



Scalable Vector Graphics (SVG) Requirements

W3C Working Draft, 29 Oct 1998

This version:

<http://www.w3.org/TR/1998/WD-SVGReq-19981029>

Latest version:

<http://www.w3.org/TR/WD-SVGReq>

Previous (member only) version:

<http://www.w3.org/Graphics/SVG/Group/1998/10/SVGReq-981019>

Editor:

[Jon Ferraiolo](#), Adobe Systems Incorporated

Status of this document

This document is a work in progress representing the current consensus of the W3C Scalable Vector Graphics Working Group. This draft of the SVG Requirements document has been approved by the SVG working group to be posted for review by W3C members and other interested parties. It is the first public review draft of this document. Publication as a working draft does not imply endorsement by the W3C membership.

Review comments from the public should be sent to www-svg@w3.org, which is an automatically [archived](#) email list. Information on how to subscribe to public W3C email lists can be found at <http://www.w3.org/Mail/Request>.

Introduction

The W3C has chartered a Scalable Vector Graphics working group to produce a specification for an SVG format, written as a modular XML tagset and usable as an XML namespace, which can be widely implemented in browsers and authoring tools and which is suitable for widespread adoption by the content authoring community as a replacement for many current uses of raster graphics.

This will mean that the graphics in Web documents will be smaller, faster, more interactive, and be displayable on a wider range of device resolutions from small mobile devices through office computer monitors to high resolution printers. This will be a significant advance in Web functionality. Interoperability, both in multiple browsers across multiple platforms and in multiple authoring tools (both read and write), is a prime focus.

The SVG working group decided to solicit public review and feedback at the earliest possible point to best ensure that SVG meets the Web community's needs for a vector

graphics language specification. This document reflects early SVG working group discussions on what SVG should and should not be. The working group has not reached consensus on all topics, so the document below sometimes describes particular features as open issues that are still under discussion.

This document lists both SVG Design Goals and SVG Detailed Requirements. The SVG Design Goals describe the high-level objectives which SVG should attempt to achieve. These design goals should also act as the criteria by which proposed features are judged. The SVG Detailed Requirements contains the actual list of proposed features.

A first draft of the detailed specification for SVG will be made available a couple of months after the posting of this requirements document for public review. The specification will be developed largely by looking at:

- the design goals and detailed requirements that are contained in this document
- review comments on this document from public feedback, invited experts and working group members

The home page for the W3C graphics activity is <http://www.w3.org/Graphics/Activity>.

[Table of Contents](#)

[SVG Design Goals](#)

[Open specification](#)

[Widely implemented and supported](#)

[Relationship to other Web standards efforts](#)

[Graphics features](#)

[Interactivity and Dynamic Behaviors](#)

[Interchange features](#)

[SVG Detailed Requirements](#)

[General requirements](#)

1. [Consistent visual results and behaviors across implementations](#)
2. [Elegant, uniform, complete and consistent](#)
3. [Packaging](#)
4. [Performance](#)
5. [Alternate representations](#)
6. [Backward compatibility](#)
7. [Well internationalized](#)

8. [Accessibility features](#)

[Graphical facilities](#)

9. [Vector graphics shapes](#)
10. [Text data](#)
11. [Image data](#)
12. [Color support](#)
13. [Transparency support](#)
14. [Grouping and layering](#)
15. [Template objects/symbols](#)
16. [Fill options](#)
17. [Stroke options](#)
18. [Transformations](#)
19. [Coordinate systems, relationship to CSS positioning](#)
20. [Antialiasing](#)
21. [Stenciling and masking](#)
22. [Client-side filter effects such as shadowing](#)
23. [Compositing](#)
24. [CSS support](#)
25. [Connectable reference points](#)
26. [Parameter substitution and formulas](#)
27. [Print control](#)

[Interaction](#)

28. [Zoom and pan](#)
29. [Links](#)
30. [Event handling](#)
31. [Object selection, clipboard](#)
32. [DOM access](#)
33. [Animation](#)

[Miscellaneous](#)

34. [Inclusion of private data \(metadata\)](#)
35. [Extensibility](#)
36. [Embedded fonts and images](#)

SVG Design Goals

The following are the Design Goals for SVG. Besides providing a set of high-level objectives for SVG, these goals act as the criteria by which proposed features are judged. Thus, the features list shown below under SVG Detailed Requirements should reflect the higher-level goals listed here.

These SVG Design Goals are not listed in any particular order. It is recognized that some of the goals might conflict or be unachievable and that tradeoffs will need to be made.

Open specification

- A. The specification should be controlled by the members in the W3C, not by a single vendor. Thus, the specification should not subject to sudden change by a single vendor.
- B. The specification should be vendor neutral and thus should not contain features biased towards a particular vendor.

Widely implemented and supported

- C. SVG should be a standard feature in Web browsers
- D. Implementations of SVG should be consistent so that the same visual results and behaviors exist in all conforming SVG processors.
- E. There should not be subset problems and incompatible generator/reader sets. Thus, there should be a single language specification, not a set of layered language specifications.
- F. There should be widespread support in authoring applications and related tools
- G. To promote widespread adoption and support, SVG should be specified to be as basic and simple as possible while still providing necessary features to satisfy the needs of graphics on the Web. While the chief goal is to aim at the middle ground, a basic and simple feature set will allow it to be used on devices with a wide range of resolutions and capabilities, from small mobile devices through office computer monitors to high resolution printers.
- H. Straightforward generation via hand-authoring with a text editor or server-side scripts (e.g., CGI)
 - I. SVG should be as self-contained as possible. While SVG should leverage and be compatible with other W3C work, it should attempt to do so without introducing excessive dependencies on other specifications.
- J. Ready availability to the casual implementor is desirable
- K. Reference source code is desirable

Relationship to other Web standards efforts

- L. Defined as an application of XML
- M. Compatible with and/or leverages other relevant standards efforts, including XML namespaces, XML links (XLink), DOM, CSS, XSL and metadata. For example:
 - the elements and attributes of an SVG drawing should be scriptable via the DOM
 - text should be expressed as XML character data so that it can be found by search engines
 - attributes which make sense to be part of a style should be expressed in CSS

The SVG working group will need to coordinate proactively with other working groups when it is more appropriate to meet the SVG requirements through modifications to other Recommendations.

Graphics features

- N. Complete, general-purpose Web vector graphics format that satisfactorily meets the graphics delivery needs for all creators and consumers of Web vector graphics
- O. Sufficiently powerful and precise to meet the needs of professional Web graphic designers such that they will utilize SVG instead of raster formats in those cases where vector graphics is a more natural or appropriate format
- P. Sufficiently powerful to meet the needs of business presentation and diagramming applications such that these drawings will be published on the Web using SVG instead of raster formats
- Q. Feature set is rich enough such that a reasonable conversion of most existing standard static graphics files into SVG is possible
- R. Sufficiently compatible with the graphics design and publishing industries' feature sets and file formats such that there is (as lossless as possible) a straightforward mapping from these applications and file formats into SVG. The goals are to facilitate conversion of existing artwork into SVG, promote the creation of lots of compelling new SVG artwork, make it as easy as possible for the graphics design and publishing industries to adapt existing authoring tools, and provide for new SVG authoring tools.
- S. Compatible with the needs of non-technical persons who want a straightforward way to generate relatively simple graphics either by hand-editing or via server-based graphics generation (e.g., CGI scripts).
- T. Compatible with high-quality, efficient printing at full printer resolution and with reliable color
- U. Focused on basic foundation, presentation-level graphics facilities, with a critical eye toward higher-level (model-level) constructs which might be better suited to a higher-level standard which would sit on top of SVG
- V. The working group is investigating whether other Web standards (e.g., CSS, XSL, MathML, Web Schematics) could use SVG as its low-level rendering specification. (This goal has significant dependencies on other W3C initiatives such as the W3C Formatting Model and might change accordingly.)
- W. The specification for SVG should take into account the possibility of building a future 3D graphics specification which either sits on top of SVG or which is entirely independent but with a similar syntax; however, the SVG working group should not let itself be slowed down or constrained by 3D upgrade path issues.

Interactivity and Dynamic Behaviors

- X. Allows for interactive and dynamic Web pages

Interchange features

- Y. Suitable as a platform-independent graphics exchange format
- Z. Mechanisms for inclusion of application-specific (or industry specific) private data which would facilitate use of SVG as a application-specific (or industry-specific) native file format for authoring applications
- AA. Capable of use as a print metafile: sufficiently expressive such that a higher-level print metafile XML grammar could use SVG for its page imaging operators

SVG Detailed Requirements

The following is the detailed list of required features and capabilities in SVG. Items which are already listed as a Design Goal are not repeated here. It is recognized that some of these requirements might conflict or may not be possible.

General requirements

1. Consistent visual results and behaviors across implementations

- a. The specification needs to be complete and unambiguous
- b. The goal is to have at least two independent implementations each of SVG viewers and SVG generators under development as the specification is defined to help find and remove ambiguities from the specification.
- c. An extensive conformance test suite needs to be delivered in conjunction with the final specification.

2. Elegant, uniform, complete and consistent

- a. Wherever possible, all graphical attributes and operations should be available to all object types. For example, if 2x3 transformation matrices are available for shape objects, then 2x3 transformation matrices should also be available for text and image objects.

3. Packaging

- a. Stand-alone packaging: Compatible with the need of stand-alone graphics authoring packages. It should be possible for a graphics authoring application to save a stand-alone SVG file. ("Stand-alone packaging" means that an SVG file is saved as a separate, self-contained file, probably with a .svg extension on platforms where the extension is important.)
- b. Fragment packaging: Compatible with the need of consolidated full-page Web authoring packages which would like to author/deliver full pages of XML with optional vector graphics elements interspersed as isolated components within the page. ("Fragment packaging" means that snippets of SVG would appear inside the content of a parent XML grammar. For example, a picture of an arrow expressed in SVG might appear inline within an XML file. The expected way this would be achieved would be by referencing the SVG namespace within the XML file.)

4. Performance

- a. Reasonable display performance on commonly used end-user machines
- b. Highly efficient representation to minimize download times
 - i. The working group should investigate abbreviation schemes (particularly for path data) to minimize file sizes
 - ii. The working group should also collaborate with other working groups, such as the XML working group, on binary compression alternatives.
- c. Streamable (i.e., support progressive rendering)
- d. Simple primitives (e.g., lines) should exhibit fast performance

5. Alternate representations

- a. The working group should investigate the issue of alternative representations in the form of text strings or image data. Alternate representations as text or images might provide a good answer for backwards compatibility in some cases or provide for a faster display option for some drawings. Alternate text

strings would be particularly valuable to visually impaired users.

6. Backward compatibility

- a. The working group should develop a reasonable backward compatibility strategy for when a user attempts to view an SVG drawing in a browser which doesn't yet support SVG.

7. Well internationalized

- a. By virtue of being written in XML, SVG will already have baseline internationalisation capability (Unicode characters, language tagging). The working group will collaborate with the I18N working group to ensure that SVG is suitable for worldwide Web graphics.

8. Accessibility

- a. The working group should ensure that adaptive interfaces for people with disabilities are fully supported, and that mapping SVG content to these contexts is easy and graceful.

Graphical facilities

9. Vector graphics shapes

(Note: in the remainder of this document, the terms "vector graphic shape" will be abbreviated to "shape" or "shape objects")

- a. Path data
 - i. Paths can be made up of any combination of the following:
 - Straight line segments
 - Cubic bezier curved segments
 - Quadratic bezier curved segments
 - Elliptical and circular arcs
 - No other curve types (Other curve types such as splines or nurbs are either technically very difficult, industry-specific and/or have not established themselves as industry standards as much as beziers.)
 - ii. Compound paths (i.e., paths with donut holes) should be supported. Both the non-zero winding and even-odd fill rules should be supported in both fill and clip operations to ensure compatibility with the design and publishing industries' existing file formats and authoring applications. (Also, it is much easier to implement these fill rules inside a renderer than it is in a file format converter.)
- b. A set of predefined shape types such as rectangles, rounded rectangles, circles and ellipses should be available so that simple objects can be defined without having to learn bezier mathematics. The exact list of predefined object types has not been decided yet.
- c. Any shape can be filled, stroked or used as a clipping path and/or mask. (See [Fill options](#), [Stroke options](#) and [Stenciling and masking](#), below.)

10. Text data

- a. Text strings should be expressed as XML character data which could be marked up with arbitrary XML name spaces. (Text strings as XML character data allow search engines to find the strings.)
- b. All text/font attributes from CSS should be supported. Complete support for CSS is what the Web community will expect. (See [CSS support](#), below)

- c. It is clear that precise text sizing/positioning is a requirement for the graphics design community. Thus, SVG should allow for precise control of text sizing, positioning, rotation, and skewing on a character-by-character basis. In particular, the working group should look to see if this feature could be packaged as to provide for precise text on a path.
- d. It is clear that precise control of fonts and glyphs is a requirement for the graphics design community. It should be possible to achieve precise control of the exact font and the exact glyphs within a given font to use for a given character sequence so that graphic artists have a way to ensure that the end user sees what was intended
- e. The same fill/stroke/clip/mask capabilities that can be used with vector data should also be available to text data. For example, it should be possible to fill a text string with a gradient or stroke it with a dashed line pattern. This is a widely used feature in the graphics design world and is required for compatibility with existing graphics content.
- f. The working group hasn't investigated yet whether it is advisable to provide a capability to automatically position a text string relative to another graphical element (e.g., automatically centering a text string within a rectangle). It is clear that such a feature would be a great convenience for hand-coders of certain types of drawings, such as diagrams. The working group should investigate whether there is an easy-to-use (for a hand-coder) yet robust and elegant way of achieving this.
- g. Having a text string determine the dimensions of a parent graphical object (e.g., a box is drawn to fit around a text string) is too high-level of a construct and is not planned for version 1 of SVG.
- h. Adding a text-on-a-path capability to SVG is still under consideration. The working group recognizes that there are many technical complexities which make it difficult to design the feature in such a way that it will be both sufficiently powerful and sufficiently compatible with the needs of various authoring scenarios.

11. Image data

- a. The same image formats available to HTML should be available to SVG
- b. The working group hasn't had time yet to address the issue of color management on image data. The goals will be to:
 - i. achieve compatibility with how other specifications (e.g., HTML) perform color management on image data
 - ii. (just like the rest of SVG) take full advantage of color management systems when supported by the browser and/or the platform operating system (see [Color support](#) below)
 - iii. if different ICC profile are specified both as an attribute on an image and embedded within the image, the attribute should win

12. Color support

- a. All colors need to be specified in the sRGB color space for compatibility with other Web standards.
- b. It is clear that reliable color is important to both end users and Web content creators. sRGB does not answer all precise color needs, and most desktop platforms support the more complete ICC-based color management system. Thus, an alternative ICC-based color representation should be available for all color attributes. If the SVG processor supports color management, then the ICC color takes precedence over the sRGB color.
- c. There has been little discussion so far on sport color support. Spot colors

might be possible by specifying then using ICC profiles (perhaps via a Named Color profile), or by storing spot color information as private data (metadata) in the SVG file.

13. Transparency support

Transparency support is becoming commonplace in authoring applications and is widely used today in Web animations.

- a. Many existing authoring applications achieve transparency by adding an opacity property wherever a color property is allowed. For compatibility with these applications, and because this represents a good technical solution to many transparency needs, wherever a color property is allowed, there will be a corresponding opacity property which indicates how transparent a given object/component/attribute should be when blended into its background
- b. In group opacity, the collection of objects that makes up a group is (in effect) drawn into a temporary area in memory, and then the temporary area is blended as a single unit into the background graphics. Group opacity is necessary when you have an aggregate object such as a automobile which is drawn as a collection of overlapping, opaque components (e.g., the hubcap might be an opaque circle that is drawn on top of an opaque tire) and you want to blend that object as a unit into the background. Group opacity is a requirement for many animation applications and has utility with static graphics also. It is straightforward to implement, particularly once you have support for transparent image objects, which is needed for GIF and PNG. It is infeasible to achieve these effects without this feature. Thus, group opacity should be supported in SVG.
- c. The working group decided that identification of a transparency/blue-screen /chromaKey color will not be supported for shapes and text because the opacity features are sufficient. However, the transparency color feature that exists in image file formats such as GIF and PNG will be supported.
- d. See [Compositing](#), below.

14. Grouping and layering

- a. There needs to be an ability to define groups (i.e., collections) of objects to allow for convenient access for scripting applications. Groups should have the following attributes (among others), all of which are accessible and/or controllable via scripting:
 - i. Unique ID
 - ii. z-index for explicit layering
 - iii. visibility
 - iv. transformations
 - v. opacity
- b. Implicit layering: unless an explicit z-index value is provided, objects will be drawn bottom-to-top in the order which they appear in the input stream. This strategy is compatible with most popular graphics file formats.
- c. Explicit layering: objects or collections of objects can be assigned an explicit z-index value, which takes precedence over the implicit bottom-to-top drawing order. (The bottom-to-top drawing order of objects with the same z-index relative to each other is the the order in which they appear in the input stream.) This explicit layering is a requirement for achieving certain types of animation effects.

15. Template objects/symbols

- a. There should be a capability for defining template objects that can be instantiated multiple times with different attributes.
- b. The template objects should be able to establish a full set of default attributes, and the instantiated objects should be able to override any of these default attributes.

16. Fill options (i.e., painting the interior of a shape or text object)

The following fill options should be available:

- a. Solid-color fill (see [Color support](#) above)
- b. Gradients
 - i. Multiple gradient stops will be supported (not just two)
 - ii. Both linear and radial gradients will be supported
 - iii. It is still an open issue within the working group whether additional gradient types should be supported. Examples: conical, elliptical, square, rectangular, Gouraud shaded triangle and rectangular meshes.
 - iv. Gradients should allow for both a linear ramp between gradient stops and a well-defined sigma function for the calculation of the intermediate colors between the stops.
 - v. For all gradients, the specification should provide details for the exact behavior of the gradient ramp. If there is a disagreement about what the gradient ramps should be, priority should be given to: (1) making sure the gradient ramp rules are consistent with real-world lighting effects, (2) retaining compatibility with as much existing content as possible (with a bias towards existing content whose users really care about how their gradients look)
 - vi. The suggestion has been made to allow for substitution of a different color table for use with a given gradient. (The color stop positions stay the same, but the colors used change.) This suggestion hasn't been discussed yet in the working group.
- c. Patterns (i.e., tiled object fill)
 - i. Any objects (i.e., shapes, images, text) can be used to fill any other objects
 - ii. When objects are used to fill other objects, parameters can be set to achieve tiling effects.
 - iii. The tiling options should be at least as capable as those found in the file formats used by the design and publishing industries to ensure compatibility with existing artwork and authoring applications.
- d. Other fill styles - The working group hasn't decided yet whether other fill styles such as fractal patterns are appropriate.
- e. Custom fill styles - See [Extensibility](#), below.

17. Stroke options (i.e., drawing the outline of a shape or text object)

- a. The same attributes available for filling an object (see [Fill options](#), above) should be available to stroke an object. For example, you should be allowed to stroke with a pattern.
- b. The set of stroke options should be at least as capable as the stroke options in the file formats used by the design and publishing industries
 - i. Arbitrary, continuous values for line width
 - ii. The working group hasn't reached consensus yet on whether SVG should support hairlines (i.e., instructing the device to draw the thinnest possible line)

- iii. User-defined dash patterns and initial phase offset
- iv. Caps, joins and miterlimits
- c. Arrowheads and polymarkers
 - i. There should be a small set of predefined arrowheads
 - ii. Arrowheads are optional and can be placed at the start and/or end of path objects
 - iii. The working group should investigate the possibility of providing for custom "arrowheads" at the start and/or end of a path object which are defined by referencing a different graphical object or group. This capability would solve the following needs:
 - Diagramming applications might use arrowheads extensively, but the predefined arrowheads might not be satisfactory
 - Graphic designers who want to attach arrowheads to drawings will want complete control of the visual characteristics of their arrowheads
 - This feature can be leveraged to provide a polymarker capability for applications such as scatter diagrams

18. Transformations

- a. Arbitrarily nested 2x3 matrix transformations should be available. This facility is necessary to ensure compatibility with existing artwork and authoring tools.
 - i. 2x3 matrices are sufficient to achieve any types of translation, scaling, rotation and skewing
 - ii. Transformation matrices can be nested to an arbitrary depth. A given transformation matrix is concatenated with the transformation matrices of all parent objects.
- b. In the spirit of making SVG reasonably easy to use for content creators who would have difficulty constructing 2x3 matrices, SVG should offer an alternative set of simpler transformation attributes (e.g., rotation=, scale=). The SVG specification, however, needs to defined unambiguously what should happen if both the simpler transformation attributes and a 2x3 transformation matrix is provided.
- c. These transformations should apply to all object types (shapes, images and text) in a uniform and consistent manner
- d. Transformed objects should have the option of exhibiting true scalability (where all attributes should scale uniformly, and linewidths, images and text should scale along with the sizes of the shapes). Additionally, transformed objects should also have the option of selective scalability such that certain attributes (e.g., stroke size and textsize) are invariant.

19. Coordinate systems, relationship to CSS positioning

- a. SVG should use CSS positioning for establishing a viewport for an outermost SVG element, such as an <svg>...</svg> complete drawing or an SVG fragment within an XML Web page.
- b. SVG will support local user coordinates
- c. Real number values (i.e., an integer followed by a possible decimal fractional component) should be possible for all appropriate attributes and coordinates. The language definition itself should not inhibit infinite precision.

20. Antialiasing

- a. It is clear that the graphics design community not only wants antialiasing to be available, but also wants to have the ability to turn antialiasing on/off on an object basis in order to achieve precise control of the rendering and possibly to control drawing speed. Thus, there will be an antialias control

attribute on each graphics object in SVG.

- b. Antialiasing adds a significant burden to the casual implementor and isn't a requirement for all potential applications of SVG, such as viewing CAD drawings. Thus, antialiasing control will be regarded as a hint to the SVG processor, which can choose whether or not to honor it. Major general-purpose implementations of SVG processors such as commercial Web browsers should honor and implement the antialiasing control hint.

21. Stenciling and masking

- a. Clipping paths

Clipping paths are a commonly used feature in existing artwork and an integral part of all authoring products used by graphic designers. Clipping paths are as fundamental to the design and publishing industries' existing file formats and authoring tools as are nested transformation matrices. Clipping is relatively easy to implement in viewers using a raster mask approach in device space if you have code for drawing a filled path into an offscreen buffer. Nested clipping paths are easy to achieve on top of an unnested clipping path implementation by just looping through each parent clipping path one after another. The performance overhead with nested clipping paths in animation scenarios is probably small relative to other necessary computations (e.g, filling and stroking) and can be overcome on the user end by simply avoiding/minimizing the use of the feature. Without nested clipping paths, converters from the design and publishing industries' existing file formats will have a significant burden in determining how to flatten the nested clipping paths they encounter. Nested clipping paths are necessary to prevent lossless translation from the design and publishing industries' existing file formats.

- i. Any shape or text object can be used as a hard-edge clipping path
 - ii. Clipping paths can be nested arbitrarily to ensure compatibility with existing artwork and authoring tools.
- b. Masks

8-bit masking is a fundamental feature on all operating systems, and a key feature for both static and dynamic Web pages.

- i. Any graphics object (i.e., shape, text, or image) can be used as an 8-bit alpha mask to control the alpha-compositing of a different object (or group/collection of objects) into other objects

22. Client-side filter effects such as shadowing

Client-side filters provide for the possibility of significant file size and download time savings in many applications. An example is text drawn with a glow effect and a drop shadow. If client-side glow and drop shadow filters were available, then only the text string and the names of the filters would need to be downloaded, instead of today, where the text needs to be converted to a raster by the author.

- a. SVG should include a set of built-in client-side filter effects for commonly used effects such as shadowing. The definitions of these effects should be unambiguous so that all implementations produce the same visual results.
- b. Additionally, the working group should investigate the feasibility of a general filter mechanism which would allow for custom filter effects to be downloaded

and applied to a given graphical object (see [Extensibility](#), below)

23. Compositing

Compositing means the rules by which a foreground object's colors are blended into a background object's colors. There are many different approaches to compositing. Alpha compositing (where each pixel has an 8-bit alpha channel which determines how opaque that pixel is) is the most common, but even this method has multiple variations.

- a. The working group needs to define standard rules for how alpha compositing should work. Which color space (sRGB? Lab?)? What are the exact formulas?
- b. The working group needs to decide whether SVG should offer multiple compositing options as standard features. Should SVG support all of the Porter/Duff compositing options? Additionally, Photoshop offers many Adjustment Layer options. Should these be supported?
- c. See [Extensibility](#), below.

24. CSS support

- a. All CSS text/font attributes from the most recently approved CSS recommendation should be honored by all conforming SVG processors.
- b. The issue of CSS inheritance is still open. It is unclear that the standards world will have addressed how a parent XML/HTML grammar passes its current style table to a child grammar. Until this issue is addressed, the SVG working group cannot promise that version 1 of SVG will support the inheritance of CSS styles from a parent grammar. The working group should work towards achieving this highly desirable capability, however. Representatives from the CSS and XSL have requested working with the SVG working on style sheet issues. They would like to discuss:
 - i. use either XSL or CSS stylesheets within SVG
 - ii. use stylesheets to apply properties to both the text AND graphic objects/assemblies within an embedded SVG component (to provide/enforce "house style" or common properties across all illustrations in a document).

25. Connectable reference points

Connectable reference points are useful constructs for many applications. In particular, diagramming applications could use connectable reference points to define the start and end points for lines that connect two different objects. The difficulty with connectable reference points is that they might introduce a large degree of complexity into the specification and the various implementations. It might be better to leave connectable reference points to a higher-level XML grammar which sits on top of SVG.

- a. It is not a requirement for version 1 of SVG to provide for a mechanism for defining connectable reference points. This is a complex issue which might be better served by a higher-level XML grammar.
- b. However, the working group should not exclude this as an option for version 1 of SVG. If one of the working group members can come up with a good proposal, then the working group should consider the proposal seriously.
- c. See [DOM access](#), below, for more on connectable reference points.

26. Parameter substitution and formulas on coordinates

Parameterized graphic objects, possibly built using a set of formulas to define how the object grows and stretches based on the values of its parameters and other aspects of the graphics, would provide for a powerful "intelligent graphics" capability. However, such a feature opens up many complex issues which might be better off in a higher-level XML grammar which sits on top of SVG.

- a. Parameter substitution and formulas on coordinates are not requirements for version 1 of SVG.
- b. A simple parameter substitution strategy, however, might be easy to define, simple to implement, and provide lots of value. In other words, a simple parameter substitution strategy might provide 90% of the value at 10% of the effort. If one of the working group members can come up with a good proposal, then the working group should consider the proposal seriously.
- c. It is harder to conceive of a simple formula-based coordinate capability. However, if someone of the working group members can come up with a good proposal, then the working group should consider the proposal seriously.

27. Ability to control whether a given object is printed.

- a. When the working group considers this feature, it should see whether CSS's print control features are adequate.

Interaction

28. Zoom and pan

- a. An SVG processor should support zoom and pan on graphics, with true scalability. Thus, all objects and attributes (including such things as text and linewidths) should grow/shrink uniformly with the zoom level.

29. Links

- a. It should be possible to assign a link to any graphic object or group.
- b. SVG should support all of the kinds of links into and out of a drawing as would be appropriate. For example, it should provide for links to other views in the same file or links to external media (i.e., a URL). Also, it should be possible to link into a particular view within an SVG drawing.
- c. As much as is appropriate, SVG should be compatible with XLink.
- d. The working group discussed briefly the concept of linking into a moment of time within an animation application. This should be investigated by the working group as the specification is developed.

30. Event handling

- a. It should be possible to assign event handlers to an individual graphic object or group.
- b. The list of event handlers should at least be as extensive as what is available for HTML (e.g., mouseovers, mouseclicks).
- c. Additional event handlers might prove to be valuable. For example, an onzoom event handler might prove very useful to control what content appears based on zoom level. The working group should investigate onzoom and other possible event handlers.

31. Object selection, copying/pasting to clipboard

- a. It is highly desirable but not required that SVG viewing processors have the ability to selectively copy/paste graphical elements, particularly from the browser to the desktop environment.

- b. The working group has not investigated yet whether it makes sense to specify an object selection mechanism in SVG. However, it is clear that the ability to select part of a drawing is a requirement for clipboard/exchange purposes.
- c. A particular detail is selecting text for the purposes of copying to the clipboard. The working group hasn't discussed yet whether it should be possible to select text strings (complete or partial) from within an SVG processor.

32. DOM access

- a. SVG should provide for complete DOM support for all attributes and elements.
- b. To provide robust hooks for animation applications, the DOM should expose graphic objects down to the individual point.
- c. Supplemental utility methods (e.g., query an object's bounds, its center, its perimeter as expressed as a path, access to "connectable" points for callouts or connection lines) would be very helpful to people writing scripts that drive SVG drawings or to higher level grammars (e.g., Web Schematics or MathML) which might want to perform layout based on the bounds of a given graphics object. The SVG working group should collaborate with the DOM working group to investigate whether it is possible within the constraints of DOM to provide such utility methods.

33. Animation

Built-in animation primitives will not be part of SVG. Animation will only be possible via the DOM or a different specification, which might sit on top of SVG.

Miscellaneous

34. Inclusion of private data (metadata)

To promote the use of SVG as an interchange format or as a component of higher-level graphics languages, there needs to be a provision for inclusion of application-specific or industry-specific private data within an SVG drawing.

35. Extensibility

It is highly desirable that SVG be extensible to cope with changing requirements and for providing many valuable hooks to allow for creation of more efficient and compelling Web pages. A well-designed extensibility mechanism can allow for tomorrow's innovations to be available in today's browsers (i.e., no need to wait for a new version of the standard to be defined and implemented in browsers). A well-designed extensibility mechanism could be the best solution for many valuable features such as client-side filters on graphics data. Possible extensibility facilities are custom paint servers (for filling and stroking), custom object types, custom filters, custom compositing engines and custom color spaces. However, defining a useful and workable extensibility mechanism is very difficult and fraught with many obstacles, such as deficiencies in cross-platform language standards. Thus, the working group should look into an extensibility mechanism as a highly desirable feature and review proposals from the members, but an extensibility mechanism is not an absolute requirement for version 1 of SVG.

36. Embedded components to achieve self-contained graphics files

- a. Images - There has been some strong initial feedback has been that SVG should provide for embedded images. The working group should collaborate with other working groups (e.g., XML) to investigate the feasibility of allowing for embedded images within an SVG drawing.
- b. Fonts - There has been some strong initial feedback has been that SVG should provide for embedded fonts. The working group has yet to make decisions on this issue. The working group should collaborate with other working groups (e.g., XML) to investigate the feasibility of allowing for embedded fonts within an SVG drawing.

Chris Lilley <chris@w3.org>

Chair, W3C SVG Working Group

Last modified: \$Date: 1998/11/03 23:41:19 \$

