HØGSKOLEN I VESTFOLD
Avdeling for samfunnsfag

Bachelorstudiet i Informatikk

BAC309IN – Bacheloroppgave – 20 studiepoeng

Tittel:

SVG i dagens nettlesermarked

Stikkord om innholdet:

Scalable Vector Graphics, SVG, vektorgrafikk, nettlesere, kompatibilitet, markedspenetrasjon, World Wide Web, W3C, standardisering, endringer i markedstilstand

Oppgaven er skrevet av:

Petter Dahl Thunæs	Robin Smidsrød

Veileder:

Viggo Holmstedt

Semester og årstall oppgaven er levert:	VÅR 2010
Oppgaven er konfidensiell:	Nei ☒ Ja ☐ Tidsrom:

For studieadministrasjonen:

Registrert dato: _____

Sign.: _____

HØGSKOLEN I VESTFOLD
Avdeling for samfunnsfag

Obligatorisk erklæring

Jeg/vi erklærer herved at min/vår

Bacheloroppgave (BAC309IN)

som inngår som en del av **Bachelorstudiet i Informatikk**
ved Høgskolen i Vestfold, Avdeling for samfunnsfag:

1. ikke har vært brukt til samme/en annen eksamen ved Høgskolen i Vestfold eller et annet universitet/høgskole innenlands eller utenlands
2. ikke refererer til andre arbeid uten at dette er oppgitt
3. har oppgitt alle referanser i litteraturlisten
4. har oppgitt alle navn på gruppemedlemmer
5. har vist god nettetikk.

Jeg/vi er kjent med at brudd på disse bestemmelsene er å betrakte som fusk og behandles i hht. §18 i Forskrift om eksamen ved HVE og U-loven § 54.

Ved gruppebesvarelse må alle gruppas deltagere undertegne.

……..……..……..
Sted/dato

....
Petter Dahl Thunæs Robin Smidsrød

SVG i dagens nettlesermarked

1. juni 2010

Bachelor i Informatikk 2009/2010

Høgskolen i Vestfold

Gruppemedlemmer:

Petter Dahl Thunæs
Robin Smidsrød

Veileder: Viggo Holmstedt

Sammendrag

SVG-standarden (Scalable Vector Graphics) har vært tilgjengelig i mange år. Den har likevel ikke total markedspenetrasjon. Vi ønsket å finne ut hvorfor SVG den dag i dag fremdeles ikke kan benyttes av utviklere uten å måtte forholde seg til hvor godt standarden er støttet. Fordi SVG ikke har stor nok markedspenetrasjon så vi også på alternative teknologier.

Vi fant ut at manglende støtte i Internet Explorer var det som hovedsakelig holdt bruken av SVG tilbake, men at alternative teknologier som Flash har fyllt det tomrommet. De alternative teknologiene har flere svakheter i seg som SVG ikke har. Microsofts endringer i strategi fra mai 2010 har endret fremtidsutsiktene for SVG helt og holdent. Fremtidsutsiktene for SVG er nå svært positive.

Innholdsfortegnelse

Forord

A Problemstilling

SVG-standarden (Scalable Vector Graphics) har vært tilgjengelig i mange år. Den har likevel ikke total markedspenetrasjon. Vi ønsker å finne ut hvorfor SVG den dag i dag fremdeles ikke kan benyttes av utviklere uten å måtte forholde seg til hvor godt standarden er støttet. En viktig del av dette er å finne ut hvordan Microsoft har påvirket utvikleres syn på SVG som en moden standard. I tillegg ønsker vi også å finne ut av om SVG fremdeles er den teknologien som anbefales for presentasjon av skalerbar grafikk på World Wide Web, eller om det er kommet andre teknologier på markedet som bedre løser SVG sitt problemområde.

For å illustrere bruk av SVG ønsker vi å lage et bibliotek som genererer diagrammer til forretningsbruk, slik som kakediagram og linjediagram. Vi velger også å lage et tradisjonelt program med grafisk brukergrensesnitt. Dette programmet skal ta imot data fra brukeren og generere SVG-filer ved hjelp av biblioteket omtalt over.

B Spørsmål som dekker grunnlaget for oppgaven

- Introduksjon: Hva er SVG?
- Hvilke nettlesere må støtte SVG for at man som utvikler kan benytte SVG uten å tenke på klient-støtte?
    - Hvor mye av SVG-standarden støtter de forskjellige nettleserne?
        - Hvilke nettlesere støtter standarden i det hele tatt?
        - Er det noe spesifikk funksjonalitet i SVG-standarden som ikke er støttet?
        - Finnes det noen mangler eller andre svakheter i støtten av SVG-standarden i disse nettleserne, f.eks. feil implementasjon?
- Hvorfor støtter ikke Microsoft Internet Explorer SVG-standarden?
    - Hvordan forholder Microsoft seg til W3C sin SVG-standard?
        - Har Microsofts synspunkt endret seg over tid?
    - Hvilke andre teknologier har Microsoft valgt å benytte isteden?
    - Hvilken innvirkning på markedet har Microsoft sitt synspunkt på adopsjon av SVG i sin helhet?
- Hvilke alternative teknologier eksisterer som dekker SVG sin funksjonalitet?
    - Hvordan benytter utviklere Adobe Flash fordi SVG ikke er tilgjengelig?
    - Kan Microsoft SilverLight også benyttes for å omgå mangel på SVG-støtte?
    - Kan JavaScript-bibliotek som [Raphaël http://raphaeljs.com/] være et alternativ til SVG?

        - Hvordan skiller RaphaëlJS sin imperative fokus seg fra SVG sin deklarative fokus?
    - Kunne VML (Vector Markup Lanuage), som Microsoft støtter, vært avansert som en global standard istedenfor SVG?
        - Finnes det proprietære elementer i VML som gjør den uskikket som en åpen standard?
        - Er VML som standard knyttet for sterkt til Microsoft sin implementasjon?
- Konklusjon: Hva må skje i markedet for at SVG skal "ta av" som standard?

C Arbeidsmetodikk

Vi vil bruke en smidig utviklingsmetodikk (agile), hvor vi setter opp korte perioder (sprint), hvor vi jobber på spesifikke problemstillinger. Idéen med smidig utvikling er at etter hver periode sitter man igjen med et system som virker med den funksjonaliteten som var planlagt for perioden. Hver periode begynner med en planleggingsrunde over hva man skal ha med, og avsluttes med versjoner av applikasjon og bibliotek som kan benyttes slik de er. Vi setter opp til en felles dag i uken med arbeid i plenum og jobber hver for oss de resterende dagene. Vi kommer til å bruke 3 uker pr. periode. Oppgaver som ikke er utført i en periode på grunn av manglende tid vil bli overført til neste periode. Start og slutt på en periode legges til felles arbeidsdager i den gjeldende uken. All dokumentasjon av programvaren benytter engelsk som språk. Dette gjør det enklere for tredjeparter å evaluere koden i prosjektet uavhengig av norsk språkforståelse.

D Modellapplikasjon

D.1 Bibliotek: SVG-generator

Vi skal lage et Java-bibliotek som gjør det mulig å generere SVG-filer basert på instanser med strukturerte data. Vi velger å benytte mønstre for god programdesign slik at biblioteket gjør det enkelt å utvide biblioteket til å støtte mer funksjonalitet på et senere tidspunkt. Vi planlegger å benytte white-box reuse-konseptet for å oppnå høy fleksibilitet.

D.2 Applikasjon: Grafisk brukergrensesnitt for håndtering av inndata

Her planlegger vi å lage en normal applikasjon som kan gi et grensesnitt til brukeren for å skrive inn data for å fylle inn et datasett. Den skal igjen kunne generere SVG-filer basert på dette datasettet. Vi planlegger å benytte mønstre for god programdesign for å gjøre applikasjonen enkel å utvide og lettere å vedlikeholde. Dette vil passe godt sammen med vårt valg om å benytte en smidig utviklingsmetodikk.

D.3 Lisensiering

Kildekoden til biblioteket og applikasjonen velger vi å tilgjengeliggjøre under BSD-lisensen. Dette gjør det mulig for andre utviklere å benytte koden fra denne bacheloroppgaven i kommersielle sammenhenger.

E Valg av verktøy til prosessen

Vi velger å benytte oss av Eclipse på grunn av god erfaring med denne utviklingsplatformen. Siden vi har opparbeidet oss god erfaring med bruk av Java velger vi å benytte dette programmeringsspråket til modellapplikasjonen.

For å koordinere arbeidsoppgavene i prosjektet velger vi å benytte LiquidPlanner.com (LP) som er et svært godt planleggingsverktøy som er gratis tilgjenglig for akademisk bruk. LP er svært godt egnet til en smidig utviklingsmetodikk.

Versjonskontroll er viktig i større prosjekter. For å dekke det behovet velger vi å benytte Git og github.com[1]. Git gir oss gode muligheter til å jobbe på flere deler av prosjektet samtidig uten frykt for integrasjonsproblemer. Denne muligheten kommer fra det faktum at Git er et distribuert revisjonskontrollsystem.

Grunnen til at vi velger GitHub.com som tjenesteleverandør er at de gir gratis tilgang til prosjekter som klassifiseres som fri programvare. De er en seriøs aktør på markedet med svært god stabilitet og sikkerhet. I tillegg har brukergrensesnittet deres mange gode funksjoner som øker effektiviteten i en distribuert utviklingsarbeidsflyt. Dersom noen utviklere i fremtiden ønsker å videreutvikle prosjektet gir GitHub dem en enkel mulighet til å gjøre det.

1 Introduksjon til SVG

1.1 Hva er SVG?

Scalable Vector Graphics (SVG[2]) er en åpen standard utgitt av World Wide Web Consortium (W3C[3]) som skal dekke behovet i markedet for å kunne representere to-dimensjonal grafikk på et utvalg av medier og enheter. Man kan si at det er et programmeringspråk for å beskrive to-dimensjonal grafikk og grafiske applikasjoner. Med to-dimensjonal grafikk snakker vi om vektoriserte figurer, raster-grafikk (bitmap) og tekst-elementer.

SVG baserer seg på mange andre åpne standarder og prøver å gjenbruke de[4] så mye som mulig for å unngå overlappende standardiseringsarbeid, både innenfor og utenfor W3C-prosessen.

SVG-standarden er implementert som en applikasjon i XML[5]. Det er en deklarativ[6] standard på lik linje med HTML[7]. Hvordan SVG fysisk representeres i XML omtales i seksjon 1.4.

SVG har støtte for scripting, noe som gjør det mulig å lage avanserte brukergrensesnitt og interaksjoner basert på kjente programmeringsprinsipper benyttet i tradisjonell web-design. SVG benytter ECMAScript[8] som sitt scriptspråk, noe som gjør det lett for en web-utvikler å benytte eksisterende kunnskap til å gjøre SVG-dokumenter interaktive. SVG benytter også DOM-standarden[9], på linje med HTML og XML, for å kunne adressere og manipulere individuelle elementer i et SVG-dokument fra et script.

SVG har også støtte for animasjon. Dette gjøres med notasjon basert på SMIL-standarden[10]. Hvordan SMIL benyttes for å animere et element blir omtalt i seksjon 1.4. Animasjon basert på SMIL er deklarativ, mens animasjon basert på bruk av script-støtten i SVG er prosedyre-basert. Den deklarative notasjonen med SMIL er noe som brukere som har erfaring med tradisjonell animasjon sannsynligvis vil kjenne igjen fordi det baserer seg på tidskoder, synkronisering av aktiviteter over en tidslinje med mer. Tradisjonelle programmerere, derimot, vil sannsynligvis føle seg mer bekvemme med å bruke script-støtten for å utføre en tilsvarende animasjon. Dette betyr i praksis at SVG-standarden er fleksibel nok til at brukere med forskjellige teknologisk utgangspunkt kan ta den i bruk og benytte de delene som er nødvendig for dem.

W3C forklarer selv hva SVG er på denne måten[11]:

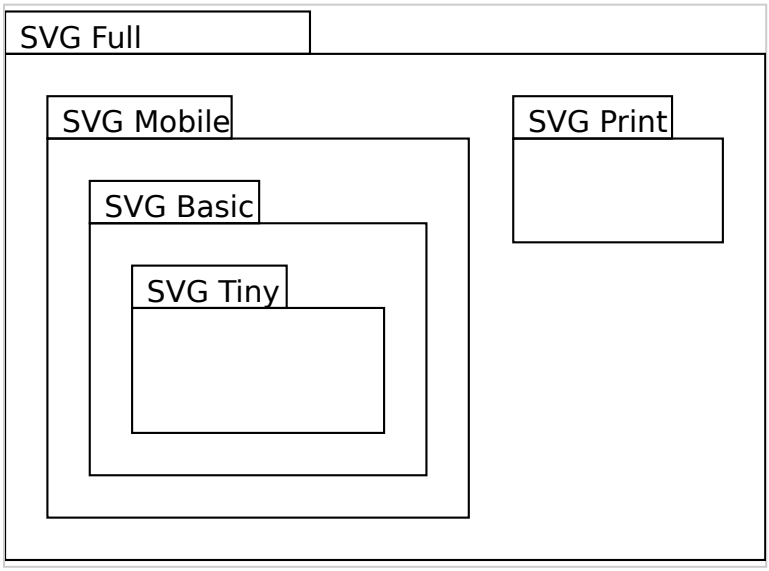
> *Scalable Vector Graphics (SVG) is like HTML for graphics. It is a markup language for describing all aspects of an image or Web application, from the geometry of shapes, to the styling of text and shapes, to animation, to multimedia presentations including video and audio. It is fully interactive, and includes a scriptable DOM as well as declarative animation (via the SMIL specification). It supports a wide*

range of visual features such as gradients, opacity, filters, clipping, and masking.

The use of SVG allows fully scalable, smooth, reusable graphics, from simple graphics to enhance HTML pages, to fully interactive chart and data visualization, to games, to standalone high-quality static images. SVG is natively supported by most modern browsers (with plugins to allow its use on all browsers), and is widely available on mobile devices and set-top boxes. All major vector graphics drawing tools import and export SVG, and they can also be generated through client-side or server-side scripting languages.

1.1.2 Forskjellige SVG profiler til ulike formål

Det finnes flere forskjellige profiler av SVG[12] som beskriver deler av spesifikasjonen. Disse forskjellige profilene er utviklet slik at leverandører av programvare og enheter kan støtte et del-sett av SVG sin totale funksjonalitet og fremdeles være kompatibel med en gitt profil. Under kan du se en oversikt over de forskjellige profilene som er definert.



SVG Mobile-profilen (som ombefatter både SVG Tiny og SVG Basic) ble adoptert av 3GPP[13], en samling av mobiloperatører, som grunnlaget for deres støtte for grafikk i mobile enheter. Flere av disse mobiloperatørene jobber aktivt med å videreutvikle SVG Mobile-profilen for å dekke deres behov på mobile enheter.

SVG Print-profilen fokuserer på hvordan SVG kan benyttes til å gi høyoppløselige utskrifter av grafikk på papir og andre ikke-bevegelige medier. Animasjon og andre interaktive elementer i SVG er naturligvis ikke med i denne profilen da det ikke lar seg reprodusere i statisk format. Canon, HP, Adobe og Corel er aktive medlemmer som jobber med videreutvikling av denne profilen.

SVG Full er den profilen som dekker opp for alle de andre variantene. Hvis man hevder at en applikasjon støtter SVG Full må den også støtte alle de andre profilene, fordi de andre profilene bare er del-sett av SVG Full.

1.2 Medlemmer i SVG Working Group

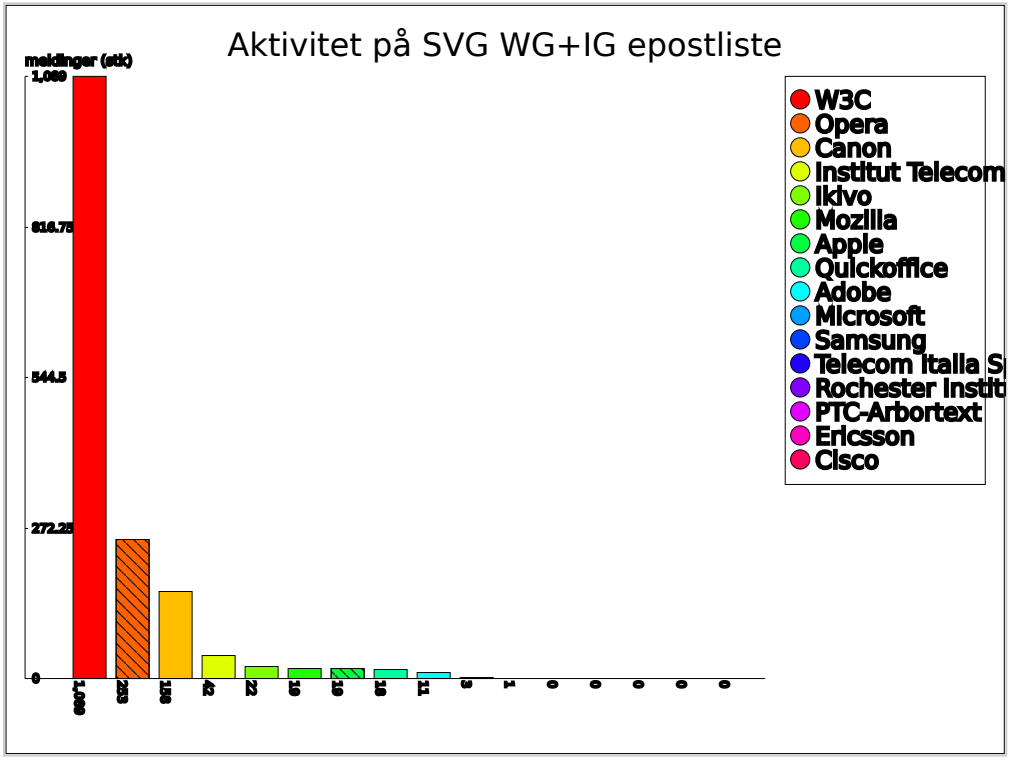
De følgende institusjoner er registrert med et eller flere medlemmer i SVG Working Group (WG) pr. 7. mars 2010[14].

- Adobe Systems Inc.
- Apple, Inc.
- Canon, Inc.
- Cisco
- ERICSSON
- Ikivo AB
- INSTITUT TELECOM
- Microsoft Corporation
- Mozilla Foundation
- Opera Software
- PTC-Arbortext
- Quickoffice Inc
- Rochester Institute of Technology
- Samsung Electronics Co., Ltd.
- Telecom Italia SpA

Ved å analysere epost-listene public-svg-wg[15] og public-svg-ig[16] kommer vi frem til følgende oversikt over aktivitet fra de forskjellige medlemsorganisasjoner:

Organisasjon	**Ant. meld. til WG+IG**	**Andel**
W3C	1089	66%
Opera	253	15%
Canon	158	9%
Institut Telecom	42	2%
Ikivo	22	1%
Mozilla	19	1%
Apple	19	1%
Quickoffice	18	1%
Adobe	11	0%
Microsoft	3	0%
Samsung	1	0%

Organisasjon	Ant. meld. til WG+IG	Andel
Telecom Italia SpA	0	0%
Rochester Institute of Technology	0	0%
PTC-Arbortext	0	0%
Ericsson	0	0%
Cisco	0	0%



Dessverre ble epost-listene til SVG WG først gjort offentlig i 2008, noe som fører til at denne oversikten over aktivitet ikke gjenspeiler en helhetlig oversikt over all kommunikasjon som har forekommet internt i SVG WG, men kun for de siste to årene. Det er uansett interessant å se på disse dataene, da de gjenspeiler interesse fra de forskjellige organisasjonene som står som medlemmer.

Det som umiddelbart viser seg er at W3C internt står for mesteparten av aktiviteten rundt standardiseringsarbeidet, hele 2/3 av all aktiviteten i perioden tallene er hentet fra.

Opera viser seg som den mest aktive nettleser-leverandøren med 15% aktivitet. Deres fokus på SVG gjenspeiler seg også i kvaliteten på deres implementasjon av SVG i deres nettleser, noe som blir mer omtalt i kapittel 2. Mozilla, Apple og Microsoft derimot viser ikke altfor stor aktivitet i gruppen, noe som igjen kan gjenspeiles i deres implementasjoner (mer informasjon om forskjeller i implementasjonene omtales i kapittel 2).

Canon viser et ganske høyt aktivitetsnivå, noe som overrasker, siden de er et firma som hovedsaklig baserer seg på produkter som driver med raster-teknologi (fotografi og utskrift). Adobe, som også driver med grafiske applikasjoner, viser helt klart et lavere aktivitetsnivå, noe som kan bety at de ikke er fullt så fokusert på standardisering pga. deres høye markedsandel i deres segment.

Institut Télécom[17] er et firma som driver med trening og utdanning av fagpersoner innen telekom-sektoren. Deres aktivitet viser at de er interessert i åpne standarder, og at vektorisert grafikk er et viktig område også for telekom-sektoren. Ikivo AB[18] er et selskap som fokuserer på multimediale applikasjoner for mobile enheter. Deres aktivitet viser at de ser på SVG som en viktig standard for mobile enheter. De fremhever spesifikt på sin informasjonsside[19] at de benytter SVG Mobile 1.2 og andre åpne standarder i deres *Enrich*-produkter.

De resterende medlemmene viser liten til ingen aktivitet i arbeidsgruppen, noe som kan bety at de ikke interesserer seg fullt så mye for videre utbedring av SVG-standarden, eller at de benytter andre måter å kommunisere med de andre medlemmene i arbeidsgruppen.

Som vår epost til SVG WG viser, har vi prøvd å få tak i mer informasjon om aktiviteten før 2008, men foreløpig (2010-01-04) er det uvisst hvorvidt vi vil få tilgang til disse opplysningene. Hvis vi får opplysningene kan vi gjøre en større analyse av aktiviteten internt i arbeidsgruppen for å se hvilke organisasjoner som har bidratt mest til at standarden har utviklet seg. Hvis ikke vi får disse opplysningene må vi basere våre funn på de offentlige dataene gjengitt ovenfor.

1.3 Historisk bakgrunn

Med utgangspunkt i dokumentet The Secret Origin of SVG[20] er det tydelig at mange organisasjoner var svært interessert i å få en enhetlig standard for vektorgrafikk på internett i midten av 1990-tallet. Det eksisterte mange forskjellige varianter, men rundt 1998 var det disse fem standardene som sto igjen som de mest sannsynlige til å ta over markedet som en åpen standard for vektorgrafikk: PGML[21], VML[22], HGML[23], DrawML[24] og WebCGM[25].

På grunnlag av disse forskjellige forslagene til standarder ble det avgjort å stifte en arbeidsgruppe (SVG WG) for å utvikle en ny standard for vektorgrafikk. Standarden ble utviklet fra bunnen av med erfaringer fra de overnevnte forslagene.

SVG sin kompakte syntaks for å beskrive stier (path data) arves i stor grad fra VML. Dette ble bestemt etter undersøkelser som viste at den kompakte syntaksen hadde en stor innvirkning på filstørrelse, både med og uten komprimering. Fra PGML (som igjen var basert på PostScript og PDF) benyttet man konsepter rundt koordinatsystemet, transformasjoner, fargerom og tekst/skrifttyper. En av tingene som er interessant å legge merke til er at det ble tidlig bestemt at variabler som beskriver hvordan elementer oppfører seg ble kodet

som XML-attributter, ikke som element-innhold. Dette gjør det enklere å tolke innholdet i et SVG-dokument for f.eks. søkemotorer og annen programvare som håndterer XML på et generelt nivå. XML-attributtene blir i en slik sammenheng ofte ignorert og man sitter igjen med tekst-innhold som beskriver det faktiske innholdet i dokumentet istedenfor formateringen. Siden SVG også støtter hyperlenking (både internt og eksternt) kan man lenke til andre dokumenter like enkelt som i HTML. Dette gjør formatet godt egnet til å formidle informasjon som kan konsumeres av både mennesker og maskiner på en enkel måte.

Tidlig i utviklingen av SVG-standarden var det W3Cs Chris Lilley og Adobes Jon Ferraiolo som var mest aktive. Som forrige avsnitt viser har Adobe sin aktivitet synket noe siden den gang.

1.3.1 Revisjoner og profiler av SVG-standarden

SVG 1.0[26] ble en godkjent W3C standard 4. september 2001.

Forfatter-listen viser at følgende organisasjoner var involvert: Adobe, Apple, Autodesk, BitFlash, Canon, Corel, Excosoft, Hewlett-Packard, IBM, ILOG, IntraNet Solutions, Kodak, Lexica, Macromedia, Microsoft, Netscape, OASIS, Opera, Oxford Brookes University, Quark, RAL (CCLRC), Sun Microsystems, Visio, W3C og Xerox.

SVG 1.1[27] ble en godkjent W3C standard 14. januar 2003.

Forfatter-listen viser at følgende organisasjoner nå også er involvert i tillegg til de som nevnes for versjon 1.0: AGFA, America Online, Ericsson, Expway, Fuchsia Design, KDDI Research Labs, Nokia, Openwave, Savage Software, Schema Software, Sharp og Zoomon.

SVG 1.1 er først og fremst en ny versjon av standarden som modulariserer standarden, slik at forskjellige profiler av standarden nå kan implementeres. Dette baner vei for definisjon av subsettene SVG Tiny, Basic og Print.

SVG Mobile (Basic og Tiny) 1.1[28] ble en godkjent W3C standard 14. januar 2003, samtidig som SVG 1.1 ble godkjent.

SVG Tiny er et direkte subsett av SVG Basic som igjen er et direkte subsett av SVG 1.1. Dette gjør det enklere for utviklere å implementere løsninger som ikke trenger å støtte hele SVG 1.1, men bare de mindre tunge profilene SVG Basic eller SVG Tiny. SVG Tiny ble opprinnelig utviklet for mobil-telefoner og SVG Basic var beregnet på mobile håndholdte enheter (PDAer).

Forfatter-listen her viser flere interessante funn. Ikke alle som står oppført på forfatter-listen for SVG 1.1 er oppført på listen over forfattere for SVG Mobile. Dette kan bety at SVG-arbeidsgruppen har fordelt arbeidet med de forskjellige standardiseringsjobbene basert på deres interesseområde.

Følgende selskaper er ikke involvert i det hele tatt i SVG Mobile: Apple, Autodesk, Excosoft, IBM, Lexica, Macromedia, Microsoft, Netscape, Opera, Oxford Brookes University, RAL(CCLRC), Visio og Xerox.

Det som er overraskende med denne oversikten er at Apple og Opera ikke har involvert seg nok i arbeidet med den mobile SVG-standarden, selv om de helt klart er veldig fokusert på det mobile markedet. Det er mulig at både Apple og Opera har fokusert på SVG Full istedenfor Mobile-profilen fordi deres nettlesere faktisk prøver å støtte den fulle SVG-standarden uansett enhet nettleseren kjører på.

SVG Tiny 1.2[29] ble en godkjent W3C standard 22. desember 2008.

Forfatter-listen viser at enda flere organisasjoner har kommet til: France Telecom, Groupe des Ecoles des Télécommunications (GET), Motorola, OpenText, Quickoffice, Research in Motion (RIM), Samsung, SAP, Streamezzo, Telecom Italia og Vodafone.

I tillegg har flere organisasjoner utvidet antall personer de har involvert i arbeidsgruppen, noe som helt klart tyder på at standarden begynner å få større markedsandeler og blir benyttet mer. Det som er interessant å legge merke til er at de fleste nye organisasjonene er selskaper som leverer tjenester på mobile enheter eller utvikler mobile enheter.

SVG Tiny 1.2 foretok en endring i måten de forskjellige profilene i SVG-standarden påvirker hverandre. SVG Tiny 1.2 er definert som et subsett av SVG 1.1, men med ny funksjonalitet.

SVG Full 1.2[30] er enda ikke godkjent som en W3C standard, men ble sist oppdatert 13. april 2005.

SVG Full 1.2 er ment som den komplette videreføringen av SVG 1.1, som benytter SVG Tiny 1.2 som basis og utvider den standarden for å oppnå full støtte for SVG 1.1 pluss ny funksjonalitet.

Siden standarden enda ikke er godkjent er det uvisst hvilke organisasjoner som vil være involvert i den endelige versjonen av standarden.

SVG Print 1.2[31] er enda ikke godkjent som en W3C standard, men ble sist oppdatert 21. desember 2007.

SVG Print er en ment som en standard for utskrift av SVG-dokumenter. Et program som sier at det støtter SVG Print må teknisk sett støtte SVG Tiny 1.2 sin statiske profil[32]. SVG Print er således en utvidelse av det statiske subsettet av SVG Tiny 1.2.

Siden standarden enda ikke er godkjent er det uvisst hvilke organisasjoner som vil være involvert i den endelige versjonen av standarden. Fra oversikten over redaktører er det tydelig at Canon viser en stor interesse for dette arbeidet.

Det vedlagte dokumentet viser en oversikt over alle forfatterne for de forskjellige revisjonene av SVG-standarden fargekodet med når de involverte seg i arbeidsgruppen.

1.4 Hvordan ser SVG ut?

Som tidligere nevnt er SVG en applikasjon i XML. Det betyr at SVG dokumenter er beskrevet som XML-kode. For at programmer skal skjønne at man jobber med SVG og ikke en generisk form for XML må man sørge for å spesifisere et XML namespace[33] som identifiserer SVG. SVG sitt namespace er definert som `http://www.w3.org/2000/svg` i SVG versjon 1.1[34]. For at nettlesere og andre enheter som benytter HTTP skal kunne identifisere SVG uten å måtte dekode dokumentet bør man benytte mimetypen `image/svg+xml`, som er registrert[35] hos IANA. Hvis XML-dokumentet innholder data fra forskjellige navnerom kan man benytte den mer generiske mimetypen `application/xml`[36]. Tradisjonelt blir filer med SVG-data navngitt med .svg eller .svgz (for komprimert data) som filendelse[37].

SVG inneholder et sett med primitiver som kan benyttes for å lage komplekse figurer i likhet med andre systemer for å tegne vektorisert grafikk. I tillegg støtter den også tekst og vanlig raster-grafikk (bitmaps). I første omgang skal vi se på hvilke primitiver som er tilgjengelig, senere kommer vi inn på bruken av tekst og rastergrafikk.

Følgende primitiver er tilgjengelig[38] (XML-element benyttet i SVG i parentes):

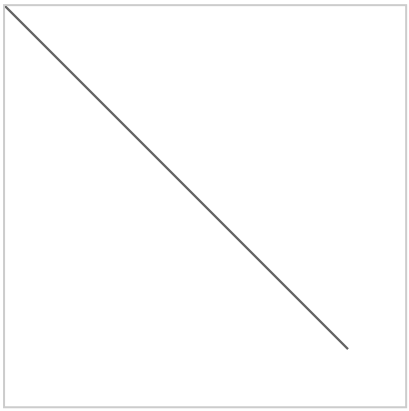
- Linje (line)
- Sammenhengende linjer (polyline)
- Polygon (polygon)
- Rektangel (rect)
- Sti (path)
- Sirkel (circle)
- Ellipse (ellipse)

Nedenfor vil vi gå gjennom hver enkelt primitiv i detalj. Som man kan se i eksemplene er det alltid et rot-element med navn `svg` som definerer synlig område (viewport[39]) for hele tegneflaten. I tillegg er det her man definerer hvilken versjon av SVG-standarden man benytter, evt. hvilken SVG-profil man baserer seg på (`baseProfile`) samt navnerommet, som nevnt over. En ting som er verdt å merke seg er at ingen av eksemplene spesifiserer `width` eller `height`, noe som betyr at viewport skal være så stor som overhodet mulig (dvs. begge verdier har en standard-verdi på 100%).

1.4.1 Linje

Under kan man se et eksempel på line-elementet[40] i SVG.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 350 350">
 <line x1="0" y1="0" x2="300" y2="300"
  style="stroke: rgb(99,99,99); stroke-width: 2;"
 />
</svg>
```



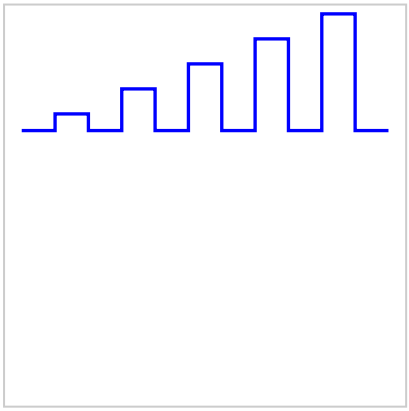
Eksempelet viser en linje fra punkt 0,0 (øverst til venstre) til 300,300. Farge og tykkelse er spesifisert ved hjelp av CSS med attributtene `stroke` og `stroke-width`. Man bruker tradisjonell CSS-syntax innenfor style-attributtet, akkurat som i HTML/XHTML. Det man må være oppmerksom på er at attributtene ikke er de samme som i HTML. Se en oversikt over hvilke CSS-attributter som er gyldige for SVG[41].

1.4.2 Sammenhengende linjer

Under kan man se et eksempel på polyline-elementet[42] i SVG.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 1200 1200">
 <polyline
  fill="none"
  stroke="blue"
  stroke-width="10"
  points="  50,375
            150,375 150,325 250,325 250,375
            350,375 350,250 450,250 450,375
            550,375 550,175 650,175 650,375
            750,375 750,100 850,100 850,375
            950,375 950,25 1050,25 1050,375
            1150,375"
```

```
 />
</svg>
```

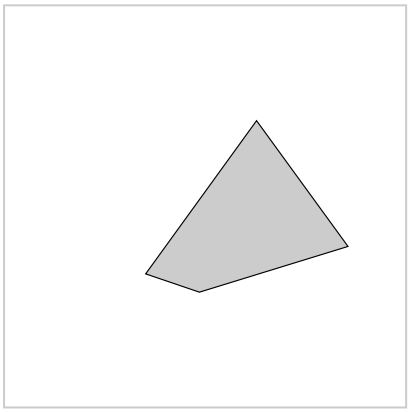


I dette eksempelet ser vi en linje trukket gjennom et sett med punkter spesifisert med `points`-attributtet. Vær oppmerksom på at dersom man skal spesifisere desimaltall for koordinatene *må* man benytte punktum som desimalskille, ikke komma. Kommaet skiller mellom X- og Y-koordinatet mens et eller flere mellomrom skiller mellom koordinatene i listen. Det som også er verdt å legge merke til er at verdiene for å sette tykkelse og farge er her spesifisert med presentasjonsattributter istedenfor CSS-verdier via style-attributtet. Presentasjonsattributter har lavere rang[43] enn CSS-verdier, så hvis begge deler er spesifisert får verdiene som er spesifisert i stilarket prioritet. Fordelen med å bruke stilark er de samme som i HTML, at man kan samle sammen utseende-definisjoner og gjenbruke de på mange elementer. Presentasjonsattributtene arver etter samme regler som i CSS2, men dokumentasjonen for attributtet forteller om det støtter arv eller ikke[44]. SVG-verktøy må ikke støtte CSS, så det er verdt å tenke på at dersom man ønsker et mest mulig kompatibelt dokument, bør man bruke presentasjonsattributter.

1.4.3 Polygon

Under kan man se et eksempel på polygon-elementet[45] i SVG.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 350 350">
 <polygon points="220,100 300,210 170,250 123,234"
  style="fill: #cccccc; stroke: #000000; stroke-width: 1;"
 />
</svg>
```

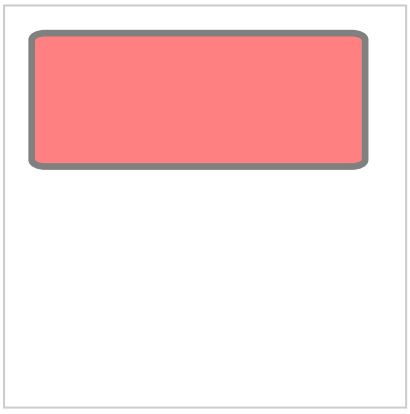


På lik linje med `polyline` benytter man et attributt med navn `points` til å spesifisere hvilke punkter polygonet skal bestå av. I praksis er et polygon det samme som en polyline, men etter det siste koordinatet blir det tegnet en ekstra linje tilbake til første koordinat og hele elementet blir fylt med spesifisert fyll-farge. Legg merke til at fargene er her spesifisert med tradisjonelle hex-koder som i HTML.

1.4.4 Rektangel

Under kan man se et eksempel på rect-elementet[46] i SVG.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 300 300">
 <rect x="20" y="20" rx="10" ry="5" width="250" height="100"
  style="fill: red; stroke: black; stroke-width: 5; opacity: 0.5;"
 />
</svg>
```



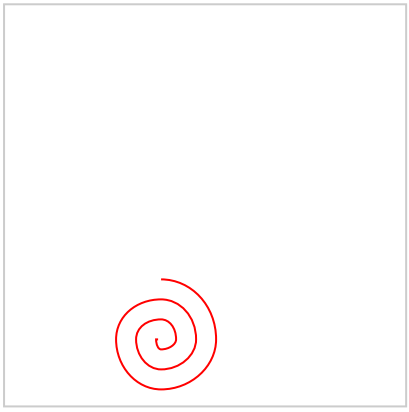
Et rektangel er spesifisert ved å sette `x`, `y`, `width` og `height` attributter som forventet. Attributtene `rx` og `ry` derimot krever litt mer forklaring. De gjør det mulig å få runde hjørner på rektangelet. Verdiene representerer radius i X- og Y-retning for sirkelen som benyttes for å tegne hjørnet. Dersom ingen av disse er spesifisert får man hjørner med en 90 graders vinkel. Det som er viktig å huske

på er at negative verdier for rx/ry/width/height ikke er tillatt. I dette eksempelet ser man også bruk av de konstante navnene i CSS for farger, samt bruk av transparens med `opacity`-attributtet. I dette tilfellet er elementet 50% gjennomskinnelig.

1.4.5 Sti

Under kan man se et eksempel på path-elementet[47] i SVG.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 400 400">
 <path d="M153 334
C153 334 151 334 151 334
C151 339 153 344 156 344
C164 344 171 339 171 334
C171 322 164 314 156 314
C142 314 131 322 131 334
C131 350 142 364 156 364
C175 364 191 350 191 334
C191 311 175 294 156 294
C131 294 111 311 111 334
C111 361 131 384 156 384
C186 384 211 361 211 334
C211 300 186 274 156 274"
  style="fill: white; stroke: red; stroke-width: 2;"
 />
</svg>
```



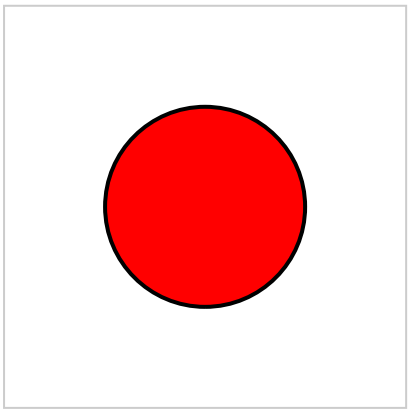
I dette eksempelet ser man en spiral tegnet opp ved å bruke en rekke tegneoperasjoner etter hverandre presentert i et kompakt format. En `M` betyr moveto og setter utgangspunktet for tegneoperasjonen. En `C` betyr curveto og tegner en bézierkurve fra utgangspunktet gjennom to kontrollpunkter til destinasjonspunktet. Destinasjonspunktet blir da det nye utgangspunktet og

tegneoperasjonen fortsetter. Det finnes flere andre tegneoperasjoner[48] man kan benytte til å tegne en sti som vi ikke omtaler her. Det som er interessant å legge merke til er at alle de andre primitivene vi har beskrevet her også kan tegnes ved hjelp av en sti.

1.4.6 Sirkel

Under kan man se et eksempel på circle-elementet[49] i SVG.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 200 200">
 <circle cx="50%" cy="50%"
         r="50"
         stroke="black" stroke-width="2"
         fill="red" />
</svg>
```



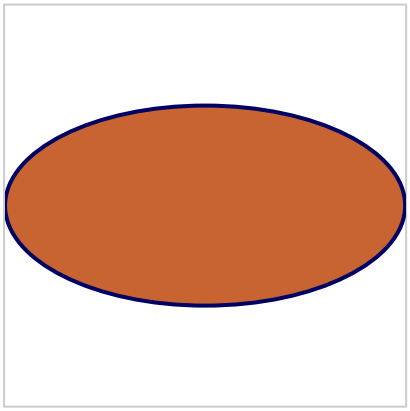
Dette er et ganske enkelt eksempel som viser en sirkel plassert i senter av viewport med en radius på 50 punkter. Bruken av prosent-anvisninger følger samme tankegang som HTML sin boksmodell ved at tallet blir evaluert i forhold til størrelsen på forelder-elementet.

1.4.7 Ellipse

Under kan man se et eksempel på ellipse-elementet[50] i SVG.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 200 200">
 <ellipse cx="50%" cy="50%" rx="50%" ry="25%"
  style="fill: rgb(200,100,50); stroke: rgb(0,0,100); stroke-width: 2;"
```

```
 />
</svg>
```



På lik linje med sirkelen spesifiserer `cx` og `cy` senterpunktet for tegneoperasjonen. Men det som er spesielt med en ellipse er at radius i X- og Y-retningen er forskjellige (i motsetning til sirkelen hvor radius er identisk i X- og Y-retningen). Man må derfor spesifisere både `rx` og `ry` for å tegne ellipsen. Radius kan heller ikke her være negativ, på lik linje med rektangelet.

1.4.8 Tekst

Under kan man se et eksempel på text-elementet[51] i SVG.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 300 300">
 <text x="10" y="20" style="font-family:sans-serif;font-size:24"
  >Stationary text</text>
 <text x="10" y="40" style="font-family:serif;font-size:24">Moving text
  <animateMotion path="M 10 40 L 200 40" dur="0.5s" fill="freeze" />
 </text>
</svg>
```

Eksempelet viser først en tekst-streng plassert tett opp til venstre kant og toppen av tegneområdet. Det er valgt en generisk sans-serif skrifttype i en passende størrelse. Den neste tekst-strengen er plassert 20 piksler lenger nede på skjermen og benytter en serif skrifttype. I tillegg er det spesifisert at selve teksten skal animeres fra utgangspunktet til X-posisjon 200 i løpet av 0.5s. Parameteret *freeze* til fill-attributtet betyr at det animerte elementet skal bli stående i sluttposisjonen[52] når animasjonen er ferdig. Vanligvis flytter elementet som er animert seg tilbake til utgangsposisjonen etter fullført animasjon (fill="remove"). Igjen ser vi bruk av en enkel sti-spesifikasjon for å beskrive bevegelsen i animasjonen.

1.4.9 Filter

Under kan man se et eksempel på bruk av filtre[53] i SVG.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
            xmlns:xlink="http://www.w3.org/1999/xlink"
     viewBox="0 0 300 300">
 <defs>
  <filter id="shadow">
   <feGaussianBlur in="SourceGraphic" stdDeviation="2" result="blur" />
   <feOffset in="blur" dx="3" dy="3" />
  </filter>
  <text id="text"
   x="10%" y="10%"
   font-family="sans-serif" font-size="150%"
   >Text with drop shadow
  </text>
 </defs>
 <use xlink:href="#text" filter="url(#shadow)" stroke="grey" />
 <use xlink:href="#text" stroke="black" />
</svg>
```

Eksempelet viser flere avanserte metoder i SVG for gjenbruk av definisjoner uten behov for duplisering. La oss gå gjennom hver del steg for steg.

Det første man legger merke til er at det er lagt til navnerommet for XLink[54], noe som gjør det mulig å referere til andre elementer ved hjelp av URI-referanser. Selv om URI-referansene er interne må man benytte xlink:href-attributtet for å peke til ressursene.

Det neste vi legger merke til er `<defs>`-elementet[55]. Dette er et samlingselement for andre elementer, på lik linje med `<g>`-elementet. Det som er spesielt med `defs` er at alle definisjonene innenfor elementet ikke blir tegnet opp umiddelbart. Men siden de kan refereres til senere i dokumentet gjør det gjenbruk kjapt og enkelt å få til. Hvis vi hopper over de faktiske definisjonene kan vi se at begge `<use>`-elementene[56] henviser til et internt element med id="text".

Hvis vi tar en kikk på `<text>`-elementet innenfor definisjonen ovenfor finner vi igjen det refererte id-attributtet. Tekst-elementet blir her tegnet opp to ganger, først en gang med en grå tegnefarge, og deretter med en sort tegnefarge. Man kan også se at den grå teksten har et filter aktivert. Hvis ikke filteret hadde vært benyttet ville den sorte teksten tegnet rett over den grå teksten som da ikke ville blitt synlig. Men siden filteret er aktivert påvirker det den grå teksten før den tegnes ut. Det er verdt å legge merke til at href-definisjonen benytter lenkesyntaks fra HTML sitt `<a>`-element, mens filter-attributtet benytter CSS-syntaks for å referere til en ressurs.

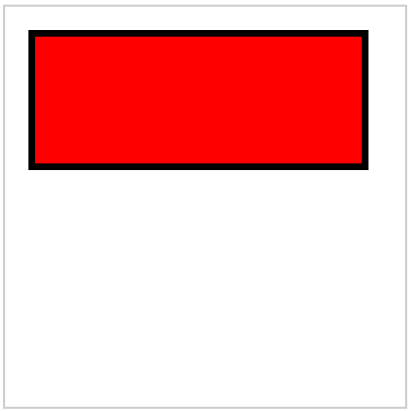
Hvis vi nå tar en kikk på `<filter>`-elementet innenfor definisjonsblokken ser vi at filteret består av en gaussian blur-effekt[57] pluss en offset-effekt[58]. Blur-effekten bruker kildegrafikken, utfører en blur-operasjon på størrelse 2 og lagrer resultatet i en midlertidig buffer som navngis **blur**. Offset-effekten benytter da denne midlertidige bufferen som kilde og flytter x- og y-posisjon med 3 piksler. Dette fører da til at en skygge tegnes i grått først, og sort tekst tegnes over skyggen avslutningsvis.

1.4.10 Animasjon

Under kan man se et eksempel på bruk av animasjon[59] i SVG.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 300 300">
 <rect
  x="20" y="20"
  rx="0" ry="0"
  width="250" height="100"
  fill="red"
```

```
  stroke="black" stroke-width="5">
  <animateColor attributeName="fill" attributeType="XML"
   from="red" to="yellow"
   dur="2s"
   fill="freeze"
  />
  <animate attributeName="rx" attributeType="XML"
   from="0" to="10"
   begin="2s" dur="2s"
   fill="freeze"
  />
  <animate attributeName="ry" attributeType="XML"
   from="0" to="10"
   begin="2s" dur="2s"
   fill="freeze"
  />
 </rect>
</svg>
```



Eksempelet viser et rektangel som omtalt før, men denne gangen med et sett med animasjonsregler som påvirker fyll-fargen og avrundingen i hjørnene. Attributtet **attributeName** spesifiserer hvilket attributt på det omsluttende elementet som skal animeres, og attributtet **attributeType** spesifiserer om det er et XML-attributt eller et CSS-attributt som skal animeres. I vårt eksempel animerer vi kun XML-attributter. Først spesifiserer vi at fyll-fargen skal endres fra rød til gul over en tidsperiode på 2 sekunder. Når farge-endringen er ferdig skal den bli stående fordi vi benytter fill="freeze". Deretter har vi to identiske regler som forteller at rx og ry-attributtene (hjørneradius) skal endres fra verdien 0 (skarp) til verdien 10 (avrundet) over en periode på 2 sekunder, men man skal vente 2 sekunder før man starter endringen. I dette tilfellet skal endringen bli stående igjen etter at animasjonen er ferdig.

1.5 Programvare som kan produsere SVG-dokumenter

Det finnes en god del programvare[60] på markedet som allerede støtter SVG-standarden til en varierende grad. Under har vi klassifisert de mest populære verktøyene på markedet som faktisk kan brukes til å produsere SVG-dokumenter.

1.5.1 Kommersiell programvare

Microsoft Visio[61] er et verktøy for å lage varierte vektoriserte diagrammer. Visio beskriver ikke på sine nettsider eksplisitt at de støtter SVG som et eksport-format, men en omfattende artikkel[62] beskriver hvordan Visio kan benyttes i en arbeidsflyt som involverer bruk av SVG. Avsnitt 3.3 i artikkelen forteller også om noen av begrensningene med å importere SVG-dokumenter inn i Visio, f.eks. det at text og tspan-elementer i SVG ikke eksplisitt gir opplysninger om teksten faktisk inneholder linjeskift eller om teksten har blitt foldet fordi linjene er for lange. Det er en del andre deler av SVG-standarden som ikke er støttet, men alt i alt er støtten overraskende god.

Adobe Illustrator[63] er et av verdens mest kjente verktøy for grafiske designere. Illustrator har støttet SVG siden versjon 9 og fikk også utvidet støtte for SVG Tiny i CS2 (versjon 12) i følge Wikipedia. Illustrator CS2 og senere støtter flere avanserte funksjonaliteter[64] i SVG, deriblant filtere, interaktivitet, rik typografi, hyperlenking og mer.

CorelDRAW[65] er et designer-verktøy på lik linje med Adobe Illustrator. CorelDRAW har støttet SVG siden versjon 10[66], som ble lansert i 2000. De fleste statiske finessene i SVG er godt støttet[67] i versjon 12. En ting som er verdt å legge merke til er at skygger faktisk blir gjort om til rastergrafikk istedenfor å benytte f.eks. blur-filter samt et offset-filter som vist i forrige avsnitt (se rubrikken ang. filtere). Det er ikke klart hvorfor de velger å benytte denne metoden på skygger.

Xara Xtreme[68] er på lik linje med Illustrator og CorelDRAW et verktøy for grafiske designere. Xara Xtreme har hatt støtte for eksport av SVG-dokumenter siden versjon 3.2[69]. Xara bekrefter at støtten for SVG ikke er komplett, men at de jobber med å forbedre støtten fortløpende.

1.5.2 Fri programvare

OpenOffice.org Draw[70] er et tegneprogram som er en del av den større OpenOffice.org-pakken (OOo) for produktivitet. Følgende dokument[71] bekrefter at OOo Draw har hatt grunnleggende støtte for eksport av SVG-dokumenter siden 2002. Det som er interessant å legge merke til i dette dokumentet er at OOo baserer seg sterkt på Batik SVG Toolkit (omtales under) for håndtering av SVG. Siden Batik har en av de mest komplette implementasjonene av SVG-standarden bør OOo Draw sin støtte være ganske god.

Inkscape[72] er et vektorbasert tegneprogram på lik linje med Illustrator, CorelDRAW og Xara Xtreme. Det som er spesielt med Inkscape er at det benytter SVG som standard filformat for lagring av vektorisert grafikk. Inkscape har ikke noe proprietært filformat. Inkscape støtter majoriteten av SVG 1.1[73], med unntak av animasjon og noen filtre.

Scribus[74] er en programvare for produksjon av side-oppsett (DTP), på lik linje med f.eks. det kommersielle Quark Express. Scribus sin støtte for SVG[75] er relativt grunnleggende. De forteller at de ikke støtter tekst-elementer fullt ut enda, samt at graderinger, innebygde bilder og masking/clipping også ikke er støttet. Derimot bekrefter de at nesten alle funksjoner i et Scribus-dokument kan eksporteres til SVG med stort hell. Det er stort sett import av SVG som er problematisk.

1.5.3 Verktøy for utviklere

Batik SVG Toolkit[76] er et bibliotek for Java-utviklere som kan benyttes til å manipulere, produsere og presentere SVG. Versjon 1.7 av Batik støtter den statiske profilen til SVG 1.1 uten mangler, samt at den også støtter en god del av den deklarative spesifikasjonen for animasjon (SMIL).

Cairo[77] er et 2D grafikk-bibliotek for utviklere. I tillegg til å kunne presentere vektorgrafikk på interaktive flater som X11, Win32 og Quartz, kan den også generere SVG-dokumenter av tegne-operasjonene. Cairo er et C-bibliotek med bindinger til mange programmeringsspråk, bl.a. PHP, Python, Perl, Ruby, Lisp, Java, C++, .NET, med mer. Cairo benyttes av blant annet Gecko[78] for fremvisning av SVG i Mozilla-baserte nettlesere, samt at den benyttes av WebKit-motoren, som danner grunnlaget for nettleserne Safari og Chrome.

Processing[79] er et programmeringsspråk som fokuserer på å lære programvareutvikling ved hjelp av en visuell kontekst. Verktøyet kommer med en integrert IDE som gjør det enkelt å komme i gang med programmeringen og umiddelbart se resultater. Processing er faktisk Java på bunnen, og benytter seg av den spesialiserte klassen PApplet[80] som implementerer mesteparten av språket. Processing har innebygd støtte for SVG[81], noe som gjør det enkelt å inkludere avansert vektorgrafikk i et Processing-prosjekt. SVG-støtten er relativt grunnleggende[82], det er kun mulig å laste inn enkle primitiver. Fordelen med dette er at det ikke øker størrelsen på klasse-filen som må distribueres altfor mye. Hvis man derimot trenger avansert SVG-støtte kan man benytte Batik-biblioteket som nevnt ovenfor.

2 Støtte for SVG i populære nettlesere

2.1 Kort om nettlesere og SVG-støtte

SVG er en standard som lenge har vært tilgjengelig i de fleste populære nettlesere på dagens marked. SVG gjør det mulig for nettlesere å utføre avanserte animasjonsoppgaver, samt å generere egne tekst-typer relativt enkelt. Dette vil sannsynligvis få stor betydning for hvordan vi utvikler websider i fremtiden.

Et viktig spørsmål er hvor mye av standarden som fungerer i de ulike nettleserne. Dette varierer en god del fra nettleser til nettleser. En mulig årsak til at SVG ikke er tatt i bruk aktivt kan være at man ikke kan garantere at resultatet blir som forventet i de ulike nettleserne.

Internet Explorer har ingen innebygget støtte for SVG, men SVG-støtte kan oppnås ved å benytte en plugin utviklet av andre aktører. Adobe har lenge tilbydt SVG Viewer[83], en plugin som gir Internet Explorer mulighet til å vise nesten alle elementer i standarden. Apache har også utviklet en plugin navngitt Batik[76] som Internet Explorer kan benytte. Denne kan også benyttes av andre applikasjoner som har behov for SVG-støtte.

Alle de andre populære nettleserne har utviklet en implementasjon av SVG-standarden, men ingen har kommet til det punktet at all funksjonalitet i standarden er støttet fullt ut. Både Mozilla Firefox, Google Chrome, Apple Safari og Opera har alle prøvd å implementere SVG. Noen av dem har lykkes i større grad enn andre.

Jeff Schiller, som er en aktiv deltaker i SVG Interest Group[84] (SVG IG) har laget en oversikt[85] over hvilke deler av SVG-standarden som er implementert i de ulike nettleserne. Det er en oversikt som tar for seg både nettlesere og andre applikasjoner som implementerer SVG-standarden. Denne oppgaven tar ikke for seg de andre applikasjonene som er spesifisert her, kun nettlesere som er relevante og i bruk i dagens marked. Det finnes gode kilder på nettet over hvilke nettlesere som er i bruk i dagens marked. StatCounter[86] er en slik kilde som gir en veldig god oversikt. Ut fra denne oversikten kom vi frem til følgende nettlesere som bør få deres SVG-implementasjon evaluert:

- Microsoft Internet Explorer 7 / 8
- Mozilla Firefox 2.0 / 3.0 / 3.5 / 3.6
- Google Chrome 2 (Nightly build) / 4
- Opera 9.6 (9.5 er nærmeste) / 10.0
- Apple Safari 4

På Jeff Schillers informasjonsside om SVG-status i ulike nettlesere er det angitt en prosentandel som forteller hvor mye av standarden som er støttet i den respektive nettleser. Det er interessant å se at Opera, som har utviklet sin egen

nettlesermotor, kommer bedre ut av testen enn f.eks Firefox, hvor SVG-implementasjonen er utviklet på toppen av en eksisterende motor.

- Microsoft Internet Explorer 0.00% / 0.00%
- Mozilla Firefox 46.17% / 60.40% / 60.77% / 61.50%
- Google Chrome 81.39% / 82.12%
- Opera 94.16% / 94.34%
- Apple Safari 82.12%

Ut i fra prosentfordelingen over kan man legge merke til at både Chrome 4 og Safari 4 har samme prosentfordeling, noe som er rimelig ettersom begge er basert på WebKit sin implementasjon av SVG. Firefox kommer dårlig ut i prosentfordeling, selv om det er en av de mest brukte nettleserne etter Internet Explorer. Som nevnt tidligere legger vi merke til at Internet Explorer ikke har innebygget støtte for SVG i det hele tatt.

2.2 Dekningsgrad i implementasjonene

Mozilla Firefox har bygd SVG-standarden direkte inn i nettleseren. Mozilla har en egen prosjektgruppe som jobber med SVG-implementasjonen i Firefox. Det er blitt opprettet en status-side[87] hvor det ligger en oversikt over funksjonalitet som per i dag er implementert. Denne blir løpende oppdatert. Det er enkelt å legge merke til at flere moduler har deler som enda ikke er støttet. Vi har valgt å basere sammenligningen vår på de modulene i SVG-standarden Mozilla Firefox ikke har støtte for eller har en mangelfull implementasjon av.

Moduler vi bør legge spesielt merke til:

- Text[88]
- Color Profile[89]
- Cursor[90]
- View[91]
- Animation[59]
- Font[92]

Ut fra status-sidene[87] kan vi konkludere med at SVG-implementasjonen i Firefox har et stykke igjen før alt fungerer fullgodt. Vi ser blant annet at støtten for symboler, deriblant japanske tegn, er dårlig. Disse er ikke implementert i tekstmodulen[88] i Firefox' implementasjon av SVG. Det er heller ikke mulig å spesifisere eget utseende på musepekeren i cursor-modulen[90]. Funksjonalitet nevnt over er ikke blant de mest benyttede i SVG-standarden, derimot har vi animasjon[59] og tekst[92] som burde være veldig relevant i forhold til bruk av SVG aktivt i web-utvikling. All funksjonalitet under disse modulene er enten ikke implementert eller inneholder feil.

Opera har en av de bedre implementasjonene av SVG-standarden. Jeff Schillers undersøkelse[85], som benytter en testpakke fra W3C for å gradere støtten for SVG i de ulike nettleserne, bekrefter dette. Det er fortsatt moduler med mangler

i Opera, som det er i Mozilla. Vi kan se ut fra status-sidene[93] hos Opera at mye av den samme funksjonaliteten som mangler i Mozillas implementasjon heller ikke eksisterer hos Opera. Det står blant annet definert på disse sidene at SVG-implementasjon i Opera ikke støtter video eller lyd. Det er heller ikke her støtte for symboler i fonter. Støtte for lyd og video skal etter hvert bli integrert direkte i nettleseren ut fra hva man kan lese om Operas videre utviklingsarbeid[94] med nettleseren. Dette er helt klart tett knyttet til arbeidet med støtte for HTML5, som vil gjøre inkludering av multimedia i nettlesere enklere.

Mozilla Firefox baserer seg på Gecko[95], som er en motor for å vise bilder og tekst i nettleseren. SVG-støtte ble derfor implementert på toppen av denne motoren. Opera derimot har laget en helt egen implementasjon siden de ikke baserer seg på noen separat motor. Dette gjør det sannsynlig at Operas motor er mer tilrettelagt for SVG.

Google Chrome og Apple Safari baserer seg derimot på WebKit[96]. WebKit kan sammenlignes med Gecko-motoren som Mozilla Firefox benytter, og er grunnsteinen i nettleseren. Implementasjon av SVG er derfor gjort i WebKit. Vi vurderer derfor WebKit, ikke Chrome og Safari, som en enkelt implementasjon.

WebKit har mange likheter med Mozilla, men ut fra status-sidene[97] ser det ut til at det er mangler eller ufullstendige implementasjoner på moduler som f.eks Mozilla og Opera har implementert fullstendig. Text-modulen har heller ikke her støtte for symboler, som i Opera og Mozilla, noe som kan indikere at denne funksjonaliteten er vanskelig å implementere. WebKit har heller ingen fullstendig implementasjon av Filter-modulen. Det er få moduler som ikke er prøvd implementert, men de fleste inneholder feil eller er bare delvis implementert. Det WebKit har klart i motsetning til Mozilla er å implementere store deler av animasjonsmodulen, selv om det finnes mange feil. WebKit har også implementert store deler av Font-modulen, noe Mozilla og Opera ennå ikke har fått til.

Internet Explorer fra Microsoft har derimot ingen innbygget støtte for SVG. Det ser nå lysere ut for Internet Explorer, ettersom det har blitt kjent at Microsoft har meldt seg inn i SVG WG[98], forhåpentligvis for å bygge SVG-støtte inn i IE9. I Internet Explorer 8 og tidligere finnes det ingen støtte for SVG-standarden uten at det blir benyttet en ekstern plugin.

2.3 Funksjonalitet ikke støttet i implementasjonene

Det finnes en mer detaljert oversikt over hvilke elementer de ulike nettleserne støtter i forhold til SVG på samme sted som prosentfordelingsoversikten[85]. Oversikten gir en veldig god indikasjon på hvilke områder av SVG-standarden som ikke er implementert i de respektive nettleserne.

Jeff Schiller gir følgende beskrivelse av fargekodene[99]:

> *I use a top secret scoring system ... ok, well actually I just use GREEN for "Pass" (2 points), YELLOW for "Almost Pass" (1 point), RED for "Fail" (0 points) and BLACK for "Fail with Crash" (-1 point). The "Almost Pass" scoring is subjective, so think of it as me giving out points for trying.*

Det første vi legger merke til er at det finnes mange røde felter totalt. Fra prosentfordelingen nevnt ovenfor fremgår det klart at noen nettlesere har flere røde elementer enn andre. Det første som legges merke til er filtre, som inneholder store deler røde felter. Disse gjør det mulig å legge til ulike effekter på bilder, som f.eks uklarhet, lys eller uskarpe kanter.

Mozilla Firefox har i sine siste versjoner hatt full støtte for filtre. Eneste versjon som ikke hadde støtte for filtre var Firefox 2.0. Både 3.0, 3.5 og 3.6 har hatt god støtte for filtre, med unntak av 3.5 som manglet støtte for displace-filteret.

Opera har også hatt god støtte for filter-elementene, bortsett fra versjon 8.5 hvor det ikke fantes støtte i det hele tatt.

Både Chrome og Safari, som begge baserer seg på WebKit, har ingen støtte for filtre. Det kan se ut som om Chrome har prøvd å få til støtte i versjon 5 beta, men denne ser ut fra oversikten, relativt eksperimentell ut.

Det er generelt samme resultat på de resterende filter-elementene som man finner på tester litt lenger ned på status-siden[85]. Dette er egentlig interessant, fordi det viser at man rett og slett i noen nettlesere bare har droppet hele denne modulen.

Det er derimot motsatt med font-elementene, som gjør det mulig for SVG å tegne opp bokstaver på ulike måter. Det ser ut som WebKit har implementert flere av font-elementene som igjen gjør det mulig for både Safari og Chrome å bruke disse. Opera har også fått med disse i sin implementasjon av SVG, mens Firefox i de fleste versjoner ikke har klart å implementere disse uten feil.

Det er egentlig en viktig del av SVG-standarden å kunne vise fonter korrekt i en nettleser. Siden font-modulen er et så viktig element når det gjelder tekst på internett, kan det diskuteres om SVG, slik standarden er implementert i dag, er veldig brukelig for webutviklere.

En annen del av oversikten er animasjonselementene. Det første vi oppdager er at Firefox ikke har støtte for noen av aninimasjonselementene bortsett fra 3.7-Alpha. Animasjons-elementene er bedre implementert i både Chrome og Safari, i det minste i de siste versjonene. Noen elementer har kun blitt delvis implementert eller inneholder mangler, men det meste av animasjons-elementene er implementert i både Chrome, Safari og Opera. Det kan se ut som vi får animasjon-støtte når Firefox 3.7 kommer på markedet.

Det siste røde feltet er text-elementer i SVG. Det kan se ut som Mozilla har hatt problemer med å implementere disse elementene i alle sine versjoner av Firefox.

De andre nettleserne har relativt feilfrie implementasjoner, men det er et par stykker som enda ikke er helt komplette eller har feil.

Generelt kan det se ut som de fleste av nettleserne har fått det til når det kommer til implementasjon av SVG-standarden, men det er relativt stor forskjell på hvilke elementer som faktisk er støttet i de ulike nettleserne. Dette gjør det vanskelig å ta høyde for spesiell funksjonalitet når man skal utvikle websider med SVG-innhold. Man kan heller ikke regne med at animasjoner fungerer som de skal, eller at skrifttypen man har valgt å bruke vises likt i alle nettlesere. Dette er en stor svakhet som gjør jobben til webutviklere og webdesignere vanskelig. Det er indikasjoner på at ting blir bedre og bedre ettersom versjonsnummeret øker på de ulike nettleserne.

Internet Explorer er ikke nevnt ovenfor fordi den ikke støtter SVG uten å installere et tredjepartstillegg for å vise SVG. Det er påbegynt arbeid med SVG i Internet Explorer i januar 2010, noe som kan bety at vi i fremtiden kanskje får se en Internet Explorer-versjon også med SVG-støtte. Dette antar vi vil gjøre SVG mer kjent og forhåpentligvis vil øke bruken av SVG.

Så langt er det Opera som har førsteplass og innehar den mest komplette implementasjonen.

2.4 Funksjonalitet med feil i implementasjonene

I Mozilla Firefox jobbes det i hovedsak med feil i to av SVG-modulene. Det er Font- og Animation-modulen. Animasjonsmodulen står det verst til med ettersom denne inneholder elementer som enten ikke er implementert, eller som er prøvd implementert men inneholder feil som gjør at de ikke funger på en riktig måte i forhold til standarden.

Første element som ikke fungerer korrekt er `animate`, som gjør det mulig å animere et enkel attributt eller ting over tid. Det er lagt inn en feil-rapport[100] på denne modulen i Firefox sitt feilhåndteringssystem, Bugzilla. Et annet element som inneholder feil er `animateTransform`. Elementet skal blant annet gjøre det mulig å skalere eller rotere en ting i SVG, f.eks et rektangel eller en firkant. Det er også her lagt inn en feil-rapport[100] i Bugzilla. Elementet funger ikke som det skal, noe som har opprinnelse i feilen som også eksisterer i animate-elementet. Totalt sett gjør dette at det å animere elementer ikke fungerer i SVG-implementasjonen i Firefox.

Den andre modulen som enda ikke er ferdig implementert i Firefox er Font-modulen. Foreløpig har Mozilla ikke klart å vise mer avanserte fonter, eller fonter som inneholder spesielle symboler som man f.eks finner i japansk tegnsett. Dette fører til at fonter generelt ikke vil fungere korrekt. Istedenfor å vise deler av fonten som faktisk fungerer eller gi tilbake en generell font, blir tekst som bruker en avansert font ikke vist i det hele tatt. Det har også her blitt lagt inn en feil-rapport[101] i Bugzilla. Feilen skaper problemer med å vise tekst

som blir generert av SVG generelt og vil igjen gjøre SVG upålitelig til å vise tekst i når det blir brukt avanserte fonter.

WebKit har flere moduler som inneholder feil. Dette går på flere av de samme modulene som Firefox har hatt problemer med å implementere, men også Cursor-modulen, samt View. Cursor-modulen gjør det mulig å lage en egendefinert peker som kan benyttes i applikasjonen, f.eks at man har lyst til å bruke et eget ikon som skal fungere som mus. Det finnes ingen feilhenvisninger som viser hva som er feil på Cursor-modulen. Men modulen er definitivt ikke implementert korrekt og markert med gult. View-modulen har fått registrert en feil[102] som forteller at det ikke er mulig å hente ut currentView. Dette er hvilke elementer vi har i nettleseren på et gitt aktivt tidspunkt, og W3C har selv kommentert at dette er vanskelig å implementere.

WebKit har også flere av svakhetene som eksisterer i implementasjonen til Firefox. Dette går i hovedsak på animasjonsmodulen. Flere av elementene i animasjonsmodulen inneholder feil. Dette gjør blant annet at det ikke er mulig å sette attributter under en animasjon for en viss tidsperiode. Det er heller ikke mulig å flytte en ting på en linje, eller skalere og rotere et element for å lage en animasjon. Dette er relativt viktige deler når det kommer til å lage animasjoner. F.eks når man skal få ting til å sakte føres inn og viskes ut eller bare vil at ting flytter seg fra et sted til et annet. Implementasjonen av animasjonsmodulen i WebKit gir derfor ikke utviklere mye å benytte. Det samme gjelder for gradvise endringer i farger, hvor det har blitt registrert en feil[103] i WebKit sin Bugzilla.

Opera har ikke spesifisert steder hvor elementer i de ulike modulene kun er delvis implementert. Ut fra den generelle SVG-status-siden[85] kan vi trekke en konklusjon på hvordan det står til med SVG-støtten i Opera. Ut fra status-siden ser man få tegn til elementer som fungerer delvis. Det eneste som kan nevnes er at Opera generelt har flere grønne felter enn både WebKit- og Gecko-baserte nettlesere. Det finnes et par tester som viser at det kun er delvis støtte tilgjengelig. Dette er blant annet på animasjonstesten, og på testen som går på fonter som inneholder symboler. Det ser fortsatt ut som både fonter og animasjon er bedre implementert i Opera enn i WebKit og Mozilla Firefox.

3 Microsofts rolle i utbredelsen av SVG

3.1 Internet Explorers manglende støtte for SVG

Fra analysen av nettlesere tidligere ser vi at Microsoft sin Internet Explorer (IE) aldri har hatt innebygget støtte for SVG. Man har alltid vært avhengig av et tredjepartstillegg (plugin) for å kunne se SVG i IE. Det mest populære tillegget[104] har til nå vært Adobe sin SVG Viewer, men andre har nå kommet på markedet fordi Adobe valgte å stoppe å utvikle sin plugin[105].

IE har siden versjon 5.0[106] hatt støtte for et annet grafikk-språk som heter VML (Vector Markup Language). VML er implementert i XML på lik linje med SVG, men er en mye eldre implementasjon som aldri har blitt standardisert. Vi kommer inn på VML i litt mer detalj senere.

Fordi VML allerede var implementert i IE var Microsoft ikke interessert[107] i å legge inn støtte for en konkurrerende teknologi som SVG. De mente det var nok å ha ett vektorformat og var derfor uvillig til å støtte en åpen standard som SVG. Sammen med det faktum at IE sitt utviklerteam nærmest ble lagt ned etter utviklingen av IE6 og at det var stille på nettleserfronten i over 5 år[108] førte til en stagnering av nettlesermarkedet.

Det førte til store problemer for SVG WG, fordi Microsoft hadde gitt klart uttrykk for at VML var en standard som ikke skulle endres/videreutvikles fordi den allerede var i bruk[20]. W3C ble da sittende mellom barken og veden. De kunne utvikle en standard som de hadde kontroll over og kunne videreutvikle, som markedslederen ikke var interessert i å støtte. Alternativet ville være å ratifisere en standard de ikke kunne videreutvikle, men som ville være mulig å få god markedspenetrasjon på i løpet av relativt kort tid. Det at SVG eksisterer som en standard i dag som ikke er kompatibel med VML viser klart og tydelig at de valgte å gå for den første løsningen. Dette har gitt oss en markedsituasjon hvor nettleseren med størst markedsandel ikke støtter den ledende og mest anerkjente teknologien på markedet.

Helt siden IE fikk støtte for VML har Microsoft vært uvillig til å endre sitt standpunkt[107] på om de ønsker å implementere SVG i nettleseren. Dette, sammen med det faktum at Microsoft ikke var villig til å videreutvikle IE, har ført til at sluttbrukere har begynt å benytte andre nettlesere, slik som Firefox, Safari, Opera og Chrome. Alle disse nettleserne støtter SVG i varierende grad, som tidligere avklart. Det at IE sine markedsandeler har falt drastisk de siste årene[109] har gjort at Microsoft har vært nødt til å revurdere sin strategi hvis de ønsker å beholde brukerne. Det at EU-domstolen også har gitt medhold[110] for at Microsoft driver med monopolistisk oppførsel rundt tilgang til alternative nettlesere har gitt sluttbrukere en mer synlig opsjon når det kommer til deres valg av nettleser. Dette, sammen med utbredelsen av diverse mobile enheter[111] som ikke støtter flere av de alternative teknologiene[112] vi kommer inn på senere, har ført til at behovet for SVG fra web-utviklere bare blir sterkere og sterkere.

3.2 Microsoft melder seg inn i SVG Working Group

I januar 2010 ble det annonsert at Microsoft har søkt om medlemskap i SVG WG[98]. Dette kom som et stort sjokk[113] (et positivt sådan) på web-utviklere verden over. Det ble synset mye frem og tilbake om dette betydde at Microsoft endelig hadde endret sin strategi og om dette betydde at en fremtidig versjon av Internet Explorer ville få støtte for SVG. Da Microsoft ble spurt om dette betydde at en fremtidig versjon av IE ville få støtte for SVG valgte Microsoft å ikke ta stilling til det spørsmålet[114] på det tidspunktet.

Som vi har sett fra statistikken på epostlisten (nevnt tidligere) ser det ut som om Microsoft har valgt å benytte sin stemme og påvirker utviklingen av standarden med sin tilstedeværelse. I tillegg er de fremdeles i *good standing* med SVG WG[14], som betyr at de møter opp til de regelmessige møtene og følger opp pliktene i deres medlemskap.

3.3 Microsoft IE9 vil støtte SVG

I mai 2010 annonserte Microsoft at deres neste versjon av Internet Explorer kommer til å støtte grunnleggende elementer i SVG 1.1[115]. Annonseringen viser at de støtter følgende elementer fra standarden (direkte utdrag):

- Most SVG document structure, scripting (eventing), and styling (inline and through CSS)
- Many presentation elements and their corresponding attributes and DOM, including:
    - paths
    - shapes
    - colors
    - transforms

Dette betyr at grunnleggende grafikk-operasjoner samt hendelseshåndtering og scriptstøtte nå skal virke. Selv om de ikke støtter animasjon, tekst, filtre og raster-bilder betyr det uansett at man nå skal ha mulighet til å lage avanserte brukergrensesnitt med SVG.

Det at man ikke har støtte for tekst og raster-bilder, samt noen av de grunnleggende filtrene, vil føre til at web-utviklere må benytte tegneoperasjonene og tradisjonelle HTML-metoder for å kombinere tekst, raster og vektorgrafikk for å få til det de ønsker på de populære nettleserne. Forhåpentligvis vil man få støtte for mer SVG-funksjonalitet før den endelige versjonen av IE9 foreligger.

4 Alternativer til SVG

Det er antagelig mange grunner til at SVG aldri har fått like mye oppmerksomhet som sine konkurrenter. Det kan være at andre teknologiere har hvert mer lovende, eller at nettsteder med store brukermasse har valgt bort SVG til fordel for mer utbedredte løsninger som Flash.

Det er i hovedsak SVG som er den åpne løsningen for multimedia og animasjon på internett. Den har også blitt definert som en åpen web-standard av W3C. Det finnes flere store konkurrenter som har fått mye større anerkjennelse. Vi kan blant annet nevne Adobe Flash og Microsoft Silverlight, som begge har blitt tatt i bruk av utviklere og blir benyttet i stor grad på nettsider i dag.

Flash har alltid hatt den største markedsandelen på grunn av sin tidlige ankomst. Adobe har også utviklet god funksjonalitet for både video og lyd via Flash som SVG enda ikke har noe god implementasjon på. Mye av grunnen til at det finnes så mange alternativer til SVG er at SVG aldri har blitt skikkelig implementert i noen nettleser med stor markedspenetrasjon. Det har heller vært et litt uferdig prosjekt som har gjort det vanskelig for utviklere å ta det i bruk. Trenden vi ser nå er at flere og flere begynner å se på SVG. Microsoft har, som nevnt tidligere, sagt seg villig til å ta i bruk SVG i Internet Explorer 9.

Vi skal derfor se hvorfor vi mener at SVG er bedre enn mange av de andre teknologiene som allerede eksisterer på internett.

4.1 Hvordan utviklere benytter Adobe Flash fordi SVG ikke er tilgjengelig?

Adobe Flash[116] har lenge vært en viktig del av vår hverdag på internett. Reklame har fått et helt nytt medium med Flash. Det er enkelt å produsere spennende video og grafikk som fanger brukerens oppmerksomhet. Det også mange internett-sider som er utviklet fullt og helt i Flash, noe som vil gi en helt spesiell brukeropplevelse.

Først og fremst er Flash en multimedia-plattform som nå vedlikeholdes av Adobe[116]. Det er mulig å lage interaktive nettsider ved å bruke Flash for å vise video, lyd og animasjoner. Flash benytter vektor- og raster-grafikk for å lage animasjoner. Det gjør det mulig å lage animasjoner som kan skaleres etter behov uten å miste detaljer (forutsatt at man ikke benytter raster-grafikk).

Flash har også fått et objektorientert script-språk, ActionScript[116], som utviklere kan bruke for å øke funksjonaliteten i Flash-videoer. Dette kan for eksempel være at man vil koble opp mot eksterne script og hente ned informasjon eller lagre informasjon som blir skrevet inn.

Det eneste kravet for å vise Flash-animasjoner og videoer er at man installer en plugin fra Adobe, Adobe Flash Player[116].

4.1.1 Fordeler ved Adobe Flash

Adobe Flash har lenge vært markedsdominerende[117] når video, lyd og animasjon skal vises i nettleseren. Alle har gjort seg vant med at en plugin må installeres for å kunne se siste trailer av en kommende film eller en videoreportasje fra en avis. Etter at Microsoft tok dette i bruk under utvikling av MSN-nettsidene[116] sine, har flere og flere fulgt etter. De fleste internettsider man besøker idag inneholder Flash i en eller annen form. Dette kan være alt fra reklame til nettsider bygget i Flash.

Utvikler som ønsker å bruke animasjoner eller video i sitt arbeid, kan være sikker på at de aller fleste vil kunne vise innholdet om de benytter Flash som teknologi. Utviklere kunne valgt å benytte f.eks SVG til denne oppgaven, da også SVG gjør mye av den samme jobben som Flash gjør. SVG har midlertidig ingen hundre prosent god implementasjon og ingen nettlesere støtter all funksjonalitet SVG-standarden har å tilby. Internet Explorer støtter ingen deler av SVG-standarden og det må også her brukes en plugin[118].

4.1.2 Ulemper ved Adobe Flash

Med tiden har vi blitt mer bevisst på hva vi benytter av applikasjoner og tillegg for å kunne se det vi vil på nettet. Vi har også blitt mer bevisst på hvilke teknologier vi støtter på bakgrunn av åpenhet mot brukermassen, stabilitet og sikkerhet. Jobben som Adobe Flash gjør kunne og burde blitt gjort med andre teknologier med bakgrunn i flere forhold.

Først kan vi se på sikkerheten[119] i Adobe Flash opp igjennom årene, og det er ingen lys verden vi får innblikk i. Adobe Flash er kanskje en av de store synderne med tanke på sikkerhetshull og mulighet for å kunne kjøre skadelig kode i klientens nettleser. Det ble gjort en undersøkelse av blant annet Trusteer[120] som sa at Adobe Flash var en av de mest brukte teknologiene er for å kjøre skadelig kode på nettbrukeres maskiner.

For å kunne utvikle noe i Flash kreves det egne verktøy[121] som kan kompilere og generere koden som skrives slik at sluttresultatet kan vises i Adobe Flash Player. Det er flere verktøy ute på markedet som gjør denne jobben, både gratis-verktøy og kommersielle verktøy. Det er ofte at kommersielle verktøy må velges siden mange av gratis-verktøyene ikke gjør konverteringsjobben godt nok.

Utvikling av SVG krever mindre verktøy enn Flash. Utvikleren kan velge en teksteditor de kjenner og kan bruke effekten uten tidskostnaden ved å måtte lære et nytt verktøy. En SVG-fil kan enkelt testes underveis. Ettersom det ikke er behov for en ekstra plugin kan utvikleren bare åpne SVG-filen i nettleseren. SVG er strukturert tekst i XML format, så enhver utvikler med litt forståelse for XML-formatet kan enkelt tolke SVG-filer.

En ekstra fordel ved SVG er at formatet lagres som tekst. Dette gjør det mulig å komprimere innholdet enda mer, ved å bruke f.eks gzip[122].

Det er også viktig å nevne at Adobe Flash krever en egen plugin[123] for å kjøre Flash formatet. Dette fordi det ikke eksisterer noen implementasjon i nettleserne for å kunne lese Flash direkte uten å benytte tredjepartstillegg.

SVG er implementert direkte i nettleseren og krever derfor ingen ekstra programvare for å kunne vises. Dette gjør det enklere for brukeren å få informasjonen, samtidig som det reduserer antall ekstra- og tredjeparts-programvare man må ha installert. Det er også viktig å påpke at støtte for SVG ikke er helt fullstendig enda, men dette forbedres daglig hos alle de store nettleserne. Per idag har flere av de store mer enn god nok implementasjon for at SVG kan bli tatt i bruk av flere webutviklere.

Adobe Flash er et proprietært[124] prosjekt. Det er Adobe som eier alle rettigheter til Flash Player og det er bare Adobe som kan rette feil eller implementere ny funksjonalitet. Flash er blitt et format som brukes overalt på internett, det er til og med flere store sider som baserer seg på å vise video ved hjelp av Flash. Det at formatet kontrolleres av et enkelt firma gjør det vanskelig for plattformer firmaet ikke er interessert i å støtte.

I motsetning til Flash er det flere andre teknologier, som blir kontrollert av standardiseringsorganisasjoner, som kan benyttes til web-utvikling. CSS, JavaScript og HTML er teknologier som er åpne for alle. Dette gjelder også SVG, som er en åpen standard, og utviklere som er interesert i å være med i utviklingen av disse kan selv involvere seg på forskjellige måter.

4.2 Kan Microsoft Silverlight benyttes for å omgå mangel på SVG-støtte?

Microsoft Silverlight[125] er på mange måter det samme som Flash fra Adobe, pakket inn i kjent Microsoft-stil. Silverlight er først og fremst et rammeverk for utviklere som bruker ASP, C# eller andre språk fra Microsoft. Dette rammeverket gjør det mulig å lage grafikk, animasjoner og vise både lyd og bilde. Rammeverket har også fått deler av .NET-rammeverket integrert, slik at mye av den samme funksjonaliteten som finnes i det vanlige .NET-biblioteket også finnes i Silverlight.

Silverlight er tilgjengelig på andre plattformer enn Microsoft sin, ved å bruke Moonlight. Moonlight[126] er utviklet av Novell, og gjør det mulig å bruke Silverlight på både Linux og BSD-plattformen.

Microsoft har med Silverlight gjort det mulig for utviklere å bruke de samme verktøyene som de er vant til i utviklingen av .NET-applikasjoner. Utvikleren kan også benytte en tekstbehandler for å lage en Silverlight-applikasjon, men dette er en mer komplisert prosess enn å bare bruke f.eks Visual Studio hvor mye av jobben blir automatisert.

4.2.1 Fordeler ved Microsoft Silverlight

Microsoft har laget et rammeverk for å lage større applikasjoner som benytter både grafikk, animasjoner, lyd og video. Det er viktig å se at dette ikke bare er en standard for å vise grafikk på nett men at det er et rammeverk for utviklere som kan brukes for å lage Rich Internet Applications[127]. Silverlight begrenser seg altså ikke til kun å vise grafikk, animasjoner, video og lyd på nett som SVG gjør. Den har et mye bredere fokus.

4.2.2 Ulemper ved Microsoft Silverlight

Silverlight har sine fordeler, men det er ikke uten ulemper når vi skal sammenligne den mot SVG. Siden SVG ikke har like mange muligheter som Silverlight, er det en mye mer spesialisert teknologi.

Silverlight er som sagt et rammeverk, og det blir ikke generert en fil som inneholder alle elementer brukt i applikasjonen. Det er flere filer[128] og mapper som må være til stede på server-siden av en applikasjon for at den skal fungere. Det er også relativt komplisert struktur på noen av filene, og det kan derfor være vanskelig å sette sammen en applikasjon uten et skikkelig verktøy.

SVG har en stor fordel når det kommer til antall filer som må være tilstede, siden den kun krever filen som inneholder grafikken. Dette er også en tekstfil, som ikke trenger noe annet enn en nettleser for å tolkes. SVG kan også bestå av store filer med mange linjer kode, men man vil fortsatt slippe å trekke inn biblioteker eller andre filer for å vise grafikken som er laget. Siden det kun er en fil som bruker den velkjente XML-strukturen, kan SVG-grafikk enkelt lages i en teksteditor.

Silverlight er som Flash en klientside-applikasjon. Det vil si at det er nettleseren som tar seg av kjøringen av Silverlight applikasjonen. Nettleseren har ingen innebygget støtte for Silverlight, og trenger derfor tredjepartsprogramvare[129] for å kunne vise applikasjonen. Det er sjelden positivt når en bruker oppsøker et nettsted og trenger enda en plugin for å kunne vise innholdet. Brukeren hadde fått en bedre opplevelse om andre teknologier kunne brukes for å slippe slike tillegg.

SVG har denne fordelen. Siden det støttes direkte i nettleseren trenger ikke brukeren å bekymre seg om å laste ned ekstra tillegg for å tolke innholdet på ulike nettsider. Ettersom det er store forskjeller på alder og ferdigheter blant internett-brukere er det viktig å gjøre mest mulig informasjon tilgjenglig for flest mulig.

Egentlig kan man stille spørsmål om hvorfor det var behov for enda en teknologi som gjør den samme jobben flere andre har gjort før den. Silverlight og Flash utfører mange av de samme oppgavene i nettleseren. Silverlight har et par mer avanserte funksjoner, som mulighet for å distribuere Silverlight innhold mellom

nettsteder. Silverlight er også mer integrert mot Microsoft sine lyd- og video-formater. Mange vil derfor gå så langt som å si at Silverlight er Flash i Microsoft-innpakning.

Hvis vi sammenligner f.eks hvordan SVGs XML og Silverlights XAML ser ut når nettleseren skal tolke det, er dette veldig likt. Mange stiller derfor spørsmål hvorfor Microsoft ikke heller tok i bruk SVG som allerede var etablert istedenfor å lage noe helt eget og proprietært.

4.3 Kan JavaScript-bibliotek som Raphaël være et alternativ til SVG?

Raphaël[130] er et JavaScript-bibliotek som baserer seg på SVG-standarden for å kunne vise animasjoner og vektorgrafikk i nettleseren. Den benytter seg også av VML for å kunne vise vektorgrafikk og animasjoner i Internet Explorer. Dette gjør det mulig for webutviklere å ta i bruk et enkelt verktøy for å vise animasjoner og vektorgrafikk, nemlig Raphaël.

Ved å benytte scriptet når det skal genereres vektor-grafikk på en nettside slipper man å tenke på om grafikken vil vises likt i de ulike nettleserne. Biblioteket emulerer funksjonalitet som ikke er tilgjenglig i vektorgrafikk-implementasjonen til nettleseren. Derfor kan man være enda sikrere på at innholdet blir vist på korrekt måte uavhengig av nettleser.

I prinsippet kan man ikke kalle Raphaël et alternativ til SVG fordi det faktisk tar i bruk SVG og emulerer funksjonalitet som enda ikke er støttet helt ut i de ulike nettleserne. Det kan heller være et godt hjelpemiddel for webutviklere som er vant med en imperativ programmeringsform. Webutviklere som vil ikke ønsker å sette seg grundig inn i en XML-syntaks for å kunne utvikle SVG-filer direkte kan heller bruke et sett med bibliotek-funksjoner for å lage grafikken de har behov for.

4.3.1 Fordeler ved Raphaël

Selv om Raphaël ikke er et nytt alternativ til SVG-standarden er det fortsatt et nyttig verktøy for utvikling av vektorgrafikk.

Raphaël gjør det mulig å lage vektorgrafikk ved å bruke metodekall[131] som man ville gjort i f.eks Java eller C++. Biblioteket har altså et imperativt[132] programmeringsfokus, i motsetning til SVG som er deklarativt[6]. Ved å bruke denne programmeringsformen er det enklere å f.eks duplisere elementer og foreta valg basert på hendelser i koden.

SVG benytter en deklarativ programmeringsform i sine tekstfiler som kan tolkes av nettleseren. For utviklere eller brukere som er kjente med f.eks XML og HTML, som begge er deklarative språk, er det en kort læringskurve til å kunne lage avanserte animasjoner. De som ikke er kjent med denne formen for

programmering kan ha mer nytte av f.eks Raphaël som lager vektorgrafikk ved å bruke kjente programmeringsmetoder.

Det kan også være en tung affære å gjøre endringer på et SVG-element etter å ha skrevet komplisert kode for animasjoner og grafikk. Det kan f.eks være at det har blitt valgt feil farge, eller gjort en skrivefeil i en tekst som blir definert flere steder for animasjon. I slike tilfeller kunne man sluppet unna med mindre omskriving ved å bruke Raphaël og duplisert elementer med en for-løkke.

4.3.2 Ulemper ved Raphaël

Biblioteket er ikke direkte støttet av nettleseren. Det kreves at biblioteket er inkludert i koden på nettsiden før eventuelle animasjoner vil tegnes opp. Vi ser her noe av det samme som må til for å få blant annet Flash og Silverlight til å fungere, men Raphaël er likevel bedre ettersom det ikke er opp til brukeren å legge til noe ekstra, men opp til webutvikleren.

I en optimal verden er tredjepartsmoduler som Flash og Silverlight byttet ut med innebygd SVG-støtte som er er implementert identisk i alle nettlesere. Siden vi enda ikke er kommet dit hen at SVG har full støtte i alle nettlesere er det fortsatt nødvendig å måtte legge et lag i mellom nettleser og klient for å få vist grafikk og animasjon på tvers av nettlesere.

Raphaël har tatt utgangspunkt i grafikk, samt animasjonsfunksjonaliteten, i SVG-standarden. Denne funksjonaliteten har allerede blitt implementert i de fleste nettlesere ved hjelp av SVG. Etterhvert som funksjonaliteten utvikles videre kan man spørre seg om det faktisk er nødvendig med et eget bibliotek for å oppnå funksjonalitet som senere vil eksistere i alle nettlesere med SVG. Selv Microsoft har annonsert[133] at de vil ha en grunnleggende SVG-implementasjon klar i IE 9. Raphaël har i en lenger periode gjort det enklere for webutvikler å lage animasjoner som fungerer i alle nettlesere men vil bli overflødig så fort komplette implementasjoner av SVG-standarden blir tilgjenglig i nettleserne.

Raphaël vil først og fremst være et verktøy[134] for å bruke vektorgrafikk-teknologien som er til stede i nettleserne. Biblioteket hadde ikke fungert uten SVG- og VML-støtten som er tilgjengelig i nettleserne. Inntil vi ser en jevn og god støtte for SVG i alle de store nettleserne er det behov for biblioteker som Raphaël som gjør det mulig å utnytte felles funksjonalitet i to inkompatible implementasjoner for vektorgrafikk.

4.4 Microsoft VML, et alternativ til SVG

VML har samme deklarative fokus[135] som SVG, og benytter også XML-syntaks for å lage grafikk og animasjoner. VML er utviklet av Microsoft for Internet Explorer of Office, og fungerer på mange måter likt med SVG for andre nettlesere. Som nevnt tidligere er det så mange likheter at JavaScript-bibliotek som Raphaël kan utnytte likhetene.

VML er et proprietært språk utviklet av Microsoft som ble presentert for W3C[106] som en ny standard for animasjon og multimedia på nett. SVG valgte bort dette språket til fordel for SVG.

Etter at W3C startet utviklingen av SVG isteden for å ratifisere VML, ble arbeidet med VML avsluttet[106]. Selv om utviklingen ble avsluttet har Microsoft valgt å beholde implementasjonen av teknologien i nettleserne sine fra Internet Explorer 5.0 og oppover. Dette gjør det mulig for utviklere også i dag å bruke VML for å vise animasjoner og grafikk på samme måte som SVG.

4.4.1 Fordeler ved Microsoft VML

VML er det eneste språket som ligner SVG og som fungerer i Internet Explorer. Det funger også bra, og kan vise både grafikk og animasjoner[135] som SVG kan gjøre i andre nettlesere. Det gjør at VML foreløpig er veldig nyttig for utviklere som et språk likt SVG for å ha en form for cross-browser-support i grafikken som lages. SVG har enda ikke kommet dithen at Internet Explorer støtter standarden, og det er derfor enda behov for VML.

En annen fordel med VML er at det i forhold til f.eks språk som Flash og Silverlight bruker XML-syntaks. Dette er samme struktur som SVG bruker for å vise grafikk og animasjoner. Dette gjør for det første at koden er liten i størrelse og kan enkelt komprimeres. Det er også enkelt for utviklere som har jobbet med SVG å forstå strukturen i VML. Sammenligning av syntaks viser store forskjeller, men dette er kun et spørsmål om å lære hvordan VML skal brukes.

Siden VML benytter XML er det også mulig å konvertere mellom VML og SVG. Dette trenger ikke å være en enkel prosess, men det er i prinsippet mulig siden begge er implementert i XML.

4.4.2 Ulemper ved Microsoft VML

Først og fremst er VML enda en ting webutviklere må ta hensyn til når de skal ta i bruk grafikk og animasjon i nettleseren. Det kan være enkelt å f.eks legge ved både VML- og SVG-kode så det ikke er behov for å ta hensyn til hvilke nettleser som blir brukt for å vise innholdet. Det kunne vært mulig å generere kode for enten VML eller SVG basert på hvilken nettleser som ble brukt av klienten, men det måtte da lages en eller annen form for kodegenerator for SVG og VML som dynamisk kunne generere korrekt kode slik at grafikk og animasjoner ble vist på rett måte. Dette fører til høy kostnad på serversiden.

VML er veldig nettleserspesifikt, det fungerer faktisk kun i en nettleser. Dette er lite positivt ettersom all utvikling av VML-grafikk vil måtte skrives om til noe annet for at andre enn IE-brukere skal kunne se innholdet.

VML har blitt en del av historien til Microsoft ettersom all videre utvikling av denne teknologien ble avsluttet i 1998. Etter dette har Microsoft ikke gjort

forbedringer eller prøvd å lage mer funksjonalitet. Dette kan ha sammenheng med nederlaget hos W3C da VML ikke ble godkjent som offisiell standard for multimedia på internett.

Det er derfor til ettertanke at VMLs eksistens kun er tilstede i fravær av annen teknologi som fungerer like godt i Internet Explorer. Hadde Microsoft gjort en innsats for utvikling av SVG også for Internet Explorer ville det vært mulig for utviklere heller å fokusere på SVG istedenfor å bruke ressurser på begge.

5 Konklusjon

SVG er en standard som har utviklet seg i god takt siden første versjon fra 2001. Standarden har blitt implementert i mange produkter, fra tegneprogrammer, produktivitetprogrammer, nettlesere til mobile enheter. Populariteten i dagens marked er svært god, og utviklere finner det svært fornøyelig å jobbe med et XML-format for å produsere vektor-grafikk.

SVG hadde lenge et problem fordi Microsoft ikke var villig til å implementere det i sitt flaggskip-produkt, Internet Explorer. Microsoft sitt valg om å søke medlemskap i SVG WG for så å implementere støtte for SVG i den neste versjonen av Internet Explorer (IE9) viser en klar endring i strategi fra deres side. SVG-støtte i IE er noe markedet lenge har etterspurt, og disse grepene fra Microsoft gir web-utviklere verden over virkelig noe å glede seg over. Microsoft har ikke gitt noen offisiell grunn for hvorfor de nå har valgt å endre strategi, men det er god grunn til å tro at deres fallende markedsandeler på nettlesermarkedet er en av grunnene. Vår spådom er at IE9 vil ha erstattet Microsofts eksisterende versjoner i løpet av 3-5 år og web-utviklere kan endelig begynne å ta i bruk vektor-grafikk på nett uten altfor store problemer.

Som alltid må web-utviklere fremdeles måtte forholde seg til feil og mangler i de forskjellige nettleserne. Dette er ikke noen ny realitet, da det finnes mange problemer i HTML- og CSS-støtten i nettlesere. SVG kommer garantert til å slite med de samme barnesykdommene i flere år fremover. Men det faktum at vi nå er på tampen til å få grunnleggende funksjonalitet ut i hendende på web-utviklere er definitivt positivt.

Det som er viktig å nevne er at SVG har vært støttet av de fleste andre nettleser-leverandører i varierende grad i en god stund, og med Microsoft sittende ved bordet blir bildet komplett. Så snart IE6-8 får en minimal markedsandel kan man benytte grunnfunksjonaliteten i SVG uten å tenke seg om.

VML, Flash og Silverlight, alternative teknologier til SVG, vil ikke forsvinne over natten. Men markedet viser helt klart og tydlig at de er klare til å ta i bruk ikke-proprietære teknologier på tross av de utfordringer det gir i forhold til kompatibilitet. Fordelene med å ikke kreve installasjon av et tredjepartstillegg (som vanligvis ikke er mulig på mobile plattformer) er helt klart svært store.

RaphaëlJS, som er en kompatibilitetsløsning for å bruke enten SVG eller VML avhengig av nettleser (via JavaScript), har blitt tatt i bruk av bl.a. New York Times. Bibliotekets funksjonalitet gir web-utviklere en god mulighet til å starte med vektor-grafikk på deres nettsider allerede i dag. Men siden den kun kan støtte felles funksjonalitet i de to standardene vil den ikke la web-utviklere utnytte SVG fullt ut.

Da vi utarbeidet problemstillingen for oppgaven (oktober 2009) var fremtidsutsiktene for SVG pessimistiske fordi Microsoft ikke var ombord. Med Microsofts endringer i strategi fra mai 2010 har fremtidsutsiktene endret seg

helt og holdent. Fremtidsutsiktene for SVG ser nå svært positive ut. Om 3-5 år, når markedet mest sannsynlig har oppgradert til IE9, vil man ha helt nye muligheter for å lage intuitive og avanserte web-grensesnitt på tvers av nettleserimplementasjoner.

Vi er helt klart inne i en svært spennende tid for verdensveven (World Wide Web) som en leveranseplattform for applikasjoner og innhold.

6. Avsluttende ord

6.1 Modellapplikasjonen

Å utvikle en applikasjon for å generere SVG var utfordrende. SVG er en omfattende spesifikasjon og få overblikk over. Valget vi tok om å skrive kapittel 1 før vi utviklet SVG-generatoren[136] var nok svært smart, da det tvang oss til å sette oss inn SVG sin syntaks før vi startet med implementasjonen.

Under utviklingen av generatoren var det helt klart at det mest utfordrende aspektet i SVG var å forstå koordinatsystemet, og hvordan man utførte transformasjoner. Den andre store utfordringen var kalkulering av størrelse på tekst. I flere av bitene av diagrammet har vi benyttet oss av tilnærminger i steden for nøyaktige mål for tekst-størrelser. Hvis vi hadde benyttet et bibliotek som Batik ville nok denne biten vært svært mye enklere, fordi Batik har alle kalkulasjonene for beregning av størrelser implementert.

Hvis hovedformålet med applikasjonen ikke hadde vært læring ville vi nok ha benyttet Batik fra starten av. Det som er interessant å nevne er at selve genereringen av SVG-kode er relativt godt isolert, og det burde være relativt enkelt å bytte ut denne koden med funksjonalitet fra Batik. API-et generatoren tilgjengeliggjør benytter seg kun av standard Java-klasser som File og OutputStream for levering av XML-dataene. Implementasjonsdetaljer er helt skjult ved hjelp av riktig innkapsling.

Under utviklingen av biblioteket bygde vi en liten test-applikasjon[137] som gjorde det enkelt å teste ut at API-et i biblioteket var brukbart fra klientsiden. Dette gjorde det enkelt å sørge for at API-et var svært enkelt og ikke ble bundet for sterkt til implementasjonsdetaljer i klienten.

Parallellt med arbeidet på biblioteket/generatoren arbeidet vi på desktop-applikasjonen[138]. Denne applikasjonen bruker mønsteret Chain-Of-Responsibility (CoR) gjennomgående for å unngå kompleks overføring av variabler som igjen fører til høy kobling. Siden API-et fra biblioteket var designet med tanke på Model-View-Controller-mønsteret (MVC) var det svært enkelt å bygge opp denne applikasjonen etter det samme mønsteret. Swing-biblioteket til Java, som vi benyttet, baserer seg sterkt på MVC-mønsteret. Desktop-applikasjonen tok seg sånn sett av View og Controller-rollene, og benyttet klasser fra SVG-generatoren som Model. Utfordringen var sånn sett mest å koble opp forskjellige Layout-klasser for å få til den presentasjonen vi ønsket på brukergrensesnittet, samt å sørge for at endringer i visningsobjektene propagerte riktig til modellobjektene via kontrolleren.

6.2 Utviklingsprosess

Valget med å skrive dokumentasjonen som rene tekstfiler i Markdown-formatet, samt bruken av Git+GitHub for å administrere hele arbeidsflyten har vært svært

positiv. Vi har vært i stand til å jobbe parallellt med forskjellige biter av teksten, med full mulighet for å enkelt gå tilbake til gamle versjoner.Vi utviklet et lite program i Perl som fletter sammen alle dokumentene til en elegant XHTML-fil som igjen blir konvertert til PDF for utskrift. All kildekoden til modellapplikasjonen og denne rapporten er tilgjengelig på den vedlagte CD-platen (med revisjonshistorikk) eller på GitHub. Kildekoden til generatoren og desktop-applikasjonen inneholder også noen UML-diagrammer som viser forskjellige aspekter av programmene. Disse diagrammene er først og fremst produsert for å hjelpe oss under utviklingen av programvaren. De er ikke ment benyttet som sluttbrukerdokumentasjon. Dokumentasjonsmappen til generatoren inneholder også en svært elementær API-dokumentasjon i standard javadoc-format.

GitHub-lenker til de forskjellige delene av oppgaven nevnt ovenfor:

- Rapport[139]
- SVG generator[136]
- Desktop-applikasjon[138]
- Test-applikasjon[137]

Selv om vi benyttet LiquidPlanner.com[140] (LP) gjennom hele oppgaven og var svært flinke med å sette opp og estimere de forskjellige oppgavene vi skulle gjennom var det likevel utfordrende å komme i mål med oppgaven i tide. Den største utfordringen forekom under arbeidet med kapittel 1 og 2, som var planlagt utført før programmeringen skulle påbegynnes. Agile-prinsippet med sprinter hjalp oss kraftig til å se at risikoen økte for hver uke som gikk. LP sin metode med å gi varsel når man er på tampen til å ikke klare en tidsfrist (sprint-milepæler) hjalp oss til å komme i mål med et komplett arbeid i tide.

Bibliografi

1. **http://github.com**
github - social coding, open source code hosting service

2. **http://www.w3.org/Graphics/SVG/**
Scalable Vector Graphics - XML Graphics for the Web, W3C, 2009-12-11

3. **http://www.w3.org**
World Wide Web Consortium front page

4. **http://www.w3.org/2007/11/SVG_rechartering/SVG-WG-charter.html#coordination**
SVG Dependencies, W3C/Doug Schepers/Chris Lilley, 2008-04-16

5. **http://www.w3.org/standards/xml**
eXtensible Markup Language overview page, W3C

6. **http://en.wikipedia.org/wiki/Declarative_programming**
Declarative programming, Wikipedia, lest 2010-02-26

7. **http://www.w3.org/standards/webdesign/htmlcss**
HyperText Markup Language overview page, W3C

8. **http://www.ecmascript.org/**
ECMAScript - the language of the web, Ecma International

9. **http://www.w3.org/DOM/**
Document Object Model overview page, W3C/Philippe Le Hégaret/Ray Whitmer/Lauren Wood, 2009-01-06

10. **http://www.w3.org/TR/smil-animation/**
Synchronized Multimedia Integration Language W3C Recommendation, W3C, 2001-09-04

11. **http://www.w3.org/standards/webdesign/graphics**
Graphics overview page, W3C

12. **http://www.w3.org/Graphics/SVG/About.html**
About SVG - 2d Graphics in XML, W3C/Chris Lilley/Dean Jackson, 2004-10-29

13. **http://www.3gpp.org/**
3rd Generation Partnership Project front page, ETSI

14. **http://www.w3.org/2000/09/dbwg/details?group=19480&public=1&gs=1&**
SVG Working Group Participants, W3C/Gerald Oskoboiny/Dominique Hazaël-Massieux, 2010-04-09

15. **http://lists.w3.org/Archives/Public/public-svg-wg/**
SVG Working Group public mailing list archive, uthentet 2010-04-01

16. **http://lists.w3.org/Archives/Public/public-svg-ig/**
SVG Interest Group public mailing list archive, uthentet 2010-04-01

17. **http://www.institut-telecom.fr/p_en_present_inst_36.html**
Institut Télécom about page, lest 2010-04-01

18. **http://www.ikivo.com/04about.html**
Ikivo AB about page, lest 2010-04-01

19. **http://www.ikivo.com/open_standards.html**
Based on open standards, Ikivo AB, 2009-10-23

20. **http://www.w3.org/Graphics/SVG/WG/wiki/Secret_Origin_of_SVG**
The Secret Origin of SVG, W3C/Doug Schepers, 2008-09-15

21. **http://www.w3.org/TR/1998/NOTE-PGML-19980410**
Precision Graphics Markup Language (PGML) W3C note, Adobe/IBM/ Netscape/Sun, 1998-04-10

22. **http://www.w3.org/Submission/1998/08/**
Vector Markup Language specification, Autodesk/Hewlett-Packard/ Macromedia/Microsoft/Visio, 1998-05-13

23. **http://www.w3.org/TR/1998/NOTE-HGML-19980619**
Hyper Graphics Markup Language W3C note, PRP (UK)/Orange PCSL, 1998-06-19

24. **http://www.w3.org/Submission/1998/20/**
Draw Markup Language specification, Excosoft AB, 1998-11-12

25. **http://www.w3.org/Submission/1998/13/**
Web profile of Computer Graphics Metafile standard, ISO/IEC 8632:1992 specification, Boeing/CCLRC/Inso/JISC/Xerox, 1998-09-19

26. **http://www.w3.org/TR/SVG10/**
SVG 1.0 W3C Recommendation, W3C/Jon Ferraiolo, 2001-09-04

27. **http://www.w3.org/TR/SVG11/**
SVG 1.1 W3C Recommendation, W3C/Jon Ferraiolo/Fujisawa Jun/Dean Jackson, 2003-01-14

28. **http://www.w3.org/TR/SVGMobile/**
Mobile SVG Profiles: SVG Tiny and SVG Basic W3C Recommendation, W3C/ Tolga Capin, 2003-01-14

29. **http://www.w3.org/TR/SVGTiny12/**
SVG Tiny 1.2 W3C Recommendation, W3C, 2008-12-22

30. **http://www.w3.org/TR/SVG12/**
SVG Full 1.2 W3C Working Draft, W3C/Dean Jackson/Craig Northway, 2005-04-13

31. **http://www.w3.org/TR/SVGPrint12/**
SVG Print 1.2 Part II: Language W3C Working Draft, W3C, 2007-12-21

32. **http://www.w3.org/TR/SVGMobile12/feature.html#SVG-static**
SVG Tiny 1.2 Static Profile, W3C, 2008-12-22

33. **http://www.w3.org/TR/xml-names/**
Namespaces in XML 1.0 (3rd Edition), W3C/Tim Bray/Dave Hollander/ Andrew Layman/Richard Tobin/Henry S. Thompson, 2009-12-8

34. **http://www.w3.org/TR/SVG11/attindex.html**
SVG 1.1 attribute index, W3C, 2003-01-14

35. **http://www.w3.org/TR/SVGTiny12/mimereg.html**
SVG Tiny 1.2 mimetype registration, W3C, 2008-12-22

36. **http://tools.ietf.org/html/rfc3023#section-3.2**
application/xml mimetype registration - XML Media Types, M. Murata/IBM/ S. St. Laurent/simonstl.com/D. Kohn/Skymoon Ventures, 2001-01-XX

37. **http://www.w3.org/TR/SVG/intro.html#MIMEType**
SVG mime type, file name extension, W3C, 2003-01-14

38. **http://www.w3.org/TR/SVG11/intro.html#TermShape**
SVG 1.1 shape term, W3C, 2003-01-14

39. **http://www.w3.org/TR/SVG11/struct.html#SVGElement**
SVG 1.1 <svg> element, W3C, 2003-01-14

40. **http://www.w3.org/TR/SVG11/shapes.html#LineElement**
SVG 1.1 <line> element, W3C, 2003-01-14

41. **http://www.w3.org/TR/SVG11/styling.html**
SVG 1.1 styling properties, W3C, 2003-01-14

42. **http://www.w3.org/TR/SVG11/shapes.html#PolylineElement**
SVG 1.1 <polyline> element, W3C, 2003-01-14

43. **http://www.w3.org/TR/SVG11/styling.html#UsingPresentationAttributes**
SVG 1.1 specifying properties using the presentation attributes, W3C, 2003-01-14

44. **http://www.w3.org/TR/SVG11/styling.html#Inheritance**
SVG 1.1 styling inheritance rules, W3C, 2003-01-14

45. **http://www.w3.org/TR/SVG11/shapes.html#PolygonElement**
SVG 1.1 <polygon> element, W3C, 2003-01-14

46. **http://www.w3.org/TR/SVG11/shapes.html#RectElement**
SVG 1.1 <rect> element, W3C, 2003-01-14

47. **http://www.w3.org/TR/SVG11/paths.html#PathElement**
SVG 1.1 <path> element, W3C, 2003-01-14

48. **http://www.w3.org/TR/SVG11/paths.html#PathData**
SVG 1.1 path data, W3C, 2003-01-14

49. **http://www.w3.org/TR/SVG11/shapes.html#CircleElement**
SVG 1.1 <circle> element, W3C, 2003-01-14

50. **http://www.w3.org/TR/SVG11/shapes.html#EllipseElement**
SVG 1.1 <ellipse> element, W3C, 2003-01-14

51. **http://www.w3.org/TR/SVG11/text.html#TextElement**
SVG 1.1 <text> element, W3C, 2003-01-14

52. **http://www.w3.org/TR/SVG11/animate.html#TimingAttributes**
SVG 1.1 animation timing attributes, W3C, 2003-01-14

53. **http://www.w3.org/TR/SVG11/filters.html**
SVG 1.1 filter effects, W3C, 2003-01-14

54. **http://www.w3.org/TR/xlink/**
XML Linking Language 1.0 W3C Recommendation, W3C/Steve DeRose/Eve Maler/David Orchard, 2001-06-27

55. **http://www.w3.org/TR/SVG11/struct.html#DefsElement**
SVG 1.1 <defs> element, W3C, 2003-01-14

56. **http://www.w3.org/TR/SVG11/struct.html#UseElement**
SVG 1.1 <use> element, W3C, 2003-01-14

57. **http://www.w3.org/TR/SVG11/filters.html#feGaussianBlur**
SVG 1.1 <feGaussianBlur> filter effect, W3C, 2003-01-14

58. **http://www.w3.org/TR/SVG11/filters.html#feOffset**
SVG 1.1 <feOffset> filter effect, W3C, 2003-01-14

59. **http://www.w3.org/TR/SVG11/animate.html**
SVG 1.1 animation elements, W3C, 2003-01-14

60. **http://en.wikipedia.org/wiki/Scalable_Vector_Graphics#Software_and_support_in_applications**
SVG support in applications, Wikipedia, 2010-03-26

61. **http://office.microsoft.com/en-us/visio/FX100487861033.aspx**
Microsoft Office Visio 2007 front page, Microsoft, lest 2010-04-04

62. **http://www.svgopen.org/2003/papers/SVG_Scenarios_using_Microsoft_Office_Visio_2003/index.html**
SVG Scenarios for Microsoft Visio 2003, Richard See, 2003

63. **http://www.adobe.com/products/illustrator/**
Adobe Illustrator graphic design software, Adobe, lest 2010-04-04

64. **http://www.adobe.com/svg/tools.html**
Adobe SVG Authoring Tools, Adobe, lest 2010-04-04

65. **http://www.corel.com/servlet/Satellite/us/en/Product/1191272117978**
CorelDRAW Graphics Suite X5, Corel, lest 2010-04-04

66. **http://www.unleash.com/davidt/svg/index.asp**
SVG - From CorelDRAW to Your Browser, David Troidl, 2007

67. **http://corel.custhelp.com/app/answers/detail/a_id/754171/**
CorelDRAW SVG support information - Corel Knowledgebase, Corel, lest 2010-04-04

68. **http://www.xara.com/us/products/xtreme/**
Xara Xtreme graphics software, Xara Group Limited, lest 2010-04-04

69. **http://support.xara.com/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=2562**
Xara Xtreme SVG support - Xara knowledgebase, Xara Group Limited, 2007-06-13

70. **http://www.openoffice.org/product/draw.html**
OpenOffice.org Draw, Oracle/Sun, lest 2010-04-04

71. **http://graphics.openoffice.org/files/documents/12/406/svg_overview.htm**
OpenOffice.org Graphics SVG integration improvement proposal, Vincent Hardy/Kai Ahrens, 2002-07-01

72. **http://www.inkscape.org/**
Inkscape vector graphics editor, lest 2010-04-04

73. **http://wiki.inkscape.org/wiki/index.php/FAQ#What_SVG_features_does_Inkscape_implement.3F**
Inkscape SVG support, Inkscape Wiki, 2010-02-12

74. **http://www.scribus.net/**
Scribus Open Source Desktop Publishing, lest 2010-04-04

75. **http://docs.scribus.net/index.php?lang=en&page=scribus-svg**
Scribus SVG support, Scribus documentation, lest 2010-04-04

76. http://xmlgraphics.apache.org/batik/
Batik Java SVG Toolkit, Apache Software Foundation, 2010-01-02

77. http://cairographics.org/
Cairo 2D graphics library, 2010-03-02

78. https://wiki.mozilla.org/Gecko_1.9_Roadmap#cairo_Graphics_Substrate
Cairo usage in Mozilla Gecko, Mozilla wiki, 2007-01-10

79. http://processing.org/about/
Processing programming language, Ben Fry/Casey Reas, lest 2010-04-04

80. http://dev.processing.org/reference/core/javadoc/processing/core/PApplet.html
PApplet class definition, Processing Project, lest 2010-04-04

81. http://processing.org/reference/libraries/candy/SVG.html
Processing SVG support, Processing Project, 2008-09-22

82. http://dev.processing.org/reference/core/javadoc/processing/core/PShapeSVG.html
PShapeSVG class definition, Processing Project, lest 2010-04-04

83. http://www.adobe.com/svg/viewer/install/
Adobe SVG Viewer IE Plugin install page, Adobe, lest 2010-05-27

84. http://www.w3.org/Graphics/SVG/IG/
SVG Interest Group, W3C, lest 2010-05-27

85. http://www.codedread.com/svg-support-table.html
SVG Support in browsers, Jeff Schiller, uthentet 2010-03-14

86. http://gs.statcounter.com/#browser_version-ww-monthly-200902-201003-bar
StatCounter Global Stats, Browser version, February 2009 to March 2010

87. http://www.mozilla.org/projects/svg/status.html
Mozilla SVG Status, Mozilla SVG Team, lest 2010-03-25

88. http://www.w3.org/TR/SVG11/text.html
SVG 1.1 - Text module, W3C, 2003-01-14

89. http://www.w3.org/TR/SVG11/color.html
SVG 1.1 - Color module, W3C, 2003-01-14

90. http://www.w3.org/TR/SVG11/interact.html
SVG 1.1 - Interaction module, W3C, 2003-01-14

91. http://www.w3.org/TR/SVG11/linking.html
SVG 1.1 - Linking module, W3C, 2003-01-14

92. http://www.w3.org/TR/SVG11/fonts.html
SVG 1.1 - Font module, W3C, 2003-01-14

93. http://www.opera.com/docs/specs/svg/
SVG support in Opera 9, Opera, lest 2010-03-25

94. http://www.opera.com/docs/specs/presto25/html5/
Opera: HTML5 elements attributes and APIs support in Opera Presto 2.5, Opera, lest 2010-03-25

95. https://developer.mozilla.org/en/Gecko
Gecko - MDC, Mozilla, 2010-03-12

96. **http://webkit.org/**
The WebKit Open Source Project, WebKit, lest 2010-03-25

97. **http://webkit.org/projects/svg/status.xml**
WebKit SVG Status, WebKit, 2010-01-05

98. **http://blogs.msdn.com/b/ie/archive/2010/01/05/microsoft-joins-w3c-svg-working-group.aspx**
Microsoft joins W3C SVG Working Group, Microsoft IE Team Blog, 2010-01-05

99. **http://www.codedread.com/svg-support.php**
SVG Support in web browsers, Jeff Schiller, lest 2010-05-29

100. **https://bugzilla.mozilla.org/show_bug.cgi?id=216462**
Implement SVG (SMIL) Animation, Mozilla Bug Tracker, rapportert: 2003-08-17, sist endret: 2010-03-05, status: RESOLVED FIXED

101. **https://bugzilla.mozilla.org/show_bug.cgi?id=119490**
Implement SVG fonts, Mozilla Bug Tracker, rapportert: 2002-01-11, sist endret: 2010-05-25, status: ASSIGNED

102. **https://bugs.webkit.org/show_bug.cgi?id=15495**
SVGViewSpec DOM bindings aka SVGSVGElement.currentView is unimplemented, WebKit Bug Tracker, rapportert: 2007-10-13, sist endret: 2010-05-18, status: NEW

103. **https://bugs.webkit.org/show_bug.cgi?id=6034**
WebKit+SVG needs to support color-interpolation for gradients and opacity calculations, WebKit Bug Tracker, rapportert: 2005-12-10, sist endret: 2010-01-30, status: NEW

104. **http://www.oreillynet.com/mac/blog/2001/04/exploring_svg.html**
Exploring SVG, Jason McIntosh, 2001-04-30

105. **http://www.adobe.com/svg/eol.html**
Adobe to Discontinue Adobe SVG Viewer, Adobe, 2009-01-01

106. **http://en.wikipedia.org/wiki/Vector_Markup_Language**
Vector Markup Language, Wikipedia, lest 2010-05-25

107. **http://www.robweir.com/blog/2006/07/cum-mortuis-in-lingua-mortua.html**
VML and OOXML: Cum mortuis in lingua mortua, Rob Weir, 2006-07-24

108. **http://www.readwriteweb.com/archives/web_browser_faceoff.php**
Web Browser Faceoff, Alex Iskold, 2006-10-06

109. **http://gs.statcounter.com/#browser-ww-monthly-200807-201005**
StatCounter Global Stats, Browsers, July 2008 to May 2010

110. **http://www.sitepoint.com/blogs/2009/12/17/microsoft-browser-ballot-screen/**
Microsoft agrees to browser ballot terms, Craig Buckler, 2009-12-17

111. **http://gs.statcounter.com/#mobile_vs_desktop-ww-monthly-200812-201005**
StatCounter Global Stats, Mobile vs. Desktop, December 2008 to May 2010

112. **http://www.dlocc.com/articles/apple-ipad-no-flash-plugin-support/**
Apple iPad: No Flash plugin support, Devin Walker, 2010-01-27

113. http://tech.slashdot.org/story/10/01/06/1829223/Microsoft-Wants-To-Participate-In-SVG-Development
Microsoft wants to participate in SVG development, Slashdot, 2010-01-06

114. http://news.cnet.com/8301-30685_3-10426321-264.html
Microsoft Web-graphics move signals IE ambitions, CNet News, 2010-01-06

115. http://msdn.microsoft.com/en-us/ie/ff468705.aspx#_Scaling_Vector_Graphics
Microsoft IE9 will support SVG, Microsoft Developer Network, 2010-05-05

116. http://en.wikipedia.org/wiki/Adobe_Flash
Adobe Flash, Wikipedia, lest 2010-05-27

117. http://www.adobe.com/products/player_census/flashplayer/version_penetration.html
Adobe Flash plugin usage worldwide, Adobe, lest 2010-05-27

118. http://www.planetsvg.com/content/svg-solutions-internet-explorer
SVG solutions for Internet Explorer, PlanetSVG, 2009-02-15

119. http://www.adobe.com/support/security/#flashplayer
Adobe Security bulletins and advisories, Adobe, lest 2010-05-27

120. http://www.trusteer.com/files/Flash_Security_Hole_Advisory.pdf
Adobe Flash Security Hole Advisory, Trusteer, 2009-08-13

121. http://en.wikipedia.org/wiki/SWF
SWF, Wikipedia, lest 2010-05-27

122. http://www.w3.org/TR/SVG11/minimize.html
Minimize SVG File Size, W3C, 2003-01-14

123. http://en.wikipedia.org/wiki/Adobe_Flash_Player
Adobe Flash Player, Adobe, lest 2010-05-27

124. http://www.apple.com/hotnews/thoughts-on-flash
Thoughts on Flash, Apple, lest 2010-05-27

125. http://en.wikipedia.org/wiki/Microsoft_Silverlight
Microsoft Silverlight, Wikipedia, lest 2010-05-27

126. http://www.mono-project.com/Moonlight#Goals
Moonlight - Mono Goals, Mono Project, lest 2010-05-27

127. http://en.wikipedia.org/wiki/Rich_Internet_application
Rich internet applications, Wikipedia, lest 2010-05-28

128. http://www.smashingmagazine.com/2009/05/09/flash-vs-silverlight-what-suits-your-needs-best
Flash vs. Silverlight - Deployment Part, Smashingmagazine, 2009-05-09

129. http://www.microsoft.com/silverlight/what-is-silverlight
What Is Silverlight?, Microsoft, lest 2010-05-27

130. http://raphaeljs.com
Raphaël - JavaScript Library, Dmitry Baranovskiy, lest 2010-05-27

131. http://raphaeljs.com/reference.html
Raphaël Reference, Dmitry Baranovskiy, lest 2010-05-27

132. http://en.wikipedia.org/wiki/Imperative_programming
Imperative Programming, Wikipedia, lest 2010-05-27

133. http://blogs.msdn.com/b/ie/archive/2010/03/18/svg-in-ie9-roadmap.aspx
SVG in IE9 Roadmap, Jennifer Yu (Microsoft IE Team Member), 2010-03-18

134. http://www.samuelclay.com/raphael/svg_open_paper.pdf
SVG Open 2009 Raphaël Paper, Samuel Clay, 2009-07-14

135. http://msdn.microsoft.com/en-us/library/bb250524%28VS.85%29.aspx
Vector Markup Language(VML), Microsoft, lest 2010-05-27

136. http://github.com/robinsmidsrod/SVGChartLibrary
SVG Chart Generator Library, Robin Smidsrød, 2010-05-28

137. http://github.com/robinsmidsrod/SVGChartTest
SVG Chart Library test application, Robin Smidsrød, 2010-05-28

138. http://github.com/petterthunaes/SVGChartApp
SVG Chart Desktop Application, Petter Dahl Thunæs, 2010-05-28

139. http://github.com/robinsmidsrod/BAC309IN
Bacheloroppgave, Petter Dahl Thunæs/Robin Smidsrød , 2010-05-28

140. http://www.liquidplanner.com/
LiquidPlanner - Smart Project Management Software, LiquidPlanner Inc, lest 2010-05-28

Vedlegg

```
Date: Sun, 07 Mar 2010 19:00:24 -0500
From: Doug Schepers <schepers@w3.org>
To: Robin Smidsrød <robin@smidsrod.no>
Subject: Re: Historical archives of SVG WG membership and activities
```

Hi, Robin-

This sounds like a very interesting (if esoteric) project. I have some opinions about the reasons myself, but I will refrain from imposing them on you.

Regarding past records... the SVG WG has only been a public working group for the past couple of years; before that, it was member-confidential, and so many of the proceedings and details are not publicly available.

However, I'm sympathetic to your project, and I will see what I can legally and ethically do to aid you. I've forwarded your email to my colleagues on the W3C Team, and we will discuss it internally.

What sort of timeline are you working on? What are your deadlines?

Regards- -Doug Schepers W3C Team Contact, SVG and WebApps WGs

Robin Smidsrød wrote (on 3/7/10 2:25 PM):

> *I'm doing my bachelor assignment at Vestfold University College in Norway.*
>
> *My assignment is about SVG and why developers and users of web browsers still (in 2010) cannot expect SVG to be a standard that they can just assume works everywhere.*
>
> *As a part of my assignment I'm trying to figure out a bit of historical information about the members of the SVG WG from the start of the project until today. The web page, http://www.w3.org/2000/09/dbwg/details?group=19480&public=1&gs=1&, only lists current members og the SVG WG, but I could really need that same list, but all the way back to the start (maybe in one year intervals or anything you have, really).*
>
> *I noticed that the archives of public-svg-ig and public-svg-wg are only available/searchable for the last two years for the general public. Is it possible to get access to the entire archive as an mbox file, so I can do some data mining with regards to membership participation?*
>
> *The question I am really trying to find an answer to is which institution/company actually did the most work on the standard and when that*

happened. I'm also trying to figure out when certain companies caught an interest in the standard and decided to join the SVG WG.

Are the standardization documents tracked using some kind on revision control system that makes it easy to see who did what (and when)? Is it possible to get read-only access to this information if it exists?

Hope to hear from you soon.

Regards, Robin Smidsrød

Viewer	Firefox 1.5	Firefox 2.0	Firefox 3.0.0	Firefox 3.5	Firefox 3.6	Firefox 3.7 Dev	Opera 8.5	Opera 9.10	*Opera 9.50*	*Opera 10.01*	*Opera*
Date	3865700.0%	2006-10-01	2008-06-17	2009-08-28	2010-01-15	2010-02-08	2005-09-20	2006-12-01	2008-06-22	2009-09-01	2009-
Grade	F	F	C	C	C	B	F	A	A+	A+	A+
Score	44.89%	46.17%	60.40%	60.77%	61.50%	72.63%	47.45%	89.96%	94.16%	94.34%	94.71
1. color-prof-01-f	F	F	F	F	F	F	F	F	F	F	F
2. color-prop-02-f	P	P	P	P	P	P	P	P	P	P	P
3. extend-namespace-01-f	P	P	P	P	P	P	F	P	P	P	P
4. filters-conv-01-f	F	F	P	P	P	P	F	P	P	P	P
5. filters-diffuse-01-f	F	F	P	P	P	P	F	P	P	P	P
6. filters-displace-01-f	F	F	P	F	P	P	F	H	P	P	P
7. filters-light-01-f	F	F	P	P	P	P	F	P	P	P	P
8. filters-morph-01-f	F	F	P	P	P	P	F	P	P	P	P
9. filters-specular-01-f	F	F	P	P	P	P	F	P	P	P	P
10. filters-turb-01-f	F	F	P	P	P	P	F	P	P	P	P
11. interact-cursor-01-f	H	H	H	H	H	H	F	H	H	H	H
12. masking-intro-01-f	F	F	P	P	P	P	F	P	P	P	P
13. masking-path-05-f	P	P	P	F	P	P	F	P	P	P	P
14. painting-marker-01-f	P	P	P	P	P	P	F	P	P	P	P
15. painting-marker-02-f	P	P	P	P	P	P	F	P	P	P	P
16. painting-marker-03-f	H	H	P	P	P	P	F	P	P	P	P
17. paths-data-03-f	H	H	P	P	P	P	P	P	P	P	P
18. styling-css-04-f	P	P	P	P	P	P	F	F	P	P	P
19. text-tselect-02-f	F	F	F	F	F	F	F	H	P	P	P
20. animate-elem-22-b	F	F	F	F	F	P	P	P	P	P	P
21. animate-elem-29-b	F	F	F	F	F	F	P	P	P	P	P
22. color-prop-01-b	P	P	P	P	P	P	F	P	P	P	P
23. coords-trans-01-b	P	P	P	P	P	P	P	P	P	P	P
24. coords-units-01-b	H	H	P	P	P	P	F	P	P	P	P
25. coords-units-02-b	P	P	P	P	P	P	F	P	P	P	P

26. coords-units-03-b	H	H	H	P	P	P	F	P	P	P	P
27. coords-viewattr-01-b	P	P	P	P	P	P	F	P	P	P	P
28. coords-viewattr-02-b	P	P	P	P	P	P	F	P	P	P	P
29. coords-viewattr-03-b	P	P	P	P	P	P	F	P	P	P	P
30. filters-blend-01-b	F	F	P	H	P	P	F	P	P	P	P
31. filters-color-01-b	F	F	P	P	P	P	F	H	P	P	P
32. filters-composite-02-b	F	F	F	F	F	F	F	P	P	P	P
33. filters-comptran-01-b	F	F	P	P	P	P	F	F	P	P	P
34. filters-example-01-b	F	F	P	P	P	P	F	P	P	P	P
35. filters-felem-01-b	F	F	P	P	P	P	F	P	P	P	P
36. filters-gauss-01-b	F	F	P	P	P	P	F	P	P	P	P
37. filters-image-01-b	F	F	P	P	P	P	F	P	P	P	P
38. filters-offset-01-b	F	F	P	P	P	P	F	P	P	P	P
39. filters-tile-01-b	F	F	P	P	P	P	F	P	P	P	P
40. fonts-elem-03-b	F	F	F	F	F	F	F	P	P	P	P
41. fonts-elem-04-b	F	F	F	F	F	F	F	F	F	F	F
42. fonts-elem-07-b	F	F	F	F	F	F	F	F	F	F	F
43. interact-dom-01-b	P	P	P	P	P	P	F	P	P	P	P
44. interact-events-01-b	P	P	P	P	P	P	F	P	P	P	P
45. interact-order-01-b	P	P	P	P	P	P	F	P	P	P	P
46. interact-order-02-b	P	P	P	P	P	P	F	P	P	P	P
47. interact-order-03-b	H	H	H	H	H	H	F	P	P	P	P
48. linking-a-01-b	P	P	P	P	P	P	P	P	P	P	P
49. linking-a-02-b	P	P	P	P	P	P	P	P	P	P	P
50. linking-a-03-b	F	F	F	F	F	F	F	P	P	P	P
51. linking-uri-01-b	F	F	F	F	F	F	F	H	H	H	H
52. linking-uri-02-b	F	F	F	F	F	F	F	H	H	H	H
53. masking-mask-01-b	F	F	P	P	P	P	F	P	P	P	P
54. masking-opacity-01-b	P	P	P	P	P	P	P	P	P	P	P
55. masking-path-01-b	P	P	P	P	P	P	F	P	P	P	P
56. masking-path-02-b	P	P	P	P	P	P	F	P	P	P	P
57. masking-path-03-b	P	P	P	P	P	P	F	P	P	P	P
58. masking-path-04-b	P	P	P	P	P	P	F	P	P	P	P
59. metadata-example-01-b	P	P	P	P	P	P	F	P	P	P	P
60. painting-fill-05-b	P	P	P	P	P	P	P	P	P	P	P
61. painting-render-01-b	F	F	F	F	F	F	F	F	F	F	F
62. pservers-grad-01-b	P	P	P	P	P	P	P	P	P	P	P

63. pservers-grad-02-b	F	F	P	P	P	P	P	P	P	P	P
64. pservers-grad-03-b	F	F	P	P	P	P	F	P	P	P	P
65. pservers-grad-04-b	F	F	P	P	P	P	F	P	P	P	P
66. pservers-grad-05-b	F	F	P	P	P	P	F	P	P	P	P
67. pservers-grad-06-b	F	F	P	H	H	H	F	P	P	P	P
68. pservers-grad-07-b	P	P	P	P	P	P	P	P	P	P	P
69. pservers-grad-08-b	P	P	P	P	P	P	F	P	P	P	P
70. pservers-grad-09-b	P	P	P	P	P	P	P	P	P	P	P
71. pservers-grad-10-b	F	F	P	P	P	P	P	P	P	P	P
72. pservers-grad-11-b	F	F	P	P	P	P	F	P	P	P	P
73. pservers-grad-12-b	F	F	P	P	P	P	F	P	P	P	P
74. pservers-grad-13-b	P	P	P	P	P	P	F	P	P	P	P
75. pservers-grad-14-b	F	F	P	H	P	P	P	P	P	P	P
76. pservers-grad-15-b	F	F	P	P	P	P	F	P	P	P	P
77. pservers-grad-16-b	P	P	P	P	P	P	F	P	P	P	P
78. pservers-grad-17-b	F	F	P	P	P	P	P	P	H	H	H
79. pservers-grad-18-b											
80. pservers-grad-19-b	F	F	F	F	F	F	F	F	F	F	F
81. pservers-pattern-01-b	F	P	P	P	P	P	F	P	P	P	P
82. render-groups-01-b	P	P	P	P	P	P	H	P	P	P	P
83. script-handle-01-b	P	P	P	P	P	P	F	P	P	P	P
84. script-handle-02-b	F	F	F	F	F	F	F	F	H	H	H
85. script-handle-03-b	P	P	P	P	P	P	F	P	P	P	P
86. script-handle-04-b	P	P	P	P	P	P	F	P	P	P	P
87. struct-dom-01-b	P	P	P	P	P	P	F	P	P	P	P
88. struct-dom-02-b	P	P	P	P	P	P	F	P	P	P	P
89. struct-dom-03-b	P	P	P	P	P	P	F	P	P	P	P
90. struct-dom-04-b	P	P	P	P	P	P	F	P	P	P	P
91. struct-dom-05-b	P	P	P	P	P	P	F	P	P	P	P
92. struct-dom-06-b	P	P	P	P	P	P	F	P	P	P	P
93. struct-group-02-b	P	P	P	P	P	P	F	P	P	P	P
94. struct-image-02-b	F	F	F	F	F	F	F	P	P	P	P
95. struct-image-05-b	F	F	F	F	F	F	F	F	P	P	P
96. struct-symbol-01-b	P	P	P	P	P	P	F	P	P	P	P
97. struct-use-05-b	F	F	F	P	P	P	F	F	P	P	P
98. styling-css-01-b	P	P	P	P	P	P	F	P	P	P	P
99. styling-css-02-b	F	F	F	P	P	P	F	P	P	P	P

100. styling-css-03-b	P	P	P	P	P	P	F	P	P	P	P
101. styling-css-05-b	P	P	P	P	P	P	F	P	P	P	P
102. styling-css-06-b	F	F	H	H	H	H	F	P	P	P	P
103. styling-inherit-01-b	P	P	P	P	P	P	F	P	P	P	P
104. text-align-01-b	P	P	P	P	P	P	P	P	P	P	P
105. text-align-02-b	F	F	F	F	F	F	F	P	P	P	P
106. text-align-03-b	P	P	P	P	P	P	F	P	P	P	P
107. text-align-04-b	F	F	F	F	F	F	F	P	P	P	P
108. text-align-05-b	F	F	F	F	F	F	F	P	P	P	P
109. text-align-06-b	F	F	F	F	F	F	F	P	P	P	P
110. text-align-08-b	F	F	F	F	F	F	F	F	F	F	F
111. text-altglyph-01-b	F	F	F	F	F	F	F	F	H	H	P
112. text-deco-01-b	F	F	F	F	F	F	F	H	P	P	P
113. text-intro-02-b	F	F	F	F	F	F	F	P	P	H	H
114. text-intro-03-b	F	F	F	F	F	F	F	H	H	P	P
115. text-path-01-b	F	P	P	P	P	P	F	P	P	P	P
116. text-spacing-01-b	F	F	F	F	F	F	F	P	P	P	P
117. text-text-01-b	F	F	F	F	F	F	F	P	P	P	P
118. text-text-03-b	F	F	F	F	F	F	F	P	P	P	P
119. text-text-08-b	F	F	P	P	P	P	F	P	P	P	P
120. text-tref-01-b	F	F	F	F	F	F	F	P	P	P	P
121. text-tselect-01-b	F	F	F	F	F	F	F	P	P	P	P
122. text-tspan-01-b	P	P	P	P	P	P	F	P	P	P	P
123. types-basicDOM-01-b	F	F	P	P	P	P	F	H	P	P	P
124. animate-elem-02-t	F	F	F	P	F	P	P	P	P	P	P
125. animate-elem-03-t	F	F	F	F	F	H	P	P	P	P	P
126. animate-elem-04-t	F	F	F	F	F	F	P	P	P	P	P
127. animate-elem-05-t	F	F	F	F	F	F	P	P	P	P	P
128. animate-elem-06-t	F	F	F	F	F	F	P	P	P	P	P
129. animate-elem-07-t	F	F	F	F	F	F	P	P	P	P	P
130. animate-elem-08-t	F	F	F	F	F	F	P	P	P	P	P
131. animate-elem-09-t	F	F	F	F	F	H	P	P	P	P	P
132. animate-elem-10-t	F	F	F	F	F	H	P	P	P	P	P
133. animate-elem-11-t	F	F	F	F	F	H	P	P	P	P	P
134. animate-elem-12-t	F	F	F	F	F	H	P	P	P	P	P
135. animate-elem-13-t	F	F	F	F	F	P	P	P	P	P	P
136. animate-elem-14-t	F	F	F	F	F	P	P	P	P	P	P

137. animate-elem-15-t	F	F	F	F	F	P	P	P	P	P	P
138. animate-elem-17-t	F	F	F	F	F	P	P	P	P	P	P
139. animate-elem-19-t	F	F	F	F	F	P	P	P	P	P	P
140. animate-elem-20-t	F	F	F	F	F	F	P	P	P	P	P
141. animate-elem-21-t	F	F	F	F	F	F	P	P	P	P	P
142. animate-elem-23-t	F	F	F	F	F	F	P	P	P	P	P
143. animate-elem-24-t	F	F	F	F	F	H	P	P	P	P	P
144. animate-elem-25-t	F	F	F	F	F	P	P	P	P	P	P
145. animate-elem-26-t	F	F	F	F	F	P	P	H	P	P	P
146. animate-elem-27-t	F	F	F	F	F	P	P	P	P	P	P
147. animate-elem-28-t	F	F	F	F	F	P	P	P	P	P	P
148. animate-elem-30-t	F	F	F	F	F	FU	H	P	P	P	P
149. animate-elem-31-t	F	F	F	F	F	P	F	P	P	P	P
150. animate-elem-32-t	F	F	F	F	F	P	P	P	P	P	P
151. animate-elem-33-t	F	F	F	F	F	F	F	P	P	P	P
152. animate-elem-34-t	F	F	F	F	F	F	F	P	P	P	P
153. animate-elem-36-t	F	F	F	F	F	FU	H	P	P	P	P
154. animate-elem-37-t	F	F	F	F	F	P	P	P	P	P	P
155. animate-elem-39-t	F	F	F	F	F	F	F	P	P	P	P
156. animate-elem-40-t	F	F	F	F	F	H	H	P	P	P	P
157. animate-elem-41-t	F	F	F	F	F	P	F	P	P	P	P
158. animate-elem-44-t	F	F	F	F	F	F	P	P	P	P	P
159. animate-elem-46-t	F	F	F	F	F	FU	F	P	P	P	P
160. animate-elem-52-t	F	F	F	F	F	F	P	P	P	P	P
161. animate-elem-60-t	F	F	F	F	F	H	F	H	P	P	P
162. animate-elem-61-t	F	F	F	F	F	H	F	H	P	P	P
163. animate-elem-62-t	F	F	F	F	F	H	F	H	H	H	P
164. animate-elem-63-t	F	F	F	F	F	H	F	H	P	P	P
165. animate-elem-64-t	F	F	F	F	F	P	P	P	P	P	P
166. animate-elem-65-t	F	F	F	F	F	P	F	F	P	P	P
167. animate-elem-66-t	F	F	F	F	F	P	H	P	P	P	P
168. animate-elem-67-t	F	F	F	F	F	P	F	H	P	P	P
169. animate-elem-68-t	F	F	F	F	F	P	P	P	P	P	P
170. animate-elem-69-t	F	F	F	F	F	P	P	P	P	P	P
171. animate-elem-70-t	F	F	F	F	F	P	P	P	P	P	P
172. animate-elem-77-t	F	F	F	F	F	H	F	P	P	P	P
173. animate-elem-78-t	F	F	F	F	F	P	F	P	P	P	P

174. animate-elem-80-t	F	F	F	F	F	P	P	P	P	P	P
175. animate-elem-81-t	F	F	F	F	F	P	H	P	P	P	P
176. animate-elem-82-t	F	F	F	F	F	P	H	P	P	P	P
177. animate-elem-83-t	F	F	F	F	F	F	F	P	H	P	P
178. animate-elem-84-t	F	F	F	F	F	F	F	P	P	P	P
179. animate-elem-85-t	F	F	F	F	F	F	F	F	F	H	H
180. color-prop-03-t	P	P	P	P	P	P	P	P	P	P	P
181. coords-coord-01-t	P	P	P	P	P	P	P	P	P	P	P
182. coords-coord-02-t	P	P	P	P	P	P	P	P	P	P	P
183. coords-trans-02-t	P	P	P	P	P	P	P	P	P	P	P
184. coords-trans-03-t	P	P	P	P	P	P	P	P	P	P	P
185. coords-trans-04-t	P	P	P	P	P	P	P	P	P	P	P
186. coords-trans-05-t	P	P	P	P	P	P	P	P	P	P	P
187. coords-trans-06-t	P	P	P	P	P	P	P	P	P	P	P
188. fonts-desc-02-t	F	F	F	F	F	F	F	F	F	F	F
189. fonts-elem-01-t	F	F	F	F	F	F	P	P	P	P	P
190. fonts-elem-02-t	F	F	F	F	F	F	P	P	P	P	P
191. fonts-elem-05-t	F	F	F	F	F	F	P	P	P	P	P
192. fonts-elem-06-t	F	F	F	F	F	F	P	P	P	P	P
193. fonts-glyph-02-t	F	F	F	F	F	F	F	P	P	P	P
194. fonts-glyph-03-t	F	F	F	F	F	F	H	H	H	H	H
195. fonts-glyph-04-t	F	F	F	F	F	F	H	H	H	H	H
196. fonts-kern-01-t	F	F	F	F	F	F	F	H	H	P	P
197. interact-zoom-01-t	F	F	F	F	F	F	H	P	P	P	P
198. linking-a-04-t	P	P	P	P	P	P	P	P	P	P	P
199. linking-a-05-t	P	P	P	P	P	P	P	P	P	P	P
200. linking-a-07-t	F	H	H	H	H	H	F	H	H	H	H
201. linking-uri-03-t	P	P	P	P	P	P	P	P	P	P	P
202. painting-fill-01-t	P	P	P	P	P	P	P	P	P	P	P
203. painting-fill-02-t	P	P	P	P	P	P	P	P	P	P	P
204. painting-fill-03-t	P	P	P	P	P	P	P	P	P	P	P
205. painting-fill-04-t	P	P	P	P	P	P	P	P	P	P	P
206. painting-stroke-01-t	P	P	P	P	P	P	P	P	P	P	P
207. painting-stroke-02-t	P	P	P	P	P	P	P	P	P	P	P
208. painting-stroke-03-t	P	P	P	P	P	P	P	P	P	P	P

209. painting-stroke-04-t	P	P	P	P	P	P	P	P	P	P	P
210. painting-stroke-07-t	P	P	P	P	P	P	F	P	P	P	P
211. paths-data-01-t	P	P	P	P	P	P	P	P	P	P	P
212. paths-data-02-t	H	H	P	P	P	P	P	P	P	P	P
213. paths-data-04-t	P	P	P	P	P	P	P	P	P	P	P
214. paths-data-05-t	P	P	P	P	P	P	P	P	P	P	P
215. paths-data-06-t	P	P	P	P	P	P	P	P	P	P	P
216. paths-data-07-t	P	P	P	P	P	P	P	P	P	P	P
217. paths-data-08-t	P	P	P	P	P	P	P	P	P	P	P
218. paths-data-09-t	P	P	P	P	P	P	P	P	P	P	P
219. paths-data-10-t	P	P	P	P	P	P	P	P	P	P	P
220. paths-data-12-t	P	P	P	P	P	P	P	P	P	P	P
221. paths-data-13-t	P	P	P	P	P	P	P	P	P	P	P
222. paths-data-14-t	P	P	P	P	P	P	P	P	P	P	P
223. paths-data-15-t	P	P	P	P	P	P	P	P	P	P	P
224. render-elems-01-t	P	P	P	P	P	P	P	P	P	P	P
225. render-elems-02-t	P	P	P	P	P	P	P	P	P	P	P
226. render-elems-03-t	P	P	P	P	P	P	P	P	P	P	P
227. render-elems-06-t	P	P	P	P	P	P	P	P	P	P	P
228. render-elems-07-t	P	P	P	P	P	P	P	P	P	P	P
229. render-elems-08-t	P	P	P	P	P	P	P	P	P	P	P
230. render-groups-03-t	P	P	P	P	P	P	H	P	P	P	P
231. shapes-circle-01-t	P	P	P	P	P	P	P	P	P	P	P
232. shapes-circle-02-t	P	P	P	P	P	P	P	P	P	P	P
233. shapes-ellipse-01-t	P	P	P	P	P	P	P	P	P	P	P
234. shapes-ellipse-02-t	H	H	P	P	P	P	P	P	P	P	P
235. shapes-intro-01-t	H	H	P	P	P	P	P	P	P	P	P
236. shapes-line-01-t	P	P	P	P	P	P	P	P	P	P	P
237. shapes-polygon-01-t	P	P	P	P	P	P	P	P	P	P	P
238. shapes-polyline-01-t	P	P	P	P	P	P	P	P	P	P	P
239. shapes-rect-01-t	P	P	P	P	P	P	P	P	P	P	P
240. shapes-rect-02-t	P	P	P	P	P	P	P	P	P	P	P
241. struct-cond-01-t	P	P	P	P	P	P	P	P	P	P	P
242. struct-cond-02-t	P	P	H	P	P	P	P	P	P	F	F
243. struct-cond-03-t	F	F	P	P	P	P	P	P	P	P	P
244. struct-defs-01-t	P	P	P	P	P	P	P	P	P	P	P
245. struct-frag-01-t	P	P	P	P	P	P	P	P	P	P	P

246. struct-frag-02-t	P	P	P	P	P	P	P	P	P	P	P
247. struct-frag-03-t	P	P	P	P	P	P	P	P	P	P	P
248. struct-frag-04-t	P	P	P	P	P	P	P	P	P	P	P
249. struct-frag-05-t	F	F	P	P	P	P	F	P	P	P	P
250. struct-frag-06-t	P	P	P	P	P	P	P	P	P	P	P
251. struct-group-01-t	P	P	P	P	P	P	P	P	P	P	P
252. struct-group-03-t	P	P	P	P	P	P	F	P	P	P	P
253. struct-image-01-t	P	P	P	P	P	P	H	P	P	P	P
254. struct-image-03-t	P	P	H	P	P	P	H	P	P	P	P
255. struct-image-04-t	P	P	P	P	P	P	H	P	P	P	P
256. struct-image-06-t	P	P	P	P	P	P	F	P	P	P	P
257. struct-image-07-t	H	H	P	P	P	P	F	P	P	P	P
258. struct-image-08-t	P	P	P	P	P	P	H	P	P	P	P
259. struct-image-09-t	P	P	P	P	P	P	H	P	P	P	P
260. struct-image-10-t	P	P	P	P	P	P	H	P	P	P	P
261. struct-use-01-t	H	P	P	P	P	P	H	P	P	P	P
262. struct-use-03-t	P	P	P	P	P	P	P	P	P	P	P
263. styling-pres-01-t	F	F	F	F	F	F	F	F	F	F	F
264. text-fonts-01-t	F	F	P	P	P	P	P	P	P	P	P
265. text-fonts-02-t	P	P	P	P	P	P	P	P	P	P	P
266. text-fonts-03-t	F	F	P	P	P	P	P	P	P	P	P
267. text-intro-01-t	F	H	H	H	H	H	P	P	P	P	P
268. text-intro-04-t	H	H	H	H	H	H	P	P	P	P	P
269. text-intro-05-t	F	F	F	F	F	F	P	P	P	P	P
270. text-text-04-t	F	F	F	F	F	F	F	P	P	P	P
271. text-text-05-t	F	F	F	F	F	F	F	F	P	H	H
272. text-text-06-t	F	F	F	F	F	F	F	F	H	P	P
273. text-text-07-t	F	F	F	F	F	F	F	P	P	P	P
274. text-ws-01-t	P	P	P	P	P	P	P	P	P	P	P
275. text-ws-02-t	F	F	P	P	P	P	P	P	P	P	P
Date	**2005-11-01**	**2006-10-01**	**2008-06-17**	**2009-08-28**	**2010-01-15**	**2010-02-08**	**2005-09-20**	**2006-12-01**	**2008-06-22**	**2009-09-01**	**2009-**
Viewer	**Firefox 1.5.0.11**	**Firefox 2.0.1**	**Firefox 3.0.0**	**Firefox 3.5.2**	**Firefox 3.6.0**	**Firefox 3.7 DevPrev**	**Opera 8.5**	**Opera 9.10**	**Opera 9.50**	**Opera 10.01**	**Oper 10.1**
	Native Support										

Scores	**246**	**253**	**331**	**333**	**337**	**398**	**260**	**493**	**516**	**517**	**519**
Maximum Score	**548**	**548**	**548**	**548**	**548**	**548**	**548**	**548**	**548**	**548**	**548**
Percentage	**44.9%**	**46.2%**	**60.4%**	**60.8%**	**61.5%**	**72.6%**	**47.4%**	**90.0%**	**94.2%**	**94.3%**	**94.7%**
Number of Tests Run	**274**	**274**	**274**	**274**	**274**	**271**	**274**	**274**	**274**	**274**	**274**
Total Tests	**275**	**275**	**276**	**275**	**275**	**275**	**275**	**275**	**276**	**276**	**276**
Percentage Run	**99.6%**	**99.6%**	**99.3%**	**99.6%**	**99.6%**	**98.5%**	**99.6%**	**99.6%**	**99.3%**	**99.3%**	**99.3%**
Scores	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Maximum Score	**304**	**304**	**304**	**304**	**304**	**304**	**304**	**304**	**304**	**304**	**304**
Percentage	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**

[1]

JeffSchiller:

WebKitTools/Scripts/build-webkit --filters

Organisasjon	Person	SVG 1.0	SVG 1.1	SVG Mobile 1.1	SVG Tiny 1.2
Adobe	Jon Ferraiolo	X	X	X	X
Adobe	Richard Cohn	X	X		X
Adobe	Peter Sorotokin		X	X	X
Agfa	Luc Minnebo		X	X	X
AOL	Don Cone		X	X	X
Apple	Peter Graffagnino	X	X		X
Autodesk	Milt Capsimalis	X	X		X
BitFlash	Rick Graham	X	X	X	X
BitFlash	Andrew Emmons				X
Canon	Craig Brown	X	X	X	X
Canon	Jun Fujisawa	X	X	X	X
Canon	Rick Yardumian	X	X	X	X
Canon	Alex Danilo		X	X	X
Canon	Andrew Shellshear				X
Canon	Anthony Grasso				X
Canon	Craig Northway				X
Corel	Gavriel State	X	X		X
Corel	Haroon Sheikh	X	X		X
Corel	Phil Armstrong		X	X	X
Corel	Gordon Bowman				X
Ericsson	Charilaos Christopoulos		X	X	X
Ericsson	Henric Axelsson		X	X	X
Ericsson	Mathias Larsson Carlander		X	X	X
Excosoft	Jan Christian Herlitz	X	X		X
Expway	Robin Berjon		X	X	X
France Telecom	Vincent Mahe				X
Fuchsia Design (formerly of ILOG)	Antoine Quint		X	X	X
Groupe des Ecoles des Télécommunications (GET)	Cyril Concolato				X
Groupe des Ecoles des Télécommunications (GET)	Jean-Claude Moissinac				X
Hewlett-Packard	Robert Stevahn	X	X		X
Hewlett-Packard	Lee Klosterman		X	X	X
IBM	Andrew Donoho	X	X		X

IBM	Kelvin Lawrence	X	X		X
Ikivo / ZOOMON	Brad Sipes		X	X	X
Ikivo / ZOOMON	Jakob Cederquist		X	X	X
Ikivo / ZOOMON	Ola Andersson		X	X	X
Ikivo / ZOOMON	Andrew Sledd				X
Ikivo / ZOOMON	Niklas Hagelroth				X
ILOG S.A.	Christophe Jolif	X	X	X	X
ILOG S.A.	Thierry Kormann		X	X	X
Itedo / Corel	Benoît Bézaire		X	X	X
KDDI Research Labs	HAYAMA Takanari		X	X	X
KDDI Research Labs	KOBAYASHI Arei		X	X	X
KDDI Research Labs	SAGARA Takeshi		X	X	X
Kodak	Thomas E Deweese	X	X	X	X
Kodak	Timothy Thompson	X	X		X
Lexica	David Dodds	X	X		X
Macromedia	Brent Getlin	X	X		X
Macromedia	Peter Santangeli	X	X		X
Microsoft	John Bowler	X	X		X
Microsoft	Tuan Nguyen	X	X		X
Motorola	Bin Hu				X
Motorola	Christophe Gillette				X
Netscape	Kevin McCluskey	X	X		X
Netscape	Scott Furman	X	X		X
Nokia	Tolga Capin		X	X	X
Nokia	Michael Ingrassia				X
Nokia	Selim Balcısoy				X
Nokia	Suresh Chitturi				X
OASIS	Lofton Henderson	X	X	X	X
OpenText	Stephane Heintz				X
Openwave	Charles Ying		X	X	X
Opera	Håkon Lie	X	X		X
Opera	Charles McCathieNevile				X
Opera	Erik Dahlström				X

Oxford Brookes University	David Duce	X	X		X
Quark	Lee Cole	X	X	X	X
Quark	Shenxue Zhou	X	X		X
Quickoffice	Lee Martineau				X
RAL (CCLRC)	Bob Hopgood	X	X		X
Research In Motion Limited	Scott Hayman				X
Samsung	Atanas Zlatinski				X
SAP AG	Sebastian Schnitzenbaumer				X
Savage Software	Mike Bultrowicz		X	X	X
Schema Software	Philip Mansfield	X	X	X	X
Schema Software	Darryl Fuller		X	X	X
Schema Software	Yuri Khramov		X	X	X
Sharp Corporation	MINAKUCHI Mitsuru		X	X	X
Sharp Corporation	ONO Shuichiro		X	X	X
Sharp Corporation	UEDA Hirotaka		X	X	X
Streamezzo	Jean-Claude Dufourd				X
Sun Microsystems	Vincent Hardy	X	X	X	X
Sun Microsystems	Jerry Evans	X	X		X
Sun Microsystems	Nandini Ramani				X
Telecom Italia	Diego Gibellino				X
Visio	Troy Sandal	X	X		X
Vodafone	Bruno David Simões Rodrigues				X
Vodafone	Lars Piepel				X
W3C	Chris Lilley	X	X	X	X
W3C	Dean Jackson	X	X	X	X
W3C	ISHIKAWA Masayasu		X	X	X
W3C	Cameron McCormack				X
W3C	Doug Schepers				X
W3C	Ivan Herman				X
W3C	Olaf Hoffmann				X
Xerox	Alan Hester	X	X		X