HAAGA-HELIA
ammattikorkeakoulu

Browsermark 2.1 HTML5 SVG test

Rightware Oy

Erno Tuovinen

Opinnäytetyö

Tietojenkäsittely

2013

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

Opinnäytetyön mahdollisen esityksen päivämäärä

Koulutusohjelma

Tekijä tai tekijät Erno Tuovinen	**Ryhmä tai aloitusvuosi** 2013
Opinnäytetyön nimi Browsermark 2.1 HTML5 SVG testi	**Sivu- ja liitesivumäärä** 27-(3)
Ohjaaja tai ohjaajat Anne Valsta	
Opinnäytetyö tehdään toimeksiantona Rightware Oy:lle. Toimeksiannon työ tulee keskittymään Rightwaren Browsermark 2.1 testaustyökaluun. Rightwaren Browsermark-testillä testataan internetselaimen tehokkuutta mobiili-, pöytäkone- ja tablet- laitteilla. Opinnäytetyöllä tuotetaan yksi Browsermasrk 2.1 kokonaisuuden testinosa tukemaan isompaa kokonaisuutta. Testinosa testaa selaimien HTML5 SVG grafiikan käyttöä ja valmiutta käyttää sitä. SVG (Scalable Vector Graphics) on tulevan HTML5 stantardin osa joka käsittelee verktorigrafiikkaa.	
Asiasanat HTML5, SVG	

HAAGA-HELIA
University of Applied Sciences

Abstract

6th May 2013

Degree programme in Business

Author Erno Tuovinen	**Group or year of entry** 2013
The title of thesis Browsermark 2.1 HTML5 SVG test	**Number of pages and appendices** 27-(3)
Supervisor Anne Valsta	
Thesis is assignment from Rightware Oy. Assignment work will be concentrate Rightware Browsermark 2.1 browser testing tool. Browsermark test tool test's internet browser efficiency in mobile, tablet and desktop computers. Thesis is one part of the Browsermark 2.1 test set. Test will test browser HTML5 SVG graphic usage and preparedness to use it. SVG (Scalable Vector Graphics) is one of the standard that is included in in future HTML5 standard.	
Key words HTML5, SVG	

Sisällys

1 Johdanto

Johdanto-osuudessa käsitellään projektin taustaa. Johdannossa selviää miksi projekti aloiteltiin ja mitä projektilla haluttiin toteuttaa.

1.1 Projektin tavoitteet ja tausta

Rightware Oy on yritys, joka on erikoistunut Benchmark-testien tekemiseen. Benchmark testauksen yleinen kuvaus on havainnoitu osiossa 1.8.

Toimeksiannon työ keskittyy Rightwaren browsermark-testiin. Browsermark-testillä testataan internet-selaimen tehokkuutta mobiili-, pöytäkone- ja tablet- laitteilla. Rightware Oy:n tuotekehityksessä on tarvetta benchmark-testien ajoittaisille uusimisille, jotta loppukäyttäjät ja asiakkaat pidetään testeistä kiinnostuneina, ja jotta testit tukisivat alati kehittyvää Web-teknologiaa. Browsermark-testin viimeisestä päivityksestä oli noin vuosi aikaa, joten testikokonaisuuden uusiminen oli ajankohtaista. Uuden testikokonaisuuden rungon ominaisuudet ovat työn alla.

Opinnäytetyöllä tuotettiin yksi testin osa tukemaan isompaa kokonaisuutta. Tähän mennessä testejä on ajettu 180.000 kappaletta. Kun Browsermark 2.0 -versio julkaistiin, monet selaimet eivät vielä kunnolla tukeneet Canvas, WebGL ja CSS 2D/3D -transformaatioita. Tilanne on selainteknologiassa parantunut, ja miltei kaikki selaimet tulevat näitä teknologioita ja standardeja. Rightware Oy haluaa kehittää testityökalua niin, että se kattaa uudet, sellaiset teknologiat, joita ei vielä tueta kunnolla. Yksi näistä

osista on tuleva HTML5 SVG -standardi. Tämä projektikuvaus ja opinnäytetyö liittyvät tähän testin osaan ja sen toiminnan kehitykseen ja toteuttamisen kuvaukseen.

1.2 Projektin teoriataustaa

Opinnäytetyön pohjalla pyörii Rightware OY:n Browsermark-benchmark – testiympäristö. Testiympäristö testaa selaimen tehokkuutta, kykyä suorittaa yksittäisiä tehtäviä ja kykyä tukea uusia teknologioita, tässä testitapauksessa SVG-teknologiaa.
Browsermark pyrkii synteettisen suorituskykymittauksen sijasta suorittamaan kokonaisvaltaisesti testejä, joita selaimet joutuvat tekemään tosimaailmassa (Browsermark on siis holistinen, tosimaailmaan pohjautuva benchmark). Browsermark käyttää teknologioiden suosituksina World Wide Web Consortium (W3C) suosituksia: HTML5, CSS3, SVG. Browsermark tekee suorituskykymittauksen (performance) lisäksi yhteensopivuustestauksen (conformance), joka ei kuitenkaan vaikuta kokonaistulokseen. Rightware Oy käsittää bencharkkaamisen ideaa seuraavasti:

- Benchmarkkauksen pitää olla objektiivinen kaikkia eri arkkitehtuureja kohtaan.

- Benchmarkilla pitää olla täysi läpinäkyvyys koodipuolelta, jotta kaikki tietävät mitä ”konepellin” alla tapahtuu.

- Tuote pitää kehittää asiakkaiden ja käyttäjien ehdoilla, kuitenkaan suosittamatta heistä ketään.

- Benchmarkkauksen pitää pystyä tarjoamaan lisäarvoa asiakkaille ja käyttäjille, eli saada heidät näyttämään todellinen performanssi heidän omilleen asiakkailleen.

- Benchmarkin pitää olla pitkäkestoinen työkuormiltaan, jotta se tarjoaa parhaiten vastinetta asiakkaiden rahoille.

- Benchmark käyttää virallisia spesifikaatioita, jotka kolmannet osapuolet ovat määritelleet (esim. OpenGL ES 3.0 spesifikaatio, jonka on määritellyt Khronos).

1.3 Projektin käsitteitä

OpenGL

Open Graphics Libary. Monialustainen API 2D- ja 3D -grafiikan mallintamiseen. Yleisesti API:a käytetään GPU kanssa, jotta voidaan saavuttaa hardware-kiihdytetty grafiikkamallennus.

GPU

Graphics proseccing unit. Tietokoneen näytönohjainpiiri.

API

Application programming interface. API määrittää, miten ohjelmiston eri komponenttien pitäisi käyttäytyä keskenään.

HTML5

Kuvauskieli, jolla voidaan esittää ja rakentaa WWW sivuja.

SVG

SVG on kuvauskieli kuvaamaan kaksiulotteista graafisia applikaatioita ja kuvia.

Canvas

HTML-elementti, jolla voidaan piirtää grafiikkaa reaaliajassa Web sivulle.

CSS

Cascading Style Sheets. Tyylikieli, jolla luodaan graafinen ulkoasu kuvauskielellä tuotetun dokumentin päälle. Näitä ovat mm. HTML, XHTML, XML

BDP Member

Rightwaren Benchmark Development Program. BDP-jäsenet ovat Rightware Oy:n asiakkaita, mutta samalla he antavat Becnhmarking-tuotteista mielipiteitä ja kehitysehdotuksia.

Git HUB

Web Hosting –sovellus, jolla voidaan ylläpitää GIT versionhallintaa.

GIT

Versionhallintajärjestelmä. Ilmainen Linus Torvaldsin kehittämä source code managment alusta.

Milestone

Sovelluksenhallinta-arkkitehtuurin sisällä oleva tapahtuma. Tapahtuma on usein joku isomman ohjelmistokokonaisuuden valmistuminen. Milestone on tapa seurata ohjelmistokehityksen etenemistä.

SCRUM

Ketterän kehityksen puite. Ketterän sovelluskehittämisen toimintamalli.

Bacecamp

37 Signallsin kehittämä projektinhallintasovellus, joka toimii web-sivun päällä.

CMS

Content managment system. Applikaatio jolla voidaan hallita, luoda ja tallentaa sisältöä web-sivuille.

Pseudokoodi

Tapa kuvata ohjelmiston funktion toimintaa. Pseudokoodi on verbaalisesti kuvattu ja havainnoitu koodi ongelmasta, jota pyritään ratkaisemaan konekielisesti. Tällä tavalla voidaan isompia kokonaisuuksia havainnoida yksinkertaisemmin.

Flowchart

Flowchart kuvaa algoritmin, funktion tai prosessin toimintaa visuaalisella tavalla.

1.4 Projektin tehtävä

Projektin tehtävänä oli tuottaa ja dokumentoida yksi uuden Browsermark 2.1-testin osa. Tämä osa käsitti uuden HTML5 SVG-grafiikka -testin. Työssä kuvattiin testin suunnittelu, toteutus ja valmistelu. Lopuksi tehtiin kooste projektin vaiheista.

1.5 Projektin tavoite ja lopputulokset

Tavoitteena oli saada toimiva testi ja dokumentoitua testin suunnittelu, rakenne sekä toiminta. Oppimistavoitteena oli ymmärtää paremmin HTML5:sen tuomia mahdollisuuksia vektorigrafiikan käytettävyydestä web-kehityksessä.
Testissä mitattiin kahta arvoa; nopeutta ja tarkkuutta. Näiden yhteistuloksesta muodostettiin arvo, jota painotettiin standardikertoimella, josta laskettiin testin päätyttyä varsinainen tulos. Aikasempia tuloksia testistä ei ole, mutta tulosta voitiin verrata selainten ja laitteiden välillä.

1.6 Projektin rajaus

Projekti ei sisällä kokonaista Browsermark–tuotetta, vaan tämä on yksi osa isompaa testikokonaisuutta. Projekti ei mittaa SVG -tuen yhdenmukaisuutta selaimissa, vaan keskittyy selaimen suorituskykymittaukseen.

1.7 Projektin organisaatio ja kumppanit

Opinnäytetyön tekijä on projektipäällikkö ja sihteeri ohjauskokouksissa: Erno Tuovinen
Opinnäytetyön tekijä: Erno Tuovinen
Opinnäytetyön ohjaaja: Anne Valsta
Toimeksiantajan edustaja: Teemu Uotila, Jouni Tuovinen

Rightwaren Benchmark Development Program (BDP) -jäsenet saivat ehdotuksen testistä ja heillä oli vaikutusmahdollisuus lopullisen testin työkuormiin ja testausmenetelmiin. Tämä sen takia, jotta objektivisuusperiaate täyttyy benchmarkin osalta.

1.8 Benchmarking yleisesti

Benchmarking on tietokoneohjelma, jolla mitataan relatiivista tehokkuutta. Benchmarking assosioidaan yleisesti hardware-teknologian testaamiseen esimerkiksi prosessorien liukulukuprosessien tehokkuuden mittaamiseen. Benchmarking-teoriaa voidaan kuitenkin soveltaa myös softwaren testaamiseen. Benchmarkingia tarvitaan, jotta voidaan verrata monen ohjelman arkkitehtuuria yleiseen standardiin.
Benchmarkit on suunniteltu jäljittelemään tietyn tyyppistä työtaakkaa komponentille tai ohjelmalle. Vaikka benchmark jäljittelee oikean elämän kaltaisia prosesseja systeemissä, antavat testit yleensä paremman tuloksen kuin oikean elämän ja käyttäjän luomat rasitteet. Synteettinen benchmarkkaaminen on kuitenkin hyödyllinen testaamaan ohjelman tai laitteen yksittäisiä komponentteja. Benchmarkkaamiseen liittyy haasteita, koska tarvitaan useita interatiivisia kierroksia, jotta saavutetaan luotettava ja hyödynnettävä lopputulos. Benchmarkkaamisessa on todella tärkeää, että sen tuottama data pysyy puhtaana. (Wikipedia 2013 Benchmark computing)

1.9 Projektin aikataulutus

no	Työosio	Aika	Status
1	Testin suunnittelu	1.5.2013 – 1.6.2013	Valmis
2	Testin teknisen osan toteutus	1.6.2013 - 30.8.3013	Valmis
3	Opinnäytetyön viimeistely ja tarkastaminen	1.9.2013 - 29.10.2013	Valmis

2 Suunnittelu

Opinnäytetyön suunnitteluosio sisältää projektin suunnittelunvaiheet. SVG -testin suunnittelussa havainnoidaan testin toimintaa ja graafista rakennetta.

2.1 Suunnittelun rakenne

Browsermark-versio 2.1 pitää sisällään paljon uudistuksia. Suunnittelun ensimmäinen vaihe oli kirjoittaa uuden version 2.1 dokumentointi uudelleen. Tämän jälkeen suunnitelmadokumentaatio lähetettiin Rightwaren Benchmark Development Program (BDP) -jäsenten arvioitavaksi. Suunnittelussa päätettiin, että olemassa olevia testin komponentteja päivitetään ja uutena testinä tuodaan SVG-testi testaamaan vektorigrafiikkaa.

2.2 Versionhallinta

SVG -testin versionhallintaa varten paketin lähdekoodi laitettiin GitHubiin, jossa GIT:in avulla versionhallinta on helppo tehdä. Git repository on Rightwaren Oy:n yksityinen versionhallintatietokanta eikä se ole open source -koodia.

2.3 SVG W3C -standardi

Kuva 1 SVG Logo public.image

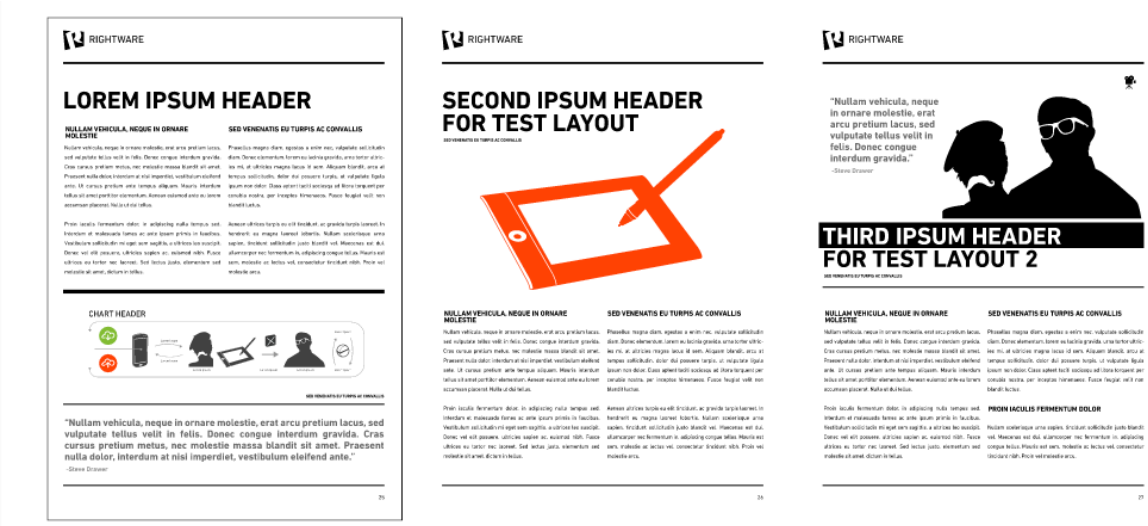
SVG on kuvauskieli kuvaamaan kaksiulotteista graafisia applikaatioita ja kuvia. SVG 1-1 -versio on W3C:n suosittelema versio. Browsermark SVG -testi käyttää versiota 1.1. SVG Tony 1.2 -versio on suunnattu mobiililaitteille. SVG 2 on tällä hetkellä kehitysasteella ja se tulee tuomaan uusia ominaisuuksia versioon 1.1, ja se tullaan myös integ-

roimaan syvemmin DOM-puuhun HTML ja CSS -standardeihin. W3C:n ”working group” työskentelee luomalla moduuleita, kasvattamaan nykyistä spesifikaatiota ja lisäämään funktionalisuutta CSS-kieleen. (W3C SVG Graphics)
SVG luotiin web-kehittäjien tarpeesta saada avoin ja intuitiivinen vektorigrafiikkafunktionaalisuus web-sivuille. SVG on suunniteltu XML-kielen päälle ja halutaan sarjoittaa osaksi HTML5-standardia, jolloin SVG toimisi natiivisti suurimmalla osalla isoja selaimia, jotka tukevat XHTML-standardia. (W3C SVG WG Proposal)

2.4 Testin perustoiminta

SVG-testi toimii normaalin web-sivupohjan päällä. Sivu alustetaan lokaaliin ympäristöön, sillä testisivu vaatii lokaalin palvelimen toimiakseen. Testi on dokumentoitu nimellä HTML5 SVG -testi ja työnimike testille ovat ”document stack”. SVG-testissä web-sivulla selaimessa on vektorigrafiikalla luotuja ”papereita”, jossa jokaisessa paperissa on vektorigrafiikalla kirjoitettua tekstiä ja grafiikkaa. Testi suorittaa zoom in-toiminnon, zoom out- ja scroll-toiminnot.
Zoom in -toiminto zoomaa ensimmäiseen paperiin ja suorittaa scroll toiminnon, jossa paperi luetaan yläreunasta alareunaan. Tämän jälkeen suoritetaan zoom out -toiminto ja siirrytään toiseen paperiin, jonka jälkeen suoritetaan taas zoom in ja scroll -toiminnot. Yksi testikierros pitää sisällään zoom out + zoom in + scroll-toiminnot. Nämä toiminnot suoritetaan jokaiselle kolmelle paperille, joista jokaisesta muodostuu yksi testikierros.
Testikierroksia on yhteensä 11 kappaletta. Testitapaus loppuu, kun viimeinen 11. kierros on päättynyt. 11 kappaleen summa sisältää kolme kertaa kaikki paperit ja yksi ylimääräinen fokus papereihin 1 ja 2.
Testi on balansoitu kestämänään alustasta riippuen noin 15 sekuntia. Jokaisessa testin kierroksessa mitataan SVG-elementin paikka. Elementin paikan osalta tarkastetaan, onko se testin viewportin (999x 600) sisällä. Jokaisella kerralla, kun elementti on sijoittunut oikein viewportin sisälle, annetaan testille plus-piste. Jos elementti taasen on viewportin ulkopuolella, pistettä ei anneta lainkaan. Testi ei jaa ulkopuolelle menneistä elementeistä miinuspisteitä. Testin päätyttyä kokonaisaika testin tekemiseen mitataan ja lasketaan operaatiot per sekunti (OPS). OPS-arvoa muokataan kertoimen mukaan ja verrataan ennalta tiedettyyn vertailutulokseen, josta saadaan testille lopullinen testitulos.

2.5 Testin graafinen ulkoasu



Kuva 2 SVG-testin graafinen näkymä testin aikana

Kuva 3 SVG-testin graafinen näkymä Browsermark 2.1 web-sivun päällä

3 Tekninen osa

Opinnäytetyön tekninen osa pitää sisällään SVG-testin teknillisen rakenteen kuvauksen. Osiossa selviää miten testi on teknisesti rakennettu ja miten suunnittelusta siirryttiin tuotteen toteutukseen.

3.1 Suunnittelusta projektin toteutukseen

Suunnittelin koko Browsermark 2.1 SVG testin rakenteen milestone kerrallaan. Nämä osiot jaettiin pienempiin segmentteihin, joista saimme projektin milestonet ja tehtävälistat. Tämä helpotti projektin seurantaa ja pystyimme tekemään realistiset viikoittaiset SCRUM-sprintit.
Tehtävälistat ja yksittäiset työtehtävät laitettiin Basecamp-projektinhallintasovellukseen talteen. SVG-testin pseudo-koodia tarkastettiin ja havaittiin, että kaikki toiminnallisuuksia ei voitu suorittaa niin kuin alun perin suunniteltiin. Koodin rakennetta korjattiin, jonka myötä testin flow chart -kaaviota jouduttiin päivittämään, koska koodiin lisättiin uusia toiminnallisia funktioita.

3.2 SVG-testin funktioiden kuvaus

Funktioiden kuvaus on laitettu liitteeksi, koska se on todettu salaiseksi yrityksen NDA (Non-disclosure agreement) -sopimuksen mukaisesti

LIITE 1: functioiden_kuvaus.docx

3.2.1 SVG Zoom -toiminnon testikoodi

SVG Zoom toiminnon testi kuvaus on laitettu liitteeksi, koska se on todettu salaiseksi yrityksen NDA (Non-disclosure agreement) -sopimuksen mukaisesti

LIITE 2: Svg_Zoom_Test.docx

3.3 SVG-testin flowchart

Flowcart-kuvaus on laitettu liitteeksi, koska se on todettu salaiseksi yrityksen NDA (Non-disclosure agreement) -sopimuksen mukaisesti

LIITE3_SVG_Test-Flowchart.png

3.4 SVG-testin pseudo-koodi

LIITE 4: svg_test_pseudo_code.docx

3.5 SVG-testin ohjelmallisen toteutuksen tietoperusta

Testin ohjelmallisen toteutuksen tietoperustan osio sisältää SVG testin teknisen toiminnan kuvauksen. Osio havainnoi miten testi on rakennettu ja miten se pyörii selain alustan päällä

3.5.1 SVG testin testialue selaimessa

SVG-testi on suunniteltu 1894 pikseliä leveänä ja 869 pikseliä korkeana. Browsermarkin vakioitu benchmark-alue on 999 pikseliä leveä ja 600 pikseliä korkea. Näkymä, jossa kaikki kolme paperia ovat näkyvissä vakioidulla benchmark-alueella, vaatii 50 % suurennoksen, jolloin vasemmalle ja oikealle jää 26 pikselin tyhjä tila.

3.5.2 ZOOM-funktion SVG:n skaalaus, panorointi ja vieritys

SVG:n kontrollointi on helpointa toteuttaa matriisin avulla, jossa kuudella parametrilla matrix (**sx, cx, cy, sy, x, y**) pystytään tekemään skaalaus, panorointi, vieritys, kierto ja vääristymä. Matriisin parametrien merkitykset ovat seuraavat:

- sx kontrolloi x-akselin skaalausta
- cx kontrolloi x-akselin kiertoa ja vääristymää (ei käytössä tässä projektissa, koska kamera liikkuu vain x- ja y-akselilla, ei z-akselilla)
- cy kontrolloi y-akselin kiertoa ja vääristymää (ei käytössä, ks. yllä)

- sy kontrolloi y-akselin skaalausta
- x kontrolloi x-akselin muutosta (panorointi)
- y kontrolloi y-akselin muutosta (vieritys)

3.5.3 Skaalaus

SVG-skaalaus kohdistuu oletuksena SVG:n vasempaan yläkulmaan. Näin ollen esimerkiksi käytössä olevan SVG:n (1894 pikseliä leveä, 869 pikseliä korkea) skaalaaminen matriisilla matrix(0.5 0 0 0.5 0 0) tarkoittaa, että SVG:stä loitonnutaan 50% kohdalle, ja SVG mahtuu tämän jälkeen 947 pikseliä leveälle ja 435 pikseliä korkealle alueelle. Jos taas halutaan lähestyä SVG:tä, voidaan käyttää esimerkiksi matriisia matrix(2 0 0 2 0 0) jolloin lähennytään 200 % kohdalle ja SVG:n koko tiivistyy 3788 pikseliä leveäksi ja 1738 pikseliä korkeaksi

3.5.4 Panorointi

Panorointia voidaan kontrolloida matriisin toiseksi viimeisellä parametrilla. Jos halutaan panoroida käytössä olevan SVG:n toisen paperin yläosaan 150 % skaalauksella, voidaan käyttää matriisia matrix(1.5 0 0 1.5 -917 0), jossa toiseksi viimeinen luku -917 on tulo laskulausekkeesta vasenlaita - (SVG:n perusleveys aloitusskaalauksessa * (paperin järjestysnumero - 1)) + marginaali = 0 - (947 * 1) + 26 = -917. Samalla kaavalla laskettuna kolmas paperi olisi 0 - (947 * 2) + 26 = -1868 ja ensimmäinen paperi olisi 0 - (947 * 0) + 26 = 26.

Paperin järjestysnumerosta vähennetään yksi arvo sen takia, että SVG:ssä ensimmäisen elementin järjestysnumero on todellisuudessa 0 (vrt. Array, listojen indeksointi), jolloin toisen paperin järjestysnumero SVG:ssä on 1.

3.5.5 Vieritys

Veritystä voidaan kontrolloida matriisin viimeisellä parametrilla. Jos halutaan vierittää kolmannen paperin alaosaan 150 % skaalauksella, voidaan käyttää matriisia matrix(1.5 0

0 1.5 -1868 -704), jossa viimeisin luku -704 on tulo laskulausekkeesta yläreuna - (alkuperäinen korkeus * skaalaus) - näkymän korkeus = 0 - (869 * 1.5) - 600 = -704.

3.5.6 Tasapainotulos

Tasapainotuloksen kuvaus on laitettu liitteeksi, koska se on todettu salaiseksi yrityksen NDA (Non-disclosure agreement) -sopimuksen mukaisesti

LIITE 5: liite_tasapainotulos.docx

4 SVG -testin lopullinen toteutus tuotteeksi

Tässä osiossa havainnoidaan raakatuotteen tuotteistamista lopulliseksi julkaisukelpoiseksi tuotteeksi. Tämä kappale pitää sisällään testin testauskäytännöt ja toimintaympäristön kartoittamisen ja rakentamisen.

4.1 Testiympäristön rakentaminen lokaaliin ympäristöön

Alusta: Windows 8 64bit
Lokaalipalvelin: Apache 2.2.22 (WAMP)
Testiympäristö vaatii palvelinympäristön toimiakseen. Ympäristö on rakennettu web-teknologioilla HTML, CSS, JavaScript ja PHP. Apache-serverille tehdään oma virtuaalipalvelin, johon testiympäristön public_html osoitetaan. Ympäristön kehittäminen ei vaadi tietokantaa, ellei haluta testin tuloksia talteen lokaaliin tietokantaan.

4.2 SVG-testin alustaminen testiympäristöön

Browsermark-testi toimii samalla tavalla kuin CMS (Content management system) -järjestelmä. Sivu koostuu neljästä magic-tiedostosta (globaalista php-tiedostoista, joista sivun osat koostetaan).
Näitä ovat:

- Parametrit (Sisältää tiedot ennen headeria, voi mm. määrittää sivun title-osan)
- Sivun yläosa (sivun head-osuus html → body)
- Sivun konteksti alue (sivun content-alue body → footer)
- Sivun alaosa (sivun alaosa footer → footer)

Oman testin luominen järjestelmään vaatii vain muutaman tiedoston.
Browsermark-alustalle tehdään kansio, jonne tehdään uusi test.js ja content.php -alustalle luodaan uusi kansio oikean testiryhmän alle. Tässä tapauksessa ryhmä on ”graphic”, koska testi on SVG-grafiikkaan liittyvä.

Testialustan kohdekansioon luodaan kaksi tiedostoa. Test.js ohjaa itse testiä ja sen toimintaa, ja sivun kontekstialue index.html sisältää testin HTML5-osuuden, johon test.js:n sisältämä testin toiminnallisuus kohdistetaan.

Testi sisältää init.js-tiedoston, joka on osa Browsermark-testin omaa koneistoa. Init.js suorittaa testin aloittamiseen liittyvät tarkastukset joita ovat mm Modenizer.js tarkastus voiko, SVG-testiä ajaa selaimella. Init.js lähettä taustajärkestelmälle tiedon testin aloittamisesta ja antaa Benchmark-moottorille luvan käynnistää testi.

LIITE 6: liite_InitJS_alustus.docx SALAINEN

4.3 Testialustan testaaminen

Kun init.js on muokattu testiä varten sopivaksi, liitetään web-sivulle SVG-koodi ja aloitetaan testi.

SVG-testin toimintaan luodaan temp-koodi

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <ellipse cx="240" cy="50" rx="220" ry="30" style="fill:yellow" />
  <ellipse cx="220" cy="50" rx="190" ry="20" style="fill:white" />
</svg>
```

Koodi tulostaa vain kaksi keltaista elliptistä rinkulaa sivuille. Tällä voidaan testata, että SVG toimii selaimessa, ja että selain tukee SVT-grafiikkaa.

4.4 SVG-selaintuen laajempi selvitys

Ennen SVG-testin ohjelmoimista tehtiin SVG-selaintuelle tarkempi ja laajempi tutkimus. Tutkimuksessa selvitettiin, miten selainkanta tukee SVG-teknologiaa ja miten teknologiaa tukemattomat selaimet voitaisiin saada suorittamaan testi. Testiin käytettiin Voormedian luomaa web-sivua (http://voormedia.com/blog/2012/10/creating-svg-vector-graphics-for-maximum-browser-compatibility/svg-browser-test) julkiseen käyttöön, jossa on listattu SVG:n mahdollistamat ominaisuudet. Sivulla on SVG:n kaikki

tämänhetkiset toiminnot ja sivun lataaminen selaimessa näyttää, mitä toimintoja selain voi suorittaa.

4.4.1 Voormedia-tulokset

Ilmoitetut toiminnot ovat SVG-testin hylättyjä toimintoja

Chrome

- img link action

Opera

- img link

Safari

- customfont
- dropshadow
- gauss. blur
- outer glow
- inner glow
- img embed
- img link
- mesh

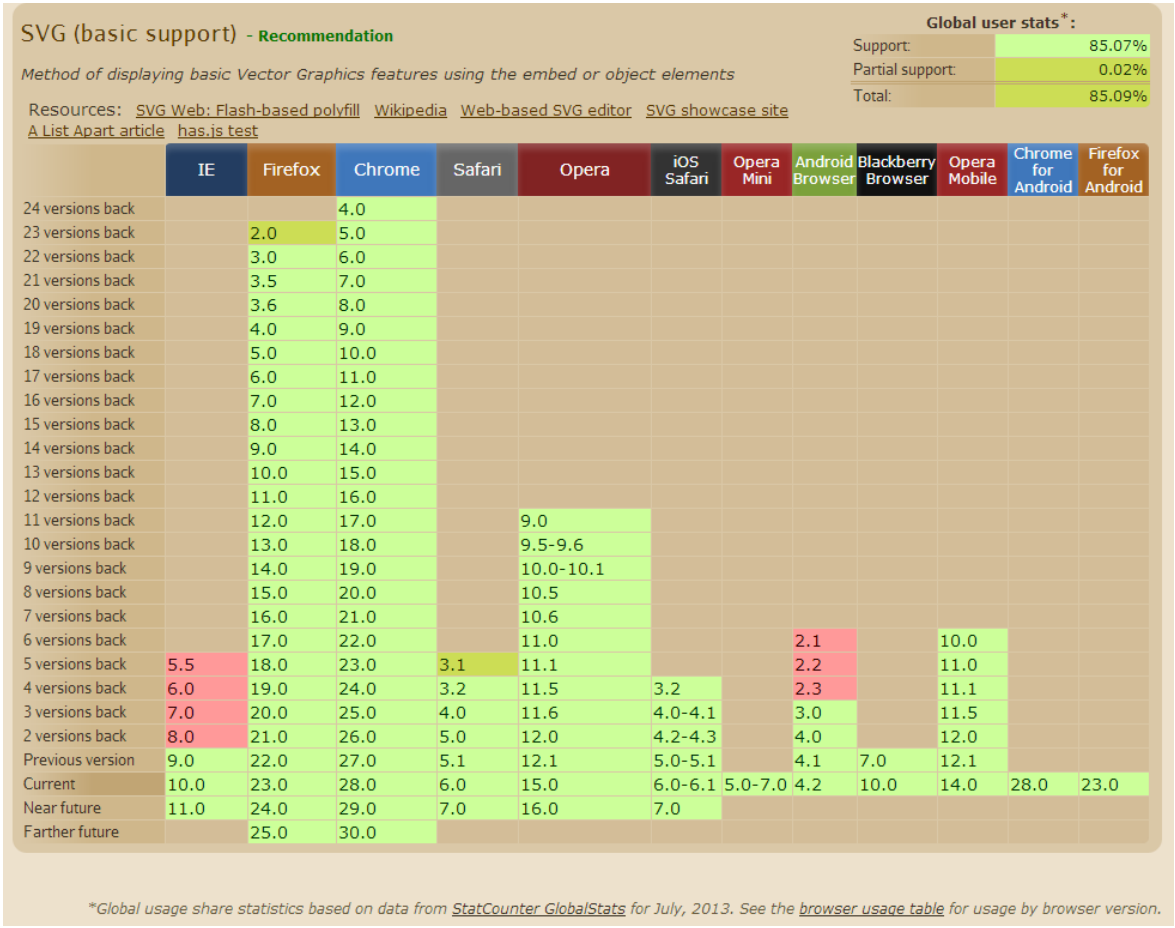
IE10

- img link

IE09

- img link

4.4.2 SVG-selaintukikaavio



SVG (basic support) - Recommendation

Method of displaying basic Vector Graphics features using the embed or object elements

Resources: SVG Web: Flash-based polyfill, Wikipedia, Web-based SVG editor, SVG showcase site, A List Apart article, has.js test

Global user stats*:

Support:	85.07%
Partial support:	0.02%
Total:	85.09%

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android
24 versions back			4.0									
23 versions back		2.0	5.0									
22 versions back		3.0	6.0									
21 versions back		3.5	7.0									
20 versions back		3.6	8.0									
19 versions back		4.0	9.0									
18 versions back		5.0	10.0									
17 versions back		6.0	11.0									
16 versions back		7.0	12.0									
15 versions back		8.0	13.0									
14 versions back		9.0	14.0									
13 versions back		10.0	15.0									
12 versions back		11.0	16.0									
11 versions back		12.0	17.0		9.0							
10 versions back		13.0	18.0		9.5-9.6							
9 versions back		14.0	19.0		10.0-10.1							
8 versions back		15.0	20.0		10.5							
7 versions back		16.0	21.0		10.6							
6 versions back		17.0	22.0		11.0			2.1		10.0		
5 versions back	5.5	18.0	23.0	3.1	11.1			2.2		11.0		
4 versions back	6.0	19.0	24.0	3.2	11.5	3.2		2.3		11.1		
3 versions back	7.0	20.0	25.0	4.0	11.6	4.0-4.1		3.0		11.5		
2 versions back	8.0	21.0	26.0	5.0	12.0	4.2-4.3		4.0		12.0		
Previous version	9.0	22.0	27.0	5.1	12.1	5.0-5.1		4.1	7.0	12.1		
Current	10.0	23.0	28.0	6.0	15.0	6.0-6.1	5.0-7.0	4.2	10.0	14.0	28.0	23.0
Near future	11.0	24.0	29.0	7.0	16.0	7.0						
Farther future		25.0	30.0									

*Global usage share statistics based on data from StatCounter GlobalStats for July, 2013. See the browser usage table for usage by browser version.

Kuva 4 Kuvaaja SVG-selaintuesta (http://caniuse.com/svg)

4.4.3 Vaihtoehdot selaimille, joissa SVG-tukea ei ole

Jos selain ei tue SVG-standardia, on olemassa muutamia keinoja miten vektorigrafiikan saa kuitenkin toimimaan. SVG-koodin voi tallentaa omaan .svg tiedostoon ja käyttää html struktuurissa object-elementtiä. Esim. <object data=url/file.svg'></object>. Hyvin tuettu keino on käyttää SVG:lle tarkoitettuja java scripit -kirjastoja. Nämä ovat Modenizr.js ja Raphael.js. Jos SVG-koodin näyttämiseen käytetään Raphael-kirjastoa, pitää SVG-tiedosto ensin konvertoida js (java script) -tiedostoksi. Tämän jälkeen Modenizr.js -kirjastolla tehdään tarkastus SVG inline-koodin toteutuksesta. Raphael käyttää SVG W3C -suositusta. Sen tarkoitus on tehdä vektorigrafiikan piirtämisestä helppoa selaimesta riippumatta. Graafinen objekti on samalla myös DOM-objekti, minkä takia siihen voi tehdä muutoksia helposti myöhemmin jQueryn avulla. (RaphaelJs 2013)

Esimerkki:

```
if (!Modernizr.inlinesvg)
{  document.write(
   '<script type="text/javascript" src="scripts/raphael.js"><\/script>',
   '<script type="text/javascript" src="scripts/svg.js"><\/script>'
);
}
```

Tämän jälkeen SVG-koodi pitää laittaa html-dokumenttiin div-elementin sisälle.
Esimerkki:

```
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"…;
<path fill="#333333" d="M296.908,120.622c-8.77,6.201-13.158,13.676-
13.158,22.41c0,10.458,5.425,18.479,16.262,24.076
  c-2.908,8.435-7.122,15.764-12.65,22.009c-5.516,6.243-10.553,9.368-15.11,9.368c-
2.147,0-5.075-0.718-8.794-2.133l-1.782-0.687  c-3.646-1.416-6.854-2.133-9.656-2.133c-
2.641,0-5.535,0.555-8.679,1.665l-2.237,0.807l-2.818,1.154   c-2.218,0.884-4.468,1.326-
6.725,1.326c-5.328,0-11.208-4.387-17.642-13.161c-9.273-12.567-13.905-26.264-13.905-
41.085  "/>
</svg>
```

Browsermark 2.1 SVG-testissä päätettiin käyttää apuna Keith Woodin jQuery SVG – kirjastoa, joka on käytettävissä MIT-lisenssillä. Kirjasto on Keith Woodin kehittämä plugin SVG:n manipuloimisee JavaSciptillä (Keith Wood jQuery SVG 2013). Valitsimme kyseisen kirjaston, koska testi ei vaadi kaikkein kehittyneimpiä SVG-grafiikan toimintoja.

4.4.4 MIT-lisenssi

MIT-lisenssin ehdot ovat yksinkertaiset. Se antaa käyttäjälle oikeudet vapaasti muokata, kopioida ja käyttää teosta omassa projektissa sillä ehdolla, että lisenssin teksti säilyy lähdekoodissa. Koska MIT-lisenssi ei ole copyleft-lisenssi, se ei vaadi lähdekoodin julkistamista vaikka muokattua teosta levitettäisiin eteenpäin. (Wikipedia MIT Lisenssi 2013)

4.5 SVG-testin tuotteistaminen

SVG-testin suunnittelun jälkeen suunnitelman pohjalta luodun rakenteen avulla testistä luotiin pseudokoodi, joka auttaa ymmärtämään testin toimintaa ja havainnoi selkokielisesti, miten testi toimii kokonaisuudessaan.

Pseudokoodaaminen on tapa kuvata ohjelmallisia ongelmia käyttämällä esimerkiksi suoraan ongelmaa tai toimintaa kuvaava lausetta. Kun SVG-testin pseudokoodi saatiin valmiiksi, jouduttiin flow charttia päivittämään, koska toiminnan tarkastaminen toi esille uusien toiminnallisten luokkien tarpeen. Nämä lisättiin lopulliseen flow charttiin ja varsinaisen tuotteen koodaaminen aloitettiin JavaScript-kielellä.

5 Testaus

Testaus-osiossa käydään läpi testin testaussuunnitelma ja käytännöt. Osiossa on mukana liitteinä testin testaustulokset.

5.1 Testauskäytännöt ja suunnitelma

Testaussuunnitelma kuvaa SVG-testin toimintaa. Suunnitelma perustuu SVG-testin suunnittelussa luotuun vaatimusmäärittelyyn. Testauksen tavoitteena on varmistaa, että SVG-testin toiminnot ovat suunniteltu ja toteutettu oikein. SVG-testin dynaamisen testauksen integraatiotestausosuus on tehty testin ohjelmoimisen aikana kevyitä TDD (Test-driven development)-menetelmiä käyttäen. Testin valmistuttua suoritetaan järjestelmätestaus lokaalissa- ja tuotantoympäristössä.

5.2 Standalone-version testaus lokaalissa ympäristössä

Projektin suunnittelun jälkeen pseudokoodin ja testin flow chartin avulla testistä tehdään standalone-testi lokaaliin ympäristöön. Tämä versio testataan kaikilla selaimilla ennen virallista julkaisua. Testaamalla tuote tässä vaiheessa varmistetaan, ettei testissä sellaisia vikoja tai poikkeamia, joita pitää korjata tai uudelleen arvioida ennen julkaisua. Benchmarking-testeissä on tärkeää, että testin toiminta on tasapuolista kaikilla selaimilla ja että testi on selainriippumaton ja tasapuolinen pistejaossaan. Testaus suoritetaan ajamalla standalone-testi Chrome, Opera, Safari ja IE 10 -selaimilla. Jokaisella testikerralla otetaan testin pisteluku ylös ja testi suoritetaan jokaisella selaimella viisi kertaa. Testien tuloksia verrataan toisiinsa ja tuloksista pyritään päättelemään vaikuttaako selaimen oman toiminnan vaihtelu pisteisiin. Selaimen vaihtelut saa vaikuttaa tuloksen sisällä niin, että pienimmän ja suurimman tuloksen väli saa olla korkeinaan 10 %. Testaamalla pyritään selvittämään myös onko tasapainotulos kodillaan. Standalone-testillä varmistetaan onko varianssi tarpeeksi pieni, vai pitääkö testiin tehdä tässä vaiheessa muutoksia.

5.2.1 Standalone-testin raportit

Standalone-testin raportit on laitettu liitteeksi, koska se on todettu salaiseksi yrityksen NDA (Non-disclosure agreement) -sopimuksen mukaisesti

LIITE1_browsermark_21_svg_stand-alone_test_report_WINDOWS8.xlsx
LIITE1_browsermark_21_svg_stand-alone_test_report_IOS.xlsx

5.3 SVG-server -puolen testaus lokaalissa ympäristössä Browsermark-alustalla

Standalone-testauksen jälkeen tehdään vielä uusi testauskierros, kun tuote on julkaistu ja laitettu testiserverille. Tällä testillä varmistetaan, että testi toimii halutulla tavalla myös tuotantoserverillä. Lopullista tuotetta testataan 11 kertaa, koska varsinaisessa järjestelmässä testiin tulee useampia muuttujia. Näitä ovat muun muuassa kirjastot, jotka on ladattuna serverille. Vaikka kirjastot ovat vain ladattuna serverille eikä niiden pitäisi vaikuttaa testituloksiin, on tämä asia varmistettava testaamalla tuotetta. Näin testin tulosten luotettavuus säilyy.

Tässä lokaalin ympäristön testausvaiheessa havaitsimme, että testi ei toimi enää FireFox-selaimessa. Vikaa selvittäessä huomattiin, että FireFox seuraa kahta standardia, WAHTWG ja W3C-stantardia. WAHTWG-standardi mahdollistaa document.domain -arvon asettamisen, kun taas W3C:n SVG-stantardeja määrittelee document.domain -arvon vain lukumuotoiseksi. Koska Browsermark-alustalla testin document.domain pitää aina olla sama, jouduimme uudelleenorganisoimaan testien järjestystä siten, että SVG-testi suoritettiin ennen testausvaihetta, jossa document.domainin käyttö on vaadittu.

5.3.1 SVG-server -puolen testaus lokaalissa ympäristössä

SVG-server puolen lokaalinympäristön testausraportit on laitettu liitteeksi, koska se on todettu salaiseksi yrityksen NDA (Non-disclosure agreement) -sopimuksen mukaisesti

LIITE2_browsermark_21_svg_implemented_test_report_IOS.xlsx
LIITE2_browsermark_21_svg_implemented_test_report_WINDOWS8.xlsx

5.4 SVG-betaserver -testaus

Kun testille on tehty ensimmäinen standalone-testi lokaalissa ympäristössä ja testaus Browsermark-alustalla lokaalissa ympäristössä, tehdään vielä yksi testauskierros beta-serverillä, jolla päästään mahdollisimman lähelle tuotantopalvelimen toimintaa. Tämä testaus on välttämätön uutta Internet Explorer 10 -versiota varten, koska IE uudella versiolla 10 ei voi tehdä lokaalintason testejä ilman suurta selaimen konfiguraatoiden säätöä. Näiden vaikeuksien takia testituloksen luotettavuudesta ei voitu olla varmoja. Lokaalin tason testauksesta luovuttiin ja testausta painotettiin betaserver-tasolla.

Ongelmat johtuivat uuden Internet Explorer 10 -selaimen suojauspäivityksistä, jotka ovat niin rajoittavia, että lokaalin serverin käynnistäminen on käytännössä mahdotonta sillä. Kokonaista testiä ei saatu koskaan pyörimään onnistuneesti näiden vaikeuksien takia. Ongelman voi kuitenkin kiertää tekemällä testi beta-palvelinympäristössä. Beta-palvelin suojattiin HTACCESS-käyttäjätunnuksella, joten ulkopuolisilla ei ole mahdollisuutta päästä näkemään testiä ennen julkaisua. Testejä tehdään tässäkin ympäristössä 11 kappaletta per selain, jotta varmistetaan, että testi toimii varmasti myös muiden testien kanssa kokonaisuutena eikä vain yksittäisenä testinä.

5.4.1 SVG-betaserver -testauksen raportit

SVG-betaserver -testausraportit on laitettu liitteeksi, koska se on todettu salaiseksi yrityksen NDA (Non-disclosure agreement) -sopimuksen mukaisesti

LIITE3_browsermark_21_svg_implemented_live_beta_IOS.xlsx
LIITE3_browsermark_21_svg_implemented_live_beta_WINDOWS8.xlsx

5.4.2 Browsermark 2.1 -testikokonaisuuden testaus

Viimeisin testi Browsermark 2.1 on koko testikokonaisuuden testaus isoimmilla selaimilla. Selaimessa ajetaan kokonainen testi viisi kertaa per selain. Testin metatiedot otetaan ylös ja näiden avulla voidaan määrittää tarkka pistelaskun tasapainopiste.

6 Tuotteen julkaisu

Browsermark 2.1 julkaistiin 31. lokakuuta 2013 kun Browsermark 2.1 testikokonaisuudelle oli tehty kattavat lokaalin tason testit. Betaserver-testit tehdään Rightware Oy:n sisäisesti, jolloin testauksessa Rightwaren Oy:n työtekijät pääsevät testaamaan uutta tuotetta ja kertomaan siitä mielipiteensä. Tällä tavalla tuotteesta saadaan IT-alan teknisesti valveutuneiden käyttäjien mahdollisesti huomaamat ohjelmalliset bugit jo talon sisällä esille ja siten myös korjattua. Tämän vaiheen jälkeen annetaan asiakkaille beta access, jossa suoritetaan rajatussa ryhmässä tuotteen testaaminen. Vasta näiden vaiheiden jälkeen virallinen tuote julkaistaan kaikkien käytettäväksi

7 Projektin päätös

Opinnäytetyö SVG-testin toimeksiannon osalta oli valmis jo ennen kuin tuote julkaistiin. Projekti suoritettiin projektisuunnitelman mukaisesti ja se valmistui hieman etuajassa. Projektin alussa suunniteltuihin oppimistavoitteisiin päästiin. Projektin myötä SVG-teknologian käyttö web-suunnittelussa ja toteutuksessa osoittautui erittäin hyödylliseksi työvälineeksi. SVG-teknologiaa voidaan hyödyntää kuitenkin parhaiten iconien ja logojen käytössä.

Vektorigrafiikka ei hävitä laatuaan vaikka sitä pienennetään tai suurennetaan, minkä takia se sopii hyvin responsiivisen sivun luomiseen, missä sivua pitää skaalata pienestä koosta suurempaan.

8 Oman oppimisen arviointi

Projektia suunnitellessa määrittelin oppimistavoitteeksi ymmärtää paremmin SVG-teknologian soveltamista websivuihin, becnhmarkkauksen toimintaa ja ideologiaa, Rightware Oy Browsermark-tuotteen toimintaa sekä miten tuote toimii ansaintamallin perustana. Oppimistavoitteina oli myös parantaa ohjelmistosuunnittelun osaamista ja koodauksen perusasioita.

8.1 Saavutetut tulokset

Testin rakentamisen jälkeen SVG-toiminnan ymmärtäminen web-sivuilla selkeni huomattavasti. SVG-grafiikan käyttö web-suunnittelussa tulee saavuttamaan enemmän huomiota tulevaisuudessa kun ajatellaan sivun skaalatutuvuutta erikokoisille laitteille. Tämä on myös testin kannalta tärkeä asia, koska sillä varmistetaan testin hyödyllisyys selainvalmistajille. Opinnäytetyön aikana ymmärrykseni benchmarkkauksesta selkeni. SVG-testin tuotteistamisen aikana Browesermark-testi alusta tuli tutuksi ja ymmärrän nyt myös paremmin CMS-rakenteen toimintaa.

Toimeksianto ja opinnäytetyö kokonaisuutena selkeyttivät ohjelmistosuunnittelun osaamistani ja perustason ohjelmoinnin osaaminen parantui.

Oppimistavoitteisiin päästiin kokonaisuutena ajatellen erittäin hyvin.

9 Lähteet

- W3C SVG Graphics
  http://www.w3.org/Graphics/SVG/

- Wikipedia 2013 Benchmark computing
  http://en.wikipedia.org/wiki/Benchmark_(computing)

- Voormedia test tool
  http://voormedia.com/blog/2012/10/creating-svg-vector-graphics-for-maximum-browser-compatibility/svg-browser-test

- RaphaelJs 2013
  http://raphaeljs.com/

- Keith Wood jQuery SVG 2013
  http://keith-wood.name/svg.html

- Wikipedia MIT Lisenssi 2013
  http://fi.wikipedia.org/wiki/MIT-lisenssi

- W3C SVG WG Proposal
  http://dev.w3.org/SVG/proposals/svg-html/svg-html-proposal.html

10 Liitteet

10.1 SVG-testin tuotteistamisen ja suunnittelun liitteet

- LIITE 1: liite_functioiden_kuvaus.docx VOI JULKAISTA
- LIITE 2: Svg_Zoom_Test.docx VOI JULKAISTA
- LIITE 3: SVG Test Flowchart.png SALAINEN
- LIITE 4: svg_test_pseudo_code.docx VOI JULKAISTA
- LIITE 5: liite_tasapainotulos.docx SALAINEN
- LIITE 6: liite_InitJS_alustus.docx SALAINEN

10.2 Testaustulokset

- LIITE1_browsermark_21_svg_stand-alone_test_report_IOS.xlsx SALAINEN
- LIITE1_browsermark_21_svg_stand-alone_test_report_WINDOWS8.xlsx SALAINEN
- LIITE2_browsermark_21_svg_implemented_test_report_IOS.xlsx SALAINEN
- LIITE2_browsermark_21_svg_implemented_test_report_WINDOWS8.xlsx SALAINEN
- LIITE3_browsermark_21_svg_implemented_live_beta_IOS.xlsx SALAINEN
- LIITE3_browsermark_21_svg_implemented_live_beta_WINDOWS8.xlsx SALAINEN

LIITE1_liite_functioiden_kuvaus.docx

SVG init

Init funktio suorittaa ensimmäisen initialize tarkastuksen. Tarkastuksessa tarkastetaan onko testin suorittaminen selaimella mahdollista. Tarkastaminen tehdään käyttämällä modenizr.js kirjastoa. Kirjasto tarkastaa onko selain sellasta selain kantaa joka tukee SVG-stantardia edes perustasolla jotta testi voidaan käynnistää. Jos testin SVG tuki menee tarkastusessa läpi, Init päivittää aloitus näkymäksi zoomatun näkymän paperin yläreunasta ja alustaa SVG-matriisin, alueen, jossa testi itsessään suoritetaan. Jos init-tarkastus ei mene läpi, annetaan testille nolla-pisteen tulos ja testi lopetetaan.

SVG run

Run funktio pyörittää itse testiä. Run funktiossa on ehto, että testiä pyöritetään vain yksitoista kertaa. Funktio laskee ja ylläpitää sivunumeroa jotta tiedetään missä kohtaa testiä kulloinkin ollaan.

SVG scroll

Scroll funktion tarkoitus on testin viimeisen sivun jälkeen rullata sivu ylhäältä alas sivun alareunaan. Vieritys tehdään päivittämällä matriisin animointiarvoa. Samalla päivitetään testin sivunumerointia ja testaus kierrosta jotta run funktio voi tehdä oman tarkastuksen siitä monennesko testikierros on menossa.

SVG calculatescore

Calculatescore funktion tarkoitus on testin suorittamisen jälkeen laskea testin tulos. Elementin paikka tarkastetaan onko se testin viewportin (999x 600) sisällä. Jokaisella kerralla kun elementti on sijoittunut oikein viewportin sisälle, annetaan testille plus piste. Jos elementti taasen on viewportin ulkopuolella pistettä ei anneta lainkaan. Testin päätyttyä kokonais aika testin tekemiseen mitataan ja lasetaan operaatiot per sekuntti (ops). OPS arvoa muokataan kertoimen mukaan ja verrataan ennaltatiedettyyn vertailutulokseen josta saadaan testille lopullinen testitulos. Laskee benchmark-moottoria

varten raakatuloksen käyttäen benchmarkin aikana saavutettuja pisteitä (max. 902 pistettä) ja aikaa (keskiarvoisesti noin 15 sekuntia).

SVG submitResult

Kun calculateScore funktio on laskenyt onnistuneen testin tuloksen, välittä funktio tuloksen submitResult funktiolle. Tämän funktion tarkoitus on välittää saatu tulos tietokantaan talteen. Lähettää lasketun raakapistemäärän + debug-tiedon benchmark-moottorille joka vastaanottaa, käsittelee ja tallentaa tuloksen myöhempää tarkastelua varten

SVG Zoom tausta

Zoom-toiminto on testin yksi keskeisin toiminta. Testin suunnittelussa ei vielä ollut varmaa saadaanko SVG testin zoom toimintoa toimimaan suoraan JavaScrip jQuery kirjastoa käyttämällä. Zoom-toiminnasta tehtiin testi-koodi suoraan jQuerylla, jonka huomasimme ratkaisevan tämän ongelman. Koodi on liitteessä Svg_Zoom_Test.docx

SVG zoomIn

zoomIn funktio suorittaa sisään zoomaus toiminnon testin aikana. Zoom in toiminto zoomaa ensimmäiseen paperiin ja suorittan scroll toiminnon missä paperi luetaan yläreunasta alareunaan. Kasvattaa SVG-matriisin skaalausarvon 1.5 (=150% suurennos)

SVG zoomOut

zoomOut toiminto käynnistyy zoomIn toiminnon jälkeen. zoomOut palauttaa zoomIn function lähentämisen takaisin normaalin tilaan ja asettaa fokuksen takaisin paperin yläreunaan. Pienentää SVG-matriisin skaalausarvon 0.5 (=50% suurennos)

LIITE1_liite_functioiden_kuvaus.docx

SVG Zoom toiminnon testi koodi

LIITE2_Svg_Zoom_Test.docx

SVG Testin flowchart

LIITE3_SVG Test Flowchart.png

SVG Testin pseudo-koodi

LIITE4_svg_test_pseudo_code.docx

LIITE2_Svg_Zoom_Test

```
<html>
<head>
   <meta    charset="utf-8">
   <meta    name="viewport"    content="width=1024">
   <title>SVG</title>
   <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.0.2/jquery.min.js"></script>
   <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.10.3/jquery-
ui.min.js"></script>
   <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/swfobject/2.2/swfobject.js"></script>
   <link href='http://fonts.googleapis.com/css?family=Open+Sans' rel='stylesheet' type='text/css'>
   <script type="text/javascript">
      $(document).ready(function()
      {
         function zoomify()
         {
            var width = $('svg').innerWidth();
            var height = $('svg').innerHeight();

            var matrix = [1,0,0,1,0,0];
            var zoom = Math.random();
            $.each(matrix, function(i)
            {
               matrix[i] *= zoom;
            });
            matrix[4] = (1-zoom)*width/2;
            matrix[5] = (1-zoom)*height/2;
            $('g#svg-matrix').attr('transform', 'matrix(' + matrix.join(' ') + ')');
         }
         setInterval(zoomify, 1000);
      });
   </script>
</head>

<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
   <g id="svg-matrix" transform="matrix(2 0 0 2 0 0)">
      <ellipse cx="240" cy="50" rx="220" ry="30" style="fill:yellow" />
      <ellipse cx="220" cy="50" rx="190" ry="20" style="fill:white" />
   </g>
</svg>
</body>
</html>
```

LIITE4_svg_test_pseudo_code.docx

```
/*
        Author: Erno Tuovinen
        Created: 1.7.2013
        Modified: 9.8.2013
        This is Browsermark SVG test java script pseudo code
*/
```

// Create variables that are used to calculate scores and test rounds.

```
variable round = 0;
variable positionscore = 0;
variable operations = 0;
variable timer = NEW TIMER()
```

// Init function will check is test doable. If it is make test initializations and start run function

```
FUNCTION INIT()
        IF (is doable)
                move crosshair to first page
                timer.start()
                RUN()
        ELSE
                CALCULATESCORE(operations, positionscore)
```

// Run function will calculate how many runs is left and starts zoom functions. If rounds are finished
// calculate scores

```
FUNCTION RUN()
        round++
        IF (round < 11)
                ZOOMOUT()
                ZOOMIN()
                READ()
        ELSE
                timer.stop()
                CALCULATESCORE(operations, positionscore)
```

// Zoomout function does jquery svg zoom 1:1 ratio. When this function starts it also adds point to
// operations that is used to calculate overall score

```
FUNCTION ZOOMOUT()
        operations++
        zoom SVG to 1:1
```

LIITE4_svg_test_pseudo_code.docx

```
/*
        Author: Erno Tuovinen
        Created: 1.7.2013
        Modified: 9.8.2013
        This is Browsermark SVG test java script pseudo code
*/
```

// Zoomin function will make zoom operation to ratio 1:20. Also adds operations points

```
FUNCTION ZOOMIN()
        operations++
        zoom SVG to crosshair 1:20
```

// Read function is used to check element position in page. Add one operation point and is give position
// compensatory points for overall score. Read function will start check is the read element last element.

```
FUNCTION READ()
        operations++
        read element position
        IF (position mismatch)
                positionscore--
        ELSE
                positionscore++
        ISLASTELEMENT()
```

// Islastelement function will do a check is the tested element last element. If it is then start scroll // function

```
FUNCTION ISLASTELEMENT()
        IF (is last element of page)
                ISLASTPAGE()
        ELSE
                SCROLL()
```

// Scroll function will add one point to operations calculator. Scrolls the page down then starts read // function

```
FUNCTION SCROLL()
        operations++
        scroll page down
        READ()
```

LIITE4_svg_test_pseudo_code.docx

```
/*
        Author: Erno Tuovinen
        Created: 1.7.2013
        Modified: 9.8.2013
        This is Browsermark SVG test java script pseudo code
*/
```

// Islastpage function test is tested page last one if it is code will move crosshair to first page and page is // not last page code will move crosshair to next page. After that start run function

```
FUNCTION ISLASTPAGE()
        IF (last page)
                move crosshair to first page
        ELSE
                move crosshair to next page
        RUN()
```

// Calclulatescore function will give test final score. Score is operation per second unit. Function will also // check if operations are zero before calculate overall score

```
FUNCTION CALCULATESCORE(operations, positionscore)
        IF (operations == 0)
                RESULT(0)
        ELSE
                ops = (operations + positionscore) / timer.elapsed
                RESULT(ops)
```

// Results function will have points as a parameter. The point will be submitted to database

```
FUNCTION RESULT(points)
        submit points to database
```