VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Saila Koski-Aho

RESPONSIIVISET HTML5-ANIMAATIOT

Liiketalous
2015

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

TIIVISTELMÄ

Tekijä	Saila Koski-Aho
Opinnäytetyön nimi	Responsiiviset HTML5-animaatiot
Vuosi	2015
Kieli	suomi
Sivumäärä	69 + 2 liitettä
Ohjaaja	Päivi Rajala

Sivustoista tehdään nykyisin kaikille näyttökooille mukautuvia eli responsiivisia, mikä asettaa uusia vaatimuksia verkkosivujen sisällölle. Animaatio on yksi tällainen sisältötyyppi, jonka tekemiseen on olemassa valmiita ohjelmia, kuten Adoben Edge Animate ja Flash Professional. Molemmilla ohjelmilla voi toteuttaa HTML5-animaatioita. Tämä tutkimus selvittääkin, voiko alun perin Flash-animaatioiden tekoon tarkoitetun Flash Professionalin CC-versiolla toteuttaa animaatiot myös responsiivisesti kuten Edge Animatella, ja ennen kaikkea, miten responsiiviset animaatiot voidaan toteuttaa näillä ohjelmilla.

Tutkimuksessa responsiivisia HTML5-animaatioita käsiteltiin ensin niiden taustalla vaikuttavien web-teknologioiden (HTML5, CSS3, JavaScript, jQuery) ja yleisesti responsiivisen suunnitteluun liittyvien tekniikoiden osalta. Yksityiskohtaisemmin tarkasteltiin responsiivista grafiikkaa. Lisäksi työssä vertailtiin Edge Animatea ja Flash Professionalia, joilla molemmilla toteutettiin tutkimustehtävänä kaksi animaatiota: infografiikka ja banneri. Niiden avulla selvitettiin responsiivisen animaatioon liittyvät tekniikat käytännössä.

Tutkimus osoitti, ettei Flash Professionalista löydy tarvittavia ominaisuuksia responsiivisten animaatioiden tekoon. Näin ollen Edge Animate toimi ainoana ohjelmana niiden toteuttamistapojen selvittämisessä. Tutkimustehtävät tehtiin eri tavoilla mukautuviksi: infografiikka eli aikajana-animaatio skaalaantui kokonaisuudessaan ja bannerin leveyden osalta. Edge Animate osoittautui siis toimivaksi ja suhteellisen helpoksi ohjelmaksi responsiivisten animaatioiden toteutuksessa.

Avainsanat	animaatio, responsiivinen, HTML5, Edge Animate, Flash Professional CC

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittelyn koulutusohjelma

ABSTRACT

Author	Saila Koski-Aho
Title	Responsive HTML5 Animations
Year	2015
Language	Finnish
Pages	69 + 2 Appendices
Name of Supervisor	Päivi Rajala

Websites are mostly responsive these days: they adapt to every screen size and that sets new requirements for the site content. An animation is one of these content types that can be made with a tool like Adobe Edge Animate or Flash Professional. HTML5 animations can be implemented with both packages. This thesis examined if it is also possible to implement responsive animations with the CC version of Flash Professional, which is originally made for Flash-content production. Edge Animate is namely made for responsive design, so the aim of the work was to study if it is really pre-eminent. And first and foremost, the thesis solved how responsive animations are possible to make with such software.

The work approached responsive HTML5 animations by processing the web technologies at their background (HTML5, CSS3, JavaScript, jQuery) and techniques related to general responsive design. Responsive graphics were observed in more detail. In addition Edge Animate and Flash Professional were compared to each other so that both were used to implement two animations: an infographic and a banner. The techniques related to responsive animations were studied with them in practice.

The examination proved that Flash Professional has not got the needed properties for making responsive animations. That is why Edge Animate worked as the only program for studying ways to implement that kind of animations. The examination tasks were made adaptive in different ways: the infographic, timeline animation, scaled as a whole and banner as regards to its width. Edge Animate was proven to be functional and relatively easy software for implementing responsive animations.

Keywords: animation, responsive, HTML5, Edge Animate, Flash Professional CC

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

LIITELUETTELO

1 JOHDANTO

Nykyisin www-suunnittelussa pyritään yhä enemmän siihen, että sivustot toimisivat ja näyttäisivät hyvältä kaikilla laitteilla. Syykin on selvä: älypuhelimien ja tablettien osuus sivustojen käytössä on huomattava. Tosin sivuista ei juurikaan tehdä enää erillisiä mobiiliversioita, vaan yksi ja sama HTML-sivu luodaan mukautumaan käyttäjän näyttökoon mukaan. Toisin sanoen sivut ovat responsiiviset. Tämä tarkoittaa sitä, että sivuston sisältökin mukautuu ympäristöönsä, esimerkiksi muuttaa kokoaan tai paikkaansa.

Yksi www-sivujen sisältötyyppi on animaatio eli liikkuva kuva, johon liittyy usein myös ääntä ja interaktiivisuutta. Responsiivisen web-suunnittelun kannalta on otettava huomioon niin animaation sisältö kuin sekin, luodaanko animaatiosta hallitseva elementti (esimerkiksi etusivun täyttävä animaatio) vai pienempi mainosbanneri. Responsiivisten animaatioiden toteuttamisesta löytyy vähän ohjeistusta, eikä itselläni ole kokemusta siitä. Täten otin lähtökohdakseni responsiivisten animaatioiden tutkimisessa animoimiseen tarkoitetut ohjelmat.

Yksi animaatioiden luomiseen käytettävä ohjelma on Adoben Edge Animate, jolla voi tuottaa animoitua sisältöä verkkosivuille käyttäen HTML5:ta, JavaScriptia ja CSS3:ta eli niin kutsuttuja moderneja web-teknologioita. Edge Animatea on ryhdytty käyttämään Vaasan ammattikorkeakoulussakin Adoben vanhemman animointiohjelman Flash Professionalin sijaan, joka kuului vielä omaan opintosuunnitelmaani ja josta minulla oli myös aiempaa kokemusta. Edgeä taas en ollut käyttänyt vielä tutkimusaiheeni muodostusvaiheessa ollenkaan, paitsi sittemmin multimedian tuotannon työkaluja käsittelevällä kurssilla.

Flash Professional eroaa Edge Animatesta ensisijaisesti siten, että se keskittyy nimensä mukaisesti Flash-animaatioiden tekoon, mutta CC (Creative Cloud)-versiossa on myös HTML5-tuki mukana ja mahdollisuus tehdä animaatioita eri käyttöympäristöjä ja laitteita varten. Kysymys kuuluukin, voisiko Flash Professional CC-versiolla toteuttaa yhtä lailla responsiivisen animaation kuin Edge Animatella vai onko Edge Animate yksinkertaisesti ylivoimainen Flashiin nähden? Tutkimukseni ensisijainen tavoite on selvittää tekniikkoja ja näitä Adoben

työkaluja koskevan tutkimusaineiston ja käytännön työn avulla, miten responsiiviset animaatiot voidaan toteuttaa niillä. Tutkimustehtävänä toteutan molemmilla ohjelmilla kaksi erilaista animaatiota: toisen bannerina ja toisen infografiikkana. Tutkimukseni on toimintatutkimus ja käytän tutkimusmenetelmänä suoraa havainnointia käytännön toteutuksesta.

Mainostoimistot pitävät responsiivisuutta ja ajantasaisuutta tärkeänä sivujen ominaisuutena sen lisäksi, että sivustot näyttävät tyylikkäiltä. Vaasalainen C2 Advertising-mainostoimisto ei ole tässä asiassa poikkeus ja sen oma sivusto onkin rakennettu tällä periaatteella. Työnäni on selvittää yritykselle eri tekniikat ja huomioitavat asiat responsiivisten animaatioiden suunnittelussa sekä ohjelmien hyödyt asian suhteen. Annan mainostoimistolle linkin responsiiviselle www-sivustolle, jolle kokoan selvitykseni pääkohdat ja asetan toteuttamani animaatiot. Tutkimuksestani on hyötyä yritykselle sen tulevissa web-projekteissa ja aihe on myös ajankohtainen.

2 WEB-TEKNOLOGIAT

Ilmaus ”HTML5” on monitahoinen, koska se tarkoittaa eri yhteyksissä ja eri ihmisten kielenkäytössä eri asioita. Se on joko HTML-kielen uusin versio tai kehitysvaihe, yleisnimitys useille nykyajan web-tekniikoille tai sovellusten toteuttamista web-tekniikoilla. Koska tutkimuksessani käsitellään HTML5-kielellä toteutettavia animaatioita, tarkastelun pääpaino on ensin mainitussa merkityksessä. Tässä osiossa tarkastellaan myös muita web-tekniikoita: JavaScriptiä, jQuerya ja CSS:ää, joita käytetään HTML5:n rinnalla niin animaatioiden taustalla Edge Animatessa kuin web-sivuissa ja sovelluksissa yleensä. (Korpela 2014, 3; Grover 2013, 1.) Flash Professional CC:ssäkin käytetään luonnollisesti kyseisiä tekniikoita HTML5-animaatioiden tekoon (Adobe Systems Software Ireland Ltd. 2015a).

2.1 HTML-kielen historiaa

Klassisen HTML:n ensimmäiset toteutukset tehtiin vuosina 1990–1991. Ensin kielen kehitys tapahtui CERNissä (Euroopan hiukkasfysiikan tutkimuskeskus), ja vuonna 1995 siitä laadittiin ensimmäinen julkinen määrittely: IETF:n HTML 2.0. Määrittelystä jäi pois esimerkiksi section-elementti, mutta se tuli määrittelyihin ja selaimiin myöhemmin HTML5:ssä. (Korpela 2014, 28.)

Vuonna 1997 tehtiin HTML 3.2, joka lähinnä virallisti selaimiin jo tehdyt laajennukset, (HTML 3.0 jäi nimittäin luonnosasteelle). Seuraavana vuonna määritelty HTML 4.0 toimi myös laajennusten virallistajana, mutta siinä oli myös uutta teoreettisella puolella. Uutuudet toteutettiin selaimiin vähitellen. HTML 4.0:stä tehtiin hieman muokattu versio 4.1 vuonna 1999, joka on ollut pitkään ”virallinen HTML”. World Wide Web Consortium eli W3C on taho, joka hoiti HTML:n määrittelyn sen versiosta 3.2 lähtien. W3C on lähinnä webistä kiinnostuneiden yritysten ja laitosten muodostama laajapohjainen organisaatio. (Korpela 2014, 28.)

Yleinen merkkauskieli XML on XHTML:n taustalla. XML:n virallinen määrittely tehtiin vuonna 1998 ja XHTML:n ensimmäinen versio määriteltiin puolestaan vuonna 2000. W3C keskittyi XML-pohjaisuuteen HTML:n kehittämisessä XML:n

levitessä monenlaiseen käyttöön. XHTML 1.0 oli HTML 4.01:n määrittely XML-pohjaisena, ja XHTML 1.1 laajensi sitä jonkin verran, mutta toisaalta lisäsi joitain rajoituksia. Kieli jaettiin version myötä moduuleihin, joista voisi koota uusia versioita ja eri ryhmätkin voisivat määritellä uusia moduuleja. (Korpela 2014, 28–29.)

HTML:ää haluttiin kuitenkin eri tahoilla kehittää vanhalta pohjalta eli laajennusten kautta. XML:stä todettiin, ettei se ollut verkkosivujen kielenä HTML:n korvaaja, vaikka XML:ää muuten oli ruvettu käyttämään monissa uudenlaisissa yhteyksissä. W3C:n järjestämässä kokouksessa vuonna 2004 ehdotettiin HTML:n laajentamista rakentaen selainten HTML 4:n käytännön perustalle. Enemmistö kannatti kuitenkin HTML:n seuraavan version kehittämistä uudelta pohjalta. Tätä linjaa ei hyväksytty kaikkialla: Applen, Mozillan ja Operan työntekijöiden perustama WHATWG-yhteisö (Web Hypertext Application Technology Working Group) pyrki kehittämään HTML:ää laajentaen sitä. Yksi varhainen suunnitelma oli lomakkeiden toiminnallisuuden laajentaminen, Web Forms 2.0. Yleisemmin HTML:n kehitystä hahmotteleva suunnitelma oli puolestaan Web Applications 1.0. Nämä suunnitelmat yhdistettiin myöhemmin ja kokonaisuutta ryhdyttiin kutsumaan nimellä HTML5. (Korpela 2014, 29.)

Vuosien 2004–2006 varhaisvaiheen jälkeen W3C hylkäsi aiemman linjansa ja lähestyi WHATWG:tä. W3C:n johtaja Tim Bernes-Lee totesi vuonna 2006 WHATWG:n työn saavuttaneen menestystä, ja ilmoitti W3C:n työskentelevän yhteistyössä sen kilpailijansa kanssa. HTML-työryhmä perustettiin uudelleen vuonna 2007 W3C:llä ja eri pohjalta toiminut XHTML-työryhmä laukkautettiin (eli kehitystyö XHTML 2.0:sta keskeytyi). 2008 julkaistiin ensimmäinen W3C:n HTML5-luonnos. (Korpela 2014, 29.)

Organisaation ja yhteisön välillä on kuitenkin säilynyt jännite. WHATWG on esimerkiksi luopunut HTML5-nimityksessä ja käyttää nimitystä Living HTML standard eli elävä HTML-standardi. Sen sisältö muuttuukin päivittäin, joskaan ei huomattavasti. W3C puolestaan on määritellyt suositusehdokkaaksi (CR=Candidate Recommendation) HTML5:n vuonna 2012, myöhemmin muokat-

tuna 2013, ja 28.10.2014 siitä tuli virallinen W3C:n suositus. Sen pohja on pääosin WHATWG:n määrittelystä, mutta joitain asioita on jätetty pois, lähinnä koska ne halutaan kuvata eri määrittelyissä. W3C:n tavoitteena on viimeistellä ja virallistaa HTML5 suurin piirtein tällaisenaan. Samalla organisaatio jatkaa HTML5:n laajentamista, toistaiseksi työnimellä HTML 5.1, joka muistuttaa paljon WHATWG:n työtä. Eri näkemysten myötä on kuitenkin syntynyt eroja, joista ei toistaiseksi ole virallista kuvausta. HTML 5.1 on tarkoitus vahvistaa suositukseksi vuoden 2016 viimeisellä neljänneksellä (Korpela 2014, 29–30, 34; W3C 2015a, W3C 2015b).

2.2 HTML5-elementit animointiin

HTML5:n multimedian ja grafiikan esittämiseen ja liittämiseen tarkoitettuja elementtejä ovat ainakin <object>-, <source>-, <video>-, <track>-, <audio>-, <svg>- ja <canvas>-elementit. Tarkastelen ja esittelen tarkemmin svg:n, canvasin, audion ja videon sekä sourcen, koska näillä on keskeinen osa responsiivisessa suunnittelussa ja animaatioissa.

2.2.1 Canvas-kuvapinta

Canvas-elementti on HTML5:n keskeisiä uutuuksia tarjoten ”piirtoalustan” graafisille esityksille, jotka voivat olla vuorovaikutteisia. Elementti itsessään luo vain piirtopinnan, varsinainen piirtäminen tapahtuu JavaScriptillä tarkoitukseen tehtyä liittymää käyttäen. Piirtäminen ei tarkoita tässä käsitteenä ainoastaan kuvioiden piirtämistä, vaan myös esimerkiksi tekstin kirjoittamista ja valmiiden kuvien asettamista alustalle. (Korpela 2014, 396.)

Canvas eli kuvapinta julkaistiin alun perin vuonna 2004 osana Applen Mac OS X WebKitiä. Kaksi vuotta myöhemmin Gecko- ja Opera-selaimet ottivat sen käyttöön, ja nykyään se on virallinen HTML5-rajapinta. (Crowther, Lennon, Ash & Wanish 2014, 167.)

Canvas-elementtiin piirtoa varten siihen on liitettävä piirtokonteksti CanvasRenderingContext2D- tai WebGLRenderingContext-olio. Elementin getContext-metodia käytetään kontekstin luontiin, ja metodin ensimmäinen argumentti ilmoit-

taa kontekstin tyypin, ”2d” tai ”webgl”, jotka viittaavat määrittelyihin HTML Canvas 2D Context ja WebGL Specification. 2D-Kontekstin tarjoamilla metodeilla on mahdollista luoda muotoja, määritellä polkuja, käyttää värejä ja liukuvärejä, tuottaa tekstiä ja paljon muuta. Kontekstin määrittävä Canvas API myös tarjoaa kehittäjille tavan viedä kuvapinnan nykyisen sisällön PNG tai JPG-muotoisena kuvana käyttäen URLia tai Blob-objekteja. (Korpela 2014, 398; Crowther ym. 2014, 21.)

HTML5:n yhteydessä määritelty erityisiä keinoja kuvapinnan sisällön animointiin, mutta yleisiä JavaScriptin keinoja voidaan käyttää. Esimerkiksi jos kuvion piirto tapahtuu vaiheittain, syntyy liikkeen vaikutelma. Kuitenkin tietokoneet ovat nykyisin tavallisesti niin nopeita, ettei ihmissilmä ehdi huomata liikettä eli muutosta. Siksi tarvitaan erilaisia hidastuksia eli haluttu vauhti voidaan asettaa ajastusmetodeita apuna käyttäen. (Korpela 2014, 769.)

2.2.2 SVG-elementti

W3C:n suosituksen Scalable Vector Graphics (SVG) Tiny 1.2 Specification mukaista SVG-elementtiä sekä sen sisällä sallittuja elementtejä voi myös käyttää HTML5:ssä. Se on fraasisisältöä HTML:n kannalta. Tällaista käyttöä koskevat erityissäännöt ovat seuraavat:

- ”Jos käytetään SVG:n foreignObject-elementtiä ja sen sisällä HTML-elementtejä, niiden tulee olla juoksevaa sisältöä. Tämä merkitsee käytännössä varsin pientä rajoitusta.”
- ”Jos HTML:ssä käytetään SVG-nimiavaruuden title-elementtiä, sen sisällön tulee olla fraasisisältöä.” (Korpela 2014, 404.)

2.2.3 Audio ja video

Suurin osa internetin kaistasta viime vuosina on täyttynyt multimediasisällöllä: videoilla ja audiolla. Nykyäänkin vielä suuri osa nettivideoista käyttää Flash-videoformaattia (FLV), Adobe Flash-lisäosaa videokoodekkin eri malleille. Jos käyttäjillä on Flash-liitännäinen asennettuna, he voivat toistaa videoita. Monet kehittäjät ovat herättäneet kysymyksiä Flashin tietoturvasta ja suorituskyvystä vide-

oiden välitysalustana ja etsivät vaihtoehtoisia ratkaisuja. Lisäksi Flash-tuen puuttuminen mobiililaitteille merkitsi pitkään sitä, että jos halusi multimediasisällön olevan saatavissa kaikilla laitteilla, kuten iPadilla, kyseinen toive oli turha. (Crowther ym. 2014, 20.)

HTML5 tarjoaa ratkaisun tälle ongelmalle <video>- ja <audio>-elementeillä, jotka sallivat tuettujen multimediatiedostojen toistamisen paikallisesti selaimella ilman kolmannen osapuolen liitännäistä. Video- ja audio-elementit tukevat track-elementtiä, jolla voi välittää mukana tulevaa tekstiä, kuten tekstityksiä. Source-elementtiä voi käyttää tarjoamaan eri tiedostomuotoja varmistaen, että vierailijat voivat käyttää sisältöä riippumatta siitä, mitä käyttöjärjestelmää tai selainta he käyttävät. (Crowther ym. 2014, 20–21.)

HTML5 myös määrittelee rajapinnan metodisarjan videon tai audio-tiedoston toiston hallintaa varten. Se sisältää metodeja toistoon, pysäyttämiseen, eteen- ja taaksepäin kelaukseen, volyymin säätöön ja muuhun. (Crowther ym. 2014, 21.)

2.3 JavaScript ja jQuery

HTML:n yhteydessä JavaScript on pääasiassa selainskriptien tekemiseen käytetty kieli. Skriptillä tarkoitetaan ohjelmakoodia, jonka selain suorittaa HTML-sivun näyttämisen yhteydessä. Tällaisen selainohjelmoinnin lisäksi JavaScripiä on mahdollista käyttää palvelinohjelmointiin. (Korpela 2014, 55.)

JavaScriptin rooli HTML-sivulla riippuu sivun luonteesta ja tarkoituksesta. JavaScriptillä voidaan muokata sivun ulkoasua, rakennetta ja sisältöä. Kielellä voidaan myös luoda vuorovaikutteisuutta ja toiminnallisuutta – ja se onkin tarpeen usein animaatioiden toimivuuden takaamisessa. (Korpela 2014, 55.)

JavaScriptin tuki selaimissa on erittäin laaja, ja se onkin muodostunut käytännössä lähes ainoaksi selainohjelmoinnin kieleksi. Kielestä on eri murteita ja yksityiskohdissa eroja, mutta toteutusten erot ovat vähentyneet selvästi JavaScriptin alkuajoista. JavaScriptin ECMA-standardia kutsutaan ECMAScriptiksi. (Korpela 2014, 55.)

Aikaisemmissa HTML:n määrittelyissä JavaScript kattoi siitä vain pienen osan <script>-elementin käytössä ja joidenkin attribuuttien, joita oli mahdollista lisätä HTML-elementteihin tapahtumakäsittelyn toiminnallisuuden aikaansaamiseksi. HTML5:ssä JavaScriptistä on tullut niin sanottu ensimmäisen luokan kansalainen: jokaisella määrittelyn osiolla on yksityiskohtaista tietoa siitä, mitä DOM (dokumenttioliomalli) API-metodeita ja ominaisuuksia on käytettävissä mille tahansa annetulle elementille. (Crowther ym. 2014, 19.)

JavaScriptin tunnetuin kirjasto jQuery on nopea, pieni ja ominaisuuksiltaan rikas. Helppokäyttöisen APInsa ansiosta jQuery suorittaa HTML-dokumentin läpikäynnin ja käsittelyn, tapahtumakäsittelyn sekä animoinnin paljon yksinkertaisemmin. jQuery on monipuolisuuden ja laajennettavuuden yhdistelmällään muuttanut tapaa, jolla miljoonat ihmiset kirjoittavat JavaScriptiä (The jQuery Foundation 2015). jQueryn tarkoitus onkin tehdä JavaScriptin käytöstä verkkosivuilla helpompaa ja se vaikuttaa olevan kaikkein suosituin JavaScriptin kirjastoista. jQuery on tiivistänyt JavaScriptin yleisimpiä monen koodirivin komentoja metodeiksi, joiden kutsumiseen jQueryllä vaaditaan vain rivi koodia. (W3Schools 2015a.)

jQuery on ollut myös paras ratkaisu monen vuoden ajan, kun puhutaan yhteensopivuudesta kaikkiin selaimiin. HTML5 ei korvaa jQueryä tai muita kirjastoja, vaan sen pitäisi tehdä niistä suorituskykyisempiä. HTML5:n standardisoimisponnistelut tekevät näiden kirjastojen tarjoamasta yhteensopivuudesta vähemmän tärkeitä. (Crowther ym. 2014, 315.)

2.4 CSS3-tyyliohje

CSS, Cascading Style Sheets eli tyyliohje koskee HTML-sivun tai XML-dokumentin ulkoasua, ja tämä ohje annetaan selaimelle. Selaimella on HTML:n tapauksessa aina oletustyyliohje, jonka lisäksi vaikuttaa mukana tuleva ohje ja se voi ohittaa oletusohjeessa olevia ohjeita. (Korpela 2014, 71.)

CSS3 on viimeisin CSS:n standardi, tosin ei virallinen, vaan lähinnä yleisnimitys CSS 2.1:n jälkeiselle kehitykselle. CSS3 siis rakentuu CSS 2.1:n pohjalle ja on säilyttänytkin sen vanhan syntaksin eli yleisen muotorakenteen. Muutoksia on teh-

ty vähän, määritelmiä on lähinnä tarkennettu kuin muutettu. Monia uusia ominaisuuksia ja uusia arvoja vanhoille ominaisuuksille on lisätty. Myös uusia rakenteita on lisätty, kuten monipuolisimpien ehtojen lisääminen @media-sääntöihin esimerkiksi laitteen ominaisuuksiin mukautumiseksi. CSS3:n kehittäjänä toimii WC3 eikä sillä ole kilpailijoita, kuten HTML5-kehityksessä. (Korpela 2013, 13–14; W3Schools 2015b.)

CSS3 on jaettu ”moduuleiksi” tai osa-alueiksi. Se sisältää vanhan CSS:n määrittelyn, joka on jaettu kolmeen pienempään osaan, ja myös uusia moduuleja. Tärkeimpiä osa-alueita ovat esimerkiksi selektorit, laatikkomalli, käyttöliittymä, animaatiot ja 2D/3D-muunnokset. (W3Schools 2015b.)

CSS3:n myötä tulleilla parannuksilla on pyritty vähentämään JavaScriptin ja erillisten kuvien käyttöä muunnoksien ja animaatioiden toteutuksessa (Crowther ym. 2014, 18). Haasteena on kuitenkin eri selainten tuki CSS3:n ominaisuuksille. Tarvitseeko esimerkiksi sivujen luonteen tai käyttäjäkunnan kannalta ottaa huomioon Internet Explorer-selaimen vanhat versiot, jotka saattavat tukea CSS3:ta puutteellisesti? Joka tapauksessa tilanne muuttuu jatkuvasti selainten CSS3-tuen suhteen eivätkä kaikki niistä pysy täysin kehityksen mukana, joten mahdollisimman useita eri selaimia on hyvä käyttää jo kokeiluvaiheessa testaamiseen. (Korpela 2013, 16.)

Selaintukiongelmaa on pyritty ratkaisemaan siten, että ominaisuuksien arvot on toteutettu aluksi niin, että käyttö edellyttää selainkohtaisen etuliitteen (vendor prefix, browser prefix) käyttämistä. Firefoxin tapauksessa etuliite on -moz-, Chromen, Safarin ja Androidin -webkit-, IE:n -ms- ja Operan -o- (myös -webkit- uusimmissa versioissa). Etuliitteet alkavat ja loppuvat siis yhdysmerkeillä. Näiden liitteiden käytön tarve on vähentynyt aidon CSS3-tuen lisääntyessä ja parasta onkin käyttää niitä vain ongelmatapauksissa. (Korpela 2013, 19–20.)

3 RESPONSIIVISUUS VERKKOSIVUILLA

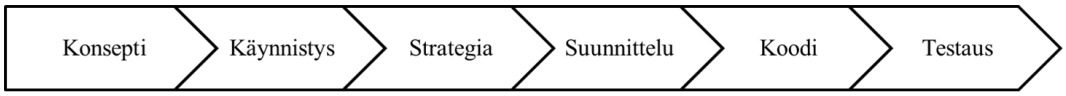
Responsiivinen web pyrkii standardisoimaan kehitysmallit mukautuakseen käyttäjien kaiken kokoisille näytöille. Viimeisen vuosikymmenen ajan verkkosivuja on tavallisesti käytetty pöytäkoneilta ja kannettavilta. Kaistanleveys ja näyttöresoluutiot ovat vakiintuneet ja useimmat käyttäjät ovat sitoutuneet sivujen käyttöön perinteisellä hiirellä ja näppäimistöllä.

Kädessä pidettävien tablettilaitteiden nousulla web-suunnittelu läpikäy ripeää ja kivuliasta kasvun vaihetta. Mobiilisivustot eivät ole uusi asia - mobiililaitteille optimoituja sivuja on ollut yli vuosikymmenen. Ongelma piilee näiden mobiilisivujen rakenteessa. (Carver 2015, 4–5.)

Responsiivinen web, arkkitehtuuri tai suunnittelu – miten sitä ikinä missäkin tilanteessa kutsutaan – on kehitetty siksi, ettei erillisiä ylläpidettäviä mobiilisivustoja tai natiiveja sovelluksia tarvitse luoda (Leiniö 2012). Mukautuva eli responsiivinen sivu käyttää yhtä osoitetta, siirrytäänpä sinne sitten puhelimelta, tabletilta tai pöytäkoneelta käsin. Verkkoliikenteestä aina vain suurempi osa tapahtuu mobiililaitteilla ja tablettien sekä älypuhelimien määrä kasvaa räjähdysmäisesti. Nämä laitteet ovat siis tulleet selkeästi ratkaisevaksi osaksi kaikkea verkkoliikennettä. (Carver 2015, 5.)

Teknisesti responsiivinen web-suunnittelu luottaa mediakyselyihin (media queries), jotka mukauttavat sivun leveyttä. Tämän vuoksi responsiivinen sivu vaatii CSS3:n tuen ja päivityksen HTML5:seen, joka toisaalta on jo standardi HTML. Muuten responsiivisen sivun ideana on pyrkiä olemaan mahdollisimman yhteneväinen kaikilla laitteilla. Käyttämällä yhtä verkko-osoitetta varmistuu, että kaikki sidotut linkit sivulla tarjoavat johdonmukaista sisältöä. Sivu tarjoaakin sisältönsä nopeammin ja tehostaa käyttäjäkokemusta. Suunnittelemalla mobiilisivun ensin, sivujen painotus sijoittuu tehokkaammin. Responsiivinen sivusto on myös tulevaisuudelle avoin, koska jokainen sivu on lopulta optimoitava uudelle teknologialle. Rakentamalla sivusto responsiivisesti taataan se, ettei koko sivustoa tarvitse suunnitella uudelleen. (Carver 2015, 5–6; Leiniö 2012.)

Perinteinen web-kehittäminen käyttää vesiputousmallia. Projekti seuraa ketjua, tyypillisesti kuvion 1 linjojen mukaan.



Kuvio 1. Vesiputousmetodi (Carver 2015, 6).

Vesiputousmallista tulee tehotonta ja kallista, jos tiimin tarvitsee ottaa huomioon muutokset projektissa. Jos epäjohdonmukaisuuksia tai suoritusongelmia ilmenee kesken kaiken, koko projektia täytyy muuttaa ja jopa mahdollisesti rakentaa uudelleen. Responsiivisessa webissä kyse on mukauttamisesta. Responsiivisella suunnittelulla tiimi työskentelee tiiviisti yhdessä rakentaakseen sivun. Sen sijaan että toteutuksia annettaisiin seuraavalle ”sivustokokoonpanolinjalla”, tiimit toistavat ja parantavat toistensa työtä (kuvio 2). (Carver 2015, 6–7.)

Kuvio 2. Ketterä lähestysmistapa (Carver 2015, 7).

Perinteisessä pikselin tarkassa webissä painotus oli ulkoasun uudelleenluomisessa design-osastolla, mutta tässä lähestymistavassa painostus on mukauttamisessa. Standardin mukaisien pikselileveyksien ja fonttikokojen käyttö ei onnistu enää. Tarvitaan jotain vaihtelevampaa. Jotta tällainen ulkoasu voidaan antaa sivustolle ja keskittyä mukauttamiseen, seuraavat asiat on otettava huomioon:

- Ulkoasu, joka mukautuu eri näyttökokoihin ja teknologioihin. Jos uusi tuote tuntemattomalla näytöllä tulee markkinoille, siihen on näin varauduttu
- Nopeampi sivusto – Optimoimalla mobiilille ensin, latausajat ovat etusijalla aina suunnittelun alusta alkaen.
- Yksinkertaisempi selainkohtainen kehitys – Selainyhteensopivuuden ongelmat voi itse asiassa olla helpompi ratkaista CSS:llä. (Carver 2015, 7.)

3.1 Avainominaisuudet

Responsiivinen web ei olisi olemassa sellaisenaan ilman kahta komponenttia: media queryja eli mediakyselyjä ja breakpointteja. Nämä ominaisuudet luovat selainyhteensopivan responsiivisuuden ja antavat verkkosivuille kyvyn mukautua käyttäjän näytölle. (Carver 2015, 7.)

3.1.1 Mediakyselyt

Mediakysely eli media query on yhden tyypin CSS-sääntö, joka rajoittaa tyylin laajuutta sen mukaan, mitä tekijöitä kyselyssä on määritelty. Jokainen mediakysely määrittää mediatyypin ja määrän ilmauksia, jotka selain tarkistaa. Mahdolliset mediatyypit sisältävät tyypit screen (digitaalisille näytöille), print (tulostetuille sivuille) ja all (kaikille mediatyypeille). Ilmaukset ovat yksityiskohtaisempia ja sisältävät ohjeita, kuten max-width tai orientation. (Carver 2015, 7.)

Mediakyselyt tulivat W3C:n vuoden 2001 määrittelyn työluonnoksen CSS3-ehdotuksesta. Ehdotus esitti ratkaisun erilaisten CSS-sääntöjen tarjoamiseen riippuen selaimesta ja laitteen näyttökoosta. Responsiivisten sivustojen mediakyselyitä voidaan käyttää välittämään CSS-sääntöjä, jotka perustuvat useisiin tekijöihin, kuten näyttöresoluutioon, aseman määritykseen ja jopa väri-indexiin. Ilman niitä mobiiliverkkosuunnittelu olisi hankalaa. (Carver 2015, 7–8.)

Mediakysely näyttää tämän kaltaiselta:
@media screen{

p { font-style: italic; }
}

Tämä rivi tyyliohjeessa käskee selainta antamaan paragrafi-tägin fonttityyliksi italicin eli kursiiviisen tyylin, mutta ainoastaan näytöillä. Tyyliä ei anneta tulostetuille sivuille tai kannettaville, jotka tunnistavat itsensä mediatyypillä handheld.

Mediakyselyä voi myös käyttää tarjoamaan asiaankuuluvan CSS-tiedoston <link>-tägin sisään asetetun kriteerin perusteella. Tässä muodossa mediakysely on head-tägin sisällä ja näyttää tältä:

<link rel=”stylesheet” type=”text/css” media=”handheld” href=”italic.css”>

Tämä esimerkki sallii tyyliohjeen latauksen vain selaimille, jotka määrittelevät olevansa handheld. (Carver 2015, 8.)

Suurin ero näiden kahden metodin välillä on se, että erillisen tyyliohjeen tarjoaminen mobiililaitteelle vaatii lisäksi HTTP-pyyntöjä jokaiselle tyyliohjeelle. Selain lataa vain tyyliohjeet, jotka ohittavat mediakyselyn ja tätä voidaan käyttää strategisesti rajoittamaan CSS:n määrää sivulla. (Carver 2015, 8.)

Avain mediakyselyiden käyttöön responsiivisessa suunnittelussa on niiden kyky tarjota näkymän (viewport) leveyteen perustuvan CSS:n. Leveys eli width tarkoittaa selainikkunan leveyttä. Näitä mediakyselyitä kutsutaan ilmaisuiksi ja ne ovat parametrejä, joita selain tarkistaa. (Carver 2015, 8.)

Laitteen välittämä tieto palvelimelle käsittää selaintoimittajan, käytetyn laitteen resoluution ja selainikkunan koon. Responsiivisessa webissä on tärkeää huomata nämä tekijät ja ymmärtää niiden erot. Mediakyselyillä määritellään CSS sen mukaan, onko kyseessä näkymä vai laitteen leveys. (Carver 2015, 8.)

400 pikseliä tai sitä pienempään näkymään perustuvan CSS:n lisäämiseksi käytettäisiin tällaista mediakyselyä:

@media (max-width: 400px) { … }

Jos vaihtoehtoisesti on tarve saada kohteeksi vain laitteet, joiden leveys on 400 pikseliä tai vähemmän, ilmaus olisi:

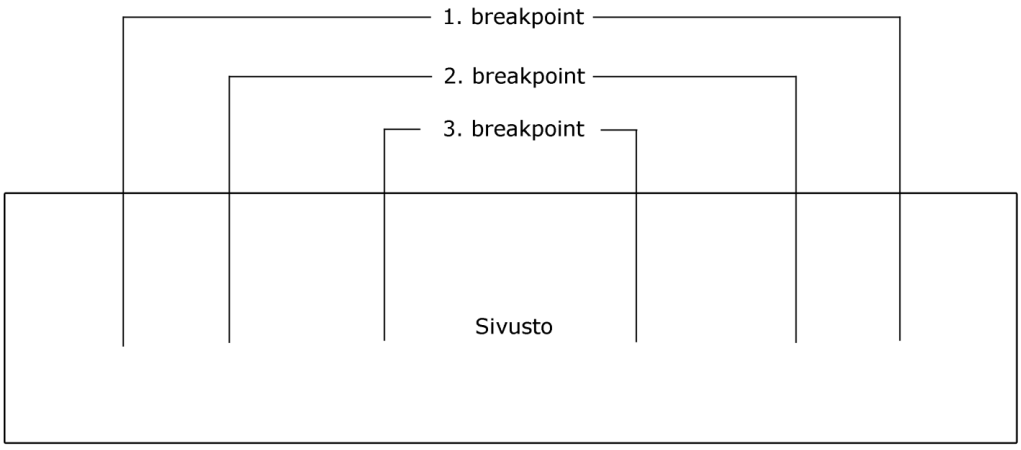
@media (max-device-width: 400px) { ... }

On tärkeää ottaa huomioon näiden kahden ero, koska jossain tapauksissa voidaan haluta pienemmän koon sääntöjä kavennetulle selainikkunalle horisontaalisen vierityspalkin estämiseksi tai paremman käyttökokemuksen takaamiseksi. Pieninäyttöisiin laitteisiin kohdentaminen on tarpeen, jos halutaan pöytäkoneiden käyttäjille antaa kokonäyttöversio sivusta, riippumatta ikkunan koosta. (Carver 2015, 9.)

Min-width- ja max-width-mediakyselyt eroavat siten, että @media (min-width: 400px) { … } kohdistuu selaimeen, jonka leveys on 400 pikseliä tai enemmän. Max-widthillä säännöt vaikuttavat jokaiseen näkymään, joka on asetettua leveyttä pienempi, mutta min-width vaikuttaa kaikkiin määriteltyä leveyttä suurempiin näkymiin. (Carver 2015, 9.)

3.1.2 Responsiivisen sivuston breakpointit

Responsiivisen suunnittelun tavoite on olla suunnittelematta sivua uudelleen jokaiselle mahdolliselle laitteelle ja näkymälle. Tämän välttämiseksi onkin tunnistettava rajat, missä ulkoasua muutetaan sopimaan muuttuvan kontekstin tarpeisiin. Kun sivuston mobiililaitteen leveys muuttuu työpöytänäyttöleveyteen, kyse on siitä, missä kohtaa se muuttuu tai ”rikkoutuu”. Tätä kutsutaan termillä ”rikkoutumispiste” (breakpoint) responsiivisessa suunnittelussa. Breakpointit ovat kohtia, joissa uudet säännöt tarjotaan responsiiviselle sivulle. Kuviossa 3 on esimerkki kolmesta potentiaalisesta breakpointista sivulla. (Carver 2015, 9.)



Kuvio 3. Kolme breakpointia visuaalisena esityksenä (Carver 2015, 10).

Jotkut suunnittelijat käyttävät mieluiten standardisoituja breakpointteja, rakentavat erityisesti suosituimmille mobiili-, tabletti- ja työpöytänäyttövariaatioille. Yksi toinen tapa, jota voi käyttää, on ”mobile first”. Tämän lähestymistavan ideana on, että tuotetaan ensin päätoiminnallisuudet sisältävä mobiilinäkymä, jota laajennetaan vähitellen lisätoiminnallisuuksilla. Kun ulkoasu alkaa laajennettaessa näyttää huonolta tai sivuilla on liikaa tilaa, lisätään breakpoint ja jatketaan mukauttamista asteittain. (Carver 2015, 10; Leiniö 2012.)

3.2 Grafiikka responsiivisessa suunnittelussa

Responsiivisen sivuston suunnittelussa on kyse ulkoasun suunnittelemisesta, joka mukautuu laitteen tai näkymän mukaan. Tämän opinnäytteen aiheen vuoksi animaatiot eli liikkuva grafiikka ovat keskiössä, joten tutkin tässä kappaleessa teoriaa niiden sovittamisesta responsiivisille sivuille.

3.2.1 Grafiikan lisääminen CSS3:lla

CSS3:lla voi lisätä liukuvärejä, pyöristettyjä kulmia, heittovarjoja (niin tekstiin kuin elementteihinkin), värimalleja RGB-värejä ja läpinäkyvyyttä varten sekä peittävyyttä. Lisäksi CSS:llä on muita ominaisuuksia, jotka mahdollistavat kauniin muotoilun: animoinnin tilojen välillä, ja koska CSS kääntyy natiivisti selaimessa, se myös näyttää terävältä ja eloisalta korkean resoluution näytöillä, ku-

ten Retina-näytöillä. Näin välttyy sotkuiselta korkean resoluution näyttöjen havainnointityöltä ja vaihtoehtoisten tiedostojen tarjoamiselta noille selaimille. (Carver 2015, 109–110.)

Suurin osa peruskäytössä olevasta CSS:stä on alkeellista, mutta pari asiaa olisivat esittämisen arvoisia. Ensimmäinen on background-size-ominaisuus, jolla voi asettaa taustakuvan koon. Sen syntaksi on

background-size: [x arvo] [y arvo];

ja arvot voivat olla kokoja pikseleissä tai prosenteissa sekä seuraavat kolme avainsanaa:

- auto – Skaalaa automaattisesti kuvan niin, että sen mittasuhteet säilyvät
- cover – Skaalaa kuvan peittääkseen kokonaan siihen liittyvien objektien koon. Jotkut osat kuvista eivät ehkä näy tausta-alueella.
- contain – Varmistaa, että kuva peittää niin paljon objektialueesta kuin mahdollista, vääristämättä kuvaa. Kaikki kuvasta näkyy, mutta se ei välttämättä peitä koko tausta-aluetta. (Carver 2015, 111–112.)

Toinen ominaisuus on box-shadow, jolla voi lisätä varjon elementtiin. Ominaisuuden syntaksi on

box-shadow: [x etäisyys] [y etäisyys] [sumennus] [väri];

X, y ja sumennus (blur)-arvot voidaan asettaa pikseliarvoilla, ja värin (color) voi asettaa hex-, RGB-, tai RGBA-arvona, joka kontrolloi paremmin varjostusta sen kyvyllä lisätä läpinäkyvyyttä. (Carver 2015, 112.)

3.2.2 Mittasuhteiden säilyttäminen

Responsiivista sivua rakentaessa suurin mahdollisesti kohdattava ongelma on kuvan näyttäminen johdonmukaisessa mittasuhteessa tai kuvasuhde. CSS:llä tällaiset ongelmat voidaan ratkaista.

Skaalautuvassa ja responsiivisessa sivustossa kuvien tai media-elementtien korkeus ja leveys on usein tarpeen pitää suhteessa toisiinsa. 3:4-kuva saattaisi olla tarpeen esittää 200 pikseliä leveänä yhdellä laitteella ja 400 pikseliä leveänä toisella, mutta mittasuhteiden tulisi säilyä. Yksi tapa tämän tekemiseen on määritellä CSS:llä kuvan korkeudeksi ”auto”, mutta miten toimitaan elementtien kanssa, joilla ei ole upotettua korkeusarvoa, kuten elementillä, jolla on taustakuva? (Carver 2015, 112.)

Kuvio 4. Logo sivun yläreunassa.

Apu löytyy CSS:tä. Esimerkiksi voi ottaa logon, joka on sijoitettu sivun yläreunaan (kuvio 4). Mobiililaitteella logon halutaan olevan 100 pikseliä leveä, mutta suuremmilla näytöillä logon halutaan kasvavan niin leveyden kuin korkeuden osalta. Tässä voidaan käyttää seuraavaa tekniikkaa (tässä ensimmäisessä esimerkissä kyseessä ei ole vielä logo):

```
.masterhead{
width: 100%;
padding-bottom: 10px;
}

figure {
width: 100%;
}
```

```
figure img{
width:100%;
max-width:1000px;
height: auto;
display: block;
margin: 0 auto;
}
```

<figure>-tägiä käytetään, koska sen on tarkoitus olla merkityksellinen wrapperi eli kääre kuville. Tässä on enemmän järkeä kuin käyttää diviä, jolla on image-wrap-luokka, koska pitäisi aina pyrkiä käyttämään merkityksien mukaista merkitsemiskieltä. Jokin sorttinen wrapperi on tarpeen skaalaamaan kunnolla sisällä olevan kuvan. <img>-tägi käsittää 100 prosenttia <figure>-wrapperista. (Carver 2015, 25.)

Onneksi kaikki selaimet osaavat skaalata kuvia ja säilyttää niiden kuvasuhteen. Kuva, joka on 1000 x 500 pikseliä, voidaan pienentää 100 x 50 pikseliin mobiilinäytöille ja se näyttää silti hyvältä. Tosiasiassa korkeamman pikselitiheyden laitteille, kuten Applen Retina-näytölle, tämä voi luoda toivotun efektin tuplaamalla kuvan pikselitiheyden. (Carver 2015, 25.)

Antamalla kuvalle max-width-arvoksi 1000 pikseliä ja keskittämällä sen figure-tägin sisälle, varmistuu, ettei kuva näytä rikkoutuneelta, jos figure-tägiä venytetään yli 1000 pikselin. Tämä on turvallinen mitta sen takaamiselle, että sivu säilyttää sen ulkoasun laajakuvamonitoreilla. (Carver 2015, 25.)

Toinen tapa on käyttää heading-tägejä. Jotta saataisiin luotua johdonmukainen suhde leveyden ja korkeuden välille, käytetään h1-tägiä näyttämään logo ja wrapperia pitämään mittasuhteet. Logo, joka näkyy h1:ssä, pitäisi skaalautua mittasuhteiden mukaan emoelementin leveyden kanssa: (Carver 2015, 113.)

```
<div class=”logo-wrapper”>
<h1 id=”logo”>Logo</h1>
</div>
```

Sitten luodaan CSS-säännöt molemmille wrappereille ja logoylätunnisteelle.

```
.logo-wrapper{
position: relative;
```

```
width: 100px;
left:50%;
}

#logo{
position: absolute;
overflow: hidden;
width: 100%;
height: 0px;
padding-bottom: 44%;
font-size: 0px;
background: url (/images/logo.png) no-repeat;
background-size: cover;
}
```

Ensiksi emoelementille asetettiin toivottu leveys. Tässä tapauksessa se asetetaan 100 pikseliin, mutta olisi helppoa käyttää myös piste (em)- tai prosenttiarvoa. Sijainniksi on asetettava relative: tämä on tärkeää, koska lapsielementille on asetettava absoluuttinen sijainti koon säilyttämiseksi. (Carver 2015, 113.)

Prosenttipohjainen korkeus on aina suhteellinen kantaisään täsmällisellä korkeudella, mutta prosenttipohjainen padding eli täyte on aina johdonmukainen (top ja bottom, left ja right) sekä aina suhteellinen emoelementtiin. Täten padding-bottom on suhteellinen emoelementin korkeuteen. Tästä voidaan hyötyä responsiivisilla sivuilla. (Carver 2015, 113.)

Toinen tapaus, jossa CSS-keinot ovat käteviä, on videoiden upottaminen sivuille. Videot vaativat aina asetetun kuvasuhteen. Kuvasuhde on helposti lisättävissä videoon käärimällä video container-diviin, (tavallisen iframe HTML-elementin sijaan) joka määrittelee mahdollisen alueen videolle: (Carver 2015, 114.)

```
<div class=”video-wrapper”>
<iframe src=”http://www.youtube.com/embed/9bZkp/q19f0” frameborder=”0”
allowfullscreen></iframe>
<div>

.video-wrapper{
position: relative;
padding-bottom: 56.25%;
height: 0;
overflow: hidden;
}
```

```
.video-wrapper iframe{
position: absolute;
top: 0;
left: 0;
width: 100%;
height: 100%;
}
```

Käyttämällä edellistä HTML:ää ja CSS:ää video skaalautuu mukavasti näkymien välillä. Videon kehys (iframe) skaalautuu täysin oikealta vasemmalle ja säilyttää kunnollisen kuvasuhteen. (Carver 2015, 114.)

3.2.3 Ikonifonttien käyttö

CSS3:lla voi ladata kustomoituja kirjasimia sivumuotoiluin. Tästä on selvästi hyötyä uniikkien ulkoasujen luomisessa selaimeen, mutta sitä voi hyödyntää kauniiden käyttöliittymien luomisessa käyttämällä ikonifontteja. Ikonifontti on kustomoitu kirjasin, joka pohjautuu skaalautuvaa vektorigrafiikkaan (SVG). Ikonifontilla voi helposti muuttaa väriä, kokoa ja kaikkea muuta, mitä voit tehdä millä tahansa kirjasimella. (Carver 2015, 114.)

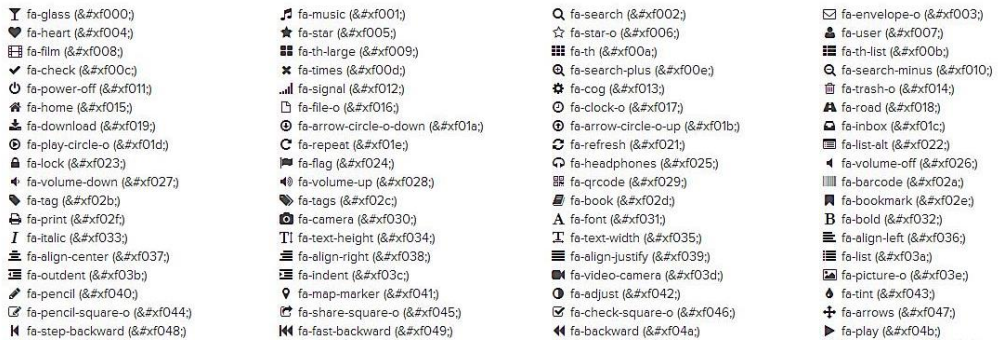
Tavallisesti käyttöliittymäkontrollit verkkosivuille toteutetaan CSS-spriteillä, jotka ovat kokoelmia kuvia yhdistettynä yhdessä kuvassa ja näyttävät ne osat kuvasta tuossa spritessa, kun niitä tarvitaan. Yhtä kuvaa käytetään taustana elementille ja osa, joka näytetään, voidaan asettaa riviin käyttämällä background-position CSS-ominaisuutta. (Carver 2015, 114–115.)

Tämä on ollut perinteisesti tehokas keino, mutta responsiivisen sivun täytyy olla vähän vivahteikkaampi. Korkean resoluution näytöt näyttävät sumeamman kuvan kuin normaalit näytöt, ja pienemmillä näytöillä suunnittelija saattaa haluta käyttää pienempiä tai suurempia painikkeita, jotka vaatisivat uusia spriteja eri breakpointeille. (Carver 2015, 115.)

Paras tapa on käyttää ikonifonttia spriten sijaan. Koska se on kustomoitu kirjasin, ikonit ovat vektoripohjaisia vastakohtana rasteripohjaisille kuville tyypillisessä PNG-spritessa. Jokaista ikonia voi myös muuttaa ikään kuin muuttaisi mitä tahan-

sa toista fonttia. Fontti myös skaalautuu hienosti eri kokovaatimuksia varten. (Carver 2015, 115.)

Esimerkissä käytetään ikonifonttia nimeltä Font Awesome, josta on pieni näyte kuviossa 5:



Kuvio 5. Osa Font Awesomen ikoneista (Hartley 2014).

Tämän fontin käyttämiseksi linkkiviittaus on lisättävä fonttitiedostoon. Tämä on samanlainen prosessi kuin fontin linkittäminen Google Fonts-kirjastosta. Ensimmäisenä CSS-kirjasto on lisättävä sivuille: (Carver 2015, 116.)

```
<link rel="stylesheet" href="http://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.0.3/css/font-awesome.min.css" type="text/css" charset="utf-8">
```

Tämän jälkeen lisätään muutama luokka painikkeille: (Carver 2015, 117.)

```
<span id="infoTrayBtn" class="icon-info-sign btn">
   <i class="fa fa-info"></i>
</span>

<div class="logo-wrapper">
   <h1 id="logo">Logo</h1>
</div>

<span id="navTrayBtn" class="icon-align-justify btn">
  <i class="fa fa-bars"></i>
</span>
```

Sitten käytetään btn-luokkaa muotoilemaan painikkeita samalla tavalla kuin mitä tahansa kirjasinta sivulla:

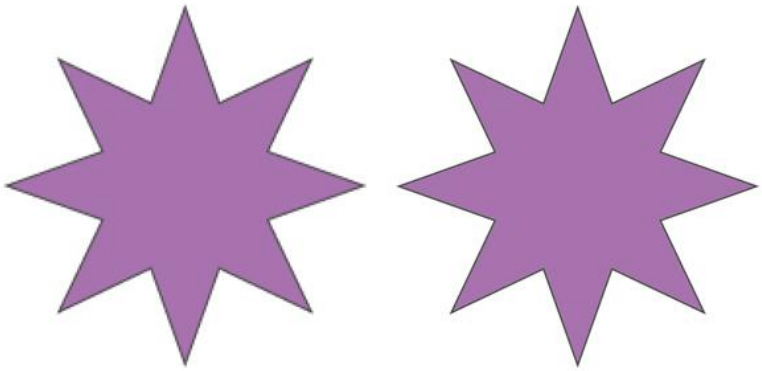
```
.btn{
font-size: 2.2em;
color: #2e3034;
text-align: center;
}
```

Parissa minuutissa saa luotua yksinkertaisen käyttöliittymän, joka skaalautuu ja näyttää hyvältä korkean resoluution näytöillä. Tästä voi pääosin kiittää faktaa, että fontit renderöidään sivulla skaalautuvana vektorigrafiikkana (SVG-kuvana). Font Awesomen tapauksessa nämä vektorigraafikat tulevat fonttitiedostomuodossa toimiakseen kaikissa selaimissa. Vektorit ovat itse samoja, mutta ne toteutetaan muokkaamalla CSS3:n fonttiominaisuuksia upottamisen helpottamiseksi. (Carver 2015, 117.)

3.2.4 SVG-kuvien käyttö

SVG (Scalable Vector Graphics) on XML-pohjainen kuvamuoto, joka on ollut olemassa jo vuodesta 2001. Tiedostomuodosta on aivan äskettäin tullut käyttökelpoinen moderneissa selaimissa, mutta ajoitus oli täydellinen. (Carver 2015, 117)

SVG antaa suunnittelijoille mahdollisuuden luoda kuvia, jotka säilyvät terävinä ja kauniina missä tahansa koossa ja näytöllä. Niitä voidaan zoomata sisään tai ulos ja ne säilyttävät rapeutensa, tai pienentää ja ne säilyttävät yksityiskohtansa. Vertaa 300 prosenttiin zoomattuja tähtiä (kuvio 6), joista ensimmäinen on rasterigrafiikkaa ja toinen vektoripohjainen. (Carver 2015, 117)



Kuvio 6. PNG- ja SVG-grafiikkaa.

SVG-tiedosto voidaan luoda esimerkiksi Illustratorissa. Piirto-ohjelmalla taas grafiikkaan voi tehdä muutoksia nopeasti ja helposti ilman CSS:n päivittämistä. SVG:n voi lisätä sivuille yhdellä tavalla kahdesta: HTML-elementtiobjektina tai CSS-taustakuvana. Objekti näytetään sivuilla itsessään kun taas CSS-taustakuva lisätään HTML-objektitägin tyylinä. Tämä on pieni, mutta tärkeä ero. SVG-kuva saadaan näkymään sivulla yksinkertaisesti linkittämällä lähdetiedosto sivulle. HTML-tiedostossa lisätään <object>-tägi ja muutama attribuutti, ja kuva näkyy selaimessa. (Carver 2015, 118.)

Kun käytetään object-tägiä upotettuun SVG-kuvaan, käytetään data-attribuuttia SVG-tiedoston asettamiseen. Tämä näyttää SVG:n object-tägin sisällä: (Carver 2015, 119.)

<object type="image/svg+xml" width="400" height="400" style="float: right" data="images/star.svg"></object>

Kuva säilyttää läpinäkyvyyden samoin kuin PNG-tiedosto ja on skaalattu 400 x 400 pikseliin, miten object-tägissä on määritelty. Sen sijaan kuvamuodoissa, jotka eivät säilytä läpinäkyvyyttä, kuva renderöidään valkoisella taustalla. Kuvan pienentämiseksi voi muuttaa leveys- ja korkeusarvoja, ja kuva näyttää silti hyvältä.

Kuvan voi skaalata myös CSS:llä:

<object type="image/svg+xml" class="star"
data="images/star.svg"></object>

Huomaa, että object-tägillä on nyt CSS-luokka nimeltä star, joka määrittää saman tyylin kuin style-, width- ja height-ominaisuudet edellisessä esimerkissä: (Carver 2015, 119.)

.star{
float: right;
width: 100px;
height: 100px;
}

Molemmat tavat saavat aikaan saman lopputuloksen. Jos taas kuvaa halutaan suurentaa, samat tavat pätevät ja kuva näyttää edelleen hyvältä. Tiedostokoko pysyy

samana eikä käyttäjän tarvitse ladata isompaa tiedostoa saadakseen isomman kuvan. (Carver 2015, 120.)

SVG-kuvat voidaan liittää CSS:n kautta, mikä voi olla käytännöllisempää ajoittain, vaikkapa kun halutaan kuvan skaalautuvan sivulla olevan objektin kanssa. Jos otetaan esimerkiksi logo, jota käytettiin aikaisemman aiheen kanssa ja korvataan se SVG-tiedostolla. Luodaan SVG-tiedosto, ja seuraavaksi korvataan CSS:ssä oleva PNG-tiedosto sillä: (Carver 2015, 120–121.)

```
#logo{
position: absolute;
overflow: hidden;
width: 100%;
height: 0px;
padding-bottom: 44%;
font-size: 0px;
background-image: url (/images/logo.svg);
background-size: cover;
}
```

Mobiililaitteella voidaan haluta käyttää pientä logoa, mutta suuremmilla täysikokoista logoa. Tämä tehdään puhtaasti background-size-ominaisuutta muuttamalla:

```
#logo{
position: absolute;
overflow: hidden;
width: 131px;
height: 0px;
padding-bottom: 44%;
font-size: 0px;
background-image: url (/images/logo.svg);
background-size: 242px auto;
}
```

Logo skaalautuu niin, että kuvalaatu ei heikkene eli muuttamalla taustakuvan kokoa vaikuttamatta itse objektin kokoon. Vaikka objekti on 131 pikseliä leveä, sen taustakuva on asetettu 242 pikseliä leveäksi jättäen 111 pikselin eron. Tätä ei huomaa, koska se laajenee DOM-objektin ulkopuolelle. (Carver 2015, 121.)

3.2.5 SVG-kuvien rajoitukset

Uuden teknologian kanssa työskentelyllä on aina seurauksia ja SVG ei tee poikkeusta. Vaikka se antaa suunnittelijoille ja kehittäjille paljon korkeamman asteen laatua, sillä on hintansa ja siksi tarvitaan hieman kaukokatseisuutta sen rajoituksien voittamiseksi. (Carver 2015, 121.)

Tiedostokoko on selvin seuraus. Vektorigrafiikka on aina raskaampaa kuin bittikarttagrafiikka, mutta turhan suurten tiedostojen ongelman voi ratkaista palvelinpuolen gzip-pakkaamisella. On kuitenkin yhä tärkeää olla selvillä, kuinka monta tiedostoa on ladattu ja kuinka isoja ne ovat. (Carver 2015, 121–122.)

Toinen monimutkainen asia on se, että SVG toimii ainoastaan hyvin kuvituksissa, missä värirajat pysyvät yksinkertaisina. Muotoihin voi lisätä helposti väriä ja liukuvärejä, mutta monimutkaiset kuvat ovat tiedostokooltaan suurempia. Tämän vuoksi SVG on kaikkein käyttökelpoisin käyttöliittymäelementeille tai sivuikoneille kuten Font Awesomelle, jossa kaikki ikonit renderöidään käyttämällä yhtä väriä. (Carver 2015, 122.)

Täytyy olla myös varmistunut siitä, että palvelin tarjoaa SVG-tiedostoja. Jos käytössä on Apache-palvelin, ei tarvitse kuin vain kirjoittaa muutama rivi palvelimen .htaccess-tiedostoon. Jos taas käyttää jotain muuta web-palvelinta, vastaavan konfigurointitavan löytämisen pitäisi olla helppoa. (Carver 2015, 122.)

Apechea käyttäessä .htaccess-tiedostoon lisätään seuraavat rivit:

AddType image/svg+xml svg
AddType image/svg+xml svgz

Näin palvelin konfiguroidaan lukemaan ja tarjoamaan SVG-tiedostomuotoa. (Carver 2015, 122.)

Lisäksi tulee huomioida selaintuki. Kaikki modernit selaimet tukevat SVG:tä, paitsi Internet Explorerin versio 8 ja sitä aikaisemmat versiot. Toisaalta vanhoja versioita eivät käytä todennäköisesti enää kuin ne, jotka käyttävät WindowsXP:tä,

jonka tukikin on loppunut kokonaan vuonna 2014. Samalla tuki IE 8:llekin lakkasi. (Microsoft 2015a; Microsoft 2015b; Carver 2015, 122.)

Jos kuitenkin haluaa tai on tarve varautua käyttäjiin, jotka saattavat käyttää vanhoja IE:n versioita, koodiin voi aina laittaa PNG-kuvan varmistukseksi:

```
#logo{
background-image: url (../images/logo.png) no-repeat;
background-image: url (../images/logo.svg);
}
```

Jos selain tukee SVG-formaattia, se käyttää viimeisenä olevaa background-image-määrittelyä. Muussa tapauksessa ylikirjoitusta ei tapahdu, vaan selain käyttää png-versiota. (Carver 2015, 122.)

4 ADOBEN ANIMOINTIOHJELMAT

HTML5-animaatiot ovat toteutettavissa Adoben animointityökaluilla, Edge Animatella ja Flash Professionalilla niin, ettei käyttäjän välttämättä tarvitse osata ohjelmointia (Rohde 2013, 1). Tässä kappaleessa esittelen nämä työkalut pääpiirteissään ja lopussa vertaan ohjelmia käsittelemällä niiden historiaa, pääohjelmointikieliä ja käyttöliittymiä. Tutkin siis niitä yleisen käytön kannalta ja mihin vaatimuksiin ohjelmat on tehty vastaamaan tekniikan kehittyessä.

Molempien ohjelmien uusimmat versiot ovat osa Creative Cloudia, Adoben pilvipalvelua, jonka avulla viimeisimmät ominaisuudet ja päivitykset ovat saatavilla ohjelmiin heti. Creative Cloudista on suoraa saatavilla valtaosa Adoben luovan alan ohjelmistoista (esimerkiksi Photoshop, After Effects, InDesign, Illustrator ja Dreamweaver). Creative Cloud-jäsenyyden saa ilmaiseksi 30 päiväksi eli rajattoman käyttöoikeuden kaikkiin sen ohjelmiin, jotka ovat helposti ladattavissa ja asennettavissa koneelle. (Adobe Systems Software Ireland Ltd. 2015b; Rohde 2013, 9–11.)

4.1 Adobe Edge Animate CC

Adoben tuoteperheeseen kuuluva Edge Animate CC on tarkoitettu modernien, mukautuvien ja vuorovaikutteisien HTML5-animaatioiden tekoon. Ohjelma on myös tarkoitettu kaikille animaatioiden tekijöille aloittelijoista ammattilaisiin. Se luokin kaiken koodin käyttäjän puolesta, mutta toisaalta ohjelmointitaitoinen voi käyttää ohjelmointipaneelia Edge Animatella työskennellessään. Edge Animatella animaatiot luodaan siis HTML5:n ja JavaScriptin avulla - pääasiassa JavaScriptin jQuery-kirjastoa käyttäen. (Adobe Systems Inc. 2015, Rohde 2013, 10–11.)

Creative Cloudin lisäksi Edge Animate on osa Adoben Edge Suite-sarjaa, joka käsittää työkaluja ja palveluita web-suunnitteluun ja -kehittämiseen. Muut Edge-ohjelmat ovat Reflow, Code ja Inspect, fonttipalvelut Web Fonts ja Typekit sekä työkalu PhoneGap Build. Edge Animate on myös yhteensopiva useamman eri Adoben julkaisutyökalun (InDesignin, Dreamweaverin, Fireworksin, Musen ja Inspectin) sekä Applen iBookin kanssa. (Rohde 2013, 10–12.)

Edge Animaten web-kehittäjille elintärkeät ominaisuudet sisältävät linkit, loopit, interaktiivisuuden ja responsiivinen web-suunnittelun. Ohjelmalla on mahdollista tehdä diaesityksiä, logoja ja sivustoja, jotka ovat joko animoituja tai eivät, linkeillä tai ilman, kaikki tekstit, kuvat tai tekstin ja kuvien yhdistelmät. Esimerkkeinä ominaisuuksista mainittakoon myös tuki HTML5-videoille, sprite-grafiikan tuominen muista suunnittelutyökaluista ja SVG-kuvien kopioiminen ja liittäminen Illustratorista. (Adobe Systems Inc. 2015; Rohde 2013, 13.)

Adobe Edge Animate CC:llä tehdyt työt toimivat kaikissa moderneissa selaimissa: viimeisimmissä Chromen, Firefoxin, Operan ja Safarin versioissa sekä Internet Explorer 9:ssä ja sitä uudemmissa IE:n versioissa. Ohjelma toimii myös kaikilla tableteilla ja älypuhelimissa, joissa on modernit selaimet. Edge Animate ei tarjoa varmistusvaihtoehtoja käyttäjilleen, jotka eivät käytä uusimpia selainversioita. Tosin ohjelmalla on olemassa ominaisuus, joka sallii Google Chrome Frame-tuen selaimille, jotka eivät tue animaatioita. Näin Edge Animatella voi julkaista sisältöä IE 6, 7 tai 8:n käyttäjäkunnalle käyttämällä Google Chrome Framea. Jos käyttäjällä ei ole jo Google Chrome Frame, asentamiskokemuksen ohjeistaminen käyttäjälle on mahdollista. (Rohde 2013, 13–16.)

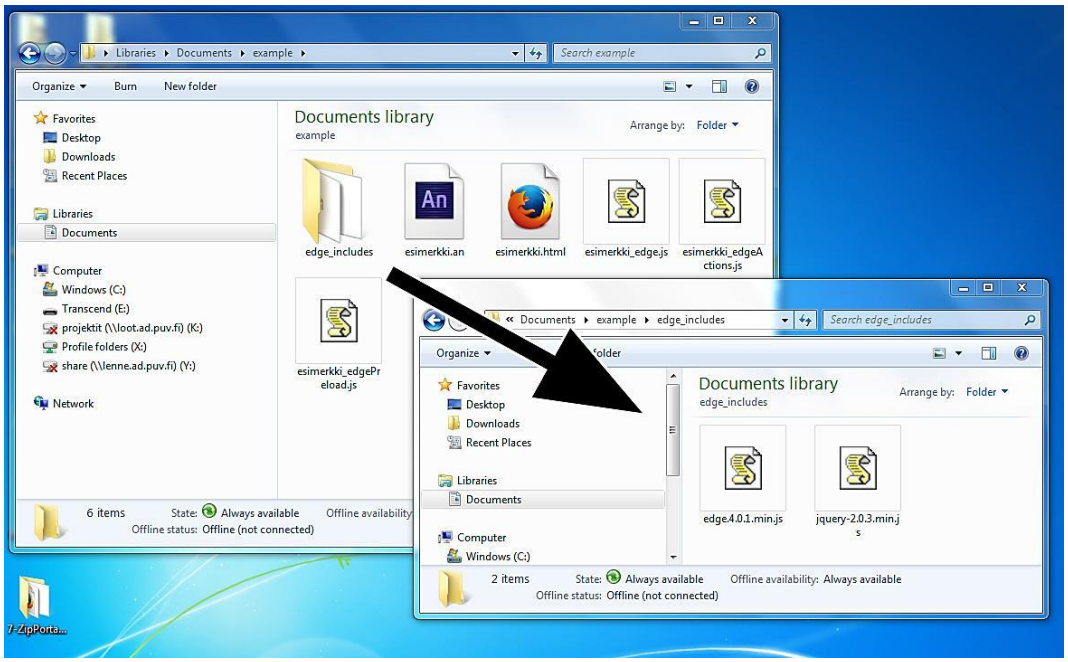
Toinen tapa on tehdä animaatioita down-level Stagella. Se on puutteellisempi versio työskentelyalusta Stagesta, ja sillä luodut työt näkyvät selaimissa, jotka eivät tue HTML5:ta. Asettamalla animaation verkkoon voi havaita, mitä selaimia yleisö käyttää. Jos koodi huomaa, että käytössä on yhteensopimaton selain, se näyttää ilmoituskuvan animaation sijasta eli sivusto ei näytä rikkoutuneelta. Jälleen kerran ohjelmointia ei tarvitse tuntea down-level Stagen käyttöä varten. Selainselvitys on sisäänrakennettu ja valittavissa, kun animaatio ollaan valmiita julkaisemaan. (Rohde 2013, 16–17.)

4.1.1 Projektin tallennus

Kun Edge Animatella tallennetaan projekti, sitä muodostuu tallennuskansioon viisi tiedostoa (kuvio 7).

- .an-tiedosto, joka on Animaten käyttämä projektitiedosto.

- .html-tiedosto, joka kuvaa sivua käyttäen HTML-koodia.
- .js-tiedostot, jotka sisältävät projektiin liittyvää JavaScriptiä. Vaikka projektiin ei olisi tehty mitään, koodissa on kuitenkin määritelty tyhjä Stage ja se suorittaa muut tehtävät, jotka ovat tarpeellisia kaikille Edge Animate-projekteille. (Grover 2013, 2.)



Kuvio 7. Edge Animate-projektin tiedostorakenne.

Tiedostojen lisäksi tiedostosijainnissa on kansio nimeltä edge_includes, josta löytyy JavaScriptin kirjastoja. Yksi on Animaten oma, muut ovat standardeja JavaScript-kirjastoja. Animaten luomalta HTML-sivulta viitataan näihin kirjastoihin, jotka toimivat moottorina Animate-projektin taustalla. Toisin sanoen ne saavat asiat liikkumaan. (Grover 2013, 2.)

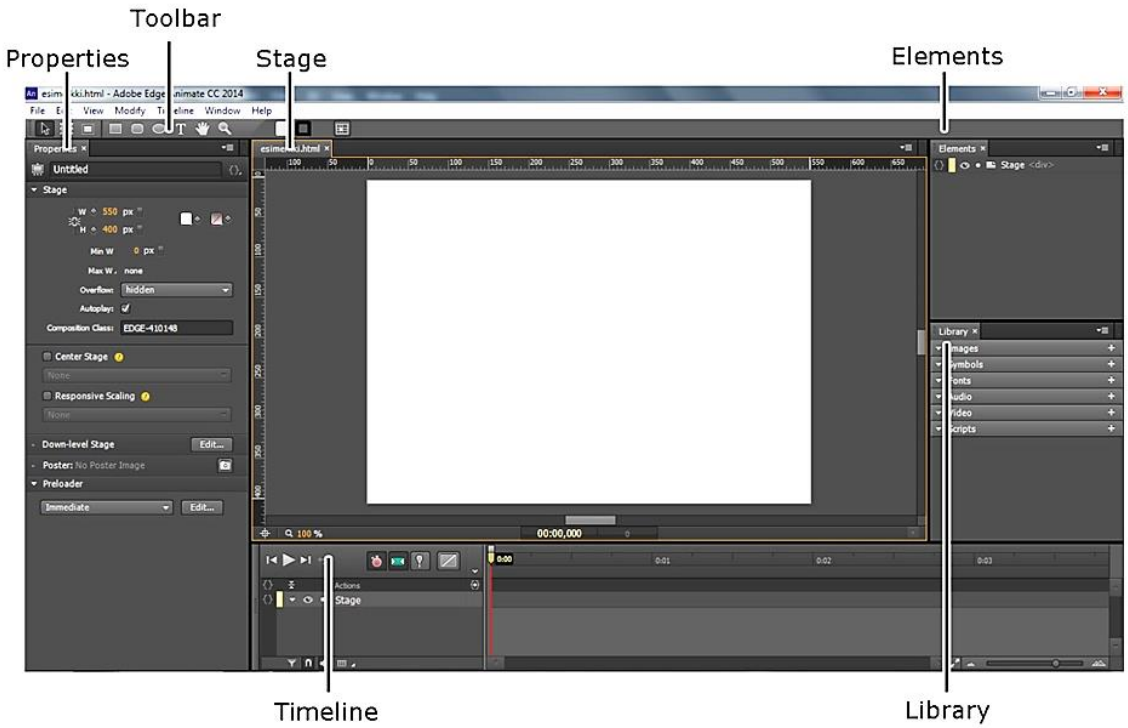
Edge Animate siis tarvitsee luomansa tiedostot rakentaakseen projektin ja näyttääkseen sivun selaimessa. Siksi näitä tiedostoja ei pidä poistaa tai siirtää muualle – ainakaan aiemmin kuin ymmärtää täysin niiden merkitystä ja suhdetta toisiinsa. Tämän (ja sekaannusten välttämiseksi) vuoksi on myös hyvä luoda oma kansionsa jokaiselle projektille. (Grover 2013, 2.)

4.1.2 Työskentelytila

Kun Edge Animaten avaa, ensimmäisenä ruudun täyttää aloitusikkuna, jossa valitaan, luodaanko uusi projekti vai avataanko jokin jo olemassa olevista projekteista. Kun valinta on tehty, näkyviin tulee Edge Animaten käyttöliittymä eli työskentelytila, joka sisältää useita paneeleja ja pienen työkalupalkin (kuvio 8). Jokaisen paneelin nimi lukee yläpalkissa. Elementit- (Elements), ominaisuudet- (Properties) ja aikajana (Timeline)-paneelit sekä työkalupalkki kaikki käsittävät työkalut ja vempaimet, joita käytetään animaation tekoon. Suurin paneeli on ”näyttämö” eli Stage, jossa animaatiot rakennetaan. Sen yläpalkki näyttää käsiteltävän projektin nimen. Alla on kerrottu tarkemmin kaikista tärkeimmistä paneeleista:

- **Stage** on siis projektin näyttämö, jolla näytetään ja animoidaan grafiikkaa ja tekstejä. Kun projekti tallennetaan, Animate tallentaa tekstin ja grafiikan sekä kuvauksen verkkosivuna HTML-koodiksi. Sivun voi avata selaimeen ja se toistuu juuri niin kuin Edge Animaten Stagella. Stage määrittelee projektin rajat ja se on mahdollista piilottaa, ja sijoittaa elementtejä niin, että ne ovat Stagen ulkopuolella.
- **Elementit** ovat objekteja, jotka lisätään Stagelle ja näin ne näkyvät valmiilla HTML-sivulla. Elementit voivat olla grafiikkaa, valokuvia tai tekstiä.
- Elementeillä on **ominaisuuksia**, jotka vaikuttavat niiden sijaintiin ja ulkomuotoon Stagella. Ominaisuuksia hallitaan ominaisuudet eli Properties-paneelista.
- **Aikajana** pitää lukua elementeistä ja niiden ominaisuuksista ajan mittaan. Elementin ominaisuuksien muuttuessa sen sijainti Stagella tai sen ulkomuoto voi muuttua.
- **Kirjasto** (Library) sisältää projektiin tuodut kuvat. Animatessa tehtyjä symboleja pääsee helposti käyttämään kirjaston avulla.
- **Työkalut** (Tools) sijaitsevat päätyötilan yläpuolella. Niitä käytetään luomaan, valitsemaan tai muuttamaan elementtejä Stagella. Palkki on pieni, mutta sillä voi tehdä yllättävän paljon.

- **Lessons-paneeli** eli oppitunti-paneeli oikealla tarjoaa linkkejä Adoben esittämiin ohjeisiin, jotka voivat olla aloitteluvaiheessa avuksi. Se antaa vaiheittaisia, ”step-by-step”-ohjeita, joiden tuotokset näkyvät näytetiedostossa Stagella. Kun paneelilla ei ole enää tarvetta, se kannattaa piilottaa rastista työtilan kasvattamiseksi. (Grover 2013, 3–4.)

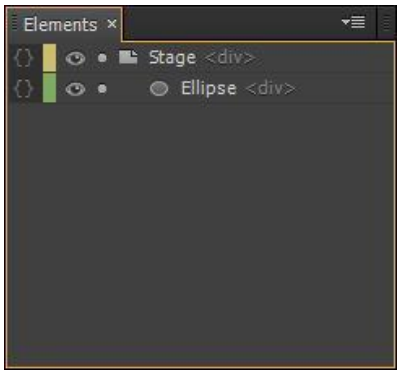


Kuvio 8. Edge Animaten käyttöliittymä.

Edge Animaten käyttöliittymä on samankaltainen kuin muiden Adoben ohjelmien. Silti ei ole tarvetta pitää samaa paneelien järjestystä yllä, vaan paneelien kokoa voi muuttaa tai ne voidaan pienentää itsenäisiksi palkeiksi. Jos taas aikajanan halutaan olevan suurempi, sitä voi vetää yhdestä kulmasta sen ollessa tiukasti ryhmitettynä muiden paneelien kanssa. Kun aikajanan koko muuttuu, ympäröivät paneelit muuttuvat mukautuen uuteen järjestykseen. Vastaavasti Properties-paneelin voi viedä toiseen näyttöön vetämällä sen kielekkeestä. Jos paneelin vapauttaminen on ongelmallista, silloin auttaa ”Undock Panel”-valinta, jonka saa auki paneelin pienestä menu-painikkeesta. (Grover 2013, 5.)

4.1.3 Ohjelmointikielet

Edge Animate luo animaatiot siis HTML:ää, CSS:ää ja JavaScriptiä käyttäen ja seuraavaksi käyn läpi hieman sitä, kuinka ja missä näitä tekniikoita käytetään ohjelmassa.



Kuvio 9. Elements-paneeli.

Ensinnäkin, Elements-paneeli näyttää kaikki tiedostoon tuodut tai piirretyt elementit eli ne, jotka kyseinen HTML-tiedosto sisältää: elementtien ID:t (nimitunnisteet) ja tägit. Stage (jolle grafiikka ja muu luodaan tai tuodaan) on div-elementti (kuvio 9), jonka sisään luodaan elementtejä. Tämän näkee Elements-paneelin lisäksi avaamalla HTML-tiedoston koodieditoriin: kuviossa 10 näkyy, miltä koodi näyttää, kun Stage on tyhjä. (Grover 2013, 164–166.)

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=Edge"/>
    <title>Untitled</title>
<!--Adobe Edge Runtime-->
    <meta http-equiv="X-UA-Compatible" content="IE=Edge">
    <script type="text/javascript" charset="utf-8" src="esimerkki_edgePreload.js"></script>
    <style>
        .edgeLoad-EDGE-410148 { visibility:hidden; }
    </style>
<!--Adobe Edge Runtime End-->

</head>
<body style="margin:0;padding:0;">
    <div id="Stage" class="EDGE-410148">
    </div>
</body>
</html>
```

Kuvio 10. Tyhjän projektitiedoston koodi.

Tiedostossa on siis vähän koodia ja suurimmaksi osaksi se koostuu linkeistä muihin tiedostoihin, jotka saavat työn toimimaan. <title>-tägien välissä on dokumentin otsikko, jonka voi muuttaa niin editorissa kuin Edge Animaten Properties-paneelista. Linkki JavaScript tiedostoon (<script>-tägeissä) puolestaan lataa Edge Animaten tarvitsemat lähteet eli muut .js-tiedostot ja kirjastot. Mitään linkeistä ei ole tarvetta muuttaa HTML-dokumentissa. Samassa kansiossa oleviin JavaScript-tiedostoihin voi kyllä tehdä muutoksia, jos JavaScriptin tai jQueryn kanssa työskentelee edistyneemmin. Edge_includes-kansiossa oleviin tiedostoihin on sen sijaan parempi olla koskematta, koska niiden avulla ohjelmalla tehtyjen luomuksien animoinnit ja efektit toimivat. (Grover 2013, 164–165; Rohde 2013, 102.)

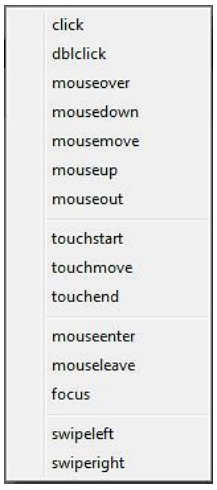
Kuvio 11. Properties-paneeli.

Edge Animaten Properties-paneeli (kuvio 11) toimii ennen kaikkea CSS-tyyliohjekielen asettajana ja animaatioita tehdessä paneeli onkin merkittävässä roolissa. Kaikkia elementtejä koskevista ominaisuuksista mainittakoon ensimmäiseksi luokka (class)-määrityksen lisääminen, mikä tulee tosin tarpeeseen siinä vaiheessa, jos tuntee CSS:ää ja JavaScriptiä ja haluaa tehdä .js-tiedostoiden kautta muutoksia koodiin. (Rohde 2013, 104.)

Tämän opinnäytteen kannalta yksin tärkeimpiä ominaisuuksia sen sijaan ovat sijainti- ja kokomäärittelyt (Position and Size), joiden asetuksia voi hyödyntää suunnitellessa responsiivista sivua. Tässä kohdassa elementille voi asettaa tukipisteen (anchor point), joka määrittelee elementin kohdan, jonka Edge Animate tulkitsee keskikohdaksi eli se vaikuttaa siihen, kuinka tietyt animaatiot näyttävät elementin. Elementin voi myös sijoittaa Stagelle ja muuttaa sen varsinaista kokoa.

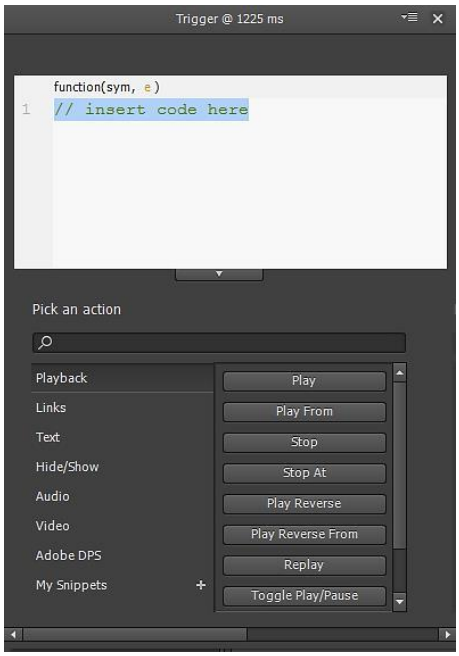
Myös ulkoasun esiasetuksien käyttö on mahdollista. Muita kaikkia elementtejä koskevia CSS-ominaisuuksia ovat muun muassa Display block (näkyvyys), Transform (esimerkiksi kiertäminen) ja Cursor (elementin osoitin). Piirretyille muodoille, kuville ja teksteille on olemassa vain niihin liittyviä ominaisuuksia, kuten värisäätö. (Rohde 2013, 104–106.)

Interaktiivisuus on Edge Animaten tärkeimpiä ominaisuuksia. Ohjelmalla voi lisätä animaatioihin toiminnallisuutta JavaScriptin avulla eli toimintoja (actions), laukaisimia (triggers) ja kylttejä (labels). Elementille lisättävä toiminto (action) tarkoittaa toimintaa, jonka animaation käyttäjä saa aikaan tekemällä jotain, esimerkiksi hiiren vasemman painikkeen napsautuksella. Elementin Actions-laatikosta (kuvio 12) on valittavana useita vastaavia interaktiivisuuden muotoja. Napsauttamalla jotain näistä Actions-koodipaneeli avautuu koodin lisäystä varten. (Rohde 2013, 77-84.)



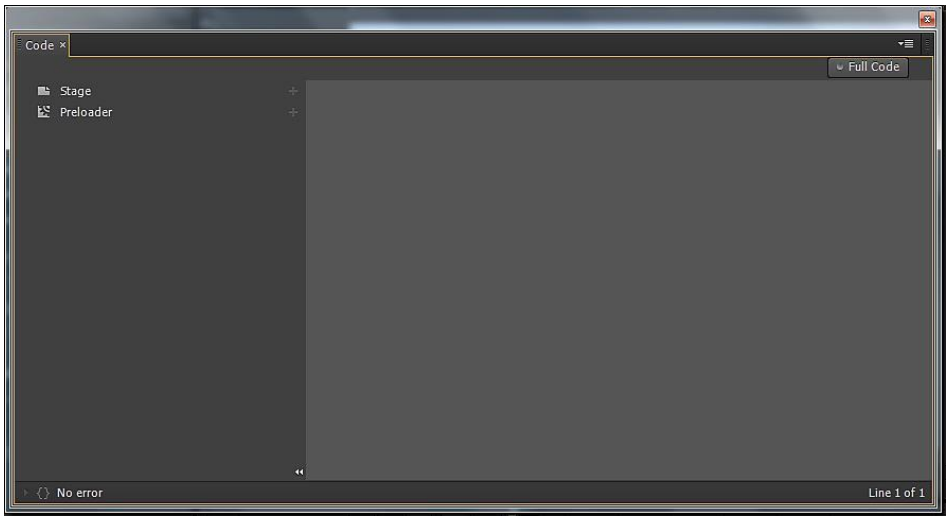
Kuvio 12. Actions-laatikko.

Laukaisimet (triggers) eroavat toiminnoista siten, että ne lisätään Stagelle ja ne tapahtuvat automaattisesti ilman interaktiivisuutta eli silloin, kun animaatio saavuttaa aikajanallaan kohdan, johon laukaisin on lisätty (Rohde 2013, 78). Tyypillinen ”Stage Trigger” on looppaus eli animaatio toistaa itseään kunnes se saavuttaa lopun tai muun määritellyn kohdan. Kuviossa 13 näkyy mahdollisia Stage-toimintoja. (Rohde 2013, 83–84.)



Kuvio 13. Triggers-ikkuna.

Ohjelman Code-paneelista (kuvio 14) voi tarkastella ja muokata kaikkea Actions- ja Triggers-koodia, mitä projektilla on. Kyseessä on ulkoisesti käytettävissä oleva tiedosto nimeltään *projektinimi*_edgeActions. (Grover 2013, 181.)



Kuvio 14. Code-ikkuna.

Kyltit (labels) ovat aikajanalle lisättäviä määrityksiä, joita voi käyttää niin laukaisimien kuin toimintojen kanssa. Jos esimerkiksi animaation halutaan palaavan tiettyyn pisteeseen siinä ajassa, kun se saavuttaa Stage-laukaisimen, voi kylttiä

käyttää tämän pisteen määrittämiseen. Kylttien etuna on, että niiden paikkaa voi muuttaa aikajanalla avaamatta koodi-ikkunaa toisin kuin käytettäessä janan aikaleimoja (timestamps). (Rohde 2013, 81.)

4.2 Adobe Flash Professional CC

Adoben Flash Professional CC:tä käytetään projekteihin, jotka yhdistävät videota, ääntä, kuvaa ja animaatiota. Sisältöä voi luoda Flashissa tai tuoda materiaalia muista Adoben ohjelmista, kuten Photoshopista ja Illustratorista ja suunnitella nopeasti animaatioita ja multimediaa. Käyttämällä Flashin omaa ActionScript 3.0:a on mahdollista yhdistää animaatioihin interaktiivisuutta. Lisäksi verkkosivujen rakentaminen ja itsenäisten työpöytäsovellusten sekä mobiilisovellusten luominen onnistuu Flash Prolla. (Chun 2015, 1.)

Nimensä mukaisesti Flash Professional tarkoitettiin alun perin Flash-animaatioiden tekemiseen, mutta HTML5:n kehitys ja Flash-liitännäisen tietoturvaongelmat ovat saaneet aikaan sen, että ohjelmasta on tehty uudempien tekniikoiden ja laitteiden tarpeisiin vastaava tuote. Tottuneen Flash Pron käyttäjän ei siis välttämättä tarvitse vaihtaa ohjelmakseen Edge Animatea tehdäkseen HTML5-animaatioita. Näin ollen uusin versio Flash Professional CC:stä sisältääkin muun muassa seuraavat ominaisuudet:

- HTML-kuvapinnan (HTML Canvas) ja laajennuksen tuki sekä HTML-optimointi. Täten ohjelmalla voi lisätä myös JavaScript-koodia.
- SVG-kuvien vienti
- Sprite-arkkien vienti
- Erilliset Flash-dokumenttityypit, jotka on tarkoitettu tietyille alustoille julkaisemista varten: Flash Player, HTML5, WebGL, työpöytä- ja mobiilisovellukset iOSille tai Androidille
- Videoiden tuonti suoraan aikajanalle
- Uusi Liike-editori (Motion editor), jolla voi luoda ja hallita liiketweenien ominaisuuksia. Editori oli Flash Prossa aiemmin, mutta se tuotiin takaisin ohjelman versiossa 2014. (Adobe Systems Software Ireland Ltd. 2015c; Chun 2015, 2.)

4.2.1 Projektien tiedostot

Flash Professionalin työtiedostopääte on .fla, joka on avattavissa vain ohjelmassa. Kun luo Flash-tiedoston eli valitsee ActionScript 3.0:n aloitusikkunasta Create new-kohdan alta ja tallentaa sen, tallennussijaintiin ilmestyy .fla-päätteinen tiedosto. Sama tapahtuu, kun tallennetaan luotu HTML Canvas-tiedosto: ohjelmassa tosin Stagella tiedoston nimen perässä lukee suluissa ”Canvas”.

Näiden tiedostotyyppien välinen ero kuitenkin selkenee julkaisuvaiheessa. Kun perinteisen Flash-muotoisen tiedoston ”ajaa” ohjelmassa ensi kertaa eli testaa, miltä animaatio oikeasti toiminnassa näyttää, ohjelma luo julkaistavan swf-päätteisen tiedoston samaan sijaintiin .fla-tiedoston kanssa (kuvio 15). Kyseisen tiedoston voi upottaa HTML-sivulle ja se toistuu Flash Playerin avulla selaimessa. Tiedoston voi myös avata suoraan sijainnista tarkasteluun, kunhan Flash Player on asennettu tietokoneeseen. (Chun 2015, 143.)

Kuvio 15. Flash- ja HTML-projektin tiedostot.

HTML Canvas-tyypin tiedosto vaatii puolestaan julkaisemisen, jotta julkaistavat tiedostot syntyisivät. Julkaisuasetuksista voidaan säätää, millaisia tiedostoja julkaiseminen luo, mutta oletustiedostot ovat html- ja js-tiedostot (kuvio 15), vaikka projektiin ei olisi tehty tai tuotu mitään.

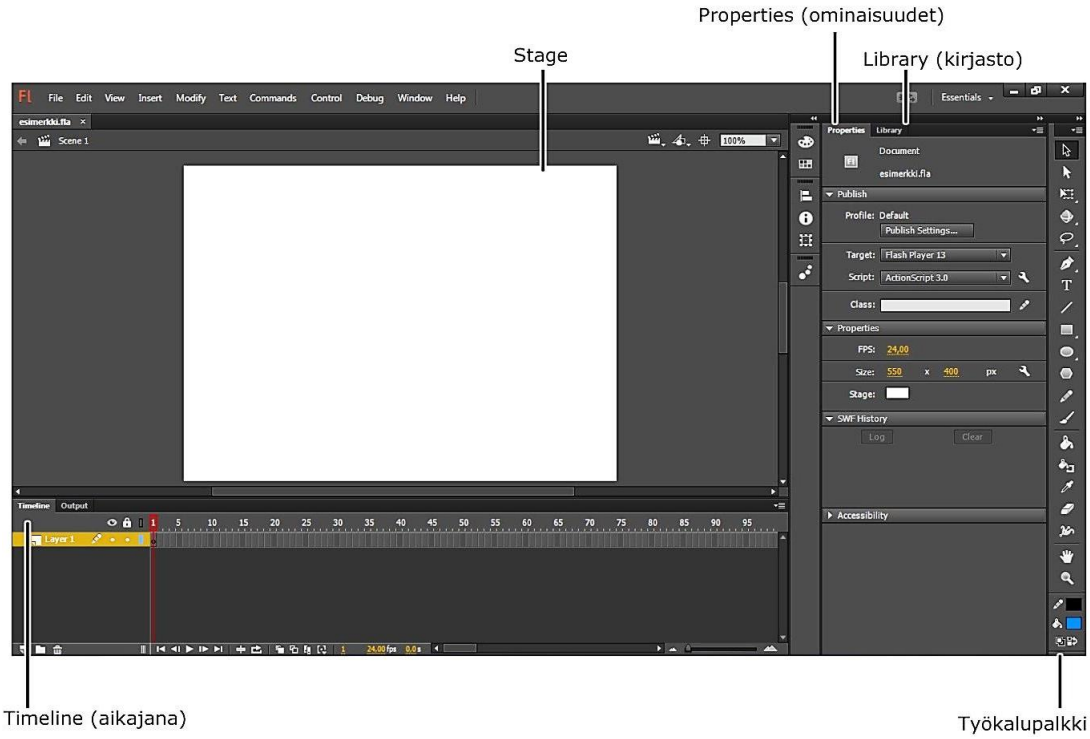
```
<html>
<head>
 <meta charset="UTF-8">
 <title>esimerkki</title>

 <script src="http://code.createjs.com/easeljs-0.7.1.min.js"></script>
 <script src="esimerkki.js"></script>

<script>
 var canvas, stage, exportRoot;

function init() {
     canvas = document.getElementById("canvas");
     exportRoot = new lib.esimerkki();

     stage = new createjs.Stage(canvas);
     stage.addChild(exportRoot);
     stage.update();

     createjs.Ticker.setFPS(lib.properties.fps);
     //createjs.Ticker.addEventListener("tick", stage);
}
</script>
</head>

<body onload="init();" style="background-color:#D4D4D4">
     <canvas id="canvas" width="550" height="400" style="background-color:#FFFFFF"></canvas>
</body>
</html>
```

Kuvio 16. Tyhjän Canvasin määrittely koodissa.

HTML-tiedoston voi avata koodieditoriin ja tyhjäksi jätetty Stage näyttää koodissa kuvion 16 mukaiselta. Title-tägeissä on tiedoston nimi eli sivun otsikko. Tiedostoon liitetyt scriptit ovat jQuery-kirjasto ja julkaistaessa luotu JavaScript-tiedosto. Lisäksi head-tägeissä on funktio, jossa määritellään Canvas, Stage ja exportRoot. Body-tägi taas sisältää Canvasin, sen mitä ulospäin nähdään eli 550 pikseliä leveän ja 400 pikseliä korkean valkoisen kuvapinnan.

4.2.2 Työskentelytila

Flash Professional CC:n aloitusikkunassa valitaan, luodaanko uusi projekti vai avataanko jokin jo olemassa olevista projekteista. Mahdollista on myös valita ohjelmaan tutustuttavia ohjeita, jotka aukeavat selaimeen. Aloitusikkuna on mahdollista piilottaa jatkossa valitsemalla ”Don’t show again”-valintaruudun.



Kuvio 17. Flash Professionalin käyttöliittymä.

Kun aloitusruutu on sulkeutunut, nähtävissä on koko käyttöliittymä eli työskentelytila, joka sisältää useita paneeleja ja työkalupalkin (kuvio 17). Jokaisen paneelin nimi lukee yläpalkissa. Kyseessä on oletusnäkymä, Essentials, joka näyttää Stagen, ominaisuudet- (Properties), kirjasto (Library)- ja aikajana (Timeline)-paneelit sekä työkalupalkin, jotka käsittävät tärkeimmät työkalut animaation tekoon. Suurin alue on ”näyttämö” eli Stage, jossa animaatiot rakennetaan ja jonka yläpalkki näyttää käsiteltävän projektin nimen. Työskentelytilan asetteluun voi vaikuttaa helposti muuttamalla Essentials-näkymä joksikin muuksi (uuden työskentelytilajärjestyksen voi luoda ja tallentaa itse) tai raahaamalla paneeleja eri paikkoihin tai pienentämällä niitä, kuten muillakin Adoben ohjelmilla on mahdollista tehdä.

Kuviosta 17 nähtävistä paneeleista kaikki ovat tuttuja nimensä puolesta Edge Animatea käsittelevästä kappaleesta 4.1 ja mahdolliset eroavaisuudet käsittelenkin kappaleessa 4.3.2.

4.3 Flashin ja Edge Animaten vertailu

Flash Professional ja Edge Animate ovat samankaltaisia erityisesti käyttöliittymiensä puolesta, kuten muutkin Adoben ohjelmat. Toisen hallitseminen helpottaa toisen omaksumista, mutta työkulku animaation tekemisessä on hiukan erilainen. Seuraavaksi käyn läpi Flashin ja Edge Animaten historian, kuinka ne nivoutuvat toisiinsa sekä eroavaisuudet käyttöliittymien ja ohjelmissa käytettävien skriptien välillä.

4.3.1 Ohjelmien historia

Aiemmin yksinkertaiset animaatiot verkossa olivat pääasiallisesti GIF-muotoisia, mutta 2000-luvulla ne korvautuivat parempilaatuisilla Flash-animaatioilla. Muun muassa vektori- ja rasterigrafiikan sekä äänen lisääminen animaatioihin aiheutti lyhyiden animaatioiden nopean lisääntymisen nettisivuilla ja tämä vaikutti jatkuvasti laajentuvaan internet-kulttuuriin. (Ströh 2014a.)

Flashin alkuperäinen nimi oli FutureSplash, jonka kehitti yritys nimeltään FutureWave. Macromedia osti vuonna 1996 FutureWaven ja uudelleennimesi FutureSplashin Flash 1.0:ksi. Vuoteen 2000 mennessä Macromedia lisäsi ohjelmointimahdollisuuksia ActionScriptillä, josta tuli osa JavaScriptin ja ECMAScriptin kaltaista olio-ohjelmointia. Kehittäjien keskuudessa tästä ominaisuudesta tuli hyvin arvostettu. He saivat ActionScriptin ja Flash Player-lisäosan ansiosta selaimen, tuen multimedialle ja mahdollisuuden luoda monimutkaisempia integraatioita. (Ströh 2014a.)

Kokonaan Flashilla rakennetut verkkoanimaatiot, pelit ja sivustot alkoivat näkyä kaikkialla verkossa. Macromedia kehitti vuonna 2003 Flash Video Codecin parempaa videoiden jakoa varten verkossa Flash Playerin avulla. Koodekki tarjosi tavan suoratoistaa videoita Flash Playerilla ja näin Flash-videoista tuli sekä kuluttajien että tuottajien suosima formaatti verkossa. Flash Video Codecia käyttävän YouTuben lanseerauksen ajoitus oli oikea synnyttääkseen videovallankumouksen ja videostandardin sen tukemiseen. (Ströh 2014a.)

Vastauksena vaatimukseen laajemmille ja monimutkaisemmille animaatioille soveltuvasta kielestä, Macromedia julkaisi ActionScript 2.0 samaan aikaan Flash MX2004:n ja Flash Player 7:n kanssa. Tämä versio noudatti osiltaan ECMAScriptin luonnosmäärittelyä. Vuonna 2005 Adobe osti Macromedian ja Flashista tuli Adobe Flash. Vuonna 2006 ActionScript 3.0 esiteltiin perusteellisesti uudelleenjärjesteltyä kielenä, joka noudatti nyt täysin ECMAScriptin luonnosmäärittelyä. (Ströh 2014a.)

iPhone julkaistiin vuonna 2007, ja kuten tiedetään, laite ja sen käyttöliittymä eivät tukeneet Flash-sisältöä. Ensin tämä vaikutti Applen taholta pieneltä erehdykseltä, ja monet kritisoivat iPhonea, joka ei tarjonnut kattavaa verkkoelämystä. Vuonna 2010 iPhonessa eikä tuolloisessa uutuudessa, iPad-tabletissa, voinut vieläkään katsoa Flash-sisältöä. (Ströh 2014a.)

Kilpailijat alkoivat tavoitella uusia tuotteita, jotka tarjoaisivat vaihtoehtoja Applen Flashia tukemattomille laitteille. Tästä alkoi Applen ja Adoben taistelu samoin kuin kiivas keskustelu aiheesta Flash vastaan HTML5 ja avoin vastaan suljettu. Vain päiviä myöhemmin Steve Jobs julkaisi avoimen kirjeen, Ajatuksia Flashista (Thoughts on Flash). Hän selitti, miksi Apple ei tulisi koskaan integroimaan Flashia iOS-laitteille: hän mainitsi luotettavuuden, turvallisuuden, suorituskyvyn ja erityisesti akunkesto-ongelmat. (Ströh 2014a.)

Jobs julisti, mitkä modernit teknologiat hänen mielestään ottaisivat vallan mobiiliaikakaudella: HTML5, CSS ja JavaScript. Tämä tietenkin sai tekniikan maailman kääntämään korvansa ja ajattelemaan, jospa HTML5 ei olisikaan vain muotisana tai jos se olisi osa web-kehittäjien suunnitelmia, se olisi sitä nyt. Lähes kaksi vuotta myöhemmin Adobe ilmoitti, ettei tue Flash Playeriä Android 4.1:llä tai sen jälkeisillä versioilla. Monista keskustelu oli loppuun käyty, ja Adobe oli hävinnyt. (Ströh 2014a.)

Toinen vaihtoehto on, että Adobe otti Steve Jobsin neuvot vastaan tämän Ajatuksia Flashista -kirjeen lopusta, missä hän pyysi Adobea keskittymään enemmän HTML5-työkalujen luontiin tulevaisuutta varten. Elokuussa 2011 Adobe ilmoitti ohjelmasta nimeltä Adobe Edge Preview, ja vuosi myöhemmin siitä julkaistiin

versio 1.0. Tällä hetkellä viimeisin versio CC:ssä on 2015 ja nimenä toimii Adobe Edge Animate. Myös Flash Professionalin viimeisin versio on vuoden 2015 kesäkuulta, ja kuten aiemmin jo mainitsinkin: Flashilla on ollut jo jonkin aikaa mahdollista luoda myös HTML-pohjaisia animaatioita. (Ströh 2014a; Adobe Systems Inc. 2015; Adobe Systems Software Ltd. 2015c.)

4.3.2 Ohjelmien käyttöliittymät

Käyttäjän käyttöliittymien kannalta Flashista ja Edge Animatesta löytyy samankaltaisuuksia ja eroavaisuuksia. Adobe on ajatellut käyttöliittymän luomista: työnkulku on yhteneväinen ja mahdollisimman käyttäjäystävällinen kaikkien sen animaatiografiikkatuotteiden välillä. Vertaan ja käytän toteutuksissa 2014.0-versioita ohjelmista.

Molemmissa ohjelmissa on ominaisuudet eli Properties-paneeli, mutta se sijaitsee Edge Animatessa vasemmalla, kun Flashissa taas oletusnäkymässä oikealla. Paneelin toimintaperiaate on sama: se on riippuvainen asiayhteydestään eli näyttää aina valitun elementin ominaisuudet. Kirjasto (Library) löytyy molemmista, samoin työkalupalkki. Flashilla tosin on perinteisiä grafiikkaohjelman työkaluja enemmän, mutta Animaten tarkoitus on tuottaa tyylit pitkälle CSS:n avulla ennemmin kuin piirtää grafiikkaa ohjelmassa. Flash-tiedoston kuvat ovatkin upotettuja, Edge Animatessa kuvat ovat itse asiassa div-elementtejä HTML:ssä, joilla on CCS:llä asetettu taustakuva. Symboli Edge Animatessa on myös div-elementti, vaikka sillä on paljolti samanlaisia ominaisuuksia kuin Flashissa. (Ströh 2014b.)

Työtiedostot ovat eri muotoisia ohjelmissa, kuten aikaisemmissa kappaleissa käykin ilmi. Edge Animatessa työtiedosto .an-päätteinen. HTML-tiedosto on julkaistu tiedosto, ja samalla käytännössä myös työtiedosto, jota ohjelmassa muokataan. Flashin työtiedosto on puolestaan .fla-muotoinen, ja ainoastaan ohjelmassa avattavissa. Swf-päätteinen tiedosto on puolestaan julkaistu tiedosto, joka toimii Flash Playerin avulla. Luonnollisesti julkaistu tiedosto on HTML-muotoinen Flashissakin, jos työtiedosto on HTML-kuvapintaa tukeva. Flashilla ja Animatella on mahdollista tehdä muitakin kuin HTML/Flash-tiedostoja, mutta tämän opinnäytetyön kannalta niitä ei ole tarpeen käsitellä. (Ströh 2014b.)

Animoinnissa hyvin tärkeä aikajana-paneeli (Timeline) on sijoitettu oletusasettelussa molempien ohjelmien alareunaan. Aikajana näyttää myös projektin tasot, joita on mahdollista näyttää, piilottaa tai lukita. Edgessä tasoissa tosin on kyse elementeistä, ja jokaisella niistä on automaattisesti omana tasonaan Stagella. Flashissa taas tasot täytyy luoda itse ja samalla tasolla voi olla useampia elementtejä, kuten kuvankäsittelyohjelmissa. Muuten aikajanalla käytetään keyframeja, avainkehyksiä, joskin Edgessä on myös toistopään (Playhead) kanssa käytettävä Toggle Pin, jonka avulla keyframeja ei tarvitse lisätä erikseen.

4.3.3 ActionScript ja JavaScript

JavaScript mahdollistaa interaktiivisuuden HTML-animaatioissa ja ActionScript puolestaan Flash-animaatioissa. Molemmat kielet perustuvat ECMAScript-standardiin ja jakavat paljon samanlaisia syntaksiominaisuuksia. ActionScript 3 on itse asiassa JavaScriptiä kehittyneempi, ja JavaScript onkin samankaltaisempi sen vanhempien versioiden kanssa. (Brimelow 2012; Ströh 2014c.)

JavaScript on olio-pohjainen ja ActionScripillä on kyky hyödyntää olio-ohjelmoinnin tekniikoita. JavaScriptin toiminnallisuus on sisäänrakennettu verkkoselaimeen ja onkin pääasiassa käytetty dynaamisen sisällön tuottamiseen (viestilaatikot, kellot ym.), kun taas ActionScriptillä tuotettu sisältö tarvitsee Adobe Flash Player-liitännäisen toimiakseen selaimessa. (Ströh 2014c.)

Merkittävin ero työnkulussa ohjelmien välillä on se, että Edge Animate sallii JavaScriptin sijoittamisen painikkeeseen tai grafiikkaan itseensä. Muuten työnkulku molemmissa on hyvin samanlainen. (Ströh 2014c.)

5 RESPONSIIVISEN HTML5-ANIMAATION TOTEUTUS

HTML5-animaatioiden toteutus onnistuu niin Adoben Edge Animatella kuin Flash Professionalilla melko helposti, ja tuotteita koskevia ilmaisia tutoriaalejakin löytyy hyvin. Sen sijaan responsiivisen animaation toteutus vaatii enemmän miettimistä ja suunnittelua, jotta animaatio näyttäisi johdonmukaiselta kaikilla näyttökooilla. Tässä kappaleessa kerron, miten suunnittelin kaksi animaatiota, infografiikan ja bannerin, ja kuinka toteutin nämä animaatiot responsiivisina. Tein molemmat animaatiot Edge Animatella ja Flash Professionalilla.

5.1 Suunnittelu

Animaatioiden suunnittelussa lähtökohtani oli animaation käyttäytyminen skaalattaessa eli miltä se näyttäisi pienillä ja suurilla näytöillä. Sovelsin Mobile first-tekniikkaa eli aloitin suunnittelun miettimällä sitä, miltä animaatiot näyttäisivät mobiililaitteilla ja tämän jälkeen mietin, miten animaatio muuttuisi, kun näytöllä olisi enemmän tilaa. Koska suunnittelin ja toteutin ensin infografiikan oppien samalla tärkeimmät asiat responsiivisen animaation toteutuksesta, bannerin suunnitteluprosessi sujui selvästi nopeammin ja selkeämmin. Aiheen keksiminen tosin oli bannerin tapauksessa hankalampaa ja siksi aloitin toteutusvaiheen infografiikasta.

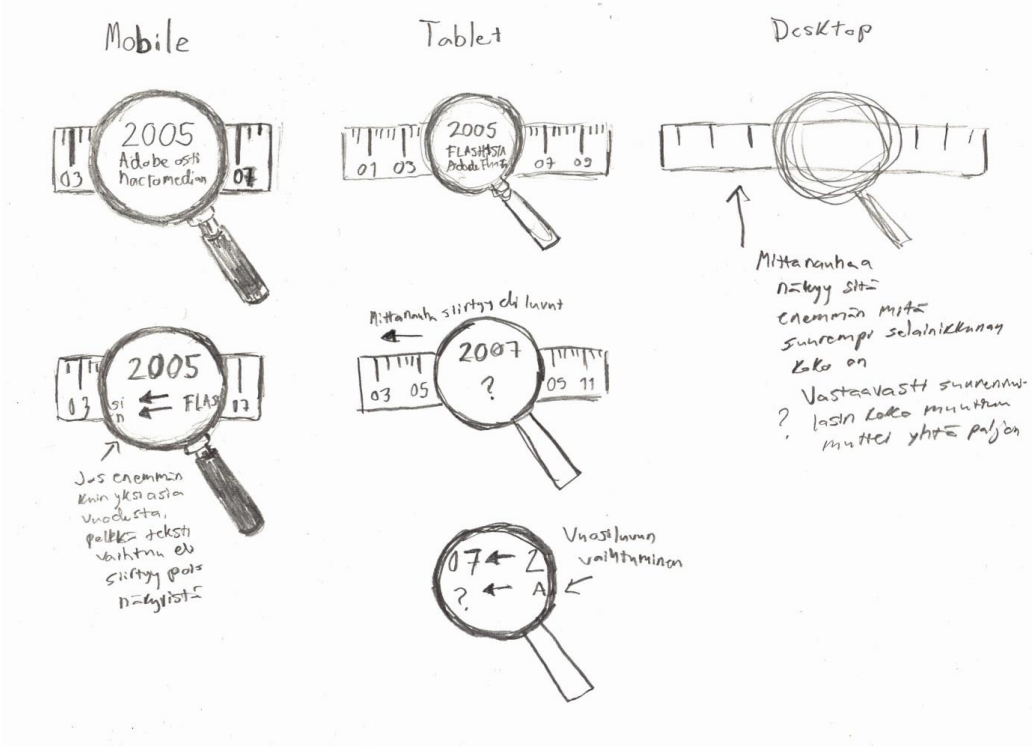
5.1.1 Infografiikka

Infografiikalla tarkoitetaan graafista esitystä, jolla kuvataan jotain ilmiötä tai asiaa. Se käsittää esimerkiksi kuvalliset käyttöohjeet, opasteiden kuvasymbolit, diagrammit, tekniset piirrokset, aikajanat ja kartat. Infografiikka tekee monimutkaisen informaation ymmärtämisestä ja sisäistämisestä helpompaa. Kuva toimiikin katseen kiinnittäjänä ja kiinnostuksen herättäjänä tehden infosta vaivattomammin lähestyttävää. (Hakala & Pyykkö 2014.)

Päätin toteuttaa infografiikan aikajanana, koska mielestäni se olisi selkein tehdä animaatiomuotoisena. Lisäksi infografiikkaa varten tarvitsee usein jotain dataa eli lukuja ja arvoja, ja mieluummin käyttäisin oikeaa tietoa, mitä minulla on. Demon-

stroidakseni aikajana-ideaa otin aiheekseni Flashin ja Edge Animaten yhteisen historian, josta kappaleessa 4.3.1 kirjoitinkin.

Animaatioon suunnittelemani tyyli toimisi hyvin muunkin aiheen kanssa, koska kuvitus ei suoraan liity valittuun aiheeseen, vaan on yleismaailmallinen. Visioni oli koko ajan etenevä, loopilla pyörivä aikalinja, jonka päällä olisi suurennuslasi. Suurennuslasista näkyisi aina vuosi ja silloin tapahtunut asia hetken aikaa, ja tämän jälkeen suurennuslasi liikkuisi seuraavaan vuoteen, jolloin olisi tapahtunut jotain aiheen suhteen. Aikalinja näyttäisi mittanauhalta, jossa vuosiluvut olisivat lyhennettyinä.



Kuvio 18. Luonnossuunnitelma aikajana-animaatiosta.

Kuvio 18 on luonnossuunnitelmani, jossa esitän animaation ulkonäön karkeassa muodossa sekä miltä se voisi näyttää eri näkymissä. Mobiilinäkymässä mittanauhaa eli aikajanaa näkyisi vain välttämättömän verran, mutta työpöytänäkymässä taas selvästi enemmän. Suurennuslasi pysyisi melko samankokoisena, toki isoimmilla näyttökooilla sillä olisi varaa olla suurempi.

Miettiessäni animaatiota käytännössä huomasin, että toteutuspuoli tekstin siirtymisestä näkymättömiin ja näkymättömistä näkyviin voisi tuottaa hankaluuksia. Koska kuitenkin suunnittelin toteuttavani tekstin tekstityökalulla, voisin vähentää ja lisätä kirjaimia siirtymisefektin luomiseksi. Aikajanan liikkuminen oli toinen mutkikkaammin toteutettava asia, joten ajattelin kokeilla toteutusvaiheessa kahta eri tapaa: joko vain siirtää lukuja tai sitten liikuttaa erikseen tehtyä mitta-asteikkoa yli Stagen rajojen.

Grafiikat päätin toteuttaa Illustratorilla, joka on luonnollinen valinta, kun halutaan piirtää vektorigrafiikkaa. Suurennuslasiin otin mallia verkosta etsimästäni kuvasta. Halusin lyhemmän kahvan ja suuremman lasin kuin mallissa, joten käytin mallia lähinnä varjostuksien ja värien käytön apuna.

Tein myös mittanauhan, mutta ilman numeroita. Kokeilin kuitenkin Illustratorissa tehdä näytteen siitä, millaiselta valmis animaatio suurin piirtein voisi näyttää stillkuvana ja kasattuna (kuvio 19). Lisäsin suurennuslasiin kellertävän ja läpinäkyvän osan, jotta se näyttäisi enemmän läpinäkyvältä lasilta. Käytin oletusfonttia tekstissä ja numeroissa, ja ajattelin valita vasta toteutusvaiheessa lopullisen fontin.

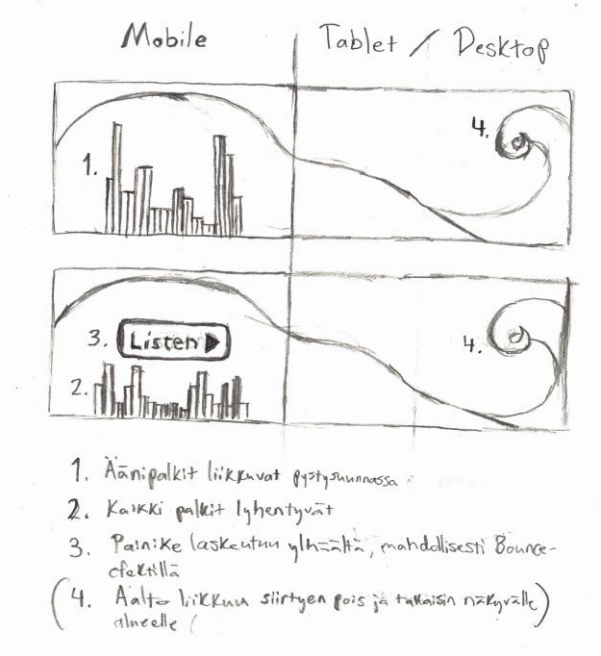
Kuvio 19. Aikajana-animaation grafiikat.

Mainittakoon vielä, että tein grafiikoista SVG-versiot Illustratorilla, koska halusin saada kuvat pysymään terävinä skaalautuvassa animaatiossa missä näyttökoossa tahansa.

5.1.2 Banneri

Bannereilla, viralliselta nimeltään display-mainonnalla tarkoitetaan sivustoilla, mobiililaitteissa ja tableteissa näkyviä, erimuotoisia ilmoituksia. Mahdollisimman hyvän ja laajan yhteensopivuuden takaamiseksi bannerit toteutetaan tavallisesti standardikokoisina. Bannerien sisältö voi olla niin staattista kuin liikkuvaakin kuvaa. Bannereita voi esiintyä joko yrityksen omilla kotisivuilla tai muilla sivuilla (esim. uutissivustot) yrityksen maksamina mainoksina. Mainosbannerin napsautus vie yleensä tuotteesta lisätietoa antavalle sivustolle, kampanjasivulle tai mainostettavan palvelun etusivulle. (Järvilehto 2012, 97.)

Suunnittelin toteuttavani bannerista yksinkertaisemman kuin infografiikasta. En kehittänyt bannerille varsinaista aihetta. Mainosbannerin olisin toteuttanut muuten, mutta mainostettava asia olisi pitänyt olla itse keksimäni. Päätin sitten tehdä musiikkiaiheisen bannerin, jossa olisi taajuuskorjain sekä muuna grafiikkana hiekkakumpu ja aallokko. Kuviosta 20 näkeekin suunnitelmani luonnosteltuna, jossa vasemmalla ovat taajuuskorjain, jonka palkit liikkuisivat ylös alas eli samaan tyylin, kuten oikeat taajuuskorjainpalkit. Samalla myös suurilla näytöillä näkyvässä osassa liikkuisi aalto. Sitten bannerin taajuuskorjaimen palkit mataloituisivat, yhä kuitenkin liikkuen, ja ylhäältä laskeutuisi painike, jota painamalla voisi kuunnella kappaleen.



Kuvio 20. Luonnossuunnitelma bannerista.

Bannerin tausta olisi käytännössä responsiivinen vain näytönleveyden suhteen. Senpä vuoksi kaikki oleellisin näkyisi noin 320 pikseliä leveällä alueella eli mobiililaitteilla. Aalto olisi käytännössä ylimääräinen lisä näkyen kaikilla yli 600 pikseliä leveillä näytöillä. Kyseessä olisi siis banneri, jonka osa sisällöstä katoaisi näkyvistä pienillä näytöillä, joilla vaikutelma olisi täten hieman erilainen kuin suurilla näytöillä.

Tein grafiikat eli aallon, hiekkakummun ja painikkeen tälläkin kertaa Illustratorissa ja tallensin kuvista myös SVG-versiot. Taajuuskorjaimen eli käytännössä palkit päätin työstää toteutusvaiheessa animointiohjelmissa, koska niitä pitäisi venyttää toivotun animaation aikaansaamiseksi.

5.2 Toteutus Edge Animatella

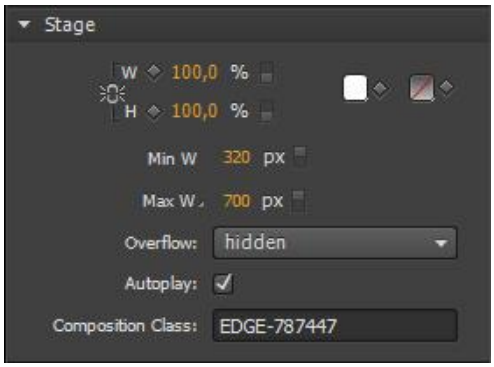
Tässä kappaleessa käyn läpi päävaiheittain, miten toteutin infografiikan ja bannerin Adobe Edge Animatella. En keskity ohjeistamaan ohjelman käyttöä kohta kohdalta, vaan kerron esimerkiksi, miksi olen valinnut tietynlaiset ratkaisut eri elementtien asetuksiksi.

5.2.1 Infografiikan toteutus

Tein infografiikan eli aikajana-animaation ensimmäisenä. Suunnittelua ennen ja sen jälkeen opiskelin Imaginen Web Designer-sivuston ”Create a responsive animation with Adobe Edge Animate”-tutoriaalin avulla ja kirjoista Adobe Edge Animate CC For Dummies ja Adobe Edge Animate: the Missing manual responsiivisen grafiikan ja animaation teosta ohjelmassa. Kirjojen teorian ja ohjeiden avulla opin eniten aiheesta käytännön kokemuksen jälkeen ja samalla harjoittelin tietojen soveltamista infografiikan tekoon.

Aloitin animaation toteuttamisen tuomalla tekemäni grafiikat Animateen. Testattuani erilaisia asetuksia päädyin kuitenkin ratkaisuun liittää mittanauhan ja suurennuslasin yhdeksi SVG-tiedostoksi, koska suurennuslasin tulisi pysyä keskellä mittanauhaa koko ajan ja missä koossa tahansa. Suurennuslasi ei liikkuisi ja suunnitelmani mukaan mitankaan ei välttämättä tarvitsisi liikkua, vaan lukujen.

Ensimmäiseksi Stagen eli niin kutsutun isäntä-divin mittojen piti olla prosenteissa, jotta responsiivisen animaation teko olisi mahdollista (Rohde 2013, 311). Stagen leveydeksi olin asettanut 700 pikseliä ja korkeudeksi 400 pikseliä. Muutin pikselit prosenteiksi sekä korkeuden että leveyden suhteen. Käytännössä tämä tarkoittaa sitä, että Stage täyttää koko selainikkunan missä tahansa koossa, kun se avataan selaimessa. Tässä työssä taustavärin ollessa valkoinen tätä ei huomaa, mutta vaihtamalla värin erottuvammaksi asiasta voi varmistua. Asetin Stagelle kuitenkin vielä maksimi-ja minimileveydet, 700 pikseliä ja 320 pikseliä (kuvio 21), jotta animaatio ei venyisi isoilla näytöillä kohtuuttoman suureksi ja vastaavasti sen minimikoko vastaisi pienintä älypuhelimen näyttöleveyttä. Tätä pienemmillä näytöillä tai selainikkunan leveydellä Stagen leveys leikkaantuu.



Kuvio 21. Stagen leveys- ja korkeusominaisuudet.

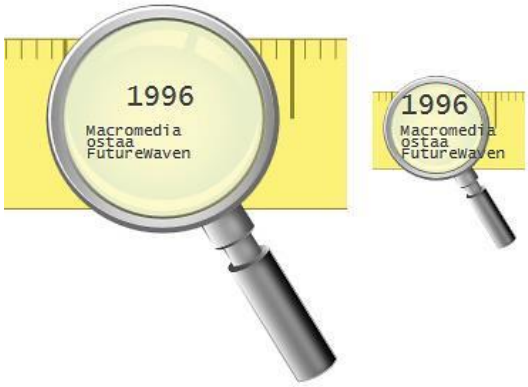
Grafiikan asetin aikajanan Stagelle niin, että mittanauhan reunoilla olevan viivat ovat yhtä kaukana reunasta kummaltakin puolelta (kuvan korkeus 100 % x 96 % ja x arvo 0 % ja y 2 %). Kuva-asetukset olin asettanut Layout Defaults-valikosta valmiiksi sellaisiksi, että kuvat käyttävät img-tägiä ja koko sekä left- ja top-arvot ovat prosentteina. SVG-kuvan ominaisuus onkin se, että se säilyttää kuvan mittasuhteet, vaikka sekä korkeuden että leveyden asettaa prosenteiksi. Muun tyyppisien kuvien tapauksessa mittasuhteiden säilyttämiseksi joko leveyden tai korkeuden on oltava ”auto”, jos toinen arvo on prosenteissa (Rohde 2013, 315).

Näillä yksinkertaisilla Stagen ja SVG-kuvan arvoilla sain kuvan skaalautumaan järkevästi, muttei suunnitelman mukaisesti. Suurennuslasin kokohan luonnollisesti pieneni aivan yhtä lailla kuin mittanauha eli mittasuhteet säilyivät, koska kyse on yhdestä kuvasta (kuvio 22). Tarkoitushan oli, että suurennuslasi jäisi suuremmaksi. Erillisinä kuvina joko asemointi tai kuvien koot olivat pielessä, vaikka tosiaan testasin eri Layout Preset-vaihtoehtoja. Kuvia ei myöskään voi asettaa sisäkkäin, kuten ohjelmassa piirrettyjä muotoja ja tekstejä voi.

Kuvio 22. Aikajana 320 pikseliä leveällä näytöllä.

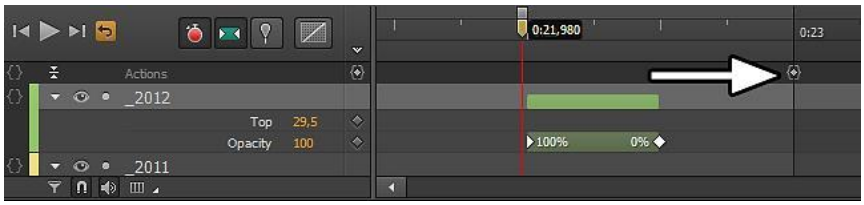
Seuraava vaihe työssä oli tehdä tekstit ja luvut sekä animoida ne liikkumaan ja vaihtumaan. Responsiivisuuden kannalta tämä olikin osittain ongelmallinen asia, sillä tekstilaatikon sisältöä ei saa skaalautumaan eli fonttikokoa muuttumaan ohjelmassa. Ainoastaan tekstilaatikon saa skaalautumaan, josta ei ollut tässä tapauksessa hyötyä (Grover 2013, 233). Tieto tästä oli luettavissa Missing manual-oppaasta, mutta myös nopeasti itse havaittavissa. Näin ollen päätin tehdä vain aikajanalla liikkuvat vuosiluvut tekstinä ohjelmassa. Niiden koon säilyminen ei nimittäin haitannut, koska ne eivät menisi minimikoossa mitan yli ja luvut erottaisi hyvin missä tahansa koossa.

Suurennuslasissa näkyvien kokonaisten vuosilukujen ja tekstin suhteen oli tehtävä toisin kuin aikajanan lukujen, koska kokonsa säilyttävät tekstit menisivät suurennuslasin yli pienillä näyttökooilla. Tekstin pienentäminenkään ei olisi ratkaisu, koska suurilla näytöillä se ei puolestaan toimisi (kuvio 23). Päätin tehdä tekstit siis SVG-kuviksi, jotka skaalautuisivat. Tämä toimi, mutta animaation ollessa minimikoossa teksti näytti kyllä pieneltä, jopa epäselvältä. Tekstiä oli tosin varaa kasvattaa – kuvia piti vain muokata Illustratorissa.



Kuvio 23. Skaalautumattoman tekstin ongelma.

Animaatio-osuudessa siirsin aikajanan lukuja vasemmalle aina siinä vaiheessa kun oli aika seuraavan asian tarkasteluun. Luvun mennessä suurennuslasin alta eli käytännössä päältä, laitoin luvun siksi ajaksi näkymättömäksi ja luku ilmestyi näkyviin lasin jälkeen eli tässä kohtaa siirtymää lisäsin Display-keyframen On-asennossa. Suurennuslasissa vaihtuvat tekstit animoin muuttamalla niiden peittävyyttä. Näin ollen teksti tuli vaiheittain näkyviin nollaprosenttisesta peittävyydestä sataprosenttiseen ja haalistui näkyvistä päinvastoin, joka on nähtävissä kuviosta 24.



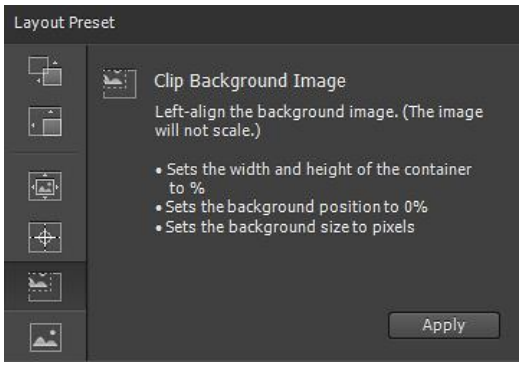
Kuvio 24. Aikajana-paneeli.

Jokaisen siirtymän välissä oli kolme sekuntia, jotta katselija ehtisi lukea aina nähtävissä olevan tekstin, ja siirtymät laitoin kestämään sekunnin verran. Laitoin animaation toistumaan automaattisesti eli palaamaan lopusta aina alkuun, ja käytin tähän laukaistinta eli triggeriä, josta Animaten esittelykappaleessa olikin mainintaa. Laukaisin tuli siis Stagelle asetettuna aikajana-paneelissa animaation loppuun (kuvio 24) ja kirjoitin Actions-paneeliin funktion sym.play();. Valmis animaatio on tarkasteltavissa kuvina kahdessa eri näyttökoossa liitteessä 1.

5.2.2 Bannerin toteutus

Bannerin toteuttaminen onnistui suunnitelman mukaisesti. Ensin muutin Stagen koon tekemällä siitä 200 pikseliä korkean ja 600 pikseliä leveän. Tällä kertaa muutin leveyden prosenteiksi ja jätin korkeuden pikseleiksi, jotta bannerin korkeus ei muuttuisi näyttökoon mukaan. Asetin bannerille minimi- ja maksimileveydet samalla periaatteella kuin aikajanalle: minimiksi 320 pikseliä pienen mobiilinäyttökoon mukaan ja maksimiksi 600 pikseliä asetetun leveyden mukaan. Vaihdoin Stagen väriksi eli bannerin taustaväriksi haalean sinisen sävyn.

Stagen ollessa valmis toin kaikki tarvittavat grafiikat sille. Maa- ja aalto-elementtejä ei voinut käyttää sellaisinaan, vaan niiden ominaisuuksia piti hiukan muuttaa. Syy tähän oli yksinkertaisesti se, että vaikka tausta kapenisi näyttöleveyden mukaan, sillä olevat kuvat eivät. Tämän korjaamiseksi kuville piti asettaa yksi Layout Preset-asetuksista: Clip Background Image (kuvio 25). Asetus sijoittaa kuvan vasempaan reunaan ja leikkaa kuvaa, muttei skaalaa sitä.



Kuvio 25. Clip Background Image.

Toteutin bannerin osittain aiemmin mainitsemani Web Designer-sivuston tutoriaalin toimintatavalla ja kyseisestä tutoriaalista sain apua Layout Presetin valintaan. Olin unohtanut, että kuvien asetuksia olisikin muutettava, tämän vuoksi jouduin jälkikäteen muuttamaan aalto-elementin taustan Illustratorissa bannerin suuruiseksi, jotta taustakuvan leikkausasetus toimisi.

Responsiivisuuden osalta muita asetuksia ei tarvinnut muuttaa, joten seuraavaksi pystyi keskittymään varsinaiseen animaatioon. Piirsin taajuuskorjaimen palkit

Rectangle-työkalulla. Animoin palkkien kokomuutokset tapahtumaan 0,25 sekunnin välein, ja tämä vaati aika paljon työtä.

Listen-painikkeen laitoin laskeutumaan näkyviin ylhäältä kolmen sekunnin kohdalla ja siirtymä kesti sekunnin. Siirtymän Easing-tavaksi valitsin EaseOutBackin (oletustapana on Linear eli suoraviivainen siirtymä) eli painike menee suoraa hieman alemmas kuin on tarkoitus, mutta ponnahtaa ylöspäin asettuen varsinaiselle paikalleen. Easing tarkoittaa siis liikkeen etenemistapaa, tavallisimmin kiihdytystä tai hidastumista (Chun 2015, 135). Muutin myös painikkeen kursorityyliksi ”pointer” eli osoittimen, vaikka en nyt lisännytkään mitään linkkiä tai soitinta painikkeeseen.

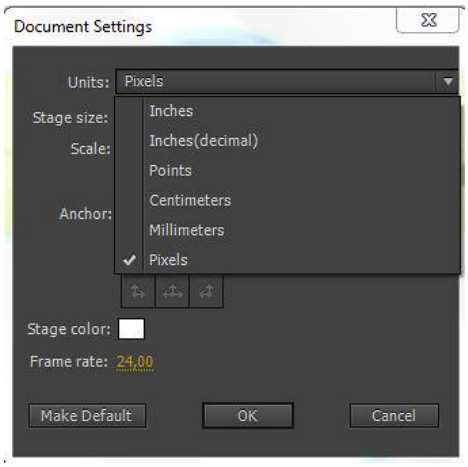
Animoin vielä aallon liikkuvaksi. Sen ja taajuuskorjaimen liikkumisen kestoksi tuli viisi sekuntia, jonka jälkeen animaatio on kolme sekuntia paikallaan, kunnes se alkaa alusta. Asetin siis kahdeksan sekunnin kohdalle sym.play();-laukaisimen. Valmis animaatio on tarkasteltavissa kuvina kahdessa eri näyttökoossa liitteessä 2.

5.3 Toteutus Flash Professionalilla

Tässä kappaleessa selvitän, miten toteutin infografiikan ja bannerin Adobe Flash Professionalilla. Koska tein kummankin animaation ensin Edge Animatella, vertan tässä kappaleessa hieman sen toteutustapoja Flashin vastaaviin. Kappaleet käsittelevät animaation tekoa suppeammin, koska responsiivisten HTML5-toteutusten teko ei ole mahdollista Flashilla.

5.3.1 Infografiikan toteutus

Flash Prolla infografiikan toteutus oli nopeampi prosessi. Osittain tämä johtui siitä, että selvisi virallisesti, ettei responsiivisen animaation toteutus onnistu Flashilla ollenkaan, vaikka animaatio olisi julkaistu HTML-muotoisena. Stagen kokoa ei voi säätää prosenteissa eikä siten minkään muunkaan elementin, kuten kuvion 26 Document setting-ikkunasta näkee. Sama tilanne on myös tällä hetkellä ohjelman uusimmassa versiossa, jota minulla oli mahdollisuus kokeilla. Tästä syystä tein animaation vain 400 pikseliä leveänä, jotta se voisi olla käyttökelpoinen mobiilikokoisenakin.



Kuvio 26. Stagen mittayksikkövaihtoehdot.

En ollut varautunut siihenkään, että käytössäni olevalla versiolla SVG-tiedostoja ei voi tuoda ohjelmaan. Sen sijaan vektorigrafiikkaa voi tuoda Illustrator- eli .ai-tiedostoina, joita käytinkin. Nyt oli mahdollista käyttää erillisenä tiedostona mittaa ja suurennuslasia eli vuosilukujen tasot saattoi laittaa suurennuslasin tason alle, toisin sanoen animoida ne siirtymään lasin alta. Animaation toteutus oli yllättävän yksinkertaista myös siksi, että vuosilukuja ei näkyisi suurennuslasin sivuilla kuin yksi kerrallaan. Saatoin myös laittaa useamman vuosiluvun samalle tasolle (eli yhdeksi symboliksi animointia ajatellen), koska niitä ei ollut tarpeen piilottaa.

Koska animaatiosta ei saanut skaalautuvaa, suurennuslasissa näkyvien tekstien tekeminen sujui täysin Flashin tekstityökalulla. Näiden tekstien näkyviin/näkymättömiin siirtymisestä en tosin tehnyt vaiheittaisia, mutta tein peittävyyden muutosefektin. Tätä väriefektiä kutsutaan Flashissa nimellä Alpha. Tämä johtui siitä, etten tehnyt tekstien tasoille Tweeniä siirtymää varten, vaan ainoastaan keyframet Alpha-arvoilla 0 ja 100 prosenttia. Liitteestä 1 on nähtävissä kuvana, millaiselta lopputulos animaatiosta näyttää selaimessa.

5.3.2 Bannerin toteutus

Aikajana-animaation toteutusvaiheessa selvisi tärkeitä eroja ja rajoituksia Flashin suhteen, ja bannerin toteutuksen myötä niitä tuli vastaan ainakin yksi lisää: en saanut bannerin taustaa eli Stagea samanväriseksi kuin mitä Edge Animatella olin toteuttanut. Piirretyille muodoille voi sen sijaan valita minkä tahansa värin eli

esimerkiksi kopioida värin RGB- tai HEX-arvon väripaletin kenttään. Tämän vuoksi bannerin taustaksi oli piirrettävä neliö, jolle sain oikean värisävyn asetettua.

Koska Flashin rajoitukset olivat tiedossa, päätin tehdä bannerista 320 pikseliä leveän eli siihen valikoituivat vain ”mobiiliversion” elementit. Animaatioiden toteutukset onnistuivat ennakoimillani tavoilla. En tosin kokeillut tehdä painikkeelle vastaavaa ponnahdusanimaatiota kuin Edge Animaten EaseOutBack. Flashissa ei nimittäin ole valmiita Easing-tapoja, mutta Ease-arvon voi kuitenkin määrittää itse. Arvoasteikko on -100-100: negatiivinen arvo aiheuttaa asteittaisemman muutoksen aloituspisteestä ja tämä tunnetaan termillä ”Ease in”, kun taas positiivinen arvo tekee lopetuksesta asteittaisemman ”Ease Outin” (Chun 2015, 135).

Animoin painikkeen siis siirtymään Stagelle Motion Tweeniä käyttäen, mutta Flash ilmoitti Output-paneelissa animaation selaintestausvaiheessa, että Motion Tweenit julkaistaan freme-by-frame-animaatioina eli Classic Tweeniä tulee käyttää aina kun mahdollista. Sama varoitus näkyi myös aikajana-animaation yhteydessä.

Taajuuskorjainpalkkien liikkeet toteutin juurikin klassista Tweeniä käyttäen, mutta huomasin sitten, että jostain syystä Tween aiheutti palkin pyörähdyksen muutaman kerran. Ensimmäisen kerran kykenin poistamaan efektin vain skaalamalla palkkia uudelleen. Sen sijaan animaation työstön loppuvaiheessa en pystynyt korjaamaan paria pyörähdystä halutunlaiseksi muuta kuin poistamalla Tweenin kokonaan. Lopputulos oli kuitenkin täysin oikeanlainen frame-by-frame-animaationa. Liitteestä 2 on nähtävissä kuvana, millaiselta lopputulos animaatiosta näyttää selaimessa.

6 YHTEENVETO JA JOHTOPÄÄTÖKSET

Keskeisin tutkimusongelmani oli selvittää, onko Edge Animate ylivoimainen ohjelma responsiivisien animaatioiden toteutuksessa verrattuna Flash Professionaliin sekä miten ja millä tekniikoilla skaalautuvat animaatiot voidaan toteuttaa kyseisillä ohjelmilla. Tutkimuksessani selvisi, ettei Flash Professionalin uusimmissa versioissakaan ole työkaluja tai ominaisuuksia responsiivisuuden aikaansaamiseksi. Skaalautuvan vektorigrafiikankaan tuonti ohjelmaan ei onnistu. Saatoin kuitenkin todeta, että Flashia on joltain osin yksinkertaisempaa käyttää kuin Edge Animatea. Erityisesti aikajanaa on selkeämpi käyttää, koska sen tasoille ei muodostu keyframejen ja animoitujen osioiden myötä lisämerkintöjä toisin kuin Edge Animatessa.

Responsiivisten animaatioiden toteutustapojen selvittäminen oli siis mahdollista vain Edge Animaten osalta. Edge Animatea käsittelevillä teoksilla olikin suurin merkitys käytännön osuudessa ja aiheen mukaisessa selvitystyössä, koska ohjeiden lisäksi teokset valottivat teoreettista taustaa tekniikoiden takana. Muu yleisempi teoria web-teknologioista ja responsiivisesta suunnittelusta auttoi toki myös, esimerkiksi SVG-kuvien käytöstä sain paljon tietoa. Animaatioiden osalta työni empiirinen osuus vastasi kuitenkin eniten tavoitteisiini.

Toteutin bannerin ja aikajanan responsiivisuuden eri tavoilla: aikajana skaalaantuu kokonaan ja banneri leveyden osalta. Molemmat onnistuivat mielestäni aika hyvin. Bannerin ulkonäöstä tuli yksinkertaisempi kuin aikajana-animaation, koska käytin siihen vähemmän aikaa, mutta banneri onnistui suunnitelmani mukaisesti toisin kuin aikajana. Aikajanan suunnitteluvaiheessa en tosin ollut varma, miten toteutus onnistuu, koska olin vasta aloittanut teorian opiskelun. Kyseessä oli enemmänkin kokeilusuunnitelma eli päätin yrittää, onnistuisiko visioni. Edge Animatella tehty lopputulos skaalaantuu kuitenkin järkevästi ja olen sen yleisilmeeseen tyytyväinen. Toteutuksen suurimmat miinuspuolet ovat vuosiluvut mobiilinäkymässä: suurennuslasin viereen kiinni tulevat luvut menevät hiukan lasin päälle.

Lisäksi SVG-kuvien tekstit näkyvät selaimissa Times New Roman-fonttina, paitsi Androidin Firefox- ja Chrome-selaimilla sekä projektikoneeni Firefoxilla, jonka kautta otin Edge Animate-toteutuksista kuvankaappaukset liitteeseen 1. SVG-kuva vaatii siis web-fonttia näyttääkseen sen tekstin selaimella oikein. En silti keksi selitystä, miksi fontit toimivat vain joissain laitteissa enkä sitä, että Flashillä tehdyt animaatiot näkyvät tyhjinä eli valkoisina ruutuina, mutta toimivat kyllä omalla koneellani kaikilla selaimilla. Tämä toimii hyvänä osoituksena siitä, että on tärkeää muistaa testata mitä tahansa HTML-toteutusta mahdollisimman monilla laitteilla ja selaimilla.

Animaatioiden tekeminen sujui kaiken kaikkiaan hyvin, sillä pääsin ongelmista eteenpäin lähdemateriaalin ja omien oivalluksieni avulla. Esimerkiksi Edge Animatessa tehtyjen tekstien koon staattisuus oli tällainen ongelma. Muuten opinnäyteprosessini sisälsikin pitkälti oleellisen teorian etsimistä, valitsemista ja muokkaamista lähdeaineistosta. Opin ymmärtämään paremmin web-teknologioita ja sain uutta tietoa responsiivisessa suunnittelusta, mistä minulla on aikaisempaakin kokemusta verkkosivujen osalta. Juuri web- ja graafinen suunnittelu ovat eräitä ammatillisia kiinnostuksen kohteitani ja koen työni kehittäneen näitä osaamisalueita. Animointiprosessi puolestaan syvensi Edge Animaten ja Flash Professionalin osaamista sen ohella, että onnistuin selvittämään tapoja toteuttaa animaatioita responsiivisesti Edge Animatella.

Saavutin opinnäytteeni tavoitteet, mutta se, että Flash ei tukenut responsiivista animaatiota, vei osan opinnäytteeni ideasta eli en voinut verrata ohjelmia tässä mielessä keskenään. Muita responsiivisia animaatioita tukevia animaatio-ohjelmia en löytänyt myöskään. Osa ideaani oli verrata saman yrityksen samaan tarkoitukseen käytettäviä tuotteita. Pääasia kuitenkin oli, että kykenin toteuttamaan mukautuvan animaation Edge Animatessa.

Saatoin myös verrata Flashin edustamaa menneisyyttä ja Edge Animaten nykyisyyttä ikään kuin konkretisoimaan sitä vastakkainasettelua, mitä käsittelin teoriaosuudessa. HTML5:n sijasta käytetään Flashia ja perinteisten työpöytäkoneiden ja kannettavien sijasta käytetään yhtä enemmän tabletteja ja älypuhelimia. Itse

olen huomannut, että monet sivut eivät näy tai toimi puhelimella moitteettomasti, vaikka ne olisivatkin rakennettu responsiivisiksi. Siksi on hyvin ajankohtaista alkaa suunnitella kaikki sivustojen elementit huolella testaten myös pienimpiä laitekokoja. Helppoa se ei välttämättä ole, mutta työni osoittaa, miten valmiilla työkalulla voi luoda näytön mukaan skaalautuvia animaatioita tai grafiikkaa suhteellisen helposti.

Toimeksiantoni oli selvittää Mainostoimisto C2:lle tekniikat, jotta ohjelmointia osaamattomatkin voisivat toteuttaa animaatioita responsiivisille sivustoille. Mainostoimistossahan graafiset suunnittelijat eivät tavallisesti ole ohjelmoijia, mutta myös he voivat nyt tehdä HTML5-animaatioita alusta loppuun itse tarpeen mukaan Edge Animatella. Edge Animate on lisäksi helposti omaksuttavissa, sillä käsitykseni mukaan lähes kaikki graafisista suunnittelijoista kuitenkin käyttävät tunnetuimpia Adoben grafiikkaohjelmia. Opinnäytteeni ei tarjoakaan suoranaisesti käyttöohjetta, mutta peruskäytön opettelu onnistuu hyvin esimerkiksi ohjelman Lessons-tutoriaalien avulla.

Olen tyytyväinen tutkimustuloksiini responsiivisten animaatioiden toteutuksen osalta. Animaatioiden tekemiseen voisi löytyä vielä muitakin kuin käsittelemiäni tapoja, mutta Edge Animatella on selvästi rajoituksensa verrattuna kokonaan ohjelmoinnilla tehtäviin animaatioihin. Minulla oli myös työssäni vapaus suunnitella itse tutkimustehtävieni aiheet ja tyylit. Jos nämä tehtävät olisivatkin olleet toimeksiantajan tarkasti määrittelemiä, se olisi voinut tuoda lisää haastetta työlleni. Tällaisenaan tutkimukseni luotettavuutta kuitenkin kohentaa sen tarkka toteuttaminen tutkimuksen jokaisessa vaiheessa. Tein tutkimusolosuhteista totuudenmukaiset ja selkeät. Olen myös arvioinut työnkulkuani ja esittänyt valinnoilleni perusteluja.

LÄHTEET

Adobe Systems Inc. 2015. Adobe Creative Cloud Edge Animate CC. Viitattu 22.7.2015. https://creative.adobe.com/fi/products/animate

Adobe Systems Software Ireland Ltd. 2015a. Adobe Flash Professional CC. Viitattu 15.7.2015. http://www.adobe.com/fi/products/flash.html

Adobe Systems Software Ireland Ltd. 2015b. Adobe Creative Cloud. Viitattu 10.7.2015. http://www.adobe.com/fi/creativecloud.html

Adobe Systems Software Ireland Ltd. 2015c. Adobe Flash Professional CC. Viitattu 22.7.2015. http://www.adobe.com/fi/products/flash/features.html

Brimelow, L. 2012. HTML5 for Flash Developers: Comparing ActionScript and JavaScript. Viitattu 3.8.2015. http://www.lynda.com/Flash-Professional-tutorials/Comparing-ActionScript-JavaScript/97148/107020-4.html

Carver, M. 2015. The responsible web. Shelter Island, New York. Manning Publications.

Chun, R. 2015. Adobe Flash Professional CC Classroom in a Book. San Jose, California. Adobe Press.

Crowther, R, Lennon, J, Ash, B & Wanish, G. 2014. HTML5 in action. Shelter Island, New York. Manning Publications.

Grover, G. 2013. Adobe Edge Animate: the missing manual. Sebastopol. O'Reilly Media, Inc.

Hakala, I. & Pyykkö, S. 2014. Infografiikalla saat viestisi perille. Viitattu 12.8.2015. http://www.sopivadesign.fi/infografiikka.html

Hartley, J. 2014. Using Font Awesome Pretty Much Anywhere. Viitattu 22.10.2015. http://www.johnbhartley.com/2014/using-font-awesome-pretty-much-anywhere/

The jQuery Foundation. 2015. What is jQuery?. Viitattu 3.7.2015. http://jquery.com/

Järvilehto, T. 2012. Bannerit eli display-mainonta. Teoksessa Häivälä, J. & Paloheimo, T., Klikkaa tästä– Internetmarkkinoinnin käsikirja 2.0. Vaasa. Mainostajien liitto.

Korpela, J. 2013. CSS3 – uudet ominaisuudet. Jyväskylä. Docendo.

Korpela, J. 2014. HTML5-käsikirja. Jyväskylä. Docendo.

Leiniö, T. 2012. Mitä on responsiivinen design?. Viitattu 4.7.2015. https://www.sofokus.com/blogi/mita-on-responsiivinen-design/

Microsoft. 2015a. SVG. Viitattu 9.7.2015. https://msdn.microsoft.com/library/hh673562(v=vs.85).aspx;

Microsoft. 2015b. Windows XP:n tuki on päättynyt. Viitattu 9.7.2015. http://windows.microsoft.com/fi-fi/windows/end-support-help

Rohde, M. 2013. Adobe Edge Animate CC For Dummies. Hoboken, New Jersey. John Wiley & Sons, Inc.

Ströh, J. 2014a. Migrating from Flash to Edge Animate: A history and comparison of Flash and Edge Animate. Viitattu 19.4.2015. http://www.lynda.com/Edge-Animate-tutorials/history-comparison-Flash-Edge-Animate/158312/179713-4.html

Ströh, J. 2014b. Migrating from Flash to Edge Animate: Comparing Flash and Edge Animate interfaces. Viitattu 22.5.2015. http://www.lynda.com/Edge-Animate-tutorials/Comparing-Flash-Edge-Animate-interfaces/158312/179715-4.html

Ströh, J. 2014c. Migrating from Flash to Edge Animate: Comparing JavaScript and ActionScript 3.0. Viitattu 22.5.2015. http://www.lynda.com/Edge-Animate-tutorials/Comparing-JavaScript-ActionScript-30/158312/179726-4.html

W3C. 2015a. HTML5. Viitattu 30.6.2015. http://www.w3.org/TR/html5/

W3C. 2015b. HTML 5.1. Viitattu 30.6.2015. http://www.w3.org/TR/html51/

W3Schools. 2015a. jQuery Introduction. Viitattu 3.7.2015. http://www.w3schools.com/jquery/jquery_intro.asp

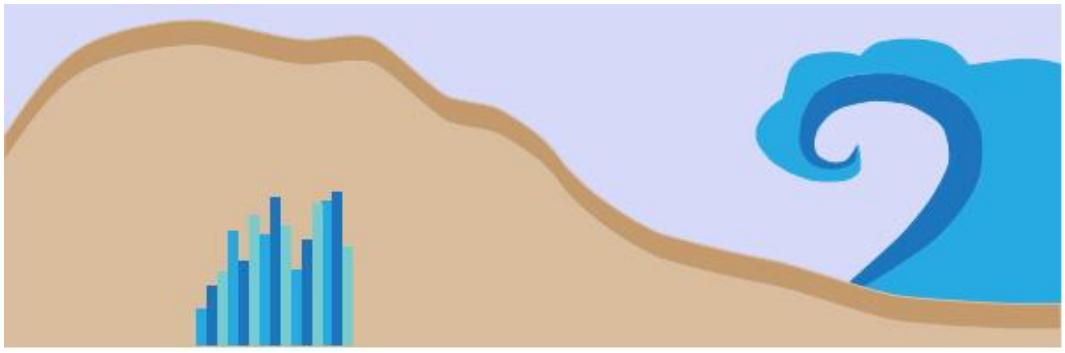
W3Schools. 2015b. CSS3 Introduction. Viitattu 2.7.2015. http://www.w3schools.com/css/css3_intro.asp

AIKAJANA-ANIMAATIO KUVINA

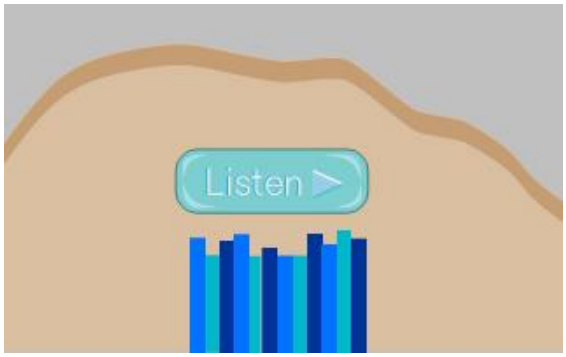
Edge Animate-toteutus 700 ja 320 pikseliä leveänä

Flash Professional-toteutus (400 pikseliä leveä)

BANNERI-ANIMAATIO KUVINA



Edge Animate-toteutus 600 ja 320 pikseliä leveänä



Flash Professional-toteutus (320 pikseliä leveä)