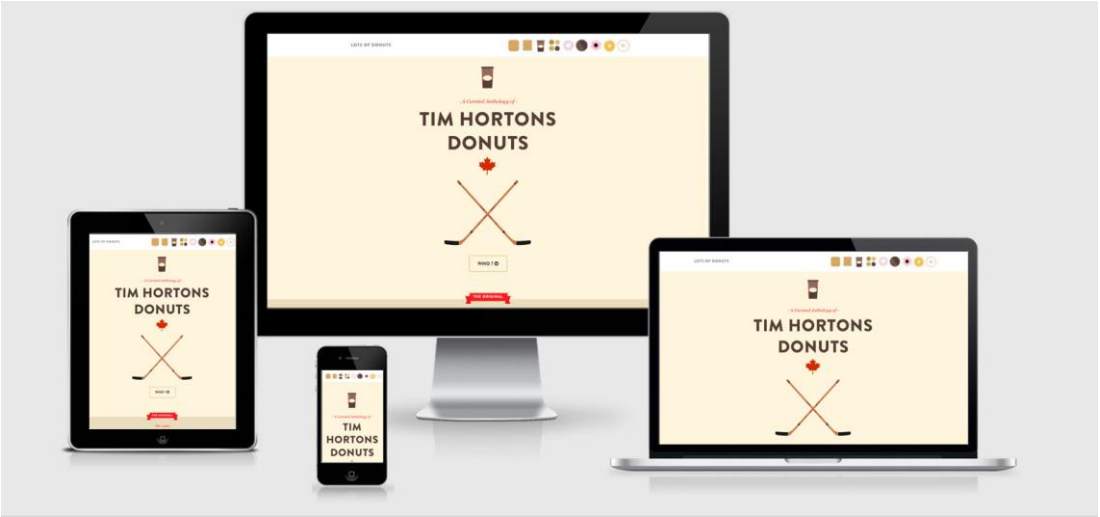
Tänä päivänä Internet-sivuston pitäisi olla joustava, toisin sanoen tasa-arvoinen riippumatta laitteen koosta tai mallista. Tätä lähestymistapaa kutsutaan responsiiviseksi web-suunnitteluksi. Tässä työssä käydään läpi, mitä responsiivisessa suunnitelmassa täytyy huomioida ja mitä kehittäjän tarvitsee tietää, kun responsiivista suunnitelmaa käytetään kehitysmallina.

2 SUUNNITELMAN SISÄLTÖ

2.1 Responsiivinen web-suunnittelu

Kun suunnittelet verkkosivua ja tavoitteena on joustavuus ja sopeutumiskykyisyys erilaisille laitteille, tätä kutsutaan responsiiviseksi web-suunnitteluksi. Tämän termin nimesi freelanceri Ethan Marcotte artikkelissa ”Responsive Web Design” vuonna 2010. RWD on nykyaikainen web-suunnitelma ja kehitysmalli. Sen tarkoituksena on Internet-sivuston optimointi, jossa keskitytään joustavuuteen, muokkautuvuuteen ja jossa annetaan käyttäjälle paras mahdollinen käyttäjäkokemus. Se parantaa sivustossa liikkumista ja sen lukemista vähentämällä sivuston vieritystä, mikä on riippumaton laitteen näytön resoluutiosta. Tämä lähestymistapa johtaa suunnitelmaan, mikä on yksinkertainen ja reagoi käyttäjän tarpeisiin. (Marcotte, 2010)

Responsiivisesta suunnitelmasta löytää suuria hyötyjä, jos sitä vertaa perinteiseen suunnittelumalliin, missä eri laitteille tehtiin erilaiset sivustot. Responsiivisen sivuston kehittäminen voi koitua ensiksi kalliiksi, mutta sillä ei tarvitse kehittää montaa eri versiota mobiilille ja muille laitteille. Vaikka responsiivisella suunnitelmalla on kallis aloittaa, niin se palkitsee tulevaisuuteen katsottuna sivuston omistajaa taloudellisesti vähentämällä ylläpidon määrää. (Marcotte, 2010)



Kuva 1. Responsiivinen sivusto erilaisilla resoluutioilla (Avery, 2016)

Kuvassa 1 näkyy responsiivinen Internet-sivu erilaisilla laitteilla. Sivu muokkautuu tietokoneelle, tableteille ja mobiililaitteille automaattisesti. Kehityksessä on yleistä, että sivuston sisältö suunnitellaan ensin pienelle laitteelle, jotta sivun suunnittelu ei kärsisi myöhemmin.

Suunnitelman ydin koostuu kolmesta eri kulmakivestä. Nämä kolme elementtiä ovat nimeltään joustava grid, joustava media ja mediakyselyt. (Marcotte, 2010)

2.2 Tuettavat laitteet

Ennen kuin aloittaa Internet-sivustoon perustuvaa projektia, täytyy suunnittelijan tai kehittäjän arvioida, millaiselle käyttäjälle sitä ollaan luomassa. Pöytätietokoneiden ja kannettavien tietokoneiden myynti on laskenut jatkuvasti, mutta ne ovat silti suuressa käytössä, sekä niiden monet erilaiset selainohjelmistot. Tietokoneisiin sisältyvät sekä pienet 11 tuuman netbookit että 28 tuuman resoluution näytöt, joilla on omat suhteet ja suunnat. Nämä vaikuttavat suuresti Internet-sisällön alueen suunnitteluun. (Ward 2017, 6)

Internet-selaamisen suosio mobiililaitteilla on kasvanut tietokoneiden ohitse, joten on hyvin tärkeää ottaa huomioon sivuston mobiilipuoli. iOS käyttöjärjestelmällä on yleisesti vain yksi selainohjelma, ja laitteistojen koko on usein johdonmukainen. Androidilla tilanne on hieman erilainen, sillä Androidilta löytyy laaja valikoima sekä selainohjelmia että näytön kokoja. (Ward 2017, 7) Kun suunnitellaan sivustoa älypuhelimelle, täytyy suunnittelijan huomioida, että selaaminen tapahtuu kosketuksella eikä näpäytyksellä ulkoisella laitteella. Tableteilla tilanne on taas eri, sillä tablettia voidaan käyttää myös hiirellä. (Ward 2017, 7)

Smart TV:t ja muut samantyyppiset laitteistot kuten Apple TV sisältävät yksinkertaistetun Internet-selaimen, jossa käyttäjät selaavat tiettyjä sivustoja. Smart TV:n selausohjelman käytön suosio on kasvanut jatkuvasti, jonka takia suunnittelussa sekin täytyy ottaa huomioon. Televisioissa on suuri näyttö, jossa on usein pieni resoluutio. Sivuston ulkoasu pitäisi olla selvä, jotta sivustoa pystytään selaamaan ja käyttämään kaukaa. (Pingdom, 2012)

2.3 Suunnitelman strategia ja prosessi

Responsiivinen suunnittelu tapahtuu oletuksena mobiili ensin - strategiana. Sen ideologiana on, että mobiilisuunnittelu tapahtuu ensimmäisenä, sillä se on suunnittelusta hankalin osuus. Kun mobiilisuunnittelun kysymyksiin on löydetty ratkaisut, suunnittelu muille laitteille helpottuu. (Gremillion, 2015)

Jos sivusto toimii hyvin mobiililaitteella, toimii se paremmin kaikilla muillakin laitteilla. Kaikista tärkeintä on se, että mobiili ensin - lähestymistapa on myös ns. sisältö ensin - lähestymistapa. Puhelimella on eniten rajoituksia sen koon takia, jonka takia se pakottaa suunnittelijan priorisoimaan sivuston sisältöä. (Gremillion, 2015) Mobiili ensin - lähestymistavan hyötynä on se, että pieniruutuiset breakpointit sopivat paremmin sisällölle. Breakpointit kehittyvät luonnollisesti sisällön ympärille, eikä suunnittelijan tarvitse tehdä jatkuvasti pieniä muutoksia. (Gremillion 2015)

Mobiili ensin - lähestymistavan prosessista löytyy huomioon otettavia vaiheita, joita ovat:

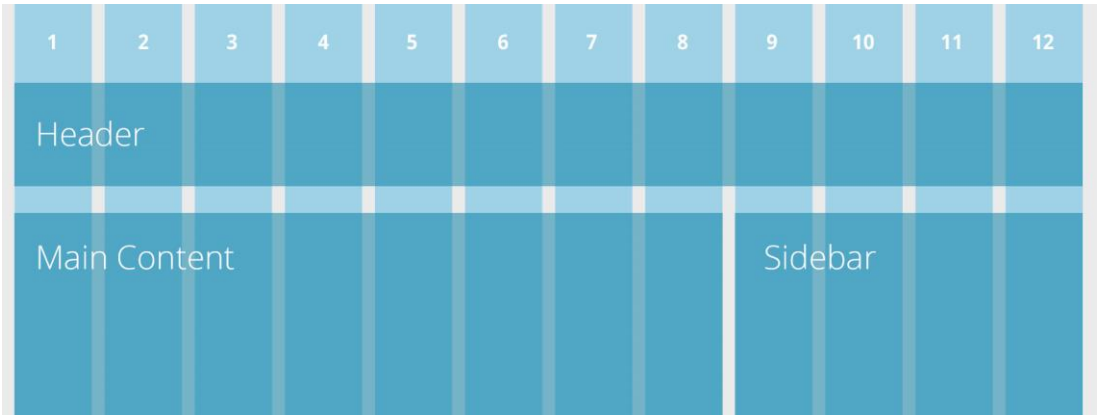
1. Sisältöluettelo (eng. content inventory). Tämä voidaan tehdä joko laskutaulukolla tai jollakin muulla vastaavalla tiedostolla, johon lisätään kaikki elementit, jotka halutaan sisällyttää sivustoon. Kuvassa 2 on esimerkki sisältöluettelosta.
2. Visuaalinen hierarkia. Elementtien priorisointi sisältöluettelossa sekä päätelmä, eli miten näytetään tärkeimmät elementit sivustolla.
3. Suunnitellaan pienimmässä breakpointissa ja skaalataan ylöspäin. Rakennetaan mobiilin rakenne ensin, sitten käytetään mobiilin rakennetta suuremmille breakpointeille.
4. Kosketusnäytön kohteet. Sormet ovat leveämmät kuin pikselintarkat hiiret, joten sivusto tarvitsee suurempia elementtejä, mihin voi painaa. Hyperlinkeille tarvitsee antaa paljon tilaa, nappeja täytyy suurentaa ja interaktiivisille elementeille täytyy antaa kaikin puolin tilaa.
5. Vältä suurta mediaa. Vaakakuvat ja monimutkaiset grafiikat eivät näy hyvin, kun näyttö on vain muutaman tuuman leveä. Tämän takia täytyy käyttää mediaa, joka on luettavaa mobiililaitteella. (Gremillion, 2015)

	A	B	C	D	E	F	G
1	**ID**	**Navigaatio**	**Sivun titteli**	**Kommentti**	**Tiedostonimi**	**Tiedostotyyppi**	
2	0.0.			CSS-tiedosto	style	.css	
3	1.0.	Etusivu	Opinnäytetyö	Hello World	Index	.html	
4	2.0.	Tietoa	Sivun tarkoitus		Tietoa	.html	
5	3.0.	Multimedia	Testissä multimedia	Joustavan kuvan ja videon testaus ja käyttöönotto	Multimedia	.html	
6	3.1.	Kuvat	Kuvatesti		Kuvat	.html	
7	3.2.	Video	Videotesti		Video	.html	
8	4.0	Ota Yhteyttä	Yhteydenotto		Yhteys	.html	

Kuva 2. Esimerkki sivuston sisältöluettelosta

2.4 Grid

Gridillä luodaan sivuston perusta. Grid koostuu seuraavista osista: muoto, marginaalit, sarakkeet ja moduulit. Gridillä luodaan näkymättömiä rivejä ja sarakkeita, joista muodostuu alueita Internet-sivun sisällön rakennetta varten. Vaikka rivit ja sarakkeet eivät ole visuaalisesti näkyvissä, ne auttavat moduulien organisoinnissa, esimerkiksi graafisissa elementeissä (tekstit, kuvat ja muut elementit). (Gregory, 2018). Kuvassa 3 nähdään, miten sivuston rakenne saadaan organisoitua gridin avulla, jossa on 12 riviä.



Kuva 3. Gridin suunnittelu (Gregory, 2018)

Ennen responsiivista suunnitelmaa kehittäjät käyttivät kiinteää gridiä web-suunnittelua varten, joka koostui kiinteistä arvoista, riveistä ja sarakkeista. Suunnittelussa käytettiin oletuksena 1024x768 kokoista resoluutiota. Tämä kiinteä elementti, joka koostui esimerkiksi pikseleistä, tuumista tai senttimetreistä, hankaloitti suunnittelua, kun kiinteä grid ei tukenut joustavaa ulkoasua. (Marcotte 2011, 23-31.)

Kiinteän ulkoasun ongelmaan löydettiin ratkaisu. Käyttämällä relatiivisia yksikköjä CSS-ominaisuuksien arvoille saadaan joustavuutta containereille ja sen child

containereille. (Marcotte 2011, 23-31.) Containerilla tarkoitetaan sivuston aluetta, johon voidaan sisällyttää sisältöä, esimerkiksi tekstiä, kuvia, ja niin edelleen. Joustavan gridin tarkoituksena on määritellä enimmäisarvo sivuston ulkoasun koolle.

Muuttuvan yksikön eli prosenttiyksikön saadaan käyttämällä tunnettua kontekstikaavaa, jossa kohteen arvo jaetaan kontekstilla. Laskukaavasta saadaan tarvittava prosenttiarvo esimerkiksi yhdelle riville, sarakkeelle tai koko containerille. (Marcotte 2011, 23.) Esimerkkinä voisi käyttää 2 sivuttaista pikseliarvoista containeria, jotka ovat kaavion kohteina. Kontekstina voi olla niiden sisällyttävä main container, wrapper tai jopa sivuston viewport.

2.5 Joustava media

Kuvat ja videot eivät muokkaudu luonnostaan sivuston ulkoasun mukaan. Median alustava rakenne pysyy samana jokaisen laitteen viewportissa. Sivuston näkyvää aluetta kutsutaan viewportiksi. Viewportilla tarkoitetaan käyttäjän visuaalista aluetta verkkosivustolla, joka vaihtelee laitteen mukaan. Medioiden alustava rakenne ei sovi jokaiselle laitteelle erilaisten resoluutioiden takia. Kuva tai video voi venyä sivu- tai pystysuuntaan tai siitä voi puuttua osia.

Näiden puutteiden yli päästään käyttämällä CSS-tiedostossa joustavaa yksikköä eli prosenttia. Media käyttää siis samoja ominaisuuksia, kuin aikaisemmassa joustavassa gridissä. Suunnittelijan täytyy ottaa huomioon se, että media ei välttämättä aina skaalaudu sivuston containerien mukaan.

Käyttöliittymän suunnittelija Richard Rutter keksi tekniikan, jolla media saadaan skaalautumaan containerin koon mukaan: käyttämällä ”max-width: 100%” attribuuttia CSS-tiedoston mediaelementissä. Attribuutin avulla media käyttää maksimissaan 100 prosenttia containerin leveydestä, eikä sitä voida ylittää. (Marcotte 2011, 45.) Samaa attribuuttia käytetään esimerkiksi videosoittimessa, kun se suurennetaan koko näytölle.

2.6 Mediakyselyt

Kolmas elementti, joka kuuluu responsiivisen suunnitelmaan, on CSS:n mediakyselyt, sillä sivusto jää keskeneräiseksi ilman sitä. Sivusto ei toimi pelkkien prosenttiyksiköiden kanssa. Jos tässä vaiheessa sivustoa testattaisiin älypuhelimella, niin ulkoasu rutistuisi kasaan. (Marcotte 2011, 64.)

Suurella resoluutiolla sivusto toimisi hyvin käyttämällä leveytenä 70 prosenttia, mutta pienellä laitteella kuten älypuhelimella ulkoasu tiivistyy liian pieneksi. Älypuhelimen viewport on liian pieni, mikä tekee sisällön luomisesta ja lukemisesta melkein mahdotonta. Mediakyselyn tarkoituksena on käyttää breakpointeja, jotta saadaan älypuhelimelle 100 prosentin leveys niin, että pöytätietokoneen prosenttiarvot pysyvät 70 prosentissa.

Breakpointilla viitataan tietyn laitteen viewportiin, jolla saadaan luokiteltua ominaisuuksia. Breakpointin avulla saadaan luotua sivun ulkoasuun ominaisuuksia, jotka ovat esimerkiksi käytössä vain tietyssä resoluutiossa (Bootstrap.)

W3C julkaisi mediakyselyt osaksi CSS3:n standardiin vuonna 2012. Tämä ratkaisi ongelman, jossa kehittäjä ei voinut luoda eri viewporteille omaa tyylitystä. Mediakyselyt mahdollistivat responsiivisen sivuston luomisen. (W3C 2012.)

Media Query (Mediakysely) on CSS3 moduuli, joka mahdollistaa sivuston automaattisen muokkautuvuuden erilaisilla tyyleillä ja ominaisuuksilla. Sillä voidaan esimerkiksi yksilöllistää laitteiden viewporteja, laitteiden suuntaa (pysty- ja sivuasento), jne. Mediakysely koostuu erilaisista mediatyypeistä, johon kuuluu all, aural, braille, handheld, print, projection, screen, speech, tty (terminaalit, kannettavat laitteet) ja tv. Mediatyyppejä käyttämällä laitteita saadaan kategorisoitua luokkiin, mikä mahdollistaa ulkoasujen rakentamisen mediakohtaisesti. Jokaisella mediatyypillä on omat ehtolausekkeet, jotka tarkistelevat ominaisuuksia, esimerkiksi leveyttä ja pituutta, laitteen leveyttä ja pituutta, resoluutiota, gridiä tai jopa väriä. Mediakysely valitsee mediatyypin totuusfunktion avulla. Jos kyselyn ominaisuuksien arvot sopivat tiettyyn tyyppiin, funktio palauttaa arvon tosi ja mediatyyppi otetaan käyttöön. Jos funktio palauttaa

arvon epätosi, mediatyyppi sivuutetaan. (W3C, 2012.) Taulukossa 1 on esimerkki eritetyistä breakpointeista.

Taulukko 1. Resoluutioiden breakpointit (Marcotte 2011, 114.)

320 pikseliä	Pienille laitteille kuten puhelimille (pystyasennossa).
480 pikseliä	Pienille laitteille kuten puhelimille (sivuasennossa).
600 pikseliä	Pienet tabletit (600x800, 600x1024, pystyasennossa).
768 pikseliä	Kymmenen tuuman tabletit kuten iPad (768x1024, pystyasennossa).
1024 pikseliä	Tabletit kuten iPad (sivuasennossa), tietyt kannettavat tietokoneet ja pöytätietokoneet.
1200 pikseliä	Kannettavat tietokoneet ja pöytätietokoneet.

Kuvassa 4 nähdään, että mediatyypiksi on valittu screen. Yllä olevasta taulukosta on valittu breakpointit sen ehdoiksi. Annetulla resoluution leveydellä (320-480) ja mediatyypillä voidaan määritellä tämän kokoisille laitteille oma tyylitys.

```
1  @media screen and (min-width: 320px) and (max-width: 480px)
2  {
3      //Tyylitys
4  }
```

Kuva 4. Mediakyselyn ehdot älypuhelimelle.

Kun selaimen leveys on sama kuin mediakyselyn annetulla ehdolla, eli näytön minimileveys on 320 pikseliä ja maksimileveys on 480 pikseliä, niin tyylitys otetaan käyttöön. Kuvan 7 malli on suunniteltu älypuhelimille, ja siinä on otettu huomioon sen pysty- ja sivuasento. Tällaisten mediakyselyjen avulla saadaan suunniteltua Internet-sivusto, joka muokkautuu automaattisesti käyttäjän tarpeen mukaan.

2.7 Navigaatio

Internet-sivun navigaation pitäisi toimia kuin kompassi: se auttaa uusia käyttäjiä perehtymään sivuston järjestelmässä, ja auttaa heidät haluamaansa määränpäähän. Nykyään sivustoilla on kaiken tasoisia ja tyyppisiä valikkoja, joka voi tehdä navigoinnista

haastavaa varsinkin, jos navigoinnin täytyy olla responsiivista. Miten kompleksinen valikko muokkautuu pienelle näytölle? Mitä jos halutaan näyttää enemmän tai vähemmän tietoa riippuen näytön mittasuhteista? Responsiivisen navigointijärjestelmän ei tarvitse näyttää samalta jokaisessa breakpointissa, mutta sen täytyy tarjota saman sisällön jokaisella laitteella. (Marcotte 2015, 2)

2.7.1 Show/hide toggle

Show/hide toggle on yksi yleisimmistä tavoista, miten voidaan käsitellä komplekseja valikkoja responsiivisessa suunnittelussa: kun valikko ei mahdu sivustolle, piilota se. Tämä patterni tarvitsee minimissään kaksi elementtiä: navigaation mikä piilotetaan tietyssä breakpointissa sekä ”trigger”-elementin, joka on käyttäjän käytössä, jolla saadaan navigaatio näkyville. (Marcotte 2015, 2)

Show/hide toggle käyttää JavaScriptiä, joka kyselee käyttäjän selaimelta, että tukeeko se JavaScriptin DOM-ominaisuuksia. Näitä ovat muun muassa document.querySelector, window.addEventListener ja window.getComputedStyle. Jos ominaisuuksia ei löydy, se palauttaa arvon return, joka estää selaimen aloittamasta muuta JavaScriptiä. Tuloksena on se, että vanhat selaimet saavat käyttöönsä toimivan, muttei täydellisen, käyttäjäkokemuksen. (Marcotte 2015, 2)

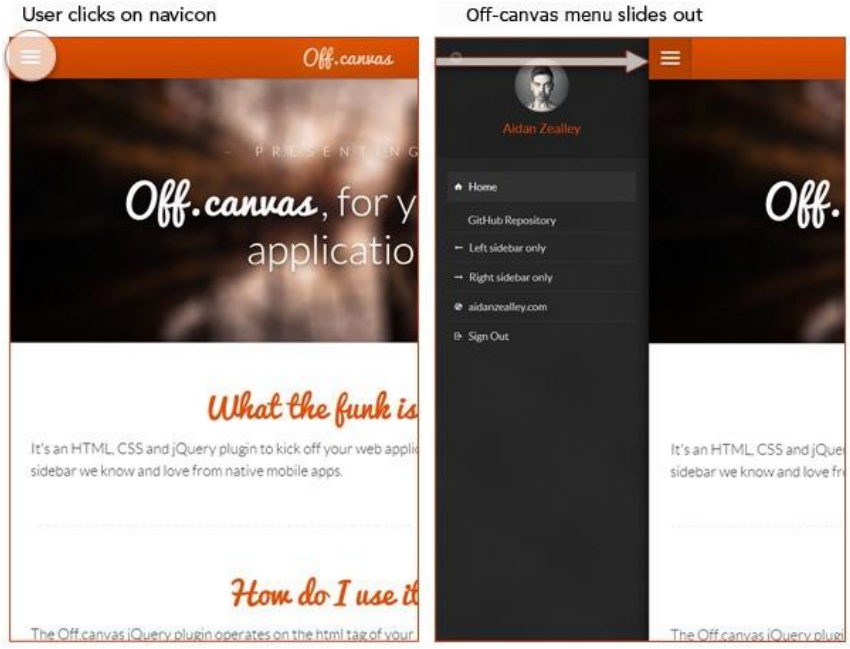
Show/hide toggle toimii erinomaisesti responsiivisella sivustolla, mutta se ei tarkoita sitä, että se on sopiva jokaiselle breakpointille. Kytkin on ainoastaan sopiva pienille viewporteille, joissa ulkoasu on kiinteämpi. Kun viewport levenee, niin koko navigaatio näkyy lukittuna esimerkiksi logossa. Ongelman ratkaisemiseksi käytetään CSS:n mediakyselyä, jossa voidaan ohittaa tietty breakpoint. Kuvassa 5 on esimerkki tällaisesta tilanteesta. Leveälle viewportille saadaan näkyville koko navigaatio ja show/hide toggle pystytään piilottamaan. (Marcotte 2015, 2)

```
@media screen and (min-width: 39em) {
  .page .nav ul {
    overflow: auto;
    max-height: inherit;
  }
}
```

Kuva 5. Breakpointin ohitus (Marcotte, 2015)

2.7.2 Off-canvas menu

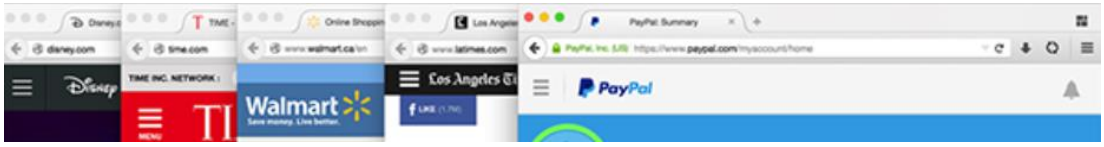
Off-canvas menu on muunnos show/hide togglesta, joka sai alkunsa mobiilisovelluksissa ja joka lopulta levisi responsiivisille verkkosivustoille. Mekaanisesti katsottuna off-canvas menu ei ole erilainen show/hide togglesta. Kummassakin piilotetaan navigointi ja pyydetään käyttäjää käyttämään elementtiä, josta kytketään valikko päälle. Jos suunnittelija osaa hyödyntää off-canvas menua, niin sillä saa ylimääräistä syvällisyyttä ja ulottuvuutta valikkoon. Tosin valikko vaatii myös ylimääräistä huolellisuutta, jotta valikossa liikkuminen pysyy helppokäyttöisenä ja jotta se ei riko yhteensopivuuksia selainohjelmistojen kanssa. (Marcotte 2015, 2) Kuvassa 5 on esimerkki off-canvas menun käytöstä.



Kuva 5. Off-canvas menu (WebFX.)

2.7.3 Hamburger

Responsiivisten sivustojen navigointi usein erottuu toisistaan erilaisilla malleilla, mutta niillä on usein yleinen elementti käytössä, joka näkyy melkein jokaisella sivustolla: ikoni, jossa on 3 pinottua vaakaa viivaa, kutsutaan nimellä ”Hamburger” (Marcotte 2015, 2). Kuvassa 6 on esimerkki Hamburger-valikosta suosituilla sivustoilla.



Kuva 6. Hamburger-valikko (Marcotte, 2015)

Ikonia käytettiin ennen merkityksellisiin valikkoihin, mutta nykyään sitä käytetään kytkimenä, jolla aukaistaan sivuston koko navigaatio, etenkin pienillä resoluutioilla. Valikon hyötynä on, että se on todella tiivis, joka on myös helposti luettava pienilläkin laitteilla. Hamburger on myös helppo sisällyttää sivuston ulkoasuun, joka on riippumaton siitä, onko sivustolla käytössä SVG, jokin CSS-pohjainen animaatio tai minimaalinen HTML kokonaisuus. (Marcotte 2015, 2)

3 WEB-TEKNOLOGIAT

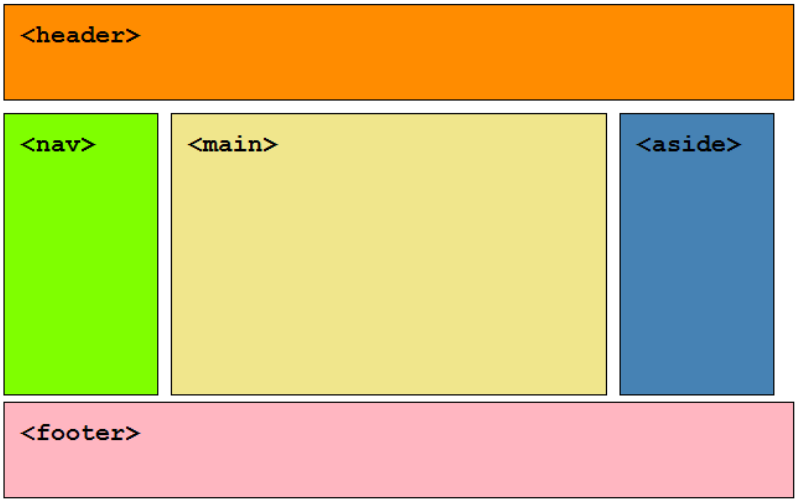
Hyper Text Mark-up Language 5 (HTML5), Cascading Style Sheets (CSS) ja JavaScript (JS) ovat responsiivisen kehityksen pääaiheiset teknologiat. Internet-sivustot koostuvat kolmesta eri komponentista, jotka ovat rakenne, visuaalisuus ja toiminnot. HTML:n avulla suunnitellaan sisällön paikkaa, CSS:n avulla saadaan visuaalisuutta ja JavaScrptillä lisätään dynaamista toiminnallisuutta.

3.1 HTML5

HTML5 on merkintäkieli, jota käytetään web-sivuston rakenteeseen ja sisällön esittämiseen. W3C julkaisi HTML:n viidennen version vuonna 2014. Version ideana on tukea nykypäivän multimediaa sekä helpottaa elementtien lukemista. (W3C 2017.) W3C lisäsi uusia elementtejä multimedian ja grafiikan esittämistä varten. Näitä ovat muun muassa <audio>, <video> ja <canvas>. Näillä elementeillä pystytään käyttämään ääniä, videota ja muuttuvaa grafiikkaa web-sivustolla, eikä suunnittelijan tarvitse käyttää ulkoisia liitännäisiä tai sovelluksia. Multimedian lisäksi HTML5 sallii MathML ja SVG-vektorigrafiikan elementtien linkittämisen html-tiedostoon. (W3C 2017.)

HTML5 käytetään Flashin korvaavuutena, mutta tähän tarvitaan myös CSS ja JavaScriptiä. Adobe ilmoitti vuonna 2017, että he lopettavat Flashin tukemisen vuoden 2020 lopussa ja suosittelevat sisällön luojia siirtymään muihin formaatteihin, joita ovat mm. HTML5, WeBGL ja WebAssembly. (Adobe 2017.)

<article>, <section>, <header>, <footer>, <aside> ja <figure>-elementtejä käytetään itsekuvaavan html-tiedoston kehittämiseen. Nämä elementit auttavat web-sivuston sisällön organisoinnissa, helpottavat hakukoneiden navigaatiota sekä device-to-device kommunikaatiota. Kuvassa 6 esimerkki organisoinnista. (W3C 2017.)



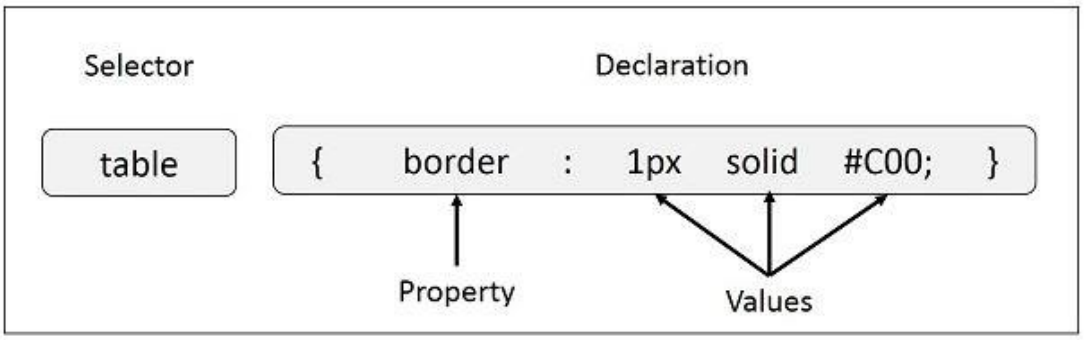
Kuva 7. Sivun rakenteen organisointi HTML5:n avulla (Faulkner, S. 2015)

Kaikki nykyaikaiset verkkoselaimet, älypuhelimet, tabletit ja muut moninäyttöiset laitteet tukevat HTML5. HTML5:n tarkoituksena on antaa kehittäjälle mahdollisuus luoda ainutlaatuisia käyttäjäkokemuksia, mahdollistaa vaativien käyttöliittymien kehittämisen ja tukea monia eri ohjelmistorajapintoja.

3.2 CSS

CSS on sivuston tyyliohje, jonka tarkoituksena on esittää Internet-sivustolle esimerkiksi ulkoasua, värejä tai fontteja. Se on Internet-sivustojen ytimeen kuuluva teknologia, jolla saadaan rakennettua auditiivisen ja visuaalisen ulkoasun web-sivustoille. Ulkoasun suunnittelussa voidaan luoda yhteensopivuutta erilaisten laitteiden kanssa. CSS voidaan käyttää jokaiseen XML:n perustuvaan merkintäkieleen, se ei siis ole riippuvainen HTML-kielestä. (Tutorialspoint.) CSS-tyylitiedostoa voidaan käyttää kolmella eri tavalla:

- tyylitiedosto linkitetään ulkoisesti
- tyylitiedoston sisältö lisätään merkintäkielen tiedoston sisään tai
- tyylitiedoston syntaksia lisätään suoraan merkintäkielen elementteihin.



Kuva 7. CSS syntaksi (Tutorialspoint).

CSS tyylit koostuvat listasta, jossa jokaiseen tyyliin sisältyy valitsin (engl. selector) ja deklaraatio (engl. declaration). Valitsijat ovat HTML:n elementtejä ja deklaraatioissa ovat CSS-tyylin attribuutteja eli ominaisuuksia (engl. Property). Kuvassa 7 näkyy valitsimen paikalla <table>-elementti, joka sijaitsee html-tiedoston sisällä. Käyttämällä CSS-attribuuttia border ja sille annettuja arvoja (engl. values), saadaan <table>-elementille yhden pikselin paksuinen reunus, joka väritetään mustaksi. Valitsimelle voidaan antaa enemmän kuin yksi deklaraatio. Suunnittelija voi esimerkiksi muuttaa <table>-elementin sisällä olevan tekstifontin kokoa font-size-attribuutin avulla ja samalla muuttaa taulun alueen väriä.

3.3 JavaScript

JavaScript on dynaaminen olio- ja funktio-ohjelmointia tukeva ohjelmointikieli, jota voidaan sisällyttää web-sivuille toiminnallisuuden lisäämiseksi. Sen syntaksi perustuu Java- ja C-kielen rakenteeseen. JavaScriptin avulla voidaan luoda vuorovaikutteisia ominaisuuksia esimerkiksi valikoita, jotka muokkautuvat sivuston ulkoasun mukaan. (Mozilla 2018.)

Suurin osa JavaScriptin komennoista ovat event handlereita, mikä tarkoittaa sitä, että niitä pystytään lisäämään suoraan HTML-elementteihin. Komennoilla voidaan manipuloida HTML:n sisältöä ja CSS-tyylitiedoston valitsimia ja attribuutteja. (Mozilla 2018.) Näin pystytään vaikuttamaan sivuston sisältöön eri viewporteissa. Tarkoituksena on priorisoida sisältöä viewportin kanssa niin, että tärkeä tieto nähdään ensimmäisenä (esim. uudemmat uutiset ennen vanhempia uutisia, jne.).

JavaScriptiä käytetään mediakyselyjen kanssa myös siksi, että saadaan korjattua mahdolliset yhteensopivuusongelmat verkkoselainten kanssa. JavaScriptillä pystyy siis paikkaamaan mediakyselyn arvoja, jos käytössä olisi esimerkiksi vanhempi versio Internet Explorerista tai Firefoxista. Internet on pullollaan valmiita JavaScript-kirjastoja, joilla voi helpottaa JavaScriptin kirjoittamista. jQuery-kirjasto on hyvä esimerkki, jolla saadaan helppokäyttöinen ohjelmointirajapinta Internet-sivustolle.

```
1 if (document.documentElement.clientWidth < 900) {
2     //priorisoidaan sivuston sisällön paikkaa skriptillä//
3 }
```

Kuva 8. Esimerkki sisällön manipuloinnista JavaScriptillä.

Kuvan 8 ehtolause (if) on sama kuin mediakyselyn max-width. Jos laitteen näyttö on leveämpi kuin 900 pikseliä, niin skripti ei käynnisty.

4 TOIMINNALLINEN OSA: RESPONSIIVINEN DEMO

4.1 Projektin määrittely

Projektin tavoitteena on luoda responsiivinen verkkosivusto mobiililaitteille sekä pöytätietokoneille, joka sopii esimerkiksi portfoliosivustoksi tai yrityssivustoksi. Tavoitteena on rakentaa joustava ulkoasu, joka sopeutuu moneen eri laitteeseen. Projektiin käytettiin Pixabayn tekijänoikeusvapaata materiaalia. Sivusto kehitettiin Visual Studio Code -ohjelmistolla, jota testattiin Google Chrome, Mozilla Firefox ja Internet Explorer -verkkoselaimella. Mobiilin käyttöä simuloitiin ja testattiin verkkoselaimella.

4.2 Käytetyt kehitystyökalut

Projektin kehityksessä käytettiin Microsoftin kehittämää Visual Studio Code -ohjelmistoa. Visual Studio Code on avoimen lähdekoodin tekstieditori, joka tukee Windows, Linux sekä macOS käyttöjärjestelmiä. Ohjelmassa on sisäänrakennettu tuki monelle eri ohjelmistokielelle sekä laaja valikoima laajennuksia, kuten esimerkiksi C#, C++ ja Pythonille. Näistä laajennuksista valitsin käyttöönotettavaksi Live Server -laajennuksen, jonka avulla saadaan suora näkymä sivustoon tapahtumista muutoksista. Sivuston testausta varten otin käyttöön Debugger for Chrome -laajennuksen.

Sivuston versionhallintaan käytettiin Linus Torvaldsin kehittämää Git -versiohallintajärjestelmää, joka on valmiiksi sisäänrakennettu Visual Studio Code -ohjelmaan. Sivuston rakenteen suunnitteluun käytettiin Moqups -web-ohjelmistoa. Ohjelmalla voidaan visuaalisesti suunnitella malleja käyttöliittymistä, diagrammeista ja prototyypeistä.

4.3 Käyttöliittymän suunnittelu

Ennen sivuston toteutusta, hahmottelin sivuston käyttöliittymän Moqups -ohjelmistolla. Aloitin suunnittelun ensin mobiiliversiolla ja sen jälkeen siirryin laajempaan

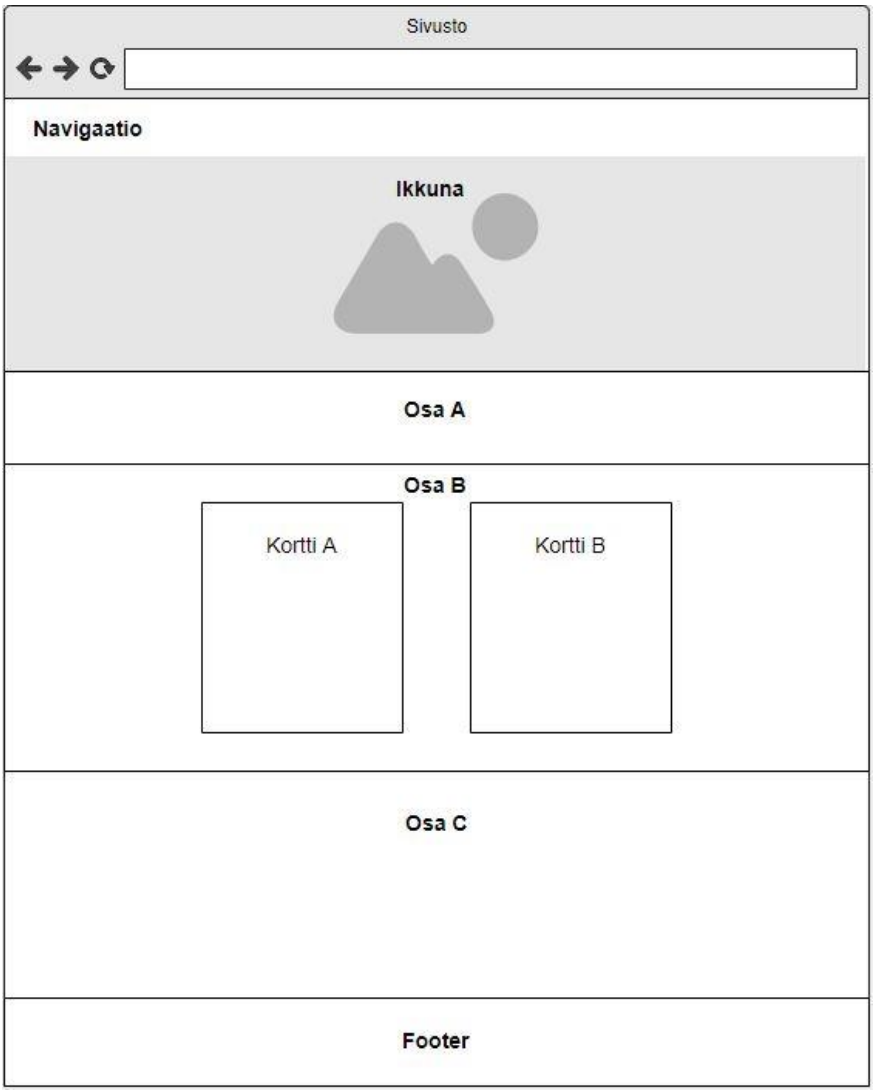
käyttöliittymään. Halusin toteuttaa mobiilin käyttöliittymän vertikaalisena ja laajemman käyttöliittymän horisontaalisena.

Sivusto koostuu dynaamisista osista, joiden paikkoja voidaan tarpeen tullen helposti vaihtaa. Sivuston navigaatio pysyy absoluuttisena sivuston yläreunassa, joka seuraa käyttäjää hänen selatessaan sivustoa. Koska sivusto on navigaatioltaan pieni, navigaatioksi valittiin Hamburger-valikko. Ikkunan tarkoituksena on ottaa vastaan tuleva vierailija ja ilmoittaa hänelle sivustosta, että mistä on kysymys. Osa A täytetään tekstillä ja Osa B koostuu korteista (engl. cards), johon lisätään responsiivisia kuvia ja tekstiä. Osa C koostuu kehittäjän yhteystiedoista. Annetaan vierailijalle mahdollisuus lukea myös lisätietoja. Sivuston loppuun lisätään alatunniste.

Kuva 9. Luonnos mobiilin ulkoasusta

Mobiilin ulkoasu pidetään mahdollisimman vertikaalisena. Kuvassa 9 nähdään, että sivuston sisältö keskitetään, mutta osat käyttävät 100% ulkoasun leveydestä. Osien pituus skaalautuu osien sisällön mukaan. Sisältöön käytetään suhteellisia yksikköjä, jotta teksti skaalautuu resoluution koon mukaan ja luettavuus pysyy hyvänä. Samaa

ulkoasua käytetään mobiililaitteen vaakasuuntaisessa ja pystysuuntaisessa käytössä. Jos tarve vaatii, niin vaakasuuntaiselle ulkoasulle voidaan suunnitella omanlaiset asetuksensa CSS:n mediakyselyillä. Mobiilin ulkoasu on käytössä, jos laitteen leveys on alle 700 pikseliä.

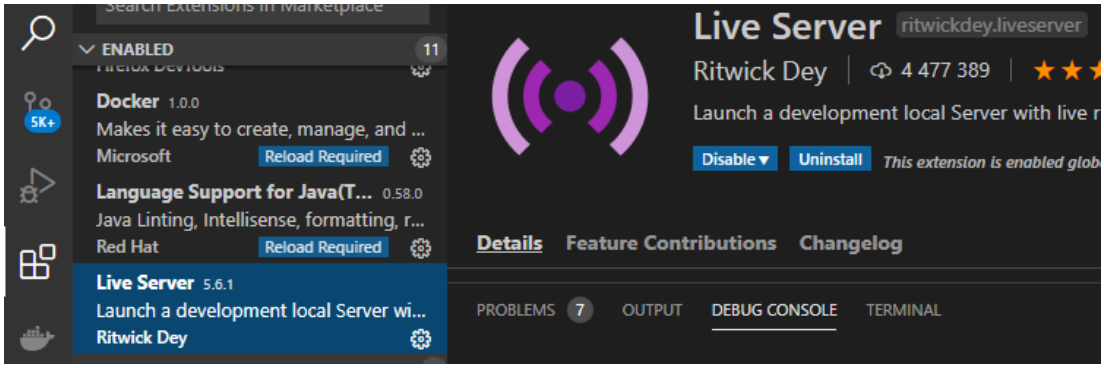


Kuva 10. Luonnos leveästä ulkoasusta

Leveä ulkoasu suunniteltiin niin, että sitä voidaan käyttää sekä tabletilla että tietokoneilla (kannettavat tietokoneet ja laajemmat näytöt). Sisällön paikka on periaatteessa keskitetty samalla tavalla kuin mobiilin ulkoasussa, mutta niin kuin kuvassa 10 nähdään, sekä osat että sisältö pyritään sijoittamaan horisontaalisesti pareiksi, jotta tyhjää tilaa jäisi vähemmän. Rakenteeseen käytetään CSS:n grid tekniikkaa. Navigaatio toimii samalla tavalla kuin mobiilin rakenteessa. Mahdollisuutena on, että hamburger valikko voidaan halutessa hävittää mediakyselyllä ja tilalle lisätään tekstilinkit.

4.3.1 Projektin alustus

Aloitin projektin lataamalla Visual Studio Code -ohjelmistoon haluamani laajennukset. Laajennusten lataus tapahtui Visual Studion markkinassa, joka on käytettävissä Visual Studio -ohjelmistoissa sekä verkkoselaimessa.



Kuva 11. Visual Studion laajennuksen asennus

Seuraavaksi loin kansion projektia varten, johon lisäsin HTML-tiedoston ’index’, CSS-tiedoston ’style’ sekä JavaScript-tiedoston ’app’. Avatessani index tiedoston lisäsin siihen heti vaadittavat elementit. Jotta verkkoselaimet tunnistavat HTML-tiedoston, sille täytyy antaa <html> elementti. Seuraavaksi lisäsin <head> ja <meta>-elementit, joihin sisällytetään sivuston metadataa. <Meta>-elementissä asetin sivuston viewportin asetukset. Asetuksen avulla tiedosto osaa dynaamisesti asettaa sivuston oletusleveyden seuraamalla käytetyn laitteen leveyttä. <Head>-elementissä yhdistetään style tiedosto index tiedostoon. Lopuksi lisäsin index tiedostoon <body>-elementin, jonka sisälle sivusto rakennetaan. JavaScript tiedosto app:n yhdistäminen tapahtuu <body>-elementissä.

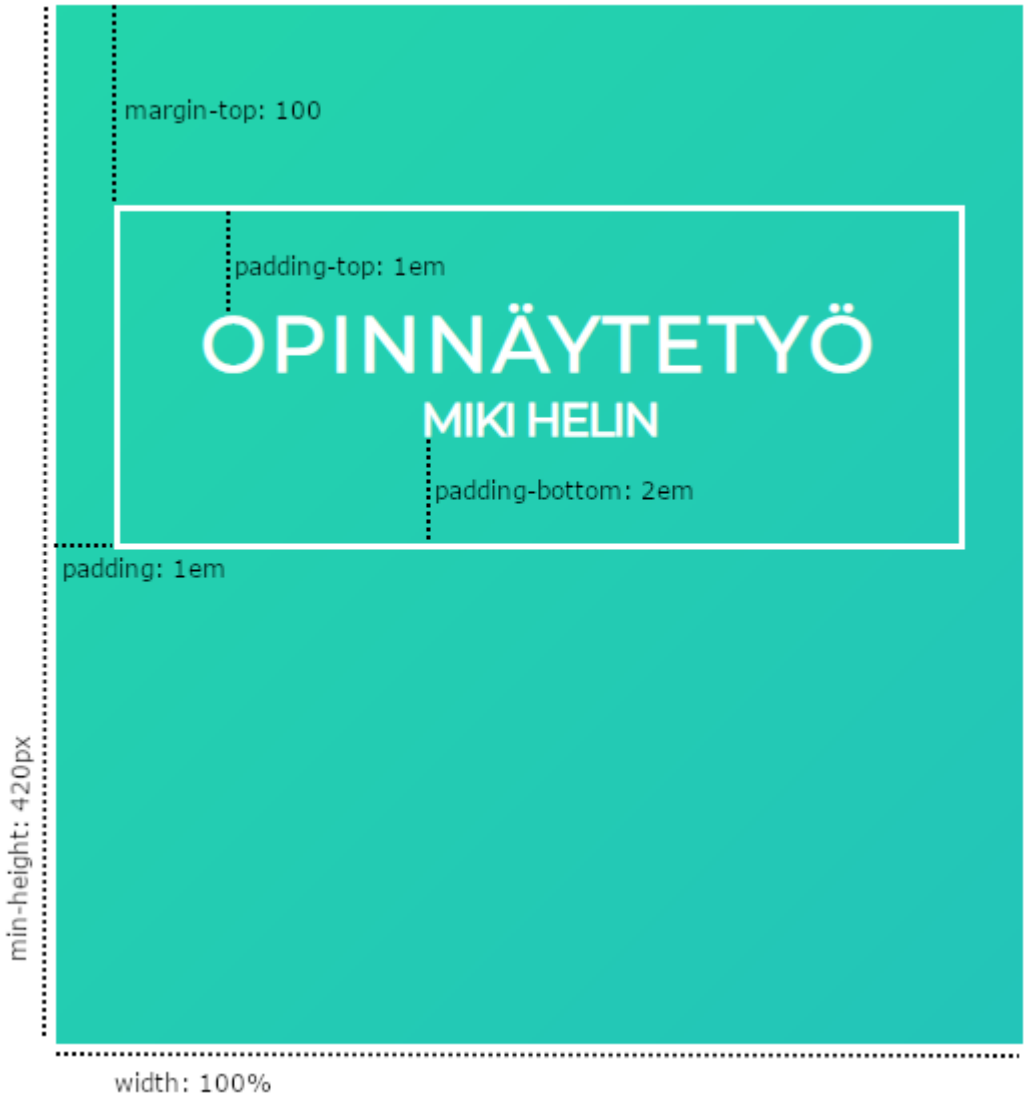
```
<html>
    <head>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <script src="app.js">
        </script>
    </body>
</html>
```

Kuva 12. Index tiedoston perusrakenne

4.3.2 Ikkuna

Aloitin ikkunan rakentamisen antamalla HTML elementille <header> ID:ksi ”ikkuna”, jolla sain asetettua koko alueelle pituuden sekä leveyden tyyliohjeessa. Asetin samalla tekstin keskityksen. Tämän jälkeen loin -elementille luokan ”sisalto-wrapper”, jonka avulla pystyn manipuloimaan sisällön paikkaa ikkunan alueella. Halusin sisalto-wrapperin alueen irti viewportin sivuista. Käytin alueen siirtelyssä apuna ikkunan titteliä. Testauksen jälkeen annoin sisalto-wrapperille ylämarginaaliarvoksi 100, ja sivuttaiseksi täytteeksi 1 em (Suhteellinen mittayksikkö). Marginaalilla annetaan tarpeeksi tilaa sivuston navigaatiolle.

Lisäsin sisalto-wrapper elementin sisälle <h1> ja <h3>-elementit, joissa ovat sivuston titteli sekä etu- ja sukunimeni. Päätin myöhemmin käyttää kyseisiä elementtejä sivuston logona, joten lisäsin sisalto-wrapper luokan sisään uuden luokan ”rajaus”, johon siirsin <h1> ja <h3>-elementit. Annoin rajaukselle tyyliohjeessa 3 pikselin paksuuden. Halusin suurentaa sivuston tekstiä enemmän, joten annoin tyyliohjeessa <body>-elementille ominaisuuden font-size, jonka arvoksi annoin 1,2 em. Tämä asetus tulee käyttöön koko sivuston rakenteessa, ellei sitä toisin aseteta yksittäisessä luokassa tai ID:ssä. Lopuksi säädin elementtien <h1> ja <h3> täytteitä tyyliohjeessa, jottei rajaus olisi kiinni tekstissä, ja laajensin <h1>-elementin kirjainväliä.

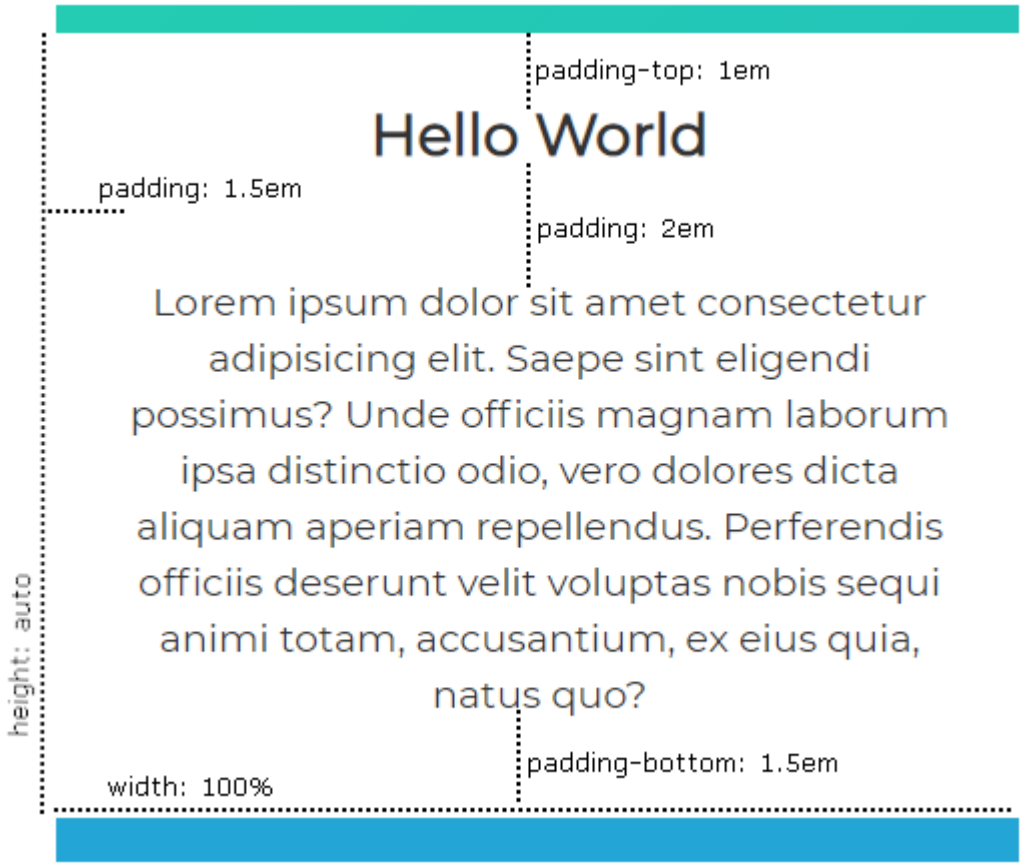


Kuva 13. Ikkunan asettelu

4.3.3 Osiointi

Aloitin ikkunan jälkeen osien luonnin asettamalla HTML-elementin <main> sisälle 3 kappaletta <section>-elementtejä, joille annoin ID:t ”osa-a”, ”osa-b” ja ”osa-c”. Osa A on yksinkertainen alue tekstille, jossa käytin hyödykseni ikkunassa käytettyä sisalto-wrapper luokkaa, jonka avulla saan keskitettyä tekstiä pois viewportin sivuista sekä muiden osien lähettyviltä. Yksi tavoitteistani oli pitää sivuston teksti mahdollisimman selkeänä ja helposti luettavissa. Jotta teksti olisi luettavampaa, lisäsin sivuston <body>-elementille ominaisuuden line-height, jonka arvoksi annoin 1,5. Ominaisuudella saadaan säädettyä koko sivuston tekstin riviväliä. Samalla annoin <p> kappale-

elementin ylätäytteelle arvoksi 1 em ja sivutäytteiksi 0. Kappale-elementti ei tarvitse omaa sivuttaista täytettä, sillä käytössä on jo sisalto-wrapper luokka. Rakenteeltaan osa A saa pituutensa sisällön mukaan automaattisesti ja leveys seuraa <body>-elementin 100 prosentin leveyttä.



Kuva 14. Osa A:n asettelu

Osa B:n kortin suunnittelussa halusin välttää ylimääräisiä -elementtejä. Websuunnittelussa yleinen ongelma on liiallinen pesiytyminen (engl. nesting) elementtien kanssa, joka tekee rakenteesta turhan monimutkaisen. Käytin korttien rakenteeseen HTML:n valmiita elementtejä <ul> ja <li>, joilla saadaan listattua alueita tai sisältöä sivustolle.

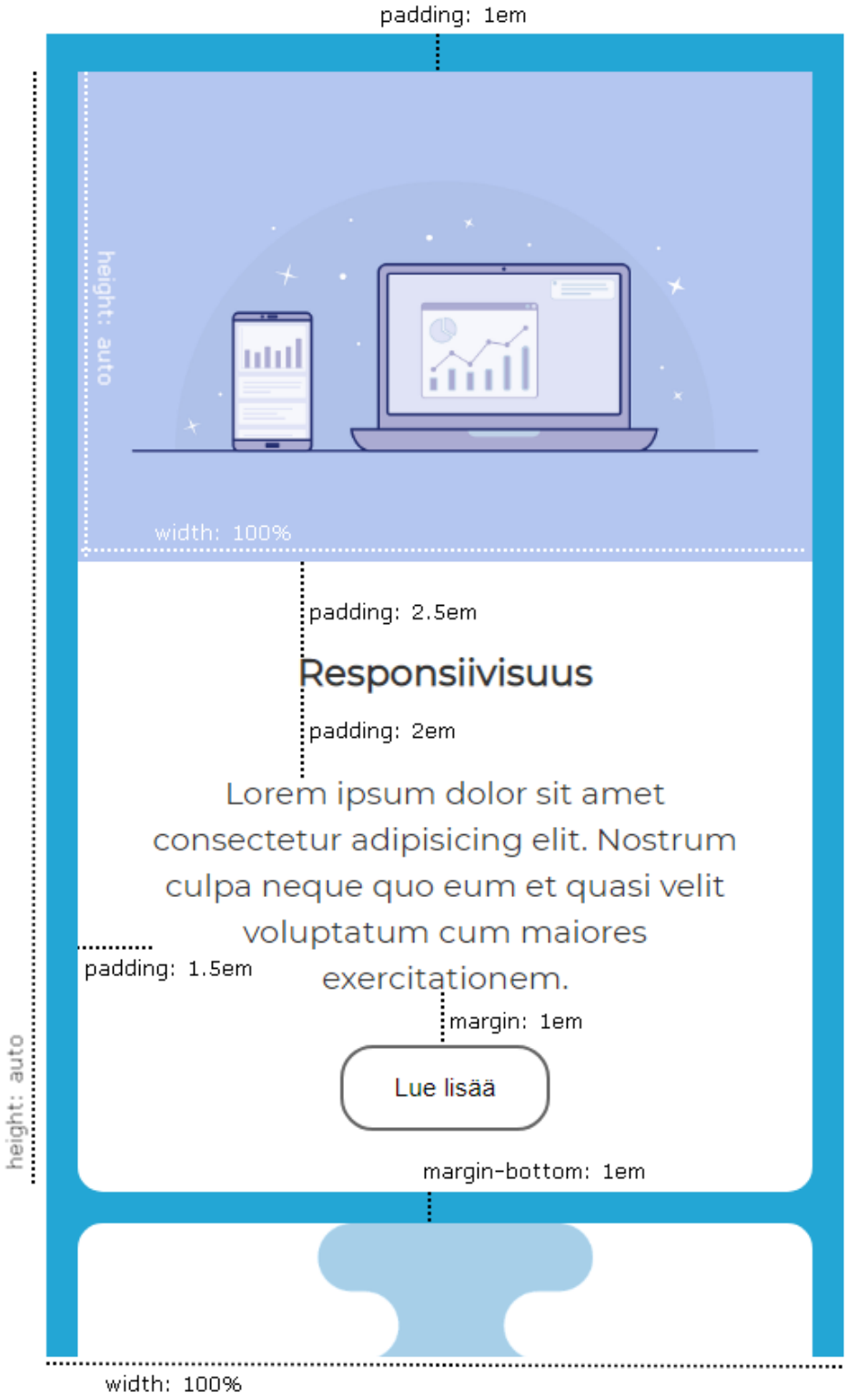
Aloitin listan rakentamisen lisäämällä HTML-tiedoston osa-b alueeseen -elementin ja pesitin sille 2 kappaletta -elementtiä kortteja varten. -elementin tarkoituksena on listata nämä kaksi korttia yhdessä. Näiden kahden -elementin sisään lisäsin uudet -elementit, joille kummallekin annoin luokan ”kortti-sisalto”.

Listan testauksen jälkeen hävitin -elementin viewportista antamalla sille list-style ominaisuuden, jonka arvoksi annoin none ja annoin marginaalin sekä täytteen arvoksi 0.

Lisäsin kummallekin -elementille oman kuvan <img>-elementillä ja oman tekstinsä <p>-elementillä. Jotta korttien kuvat ei riko sivuston rakennetta, lisäsin <img>-elementin ominaisuuksiin maksimileveydeksi 100% ja pituuden automaattiseksi. Tällä asetuksella vältetään vääristyneitä kuvia: kuva seuraa kortin leveyttä ja kuvan pituus seuraa kuvan leveyttä.

Siirryin takaisin listan rakenteeseen, sillä halusin erottaa listat toisista osista ja viewportin reunoista. Korttien irrottaminen viewportin sivuista onnistui lisäämällä osa-b alueeseen ominaisuuden, joka asettaa alueen jokaiselle puolelle 1 em:n täytteen. Irrotin kortit toisistaan asettamalla -elementille 1 em:n alamarginaalin. Kortin sisällölle annoin täytteeksi 1,5 em.

Lisäsin kummankin kortin loppuun napit ja varmistin, että napit ovat tarpeeksi suuret sormen kosketukselle ja että nappien läheisyydessä ei ole muuta sisältöä. Muutin napin ulkonäköä sivuston tyyliohjeessa ja annoin sille 1 em:n ylä- ja alamarginaalin. Napin kokoa sain säädettyä napin tyyliohjeessa täytteen avulla.



Kuva 15. Osa B. Kortin asettelu.

Osa C on sivuston pääte, jossa annetaan vierailijalle mahdollisuus lukea lisätietoa sivustosta. Tämän vierelle halusin lisätä sivuston omistajan yhteystiedot. Aloitin osan rakentamisen lisäämällä HTML-tiedoston osa-c alueeseen 2 kappaletta -

elementtiä. Luokittelin nämä elementit samaan ”info” luokkaan. Info luokan tyyliohjeessa annoin jokaisen puolen täytteeksi 2 em ja sulautin luokan taustavärillään kiinni osa B:n alueeseen. Käytin hyödykseni tekemääni nappia ja lisäsin sen lisätieto -elementtiin.

4.3.4 Navigointi

Valitsin sivuston navigaatioksi hamburger-valikon. Halusin tehdä valikon itse, enkä halunnut käyttää valmiita CSS-kirjastoja kuten Bootstrapia apuna. Aloitin valikon rakentamisen HTML-tiedostossa, missä loin sivuston rakenteen eteen uuden -elementin. Annoin tälle elementille luokan ”valikko-wrapper”. Valikko-wrapperin sisälle pesitin -elementin ”valikko” ja tälle 3 kappaletta -elementtejä, joista tulee valikon 3 linjaa. Annoin jokaiselle kolmelle elementille omat ID:t ja luokittelin ne samaan luokkaan ”linja”. Lisäsin samalla navigointipalkkia varten <ul> ja -elementeillä hyperlinkkilistauksen valikko-wrapperin alle. Tyyliohjeessa muutin valikko-wrapperin sijainnin kiinteäksi. Kiinteällä sijainnilla navigaatio seuraa vierailijaa, kun hän siirtyy sivustolla ylöspäin tai alaspäin.

```
<div id="valikko-wrapper">
    <div id="valikko">
        <div id="linja1" class="linja"></div>
        <div id="linja2" class="linja"></div>
        <div id="linja3" class="linja"></div>
    </div>
    <ul class="nav" id="nav">
        <li><a href="#ikkuna">Etusivu</a></li>
        <li><a href="#osa-b">Tietoa</a></li>
        <li><a href="#osa-b">Projektit</a></li>
        <li><a href="#osa-c">Yhteydenotto</a></li>
    </ul>
</div>
```

Kuva 16. Navigaation HTML-rakenne

Valikko-wrapperin ja tämän pesiytyneiden elementtien jälkeen siirryin sivuston tyyliohjeeseen, jossa suunnittelin hamburger-valikon linjat. Annoin linja luokalle ominaisuudeksi 3 pikselin korkeuden ja 100% leveyden. Linjat käyttävät valikko -elementin leveyttä, joka on 30 pikseliä. Linjojen sijainti on asetettu tyyliohjeessa

relatiiviseksi. Jotta linjat saadaan koottua kasaan, käytin apunani CSS-funktiota translateY, joka siirtää elementtejä pystysuuntaisesti sivustolla. Annoin ensimmäiselle linjalle arvoksi -4 pikseliä ja kolmannelle linjalle 4 pikseliä.

Navigointipalkin piilottamiseen käytin JavaScript-tiedostoa. Asetin navigointipalkin display asetuksen niin, että navigointipalkin elementti hävitetään näkyviltä HTML-tiedostossa. JavaScript-tiedosto toimii hamburger menun kytkimenä, jolla navigointipalkki saadaan näkyville. Lisäsin CSS-tiedostoon luokan ”kytkin”, jolla on vain yksi ominaisuus. Ominaisuus otetaan käyttöön, kun vierailija painaa hamburger valikkoa. Kun vierailija klikkaa ikonia uudestaan, JavaScript funktio palauttaa navigaatiopalkin entisen arvon ja elementti hävitetään. Funktio yhdistetään HTML-tiedostossa valikon -elementtiin.

```
function navKytkin(){
  document.getElementById("nav").classList.toggle("kytkin");
}
```

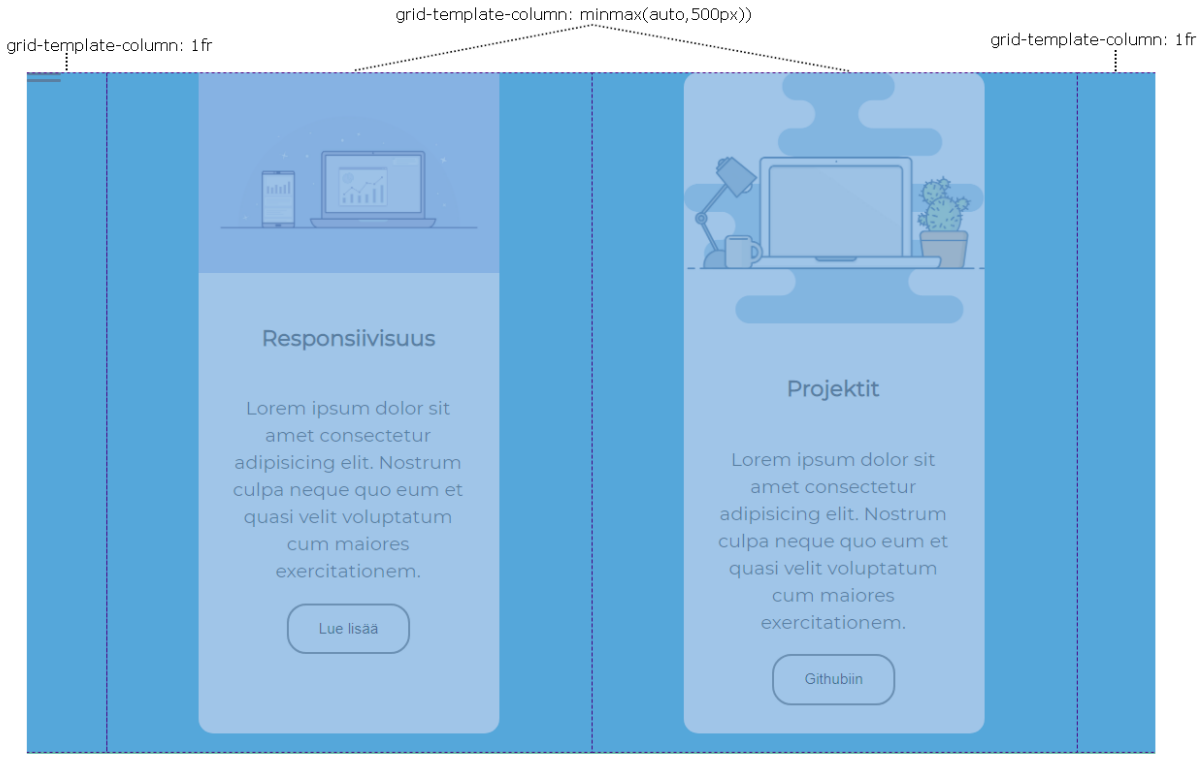
Kuva 17. Kytkimen funktio

4.3.5 Mediakyselyt

Demosivuston mobiilirakenne on nyt valmis, joten jäljellä on osien järjestely laajemmalle resoluutiolle. Käytin avukseni osien järjestelyyn CSS:n gridiä. Aloitin laajemman rakenteen suunnittelun CSS-tiedostossa. Lisäsin mediakyselyn, joka aktivoi sisällä olevien luokkien ja ID:n ominaisuudet, jos sivuston viewport on suurempi kuin 700 pikseliä. Asetin mediakyselyn sisälle ”grid” luokan ja yhdistin sen HTML-tiedostossa alueisiin ikkuna, osa B ja osa C. Grid luokan sisälle lisäsin ominaisuuden display: gridin sekä grid-template-columns. HTML-elementistä muodostuu grid container, jos sen display asetus asetetaan gridiksi. Grid-template-columnilla muodostetaan gridin rakenteen sarakkeet sisällön järjestämistä varten.

Annoin gridin uloimmille sarakkeille arvoksi suhteellisen mittayksikön 1 fr (fraction). Fraction sarakkeet joustavat resoluution mukaan. Ne säätävät automaattisesti viewportin leveyden ylimääräistä tilaa, jos resoluutio laajenee tai pienenee. Maksimissaan ne

ovat yhtä leveitä kuin kuvan 18 keskimmäiset sarakkeet, minimissään niitä ei näe lainkaan. Keskimmäiset sarakkeet hakevat minimiarvonsa automaattisesti ja lähtevät arvosta ylöspäin 500 pikseliin asti. Sain kuvan 18 kortit horisontaaliseksi lisäämällä mediakyselyyn osa B:n -elementin. Annoin -elementille flex display ominaisuuden ja otin sille käyttöön gridin keskimmäiset sarakkeet grid-column ominaisuudella. Flex asetus rivittää kortit automaattisesti, jonka takia se ei vaadi ylimääräisiä asetuksia. Koska kortit ovat mediakyselyssä horisontaalisesti, annoin leveydeksi 40%, jotta ne erottuvat toisistaan. Lopuksi lisäsin mediakyselyyn sivuston muut alueet, joille annoin grid-column ominaisuudet, jotta nekin saavat käyttöönsä sisällölle tarkoitetut sarakealueet.



Kuva 18. Gridin rakenne

5 LOPUKSI

Opinnäytetyöni tarkoituksena oli käsitellä, mitä responsiivinen web-suunnitelma sisältää. Opinnäytetyö aloitettiin teoriaosuuden kirjoittamisella, jonka avulla rakensin sitä tukevan toiminnallisen osuuden. Toiminnallisen osuuden projekti onnistui mielestäni hyvin. Tavoitin projektin määrittelyssä tehdyt vaatimukset. Sivusto reagoi joustavasti käytettävän näytön resoluution tilaan. Projektin loppuvaiheessa halusin käyttää testaukseen älypuhelinta ja tablettia, mutta tyydyin verkkoselaimen kehitystyökaluihin.

Projektia kehittäessäni löysin suureksi haasteeksi mobiili ensin - ajattelutavan. Sivuston kehitys ja sen web-teknologiat olivat minulle jo valmiiksi tuttuja, mutta verkkosivuston suunnittelu mobiilin pohjalta oli täysin uutta. Projektin edetessä sivuston rakentaminen ja mediakyselyjen luonti laajennuksille tuntui helpommalta. Olen tyytyväinen siihen, että sain projektista tarpeeksi haasteellisen.

Projektityön rakenteen suunnittelua voisi jatkaa lisäämällä uusia breakpointeja eri näyttökoille, joissa otetaan myös huomioon laitteen asettelu. Käyttöliittymäsuunnittelun ja ulkoasusuunnittelun jälkeen aion jatkaa projektia henkilökohtaiseksi web-sovellukseksi, jonka avulla harjoittelen itsenäisesti front-end ohjelmoinnissa käytettyjä JavaScript kirjastoja kuten Reactia ja Angularia. Käyttöliittymän suunnittelu ja rakentaminen liittyy front-end ohjelmointiin, josta olen kiinnostunut.

LÄHTEET

Kirjalliset lähteet

Adobe. 2017. Flash & The Future of Interactive Content. Viitattu 20.10.2019. Saatavissa: https://theblog.adobe.com/adobe-flash-update/.

Bootstrap. Responsive breakpoints. Viitattu 20.3.2018. Saatavissa: https://getbootstrap.com/docs/4.0/layout/overview/

Gremillion, B. 2015. A Hands-On Guide to Mobile-First Responsive Design. Viitattu 5.3.2019. Saatavissa: https://www.uxpin.com/studio/blog/a-hands-on-guide-to-mobile-first-design/

Gregory, S. 2018. Why Responsive Design is important and Google approved. Viitattu 19.1.2019. Saatavissa: https://freshsparks.com/why-responsive-design-is-important/

Marcotte, E. 2010. Responsive Web Design. Viitattu 10.3.2018. Saatavissa: http://alistapart.com/article/responsive-web-design

Marcotte, E. 2011. A Book Apart: Responsive Web Design. Viitattu 10.3.2018.

Marcotte, E. 2015. Responsive Design: Patterns & Principles (EPUB) Viitattu 20.1.2019

Mozilla. 2018. A re-introduction to JavaScript (JS tutorial). Viitattu 20.3.2018. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript

Pingdom, 2012. Make your website TV-friendly with our 10 great tips. Viitattu 13.9.2019. Saatavissa: https://royal.pingdom.com/make-your-website-tv-friendly-with-our-10-great-tips/

Simpson, R. 2016. Mobile and tablet internet usage exceeds desktop for first time worldwide. Viitattu 12.3.2018. Saatavissa: http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide

Tutorialspoint. CSS Tutorial. Viitattu 20.3.2018. Saatavissa: https://www.tutorialspoint.com/css/what_is_css.htm

Ward, C. 2017. Jump Start Responsive Web Design: Modern Responsive Solutions. Viitattu 20.1.2019

W3C. 2012. Media Queries, W3CRecommendation. Viitattu 18.3.2018. Saatavissa: http://www.w3.org/TR/css3-mediaqueries/.

W3C. 2017. HTML5.2, W3CRecommendation. Viitattu 18.3.2018. Saatavissa: https://www.w3.org/TR/html5/single-page.html

Kuvalähteet

Avery, J. 2016. More Donuts responsive Design example. Viitattu 10.3.2018. Saatavissa: https://responsivedesign.is/examples/lots-of-donuts/more-donuts/

Faulkner, S. 2015. Easy content organisation with HTML5. Viitattu 20.3.2018. Saatavissa: https://developer.paciellogroup.com/blog/2015/09/easy-content-organisation-with-html5/

Gregory, S. 2018. Why Responsive Design Is Important And Google Approved. Viitattu 19.1.2019. Saatavissa: https://freshsparks.com/why-responsive-design-is-important/

Marcotte, E. 2011. A Book Apart: Responsive Web Design. Viitattu 10.3.2018.

Tutorialspoint. CSS Tutorial. Viitattu 20.3.2018. Saatavissa: https://www.tutorialspoint.com/css/what_is_css.htm

WebFX. A Look at the Off-Canvas Menu Design Pattern. Viitattu 7.10.2019. Saatavissa: https://www.webfx.com/blog/web-design/off-canvas-menu/