Sami Perttilä

RESPONSIIVISET VERKKOSIVUT

RESPONSIIVISET VERKKOSIVUT

Sami Perttilä
Opinnäytetyö
Kevät 2017
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä: Sami Perttilä
Opinnäytetyön nimi: Responsiiviset verkkosivut
Työn ohjaaja(t): Risto Korva
Työn valmistumislukukausi ja -vuosi: Kevät 2017 Sivumäärä: 48 + 3 liitettä

Tämä opinnäytetyö käsittelee verkkosuunnittelun historiaa ja tärkeimpiä huomioitavia asioita verkkosuunnittelussa. Työllä ei ollut toimeksiantajaa, vaan se tehtiin omalta harrastus- ja kiinnostuspohjalta. Tämän työn päätarkoitus on olla yksinkertainen opas responsiiviseen verkkosuunnitteluun ja selittää lukijalle, miksi responsiivinen verkkosuunnittelu on nykyaikana suositelluin tapa toteuttaa verkkosivuja.

Nykyaikana Internet-kykyisten laitteiden kirjo on kasvanut niin erilaiseksi, että se pitää huomioida verkkosivujen kehityksessä. Tämän ja muutamien muiden seikkojen vuoksi perinteiset verkkosivut eivät enää kelpaa. Työssä esitellään myös kilpailevat, mutta käytännössä vanhanaikaiset verkkosivujen toteutustavat, kuten erillinen mobiilisivusto ja adaptiivinen verkkosuunnittelu. Tuon esille eri tekniikoiden hyvät puolet ja heikkoudet, minkä avulla lukija voi ymmärtää, miksi responsiivinen verkkosuunnittelu on yleisesti paras tekniikka.

Responsiivinen verkkosuunnittelu käsittellään tarkemmin ja selitetään sen peruskonsepti ja vaadittavat teknologiat. Työssä käydään läpi joustava asemointi, joustavat kuvat ja mediakyselyt, sekä niiden merkitys responsiivisen verkkosuunnittelun kannalta. Tämän lisäksi esitellään vaadittavat HTML5 ja CSS3 ominaisuudet. Verkkosivun toteutuksessa esitellään responsiivisen konsepti vielä tarkemmin ja sen, miten responsiivinen verkkosuunnittelu vaikuttaa verkkosivuston käyttäytymiseen ja muovautumiseen kuvien avulla.

Tämän työn toteutuksessa tuotetaan responsiivinen verkkosivu, jonka on tarkoitus toimia esimerkkinä, kuinka toteutetaan yksinkertainen responsiivinen verkkosivusto. Aloittelevat web-koodaajat ja opiskelijat voivat käyttää sitä responsiiviseen verkkosuunnitteluun tutustumiseen ja mallipohjana responsiivisesta verkkosivusta. Responsiivinen verkkosuunnittelu kuitenkaan ei rajaa verkkosivun ulkoasua, joten tämä toteutus on vain yksi esimerkki.

Asiasanat: verkko-ohjelmointi, verkkosivut, CSS, HTML

ALKULAUSE

Tällä opinnäytetyöllä ei ollut tilaajaa, mutta aihe valikoitui aikaisemman työkokemuksen perusteella. Opinnäytetyö muodostui raportin kirjoituksesta ja verkkosivun toteutuksesta. Erityiskiitokset työn valmiiksi saattamisesta veljelleni Vili Perttilälle, ohjaavalle opettajalle Risto Korvalle ja kielenohjaaja Tuula Hopeavuorelle.

Oulussa 25.5.2017

Sami Perttilä

SISÄLLYS

SANASTO

Container
Verkkosivukehityksessä nimitys HTML-elementille, jonka tehtävä on vain varata tilaa sijoitettavalle sisällölle. Yleinen container on div-elementti.

CSS / CSS3
Cascading Style Sheets on tyylikieli, jota käytetään muotoilemaan merkintäkielellä tehdyn dokumentin ulkonäköä. CCS3 on yleisnimitys CSS taso 3:een, joka toi uusia ominaisuuksia CSS-tyyliohjeisiin.

DOM
Dokumenttioliomalli on puurakenne, jossa jokainen solmukohta kuvastaa HTML-elementtiä. Puussa kaikista ylimmäinen elementti on dokumentti.

Elementti
Elementit muodostuvat HTML-dokumentilla olevista tageista, kun selain piirtää ne. Nämä elementit muodostavat verkkosivun sisällön.

FLUID GRID
Joustava asemointi on responsiivisen verkkosuunnittelun perusta. Sen tarkoitus on toteuttaa suhteelliset koko yksiköt, jotta sisältö skaalautuisi eri päätelaitteille.

HTML / HTML5
Hypertext Markup Language eli hypertekstin merkintäkieli. Avoimeen standardiin perustuva merkintäkieli, jota käytetään eniten verkkosivujen tekemiseen. HTML5 on yleisnimitys HTML 5.x-versioille ja usein siihen yleistetään myös uudet CSS3-ominaisuudet.

Parent / Child
Näillä termeillä kuvataan elementin suhteellista sijaintia DOM-rakenteessa. Parent tarkoittaa DOM-rakenteessa olevaa lähintä edeltäjää ja child tarkoittaa seuraavaa elementtiä DOM-rakenteessa, joka on yhteydessä käsittelyssä olevaan elementtiin.

Tagi

HTML-dokumentti sisältää tageja, jotka määrittävät elementit ja niiden ominaisuudet. Tagit eivät siis ole sama asia kuin elementit.

Tietokone

Tässä työssä tietokone-käsite viittaa aina pöytätietokoneeseen. Esimerkiksi kannettava tietokone ei sisälly tähän kategoriaan.

User agent / user-agent

User agent on ohjelma, joka toimii käyttäjän puolesta. Tunnetuin on verkkoselain, joka ilmoittaa verkkosivustolle mm. käytetyn selaimen tiedot ja käyttöjärjestelmän. User-agent on arvokenttä HTML-tunnisteessa, jossa em. tiedot ovat.

W3C

World Wide Web Consortium on olennaisin kansainvälinen standardisoimisjärjestö World Wide Webille. W3C pyrkii valvomaan yhteensopivuutta toimialan jäsenten määrittelemien uusien standardien kesken.

1 JOHDANTO

Tämä opinnäytetyö käsittelee verkkosivuston suunnitelua ja toteuttamista responsiivisesti. Työllä ei ollut erillistä toimeksiantajaa, vaan tein sen omalta harrastus- ja kiinnostuspohjalta. Aiheeseen päädyin työskenteltyäni yrityksessä, joka oli kehittänyt selainpohjaisen resurssienhallintaohjelman. Tämä ohjelma piti kuitenkin saada skaalautuvaksi ja erityisesti toimimaan myös tableteilla. Tästä päädyin aiheeseen responsiivinen verkkosuunnittelu. Responsiivinen verkkosuunnittelu on erittäin ajankohtainen, koska sen tarkoitus on ottaa huomioon käyttäjien erilaiset päätelaitteet, jotka viime aikana ovat hyvin paljon yleistyneet. Monet yritykset kuitenkin vielä tyytyvät entisiin perinteisiin kiinteän kokoisiin verkkosivustoihin tai erillisiin mobiilisivustoratkaisuihin.

Internetin käyttö on muuttunut mobiilikeskeisemmäksi, minkä vuoksi verkkosivustot täytyy suunnitella uudelleen huomioimaan myös pienemmät laitteet kuin perinteiset tietokoneet. Työssäni tuon esille syitä, miksi yritysten kannattaisi päivittää vanhat verkkosivustonsa tai tehdä responsiivinen verkkosivusto. Responsiivisella suunnittelulla on myös kilpailevia ratkaisuja, mutta kustannustehokkuudessa ne eivät pärjää. Yleensäkin ne ovat myös hankalampia toteuttaa sekä ylläpitää. Responsiivinen suunnittelu usein ajatellaan vain mobiililaitteiden huomioinniksi, mikä ei ole totta. Sen tarkoitus on toteuttaa yksi verkkosivusto, joka skaalautuu kaikenkokoisille päätelaitteille.

Responsiivisen verkkosuunnittelun loi Ethan Marcotte, joka kirjoitti kirjan Responsive Web Design (Marcotte, E. 2014). Työssäni se oli ohjekirjanani ja suosittelen jokaista web-kehittäjää sen lukemaan. Kirja sopii hyvin aloittelijoille ja kehittyneemmillekin ja sen avulla saa helposti responsiivisen verkkosivuston toteutettua. Työssäni esittelin responsiivisen suunnittelun peruskonseptin, joka koostuu joustavasta asemoinnista, joustavista kuvista ja media queryistä. Selitän myös näiden merkityksen ja toteutustavan.

Toteutuksen esittelen työn lopuksi, kun toteutan responsiivisen verkkosivun. Siinä esittelen responsiivisen suunnittelun kannalta merkitsevät koodit ja selitän niiden merkityksen. Kuvien avulla myös hahmotan lukijalle, miten toteuttamani

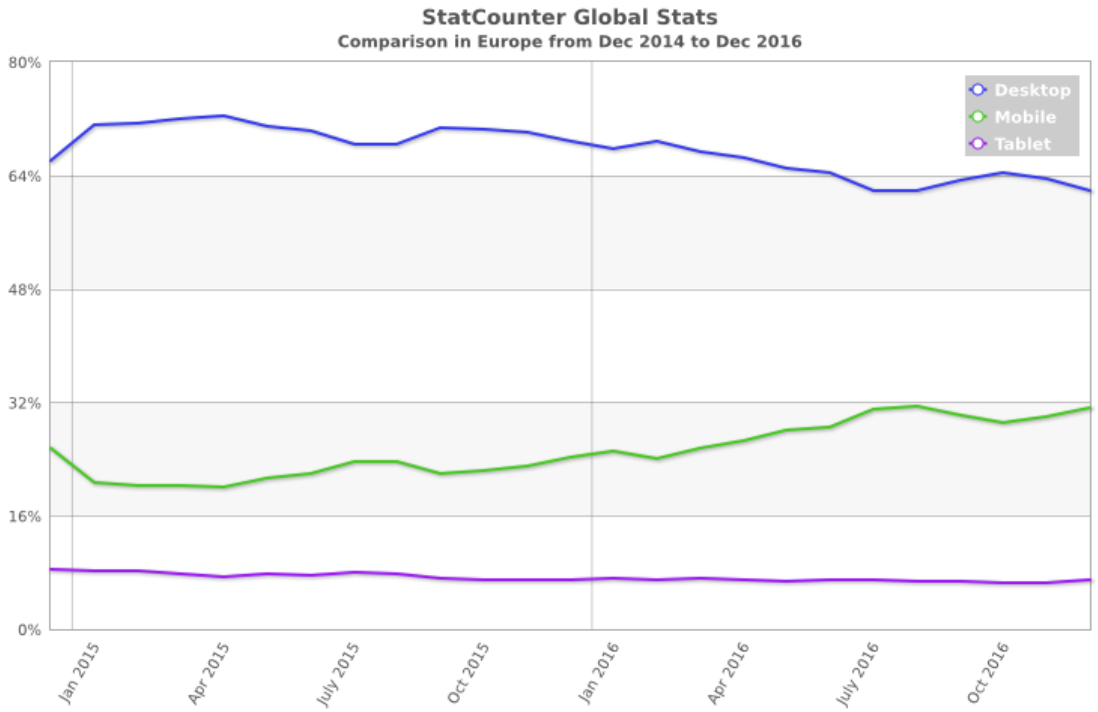
verkkosivu käyttäytyy eri päätelaitteilla. Toteutuksen on tarkoitus toimia esimerkkinä yhdenlaisesta responsiivisesta verkkosivusta, mutta peruskonsepti kaikille responsiivisille verkkosivuille on kuitenkin aina sama.

2 TAUSTAA

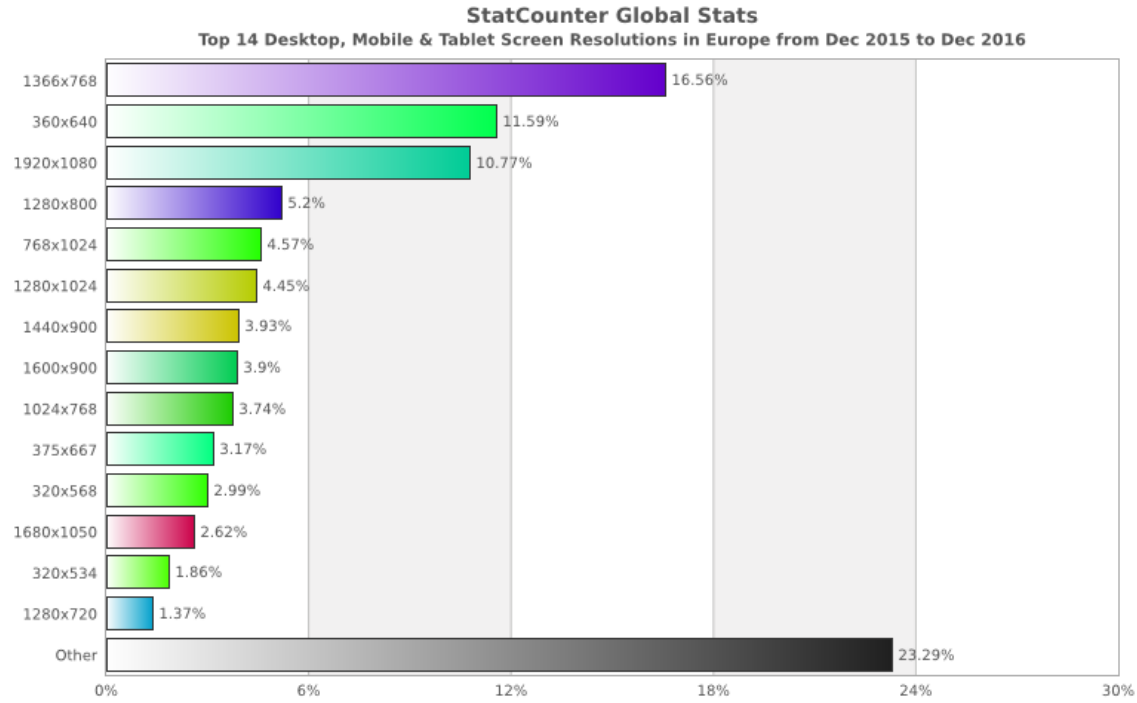
Ennen verkkoa selattiin vain tietokoneilla ja kannettavilla tietokoneilla, mikä yksinkertaisti verkkosuunnittelua, koska riitti, että sivustolla oli yksi ulkoasu ja se toimi hyvin. Nämä perinteiset verkkosivut oli usein suunniteltu kiinteän kokoisiksi 960 pikselin levyisiksi. Nykyaikana tosin erilaisten Internet-kykyisten päätelaitteiden määrä on laaja. Varsinkin koska ne ovat eri kokoisia, vanhat kiinteän kokoiset verkkosivut eivät enää kelpaa, koska näkymä ei vastaa enää suunniteltua pienellä laitteella. Mobiiliverkkoselaus yleistyy myös koko ajan.

2.1 Syitä mobiililähtöiseen verkkosuunnitteluun

Mobiililaitteet ohittivat jo tietokoneet Internetin käytössä koko maailmassa marraskuussa 2016 ja suomalaisille tärkeällä Euroopan markkina-alueella mobiililaitteiden osuus Internetin käytöstä on jo myös noin 40% (kuva 1). Tämän on jo yritykset huomioineet digitaalisessa markkinoinnissa ja luoneet mobiililaitteille suunniteltuja verkkosivustoja. Teknologian kehityksen myötä on mobiililaitteiden kirjo kuitenkin lisääntynyt, mikä tarkoittaa toistuvaa mobiilisivujen uudelleen suunnittelua uusille laitteille (kuva 2). Responsiivisilla verkkosivuilla päästään eroon tästä ongelmasta ja saadaan eri laitteille skaalautuvat verkkosivut.



KUVA 1. Kuvaaja tietokoneen, puhelimen ja tabletin suhteesta Internetin selauksessa Euroopassa (StatCounter 2016)



KUVA 2. Kuvaaja eri resoluutioiden suhteista Euroopassa verkkoselauksessa

2.2 Google ja mobiilituki

Tärkein kohderyhmä melkein jokaiselle yritykselle verkkosivujen suunnittelun kannalta on Google, koska se on käytetyin hakukone (Ratcliff, C. 2016). Hakukoneiden kautta ihmiset löytävät verkkosivut, ja jopa selainten osoiterivit toimivat hakukoneina taustalla esimerkiksi tapauksissa joissa käyttäjä ei kirjoita täydellistä www-osoitetta osoiteriville. Google julkaisi jo vuonna 2015 mobile friendly -algoritmin hakutulosten listaamiseen. Algoritmin idea on nostaa mobiililaitteille suunniteltujen sivustojen tuloksia hauissa, kun haetaan mobiililaitteella. Tällä tavalla Googlen-haku löytäisi relevantimpia ja laadukkaampia tuloksia mobiililaitteilla. (Makino T. ja Phan D. 2015.) Seuraavan vuoden 2016 keväällä Google vielä tehosti algoritmin vaikutusta, mikä tarkoitti, että mobiililaitteella haettaessa ei-mobiiliystävälliset sivustot joutuvat listauksessa alaspäin. Uusi algoritmi käytännössä pakottaa viimeisetkin verkkosivustot päivittämään sivustonsa mobiiliystävällisiksi. (Kloboves, K. 2016.) Google suosittelee käytettäväksi responsiivista verkkosuunnittelua (Google Developers 2015).

Viime vuosina myös sosiaalisen median käyttö on kasvanut eksponentiaalista vauhtia. Isot yhtiöt, kuten Facebook, ovat panostaneet mobiililähtöisyyteen ja sitä käytetäänkin useimmiten mobiililaitteilta. (Facebook 2016.) Facebookista ja muista some-palveluista kuitenkin usein siirrytään muille sivustoille linkkien välityksellä, mikä on vaatinut muita sivustoja kehittämään sivustonsa mobiililähtöisiksi. Tämä on puskenut verkkosivustojen kehitystä mobiililähtöiseksi.

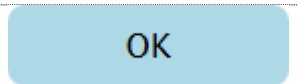
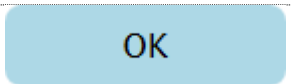
2.3 Näytöt

Verkkosivustoa suunnitellessa näytöt eivät kuitenkaan vain rajoitu mobiililaitteisiin ja muihin laitteisiin. Näytöt voidaan jakaa useaan kategoriaan, mutta verkkosivusuunnittelun kannalta yksinkertaisinta on jakaa neljään ryhmään: pienet (puhelin), keskikoiset (tabletti/kannettava), työpöytä (kannettava/tietokone) ja isot (tietokone/premium-näytöt). Kategoriat määräytyvät loogisten pikselitarkkuuksien perusteella. Loogisia ja fyysisiä pikseleitä käsitellään tarkemmin luvussa 2.1.3.

Tällä hetkellä ylivoimaisesti myydyin tietokoneen näyttöresoluutio on 1920 × 1080 ja kannettavissa tietokoneissa yleisin näytön resoluutio on 1366 × 768, koska resoluution nostaminen vaatii usein myös muun laitteiston päivittämistä, mikä nostaisi kannettavan hintaa liikaa. Kuitenkin 2560 × 1440 monitorit ovat yleistymässä ja 4k-näytöt eli 3840 × 2160 resoluution näytöt ovat jo kaupoissa tosin nämä ovat vielä marginaalinen osuus kaikista käyttäjistä. Verkkosivuston suunnittelussa kuitenkin sivusto ensimmäisenä tehdään kohderyhmälle, tämän jälkeen vasta massalle. Esimerkiksi verkkosivusto, joka myy premium-tuotteita, on selkeästi kohdistettu varakkaammille henkilöille. Varakkailla on jo mahdollisesti laadukkaampi näyttö ja upeampi elämys luo verkkokaupassa todennäköisemmin kaupan.

2.4 Pikseli-yksikkö

Verkkosivuja toteutettaessa perusmittayksikkönä käytetään pikseliä. Tämä johtuu siitä, että näyttöpaneeli muodostuu pikseleistä ja yksi pikseli on pienin yksikkö, mitä näytölle voidaan piirtää. Tästä myös seuraa, että pikselin reaalinen koko on suhteellinen näyttöpaneelin kokoon eli pikseli ei ole vakiokokoinen mittayksikkö, vaan toisella näyttöpäätteellä yksi pikseli voi olla monta kertaa isompi tai pienempi (kuva 3). Tämän ongelman ovat laitevalmistajat ratkaisseet verkkosuunnittelijoiden puolesta ja luoneet ns. pikselisuhteen. Esimerkiksi 5,2 tuuman puhelimen näytön resoluutio on 1080 × 1920 ja pikselisuhde on 3. Pikselisuhteella jaetaan näytön resoluutio, eli $\frac{1080}{3} = 360$ ja $\frac{1920}{3} = 640$. Näin saadaan esimerkiksi CSS:n havaitsema ns. looginen resoluutio 360 × 640. Loogisen resoluution avulla saadaan pikseliyksikkö suhteutettua suunnilleen samankokoiseksi pienellä näytöllä kuin esimerkiksi tietokoneen näytöllä.

24” näyttö (1920 × 1080)	5,2”:n puhelin horisontaalisti (1920 × 1080)	5,2” puhelin pikselisuhdetta käyttäen (640 × 360)
		

KUVA 3. Sama OK-nappi eri näytöillä. Kuva esittää mittasuhde-eroa, joka johtuu eri kokoisista pikseleistä.

Pikselisuhteen täytyy kuitenkin olla kokonaisluku, koska se kuvastaa, kuinka montaa fyysistä pikseliä yksi looginen pikseli vastaa pysty- ja vaakasuunnassa

paneelilla. Puolikkaita pikseleitä ei siis voisi piirtää. Tämän vuoksi ratkaisu ei ole täydellinen, koska pientä resoluutiovaihteluja tulee saman kokoisilla näytöillä. Suurimmat mittasuhde-erot voidaan kuitenkin kompensoida ja ulkoasu säilyy hyvänä. Verkkosivujen suunnittelijalta vaaditaan kuitenkin testausta näkymän toimivuudesta eri päätelaitteilla.

2.5 Muita verkkosivuston kehitystekniikoita

Verkkosivuja on suunniteltu vuosia sidoksissa tiettyihin pikselileveyksiin. Erityisesti 960 pikselin leveys on suosittu jaollisuutensa takia. Tällöin ulkoasukin on optimaalinen vain 960 pikselin levyisessä näytöllä tai ikkunassa. Nykyään Internet-kykyisiä laitteita on kuitenkin niin paljon erilaisia, että näkymä toisella laitteella ei enää vastaa suunniteltua. Tämä yleensä korjataan jollain kolmesta eri tavasta. Yleisin tapa niistä on ollut erillinen mobiilisivusto, joka on suunniteltu mobiililaitteille, ja vanha sivusto pidetään ennallaan. Toinen tapa on adaptiivinen verkkosuunnittelu, joka on samantyylinen kuin responsiivinen suunnittelu, mutta huomattavasti monimutkaisempi toteuttaa ja ylläpitää. Kolmas tapa on tietenkin responsiivinen verkkosuunnittelu.

2.5.1 Mobile first -suunnittelu

Mobile first eli mobiililähtöinen verkkosuunnittelu on vanhempi konsepti kuin responsiivinen verkkosuunnittelu. Luke Wroblewski esitteli jo vuonna 2009 idean, kuinka verkkosivut tulisi suunnitella ensin mobiililaitteille, ja sitten vasta tavallisille tietokoneille. Tämä oli mullistavaa, koska älypuhelimet olivat vielä alkutekijöissään. Tällöin moni web-kehittäjä ajatteli, että ”sivusto on liian laaja kännykälle” tai ”sivusto on liian raskas kännyköille”, minkä vuoksi ei kannattaisi moista edes yrittää. Ennen mobiililähtöistä suunnittelua verkkosivut toteutettiin mobiililaitteille käyttäen ns. graceful degradation -strategiaa (rappeuttamis-strategia). Strategian idea oli toteuttaa ensin paras versio verkkosivusta, mutta huonommille ja vanhemmille selaimille tai päätelaitteille tarjota ns. karsittua näkymää, jotta sivusto ei rikkoutuisi täysin. Toimivuus oli kuitenkin usein heikko, koska näkymä vanhemmilla laitteilla oli tökerö ja huonosti ylläpidetty tai mobiililaitteet eivät tukeneet sivustolla tarvittavia ominaisuuksia.

Mobiililähtöisessä suunnittelussa työ tehdään toisin päin. Verkkosivusto suunnitellaan kohderyhmän huonoimman laitteen mukaan ja siitä lähdetään lisäämään verkkosivustolle ominaisuuksia. Tällöin toiminnallisuus säilyy, koska lähtökohtana oli heikoin laite. Ominaisuuksia pystyy aina lisäämään laitteiden ja teknologioiden kehittyessä. Toki työstäminen näin päin on hankalampaa, mutta jos pyrkii hyvään toimivuuteen, lopulta työmäärä on pienempi kuin vanhalla tavalla. Tämä kehitystapa tunnetaan myös nimellä progressive enhancement eli asteittain parantaminen. Mobile first on edelleen ajankohtainen käsite verkkosivujen ulkoasun suunnittelussa, jolloin tarkoitetaan lähinnä parhaan käytettävyyden varmistamista mobiililaitteille.

2.5.2 Mobiilisivusto

Mobiilisivusto on täysin erillinen verkkosivusto mobiilikäyttäjiä varten. Mobiilisivuston avulla voi huomioida mahdolliset mobiililaitteen asettamat rajoitukset ja keskittyä tarjoamaan yleisesti mobiililaitteille parhaan mahdollisen palvelun, säilyttäen perussivuston ennallaan. Erillisellä mobiilisivustolla voidaan helposti rajoittaa sisältöä varsinkin, jos yhteysnopeudet ovat hitaita. Näin saadaan sivuston sisältö nopeammin ladattua ja käyttäjän luettavaksi. Mobiilisivustolla luodaan mobiilikeskeinen käyttöliittymä, jonka avulla käyttömukavuus mobiililaitteilla on erinomainen.

Mobiilisivusto on kuitenkin toinen erillinen verkkosivusto, mikä tarkoittaa, että ylläpidettävänä on kaksi verkkosivustoa. Toisen verkkosivuston kehittäminen nostaa tuotantokustannuksia ja usein myös aiheuttaa päivitysongelmia. Toisen sivuston päivittäminen voi vaatia myös toisen verkkosivuston päivittämistä yhteensopivuuden säilyttämiseksi. Ylimääräinen verkkosivusto nostaa myös yleisiä ylläpitokustannuksia.

Mobiilisivusto voidaan toteuttaa eri aliverkkotunnuksen alle esimerkiksi m.iltalehti.fi, mikä on helpompi tapa hallita ja toteuttaa sivusto. Sen heikkous on kuitenkin, että aina kun joku vierailija jakaa linkin mobiililaitteelta sivustolle, se linkki ohjaa myös vastaanottajan mobiilisivustolle riippumatta siitä, mitä laitetta vastaanottaja käyttää. Tämän lisäksi erillinen aliverkkotunnus tuo haasteita hakukoneoptimoinnin kanssa. Hakukoneille täytyy antaa erilliset ohjeet, kuinka

huomioida molemmat sivustot kaksinkertaisen sisällön vuoksi ja ohjeet ovat herkkiä virheille. Mm. Googlen hakurobotit eivät ole velvoitettuja noudattamaan annettuja ohjeita, vaan ne vain pyrkivät noudattamaan niitä (Mueller, J. 2009.) Tämä ja muut virheet hakukoneoptimoinnissa aiheuttavat alemman sijainnin hakutuloksissa verrattuna yhteen verkkotunnukseen.

Mobiilisivuston voi toteuttaa myös käyttäen Dynamic Servingiä. Dynamic Serving eli vapaasti suomennettuna dynaaminen tarjoilu ei erottele sivustoja eri aliverkkotunnuksiin, vaan palvelin tunnistaa käyttäjän user agentin HTTP-tunnisteesta. User-agent-arvo on tekstikenttä, joka selaimissa kertoo käyttäjän selaimen, käyttöjärjestelmän ja laitteiston tiedot. Riippuen tästä arvosta, palvelin lähettää kyseiselle laitteelle suunnitellut HTML-, CSS- ja JavaScript-tiedostot sekä muut mediat. Dynamic Serving ohittaa hakukoneoptimointi ongelmat, mutta luo uuden ongelman. User agentin tunnistuksen epäonnistuessa näkymästä voi tulla täysi sekasotku. Tämän vuoksi dynaamista tarjoilua käyttävä sivusto vaatii hyvää ylläpitoa, jotta user-agent-arvolista on aina ajan tasalla.

2.5.3 Adaptive web design -suunnittelu

Adaptive web design eli adaptiivinen verkkosuunnittelu on tapa ratkaista sama ongelma kuin minkä responsiivinen verkkosuunnittelu ratkaisee. Tämän vuoksi monesti nämä kaksi käsitettä sekoitetaan keskenään. Varsinkin tämä virhe tapahtuu suomen kielessä, kun molemmat adaptive ja responsive kääntyvät suomeksi tässä asiayhteydessä ”sopeutuva” tai ”mukautuva”. Kuitenkaan ne eivät ole verkkosuunnittelussa sama asia, vaikka molemmilla ratkaistaan sama asia, mutta eri tavalla.

Adaptiivisessa suunnittelussa voidaan käyttää staattisia yksiköitä eli pikseleitä ilman ongelmia, mutta luodaan mediakyselyiden avulla breakpointit eli reagointirajat ennalta päätettyihin resoluutiokokoihin, joissa asemointi muuttuu. Nämä resoluutiorajat valitaan sillä perusteella, milloin elementit eivät mahtuisi kokonaisina näkymään. Näin vältetään, että ulkoasu ei hajoa isojen kiinteiden kappaleiden takia, mutta ulkoasun muutos ei tapahdu sulavasti. Näkymä muuttuu vain breakpoint-kohdissa, eikä ollenkaan niiden välissä. Tämä on käyttäjän suurin

huomaava ero. Mediakyselyt ja breakpointit käsitellään tarkemmin luvussa 3.1.3.

Adaptiivinen suunnittelu pyrkii optimoimaan myös laitekohtaisesti, ei pelkästään näyttöpäätteen koon ja resoluution mukaan. Ensiksi tunnistetaan päätelaite sekä selain ja sen mukaan tarjoillaan oikeat tyylisäännöt. Tällä tavalla saadaan huomattavasti yksilöllisempää kohdennusta ja taataan parempi toimivuus eri laitteille. Heikommalla tai vanhemmalla laitteella voidaan ladata eri tyylisäännöt ja näin jättää asioita näyttämättä, kuten videot tai isot kuvat. Toisaalta työn määrä on moninkertainen ja vaatii toistuvaa päivittämistä uusien laitteiden ilmestyessä, joten ainoastaan erittäin isoille yrityksille itse suosittelisin adaptiivisia verkkosivuja.

3 RESPONSIIVINEN VERKKOSUUNNITTELU

Vuonna 2010 Ethan Marcotte nimesi uuden tavan ajatella verkkosuunnittelua. Tämä tapa oli responsiivinen verkkosuunnittelu. Se ei vaatinut uusia teknologioita tai lisäosia, mutta teki verkkosivustoista käytettäviä riippumatta käyttäjän laitteesta. Tarkemmin hän esitteli käsitteen artikkelissaan nimeltä Responsive Web Design verkkolehteen A List Apart. Artikkelissaan Marcotte totesi, kuinka mahdottomaksi tulee uusien laitteiden ilmestyessä suunnitella jokaiselle laitteelle erikseen toimiva näkymä, minkä vuoksi hän kehitti responsiivisen verkkosuunnittelun konseptin (Marcotte, E. 2010.) Tämän artikkelin ja hänen 2011 julkaiseman kirjansa Responsive Web Design myötä responsiivisesta suunnittelusta tuli tunnettua web-kehittäjien joukossa.

Responsiivisen verkkosuunnittelun tarkoituksena on toteuttaa verkkosivusto siten, että näkymä sopeutuu aina optimaalisesti käyttäjän näytölle. Oli laite iso tai pieni ja resoluutio mikä hyvänsä, näkymä skaalautuu suunnitellussa mittasuhteessa laitteelle. Tämä tarkoittaa, että käyttäjän tarvitsee rullata tai pyyhkiä vain yhteen suuntaan ja zoomauksen tarve minimoidaan. Isommilla näytöillä hyödynnetään suurempi pinta-ala eikä sitä jätetä vain hukkatilaksi. Toteutuksessa pitää kuitenkin huomioida, että sama tieto verkkosivulla pitää olla tarjolla isossa ja pienessä näytössä. Tarkoitus ei ole lisätä tietoa isoon näkymään vaan lähinnä siirrellä sen sijaintia.

Koska responsiivinen suunnittelu toimii kaikilla laitteilla, se voidaan toteuttaa mobile first -periaatteella tai ei. Tämä riippuu aivan verkkosivuston kohderyhmästä, mutta mobiilikäytön jatkuvasti yleistyessä mobile first -periaatetta melkein aina noudatetaan. Responsiivisessa suunnittelussa tämä näkyy pienimmän ulkoasun suunnittelussa ja siitä lähdetään asemoimaan muita näkymiä. Näin saadaan nopein, toimivin ja käytettävin käyttöliittymä varmistettua mobiililaitteille.

3.1 Responsiivisen suunnittelun konsepti

Marcotte määritteli responsiivisen suunnittelun käsitteen koostumaan kolmesta osasta. Nämä kolme osaa ovat fluid grid eli joustava asemointi, flexible images

eli joustavat kuvat ja media queries eli mediakyselyt. Fluid gridin idea on luoda suhteelliset mittayksiköt, jotta sisältö saadaan skaalautumaan eri päätelaitteille. Joustavilla kuvilla tai ylipäänsä kaikella medialla, joka toimii joustavassa sisällössä, Marcotte tarkoitti skaalautuvia kuvia. Mediakyselyillä on tarkoitus optimoida näkymää ja asettaa eri tyyliohjeet käyttöön riippuen päätelaitteen loogisesta resoluutiosta. Tällöin saadaan näkymä helposti asemoitua uudelleen.

3.1.1 Joustava asemointi

Responsiivinen suunnittelu tarttuu ensimmäisenä näkymän skaalautuvuus-ongelmaan käyttämällä joustavaa asemointia. Joustavassa asemoinnissa elementtien leveys ei ole staattinen vaan määritetään suhteelliseksi parentin eli kyseisen elementin sisältävän elementin leveyteen. Leveyden yksikkönä käytetään suhteellista kokoa. Näkymästä muodostuu fluid grid eli 12 sarakkeen taulukko. Näitä sarakkeita voi kuitenkin yhdistää keskenään, jolloin saa helposti halutun levyisiä containereita eli tilanvarauselementtejä. Fluid grid ei edellytä 12:ta saraketta, mutta yleisesti käytetään juuri 12:ta saraketta, koska siitä on helppo osioida sopivan kokoisia containereita. Kahdestatoista voi jakaa helposti 2, 3 tai 4 yhtä suurta containeria. Tarvittaessa sarakkeen voi jakaa aina uudelleen ja saada halutun kokoisen containerin. (Kuva 4.) Tämä on mahdollista, koska seuraavan elementin koko on aina suhteellinen parentin kokoon. Mitään desimaaliarvoja ei tyyliohjeissa kuitenkaan kannata pyöristää, koska selaimille ylimääräisistä desimaaleista ei ole haittaa, mutta pyöristys voi aiheuttaa muutamien pikselien virheen, joka ulkoasussa sitten näyttääkin tökeröltä.

KUVA 4. Fluid grid. Punaisella ympyröity on uudelleen jaettu sarake

Koska eri laitteilla on eri kokoiset pikselit, niin myös tekstikenttien koko täytyy antaa suhteellisena. Tämä tehdään antamalla sivustolle yksi pääkirjasinkoko, johon kaikki muu suhteutetaan. Tällöin ylläpito helpottuu huomattavasti, kun on

vain yksi staattinen arvo kirjasinkoolle. Vain yhtä arvoa tarvitsee säätää, jos jollain laitteella on liian pieni tai liian suuri teksti.

Yleensä suhteellinen koko tehdään käyttäen em- tai rem-yksikköä, mutta muitakin suhteellisia yksiköitä on. Em-yksikön laskukaava on seuraavanlainen:

$$\frac{kohde}{konteksti} = tulos \qquad \frac{24}{16} = 1.5(em)$$

Kaavassa *kohde* on kyseisen elementin kirjasinkoko pikseleinä, ja *konteksi* on valitun elementin parentin kirjasinkoko. Tästä tuloksena muodostuu suhteellinen arvo, joka on em-yksikköä. Rem toimii muuten samalla tavalla, mutta se perustuu aina juuren eli html-elementin kirjasinkokoon, ei kontekstin. 1 rem on siis aina sama kokoinen riippumatta, missä se dokumentilla sijaitsee.

3.1.2 Joustavat kuvat

Verkkosivuilla on paljon kuvia ja videota, ja näillä on ominaisuutena vakiokokoinen pikselimääritteinen tarkkuus eli koko. Verrattuna tekstielementteihin, jotka rivittyvät alaspäin automaattisesti elementin leveyden mukaan, kuvilla ja videoilla on ennalta määrätty pikselileveys. Perinteisesti verkkosuunnittelussa on kuville varattu tietyn kokoinen tila ja kuva on käsitelty sen kokoiseksi. Tilanvarauksen ollessa pikselimääritteinen kuvilla ei ole mahdollisuutta skaalautua isommiksi, mikä voi aiheuttaa käyttäjälle turhauttavaa zoomailua ja pienillä näytöillä tulee helposti myös horisontaalinen kelauspalkki. Tämän sijaan staattisen koon tilalla responsiivisessa verkkosuunnittelussa annetaan selaimen skaalata media oikean kokoiseksi varattuun tilaan.

Vakiotilanvarauksen sijaan medialle varataan tila käyttäen joustavaa asemointia, jolloin tilanvaraus on suhteellinen. Media sijoitetaan tilaan käyttäen CSS:n max-width-ominaisuutta, jonka arvoksi annetaan esimerkiksi 100 %, jolloin kuva näkyy alkuperäisen kokoisena. Näin selain skaalaa median kyseiseen tilaan mahdollisimman isoksi, kunnes se saavuttaa kuvan alkuperäisen koon. Kaikki nykyaikaiset selaimet osaavat skaalata kuvat ja videon säilyttäen kuvasuhteen. Tällä tavalla saadaan toteutettua joustavat kuvat ja videot.

Max-width-määritys on helppo ja yksinkertainen tapa toteuttaa joustavat kuvat, mutta kuvien resoluutioiden pitäisi silti olla suunnilleen niille varattujen elementtien kokojen mukaisia. Turhan isot kuvat, jotka skaalataan pieneksi, pidentävät vain latausaikoja ja kuluttavat turhaan verkkokaistaa. Ongelman voi kuitenkin kiertää käyttämällä img-elementin srcset-attribuuttia. Sen avulla voi antaa ylimääräisiä lähteitä kuvalle. Näin voimme tarjota erikokoisen kuvan mobiililaitteelle kuin tietokoneelle. Tästä esimerkki toteutuksessa.

3.1.3 Media queryt ja breakpointit

Kolmas pääpiirre responsiivisessa verkkosuunnittelussa on CSS:n media queryt eli mediakyselyt. Nämä laajentuivat CSS3:n myötä huomattavasti ja alkoivat tukea uusia ominaisuuksia, kuten max-width, min-width ja orientation. Mediakysely koostuu kahdesta osasta, mediasäännön arvosta ja kyselystä, jossa määritellään jokin ominaisuus. Mahdollisia arvoja mediasäännölle ovat

- all (kaikki)
- print (tulostus)
- screen (näytöt)
- speech (tekstin äänen lukijat).

Ominaisuuksia taas on lukuisia, mutta responsiivisessa verkkosuunnittelussa tärkeimmät ovat lähinnä päätelaitteen näytön resoluutioon liittyvät. Yhdistämällä mediasäännön arvon näyttö ja ominaisuus maksimileveys saadaan aikaiseksi mediakysely tiettyä näyttökokoa pienemmille laitteille. Näitä mediakyselyitä voidaan luoda useita eri resoluutiolla varustettuja näyttöjä varten. Rajoja, joissa eri mediakyselyn säännöt astuvat voimaan, kutsutaan breakpointeiksi eli murtumakohdiksi. Breakpointtien avulla pidetään huolta, että sivuston rakenne säilyy ehjänä kaikilla resoluutioilla, riippumatta näyttöpäätteestä. Yksinkertaistettuna breakpoint on siis pikselimääräinen näytön leveyden arvo.

Mediakyselyjen avulla voidaan verkkosivusto helposti suunnitella eri kokoisille laitteille. Luomalla erikokoisille näytöille omat tyylisäännöt voidaan asemoida elementit tarvittaessa uudelleen ja saada hyvä käyttöliittymä riippumatta laitteesta. Yleensä tämä toteutetaan luomalla neljä ryhmää: small (puhelin/tabletti),

medium (tabletti/kannettava), desktop (kannettava/tietokone) ja large (suuri näyttö). Ryhmillä on hieman päällekkäisyyksiä, koska laitetyyppi ei kuitenkaan määritä näytön kokoa. Nämä neljä ryhmää muodostavat neljä erilaista ulkoasua, jolloin saadaan jokaiselle laitteelle erinomainen käyttöliittymä. Näitä ryhmiä hyödynnetään CSS-tyyliohjeissa fluid grid -sarakkeille eli määritetään kerralla jokaisen laitteen koko. Alla on esimerkkinä yksi tapa käyttää mediakyselyitä.

```
<link rel="stylesheet" media="screen and (max-width: 768px)"
href="small.css" />

@media screen and (max-width: 768px) {}
```

Ensimmäinen on normaali CSS-tyyliohjeen linkitys HTML-dokumenttiin, mutta siihen on lisätty mediasääntö. Tämä rajoittaa small.css-tyyliohjeen latauksen vain, jos näytön resoluution on enintään 768 pikseliä. Jälkimmäistä syntaksia käytetään CSS-tyyliohjeiden sisällä.

3.2 RESS

RESS eli Responsive Design + Server Side Components on niin sanottu hybridi ratkaisu. RESS sisältää responsiivisen suunnittelun periaatteet, mutta näiden lisäksi se sisältää palvelinpuolen ominaisuuksia. Palvelinpuolen ominaisuuksien tavoitteena on parantaa mobiililaitteiden käytettävyyttä. Tämä ei ole kovin yllättävää, sillä käsitteen loi Luke Wroblewski, joka on myös sama henkilö kuin mobile first -käsitteen kehittäjä. Hän on ajanut mobiililähtöisen verkkosuunnittelun linjaa jo pitkään.

RESS:n ominaisuuksiin kuuluu palvelinpuolen laitetunnistus, jonka perusteella voidaan tarjota jopa kokonaan eri HTML-tiedostot eri laitteille. Laitetunnistuksen lisäksi RESS:n tarkoituksena on tunnistaa käyttäjän selaimen tukemat ominaisuudet, minkä avulla tarjotaan varmasti toimivat sivut. Nopean HTML5-tuen yleistymisen myötä on kuitenkin moni RESS:n eduista käynyt turhaksi.

RESS ei muutenkaan kerennyt ikinä yleistymään, koska siinä on niin paljon epävarmuuksia. User agentin tunnistaminen voi epäonnistua ja selainten ominaisuuksien tunnistaminen vaatii kolmannen osapuolen palvelun ostamista,

jotta tiedot pysyvät ajan tasalla. Yleisesti RESS-sivuston kehittäminen on huomattavasti monimutkaisempaa eli kalliimpaa, koska se vaati erillisen palvelinpuolen kehityksen ja vaihtoehtoisia HTML-dokumentteja eri laitteille.

4 KÄYTETYT TYÖKALUT JA TEKNOLOGIAT

Responsiivinen verkkosuunnittelu on vain tapa toteuttaa verkkosivuja, minkä vuoksi se ei vaadi mitään ylimääräistä teknologiaa tai liitännäistä. Pelkkä HTML ja CSS riittävät toteutukseen, mutta JavaScript on hyvä apu ja välttämätön esitellä, koska sitä käytetään nykyaikana käytännössä jokaisella verkkosivustolla. HTML5 ja CSS3 toivat mukanaan paljon uusia ominaisuuksia, mutta ainoastaan muutamat niistä ovat välttämättömyyksiä toteutuksen kannalta, loput vain helpottavat sitä. Kuitenkin kaikki eniten käytetyt selaimet tukevat jo tarvittavia ominaisuuksia (Can I use... 2017, hakusanoja: em, media jne).

4.1 HTML

Hypertext Markup Languagen (HTML) alun perin kehitti Sir Tim Berners-Lee. Ensimmäisen kuvauksen hän esitteli jo vuonna 1991, mutta virallinen version Internet Engineering Task Forcen (IETF) julkaisi HTML-määrittelyille vasta vuonna 1993. Tämän jälkeen HTML:n kehittäjinä ovat toimineet monet eri tahot, mikä on osittain hidastanut HTML:n kehitystä, koska kehityksen suunnan päämäärästä ei ole päästy yksimielisyyteen. Tällä hetkellä uusin versio on HTML5 ja se on W3C:n ylläpitämä. HTML5 julkaistiin loppuvuodesta 2014 ja se toi paljon uudistuksia, koska edellinen versio HTML 4.01 oli vuodelta 2000. (Wikipedia The Free Encyclopedia 2017, hakusana HTML.)

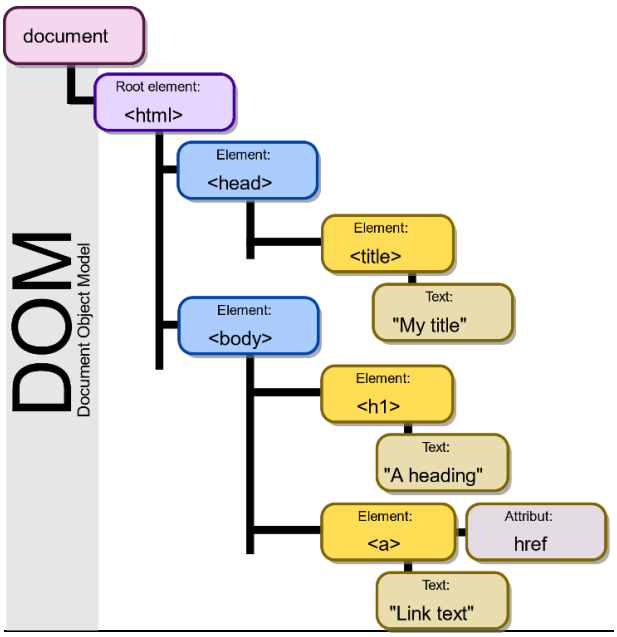
HTML-kieli on merkintäkieli. Merkintäkielissä dokumentin alussa julistetaan kieli, jotta jäsentimet osaavat tulkita tagit eli tunnisteet oikein. HTML:ssä on ennalta määritelty esityssemantiikka eli etukäteen määritellyt tunnisteet. Merkintäkielien pääideana on erottaa tekstin looginen rakenne sisällöstä. HTML on vastuussa sisällön asettelusta sivulle, mutta se ei määritä elementtien ulkoasua tai toiminnallisuutta. HTML-rakennnetta kuvattaessa usein käytetään termejä parent eli vanhempi ja child eli lapsi. Näillä viitataan elementtien rakenteelliseen järjestykseen. HTML-elementtiä, jonka sisällä on toinen HTML-elementti, kutsutaan parentiksi ja sisällä olevaa elementtiä lapsielementiksi.

HTML-dokumentti perustuu melko yksinkertaiseen syntaksiin: tunnisteisiin ja niiden sisältämiin attribuutteihin, merkkipohjaisiin tietotyyppeihin sekä merkkiviitteisiin. HTML-elementit muodostuvat, kun HTML-tulkki lukee elementin aloitustunnisteen ja sen ominaisuuksia, kunnes tulee lopetustunniste esimerkiksi:

```
<a href=”https://www.google.com/”> Google </a>
```

Merkkipohjaiset tietotyypit mahdollistavat suoraan tekstin syöttämisen, sen sijaan että tieto olisi bitteinä. Merkkiviitteillä taas tarkoitetaan kirjoitusmerkin sisällyttämistä, mikä ei sisälly koodattaessa käytettävää merkistöön. Esimerkiksi ASCII-merkistö ei sisällä ä-kirjainta, mutta se voidaan kirjoittaa käyttämällä &auml-merkkiviittausta. Tosin UTF-8 merkistön tuen yleistyttyä ominaisuuden tarve on vähentynyt, mutta se on edelleen helppo tapa tehdä erikoismerkkejä, kuten suuntanuolia tai valikko-ikoni.

HTML-sivua käsitellään yleensä DOM-ohjelmointirajapinnan avulla. DOM on alusta- ja kieliriippumaton rajapinta, joka käsittelee HTML-, XHTML- tai XML-dokumenttia puuna. Puussa jokainen solmu on olio. Nämä solmut kuvastavat yhtä dokumentin osaa. Näin verkkosivua on helppo muokata ohjelmallisesti eli usein JavaScriptillä, selaten läpi elementtejä ja valitsemalla halutun. (Kuva 5.)



KUVA 5. DOM-puurakenne (Wikipedia The Free Encyclopedia 2017, hakusana DOM)

4.2 HTML5

Responsiivisen kehityksen avuksi HTML5 toi viewport meta-tagin. Se on ainoa välttämättömyys HTML5-tuesta, koska sillä otetaan käyttöön näkymän leveydeksi näytön loogiset pikselit ja poistetaan mahdollinen selaimen perusskaalauskerroin käytöstä. HTML5 kuitenkin toi uusia elementtejä, joiden avulla HTML-koodista saa yksinkertaisempaa ja selkeämpää. Alla muutamia esimerkkejä näistä elementeistä, jotka varmaan löytyvät melkein jokaiselta HTML5 verkkosivustolta.

```
<footer>  Määrittelee footerin dokumentille

<header>  Määrittelee headerin dokumentille

<main>    Määrittelee pääsisällön dokumentille

<nav>     Määrittelee navigaatiomenun dokumentille
```

HTML5:n uusista ominaisuuksista tämän lisäksi käytetyimpiä ovat natiivi video- ja audiotuki. Tämän ansiosta Flashiä tai muita kolmannen osapuolen mediasoittimia ei enää tarvita. Videollekin voi määrittää useamman eri lähteen kuten kuvalle sekä se tukee venyttämistä ja pienentämistä. Käyttäjän asetuksien tallentamiseen ja mainostukseen tuli erittäin käytetty ominaisuus local storage eli paikallinen muisti. Tämä on käytännössä paranneltu versio cookiesta, mutta se sallii tiedon käsittelyn eri ikkunoista toisin kuin cookiessa. Myös tietoturvaa sekä suorituskykyä on parannettu verrattuna cookieen. Talletettu tieto säilyy paikallisessa muistissa, vaikka selaimen sulkisi.

4.3 JavaScript

JavaScript kehitettiin HTML:n ja CSS:n lisäksi tuomaan toiminnallisuutta verkkosivuille. Nykyään siitä on olemassa myös palvelinpuolen sovelluksensa. JavaScriptin käyttö ei rajoitu vain verkkosivuille, vaan sitä käytetään myös peleissä ja sovelluksissa. JavaScriptin standardoitu versio tunnetaan nimellä ECMAScript, josta uusin versio tällä hetkellä on 7. painos. Se on yleisemmin tunnettu nimellä ECMAScript 2016. ECMA lyhenne tulee European Computer Manufacturers Association eli eurooppalaisten tietokonevalmistajien yhdistys.

JavaScript on korkean tason oliopohjainen tulkattava skriptikieli. Semantiikaltaan se on englannin kielen kaltaista, mikä on tehnyt JavaScriptistä helpon kielen oppia ja ymmärtää. Javascriptin oppimista ja koodausta myös helpottaa dynaaminen tyypitys eli se että ajonaikana muuttujien tyypit voivat vaihtua. Tulkattavissa kielissä koodia ei erikseen käännetä ohjelmaksi ennen suoritusta, mutta ne kuitenkin vaativat tulkin koodin suorittamiseen. JavaScript-koodi on alustariippumatonta ja tulkki suorittaa JavaScript-koodia rivin kerrallaan. JavaScript-koodia voi kirjoittaa erilliseen tiedostoon tai suoraan HTML-tiedostoon, mitä tosin ei suositella.

Verkkosivujen JavaScript-koodi suoritetaan käyttäjän selaimessa, mikä tekee siitä erittäin nopeaa, koska tällöin ei tarvitse odottaa vastausta palvelimelta. Tämä myös tarkoittaa vähemmän liikennettä selaimen ja palvelimen välille, mikä säästää verkkokaistaa ja nopeuttaa käyttöliittymää. Asiakaspuolisuudessa on myös haittapuolensa. JavaScript-koodi on silloin käyttäjälle näkyvää ja avointa muokattavaksi. Tämän lisäksi saastuneet tai muuten pahantahtoiset sivustot voivat suorittaa mahdollisesti haitallista koodia käyttäjän omalla koneella, kaikista selaimen turvarajoituksista huolimatta.

Yleisimmät ominaisuudet, joita JavaScriptillä toteutetaan, ovat verkkosivun näkymän muokkaus suoraan käyttäjän selaimessa eli jo olemassa olevien HTML-elementtien tyylien muokkaus tai uusien elementtien luominen. Joillakin verkkosivustoilla osan palvelinpuolen tehtävistä voi toteuttaa asiakaspuolisella JavaScriptillä, näin vähennetään verkkokaistan kulutusta, mikä nopeuttaa sivuston käyttöä ja säästää rahaa palvelinkustannuksissa.

4.3.1 Unobtrusive JavaScript

Unobtrusive JavaScript eli ei-häiritsevä JavaScript on tapa toteuttaa JavaScriptiä siten, että sivusto säilyy edelleen käytettävänä, vaikka käyttäjän selain ei tukisi täysin kaikkia vaadittuja JavaScript-ominaisuuksia. Käsitettä ei ole miten-kään virallisesti määritelty, mutta kolme periaatetta yleensä yhdistetään siihen:

1. Toiminnallisuuden erotus
2. ”Parhaat käytännöt”

3. Esteettömyyden periaate.

Toiminnallisuuden erotuksella tarkoitetaan JavaScriptin erottamista HTML-koodista.

```
// Huono tapa

<input type="text" name="date" onchange="validateDate()" />

// Hyvä tapa

<input type="text" name="date" id="date" />

window.onload = function() {
  document.getElementById('date').onchange = validateDate; };
```

Erona tässä on, että event handler eli tapahtuman käsittelijä rekisteröidään vasta JavaScriptin puolella. Tämän ansiosta, vaikka selain ei tukisi tarvittavaa Javascript-ominaisuutta, niin HTML-dokumentti kuitenkin latautuu. Huonolla tavalla taas selain jumittuu tuntemattomaan komentoon eikä sivusto lataudu kokonaan.

Parhaiden käytäntöjen noudattaminen tarkoittaa nykyaikana lähinnä pysymistä ECMAScript-standardissa, jolloin välttyy selainten välisiltä ristiriidoilta. Tämän lisäksi se myös tarkoittaa yleisesti ohjelmoinnin hyviä käytäntöjä, jotka koskevat kaikkia muitakin ohjelmointikieliä.

Esteettömyyden periaatteella tarkoitetaan JavaScriptin toteutusta siten, että Javascriptin olemassaolo ei aiheuta haittaa HTML-koodin tulkintaan. Tällä tarkoitetaan, että HTML-dokumentti pitäisi pystyä aina lataamaan, vaikka JavaScript-tuki ei käyttäjän laitteella olisi riittävä. JavaScriptillä on tarkoitus parantaa käyttöliittymää, mutta ei rikkoa sitä.

Nämä periaatteet ovat helpottaneet JavaScript-kirjastojen syntyä, koska niitä noudattaessa koodi ei ole enää vain kyseiseen verkkosivustoon sidottua, vaan siitä tulee uudelleen käytettävää.

4.3.2 Frameworkit ja kirjastot

JavaScriptistä on tullut käytännössä de facto -standardi verkkosivujen interaktiivisuuden toteutuksessa, joten hyödyllisiä JavaScript-kirjastoja on toteutettu tuhansia. Valmiiden JavaScript-kirjastojen avulla koodin kirjoitus on nopeampaa ja helpompaa. Omassa työssäni käytin seuraavia kirjastoja:

1. jQuery
2. Prefix free
3. HTML5 Shiv
4. Selectivizr.

jQuery on nopea, pieni ja monipuolinen JavaScript-kirjasto. Sen avulla HTML-dokumentin läpikäynti ja käsittely, eventtien hallinta, animaatioiden luonti ja Ajax-kyselyt ovat yksinkertaisempia toteuttaa. Se tarjoaa API:n (Application Programming Interface) eli ohjelmointirajapinnan, jonka avulla ohjelmoija voi yksinkertaisilla käskyillä toteuttaa laajempia toiminnallisuuksia tarvitsematta tietää, miten kirjasto itse toteuttaa asian. jQuery toimii kaikilla selaimilla. (The jQuery Foundation 2017)

jQueryn syntaksi on suunniteltu valitsemaan HTML-elementtejä ja suorittamaan haluttu toiminto niille. Perussyntaksi noudattaa seuraavaa mallia.

```
$(valitsin).toiminto()
```

- $-merkki viittaa jQuery-kirjaston käyttämiseen.
- valitsin on määritys halutulle HTML-elementille. Valitsimet noudattavat CSS-syntaksin sääntöjä. Valitsimista lisää seuraavassa luvussa.
- toiminto() kuvaa haluttua metodia, joka valitulle elementille suoritetaan.

Prefix free on kirjasto, joka automaattisesti lisää CSS-tyyleihin käyttäjän selaimen tarvitsemat prefixit eli etuliitteet. Se lisää prefixit siis vain tarvittaessa. Tämän vuoksi CSS-tyylitiedoston kirjoittamisesta tulee yksinkertaisempaa ja tyylitiedoston koosta tulee pienempi, mikä säästää verkkokaistaa. Prefixien merkitystä käsitellään tarkemmin CSS-luvussa.

HTML5 Shiv on kirjasto, joka toteuttaa JavaScriptin avulla HTML5-elementit vanhoille selaimille kuten Internet Explorer 6–9, Safari 4.x, Iphone 3.x ja Firefox 3.x. Kirjastosta on kaksi versiota, perusversio ja kattavampi versio. Kattavampi versio mahdollistaa myös elementtien tyylittelyn piirron aikana Internet Explorerissa.

Selectivizr pyrkii JavaScriptin avulla jäljittelemään CSS3:n pseudo-luokka- ja attribuuttivalitsimia Internet Explorer 6–8:ssa. Tämän avulla pystyy toteuttamaan myös CSS3-pohjaiset tyylitykset vanhoissa Internet Explorereissa, jotka eivät muuten tukisi CSS3-tyylejä. Selectivizr kuitenkin vaatii lisäkseen jonkin seitsemästä sen tukemasta JavaScript-kirjastosta avukseen. Nämä ovat jQuery, Dojo Toolkit, Prototype, The Yahoo! User Interface Library (YUI), DOMAssistant, MooTools ja NWMatcher.

4.4 CSS

CSS on tyyliohjeiden laji, joka on pääasiassa tarkoitettu HTML- ja XML-dokumenttien muotoiluun. Merkkauskielten erot eivät kuitenkaan vaikuta mitenkään CSS-tyyliohjeiden muotoiluun, mutta elementtien oletusmuotoilut saattavat olla erilaiset. Oletusmuotoilulla tarkoitetaan ulkoasua, joka piirretään elementille, jonka tyyliä ei ole erikseen määritetty. CSS:n tarkoitus on erottaa dokumentin sisältö sen esitystavasta. Tämä vähentää toistamista HTML-koodissa ja yksinkertaistaa dokumenttia, koska tyylit voi siirtää kokonaan eri tiedostoon. Samalla se tekee mahdolliseksi helpot teemojen luonnit ja erilaiset esitystavat eri laitteille.

CSS:n keskeisimpiä periaatteita ovat kaskadi (cascade) ja periytyvyys (inheritance). Kaskadilla CSS-tyyliohjeissa tarkoitetaan, että kun on useampi kuin yksi tyyliohje, joka voidaan soveltaa tiettyyn osaan HTML-dokumenttia, niin on kuitenkin sääntö, joka määrittää voimaan jäävän tyyliohjeen. Sääntö noudattaa kaskadi-periaatetta, eli yleisemmästä säännöstä lähdetään poissulkemaan kohti tarkempia sääntöjä, kunnes vain tarkin sääntö jää jäljelle. Periytyvyydellä tarkoitetaan tilannetta, jossa lapsielementti perii ulkoasuun liittyvät ominaisuudet parentilta. Silloin kun tyyliohjeissa asetetaan ominaisuudelle jokin arvo, se ei peri sitä parentilta, jollei näin ole erikseen määritetty kyseisen ominaisuuden arvoksi.

Kaikki ominaisuudet eivät kuitenkaan ole periytyviä, sekä esimerkiksi taulukot ja taulukkoelementit perivät vain toisilta vastaavilta elementeiltä ominaisuuksia. Periytyminen vähentää huomattavasti tyylien toistamisen tarvetta, mikä pienentää CSS-tyylitiedostojen kokoa ja näin saadaan nopeammin latautuvat verkkosivut ja helpommin ylläpidettävää koodia.

CSS:n syntaksi koostuu valitsimesta, ominaisuudesta ja arvosta. Valitsin voi olla elementti, elementin id tai luokka. Sen lisäksi valitsin voi sisältää niin sanotun pseudo-luokan ja valitsimia voi yhdistellä keskenään luomaan mielekkäitä toiminnallisuuksia. Pseudo-luokat liittyvät valitsimeen kaksoispisteellä ja ne kuvastavat elementin poikkeavaa tilaa. Tämän avulla niillä voi tehdä helposti interaktiivisia tyylimuutoksia.

```
valitsin {
ominaisuus:arvo;
}
```

```
li:hover > li {
background-color:red;
}
```

Yllä oleva koodi valitsee kaikki li-elementit, joiden parentin päällä kursori sijaitsee. Eri CSS tasot tukevat erilaisia valitsimia, sivustoa kehittäessä kannattaa huomioida eri selaimien kyvykkyys.

Tämän lisäksi CSS-ominaisuuden eteen voidaan lisätä prefix eli etuliite. Prefixien tarkoitus on pystyä toteuttamaan ominaisuuksia, jotka ovat vielä kehityksessä. Niillä tuetaan CSS-ominaisuutta, jota ei ole vielä standardoitu, vaan valmistajat tekevät kokeellisia tai omia toteutuksia tulevalle ominaisuudelle. Prefixejä käyttämällä saadaan laajempi ominaisuuksien määrä käyttöön. Mahdollisia prefixejä ovat

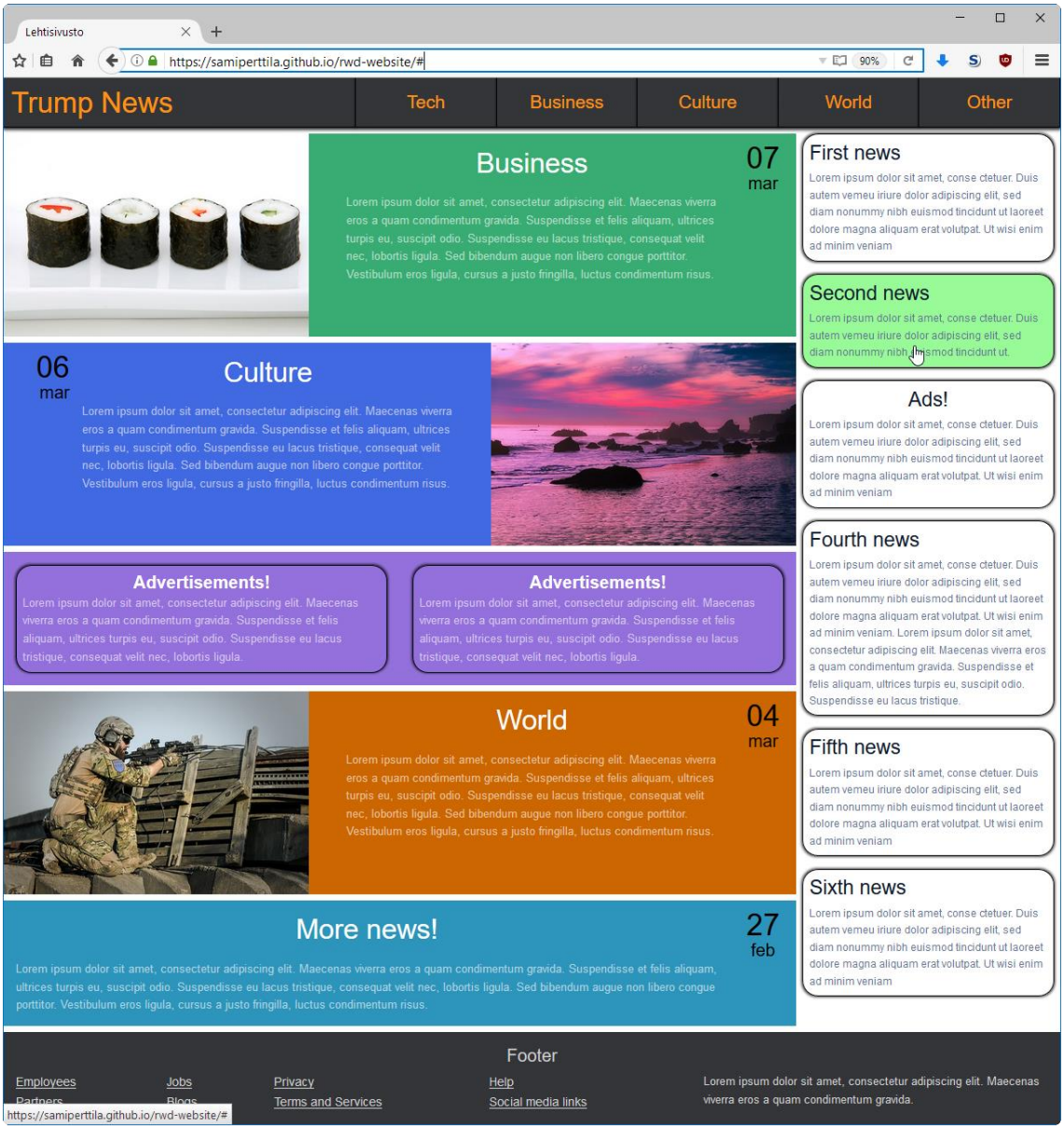
- -webkit- Chrome, Safari, Opera 14 ja uudemmat versiot
- -moz Firefox
- -o- Vanhemmat kuin Opera 14 versiot
- -ms- Internet Explorer.

World Wide Web Consortium (W3C) ylläpitää CSS-kielen määritelmiä. Uusin määritelmä on tällä hetkellä CSS taso 3. Tästä on tullut nimitys CSS3, joka yleisesti tarkoittaa kaikkia uusia ominaisuuksia CSS 2.1 -määrittelyn jälkeen.

CSS3:a kehitetään modulaarisesti, eikä se ole yksi täysi kokonaisuus. Modulaarisen kehityskaaren vuoksi luultavasti CSS4-versiota ei koskaan tulla julkaisemaan, vaan CSS:n tason 4 moduulit. CSS3 toi mukanaan paljon uusia ominaisuuksia, joista mediasääntöjen uudet ominaisuudet ovat responsiivisen suunnittelun kannalta ratkaisevimmat.

5 TOTEUTUS

Opinnäytetyössä tein yksinkertaisen responsiivisen verkkosivun, jonka avulla esittelen responsiivisen verkkosivuston ominaisuuksia ja lähinnä käyn responsiivisen verkkosuunnittelun kannalta tärkeimmät asiat läpi toteutuksesta. Koko koodi on kuitenkin liittenä opinnäytetyön lopussa. Toteutuksen kokonaisnäkymä tulee olemaan seuraava (kuva 6).



KUVA 6. Verkkosivun kokonaiskuva tietokoneella katsottuna hieman pienettynä

Responsiivinen toteutus ei tee HTML-koodiin paljoa eroa verrattuna perinteiseen verkkosivustoon. Kuitenkin HTML-dokumentin alussa täytyy määrittää

päätelaitteen viewportin eli näkymän perusmittasuhteet ja skaalauksen kerroin. Tämä tehdään käyttämällä HTML-meta-tagia.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Meta-tagin sisällön arvo width=device-width asettaa sivun sisällön leveydeksi käytetyn laitteen näytön leveyden loogisina pikseleinä ja intial-scale=1.0 asettaa tarkennukseksi 100 % eli zoomin pois päältä. Monessa mobiiliselaimessa on perusasetuksena jo jonkin asteinen skaalauskerroin päällä, mikä sotkisi media-kyselyjen breakpointit. Lisäksi suurien staattisten elementtien käyttämistä tulee välttää, sillä niiden asemointi tuottaa ongelmia, koska ne eivät skaalaudu. Tämän vuoksi en esimerkiksi käyttänyt HTML-taulukkoja ollenkaan, koska niiden koko on aina kiinteä.

JavaScript ei ole välttämättömyys responsiivisessa toteutuksessa, joten työssäni lähinnä hyödynsin sitä vain eri selain tukien parantamiseen. Tämä tein ottamalla käyttöön kolme valmista JavaScript-kirjastoa: Prefix free, HTML5 Shiv ja Selectivizr. Kirjastot otetaan käyttöön linkkaamalla JavaScript-tiedostot HTML-dokumenttiin. Koska käyttöönotto on näin helppoa, minusta ne kannattaa laittaa melkein jokaiselle verkkosivustolle. Näiden kolmen lisäksi otin myös käyttöön jQuery-kirjaston, jonka avulla toteutin navigaatiomenun interaktiot.

Tavallisesti ennen tuotantoon laittamista JavaScript-koodit ja CSS-tyylit on tarkoitus minimoida eli pienentää niiden tiedostokoot mahdollisimman pieneksi. Koska molemmat ovat lopulta vain tekstiä, niin minimointi tapahtuu minimoimalla merkkien määrä tiedostoissa. Tähän on olemassa valmiita työkaluja kuten https://cssminifier.com/ ja https://jscompress.com/. En kuitenkaan työssäni käyttänyt kumpaakaan, koska halusin pitää koodit luettavana, minimointi kun poistaa kaikki rivitykset, ylimääräiset välilyönnit, sisennykset jne.

Päätoteutustapa työssäni oli niin sanottu column-drop-strategia, mikä käytännössä tarkoittaa elementtien kelluttamista vierekkäin. Pudotus tehdään asettamalla float-ominaisuuden arvo tyhjäksi, jolloin vierekkäin olleet elementit siirtyvät allekkain ja block-ominaisuuden ansiosta ovat 100 % leveitä, jollei toisin ole määrittänyt. Tällä tavalla saa erittäin helposti toteutettua mobiilinäkymän ja

muut näkymät. Mediakyselyjen avulla elementtien koko muutetaan erikokoiseksi laitteen mukaan, jotta saadaan viereen toinen elementti ja isommalla näytöllä pinta-ala hyödynnettyä paremmin.

5.1 Ulkoasu

Kuva-elementtejä käsittelin kahdella tavalla. Ensiksi tein niistä skaalautuvia määrittämällä tyyliksi

```
img{ height:auto;  display:block;  max-width:100%; }
```

Näin käyttäjän selain skaalaa kuvat automaattisesti varattuun tilaan sopiviksi säilyttäen kuvasuhteen, mutta ei kuitenkaan ala venyttää kuvaa alkuperäistä isommaksi. Tämän lisäksi kuvien tarjoamisessa käytin srcset-attribuuttia, jonka avulla pystyy tarjoamaan eri kuvan eri päätelaitteelle. Näin voin tarjota pienemmän kuvan mobiilikäyttäjälle kuin tietokoneenkäyttäjälle. Tämä säästää verkkokaistaa sekä mobiilikäyttäjän latausaika nopeutuu huimasti, kun kuvan koko voi olla jopa viisi kertaa pienempi. Yksinkertaisin tapa käyttää scrset-attribuuttia on listata siihen sijoitettavat kuvat ja lisätä jokaisen perään pikselimääräinen leveys, milloin uusi parempilaatuinen kuva tulee edeltävän tilalle. Käyttäjän selain valitsee automaattisesti oikean kuvan sijoitettavaksi.

```
<img srcset="Images/malibu-2080075_280.jpg 280w, Images/malibu-
2080075_320.jpg 320w, Images/malibu-2080075_360.jpg 360w"
alt="Not found">
```

Responsiivisessa suunnittelussa ulkoasun uudelleenasemointi tehdään mediakyselyjen avulla. Mediakyselyjen avulla määritän uudet tyylit, jotka tulevat myös silloin voimaan. Koska CSS-tyylit ovat kaskadisia, aluksi loin yleiset tyylit, jotka pätevät kaikille ulkoasuille, riippumatta onko päätelaitteena pieni vai iso laite. Tämän jälkeen lähdin suunnittelemaan ulkoasun toteutusta pienimmästä laitteesta suurimpaan mediakyselyjen avulla. Tämä säästää tyylisääntöjen turhaa uudelleenylikirjoitusta.

Pienissä päätelaitteissa usein käyttöliittymänä on kosketusnäyttö. Usein CSS-tyyleissä käytetään hover-pseudoluokkaa luomaan toiminnallisuuksia, kuten pu-

dotusvalikoita tai yksinkertaisesti värjätään valittu elementti, mutta kosketusnäyttöpohjaisissa laitteissa se ei tietenkään toimi. Tämän ongelman voi korjata käytännössä kaikissa tapauksissa lisäämällä myös active-valitsimen. Huomioitavaa tässä on kuitenkin, että active-valitsin ei saa tulla CSS-koodissa ennen hover-valitsinta tai se ei toimi.

```
.top-ul li:hover > a, .top-ul li:active > a {
  color:#ff1e1e;  background-color:#283D5F; }
```

5.2 Näkymät

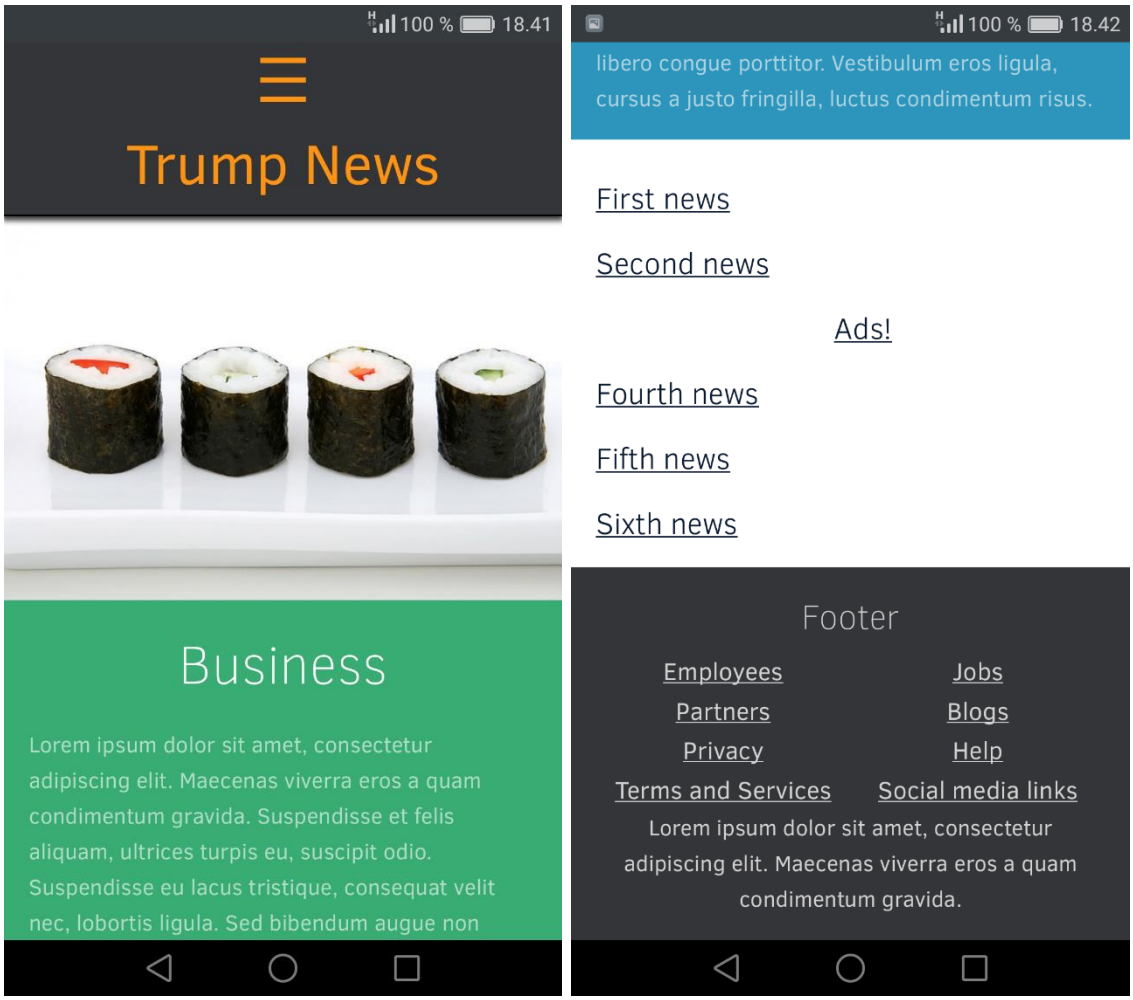
Tässä luvussa esittelen eri näkymät työssäni pienemmästä isompaan ja esittelen merkittävimmät koodit verrattuna seuraavaan näkymään. Lähinnä nostan esille koodit, joilla on responsiivisen suunnittelun kannalta. Eri näkymät vaihtuvat mediakyselyjen breakpointeissa. Näiksi pisteiksi valitsin seuraavat:

```
@media screen and (max-width:480px){}   // Small (puhelimet)

@media screen and (max-width:768px){}   // Medium

@media screen and (min-width:769px){}   // Desktop

@media screen and (min-width:1367px){}  // Large
```

Näiden breakpointtien lisäksi olisi small- ja medium-ryhmän voinut jakaa aliryhmiin niiden päällekkäisyyksien vuoksi, mutta koska tämä toteutus toimii vain esimerkkinä, niin en nähnyt tarvetta sille. Edetessä laitteesta isompaan, breakpoint-kohdissa tulee aina uusia tyylisääntöjä voimaan ja mahdollisesti myös edellisen kumoavia sääntöjä. Näin saan muokattua näkymää eri päätelaitteelle sopivaksi.

5.2.1 Small-näkymä

Small-näkymä on lähinnä älypuhelimille suunniteltu näkymä, jonka leveys rajoittuu enintään 480 pikseliin. Seuraavaksi kaksi esimerkkiä verkkosivun ulkoasusta mobiililaitteella (kuva 7).



KUVA 7. Kuvankaappaukset sivuston alusta ja lopusta. Päätelaitteena on Huawei Honor 7 Lite 5,2". Kuvia on huomattavasti pienennetty

Tässä ulkoasussa navigointimenu menee piiloon ja on koko ruudun levyinen pudotusmenu. Navigointimenun sijoituksen tein CSS-tyylien avulla, kun taas toiminnallisuuden JavaScript-koodilla, jossa käytän hyväksi jQuery-kirjastoa työn helpottamiseksi.

```
.top-nav > ul { height: 0; width: 100%; overflow: hidden;
  position: relative; z-index: 999;}

.top-nav > ul.show-menu {height: auto;}

.top-nav .top-ul ul {position: relative;}

.top-nav .top-ul li {float: none;}

.top-nav > ul ul.show-ul {display: block; height: auto;}
```

Erityisesti pitää huomiota määrittää overflow: hidden navigointimenulle, jotta se leikkaa ylimenevän sisällön pois eikä yllä oleva elementti varaa koko tilaa. z-indexin avulla voidaan määrittää taso, jolle piirretään. Mitä suurempi arvo, sitä korkeammalle tasolle piirretään, eli kahdesta päällekkäisestä elementistä suuremmalla z-indexillä oleva elementti jää näkyviin. Tällä varmistan, että menu piirtyy kaiken muun päälle.

Toiminnallisuuden toteutus on kaksiosainen. Ensimmäinen osa hallitsee valikko-ikonia eli navigaatiomenun alaspudotusta.

```
// jQuery:

$(‘.nav-text’).click(function() {
  $(‘.top-nav > ul’).toggleClass(‘show-menu’);
  $('.top-nav > ul li.submenu > ul').removeClass('show-ul');
})

// CSS:

.show-menu {height: auto;}

.show-ul { display: block; height: auto; }
```

Tämä liittää ikoniin click-eventin. Ikonia klikatessa funktio lisää navigaatiomenulle luokan show-menu, joka tekee menun näkyväksi. Toteutuksessa hyväksikäytän kuitenkin toggleClass-funktiota, jolloin uudelleen painaessa se poistaa menulta aikaisemmin lisätyn show-menu-luokan. Näin saadaan navigaatio menu taas piilotettua. Tämän lisäksi piilotettu menu sulkee kaikki mahdollisesti auki olevat sisämenut.

Toinen osuus hallinnoi navigaatiomenun sisällä olevia sisämenuja. Normaalisti ne ovat suljettuna, mutta klikatessa sisämenu pitää aueta suoraan alapuolelle.

```
// jQuery:

$('.top-nav > ul li:has(ul)').addClass('submenu');

$('.top-nav > ul li.submenu > a').click(function () {
  $('.top-nav > ul li.submenu > ul').removeClass('show-
ul');
```

```
  $('.top-nav > ul li.submenu:hover > ul').tog-
gleClass('show-ul');
});

// CSS:

.show-ula {display: block; height: auto; }
```

Tällä jQuery-koodilla käyn läpi kaikki navigaatiomenun li-elementit, joilla on sisämenu. Niille li-elementeille, joilta sisämenu löytyy, lisään luokaksi submenu helpompaa käsittelyä varten. Sen jälkeen luon submenun ankkureille click-eventin, jolla vastaavalla tavalla kuin aikaisemmin näytän ja suljen sen.

Sivuston ulkoasua on kuitenkin tähän mennessä määritelty pääosin yhdellä luokalla eli fluid gridistä tutulla s-12-tyylillä. Tämän avulla kaikista elementeistä tulee 100 % leveitä ja ne asettuvat nätisti allekkain. Alla esimerkkinä sivuston artikkelin kuvan HTML-koodin määritys. Siitä kuitenkin huomaa, kuinka suhteellinen koko muuttuu isommilla näytöillä.

```
<div class="s-12 d-5 l-4 post-image"></div>
```

Navigointimenun lisäksi päätin jälkikäteen tehdä muutoksia myös aside-elementtiin eli ns. lisäuutispalkkiin. HTML-rakenteen ansiosta tämä uutispalkki siirtyy automaattisesti alimmaiseksi, olettaen, että footeria ei huomioida. Halusin kuitenkin vähentää tarvittavaa alaspäin vieritystä, joten poistin näkymästä uutispalkin tekstikentät ja jätin vain uutisten otsikot linkkeinä näkyviin. Tämä oli hyvin yksinkertainen muutos tyyleihin, mutta käytettävyys parani huomattavasti.

```
.aside-block p {display:none;}
```

5.2.2 Medium-näkymä

Medium-näkymä on suunnitteltu lähinnä tableteille ja tavallista pienemmille kannettaville tietokoneille. Kuvassa 8 esittelen verkkosivun ulkoasun tabletilta.

KUVA 8. Apple Ipad Pron pystyasennon kuvankaappaus pienennettynä

Medium-näkymä on viimeinen näkymä, jossa käytetään navigaatiomenun pudotusvalikkoa, joten tässä luodaan pudotusmenun hampurilaisikoni. Ikonin voi luoda kopioimalla ja liittämällä sen jostakin tai käyttämällä merkkiviittausta.

```
.nav-text:after {content: "\2630";}
```

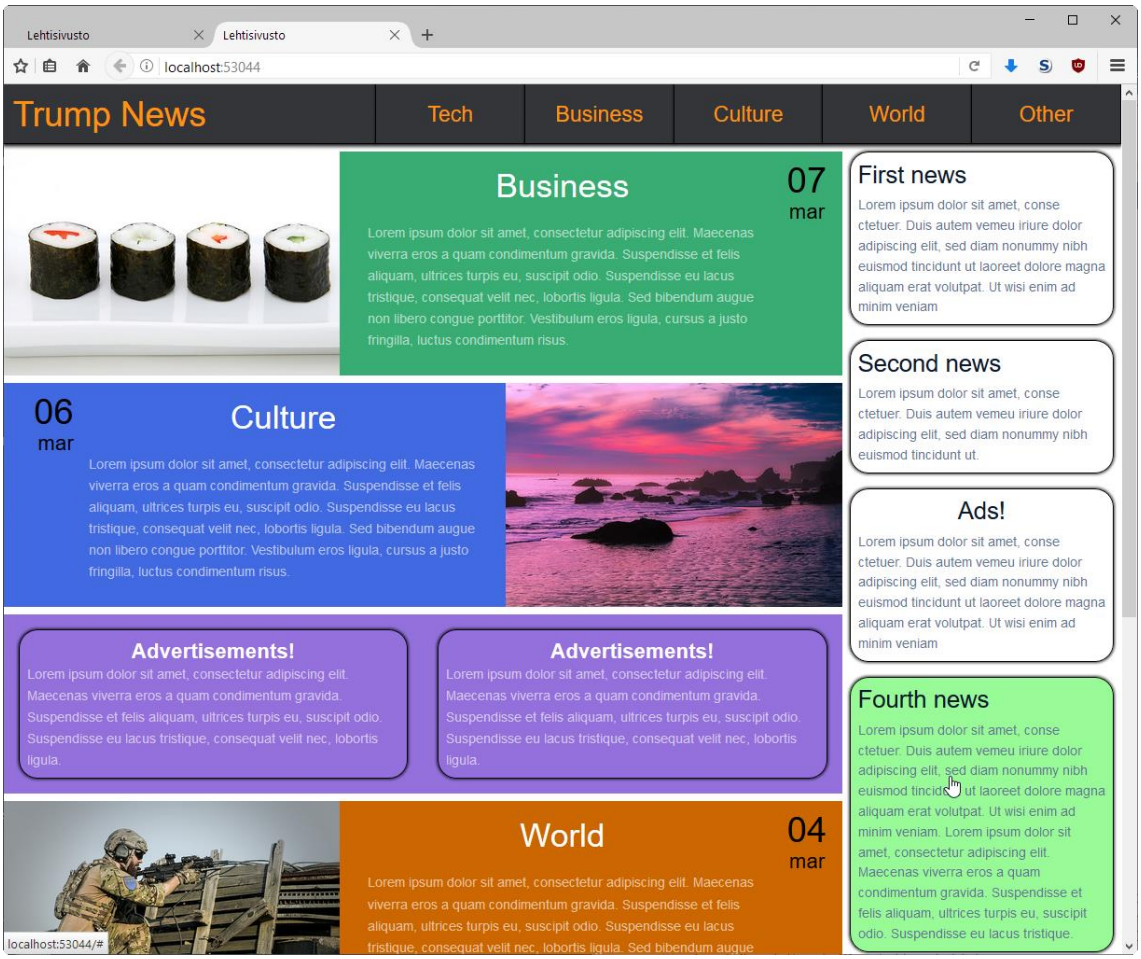
Muuten näkymä ei juuri poikkea small-näkymästä, mutta ylimääräinen leveys on otettu hyötykäyttöön aside-elementille. Tämä toteutettiin yksinkertaisesti fluid gridin avulla:

```
<div class="s-12 m-7 d-9 l-9"></div>

<div class="s-12 m-5 d-3"></div>
```

Kaikki elementit eivät ole enää 100 % leveitä, vaan artikkelin leveys on 7/12 ja uutispalkin leveys on 5/12.

5.2.3 Desktop-näkymä

Desktop-näkymä eli tietokoneille suunniteltu täysikokoinen näkymä pyrkii käyttämään myös leveys suunnan mahdollisimman tehokkaasti. Kuvassa 9 on näkymä ulkoasusta tietokoneelta.



KUVA 9. Kuvankaappaus tietokoneelta desktop-näkymästä pienennettynä

Tietokonenäkymään edetessä navigointimenu muuttuu horisontaaliseksi ja on aina näkyvissä eli hampurilais-ikoni piilotetaan. Navigointimenusta tehdään horisontaalinen kelluttamalla. Tämän lisäksi sisämenujen pudotusominaisuudet vaativat muutamia tyylimuutoksia.

```
.top-ul li {float:left;}

.top-ul > li {position:relative;}

.top-ul ul {position:absolute;}

.top-ul li ul {display:none; width:100%;}

.top-ul li ul li {float:none; width:100%;}
```

Määritetään navigaatiomenun li-elementtien sijainti suhteelliseksi ja sisämenun sijainti absoluuttiseksi, minkä avulla saadaan putoava sisämenu sijoitettua suhteellisesti parenttiin. Näin sisämenut voidaan helposti piirtää oikeaan kohtaan. Tämän lisäksi alkuarvoina piilotetaan sisämenut.

Näytön koko on taas kasvanut tarpeeksi, joten siirretään artikkeleissa oleva kuva tekstin viereen ja näin saadaan näkymä tehokkaammin hyötykäytettyä. Tämäkin toteutetaan vain fluid gridiä käyttämällä eli määrittämällä suhteellinen leveys artikkeli-elementin sisällä.

```
<div class="s-12 d-5 l-4 post-image"></div>

<div class="s-12 d-6 l-7 post-text"></div>

<div class="s-12 m-12 d-1 l-1 post-date"></div>
```

5.2.4 Large-näkymä

Suurille näytöille näkymä on lähes identtinen desktop-näkymän kanssa. Ainoastaan fluid gridin arvoja vain muokkasin. Tämä tein parantamaan ulkoasua, jotta esimerkiksi artikkelin kuva ei veny turhan isoksi ja artikkelin tekstille olisi enemmän tilaa. Tosin uutissivustolle ei välttämättä tarvitsekaan large-näkymää, koska lisätilan hyötykäyttö on ylipäänsä melko vaikeaa. Sivustolla ei saa olla kerralla liikaa tekstiä palstoittain, koska tällöin käyttäjä ei jaksa lukea kaikkea. Verkkosivuja ei lueta samalla tavalla kuin sanomalehteä, vaan luetaan vain otsikoita, joista valitaan artikkeli kerrallaan luettavaksi. Toisaalta verkkokaupoille large-näkymä on erittäin hyödyllinen, koska siinä saa enemmän tuotteita kerralla esille.

6 POHDINTA

Palveluntarjoajat tekevät verkkosivut käyttäjille, ja tämän vuoksi niiden laatu ja käytettävyys merkitsevät poikkeuksen paljon. Huonosti toteutetut verkkosivustot eivät ikinä saa käyttäjiä eivätkä täten saavuta tehtäväänsä. Tämä on verkkokehityksen suurin huomioinnin kohde, mutta surullista kyllä usein unohdetaan, että verkkosivustot tehdään käyttäjille, ei yritykselle itselleen. Responsiiviset verkkosivut tarjoavat käyttäjille erinomaisen käyttökokemuksen riippumatta siitä, käyttävätkö he tietokonetta, tablettia vai puhelinta.

Kuluttajien jatkuvasti muuttuvat laitteet aiheuttavat palveluntarjoajille paljon haasteita. Uusien verkkosivujen tuottaminen on kallista, mikä monelle pienelle yritykselle on liian suuri este päivittää verkkosivustoa. Kuitenkin samaa aikaa pitäisi pyrkiä palvelemaan koko mahdollista asiakaskuntaa. Tarkoituksena on siis tuottaa kustannustehokkaat ja mahdollisimman pitkäikäiset verkkosivut. Tehokkain tapa tässä on ylivoimaisesti responsiiviset verkkosivut yksinkertaisuutensa ja ongelmattomuutensa takia. Responsiivista verkkosuunnittelua käyttäessä ei tarvitse murehtia toistuvista päivityskustannuksista tai ylläpitää kahta erillistä verkkosivustoa.

Tämän opinnäytetyön tavoitteena on esitellä yleisimmät ratkaisut ja tuoda esille kustannustehokas ja toimiva tapa toteuttaa verkkosivut, jotka toimivat eri laitteilla eivätkä vaadi jatkuvaa päivitystä eli lisäkustannuksia. Responsiivinen verkkosuunnittelu on ylivoimaisesti paras ja yksinkertaisin vaihtoehto. Sillä pystyy toteuttamaan mobiiliystävälliset verkkosivut ja sen lisäksi verkkosivut skaalautuvat myös muille laitteille. Responsiivisten verkkosivujen konsepti koostuu kolmesta asiasta, mutta pääpiirre on kuitenkin mittayksiköiden suhteellisuus. Suhteelliset mittayksiköt toteuttavat skaalautuvuuden erikokoisille laitteille. Skaalautuvuuden ja riippumattomuuden takia responsiiviset verkkosivut toimivat vielä pitkään tulevaisuudessa eivätkä vaadi päivittämistä.

Verkkosivuston toteutus oli erittäin helppoa, vaikka en olekaan kokenut webkoodaaja. Työssä olin oppinut jo perusideat, mikä ehkä vaikutti asiaan. Toteu-

tuksessa olisi uusi CSS3:n display-ominaisuuden flexbox-asettelu vähän helpottanut. Flexboxin idea on luoda flexbox-containeri ja sen jälkeen kaikille sisällä oleville elementeille annetaan flex-arvo, joka kuvastaa kyseisen elementin suhteellista kokoa muihin sisällä oleviin elementteihin. Käytännössä flexbox toteuttaa joustavan asemoinnin eri tavalla. Tämän lisäksi flexbox sisältää mm. ominaisuuden, jolla voi määrittää järjestyksen, miten sisällä olevat elementit listataan. Tämä helpottaa sisällön uudelleenasemointia toiseen näkymään. Minä en vain halunnut käyttää flexboxia vielä, koska se ei ole vielä täysin tuettu kaikilla selaimilla (Can I use… 2017, hakusana flexbox). Jos haluaisi käyttää flexboxia ja täyden tuen verkkosivuille, joutuisi aluksi testaamaan, tukeeko käyttäjän selain flexbox-asettelua, ja sen lisäksi tekemään myös tällä tavalla toiset tyylit niille selaimille, jotka eivät vielä tue flexboxia. Tulevaisuudessa se kuitenkin todennäköisesti vakiintuun, koska se yksinkertaistaa koodia huomattavasti.

Jos toteutus olisi ollut oikea verkkosivusto jollekin yritykselle ja käytössä olisi ollut jokin sisällönhallintaohjelma, olisin halunnut tehdä toteutukseen toiminnallisuuden, että vieritettäessä alaspäin se automaattisesti lataa uusia artikkeleita. Näin käyttäjän ei tarvitsisi painaa seuraava sivu - tai vastaavaa painiketta. Tämä toteutus oli vain mallipohja verkkosivulle, niin moinen ei ollut mahdollista. Itse arvostan paljon, kun esimerkiksi verkkokaupassa selaa tuotteita eikä ole seuraava- ja edellinen-nappeja, vaan tuotteita tulee lisää, kun sivua vierittää alaspäin.

LÄHTEET

Can I use... 2017. Support tables for HTML5, CSS3, etc. Saatavissa: http://caniuse.com. Hakupäivä 11.4.2017.

Facebook 2016. Facebook Q3 2016 Results. Saatavissa: https://s21.q4cdn.com/399680738/files/doc_presentations/FB-Q316-Earnings-Slides.pdf. Hakupäivä 5.1.2017.

Google Developers 2015. Responsive Web Design. Saatavissa: https://developers.google.com/webmasters/mobile-sites/mobile-seo/responsive-design. Hakupäivä 25.1.2017.

Kloboves, K. 2016. Google Webmaster Central Blog. Continuing to make the web more mobile friendly. Saatavissa: https://webmasters.googleblog.com/2016/03/continuing-to-make-web-more-mobile.html. Hakupäivä 12.12.2016.

Makino T. ja Phan D. 2015. Google Webmaster Central Blog. Rolling out the mobile-friendly update. Saatavissa: https://webmasters.googleblog.com/2015/04/rolling-out-mobile-friendly-update.html. Hakupäivä 12.12.2016.

Marcotte, E. 2010. A List Apart. Responsive Web Design. Saatavissa: https://alistapart.com/article/responsive-web-design. Hakupäivä 3.4.2017.

Marcotte, E. 2014. Responsive Web Design Second Edition. New York: A Book Apart.

Mueller, J. 2009. Google Webmaster Central Blog. Handling legimate cross-domain content duplication. Saatavissa: https://webmasters.googleblog.com/2009/12/handling-legitimate-cross-domain.html. Hakupäivä 15.1.2016.

Ratcliff, C. 2016. What are the top 10 most popular search engines. Saatavissa: https://searchenginewatch.com/2016/08/08/what-are-the-top-10-most-popular-search-engines/. Hakupäivä 10.12.2016.

StatCounter 2016. Mobile and table internet usage exeeds desktop for first time worldwide. Saatavissa: http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide. Hakupäivä 8.12.2016.

The jQuery Foundation 2017. Saatavissa: https://jquery.org/. Hakupäivä 11.4.2017.

Wikipedia The Free Encyclopedia 2017. Saatavissa: https://en.wikipedia.org/wiki/Main_Page. Hakupäivä 1.3.2017.

```
<!DOCTYPE html>
<html>
<head>
  <title>Lehtisivusto</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link href="Styles/styles.css" rel="stylesheet" />
  <script src="https://ajax.goog-
leapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
  <script src="https://raw.githubusercontent.com/LeaVerou/prefixfree/gh-
pages/prefixfree.min.js"></script>
  <script src="Scripts/html5shiv-printshiv.min.js"></script>
  <script src="Scripts/selectivizr-min.js"></script>
  <script src="Scripts/rwd-website.js"></script>
</head>
<body>
  <header class="s-12 m-12 d-12 l-12 background-darkbrown">
    <nav>
      <a href="#" class="nav-text"></a>
      <div class="title left s-12 m-12 d-4 l-3">
        <a class="" href="#home">Trump News</a>
      </div>
      <div class="top-nav right s-12 m-12 d-8 l-9">
        <ul class="top-ul s-12 m-12 d-12 l-12">
          <li class="">
            <a href="#">Tech</a>
            <ul class="">
              <li><a href="#">SubLink1</a></li>
              <li><a href="#">SubLink2</a></li>
              <li><a href="#">SubLink3</a></li>
            </ul>
          </li>
          <li class="">
            <a href="#">Business</a>
            <ul class="">
              <li><a href="#">SubLink1</a></li>
              <li><a href="#">SubLink2</a></li>
              <li><a href="#">SubLink3</a></li>
              <li><a href="#">SubLink4</a></li>
            </ul>
          </li>
          <li class="">
            <a href="#">Culture</a>
            <ul class="">
              <li><a href="#">SubLink1</a></li>
              <li><a href="#">SubLink2</a></li>
              <li><a href="#">SubLink3</a></li>
              <li><a href="#">SubLink4</a></li>
            </ul>
          </li>
          <li class="">
            <a href="#">World</a>
            <ul class="">
              <li><a href="#">SubLink1</a></li>
              <li><a href="#">SubLink2</a></li>
              <li><a href="#">SubLink3</a></li>
            </ul>
          </li>
          <li class="">
            <a href="#">Other</a>
            <ul class="">
              <li><a href="#">SubLink1</a></li>
```

```
            <li><a href="#">SubLink2</a></li>
            <li><a href="#">SubLink3</a></li>
            <li><a href="#">SubLink4</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </nav>
</header>
<section>
  <div class="s-12 m-7 d-9 l-9">
    <article class="post background-green">
      <div class="s-12 d-5 l-4 post-image">
        <a href="#"><img srcset="Images/asian-1239271_440.jpg 440w, Im-
ages/asian-1239271_640.jpg 640w" alt="Not found"></a>
      </div>
      <div class="s-12 d-6 l-7 post-text">
        <a href="#">
          <h2>Business</h2>
          <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas
viverra eros a quam condimentum gravida. Suspendisse et felis aliquam, ultrices
turpis eu, suscipit odio. Suspendisse eu lacus tristique, consequat velit nec,
lobortis ligula. Sed bibendum augue non libero congue porttitor. Vestibulum eros
ligula, cursus a justo fringilla, luctus condimentum risus.</p>
        </a>
      </div>
      <div class="s-12 m-12 d-1 l-1 post-date">
        <p class="date">07</p>
        <p class="month">mar</p>
      </div>
    </article>
    <article class="post right-align background-blue">
      <div class="s-12 d-5 post-image">
        <a href="#"><img srcset="Images/malibu-2080075_440.jpg 440w, Im-
ages/malibu-2080075_640.jpg 640w" alt="Not found"></a>
      </div>
      <div class="s-12 d-6 post-text">
        <a href="#">
          <h2>Culture</h2>
          <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas
viverra eros a quam condimentum gravida. Suspendisse et felis aliquam, ultrices
turpis eu, suscipit odio. Suspendisse eu lacus tristique, consequat velit nec,
lobortis ligula. Sed bibendum augue non libero congue porttitor. Vestibulum eros
ligula, cursus a justo fringilla, luctus condimentum risus.</p>
        </a>
      </div>
      <div class="s-12 m-12 d-1 post-date">
        <p class="date">06</p>
        <p class="month">mar</p>
      </div>
    </article>

    <article class="post background-purple">
      <div class="s-12 m-12 d-6 post-text ad-container">
        <div class="ads">
          <a href="#">
            <h4>Advertisements!</h4>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maece-
nas viverra eros a quam condimentum gravida. Suspendisse et felis aliquam, ultri-
ces turpis eu, suscipit odio. Suspendisse eu lacus tristique, consequat velit
nec, lobortis ligula.</p>
          </a>
```

```
          </div>
        </div>
        <div class="s-12 m-12 d-6 post-text ad-container">
          <div class="ads">
            <a href="#">
              <h4>Advertisements!</h4>
              <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maece-
nas viverra eros a quam condimentum gravida. Suspendisse et felis aliquam, ultri-
ces turpis eu, suscipit odio. Suspendisse eu lacus tristique, consequat velit
nec, lobortis ligula.</p>
            </a>
          </div>
        </div>
      </article>

      <article class="post background-brown">
        <div class="s-12 d-5 post-image">
          <a href="#"><img srcset="Images/war-1447021_440.jpg 440w, Images/war-
1447021_640.jpg 640w" alt="Not found"></a>
        </div>
        <div class="s-12 d-6 post-text">
          <a href="#">
            <h2>World</h2>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas
viverra eros a quam condimentum gravida. Suspendisse et felis aliquam, ultrices
turpis eu, suscipit odio. Suspendisse eu lacus tristique, consequat velit nec,
lobortis ligula. Sed bibendum augue non libero congue porttitor. Vestibulum eros
ligula, cursus a justo fringilla, luctus condimentum risus.</p>
          </a>
        </div>
        <div class="s-12 m-12 d-1 post-date">
          <p class="date">04</p>
          <p class="month">mar</p>
        </div>
      </article>

      <article class="post background-lightblue">
        <div class="s-12 d-11 post-text">
          <a href="#">
            <h2>More news!</h2>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas
viverra eros a quam condimentum gravida. Suspendisse et felis aliquam, ultrices
turpis eu, suscipit odio. Suspendisse eu lacus tristique, consequat velit nec,
lobortis ligula. Sed bibendum augue non libero congue porttitor. Vestibulum eros
ligula, cursus a justo fringilla, luctus condimentum risus.</p>
          </a>
        </div>
        <div class="s-12 m-12 d-1 post-date">
          <p class="date">27</p>
          <p class="month">feb</p>
        </div>
      </article>
    </div>
    <!-- SIDEBAR -->
    <div class="s-12 m-5 d-3">
      <aside>
        <div class="aside-block">
          <div class="borders">
            <a href="#">
              <h3>First news</h3>
```

```
            <p>Lorem ipsum dolor sit amet, conse ctetuer. Duis autem vemeu iri-
ure dolor adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam</p>
            </a>
          </div>
        </div>
        <div class="aside-block">
          <div class="borders">
            <a href="#">
              <h3>Second news</h3>
              <p>Lorem ipsum dolor sit amet, conse ctetuer. Duis autem vemeu iri-
ure dolor adipiscing elit, sed diam nonummy nibh euismod tincidunt ut.</p>
            </a>
          </div>
        </div>
        <div class="aside-block">
          <div class="ad-container borders">
            <a href="#">
              <h3>Ads!</h3>
              <p>Lorem ipsum dolor sit amet, conse ctetuer. Duis autem vemeu iri-
ure dolor adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam</p>
            </a>
          </div>
        </div>
        <div class="aside-block">
          <div class="borders">
            <a href="#">
              <h3>Fourth news</h3>
              <p>Lorem ipsum dolor sit amet, conse ctetuer. Duis autem vemeu iri-
ure dolor adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam. Lorem ipsum
dolor sit amet, consectetur adipiscing elit. Maecenas viverra eros a quam condi-
mentum gravida. Suspendisse et felis aliquam, ultrices turpis eu, suscipit odio.
Suspendisse eu lacus tristique.</p>
            </a>
          </div>
        </div>
        <div class="aside-block">
          <div class="borders">
            <a href="#">
              <h3>Fifth news</h3>
              <p>Lorem ipsum dolor sit amet, conse ctetuer. Duis autem vemeu iri-
ure dolor adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam</p>
            </a>
          </div>
        </div>
        <div class="aside-block">
          <div class="borders">
            <a href="#">
              <h3>Sixth news</h3>
              <p>Lorem ipsum dolor sit amet, conse ctetuer. Duis autem vemeu iri-
ure dolor adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam</p>
            </a>
          </div>
        </div>
      </aside>
    </div>
  </section>
  <!-- FOOTER -->
```

```
  <footer>
    <h4 class="text-center text-clear">Footer</h4>
    <div class="s-12 m-4 d-3">
      <div class="s-6 m-6 d-7">
        <a href="#">Employees</a>
        <a href="#">Partners</a>
      </div>
      <div class="s-6 m-6 d-5">
        <a href="#">Jobs</a>
        <a href="#">Blogs</a>
      </div>
    </div>
    <div class="s-12 m-8 d-5">
      <div class="s-6 m-6 d-6">
        <a href="#">Privacy</a>
        <a href="#">Terms and Services</a>
      </div>
      <div class="s-6 m-6 d-6">
        <a href="#">Help</a>
        <a href="#">Social media links</a>
      </div>
    </div>
    <div class="s-12 d-4">
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas vi-
verra eros a quam condimentum gravida.</p>
    </div>
  </footer>
</body>
</html>
```

```css
* {
    box-sizing:border-box;
    margin:0;
}
html, body {
    font-size:100%;
    font-family:"Open Sans",Arial,sans-serif;
}

h1 {font-size:2.7em;}
h2 {font-size:2.2em;}
h3 {font-size:1.8em;}
h4 {font-size:1.4em;}
h5 {font-size:1.1em;}
h6 {font-size:0.9em;}

a{text-decoration:none;}

p,li{
  font-size: 0.85em;
  line-height: 1.6;
}

img,video {
    border:0;
    display:block;
    height:auto;
    max-width:100%;
    float:inherit;
}

nav {
    position: relative;
    display:block;
    width:100%;
}
nav a:visited{color:#ff9414;}
nav a{
    color:#ff9414;
    transition:color 0.20s linear 0s;
}
nav a:hover, nav a:active {color:#ff1e1e;}

header{
    margin-bottom:0.5em;
    border-bottom:1px solid black;
    box-shadow: 0px 2px 3px 1px rgba(0,0,0,0.75);
}

.title {
    font-size: 2.3em;
    display: inline-block;
    padding: 0.6rem;
}

.top-nav ul {
    padding:0;
    height:inherit;
}
.top-ul ul {
    position:absolute;
    background-color:inherit;
```

```
}

.top-ul li {
    float:left;
    list-style:none;
    padding:0px;
    line-height:1;
    font-size:1.45em;
}
.top-ul > li {
    position:relative;
}

.top-ul li:hover > a, .top-ul li:active > a {
    color:#ff1e1e;
    background-color:#283D5F;
}

.top-ul li a {
    display:block;
    font-size:inherit;
    background-color:#343538;
    padding:0;
}
.top-ul li ul {
    display:none;
    width:100%;
}
.top-ul ul li:last-child{border-bottom:1px solid black;}
.top-ul li ul li{
    font-size:0.8em;
    float:none;
    list-style:none;
    width:100%;
    padding:0px;
}
.top-ul li ul li a {
    font-size:inherit;
    background-color:#343538;
    padding:0.625em 1em;
}

.count-number {display:none;}

article a:visited{color: #fff;}

.ad-container{padding:0.5em;}
.ad-container h2, .ad-container h3, .ad-container h4{text-align:center;}
.ad-container .post-text a:hover{opacity:0.6;}

article .ads {
    border:1px solid black;
    box-shadow: 0px 0px 2px 1px rgba(0,0,0,0.75);
    border-radius:20px;
    padding:0.5rem;
}

.post{
    overflow:hidden;
    margin-bottom:0.5em;
}
.post:last-child{margin-bottom:0;}
```

```
.post-text {
            color: #fff;
            padding: 1.0em;
}
.post-text > a:hover{
            opacity:0.6;
}

.post-text h2 {
  color: inherit;
  font-weight:100;
  margin: 0em 0em 0.5em 0em;
  opacity: 1;
  transition: opacity 0.20s linear 0s;
  text-align:center;
}
.post-text a {color:inherit;}

.post-text p {
    color:inherit;
            font-size: 0.9em;
            opacity: 0.6;
}

.post-image img {
            margin: 0px;
            opacity: 1;
            transition: opacity 0.20s linear 0s;
}
.post-image img:hover{
            opacity: 0.6;
}

.right-align .post-image a img{
    float:right;
}
.post .post-date {
            font-weight: 100;
            text-align: center;
}
.post-date .date {
            font-size: 2.4em;
}
.post-date .month {
            font-size: 1.4em;
    margin-top:-0.7em;
}

.right-align .post-image{
            float: right;
}
.right-align .post-text {
            float: right;
}
.right-align .post-date {
            float: right;
            text-align: right;
}

aside h3 {
  color: #0b192f;
```

```
  font-size: 1.6em;
  font-weight: 300;
  margin: 0em 0em 0.25em 0em;
}
aside p {
            color: #637693;
            font-size: 0.85em;
}

.aside-block {
    background-color:#fff;
            padding: 0em 0.5em 1.0em 0.5em;
}

.aside-block div:hover, .aside-block div:active {background-color:palegreen;}

footer {
            background-color:#343538;
    color:lightgray;
            display: inline-block;
            margin-bottom: 0;
            margin-top: 0.5em;
            padding: 1.0em;
            position: relative;
            width: 100%;
}
footer a, a:visited{
    color:lightgray;
    float:none;
    display:block;
    line-height:1.6;
    text-decoration:underline;
    transition:color 0.20s linear 0s;
}

footer a:hover, footer a:active{color:red;}
footer p {
            font-size: 0.9em;
}
footer h4 {
    margin-bottom:0.333333em;
}

.background-green{background-color:#39ac73;}
.background-brown{background-color:#cc6600}
.background-purple{background-color:mediumpurple;}
.background-blue{background-color:royalblue;}
.background-lightblue{background-color:#2d95bb;}
.background-darkbrown{background-color:#343538}

.text-center {text-align:center!important;}
.text-right {text-align:right!important;}

.text-clear{
    text-decoration:none;
    font-weight:100;
}

.right {float:right;}
.left {float:left;}

.borders {
```

```
    border:1px solid black;
    box-shadow: 0px 0px 2px 1px rgba(0,0,0,0.75);
    border-radius:20px;
    padding:0.5em
}

/* Queryt */

@media screen and (max-width: 768px) {

    .title {
    display: block;
    text-align: center;
    }
    .nav-text {
    color:#ff9414;
    transition:color 0.20s linear 0s;
    font-size: 2.5em;
    line-height: 1;
    padding-top: 0.5em;
    text-align: center;
    }

    .nav-text:hover, .nav-text:active {color:#ff1e1e;}

    .top-nav ul {
    float: none;
    text-align: center;
    }
    .top-ul li {
    border-top: 1px solid black;
    display: block;
    }
    .right-align .post-text {text-align: left;}
    .right-align .post-date {text-align: center;}

    footer {text-align: center;}

}

@media screen and (min-width:769px) {
    .d-1 {width:8.3333%;}
    .d-2 {width:16.6666%;}
    .d-3 {width:25%;}
    .d-4 {width:33.3333%;}
    .d-5 {width:41.6666%;}
    .d-6 {width:50%;}
    .d-7 {width:58.3333%;}
    .d-8 {width:66.6666%;}
    .d-9 {width:75%;}
    .d-10 {width:83.3333%;}
    .d-11 {width:91.6666%;}
    .d-12 {width:100%;}

    .title {
        border-right:1px solid black;
        box-shadow: 0px 0px 2px 1px rgba(0,0,0,0.75);
    }
    .post-image img {
        max-height: 260px;
    }
    .top-ul > li{
```

```
        width:20%;
        border-right:1px solid black;
        box-shadow: 0px 0px 2px 1px rgba(0,0,0,0.75);
    }
    .top-ul >li > a{
        padding:0.85em 0em 0.85em 0em;
        text-align:center;
    }
    .top-ul li:hover > ul, .top-ul li:active > ul {
                display:block;
                z-index:10;
    }
    .top-ul li:hover > ul ul, .top-ul li:active > ul ul { /* jos ois sub-sub-
menui */
    left:100%;
    top:0%;
    width:100%;
    }
    .top-nav .top-ul li ul{
        border:1px solid black;
        box-shadow: 2px 2px 2px 1px rgba(0,0,0,0.75);
    }
    .top-ul li ul li{border-bottom: 1px solid black;}
}
@media screen and (min-width:1367px){
    .l-1 {width:8.3333%;}
    .l-2 {width:16.6666%;}
    .l-3 {width:25%;}
    .l-4 {width:33.3333%;}
    .l-5 {width:41.6666%;}
    .l-6 {width:50%;}
    .l-7 {width:58.3333%;}
    .l-8 {width:66.6666%;}
    .l-9 {width:75%;}
    .l-10 {width:83.3333%;}
    .l-11 {width:91.6666%;}
    .l-12 {width:100%;}
}

.nav-text{display:none;}

.s-1, .s-2, .s-3, .s-4, .s-5, .s-6, .s-7, .s-8, .s-9, .s-10, .s-11, .s-12, .m-1,
.m-2, .m-3, .m-4, .m-5, .m-6, .m-7, .m-8, .m-9, .m-10, .m-11, .m-12, .d-1, .d-2,
.d-3, .d-4, .d-5, .d-6, .d-7, .d-8, .d-9, .d-10, .d-11, .d-12 .l-1, .l-2, .l-3,
.l-4, .l-5, .l-6, .l-7, .l-8, .l-9, .l-10, .l-11, .l-12 {
  float:left;
  position:static;
}

@media screen and (max-width:768px) {
    .m-1 {width:8.3333%;}
    .m-2 {width:16.6666%;}
    .m-3 {width:25%;}
    .m-4 {width:33.3333%;}
    .m-5 {width:41.6666%;}
    .m-6 {width:50%;}
    .m-7 {width:58.3333%;}
    .m-8 {width:66.6666%;}
    .m-9 {width:75%;}
    .m-10 {width:83.3333%;}
    .m-11 {width:91.6666%;}
    .m-12 {width:100%}
```

```
    nav {
        display:block;

    }
    .top-nav > ul {
        height: 0;
        max-width: 100%;
        overflow: hidden;
        position: relative;
        z-index: 999;
    }
    .top-nav > ul.show-menu {
        height: auto;
    }
    .top-nav .top-ul ul {
        margin-top: 0;
        position: relative;
    }
    .top-nav li ul li a {min-width: 100%;}
    .top-nav .top-ul li {
        float: none;
        border-top:1px solid black;
        box-shadow:none;
    }
    .top-nav .top-ul li a {
        display: block;
        padding: 0.6em;
        text-align: center;
    }

    .top-ul ul ul{  /* sub-sub menut*/
        display: block;
        overflow: hidden;
        height: 0;
    }
    .top-nav .top-ul ul li{
        border:none;
        border-top:1px solid black;
        box-shadow:none;
    }

    .top-nav > ul ul.show-ul {
        display: block;
        height: auto;
    }
    .top-nav li ul li a {
        background-color: #4c5460;
        padding: 0.6rem;
    }

    .nav-text {
        color:#ff9414;
        display: block;
        max-width: 100%;
        text-align: center;
    }
    .nav-text:hover, .nav-text:active {color:#ff1e1e;}
    .nav-text:after {
        content: "\2630";
    }
}
```

```
@media screen and (max-width:480px) {
    .s-1 {width:8.3333%;}
    .s-2 {width:16.6666%;}
    .s-3 {width:25%;}
    .s-4 {width:33.3333%;}
    .s-5 {width:41.6666%;}
    .s-6 {width:50%;}
    .s-7 {width:58.3333%;}
    .s-8 {width:66.6666%;}
    .s-9 {width:75%;}
    .s-10 {width:83.3333%;}
    .s-11 {width:91.6666%;}
    .s-12 {width:100%}

    .aside-block{padding:0px!important;}
    .aside-block:first-child{margin-top:1em;}
    .aside-block .borders{
        border:none!important;
        box-shadow:none!important;
        border-radius:0px!important;
        padding:0.5em 0em 0.5em 1em;
    }
    .aside-block h3{
        font-size:1.2em;
        margin:0px!important;
        text-decoration:underline;
    }
    .aside-block p{display:none;}

}
```

```
jQuery(document).ready(function ($) {
    $('.top-nav > ul li:has(ul)').addClass('submenu');
    $('.top-nav > ul li.submenu > a').click(function () {
        // Close other open sub menus
        $('.top-nav > ul li.submenu > ul').removeClass('show-ul');
        $('.top-nav > ul li.submenu:hover > ul').toggleClass('show-ul');
    });
    // Mobile aside block
    $('.nav-text').click(function () {
        $('.top-nav > ul').toggleClass('show-menu');
        $('.top-nav > ul li.submenu > ul').removeClass('show-ul');
    });
});
```