HTML5 MOBIILIKÄYTÖSSÄ

WWW-sivujen mobiiliversion toteuttaminen

Miia Hallström

Opinnäytetyö
Kesäkuu 2013
Tietojenkäsittelyn koulutusohjelma
Digimedia

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Digimedia

HALLSTRÖM, MIIA:
HTML5 mobiilikäytössä
WWW-sivujen mobiiliversion toteuttaminen

Opinnäytetyö 57 sivua, joista liitteitä 0 sivua
Kesäkuu 2013

Opinnäytetyössäni käsitellään edelleen kehitteillä olevan HTML5-merkintäkielen käyttöä mobiililaitteilla. Mobiililaitteiden, eli puhelimien ja tablettien, käyttö lisääntyy päivä päivältä, mikä tuo omat vaatimuksensa myös nettisivuille. Ei riitä, että toteutetaan sivusto pöytäkoneelle vaan niiden tulee olla toimivat myös pienemmillä laitteilla. Omat ongelmansa aiheuttavat Internet Explorerin vanhat versiot, jotka eivät tunnista HTML5-kieltä. Opinnäytetyön tarkoituksena oli tutkia, millaisia eri tekniikoita on käytettävissä, kun toteutetaan nettisivut tietokone- ja mobiilikäyttöön niin, että siinä on huomioitu kaikki tarpeellinen. Tavoitteena oli saada selville, mikä on parhain ja toimivin menetelmä kyseisten sivujen toteuttamiseen.

Opinnäytetyötä tehdessä saatiin selville monia eri tapoja, joilla nettisivuista saadaan entistä toimivammat myös mobiililaitteilla. On olemassa mobiilikäyttäjää helpottavia ominaisuuksia, joiden käytön pitäisi olla itsestäänselvyys suunnittelijoille. HTML5 ja CSS3 tarjoavat monia mahdollisuuksia näiden toteuttamiseen. Esimerkkinä käytettiin työharjoittelussa toteuttamaani ravintola.fi-portaalin mobiiliversiota m.ravintola.fi. Sivuston toteutuksessa oli huomioitu monia tapoja ja ratkaisuja, jotka ilmenivät myös opinnäytetyön lähdemateriaaleista. Moni asiantuntija näyttää olevan sitä mieltä, että Internet Explorerin vanhojen versioiden sijaan tulisi keskittää huomio uusimpiin selaimiin ja tarjota vanhoille versioille vain pelkistetty ulkoasu.

Opinnäytetyön lopputulos on, että kaikkiin asioihin ei ole oikeaa ratkaisua. On hyviä ja huonoja ratkaisuja sekä monien eri ihmisten mielipiteitä. Vaikka sivuston toteuttaisi täysin ohjeiden mukaisesti, ei se silti miellytä kaikkia. On kuitenkin olemassa monia vinkkejä ja tapoja, joilla mobiilisivuista saadaan toimivammat. Kun perusasiat ovat hallussa, jatkossa voidaan keskittyä yksityiskohtaisemmin pieniin, mutta tärkeisiin ominaisuuksiin, jotka parantavat mobiilikäyttäjän käyttökokemusta entisestään.

Opinnäytetyön ymmärtäminen edellyttää HTML-merkintäkielen ja CSS-tyylikielen perusteiden tietämystä, koska niitä ei käsitellä erikseen.

Asiasanat: HTML5, CSS3, WWW-sivut, mobiililaitteet

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Digital Media

HALLSTRÖM, MIIA:
HTML5 in Mobile Use
Developing Mobile Website

Bachelor's thesis 57 pages, appendices 0 pages
June 2013

This Bachelor's thesis is based on the HTML5 markup language used on mobile devices. The number of mobile devices increases day by day which brings its own requirements for web pages. It is not enough that the website is implemented for desktop computers, the site must also be user friendly in smaller devices. HTML5 is still a work in progress, but most modern browsers support it. The only exceptions are the old versions of Internet Explorer which do not recognize the HTML5 language.

The purpose of this thesis was to study what kinds of techniques are available when creating a website for both computer and mobile use, so that it takes into account all utilities. The objective of this study was to find out what is the best and most viable method of creating those pages.

The theoretical section explores many different ways how to make a website more functional on mobile devices. HTML5 and CSS3 offer many opportunities for their realization. Ravintola.fi-portal's responsive mobile version m.ravintola.fi was used as an example.

The findings indicate that there is not the right solution for everything. There are good and bad solutions and people's different opinions. However, there are many tips and ways how a mobile website can be more functional. Once the basics are covered, it is possible to focus more in the details that enhance the mobile user experience even further.

Key words: HTML5, CSS3, web pages, mobile devices

SISÄLLYS

1 JOHDANTO ... 6
2 MOBIILISIVUSTON MERKITYS ... 8
3 MERKINTÄTAVAT ... 10
3.1 HTML5 ... 10
3.1.1 Uudet ominaisuudet ... 11
3.1.2 Lomakkeet ... 13
3.2 CSS3 ... 16
3.2.1 Uudet ominaisuudet ... 17
3.2.2 CSS3 ja vanhat selaimet ... 23
3.3 JavaScript ... 23
4 SIVUSTON TOTEUTUS ... 25
4.1 Responsiivinen sunnittelu ... 27
4.2 Mobiili- vai pöytäkoneversio ensin? ... 28
4.3 Mobile Meta Tags ... 30
4.4 CSS3 Media Queries ... 30
4.5 Päätteiden näytöt ... 33
4.6 Syötetavat ... 35
4.7 Sivuston sisältö ... 37
4.7.1 Mobiilisivuston keventäminen ... 38
4.7.2 Sijoittelu ... 39
4.7.3 Tekstikentät ja lomakkeet ... 42
4.8 Muuta huomioitavaa ... 43
5 TOTEUTETTU SIVUSTO ... 45
6 HTML5 JA INTERNET EXPLORER ... 51
6.1 HTML5:n tarvitsemat JavaScriptit ... 53
6.2 CSS3:n tarvitsemat JavaScriptit ... 54
7 POHDINTA ... 55
LÄHTEET ... 57

ERITYISSANASTO

Polyfill / plugin	Polyfill on osa koodia (tai liitännäistä), joka mahdollistaa toimintoja selaimille, jotka eivät tue niitä.
Natiivi-sovellus	Sovellus, joka koodataan tietylle päätelaitteelle ja mahdollisesti vielä sen tietylle käyttöjärjestelmän versiolle.
Regex	Säännölliset lausekkeet *(eng. regular expressions)* tarkoittavat joukkoa merkkijonoja, jotka tarkastelevat kuuluuko jokin merkkijono lausekkeen määrittämään kieleen.
WebKit	Selainmoottori, joka noutaa datan verkosta sekä tulkitsee HTML/CSS-kielisen koodin näkyväksi web-sivuksi.

1 JOHDANTO

Teknologia kehittyy jatkuvasti ja nopealla vauhdilla. Kun uusi laite julkaistaan, seuraava on jo kehitteillä. Ennen pärjättiin ilman ylimääräisiä laitteita ja nykyään ei välttämättä riitä edes pelkkä älypuhelin, vaan sen lisäksi tarvitaan tabletti ja tietokone. Meille kehitellään laitteita ominaisuuksilla, joita emme edes tienneet tarvitsevamme, emmekä nykyään voisi kuvitellakaan tulevamme toimeen ilman niitä. Se taas luo uusia vaatimuksia myös Internetiin. Ei riitä, että nettisivut näkyvät hyvin tietokoneella, vaan niiden tulee näkyä vaivattomasti myös tableteilla ja niitä pienemmillä puhelimilla.

Opinnäytetyöni käsittelee HTML5-kielen käyttöä mobiililaitteilla. HTML5 on uusin edelleen kehitteillä oleva versio HTML-merkintäkielestä. Suurin osa nykypäivän selaimista tukee sen uusia ominaisuuksia. On kuitenkin olemassa muutamia asioita, jotka tulee ottaa huomioon suunnittelussa ja käyn niitä läpi opinnäytetyön edetessä. Suurimmaksi ongelmaksi muodostuu Internet Explorerin vanhemmat versiot, jotka eivät tue HTML5:tä. CSS3 liittyy olennaisena osana mobiilisivuistoihin, joten käyn myös läpi sen uusia ominaisuuksia, jotka auttavat esimerkiksi sivustojen keventämisessä.

Opinnäytetyöni aihe on erittäin ajankohtainen ja hyödyllinen. Nykyään ei riitä, että toteuttaa nettisivut pelkästään pöytäkonekäyttöön vaan on suorastaan vaatimus, että ne toimivat kunnolla myös mobiilipäätteillä. Opinnäytetyön toimeksiantajana toimii tamperelainen verkkoviestinnän asiantuntija Syrjä Interactive. Työn tarkoituksena on myös antaa toimeksiantajalle monipuolinen kuva asioista, jotka tulisi huomioida mobiilisivustoja luotaessa.

Opinnäytetyön tarkoituksena oli tutkia millaisia eri tekniikoita on käytettävissä, kun toteutetaan nettisivut tietokone- ja mobiilikäyttöön niin, että siinä on otettu huomioon kaikki tarpeellinen. Esimerkkinä käytän jo toteuttamaani ravintola.fi-portaalin mobiiliversiota m.ravintola.fi. Mobiilipäätteessä sivujen tulisi latautua mahdollisimman nopeasti, sisältää vain tarpeelliset tiedot sekä niiden tulisi olla yksinkertaiset ja helppokäyttöiset. Sivuston ulkoasussa tulee ottaa huomioon eri päätteiden (tietokoneen, tabletin, puhelimen) näyttöjen resoluutiot. Tällä tarkoitetaan ensisijaisesti sitä, että sisällön rakenne ja asettelu tulisi suunnitella niin, että se sopii jokaiselle laitteelle ja niiden nor-

maaleille käyttötavoille. Opinnäytetyön tavoitteena on saada selville mikä on järkevin ja toimivin menetelmä toteuttaa nettisivut sekä tietokone- että mobiilikäyttöön.

Lähteinä on käytetty sekä kirjoja että Internetiä. Kirjoista saa erittäin monipuolisen ja kattavan kuvan tutkittavasta asiasta, mutta osa tiedoista saattaa olla vanhentunutta. Informaatioteknologian ala muuttuu jatkuvasti, minkä vuoksi Internet tarjoaa ajankohtaisempaa tietoa kuin muutaman vuoden vanha kirja. Tämän vuoksi pyrin valitsemaan opinnäytetyöhöni mahdollisimman uusia kirjoja ja artikkeleja.

2 MOBIILISIVUSTON MERKITYS

Nykypäivänä nettisivujen vaatimukset ovat suuret, sillä älypuhelinten käyttömäärä on lisääntynyt huomattavasti viime vuosien aikana. Ei riitä, että toteuttaa sivut toimimaan hyvin tietokoneilla, vaan niiden tulisi myös toimia hyvin mobiililaitteilla. Vaikka nettisivut näyttäisivät hyvältä pöytätietokoneella, sama ei välttämättä päde puhelimiin ja tabletteihin. Mobiililaitteita on joka päivä käytössä yhä enemmän ja enemmän. Minne ikinä meneekin, niin poikkeuksetta jollakin on puhelin kädessä. Ennen riitti, että puhelimella pystyi soittamaan ja lähettämään tekstiviestejä. Nykyään suurimmalla osalla puhelimen omistajista vähimmäisvaatimuksiin kuuluu puheluiden soittaminen, viestien lähetys, hyvä ja nopea kamera, mahdollisuus kuunnella musiikkia sekä tärkeimpänä mahdollisuus päästä sujuvasti Internetiin.

Internetin käytöstä puhelimella on tehty tilastotutkimuksia. Peruspuhelimet muodostavat suuren osan matkapuhelinverkosta, mutta niillä on suuri käyttöero kyvykkäämpiin älypuhelimiin. Älypuhelinomistajista 35 % selaa mobiilinettiä vähintään päivittäin, peruspuhelinomistajista vain 4 %. Tämä tieto on vuodelta 2009, joten erot ovat vieläkin suurempia tänä päivänä. (Wroblewski 2011, 12.) Qualcommin elokuussa 2012 tehdyn tutkimuksen mukaan 37 % kyselyyn vastanneista tarkistaa mobiililaitteensa 30 minuutin välein tai useammin. Kyselyn mukaan myös suurimman prosentuaalisen määrän on saanut Internetin selailu (46 %), tiedonhaku Internetistä (41 %) sekä vierailu sosiaalisessa mediassa (37 %), kun kysyttiin mitä he tekevät ainakin kaksi kertaa viikossa. (Statistic Brain, 2012.)

Yhdistelmä mobiili- ja työpöytäsivuston kokemuksista lisää käyttäjiä molemmissa laitteissa. Esimerkiksi Facebook ei ajattele mobiilikokemusta osana tietokonesivustoa; se omaksuu tyylin, jolla tehdä koko Facebook-kokemus paremmaksi. (Wroblewski 2011, 17.)

Käyttäjien käyttäytymiset mobiilisivuilla määrittävät sen miten mobiilisivut voidaan luoda ja järjestää täyttämään käyttäjän tarpeet. Käyttö koostuu yleensä muutamasta vuorovaikutustavasta (Wroblewski 2011, 50):

- Haku/Etsintä (kiireellinen tieto, paikallinen): Käyttäjä tarvitsee vastauksen johonkin heti, usein sidoksissa sen hetkiseen sijaintiin maailmassa.
- Tutkia/Pelata (tylsää, paikallinen): Käyttäjällä on aikaa tapettavana ja hän haluaa vain joutoajalle tekemistä.
- Saapuminen/Tila (toistaminen/microtasking): Jotain käyttäjälle tärkeää muuttuu tai päivittyy ja käyttäjä haluaa pysyä ajan tasalla.
- Muokkaa/Luo (kiireellinen muutos/microtasking): Käyttäjän pitää saada jotakin tehtyä saman tien eikä se voi odottaa.

Nykypäivänä ei ole yhtään ihmeellistä, jos samalta henkilöltä löytyy kotoaan älypuhelin, tabletti sekä pöytäkone ja mahdollisesti vielä kannettava tietokone erikseen. Kuulemani perusteella ei ole myöskään ihmeellistä, että sohvalla istuskeltaessa ensimmäisenä käteen tarttuu se lähin laite, jos on tarvetta katsoa jotain Internetistä. Vaikka monien nettisivujen käyttökokemus on huomattavasti mukavampi ja varsinkin nopeampi tietokoneella käytettäessä, myönnän itsekin käyttäväni puhelinta samaan tarkoitukseen, jos se tarkoittaa sitä, että voin jatkaa oleskeluani kyseisellä paikalla. Älypuhelimella on sekä hyvät että huonot vaikutuksensa käyttäjiin. Kehityksen myötä vaadimme päivä päivältä enemmän puhelimelta ja sen käytettävyydeltä.

3 MERKINTÄTAVAT

3.1 HTML5

HTML5 tulee olemaan uusi standardi HTML-kuvauskielelle. Sitä kehitetään edelleen, mutta merkittävimmät selaimet tukevat suurinta osaa sen uusista elementeistä sekä ohjelmointirajapinnoista. (W3Schools, 2012.) Useimmat ominaisuudet on tuettu uusimmilla Internet-selaimilla mukaan lukien Applen Safari, Google Chrome, Opera, Mozilla Firefox sekä jopa Internet Explorer 9 (Frain 2012, 98). Viimeisimmät versiot selaimista tukevat HTML5:ta tavalla tai toisella, mutta niiden edeltäjillä ei ole sadan prosentin tukea, josta haaveilemme. Vaikka HTML5-merkintäkielellä on paljon uusia jännittäviä ominaisuuksia, se sisältää myös, enemmän kuin koskaan aiemmin, siihen liittyviä webteknologioita, joista monet luottavat rikkaaseen JavaScriptin ohjelmointirajapintaan. (Schmitt & Simpson 2011, x.) Selaimen nykyisen tason käytettävissä olevista CSS3:n ja HTML5:n ominaisuuksista näkee parhaimmin Internet-osoitteesta http://caniuse.com. Sen ohella sivusto näyttää selaimen version tuen, joka on haettavissa ominaisuuden perusteella. (Frain 2012, 141.)

HTML5 on merkintäkielenä todella anteeksiantavainen. Se ei välitä vaikka ei käytetä lainausmerkkejä attribuuttien ympärillä, käytetään isojen sekä pienien kirjaimien sekoitusta, unohdetaan lopetustagi tai jätetään tyyppimääritelmä kokonaan pois. Se on silti yhtä validia. HTML5 validoi, vaikka jätettäisiin aloitus `<head>`-tagi kokonaan pois. (Frain 2012, 101.) HTML5:ssä ID voi alkaa myös numerolla. Mikäli ei ole tottunut kyseiseen merkintätapaan, saattaa olettaa arvon olevan virheellinen, mitä se olikin HTML 4.01:sta käyttäen. HTML5 kuitenkin poistaa tämän rajoituksen. Ainostaan tulisi muistaa, että ID ei sisällä välilyöntejä ja että se on yksilöllinen jokaisella sivulla. (Frain 2012, 138.)

Rakenteellisten elementtien käyttö ei ole välttämätöntä validin HTML5-tiedoston tekemiseen; tarvitaan vain dokumentin tyyppimääritelmä (Schmitt & Simpson 2011, 34). `<!DOCTYPE html>` sivun alussa määrittelee sen HTML5:ksi. Ei ole myöskään väliä kirjoitetaanko se isoilla vai pienillä kirjaimilla. Aikaisemmassa versiossa oli mahdollisesti käytössä seuraavanlainen määritelmä (Frain 2012, 99-100):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

HTML5:n määritelmä on hieman karsitumpi:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset=utf-8>
```

HTML5 on huomattavasti helppokäyttöisempi kuin sen edeltäjät, minkä huomaa jo dokumentin tyyppimääritelmästä. Uudet elementit, jotka käyn läpi seuraavassa alaluvussa, tekevät koodista yksinkertaisempaa ja selkeämpää. Vaikka HTML5 on edelleen työn alla, on se silti täysin käyttökelpoinen merkintäkieli sivustoilla. Kuten sanottu, suurin osa selaimista tukee sen ominaisuuksia ja aina löytyy vaihtoehtoisia ratkaisutapoja mahdollisesti toimimattomille ominaisuuksille. HTML5 ja CSS3 ovat aiheuttaneet myös suuren muutoksen nettisivujen ulkoasuissa uusien ominaisuuksien myötä. Ne yhdessä tarjoavat paljon uusia ominaisuuksia, jotka helpottavat varsinkin mobiilisivustoja.

3.1.1 Uudet ominaisuudet

Uudet elementit HTML5:ssä yksinkertaisesti kuvastavat sitä, mitä sen tekijät ovat jo entuudestaan tehneet (Schmitt & Simpson 2011, 6). Uudet rakenteelliset elementit on mahdollista korvata myös divillä, jota monet ovat jo tottuneet käyttämään. Jos tarvitaan elementti ainoastaan tyylitarkoitukseen, div on elementti, jota tulee käyttää. Tulisi muistaa keskittyä sisältöön ja välttää tarpeettomien divien käyttöä. (Schmitt & Simpson 2011, 8.) Uudet elementit ovat hyvä uutinen suunnittelijoille, mutta ehkä vielä parempi uutinen hakukoneille. Nyt ne pystyvät ymmärtämään sivuja paremmin kuin koskaan ennen ja sijoittavat sisällön sen mukaisesti. (Frain 2012, 24.)

HTML5:n uudet elementit on esitelty taulukossa 1 (Schmitt & Simpson 2011, 5; w3schools 2012).

TAULUKKO 1. HTML5:n merkittävimmät uudet elementit

header	Käytetään sivun otsikointia ja/tai sektiota varten. Voi sisältää myös lisätietoja, kuten logoja ja navigaation.
footer	Sisältää tietoa sivusta ja/tai sektiosta, kuten kuka sen kirjoitti, linkkejä sekä tekijänoikeusviittauksia.
nav	Sisältää päänavigaation linkit sivulla. Ei edellytetä, mutta sisältyy yleensä headeriin.
aside	Sisältää tiedot, jotka liittyvät ympärillä olevaan sisältöön. Voi esiintyä myös itsenäisesti, kuten sivupalkki.
section	Yleisin uusista elementeistä. Sisältää sisällön, joka on ryhmitetty temaattisesti tai siihen liittyvästi.
article	Käytetään itsenäiseen sisältöön, jota voidaan käyttää kokonaisena itsenäisesti sivulla, kuten esimerkiksi blogiin kirjautuminen.
video	Määrittelee videon tai filmin.
canvas	Käytetään piirtämään grafiikkaa, virallisesti sen säilömiseen. JavaScript piirtää grafiikat.
audio	Määrittää äänisisällön.
figure	Tarkentaa itsenäisen sisällön, kuten illustrationit, diagrammit, kuvat jne.
hgroup	Määrittelee otsikkoelementit (<h1>:stä <h6>:een).

Uudet elementit tuovat mukanaan myös pienen muutoksen CSS-tyylitiedostoon. Aiemmin HTML:n divissä käytetty id="header" merkitään tyylitiedostoon #header, kun taas <header>-tagia käyttämällä CSS:lle riittää ainoastaan tagin mukaisesti merkintä header. Mikäli tiedostossa kuitenkin käytetään useampaa samannimistä tagia, tulee muistaa kirjoittaa tarkemmat määritelmät yksittäisille tapauksille. (Frain 2012, 113-114.) Sivustolla saattaa esimerkiksi olla enemmän kuin yksi navigaatio. Siinä tapauksessa HTML5:n <nav>-tagia käytetään päänavigaatioon ja mahdollinen lisänavigaatio merkitään esimerkiksi vanhalla tavalla diviä käyttäen määrittelemällä sille oma ID. Toinen vaihtoehto on käyttää molemmissa <nav>-tagia määritellen molemmille omat ID:t.

Yhteydetön verkkosovellus

Mobiilikäyttäjien määrä Internet-sivuilla on vain kasvussa, joten heille tulisi myös tarjota mahdollisuus sivun sisältöön ilman yhteyttä Internetiin. HTML5:n yhteydetön verk-

kosovellus *(eng. offline web appliction)* tarjoaa mahdollisuuden tähän. Ominaisuus toimii uusimmilla selaimilla (Firefox 3+, Chrome 4+, Safari4+, Opera 10.6+, iOS 3.2+, Opera Mobile 11+, Android 2.1+, Internet Explorer 10+). Yksinkertaisimmallaan ominaisuus toimii, kun lisätään tiedostoon viittaus *.manifest*-tiedostoon, joka lataa listatut lähteet ja säilöö ne paikallisesti yhteyden ollessa poissa. (Frain 2012, 131.)

3.1.2 Lomakkeet

HTML5:llä on lukuisia eri syötekenttätyyppejä *(eng. input)*, jotka rajoittavat käyttäjän syöttömahdollisuuksia ilman, että tarvitaan JavaScript-koodia (taulukko 2). Lohduttavinta näissä uusissa ominaisuuksissa on se, että mikäli selain ei tue kyseistä ominaisuutta, se alentaa kentän vain tavalliseksi tekstikenttälaatikoksi. Lisäksi on olemassa hyviä liitännäisiä *(eng. polyfills)*, joilla saadaan selaimien vanhemmat versiot ajan tasalle. (Frain 2012, 243-245.)

TAULUKKO 2. HTML5:n syötekenttätyyppejä

`placeholder`	Sisältää sanan/tekstin, joka näkyy lomakkeen kentässä, kunnes se valitaan.
`required`	Osoittaa, että tieto on pakollinen. Mikäli kenttä jää tyhjäksi näytetään varoitusviesti.
`autofocus`	Mahdollistaa lomakkeen kentän valitsemisen automaattisesti käyttäjän syötteelle.
`autocomplete`	Automaattinen täydennys, jonka voi halutessaan ottaa pois käytöstä.
`list`	Datalist-elementin kanssa yhdessä tarjoaa mahdollisia vaihtoehtoja, kun käyttäjä alkaa syöttämään merkkejä.

Seuraavassa on lueteltu HTML5:n uudet syötekenttien tyypit ja niiden ominaisuudet (Frain 2012, 246-256):

- `type="email"` odottaa käyttäjän syötteen täsmäävän sähköpostiosoitteen syntaksiin. Kosketusnäytöllisissä laitteissa, esimerkiksi Androidissa, iPhonessa ja niin edelleen, se muuttaa näppäimistön vastaamaan tätä tyyppiä. Näppäimistöön ilmestyy muun muassa @-symboli.

- `type="number"` odottaa käyttäjän syötteen olevan numeerinen. Se myös lisää oletuksena kenttään elementin *(eng. spinner)*, jolla käyttäjä pystyy helposti lisäämään tai vähentämään lukuarvoa.

- `type="url"` tarkoittaa URL-syötteitä. Kuten email:ssa, kosketusnäyttölaitteissa näppäimistö muuttuu vastaamaan kyseistä tyyppiä. Näkymään tulee esimerkiksi näppäimet ”Go”, kauttaviiva (/) sekä ”.com” (kuva 1).

- `type="tel"` odottaa syötteen olevan puhelinnumero. Kosketusnäytöllisissä laitteissa näppäimistöön aukeaa tässä tapauksessa numeerinen näppäimistö.

- `type="search"` käyttäytyy melko samalla tavalla kuin tavallinen tekstilaatikko.

- `type="pattern"` antaa mahdollisuuden tarkentaa kentän syötteen sisältöä käyttämällä säännöllisiä lausekkeita *(eng. regular expressions)*.

- `type="color"` avaa erillisen värivalitsimen selaimissa, joissa se on tuettu, antaen käyttäjälle mahdollisuuden valita väriarvon heksadesimaaliluvun.

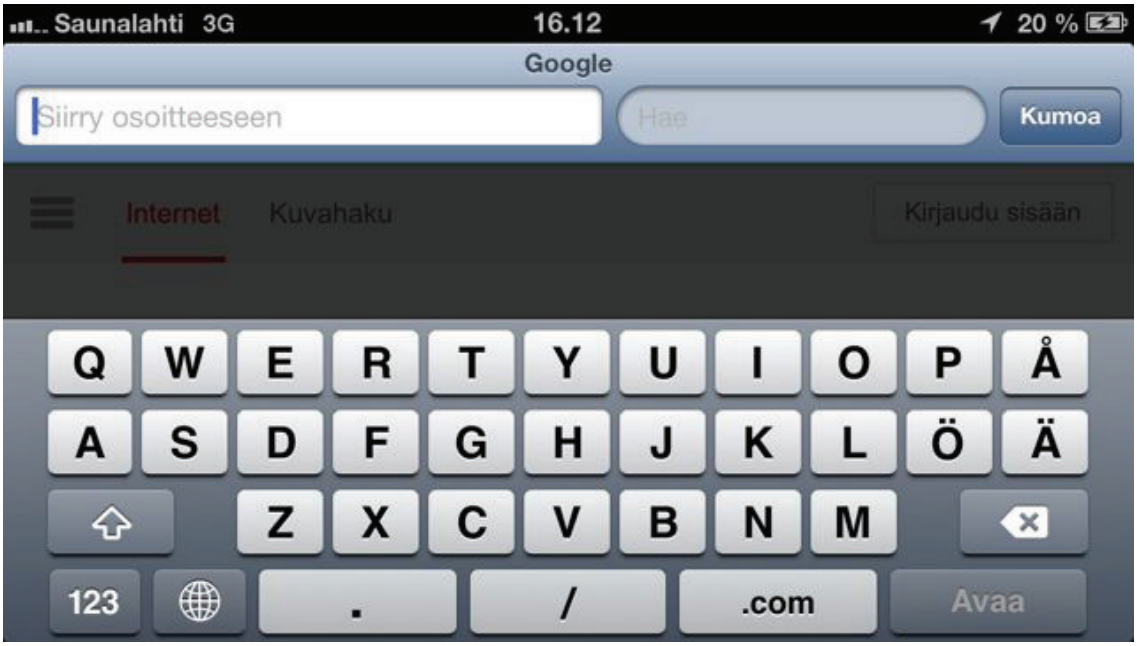
- `type="date"` avaa erillisen kalenterin valittaessa kyseinen kenttä.

- `type="time"` mahdollistaa arvon 24 tunnin formaattiin, esimerkiksi 23:50. Tätä tukevissa selaimissa kenttään tulee jälleen elementti *(eng. spinner)*, joka sallii kyseisiä aika-arvoja.

- `type="range"` luo liukusäätimen, jolla pystyy valitsemaan arvon. Oletuksena on arvoväli nollasta sataan, mutta sen pystyy määrittelemään uudelleen min- ja max-arvoja käyttäen.

Eri selaimet käyttäytyvät eri tavoin tietyissä tyyppikohdissa, jos käyttäjä syöttää väärän arvon. Esimerkiksi tyypin ollessa number, Chrome (v16) tyhjentää kentän sekä kadottaa kohdistuksen ilman, että antaa minkäänlaista palautetta. Samassa tapauksessa Mozilla (v9) antaa syöttää mitä tahansa olettaen sen olevan tavallinen tekstikenttä. Jos numeeriselle arvolle antaa minimi- ja maksimiarvot, Chrome (v16) näyttää varoituksen, kun taas

Firefox (v9) ei tee mitään. Tyypin ollessa tel, jolloin vaatimuksena on jälleen numeerinen arvo, Chrome v16 ja Firefox v9 vain käyttäytyvät kuin se olisi tekstikenttä. Myös color-tyypin selaintuki on niukka. Vain Opera (v11) näyttää tukevan värivalitsinta. (Frain 2012, 247, 249, 251.)



KUVA 1. Kosketusnäytöllisen laitteen näppäimistö, kun syötekentän tyyppi on URL

Uudet syötekenttätyypit ovat erittäin hyödyllisiä varsinkin puhelinta käytettäessä. Se helpottaa huomattavasti käyttäjää, kun esimerkiksi sähköpostiin kirjauduttaessa kosketusnäytöllinen laite näyttää näppäimistössä jo valmiiksi @-merkin. Puhelimessa erikoismerkit ovat muutaman painalluksen takana, joten se nopeuttaa toimintaa, kun tarvittavat näppäimet ovat tarjolla jo kirjaimien kanssa samalla ruudulla, kuten esimerkiksi kuvassa 1. Ilman syötekenttätyypin määritystä kuvan alimmat kolme vaaleaa näppäintä olisi korvattuna välilyönnillä. Siinä tapauksessa piste ja kauttaviiva pitäisi etsiä erikoismerkkinäppäimistön kautta palaten aina välissä kirjain-ruudulle. Samoin Internet-sivun osoitteen verkkotunnus pitäisi kirjoittaa itse.

HTML5-lomakkeiden käyttömahdollisuudessa on kaksi ongelmaa: selainten erilainen tapa toteuttaa toiminnot sekä miten toimia niiden selainten kanssa, jotka eivät tue toimintoja ollenkaan. Tähän on olemassa erilaisia JavaScript-ratkaisuja, jotka mahdollistavat HTML5-lomakkeiden käsittelyn niissä selaimissa, jotka eivät tue niitä, muun muassa myöhemmin mainittu Modernizr. Lisäämällä jQueryn ”Webshims Lib” saadaan

myös lisäominaisuudet toimimaan uusimmissa selaimissa, jotka eivät oletuksen tue niitä, kuten esimerkiksi palaute väärään syötteeseen. Samoin se tarjoaa varasuunnitelmia selaimille, jotka eivät pysty toteuttamaan jotakin tiettyä ominaisuutta. Kyseistä jQuery-kirjastoa käyttämällä voi huoletta käyttää HTML5-lomakkeita tietäen, että kaikki käyttäjät näkevät mitä heidän pitäisikin nähdä. (Frain 2012, 258.)

3.2 CSS3

CSS3 perustuu CSS2:n moduuliin eli CSS3 käyttää CSS2:n määrittelyä ytimenään. Kukin moduuli lisää toiminnallisuutta ja/tai korvaa osan CSS2:sta. Tarkoituksena on, että CSS-moduulit eivät ole ristiriidassa, vaan että ne lisäävät toiminnallisuutta sekä tarkentavat määritelmiä. (Frain 2012, 24.)

Koska nykyiset tietoliikenneverkon nopeudet vaihtelevat suuresti, pitää tarkastella kaistanleveyttä ja luodun sivuston latausaikaa. Mobiililaitteissa latausajat muistuttavat vanhoja aikoja vuosia sitten. Onneksi CSS3:n avulla voidaan huomattavasti vähentää muun muassa kuvien käyttöä. Se antaa työkaluja luoda sivustoja, jotka latautuvat ennätysajassa. (Frain 2012, 137.) Vain muutamalla rivillä koodia, ilman kuvia, CSS3 voi tuottaa näyttöelementtejä, kuten pyöristettyjä kulmia, taustaliukuvärejä, tekstivarjoja, laatikkovarjoja, mukautettua typografiaa sekä useita taustakuvia. Myönnettäköön, että viimeisin edellyttää kuvan. Mikäli tämä ei riitä, on paljon perusvuorovaikutusta, jossa on aiemmin turvauduttu JavaScriptiin, mutta jota voidaan nykyään käsitellä puhtaasti CSS3:lla. (Frain 2012, 138.)

CSS3-moduulin määritelmä ei ole vielä vahvistettu ja kaikki ehdotetut ominaisuudet eivät vielä ole kokonaisuudessaan käytössä kaikissa selaimissa. Tämän vuoksi selaintoimittajat käyttävät toimittajan etuliitettä testatakseen uutta ”kokeellista” CSS-ominaisuutta. Tämä auttaa selaimen valmistajaa toteuttamaan uuden CSS3-moduulin. Esimerkiksi pyöreät reunat tulisi toteuttaa seuraavanlaisesti (Frain 2012, 139):

```
.round {
-khtml-border-radius: 10px; /* Konqueror */
-rim-border-radius: 10px; /* RIM */
-ms-border-radius: 10px; /* Microsoft */
-o-border-radius: 10px; /* Opera */
-moz-border-radius: 10px; /* Mozilla (e.g Firefox) */
```

```
-webkit-border-radius: 10px; /* Webkit (e.g. Safari and Chrome) */
border-radius: 10px; /* W3C */
}
```

On hyväksyttävää listata jokainen etuliite eri selaimia varten. Todellisuudessa harva tekee niin. Sen sijaan he keskittyvät selaimiin, joilla sivuja useimmiten selataan tai tarkistavat mitkä selaimet tukevat ominaisuutta ennen kuin kirjoittavat määritelmän. Ei ole väliä vaikka listaisi kaikkien selainten etuliitteet, se on silti nopeampi, elegantimpi ja kestävämpi tapa kuin kuvien käyttö. Ennen sivuston työstämistä on viisasta tarkastella selaimen käyttötilastoja. Tämän jälkeen on parempi käsitys siitä, mille selaimille on luotava erityistä tukea. Esimerkiksi jos aika ja budjetti ovat tiukoilla saattaa olla parempi jättää sivustolta valmistajakohtaisia etuliitteitä selaimiin, joiden käyttöaste on alle kolmen prosentin. (Frain 2012, 140-141.)

3.2.1 Uudet ominaisuudet

Monipalstaisuus *(eng. multiple columns)*

CSS3 mahdollistaa useamman palstan luomisen ilman sisällön erottamista eri merkintäelementein ja tyylein. Tekstisisältö voi virrata palstoille joko tietyllä palstaleveydellä tai tietyllä määrällä palstoja. Kun on määritelty tietty palstaleveys, niin näkymäikkunan koolla ei ole väliä, sisältö ulottuu aina määritellyn leveyden mukaisesti. Näkymäikkunan muuttaminen vain säätää palstojen määrää dynaamisesti. Tämä takaa kauniin responsiivisen ulkoasun, joka vaatii minimaalisesti työtä. (Frain 2012, 142, 144.)

Rivitys *(eng. word wrapping)*

Monesti tulee vastaan tilanne, jossa pitkä URL pitää lisätä pieneen tilaan. CSS3 korjaa tämän ongelman yksinkertaisella ilmoituksella, joka toimii myös vanhemmissa versioissa, jopa niinkin pitkälle taaksepäin kuin Internet Explorer 5.5 asti. (Frain 2012, 145.)

Osamerkkijonoa vastaavat attribuuttivalitsimet *(eng. substring matching attribute selectors)*

CSS3 antaa mahdollisuuden valita elementtejä alimerkkijonoja vastaavien attribuuttivalitsimien perusteella. Tämä mahdollistaa elementin valitsemisen perustuen sisällön määritteelle. Määritteitä on kolme vaihtoehtoa: alkaa etuliitteellä, sisältää ilmentymän sekä loppuu päätteeseen. Tämä kuulostaa monimutkaiselta, mutta ei kuitenkaan ole sitä. `Element[attribute^="value"]` tarkistaa alkaako sana arvokohtaan merkityllä sanalla. Attribuutin jälkeisen merkin vaihtaminen muuttaa sen merkityksen. `Element[attribute*="value"]` tarkistaa sisältääkö elementti kyseisen ilmentymän. Viimeisenä `element[attribute$="value"]` tarkistaa loppuuko elementti tiettyyn päätteeseen. (Frain 2012, 147-148.)

Rakenteelliset näennäisluokat *(eng. structural pseudo-classes)*

Mitä useammin koodaa sivustoja, sitä useammin tulee vastaan sama ongelma, joka täytyy ratkaista uudelleen ja uudelleen. Tästä tyypillinen esimerkki on navigointipalkki, joka usein koostuu tasaisin välein esiintyvistä `<li>`-linkeistä. Oletettavasti siihen tarvitaan marginaalia oikealle sekä vasemmalle puolella paitsi ensimmäisessä ja viimeisessä linkissä. Tämä on onnistunut suhteellisen helposti luomalla ensimmäiselle ja viimeiselle elementille omat luokat, mutta CSS3 mahdollistaa valitsimet, joilla se onnistuu yksinkertaisemmin. Jo CSS2.1:ssä oli valitsin, jolla pystyi soveltamaan ensimmäistä kohdetta luettelosta: `li:first-child`. Kuitenkin CSS3 lisää valitsimen myös viimeiselle: `li:last-child`. Käyttämällä näitä yhdessä ei tarvita ylimääräisien luokkien merkitsemistä. Näiden lisäksi on olemassa `:nth-child()`-valitsija, jolle pystyy antamaan sulkeisiin erilaisia määrityksiä, kuten esimerkiksi joka toinen linkki on erivärinen. Vielä yksi kätevä valitsija on `:not`, jolla pystyy valitsemaan kaikki, mitkä eivät ole jotakin muuta. Nämä ovat vain muutamia esimerkkejä tärkeistä rakenteellisista näennäisluokista. Aiemmin on pitänyt käyttää JavaScript-kirjastoa, kuten jQuerya, jotta on voinut tehdä monimutkaisia valintoja, mutta CSS3 usein poistaa tämän tarpeen. On myös lohdullista tietää, että CSS3-valitsinmoduulilla on jo W3C-suositus, joten se on hyvin kehittynyt moduuli, joka ei todennäköisesti tule paljon tästä muuttumaan. (Frain 2012, 149-150, 154, 158, 161.)

RGB-väritys

RGB (Red, Green, Blue) on väritys, joka on ollut olemassa jo vuosikymmeniä. Se toimii määrittelemällä eri arvoja punaiselle, vihreällä sekä siniselle värikomponentille. Arvo on määritelty CSS:ssä heksadesimaaliarvolla esimerkiksi `color: #FE0208;`. CSS3:lla voi yhtä hyvin määritellä saman asian RGB-arvolla: `color: rgb(254, 2, 8);`. (Frain 2012, 169.)

HSL-väritys

RGB-värityksen lisäksi CSS3 mahdollistaa HSL-arvojen (Hue, Saturation, Lightness) määrittelyn. Tätä on ilo käyttää, koska siitä on suhteellisen helppo ymmärtää mitä väriä arvo edustaa. HLS toimii 360 asteen väriympyränä. Ensimmäinen luku kuvastaa värisävyä *(eng. hue)*, jossa keltainen on 60°, vihreä 120°, sinivihreä 180°, sininen 240°, magenta 300° ja punainen 360°. Seuraavat kaksi arvoa määrittelevät kylläisyyden *(eng. saturation)* ja valoisuuden *(eng. lightness)* prosenttiosuuden. Kun on määritellyt väriarvon HSL:lle, sitä on helppo muunnella muuttamalla ainoastaan kylläisyyden ja valoisuuden arvoja. HSL:n merkitseminen on myös helppoa, esimerkiksi: `color: hsl(359, 99%, 50%);`. Kuten arvata saattaa RGB ja HSL eivät ole tuettuja Internet Explorerin versioilla alle IE9:n. Tämän vuoksi nämä selaimet tarvitsevat varmistuksen ennen kyseisiä arvoja. Tähän käy hyvin varmistukseksi värin määritys heksadesimaaliarvolla. (Frain 2012, 170-172.)

Läpinäkyvyyskanavat *(eng. alpha channels)*

HSL:n ja RGB:n käyttö mahdollistavat läpinäkyvyyskanavien käytön eli toisin sanoen toinen elementti näyttää, mitä toisen alla näkyy. HSLA-väritys on syntaksiltaan samanlainen kuin HSL. Pelkän HSL:n sijaan on ilmoitettava arvo HSLA lisäämällä läpinäkymättömyyden arvo *(eng. opacity)*. Tämä merkitään kymmenjärjestelmäarvolla siten, että nolla merkitsee täysin läpinäkyvää ja ykkönen täysin läpinäkymätöntä. Tämänkin käyttö on varsin yksinkertaista, esimerkiksi: `background-color: hsla((0, 0%, 100%, 0.8));` tai vaihtoehtoisesti `rgba(255, 255, 255, 0.8);`. (Frain 2012, 172-173.)

Tekstivarjotus *(eng. text-shadow)*

Yksi eniten käytetyistä ominaisuuksista on tekstin varjostaminen. Sillä on laaja tuki modernien selaimien keskuudessa sekä Internet Explorer 9:ssä ja sitä uudemmissa versioissa. Elementille annetaan esimerkiksi syntaksi `text-shadow: 1px 1px 1px #cccccc;`. Pikasääntönä voidaan pitää, että arvot menevät aina oikealle ja sitten alas. Näin ollen ensimmäinen arvo on varjon määrä oikealle, toinen on määrä alas ja kolmas arvo kuvaa hämäryyden määrää eli etäisyyttä, jonka varjo kulkee kunnes haalistuu tyhjyyteen. Viimeinen arvo kuvastaa tietenkin varjon väriä, joka on myös korvattavissa HSL(A)- ja RGB(A)-värityksillä. Varjostuksen saa ylös ja vasemmalle käyttämällä negatiivisia arvoja kahdessa ensimmäisessä syntaksin kohdassa. On myös mahdollista toteuttaa moninkertainen tekstivarjostus *(eng. multiple text-shadow)* erottamalla pilkun jälkeen uudelleen uudet varjomääritelmät. (Frain 2012, 176, 180-181.)

Laatikon varjostus *(eng. box-shadow)*

Laatikon varjostus seuraa täysin samaa kaavaa kuin tekstivarjostus. Esimerkiksi `box-shadow: 0px 3px 5px #444444;`, jossa on vaakatason arvo, pystytason arvo, hämärrys ja viimeisenä väri. Tämä ominaisuus ei kuitenkaan ole selaimissa laajasti tuettu, joten on viisasta käyttää selaimien etuliitteitä yhteensopivuuden maksimoinniksi. (Frain 2012, 181.)

Syvennys *(eng. inset shadow)*

Laatikon varjostus -ominaisuutta voidaan käyttää myös syvennyksen luomisessa. Tämä tarkoittaa sisäpuolelle kohdennettua elementtiä, toisin kuin laatikon varjostus, jonka varjostus on ulkopuolella. Syntaksi luodaan samalla tavalla, mutta ennen arvoja lisätään sana *"inset"*. Kaikki toimii kuten ennenkin, mutta inset ohjeistaa selaimen asettamaan varjostuksen sisäpuolelle. Sekä laatikon varjostuksessa, että syvennyksessä on myös mahdollista käyttää moninkertaista varjostusta erottamalla pilkun jälkeen uudelleen uudet varjomääritelmät. (Frain 2012, 182-184.)

Taustan liukuvärit *(eng. background gradients)*

Ainoa haitta taustan liukuvärien käytössä on niiden heikko tuki selaimissa. Internet Explorer 9:llä ja sitä vanhemmilla versioilla ei ole sisäänrakennettua tukea sille. Kuitenkin liukuvärit on tuettu useimmissa muissa selaimissa, tosin etuliitteitä käyttäen. Vaihtoehtoisesti vanhemmille selaimille on joskus parempi määritellä kiinteä taustaväri, mikäli ne eivät ymmärrä liukuvärien sääntöjä. Liukuvärien syntaksi saattaa olla hieman sekava, esimerkkinä lineaarisen liukuvärin syntaksi: `background: linear-gradient(90deg, #ffffff -50%, #e4e4e4 50%, #ffffff 100%);`. Ensimmäinen arvo määrittelee suunnan, josta liukuvärjäys alkaa. Toinen arvo kertoo aloittavan värin sekä kohdan, josta väritys alkaa. Kolmannessa määritellään väri, jota kohti liukuvärjäys kulkee ja kuinka kauan. Viimeinen arvo määrittelee liukuvärjäyksen niin sanotun loppupisteen. (Frain 2012, 188.)

Moninkertaiset taustakuvat

Yleinen ulkoasuvaatimus on, että sivulla on erilainen taustakuva sivun yläosassa ja erilainen sen alaosassa. Vaihtoehtoisesti saattaa olla kaksi kuvaa saman elementin sisällä. Ennen tämä on vaatinut ylimääräisien divien luomista, mutta CSS3:lla se on varsin helposti ratkaistavissa. Syntaksi on hyvin yksinkertainen: `background: url('../img/1.png'), url('../img/2.png'), url('../img/3.png');`, jossa listan ensimmäinen kuva esiintyy ylimpänä selaimessa. Se on hyvin tuettu, Internet Explorer 8:n ja sitä vanhempien ollessa ainoat poikkeustapaukset. (Frain 2012, 203-204.)

Siirtymät *(eng. transitions)*

On yleinen käytäntö, että hyperlinkit tyylitellään hover-elementillä. Hover-tila on oletuksena aseteltu niin, että se muuttuu toiseen tilaan kohdistettaessa kursori sen päälle. Kuitenkin CSS3 antaa nimensä mukaisesti siirtymämahdollisuuden toisesta tilasta toiseen, jolloin käyttäjä näkee muutoksen tasaisen rauhallisesti. Siirtymäesittely lisätään aina alkuperäiseen elementtiin hover-tilan sijaan. Siirtymään on mahdollista vaikuttaa neljällä eri tavalla: mihin se vaikuttaa, missä ajassa siirtymä tapahtuu, miten se muuttuu keston aikana sekä viiveeseen ennen siirtymän aloittamista, esimerkiksi `transition: all 1s ease 0s;`. (Frain 2012, 210-212.)

2D-muunnokset *(eng. 2D transformations)*

Huolimatta siitä, että CSS-muunnokset kuulostavat samalta kuin CSS-siirtymät, ne ovat täysin eri asioita. Siirtymät muuttuvat yhdestä tilasta toiseen, kun taas muunnokset määrittelevät mikä elementistä tulee. Verrattuna 3D-muunnoksiin, 2D-muunnokset ovat laajemmin toteutettavissa, selainviisaita sekä helpommin kirjoitettavia. Muunnosmoduuli sallii seuraavat muunnokset elementille: skaalaus, kääntäminen näytöllä, kiertäminen määrittelemättömällä määrällä, vääristäminen sekä siirto ja muotoilu pikselin tarkkuudella. Näistä esimerkiksi kääntäminen toimii seuraavalla tavalla: `transform: rotate(3600deg);`. (Frain 2012, 216-217.)

3D-muunnokset *(eng. 3D transformations)*

Tällä hetkellä 3D-muunnokset vaativat melko paljon säätämistä toimiakseen joustavalla ulkoasulla. Lyhykäisyydessään joustavalla ulkoasulla tarkoitetaan nettisivujen sisällön esittämistä kaikkien laitteiden näkymäikkunoille sopivalla tavalla. Tuki on rajoittunutta eivätkä 3D-muunnokset ole usein vahvin ratkaisu, jos ollaan rakentamassa sivustoa eri selaimille. Ne ovat tuettuja WebKit-selaimissa (Safari, Chrome) ja Firefoxin versioilla 10+, mutta niitä ei tue Internet Explorerin versioista muut kuin IE10. Kuitenkin huolimatta siitä ettei niillä ole tukea ”työpöytä”-selaimissa, ne ovat hyvin tuettuja Androidilla (alkaen v3) sekä iOS:llä (kaikki versiot). Mahdollisuudet 3D-muunnoksilla ovat kuitenkin erittäin lupaavia. Kun selaintukea laajennetaan ne tarjoavat CSS:n kautta mahdollisuuden moniin hienoihin tehosteisiin, jotka ovat tällä hetkellä riippuvaisia JavaScriptistä. (Frain 2012, 11, 221, 227-228.)

Animaatiot

Mikäli on joskus työskennellyt Flashin kanssa, on siitä välitön hyöty CSS3-animaatioiden tekemisessä. CSS3 käyttää animaatioissaan samaa avainkehysmerkintätapaa *(eng. keyframing conventions)* kuin Flashissa sekä muissa aikajanapohjaisissa sovelluksissa. Animaatiot ovat hieman laajemmin tuettuja kuin 3D-muunnokset. (Frain 2012, 228.)

3.2.2 CSS3 ja vanhat selaimet

Selaimien vanhemmat versiot, kuten Internet Explorer 6, 7 ja 8, mielellään ohittavat CSS3:n ominaisuuksia, joita ne eivät voi käsitellä. Tämä antaa mahdollisuuden parantaa suunnittelua paremmin varusteltuihin selaimiin varmistaen samalla riittävän peräänty-misen vanhemmista. (Frain 2012, 25.) Kuten aina web-kehityksessä, se tulisiko CSS3:sta käyttää omassa ulkoasussa, riippuu kyseisen sivuston tarpeista. CSS3:lla voi lähinnä parantaa sivustoa sen sijaan, että tarjoaa olennaisia toimintoja. Kehittäjien ja asiakkaiden tulisi olla tyytyväisiä siihen, että elementit saattavat näyttää hieman erilaisilta eri selaimissa. Se, mikä osa ulkoasusta on tärkeää toiminnan kannalta tai minkä tulisi näyttää hyvältä, on omakohtaisesti tulkittavissa. On kuitenkin hyvä tietää, että on olemassa monia liitännäisiä *(eng. polyfills/plugins)*, joilla voidaan lisätä CSS3:n toiminnallisuus vanhoihin IE-selaimiin. (Frain 2012, 138.) Opinnäytetyön lopussa käsitellään tarkemmin Internet Explorerin tarvitsemia lisäkirjastoja HTML5:n ja CSS3:n toiminnallisuuden lisäämiseksi.

3.3 JavaScript

JavaScript on selainohjelmoinnin kieli, jolla ohjelmoidaan web-selain suorittamaan haluttuja toimintoja (Korpela 2011, 47). Se on suosituin ohjelmointikieli. JavaScript on kevyt kieli HTML:lle ja verkkosivuille, palvelimille, tietokoneille, kannettaville, tableteille sekä älypuhelimille. HTML-sivulle lisätty JavaScript toimii kaikilla moderneilla selaimilla. (W3Schools, 2013.)

HTML-dokumenttiin voi sijoittaa rajattoman määrän skriptejä. Skriptit voivat olla `<body>`- tai `<head>`-osassa tai molemmissa. Yleinen käytäntö on sijoittaa toiminnot `<head>`-osaan tai sivun alareunaan. Näin ne ovat kaikki samassa paikassa eivätkä häiritse sivun sisältöä. (W3Schools, 2013.)

JavaScriptistä on monia eri versioita käytettävissä. Kaikista versioista vakain kaikille selaimille (alemman hintaluokan laitteista älypuhelimiin) on 1.5 ja se on versio, johon tulisi panostaa eniten. Uusimmat versiot toimivat vain selainten viimeisimmillä versioilla ja lisäykset eivät ole yhteensopimattomuuden arvoisia. (Firtman 2013, 452.)

Kirjastot

Monet JavaScript-kirjastoista pohjautuvat standardeille, jotka eivät ole niin selkeitä ja vaikka ne on valmistettu ja testattu tunnetuilla pöytäkoneen selaimilla (Internet Explorer, Firefox, Safari, Chrome, Opera), niitä ei ole testattu kaikilla mobiiliselaimilla. Mikäli mahdollista kirjaston käyttöä olisi hyvä välttää. Jos välttäminen ei ole mahdollista, tulisi niitä käyttää ainoastaan älypuhelimille ja olla tietoinen siitä, että jotkut ominaisuudet ja liitännäiset eivät välttämättä toimi kunnolla. Paras tapa olisi ladata vain ne osat kirjastoa, joita todella tarvitsee. (Firtman 2013, 484.)

jQuery on nopea, pieni ja monipuolinen JavaScript-kirjasto. Se tekee muun muassa HTML-tiedoston läpikäymistä ja manipulointia, tapausten käsittelyä ja animointia. jQuery on ohjelmointirajapinta, joka toimii lukuisilla selaimilla. Monipuolisuuden ja laajennettavuuden yhdistelmällä jQuery on muuttanut tapaa, jolla miljoonat ihmiset kirjoittavat JavaScriptiä. Tunnetuimmat jQueryn käyttäjät ovat WordPress ja Wikipedia. jQueryn uusin julkaisu jQuery 2.0 on jättänyt taakseen tuen Internet Explorerin versioille 6, 7 ja 8. Vastineeksi se on kevyempi, nopeampi ja sitä voidaan käyttää JavaScript-ympäristöissä, joissa IE:n tarvitsema lisätuki aiheuttaa usein omia ongelmiaan. jQuery kuitenkin tukee edelleen aiempia versioita, joilla saadaan toimivuus vanhemmille selaimille. (jQuery 2013.)

jQuery Mobile on yhtenäinen HTML5-pohjainen käyttöliittymäjärjestelmä kaikille suosituille mobiililaitteiden alustoille. Se on rakennettu jQueryn sekä jQuery-käyttöliittymän perustalle. jQuery Mobile on joustava ja sen ulkoasu on helposti vaihdettavissa teemojen mukaisesti. Sen kevyt koodi on rakennettu asteittaiselle parannukselle *(eng. progressive enhancement)*, jonka merkintäkieli noudattaa web-standardeja eli se on käyttökelpoinen kaikissa selaimissa. jQuery Mobile vie viitekehyksen ”kirjoita vähemmän, tee enemmän” –sanomaa seuraavalle tasolle. Sen sijaan, että kirjoittaisi yksilöllisiä sovelluksia kullekin mobiililaitteelle tai käyttöjärjestelmälle, jQuery Mobile tarjoaa puitteet, joilla suunnitella laadukas verkkosivusto tai sovellus, joka toimii kaikilla suosituimmilla älypuhelimien, tablettien sekä pöytätietokoneiden alustoilla. jQuery Mobilella on laaja tuki valtaosassa moderneista työpöydän, älypuhelinten, tablettien sekä e-lukijoiden alustoista. Tämän lisäksi myös ominaisuudet puhelimissa ja vanhemmat selaimet ovat tuettuja asteittaisen parannuksen ansiosta. (jQuery Mobile, 2013.)

4 SIVUSTON TOTEUTUS

Laitteet eivät ole erilaisia vain niiden erilaisten teknisten valmistuksien ja rajoituksien vuoksi, vaan myös koska ihmiset käyttävät niitä eri tavalla. Laitteita käytetään satunnaisesti sohvalla tai sängyllä, pitkiä aikoja työpöydällä sekä pikaisesti pitkin päivää eri tilanteissa. Syöttötapoina on kauko-ohjain, eleet, hiiri, näppäimistö, kosketus sekä anturit. Näytön koko voi olla seinän kokoinen, pöytäkokoinen, tabletti tai vieläkin pienempi. Vaikka on mahdollista saada yhden käyttöliittymän sivusto toimimaan useammassa laitteessa, saattaa se aiheuttaa kompromisseja sekä riisuttuja ominaisuuksia, jotka eivät ole edukseen kaikkien laitteiden käyttökokemuksille. (Wroblewski 2011, 116-117.)

Kun ollaan toteuttamassa mobiiliversiota olemassa olevasta nettisivusta, täytyy ymmärtää, että ollaan mobilisoimassa sivua, ei minimoimassa sitä. Minimoimalla pöytäkoneen nettisivua yksinkertaisesti näytetään sama sisältö pienemmälle näytöllä. Mobilisointi on enemmän, se vaatii asiayhteyden ymmärtämistä sekä tarjoaa sivuston palvelut ja sisällön tavalla, joka on käytännöllinen ja mahdollistaa nopean käynnin käyttäjälle. (Firtman 2012, 142.) Ainoa mihin voi tällä hetkellä mobiilissa luottaa on muutos. Tämän vuoksi kannattaa ottaa riskejä, kokeilla uusia asioita ja hyväksyä se, että kaikkia rajaviivoja laitteiden, selaimien ja verkon välillä ei ole vielä vedetty. (Wroblewski 2011, 109.)

Helposti ajatellaan, että mobiilisivuja luotaessa kyseessä ovat samat asiat, kuin pöytäkonesivuilla, mutta pienemmässä pakkauksessa. Todellisuudessa mobiiliverkko eroaa monella tavalla pöytäkoneverkosta. Firtmanin (2013, 40) mukaan suurimmat eroavaisuudet ovat:

- Hidas verkkoyhteys pitkällä viiveellä
- Hitaampi laitteisto sekä pienempi käytettävissä oleva muisti
- Erilainen selailukokemus ja selaimen toiminta
- Erilainen käyttäjän asiayhteys
- Liian monta mobiiliverkkoselaimen eri versiota markkinoilla samanaikaisesti
- Toiset selaimet liian rajallisia, toiset liian innovatiivisia
- Useat selaimet ilman identiteettiä (ovat vain selaimia), dokumentointia tai kehittäjän työkaluja

- Testaus ja virheenkorjaus mobiiliverkossa on haaste ja eroaa perinteisestä web-kehityksestä

Nykypäivän lähes kaikki markkinoilla olevat älypuhelimet, esimerkiksi iPhone- ja Android-pohjaiset laitteet, voivat lukea ja näyttää kokonaisia työpöydän verkkosivuja. Käyttäjät haluavat saman kokemuksen mobiiliverkossa, kuin mikä heillä on pöytäkoneiden verkkosivuilla. Jotkut tilastot jopa osoittavat, että käyttäjillä on tapana valita webversio mobiiliversion sijaan käyttäessään älypuhelinta. (Firtman 2013, 38.)

Mikäli toteuttaa verkkosivuille erillisen mobiiliversion joustavan ulkoasun sijaan, olisi hyvä antaa mobiilikäyttäjälle mahdollisuus päästä myös alkuperäiselle sivustolle. Osa on saattanut vierailla tietyllä sivustolla useasti pöytäkoneella ja näin ollen ehkä tietää tarkalleen mistä löytää haluamansa tiedon. Jos tarjolla on ainoastaan mobiiliversio, saattaa se ärsyttää käyttäjää, joka joutuu niin sanotusti täysin uudelle sivustolle, josta joutuu etsimään kyseisen tiedon. Mobiiliversiosta saattaa olla myös karsittu ylimääräistä tietoa, jolloin käyttäjä ei välttämättä näe kaikkea etsimäänsä. Tämän vuoksi olisi hyvä antaa, laitteesta riippumatta, käyttäjälle pääsy kumpaankin versioon.

Niin pöytäkone- kuin mobiilisivustoa luotaessa tulisi yhtenä tärkeänä osana muistaa sivuston käytettävyys. Mitä vaikeakäyttöisemmät sivut ovat käyttäjälle, sitä todennäköisimmin hän menee seuraavalla kerralla mieluummin kilpailijan sivuille. Vierailin hiljattain nettisivulla, joka on toteutettu muutama kuukausi sitten. Sivut ovat erittäin hienot ja niistä huomaa, että ulkoasun suunnitteluun on käytetty paljon aikaa. Sivuilta on kuitenkin kokonaan unohdettu sen käytettävyys. Navigaation linkit ovat niin epäkuvaavat, että jopa yhteystietojen löytäminen vaatii keskittymistä ja monia virhepainalluksia sivuston väärille sivuille. Yksinkertaisilta kuulostavien asioiden löytäminen vaatii enemmän aikaa kuin sen kuuluisi. Mielenkiinnosta tarkistin, onko sivustosta toteutettu mobiiliversio, kun kuitenkin on kyse nykypäivän sivustosta kyseisellä ulkonäöllä. Puhelimella katsottuani en huomannut sivuissa mitään eroa. Paitsi, että sillä ne ovat entistäkin vaikeakäyttöisemmät, koska näyttötilaa on huomattavasti vähemmän. Tästä voi ehkä päätellä, haluanko enää koskaan mennä kyseisille sivuille. Tämän vuoksi ei riitä, että sivusto näyttää kivalta: yhtä tärkeää on, että ne ovat myös helpot käyttää.

4.1 Responsiivinen sunnittelu

”Responsiivisella designilla tarkoitetaan käyttöliittymäsuunnittelua, jonka avulla yhtä web-sovellusta voi käyttää eri laitteilla ilman että käyttökokemus merkittävästi kärsii.” (Leiniö, 2012). Tällä tarkoitetaan ensisijaisesti sitä, että sisällön rakenne ja asettelu suunnitellaan niin, että se sopii jokaiselle laitteelle ja niiden normaaleille käyttötavoille. Termiä käytetään usein viittaamaan samaa merkitystä oleviin kuvauksiin, kuten joustava suunnittelu, mukautuva ulkoasu, eri laitteiden muotoilu sekä sopeutuva suunnittelu (Frain 2012, 10). Responsiivinen muotoilu antaa melkein aina parempaa ja kattavampaa käyttökokemusta kuin tavallinen kiinteän leveyden muotoilu (Frain 2012, 33). Kun ulkoasusta tekee joustavan, se näyttää hyvältä sekä luettavalta kaikilla näytöillä, ei vain pöytätietokoneella. Responsiivinen suunnittelu on puhtaasti HTML:ää ja CSS:ää. Yksinkertaisesti luodaan sääntöjä CSS:ään perustuen käyttäjän laitteen näytön kokoon. (Agarwal, 2012.) Suositeltu tapa toteuttaa mobiilisivustoja on käyttää samaa URL:ia jokaiselle laitteelle. Kuitenkin myös alaverkkotunnus on edelleen käytettävissä. (Firtman 2013, 45.)

Kun suunnittellaan ulkoasua mobiililaitteelle, tulisi aina muistaa, että sivun pitäisi olla suunniteltu mukautuvaksi. Tähän on kaksi syytä: mobiililaitteella on kaksi käyttötapaa, pysty- ja vaaka-asento sekä rajallinen tila, joten tulisi käyttää ruudulla jokainen käytettävissä oleva pikseli. (Chuan 2012, 28.) Responsiivisuus on yksinkertainen ja tehokas tapa käyttää yhtä HTML-dokumenttia, joka automaattisesti sopeutuu käyttäjän näytön tilaan sekä suuntaan (Firtman 2013, 134).

Responsiivinen menetelmä on sopiva, kun asiakas haluaa, että sivut ovat selkeät, nopeat ja helposti ylläpidettävät, ja kun asiakas hyväksyy, että sivut toimivat hieman eri tavalla eri selaimissa. Mikäli asiakas haluaa sivujen olevan identtiset kaikilla mahdollisilla selaimilla ja jos 70 % odotettavissa olevista käyttäjistä käyttävät Internet Explorer 8 tai aiempia versioita, responsiivinen menetelmä ei ole parhain mahdollinen vaihtoehto. (Frain 2012, 32.)

Responsiivisella suunnittelulla on monia hyviä puolia. Se säästää aikaa ja rahaa, kun ei tarvitse ylläpitää erillisiä nettisivuja pöytätietokoneille sekä mobiililaitteille. Se on myös hyväksi sivuston hakusijoituksille, kun jokaisella sivuston sivulla on sama URL-osoite.

Näin ollen ei tarvitse huolehtia tilanteista, joissa jotkut sivustot linkittävät mobiisisivustolle ja jotkut pöytätietokoneen sivustolle. Mikäli sivusto seuraa käyttäjävirtaa, antaa luotettavamman kuvan, kun molempien versioiden käyttäjät näkyvät samassa raportissa. Ylläpito on huomattavasti helpompaa, kun ei ole palvelinpuolen osatekijöitä. Pitää ainoastaan muuttaa taustalla olevaa CSS-tiedostoa, jotta saa muutettua sivun ulkoasua tai asettelua tietyllä laitteella. (Agarwal, 2012.)

Responsiivisella suunnittelulla on myös muutama haittapuoli. Se saattaa lisätä sivuston kilotavuja, koska sivuston täytyy ladata CSS-tyylejä ja JavaScript-tiedostoja, jotka olisivat muutoin tarpeettomia. Toinen ongelma on kuvat. Mobiilisivustolle ei haluta korkearesoluutioisia kuvia, mikä on vaikea saavuttaa joustavalla ulkoasulla, ellei sitten turvaudu palvelinpuolen kiertotapaan. Viimeisimpänä on oppimisvaihe, joka vie aikaa silloin, kun lisätään responsiivisuus jo olemassa olevalle sivustolle. Joskus voi olla järkevämpää aloittaa tyhjästä sen sijaan, että luo nykyiset kiinteälevyiset nettisivut joustavaksi. (Agarwal, 2012.)

Eri sivustojen lähdekoodeja tutkimalla huomaataan, että kaikilla on jokseenkin oma tapansa toteuttaa sivut joustaviksi. Lähdekoodia tutkimalla pystyy myös osittain päättelemään, onko sivusto toteutettu joustavaksi alun alkaen vai onko joustavuus lisätty vasta jälkeenpäin. Jo valmiisiin sivuihin on huomattavasti vaikeampi toteuttaa responsiivisuus kuin sivuihin, joihin se lisätään sivun toteuttamisen yhteydessä. Silloin on mahdollista ajatella kaikki elementtien koot etukäteen mobiiliin sopivaksi, kun taas valmiissa sivustoissa pelataan jo ennaltaan määritellyillä koo'illa.

4.2 Mobiili- vai pöytäkoneversio ensin?

Sivustoa luotaessa tulisi muuttaa koko nykyinen lähestymistapa. Sen sijaan, että aloitetaan kiinteällä työpöydän leveydellä skaalaten sitä pienempään sisältöön pienimmille näkymäikkunoille, tulisi suunnitella pienin näkymä ensin ja sitten vähitellen lisätä ulkoasua ja sisältöä suurempiin näkymäikkunoihin. (Frain 2012, 11.) Kun ihmiset käyttävät mobiiliversiota, ei ole harvinaista, että he haaveilevat työpöytäversionkin olevan "niin yksinkertainen" (Wroblewski 2011, 19). Suunniteltaessa mobiiliversio ensin, ei vain avata uusia mahdollisuuksia kasvulle, vaan voidaan myös tuottaa parempi kokonais-

käyttökokemus nettisivulla. Se pakottaa keskittymään ja priorisoimaan rajoitukset luontaisesti mobiilisuunnitteluun. Se antaa myös mahdollisuuden kekseliäisiin ratkaisuihin hyödyntämällä uusia ominaisuuksia alkuperäisiin mobiililaitteisiin sekä niiden käyttötavoille. (Wroblewski 2011, 1.) Osa maailman suurimmista verkkoyhtiöistä on myös omaksumassa ”mobiili ensin” -suunnittelun periaatteen. Muun muassa Googlen puheenjohtaja Eric Schmidt, Facebookin suunnittelujohtaja Kate Aronowitz sekä Adoben teknologiajohtaja Kevin Lynch puhuvat mobiilisuunnittelun ensisijaisuuden puolesta. (Wroblewski 2011, 2.)

Edellisestä kappaleesta tuli mieleen mielenkiintoisena esimerkkinä juurikin Facebook. Kyseinen yhteisöpalvelu on perustettu jo lähes kymmenen vuotta sitten, jolloin Internetin käyttö puhelimilla oli melko minimaalista. Sivusto on siis perustettu pöytäkoneversio ensin -periaatteella. Noin viiden vuoden ajan olen seurannut sen kehitystä uusine hyvine ja huonoine ominaisuuksineen. Moni käyttäjä haluaisi pysyä siinä tutussa ja turvallisessa ja tavallaan pelkäävät muutosta. Nykypäivän vaatimukset yhteisöpalveluissa ovat kuitenkin korkeat, joten kehitys on välttämätöntä. Olen ollut Facebookin mobiilisovelluksen käyttäjä vasta puolisen vuotta ja miellyin alusta alkaen sen helppokäyttöiseen käyttöliittymään. Muutama toiminnallisuus oli sillä puolella toteutettu huomattavasti paremmin kuin pöytäkoneversiossa, jossa sama ominaisuus vaati monia hiiren klikkauksia. Viime aikoina olen kiinnittänyt huomiota siihen, että Facebook on alkanut muuttaa pöytäkoneversiota mobiiliversion kaltaiseksi. Tämä tulee varmasti helpottamaan käyttäjiä, kun samat ominaisuudet löytyvät suurin piirtein samoista paikoista. Tavallaan, kun asiaa tarkemmin miettii, he toimivat tällä hetkellä osittain mobiiliversio ensin -periaatteella.

Moni asiantuntija ja nettisivu puhuu ”mobiili ensin” –suunnittelun puolesta. Itse en ole nähnyt kenenkään tuttuni todellisuudessa toteuttavan tätä käytännössä. Yksi sanoi kokeilleensa sitä ja palanneensa vanhaan suunnittelutapaansa uuden tavan hankaluuden vuoksi. Todennäköisesti mobiiliversion suunnitteluun ei käytetä tarpeeksi aikaa. Se koetaan hankalaksi, koska se on uutta, mitä ei vielä osata kunnolla. Aiheeseen tarkemmin tutustuttuani voin ainakin itse allekirjoittaa väitteen. Suunnitteluvaiheessa on yllättävän monta asiaa, joita ei tule edes ajateltua. Monen asian oppii myös yrityksen ja erheen kautta. Tämäkin asia on ratkaistavissa asiaa tutkimalla ja opettelemalla.

4.3 Mobile Meta Tags

iOS ja Android -selaimet sekä muun muassa Opera Mobile perustuvat WebKit-selainmoottoriin. Näissä selaimissa on mahdollisuus syrjäyttää oletusnäkymää kutistumasta käyttämällä meta viewport -elementtiä `<meta name="viewport" content="initial-scale=1.0,width=device-width" />`. `Content="initial-scale"` viittaa sisällön kokoon, esimerkiksi 2.0 näyttää sen kaksinkertaisena, 0.5 puoliksi. `Width="device-width"` kertoo selaimelle, että leveyden tulisi olla yhtä suuri kuin laitteen leveyden. On myös mahdollista säädellä kuinka monta kertaa käyttäjä pystyy zoomaamaan tai määritellä, ettei se ole mahdollista. (Frain 2012, 50-51.)

Näkymäikkunan sisällölle on mahdollista määritellä leveys ja korkeus, ja se voi olla suurempi tai pienempi kuin koko näytöllä näkyvä alue. Jos ollaan luomassa mobiiliystävällistä nettisivustoa, sen ei tulisi tarvita zoomausta lähelle eikä kauas, joten selaimen skaalaukselle voi antaa arvon 1:1 (nykyinen näkymä = näkymäikkunan näkyvä alue = näkyvä alue). Meta-tagille on mahdollista antaa erilaisia attribuutteja, jotka on lueteltu taulukossa 3. (Firtman 2013, 204-205.)

TAULUKKO 3. Näkymäikkunan metadatan attribuutit

`width`	Näkymäikkunan leveys
`height`	Näkymäikkunan korkeus
`initial-scale`	Näkymäikkunan näkyvä alue
`user-scalable`	Määrittelee onko zoomaus mahdollista vai ei
`minimum-scale`	Näkymäikkunan minimizoomaus
`maximum-scale`	Näkymäikkunan maksimizoomaus
`target-densitydpi`	Määrittelee kuinka hallita eri resoluutioita riippuen näkymäikkunan leveydestä

4.4 CSS3 Media Queries

Mediakyselyt *(eng. media queries)* antavat mahdollisuuden kohdistaa tiettyjä CSS-tyylejä tiettyjen laitteiden näytön vaatimuksiin (Frain 2012, 35). Pystysuunnassa kapeal-

la näytöllä kaksi lohkoa esitetään allekkain, kun taas vaakasuunnassa leveämmällä näytöllä ne esitetään vierekkäin.

Mediakyselyt ovat jo laajalti käytössä ja niillä on myös laaja selainten tuki. Lisäksi on helppo toteuttaa JavaScript-pohjaisia korjauksia, joilla vahvistetaan tavallisia vanhentuneita selaimia, kuten Internet Explorerin versiot 6, 7 ja 8. (Frain 2012, 35.)

Yleisimmät mediakyselyt, joita on totuttu käyttämään ovat laitteen näyttöikkunan leveys *(eng. width)* sekä laitteen näytön leveys *(eng. device-width)*. Mikäli kuitenkin tarvetta ilmenee on olemassa myös muita mediakyselyitä, joita voi testata. Kaikki käytettävät ominaisuudet on lueteltu taulukossa 4. Lukuun ottamatta kahta viimeisintä ominaisuutta voidaan käyttää minimi- ja maksimivaihteluväliä. (Frain 2012, 39-40.)

TAULUKKO 4. Mediakyselyiden ominaisuudet

Leveys (`width`)	Näyttöikkunan leveys.
Korkeus (`height`)	Näyttöikkunan korkeus.
Laitteen leveys (`device-width`)	Laitteen näytön leveys.
Laitteen korkeus (`device-height`)	Laitteen näytön korkeus.
Näyttösuunta (`orientation`)	Pysty- vai vaakasuunta.
Kuvasuhde (`aspect-ratio`)	Perustuu näyttöikkunan leveyteen ja korkeuteen.
Laitteen kuvasuhde (`device-aspect-ratio`)	Perustuu laitteen näytön leveyteen ja korkeuteen.
Väri (`color`)	Bittien lukumäärä per värikomponentti.
Väri-indeksi (`color-index`)	Laitteen merkintöjen määrä värihakutaulukossa. Arvojen oltava positiivisia numeroita tai nolla.
Mustavalkoisuus (`monochrome`)	Testaa kuinka monta bittiä per pikseli on mustavalkoisessa kehyspuskurissa. Arvon oltava kokonaisluku eikä se voi olla negatiivinen.
Resoluutio (`resolution`)	Näytön resoluutio eli tarkkuus.
Skannaus (`scan`)	Pitkälti televisioissa käytetty.
Verkko (`grid`)	Osoittaa onko laite verkossa vai bittikarttaan perustuva.

Mediakyselyitä on mahdollista käyttää kahdella tavalla. Sen voi sisällyttää HTML-tiedoston head-elementtiin esimerkiksi `<link rel="stylesheet" type="text/css" media="screen" href="screen-styles.css">`. Media-kohtaan on mahdollista laittaa erilaisia ominaisuuksia, joiden toteutuessa sivusto käyttää kyseistä tyylitiedostoa. Esimerkiksi lauseella `media="screen and (orientation: portrait)"` CSS-tiedostoa käytetään, kun näyttö on pystysuunnassa. Siihen on myös mahdollista esittää minimi- ja maksimileveydet, jolloin näytön koon muuttuessa käytetään eri tyylitiedostoa. Toinen tapa on lisätä ehdot CSS-tiedostoon. Esimerkiksi kirjoittamalla `@media screen and (max-device-width: 400px) { }` ominaisuudet kirjoitetaan aaltosulkeiden sisälle. Mikäli ehdot toteutuvat, ominaisuudet tulevat käyttöön sivustolla. (Frain 2012, 38-39.) Nykyaikaiset selaimet ovat tarpeeksi älykkäitä sivuuttamaan mediakyselyt, jotka eivät ole niille tarkoitettu, mutta se ei aina estä niitä lataamasta tiedostoja. Mediakyselyjen erottamisesta erillisiin tiedostoihin voi olla hieman hyötyä. Toisaalta erilliset tiedostot lisäävät http-pyyntöjen määrää, joka puolestaan tekee sivuista hitaammat. Kuitenkin ehkä suositeltavampaa olisi lisätä mediakyselyt jo olemassa oleviin tyylitiedostoihin. (Frain 2012, 41.)

Mediakyselyjen yksittäinen käyttö on sopivaa, jos on tiedossa kohdelaitteen erityisvaatimukset. Tällä hetkellä, kun muutetaan näyttöikkunan kokoa, ulkoasu muuttuu vain sitä mukaan, kuin mitä mediakyselylle on määritelty. Sen jälkeen se pysyy staattisena, kunnes seuraava mediakysely on saavutettu. Tarkkojen tyylien kirjoittaminen jokaiselle näyttöikkunalle ei tee oikeutta tulevaisuuden laitteille. Tämän vuoksi ulkoasun tulisi olla joustava ennen kuin se saavuttaa mediakyselyn määrittelyt. (Frain 2012, 59.) Tällä tarkoitetaan siis sitä, että selaimen ikkunaa pienennettäessä sivusto muuttuu koko ajan joustavasti pienentäen sisältöä ja mahdollisia kuvia. Kun ikkunan koko saavuttaa CSS:ssä määritellyn mediakyselyn koon, sivusto muuttuu kyseiselle koolle suotuisammaksi muuttaen esimerkiksi sisällön ja kuvien sijaintia. Tämän jälkeen se pienentyy taas joustavalla periaatteella seuraavaan mediakyselyyn asti, kunnes kaikki määritelmät ovat käyty läpi.

4.5 Päätteiden näytöt

Varmistaakseen, että jokaisella mobiililaitteen käyttäjällä on mahdollisuus päästä käsiksi sivuston sisältöön, suunnittelija tarvitsee ratkaisun, joka toimii peruspuhelimilla, älypuhelimilla sekä kaikilla siitä välistä (Wroblewski 2011, 12). Tämän lisäksi tulee ottaa huomioon myös tabletit ja tietokoneet. Tableteilla ja puhelimilla on huomioitava myös molemmat näyttösuunnat: pysty- ja vaakasuunta.

Eri mobiililaitteilla on oma oletusnäkymän leveys. Sen määrittelemättä jättäminen aiheuttaa useimmiten odottamattomia tuloksia. Kun määrittelee oletusnäkymän leveyden laitteen leveydeksi, se kertoo selaimelle ettei sivua skaalata sopimaan laitteen koko alueelle (Chuan 2012, 44, 46.) Suositeltavat vaihtelupisteet voisivat olla CSS3-mediakyselyille 320px (iPhone pystytaso), 480px (iPhone vaakataso), 600px (Android-tabletit), 768px (iPad- ja Galaxy -tabletit) sekä 1024px (iPad vaakataso ja pöytätietokoneet). (Agarwal, 2012.)

Puhelimet ja tabletit

Ensimmäinen sarja älypuhelimia oli pääosin sidottu 320x480 pikselin resoluutioon (Wroblewski 2011, 19). Ei ole olemassa mitään standardia mobiililaitteiden näyttöjen resoluutioille. Jollain puhelimella resoluutio saattaa olla 128x128 pikseliä, kun taas toisella 720x1280 pikseliä. Kolmannen sukupolven iPadissä on 2048x1536 pikselin näyttö. Kosketusnäytöllisissä laitteissa on tyypillisesti suurempi resoluutio kuin näppäimistöllisissä laitteissa, koska niissä ei ole tarvinnut varata tilaa näppäimistölle. (Firtman 2013, 54.)

Resoluutio ei ole ainoa asia, joka pitää mobiilaitteiden näytöissä huomioida. Yhtä tärkeä piirre on näytön fyysiset mitat (tuumissa tai senttimetreissä, viistosti tai mitattuna leveys/korkeus) tai suhde tämän sekä resoluution välillä, joka tunnetaan sanalla PPI (pixels per inch, pikselit tuumalla) tai DPI (dots per inch, pisteet tuumalla). Tämä on erittäin tärkeää, sillä vaikka ensimmäinen ajatus voi olla, että 128x160 resoluution näyttö on pienempi kuin 240x320, voi se olla väärä johtopäätös. Esimerkiksi iPad 2:ssa ja kolmannen sukupolven iPad:ssa on molemmissa 9,7 tuuman näyttö, mutta aiemman resoluutio on 1024x768 ja jälkimmäisen 2048x1536 pikseliä. (Firtman 2013, 56.)

Tietokoneet

Web-suunnittelijat kiistelivät vuosia käytettävissä olevasta näyttötilasta pöytätietokoneella. Lopulta kuitenkin päädyttiin 1024x768 pikselin ratkaisuun. (Wroblewski 2011, 18.) Nykypäivänä suurimmat näytöt ovat jopa 1920x1080 pikseliä.

Retina-näyttö

Retina-näyttö on Applen tavaramerkki, mikä nykypäivänä tarkoittaa ”kaksinkertaista tarkkuutta (nelinkertaisesti pikseleitä käytettävissä)” (Firtman 2013, 57). iPhone 4 oli ensimmäinen laajalti käytetty laite, jossa on käytössä korkearesoluutioinen näyttö. Sen näyttö on 960x640 pikseliä ja tarkkuus 326 pikseliä tuumalla. Tämä tarkoittaa kaksinkertaista resoluutiota verrattuna aikaisempaan iPhone 3GS –versioon sekä kaksinkertaista resoluutiota vuoden 2011 MacBook Pro 15’’ -kannettavaan tietokoneeseen. On myös odotettavissa, että monet laitteet tableteista ja kannettavista tietokoneista pöytätietokoneisiin seuraavat perässä. (Frain 2012, 292.)

Kuvien käyttö on hieman pulmallista, koska retina-näytöllä 200x400 kuva onkin 400x800. Kuvaa ei ole suunniteltu siihen kokoon, joten se venyy samassa mittasuhteessa ja todennäköisesti näyttää sumealta. Helpoin ratkaisu on luoda kuva kaksinkertaisena ja img-tagissa pienentää se puoleen. Tämä kuitenkin vaatii enemmän latausaikaa, joten se ei välttämättä ole paras ratkaisu. Tilanteesta riippuen voi päätyä ratkaisuun, jossa käytetään JavaScriptiä tai CSS:ää kuvien korjaamiseen. Jos nettisivuilla on vain vähän kuvia, JavaScript voisi olla paras ratkaisu. Tällöin kaksinkertaistetaan kuvan koko ja puolitetaan se jQuery-koodilla. Toinen tapa on käyttää kahta erillistä kuvaa, esimerkiksi toinen retina-kuvana ja toinen normaalina. Tällöin JavaScriptiä käyttämällä attribuutin kautta muutetaan kumpaa kuvaa käytetään riippuen näytön resoluutiosta. Myös yksi vaihtoehto on CSS:ää käyttäen mediakyselyillä kohdentaa kuva oikealle resoluutiolle. (Simpson, 2013.)

CSS:ää ja mediakyselyitä käyttämällä on mahdollista luoda graafisia ulkoasuja, jotka mukautuvat dynaamisesti käyttäjän ympäristöön. Suurin rooli on img-elementin srcset-attribuutilla. Mobiilikäyttöön tarkoitetuille sivuille listataan kolme eri kuvaa srcset-attribuutille. Tällä tavalla suurimmalla kuvalla täytyy olla kuvapistetiheyden avainsana

leveyden sijaan, koska ei ole keinoa määrittää ääretöntä leveyttä. Esimerkiksi (Hickson, 2013.):

```
<img src="pear-mobile.jpeg" srcset="pear-mobile.jpeg 720w, pear-
tablet.jpeg 1280w, pear-desktop.jpeg 1x" alt="The pear is juicy.">
```

Eri sivuja tutkiessani huomasin, että samat tekniikat toistuivat suunnittelijasta toiseen. Ilmeisesti ei ole vain yhtä oikeaa ratkaisua tehdä joustavat sivut ottaen huomioon korkearesoluutioiset näytöt. HTML5:n uusi img-tagin srcset-attribuutti kuulostaa mielenkiintoiselta ratkaisulta, mutta se on ilmeisesti vielä toistaiseksi työn alla. Se mitä tekniikka kannattaa käyttää riippuu tapauskohtaisesti sivujen sisällöstä ja kuvien määrästä. Kokeilemalla eri tapoja löydetään paras ratkaisu juuri siihen tarkoitukseen.

4.6 Syötetavat

Yksi suurimmista eroista mobiililaitteiden ja tietokoneiden välillä on se, miten ollaan vuorovaikutuksessa näyttöön. Tietokoneella käytetään hiirtä liikkumiseen ja klikkauksiin. Mobiilinäytöillä vuorovaikutus tulee pääasiassa kosketuksesta ja eleistä.

Mobiilisivustolla liikuttaessa käytetään erilaisia tekniikoita. Jokainen mobiiliselaimen käyttäjä käyttää yhtä tai useampaa näistä tiloista navigoidakseen. Tilat ovat: kohdistaminen, osoittaminen, kosketus sekä monikosketus. Kohdistaminen on yleisesti ottaen käytössä ei-kosketuksellisissa laitteissa, joissa käyttäjä käyttää kohdistinta näppäimistöstä navigoidakseen linkityksien välillä ja selaten verkkosivua. Kursorinavigoinnilla liikutaan hiiren osoittimelle, jota liikutellaan nuolinäppäimillä. Kosketusnäyttönavigointi saattaa tuntua itsestään selvältä, mutta on oltava tietoinen siitä, että navigointia voi käyttää sormella sekä kynällä. Tarkkuudella on suuri ero suunnittelussa. Jotkut laitteet ovat myös monikosketuksellisia, jonka avulla käyttäjä voi valita monia objekteja yhtä aikaa sekä lisätä eleiden määrää. (Firtman 2013, 45-47.)

Kosketusnäyttö

Nykyaikaisissa älypuhelimissa on pääosin kosketusnäyttö (Chuan 2012, 36). On olemassa monia johdonmukaisia eleitä, joilla voi olla vuorovaikutuksessa kosketusnäytön kanssa. Voi napauttaa, tuplanapauttaa, vetää, pyyhkäistä, nipistää, levittää, painaa ja

napauttaa, painaa ja vetää sekä useita vaihtoehtoja kääntää. Elepohjainen toiminto usein alkaa yhdessä sovelluksessa tullen hitaasti hyväksytyksi useammissa paikoissa. Ajan mittaan käyttäjät odottavat toiminnon toimivan kaikkialla, joten ei tulisi pelätä kokeilla mitä eleillä voi toteuttaa. (Wroblewski 2011, 74-75.)

Kosketusnäytöllä on omat rajoitteensa. Täytyy huomioida, että esimerkiksi toiminnot, jotka tulevat näkyviin kursorin kohdistamisesta *(eng. hover)* eivät toimi samalla tavalla kosketusnäytöllä. Yksinkertaisesti ei ole kursoria, jota kohdistaa toiminnon ylle. Tämän vuoksi kaikki toimet, jotka ovat riippuvaisia hiirestä, on määriteltävä uudelleen. Hover-valikot toimivat niin sanotusti kaatopaikkoina; toimet eivät ole tarpeeksi tärkeitä näkyäkseen näytöllä, mutta ovat silti sen verran tärkeitä, että ne pitää näyttää kursorin kohdistamisella. Näitä toimintoja ei saa kadottaa siirryttäessä pöytäkoneelta mobiililaitteelle. Mobiililaitteella on muutamia vaihtoehtoja hover-valikolle. Mikäli tiedot ovat tärkeitä, oikea lähestymistapa voisi olla näyttää ne suoraan näytöllä tai pudotusvalikossa *(eng. drop-down menu)*. Jos taas toimet tai sisältö ovat looginen seuraava askel, niin ne voisi näyttää painalluksesta tai pyyhkäisystä. Tämäkin tapa voi olla ärsyttävä, koska se vaatii ylimääräisen askeleen käyttäjän etenemiselle. Jos hoverin sisältö on laaja, voi olla parasta siirtää sen sisältö erilliselle näytölle eli toisin sanoen uudelle sivulle. Viimeisenä vaihtoehtona on hävittää koko hoverin sisältö. Poistamalla ylimääräiset vaihtoehdot, jotka eivät ole arvokkaita käyttäjille, yksinkertaistaa käyttöliittymää sekä vähentää omaa luomista ja ylläpitämistä. (Wroblewski 2011, 78-82.)

On olemassa muutamia käytännöllisiä vihjeitä kosketusnäytöllisiin laitteisiin liittyen. Usein käytettäville nappuloille ja linkeille tulisi varata tarpeeksi iso klikattava alue, minimissään 40 pikseliä leveys sekä korkeus. Vähemmän käytettäville nappuloille sen sijaan riittää pienempi alue, minimissään 25 pikseliä. Klikattavien elementtien väliin tulisi jättää kunnollinen väli, 20 pikseliä tai enemmän. Lomakkeen syöttökentän mahdolliset vihjeet tulisi sijoittaa kentän ylle tai alle eikä vasemmalle tai oikealle. Kosketuslaitteiden virtuaalinen näppäimistö zoomaa yleensä kenttään, jolloin käyttäjä ei kirjoittaessaan näe sen vierellä lukevia vihjeitä. Yleisimpien tekstikenttien vieressä olisi hyvä käyttää toimintoa automaattiseen tyhjentämiseen. X-symboli kentän vieressä oikealla tyhjentää painettaessa sen sisällön. Yksi hyvä vaihtoehto on käyttää suunnittelumallia, jossa kosketusta jatketaan kauemmin, kaksi sekuntia tai enemmän, jonka seurauksena avautuu ponnahdusikkunavalikko asiaan liittyville vaihtoehdoille. Kosketuseleitä käyt-

täessä tulisi käyttää niistä ilmeisimpiä seuraamalla käyttöjärjestelmien malleja, kuten pyyhkäisy oikealle tai vasemmalle kuvagalleriassa navigoidessa. Mikäli toiminnot eivät ole ilmeisiä, tulisi antaa vaihtoehtoisia painikkeita tai esimerkki opastusohjelmaan ensimmäisellä latauksella. (Firtman 2013, 146-147.)

Zoomaaminen

Selaimen hallinnoimisessa on kaksi zoomausvaihtoehtoa. Ensimmäinen tarjoaa peruszoomausominaisuuden: websivu muuttuu 1:1 mittakaavassa sen alkuperäiseen suunnitteluun. Jos malli ei mahdu näyttöön, vierityspalkki ratkaisee ongelman. Toinen tyyppi on älykäs zoomausominaisuus: websivua voidaan zoomata miten käyttäjä haluaa ja zoomaus vaikuttaa fontin kokoon, kuviin ja websivuun kokonaisuudessaan. (Firtman 2013, 47.)

4.7 Sivuston sisältö

Mobiilisivuja luodessa olisi hyvä olla käyttöliittymän tunnistus *(eng. user agent detection)*. Tämä voi helpottaa skriptin uudelleenohjausta tai auttaa määrittelemään pitäisikö ladata tai jättää lataamatta jotakin perustuen käyttöliittymään. (Chuan 2012, 100.)

Kun pohditaan mobiililaitteen käyttötapoja, on hyvä kuvitella ihmiset ”yhdeksi silmäksi ja yhdeksi peukaloksi”. Yksi peukalo, koska he todennäköisesti pitävät laitetta yhdellä kädellään sekä käyttävät yhtä peukaloa hallitakseen sitä. Yksi silmä, koska monissa paikoissa, joissa mobiililaitetta käytetään, on käytössä vain ihmisten osittainen huomio. Tämä ajattelumalli pakottaa yksinkertaistamaan mobiilisivustoa, jotta se olisi ymmärrettävissä ja käytettävissä tällaisissa tilanteissa. (Wroblewski 2011, 25-26.)

4.7.1 Mobiilisivuston keventäminen

Matkapuhelinkäyttäjän sivuston latausaikaa voi pienentää hallitsemalla sekä tiedostojen kokoa että määrää ja siten myös HTTP-pyyntöjä, joita se lähettää laitteeseen. Sivuston keventäminen onnistuu huomioimalla seuraavat kohdat (Wroblewski 2011, 23-24):

- Kokoamalla yhteen ja minimoimalla CSS- ja JavaScript-tiedostot.
- Rajoittamalla tai poistamalla riippuvuudet raskaisiin JavaScript-kirjastoihin. Varsinkin jos käyttää vain yhtä tai paria toiminnoista.
- Rajoittamalla CSS-listajärjestelmän käyttöä.
- Käyttämällä asianmukaista HTTP headersia varmistaakseen, että tiedostot ovat tarkoituksenmukaisesti tallennettu selaimen muistiin.
- Tarvittaessa hyödyntämällä nykyaikaisten selaimien ominaisuuksia, kuten HTML5:sta.
- Käyttämällä CSS3:n uusia ominaisuuksia (esimerkiksi gradientti ja pyöreäreunaisuus kuvien tilalla), kuitenkin menemättä liiallisuuksiin.

Tekstisisältö ja fontit

Aika on yleensä kallisarvoista mobiilikäytössä ja lataukset saattavat maksaa, joten käyttäjille tulisi tarjota mahdollisimman pian se tieto, mitä he etsivät. Kohdistamalla sisällön ensin ja navigaation vasta sitten, käyttäjät saavat tiedon ja tehtävät hoidettua nopeasti (Wroblewski 2011, 52, 66.)

Mobiilikäyttäjille ei ole välttämättä tarpeellista näyttää koko pöytäkonesivuston sisältöä. Useimmissa tapauksissa käyttäjä etsii jotain tarpeellista tietoa ollessaan esimerkiksi tien päälle. Tällainen tieto voisi olla esimerkiksi paikan osoite tai aukioloajat. Mikäli käyttäjä haluaa saada yksityiskohtaisempaa tietoa sivuilta, voi hänelle antaa mahdollisuuden päästä pöytäkonesivustolle. Sivut ovat hieman kevyemmät, kun sieltä jättää ylimääräiset tekstit pois.

Kirjaisintyypin *(eng. font-family)* muotoilu on ongelma mobiiliselaimissa. Ei ole olemassa mitään standardeja fontteja mobiilikäyttöjärjestelmiin ja useimmilla alustoilla on vain yksi fontti, yleensä sans serif. Jos haluaa määritellä tietyn fontin, tulisi harkita vaihtoehtoisen listan tarjoamista. Jos ensimmäinen fontti ei ole käytettävissä, selain yrit-

tää toista, sitten kolmatta ja niin edelleen. Jos mikään listan fonteista ei ole käytettävissä, se käyttää oletusfonttia. Google Web Fontit ja Adobe Edge Web fontit ovat kaksi hanketta, joita voi käyttää ilmaiseksi ja ne molemmat tarjoavat tuhansia fontteja. Niitä hyödyntämällä voi käyttää kustomoituja fontteja kaikilla nykyaikaisten tablettien ja älypuhelimien selaimilla. (Firtman 2013, 395-396, 398.)

Kuvien käyttö

Mobiililaitteilla vähäisempi määrä kuvia tarkoittaa nopeampaa latausta, mikä on helppo tapa lisätä suorituskykyä. Mediakyselyt mahdollistavat turhien kuvien poisjättämisen pienille näytöille. Mobiilisivustoilla käyttäjällä on yleensä joku tietty määränpää, esimerkiksi tietyn tiedon etsiminen, jolloin hän ei todennäköisimmin kiinnitä niin paljon huomiota itse kuviin sivuilla. Joissain tapauksissa ne saattavat jopa olla vain tiellä.

Mikäli ei käytetä CSS3:a ja halutaan elementille tausta, kannattaa käyttää ohutta graafista siivua, jota toistetaan vaaka- tai pystysuunnassa. Se on melko edullinen kompromissi. Kuva, joka on vain pikselin tai kaksi leveä, ei kuormita kaistanleveyttä ja yhdellä sivulla sitä voidaan käyttää usealle elementille. Kuitenkin muutokset vaativat aina graafisen editorin käyttöä. Joskus on myös mahdollista, että sisältö rikkoutuu yli kiinteiden kokorajojen. Tämä ongelma pahentuu joustavassa ulkoasussa, jossa sivun rakenteen täytyy muuttaa muotoaan esimerkiksi pitenemällä tai laajentumalla. (Frain 2012, 185-186.)

4.7.2 Sijoittelu

Painikkeet ja niiden koko

Koska suurin osa mobiililaitteista on kosketusnäytöllisiä, täytyy kiinnittää huomiota myös nappuloiden kokoon. Pöytäkonesivustolla ei haittaa vaikka nappula olisi yllättävän pienikin, kyllä siihen hiirellä osuu. Sen sijaan kosketusnäytöllä se vaatii jo tarkkuutta sekä hyvää tuuria varsinkin, jos monia nappuloita/linkkejä on vieretysten.

Apple suosittelee, että kosketuskohde olisi 44x44 pistettä *(eng. points)*. Microsoft suosittelee 9 millimetrin kohdetta, vähimmäiskokona 7 mm ja vähimmäisvälistyksenä 2

mm. MIT Touch Lab on päättänyt, että keskiarvo on 10-14 mm sormille ja 8-10 mm sormenpäille. Kosketuskohteen tulisi osua tälle välille, mutta visuaalisesti se voi olla 50-100 % oikean painikkeen koosta. (Wroblewski 2011, 69-70.)

Koska suurin osa ihmisistä on oikeakätisiä (noin 70-90 %), ja he käyttävät peukaloitaan käyttäessään puhelinta yhdellä kädellä, on yleisintä sijoittaa tärkeimmät toiminnot oikealle. Tuhoavat toiminnot, kuten peruuta tai poista, olisi hyvä sijoittaa pois ihmisten mukavuusalueelta. Jos mobiililaitetta pitää oikeassa kädessä, niin joutuu näkemään enemmän vaivaa ja venyttämään peukaloa yltääkseen puhelimen vasempaan yläreunaan. Tällöin vältytään virhepainalluksilta. (Wroblewski 2011, 72.) Kuvassa 2 on havainnollistettu puhelimen näytön käytettävyyttä. Vihreille alueille on helppo osua, kun taas keltaiselle täytyy venyttää peukaloa.

KUVA 2. Puhelimen näytön käytettävyys (Wroblewski 2011, muokattu)

Navigointi

Yleinen ongelma responsiivisessa ulkoasussa on se, että jos sivusto sisältää paljon navigointilinkkejä, niin ne voivat viedä huomattavan osan näytöstä pienemmissä näkymäikkunoissa. Yksi tapa on muuntaa normaali valikko pudotusvalikoksi (kuva 3), mikäli

selain on alle tietyn ikkunaleveyden. Se ei ole välttämätöntä toiminnallisuutta, mutta se antaa enemmän sisältöä sivun yläosaan. (Frain 2012, 287-288, 290.)

KUVA 3. Muut-linkistä avautuva pudotusvalikko

Toiminnot ruudun alaosassa sisällön jälkeen antavat käyttäjille mahdollisuuden tutkia sivun osia. Alareunassa ne esittävät valintoja ja vaihtoehtoja mitä tehdä seuraavaksi, kun käyttäjä pääsee sivun loppuun. Yksinkertaisella ankkurilinkityksellä sivun yläosassa käyttäjän voi ohjata sivun alareunassa olevaan navigaatioon. (Wroblewski 2011, 57.)

Mobiilikäytössä on vain käyttäjän osittainen huomio, joten he tarvitsevat selkeää, kohdennettua muotoilua saadakseen asiat tehdyksi, eivät paljon navigaatiovaihtoehtoja. Minimoimalla vaihtoehdot mobiilinäytöltä ehkäistään virheitä. Vähemmillä vaihtoehdoilla käyttäjät todennäköisesti suoriutuvat tavoitteistaan nettisivuilla, eivätkä he vahingossa napauta itseään muualle. (Wroblewski 2011, 64-65.)

Useita kirjoja lukiessani huomasin, että eri kirjoittajilla on eri mielipide navigaation sijaintiin. Mikäli sen sijoittaa sivun yläreunaan, on mahdollista, että käyttäjä ei selaa koko sivua läpi vaan siirtyy saman tien toiselle sivulle. Jos navigaation taas sijoittaa sivun alareunaan, voi käyttäjä ärsyyntyä siitä, ettei hänelle tarjota sivuvaihtoehtoja heti alussa vaan ne joutuu niin sanotusti etsimään. Moni asia tuntuukin olevan vain eri ih-

misten mielipiteitä eikä niinkään yleisiä suosituksia. On myös ratkaisuja, jotka sopivat joillekin sivuille, mutta toisille eivät.

4.7.3 Tekstikentät ja lomakkeet

Tekstikenttä *(eng. input)* olisi hyvä sijoittaa ylös, jolloin se optimoituu koko näytön leveydelle ja pysyy näkyvillä, kun virtuaalinen näppäimistö tulee esiin vieden puolet käytettävissä olevasta tilasta. (Wroblewski 2011, 87.)

Valitse valikosta -menut *(eng. select menus)* eivät ole välttämättä parhaimpia vaihtoehtoja mobiililaitteille, koska ne vaativat useita napautuksia. Ensin avataan valikko, rullataan haluttu vastaus, valitaan vastaus ja sitten vielä painetaan valmis. Onneksi tällekin on keksitty muita kosketusnäytölle sopivia ratkaisuja. Sen sijaan, että näyttää valikosta tietyn määrän numeroita, voi käyttää esimerkiksi plus- ja miinusnappeja, mikäli kysymyksessä on pieni valikoima vaihtoehtoja. (Wroblewski 2011, 91-94.)

Tutkimukset osoittivat, että valmiiksi esitäytetyillä lomakkeilla käyttäjät täyttävät tiedot neljä kertaa nopeammin kuin tyhjillä lomakkeilla. Esimerkiksi hotellia varatessa huoneen määrän kohdalla voisi olla oletusarvoisesti numero 1, joka säästäisi aikaa ja vaivaa käyttäjältä. (Wroblewski 2011, 94.)

Uudet HTML5:n ominaisuudet auttavat käyttäjiä vastamaan kysymyksiin, jotka edellyttävät erityistä muotoa. Mobiiliselaimella kuten Safarilla, jossa määritellään lomaketyyppi url:ksi, tulee näppäimistöön näppäimet ”.”, ”/” ja ”.com” helpottamaan vastauksen kirjoittamista. Jos tyyppi määritellään email:ksi tulee muun muassa näppäin ”@” näkyviin. (Wroblewski 2011, 96-97.) Tavallisten tekstikenttien täyttämistä on myös mahdollista helpottaa mobiililaitteilla. Ottamalla pois käytöstä automaattisen ison alkukirjaimen sekä tekstinkorjauksen email-, salasana-, URL- sekä merkkikokoriippuvaisissa kohdissa, helpotetaan mobiilikäyttäjää. Kuten edellisessä, myös tässä kohdassa selaimet, jotka eivät tue näitä ominaisuuksia, yksinkertaisesti vain jättävät ne huomioimatta, joten ei ole mitään harmia lisätä niitä sivuille. (Wroblewski 2011, 97.)

4.8 Muuta huomioitavaa

Sivuston testaus

Vaikka mikään ei voita kokeilua käytännössä, voidaan testausta kokeilla myös selaimen ikkunan kokoa muuttamalla. On olemassa erilaisia kolmannen osapuolen liitännäisiä ja selainlaajennuksia, jotka näyttävät nykyisen selain- tai näyttöikkunan pikselit. (Frain 2012, 11). Jos vain mahdollista, muotoilu ja koodi tulisi testata todellisella mobiililaitteella. Simulaattorit ja työpöytäselaimet eivät ole korvikkeita oikeille laitteille. (Wroblewski 2011, 120.) Mobiilinettisivuja ei ole mahdollista toteuttaa joka ikiselle mobiililaitteelle. On olemassa kirjaimellisesti tuhansia älypuhelimia, ja niistä ei tarvitse jokaista testatakseen sivuston toimivuutta. Alle kymmenen riittää, mutta kaksi laitettakin on tarpeeksi. (Chuan 2012, 6.)

CSS Reset

CSS Reset -tiedosto sisältää joukon CSS-määrittelyjä, jotka nollaavat eri selainten erilaiset oletustyylit. Ne lisätään päätyylitiedoston alkuun, jonka jälkeen tyylien määrittelyt voi tehdä itse. Tällöin jokaisella selaimella on sama vaikutus jokaisella elementillä.

Kirjanmerkkikuvake

Nettisivuista on myös mahdollista saada kirjanmerkkikuvake *(eng. mobile bookmark bubble)* mobiililaitteen kotinäkymään. Vaikka kyseinen hieno ominaisuus tuo Internet-sovellusta lähemmäs natiivisovellusta, on siinä yksi ongelma. Ei ole olemassa ohjelmointirajapintaa, jota kutsua kirjanmerkkitoimintoon. Tämän vuoksi monikaan käyttäjistä ei ole tietoinen kyseisestä ominaisuudesta laitteissaan. Ongelman korjaamiseksi muutama kehys *(eng. framework)* tarjoaa kirjanmerkkikuvakkeen käyttäen CSS:ää ja JavaScriptiä. Kyseinen skripti lisää kuvakkeen nettisivujen alareunaan kysyen käyttäjältä haluaako tämä nettisivuista kuvakkeen laitteen kotinäkymään. (Chuan 2012, 102.)

Validointi

Ohjelmakoodin validointi on ensimmäinen askel eliminoida koodausvirheet. Mikäli koodi on epävalidia, selain yrittää tasoittaa virheitä odottamattomilla tavoilla. Mobiiliselaimet ovat vähemmän anteeksiantavaisia virheelliselle koodille. Niillä ei yksinkertaisesti ole resursseja tasoittaa virheitä kuten pöytätietokoneiden selaimilla. (Schmitt & Simpson 2011, 48.)

5 TOTEUTETTU SIVUSTO

Toteutin työharjoitteluni aikana yhdessä opinnäytetyöni tilaajan kanssa ravintola.fi:n mobiiliversion m.ravintola.fi. Mobiiliversion toteuttaminen oli kannattavaa, koska käyttäjä löytää sieltä tiedot, esimerkiksi ravintolan aukioloajat ja sijainnin, nopeammin ollessaan liikenteessä. Mobiiliversiosta haluttiin myös kevyempi.

Mobiiliversio toteutettiin käyttämällä HTML5-merkintäkieltä, koska se tarjoaa enemmän mahdollisuuksia kuin aiemmat versiot. Sivustolla on käytössä HTML5:n uusista elementeistä `header`, `nav`, `section` ja `footer`. Mobiiliversiossa on kolme tilaa, jotka muuttuvat sitä mukaan minkä kokoisen laitteen näytöllä sivustoa selaillaan. Mediakyselyn vaihtelupisteet ovat 480px ja 960px. Mikäli laitteen näytön resoluutio on alle 480 pikseliä, niin käyttäjälle näytetään versio, jonka elementtien leveydet ovat 320 pikseliä. Tässä näkymässä sisältö on yleensä kaikkein yksinkertaisin. Tästä seuraava versio on laitteille, joiden näytön resoluutio on 480-959 pikseliä. Tämä käsittää suurimman osan älypuhelimista sekä tabletit pystysuunnassa. Viimeisin on laitteille, joiden näytön resoluutio yli 960 pikseliä eli pääasiassa vaakasuuntaisille tableteille ja pöytäkoneille. Mobiiliversiolle on annettu muutama ehto mobile meta tageja käyttämällä: `<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no"/>`. Tämä määrittää sivuston näkyvän alueen: sivusto on aina laitteen näkymän levyinen sekä pysty- että vaakasuunnassa. Zoomaus on yleensä tarpeeton mobiiliversiossa, koska sivusto on suunniteltu mobiililaitteille ja sen tulisi mukautua aina laitteelle sopivaksi. Tämän vuoksi on myös määritelty, että zoomaus ei ole mahdollista.

Pyrimme käyttämään sivuston suunnitteluun paljon aikaa, jolloin säästyimme ylimääräiseltä työltä. Suunnittelimme sivustoa yhdessä graafikon kanssa antaen hänelle tiedot, mitä häneltä tarvitsisimme. Koska ravintola.fi:n pöytäkoneversio oli jo toiminnassa, suunnittelimme mobiiliversion vasta tämän jälkeen. Meillä ei siis ollut mahdollisuutta ”mobiiliversio ensin” -periaatteeseen. Mobiilisivuston ulkoasuun otettiin mallia pöytäkoneen versiosta, jotta ne olisivat mahdollisimman yhdennäköiset, vertaa kuvaa 4 ja kuvaa 5.

KUVA 4. Pöytäkoneversio

Lähtökohtana sivustolle oli mobiilikäyttäjälle tarpeettomien tietojen poisjättäminen, yksinkertaisuus, helppokäyttöisyys sekä nykyaikaisuus. Lähdimme liikkeelle miettien mitä suurimpia eroavaisuuksia pöytäkonesivustolla on verrattuna mobiilisivustoon. Ensimmäisenä tuli tietenkin mieleen se, että mobiilikäyttäjät todennäköisesti haluavat löytää etsimänsä tiedon mahdollisimman pian, joten oli selkeää ettei heille tarjota kaikkea tietoa, mitä pöytäkoneversio sisältää. Voisi olettaa, että mobiilikäyttäjä myös tietää mitä tietoa hän sivustolta etsii, koska harvemmin mobiilisivustoa selaillaan vain ajan kuluksi. Mikäli käyttäjä on Tampereella, hän todennäköisesti haluaa nähdä ainoastaan Tampereen ravintoloiden tarjonnan, joten päädyimme ratkaisuun, jossa etusivulla listataan ensin kaupungit. Kaupungin valitsemalla käyttäjä näkee listan kyseisen kaupungin ravintoloista sekä ainoastaan sen kaupungin tapahtumat ja lounaat. Ravintolan valitsemalla käyttäjä näkee ravintolan kuvauksen, yhteystiedot, aukioloajat, tapahtumat, virtuaaliesittelyn sekä mahdollisen yhteyden Facebookiin. Jokaisen sivun alareunassa on navigaatio, joka sisältää linkit myyntipalveluun, galleriaan, yrityssivulle sekä yhteystietoihin. Etusivulta on mahdollista päästä sivulle, jossa kerrotaan kanta-asiakasedut sekä sivulle, josta voi varata pöydän haluamaansa ravintolaan.

Kun meillä oli selvillä suunnilleen mitä sivut sisältävät, aloimme pohtia miten ne esitetään käyttäjille. Helpoimmin toteutettavat sivut olivat vaakatasossa olevalle tabletille, sillä kaikki tiedot näkyvät oikeastaan lähes koko näytöllä (kuva 5). Tämän vuoksi ei myöskään ollut niin paljon väliä missä mikäkin elementti sijaitsee. Mobiiliversio ei ole jatkuvasti joustava eli se on toteutettu käyttämällä mediakyselyitä, jolloin sivu muuttuu tietyin mittavälein pysyen muuten staattisena. Selaimen ikkunaa pienentämällä huomaa kyseiset muutokset.

KUVA 5. Tablettiversio vaakatasossa

Seuraavaksi lähdimme miettimään miten näytetään samat tiedot pystytasoiselle tabletille sekä puhelimille, joiden näyttö on yli 480 pikseliä leveä. Päädyimme ratkaisuun, jossa ensin näytetään tiedot kyseisestä ravintolakonsernista ja vasta tämän jälkeen on lista kaupungeista ja tapahtumat (kuva 6). Palkkien sijainti muuttuu hieman riippuen mistä sivusta on kyse. Esimerkiksi sivulla, joka näyttää kaupungin ravintolat, tapahtumat ja lounaat, näkyy ensin ravintolalistaus sekä lounaat vierekkäin ja niiden alapuolelle vasta lista tapahtumista. Ravintolan sivulla näytetään ensin yhteystiedot, kun taas itse ravintolan kuvaus tulee vasta sen jälkeen. Halusimme tarjota käyttäjille suoraan ne tiedot, joita he todennäköisimmin mobiilikäytössä haluavat löytää.

Viimeisimpänä piti vielä miettiä miltä sivusto näyttäisi puhelimella katsottaessa. Kyseinen näkymä näkyisi puhelimilla, joiden näytön leveys on maksimissaan 480px (kuva 7). On siis melko selvää, että kaikki tiedot esitetään allekkain aloittaen oleellisimmasta.

Etusivu alkaa jälleen ravintolakonsernin esittelylle, jota seuraa kaupunkilistaus. Viimeisimpänä on vasta lueteltu tapahtumat. Kaupunkisivulla on ensimmäisenä ravintolalistaus. Myös tässä näkymässä ravintolan sivu esittää tiedot melko yhtäläisesti pystytasoisen tabletin kanssa.

KUVA 6. Tablettiversio pystytasossa

KUVA 7. Puhelinversio

Ravintola.fi:n sivujen sisältö tulee sisällönhallintajärjestelmän kautta, mikä on osaltaan hankaloittanut kuvien käyttöä sekä tekstisisältöä mobiiliversiossa. Koska eri versioilla on eri leveydet, muutamassa kohdassa samasta sivusta on pitänyt luoda kaksi versiota sisällönhallintaan, toinen pöytäkoneelle ja toinen mobiililaitteille. Edelleenkin mobiilisivustolla saattaa tulla vastaan kuva, joka on toteutettu alkuperäisille sivuille unohtaen ottaa huomioon, että se näkyy myös mobiiliversiossa. Päädyimme kuitenkin linkittämään suurimmilta osin saman sisällön mobiiliversioon, jotta sivujen päivitys olisi helpompaa eikä tarvitsisi jokaisella kerralla tehdä muutoksia kahdessa paikassa. Ajan kuluessa olen huomannut, että sivuston ylläpito olisi huomattavasti helpompaa mikäli alkuperäinen sivusto olisi joustava, jolloin ei tarvitsisi päivittää kahta eri sivustoa. Olen kiinnittänyt huomiota siihen, että mobiilisivusto unohtuu melko helposti varsinkin, kun päivityksiä tulee hieman harvemmin. Toisaalta kyseisessä tapauksessa pöytäkoneversio

sisältää hyvin paljon sisältöä, isoja kuvia sekä mobiilisivustolle tarpeetonta JavaScript-toiminnallisuutta. Jos siihen olisi toteuttanut responsiivisen ulkoasun, olisi se todennäköisesti ollut melko raskas varsinkin vanhemmille puhelimille.

Sivusto m.ravintola.fi on toteutettu ennen opinnäytetyön kirjoittamisen aloittamista. Nyt aiheeseen enemmän paneuduttuani huomaan muutamia ratkaisuja, joita mobiilisivustolla voisi työstää paremmaksi. Mobiiliversioon tulee sisällönhallinnan kautta yksi lomake, joka on alun perin toteutettu pöytäkonesivustolle. Tämän vuoksi siinä ei ole otettu huomioon HTML5:n uusia syötekenttien tyyppejä, jotka helpottaisivat mobiilikäyttäjää. Yhtenä lisäyksenä mobiiliversioon voisi lisätä linkin, josta on pääsy pöytäkoneen versioon käyttäjän niin halutessaan. Tämänhetkinen kaupunkinavigaatio on hyvä, mutta sen voisi toteuttaa eri tavalla. Sivuja työstäessä meillä oli päällimmäisenä ajatuksena se, että käyttäjä joutuisi painamaan nappuloita mahdollisimman vähän ja tieto näkyisi pääasiassa rullaamalla sivua ylös ja alas. Kaikki kaupungit listattuna vie kuitenkin melko paljon ylimääräistä tilaa, joten listan olisi voinut toteuttaa erikseen avautuvalla pudotusvalikolla. Sivuilta puuttuu kokonaan niin sanottu päänavigaatio. Etusivulla on kuva, jota painamalla pääsee sivulle, joka näyttää asiakasedut sekä kuva, josta pääsee varaamaan pöydän. Sen lisäksi on kaupunkilistaus, tapahtumien lopussa linkki tapahtumasivulle sekä sivun alareunassa linkit myyntipalveluun, galleriaan, yrityssivulle ja yhteystietoihin. Jatkamalla matkaa kaupungin sivulle katoaa mahdollisuudet päästä asiakasetuihin sekä pöydän varaukseen. Kyseiseltä sivulta pääsee etusivulle ainoastaan, jos tietää ylhäällä olevan logon olevan linkki etusivulle tai edellinen-näppäimellä mobiililaitteen valikosta. Ravintolan sivulla näkyy erillinen linkki etusivulle, mutta puuttuu edelleen mahdollisuus varata pöytä, jolloin tätä varten pitäisi palata taas etusivulle. Sivustosta tulisi käyttäjälle varmasti selkeämpi, jos kaikki linkit olisi listattuna samassa paikassa.

Sivustojen ulkoasusta pyrittiin tekemään mahdollisimman yhtenäinen, mutta ei kuitenkaan samanlainen. Toinen muutos, minkä tekisin, olisi sivun yläosassa olevan kuvasliderin vaihtaminen joko kokonaan pois tai pelkkään yksittäiseen kuvaan. Tabletilla se on vielä jokseenkin käyttökelpoinen, mutta puhelimeen siirryttäessä se on niin sanotusti tiellä. Kun yrittää päästä sivustolla alaspäin liu'uttamalla sormea vahingossa sliderin päällä, se vaihtaa ainoastaan slider-kuvan seuraavaan eikä liiku itse sivulla. Se ei siis ole kovinkaan käytännöllinen ja saa todennäköisesti vain käyttäjän hermostumaan, mikäli kyseiseen tilanteeseen ajautuu useamman kerran.

Kokonaisuudessaan onnistuimme kuitenkin toteuttamaan mobiilisivuista varsin käyttökelpoiset. Sieltä löytää tarvittavat tiedot, ulkoasu on yksinkertainen, ylimääräiset suuret kuvat on pyritty poistamaan ja nappulat toteutettiin niin, että ne ovat helppokäyttöiset sormilla käytettäessä.

6 HTML5 JA INTERNET EXPLORER

Vaikka tilastot ovat vain karkea suunta, on mielenkiintoista huomata, että gs.statcounter.com:in mukaan 12 kuukaudessa heinäkuusta 2010 heinäkuuhun 2011 globaali mobiiliselaimen käyttö nousi 2,86 prosentista 7,02 prosenttiin. Samat tilastot osoittavat, että Internet Explorer 6:n käyttö laski 8,79 prosentista 3,42 prosenttiin, ja IE7:n 5,45 prosenttiin. Jos asiakas haluaa sivuston, joka toimii IE6:lla ja IE7:llä, olisi parempi vakuuttaa suuntaamaan tavoitteet muualle. Yhä useammin ihmiset käyttävät sivustoja matkapuhelimellaan, eivätkä pöytäkoneen tai kannettavan tietokoneen Internet Explorer 6 tai 7 -selaimilla. (Frain 2012, 8.) Saman sivuston mukaan globaali mobiiliselaimen käyttö on noussut tasaisesti ja tilastot näyttävät tällä hetkellä käytön olevan 13,9 prosenttia. Samoin IE7:n käyttö on laskenut jo hyvin vähäiseksi: 0,59 prosenttiin.

Entistä mielenkiintoisempaa on se, että joulukuussa 2011 yksi moderneista selaimista, Google Chrome (versiot 15 ja 16), hallitsi 25,7 prosenttia maailmanlaajuisesta selainten käytöstä. Lähes sama määrä, 27,9 %, on yhteensä Internet Explorerin versioissa 6, 7 ja 8. Kun huomioi muiden modernien selaimien osuudet, kuten Safarin (4,3 % ilman iOS-versiota) sekä kaikkien Firefoxin versioiden (21,01 %) sekä mobiiliselaimien, niin ei ole vaikea ymmärtää miksi kehitetään ja parannetaan käyttökokemuksia nykyaikaisissa selaimissa sen sijaan, että paikattaisiin reikiä vanhoissa selaimissa. Lopputulos on kuitenkin se, että vanhojen IE-versioiden käyttö on vähenemässä samalla, kun modernien selaimien käyttö, sekä työpöydällä että mobiilissa, on kasvamassa. (Frain 2012, 276.)

Kaikki nykypäivän selaimet eivät osaa tulkita sisältöä uusista elementeistä. Jotkut selaimet eivät tunnista niitä ja jotkut kohtelevat niitä kuten tekstielementtejä. Tällainen oletustulkitseminen voi aiheuttaa vakavia ongelmia tyyleissä. (Schmitt & Simpson 2011, 8.) HTML5:n toteuttaminen on hankalaa, kun Internet Explorer (IE) ei tulkitse tyylimäärittelyä eli CSS:ää, jota on lisätty uusiin elementteihin (Schmitt & Simpson 2011, 35.) Onneksi tämän korjaaminen ei vaadi paljoa. Selain yksinkertaisesti tarvitsee käskyn kohdella elementtejä kuten lohkotason elementtejä:

```
header, footer, nav, article, aside, section {
          display: block;
}
```

Tuolla yksittäisellä CSS-ilmoituksella, saa melko onnistuneesti tyyliteltyä rakenteellisia elementtejä. Internet Explorerin aiemmat versiot ennen IE9:ää tarvitsevat JavaScriptiä tunnistaakseen elementit ja antaakseen luvan tyylitellä niitä. Kun luodaan elementit, Internet Explorer lisää CSS:n niihin (Schmitt & Simpson 2011, 8, 35.):

```
<script>
          document.createElement('header');
          document.createElement('footer');
          document.createElement('nav');
</script>
```

Vaikka tekniikka on tehokas, voi olla hankalaa luoda DOM-elementit jokaiselle HTML5-elementille, varsinkin jos monet niistä ovat raskaita. Web-ohjelmoija Remy Sharp on luonut skriptin, joka mahdollistaa kaikki HTML5-elementit. Sen voi liittää HTML:n head-elementtiin Googlen koodisäilöstä. Skriptiä kutsutaan ainoastaan mikäli kyseessä on Internet Explorer 8 tai aiempi (Schmitt & Simpson 2011, 38.):

```
<!—-[if lt IE 9]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

Jos yrittää saada Internet Explorer 6 tai 7 näyttämään sivut samanlaisena kuin ne näkyvät nykyaikaisilla standardin mukaisilla selaimilla kuten Safarilla, Firefoxilla tai Chromella, ymmärtää sen hankaluuden. Työ voi viedä jopa 30 % hankkeeseen varatusta ajasta/budjetista, kun yritetään korjata siihen liittyviä ongelmia ja puutteita. Tuonkin ajan voisi sen sijaan käyttää parannuksiin ja vähentämään koodin määrää, jotta sivut toimivat nykyaikaisilla selaimilla eikä paikkaamaan ja lisäämään pyöreitä kulmia, läpinäkyviä kuvia, oikein kohdistettuja lomake-elementtejä Internet Explorerin kutistumassa olevalle käyttäjäjoukolle. Hyväksymällä sen, että selaimien vanhat versiot näyttävät sivut hieman eri tavalla, pysyy koodi ylläpidettävänä sekä halvempana päivittää myöhemmin. Jos tehdään kaikki selaimet näyttämään sivut samanlaisina, se lisää kuvien määrää huomattavasti. Tämä tekee sivuista hitaammat, kalliimmat tuottaa sekä vaikeammin ylläpidettävät. Niukempi koodi tekee sivuista nopeammat nykyaikaisilla selaimilla. Nopeammat sivut taas asettuvat korkeammalle hakukoneissa kuin hitaammat. Vanhojen selaimien käyttäjämäärät vähenevät koko ajan ja sen sijaan nykyaikaisten selaimien käyttäjämäärät kasvavat, joten tulisi ennemmin keskittyä niihin. Keskittymällä nykyai-

kaisiin selaimiin, voi nauttia responsiivisesti muotoilusta, joka reagoi eri tavalla eri laitteiden näkymäikkunoihin. (Frain 2012, 33-34.)

On olemassa sanonnat ”asteittainen parannus” *(eng. progressive enhancement)* sekä ”sulavaliikkeinen hajoaminen” *(eng. graceful degradation)*. Nämä kaksi käsitettä ovat menetelmiä käsitellä laajaa ja monipuolista selaintukea. Ne aiheuttavat kiivasta keskustelua verkkoyhteisössä. Vaikka käsitteet aluksi voivat tuntua keskenään vaihtokelpoisilta tekniikoilta, todellisuudessa ne ovat pohjimmiltaan vastakkaisia. Sulavaliikkeisellä hajoamisella tarkoitetaan sivuston luomista uusille selaimille varmistaen, että tietyt vanhemmat selaimet tuottavat myös käyttökelpoista kokemusta. Ominaisuudet hajoavat selaimien vanhemmilla versioilla ja siellä on rajakohta, jossa kaikkein vanhimpia selaimia ei tueta ollenkaan. On myös aihetta varoittaa käyttäjiä, kun heidän selaimellaan on ongelma ja ehdottaa korjausta, esimerkiksi päivittämällä selain uudempaan. Asteittainen parannus on käänteinen sulavaliikkeiselle hajoamiselle. Asteittainen parannus alkaa merkintäkielestä, joka noudattaa web-standardeja eli se on käyttökelpoinen kaikissa selaimissa riippumatta teknologioiden, kuten JavaScriptin ja jopa CSS:n, käytöstä. Tätä menetelmää pidetään tällä hetkellä parhaiden käytäntöjen tapana kehittää verkkosivuja. (Frain 2012, 274.)

On lähes varmasti mahdollista saada enemmistö HTML5:n ja CSS3:n ominaisuuksista toimimaan myös vanhemmilla selaimilla, mutta tuloksena käyttäjäkokemus on raskaasti kuormitettu JavaScriptillä sekä myös mahdollisesti vähemmän käyttökelpoinen. Sanomattakin selvää, että on tärkeää harkita suorituskyvyn seurauksia tällaisessa valinnassa. Se että voi, ei välttämättä tarkoita että pitäisi. (Frain 2012, 275.)

6.1 HTML5:n tarvitsemat JavaScriptit

Modernizr (http://modernizr.com) on JavaScript-kirjasto, joka havaitsee HTML5:n ja CCS3:n toiminnot käyttäjän selaimesta ja luo halutun ominaisuuden, mikäli selain ei sitä tue (Schmitt & Simpson 2011, 39). Se antaa mahdollisuuden ajatella ominaisuuksien ehtoja selaimen ehtojen sijaan. (Frain 2012, 281.) Fauk Ates kirjoitti Modernizrin ensimmäisen version ja tällä hetkellä hankkeessa on mukana myös Alex Sexton ja Paul Irish johtavana kehittäjänä. Se on apuvälineenä muutamalla yrityksellä, muun muassa

Twitterillä, Microsoftilla sekä Googlella. Modernizr ei ole pelkästään nykypäivän hieno esimerkki JavaScriptistä, vaan se on työväline, joka on syytä ymmärtää. (Frain 2012, 277.)

JavaScript-kirjasto ladataan koneelle ja sijoitetaan HTML-dokumentin `<head>`-elementtiin script-tagiin esimerkiksi `<script src="js/modernizr.js"></script>`. Tällä hetkellä Modernizr tarkastaa yli kaksikymmentä HTML5- ja CSS3-ominaisuutta ja se sisältää myös aiemmin mainitut tekniikat elementtien luomisesta. Jokainen tarkastettu toiminto tekee omat tarvittavat CSS-luokat sekä JavaScript-ominaisuudet. (Schmitt & Simpson 2011, 40, 43.)

6.2 CSS3:n tarvitsemat JavaScriptit

IE:n vanhojen selainten CSS3-tuen puute tarkoittaa, ettei käytössä ole tukea myöskään mediakyselyille. Ilman niitä ei ole merkittäviä ulkoasumuutoksia eri näkymäikkunoissa Internet Explorerin versioilla 6, 7 ja 8. Yksi hyvä tapa on lisätä vanhoille selaimille vain tuki mediakyselyn min- ja max-leveysarvoille. Tämä onnistuu helpoimmin Scott Jehlin respond.js-tiedostolla, joka on helppokäyttöinen, nopea ja palvelee hyvin tarpeita. Kyseistä tiedostoa on mahdollista käyttää myös ilman Modernizria. Toki on olemassa myös runsaampia tiedostoja mahdollistamaan mediakyselyt lisäten laajemman tuen niille. Koska tiedosto on suunnattu ainoastaan Internet Explorer 8:lle ja sitä vanhemmille versioille, on syytä laittaa skripti IE:lle suunnattuun kommenttiin, jolloin muut selaimet eivät huomioi sitä. Modernizr sisältää YepNope.js-nimisen JavaScript mikrokirjaston, joka kysyy selaimelta ymmärtääkö se mediakyselyitä ja toimii sen mukaisesti. Vanhat IE-selaimet reputtavat testin aina ja näin ollen ladataan asianomainen tiedosto nope-kohdasta. Kyseisen koodin voi sijoittaa omaan JavaScript-tiedostoon ja linkittää tämän samalla tavalla kuin modernizrin (Frain 2012, 282-286.):

```
Modernizr.load({
          test: Modernizr.mq('only all'),
          nope: 'js/respond.min.js'
});
```

7 POHDINTA

Paneuduttuani tarkemmin HTML5:n ja CSS3:n käyttöön mobiilisivuistoilla, ymmärrän ettei kaikkiin asioihin ole oikeaa ratkaisua. On hyviä ja huonoja ratkaisuja sekä monien eri ihmisten mielipiteitä. Vaikka sivuston toteuttaisi täysin jonkun ohjeiden mukaisesti, ei se silti välttämättä miellytä kaikkia. On olemassa monia erilaisia vinkkejä ja tapoja, joilla saada mobiilinettisivuista toimivammat. Sivuista ei saa koskaan täydellisiä vaikka kuinka yrittäisi pysyä ajan tasalla asioista. Jatkuvasti kehitellään uusia ominaisuuksia, jotka helpottavat ja/tai vaikeuttavat sivujen toimivuutta.

HTML5 ja CSS3 tarjoavat uusine ominaisuuksineen erittäin hyviä vaihtoehtoja mobiilisivustojen toteuttamiseen. Kun kuvia on mahdollista korvata CSS3:n ominaisuuksilla, keventää se sivuston kuormitusta, jolloin niistä tulee nopeammat, mikä taas on tärkeää mobiililaitteilla. Koko sivuston responsiivinen ulkoasu tarjoaa parhaan ratkaisun mobiiliversion toteutukseen. Tällöin on ylläpidettävänä vain yksi sivusto ja kaikki osoitteet linkittävät samoille sivuille, mikä ei ole täysin varmaa, jos mobiilisivusto on erillisellä alaverkkotunnuksella. Mobiiliversion toteutustapa tulee aina valita tapauskohtaisesti. Joskus on myös hyvä toteuttaa mobiiliversio erilliselle sivulle, kuten esimerkiksi ravintola.fi:n mobiiliversio m.ravintola.fi. Tällöin mobiilikäyttäjät saavat mahdollisimman nopeat sivut, jotka on suunnattu vain heille.

Äkkiseltään ei tule ajatelleeksi kuinka paljon eroavaisuuksia fyysinen näppäimistö ja kosketusnäyttö luovat sivustoille. Hover-valikkoa, jossa hiiri painamatta avaa listan ei voi käyttää ja nappulat tulisi olla vähintään tietyn kokoisia. Mobiilisivuilla olisi myös ehdottoman tärkeää käyttää kaikkia mahdollisia mobiilikäyttäjää helpottavia työkaluja, kuten lomakkeiden kenttien tyyppimääritystä, jolloin kosketusnäytöllisen laitteen näppäimistö muuttuu kyseiselle kentälle suotuisaksi. Varsinkin tämän ominaisuuden olen itse kokenut erittäin hyödylliseksi. Tuntuu melko turhauttavalta, kun pitäisi kirjoittaa lomakkeen kenttään puhelinnumero ja kosketusnäytöllinen näppäimistö avautuu normaalina kirjain-näppäimistönä. Numero-näppäimistöön pääsy ei tosin vaadi kuin yhden painalluksen, mutta kyllä sitä arvostaa, kun se avautuu automaattisesti. Tällöin asiat hoituvat nopeammin ja sitä mobiilikäyttäjä arvostaa.

Yksi suurimmista ongelmista HTML5:n ja CSS3:n yhteydessä ovat Internet Explorerin vanhat versiot. JavaScript tarjoaa monenlaisia eri kirjastoja, joilla voidaan lisätä tuki myös niille, jolloin uudet ominaisuudet saadaan toimimaan. Kuitenkin nykypäivän vaatimukset huomioon ottaen olisi järkevämpää keskittää huomio uudempiin selaimiin ja jättää IE omaan arvoonsa. Kyseinen selain tulee todennäköisesti joka tapauksessa aiheuttamaan päänvaivaa vielä monia vuosia. IE:n vanhojen versioiden käyttäjiä on kuitenkin päivä päivältä vähemmän, joten ei ole välttämättä huono vaihtoehto tarjota kyseisille käyttäjille yksinkertainen ja pelkistetty versio sivuista tai joissain tapauksissa jopa ystävällinen ehdotus selaimen päivittämisestä uudempaan.

Opinnäytetyön tekemiseen kului monia satoja tunteja ja silti koen, että tietotaitoni mobiilinettisivuista on vasta alkuvaiheessa. Olen saanut pohjan, josta voin jatkaa kohti parempaa suunnittelua. Kun perusasiat ovat hallussa, voi keskittyä yksityiskohtaisemmin pieniin, mutta tärkeisiin ominaisuuksiin. Opinnäytetyötä tehdessäni totesin Firtmanin kirjan käsittelevän erittäin laaja-alaisesti mobiilinettisivujen tuottamista. Siinä on huomioitu monia asioita, joista muissa kirjoissa ei ollut mitään mainintaa. Jatkossa voisi muun muassa miettiä tarkemmin selaimia mobiililaitteissa, mitä ratkaisuja ja ongelmia ne mahdollisesti tarjoavat. Toinen asia, mihin jatkossa voisi kiinnittää enemmän huomiota on HTML5:n tarjoamat lomakkeiden uudet syötekenttien tyypit, mitkä helpottavat omalta osaltaan mobiilikäyttäjää. Niiden merkityksen huomaa kunnolla vasta, kun näkee ne itse toiminnassa.

LÄHTEET

Agarwal, A. 2012. Responsive Web Design – A Dummies Guide. Luettu 10.2.2013. http://www.labnol.org/internet/responsive-web-design-faq/21361

Chuan, S. 2012. HTML5 Mobile Development Cookbook. UK, Birhingham: Packt Publishing.

Firtman, M. 2013. Programming the Mobile Web, 2nd Edition. USA: O'Reilly Media.

Frain, B. 2012. Responsive Web Design with HTML5 and CSS3. UK, Birhingham: Packt Publishing.

Hickson, I. 2013. W3C. The srcset attribute. An HTML extensions for adaptive images. Luettu 19.4.2013 http://www.w3.org/html/wg/drafts/srcset/w3c-srcset

jQuery. 2013. Write less, do more. Luettu 19.2.2013. http://jquery.com

jQuery. 2013. Write less, do more. jQuery 2.0 Released. Luettu 24.4.2013. http://blog.jquery.com/2013/04/18/jquery-2-0-released

jQuery Mobile. 2012. Luettu 19.2.2012. http://jquerymobile.com

Korpela, J. 2011. HTML5. Uudet ominaisuudet. Jyväskylä: Docendo.

Leiniö, T. 2012. Mitä on responsiivinen design. Luettu 6.12.2012. http://www.sofokus.com/blogi/2012/07/mita-on-responsiivinen-design

Schmitt, C. & Simpson, K. 2011. HTML5 Cookbook. USA: O'Reilly Media.

Simpson, J. 2012. Designing for Retina Display. Luettu 18.4.2013. http://www.inserthtml.com/2012/09/designing-retina-devices

StatCounter. 2013. Luettu 6.5.2013. http://gs.statcounter.com

Statistic Brain. 2012. Luettu 22.4.2013. http://www.statisticbrain.com/mobile-device-cell-phone-statistics

Wroblewski, L. 2011. Mobile First. New York: A Book Apart.

W3Schools. 2012. Luettu 6.11.2012. http://www.w3schools.com