Sini Makkonen

# ANIMOINNIN MERKITYS WEB-KEHITYKSESSÄ

Opinnäytetyö
Tietojenkäsittely

Marraskuu 2015

KUVAILULEHTI

| | |
|---|---|
| MAMK University of Applied Sciences | **Opinnäytetyön päivämäärä**<br>30.11.2015 |
| **Tekijä(t)**<br>Sini Makkonen | **Koulutusohjelma ja suuntautuminen**<br>Tietojenkäsittelyn koulutusohjelma |

**Nimeke**

Animoinnin merkitys web-kehityksessä


**Tiivistelmä**

Animoinnin merkitystä web-kehityksessä ei yleisesti ottaen huomioida riittävästi. Ilman animointia web-sivut ovat kuolleet ja elottomat. Animointi on se tekijä, joka herättää sivut eloon ja käyttäjä kokee sivujen olevan vuorovaikutuksessa itsensä kanssa.

Opinnäytetyöni tarkoitus on tutkia, mitkä ovat nykyaikaisimmat animointitekniikat web-kehityksessä tällä hetkellä ja mikä on niiden merkitys web-sivuilla. Työni valottaa animaatioiden hyödyntämisessä huomioitavia seikkoja, joita ei muutoin tule välttämättä edes ajatelleeksi. Käsittelen animaatioita visuaalisuuden ja käytettävyyden kannalta sekä perehdyn animaatioiden suunnitteluun.

Työtäni varten tein asiantuntijahaastatteluita ja kyselyn. Asiantuntijoiden näkemyksistä sain reaaliaikaista tietoa animointien käytöstä web-kehityksessä. Keskityn työssäni käsittelemään kyselyssä ja haastatteluissa esiin nousseita tekniikoita ja käsittelen 2D-animointia ohjelmointikielien kannalta. Esittelen erilaisia HTML5, CSS3, JavaScript ja SVG-animaatioita, joista olen kasannut käytännönosiossa myös esimerkkisivun, miten animaatioita voidaan hyödyntää käytännössä.



**Asiasanat (avainsanat)**

Animaatio, CSS, HTML, JavaScript, SVG

| | |
|---|---|
| **Sivumäärä**<br>35 | **Kieli**<br>**Suomi** |

**Huomautus (huomautukset liitteistä)**


| | |
|---|---|
| **Ohjaavan opettajan nimi**<br>Miia Liukkonen | **Opinnäytetyön toimeksiantaja** |

DESCRIPTION

| | |
|---|---|
| MAMK University of Applied Sciences | **Date of the bachelor's thesis**<br>November 30, 2015 |
| **Author(s)**<br>Sini Makkonen | **Degree programme and option**<br>Business information technology |

**Name of the bachelor's thesis**

The meaning of animation in web development


**Abstract**

The meaning of animation is not always sufficiently taken into account. Without animation webpages are liveless and dead. The factor that brings webpages to live is animation, and because of this, the users feel that the webpages are in interaction with them.

The purpose of my thesis was to research which was the modern animation techniques of present day wise and what the meaning of animation in webpages was. I clarified some factors to consider for exploitation of animation and which one propably wouldn't even think of otherwise. I dealt with animation through visuality and functionality and I also explored the design of animations.

For my work I carried out expert interviews and a survey. I received real-time information about using animation in web development from these expert views. I focused on the techniques that emerged throught the survey and the interviews. My work was about 2D-animation from the programming language point of view, and I introduced a variety of animations made with HTML5, CSS3, JavaScript and SVG. I also made an example webpage in the practical section on how animations could be used in practice.



**Subject headings, (keywords)**

Animation, CSS, HTML, JavaScript, SVG


| | |
|---|---|
| **Pages**<br>35 | **Language**<br>**Finnish** |

**Remarks, notes on appendices**

| | |
|---|---|
| **Tutor**<br>Miia Liukkonen | **Bachelor's thesis assigned by** |

**SISÄLTÖ**

# 1 JOHDANTO

Tämä opinnäytetyö on pitkälti lähtenyt minun oman mielenkiintoni pohjalta animointiin. Tämä aihe on myös tärkeä, sillä tätä aihetta ei ole aikaisemmin käsitelty tästä näkökulmasta opinnäytetyössä. Työni tarkoitus on tutkia, mitkä ovat nykyaikaisimmat tekniikat animaatioiden eli kaiken liikkuvan sisällön toteuttamiseen helposti ja näyttävästi web-sivuilla. Käsittelen tätä aihetta web-kehittäjän näkökulmasta ja nojaan opinnäytetyössäni tekemieni asiantuntijahaastatteluiden kautta saamiini tietoihin. Avaan näitä haastattelutuloksia paremmin käytännön osuudessa ja käytän niitä myös lähteenä teoriaosiossa. Tavoitteenani on tutkia työssäni haastatteluiden kautta nousseita tekniikoita animaatioiden toteuttamiseen. Tutkin myös, miten animaatioita kannattaa hyödyntää visuaalisuuden ja käytettävyyden kannalta. Opinnäytetyöni rajoittuu käsittelemään 2D-animaatioita web-kehityksessä. Työn ulkopuolelle rajaan 3D-animoinnin, koska niiden käsittelemiseksi täytyisi keskittyä erilaisiin animointiohjelmistoihin, sillä tahdon käsitellä aihetta ohjelmointikielien kautta.

Animointi on tehokas tapa kiinnittää ihmisen huomio, sillä ihmisen silmä hakeutuu automaattisesti liikkeen puoleen. Animointia voidaan käyttää monenlaisina elementteinä ja tehostekeinoina. Animointia voi tapahtua esimerkiksi hiiren liikkuessa jonkin elementin päälle tai se voi olla jatkuvasti pyörivä mainos web-sivuilla. Animaatioiden käyttö historian kuluessa on lisääntynyt web-kehityksessä tekniikan ja ohjelmointikielien kehittyessä. Nykyisin animaatioita voidaan tehdä monenlaisilla ohjelmointikielillä ja ohjelmilla, joita on myös kattavasti saatavilla ilmaiseksi. Animaatioita voidaan tehdä web-kehityksessä myös erilaisilla animaatiokirjastoilla, jotka käyttävät yleensä JavaScript-pohjaisia kieliä, CSS3- ja HTML5-kieliä liikkuvien elementtien luomiseen. Nämä helpottavat paljon animaatioiden tekemisessä. Animaatioita voidaan myös tehdä erilaisilla ohjelmistoilla, joilla saadaan aikaisiksi esimerkiksi piirrettyjä animaatioita. Tällaiset animaatiot lisätään web-sivuille videomuodossa, jolloin täytyy kehityksessä huomioida se, että kaikki selaimet eivät tue kaikkia videomuotoja.

Animaatioita voidaan käyttää web-sivuilla esimerkiksi siirtymisiin, nappeihin ja mainoksiin. Tällaiset pienet visuaaliset efektit voivat luoda sisällöstä mielenkiintoisemman, mutta ollessaan liian suuria tai voimakkaita ne voivat häiritä ja vaikuttaa käytettävyyteen. Käytettävyyteen vaikuttaa myös jos animaatiot ovat raskaita suorittaa, jol-

loin ne hidastavat sivujen toimintaa. Nykyaikaisissa animaatioissa tärkeitä ominaisuuksia ovatkin nopeus ja keveys.

Hyvä animaatio on tasapainoinen ja sisältöä rikastuttava elementti. Se ohjaa sivujen käytössä eikä ole keskittymistä häiritsevä tai käytettävyyttä heikentävä. Värien käyttöä kannattaa myös miettiä siinä valossa, mikä on animaation tarkoitus sivuilla. Väreillä voidaan viestittää käyttäjälle esimerkiksi, että vihreästä napista hyväksytään valinta ja punaisesta peruutetaan. Värejä käyttämällä saadaan myös korostettua haluttuja elementtejä sivuilla, joihin tahdotaan käyttäjän kiinnittävän huomiota.

Opinnäytetyössäni kerron lyhyesti animoinnin historiasta ja kehityksestä. Käsittelen työssäni pääasiallisesti HTML5-, CSS3-, SVG- ja JavaScript-animaatioita. Vastaan myös kysymykseen millainen on hyvä animaatio visuaalisessa mielessä ja perehdyn animaatioiden suunnitteluun. Tarkastelen animaatioita käytettävyyden kannalta ja annan muutamia vinkkejä animaatioiden suorituskyvyn parantamiseksi. Käytännönosuudessa kerron tarkemmin esimerkkien avulla erilaisista animaatioden hyödyntämistavoista ja tilanteista. Esimerkeissä esittelen myös yhdenlaisen animoidun sivukokonaisuuden.

## 2 ANIMOINNIN HISTORIAA

Animaatio on tapa päästä lähemmäksi elävää tunnetta, ja siksi sillä saadaankin websivut heräämään henkiin. Animoidut elementit ovat parhaimmillaan interaktiivisia osioita, joiden kanssa käyttäjä voi olla vuorovaikutuksessa. Animointi ja liikkuva kuva on kiinnostanut ihmisiä kautta aikojen. Eikä ihme, sillä luonnossa kaikki liikkuu. Olipa liike sitten nopeaa tai hidasta, niin sen ilmaisuun kuvankeinoin on pyritty kautta aikojen. Esimerkiksi esihistoriallisissa luolamaalauksissakin kuvatuille eläimille on todennäköisesti tästä syystä piirretty useita jalkoja, jotka kuvastavat jalkojen liikettä. Aikaisimpia löytöjä selvästi animaatioon pyrkivästä esitysmuodosta on tehty 5 200 vuotta sitten valmistetun Iranista löytyneen kulhon kylkeen. Kulhoon on maalattu viisikuvainen kuvasarja vuohen hypähtelevästä liikehdinnästä. (Ball 2008; Bruni 2015.)

Historiassa on ollut monenlaisia laitteita ja tapoja tuottaa liikkuvaa kuvaa kuvasarjoina. Piirrosanimaatioiden muodossa animoinnin voidaan kuitenkin katsoa saaneen al-

kunsa 1900-luvun alulla. Silloin valmistui ensimmäisiä piirrosanimaatioita kuten Fantasmagorie (1908), Little Nemo (1911) ja myöhemmin vuonna 1928 Disneyn ensimmäinen äänellinen animaatio Steamboat Willie (McLaughlin 2001). Tästä lähtien piirrosanimaatioita ja sen tekniikoita on kehitetty koko ajan eteenpäin. Vaikka piirrosanimaatiot eivät ole vielä menettäneet suosiotaan, on uusia animointitekniikoita tullut paljon lisää. Webin käyttöönottamisen jälkeen ja tietokoneiden kehittyessä on pystytty grafiikkaa ja animaatioita tekemään erilaisilla ohjelmilla ja ohjelmointikielillä.

Animoinnin kehitys webissä sai alkunsa kun ensi kertaa tuli mahdolliseksi liittää kuvia webiin tekstin lisäksi. Vuonna 1987 ComputerServe Incorporated kehitti formaatin nimeltä GIF (Graphics Interchange Format), mikä teki mahdolliseksi toistaa kuvasarja kuvia jatkuvalla kierrolla. Vaikka niiden laatu voi näyttää hieman pikselöityneeltä johtuen GIF-animaatioiden pienestä värimäärästä, on GIF edelleen hyvin käytetty pienissä webissä levitettävissä lyhyt animaatioissa. GIF:in 8-bittisen värikartan värien maksimimäärä on vain 256, joka saa suurempi värimääräiset kuvat näyttämään pikselöityneiltä käännettäessä GIF-formaattiin. GIF-animaatioiden hyvä puoli on kuitenkin se, että ne ovat laajalti tuettuja. (CompuServe Incorporated 1987; Harris 2015.) Vuonna 1995 julkaistiin uusi formaatti PNG (Portable Network Graphics), jonka oli tarkoitus korvata GIF-formaatti. GIF-formaatin suosio laski näihin aikoihin huimasti, koska CompuServe ilmoitti, että kaikkien GIF-animaatioita käyttävien olisi maksettava rojalteja formaatista. GIF-animaatioissa käytettävän pakkausmetodin kehittäjät olivat näet patentoineet kyseisen LZW-pakkausmetodin. PNG-formaatti oli taas täysin patentoimaton ja maksuton formaatti. Näihin aikoihin tietokoneet olivat jo muutenkin sen verran kehittyneitä, että GIF-formaatin 256 värinen väripaletti oli liian suppea. PNG tarjosi tuen peräti 48-bittisille RGB-kuville ja 64-bittisille RGBA-kuville. (Roelofs 2015; Lilley 2006.)

Myöhemmin vuonna 1996 Flashin ilmestyessä se sai valtaisan suosion. Se mahdollisti aivan uudenlaisien animaatioiden hyödyntämisen ja siksi se levisikin nopeasti hyvin laajalle. Flash toi mukanaan mm. Flash-pelit, joita pian web ilmestyikin pullolleen. Flash oli jotain uutta ja hienoa, jota ei ollut aikaisemmin koettu. Ajan kuluessa webkehittäjät siirtyivät kuitenkin käyttämään enenevissä määrin JavaScript-pohjaisia animaatioita. Myöhemmin vuonna 2009 julkaistiin CSS3:n animation-ominaisuus, joka onkin sen jälkeen saanut suurta suosiota. CSS3-animaatioita käytetään nykyisin laajal-

ti ja se on ehkä käytetyin animointitekniikka tällä hetkellä. (Bruni 2015; Computer Hope 2015.)

# 3 ERILAISET MAHDOLLISUUDET TEHDÄ ANIMAATIOITA

Animointia web-sivuille voidaan toteuttaa ohjelmoimalla tai tekemällä animaatioista videoita. Ohjelmoimalla voidaan kuitenkin tehdä animaatioita myös eritoiminnallisuuksiin kuten nappien painamiseen, sivujen vaihtumiseen, kuvien zoomaukseen ja moniin muihin toiminnollisuuksiin jotka voivat olla myös interaktiivisia. Videomuotoisten animaatioiden tekoon tarvitaan myös erillisiä ohjelmia.

## 3.1 Ohjelmointikielet

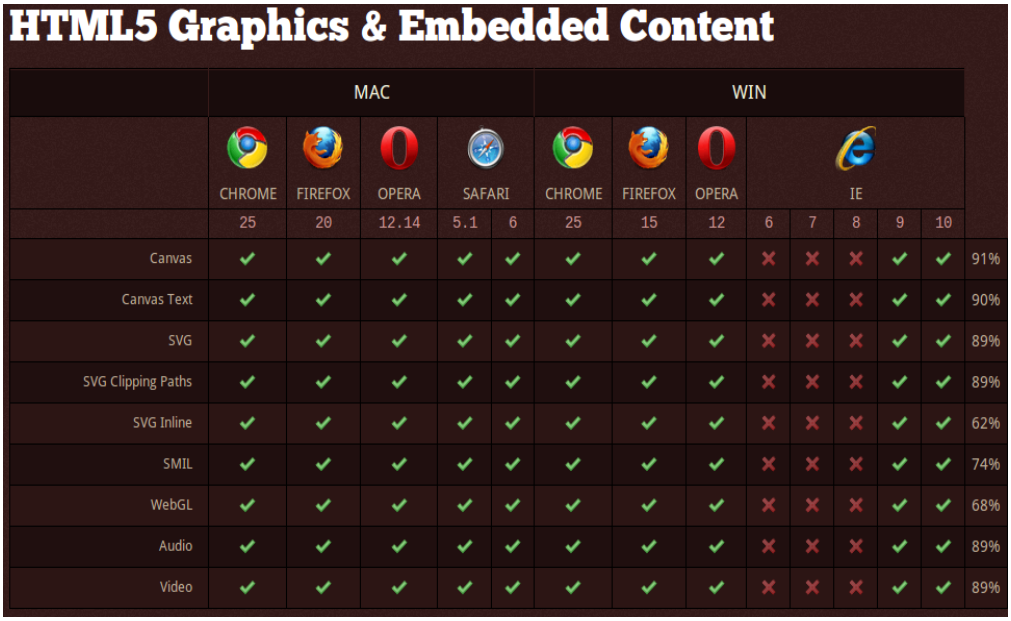
Ohjelmointikielistä moni soveltuu animaatioiden toteuttamiseen. Käsittelen tässä osiossa niistä tällä hetkellä eniten web-kehityksessä käytettyjä. Ohjelmointikieliä voidaan käyttää tehdessä erilaisia siirtymä- ja tapahtuma-animaatioita sekä muita monenlaisia liikeanimaatioita. Tällä hetkellä eniten animaatioiden tekemiseen käytetään HTML5:ttä ja CSS3:a tai JavaScriptia. Mutta se, kumpaa näistä tavoista kannattaa käyttää, riippuu siitä, millaisiin lopputuloksiin halutaan päästä. HTML5 ja CSS3 ovat tällä hetkellä yksinkertaisimpia tapoja tehdä liikkuvaa sisältöä web-sivuille. CSS3 on nopea ja helppo tapa tehdä pieniä animaatioita kuten esimerkiksi linkin värin vaihtuminen hiiren kursorin liikkuessa sen päälle. Kuitenkin jos animaatioon tarvitsee saada esimerkiksi hienompia efektejä, interaktiivisuutta ja moninaisempia toimintoja, kannattaa animaation tekoon käyttää JavaScriptia. (Lewis 2014; Shapiro 2015, 4.) Ohjelmoitaessa animaatioita voidaan apuna käyttää monenlaisia kirjastoja, jotka nopeuttavat animaatioiden tekoa valmiiksi ohjelmoiduilla toiminnoillaan. Tällaisia ovat esimerkiksi erilaiset siirtymät, pompahdukset, flippaukset yms. Näitä kirjastoja on paljon saatavilla monille eri ohjelmointikielille. Liitteessä on kuvattuna joitakin näistä kirjastomahdollisuuksista (liite 1).

HTML5 yhdessä CSS3:n kanssa on tällä hetkellä suosituimpia tapoja tehdä animointia web-sivuille. CSS3 on helppo ja hyvä tapa tehdä pieniä animaatioita mutta moninaisempiin toimintoihin se ei yllä. Monille web-kehittäjille on muotoutunut käsitys, että CSS3 on suorituskyvyltään paras tapa toteuttaa animaatioita web-sivuille ja siksi mo-

net ovatkin luopuneet JavaScript-pohjaisista animaatioista kokonaan. Tämä aiheuttaa myös sen, että animaatioille ei saada tukea vanhemmissa selaimissa. Käsitys siitä, että CSS3 on nopeampi kuin JavaScript on muotoutunut pitkälti sen pohjalta, että sitä on aina verrattu jQuery animaatioihin, jotka oikeasti ovatkin hitaita toiminnaltaan. Kuitenkin JavaScriptista on olemassa animointikirjastoja, jotka eivät pohjaudu jQuery-kirjastoon vaan ohittavat sen. Tällaisilla kirjastoilla saadaan aikaisiksi hyvin suorituskykyisiä ja vertailukelpoisia animaatioita. (Shapiro 2015, 4, 56.) Uudemmista animointikirjastoista mm. GreenSock on todettu testeissä olevan 20 kertaa jQuerya nopeampi ja päihittävän suorituskyvyllään myös CSS3-animaatiot (CSS-Tricks 2014).

### 3.1.1 HTML5

HTML5 on uusin versio HTML-merkkauskielestä, joka on luotu tekemään ohjelmoinnista helpompaa kuin aikaisemmin. Sen uudet multimediaominaisuudet ovat erittäin vaikuttavia. Nämä ominaisuudet on suunniteltu niin, että pieni tehoisillakin laitteilla pystyttäisiin pyörittämään raskasta sisältöä ilman erillisten pluginien tai API:en asentamista. HTML5 antaa hyvät mahdollisuudet animaatioiden tekemiseen uusien ominaisuuksiensa ansiosta. Sen ominaisuudet ovat myös hyvin tuettuja uudemmissa selain versioissa (kuva 1). HTML syntaksiin on lisätty uusina multimedia tageina mm. <video>, <audio> ja <canvas>, jotka helpottavat näiden sisältöjen käyttämistä websivuilla. (1st Web designer 2015.)



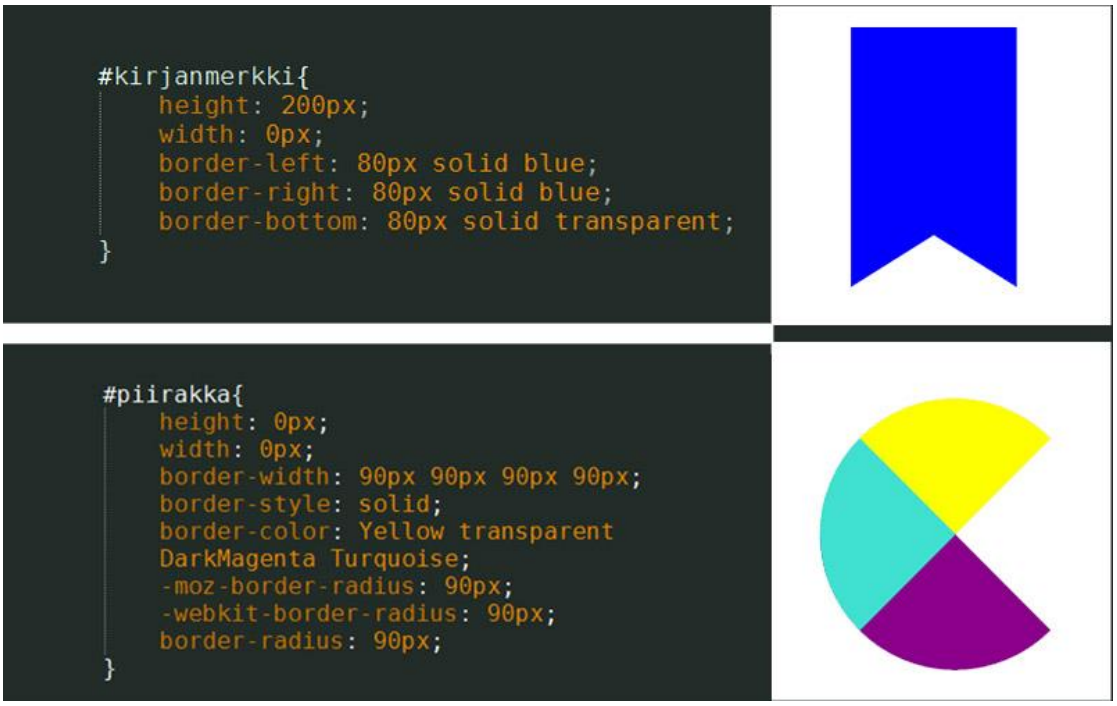

**KUVA 1. Selaimien tuki HTML5:den ominaisuuksille (Ferney 2013)**

HTML5 on tuonut tullessaan uuden animointia edistävän ominaisuuden, joka on nimeltää canvas. Se antaa web-kehittäjälle mahdollisuuden toteuttaa monenlaisia liikkuvaa grafiikkaa sisältäviä kokonaisuuksia. Canvas on suorakulmion muotoinen piirtoalusta, jolle voidaan JavaScriptia apuna käyttäen loihtia lähes mitä vain. Canvas-elementtiin voidaan lisätä vapaasti tekstiä, kuvioita, kuvia, videoita ja ääntä. Sen parhaimpia puolia ovat mm. sen täydellinen interaktiivisuus. Se voi reagoida käyttäjän hiiren liikkeisiin, näppäimistöön ja kosketukseen. Tämän johdosta canvas on erinomainen tapa tehdä pelejä ja muita vastaavanlaisia liikettä sisältäviä ratkaisuita, jotka reagoivat käyttäjän liikkeisiin ja tomintaan. Canvas on erittäin hyvin tuettu, sillä kaikki yleisimmät selaimet tukevat sitä. Canvas on myös tästä syystä tullut hyvin suosituksi, sillä se on tehokkaampi ratkaisu kuin Flash tai Silverlight. Sillä voidaan toteuttaa mm. 2D- ja 3D-pelejä, ja se on hyvä vaihtoehto Flash pohjaisille toteutuksille kuten mainos bannereille. Canvasilla voidaan toteuttaa monenlaisia opetusohjelmia sen moninaisten ominaisuuksien johdosta ja myös interaktiivisien graafien luonti onnistuu helposti. (Gerchev 2014; Pilgrim 2015.)

### 3.1.2 CSS3

CSS3 on helppo ja nopea tapa tehdä animaatioita web-sivuille. Hienon näköisten efektien kuten pyöristettyjen kulmien, varjojen ja näyttävien animaatioiden teko onnistuu vain muutamalla rivillä koodia. Näin sivuista saadaan kevyemmät ja nopeammin latautuvat, kun tiedostoja ja kuvia tarvitsee ladata sivuille vähemmän. CSS3 mahdollistaa myös mm. elementtien liikuttelun, skaalauksen ja elementin vaihtamisen toisesta CSS3 tyylistä toiseen. Uuden CSS3 animation–ominaisuuden myötä pystytään tekemään kaikkia näitä muutoksia hyödyntäviä animaatioita. Animaatioille voidaan määritellä myös keyframe:t, joiden välissä nämä muutokset tapahtuvat. Muutoksiin voidaan myös lisätä erilaisia suorituksen aikamääreitä kuten esimerkiksi ease-in, ease-out tai cubic-bezier, jolla voidaan määrittää itse suorituskäyrä. Nämä vaikuttavat animaatiossa tapahtuvan muutoksen alkamis- ja loppumisnopeuteen. Näitä määreitä on monenlaisia ja niitä voidaan hyödyntää myös JavaScriptia apuna käyttäen. (Waterhouse 2011; Wolejko 2012.)

CSS3 on erittäin tehokas käytettäväksi yksinkertaisissa animaatioissa ja sen mahdollisuuksia ei pidä aliarvioida. CSS3:lla on helppo luoda erimallisia kuvioita, joita voidaan hyödyntää animoinnin teossa. Kuvassa 2 olevat kuviot saadaan aikaisiksi hyvin nopeasti parilla rivillä koodia (kuva 2). Pyörittelemällä useampaa tämän tyylistä pyöreää piirakkakuviota päällekkäin saadaan helposti aikaisiksi esimerkiksi aikalaskuri tai latausikoni. Sinisistä kirjanmerkkityylisistä kuvioista voi kasata esimerkiksi valikon tai tehdä kuvien päälle tietolaatikoita. Monenlaisia kuvioita saadaan aikaiseksi näillä border-määreen muunnelmilla. Kuvioihin saadaan pyöreitä muotoja käyttämällä border-radius-määrettä.




**KUVA 2. CSS3 esimerkkimuotoja**

Jos animaatioista halutaan moninaisempia tai lisätä toiminnollisuuksia niihin, kannattaa apuun ottaa JavaScript. Pelkällä CSS3-kielellä tehtäessä pidemmälle vietyjä animaatiokokonaisuuksia, joudutaan käyttämään paljon kekseliäisyyttä mutta CSS3-kielellä pärjää kyllä silti hyvin pitkälle. Tietyn pisteen jälkeen kuitenkin JavaScript on järkevä ottaa rinnalle tai siirtyä kokonaan käyttämään sitä. Mutta nämä kaksi yhteen liitettynä on todella mahtava keino animaatioiden rakentamiseen. JavaScriptilla voidaan tarvittaessa manipuloida CSS3-koodia monella tavalla. Esimerkiksi niin, että animaatio muuttaa muotoaan käyttäjän syöttämien tietojen mukaan. (Conrad 2011; Shapiro 2015, 45, 56.)

### 3.1.3 JavaScript

JavaScript on saanut aiheettomasti maineen hitaampana animaatioiden suorittajana kuin CSS3, koska sitä on verrattu JavaScriptin jQuery-animaatioihin. jQuery on mahtava väline animaatioiden tekemiseen mutta se myös hidastaa JavaScriptin todellista suorituskykyä, sillä sitä ei ole suunniteltu huippu suorituskykyiseksi animaatiomoottoriksi. jQuery ylikuormittaa selainta layout-käsittelyllään samanaikaisesti, kun selain yrittää suorittaa animaatioita. Koska jQueryn peruskoodi on laaja ja se sisältää paljon muitakin ominaisuuksia kuin animoinnin, saa se muistin kuluttamisellaan aikaiseksi ”roskavyöryn” selaimen sisällä. Tämä johtaa siihen, että animaatiot saattavat nykiä arvaamattomasti. JavaScript-animaatiokirjastot, jotka sivuttavat jQueryn kokonaan, yltävät hyvään suorituskykyyn virtaviivaistamalla keskustelunsa web-sivun kanssa. Testeissä on todettu JavaScriptin olevan CSS3:a nopeampi muissa tapauksissa paitsi suoritettaessa transform-animaatioita Webkit-selaimissa. Webkit-selaimissa on kuitenkin synkronisaatio ongelmia. Muissa tapauksissa CSS käytti kaksi kertaa enemmän prosessorin tehoa verrattuna JavaScriptiin. (Shapiro 2015, 6; GreenSock 2015.)

Animointiin ei kannata käyttää JavaScriptia ainoastaan sen nopeuden vuoksi, vaan sen laajat ominaisuudet ovat vähintäänkin yhtä vaikuttavia. Kun animointia tehdään CSS3-kielellä, rajoittaa käytettäviä mahdollisuuksia luonnostaan se, mitä ominaisuuksia CSS3 sisältää. Ohjelmointi kielet taas mahdollistavat luonnostaan moninaisempia mahdollisuuksia ja animointikirjastoja apuna käyttämällä saadaan täydellinen hallinta animaatioiden liikkeille. JavaScriptin käytöstä on myös se etu, että sillä pystytään ketjuttamaan yksittäisiä animaatioita yhteen ja sillä saadaan animaatioista moninaisemmin interaktiivisia. (Shapiro 2015, 7, 9, 10.)

JavaScriptiin on olemassa useita animointikirjastoja, joilla voidaan nopeuttaa animaatioiden tekemistä (liite 1). Näitä kirjastoja on monenlaisia ja ne ovat erikoistuneet erilaisiin animointeihin. Jotkut niistä jäljittelevät mm. fysiikan lakeja kuten painovoimaa. Tällaisilla efekteillä saadaan aikaiseksi hyvin todentuntuisia animaatioita. Jotkut näistä taas keskittyvät SVG-animaatioihin ja toiset helpottavat WebGL:n ja canvas-elementtien käyttöä. (Shapiro 2015, 14.)

### 3.2 Flash

Adobe Flash on Adoben kehittämä ohjelma, jolla voidaan tehdä monenlaisia animaatioita. Flash on vielä yksi käytetyimmistä animaatiomuodoista websivuilla, vaikka se onkin siirtymässä syrjään käytettävyydeltään parempien vaihtoehtojen tieltä. Sen käyttöaste web-sivuilla laskee kokoajan voimakkaasti ja tutkimusten mukaan sitä sisältää kaikista maailman web-sivuista enää 10 %. Flashillä voidaan tehdä monenlaisia interaktiivisia vektorigrafiikka-animaatioita, jotka voivat sisältää sekä liikettä että ääntä. Animaatioista saadaan interaktiivisia käyttämällä Flashin omaa ActionScript-ohjelmointikieltä, jolla voidaan tehdä myös muita toiminnollisuuksia sisältöön. Flash on yksi tapa, jolla voidaan tehdä visuaalisesti näyttäviä web-sivuja. Flash pohjaiset interaktiiviset ratkaisut eivät kuitenkaan toimi mobiililaitteilla, mikä on nykypäivänä suuri puute. Käytettäessä Flashiä on sen liitännäinen asennettava tietokoneelle, jotta Flash sisältöä pystytään katsomaan. (Baker 2006; W3Techs 2015.)

### 3.3 SVG

SVG eli Scalable Vector Graphics on W3C:n kehittämä XML-pohjainen open source-kieli jolla voidaan luoda huippu laatuista, dynaamista ja interaktiivista vektorigrafiikkaa. SVG-grafiikka on erittäin hyvä tapa tehdä kuvia ja animaatioita, sillä se koostuu koodista, joka tekee siitä skaalautuvaa ja moninaisesti sovellettavaa. Sitä voidaan mm. indeksoida, ryhmitellä, tyylitellä, pakata sekä yhdistellä muihin objekteihin. SVG-grafiikat ovat myös tiedosto kooltaan huomattavasti pienempiä kuin normaalit PNG/JPEG kuvat. Käyttämällä jotakin ohjelmointikieltä SVG-elementtien lisänä, päästään suoraan käsiksi SVG:n DOMiin, jota kautta voidaan muokata elementin määreitä ja ominaisuuksia tai lisätä tapahtumiin käsittelijöitä. SVG:llä voidaan luoda monenlaisia kaksiulotteisia kuvia ja tekstiä vektorilaatuisena. Sillä on myös kattava tuki animaatioiden tekemiseen. SVG-grafiikkaa voidaan luoda millä tahansa tekstinkäsittely ohjelmalla, mutta kätevintä se on jollain piirustusohjelmalla kuten Adobe Illustraattorilla. (W3C 2004; W3Schools.com 2015; Shapiro 2015, 104.)

SVG:llä voidaan luoda hyvinkin monimutkaisia animaatioita ja grafiikkaa. Tehtäessä monimutkaisempia grafiikoita kannattaa avuksi ottaa jokin piirustusohjelma, jolla kuva voidaan suoraan tallentaa SVG-muotoon. Tällä tavalla kuva saadaan suoraan

muunnettua SVG-koodiksi. Grafiikka voidaan lisätä suoraan sivuille esimerkiksi objektina tai koodina. Lisäämällä koodin joukkoon hieman CSS3:a tai JavaScriptia, saadaan aikaan animaatio. Tällä hetkellä SVG animaatioiden tuki ei tosin ole kovin kattava jQuerylla tai CSS3:lla, joilla pystytään muokkaamaan vain osaa SVG-määreistä. Internet Explorer 9 ei tukea lainkaan SVG:n CSS3-transitioita, eikä CSS3-transform toimi ollenkaan Internet Explorerissa. Kattava SVG-animaatioiden tuki saadaan kuitenkin aikaan käyttämällä jotakin SVG:lle omistettua kirjastoa tai sisäänrakennetun SVG-tuen omaavaa animointikirjastoa. Hyviä SVG:n hyödyntämis paikkoja ovat animoidut napit, latausanimaatiot ja logot. (Shapiro 2015, 108–109.)

Alla on esimerkki siitä, miten pienellä määrällä koodia saadaan SVG:llä näyttävää jälkeä (kuva 3).

```
<svg viewBox="0 0 150 150">
    <radialGradient id="gradient" cx="10" cy="40" r="100"
        gradientUnits="userSpaceOnUse">
        <stop offset="0" stop-color="#66cccc"/>
        <stop offset="1"/>
    </radialGradient>
    <path d="M20,01 A90,90 0,0,1 8,60 C15,79 10,28 35,25 C72,14
        65,75 90" id="logo" fill="url(#gradient)"/>
    <use xlink:href="#logo" transform="rotate(240 34,34)"/>
    <use xlink:href="#logo" transform="rotate(120 34,34)"/>
</svg>
```

**KUVA 3. SVG esimerkki muoto**

Tämä kuva on saatu aikaan pelkällä tekstinkäsittelyohjelmalla. Piirrettyjä kuvion osia saadaan monistettua linkittämällä alkuperäinen polku sen id:n perusteella. Tässä alkuperäinen polku on monistettu kaksi kertaa ja niiden piirtokulmaa on muutettu, jotta kuvioista muodostuu yhtenäinen logo.

## 4 ANIMAATIOIDEN JA LIIKKEEN SUUNNITTELU

Animaatioiden suunnittelu on päättämistä, millaisia muutoksia halutaan objektien tekevän ja miten nämä muutokset tapahtuvat. Käyttöliittymäsuunnittelu eli sivukokonaisuuden käytettävyyden suunnittelu, on aina ollut tärkeämmässä osassa web-sivujen suunnittelua kuin animoinnin ja liikkeen suunnittelu. Tämä on johtunut osin siitä, että selaimet eivätkä laitteet ole olleet riittävän nopeita tukemaan runsasta liikkeen määrää web-sivuilla. Kuitenkin sivujen kokonaisuuden kannalta animaatio on tärkeää. Siinä

missä käyttöliittymäsuunnittelu luo rakenteen sivuille, animointi rikastuttaa ja täydentää sisältöä. Animaatiot saavat web-sivut heräämään henkiin. Tällaisia sivuja käytettäessä käyttäjä saa tunteen siitä, että sivut ovat vuorovaikutuksessa hänen kanssaan ja reagoivat hänen toimintaansa. Animointi saa siis hyvän näköiset sivut tuntumaan myös hyvältä käyttää. Animointi on eleganssia, joka muistuttaa käyttäjää teknologian taianomaisuudesta, joka saa käyttäjän palaamaan sivuille yhä uudelleen. (Shapiro 2015, 38–40.)

Mikä sitten tekee animoinnista tärkeän lisän web-sivuilla? Animointi on keino vaikuttaa käyttäjään psykologisesti. Miten käyttäjä voi esimerkiksi ilman ohjaavia animaatioita olla varma siitä, että web-sivut ovat havainneet hänen napin klikkauksensa tai tapahtuuko sivujen lataamisessa edistystä vai ovatko sivut vain kaatuneet? Näihin tilanteisiin pystytään vaikuttamaan animaation keinoin siten, että käyttäjä saa kokoajan tietoa sivujen tilasta. (Mts. 40.) Käyttäjälle voidaan animaation keinoin kertoa esimerkiksi: nyt täytyy odottaa, tästä tapahtuu siirtyminen seuraavalle sivulle, paina tästä, täydennä tämä jne. Animaatio on hyvin universaali elekieli jolla pystytään viestimään asioita ilman sanoja. Yleisesti web-kehityksessä käytetyistä animaatioista on muotoutunut standardin omaisia, jotka ymmärretään ilman minkäänlaisia kulttuurirajoja.

Web on pullollaan animointia sisältäviä sivuja, joita jokainen ihminen näkee lukemattomia määriä ollessaan webissä. Suosituimpien web-sivujen käyttämät animaatiot tulevat pikkuhiljaa käyttäjille tutuiksi ja kotoisiksi, ja he ymmärtävät näiden animaatioiden merkityksen. Tätä voidaan käyttää hyväksi sivuja suunniteltaessa, koska mitä enemmän sivut sisältävät tällaisia jo tutuksi tulleita animaatioita, sitä varmempi käyttäjä on sivujen käytössä ja sitä nopeammin hän tuntee sivut mukaviksi ja kotoisiksi. Animaationkeinoin voidaan käyttäjälle myös ilmaista, mitä jostakin käyttäjän toiminnasta tapahtuu ennen varsinaisen tapahtuman suorittamista. Näin käyttäjä saa vahvistuksen siitä, mitä elementistä tapahtuu. Tällainen animaatio voi tapahtua esimerkiksi silloin, kun kursori viedään jonkin napin päälle. Esimerkiksi palautteen lähetysnapissa näkyvä animoitu kirjekuori, joka siirtyy animoidusti postilaatikkoon. Animoinnilla voidaan myös tehdä tylsistä asioista kuten lomakkeiden täytöstä mielekkäämpää. Lomaketta voidaan animoida esimerkiksi lisäämällä ruksi hyväksytysti täytettyjen kenttien perään ja muuttamalla väärin syötettyjen kenttien värin punaiseksi. (Mts. 41–42.)

Animaatioiden liike pakottaa meidät kiinnittämään siihen huomiota, halusimme tai emme. Tämä johtuu siitä, miten meidän aivomme toimivat. Liike, joka lähestyy meitä kohti, saa meidät toimimaan välittömästi tarvittavalla tavalla. Poispäin liikkuva liike taas ei vaadi välitöntä toimintaa, mutta saa silti meidän huomiomme. Mitä mielenkiintoisempi kohde on, sitä enemmän käyttäjät haluavat olla tekemisissä sen kanssa. Esimerkiksi näyttävät, isot, liukuvärjätyt, värikkäät ja animoidut pallukkanapit suorastaan kutsuvat käyttäjää klikkaamaan niistä. Tällaisia elementtejä ja nappeja kannattaa käyttää hyödyksi ja sijoittaa sellaisiin toimintoihin, joita halutaan käyttäjän suorittavan kuten rekisteröitymisnappi tai tilauksen hyväksymisnappi. (Mts. 43.)

Käyttäjä saa sivujen toiminnasta paremman kuvan silloin, kun erilaisia animaatioita on käytetty johdonmukaisesti ja tietyt liikkeet on liitetty tiettyihin toimintoihin. Siksi sivujen liikkeitä ja animaatiota suunniteltaessa on parempi pitää erilaisten liikkeiden määrä pienenä ja hyvin harkittuna. Myös elementtien liikkeet on hyvä tehdä johdonmukaisiksi niin, että esimerkiksi kuvaan saapuva elementti poistuu takaisin sinne mistä tulikin. Jos elementti saapuu näytölle esimerkiksi oikealta ja poistuu vasemmalle, saa käyttäjä tällöin kuvan, että elementti meni johonkin eri paikkaan kuin mistä se tuli. (Mts. 44.) Tällainen efekti sopii kyllä esimerkiksi kuvien selausanimaatioon mutta ilmoitusluontoisesti esiin ponnahtavassa modaalissa kannattaa tällaista animaatiota käyttää harkiten ja ymmärtäen millaisen vaikutelman se luo.

Liian pitkät animaatiot voivat tuottaa myös ongelmia. Monesti web-suunnittelijat tekevät sen virheen tehdessään animaatioita, että niiden kestosta tulee hiukan turhan pitkiä. Tällaisia animaatioita voi olla kiva katsella muutamia kertoja, mutta jos sama pitkä animaatio toistuu useita kertoja sivuilla olon aikana, voi siitä nopeasti tulla rasite. Jos käyttäjä joutuu odottamaan pitkiä aikoja animaatioiden takia, tulee sivujen käyttämisestä käyttäjälle tuskallista ja hidasta. Hän saattaa tästä syystä turhautua ja lähteä sivuilta lopullisesti. Hyvä nyrkkisääntö animaatioiden sopivan pituuden määrittämiseksi on nopeuttaa animaatiota vielä 25 % sen jälkeen, kun sen pituus ei enää tunnu liian pitkältä kymmenen katselukerran jälkeen. (Mts. 45.)

Liiallinen animaatioiden määrä tekee web-sivuista herkästi epäammattimaisen näköisiä. Jos jonkin animaation pois jättäminen ei vaikuta käyttäjän ymmärrykseen sivujen toiminnasta, kannattaa harkita sen pois jättämistä ja tehdä sen tilalle vain muita tyylittelyjä. Suurin osa animaatioista tulisi olla hieno varaisia ja pieni eleisiä, jotta tärkeäm-

pien kohteiden animaatiot erottuisivat paremmin. Mitä enemmän sivuilla on animaatioita, sitä vähemmän käyttäjä kiinnittää niihin huomiota ja niiden merkitys huomion kiinnittäjänä häviää. (Mts. 45.)

Jos sinun suunnittelemillasi web-sivuilla on joku tietty yksi animaatio, joka toistuu vain kerran sivuilla olo aikana, panosta silloin siihen. Jos tällainen animaatio suoritetaan sivuille tulon yhteydessä, voidaan sillä luoda hyvä ensivaikutelma ja antaa persoonallisuutta sivuille. Kuitenkin kannattaa miettiä millaiseen asiayhteyteen animaatiot tulevat. Esimerkiksi leikkisää pomppivaa animaatiota ei kannata laittaa jonkin viraston sivuille, sillä se veisi sivujen sisällöltä vakavuuden ja tärkeyden. Sen sijaan tällainen leikkisä animaatio on hyvä esimerkiksi lastentarhan tai huvipuiston web-sivuilla. (Mts. 47.)

Animaatioista saadaan aikaan ammattimaisen näköisiä, kun niitä ei toteuteta täysin lineaarisesti. Amatöörimäisesti tehdyissä animaatioissa useasti laitetaan kaikki efektit tapahtumaan samanaikaisesti ja lineaarisesti. Esimerkiksi animoitaessa elementin saapuminen näyttöön ruudun ulkopuolelta, laitetaan elementin näkyvyyden lisääntyminen ja liike alkamaan ja loppumaan samanaikaisesti. Ihmiset pitävät luonnostaan vaihtelevuudesta ja kontrasteista elementtien väleillä, joten tätä voidaan hyödyntää animaatioita suunniteltaessa. Jos elementin näkyvyyttä lisätään ensin näkymättömästä 50 %:iin ja vasta sen jälkeen aloitetaan elementin liike, saadaan lopputulokseen vaihtelevuutta ja ammattimainen silaus. (Mts. 48.)

## 5 MILLAINEN ON HYVÄ ANIMAATIO?

Hyvän animointi kokonaisuuden aikaan saaminen web-sivuille on prosessi. Animaatioiden testailua ja kokeilua joudutaan suorittamaan, jotta animaatioihin saataisiin yhtenäinen tuntu. Yhtenäisyyden tuntuun vaikuttaa animaatioiden tyyli, mutta myös animoinnin pituus ja ajoitus. (Shapiro 2015, 9.) Väreillä on myös oma osansa yhtenäisyyden luomisessa (Hatva 2003, 63).

### 5.1 Värit ja niiden vaikutukset

Väreillä ja erilaisilla väriskaaloilla voidaan ilmaista monenlaisia asioita kuten esimerkiksi tunnelmaa tai ilmaista kontekstia. Väreillä on myös suora vaikutus siihen, miten animaatio koetaan. Värivalintojen onnistuessa kokonaisuus on miellyttävä ja tuo lisäarvoa sisällölle, mutta jos taas värivalinnoissa epäonnistutaan, voi animaation katselu pahimmassa tapauksessa olla niin ärsyttävää, että se karkottaa käyttäjän kokonaan pois sivuilta. Tästä syystä värivalintoja kannattaa harkita huolella, että värien käytöstä saataisiin visuaalisesti miellyttävää ja kokonaisuutta tukevaa eikä rasitetta sisällölle. (Hatva 2003, 63, 74.)

Värejä valittaessa tulee myös pitää mielessä, että osa väestöstä on värisokeita eivätkä siksi pysty erottamaan pieniä kontrastieroja. Siksi erityisesti tärkeiden objektien tulisi erottua selkeästi taustasta värikontrastieroina. Mutta jos värikontrastierosta tulee hyvin suuri, siitä syntyy helposti häiritsevä elementti, sillä ihmissilmät väsyvät katsoessaan paljon suuria kontrastieroja. Värien välillä oleva kontrasti kannattaa siis tarkistaa ja tämän voi tehdä ja kokeilla helpoiten muuntamalla värit harmaasävyiksi. Tällöin värien kontrastierojen erottuvuuskynnystä voidaan arvioida parhaiten. Kontrastia voidaan luoda värien välille mm. kylläisyydellä, värisävyllä ja kirkkaudella. Näitä kaikkia kannattaa käyttää hyödyksi muutenkin värejä valittaessa. (Mts. 67–68.)

Värisävyjen miellyttävyydestä on tehty jonkin verran tutkimuksia. Niissä on selvinnyt, että viileät värit kuten sininen, vihreä, violetti ja sini-vihreä ovat katsojan silmään miellyttävimpiä. Kuitenkin tutkimuksien mukaan naiset pitävät miehiä enemmän lämpimistä väreistä. Tärkeintä värien valinnassa on kuitenkin asiayhteys ja väriyhdistelmät joita käytetään. Yleisesti ottaen värillinen koetaan aina miellyttävämmäksi kuin musta-valkoinen, mutta siitä huolimatta värien käytössä ei kannata mennä liian pitkälle tai käyttää pelkästään voimakkaista väreistä koostuvaa kokonaisuutta. (Mts. 74–75.) Väriyhdistelmien ja väriharmonioiden aikaan saamiseksi kannattaa käyttää apuna webistä löytyviä väriharmonia-generaattoreita, jos kokee, että oma värisilmä ei ole tarpeeksi kehittynyt löytämään hyviä yhdistelmiä.

Animaatiossa esiintyvien tekstuurien ja pienten yksityiskohtien kohdalla on sama ongelma kuin kontrastieroissa. Heikon näkökyvyn omaavan henkilön on vaikea erottaa niitä. On tutkittu, että vaaleuserot ovat parempia erottuvuudeltaan kuin värierot, joten

animaatiossa kannattaa hyödyntää tästä syystä enemmän valon ja varjon vaikutelmaa kuin eri väreillä tehtyä erottelua. (Mts. 71.)

## 5.2 Käytettävyys

Kun animointia lisätään web-sivuille, on tärkeää miettiä miten ne vaikuttavat sivujen käytettävyyteen. Liiallinen vilinä sivuilla ei ainoastaan rasita tietokoneen prosessoria, vaan se voi tehdä sivuista myös sekavat ja luotaan työntävät. Siksi animointia tulee käyttää harkitusti ja maltillisesti. Animointi on silloin aiheellista ja tarpeellista, kun sillä johdatellaan ihmistä sivujen käytössä tai korostetaan merkittävimpiä elementtejä sivuilla. Interaktiivisuus ja efektit ovat kuitenkin toissijaisia käytettävyyden rinnalla. Efektit eivät saa hidastaa sivuja. Mainoskäytössä tässä voidaan kuitenkin tehdä hiukan kompromisseja, jos sivuille on lisättävä korkealaatuista videomuotoista sisältöä. (Lepola 2015; Voutilainen 2015.)

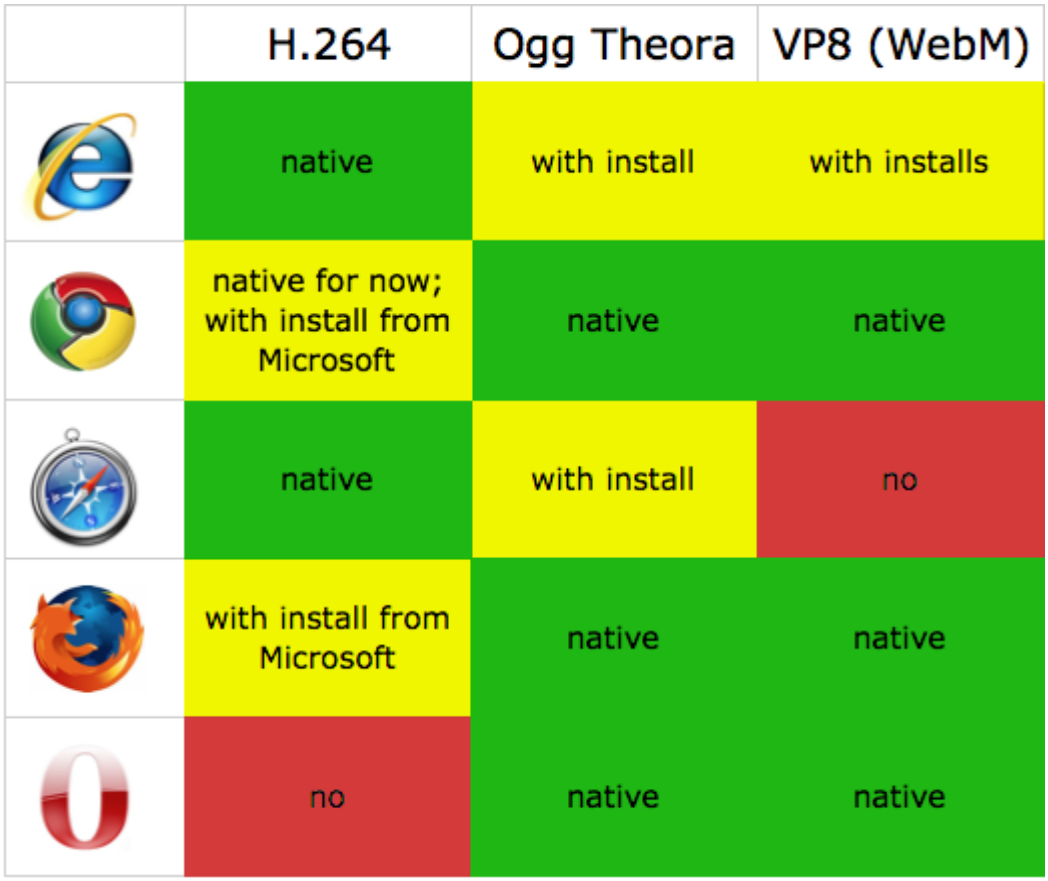
Käytettäessä erilaisia tekniikoita animoinnin aikaan saamiseksi on hyvä miettiä, miten nämä eri tekniikat eroavat käytettävyydeltään ja suorituskyvyltään. Nykyisin pääasiallisesti sovelluskehityksessä käytetään HTML5- ja CSS3-kieliä, joita tarvittaessa hiukan tuetaan JavaScriptia apuna käyttäen. Yleinen käsitys nykyisin on se, että HTML5 yhdessä CSS3:n kanssa on kevyin tapa tehdä animaatioita. (Lepola 2015; Kallio 2015.) Kuitenkin testit ovat todistaneet sen, että JavaScript on mainettaan huomattavasti parempi animaatioiden luomisessa (GreenSock 2015).

Animaatiot jo sinällään ovat sivuja hidastavia elementtejä, johtuen niiden intensiivisestä resurssien käytöstä. Kuitenkaan tästä syystä ei kannata luopua animaatioiden käytöstä täysin, sillä animaatioiden toimintaa voidaan keventää muutamalla keinolla. JavaScript-animaatioiden suurimman ongelman muodostaa ehkä jatkuvien vuorottaisten set- ja get-toimintojen suorittaminen setInterval()- tai setTimeout()-toimintojen sisällä. Animaatioiden suorituksessa tähdätään 60 frame / sek suoritukseen, joka tarkoittaa sitä, että selaimen täytyy ladata yksi animaation frame 16,7 ms. Kun selain suorittaa näin vuorotellen set- ja get-toimintoja kaiken muun ohella, ylittyy helposti tuo 16,7 ms ja animaatio nykii tai jumiutuu täysin. Tämä voidaan ehkäistä siirtämällä kaikki get-toiminnot ja set-toiminnot yhteen niin, että niitä ei suoriteta vuorotellen.

Näin selain voi suorittaa ensin kaikki get-toiminnot jonka jälkeen set-toiminnot, jolloin suoritus nopeutuu. (Shapiro 2015, 118–145.)

Animaatioita voidaan myös kiihdyttää grafiikkaprosessorin (GPU) avulla, joka nykyisin löytyy myös älypuhelimista. Grafiikkaprosessori suorittaa automaattisesti raskaita animaatioita kuten CSS3-transitions ja 3D transform-animaatiot, Canvas-animaatiot ja WebGL 3D-animaatiot. Tätä ominaisuutta voidaan myös käyttää hyväksi lisäämällä koodiin 3D-animoitu määre, joka pakottaa selaimen siirtämään animaation suorituksen grafiikkaprosessorille. Näitä kiihdytettyjä määreitä ovat CSS3-kielessä mm. translateZ ja translate3D. JavaScriptissa vastaavat määreet ovat translate3d() ja matrix3d(). TranslateZ voi aiheuttaa nykimistä Chrome- ja Safari-selaimissa, mutta tämä saadaan korjattua lisäämällä ”backface-visibility: hidden;” ja ”perspective: 1000;” -määreet. 3D-määreen ei tarvitse tehdä mitään, kunhan se vain on koodissa. Näitä 3D-määreitä eivät kuitenkaan kaikki selaimet tue, joten toiminnolle joudutaan ohjelmoimaan myös poikkeustapaus. Tämä myös tarkoittaa sitä, että näillä selaimilla ei grafiikkaprosessori kiihdytystä tapahdu. (Matyus 2013; Hernandez 2012; CSS-Tricks 2014.)

Kun sivuille päädytään lisäämään videomuotoisia animaatioita, on muistettava, että kaikki selaimet eivät tue kaikkia videopakkausmuotoja (kuva 4). Tästä syystä sama animaatio kannattaa lisätä ainakin kahdessa eri muodossa, jolloin saadaan parempi tuki sisällölle. Videomuotoista sisältöä käytetään nykyisin melkein ainoastaan, jos sivuille tarvitsee lisätä tutoriaaleja (Lepola 2015; Voutila 2015).




**KUVA 4. Selaimien tuki eri videoformaateille (Whitney 2015)**

Nykypäivänä mobiililaitteiden käyttö webin selailussa on niin arkipäiväistä, että tämä tulisi ottaa hyvin huomioon websivuja suunniteltaessa. Animaatioita tehdessä on hyvä myös pitää mielessä, miten käytetty tekniikka reagoi resoluution pienentyessä tai suurentuessa ja miten se vaikuttaa sisältöön kokonaisuutena. Samat animaatiot eivät välttämättä ole järkeviä käyttää sekä tietokoneelta että mobiililaitteelta katseltaessa, sillä mobiililaitteella käytössä oleva tila on paljon pienempi. Kaikki CSS3-animointimääreet eivät myöskään toimi samalla tavalla mobiililaitteilla kuin tietokoneella (Juvonen 2015; Talus 2015; Leppänen 2015).

Ohjelmoimalla tehtyjen animaatioiden kokoa voidaan muuttaa esimerkiksi määrittelemällä prosentuaalisesti, kuinka suuri animaatio on näytön pinta-alasta. Kuitenkin näin tehdessä, joudutaan joko puristamaan suuri animaatio pieneen tilaan mobiililaitteella tai tyytymään huonompi laatuiseen animaatioon isolla näytöllä. Tästä syystä skaalautuvasta vektorigrafiikasta tehdyt animaatiot kuten SVG-animaatiot, toimivat skaalautuviksi suunnitelluissa websivuissa parhaiten.

### 5.3 Animoitujen elementtien käyttäminen mainonnassa

Websivut ovat yleensä täynnä erilaisia liikkuvia, välkkyviä tai vilkkuvia mainoksia, jotka vaativat katsojan huomiota. Tämä sekä animaatioiden sijainti, voivat häiritä katselijaa niin paljon, että hän ei pysty keskittymään sivujen varsinaiseen sisältöön. Animoitujen mainosten vaikutuksista katselijaan on tehty monenlaisia tutkimuksia. Tutkimustulokset viittaavat siihen, että animoituja mainoksia klikataan useammin kuin staattisia. Animoidut mainokset saavat myös toisten tutkimusten mukaan enemmän huomiota katselijalta kuin staattiset. Toiset tutkimukset taas viittaavat siihen, että animoitu sisältö ei saa katselijalta huomiota varsinkaan silloin, kun hän suorittaa sivuilta tiedon hakua. (Kuisma ym. 2010, 271–272.)

Tutkimuksissa on todettu, että sivujen yläreunaan asetetut bannerimainokset ovat katselijan helpompia sivuttaa kuin sivujen reunassa olevat isot, pitkät ja kapeat mainokset. Ihmiset ovat tottuneet siihen, että mainokset on yleensä asetettu sivujen yläreunaan, mikä saa heidät sivuttamaan ne niihin suurempaa huomiota kiinnittämättä. On myös todettu, että sivujen reunassa olevat mainokset saavat aikaan positiivisempia reaktioita katselijassa kuin bannerit tai pop-up mainokset. Tästä syystä voidaan päätellä, että sivujen reunat ovat parhaat ja suotuisimmat paikat mainoksille. (Mts. 272.)

On tutkittu, että kun sivuilla on paljon mainoksia ja vilinää, niin katselijan sensurointi kyky ylikuormittuu. Hän joutuu aktiivisesti välttelemään kiinnittämästä huomiota mainoksiin, ja tästä seurauksena hänen muistikykynsä heikkenee. (Mts. 273.) Tästä syystä mainoksia ja mainontaa ei kannata käyttää paljon tiedottamiseen tarkoitetuilla sivuilla. Viihdesivustoilla mainosten rajallinen käyttö voi taas olla hyväkin asia. Kuitenkin yleisellä tasolla mainoksia kannattaa lisätä harkitusti.

## 6 ANIMOINTI WEB-SIVUILLA

Tässä osiossa keskityn tarkastelemaan erilaisia efektejä ja mahdollisuuksia soveltaa animointia eri tilanteissa. Keskityn tekniikkoihin, jotka nousivat esille asiantuntijakyselyn kautta, sillä näitä tekniikoita käytetään tällä hetkellä yleisimmin.

Tein asiantuntijakyselyn sekä kaksi haastattelua opinnäytetyötäni varten. Yhteensä kyselyyn vastasi seitsemän asiantuntijaa. Sekä haastatteluissa että kyselyissä tuli esille hyvin samanlaisia ajatuksia ja näkemyksiä animoinnista web-sivuilla. Haastattelukysymyksiä voi tarkastella paremmin liitteistä (liite 2).

### 6.1 Asiantuntijanäkemyksiä

Asiantuntijahaastatteluiden kautta nousi esiin, että nykyaikaiset animaatiot tehdään pitkälti CSS3-, JavaScript- ja jQuery-kieliä käyttäen, mutta näistä suosituin on CSS3. Flash-animaatioita on käytetty jonkin verran banneri-muotoisissa animaatioissa, mutta niistä on luovuttu parempien tekniikoiden vuoksi. SVG-animaatioita käytetään myös koko ajan enenevissä määrin. (Juvonen 2015; Kallio 2015; Kärkkäinen 2015; Lepola 2015; Leppänen 2015; Suvimaa 2015; Talus 2015; Voutila 2015; Voutilainen 2015.)

Yleisesti tärkeinä asioina web-sivujen suunnittelussa pidetään sivujen keveyttä ja nopeaa latautumisaikaa. Sivujen latausaika vaikuttaa myös hakukoneiden kautta saataviin tuloksiin, joka on merkittävä etu sivujen löydettävyyden kannalta. Kuitenkin sivujen visuaalisuutta ja kiinnostavuutta on tärkeä pitää yllä, mutta mahdollisimman käyttäjäystävällisesti ja kevyesti. (Mt.)

Perus web-sivuille ei lisätä paljon animoitua sisältöä, koska niillä ihmiset vierailevat paljon ja liiallinen animointi voisi olla häiritsevää. Silloin kun tehdään erilaisia kampanja- tai mainos-sivuja, voidaan sivuista tehdä hyvinkin näyttäviä ja animoituja, sillä sivut eivät ole webissä vuodesta toiseen ja sivuilla ei käydä monia kertoja. Tällaisissa tapauksissa pitkät ja moninaiset animaatiot eivät kerkeä tulemaan käyttäjille rasitteeksi. (Kallio 2015.) Animaatioita käytetään pääasiassa ohjaavina ja informatiivisina elementteinä, jotka ohjaavat käyttäjää sivujen käytössä. Liikkuva kuva on tällä hetkellä suosiossa, mutta sen toteutukset ovat kohtalaisen raskaita kuten esimerkiksi JavaScriptilla toteutettava parallax-efekti, jossa tausta liikkuu eritahdissa sivun muihin elementteihin verrattuna. (Mt.)

Animaatioiden tulevaisuuden asema web-sivuilla nähdään kasvavan laitteiden ja tekniikan kehittyessä. Selaimet alkavat tukea paremmin visuaalisia elementtejä ja niiden käytöstä tulee yksinkertaisempaa, nopeaa ja sujuvaa. Animaatioita ei enää tarvitse

suunnitella eritavalla erilaitteille vaan samat animaatiot toimivat kaikissa laitteissa. Liikkuva kuva on tulevaisuutta. (mt.) SVG-animaatioiden käyttö tulee todennäköisesti lisääntymään tulevaisuudessa niiden pienen tiedosto koon ja skaalautuvuuden vuoksi. Funktionaalisen koodauksen katsotaan myös lisääntyvän tulevaisuudessa ja olio-ohjelmoinnin jäävän pois (Voutila 2015).

## 6.2 Animaatioiden hyödyntäminen

Koska asiantuntijahaastatteluissa nousi esille animaatioissa pääasiallisesti käytettävän CSS3- ja JavaScript-animaatioita, keskityn nyt tarkastelemaan näitä tekniikoita lähemmin. Teen myös yhden esimerkin HTML5 canvas-elementistä, koska se on hyvin merkittävä väline monimutkaisempien animaatioiden toteuttamisessa. Vaikka haastatteluissa kerrottiin myös jQueryn olevan yksi suosituista animaatiomuodoista, en kuitenkaan tässä osiossa käsittele sitä sen hitauden vuoksi. Sen sijaan olen käyttänyt GreenSock-animointikirjastoa, joka on jQuerya huomattavasti suorituskykyisempi vaihtoehto. Käsittelen myös SVG-animaatioita niiden lisääntyvän käytön vuoksi, ja koska ne ovat erittäin hyvä ja helppo tapa tehdä pieniä animaatioita. Niiden merkitys tulevaisuudessa tulee varmasti myös lisääntymään animaatiotekniikoiden joukossa. Haastatteluissa tuotiin esille parallax-efekti, joka on yksi tämän hetken suosituimmista efekteistä web-sivuilla. Tarkastelen myös tätä efektiä parissa esimerkissä ja sen toteutusmahdollisuuksia.

### 6.2.1 Hover-animaatiot

Hover-efektit ovat hyvin yleinen animointikeino ohjaamaan käyttäjän toimintaa ja korostamaan elementtejä. Sillä saadaan helposti ilmaistua kohdat, joista esimerkiksi klikkaamalla tapahtuu jotain. Ensimmäiseksi vertaan, miten yksinkertainen animaatio kuten taustavärin vaihtaminen kursorin liikkuessa div-elementin päälle onnistuu CSS3 ja JavaScript-kielillä (kuva 5).

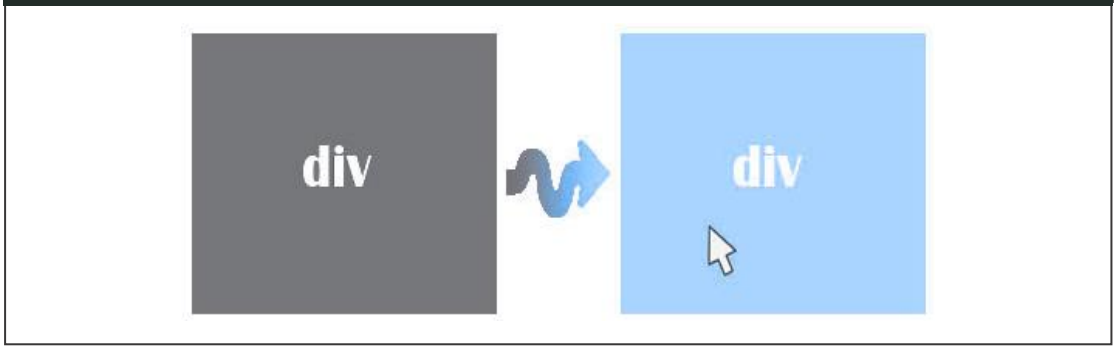
*CSS3:*

```
#div:hover{
    background-color: #a9d4ff;
}
```

*Javascript:*

```
var div = document.getElementById('div');

div.onmouseover = function(){
    this.style.backgroundColor = '#a9d4ff';
};
div.onmouseout = function(){
    this.style.backgroundColor = '#747779';
};
```




**KUVA 5. Hover-animaation esimerkkikuva**

Tässä esimerkissä kummallekin kielelle luodaan ensin HTML-pohjalle div, jolle määritellään perus taustaväriksi harmaa. Tämän jälkeen luodaan hover-efektille tapahtumat.

*CSS3:*

Tällaisissa yksinkertaisissa animaatioissa CSS3 on yksinkertaisin keino saada tuloksia nopeasti. Luodakseen tällaisen animaation tarvitsee vain määritellä hover-tapahtumaan värikoodi, joksi taustaväri halutaan muuttaa.

*JavaScript:*

JavaScriptilla ohjelmoitaessa taas sama joudutaan määrittelemään myös, mikä taustaväri on kun kursori siirtyy pois div:n päältä. JavaScriptissa ei ole samanlaista hover-

tapahtumaa kuin CSS3:ssa, joten rivillisesti koodia tarvitaan enemmän. JavaScript taas vastaavasti antaa paljon mahdollisuuksia animaation jatkokehitykselle.

### 6.2.2 Painikkeet

Painikkeet ovat ehkä yksi eniten animoiduista elementeistä web-sivuilla. Harvoin edes näkee sellaisia painikkeita, joita ei olisi jotenkin animoinnilla korostettu. Eikä ihme, sillä käyttäjän on hyvin tärkeää havainnoida se, että sivut havaitsevat hänen klikkauksensa.

Nämä yksinkertaiset painikkeet on tehty HTML- ja CSS3-kielillä. CSS3-animointiominaisuudet antavat paljon mahdollisuuksia tehdä esimerkiksi tällaisia helppoja ja yksinkertaisia efektejä (kuva 6).


**KUVA 6. CSS3-painikkeet**

Nämä painikkeet on tehty CSS3-transform translate-määrettä hyödyntäen. Painiketta siirretään sitä klikattaessa alareunan border-määreen verran alaspäin, jolloin saadaan aikaan efekti, että painike myös painuu alas sitä klikattaessa. Painikkeille on annettu hover-efekti, joka muuttaa painikkeen väriä hieman kursorin ollessa sen päällä. Paini-

ketta on käytetty sivukokonaisuudessa, jonka täydellinen koodi löytyy liitteistä (Liite 3).

### 6.2.3 Linkit

Linkit ja valikot ovat toinen elementti, joita animoidaan lähes poikkeuksetta. Tein esimerkin, miten linkkitekstin voi animoida. Tätä efektiä voidaan hyödyntää myös valikoiden teossa, jossa tarvitaan esimerkiksi lisäselitteitä linkeistä (kuva 7). Efekti on tehty HTML- ja CSS3-kielillä.

*CSS3:*

```
-webkit-transition: all .5s ease-out;
-moz-transition: all .5s ease-out;
transition: all .5s ease-out;
```

*HTML:*

```
<a href="" otsikko1="Natural" otsikko2="Cosmetics">LeafFLOW</a>
```


**KUVA 7. Linkin-animointi**

Animaatiossa on kaikki määreet määritelty animoiduiksi ja aikamääreeksi on määritelty ease-out, jolloin animaation suoritus hidastuu loppua kohden. Pienemmät otsikot on lisätty linkin sisään määreinä ja ne on sieltä noudettu CSS3-koodin puolelle. CSS3-koodissa nämä määreet on lisätty sisällöksi ja määritelty niille liikkeet. Linkki-efektiä on käytetty sivukokonaisuudessa, jonka täydellinen koodi löytyy liitteistä (Liite 3).

### 6.2.4 Kuvagalleria

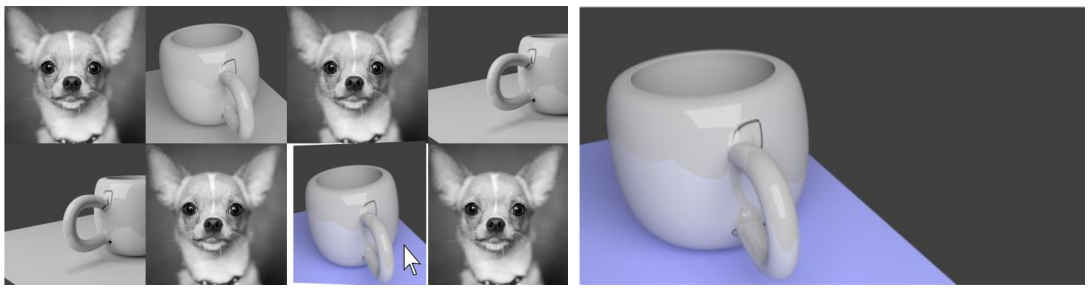
Kuvan flippaus on näyttävä efekti, jota voidaan hyödyntää jos kuvaa halutaan korostaa. Tällä efektillä saadaan esitettyä, mikä kuva on milläkin hetkellä hiiren alla klikattavissa. Se voidaan yhdistää esimerkiksi hover-efektiin, kuten olen tässä tehnyt (kuva 8). Flip-efekti on toteutettu CSS3- ja JavaScript-kielillä. Isomman kuvan avautumisefektiin on lisäksi käytetty avuksi GreenSock-animaatiokirjastoa.

*CSS3:*

```
transition: -webkit-transform .3s;
transition: transform .3s;
-webkit-transform: perspective( 1000px ) rotateY( 0deg );
transform: perspective( 1000px ) rotateY( 0deg );
-webkit-filter: grayscale(1);
filter: grayscale(1);
```

*GSAP:*

```
TweenMax.to(kehys, .3, {opacity:1, ease:Circ.easeOut});
TweenMax.to(kehys, 1, {scale:1.02});
```



**KUVA 8. Flip-kuvagalleria**

Kuvat ovat oletusarvoisesti harmaasävyisiä, joka saadaan aikaan asettamalla div:lle tai kuvalle filter-määre. Filter on tapa saada Photoshop tyylisiä efektejä aikaan koodipohjaisesti. Kun hiiri viedään kuvan päälle, kuvan filter-määre poistetaan kuvasta, jolloin kuvan värit tulevat esille. Animaatio saadaan aikaiseksi kääntämällä kuvan div-elementtiä perspektiivisesti y-akselinsa ympäri. Perspective-määreellä määritellään kuinka läheltä katsojaa kuva kääntyy. Määreen ollessa pienempi, kuvan etureuna venyy korkeammaksi ja takareuna lyhenee flipin aikana. RotateY määrittää taas akselin, jonka ympäri kuva kääntyy. Jos määre korvattaisiin rotateX-määreellä, kuva kääntyisi

horisontaalisesti ympäri. Transition-määreessä määritetään, kuinka kauan efektin suorittaminen kestää.

Klikkaamalla kuvaa, kuva aukeaa toiseen isompaan div:iin ja samalla pienemmät kuvat poistuvat näkyvistä. Kuvan toiminnollisuudet on ohjelmoitu JavaScriptilla ja isomman kuvan aukeamiseen on lisätty GreenSock-animaatio. Kuva skaalautuu hieman avattaessa ja sen näkyvyys lisääntyy pikku hiljalleen. Kuvagalleriaa on käytetty sivukokonaisuudessa, jonka täydellinen koodi löytyy liitteistä (Liite 3).

### 6.2.5 Kello

HTML5:den uusi canvas-ominaisuus antaa paljon mahdollisuuksia animaatioiden tekoon. Canvasilla onnistuu suurien koko sivun kokoisten animaatioiden teko kuin myös pienempien elementtien rakentaminen. Sillä voidaan piirtää erilaisia kuvioita tai hallita niitä. Näitä elementtejä pystytään canvasin avulla liikuttamaan frame eli kuva kerrallaan. Tässä esimerkissä olen rakentanut canvas-pohjalle kellon (kuva 9). Kelloa voidaan hyödyntää esimerkiksi kaikenlaisten aikavarausjärjestelmien yhteydessä. Tämän kellon ja kaiken canvasille piirretyn sovellus- ja muokkausmahdollisuudet ovat JavaScriptilla rajattomat.

```
var angle;
context.font = "25px arial";
context.textBaseline="middle";
context.textAlign="center";
for (var i=1;i<13;i++){
    angle = i * Math.PI / 6;
    context.rotate(angle);
    context.translate(0, -130);
    context.rotate(-angle);
    context.fillText(i.toString(),0,0);
    context.rotate(angle);
    context.translate(0, 130);
    context.rotate(-angle);
}
```


**KUVA 9. Canvas-kello**

Kellon numerotaulu saadaan aikaisiksi määrittämällä kehän koko, jonka ympärille numerot piirtyvät. Tämän jälkeen numeroiden välinen kulma saadaan jakamalla π

kuudella ja kertomalla tämä kyseessä olevalla tuntimäärällä. Näin jokainen numero tulee tasaisin välein, koska laskentalähtöpiste on aina sama. Katso kellon täydellinen koodi liitteistä (liite 4).

### 6.2.6 Parallax-efekti

Parallax on tällä hetkellä suosittu efekti web-sivuilla. Efektillä saadaan aikaisiksi perspektiiviä kuvaan, joka muuttuu sivua vierittäessä. Parallax-efekti voidaan yksinkertaisimmillaan tehdä puhtaasti JavaScriptia käyttäen tai käyttää tähän tarkoitukseen tehtyjä animointikirjastoja. Näitä ovat mm. Scrollr ja ScrollMagic. Tämä esimerkki on kuitenkin tehty pelkillä JavaScript-, CSS3- ja HTML5-kielillä (kuva 10).


**KUVA 10. Parallax-efekti**

Sivua vierittäessä kaksi lähimmäistä lammasta liikkuvat eri tahdissa taustaan nähden, jolloin saadaan aikaisiksi illuusio perspektiivista. Tämä on tehty JavaScriptilla kertomalla sivun vieritysetäisyys yläreunasta pienemmällä luvulla. Jokaisen kuvan kohdalla tämä on tehty eri kertoimella, jotta kuvat liikkuisivat eritahdissa. Sitten tämä arvo lisätään vain kuvan tyyliin etäisyysarvoksi yläreunasta. Kuvien asemointi taustaan nähden on ”relative”. Sivun header on ”fixed” asemoinniltaan ja ”z-indeksi” on muita korkeammalla, joten kuvat liukuvat sivuja vierittäessä sen alle.

### 6.2.7 SVG-animoitu logo

SVG:llä voidaan tehdä helposti monenlaisia 2D-animaatioita. Tässä esimerkissä olen kokeillut logon tekemistä animoidusti. Piirsin ensin logoon tulevat lehti-elementit Adobe Illustrator-ohjelmalla, josta tallensin logon suoraan SVG-muotoon. Tämän jälkeen muokkasin koodia tekstinkäsittelyohjelmalla ja lisäsin elementtien animoinnin. Animaatio koostuu kolmesta liukuvärjätystä lehdestä, jotka tulevat pikku hiljalleen valkoisesta taustasta esiin. Lehdet liikkuvat ja pyörivät hiukan, jotta lehdistä muodostuu viuhkamainen muodostelma. Lehtien alle paljastuu myös logoon kuuluva teksti (kuva 11). Animaatio on toteutettu pelkällä SVG-kielellä.

```
<animateTransform attributeType="XML"
attributeName="transform"
type="rotate"
from="0 55 115"
to="70 55 115"
dur="4s"
fill="freeze"
repeatCount="1"/>
```

```
<animate attributeName="opacity"
id="anim1" from="0" to="0.1"
dur="3s" fill="freeze"/>

<animate attributeName="opacity"
from="0.1" to="0.8" begin="anim1.end"
dur="3s" fill="freeze"/>
```

```
<linearGradient id="gradient" gradientUnits="userSpaceOnUse"
    x1="62.1118" y1="85.1157" x2="116.3414" y2="30.8862">
    <stop  offset="0.073" style="stop-color:#30B562"/>
    <stop  offset="0.3146" style="stop-color:#7AC687"/>
    <stop  offset="0.6404" style="stop-color:#91CA6C"/>
    <stop  offset="0.8708" style="stop-color:#B5D334"/>
</linearGradient>
```


**KUVA 11. SVG:llä animoitu logo**

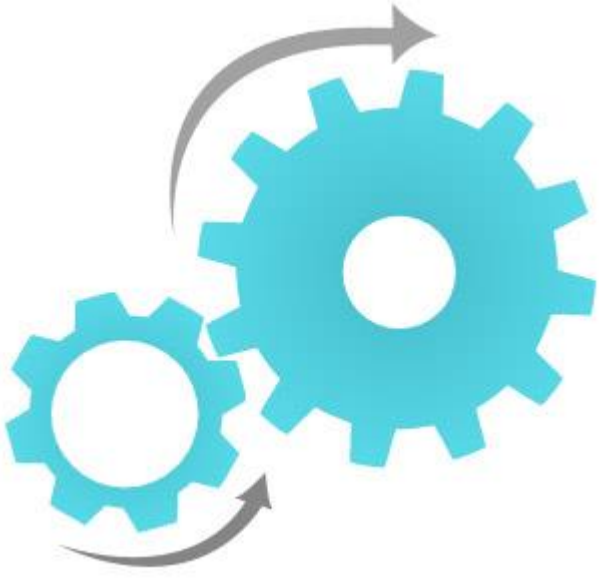
Lehtien liikkuminen tapahtuu yksinkertaisesti määrittelemällä animateTransform-määreellä liikkeen tyyli ja kulma, johon elementti kääntyy sekä koordinaateilla määritelty liikkeen keskipiste. Elementtiin määritellään myös kuinka pitkään liikkeen suorittaminen kestää, liikkeen toistojen määrä ja mitä liikkeen jälkeen tapahtuu.

Lehtien näkyvyyttä olen animoinut määrittelemällä opacitya animate-määreessä. Animate-määreellä voidaan animoida melkein mitä vain elementissä olevaa määrettä. Määreiden arvoja voidaan näin manipuloida, ja animate-määreitä voidaan myös jaksottaa suorittumaan toistensa jälkeen begin-komennolla, kuten olen tässä animaatiossa tehnyt.

Lehtien liukuväri saadaan aikaiseksi määrittelemällä pisteet joiden välillä värit vaihtuvat. Tämän jälkeen määritellään eri värit ja niiden vaihtumispisteet. Tätä SVG-logoa on käytetty sivukokonaisuudessa jonka täydellinen koodi löytyy liitteistä (liite 3).

### 6.2.8 SVG-animoitu lataus-ikoni

Jos haluaa saada web-sivuille persoonallisuutta, siihen loppusilauksen antaa oma kustomoitu latausanimaatio. Nämä pyörivät rattaat saadaan helposti toteutettua piirtämällä ensin rattaat piirustusohjelmalla ja tallentamalla ne sitten SVG-muotoon (kuva 12). Animaatio on toteutettu SVG-kielellä ja lisätty HTML-pohjaan.



**KUVA 12. SVG-latausikoni**

Kumpikin ratas on oma tiedostonsa ja ne ovat asemoitu niin, että ratakset menevät hieman päällekkäin. Ratakset on upotettu ja lisätty objekteina HTML-pohjaan (kuva 13).

```
<div style="padding:15%;">
    <div style="width:200px; margin:0 auto;">

        <object id="ratas1">
            <embed src="ratas2.svg" width="100px" height="100px">
        </object>
        <object id="ratas2">
            <embed src="ratas.svg" width="180px" height="180px">
        </object>
    </div>
</div>
```

```
<animateTransform attributeType="XML"
attributeName="transform"
type="rotate"
from="0 85 85"
to="360 85 85"
dur="7s"
repeatCount="indefinite"/>
```

**KUVA 13. Latausikonin koodi**

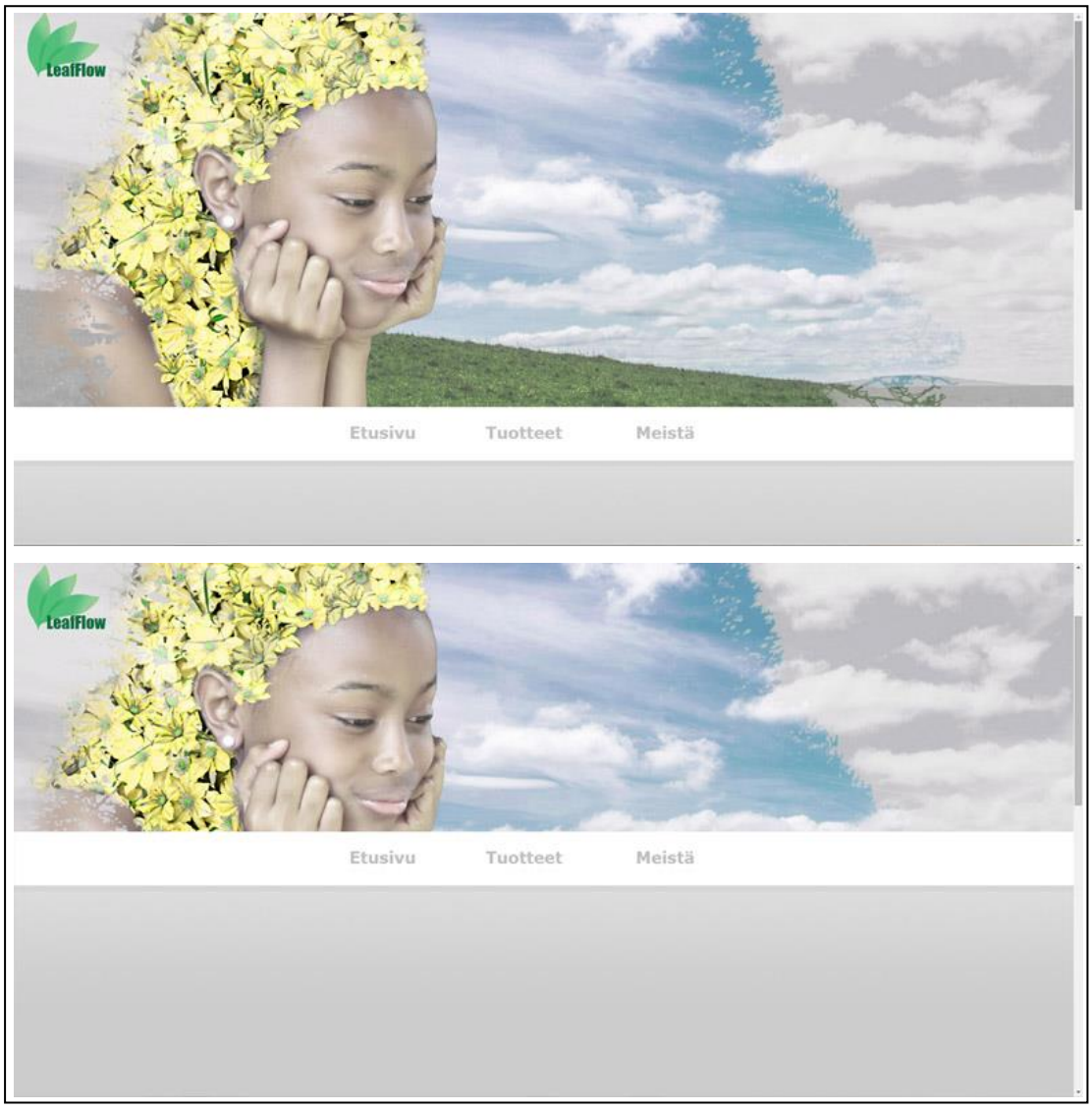
Rattaiden pyöriväliike saadaan aikaan samoin kuin edellisessä esimerkissä, nyt vain liike on ohjelmoitu jatkumaan ikuisesti (repeatCount=”indefinite”). Latausikoni voidaan ottaa käyttöön hyväksikäyttämällä esimerkiksi JavaScriptia. Latausanimaation täydellinen koodi löytyy liitteistä (liite 5).

### 6.2.9 Animoitu sivukokonaisuus

Olen tässä osiossa kuvannut sitä, miten ja millaisissa tilanteissa animaatioita voidaan hyödyntää. Sivukokonaisuus esimerkkinä tein yksisivuisen websivun, jossa olen hyödyntänyt aikaisemmin tekemiäni esimerkkejä. Sivu on kuvitteellisen kosmetiikka-alan yrityksen websivun mockup. Sivun tarkoitus on havainnollistaa animaatioiden käyttöä ja toimintaa käytännössä.

Sivun parallax-efekti on tehty ScrollMagic-animointikirjastolla. ScrollMagic oli vertailussa Scrollr-animointikirjastoa parempi siinä, että sitä voidaan hyödyntää moninai-

semmin. ScrollMagic on JavaScript-pohjainen ja siksi vaatii jonkin verran kielen tuntemusta. Se tukee mm. täysin jQuery-, GreenSock- ja Velocity-animontikirjastoja, mutta ei toimiakseen tarvitse välttämättä mitään näistä. Kirjastolla voidaan myös manipuloida pelkkää CSS3-koodia. Myös yksi mielestäni tärkeä ominaisuus ScrollMagic-animointikirjastossa on se, että animaatioita voidaan laukaista eri selausvaiheissa ilman, että ne ovat sidottuja sivujen vieritykseen. Animaatiossa tausta liikkuu perspektiivisesti hitaammin kuin etualalla oleva kuva (kuva 14). Sivun ylälaidassa on myös aiemmassa esimerkissäni tehty SVG-logo (kuva 11).




**KUVA 14. ScrollMagic parallax-efekti**

Tässä seuraavassa kuvassa näkyy vielä tarkemmin ScrollMagic-syntaksia (kuva 15). Sivuille luodaan ScrollMagic-controlleri, jolle eri animointielementit syötetään kohtauksina (scene). Mielestäni ScrollMagic-syntaksi on suhteellisen helppo oppia ja sillä voidaan tehdä hyvin moninaisia efektejä.

```
var controller = new ScrollMagic.Controller();

var scene = new ScrollMagic.Scene({duration: 800})
        .setTween('#girl', 1.5, {y: "70%", ease: Linear.easeNone})
        .addTo(controller);

var scene2 = new ScrollMagic.Scene({duration: 800})
        .setTween('#back', 1.5, {y: "90%", ease: Linear.easeNone})
        .addTo(controller);

var scene3 = new ScrollMagic.Scene({duration: 800})
        .setTween('#logo', 1.5, {y: "10%", ease: Linear.easeNone})
        .addTo(controller);
```

**KUVA 15. ScrollMagic-syntaksi**

Sivun valikko on tehty puhtaasti CSS3-ominaisuuksilla. Isot tekstit ovat linkkejä, joihin on liitetty pienemmät tekstit määreinä. CSS3-koodin puolella nämä määreet on noudettu ja animoitu transition-määreellä. Ylä- ja alapuolella olevat tekstit tulevat isomman tekstin takaa esiin hover-efektinä. Samalla isompaa tekstiä skaalataan hiukan suuremmaksi ja sen väriä vaihdetaan (kuva 16).


**KUVA 16. Animoitu linkki**

Sivun tuoteosio on tehty samalla kuvagallerialla kuin aikaisempi esimerkki (kuva 8). Pienet tuotekuvat kääntyvät hieman perspektiivisesti Y-akselin ympäri hover-efektinä. Kuvista poistuu myös samalla harmaasävy-filteri, jolloin kuvat näkyvät värillisenä (kuva 17).


**KUVA 17. Tuotegalleria**

Tuotteet-teksti on tehty SVG-animaationa, joka sisältää vain tekstin polun. Polku on animoitu CSS3-animation ominaisuudella (kuva 18).

```
#Layer1{
    stroke-dasharray:3100;
    stroke-dashoffset:0;
    -webkit-animation: viiva 4s linear;
    -o-animation: viiva 4s linear;
    -moz-animation: viiva 4s linear;
    animation: viiva 4s linear;
}

    @-webkit-keyframes viiva {
        from{
            stroke-dashoffset: 3100;
        }
        to{
            stroke-dashoffset: 0;
        }
    }
```


**KUVA 18. Viiva-animaatio SVG:llä**

Animaatio on nimetty viiva-animaatioksi ja sen kestoksi on määritelty 4s. Stroke-dasharray määrittää animoitavan viivan pituuden ja stroke-dashoffset sen alkupisteen polulla. Animaation ajoitus on määritelty lineaariseksi, jolloin animaatio suoritetaan alusta loppuun tasaiseen tahtiin. Animaatio sisältää kaksi kerrosta, joista toisessa on itse teksti ja toisessa t-kirjainten viivat. Näin kaksi kerrosta voidaan animoida suoritettavaksi eriaikaisesti. Muuten kerrokset ovat animoitu samalla tavalla.

Painike-animaatiota olen hyödyntänyt tässä yhteydenottokentän yhteydessä. Painike on tehty samalla tavalla kuin aikaisemmassa esimerkissä (kuva 7) mutta sen väri ja asemointi on erilaiset (kuva 19).


**KUVA 19. Painikeanimaatio**

Tämän sivukokonaisuuden ja kaikkien siihen liittyvien esimerkkien täydellinen koodi löytyy liitteistä (liite 3).

## 7 PÄÄTÄNTÖ

Sain vastattua tutkimuskysymykseen nykyaikaisimmista animointitekniikoista asiantuntijakyselystä saamieni tietojen pohjalta. Kyselystä sain reaaliaikaista tietoa animoinnin käytöstä kentällä, ja koska asiantuntijoiden näkemykset olivat hyvin saman-

suuntaisia, voidaan olettaa, että myös niiden kautta saamani tiedot ovat luotettavia. Tutkimustulokset olisivat tietenkin luotettavampia, jos kyselyyn olisi vastannut useampia asiantuntijoita. Mielestäni kyselyn suorittaminen oli kuitenkin hyvin olennainen osa tutkimusta, sillä sitä kautta sain näyttöä ja tukea tutkimustuloksiini.

Animaatioiden visuaalisuutta käsittelin värien ja liikkeen suunnittelun kautta ja millaisia vaikutuksia niillä on. Tutkimustuloksiin hyödynsin monia eri lähteitä ja sain niistä kasattua mielestäni hyvin laaja-alaisesti tietoa. Aihetta ei ole kovin paljon tutkittu ja siksi lähteitä aiheesta löytyi vähän.

Työni kokonaisuudessaan antaa hyödyllisiä tietoja animaatioista ja niiden suunnittelusta web-kehittäjille. Tällaista tutkimusta tästä näkökulmasta ei ole aikaisemmin opinnäytetyössä tehty, joten työni on tässä mielessä ainutlaatuinen ja lisäsi ammattialan tietovarantoa. Animaatioiden suunnitteluun harvoin paneudutaan tarpeeksi ja siksi siinä saatetaan epäonnistua. Työni antaa kattavasti tietoa, jota hyödyntämällä saadaan aikaan hyviä tuloksia.

Opinnäytetyöprosessi onnistui suhteellisen hyvin. Sain mielestäni tutkittua aihetta kohtuullisen monipuolisesti. Eniten ongelmia opinnäytetyön teossa aiheutti lähteiden vähyys. Suuriosa ajasta menikin siksi niiden etsimiseen. Työtä tehdessäni minun olisi kannattanut jättää enemmän aikaa käytännönosuudelle ja tehdä siitä laaja-alaisempi.

Aihealue on hyvin laaja ja siitä olisi syntynyt hyvinkin suuri tutkimus. Koin, että aiheeseen voisi porautua vieläkin syvemmälle ja aiheessa riittää vielä tutkimista. Aihetta voisi käsitellä myös tekniikoiden käytännönvertailun kannalta ja animointitekniikoita voisi käsitellä syvällisemminkin. Tekniikoiden ja selaintuen kehittyessä käytettävien animointimahdollisuuksien määrä kasvaa, joka muuttaa alaa kokoajan. Tästä syystä käytännön menetelmät muuttuvat alati ja niiden soveltamisessa riittää tutkimista.

Opin tätä opinnäytetyötä tehdessäni, että animaatioiden suunnittelussa täytyy huomioida monia asioita. Käytettävyys on yksi merkittävä osa animaatioiden suunnittelua. Animaatioilla voidaan helposti pilata käyttäjäkokemus liiallisella vilinällä tai niistä johtuvalla suorituskyvyn laskulla. Animaatioiden käytössä täytyykin tästä syystä pitää käytettävyys etusijalla. Jotkin animointityylit kuormittavat selainta enemmän kuin toiset ja tekevät näin sivujen käytöstä raskasta sivujen tahmean toiminnan vuoksi.

Myös animaatioiden asettelu ja värivalinnat ovat tärkeitä tekijöitä. Animaatioita tehdessä on myös pidettävä mielessä, että animointimahdollisuudet ovat paljon kehittyneemmät kuin selaimien tuki. Vanhemmat selaimet eivät tue uusimpia animointimääreitä.

Ohjelmoimalla voi animaatioista saada hyvin monenlaisia. Animaatiotekniikoiden hallitseminen tosin vaatii hiukan opettelua riippuen tekniikasta ja omasta ohjelmointitaustasta. Monipuolisimmat animaatiot saadaan aikaisiksi JavaScriptilla, mutta sen käyttö vaatii kielen tuntemusta. Moni animaatiokirjasto pohjautuu JavaScriptiin ja sen vuoksi web-kehittäjän olisi hyvä osata tätä kieltä. CSS3-animoinnillakin pääsee yllättävän pitkälle, mutta animaation muokkausmahdollisuudet ovat rajalliset. Jos animaatioita halutaan jaksottaa tai saman elementin eri muutoksia halutaan suorittaa eriaikaisesti, on avuksi otettava JavaScript. CSS3 on kuitenkin hyvä animointitekniikka silloin, jos animaatioista tehdään pieniä ja yksinkertaisia. SVG-animaatioilla saadaan taas luotua helposti monimutkaisiakin kuvioita, joita voi helposti animoida. Näiden animaatioiden syvällisempi manipulointi kuitenkin tarvitsee avukseen JavaScriptin, joka on toiminnollisuuksia sisältävän animoinnin perusta.

Animaatiokirjastoja kannattaa hyödyntää animointia tehdessä mahdollisuuksien mukaan. Ne sisältävät paljon valmiita animaatiotyylejä, joita saadaan hyödynnettyä vain muutamalla rivillä koodia. Näin saadaan nopeasti animaatioita aikaan ja säästytään paljolta ohjelmoimiselta.

Animoinnin merkitys tällä hetkellä web-kehityksessä on pääasiallisesti ohjaava. Sitä käytetään säästeliäästi ja asiakaslähtöisesti. Animointi voi myös olla keskeisessä roolissa sivuilla silloin, kun sivujen päätarkoitus on mainostaa jotakin. Sen tarkoitus websivuilla on kiinnittää käyttäjän huomio haluttuihin elementteihin ja luoda sivuista elävät, joita käyttäessään käyttäjä kokee sivujen olevan vuorovaikutuksessa itsensä kanssaan. Animaatioiden visuaalinen merkitys on suuri silloin, kun niitä käytetään järkevästi hyväksi.

**LÄHTEET**

About SVG. 2004. W3C. WWW-dokumentti. http://www.w3.org/Graphics/SVG/About.html. Päivitetty 29.10.2004. Luettu 31.07.2015.

Baker, Loren. 2006. 5 Reasons Not to Use Flash. WWW-dokumentti. http://www.searchenginejournal.com/5-reasons-not-to-use-flash/3949/. Päivitetty 27.10.2006. Luettu 20.6.2015.

Ball, Ryan. 2008. Oldest Animation Discovered In Iran. WWW-dokumentti. http://www.animationmagazine.net/features/oldest-animation-discovered-in-iran/. Päivitetty 12.3.2008. Luettu 5.8.2015.

Bruni, Ezequiel. 2015. The Ultimate Guide To Web Animation. WWW-dokumentti. http://www.webdesignerdepot.com/2015/05/the-ultimate-guide-to-web-animation/. Päivitetty 25.5.2015. Luettu 7.10.2015.

Conrad, David. 2011. CSS3 Rocks - Building a Custom CSS Button. WWW-dokumentti. http://www.i-programmer.info/programming/htmlcss/3292-css3-rocks-building-a-custom-css-button.html. Päivitetty 7.11.2011. Luettu 17.07.2015.

CSS animations performance: the untold story. 2015. GreenSock. WWW-dokumentti. http://greensock.com/css-performance. Päivitetty 5.1.2015. Luettu 24.08.2015.

Ferney, Thomas. 2013. HTML5 – Introduction. WWW-dokumentti. http://www.horizonduweb.com/html5-introduction/. Päivitetty 16.5.2013. Luettu 3.11.2015.

Flash. 2015. Computer Hope. WWW-dokumentti. http://www.computerhope.com/jargon/f/flash.htm. Päivitystietoa ei ole saatavilla.. Luettu 4.11.2015

Gerchev, Ivaylo. 2014. HTML5 Canvas Tutorial: An Introduction. Site Point. WWW-dokumentti. http://www.sitepoint.com/html5-canvas-tutorial-introduction/. Päivitetty 10.3.2014. Luettu 30.5.2015.
GIF (tm). 1987. CompuServe Incorporated. WWW-dokumentti. http://www.w3.org/Graphics/GIF/spec-gif87.txt. Päivitetty 15.6.1987. Luettu 5.8.2015.

Harris, Tom. 2015. How Web Animation Works. WWW-dokumentti. http://computer.howstuffworks.com/web-animation2.htm. Päivitystietoa ei saatavilla. Luettu 5.5.2015.

Harris, Tom. 2015. How Web Animation Works. WWW-dokumentti. http://computer.howstuffworks.com/web-animation3.htm. Päivitystietoa ei saatavilla. Luettu 15.7.2015.

Hatva, Anja. 2003. Verkko grafiikka.

Hernandez, Guil. 2012. Increase Your Site's Performance with Hardware-Accelerated CSS. WWW-dokumentti. http://blog.teamtreehouse.com/increase-your-sites-performance-with-hardware-accelerated-css. Päivitetty 10.12.2012. Luettu 3.11.2015.

HTML5 Introduction – What is HTML5 Capable of, Features, and Resources. 2015. 1st Web designer. WWW-dokumentti. http://www.1stwebdesigner.com/html5-introduction/. Päivitystietoa ei saatavilla. Luettu 31.5.2015.

Juvonen, Ville. 2015. Sähköposti 4.9.2015. Media designer. Mediataivas Oy.

Kallio, Sami. 2015. Sähköposti 28.8.2015. Web-suunnittelija. Haaja & Arwo Design Oy.

Kuisma, Jarmo. Simola, Jaana. Uusitalo, Liisa. Öörni, Anssi. 2010. The Effects of Animation and Format on the Perception and Memory of Online Advertising. Journal of Interactive Marketing. Volume 24. Issue 4, 269–282. PDF-dokumentti. http://www.sciencedirect.com.ezproxy.mikkeliamk.fi:2048/science/article/pii/S1094996810000447. Päivitetty 2010. Luettu 11.8.2015.

Kärkkäinen, Vesa. 2015. Sähköposti 3.9.2015. Toimitusjohtaja. Mainostoimisto Kixit Oy.

Lepola, Janne. 2015. Haastattelu 20.8.2015. Vanhempi ohjelmistokehittäjä. Saimaa Soft Oy.

Leppänen, Janne. 2015. Sähköposti 4.9.2015. Web developer. Mediataivas Oy.

Lewis, Paul. 2014. CSS vs JavaScript Animations. Google Developers. WWW-dokumentti. https://developers.google.com/web/fundamentals/look-and-feel/animations/css-vs-javascript. Päivitetty 19.9.2014. Luettu 22.3.2015.

Lilley, Chris. 2006. PNG (Portable Network Graphics). WWW-dokumentti. http://www.w3.org/Graphics/PNG/. Päivitetty 17.3.2006. Luettu 6.8.2015.

Matyus, Lehel. 2013. Improving HTML5 app performance with GPU accelerated CSS transitions. WWW-dokumentti. https://www.urbaninsight.com/2013/01/04/improving-html5-app-performance-gpu-accelerated-css-transitions. Päivitetty 4.1.2013. Luettu 3.11.2015.

McLaughlin, Dan. 2001. A rather incomplete but still fascinating history of animation. WWW-dokumentti. http://animation.filmtv.ucla.edu/NewSite/WebPages/Histories.html. Päivitetty 2001. Luettu 5.8.2015.

Myth Busting: CSS Animations vs. JavaScript. 2014. CSS-Tricks. WWW-dokumentti. https://css-tricks.com/myth-busting-css-animations-vs-javascript/. Päivitetty 13.1.2014. Luettu 3.11.2015

Pilgrim, Mark. 2015. Let's call it a draw(ing surface). WWW-dokumentti. http://diveintohtml5.info/canvas.html. Päivitystietoa ei saatavilla. Luettu 30.5.2015.

Roelofs, Greg 2015. Portable Network Graphics. WWW-dokumentti. http://www.libpng.org/pub/png/#history. Päivitetty 5.4.2015. Luettu 5.8.2015.

Shapiro, Julian. 2015. Web Animation using JavaScript.
Suvimaa, Jyrki. 2015. Sähköposti 4.9.2015. Toimitusjohtaja. Mainostoimisto Groteski Oy.
SVG Tutorial. 2015. W3Schools.com. WWW-dokumentti. http://www.w3schools.com/svg/. Päivitystietoa ei saatavilla. Luettu 31.7.2015.

Talus, Juho. 2015. Sähköposti 4.9.2015. Web developer. Mediataivas Oy.

Usage of Flash for websites. 2015. W3Techs. WWW-dokumentti. http://w3techs.com/technologies/details/cp-flash/all/all. Päivitystietoa ei ole saatavilla. Luettu 21.8.2015.

Waterhouse, Tom. 2011. The Guide To CSS Animation: Principles and Examples. WWW-dokumentti. http://www.smashingmagazine.com/2011/09/14/the-guide-to-css-animation-principles-and-examples/. Päivitetty 13.9.2011. Luettu 28.5.2015

Web Animation Infographics: A Map of the Best Animation Libraries for JavaScript and CSS3 plus Performance Tips. 2014. Awwwards. WWW-dokumentti. http://www.awwwards.com/web-animation-infographics-a-map-of-the-best-animation-libraries-for-javascript-and-css3-plus-performance-tips.html. Päivitetty 2014. Luettu 6.8.2015.

Whitney, Justin. 2015. How to Embed Video Using HTML5. WWW-dokumentti. http://www.htmlgoodies.com/html5/client/how-to-embed-video-using-html5.html#fbid=gZf7WCTHhpV. Päivitystietoa ei ole saatavilla.. Luettu 3.11.2015.

Wolejko, G. 2012. 5 reasons Why HTML5 matters? WWW-dokumentti. http://www.cognifide.com/blogs/ux/5-reasons-why-html5-matters/. Päivitetty 4.4.2012. Luettu 17.7.2015.

Voutila, Jukka. 2015. Haastattelu 21.8.2015. Toimitusjohtaja. Aika24 Oy.

Voutilainen, Miika. 2015. Sähköposti 4.9.2015. Web-asiantuntija. Idealmarkkinointi Oy.

**LIITE 1.**

Animointikirjastot

Awwwards 2014

**Asiantuntijoiden kyselykysymykset:**

**Millä kielillä olet animaatioita yleensä toteuttanut? Oletko kokenut tämän olevan paras vaihtoehto animaatioiden toteuttamiseen?**

**Millaisia nämä animaatiot ovat olleet ja mikä niiden tarkoitus sivuilla on ollut (esim. informatiivisuus, ohjaavuus, näyttävyys)?**

**Kuinka suuri asema liikkuvalla kuvalla tai interaktiivisuudella on ollut toteuttamillasi sivuilla?**

**Kumpi on sinulle tärkeämpi sivuja luodessa, niiden suorituskyky vai näyttävyys/visuaalisuus?**

**Oletko törmännyt sivujen mobiiliversioita ohjelmoitaessa ongelmiin animaatioiden kanssa?**

**Mitä asioita pidät tärkeinä websivujen animaatioissa?**

**Millaisia ohjelmointi taitoja vaaditaan animaatioiden toteuttamiseen?**

**Millaisena koet animoinnin aseman olevan tulevaisuudessa?**

**LIITE 3(1).**

**Sivukokonaisuuden koodi**

```
<DOCTYPE html>
<html>
<head>
<title>LeafFLOW-Cosmetics</title>
<meta http-equiv "Content-Type" content "text/html; charset="UTF-8 />
    <script src "http://cdnjs.cloudflare.com/ajax/libs/gsap/1.14.2/TweenMax.min.js"></script>
    <script src "https://cdnjs.cloudflare.com/ajax/libs/ScrollMagic/2.0.3/ScrollMagic.js"></script>
    <script src "https://cdnjs.cloudflare.com/ajax/libs/ScrollMagic/2.0.3/plugins/animation.gsap.js"></script>
    <script src "https://cdnjs.cloudflare.com/ajax/libs/ScrollMagic/2.0.3/plugins/debug.addIndicators.js"></script>

<style type "text/css">

*{
    margin: 0px;
    padding: 0px;
    font-family:verdana;
    color:#ccc;
}

body{
    height:100%;
    width:100%;
}

#back{
    position:absolute;
    top:0;
    width:100%;
    height:100%;
    background-image: url("girl_background.jpg");
    background-color:#cccccc;
    background-size: 100%;
    background-repeat: no-repeat;
    z-index: -2;
}

#girl{
    position:absolute;
    top:0;
    width:100%;
    height:100%;
    background-image: url("girl2.png");
    background-size: 100%;
    background-repeat: no-repeat;
    z-index: -1;
}

#logo{
    position:fixed;
    top:0;
    z-index: 2;
}

#tuotteet{
    margin-bottom:30px;
}

#header{
    position:relative;
    height:90%;
    overflow:hidden;
}

#container{
    height:150%;
    margin-top: -10%;
    background-image:url('bg.jpg');
    background-repeat: repeat-x;
    background-color:#cccccc;
    z-index: 1;
}

#content{
    width: 100%;
    height: 100%;
}

.st1{
    fill:#006E3A;
    font-size:30;
    font-family:'Impact';
}


.st0{
    fill:none;
    stroke:#ffffff;
    stroke-width:5;
}
```

**LIITE 3(2).**

**Sivukokonaisuuden koodi**

```
#Layer1{
    stroke-dasharray:3100;
    stroke-dashoffset:0;
    -webkit-animation: viiva 4s linear;
    -o-animation: viiva 4s linear;
    -moz-animation: viiva 4s linear;
    animation: viiva 4s linear;
}

    @-webkit-keyframes viiva {
        from{
            stroke-dashoffset: 3100;
        }
        to{
            stroke-dashoffset: 0;
        }
    }

#Layer2{
    stroke-dasharray:2000;
    stroke-dashoffset:0;
    -webkit-animation: viiva 6s linear;
    -o-animation: viiva 6s linear;
    -moz-animation: viiva 6s linear;
    animation: viiva 6s linear;
}

#galleria{
    padding-top:20%;
    text-align:center;
}

#rivi{
    width:800px;
    height:400px;
    margin:auto;
    opacity: 1;
}

.kuva{
    width:25%;
    height:50%;
    float:left;
    overflow: hidden;
    transition: -webkit-transform .3s;
    transition: transform .3s;
    -webkit-transform: perspective( 1000px ) rotateY( 0deg );
    -moz-transform: perspective( 1000px ) rotateY( 0deg );
    -o-transform: perspective( 1000px ) rotateY( 0deg );
    transform: perspective( 1000px ) rotateY( 0deg );
    -webkit-filter: grayscale(1);
    filter: grayscale(1);
    z-index:-1;
}

.kuva:hover{
    -webkit-transform: perspective( 1000px ) rotateY( -20deg );
    -moz-transform: perspective( 1000px ) rotateY( -20deg );
    -o-transform: perspective( 1000px ) rotateY( -20deg );
    transform: perspective( 1000px ) rotateY( -20deg );
    filter:;
    -webkit-filter: grayscale(0%);
}

.img{
    height:100%;
}

#kehys{
    position: absolute;
    left:25%;
    text-align: center;
    height:50%;
    width: 50%;
    opacity:0;
    z-index:-3;
}

#footer{
    background-color: #555;
    margin-top: -8%;
    width: 100%;
    height: 25%;
    min-height: 300px;
    padding-top: 5%;
    text-align:center;
}
```

**Sivukokonaisuuden koodi**

```
textarea {
    resize:;
}

a{
    width:100%;
    height:100%;
    border-radius:10px;
    text-decoration:none;
    color:#ffffff;
}

#button{
    height: 25px;
    width: 150px;
    padding: 15px;
    background-color: #28C158;
    cursor:pointer;
    border-radius:10px;
    border-bottom: 4px solid #249E4C;
    box-shadow: 3px 3px 3px #333;
    transition: background .1s linear;
}

#button:hover{
    background-color: #2FD767;
}

#button:active {
    box-shadow: 0px 0px 0px #777;
    border: 0px;
    transform: translateY(4px);
}

#menu{
    width:100%;
    height: 5vw;
    background-color: white;
    border-bottom: 10px solid #d5d5d5;
}

.menupainike{
    padding: 2%;
    float:left;
    height:100%;
    margin:4% auto;
    flex:1;
}

#menucontainer{
    width:40%;
    margin:auto;
    margin-top:2%;
    position:relative;
    display: -webkit-flex; /* Safari */
    display: flex;
}

.menupainike a {
    text-decoration:;
    font-size: 1.5vw;
    text-align:center;
    width: 20%;
    height: 10%;
    line-height: 0px;
    position: absolute;
    font-weight: 900;
    color: #bfbfbf;
    -webkit-transition: all .5s ease-out;
    -moz-transition: all .5s ease-out;
    transition: all .5s ease-out;
}

.menupainike a:before, a:after {
    color: #bfbfbf;
    font-weight: 200;
    display: block;
    -webkit-transition: inherit;
    -moz-transition: inherit;
    transition: inherit;
}

.menupainike a:before {
    content: attr(otsikko1);
    font-size:.15vw;
    text-align:center;
    opacity: .4;
}
```

**Sivukokonaisuuden koodi**

```
.menupainike a:after {
    content: attr(otsikko2);
    font-size:.15vw;
    text-align:center;
    opacity: .4;
}

.menupainike a:hover {
    color:#28C158;
    -webkit-transform: scale(1.1);
    -moz-transform: scale(1.1);
    -ms-transform: scale(1.1);
    -o-transform: scale(1.1);
    transform: scale(1.1);
}

.menupainike a:hover:before {
    -webkit-transform: scale(6) translate(-4px, -3px);
    -moz-transform: scale(6) translate(-4px, -3px);
    -ms-transform: scale(6) translate(-4px, -3px);
    -o-transform: scale(6) translate(-4px, -3px);
    transform: scale(6) translate(-4px, -3px);
}

.menupainike a:hover:after {
    -webkit-transform: scale(6) translate(4px, 3px);
    -moz-transform: scale(6) translate(4px, 3px);
    -ms-transform: scale(6) translate(4px, 3px);
    -o-transform: scale(6) translate(4px, 3px);
    transform: scale(6) translate(4px, 3px);
}

</style>
</head>
<body>
<div id "header">

    <div id "back"></div>
    <div id "girl"></div>

    <div id "logo">
        <svg  version="1.1" xmlns="&ns_svg;" xmlns:xlink="&ns_xlink;" width="140.773" height="116.628"
        viewBox="0 0 140.773 116.628" style="overflow:visible;enable-background:new 0 0 140.773 116.628;"
        xml:space="preserve">

        <g id="lehti1">
            <linearGradient id="gradient" gradientUnits="userSpaceOnUse" x1="62.1118" y1="85.1157"
            x2="116.3414" y2="30.8862">
                <stop  offset="0.073" style="stop-color:#308562"/>
                <stop  offset="0.3146" style="stop-color:#7AC687"/>
                <stop  offset="0.6404" style="stop-color:#91CA6C"/>
                <stop  offset="0.8708" style="stop-color:#B5D334"/>
            </linearGradient>

            <path style="fill:url(#gradient); opacity:0;" d="M45.916,82.757c-1.438,21.056,0.342,40.572-1.694,36.271
                c-4.499-9.497-11.027-16.147-17.627-24.067c-6.78-8.135-12.261-23.746-11.188-41.355C17.102,25.809,31.
                339,17.334,34.39,4.114
                c1.729-7.496,22.712,24.407,16.948,48.813C49.393,61.167,46.631,72.283,45.916,82.757z ">

                <animate id="anim1" attributeName="opacity" from="0" to="0.1"
                dur="3s" fill="freeze"/>

                <animate attributeName="opacity" from="0.1" to="0.8"
                begin="anim1.end" dur="3s" fill="freeze"/>

                <animateTransform attributeType="XML"
                attributeName="transform"
                type="rotate"
                from="0 55 115"
                to="70 55 115"
                dur="4s"
                fill="freeze"
                repeatCount="1"/>

                </path>
        </g>
```

## Sivukokonaisuuden koodi

```
        <g id="lehti2">
            <path style="fill:url(#gradient); opacity:0;" d="M45.916,82.757c-1.438,21.056,0.342,40.572-1.694,36.271
                c-4.499-9.497-11.027-16.147-17.627-24.067c-6.78-8.135-12.261-23.746-11.188-41.355C17.102,25.809,31.
                339,17.334,34.39,4.114
                c1.729-7.496,22.712,24.407,16.948,48.813C49.393,61.167,46.631,72.283,45.916,82.757z ">

                <animate id="anim1" attributeName="opacity" from="0" to="0.1"
                dur="2s" fill="freeze"/>

                <animate attributeName="opacity" from="0.1" to="0.8"
                begin="anim1.end" dur="2s" fill="freeze"/>

                <animateTransform attributeType="XML"
                attributeName="transform"
                type="rotate"
                from="0 55 115"
                to="40 55 115"
                dur="3s"
                fill="freeze"
                repeatCount="1"/>

                </path>
        </g>
        <g id="lehti3">
            <path style="fill:url(#gradient); opacity:0;" d="M45.916,82.757c-1.438,21.056,0.342,40.572-1.694,36.271
                c-4.499-9.497-11.027-16.147-17.627-24.067c-6.78-8.135-12.261-23.746-11.188-41.355C17.102,25.809,31.
                339,17.334,34.39,4.114
                c1.729-7.496,22.712,24.407,16.948,48.813C49.393,61.167,46.631,72.283,45.916,82.757z">

                <animate id="anim1" attributeName="opacity" from="0" to="0.2"
                dur="1s" fill="freeze"/>

                <animate attributeName="opacity" from="0.2" to="0.8"
                begin="anim1.end" dur="2s" fill="freeze"/>

                <animateTransform attributeType="XML"
                attributeName="transform"
                type="rotate"
                from="0 55 115"
                to="10 55 115"
                dur="2s"
                fill="freeze"
                repeatCount="1"/>

                </path>
        </g>
        <g id="teksti">
            <text transform "matrix(1 0 0 1 58.7791 116.1694)" opacity "0"><tspan x="0" y="0" class="st1">LeafFlow</tspan>
            <animate id="anim1" attributeName="opacity" from="0" to="0"
                dur="4s" fill="freeze"/>
            <animate attributeName="opacity" from="0" to="1"
                begin="anim1.end" dur="3s" fill="freeze"/>
            </text>
        </g>
        </svg>
    </div>
</div>

<div id "container">
<div id "menu">
    <div id "menucontainer">
    <div class "menupainike"><a href "" otsikko1 "Frontpage" otsikko2 "Frontpage">Etusivu</a></div>
    <div class "menupainike"><a href "" otsikko1 "MakeUp" otsikko2 "Cosmetics">Tuotteet</a></div>
    <div class "menupainike"><a href "" otsikko1 "LeafFLOW" otsikko2 "Company">Meistä</a></div>
    </div>
</div>
```

## Sivukokonaisuuden koodi

```
<div id "content">
    <div id "galleria">
        <div id "tuotteet">
            <svg  version="1.1" xmlns="&ns_svg;" xmlns:xlink="&ns_xlink;" width="8%" viewBox="0 0 234.714 96.495"
                style="overflow:visible;enable-background:new 0 0 234.714 96.495;" xml:space="preserve">
            <g id="Layer1">
                <path class="st0" d="M13.479,39.765c-19.702,20.578-28.25-58,50-35c14.673,4.313,43.066,8.948,45,2.5c3-10-36.417-7.825-49.
                    25,3.75
                    C43.605,25.107,3.056,92.69,11.73,96.159c10.067,4.026,32.886-29.53,32.886-29.53S25.824,92.803,34.549,94.146
                    s22.819-28.859,23.49-28.188s-18.121,31.544-8.725,28.859s24.161-22.819,24.161-22.819s-17.45,18.792-8.054,22.819
                    s30.873-31.543,14.765-29.53s9.331,15.624,16.042,6.228s55.639-69.842,43.5-71.5c-10.063-1.374-58.2,93.46-48.804,94.802
                    s87.248-91.276,71.812-93.289s-58.505,91.341-51.007,93.289s34.499-11.552,39.249-19.802c9.017-15.661-8.25-13.5-16,0.25
                    c-4.614,8.187-9.155,24.921,2.255,20.223s29.05-10.425,36.745-20.473s-5-13.75-15.75-0.25
                    c-8.056,10.117-12.556,25.088,3.166,20.052c17.334-5.552,81.334-92.802,72.483-94.631c-9.249-1.912-63.149,87.579-50.399,
                    94.579
                    c6.578,3.611,15.085-7.14,19.527-13.371"/>
            </g>
            <g id="Layer2">
                <path class="st0" d="M85.479,53.265c2.25-11.5,65,7,68-3.25"/>
                <path class="st0" d="M179.479,56.765c1.52-5.864,32.251,8.998,37.5-1.5"/>
            </g>
            </svg>
        </div>
        <div id "kehys" onClick "kuvan_pienennys(this.id)"></div>
        <div id "rivi">
            <div class "kuva" id "1" onClick "kuvan_suurennus(this.id)"><img class "img" src "kuvagalleria\huulipunat.jpg" alt ""/></div>
            <div class "kuva" id "2" onClick "kuvan_suurennus(this.id)"><img class "img" src "kuvagalleria\rasvat.jpg" alt ""/></div>
            <div class "kuva" id "3" onClick "kuvan_suurennus(this.id)"><img class "img" src "kuvagalleria\kynat.jpg" alt ""/></div>
            <div class "kuva" id "4" onClick "kuvan_suurennus(this.id)"><img class "img" src "kuvagalleria\kynsilakat.jpg" alt ""/></div>
            <div class "kuva" id "5" onClick "kuvan_suurennus(this.id)"><img class "img" src "kuvagalleria\kynsilakat.jpg" alt ""/></div>
            <div class "kuva" id "6" onClick "kuvan_suurennus(this.id)"><img class "img" src "kuvagalleria\maskarat.jpg" alt ""/></div>
            <div class "kuva" id "7" onClick "kuvan_suurennus(this.id)"><img class "img" src "kuvagalleria\tarvikkeet.jpg" alt ""/></div>
            <div class "kuva" id "8" onClick "kuvan_suurennus(this.id)"><img class "img" src "kuvagalleria\huulipunat.jpg" alt ""/></div>
        </div>
    </div>
</div>

    <div id "footer">
        <h4>Ota yhteyttä...</h4>
            <textarea cols "50" rows "5"></textarea>
            <br/><br/><br/>
            <a id "button" href "">Lähetä</a>
    </div>

</div>


<script>
var kuva;
var kuva2;
var kehys;

function kuvan_suurennus(id)
{
    kehys = document.getElementById("kehys");
    kuva = document.getElementById(id);
    kuva2 = kuva.innerHTML;
    document.getElementById("kehys").innerHTML= kuva2;
    document.getElementById("kehys").style.zIndex = 3;
    document.getElementById("rivi").style.opacity = "0";
    TweenMax.to(kehys, .3, {opacity:1, ease:Circ.easeOut});
    TweenMax.to(kehys, 1, {scale:1.02});
}

function kuvan_pienennys(id)
{
    kehys = document.getElementById("kehys");
    TweenMax.to(kehys, 1, {opacity:0, scale:1, ease:Power1.easeOut});
    kuva = document.getElementById(id);
    kuva.innerHTML="";
    kuva.style.zIndex = -3;
    document.getElementById("rivi").style.opacity = "1";
}

var controller = new ScrollMagic.Controller();

var scene = new ScrollMagic.Scene({duration: 800})
        .setTween('#girl', 1.5, {y: "70%", ease: Linear.easeNone})
        .addTo(controller);

var scene2 = new ScrollMagic.Scene({duration: 800})
        .setTween('#back', 1.5, {y: "90%", ease: Linear.easeNone})
        .addTo(controller);

var scene3 = new ScrollMagic.Scene({duration: 800})
        .setTween('#logo', 1.5, {y: "10%", ease: Linear.easeNone})
        .addTo(controller);


</script>
</body>
</html>
```

**LIITE 4(1).**

**Canvas kello koodi**

```
<!DOCTYPE HTML>
<html>

<canvas id="canvas" width="400" height="400"></canvas>

<script type="text/javascript">

function kello(){

    var time = new Date();
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext('2d');
    context.save();
    context.clearRect(0,0,400,400);
    context.translate(200,200);
    context.rotate(-Math.PI/2);
    context.lineCap = "round";
    context.fillStyle = "#cccccc";



    context.save();
    context.rotate(Math.PI / 2);


    var angle;
    context.font = "25px arial";
    context.textBaseline="middle";
    context.textAlign="center";
    for (var i=1;i<13;i++){
        angle = i * Math.PI / 6;
        context.rotate(angle);
        context.translate(0, -130);
        context.rotate(-angle);
        context.fillText(i.toString(),0,0);
        context.rotate(angle);
        context.translate(0, 130);
        context.rotate(-angle);
    }



    context.restore();


    var seconds = time.getSeconds();
    var minutes = time.getMinutes();
    var hours  = time.getHours();

    context.fillStyle = "#888888";


    context.save();
    context.rotate( hours*(Math.PI/6) + (Math.PI/360)*minutes + (Math.PI/21600)*seconds )
    context.lineWidth = 12;
    context.beginPath();
    context.moveTo(0,0);
    context.lineTo(80,0);
    context.stroke();
    context.restore();


    context.save();
    context.rotate( (Math.PI/30)*minutes + (Math.PI/1800)*seconds )
    context.lineWidth = 8;
    context.beginPath();
    context.moveTo(0,0);
    context.lineTo(112,0);
    context.stroke();
    context.restore();


    context.save();
    context.rotate(seconds * Math.PI/30);
    context.lineWidth = 3;
    context.beginPath();
    context.moveTo(-30,0);
    context.lineTo(83,0);
    context.stroke();
    context.restore();


    context.save();
    context.rotate(seconds * Math.PI/30);
    context.lineWidth = 7;
    context.beginPath();
    context.moveTo(-40,0);
    context.lineTo(-30,0);
    context.stroke();
    context.restore();

    context.restore();

    window.requestAnimationFrame(kello);
}

window.requestAnimationFrame(kello);

</script>
</html>
```

**LIITE 5(1).**

**SVG-latausikoni**

```
<DOCTYPE html>
<html>
<head>
<title>SVG-latausikoni</title>
<meta charset="UTF-8"/>
<style>

#ratas1{
    position: relative;
    left: -20px;
    top: 155px;
    }
#ratas2{
    position: relative;
    left: 35px;
    top: -80px;
    }

.st0{
    fill-rule:evenodd;
    clip-rule:evenodd;
    fill:url(#grad);
    }

.st1{
    fill-rule:evenodd;
    clip-rule:evenodd;
    fill:#FFFFFF;
    }

</style>
</head>
<body>
    <div style="padding:15%;">
        <div style="width:200px; margin:0 auto;">


<svg id="ratas1" version="1.1" xmlns="&ns_svg;" xmlns:xlink="&ns_xlink;" width="100" height="100" viewBox="0 0 169.2 169.105"
    style="overflow:visible;enable-background:new 0 0 169.2 169.105;" xml:space="preserve">
<g >
<g id="Layer_1">
<radialGradient id="grad" cx="84.6001" cy="84.5532" r="84.5764" gradientUnits="userSpaceOnUse">
    <stop  offset="0.0056" style="stop-color:#45bcca"/>
    <stop  offset="1" style="stop-color:#58d9e8"/>
</radialGradient>
    <path class="st0" d="M69.786,0.053c9.575,0,19.148,0,28.723,0c1.49,6.906,3.912,12.878,5.386,19.8
        c4.582,0.925,7.841,3.178,11.848,4.68c5.989-2.994,11.558-6.41,17.593-9.36c6.496,6.69,14.745,13.53,19.746,20.52
        c-3.102,5.65-5.922,11.582-9.335,16.92c1.769,3.868,3.827,7.442,5.026,11.88c6.63,1.992,13.742,3.501,19.747,6.12
        c0,9.36,0,18.72,0,28.081c-6.689,1.813-13.323,3.681-19.747,5.76c-1.416,3.979-2.898,7.894-5.026,11.16
        c3.062,6.17,6.369,12.094,9.694,18c-7.15,6.153-13.673,14.846-20.823,19.8c-5.864-2.881-11.087-6.403-17.234-9
        c-3.59,1.92-7.409,3.61-11.488,5.04c-1.662,6.733-3.513,13.278-5.745,19.44c-9.453,0-18.908,0-28.363,0
        c-1.818-6.576-3.865-12.925-5.743-19.44c-4.08-1.311-8.285-2.493-11.488-4.68c-6.44,3.143-12.188,6.979-18.67,10.08
        c-6.496-6.69-14.746-13.53-19.747-20.521c3.147-6.085,6.804-11.658,9.694-18c-1.846-3.548-3.061-7.73-4.308-11.88
        c-6.469-2.153-13.64-3.604-19.746-6.12c0-9.48,0-18.96,0-28.44c6.891-1.37,12.561-3.964,19.387-5.4
        c1.399-4.237,2.934-8.338,4.667-12.24c-3.267-6.084-6.551-12.151-10.052-18c6.909-6.752,13.016-14.308,20.823-20.16
        c5.617,3.609,11.89,6.558,17.592,10.08c3.511-2.119,8.005-3.253,12.207-4.68C65.963,12.78,68.198,6.74,69.786,0.053z
        M55.066,54.053c-6.351,6.069-10.528,12.74-12.207,23.04c-4.821,29.606,19.089,51.36,45.237,49.32
        c26.92-2.101,47.891-33.004,33.39-61.2c-6.421-12.484-20.827-24.365-41.646-22.32C68.472,44.01,60.688,48.681,55.066,
        54.053z">

        <animateTransform attributeType="XML"
        attributeName="transform"
        type="rotate"
        from="360 85 85"
        to="0 85 85"
        dur="4665ms"
        repeatCount="indefinite"/>
    </path>
</g>
<g id="Layer_2">
    <circle class="st1" cx="84.267" cy="85.553" r="51.833"/>
</g>
</g>
</svg>

<svg id="ratas2" version="1.1" xmlns="&ns_svg;" xmlns:xlink="&ns_xlink;" width="180" height="180"
    viewBox="0 0 169.2 169.105" style="overflow:visible;enable-background:new 0 0 169.2 169.105;" xml:space="preserve">
<radialGradient id="grad" cx="84.6001" cy="84.5532" r="84.5764" gradientUnits="userSpaceOnUse">
    <stop  offset="0.0056" style="stop-color:#45bcca"/>
    <stop  offset="1" style="stop-color:#58d9e8"/>
</radialGradient>
<path class="st0" d="M77.05,1.649c4.994,0,9.989,0,14.983,0c0.545,5.559,1.39,10.819,1.85,16.463
    c5.462,1.259,11.063,2.379,15.723,4.439c3.729-4.164,6.64-9.145,10.359-13.318c4.644,2.077,8.922,4.519,12.763,7.399
    c-2.08,5.133-4.363,10.064-6.474,15.168c3.975,3.855,8.118,7.543,11.653,11.838c5.182-2.032,10.127-4.301,15.168-6.474
    c3.173,3.548,5.304,8.138,7.584,12.579c-3.985,3.906-9.001,6.784-13.318,10.358c1.873,4.848,3.365,10.076,4.254,15.908
    c5.413,0.815,11.077,1.379,16.648,2.035c0.529,4.568,0.529,10.415,0,14.983c-5.521,0.644-11.123,1.208-16.463,2.034
    c-1.271,5.389-2.231,11.087-4.439,15.538c3.982,3.972,9.159,6.748,13.318,10.544c-2.067,4.653-4.52,8.922-7.399,12.763
    c-5.108-2.105-10.121-4.307-15.168-6.474c-3.837,3.993-7.543,8.117-11.838,11.653c2.075,5.139,4.275,10.152,6.474,15.168
    c-3.543,3.178-8.141,5.301-12.578,7.584c-3.894-3.999-6.809-8.977-10.358-13.318c-4.849,1.872-10.103,3.339-15.908,4.255
    c-0.814,5.412-1.379,11.076-2.034,16.647c-4.568,0.529-10.415,0.529-14.983,0c-0.553-5.551-1.374-10.835-1.85-16.463
    c-5.525-1.258-11.223-2.343-15.908-4.439c-3.752,4.14-6.626,9.158-10.358,13.318c-4.644-2.076-8.922-4.519-12.763-7.399
    c2.081-5.133,4.348-10.079,6.474-15.168c-3.981-3.849-8.106-7.555-11.653-11.838c-5.169,2.045-10.133,4.295-15.168,6.474
    c-3.173-3.548-5.304-8.137-7.584-12.578c3.999-3.894,8.976-6.808,13.318-10.358c-1.872-4.849-3.339-10.103-4.254-15.908
    c-5.413-0.814-11.076-1.378-16.647-2.034c-0.316-0.917,0.024-2.491-0.37-3.33c0-2.836,0-5.672,0-8.509
    c0.472-0.761-0.112-2.578,0.555-3.145c5.471-0.695,11.119-1.213,16.463-2.035c0.916-5.805,2.382-11.059,4.254-15.908
    c-4.33-3.563-9.326-6.458-13.318-10.358c2.283-4.438,4.407-9.035,7.584-12.579c5.042,2.171,9.977,4.451,15.168,6.474
    c3.537-4.293,7.662-7.999,11.653-11.838c-2.167-5.046-4.369-10.06-6.474-15.168c3.841-2.879,8.109-5.332,12.763-7.399
    c3.732,4.16,6.606,9.179,10.358,13.318c4.714-2.007,10.127-3.315,15.908-4.254C75.845,12.899,76.386,7.213,77.05,1.649z">

        <animateTransform attributeType="XML"
        attributeName="transform"
        type="rotate"
        from="0 85 85"
        to="360 85 85"
        dur="7s"
        repeatCount="indefinite"/>


    </path>
<circle class="st1" cx="85.267" cy="86.22" r="23.833"/>
</svg>


        </div>
    </div>

</body>
</html>
```