W3C Working Draft

A The SVG Micro DOM (uDOM)

Contents

During the later stages of the SVG Mobile 1.1 specification it became obvious that there was a requirement to subset the SVG and XML DOM in order to reduce the burden on implementations. SVGT 1.2 adds new features to the uDOM, allowing for as much necessary functionality as possible, still being suitable for SVG Tiny implementations.

Furthermore, it should be possible to implement the uDOM on devices that support SVG Tiny 1.1 although, in this case, the scripting would be external to the SVG document (since SVG Tiny 1.1 does not support inline scripting).

The goal of the uDOM definition is to provide an API that allows access to initial and computed attribute and property values, to reduce the number of interfaces, to reduce run-time memory footprint using necessary features of the core XML DOM, as well as the most useful SVG features (such as transformation matrices).

The IDL definition for the uDOM is provided.

A.1 Introduction

This appendix consists of the following parts:

- Overview of the SVG uDOM An introduction to the uDOM, including a summary of supported features and descriptions by topic of the key features and constraints.
- A definition of all the interfaces in the SVG uDOM. (DOM Core APIs, DOM Events APIs, SVG DOM APIs, SMIL DOM APIs and Global DOM APIs.)

A.2 Overview of the SVG uDOM

The following sections describe the SVG uDOM key features and constraints.

Note. Like other W3C DOM definitions, the SVG uDOM is programming-language independent. Although this appendix only contain ECMAScript and Java[tm]

language examples, the SVG uDOM is compatible with other programming languages.

A.2.1 Document Access

The SVG uDOM offers access to Document object which is the root for accessing other features. The way the Document object becomes available depends on the usage context. One way to gain access to the Document object is to implement the EventListenerInitializer2 interface. The SVG Tiny user agent will invoke the implementation's `initializeEventListeners (dom::Document doc)` method once the programming logic has been loaded and is ready to bind to the document. The Document object is sometimes accessible through other means, for example as the global 'document' variable in ECMAScript.

A.2.2 Tree Navigation

SVG uDOM only allows navigation of the element nodes in the DOM tree. Two options are available for navigating the hierarchy of elements:

- Individual element nodes with an ID value can be accessed directly via the `getElementById` method on the Document interface.
- The hierarchy of element nodes can be traversed using the facilities on the ElementTraversal interface, along with the `parentNode` attribute on the Node interface.

The ElementTraversal interface provides `firstElementChild`, `lastElementChild`, `previousElementSibling` and `nextElementSibling`, which are particularly suitable for constrained devices. These traversal mechanisms skip over intervening nodes between element nodes, such as text nodes which might only contain spaces, tabs and newlines.

A.2.3 Element Creation

SVG uDOM allows the creation of new `Elements`:

Example: Element creation

```
Element myRect = document.createElementNS(svgNS, "rect");
```

A.2.4 Element Addition

Node Addition is the ability to add new elements to a document tree.

SVG uDOM allows addition and insertion of a `Node` :

Example: Element addition

```
String svgNS = "http://www.w3.org/2000/svg";
// Create a new <rect> element
Element myRect = document.createElementNS(svgNS, "rect");
// Set the various <rect> properties before appending
...

// Add element to the root of the document
Element svgRoot = document.getDocumentElement();
svgRoot.appendChild(myRect);

// Create a new <ellipse> element
Element myEllipse = document.createElementNS(svgNS, "ellipse");

// Set the various <ellipse> properties before insertion
...

// Insert the ellipse before the rectangle
svgRoot.insertBefore(myEllipse, myRect);
```

A.2.5 Element Removal

Node removal is the ability to remove an element from a document tree. SVG uDOM allows the removal of element `Nodes`.

Example: Element removal

```
Element myRect = ...; // See Element creation
Element myGroup =document.getElementById("myGroup");
myGroup.appendChild(myRect);
....
myGroup.removeChild(myRect);
```

A.2.6 Attribute and Property Access

SVG 1.2 adds a new ability to access XML attribute andCSS property values through the SVG uDOM through the concept of *Traits*. A trait is a potentially animatable parameter associated with an element. Trait is the typed value (e.g., a number, not just a string) that gets assigned through an XML attribute or a CSSproperty. Traits can be thought of as a unification and generalization of some of the notions of XML attributes and CSS properties.

The trait facilities in the SVG uDOM allow for strongly-typed access to certain attribute and property values. For example, there is a `getFloatTrait(...)` method for getting an attribute or property value directly as a float. This contrasts the DOM Core `getAttributeNS(...)` method which always returns a string.

The trait facilities in the SVG uDOM are available on the TraitAccess interface.

Example: Trait Access

```
float width = myRect.getFloatTrait('width');
width += 10;
myRect.setFloatTrait('width', width);
```

SVG uDOM provides restricted access to attributes via `getAttributeNS` and `setAttributeNS` on the Element interface. The restrictions are described in `getAttributeNS` .

A.2.7 Text Node Access

In the SVG uDOM, text node access is available via trait getters and setters and via the textContent attribute on the Node interface. To access or set the text string value for an element via traits you invoke `getTrait()` or `setTrait()` on that element and pass `#text` as the name of the trait you want to get or set. For example, `MyTextElement.setTrait("#text", "Hello");`

Text access via the #text mechanism is supported on Text Content, 'desc', 'title' and 'metadata' elements. Text access to other elements defined within this specification (see list of elements) is not supported and an implementation is free to ignore any text on these elements.

The result of getting and setting text content via the #text mechanism is exactly the same as when using the textContent attribute. Therefore the user should be aware of the fact that styling by 'tspan' elements will be lost if she gets a text string from the model and sets it back again.

A.2.8 Event Listener Registration and Removal

Event Listener Registration and Removal is the ability to add and remove new event listeners from a `Document`. SVG uDOM allows adding and removing `EventListeners:`

Example: Event Listeners

```
class MyEventListener implements EventListener {
public void handleEvent(Event evt) {
// Do whatever is needed here
}
}
...
EventListener l = new MyEventListener();

SVGElement myRect = (SVGElement)document.getElementById("myRect");
//Listen to click events, during the bubbling phase
myRect.addEventListener("click", l, false);
....

// Remove the click listener
myRect.removeEventListener("click", l, false);
```

The SVG uDOM only supports the bubble phase. Any attempt to specify event operations on the capture phase will raise a DOMException of type NOT_SUPPORTED_ERR.

Refer to the DOM Events Level 3 specification or the XML Events specification introduction for an explanation of the SVG event flow and the meaning of event targets, event current target, bubble and capture.

A.2.9 Animation

SVG uDOM allows code to start or end animation elements.

Example: animation

```
AnimationElement animateColor = (AnimationElement) document.getElementById("myAnimation");

// Start the animation 2.5 seconds from now.
animateColor.beginElementAt(2.5);
```

A.2.10 Package naming

The SVG uDOM will use the same package names as the SVG 1.2 Full DOM (e.g., org.w3c.dom,org.w3c.dom.events, org.w3c.dom.svg). This allows applications which restrict themselves to the features in the SVG uDOM to also run in implementations that support the SVG 1.2 Full DOM.

A.3 Module: dom

A.3.1 DOMException

IDL Definition

```
exception DOMException
{
    unsigned short code;
};

// ExceptionCode
const unsigned short WRONG_DOCUMENT_ERR = 4;
const unsigned short INDEX_SIZE_ERR = 1;
const unsigned short HIERARCHY_REQUEST_ERR = 3;
const unsigned short NO_MODIFICATION_ALLOWED_ERR = 7;
const unsigned short NOT_FOUND_ERR = 8;
const unsigned short NOT_SUPPORTED_ERR = 9;
const unsigned short INVALID_STATE_ERR = 11;
const unsigned short INVALID_MODIFICATION_ERR = 13;
const unsigned short INVALID_ACCESS_ERR = 15;
const unsigned short TYPE_MISMATCH_ERR = 17;
```

Constants

WRONG_DOCUMENT_ERR
If a node is used in a different document than the one that created it (that doesn't support it).

INDEX_SIZE_ERR
If index or size is negative, or greater than the allowed value.

HIERARCHY_REQUEST_ERR
If any Node is inserted somewhere it doesn't belong.

NO_MODIFICATION_ALLOWED_ERR
If an attempt is made to modify an object where modifications are not allowed.

NOT_FOUND_ERR
If an attempt is made to reference a Node in a context where it does not exist. See Node#insertBefore for example.

NOT_SUPPORTED_ERR
If the implementation does not support the requested type of object or operation.

INVALID_STATE_ERR
If an attempt is made to use an object that is not, or is no longer, usable.

INVALID_MODIFICATION_ERR
If an attempt is made to modify the type of the underlying object.

INVALID_ACCESS_ERR
If a parameter or an operation is not supported by the underlying object.

TYPE_MISMATCH_ERR
If the type of an object is incompatible with the expected type of the parameter associated to the object.

No defined attributes

No defined methods

A.3.2 Node

The Node interface describes generic nodes in an SVG document tree.

This interface is a subset of the Node interface defined in the DOM Level 3 Core. The only node types that are required to be supported in the uDOM are Element nodes and Document nodes.

IDL Definition

```
interface Node
{
    readonly attribute DOMString namespaceURI;
    readonly attribute DOMString localName;
    readonly attribute Node parentNode;
    readonly attribute Document ownerDocument;
    attribute DOMString textContent;
    Node appendChild(in Node newChild) raises(DOMException);
    Node insertBefore(in Node newChild, in Node refChild) raises(DOMException);
    Node removeChild(in Node oldChild) raises(DOMException);
};
```

No defined constants

Attributes

namespaceURI
Returns the namespace URI of the Node.

localName
Returns the local part of the qualified name of this node. If the node is of type SVGElement, this returns the tag name without a prefix. But, if the node is of type Document then `null` is returned.

parentNode
Returns the parent `Node` of this `Node`.

ownerDocument
The Document object associated with this node.

textContent

This attribute returns the text content of this node and its descendants. When it is defined to be null, setting it has no effect. On setting, any possible children this node may have are removed and, if it the new string is not empty or null, replaced by a single Text node containing the string this attribute is set to. On getting, no serialization is performed, the returned string does not contain any markup. No whitespace normalization is performed and the returned string does not contain the white spaces in element content. Similarly, on setting, no parsing is performed either, the input string is taken as pure textual content.

`textContent` is supported on all non-svg elements. For elements defined within this specification (see list of elements) `textContent` is supported on the Text Content, 'desc', 'title' and 'metadata' elements. `textContent` on a non-supported element is null. Setting `textContent` on a non-supported element has no effect.

`textContent` on the `document` node is always null.

An alternate way of accessing text content on elements defined within the svg specification is via the getTrait("#text") syntax.

For further details see DOM3 textNode.

Methods

appendChild
Appends a child to this `Node`.

Parameters:

- newChild The `Node` to be appended to this `Node`. This is equivalent to insertBefore(newChild,null)

Raises:

- DOMException with error code HIERARCHY_REQUEST_ERR: Raised if this node is of a type that does not allow children of the type of the newChild node, or if the node to append is one of this node's ancestors or this node itself, or if this node is of type Document and the DOM application attempts to append a second Element node.
- DOMException with error code WRONG_DOCUMENT_ERR: Raised if newChild was created from a different document than the one that created this node.
- DOMException with error code NOT_SUPPORTED_ERR: if the newChild node is a child of the Document node or if the child is of a type that cannot be created with createElementNS.
- DOMException with error code INVALID_STATE_ERR: if the newChild node would cause the document to go into error.

insertBefore

Inserts newChild before refChild in the child list for this node. If refChild is null, newChild is inserted at the end of the list. If the newChild is already part of the tree, it is first removed.

Parameters:

- newChild The child to add.
- refChild The child before which the new child should be added.

Raises:

- DOMException with error code HIERARCHY_REQUEST_ERR: if this node is of a type that does not allow children of the type of the newChild node, or if the node to append is one of this node's ancestors or this node itself, or if this node is of type Document and the DOM application attempts to append a second Element node.
- DOMException with error code WRONG_DOCUMENT_ERR: Raised if newChild was created from a different document than the one that created this node.
- DOMException with error code NOT_FOUND_ERR: raised if refChild is not a child of this node.
- DOMException with error code NOT_SUPPORTED_ERR: if the newChild node is a child of the Document node or if the child is of a type that cannot be created with createElementNS.
- DOMException with error code INVALID_STATE_ERR: if the newChild node would cause the document to go into error.

removeChild

Removes the specified child associated with this Node.

Parameters:

- oldChild The `Node` that is to be removed.

The SVG WG has considered to not allow removal of elements that are referenced by other elements. The WG has resolved to not add this constraint but are willing to reconcider if implementation feedback shows that such a constraint would decrease implementation cost. With such a constraint, calling removeChild on a referenced element would not remove the element and the method would raise a DOMException (INVALID_ACCESS_ERR).

Raises:

- DOMException with error code NOT_FOUND_ERR: Raised if oldChild is not a child of this node.
- DOMException with error code NOT_SUPPORTED_ERR: if this node is of type Document or if the child, or any of its descendants, is of a type that cannot be created with Document.createElementNS.

A.3.3 Element

Represents an element in the document. A user agent is allowed to store normalized attribute values internally as described in DOM 3.

IDL Definition

```
interface Element : Node
{
    DOMString getAttributeNS(in DOMString namespaceURI, in DOMString localName) raises(DOMException);
    void setAttributeNS(in DOMString namespaceURI, in DOMString localName, in DOMString value);
};
```

No defined constants

No defined attributes

Methods

getAttributeNS

Retrieves an attribute value by local name and namespace URI. Per [XML Namespaces], applications must use the value null as the namespaceURI parameter for methods if they wish to have no namespace.

A uDOM implementation must support getAttributeNS for all attributes on elements that does not belong to SVGT1.2 [List of SVGT1.2 Elements]. For attributes belonging to SVGT1.2 [List of SVGT1.2 Attributes] the implementation must support attributes accessible by the getTrait method on SVGElement (see table for list of attributes supporting getTrait).

An important difference between getTraitNS and getAttributeNS is that getTraitNS returns the computed attribute value but getAttributeNS returns the specified attribute value (witch might not exactly match the original specified value due to the possibility of user agent value normalization as described below).

Example

```
<g fill="red">
<rect id="r1" x="1" y="1" width="5" height="5"/>
<rect id="r2" fill="inherit" x="1" y="1" width="5" height="5"/>
</g>
```

`r1.getTraitNS(svgNS, "fill")` return "red". (or equivalent normalized form, see below)

`r2.getTraitNS(svgNS, "fill")` return "red". (or equivalent normalized form, see below)

`r1.getAttributeNS(svgNS, "fill")` return "".

`r2.getAttributeNS(svgNS, "fill")` return "inherit".

A viewer implementing the UDOM is allowed to return normalized values in getAttributeNS as defined in DOM 3.

E.g. fill="red" may be returned as "rgb(255,0,0)", "#ff0000", or other semantically identical form. stroke-width="3" might be returned as e.g. "3.00".

Parameters:

- namespaceURI The namespace URI of the attribute to retrieve.
- localName The local name of the attribute to retrieve.

Raises:

- DOMException NOT_SUPPORTED_ERR if getAttributeNS tries to get an unsupported attribute.

setAttributeNS

Adds a new attribute. The value to set is a simple string; it is not parsed as it is being set. So any markup is treated as literal text, and needs to be appropriately escaped by the implementation when it is written out. Per [XML Namespaces], applications must use the value null as the namespaceURI parameter for methods if they wish to have no namespace.

A uDOM implementation must support setAttributeNS for all attributes.

Parameters:

- namespaceURI The namespace URI of the attribute to create or alter.
- localName The local name of the attribute to create or alter.
- value The value to set in string form.

A.3.4 Document

The Document interface represents an XML Document.

This interface is a subset of the Document interface defined in the DOM Level 3 Core.

Note the behavior of the following attributes and methods from the `Node` interface when called on a `Document` object:

- the parentNode attribute will be `null`
- appendChild throws `HIERARCHY_REQUEST_ERR`
- insertBefore throws `HIERARCHY_REQUEST_ERR`
- removeChild throws `NOT_SUPPORTED_ERR`

IDL Definition

```
interface Document : Node
{
    Element createElementNS(in DOMString namespaceURI, in DOMString qualifiedName) raises(DOMException);
    readonly attribute Element documentElement;
    Element getElementById(in DOMString id);
};
```

No defined constants

Attributes

documentElement
Return a child element of this document Node which corresponds to the top-most tag in XML file. For SVG files it must be `SVGSVGElement`, but return type is Element for DOM Core compatibility and to allow for future extensions.

Methods

createElementNS
Create a new `Element` based on the specified (qualified) tag name.

Parameters:

- namespaceURI The namespace uri for the newly created element.
- qualifiedName The qualified name for the newly created element (For example: "rect", to create a <rect> element).

Raises:

- DOMException NOT_SUPPORTED_ERR if the type of element is not supported by the implementation.

getElementById
Return the `Element` in the current document with the given unique ID. If no such element exists, this returns null. If more than one element has an ID attribute with that value, this method returns the first element, in document order, which has the requested ID.

Parameters:

- id The ID of the object to be retrieved.

A.4 Module: events

A.4.1 EventTarget

This interface represents an event target, and is a subset of the EventTarget interface defined in the DOM Level 2 Event model.

This interface is implemented by an object (SVGElement, SVGDocument, SVGElementInstance) that can notify listeners about events and allows registration and removal of EventListener objects.

IDL Definition

```
interface EventTarget
{
    void addEventListenerNS(in DOMString namespaceURI, in DOMString type, in EventListener listener, in boolean useCapture);
    void removeEventListenerNS(in DOMString namespaceURI, in DOMString type, in EventListener listener, in boolean useCapture);
};
```

No defined constants

No defined attributes

Methods

addEventListenerNS

This method registers the specified listener with the event target. If an EventListener is added to an EventTarget while it is processing an event, it will not be triggered by the current actions. If multiple identical EventListeners are registered on the same EventTarget with the same parameters the duplicate instances are discarded. They do not cause the EventListener to be called twice and since they are discarded they do not need to be removed with the removeEventListenerNS method.

Parameters:

- namespaceURI The namespace URI of the event.
- type The type of event to listen to.
- listener Will be notified when an event of the desired type happens on this target or one of its descendant.
- useCapture This can only be 'false' since capture phase is not supported.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if useCapture is true since capture phase is not supported in SVG Tiny.

removeEventListenerNS

This method removes the specified listener from the event target. If an EventListener is removed from an EventTarget while it is processing an event, it will not be triggered by the current actions. Calling removeEventListenerNS with arguments which do not identify any currently registered EventListener on the EventTarget has no effect.

Parameters:

- namespaceURI The namespace URI of the event.
- type The type of event that was listened to.
- listener The listener that was previously registered.
- useCapture This can only be 'false' since capture phase is not supported.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if useCapture is true since capture phase is not supported in SVG Tiny.

A.4.2 EventListener

This interface represents an event listener, and is a subset of the EventListener interface defined in the DOM Level 2 Event model.

This interface must be implemented and registered on an EventTarget using the EventTarget#addEventListenerNS method to be notified about events that occur on or bubble through the event target.

IDL Definition

```
interface EventListener
{
    void handleEvent(in Event evt);
};
```

No defined constants

No defined attributes

Methods

handleEvent

This method is called whenever an event occurs of the type for which the EventListener interface was registered.. The Event object contains the necessary information pertaining to the event, such as its target and type.

Parameters:

- evt The event object containing necessary event information.

A.4.3 Event

The Event interface is used to provide contextual information about an event to the handler processing the event. An object which implements the Event interface is passed as the first parameter to the EventListener#handleEvent call. If an event target is an element instance (see SVGElementInstance), the currentTarget is an implementation of EventTarget that does not implement the Node interface. In profiles of SVG that support the SVGElementInstance interface, the currentTarget is an SVGElementInstance.

IDL Definition

```
interface Event
{
    readonly attribute EventTarget target;
    readonly attribute EventTarget currentTarget;
    readonly attribute DOMString type;
};
```

No defined constants

Attributes

target
Used to indicate the event target. This attribute contains the target node when used with the DOM event flow.

currentTarget
Used to indicate the EventTarget whose EventListeners are currently being processed. In SVG Tiny, this is always an object to which event listener was attached.

type
This method returns the event type information. The name of the event is case-sensitive. For a list of supported event types see the Supported Events chapter.

No defined methods

A.4.4 OriginalEvent

OriginalEvent is supported on events going through the following element types: 'animation' and 'foreignObject' (when 'foreignObject' has an 'xlink:href' attribute).

IDL Definition

```
interface OriginalEvent
{
    readonly attribute Event originalEvent;
};
```

No defined constants

Attributes

originalEvent
For the supported element ('animation' and 'foreignObject'), events can be dispatched to the SVGElementInstance objects corresponding to the content referenced by the supported element. UI events that are dispatched to these SVGElementInstance objects can bubble up to the supported element. Upon bubbling to the supported element, the User Agent must manufacture new Event and OriginalEvent objects. The new Event object must be of the same type and carry the same information as the original event object but which has the supported element as the "target". The new OriginalEvent object must set the "originalEvent" attribute to the original event object.

No defined methods

A.4.5 MouseEvent

Event that provides specific contextual information associated with pointing device events.

IDL Definition

```
interface MouseEvent : Event
{
    readonly attribute long screenX;
    readonly attribute long screenY;
    readonly attribute long clientX;
    readonly attribute long clientY;
    readonly attribute unsigned short button;
};
```

No defined constants

Attributes

screenX
The horizontal coordinate at which the event occurred relative to the origin of the screen coordinate system.

screenY
The vertical coordinate at which the event occurred relative to the origin of the screen coordinate system.

clientX
The horizontal coordinate at which the event occurred relative to the DOM implementation's client area.

clientY
The vertical coordinate at which the event occurred relative to the DOM implementation's client area.

button
During mouse events caused by the depression or release of a mouse button, this is used to indicate which mouse button changed state. The values for button range from zero to indicate the left button of the mouse, one to indicate the middle button if present, and two to indicate the right button. For mice configured for left handed use in which the button actions are reversed the values are instead read from right to left.

No defined methods

A.4.6 TextEvent

One or more characters have been entered.

Event type that is a Text Event: textInput.

IDL Definition

```
interface TextEvent : Event
{
    readonly attribute DOMString data;
};
```

No defined constants

Attributes

data
Holds the value of the characters generated by the character device. This may be a single Unicode character or a non-empty sequence of Unicode characters. This attribute cannot be null or contain the empty string.

No defined methods

A.4.7 KeyboardEvent

Provides specific contextual information associated with keyboard devices. Each keyboard event references a key using an identifier.

Event types that are Keyboard Events: keyDown, keyUp.

IDL Definition

```
interface KeyboardEvent : Event
{
    readonly attribute DOMString keyIdentifier;
};
```

No defined constants

Attributes

keyIdentifier
Holds the identifier of the key.

No defined methods

A.4.8 ConnectionEvent

Provides information about the data that arrives through a connection.

Event type that is a Connection Event: connectionData.

IDL Definition

```
interface ConnectionEvent : Event
{
    readonly attribute DOMString data;
};
```

No defined constants

Attributes

data
The data that has arrived.

No defined methods

A.4.9 TimeEvent

Event that is fired by a SMIL animation element.

Event types that are TimeEvents: beginEvent, endEvent, repeatEvent.

IDL Definition

```
interface TimeEvent : Event
{
    readonly attribute long detail;
};
```

No defined constants

Attributes

detail
Specifies some detail information about the Timing Event, the information depends on the type of event. For beginEvent and endEvent the detail field is not used. For repeatEvent the detail field contains the current repeat iteration.

No defined methods

A.4.10 WheelEvent

Many devices today have a rotational input method, such as the wheel on a mouse or the "jog dial" of a phone or PDA.

The "wheel" event is triggered when the user rotates the rotational input device. This event may only be registered on the root-most svg element.

Event type that is a WheelEvent: wheel.

IDL Definition

```
interface WheelEvent : Event
{
    readonly attribute int wheelDelta;
};
```

No defined constants

Attributes

wheelDelta
Indicates the number of "clicks" the wheel has been rotated. A positive value indicates that the wheel has been rotated away from the user (or in a right-hand manner on horizontally aligned devices) and a negative value indicates that the wheel has been rotated towards the user (or in a left-hand manner on horizontally aligned devices).

A "click" is defined to be a unit of rotation. On some devices this is a finite physical step. On devices with smooth rotation, a "click" becomes the smallest measurable amount of rotation.

No defined methods

A.4.11 ProgressEvent

Many resources, such as raster images, movies and complex SVG content can take a substantial amount of time to download. In some use cases the author would prefer to delay the display of content or the beginning of an animation until the entire contents of a file have been downloaded. In other cases, the author may wish to give the viewer some feedback that a download is in progress (e.g. a loading progress screen).

The ProgressEvent occurs when the user agent makes progress loading a resource (local or external) referenced by an xlink:href attribute.

The user agent must dispatch a ProgressEvent at the beginning of a load operation (i.e., just before starting to access the resource). This event is of type 'preload'. The value of the 'preload' event's progress property is 0.

The user agent must dispatch a ProgressEvent at the end of a load operation (i.e. after load is complete and the user agent is ready to render the corresponding resource). This event is of type 'postload' event. The value of the 'postload' event's progress property is 1.

The user agent may dispatch a loadProgress event between the 'preload' event and the 'postload' events. Such events are of type 'loadprogress'.

All 'loadprogress' events must follow to the following constraints:

- the progress property on an 'loadprogress' event is strictly greater or equal to zero and strictly smaller than or equal to one.
- for two consecutive 'loadprogress' events, if provided, the progress property of an event must be strictly bigger than the value of the progress property on the preceding event.
- for two consecutive 'loadprogress' events, the loaded property of an event must be strictly bigger than the value of the loaded property on the preceding event.

In the case where the size of the downloading resource is known, such as from HTTP headers, then the progress property reflects the proportion of the current download that has been completed.

In the case where the size of the downloading resource is not known, then the progress property will only ever have the value 0 or 1.

The ProgressEvent has three corresponding event attributes on elements: onpreload, onpostload and onloadprogress.

Example: ProgressEvent

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
xmlns:xlink="http://www.w3.org/1999/xlink">

<script type="application/ecmascript"><![CDATA[
function showImage(imageHref) {
var image = document.getElementById('myImage');
image.setTraitNS("http://www.w3.org/1999/xlink", "href", imageHref);
}

function imageLoadStart(evt) {
var progressBar = document.getElementById('progressBar');
progressBar.setFloatTrait("width", 0);
var loadAnimation = document.getElementById('loadAnimation');
loadAnimation.beginElement();
}

function imageLoadProgress(evt) {
  var progressBar = document.getElementById('progressBar');
progressBar.setFloatTrait("width", 100*evt.progress);
}

function imageLoadComplete(evt) {
var progressBar = document.getElementById('progressBar');
  progressBar.setFloatTrait("width", 100);
var loadAnimation = document.getElementById('loadAnimation');
loadAnimation.endElement();
}
]]></script>

<image id="myImage" xlink:href="imageA.png" width="300" height="400">
<handler type="application/ecmascript" ev:event="preload">
```

```
imageLoadStart(evt);
</handler>

<handler type="application/ecmascript" ev:event="loadprogress">
imageLoadProgress(evt);
</handler>

<handler type="application/ecmascript" ev:event="postload">
imageLoadComplete(evt);
</handler>
</image>

<rect width="120" height="30" y="400">
<handler type="application/ecmascript" ev:event="click">
showImage('imageB.png');
</handler>
</rect>

<animate id="loadAnimation" ... />

<rect id="progressBar" ... />
</svg>
```

Event types that are ProgressEvents: loadprogress, preload, postload.

IDL Definition

```
interface ProgressEvent : Event
{
    readonly attribute boolean lengthComputable;
    readonly attribute DOMString typeArg;
    readonly attribute unsigned long loaded;
    readonly attribute unsigned long total;
};
```

No defined constants

Attributes

lengthComputable
If false the total number of bytes (total) cannot be computed and the value of total should be ignored. This might occur if the size of the downloaded resource is unknown or if the data has already arrived.

typeArg
Specifies the event type. One of 'preload', 'loadprogress' or 'postload'.

loaded
Specifies the number of bytes downloaded since the beginning of the download. This value is ignored for a 'preload' or 'postload' event.

total
Specifies the expected total number of bytes expected in a load operation. For a 'loadprogress' event, it should specify the total number of bytes expected.

No defined methods

A.5 Module: smil

A.5.1 ElementTimeControl

This interface defines common methods for elements which define animation behaviors compatible with SMIL Animation (animation elements, media elements and time containers).

IDL Definition

```
interface ElementTimeControl
{
    void beginElementAt(in float offset);
    void beginElement();
    void endElementAt(in float offset);
    void endElement();
    void pauseElement();
    void resumeElement();
    readonly attribute boolean isPaused;
};
```

No defined constants

Attributes

isPaused
true if the animation is paused. false otherwise. See Paused element and the active duration. Note that an element that is stopped (has reached the end of its active duration) is not paused.

Methods

beginElementAt
Creates a begin instance time for the current time plus or minus the passed offset. The new instance time is added to the begin instance times list.

Parameters:

- offset The offset in seconds at which to begin the element.

beginElement

Creates a begin instance time for the current time. The new instance time is added to the begin instance times list. This is equivalent to beginElementAt(0).

endElementAt
Creates an end instance time for the current time plus or minus the passed offset. The new instance time is added to the end instance times list.

Parameters:

- offset The offset in seconds at which to end the element.

endElement
Creates an end instance time for the current time. The new instance time is added to the end instance times list. This is equivalent to endElementAt(0).

pauseElement
Pauses the timed element. See Paused element and the active duration.

resumeElement
Resumes the timed element. See Paused element and the active duration.

A.6 Module: global

A.6.1 Global

The Global interface is designed to be the parent interface for language specific window objects. It is currently defined to be empty.

IDL Definition

```
interface Global
{
};
```

No defined constants

No defined attributes

No defined methods

A.6.2 Connection

The Connection interface provides an API for socket-level communication. A connection object is created using the SVGGlobal interface.

IDL Definition

```
interface Connection : events::EventTarget
{
    typedef dom::DOMString DOMString;
    typedef dom::DOMException DOMException;
    void connect(in DOMString uri) raises(DOMException);
    void send(in DOMString data);
    void close();
    readonly attribute boolean connected;
};
```

No defined constants

Attributes

connected
True if a connection is established, false otherwise.

Methods

connect
Connect with the provided URL.

Raises:

- DOMException NOT_FOUND_ERR if the connection fails.

send
Send the data over the connection.

close
Close the connection.

A.7 Module: svg

A.7.1 SVGException

An exception thrown for SVG-specific errors, such as noninvertable matrix in SVGMatrix#inverse .

IDL Definition

```
exception SVGException
{
    unsigned short code;
};
```

```
// ExceptionCode
const unsigned short SVG_INVALID_VALUE_ERR = 1;
const unsigned short SVG_MATRIX_NOT_INVERTABLE = 2;
```

Constants

SVG_INVALID_VALUE_ERR
Value passed to an SVG-specific method is invalid, such as out of range color component in SVGSVGElement#createSVGRGBColor .

SVG_MATRIX_NOT_INVERTABLE
Matrix that has a determinant equal to zero, and therefore not invertable.

No defined attributes

No defined methods

A.7.2 SVGDocument

IDL Definition

```
interface SVGDocument : Document, events::EventTarget
{
    readonly attribute SVGGlobal global;
};
```

No defined constants

Attributes

global
Access to the SVGGlobal object.

No defined methods

A.7.3 SVGElementInstance

For each 'use' element, the UDOM represents the referenced content with a shadow tree of SVGElementInstance objects (the "instance tree") of type SVGElementInstance. An SVGElementInstance represents a single node in the instance tree.

If the 'use' element references a simple graphics element such as a 'rect', then there is only a single SVGElementInstance object, and the correspondingElement attribute on this SVGElementInstance object is the SVGElement that corresponds to the referenced 'rect' element.

If the 'use' element references a 'g' which contains two 'rect' elements, then the instance tree contains three SVGElementInstance objects, a root SVGElementInstance object whose correspondingElement is the SVGElement object for the 'g', and then two child SVGElementInstance objects, each of which has its correspondingElement that is an SVGElement object representing the 'rect'. Note however that the uDOM does not provide methods to navigate between a SVGElementInstance and its children SVGElementInstances.

IDL Definition

```
interface SVGElementInstance : events::EventTarget
{
    readonly attribute SVGElement correspondingElement;
    readonly attribute SVGElement correspondingUseElement;
};
```

No defined constants

Attributes

correspondingElement
The corresponding element to which this object is an instance. For example, if a 'use' element references a 'rect' element, then an SVGElementInstance is created, with its correspondingElement being the SVGElement object for the 'rect' element.

correspondingUseElement
The corresponding 'use' element to which this SVGElementInstance object belongs. When 'use' elements are nested (e.g., a 'use' references another 'use' which references a graphics element such as a 'rect'), then the correspondingUseElement is the outermost 'use' (i.e., the one which indirectly references the 'rect', not the one with the direct reference).

No defined methods

A.7.4 SVGSVGElement

This interface represents <svg> element in (SVG) document tree.

User Agent Transforms

The DOM attributes currentScale, currentRotate and currentTranslate are combined to form user agent transformation which is applied at the outermost level on the SVG document (i.e., outside the outermost 'svg' element) if "magnification" is enabled (i.e., zoomAndPan attribute is set to "magnify"). Their values can potentialy be modified through user-agent specific UI. User agent transformation can be obtained by multiplying matrix

```
[currentScale      0       currentTranslate.x]     [cos(currentRotate) -sin(currentRotate 0]
[     0      currentScale  currentTranslate.y]  by     [sin(currentRotate) cos(currentRotate) 0]
[     0            0              1          ]     [         0                  0         1]
```

i.e. (translate, then scale, then rotate the coordinate system). The reference point for scale and rotate operations is the origin (0, 0).

IDL Definition

```
interface SVGSVGElement : SVGLocatableElement : SVGAnimationElement
{
    const unsigned short FOCUS_AUTO     = 1;
    const unsigned short FOCUS_NEXT     = 2;
    const unsigned short FOCUS_PREV     = 3;
    const unsigned short FOCUS_NORTH        = 4;
    const unsigned short FOCUS_NORTH_EAST   = 5;
    const unsigned short FOCUS_EAST     = 6;
    const unsigned short FOCUS_SOUTH_EAST   = 7;
    const unsigned short FOCUS_SOUTH    = 8;
    const unsigned short FOCUS_SOUTH_WEST   = 9;
    const unsigned short FOCUS_WEST     = 10;
    const unsigned short FOCUS_NORTH_WEST   = 11;
    attribute float currentScale;
    attribute float currentRotate;
    readonly attribute SVGPoint currentTranslate;
    readonly attribute SVGRect viewport;
    attribute float currentTime;
    SVGMatrix createSVGMatrixComponents(in float a, in float b, in float c, in float d, in float e, in float f);
    SVGRect createSVGRect();
    SVGPath createSVGPath();
    SVGRGBColor createSVGRGBColor(in long red, in long green, in long blue) raises(SVGException);
    void moveFocus(in unsigned short motionType) raises(DOMException);
    void setFocus(in DOMObject object) raises(DOMException);
    DOMObject getCurrentFocusedObject();
};
```

Constants

FOCUS_AUTO
Indicates that focus should move to the next focusable object according to the User Agent own algorithm.

FOCUS_NEXT
Indicates that focus should move to the next focusable object according to current nav-index value.

FOCUS_PREV
Indicates that focus should move to the previous focusable object according to current nav-index value.

FOCUS_NORTH
Indicates a request that focus should move in the given direction.

FOCUS_NORTH_EAST
Indicates a request that focus should move in the given direction.

FOCUS_EAST
Indicates a request that focus should move in the given direction.

FOCUS_SOUTH_EAST
Indicates a request that focus should move in the given direction.

FOCUS_SOUTH
Indicates a request that focus should move in the given direction.

FOCUS_SOUTH_WEST
Indicates a request that focus should move in the given direction.

FOCUS_WEST
Indicates a request that focus should move in the given direction.

FOCUS_NORTH_WEST
Indicates a request that focus should move in the given direction.

Attributes

currentScale
On read : Returns current user agent scale (zoom) coefficient. The initial value for currentScale is 1. On write : Sets current user agent scale (zoom) coefficient.

Raises:

- DOMException with error code INVALID_ACCESS_ERR if the scale value is set to zero.

currentRotate
On read : Returns current user agent rotation angle in degrees. The initial value for currentRotate is 0. On write : Sets current user agent rotate coefficient in degrees.

currentTranslate
Current user agent translation used for scrolling or panning (The returned SVGPoint object is "live" and setting its x and y components will change user agent's translation). The initial values for currentTranslate is SVGPoint(0,0).

viewport

The position and size of the viewport (implicit or explicit) that corresponds to this 'svg' element. When the user agent is actually rendering the content, then the position and size values represent the actual values when rendering. If this SVG document is embedded as part of another document (e.g., via the HTML 'object' element), then the position and size are unitless values in the coordinate system of the parent document. (If the parent uses CSS or XSL layout, then unitless values represent pixel units for the current CSS or XSL viewport, as described in the CSS2 specification.) If the parent element does not have a coordinate system, then the user agent should provide reasonable default values for this attribute.

The object itself and its contents are both readonly. DOMException with error code NO_MODIFICATION_ALLOWED_ERR is raised if attempt is made to modify it. The returned SVGRect object is "live", i.e. its x, y, width, height is automatically updated if viewport size or position changes.

currentTime
On read : Returns current animation timeline time in seconds. On write : Sets current animation timeline time (in seconds). This API is required to support moving forwards in timeline. The underlying implementations are normally designed to seek forward in time and setting the time backwards is not meant to play the animation backwards. Note: Moving backwards in time is a costly feature for the implementations to support.

Methods

createSVGMatrixComponents
Creates new SVGMatrix object. This object can be used to modify value of traits which are compatible with SVGMatrix type using TraitAccess#setMatrixTrait method. The internal representation of the matrix is as follows:

```
[  a  c  e  ]
[  b  d  f  ]
[  0  0  1  ]
```

Parameters:

- a The 'a' component of the matrix to be set.
- b The 'b' component of the matrix to be set.
- c The 'c' component of the matrix to be set.
- d The 'd' component of the matrix to be set.
- e The 'e' component of the matrix to be set.
- f The 'f' component of the matrix to be set.

createSVGRect
Creates new SVGRect object. This object can be used to modify value of traits which are compatible with SVGRect type using TraitAccess#setRectTrait method. The intial values for x, y, width, height of this new SVGRect are zero.

createSVGPath
Creates new SVGPath object. This object can be used to modify value of traits which are compatible with SVGPath type using TraitAccess#setPathTrait method.

createSVGRGBColor
Creates new SVGRGBColor object. This object can be used to modify value of traits which are compatible with SVGRGBColor type using TraitAccess#setRGBColorTrait method.

Parameters:

- red The red component of SVGRGBColor object.
- green The green component of SVGRGBColor object.
- blue The blue component of SVGRGBColor object.

Raises:

- SVGException with error code SVG_INVALID_VALUE_ERR: if any of the parameters is not in the 0..255 range.

moveFocus
Moves current focus to a different object based on the value of motionType. User Agent must take into account the currently focused object in the document in order to find the new focused object.

If this method succeeds :

- A DOMFocusOut event MUST be created which has the previously focused object as the event target.
- After that, a DOMFocusIn event MUST be created which has the the new focused object as the event target.

A reference to the new focused object can be obtained using the EventTarget interface of the generated DOMFocusIn Event.

Refer to the focus section to see how navigation is managed. The behavior for this method MUST be the same as if an equivalent move was done by the end user (using joystick or TAB keys) and not by scripting.

Whenever the method fails (i.e. an Exception is raised), focus MUST stay on the currently focused object and no DOMFocusOut/DOMFocusIn event is sent.

NOTE: For stand-alone SVG documents, the User Agent MUST always have a currently focused object. At the beginning, the outermost SVGSVGElement object has focus.

Parameters:

- motionType The type of motion.

Raises:

- DOMException with error code NOT_SUPPORTED_ERROR if the requested motion type is not supported (i.e. not one of the interface constants).
- DOMException with error code INVALID_ACCESS_ERR if the currently focused object doesn't have a nav-* value corresponding to the requested motion type. For instance, if a moveFocus(FOCUS_UP) is called on an element which has no nav-up property.
- DOMException with error code INVALID_STATE_ERR if the currently focused object has a nav-* value corresponding to the requested motion type but the target indicated in this attribute can not be found or is not a focusable object. For instance, if a moveFocus(FOCUS_UP) is called on an object which has a nav-up property but the value of this property references an element which is not focusable.

setFocus
A request to put the focus on the given object.

If this method succeeds:

- A DOMFocusOut event MUST be created which has the previously focused object as the event target.
- After that, a DOMFocusIn event MUST be created which has the the new focused object as the event target.

A reference to the new focused object can be obtained using the EventTarget interface of the generated DOMFocusIn Event.

Whenever the method fails (i.e. an Exception is raised), focus MUST stay on the currently focused object and no DOMFocusOut/DOMFocusIn event is sent.

NOTE: For stand-alone SVG documents, the User Agent MUST always have a currently focused object. At the beginning, the SVGDocument has focus.

Parameters

- object The object which should receive focus.

Raises

- DOMException with error code NOT_SUPPORTED_ERROR if the requested element is not focusable (i.e. its 'focusable' property does not evaluate to 'true')

getCurrentFocusedObject
Returns a reference to the object which has the focus. In the case of standalone SVG documents, this method should never return 'null' because for standalone SVG files a User Agent MUST always have a currently focused object.

A.7.5 SVGRGBColor

This interface represents an "SVGRGBColor" datatype made up of red, green, and blue components. It can be used to read properties that store color values (TraitAccess#getRGBColorTrait) such as `fill`, `stroke`, and `color`.

IDL Definition

```
interface SVGRGBColor
{
    readonly attribute unsigned long red;
    readonly attribute unsigned long green;
    readonly attribute unsigned long blue;
};
```

No defined constants

Attributes

red
Returns the red component of the SVGRGBColor.

green
Returns the green component of the SVGRGBColor.

blue
Returns the blue component of the SVGRGBColor.

No defined methods

A.7.6 SVGRect

This interface represents an "SVGRect" datatype, consisting of a minimum X, minimum Y, width and height values.

IDL Definition

```
interface SVGRect
{
    attribute float x;
    attribute float y;
    attribute float width;
    attribute float height;
};
```

No defined constants

Attributes

x
On read : Returns the minimum X value for this SVGRect. On write : Sets the minimum X value of this SVGRect to the specified value.

y
On read : Returns the minimum Y value for this SVGRect. On write : Sets the minimum Y value of this SVGRect to the specified value.

width
On read : Returns the width for this SVGRect. On write : Sets the width of this SVGRect to the specified value.

height
On read : Returns the height for this SVGRect. On write : Sets the height of this SVGRect to the specified value.

No defined methods

A.7.7 SVGPoint

This interface represents an "SVGPoint" datatype, identifiend by its x and y components.

IDL Definition

```
interface SVGPoint
{
    attribute float x;
    attribute float y;
};
```

No defined constants

Attributes

x
On read : Returns the x component of the point. On write : Sets the x component of the point to the specified float value.

y
On read : Returns the y component of the point. On write : Sets the y component of the point to the specified float value.

No defined methods

A.7.8 SVGPath

This interface represents an "SVGPath" datatype used to define the path geometry. Corresponds to SVG path specification or the "d" attribute.

The native implementations must support the following simplifications or canonicalization of path segments. Any simplifications should be lossless.

- Relative commands (c, h, l, m, q, s, t, and v) must be converted to their absolute counterparts.
- Horizontal and Vertical lines (H, h, V, and v) must be converted to general lines (L and l).
- Translate command S to command C.
- Translate command T to command Q.

IDL Definition

```
interface SVGPath
{
    const unsigned short MOVE_TO = 77;
    const unsigned short LINE_TO = 76;
    const unsigned short CURVE_TO = 67;
    const unsigned short QUAD_TO = 81;
    const unsigned short CLOSE = 90;
    readonly attribute unsigned long numberOfSegments;
    unsigned short getSegment(in unsigned long cmdIndex) raises(DOMException);
    float getSegmentParam(in unsigned long cmdIndex, in unsigned long paramIndex) raises(DOMException);
    void moveTo(in float x, in float y);
    void lineTo(in float x, in float y);
    void quadTo(in float x1, in float y1, in float x2, in float y2);
    void curveTo(in float x1, in float y1, in float x2, in float y2, in float x3, in float y3);
    void close();
};
```

Constants

MOVE_TO
Numeric value is ASCII code of the letter 'M'.

LINE_TO
Numeric value is ASCII code of the letter 'L'.

CURVE_TO
Numeric value is ASCII code of the letter 'C'.

QUAD_TO
Numeric value is ASCII code of the letter 'Q'.

CLOSE
Numeric value is ASCII code of the letter 'Z'.

Attributes

numberOfSegments
Return number of segments in this path.

Methods

getSegment
Returns segment command by zero-based command index. Returns one of MOVE_TO, LINE_TO, CURVE_TO, QUAD_TO or CLOSE.

Parameters:

- cmdIndex The command index for the segment command to retrieve.

Raises:

- DOMException with error code INDEX_SIZE_ERR if segment index out of bounds.

getSegmentParam
Returns segment parameter by zero-based command index and zero-based parametr index.

Parameters:

- cmdIndex The command index for the segment command to retrieve.

Raises:

- DOMException with error code INDEX_SIZE_ERR if segment index out of bounds or param index out of bounds for this segment's type.

moveTo
Appends 'M' (absolute move) segment to the path with the specified coordinates.

Parameters:

- x The x-axis coordinate for the specified point.
- y The y-axis coordinate for the specified point.

lineTo
Appends 'L' (absolute line) segment to the path with the specified coordinates.

Parameters:

- x The x-axis coordinate for the specified point.
- y The y-axis coordinate for the specified point.

quadTo
Appends 'Q' (absolute quadratic curve) segment to the path.

Parameters:

- x1 The x-axis coordinate of the first control point.
- y1 The y-axis coordinate of the first control point.
- x2 The x-axis coordinate of the final end point.
- y2 The y-axis coordinate of the final end point.

curveTo
Appends 'C' (absolute cubic curve) segment to the path.

Parameters:

- x1 The x-axis coordinate of the first control point.
- y1 The y-axis coordinate of the first control point.
- x2 The x-axis coordinate of the second end point.
- y2 The y-axis coordinate of the second end point.
- x3 The x-axis coordinate of the final end point.
- y3 The y-axis coordinate of the final end point.

close
Numeric value is ASCII code of the letter 'Z'.

A.7.9 SVGMatrix

This interface represents an "SVGMatrix" datatype, identified by an affine transform. It can be used to read and modify the values of transform attribute as per SVG specification. Note that the mTranslate, inverse, mMultiply, mScale and mRotate methods in this interface mutate the SVGMatrix object and return a reference to the SVGMatrix instance itself, after performing the necessary matrix operation.

This matrix transforms source coordinates (x, y) into destination coordinates (x', y') by considering them to be a column vector and multiplying the coordinate vector by the matrix according to the following process:

```
[ x' ]    [  a  c  e  ]   [ x ]    [ a.x + c.y + e ]
[ y' ] =  [  b  d  f  ]   [ y ] =  [ b.x + d.y + f ]
[ 1  ]    [  0  0  1  ]   [ 1 ]    [        1      ]
```

IDL Definition

```
interface SVGMatrix
{
    float getComponent(in unsigned long index) raises(DOMException);
    SVGMatrix mMultiply(in SVGMatrix secondMatrix);
    SVGMatrix inverse() raises(SVGException);
    SVGMatrix mTranslate(in float x, in float y);
    SVGMatrix mScale(in float scaleFactor);
    SVGMatrix mRotate(in float angle);
};
```

No defined constants

No defined attributes

Methods

getComponent
Returns a component of the matrix by component's zero-based index. `getComponent(0)` is a, `getComponent(1)` is b, etc.

Parameters:

- index The index of the matrix component to retrieve.

Raises:

- DOMException - INDEX_SIZE_ERR if the `index` is invalid.

mMultiply
Performs matrix multiplication. This matrix is post-multiplied by another matrix, returning the resulting current matrix.

Parameters:

- secondMatrix The matrix to post-multiply with.

inverse
Returns a new instance of SVGMatrix containing the inverse of the current matrix.

Parameters:

Raises:

- SVGException - SVG_MATRIX_NOT_INVERTABLE when determinant of this matrix is zero.

mTranslate
Post-multiplies a translation transformation on the current matrix and returns the resulting current matrix. This is equivalent to calling `multiply(T)`, where `T` is an `SVGMatrix` object represented by the following matrix:

```
[   1     0     x   ]
[   0     1     y   ]
[   0     0     1   ]
```

Parameters:

- x The distance by which coordinates are translated in the X axis direction.
- y The distance by which coordinates are translated in the Y axis direction.

mScale
Post-multiplies a uniform scale transformation on the current matrix and returns the resulting current matrix. This is equivalent to calling `multiply(S)`, where `S` is an `SVGMatrix` object represented by the following matrix:

```
[   scaleFactor      0           0   ]
[   0           scaleFactor      0   ]
[   0                0           1   ]
```

Parameters:

- scaleFactor The factor by which coordinates are scaled along the X and Y axis.

mRotate
Post-multiplies a rotation transformation on the current matrix and returns the resulting current matrix. This is equivalent to calling `multiply(R)`, where `R` is an `SVGMatrix` object represented by the following matrix:

```
[ cos(angle) -sin(angle) 0 ]
[ sin(angle)  cos(angle) 0 ]
[ 0           0          1 ]
```

Parameters:

- angle The angle of rotation in degrees.

A.7.10 SVGLocatable

Interface for getting information about the location of elements. **Note:** Animations will have an effect on the values of bounding box.

The following example further clarify the behavior of the getBBox() method. The example have a short explanation, an SVG fragment and are followed by a set of bounding box values which have the following format:

```
[elementId] : {x, y, width, height} | {null}
```

where x, y, width and height define the values of the SVGRect object's returned from a getBBox call on the element with the specified id. There are a few cases where the bounding box may be null (see example 6).

Example #1: Simple groups and bounds

This first example shows the values returned by the getBBox method for various simple basic shapes and groups. In particular, it shows that the transform, on an element, does not change the value of its user space bounding box.

```
<g id="group1" transform="translate(10, 20)" fill="red">
<rect id="rect1" transform="scale(2)" x="10" y="10" width="50" height="50"/>
<rect id="rect2" x="10" y="10" width="100" height="100"/>
<g id="group2" transform="translate(10, 20)">
<rect id="rect3" x="0" y="10" width="150" height="50"/>
<circle id="circle1" cx="20" cy="20" r="100" />
</g>
</g>
```

Result:

```
[group1] : {-70.0, -60.0, 230.0, 200.0}

[rect1] : {10.0, 10.0, 50.0, 50.0}

[rect2] : {10.0, 10.0, 100.0, 100.0}

[group2] : {-80.0, -80.0, 230.0, 200.0}

[rect3] : {0.0, 10.0, 150.0, 50.0}

[circle1] : {-80.0, -80.0, 200.0, 200.0}
```

Example #2: Bounding box on zero width or height rectangle

This example illustrates that the bounding box on elements is based on the element's geometry coordinates. For example, the bounding box on a zero-width rectangle is defined (see below), even though the rectangle is not rendered.

```
<g id="group1" transform="translate(10, 20)" fill="red">
<rect id="rect2" x="10" y="10" width="400" height="0"/>
<g id="group2" transform="translate(10, 20)">
<rect id="rect3" x="0" y="10" width="150" height="50"/>
</g>
</g>
```

Result:

```
[group1] : {10.0, 10.0, 400.0, 70.0}

[rect2] : {10.0, 10.0, 400.0, 0.0}

[group2] : {0.0, 10.0, 150.0, 50.0}

[rect3] : {0.0, 10.0, 150.0, 50.0}
```

Example #3: Bounding Box on zero radius ellipses.

This is another example of how bounding boxes are based on the element's geometry. Here, the bounding box of an ellipse with a zero x-axis radius is still defined, even though the ellipse is not rendered.

```
<svg id="mySVG" width="10" height="20">
<g id="group1" transform="translate(10, 20)" fill="red">
<rect id="rect1" x="10" y="10" width="100" height="100"/>
<ellipse id="ellipse1" cx="20" cy="20" rx="0" ry="70" />
</g>
</svg>
```

Result:

```
[mySVG] : {20.0, -30.0, 100.0, 160.0}

[group1] : {10.0, -50.0, 100.0, 160.0}

[rect1] : {10.0, 10.0, 100.0, 100.0}

[ellipse1] : {20.0, -50.0, 0.0, 140.0}
```

Example #4: Viewports do not clip bounding boxes

This example shows that no matter what the viewport is on the root SVG element, the bounding boxes, based on the geometry, are still defined. Here, even though the root svg has a zero width, the bounding boxes for the root itself and its children is precisely defined.

```
<svg id="mySVG" width="0" height="50">
<g id="group1" transform="translate(10, 20)" fill="red" >
<rect id="rect1" x="10" y="10" width="50" height="50"/>
<g id="group2" transform="translate(10, 20)">
<rect id="rect2" x="0" y="10" width="150" height="0"/>
<circle id="circle1" cx="20" cy="20" r="500"/>
</g>
</g>
</svg>
```

Result:

```
[mySVG] : {-460.0, -440.0, 1000.0, 1000.0}

[group1] : {-470.0, -460.0, 1000.0, 1000.0}

[rect1] : {10.0, 10.0, 50.0, 50.0}

[group2] : {-480.0, -480.0, 1000.0, 1000.0}

[rect2] : {0.0, 10.0, 150.0, 0.0}

[circle1] : {-480.0, -480.0, 1000.0, 1000.0}
```

Example #5: getBBox on <use>

This example shows that the bounding box for a <use> element accounts for the x and y attributes defined on the element, just like the x and y attributes impact the bounding box computation on a <rect> or on an <image> element.

```
<svg>
<defs>
<rect id="myRect" x="0" y="0" width="60" height="40"/>
</defs>
<use id="myUse" xlink:href="#myRect" x="-30" y="-20"/>
</svg>
```

Result:

```
[myRect] : {0.0, 0.0, 60.0, 40.0}

[myUse] : {-30.0, -20.0, 60.0, 40.0}
```

Example #6: Empty group

This example shows that the bounding box for an empty group is null. By the same token, the bounding box of a <path> with an empty SVGPath (i.e., one with no path commands, which may happen after creating a new <path> element with a Document.createElementNS call) is also null.

```
<g id="emptyG"/>
```

Result:

```
[emptyG] : {null}
```

Example #7: Impact of display='none' and visibility='hidden'

This example shows how the bounding box of children with display='none' are not accounted for in the computation of their parent's bounding box. This reflects the definition of the display property and its impact on rendering and bounding box computation. The example also shows that elements with a 'hidden' visibility still contribute to their parent's bounding box computation.

```
<g id="g1">
<g id="g1.1.display.none" display="none">
<rect id="rect1" x="10" y="10" width="40" height="40"/>
<g/>
<rect id="rect2.visibility.hidden" visibility="hidden" x="30" y="60" width="10" height="20"/>
</g>
```

Result:

```
[g1] : {30.0, 60.0, 10.0, 20.0}

[g1.1.display.none] : {10.0, 10.0, 40.0, 40.0}

[rect1] : {10.0, 10.0, 40.0, 40.0}

[rec2.visibility.hidden] : {30.0, 60.0, 10.0, 20.0}
```

Example #8: Concatenating bounding boxes in the container's user space

This example shows how the concatenation and computation of bounding boxes for container element happens in the container's user space.

```
<g id="g1">
<line id="line1" x2="100" y2="100" transform="rotate(-45)"/>
</g>
```

Result:

```
[g1] : {0.0, 0.0, 141.42136, 0}

[line1] : {0.0, 0.0, 100.0, 100.0}
```

Example #9: No influence of stroke-width

This example illustrates that stroking has no impact on the computation of bounding boxes.

```
<g>
<line id="thickLine" stroke-width="10" x2="100" y2="0"/>
</g>
```

Result:

```
[thickLine] : {0.0, 0.0, 100.0, 0.0}
```

Example #10: No influence of viewBox

This example illustrates that viewBox has no impact on the computation of bounding boxes.

```
<svg id="rootSvg" width="500" height="300" viewBox="0 0 200 100">
<rect x="-100" y="-200" width="500" height="100"/>
</svg>
```

Result:

```
[rootSVG] : {-100, -200, 500, 100}
```

IDL Definition

```
interface SVGLocatable
{
    SVGRect   getBBox();
    SVGMatrix getScreenCTM();
    SVGRect   getScreenBBox();
};
```

No defined constants

No defined attributes

Methods

getBBox

Returns the tight bounding box in current user coordinate space (i.e., after application of the transform attribute or the viewbox). Tight bounding box is the smallest possible rectangle that includes the geometry of all contained graphics elements excluding stroke. The calculation is done in the user coordinate space of the element. When bounding box is calculated elements with display property (trait) set to none are ignored. Exact rules for the bounding box calculation are given in the SVG spec.

getScreenCTM

Returns the transformation matrix from current user units (i.e., after application of the transform attribute or the viewbox) to the parent user agent's notion of a "pixel". For display devices, ideally this represents a physical screen pixel. For other devices or environments where physical pixel sizes are not known, then an algorithm similar to the CSS2 definition of a "pixel" can be used instead. Note that `null` is returned if this element is not hooked into the document tree.

getScreenBBox

Returns the tight bounding box in screen coordinate space. Tight bounding box is the smallest possible rectangle that includes the geometry of all contained graphics elements excluding stroke. The box coordinates are in the screen coordinate space, which is connected to the current user coordinate space by the matrix returned by SVGLocatable#getScreenCTM method. Note that `null` is returned if this element is not hooked into the document tree.

A.7.11 SVGLocatableElement

This interface represents an enlement that has a physical location on the screen.

This interface is implemented by: <rect>, <circle>, <ellipse>, <line>, <path> <use> <image> <text>, <svg>, <a>, <video>, <animation>, <switch>, <foreignObject>, <polygon>, <polyline> and <g>.

IDL Definition

```
interface SVGLocatableElement : SVGElement, SVGLocatable
{
};
```

No defined constants

No defined attributes

No defined methods

A.7.12 TraitAccess

Trait manipulation interface. This interface is implemented by all svg elements. Each *trait* corresponds to an attribute which is parsed and understood by the element and in most cases animatable. Unlike attributes, each element has a well-defined set of traits and attempting to access an undefined trait is an error. Also unlike attributes traits can be typed. When getting and setting trait values, accessor of the correct type must be used or an exception will be thrown.

Initial trait values come from parsing corresponding attributes. In the environments where styling is supported, trait values also come from the stylesheets or styling attributes. If value is not specified, but corresponding attribute (or property for environments where styling is supported) is inherited, inherited value is returned as a result of the trait query method. If it is not inherited, default value is returned. The value which is returned is always a base value, before SMIL animation is applied.

The different getPresentationTrait methods return either the current animated value if the given trait is currently being animated (per the SMIL specification) or the base value if the given trait is not currently being animated.

Note about invalid/unsupported trait values: There are two situations where the various trait setter methods(such as setTrait, setFloatTrait or setPathTrait methods) consider a value invalid and throw a DOMException with the INVALID_ACCESS_ERR code. The trait methods will consider the value to be invalid if its 'in error' [link to definitions section 1.5].

The second situation is when the trait value is invalid with regards to animations currently applied to the trait. The value is considered invalid because it would put the animation, and therefore the document, in an error state. For example, if a <path> element has animations on its "d" attribute, trying to change the "d" attribute to a value incompatible with the animations will cause the exception to happen.

The trait setter methods will consider a value unsupported when it complies with the definition for an unsupported value [link to definitions section 1.5]. This will result in a DOMException thrown with the NOT_SUPPORTED_ERR code. For example, trying to set the "stroke-linejoin" trait to "foo" would cause this exception.

IDL Definition

```
interface TraitAccess
{
    DOMString getTrait(in DOMString name) raises(DOMException);
    DOMString getTraitNS(in DOMString namespaceURI, in DOMString name) raises(DOMException);
    float getFloatTrait(in DOMString name) raises(DOMException);
    SVGMatrix getMatrixTrait(in DOMString name) raises(DOMException);
    SVGRect getRectTrait(in DOMString name) raises(DOMException);
    SVGPath getPathTrait(in DOMString name) raises(DOMException);
    SVGRGBColor getRGBColorTrait(in DOMString name) raises(DOMException);
    DOMString getPresentationTrait(in DOMString name) raises(DOMException);
    DOMString getPresentationTraitNS(in DOMString namespaceURI, in DOMString name) raises(DOMException);
    float getFloatPresentationTrait(in DOMString name) raises(DOMException);
    SVGMatrix getMatrixPresentationTrait(in DOMString name) raises(DOMException);
    SVGRect getRectPresentationTrait(in DOMString name) raises(DOMException);
    SVGPath getPathPresentationTrait(in DOMString name) raises(DOMException);
    SVGRGBColor getRGBColorPresentationTrait(in DOMString name) raises(DOMException);
    void setTrait(in DOMString name, in DOMString value) raises(DOMException);
    void setTraitNS(in DOMString namespaceURI, in DOMString name, in DOMString value) raises(DOMException);
    void setFloatTrait(in DOMString name, in float value) raises(DOMException);
    void setMatrixTrait(in DOMString name, in SVGMatrix matrix) raises(DOMException);
    void setRectTrait(in DOMString name, in SVGRect rect) raises(DOMException);
    void setPathTrait(in DOMString name, in SVGPath path) raises(DOMException);
    void setRGBColorTrait(in DOMString name, in SVGRGBColor color) raises(DOMException);
};
```

No defined constants

No defined attributes

Methods

getTrait
Returns the trait value as String. In SVG Tiny only certain traits can be obtained as a String value. Syntax of the returned String matches the syntax of the corresponding attribute. This element is exactly equivalent to TraitAccess#getTraitNS with namespaceURI set to null.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to a String (SVG Tiny only).

getTraitNS
Same as TraitAccess#getTrait , but for namespaced traits. Parameter name must be a non-qualified trait name, i.e. without prefix.

Parameters:

- namespaceURI The namespaceURI of the trait to retrieve.
- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to a String (SVG Tiny only).

getFloatTrait
Get the trait value as float.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to a float

getMatrixTrait
Returns the trait value as SVGMatrix . The returned object is a copy of the actual trait value and will not change if the corresponding trait changes.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to SVGMatrix

getRectTrait
Returns the trait value as SVGRect . The returned object is a copy of the actual trait value and will not change if the corresponding trait changes.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to SVGRect

getPathTrait
Returns the trait value as SVGPath . The returned object is a copy of the actual trait value and will not change if the corresponding trait changes.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to SVGPath

getRGBColorTrait
Returns the trait value as SVGRGBColor . The returned object is a copy of the trait value and will not change if the corresponding trait changes. If the actual trait value is not an RGBColor (i.e. "none"), this method will return null.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to SVGRGBColor

getPresentationTrait
Returns the trait presentation value as String. In SVG Tiny only certain traits can be obtained as a String value. Syntax of the returned String matches the syntax of the corresponding attribute. This element is exactly equivalent to TraitAccess#getPresentationTraitNS with namespaceURI set to null.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to a String (SVG Tiny only).

getPresentationTraitNS
Same as TraitAccess#getPresentationTrait , but for namespaced traits. Parameter name must be a non-qualified trait name, i.e. without prefix.

Parameters:

- namespaceURI The namespaceURI of the trait to retrieve.
- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to a String (SVG Tiny only).

getFloatPresentationTrait
Get the trait presentation value as float.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to a float

getMatrixPresentationTrait
Returns the trait presentation value as SVGMatrix . The returned object is a copy of the actual trait value and will not change if the corresponding trait changes or as animation continue to affect the trait presentation value.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to SVGMatrix

getRectPresentationTrait
Returns the trait presentation value as SVGRect . The returned object is a copy of the actual trait value and will not change if the corresponding trait changes or as animation continue to affect the trait presentation value.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to SVGRect

getPathPresentationTrait
Returns the trait presentation value as SVGPath . The returned object is a copy of the actual trait value and will not change if the corresponding trait changes or as animation continue to affect the trait presentation value.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to SVGPath

getRGBColorPresentationTrait
Returns the trait presentation value as SVGRGBColor . The returned object is a copy of the trait value and will not change if the corresponding trait changes or as animation continue to affect the trait presentation value. If the actual trait presentation value is not an RGBColor (i.e. "none"), this method will return null.

Parameters:

- name The name of the trait to retrieve.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- DOMException with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to SVGRGBColor

setTrait
Set the trait value as String. In SVG Tiny only certain traits can be set through a String value. The syntax of the String that should be given as a value must be the same as syntax of the corresponding XML attribute value. Exactly equivalent to TraitAccess#setTraitNS with namespaceURI attribute set to null.

Parameters:

- name The name of the trait to be set.
- value the value of the trait to be set as String.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element, an attempt has been made to set an unsupported value.
- DOMException with error code TYPE_MISMATCH_ERR if the requested trait's value cannot be specified as a String
- DOMException with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait or null is specified.
- DOMException with error code NO_MODIFICATION_ALLOWED_ERR: if attempt is made to change readonly trait.

setTraitNS
Same as TraitAccess#setTrait , but for namespaced traits. Parameter name must be a non-qualified trait name, i.e. without prefix.

Parameters:

- namespaceURI The namespaceURI of the trait to be set.
- name The name of the trait to be set.
- value The value of the trait to be set as a string.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element, an attempt has been made to set an unsupported value.
- DOMException with error code TYPE_MISMATCH_ERR if the requested trait's value cannot be specified as a String
- DOMException with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait or null is specified. This error is also thrown when the <use> element is hooked into the document tree and the the value of xlink:href is set invalid.
- DOMException with error code NO_MODIFICATION_ALLOWED_ERR: if attempt is made to change readonly trait.

setFloatTrait
Set the trait value as float.

Parameters:

- name The name of the trait to be set.
- value The value of the trait to be set as float.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element.
- DOMException with error code TYPE_MISMATCH_ERR if the requested trait's value cannot be specified as a float
- DOMException with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait or null is specified.

setMatrixTrait
Set the trait value as SVGMatrix . Values in SVGMatrix are copied in the trait so subsequent changes to the given SVGMatrix have no effect on the value of the trait.

Parameters:

- name The name of the trait to be set.
- value The value of the trait to be set as SVGMatrix.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element.
- DOMException with error code TYPE_MISMATCH_ERR if the requested trait's value cannot be specified as an SVGMatrix
- DOMException with error code INVALID_ACCESS_ERR if the input matrix value is null.

setRectTrait
Set the trait value as SVGRect . Values in SVGRect are copied in the trait so subsequent changes to the given SVGRect have no effect on the value of the trait.

Parameters:

- name The name of the trait to be set.
- value The value of the trait to be set as SVGRect.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element.
- DOMException with error code TYPE_MISMATCH_ERR if the requested trait's value cannot be specified as an SVGRect
- DOMException with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait or null is specified. SVGRect is invalid if the width or height values are set to negative.

setPathTrait
Set the trait value as SVGPath . Values in SVGPath are copied in the trait so subsequent changes to the given SVGPath have no effect on the value of the trait.

Parameters:

- name The name of the trait to be set.
- value The value of the trait to be set as SVGPath.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element.
- DOMException with error code TYPE_MISMATCH_ERR if the requested trait's value cannot be specified as an SVGPath
- DOMException with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait or null is specified. SVGPath is invalid if it begins with any segment other than MOVE_TO segment. Note that an empty SVGPath is still a valid value.

setRGBColorTrait

Set the trait value as SVGRGBColor . Values in SVGRGBColor are copied in the trait so subsequent changes to the given SVGRGBColor have no effect on the value of the trait.

Parameters:

- name The name of the trait to be set.
- value The value of the trait to be set as SVGRGBColor.

Raises:

- DOMException with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element.
- DOMException with error code TYPE_MISMATCH_ERR if the requested trait's value cannot be specified as an SVGRGBColor
- DOMException with error code INVALID_ACCESS_ERR if the input value is null.

A.7.13 ElementTraversal

This interface provides a way to traverse elements in the DOM tree. It is needed mainly because SVG Tiny DOM does not expose character data Nodes. Each element in SVG Tiny document tree implements this interface. This applies to elements in the foreign namespaces as well.

IDL Definition

```
interface ElementTraversal
{
    readonly attribute Element firstElementChild;
    readonly attribute Element lastElementChild;
    readonly attribute Element nextElementSibling;
    readonly attribute Element previousElementSibling;
};
```

No defined constants

Attributes

firstElementChild
Returns the first child element node of this element. `null` if this element has no child elements.

lastElementChild
last child element node of this element. null if this element has no child elements.

nextElementSibling
Returns the next sibling element node of this element. `null` if this element has no element sibling nodes that come after this one in the document tree.

previousElementSibling
previous sibling element node of this element. null if this element has no element sibling nodes that come before this one in the document tree.

No defined methods

A.7.14 SVGElement

This interface represents an SVG element in the document tree. For implementations of the SVGTiny 1.2 profile an element's id can be set only if it does not already have an id. A DOMException with error code NO_MODIFICATION_ALLOWED_ERR is raised if an attempt is made to change an existing id. Other, future, profiles might allow changing the ID of an element. This interface also provides methods to traverse elements in the DOM tree.

This interface can also be used read and manipulate the value of "traits" associated with this SVGElement. Each *trait* corresponds to an attribute or property,which is parsed and understood by the element and in most cases animatable. Unlike attributes, each element has a well-defined set of traits and attempting to access undefined trait is an error. Also unlike attributes traits are typed and their values are normalized; for instance SVG path specification is parsed and all path commands are converted to their absolute variants, it is not possible to say through the value of the trait if a path command was absolute or relative. When getting and setting trait values, accessor of the correct type must be used or exception will be thrown.

Initial trait values come from parsing corresponding attributes. If value is not specified, but corresponding attribute (or property for environments where styling is supported) is inherited, inherited value is returned as a result of the trait query method. If it is not inherited, default value is returned. Default values are also returned in the case when there is no parent to inherit from, for ex: when you create a new element, set a trait value to 'inherit', but there is no parent for inheritance. It is important to note that the value which is returned is always a base value (i.e. before animation is applied), and this is true for both static and animated content.

Setting a trait value has the same effect as changing a corresponding attribute, but trait setters can operate on typed values. The value which is modified is always a base value. For inheritable traits the trait value can always be set to "inherit" (but querying the value will always return the actual inherited value as explained above).

A.7.15 Traits supported in this specification, SVG Tiny 1.2 uDOM

The table below shows the list of attributes and properties that SVG Tiny DOM 1.2 implementations must support. Each attribute row lists the allowed getter and setter (s). The 3:rd column specifies the default values that must be used for each attribute or property. Unless explicitly stated in the 'Comments' column, a supported attribute is accessible on all elements it can belong to. See the attribute section for a list of attributes and which elements they belong to.

For 'REQUIRED' attributes, there are two cases:

- i) The document is in error, if this attribute was not present at the time of loading.

- ii) When using uDOM API, the specified default value (in parenthesis) must be used.

Note: For some of the attributes and data types additional rules apply. These rules are defined below the table.

Attribute	Trait Getter	Trait Setter	Default Values	Comments
accumulate	getTrait [none | sum]	setTrait [none | sum]	none	
additive	getTrait [replace | sum]	setTrait [replace | sum]	replace	
attributeName	getTrait	setTrait		
audio-level	getFloatTrait [value >= 0 && value <= 1]	setFloatTrait [value >= 0 && value <= 1]	1.0f	
baseProfile	getTrait	Not available (readonly)	"tiny"	
begin	N/A	setTrait		
calcMode	getTrait [discrete | linear | paced | spline]	setTrait [discrete | linear | paced | spline]	linear (except for animateMotion) paced (for animateMotion)	
color	getRGBColorTrait [SVGRGBColor] getTrait [SVGRGBColor]	setRGBColorTrait [SVGRGBColor] setTrait [inherit | SVGRGBColor]	rgb(0,0,0)	-See RGB access rule.
cx	getFloatTrait	setFloatTrait	0.0f	
cy	getFloatTrait	setFloatTrait	0.0f	
d	getPathTrait [SVGPath]	setPathTrait [SVGPath]	REQUIRED(Empty SVGPath)	
display	getTrait [inline | none]	setTrait [inline | none | inherit]	"inline"	
dur	N/A	setTrait		
editable	getTrait [true | false]	setTrait [true | false]	"false"	
fill	getRGBColorTrait [SVGRGBColor] getTrait [none | <Paint Server> | SVGRGBColor]	setRGBColorTrait [SVGRGBColor] setTrait [none | currentColor | <Paint Server> | inherit | SVGRGBColor]	rgb(0,0,0)	-See RGB access rule. -Attribute defining the fill-color.
fill	getTrait[freeze | remove]	setTrait[freeze | remove]	"remove"	-Attribute defining the fill behavior of a smil animation element.
fill-opacity	getFloatTrait [value >= 0 && value <= 1]	setFloatTrait [value >= 0 && value <= 1] setTrait[inherit]	1.0f	
fill-rule	getTrait [nonzero | evenodd]	setTrait [nonzero | evenodd | inherit]	"nonzero"	
focusable	getTrait [true | false]	setTrait [true | false | auto]		
focusNorth	getTrait [null | IDREF]	setTrait [IDREF | auto]		
focusNorthEast	getTrait [null | IDREF]	setTrait [IDREF | auto]		
focusEast	getTrait [null | IDREF]	setTrait [IDREF | auto]		
focusSouthEast	getTrait [null | IDREF]	setTrait [IDREF | auto]		
focusSouth	getTrait [null | IDREF]	setTrait [IDREF | auto]		
focusSouthWest	getTrait [null | IDREF]	setTrait [IDREF | auto]		
focusWest	getTrait [null | IDREF]	setTrait [IDREF | auto]		
focusNorthWest	getTrait [null | IDREF]	setTrait [IDREF | auto]		
focusNext	getTrait [null | IDREF]	setTrait [IDREF | auto]		
focusPrev	getTrait [null | IDREF]	setTrait [IDREF | auto]		
focusable	getTrait [true | false]	setTrait [true | false | auto]		
font-family	getTrait [single, computed font-family value]	setTrait [same syntax as font-family attribute]	User-Agent	
font-size	getFloatTrait [value >= 0]	setFloatTrait [value >= 0] setTrait [inherit]	User-Agent	
font-style	getTrait [normal | italic | oblique]	setTrait [normal | italic | oblique | inherit]	"normal"	
font-weight	getTrait [100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900]	setTrait [normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit]	"normal"	
gradientUnits	getTrait [userSpaceOnUse | objectBoundingBox]	setTrait [userSpaceOnUse | objectBoundingBox]	"objectBoundingBox"	
height	getFloatTrait [value >= 0]	setFloatTrait [value >= 0]	REQUIRED(0.0f)	
id	getTrait	setTrait		
keyPoints	N/A	setTrait		
keySplines	N/A	setTrait		
keyTimes	N/A	setTrait		
max	N/A	setTrait		
min	N/A	setTrait		
offset	getFloatTrait[value >= 0 && value <= 1]	setFloatTrait[value >= 0 && value <= 1]		
opacity	getFloatTrait [value >= 0 && value <= 1]	setFloatTrait [value >= 0 && value <= 1] setTrait[inherit]		
path	getPathTrait [SVGPath]	setPathTrait [SVGPath]	REQUIRED(Empty SVGPath)	
r	getFloatTrait [value >= 0]	setFloatTrait [value >= 0]	REQUIRED (0.0f)	
repeatCount	N/A	setTrait		
repeatDur	N/A	setTrait		
restart	getTrait [always | whenNotActive | never]	setTrait [always | whenNotActive | never]	always	
rx	getFloatTrait [value >= 0]	setFloatTrait [value >= 0]	0.0f	

ry	getFloatTrait [value >= 0]	setFloatTrait [value >= 0]	0.0f	
snapShotTime	getFloatTrait [value >= 0]	setFloatTrait [value >= 0]	0.0f	
solid-color	getRGBColorTrait [SVGRGBColor] getTrait [SVGRGBColor]	setRGBColorTrait [SVGRGBColor] setTrait [SVGRGBColor | inherit]	rgb(0,0,0)	-See RGB access rule.
solid-opacity	getFloatTrait [value >= 0 && value <= 1]	setFloatTrait [value >= 0 && value <= 1]	1.0f	
stop-color	getRGBColorTrait [SVGRGBColor] getTrait [none | <Paint Server> | SVGRGBColor]	setRGBColorTrait [SVGRGBColor] setTrait(none | currentColor | <Paint Server> | inherit | SVGRGBColor]	rgb(0,0,0)	-See RGB access rule.
stop-opacity	getFloatTrait [value >= 0 && value <= 1]	setFloatTrait [value >= 0 && value <= 1] setTrait[inherit]	1.0f	
stroke	getRGBColorTrait [SVGRGBColor] getTrait [none | <Paint Server> | SVGRGBColor]	setRGBColorTrait [SVGRGBColor] setTrait [none | currentColor | <Paint Server> | inherit | SVGRGBColor]	"none"	-See RGB access rule.
stroke-dashoffset	getFloatTrait	setTrait [inherit] setFloatTrait	0.0f	
stroke-linecap	getTrait [butt | round | square]	setTrait [butt | round | square | inherit]	"butt"	
stroke-linejoin	getTrait [miter | round | bevel]	setTrait [miter | round | bevel | inherit]	"miter"	
stroke-miterlimit	getFloatTrait [value >= 1]	setTrait [inherit] setFloatTrait [value >= 1]	4.0f	
stroke-opacity	getFloatTrait [value >= 0 && value <= 1]	setFloatTrait [value >= 0 && value <= 1] setTrait[inherit]	1.0f	
stroke-width	getFloatTrait [value >= 0]	setTrait [inherit] setFloatTrait [value >= 0]	1.0f	
target	getTrait	setTrait	""	
"text"	getTrait("#text") ["text content"]	setTrait("#text", "text content")		-See Text Node Access for details.
text-anchor	getTrait [start | middle | end]	setTrait [start | middle | end | inherit]	"start"	
transform	getMatrixTrait [SVGMatrix] getTrait	setMatrixTrait [SVGMatrix] setTrait	Identity matrix (1,0,0,1,0,0)	-See Transform access rule.
type	For animateTransform: getTrait [translate | scale | rotate | skewX | skewY]	For animateTransform: setTrait [translate | scale | rotate | skewX | skewY]		
values	N/A	setTrait		
vector-effect	getTrait [defaultInon-scaling-stroke]	setTrait [defaultInon-scaling-stroke]	"default"	
version	getTrait	Not available (readonly)	"1.2"	
viewBox	getRectTrait [null, SVGRect]	setRectTrait [SVGRect] setTrait [none]	null	If the actual trait value is not an SVGRect (i.e. "none"), the getRectTrait method will return null.
viewport-fill	getRGBColorTrait [SVGRGBColor] getTrait [none | <Paint Server> | SVGRGBColor]	setRGBColorTrait [SVGRGBColor] setTrait [none | currentColor | <Paint Server> | SVGRGBColor]	"none"	-See RGB access rule.
viewport-fill-opacity	getFloatTrait [value >= 0 && value <= 1]	setFloatTrait [value >= 0 && value <= 1]	1.0f	
visibility	getTrait [visible | hidden]	setTrait [visible | hidden | inherit]	"visible"	
width	getFloatTrait [value >= 0]	setFloatTrait [value >= 0]	REQUIRED (0.0f)	
x	getFloatTrait (array of x-values not supported)	setFloatTrait (array of x-values not supported)	0.0f	
x1	getFloatTrait	setFloatTrait	0.0f	
x2	getFloatTrait	setFloatTrait	0.0f	
xlink:href	getTrait [absolute URI]	setTrait	""	
y	getFloatTrait (array of y-values not supported)	setFloatTrait (array of y-values not supported)	0.0f	
y1	getFloatTrait	setFloatTrait	0.0f	
y2	getFloatTrait	setFloatTrait	0.0f	
zoomAndPan	getTrait [disable | magnify]	setTrait [disable | magnify]	"magnify"	

Additional accessing rules

Accessing rules for RGBColorTrait

The getRGBColorTrait is used to get the RGB color. If the actual trait value is not an RGBColor, i.e. "none" or a link to a paint server (e.g. to a gradient or a solid-color), this method will raise a DOMException with error code TYPE_MISMATCH_ERR and getTrait should be used instead. setTrait must be used to set a color type that is not an RGBColor.

Accessing rules for 'transform' attribute

The transform attribute in SVGT1.2 can have two types of values. The 'normal' transformation list (e.g. scale, translate, rotate, matrix etc) or the newly introduced ref(svg) type. getMatrixTrait returns the current evaluated matrix in both cases. If the user needs to know that the transform attribute value was a 'ref' he must call getTrait. By using setTrait he can set the transform attribute to ref(svg).

Accessing rules for animation and font related elements

The following rule applies to Smil-animation elements (<animate>, <animateTransform>, <animateColor>, <animateMotion>, <set>) and font element with its children (<font>, <font-face, <missing-glyph>, <glyph>, <hkern>, <font-face-src>, <font-face-uri>, <font-face-name>).

These elements can be inserted and removed from the tree but they cannot be modified once inserted into the tree. Manipulating animations while they are in the DOM tree and possiblely active would result in undefined behaviour. This may also have an effect on past resolved times. This restriction means that if the author wishes to add animations via script, the element must be modified when it is not attached to the tree. A similar reasoning applies to the different font elements, modifying them while attached to the tree might lead to unpredictable result. Following is an example of adding animation to the tree including setting the properties prior to insertion.

Example: Animating via the uDOM

```
<svg version="1.2" baseProfile="tiny" xmlns="http://www.w3.org/2000/svg" id="svg-root"
width="100%" height="100%" viewBox="0 0 480 360">
<rect id="myRect" fill="green" x="10" y="10" width="200" height="100" stroke="black" stroke-width="2"/>
</svg>
```

A script (java) such as the following might be used to add an animation to the rectangle:

```
SVGElement newAnimate = (SVGElement)document.createElementNS(svgNS, "animate");
newAnimate.setTrait("attributeName", "fill");
newAnimate.setTrait("from", "red");
newAnimate.setTrait("to", "blue");
newAnimate.setTrait("dur", "5");
newAnimate.setTrait("repeatCount", "10");
Element myRect = document.getElementById("myRect");
myRect.appendChild(newAnimate);
```

IDL Definition

```
interface SVGElement : dom::Element, events::EventTarget, TraitAccess, ElementTraversal
{
    attribute DOMString id;
};
```

No defined constants

Attributes

id

On read : Returns the Element's id, null if no id specified. On write : Sets the Element's id attribute.

Raises:

- DOMException with error code NO_MODIFICATION_ALLOWED_ERR is raised if an attempt is made to change an existing Id.
- DOMException with error code INVALID_ACCESS_ERR is raised if the Id is not unique i.e. if this Id already exists in the document.

No defined methods

A.7.16 SVGAnimationElement

This interface represents an Animation element which is implemented by elements that suports timing control (smil-animation elements, media elements and time containers).

This interface is implemented by: <animate>, <animateTransform>, <animateColor>, <set>, <animateMotion>, <audio>, <video>, <animation> and <svg>

IDL Definition

```
interface SVGAnimationElement : SVGElement, smil::ElementTimeControl
{
};
```

No defined constants

No defined attributes

No defined methods

A.7.17 SVGVisualMediaElement

This interface represents a media element that is visual, i.e.have a physical location on the screen.

This interface is implemented by: <animation> and <video>.

IDL Definition

```
interface SVGVisualMediaElement : SVGLocatableElement, SVGAnimationElement
{
};
```

No defined constants

No defined attributes

No defined methods

A.7.18 EventListenerInitializer2

EventListenerInitializer2 allows event listeners to be initialized. In typical usage with Java, a <script> element references a JAR file which contains a manifest entry (SVG-Handler-Class) which identifies the class responsible for creating the event listeners. For ECMAScript, the script global object must implement the EventListenerInitializer2 interface, thus providing methods initializeEventListeners and createEventListener on the script global object. Further information about EventListenerInitializer2 can be found in the Scripting chapter.

Example #1: The usage of java together with the 'script' element

The example rely on the fact the 'myclasses.jar' jar file contains a MANIFEST file with the following entry:

```
SVG-Handler-Class: org.sample.SVGHandler
```

Given the following SVG file:

```
<svg xmlns:svg="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 0 500 500">
<script type="application/java-archive" xlink:href="myclasses.jar"/>
<rect id="therect" x="0" y="0" width="100" height="100"/>
</svg>
```

Given that the SVGHandler implementation available in the 'myclasses.jar' jar file is the following:

```
package org.sample.SVGHandler

import org.w3c.dom.*;
import org.w3c.dom.svg.*;

public class SVGHandler implements EventListenerInitializer2 {
public void initializeEventListeners(Element scriptElement) {
Document document = scriptElement.ownerDocument;
Element rec = document.getElementById("therect");
rec.addEventListener("click", new MyListener(), true);
}

public EventListener createEventListener(Element handlerElement) {}
}
```

An instance of "MyListener" listener will be called when a 'click' event will occur on the rect element.

Example #2: The usage of java together with the 'handler' element

The example rely on the fact the 'myclasses.jar' jar file contains a MANIFEST file with the following entry:

```
SVG-Handler-Class: org.sample.SVGHandler
```

Given the following SVG file:

```
<svg xmlns:svg="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 0 500 500">
  xmlns:ev="http://www.w3.org/2001/xml-events" xmlns:myns="http://sample.org/myNS">
<script id="init" type="application/java-archive" xlink:href="myclasses.jar"/>
<rect id="therect" x="0" y="0" width="100" height="100">
<handler type="application/java" ev:event="click"
xlink:href="#init" myns:listenerClass="MyListener">
</rect>
</svg>
```

Given that the SVGHandler implementation available in the 'myclasses.jar' jar file is the following:

```
package org.sample.SVGHandler

import org.w3c.dom.*;
import org.w3c.dom.svg.*;

public class SVGHandler implements EventListenerInitializer2 {
public void initializeEventListeners(Element scriptElement) {}

public EventListener createEventListener(Element handlerElement) {
EventListener listenerInstance = null;
try {
String listenerClass = ((TraitAccess)handlerElement).
getTraitNS("http://sample.org/myNS", "listenerClass");
listenerInstance = Class.forName(listenerClass).newInstance();
    } catch (Exception e) {}

return listenerInstance;
}
}
```

An instance of "MyListener" listener will be called when a 'click' event will occur on the rect element. What createEventListener does is totally in the hand of the Java developer, the listener instance can be hardcoded or fully configured from information put on the handler element as shown here.

IDL Definition

```
interface EventListenerInitializer2
{
    void initializeEventListeners( in dom::Element scriptElement );
    events::EventListener createEventListener( in dom::Element handlerElement );
};
```

No defined constants

No defined attributes

Methods

initializeEventListeners
For each <script> element, at load time for that element, the user agent finds the appropriate object which implements the EventListenerInitializer2 interface. (For ECMAScript, it is the script global object. For Java, it is the class identified by the SVG-Handler-Class manifest entry). The user agent then invokes the initializeEventListeners method, which allows the scripting code to register event listeners.

Parameters:

- scriptElement The <script> element which has been loaded.

createEventListener
For each <handler> element, at load time for that element, the user agent finds the appropriate object which implements the EventListenerInitializer2 interface. (For ECMAScript, it is the script global object. For Java, it is the class identified by the SVG-Handler-Class manifest entry). The user agent then invokes the createEventListener method, which allows the scripting code to register an appropriate event listener. For some scripting languages such as ECMAScript, the User Agent must define a default createEventListener method (see the Scripting chapter).

This method returns the event listener that will handle events. The user agent will register this event listener with the event target identified by the <handler> element.

Parameters:

- handlerElement The <handler> element which has been loaded.

A.7.19 SVGGlobal

The majority of scripted SVG documents in existence make use of the browser specific Window interface. SVG 1.2 specifies an SVGGlobal interface, taking into account the de-facto standard that already exists, as well as adding the new features present in SVG 1.2. SVGGlobal inherits from the Global interface, which is currently defined to be empty. The Global interface is designed to be the parent interface for language specific window objects. In scripting implementations, the methods and attributes defined by the Global object are normally part of the global execution context.

Interface SVGGlobal provides a global object for scripts embedded in a SVG document.

IDL Definition

```
interface SVGGlobal : global::Global
{
    global::Connection createConnection();
    void gotoLocation(in DOMString newURI);
    readonly attribute Document document;
    readonly attribute global::Global parent;
};
```

No defined constants

Attributes

document
The Document that this SVGGlobal operates on.

parent
The Global context of this document's parent. If the document has no notion of parent (e.g. when the document is displayed in a stand-alone viewer) parent is null.

Methods

createConnection
Creates a Connection object.

gotoLocation
Request that the user agent navigates to the given URL.

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

B IDL Definitions

This appendix contains the complete OMG IDL for the SVG UDOM.

The SVG UDOM IDL defines the model for the SVG UDOM. Note that the SVG UDOM IDL is defined such that some interfaces have more than one base class. The different standard language bindings for the SVG UDOM are responsible for defining how to map all aspects of the SVG UDOM into the given language, including how the language should implement interfaces with more than one base class.

```
pragmas
{
  java.jni.api.name="org.w3c.dom";
  core.package.vendor="W3C";
  core.package.name="SVG Tiny";
  core.package.id="svgt";
};

[
  comment="subsetted Core DOM";
  java.jni.api.name="org.w3c.dom";
]

module dom
{
    typedef string DOMString;
    typedef Object DOMObject;
    interface Node;
    interface Element;
```

```
    interface Document;

    exception DOMException
    {
        unsigned short code;
    };

    const unsigned short WRONG_DOCUMENT_ERR = 4;
    const unsigned short INDEX_SIZE_ERR = 1;
    const unsigned short HIERARCHY_REQUEST_ERR = 3;
    const unsigned short NO_MODIFICATION_ALLOWED_ERR = 7;
    const unsigned short NOT_FOUND_ERR = 8;
    const unsigned short NOT_SUPPORTED_ERR = 9;
    const unsigned short INVALID_STATE_ERR = 11;
    const unsigned short INVALID_MODIFICATION_ERR = 13;
    const unsigned short INVALID_ACCESS_ERR = 15;
    const unsigned short TYPE_MISMATCH_ERR = 17;

    interface Node
    {
        readonly attribute DOMString namespaceURI;
        readonly attribute DOMString localName;
        readonly attribute Node parentNode;
        readonly attribute Document ownerDocument;
        attribute DOMString textContent;
        Node appendChild(in Node newChild) raises(DOMException);
        Node insertBefore(in Node newChild, in Node refChild) raises(DOMException);
        Node removeChild(in Node oldChild) raises(DOMException);
    };

    interface Element : Node
    {
        DOMString getAttributeNS(in DOMString namespaceURI, in DOMString localName) raises(DOMException);
        void setAttributeNS(in DOMString namespaceURI, in DOMString localName, in DOMString value);
    };

    interface Document : Node
    {
        Element createElementNS(in DOMString namespaceURI, in DOMString qualifiedName) raises(DOMException);
        readonly attribute Element documentElement;
        Element getElementById(in DOMString id);
    };

};

module events
{
    typedef dom::DOMString DOMString;
    typedef dom::DOMException DOMException;
    typedef dom::Document Document;
    typedef dom::Element Element;
    interface EventTarget;
    interface EventListener;
    interface Event;

    interface EventTarget
    {
        void addEventListenerNS(in DOMString namespaceURI, in DOMString type, in EventListener listener, in boolean useCapture);
        void removeEventListenerNS(in DOMString namespaceURI, in DOMString type, in EventListener listener, in boolean useCapture);
    };

    interface EventListener
    {
        void handleEvent(in Event evt);
    };

    interface Event
    {
        readonly attribute EventTarget target;
        readonly attribute EventTarget currentTarget;
        readonly attribute DOMString type;
    };

    interface OriginalEvent
    {
        readonly attribute Event originalEvent;
    };

    interface MouseEvent : Event
    {
        readonly attribute long screenX;
        readonly attribute long screenY;
        readonly attribute long clientX;
        readonly attribute long clientY;
        readonly attribute unsigned short button;
    };

    interface TextEvent : Event
    {
        readonly attribute DOMString data;
    };

    interface KeyboardEvent : Event
    {
        readonly attribute DOMString keyIdentifier;
    };

    interface ConnectionEvent : Event
    {
        readonly attribute DOMString data;
    };

    interface TimeEvent : Event
    {
        readonly attribute long detail;
```

```
    };

    interface WheelEvent : Event
    {
        readonly attribute int wheelDelta;
    };

    interface ProgressEvent : Event
    {
        readonly attribute boolean lengthComputable;
        readonly attribute DOMString typeArg;
        readonly attribute unsigned long loaded;
        readonly attribute unsigned long total;
    };

};

module smil
{

    interface ElementTimeControl
    {
        void beginElementAt(in float offset);
        void beginElement();
        void endElementAt(in float offset);
        void endElement();
        void pauseElement();
        void resumeElement();
        readonly attribute boolean isPaused;
    };

};

module global
{
    interface Connection;

    interface Global
    {
    };

    interface Connection : events::EventTarget
    {
    typedef dom::DOMString DOMString;
        typedef dom::DOMException DOMException;
    void connect(in DOMString uri) raises(DOMException);
    void send(in DOMString data);
    void close();
    readonly attribute boolean connected;
    };

};

module svg
{
    typedef dom::DOMString DOMString;
    typedef dom::DOMException DOMException;
    typedef dom::Document Document;
    typedef dom::Element Element;
    typedef dom::DOMObject DOMObject;
    interface SVGSVGElement;
    interface SVGRGBColor;
    interface SVGRect;
    interface SVGPoint;
    interface SVGPath;
    interface SVGMatrix;
    interface SVGLocatableElement;
    interface SVGElement;
    interface SVGAnimationElement;
    interface SVGDocument;
    interface SVGGlobal;

    exception SVGException
    {
        unsigned short code;
    };

    const unsigned short SVG_INVALID_VALUE_ERR = 1;
    const unsigned short SVG_MATRIX_NOT_INVERTABLE = 2;

    interface SVGDocument : Document, events::EventTarget
    {
        readonly attribute SVGGlobal global;
    };

    interface SVGElementInstance : events::EventTarget
    {
        readonly attribute SVGElement correspondingElement;
        readonly attribute SVGElement correspondingUseElement;
    };

    interface SVGSVGElement : SVGLocatableElement : SVGAnimationElement
    {
        const unsigned short FOCUS_AUTO     = 1;
    const unsigned short FOCUS_NEXT     = 2;
    const unsigned short FOCUS_PREV     = 3;
    const unsigned short FOCUS_NORTH        = 4;
    const unsigned short FOCUS_NORTH_EAST   = 5;
    const unsigned short FOCUS_EAST     = 6;
        const unsigned short FOCUS_SOUTH_EAST   = 7;
        const unsigned short FOCUS_SOUTH    = 8;
        const unsigned short FOCUS_SOUTH_WEST   = 9;
        const unsigned short FOCUS_WEST     = 10;
        const unsigned short FOCUS_NORTH_WEST   = 11;
        attribute float currentScale;
```

```
        attribute float currentRotate;
        readonly attribute SVGPoint currentTranslate;
        readonly attribute SVGRect viewport;
        attribute float currentTime;
        SVGMatrix createSVGMatrixComponents(in float a, in float b, in float c, in float d, in float e, in float f);
        SVGRect createSVGRect();
        SVGPath createSVGPath();
        SVGRGBColor createSVGRGBColor(in long red, in long green, in long blue) raises(SVGException);
        void moveFocus(in unsigned short motionType) raises(DOMException);
        void setFocus(in DOMObject object) raises(DOMException);
        DOMObject getCurrentFocusedObject();
    };

    interface SVGRGBColor
    {
        readonly attribute unsigned long red;
        readonly attribute unsigned long green;
        readonly attribute unsigned long blue;
    };

    interface SVGRect
    {
        attribute float x;
        attribute float y;
        attribute float width;
        attribute float height;
    };

    interface SVGPoint
    {
        attribute float x;
        attribute float y;
    };

    interface SVGPath
    {
        const unsigned short MOVE_TO = 77;
        const unsigned short LINE_TO = 76;
        const unsigned short CURVE_TO = 67;
        const unsigned short QUAD_TO = 81;
        const unsigned short CLOSE = 90;
        readonly attribute unsigned long numberOfSegments;
        unsigned short getSegment(in unsigned long cmdIndex) raises(DOMException);
        float getSegmentParam(in unsigned long cmdIndex, in unsigned long paramIndex) raises(DOMException);
        void moveTo(in float x, in float y);
        void lineTo(in float x, in float y);
        void quadTo(in float x1, in float y1, in float x2, in float y2);
        void curveTo(in float x1, in float y1, in float x2, in float y2, in float x3, in float y3);
        void close();
    };

    interface SVGMatrix
    {
        float getComponent(in unsigned long index) raises(DOMException);
        SVGMatrix mMultiply(in SVGMatrix secondMatrix);
        SVGMatrix inverse() raises(SVGException);
        SVGMatrix mTranslate(in float x, in float y);
        SVGMatrix mScale(in float scaleFactor);
        SVGMatrix mRotate(in float angle);
    };

    interface SVGLocatable
    {
        SVGRect   getBBox();
        SVGMatrix getScreenCTM();
        SVGRect   getScreenBBox();
    };

    interface SVGLocatableElement : SVGElement, SVGLocatable
    {
    };

    interface TraitAccess
    {
        DOMString getTrait(in DOMString name) raises(DOMException);
        DOMString getTraitNS(in DOMString namespaceURI, in DOMString name) raises(DOMException);
        float getFloatTrait(in DOMString name) raises(DOMException);
        SVGMatrix getMatrixTrait(in DOMString name) raises(DOMException);
        SVGRect getRectTrait(in DOMString name) raises(DOMException);
        SVGPath getPathTrait(in DOMString name) raises(DOMException);
        SVGRGBColor getRGBColorTrait(in DOMString name) raises(DOMException);
        DOMString getPresentationTrait(in DOMString name) raises(DOMException);
        DOMString getPresentationTraitNS(in DOMString namespaceURI, in DOMString name) raises(DOMException);
        float getFloatPresentationTrait(in DOMString name) raises(DOMException);
        SVGMatrix getMatrixPresentationTrait(in DOMString name) raises(DOMException);
        SVGRect getRectPresentationTrait(in DOMString name) raises(DOMException);
        SVGPath getPathPresentationTrait(in DOMString name) raises(DOMException);
        SVGRGBColor getRGBColorPresentationTrait(in DOMString name) raises(DOMException);
        void setTrait(in DOMString name, in DOMString value) raises(DOMException);
        void setTraitNS(in DOMString namespaceURI, in DOMString name, in DOMString value) raises(DOMException);
        void setFloatTrait(in DOMString name, in float value) raises(DOMException);
        void setMatrixTrait(in DOMString name, in SVGMatrix matrix) raises(DOMException);
        void setRectTrait(in DOMString name, in SVGRect rect) raises(DOMException);
        void setPathTrait(in DOMString name, in SVGPath path) raises(DOMException);
        void setRGBColorTrait(in DOMString name, in SVGRGBColor color) raises(DOMException);
    };

    interface ElementTraversal
    {
        readonly attribute Element firstElementChild;
        readonly attribute Element lastElementChild;
        readonly attribute Element nextElementSibling;
        readonly attribute Element previousElementSibling;
    };
```

```
    interface SVGElement : dom::Element, events::EventTarget, TraitAccess, ElementTraversal
    {
        attribute DOMString id;
    };

    interface SVGAnimationElement : SVGElement, smil::ElementTimeControl
    {
    };

    interface SVGVisualMediaElement : SVGLocatableElement, SVGAnimationElement
    {
    };

    interface EventListenerInitializer2
    {
        void initializeEventListeners( in dom::Element scriptElement );
        events::EventListener createEventListener( in dom::Element handlerElement );
    };

    interface SVGGlobal : global::Global
    {
       global::Connection createConnection();
       void gotoLocation(in DOMString newURI);
       readonly attribute Document document;
       readonly attribute global::Global parent;
    };

};
```

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

C Implementation Requirements

Contents

This appendix is normative.

C.1 Introduction

The following are notes about implementation requirements corresponding to various features in the SVG language.

C.2 Unsupported elements, attributes, properties, attribute values and property values

User Agents shall ignore unknown attributes, attribute values, styling properties, styling property values, and descendant elements as follows:

- Unknown elements, including unknown elements in the SVG namespace, and their descendant elements are not rendered. The processing model for unknown elements is equivalent to an <svg:g> element with the 'display' property set to 'none'.
- For known elements in the SVG namespace, unknown attributes that have no namespace and known attributes with unsupported values are treated as if they hadn't been specified when rendering. Unsupported values are defined in the definitions section. Unsupported values for known attributes must also cause a highly perceivable warning in the user agent.
- Attributes put in the SVG namespace on any element are processed as unknown attributes.
- For unknown attributes in the XLink or XML Events namespaces, or known attributes in the XLink or XML Events namespaces with unsupported values, the user agent must treat the attributes as if they were not specified.
- User agents must implement the CSS2 rules for unknown styling properties and known styling properties that have unsupported values.

C.3 Error processing

There are various scenarios where an SVG document fragment is technically in error:

- When the content does not conform to the XML 1.0 specification [XML10], such as the use of incorrect XML syntax
- Other situations that are described as being *in error* in this specification

A document can go in and out of error over time. For example, document changes from the SVG DOM or from animation can cause a document to become *in error* and a further change can cause the document to become correct again.

When a document is in error, the User Agent shall provide a highly perceivable indication of error. The following error handling behavior represents a recommended approach that User Agents may choose to implement:

- The document should be rendered up to, but not including, the first element which has an error. Exceptions:
  - If a **'path'** element is the first element which has an error and the only errors are in the path data specification, then render the **'path'** up to the point of the path data error. For related information, see **'path'** element implementation notes.
  - If a **'polyline'** or **'polygon'** element is the first element which has an error and the only errors are within the points attribute, then render the **'polyline'** or **'polygon'** up to the segment with the error.

  This approach will provide a visual clue to the user or developer about where the error might be in the document.
- If the document has animations, the animations should stop at the point at which an error is encountered and the visual presentation of the document should

reflect the animated status of the document at the point the error was encountered.
- For visual rendering situations, an example of an indication of error would be to render a translucent colored pattern such as a checkerboard on top of the area where the SVG content is rendered.
- If the user agent has access to an error reporting capability such as status bar, it is recommended that the user agent provide whatever additional detail it can to enable the user or developer to quickly find the source of the error. For example, the user agent might provide an error message along with a line number and character number at which the error was encountered.

Because of situations where a block of scripting changes might cause a given SVG document fragment to go into and out of error, error processing shall occur only at times when document presentation (e.g., rendering to the display device) is updated. In particular, error processing shall be disabled whenever redraw has been suspended via DOM calls to suspendRedraw().

C.3.1 Example

This example shows the differences between erroneous and unsupported values. The first rectangle is conformant. The next two rectangles have unsupported values for the known attribute **width** . These values are not specifically listed as errors in the specification, but do not conform to the syntax for length. In both these cases the attribute should be ignored for rendering and the default value of zero should be used for the width. The last (fourth) rectangle in the example contains a value for 'width' that is explicitly listed as in error, thus at this point the document goes into error.

C.4 Namespace, version, baseProfile, requiredFeatures and requiredExtensions

The outermost <svg> element must be defined in the SVG namespace (e.g., <svg xmlns="http://www.w3.org/2000/svg">); otherwise the document is in error.

The attributes 'version', 'baseProfile', 'requiredFeatures' and 'requiredExtensions' identify the minimal functional requirements of an SVG user agent in order to render the content successfully. If these attributes identify a version, a profile, features or extensions, and the features are not supported by the user agent, then the user agent should alert or otherwise notify the user that the version of the file is not supported and suggest an alternate processing option (e.g., installing an updated version of the user agent) if such an option exists.

C.5 Clamping values which are restricted to a particular range

Some numeric attribute and property values have restricted ranges, such as color component values. When out-of-range values are provided, the user agent shall defer any error checking until after presentation time, as composited actions might produce intermediate values which are out-of-range but final values which are within range.

Color values are not in error if they are out-of-range, even if final computations produce an out-of-range color value at presentation time. It is recommended that user agents clamp color values to the nearest color value (possibly determined by simple clipping) which the system can process as late as possible (e.g., presentation time), although it is acceptable for user agents to clamp color values as early as parse time. Thus, implementation dependencies might preclude consistent behavior across different systems when out-of-range color values are used.

Opacity values out-of-range are not in error and should be clamped to the range 0 to 1 at the time which opacity values have to be processed (e.g., at presentation time or when it is necessary to perform intermediate filter effect calculations).

C.6 **'path'** element implementation notes

A conforming SVG user agent must implement path rendering as follows:

- Directionality and zero-length path segments:
  - Certain line-capping and line-joining situations require that a path segment have directionality at its start and end points. Zero-length path segments have no directionality. In these cases, the following algorithm is used to establish directionality: to determine the directionality of the start point of a zero-length path segment, go backwards in the path data specification within the current subpath until you find a segment which has directionality at its end point (e.g., a path segment with non-zero length) and use its ending direction; otherwise, temporarily consider the start point to lack directionality. Similarly, to determine the directionality of the end point of a zero-length path segment, go forwards in the path data specification within the current subpath until you find a segment which has directionality at its start point (e.g., a path segment with non-zero length) and use its starting direction; otherwise, temporarily consider the end point to lack directionality. If the start point has directionality but the end point doesn't, then the end point uses the start point's directionality. If the end point has directionality but the start point doesn't, then the start point uses the end point's directionality. Otherwise, set the directionality for the path segment's start and end points to align with the positive x-axis in user space.
  - If **'stroke-linecap'** is set to **butt** and the given path segment has zero length, do not draw the linecap for that segment; however, do draw the linecap for zero-length path segments when **'stroke-linecap'** is set to either **round** or **square**. (This allows round and square dots to be drawn on the canvas.)
- The S/s commands indicate that the first control point of the given cubic BÃ©zier segment is calculated by reflecting the previous path segments second control point relative to the current point. The exact math is as follows. If the current point is (curx, cury) and the second control point of the previous path segment is (oldx2, oldy2), then the reflected point (i.e., (newx1, newy1), the first control point of the current path segment) is:

```
(newx1, newy1) = (curx - (oldx2 - curx), cury - (oldy2 - cury))
               = (2*curx - oldx2, 2*cury - oldy2)
```

- A non-positive radius value is an error.
- Unrecognized contents within a path data stream (i.e., contents that are not part of the path data grammar) is an error.

C.7 Text selection implementation notes

The following implementation notes describe the algorithm for deciding which characters are selected during a text selection operation.

As the text selection operation occurs (e.g., while the user clicks and drags the mouse to identify the selection), the user agent determines a *start selection position* and an *end selection position*, each of which represents a position in the text string between two characters. After determining start selection position and end selection position, the user agent selects the appropriate characters, where the resulting text selection consists of either:

- no selection or
- a *start character*, an *end character* (possibly the same character), and all of the characters within the same **'text'** element whose position in the DOM is logically between the start character and end character.

On systems with pointer devices, to determine the *start selection position*, the SVG user agent determines which boundary between characters corresponding to rendered glyphs is the best target (e.g., closest) based on the current pointer location at the time of the event that initiates the selection operation (e.g., the mouse down event). The user agent then tracks the completion of the selection operation (e.g., the mouse drag, followed ultimately by the mouse up). At the end of the selection operation, the user agent determines which boundary between characters is the best target (e.g., closest) for the *end selection position*.

If no character reordering has occurred due to bidirectionality, then the selection consists of all characters between the *start selection position* and *end selection position*. For example, if a **'text'** element contains the string "abcdef" and the start selection position and end selection positions are 0 and 3 respectively (assuming the left side of the "a" is position zero), then the selection will consist of "abc".

When the user agent is implementing selection of bidirectional text, and when the selection starts (or ends) between characters which are not contiguous in logical order, then there might be multiple potential combinations of characters that can be considered part of the selection. The algorithms to choose among the combinations of potential selection options shall choose the selection option which most closely matches the text string's visual rendering order.

When multiple characters map inseparably to a given set of one or more glyphs, the user agent can either disallow the selection to start in the middle of the glyph set or can attempt to allocate portions of the area taken up by the glyph set to the characters that correspond to the glyph.

For systems which support pointer devices such as a mouse, the user agent is required to provide a mechanism for selecting text even when the given text has associated event handlers or links, which might block text selection due to event processing precedence rules (see Pointer events). One implementation option: For platforms which support a pointer device such as a mouse, the user agent may provide for a small additional region around character cells which initiates text selection operations but does not initiate event handlers or links.

C.8 Printing implementation notes

For user agents which support both zooming on display devices and printing, it is recommended that the default printing option produce printed output that reflects the display device's current view of the current SVG document fragment (assuming there is no media-specific styling), taking into account any zooming and panning done by the user, the current state of animation, and any document changes due to DOM and scripting . Thus, if the user zooms into a particular area of a map on the display device and then requests a hardcopy, the hardcopy should show the same view of the map as appears on the display device. If a user pauses an animation and prints, the hardcopy should show the same graphics as the currently paused picture on the display device. If scripting has added or removed elements from the document, then the hardcopy should reflect the same changes that would be reflected on the display.

When an SVG document is rendered on a static-only device such as a printer which does not support SVG's animation and scripting and facilities, then the user agent shall ignore any animation and scripting elements in the document and render the remaining graphics elements according to the rules in this specification.

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

D Conformance Criteria

Contents

D.1 Introduction

This specification defines conformance for several classes of products:

- Content (both complete SVG files, and those portions of XML files that are in the SVG namespace
- Software that *reads* SVG (with further conformance requirements for software that displays SVG after reading it)
- Software that *writes* SVG (including authoring tools and severs)

Some SVG displaying software will not display animations or other run-time modifications - for example, server side rasterizers or SVG-enabled printers.Therefore, this specification has different conformance levels for static and dynamic SVG viewers.

D.2 Terminology

Within this specification, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 (see [RFC2119]). However, for readability, these words do not appear in all uppercase letters in this specification.

At times, this specification recommends *good practice* for authors and user agents. These recommendations are not normative and conformance with this specification does not depend on their realization. These recommendations contain the expression "We recommend ...", "This specification recommends ...", or some similar wording.

All examples are informative, not normative. In the case of a conflict between the prose of this specification and the RelaxNG schema, the prose is authoritative (for example, the prose description of some attributes has a BNF grammar for allowed values, which the RelaxNG is not able to express). Similarly in the case of a conflict between a DTD or W3C XML Schema and the RelaxNG schema, the RelaxNG is authoritative (RelaxNG can express some constraints on content models that are problematic to express in W3C XML schema, and expresses namespaces in a natural and more general way than a DTD is able to).

D.3 SVG Content Conformance

D.3.1 Conforming SVG Document Fragments

An SVG document fragment is a *Conforming SVG Document Fragment* if it adheres to the specification described in this document (Scalable Vector Graphics (SVG) Specification) including SVG's schema (see Relax NG schema) and also:

- Is well-formed as defined in XML 1.1 and conforms to Namespaces in XML 1.1;
- Matches the NVDL script below, or alternatively if after having removed all elements not in the SVG namespace, and all attributes on elements in the SVG namespace that are in a namespace that isn't that of XLink, XML Events, or XML attributes, it validates against the Relax NG schema;
- Where the specification provides further constraints not expressed in the schema (such as for instance BNF grammars for attribute values), it complies to them.

NVDL script:

```
<rules xmlns='http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0'>
   <namespace ns='http://www.w3.org/2000/svg'>
     <validate schema='Tiny-1.2.rng'>
       <mode>
         <namespace ns='http://www.w3.org/2000/svg' match='attributes'>
           <reject/>
```

```
        </namespace>
        <namespace ns='' match='attributes'>
          <attach/>
        </namespace>
        <namespace ns='http://www.w3.org/1998/xml' match='attributes'>
          <attach/>
        </namespace>
        <namespace ns='http://www.w3.org/1999/xlink' match='attributes'>
          <attach/>
        </namespace>
        <namespace ns='http://www.w3.org/2001/xml-events' match='attributes'>
          <attach/>
        </namespace>
        <anyNamespace match='elements attributes'>
          <mode>
            <anyNamespace>
              <allow/>
            </anyNamespace>
          </mode>
        </anyNamespace>
      </mode>
    </validate>
  </namespace>
</rules>
```

The SVG language or these conformance criteria provide no designated size limits on any aspect of SVG content. There are no maximum values on the number of elements, the amount of character data, or the number of characters in attribute values.

D.3.2 Conforming SVG Stand-Alone Documents

A document is a *Conforming SVG Stand-Alone Document* if:

- it conforms to the criteria for Conforming SVG Document Fragment; and
- its root element is an 'svg' element in the SVG namespace.

D.3.3 Conforming SVG Included Document Fragments

SVG document fragments can be included within parent XML documents using the XML namespace facilities described in Namespaces in XML 1.1.

An SVG document fragment that is included within a parent XML document is a Conforming Included SVG Document Fragment if the SVG document fragment, when taken out of the parent XML document, conforms to the criteria for *Conforming SVG Document Fragments*.

In particular, note that individual elements from the SVG namespace *cannot* be used by themselves. Thus, the SVG part of the following document is *not* conforming:

```
<?xml version="1.0" standalone="no"?>
<ParentXML xmlns="http://ns.example/">
   <!-- Elements from ParentXML go here -->
   <!-- The following is not  conforming -->
   <z:rect xmlns:z="http://www.w3.org/2000/svg"
             x="0" y="0" width="10" height="10"/>
   <!-- More elements from ParentXML go here -->
</ParentXML>
```

Instead, for the SVG part to become a Conforming Included SVG Document Fragment, the document could be modified as follows:

```
<?xml version="1.0" standalone="no"?>
<ParentXML xmlns="http://ns.example/">
   <!-- Elements from ParentXML go here -->
   <!-- The following is conforming -->
   <z:svg xmlns:z="http://www.w3.org/2000/svg"
          width="100px" height="100px" >
     <z:rect x="0" y="0" width="10" height="10" />
   </z:svg>
   <!-- More elements from ParentXML go here -->
</ParentXML>
```

D.4 SVG Writer Conformance

D.4.1 Conforming SVG Generators

A *Conforming SVG Generator* is a program which:

- always creates at least one of Conforming SVG Document Fragments, Conforming SVG Stand-Alone Files or Conforming SVG Included Documents.
- does not create non-conforming SVG document fragments of any of the above types.

SVG generators are encouraged to follow W3C developments in the area of internationalization. Of particular interest is the *W3C Character Model* and the concept of *Webwide Early Uniform Normalization*, which promises to enhance the interchangability of Unicode character data across users and applications. Future versions of the SVG specification are likely to require support of the *W3C Character Model* in Conforming SVG Generators.

D.4.2 Conforming SVG Authoring Tools

Conforming SVG Authoring Tools must meet all the requirements of a Conforming SVG Generator. Additionally, a Conforming SVG Authoring Tool must conform to all of the Priority 1 accessibility guidelines from the document "Authoring Tool Accessibility Guidelines 1.0" [ATAG] that are relevant to generators of SVG content. (Priorities 2 and 3 are encouraged, but not required for conformance.)

D.4.3 Conforming SVG Servers

Conforming SVG Servers must meet all the requirements of a Conforming SVG Generator. In addition, Conforming SVG Servers using HTTP or other protocols that use Internet Media types must serve SVG stand-alone files with the media type `image/svg+xml`. Also, if the SVG is compressed with gzip or deflate, Conforming SVG Servers must indicate this with the "Content-Encoding:Â gzip" or "Content-Encoding:Â deflate" headers, as appropriate.

D.5 Conforming SVG Readers

D.5.1 Conforming SVG Interpreters

An SVG interpreter is a program which can parse and process SVG document fragments. Examples of SVG interpreters are server-side transcoding tools (e.g., a tool which converts SVG content into modified SVG content) or analysis tools (e.g., a tool which extracts the text content from SVG content). An SVG viewer also satisfies the requirements of an SVG interpreter in that it can parse and process SVG document fragments, where processing consists of rendering the SVG content to the target medium.

In a *Conforming SVG Interpreter*, the XML parser must be able to parse and process all XML constructs defined within [XML10] and [XML-NS].

There are two sub-categories of *Conforming SVG Interpreters*:

- *Conforming Static SVG Interpreters* must be able to parse and process the static language features of SVG that correspond to the feature string "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-static" (see Feature strings).
- In addition to the requirements for the static category, *Conforming Dynamic SVG Interpreters* must be able to parse and process the language features of SVG that correspond to the feature string "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-dynamic" (see Feature strings) and which support all of the required features in the SVG DOM described in this specification.

A Conforming SVG Interpreter must parse any SVG document correctly. It is not required to interpret the semantics of all features correctly. It needs only check the syntax of attribute values on elements in the SVG namespace, and element content models in the SVG namespace that it knows about from the profile it implements (SVG Tiny 1.2).

Note: A transcoder from SVG into another graphics representation, such as an SVG-to-raster transcoder, represents a viewer, and thus viewer conformance criteria apply. (See Conforming SVG Viewers.)

D.5.2 Conforming SVG Viewers

An SVG viewer is a program which can parse and process an SVG document fragment and render the contents of the document onto some sort of output medium such as a display or printer; thus, an *SVG Viewer* is also an *SVG Interpreter*.

There are two sub-categories of *Conforming SVG Viewers*:

- *Conforming Static SVG Viewers* support the static language features of SVG that correspond to the feature string "http://www.w3.org/TR/SVG11/feature#SVG-static" (see Feature strings). This category often corresponds to platforms and environments which only render static documents, such as printers.
- *Conforming Dynamic SVG Viewers* support the language features of SVG that correspond to the feature string "http://www.w3.org/TR/SVG11/feature#SVG-dynamic" (see Feature strings). This category often applies to platforms and environments such as common Web browsers which support user interaction and dynamic document content (i.e., documents whose content can change over time). (User interaction includes support for hyperlinking, events [e.g., mouse clicks], text selection, zooming and panning [see Interactivity]. Dynamic document content can be achieved via declarative animation or by scripts modifying the SVG DOM.)

Specific criteria that apply to both *Conforming Static SVG Viewers* and *Conforming Dynamic SVG Viewers*:

- The program must also be a Conforming SVG Interpreter,
- For interactive user environments, facilities must exist for zooming and panning of stand-alone SVG documents or SVG document fragments embedded within parent XML documents.
- In environments that have appropriate user interaction facilities, the viewer must support the ability to activate hyperlinks.
- If printing devices are supported, SVG content must be printable at printer resolutions with the same graphics features available as required for display (e.g., the specified colors must be rendered on color printers).
- On systems where this information is available, the parent environment must provide the viewer with information about physical device resolution. In situations where this information is impossible to determine, the parent environment shall pass a reasonable value for device resolution which tends to approximate most common target devices.
- The viewer must support JPEG/JFIF [JPEG] [JFIF] and PNG [PNG] image formats. Other image formats may be supported in addition.
- Resampling of image data must be consistent with the specification of property **'image-rendering'**.
- The viewer must support alpha channel blending of the image of the SVG content onto the target canvas.
- SVG implementations must correctly support gzip-encoded and deflate-encoded SVG data streams. SVG implementation that support HTTP must support these encodings according to the HTTP 1.1 specification [RFC2616]; in particular, the client must specify with an "Accept-Encoding:" request header [HTTP-ACCEPT-ENCODING] those encodings that it accepts, including at minumum gzip and deflate, and then decompress any gzip-encoded and deflate-encoded data streams that are downloaded from the server. If the implementation supports progressive rendering, the implementation should also support progressive rendering of compressed data streams.
- The viewer must support content using the "data:" protocol [RFC2397] wherever IRI referencing of whole documents (such as raster images, SVG documents, fonts and color profiles) is permitted within SVG content. This support must include use of base64 encoded content. (Note: fragments of SVG content which do not constitute an entire SVG document are not available using the "data:" protocol.)
- The viewer must support the following W3C Recommendations with regard to SVG content:
  - complete support for the XML 1.0 specification [XML10].
  - complete support for "Namespaces in XML 1.1" [XML-NS], including inclusion of non-SVG namespaces within SVG content. (Note that data from non-SVG namespaces are included in the DOM but are otherwise ignored from the point of view of rendering and interaction.)
- All visual rendering should be accurate to within one device pixel to the mathematically correct result at the initial 1:1 zoom ratio. It is suggested that viewers attempt to keep a high degree of accuracy when zooming.
- On systems which support accurate sRGB [SRGB] color, all sRGB color computations and all resulting color values must be accurate to within one sRGB color component value, where sRGB color component values range from 0 to 255.

Although anti-aliasing support is not a strict requirement for a Conforming SVG Viewer, it is highly recommended for display devices. Lack of anti-aliasing support will generally result in poor results on display devices.

Specific criteria that apply to only *Conforming Dynamic SVG Viewers*:

- In Web browser environments, the viewer must have the ability to search and select text strings within SVG content.
- In interactive environments, the viewer must have the ability to select and copy text from SVG content to the system clipboard.

The Web Accessibility Initiative [WAI] is defining "User Agent Accessibility Guidelines 1.0" [UAAG]. Viewers are encouraged to conform to the Priority 1 accessibility guidelines defined in this document, and preferably also Priorities 2 and 3. Once the guidelines are completed, a future version of this specification is likely to require conformance to the Priority 1 guidelines in Conforming SVG Viewers.

A *Conforming SVG Viewer* must be able to apply styling properties to SVG content using presentation attributes (see properties shared with CSS and XSL).

If the user agent includes an XHTML or SMIL viewing capability, then a *Conforming SVG Viewer* must support resources of MIME type "image/svg+xml" wherever raster image external resources can be used, such as in the XHTML **'img'** element.

If the user agent can apply CSS/XSL styling properties to XML documents (including XHTML, SMIL, or XForms documents), then a *Conforming SVG Viewer* must support resources of MIME type "image/svg+xml" in CSS/XSL properties that can refer to raster image resources (e.g., **'background-image'**).

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

E Conformance to WQ Framework Specification Guidelines

Contents

E.1 Introduction

This table is derived from the checklist of all defined requirements and good practices from the *QA Framework: Specification Guidelines* [QAF-SPEC] . For each requirement and good practive, the table links to the part of the SVG Tiny 1.2 specification that satisfied the requirement or best practice.

E.2 Checklist table

Guidelines	YES	NO	N/A
1 Specifying Conformance			
1.1 A conformance clause is essential			
Requirement : Include a conformance clause.	YES		
Good Practice : Define the specification's conformance model in the conformance clause.	YES		
Good Practice : Specify in the conformance clause how to distinguish normative from informative content.	YES		
1.2 Specify how to make conformance claims			
Good Practice : Provide the wording for conformance claims.		no	
Good Practice : Provide an Implementation Conformance Statement proforma.		no	
Good Practice : Require an Implementation Conformance Statement as part of valid conformance claims.		no	
2. Setting up Groundrules			
2.1. Scope			
Requirement : Define the scope.	YES		
Good Practice : Provide examples, use cases, and graphics.	YES		
2.2 What needs to conform			
Requirement : Identify who or what will implement the specification.	YES		
2.3 Normative (and non-normative) references			
Requirement : Make a list of normative references.	YES		
Good Practice : Do systematic reviews of normative references and their implications.	YES		
4. Managing Variability			
4.1 Subdivide			
Good Practice : Create subdivisions of the technology when warranted.	YES		
Requirement : If the technology is subdivided, then indicate which subdivisions are mandatory for conformance.	YES		
Requirement : If the technology is subdivided, then address subdivision constraints.	YES		
Good Practice : If the technology is profiled, define rules for creating new profiles.		no	
4.2 Optionality and Options			
Good Practice : Make sure there is a need for the optional feature.	YES		
Good Practice : Clearly identify optional features.	YES		
Good Practice : Indicate any limitations or constraints on optional features.	YES		
4.3 Extensibility and Extensions			
Requirement : Address Extensibility.	YES		
Good Practice : If extensibility is allowed, define an extension mechanism.	YES		
Good Practice : Warn implementers to create extensions that do not interfere with conformance.		no	
Good Practice : Define error handling for unknown extensions.	YES		
4.4 Deprecation			
Requirement : Identify deprecated features.			None
Requirement : Define how deprecated feature is handled by each class of product.			n/a
Good Practice : Explain how to avoid using a deprecated feature.			n/a
Good Practice : Identify obsolete features.			None
4.5 Error Handling			
Good Practice : Define an error handling mechanism.	YES		
5. Do Quality Control			
Good Practice : Define an internal publication and review process.	YES		
Good Practice : Do a systematic and thorough review.	YES		
Good Practice : Write sample code or tests.	YES		
Good Practice : Write Test Assertions.		no	
Good Practice : Use formal languages and define which from prose and formal languages has priority.	YES		

SVG Tiny 1.2 - 20050413 Top | Contents | Previous | Next | Elements | Attributes

F Accessibility Support

Contents

This appendix is informative, not normative.

F.1 WAI Accessibility Guidelines

This appendix explains how accessibility guidelines published by W3C's Web Accessibility Initiative (WAI) apply to SVG.

1. The "Web Content Accessibility Guidelines 1.0" [WCAG] explains how authors can create Web content that is accessible to people with disabilities.
2. The "Authoring Tool Accessibility Guidelines 1.0" [ATAG] explains how developers can design accessible authoring tools such as SVG authoring tools. To conform to the SVG specification, an SVG authoring tool must conform to ATAG (priority 1). SVG support for element grouping and reuse is relevant to designing accessible SVG authoring tools.
3. The "User Agent Accessibility Guidelines 1.0" [UAAG] explains how developers can design accessible user agents such as SVG-enabled browsers. To conform to the SVG specification, an SVG user agent should conform to UAAG. SVG support for scaling, style sheets, the DOM, and metadata are all relevant to designing accessible SVG user agents.

The W3C Note "Accessibility Features of SVG" [SVG-ACCESS] explains in detail how the requirements of the three guidelines apply to SVG.

F.2 SVG Content Accessibility Guidelines

This section explains briefly how authors can create accessible SVG documents; it summarizes and builds upon "Accessibility Features of SVG" [SVG-ACCESS].

Provide text equivalents for graphics.

- When the text content of a graphic (e.g., in a **'text'** element) explains its function, no text equivalent is required. Use the **'title'** child element to explain the function **'text'** elements whose meaning is not clear from their text content.
- When a graphic does not include explanatory text content, it requires a text equivalent. If the equivalent is complex, use the **'desc'** element, otherwise use the **'title'** child element.
- If a graphic is built from meaningful parts, build the description from meaningful parts.

Do not rely on color alone.

- Do not use color alone to convey semantic information.
- Ensure adequate color contrast.

Use markup correctly.

- Separate structure from presentation.
- Use the **'g'** element and rich descriptions to structure SVG documents. Reuse named objects.
- Publish highly-structured documents, not just graphical representations. Documents that are rich in structure may be rendered graphically, as speech, or as Braille. For example, express mathematical relationships in MathML [MATHML] and use SVG for explanatory graphics.
- Author documents that validate to the SVG RelaxNG grammar.

Use text for text, and graphics for graphics

- Represent text as character data, not as glyphs, images or curves.
- Style text with fonts. Authors may describe their own fonts in SVG.
- Do not use 'pi' fonts or picture fonts to represent small graphics. The resulting garbage text does not conform to [CHARMOD] and confuses text to speech engines.

Provide a default reading order

- Use **'textArea'** elements to provide text that wraps in a box, rather than using script to wrap the text or using a sequence of unrelated **'text'** elements.

Clarify natural language usage.

- Use **xml:lang** to identify the natural language of content and changes in natural language. This ensures that textual content can be spell-checked, or converted to speech.
- Use the **'systemLanguage'** test attribute to provde language-specific alternative text and/or graphics.

Allow for rich navigation

- Do not assume that all devices have a pointer device. Allow for keyboard navigation as well.
- Provide links for 8-way directional navigation.

Ensure that dynamic content is accessible.

- Ensure that text equivalents for dynamic content are updated when the dynamic content changes.
- Avoid storing dynamic text in ecmascript arrays or in XSLT stylesheets. This makes it less accessible, and also increases the difficulty of localising the text or providing multilingual alternatives.
- Ensure that SVG documents are still usable when scripts or other programmatic objects are turned off or not supported.

F.3 SVG User Agent Accessibility Guidelines

This section explains how implementors of SVG User Agents can make their software more accessible.

Provide access to color information

- SVG User Agents should implement and document APIs so that assistive technologies can have access to color information on individual elements.
- SVG User Agents should provide an optional high contrast view.

Provide a text-only view

- SVG user Agents should provide mechanisms that allow assistive technologies to achieve a useful text-only view. Examples include a DOM explorer, a synchronized text only view, or an XSLT stylesheet to convert the textual content to XHTML.

Allow keyboard navigation

- SVG user Agents should provide keyboard access to zooming and panning

- SVG user Agents should allow animations, audio, and video to be started, stopped and paused using the keyboard.

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

G Internationalization Support

Contents

This appendix is informative, not normative.

G.1 Introduction

This appendix provides a brief summary of SVG's support for internationalization. The appendix is hyperlinked to the sections of the specification which elaborate on particular topics.

G.2 Internationalization and SVG

SVG is an application of XML [XML10] and thus supports Unicode [UNICODE], which defines the universal character set.

Additionally, SVG provides a mechanism for precise control of the glyphs used to draw text strings, which is described in SVG Fonts. This allows content to supply, or link to, SVG Fonts to display international text, even if no fonts for that language are installed on the client machine.This facility provides:

- the ability to specify glyphs for any character
- the ability to specify ligatures for arbitrary strings of characters
- the ability to follow the guidelines for normalizing character data for the purposes of enhanced interoperability (see [CHARMOD]), while still having precise control over the glyphs that are drawn.

SVG Tiny 1.2 supports:

- Horizontal, left-to-right text, for example as found in Roman scripts
- Bidirectional text (for languages such as Arabic and Hebrew).

SVG Tiny 1.2 does not support vertical text, but SVG Full 1.2 does.

SVG fonts support contextual glyph selection, for example for Arabic contextual forms, and language-dependent Han glyphs.

Multi-language SVG documents are possible by utilizing the **systemLanguage** attribute to have different text strings or graphics appear based on the client machine's language setting.

G.3 SVG Internationalization Guidelines

SVG generators should follow W3C guidelines for normalizing character data [CHARMOD].

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

H Minimizing SVG File Sizes

This appendix is informative, not normative.

Considerable effort has been made to make SVG file sizes as small as possible while still retaining the benefits of XML and achieving compatibility and leverage with other W3C specifications.

Here are some of the features in SVG that promote small file sizes:

- SVG's path data definition was defined to produce a compact data stream for vector graphics data: all commands are one character in length; relative coordinates are available; separator characters do not have to be supplied when tokens can be identified implicitly; smooth curve formulations are available (cubic Bziers, quadratic Bziers and elliptical arcs) to prevent the need to tesselate into polylines; and shortcut formulations exist for common forms of cubic Bzier segments, quadratic Bzier segments, and horizontal and vertical straight line segments so that the minimum number of coordinates need to be specified.
- Text can be specified using XML character data -- no need to convert to outlines.
- SVG contains a facility for defining symbols once and referencing them multiple times using different visual attributes, different sizing and positioning.

Additionally, HTTP/1.1 allows for compressed data to be passed from server to client, which can result in significant file size reduction. Here are some sample compression results using gzip compression on SVG documents:

Uncompressed SVG	With gzip compression	Compression ratio
12,912	2,463	81%
12,164	2,553	79%
11,613	2,617	77%
18,689	4,077	78%
13,024	2,041	84%

A related issue is progressive rendering. Some SVG viewers will support:

- the ability to display the first parts of an SVG document fragments as the remainder of the document is downloaded from the server; thus, the user will see part of the SVG drawing right away and interact with it, even if the SVG file size is large.
- delayed downloading of images and fonts. Just like some HTML browsers, some SVG viewers will download images and WebFonts last, substituting a temporary image and system fonts, respectively, until the given image and/or font is available.

Here are techniques for minimizing SVG file sizes and minimizing the time before the user is able to start interacting with the SVG document fragments:

- Construct the SVG file such that any links which the user might want to click on are included at the beginning of the SVG file
- Use default values whenever possible rather than defining all attributes and properties explicitly.
- Take advantage of the path data data compaction facilities: use relative coordinates; use *h* and *v* for horizontal and vertical lines; use *s* or *t* for cubic and quadratic Bzier segments whenever possible; eliminate extraneous white space and separators.
- Utilize symbols if the same graphic appears multiple times in the document
- For user agents that support styling with CSS, utilize CSS property inheritance and selectors to consolidate commonly used properties into named styles or to assign the properties to a parent 'g' element.

SVG Tiny 1.2 - 20050413 Top Contents Previous Next Elements Attributes

I Feature strings

This appendix is normative.

The following are the feature strings for the **requiredFeatures** attribute. These same feature strings apply to the hasFeature method call that is part of the SVG DOM's support for the DOMImplementation interface defined in [DOM3-CORE]. In some cases the feature strings map directly to SVG modules, in others they represent some functionality of the User Agent (that it is a dynamic viewer for example).

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#SVG
User Agent Supports:
At least one of the following (all of which are described subsequently):

- "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-static"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-static-DOM"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-animated"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-all"

(Because the feature string "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG" can be ambiguous in some circumstances, it is recommended that more specific feature strings be used.)

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-static
User Agent Supports:
All of the following features (described below):

- "http://www.w3.org/Graphics/SVG/feature/1.2/#CoreAttribute"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#Structure"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#ConditionalProcessing"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#ConditionalProcessingAttribute"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#Image"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#Prefetch"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#Shape"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#Text"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#PaintAttribute"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#OpacityAttribute"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#GraphicsAttribute"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#Gradient"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#SolidColor"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#XlinkAttribute"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#ExternalResourcesRequiredAttribute"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#Font"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#Hyperlinking""
- "http://www.w3.org/Graphics/SVG/feature/1.2/#Extensibility"

For SVG viewers, "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-static" indicates that the viewer can process and render successfully all of the language features in the modules corresponding to the features listed above.

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-static-DOM
User Agent Supports:
All of the DOM interfaces and methods that correspond to the language features for "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-static" as well as the following features:

- "http://www.w3.org/Graphics/SVG/feature/1.2/#Script"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#Handler"

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-animated
User Agent Supports:
All of the language features from "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-static" plus the following feature:

- http://www.w3.org/Graphics/SVG/feature/1.2/#TimedAnimation

For SVG viewers running on media capable of rendering time-based material, such as displays, "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-animated" indicates that the viewer can process and render successfully all of the corresponding language features.

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-all
User Agent Supports:
All of the language features from:

- "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-static-DOM"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-animated"

Plus the following features:

- "http://www.w3.org/Graphics/SVG/feature/1.2/#Audio"

- "http://www.w3.org/Graphics/SVG/feature/1.2/#Video"
- "http://www.w3.org/Graphics/SVG/feature/1.2/#Animation"

For SVG viewers running on media capable of rendering time-based material, such as displays, "http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-all" indicates that the viewer can process and render successfully all of the corresponding language features.

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#CoreAttribute
User Agent Supports:
Core Attribute Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Structure
User Agent Supports:
Structure Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#ConditionalProcessing
User Agent Supports:
Conditional Processing Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#ConditionalProcessingAttribute
User Agent Supports:
Conditional Processing Attribute Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Image
User Agent Supports:
Image Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Prefetch
User Agent Supports:
Prefetch Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Shape
User Agent Supports:
Shape Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Text
User Agent Supports:
Text Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#PaintAttribute
User Agent Supports:
Paint Attribute Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#OpacityAttribute
User Agent Supports:
Opacity Attribute Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#GraphicsAttribute
User Agent Supports:
Graphics Attribute Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Gradient
User Agent Supports:
Gradient Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#SolidColor
User Agent Supports:
Solid Color Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Hyperlinking
User Agent Supports:
Hyperlinking Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#XlinkAttribute
User Agent Supports:
Xlink Attribute Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#ExternalResourcesRequired
User Agent Supports:
ExternalResourcesRequired Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Script
User Agent Supports:
Script Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Handler
User Agent Supports:

Handler Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#TimedAnimation
User Agent Supports:
Timed Animation Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Animation
User Agent Supports:
Animation Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Audio
User Agent Supports:
Audio Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Video
User Agent Supports:
Video Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Font
User Agent Supports:
Font Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#Extensibility
User Agent Supports:
Extensibility Module

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#TransformedVideo
User Agent Supports:
The ability to perform any transformation (including scaling) on video content.

Feature String:
http://www.w3.org/Graphics/SVG/feature/1.2/#ComposedVideo
User Agent Supports:
The ability to compose video content with other content.

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

J Elements Table

Elements	Attributes	prop.	Possible Children
a	class, externalResourcesRequired, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, id, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, target, transform, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space	âœ”	The 'a' element may contain any element that its parent may contain, except itself.
animate	accumulate, additive, attributeName, attributeType, begin, by, calcMode, class, dur, end, fill, from, id, keySplines, keyTimes, max, min, repeatCount, repeatDur, restart, to, values, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space	âœ˜	desc, handler, metadata, switch, title
animateColor	accumulate, additive, attributeName, attributeType, begin, by, calcMode, class, dur, end, fill, from, id, keySplines, keyTimes, max, min, repeatCount, repeatDur, restart, to, values, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space	âœ˜	desc, handler, metadata, switch, title
animateMotion	accumulate, additive, begin, by, calcMode, class, dur, end, fill, from, id, keyPoints, keySplines, keyTimes, max, min, origin, path, repeatCount, repeatDur, restart, rotate, to, type, values, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space	âœ˜	desc, handler, metadata, mpath, switch, title
animateTransform	accumulate, additive, attributeName, attributeType, begin, by, calcMode, class, dur, end, fill, from, id, keySplines, keyTimes, max, min, repeatCount, repeatDur, restart, to, type, values, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space	âœ˜	desc, handler, metadata, switch, title
animation	audio-level, begin, class, dur, end, externalResourcesRequired, fill, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, height, id, overflow, preserveAspectRatio, repeatCount, repeatDur, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, restart, syncBehavior, syncBehaviorDefault, syncTolerance, syncToleranceDefault, systemLanguage, width, x, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space, y	âœ˜	animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title
audio	audio-level, begin, class, dur, end, externalResourcesRequired, id, repeatCount, repeatDur, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, restart, syncBehavior, syncBehaviorDefault, syncTolerance, syncToleranceDefault, systemLanguage, type, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space	âœ˜	animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title
circle	class, cx, cy, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, id, r, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, transform, xml:base, xml:id, xml:lang, xml:space	âœ”	animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title

defs	class, id, transform, xml:base, xml:id, xml:lang, xml:space	â€" a, animate, animateColor, animateMotion, animateTransform, animation, audio, circle, defs, desc, discard, ellipse, font, font-face, foreignObject, g, handler, image, line, linearGradient, metadata, path, polygon, polyline, prefetch, radialGradient, rect, script, set, solidColor, switch, text, textArea, title, use, video
desc	class, id, xml:base, xml:id, xml:lang, xml:space	â€˜ <text>
discard	begin, class, id, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space	â€˜ desc, handler, metadata, switch, title
ellipse	class, cx, cy, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, id, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, rx, ry, systemLanguage, transform, xml:base, xml:id, xml:lang, xml:space	â€" animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title
font	class, externalResourcesRequired, horiz-adv-x, horiz-origin-x, id, xml:base, xml:id, xml:lang, xml:space	â€˜ desc, font-face, glyph, hkern, metadata, missing-glyph, switch, title
font-face	accent-height, alphabetic, ascent, bbox, cap-height, class, descent, externalResourcesRequired, font-family, font-size, font-stretch, font-style, font-variant, font-weight, hanging, id, ideographic, mathematical, overline-position, overline-thickness, panose-1, slope, stemh, stemv, strikethrough-position, strikethrough-thickness, underline-position, underline-thickness, unicode-range, units-per-em, widths, x-height, xml:base, xml:id, xml:lang, xml:space	â€˜ desc, font-face-src, metadata, switch, title
font-face-name	class, id, name, xml:base, xml:id, xml:lang, xml:space	â€˜ desc, metadata, switch, title
font-face-src	class, id, xml:base, xml:id, xml:lang, xml:space	â€˜ desc, font-face-name, font-face-uri, metadata, switch, title
font-face-uri	class, id, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space	â€˜ desc, metadata, switch, title
foreignObject	class, externalResourcesRequired, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, height, id, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, transform, width, x, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space, y	â€"
g	class, externalResourcesRequired, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, id, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, transform, xml:base, xml:id, xml:lang, xml:space	â€" a, animate, animateColor, animateMotion, animateTransform, animation, audio, circle, defs, desc, discard, ellipse, font, font-face, foreignObject, g, handler, image, line, linearGradient, metadata, path, polygon, polyline, prefetch, radialGradient, rect, script, set, solidColor, switch, text, textArea, title, use, video
glyph	arabic-form, class, d, glyph-name, horiz-adv-x, id, lang, unicode, xml:base, xml:id, xml:lang, xml:space	â€˜ desc, metadata, switch, title
handler	class, ev:event, externalResourcesRequired, id, type, xml:base, xml:id, xml:lang, xml:space	â€˜ <text>
hkern	class, g1, g2, id, k, u1, u2, xml:base, xml:id, xml:lang, xml:space	â€˜ desc, metadata, switch, title
image	class, externalResourcesRequired, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, height, id, opacity, preserveAspectRatio, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, transform, type, width, x, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space, y	â€˜ animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title
line	class, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, id, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, transform, x1, x2, xml:base, xml:id, xml:lang, xml:space, y1, y2	â€" animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title
linearGradient	class, id, x1, x2, xml:base, xml:id, xml:lang, xml:space, y1, y2	â€" animate, animateColor, animateMotion, animateTransform, desc, discard, metadata, set, stop, switch, title
metadata	class, id, xml:base, xml:id, xml:lang, xml:space	â€˜ <text>
missing-glyph	class, d, horiz-adv-x, id, xml:base, xml:id, xml:lang, xml:space	â€˜ desc, metadata, switch, title
mpath	class, id, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space	â€˜ desc, metadata, switch, title
path	class, d, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, id, pathLength, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, transform, xml:base, xml:id, xml:lang, xml:space	â€" animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title
polygon	class, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, id, points, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, transform, xml:base, xml:id, xml:lang, xml:space	â€" animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title
polyline	class, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, id, points, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, transform, xml:base, xml:id, xml:lang, xml:space	â€" animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title
prefetch	bandwidth, class, id, mediaCharacterEncoding, mediaContentEncodings, mediaSize, mediaTime, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space	â€˜ desc, metadata, switch, title

radialGradient	class, cx, cy, id, r, xml:base, xml:id, xml:lang, xml:space	âœ”	animate, animateColor, animateMotion, animateTransform, desc, discard, metadata, set, stop, switch, title
rect	class, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, height, id, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, rx, ry, systemLanguage, transform, width, x, xml:base, xml:id, xml:lang, xml:space, y	âœ”	animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title
script	class, externalResourcesRequired, id, type, xml:base, xml:id, xml:lang, xml:space	âœ˜	<text>
set	attributeName, attributeType, begin, class, dur, end, fill, id, max, min, repeatCount, repeatDur, restart, to, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space	âœ˜	desc, handler, metadata, switch, title
solidColor	class, id, xml:base, xml:id, xml:lang, xml:space	âœ”	animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title
stop	class, id, offset, xml:base, xml:id, xml:lang, xml:space	âœ”	animate, animateColor, animateMotion, animateTransform, desc, metadata, set, switch, title
svg	baseProfile, class, contentScriptType, externalResourcesRequired, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, height, id, playbackOrder, preserveAspectRatio, snapshotTime, syncBehavior, syncBehaviorDefault, syncTolerance, syncToleranceDefault, timelineBegin, version, viewBox, width, xml:base, xml:id, xml:lang, xml:space, zoomAndPan	âœ”	a, animate, animateColor, animateMotion, animateTransform, animation, audio, circle, defs, desc, discard, ellipse, font, font-face, foreignObject, g, handler, image, line, linearGradient, metadata, path, polygon, polyline, prefetch, radialGradient, rect, script, set, solidColor, switch, text, textArea, title, use, video
switch	class, externalResourcesRequired, id, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, transform, xml:base, xml:id, xml:lang, xml:space	âœ”	The 'switch' element may contain any element that its parent may contain.
tBreak	class, id, xml:base, xml:id, xml:lang, xml:space	âœ˜	
text	class, editable, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, id, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, rotate, systemLanguage, transform, x, xml:base, xml:id, xml:lang, xml:space, y	âœ”	<text>, a, animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, tBreak, title, tspan
textArea	class, editable, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, height, id, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, transform, width, x, xml:base, xml:id, xml:lang, xml:space, y	âœ”	<text>, a, animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, tBreak, title, tspan
title	class, id, xml:base, xml:id, xml:lang, xml:space	âœ˜	<text>
tspan	class, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, id, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, xml:base, xml:id, xml:lang, xml:space	âœ”	<text>, a, animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, tBreak, title, tspan
use	class, externalResourcesRequired, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, id, overflow, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, systemLanguage, transform, x, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space, y	âœ”	animate, animateColor, animateMotion, animateTransform, desc, handler, metadata, set, switch, title
video	audio-level, begin, class, dur, end, externalResourcesRequired, focusEast, focusNext, focusNorth, focusNorthEast, focusNorthWest, focusPrev, focusSouth, focusSouthEast, focusSouthWest, focusWest, focusable, height, id, overlay, preserveAspectRatio, repeatCount, repeatDur, requiredExtensions, requiredFeatures, requiredFonts, requiredFormats, restart, syncBehavior, syncBehaviorDefault, syncTolerance, syncToleranceDefault, systemLanguage, transform, transformBehavior, type, width, x, xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type, xml:base, xml:id, xml:lang, xml:space, y	âœ˜	animate, animateColor, animateMotion, animateTransform, desc, discard, handler, metadata, set, switch, title

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

K Attributes and Properties Tables

Contents

K Properties Table

Property	ani.	inh.	Value
audio-level	âœ”	âœ˜	'inherit' | <Number.datatype>
color	âœ”	âœ”	'inherit' | <Color.datatype>
color-rendering	âœ”	âœ”	'auto' | 'optimizeSpeed' | 'optimizeQuality' | 'inherit'
display	âœ”	âœ˜	'inline' | 'block' | 'list-item' | 'run-in' | 'compact' | 'marker' | 'table' | 'inline-table' | 'table-row-group' | 'table-header-group' | 'table-footer-group' | 'table-row' | 'table-column-group' | 'table-column' | 'table-cell' | 'table-caption' | 'none' | 'inherit'
display-align	âœ”	âœ”	'auto' | 'before' | 'center' | 'after' | 'inherit'
fill	âœ”	âœ”	'inherit' | <Paint.datatype>

fill-opacity	âœ”	âœ”	'inherit' | <OpacityValue.datatype>
fill-rule	âœ”	âœ”	'inherit' | <ClipFillRule.datatype>
font-family	âœ”	âœ”	'inherit' | <FontFamilyValue.datatype>
font-size	âœ”	âœ”	'inherit' | <FontSizeValue.datatype>
font-style	âœ”	âœ”	'normal' | 'italic' | 'oblique' | 'inherit'
font-weight	âœ”	âœ”	'normal' | 'bold' | 'bolder' | 'lighter' | '100' | '200' | '300' | '400' | '500' | '600' | '700' | '800' | '900' | 'inherit'
image-rendering	âœ”	âœ”	'auto' | 'optimizeSpeed' | 'optimizeQuality' | 'inherit'
line-increment	âœ”	âœ”	'auto' | 'inherit' | <Number.datatype>
pointer-events	âœ”	âœ”	'visiblePainted' | 'visibleFill' | 'visibleStroke' | 'visible' | 'painted' | 'fill' | 'stroke' | 'all' | 'none' | 'inherit'
shape-rendering	âœ”	âœ”	'auto' | 'optimizeSpeed' | 'crispEdges' | 'geometricPrecision' | 'inherit'
solid-color	âœ”	âœ˜	'inherit' | <SVGColor.datatype>
solid-opacity	âœ”	âœ˜	'inherit' | <OpacityValue.datatype>
stop-color	âœ”	âœ˜	'inherit' | <SVGColor.datatype>
stop-opacity	âœ”	âœ˜	'inherit' | <OpacityValue.datatype>
stroke	âœ”	âœ”	'inherit' | <Paint.datatype>
stroke-dasharray	âœ”	âœ”	'inherit' | <StrokeDashArrayValue.datatype>
stroke-dashoffset	âœ”	âœ”	'inherit' | <StrokeDashOffsetValue.datatype>
stroke-linecap	âœ”	âœ”	'butt' | 'round' | 'square' | 'inherit'
stroke-linejoin	âœ”	âœ”	'miter' | 'round' | 'bevel' | 'inherit'
stroke-miterlimit	âœ”	âœ”	'inherit' | <StrokeMiterLimitValue.datatype>
stroke-opacity	âœ”	âœ”	'inherit' | <OpacityValue.datatype>
stroke-width	âœ”	âœ”	'inherit' | <StrokeWidthValue.datatype>
text-anchor	âœ”	âœ”	'start' | 'middle' | 'end' | 'inherit'
text-rendering	âœ”	âœ”	'auto' | 'optimizeSpeed' | 'optimizeLegibility' | 'geometricPrecision' | 'inherit'
vector-effect	âœ”	âœ˜	'default' | 'non-scaling-stroke' | 'inherit'
viewport-fill	âœ”	âœ˜	'inherit' | <Paint.datatype>
viewport-fill-opacity	âœ”	âœ˜	'inherit' | <OpacityValue.datatype>
visibility	âœ”	âœ”	'visible' | 'hidden' | 'inherit'

K Attributes Table

Attribute	ani.	inh.	Value	Parents
accent-height	âœ˜	âœ˜	<Number>	font-face
accumulate	âœ˜	âœ˜	'none' | 'sum'	animate, animateColor, animateMotion, animateTransform
additive	âœ˜	âœ˜	'replace' | 'sum'	animate, animateColor, animateMotion, animateTransform
alphabetic	âœ˜	âœ˜	<Number>	font-face
arabic-form	âœ˜	âœ˜	<text>	glyph
ascent	âœ˜	âœ˜	<Number>	font-face
attributeName	âœ˜	âœ˜	<text>	animate, animateColor, animateTransform, set
attributeType	âœ˜	âœ˜	'XML' | 'CSS' | 'auto'	animate, animateColor, animateTransform, set
bandwidth	âœ˜	âœ˜	<NumberOrPercentage>	prefetch
baseProfile	âœ˜	âœ˜	<Text>	svg
bbox	âœ˜	âœ˜	<text>	font-face
begin	âœ˜	âœ˜	<text>	animate, animateColor, animateMotion, animateTransform, animation, audio, discard, set, video
by	âœ˜	âœ˜	<text>	animate, animateColor, animateMotion, animateTransform
calcMode	âœ˜	âœ˜	'discrete' | 'linear' | 'paced' | 'spline'	animate, animateColor, animateMotion, animateTransform
cap-height	âœ˜	âœ˜	<Number>	font-face
class	âœ˜	âœ˜	<text>	a, animate, animateColor, animateMotion, animateTransform, animation, audio, circle, defs, desc, discard, ellipse, font, font-face, font-face-name, font-face-src, font-face-uri, foreignObject, g, glyph, handler, hkern, image, line, linearGradient, metadata, missing-glyph, mpath, path, polygon, polyline, prefetch, radialGradient, rect, script, set, solidColor, stop, svg, switch, tBreak, text, textArea, title, tspan, use, video
contentScriptType	âœ˜	âœ˜	<ContentType>	svg

cx	âœ”	â˜	<Coordinate>	circle, ellipse, radialGradient
cy	âœ”	â˜	<Coordinate>	circle, ellipse, radialGradient
d	âœ”	â˜	<PathData>	glyph, missing-glyph, path
descent	â˜	â˜	<Number>	font-face
dur	â˜	â˜	<text>	animate, animateColor, animateMotion, animateTransform, animation, audio, set, video
editable	â˜	â˜	<Boolean>	text, textArea
end	â˜	â˜	<text>	animate, animateColor, animateMotion, animateTransform, animation, audio, set, video
ev:event	â˜	â˜	<XSLT-QName>	handler
externalResourcesRequired	â˜	â˜	<Boolean>	a, animation, audio, font, font-face, foreignObject, g, handler, image, script, svg, switch, use, video
fill	â˜	â˜	'remove' | 'freeze'	animate, animateColor, animateMotion, animateTransform, animation, set
focusEast	âœ”	â˜	<Focus>	a, animation, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, svg, text, textArea, tspan, use, video
focusNext	âœ”	â˜	<Focus>	a, animation, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, svg, text, textArea, tspan, use, video
focusNorth	âœ”	â˜	<Focus>	a, animation, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, svg, text, textArea, tspan, use, video
focusNorthEast	âœ”	â˜	<Focus>	a, animation, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, svg, text, textArea, tspan, use, video
focusNorthWest	âœ”	â˜	<Focus>	a, animation, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, svg, text, textArea, tspan, use, video
focusPrev	âœ”	â˜	<Focus>	a, animation, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, svg, text, textArea, tspan, use, video
focusSouth	âœ”	â˜	<Focus>	a, animation, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, svg, text, textArea, tspan, use, video
focusSouthEast	âœ”	â˜	<Focus>	a, animation, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, svg, text, textArea, tspan, use, video
focusSouthWest	âœ”	â˜	<Focus>	a, animation, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, svg, text, textArea, tspan, use, video
focusWest	âœ”	â˜	<Focus>	a, animation, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, svg, text, textArea, tspan, use, video
focusable	âœ”	â˜	'auto' | <Boolean.datatype>	a, animation, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, svg, text, textArea, tspan, use, video
font-family	â˜	â˜	<text>	font-face
font-size	â˜	â˜	<text>	font-face
font-stretch	â˜	â˜	<text>	font-face
font-style	â˜	â˜	<text>	font-face
font-variant	â˜	â˜	<text>	font-face
font-weight	â˜	â˜	<text>	font-face
from	â˜	â˜	<text>	animate, animateColor, animateMotion, animateTransform
g1	â˜	â˜	<text>	hkern
g2	â˜	â˜	<text>	hkern
glyph-name	â˜	â˜	<text>	glyph
gradientUnits	âœ”	â˜	'userSpaceOnUse' | 'objectBoundingBox'	
hanging	â˜	â˜	<Number>	font-face
height	âœ”	â˜	<Length>	animation, foreignObject, image, rect, svg, textArea, video
horiz-adv-x	â˜	â˜	<Number>	font, glyph, missing-glyph
horiz-origin-x	â˜	â˜	<Number>	font
id	â˜	â˜	<ID>	a, animate, animateColor, animateMotion, animateTransform, animation, audio, circle, defs, desc, discard, ellipse, font, font-face, font-face-name, font-face-src, font-face-uri, foreignObject, g, glyph, handler, hkern, image, line, linearGradient, metadata, missing-glyph, mpath, path, polygon, polyline, prefetch, radialGradient, rect, script, set, solidColor, stop, svg, switch, tBreak, text, textArea, title, tspan, use, video
ideographic	â˜	â˜	<Number>	font-face
k	â˜	â˜	<Number>	hkern
keyPoints	â˜	â˜	<text>	animateMotion
keySplines	â˜	â˜	<text>	animate, animateColor, animateMotion, animateTransform
keyTimes	â˜	â˜	<text>	animate, animateColor, animateMotion, animateTransform
lang	â˜	â˜	<LanguageCodes>	glyph

mathematical	âœ˜	âœ˜	<Number>	font-face
max	âœ˜	âœ˜	<text>	animate, animateColor, animateMotion, animateTransform, set
mediaCharacterEncoding	âœ˜	âœ˜	<text>	prefetch
mediaContentEncodings	âœ˜	âœ˜	<text>	prefetch
mediaSize	âœ˜	âœ˜	<NumberOrPercentage>	prefetch
mediaTime	âœ˜	âœ˜	<text>	prefetch
min	âœ˜	âœ˜	<text>	animate, animateColor, animateMotion, animateTransform, set
name	âœ˜	âœ˜	<text>	font-face-name
offset	âœ”	âœ˜	<GradientOffset>	stop
opacity	âœ”	âœ˜	'inherit' | <OpacityValue.datatype>	image
origin	âœ˜	âœ˜	<text>	animateMotion
overflow	âœ˜	âœ˜	'visible'	animation, use
overlay	âœ˜	âœ˜	'none' | 'top'	video
overline-position	âœ˜	âœ˜	<Number>	font-face
overline-thickness	âœ˜	âœ˜	<Number>	font-face
panose-1	âœ˜	âœ˜	<text>	font-face
path	âœ˜	âœ˜	<text>	animateMotion
pathLength	âœ”	âœ˜	<Number>	path
playbackOrder	âœ˜	âœ˜	'all' | 'forwardOnly'	svg
points	âœ”	âœ˜	<Points>	polygon, polyline
preserveAspectRatio	âœ”	âœ˜	<PreserveAspectRatioSpec>	animation, image, svg, video
r	âœ”	âœ˜	<Length>	circle, radialGradient
repeatCount	âœ˜	âœ˜	<text>	animate, animateColor, animateMotion, animateTransform, animation, audio, set, video
repeatDur	âœ˜	âœ˜	<text>	animate, animateColor, animateMotion, animateTransform, animation, audio, set, video
requiredExtensions	âœ˜	âœ˜	<ExtensionList>	a, animation, audio, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, switch, text, textArea, tspan, use, video
requiredFeatures	âœ˜	âœ˜	<FeatureList>	a, animation, audio, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, switch, text, textArea, tspan, use, video
requiredFonts	âœ˜	âœ˜	<FontList>	a, animation, audio, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, switch, text, textArea, tspan, use, video
requiredFormats	âœ˜	âœ˜	<FormatList>	a, animation, audio, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, switch, text, textArea, tspan, use, video
restart	âœ˜	âœ˜	'always' | 'never' | 'whenNotActive'	animate, animateColor, animateMotion, animateTransform, animation, audio, set, video
rotate	âœ˜	âœ˜	<text>	animateMotion
rotate	âœ”	âœ˜	<Numbers>	text
rx	âœ”	âœ˜	<Length>	ellipse, rect
ry	âœ”	âœ˜	<Length>	ellipse, rect
slope	âœ˜	âœ˜	<Number>	font-face
snapshotTime	âœ˜	âœ˜	<text>	svg
stemh	âœ˜	âœ˜	<Number>	font-face
stemv	âœ˜	âœ˜	<Number>	font-face
strikethrough-position	âœ˜	âœ˜	<Number>	font-face
strikethrough-thickness	âœ˜	âœ˜	<Number>	font-face
syncBehavior	âœ˜	âœ˜	'canSlip' | 'locked' | 'independent' | 'default'	animation, audio, svg, video
syncBehaviorDefault	âœ˜	âœ˜	'canSlip' | 'locked' | 'independent' | 'inherit'	animation, audio, svg, video
syncTolerance	âœ˜	âœ˜	'Clock-value' | 'default'	animation, audio, svg, video
syncToleranceDefault	âœ˜	âœ˜	'Clock-value' | 'inherit'	animation, audio, svg, video
systemLanguage	âœ˜	âœ˜	<LanguageCodes>	a, animation, audio, circle, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, switch, text, textArea, tspan, use, video
target	âœ”	âœ˜	<LinkTarget>	a
timelineBegin	âœ˜	âœ˜	'onLoad' | 'onStart'	svg
to	âœ˜	âœ˜	<text>	animate, animateColor, animateMotion, animateTransform, set

transform	â€"	â€˜	<TransformList>	a, circle, defs, ellipse, foreignObject, g, image, line, path, polygon, polyline, rect, switch, text, textArea, use, video
transformBehavior	â€˜	â€˜	'geometric' | 'pinned'	video
type	â€˜	â€˜	<ContentType>	audio, handler, image, script, video
type	â€˜	â€˜	'translate' | 'scale' | 'rotate' | 'skewX' | 'skewY'	animateMotion, animateTransform
u1	â€˜	â€˜	<text>	hkern
u2	â€˜	â€˜	<text>	hkern
underline-position	â€˜	â€˜	<Number>	font-face
underline-thickness	â€˜	â€˜	<Number>	font-face
unicode	â€˜	â€˜	<text>	glyph
unicode-range	â€˜	â€˜	<text>	font-face
units-per-em	â€˜	â€˜	<Number>	font-face
values	â€˜	â€˜	<text>	animate, animateColor, animateMotion, animateTransform
version	â€˜	â€˜	'1.0' | '1.1' | '1.2'	svg
viewBox	â€"	â€˜	<ViewBoxSpec>	svg
width	â€"	â€˜	<Length>	animation, foreignObject, image, rect, svg, textArea, video
widths	â€˜	â€˜	<text>	font-face
x	â€"	â€˜	<Coordinate>	animation, foreignObject, image, rect, textArea, use, video
x-height	â€˜	â€˜	<Number>	font-face
x1	â€"	â€˜	<Coordinate>	line, linearGradient
x2	â€"	â€˜	<Coordinate>	line, linearGradient
x	â€"	â€˜	<Coordinates>	text
xlink:actuate	â€˜	â€˜	'onLoad'	animate, animateColor, animateMotion, animateTransform, animation, audio, discard, font-face-uri, foreignObject, handler, image, mpath, prefetch, script, set, use, video
xlink:actuate	â€˜	â€˜	'onRequest'	a
xlink:arcrole	â€˜	â€˜	<IRI>	a, animate, animateColor, animateMotion, animateTransform, animation, audio, discard, font-face-uri, foreignObject, handler, image, mpath, prefetch, script, set, use, video
xlink:href	â€"	â€˜	<IRI>	a, animate, animateColor, animateMotion, animateTransform, animation, audio, discard, font-face-uri, foreignObject, handler, image, mpath, prefetch, script, set, use, video
xlink:role	â€˜	â€˜	<IRI>	a, animate, animateColor, animateMotion, animateTransform, animation, audio, discard, font-face-uri, foreignObject, handler, image, mpath, prefetch, script, set, use, video
xlink:show	â€˜	â€˜	'other'	animate, animateColor, animateMotion, animateTransform, discard, font-face-uri, handler, mpath, prefetch, script, set
xlink:show	â€˜	â€˜	'embed'	animation, audio, foreignObject, image, use, video
xlink:show	â€˜	â€˜	'new' | 'replace'	a
xlink:title	â€˜	â€˜	<text>	a, animate, animateColor, animateMotion, animateTransform, animation, audio, discard, font-face-uri, foreignObject, handler, image, mpath, prefetch, script, set, use, video
xlink:type	â€"	â€˜	'simple'	a, animate, animateColor, animateMotion, animateTransform, animation, audio, discard, font-face-uri, foreignObject, handler, image, mpath, prefetch, script, set, use, video
xml:base	â€˜	â€˜	<IRI>	a, animate, animateColor, animateMotion, animateTransform, animation, audio, circle, defs, desc, discard, ellipse, font, font-face, font-face-name, font-face-src, font-face-uri, foreignObject, g, glyph, handler, hkern, image, line, linearGradient, metadata, missing-glyph, mpath, path, polygon, polyline, prefetch, radialGradient, rect, script, set, solidColor, stop, svg, switch, tBreak, text, textArea, title, tspan, use, video
xml:id	â€˜	â€˜	<ID>	a, animate, animateColor, animateMotion, animateTransform, animation, audio, circle, defs, desc, discard, ellipse, font, font-face, font-face-name, font-face-src, font-face-uri, foreignObject, g, glyph, handler, hkern, image, line, linearGradient, metadata, missing-glyph, mpath, path, polygon, polyline, prefetch, radialGradient, rect, script, set, solidColor, stop, svg, switch, tBreak, text, textArea, title, tspan, use, video
xml:lang	â€˜	â€˜	<LanguageCode.datatype>	a, animate, animateColor, animateMotion, animateTransform, animation, audio, circle, defs, desc, discard, ellipse, font, font-face, font-face-name, font-face-src, font-face-uri, foreignObject, g, glyph, handler, hkern, image, line, linearGradient, metadata, missing-glyph, mpath, path, polygon, polyline, prefetch, radialGradient, rect, script, set, solidColor, stop, svg, switch, tBreak, text, textArea, title, tspan, use, video
xml:space	â€˜	â€˜	'default' | 'preserve'	a, animate, animateColor, animateMotion, animateTransform, animation, audio, circle, defs, desc, discard, ellipse, font, font-face, font-face-name, font-face-src, font-face-uri, foreignObject, g, glyph, hkern, image, line, linearGradient, metadata, missing-glyph, mpath, path, polygon, polyline, prefetch, radialGradient, rect, set, solidColor, stop, svg, switch, tBreak, text, textArea, title, tspan, use, video
xml:space	â€˜	â€˜	'preserve'	handler, script
y	â€"	â€˜	<Coordinate>	animation, foreignObject, image, rect, textArea, use, video
y1	â€"	â€˜	<Coordinate>	line, linearGradient

y2	âœ” âœ˜	<Coordinate>	line, linearGradient
y	âœ” âœ˜	<Coordinates>	text
zoomAndPan	âœ˜ âœ˜	'disable'	'magnify'	svg

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

L Media Type registration for image/svg+xml

Contents

This appendix is normative.

L.1 Introduction

This appendix registers a new MIME media type, "image/svg+xml" in conformance with RegMedia and W3CRegMedia.

L.2 Registration of Media Type image/svg+xml

MIME media type name:

image

MIME subtype name:

svg+xml

Required parameters:

None.

Optional parameters:

None

The encoding of an SVG document is determined by the XML encoding declaration. This has identical semantics to the application/xml media type in the case where the charset parameter is omitted, as specified in RFC3023 sections 8.9, 8.10 and 8.11.

Encoding considerations:

Same as for application/xml. See RFC3023 , section 3.2.

Restrictions on usage:

None

Security considerations:

As with other XML types and as noted in RFC3023 section 10, repeated expansion of maliciously constructed XML entities can be used to consume large amounts of memory, which may cause XML processors in constrained environments to fail.

SVG documents may be transmitted in compressed form using gzip compression. For systems which employ MIME-like mechanisms, such as HTTP, this is indicated by the Content-Transfer-Encoding header; for systems which do not, such as direct filesystem access, this is indicated by the filename extension and by the Macintosh File Type Codes. In addition, gzip compressed content is readily recognised by the initial byte sequence as described in RFC1952 section 2.3.1.

Several SVG elements may cause arbitrary URIs to be referenced. In this case, the security issues of RFC2396, section 7, should be considered.

In common with HTML, SVG documents may reference external media such as images, audio, video, style sheets, and scripting languages. Scripting languages are executable content. In this case, the security considerations in the Media Type registrations for those formats apply.

In addition, because of the extensibility features for SVG and of XML in general, it is possible that "image/svg+xml" may describe content that has security implications beyond those described here. However, if the processor follows only the normative semantics of this specification, this content will be outside the SVG namespace and will be ignored. Only in the case where the processor recognizes and processes the additional content, or where further processing of that content is dispatched to other processors, would security issues potentially arise. And in that case, they would fall outside the domain of this registration document.

Interoperability considerations:

This specification describes processing semantics that dictate behavior that must be followed when dealing with, among other things, unrecognized elements and attributes, both in the SVG namespace and in other namespaces.

Because SVG is extensible, conformant "image/svg+xml" processors must expect that content received is well-formed XML, but it cannot be guaranteed that the content is valid to a particular DTD or Schema or that the processor will recognize all of the elements and attributes in the document.

SVG has a published Test Suite and associated implementation report showing which implementations passed which tests at the time of the report. This information is periodically updated as new tests are added or as implementations improve.

Published specification:

This media type registration is extracted from Appendix G of the SVG 1.2 specification.

Additional information:

Person & email address to contact for further information:

Dean Jackson, (dean@w3.org).

Intended usage:

COMMON

Author/Change controller:

The SVG specification is a work product of the World Wide Web Consortium's SVG Working Group. The W3C has change control over these specifications.

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

M RelaxNG Schema for SVG Tiny 1.2

The schema for SVG Tiny 1.2 is written in RelaxNG [RelaxNG], a namespace-aware schema language that uses the datatypes from XML Schema Part 2 [Schema2]. This allows namespaces and modularity to be much more naturally expressed than using DTD syntax. The RelaxNG schema for SVG Tiny 1.2 may be imported by other RealaxNG schemas, or combined with other schemas in other languages into a multi-namespace, multi-grammar schema using NVDL [NVDL].

Unlike a DTD, the schema used for validation is not hardcoded into the document instance. There is no equivalent to the DOCTYPE declaration. Simply point your editor or other validation tool to the IRI of the schema (or your local cached copy, as you prefer).

The schema is available in an unchanging *dated* version, considered to be normative at the time of publication, and the *latest* version (possibly incorporating bug fixes made since publication) which will evolve over time. At the date of publication, both versions are identical.

The driver schema is available as a latest version and also the dated version for this Working Draft.. A zip file of all the schema modules is available for download in latest and dated versions also.

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Next | Elements | Attributes

N References

Contents

N.1 Normative references

[ATAG]
"Authoring Tool Accessibility Guidelines 1.0", J. Treviranus, J. Richards, I. Jacobs, C. McCathieNevile, editors, 3 February 2000.

Available at http://www.w3.org/TR/ATAG10/

[COLORIMETRY]
"Colorimetry, Second Edition", CIE Publication 15.2-1986, ISBN 3-900-734-00-3.

Available at http://www.cie.co.at/publ/abst/15-2-86.html.

[CHARMOD]
"Character Model for the World Wide Web 1.0: Fundamentals", M. DÃ¼rst, F. Yergeau, R. Ishida, M. Wolf, T. Texin, editors, 15 February 2005.

Available at http://www.w3.org/TR/charmod/

[CHARMOD-RI]
"Character Model for the World Wide Web 1.0: Resource Identifiers", M. DÃ¼rst, F. Yergeau, R. Ishida, M. Wolf, T. Texin, editors, 22 November 2004

Available at http://www.w3.org/TR/2004/CR-charmod-resid-20041122/

[CSS2]
"Cascading Style Sheets, level 2", B. Bos, H. W. Lie, C. Lilley, I. Jacobs, 12 May 1998.

Available at http://www.w3.org/TR/REC-CSS2/.

[DOM3]
"Document Object Model (DOM) Level 3 Core Specification", A. Le Hors, P. Le HÃ©garet, L. Wood, G. Nicol, J. Robie, M. Champion, S. Byrne, editors, 07 April 2004

Available at http://www.w3.org/TR/DOM-Level-3-Core/

[DOM3Events]
"Document Object Model (DOM) Level 3 Events Specification", P. Le HÃ©garet, T. Pixley, editors, 07 November 2003.

Available at http://www.w3.org/TR/DOM-Level-3-Events

[ECMAScript]
ECMA (European Computer Manufacturers Association) "ECMAScript Language Specification", 3rd Edition.

Available at http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM

[ISO8601]
"Data elements and interchange formats - Information interchange - Representation of dates and times", International Organization for Standardization, 1998.

[JPEG]
ISO/IEC 10918. Available from the International Organization for Standardization (ISO).

[JFIF]
JPEG File Interchange Format, version 1.02. September 1, 1992. Available at http://www.w3.org/Graphics/JPEG/jfif3.pdf

[JSR226]
"JSR 226: Scalable 2D Vector Graphics API for J2MEâ„¢", S. Chitturi, Specification Lead.

Available at http://www.jcp.org/en/jsr/detail?id=226

[NVDL]
"Document Schema Definition Languages (DSDL) â€" Part 4: Namespace-based Validation Dispatching Language â€" NVDL. ISO/IEC FCD 19757-4 "

Available at http://www.asahi-net.or.jp/~eb2m-mrt/dsdl/

[OpenGIS Coordinate Systems]
"Recommended Definition Data for Coordinate Reference Systems and Coordinate Transformations Version 1.0.1." OpenGIS Recommendation

Available at http://www.opengis.org/techno/specs/01-014r3.pdf

[PNG]
"Portable Network Graphics (PNG) Specification (Second Edition)"

" Information technology â€" Computer graphics and image processing â€" Portable Network Graphics (PNG): Functional specification. ISO/IEC 15948:2003 (E)"

David Duce editor, 10 November 2003.

Available at http://www.w3.org/TR/PNG/.

[QAF-SPEC]
QA Framework: Specification Guidelines , Karl Dubost, Lynne Rosenthal, Dominique HazaÃ«l-Massieux, Lofton Henderson, W3C Working Draft, http://www.w3.org/TR/2004/WD-qaframe-spec-20041122/ Latest version available at http://www.w3.org/TR/qaframe-spec/ .

[RelaxNG]
"Document Schema Definition Languages (DSDL) â€" Part 2: Regular grammar- based validation â€" RELAX NG. ISO/IEC FDIS 19757-2:2002(E)", J. Clark, MURATA M., editors, 12 December 2002.

Available at http://www.y12.doe.gov/sgml/sc34/document/0362_files/relaxng-is.pdf

[RFC1951]
"DEFLATE Compressed Data Format Specification version 1.3", P. Deutsch, May 1996.

Available at http://www.ietf.org/rfc/rfc1952.txt.

[RFC1952]
"GZIP file format specification version 4.3", P. Deutsch, May 1996.

Available at http://www.ietf.org/rfc/rfc1952.txt.

[RFC2045]
"Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", N. Freed and N. Borenstein, November 1996. Note that this RFC obsoletes RFC1521, RFC1522, and RFC1590.

Available at http://www.ietf.org/rfc/rfc2045.txt.

[RFC2046]
"Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", N. Freed and N. Borenstein, November 1996. Note that this RFC obsoletes RFC1521, RFC1522, and RFC1590.

Available at http://www. ietf.org/rfc/rfc2046.txt.

[RFC2119]
"Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997.

Available at http://www.ietf.org/rfc/rfc2119.txt.

[RFC2141]
"URN Syntax", R. Moats, May 1997.

Available at http://www.ietf.org/rfc/rfc2141.txt.

[RFC2279]
"UTF-8, a transformation format of ISO 10646", F. Yergeau, January 1998. Note that this RDF obsoletes RFC 2044.

Available at http://www.ietf.org/rfc/rfc2279.txt.

[RFC2318]
"The text/css Media Type", H. Lie, B. Bos, C. Lilley, March 1998.

Available at http://www.ietf.org/rfc/rfc2318.txt.

[RFC2397]
'The "data" URL scheme', L. Masinter, August 1998.

Available at http://www.ietf.org/rfc/rfc2397.txt.

[RFC2616]
"Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding, J. Gettys, J. Mogul, H. Frystyk Nielsen, L. Masinter, P. Leach and T. Berners-Lee, June 1999. This RFC obsoletes RFC 2068.

Available at http://www.ietf.org/rfc/rfc2616.txt.

[RFC2732]
"RFC 2732: Format for Literal IPv6 Addresses in URL's", Internet Engineering Task Force, R. Hinden, B. Carpenter, L. Masinter, December 1999.

Available at http://www.ietf.org/rfc/rfc2732.txt.

[RFC3023]
"XML Media Types", M. Murata, S. St.Laurent, D. Kohn, January 2001.

Available at http://www.ietf.org/rfc/rfc3023.txt. Note that a revision to RFC3023 is in preparation.

[RFC3066]
"Tags for the Identification of Languages", H. Alvestrand, January 2001.

Available at http://www.ietf.org/rfc/rfc3066.txt.

[RFC3986]
'Uniform Resource Identifiers (URI): Generic Syntax', T. Berners-Lee, R. Fielding, L. Masinter, January 2005. Note that RFC 3986 updates [RFC 2396], [RFC1738] and [RFC1808].

Available at http://www.ietf.org/rfc/rfc3986.txt.

[RFC3987]
"Internationalized Resource Identifiers (IRIs)". M. Duerst, M. Suignard, January 2005.

Available at http://www.ietf.org/rfc/rfc3987.txt

[SMIL20]
Synchronized Multimedia Integration Language (SMIL 2.0) - [Second Edition]. J. Ayars, D. Bulterman *et. al.*, 07 January 2005.

Available at http://www.w3.org/TR/2005/REC-SMIL2-20050107/

[SRGB]
IEC 61966-2-1 (1999-10) - "Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour

space - sRGB", ISBN: 2-8318-4989-6 Â ICS codes: 33.160.60, 37.080 Â TC 100 Â 51 pp.

Available at: http://domino.iec.ch/webstore/webstore.nsf/artnum/025408.

[UNICODE]
The Unicode Consortium. "The Unicode Standard", Version 4.0.0 or later. The Unicode Standard, Version 4.0.0 is defined by "The Unicode Standard, Version 4.0" (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1).

Refer to http://www.unicode.org/unicode/standard/versions/.

[WCAG]
"Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, I. Jacobs, editors, 5-May-1999.

Available at:

http://www.w3.org/TR/WAI-WEBCONTENT/.

[XLINK]
"XML Linking Language (XLink)", S. DeRose, E. Maler, D. Orchard, editors, 27 June 2001.

Available at http://www.w3.org/TR/xlink/

[XML10]
"Extensible Markup Language (XML) 1.0 (Third Edition)", T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, editors, 04 February 2004.

Available at http://www.w3.org/TR/REC-xml.

[XML11]
"Extensible Markup Language (XML) 1.1", T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, F. Yergeau, J. Cowan, editors, 15 April 2004.

Available at http://www.w3.org/TR/xml11/

[XMLID]
"xml:id Version 1.0", J. Marsh, D. Veillard, N. Walsh, editors, 08 February 2005.

Available at http://www.w3.org/TR/2005/CR-xml-id-20050208/

[XML-BASE]
"XML Base", J. Marsh , editor, 20 December 2000.

Available at http://www.w3.org/TR/xmlbase/.

[XML-NS]
"Namespaces in XML 1.1", T. Bray, D. Hollander, A. Layman, R. Tobin, editors, 4 February 2004.

Available at http://www.w3.org/TR/xml-names11/.

[XPTRFW]
"XPointer Framework", P. Grosso, E. Maler, J. Marsh, N. Walsh, editors, 25 March 2003

Available at http://www.w3.org/TR/xptr-framework/

N.2 Informative references

[DCORE]
The Dublin Core. For more information, refer to http://purl.org/DC.

[FOLEY-VANDAM]
"Computer Graphics : Principles and Practice, Second Edition", James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Richard L. Phillips, Addison-Wesley, pp. 488-491.

[HTML4]
"HTML 4.01 Specification", D. Raggett, A. Le Hors, I. Jacobs, 24 December 1999.

Available at http://www.w3.org/TR/html401/. The Recommendation defines three document type definitions: Strict, Transitional, and Frameset, all reachable from the Recommendation.

[MATHML]
"Mathematical Markup Language (MathML) Version 2.0", D. Carlisle, P. Ion, R. Miner, N. Poppelier, editors, 21 February 2001.

Available at http://www.w3.org/TR/MathML2/.

[MIMETYPES]
List of registered Internet Media Types (previously called MIME types). Download a list of registered Internet Media Types from http://www.iana.org/assignments/media-types/.

[OPENTYPE]
See http://www.microsoft.com/OpenType/OTSpec/.

[RDF10]
"Resource Description Framework (RDF) Model and Syntax Specification", O. Lassila, R. Swick, eds., 22 February 1999. This document is http://www.w3.org/TR/REC-rdf-syntax/.

[Schema2]
"XML Schema Part 2: Datatypes Second Edition". P. Biron, A. Malhotra, editors, 28 October 2004.

Available at http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

[SVG11]
"Scalable Vector Graphics (SVG) 1.1", J. Ferraiolo, è—¤æ²¢ æ·³ (FUJISAWA Jun), D. Jackson, editors, 14 January 2003..

Available at: http://www.w3.org/TR/SVG11/

[SVG-ACCESS]
"Accessibility Features of SVG", C. McCathieNevile, M. Koivunen, 7 August 2000.

Available at http://www.w3.org/TR/SVG-access/.

[UAAG]
"User Agent Accessibility Guidelines 1.0", I. Jacobs, J. Gunderson, E. Hansen, editors, 17 December 2002.

Available at http://www.w3.org/TR/UAAG10/

[WAI]
Home page for Web Accessibility Initiative:

http://www.w3.org/WAI/.

[WebCGM]
"WebCGM 1.0 Second Release", D. Cruikshank et. al, editors, 17 December 2001.

Available at http://www.w3.org/TR/REC-WebCGM/

[XHTML]
"XHTML(tm) 1.0: The Extensible HyperText Markup Language", 26 January 2000,

Available at http://www.w3.org/TR/xhtml1/.

[XHTMLplusMathMLplusSVG]
"An XHTML + MathML + SVG Profile", M. Ishikawa editor, 9 August 2002.

Available at: http://www.w3.org/TR/2002/WD-XHTMLplusMathMLplusSVG-20020809/

[XSL]
"Extensible Stylesheet Language (XSL) Version 1.0", S. Adler, A. Berglund, J. Caruso, S. Deach, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman, S. Zilles, editors, 12 October 2001.

Available at http://www.w3.org/TR/xsl/

[XSLT]
"XSL Transformations (XSLT) Version 1.0", J. Clark, editor, 16 November 1999.

Available at http://www.w3.org/TR/xslt

SVG Tiny 1.2 - 20050413 | Top | Contents | Previous | Elements | Attributes

O Change History

Contents

Changes relative to the previous public Working Draft of SVG Tiny 1.2 are described below. Many of these changes were made in response to LastCall feedback.

O General Changes

- Attributes with boolean values have been changed to use enumerations, to allow extensibility.

O Coordinate Systems, Transformations and Units (coords.html)

- Elements that establish new viewports are now: svg, animation, image, video. Clarified that foreignObject does not establish a viewport.
- The intrinsic aspect ratio of SVG content has been defined.
- No support for % on ObjectBoundingBoxes

O Linking (linking.html)

- ID references use XPointer Framework Shorthand Pointers.
- The specification now uses IRIs rather than "URIs or a string which may be converted to a URI as specified".

O Animation (animate.html)

- The behaviour of 'paced' animation has been clarified.
- The behaviour of animateTransform has been clarified.
- 'wallclock' remove as it was complex, rarely used, and hard to implement correctly.

O Document Structure (struct.html)

- The page and pageSet elements have been remvoed from SVG Tiny 1.2.
- Added discard element and playbackOrder attribute.
- Removed streamedContents attribute.
- The use element can now reference external content with some restrictions.
- The image element can no longer reference SVG content. The animation element should be used instead.
- The opacity property has been added to the image element.
- The switch element has increased availability.
- requiredFormats, requiredFonts, requiredFeatures and requiredExtensions are now specifically allowed in SVG Tiny.
- The overflow attribute is now available on use. The only valid value is 'visible'.

O Text (text.html)

- The textArea and associated elements replace the previous Flowing Text features.
- 'editable' is now allowed on text elements with children, but the children are flattened when editing occurs

O MultiMedia (multimedia.html)

- The animation element has been added to the specification.(external use/images)
- The overlay attribute was added to the video element.
- The transformBehaviour attribute was added to the video element.
- The audio-level property has been extended to container elements as well as media elements.
- The addition of run-time synchronisation attributes to audio, video and animation.

O Interactivity (interact.html)

- The focusNext and focusPrev attributes have been added in preference to navIndex.
- 8-way navigation is now allowed through the focusNorth, focusNorthEast, etc... attributes.
- The capture phase is no longer supported in SVG Tiny 1.2.

O Painting: Filling, Stroking, Colors and Paint Servers (painting.html)

- viewport-fill and viewport-fill-opacity replace background-fill and background-opacity.

O Scripting (script.html)

- The application/ecmascript Internet Media Type for ecmascript is now specified in the scripting chapter. Use of the older, non-registered text/ecmascript Internet Media Type is deprecated.
- The type attribute (on script and handler) is now optional and defaults to value of contentScriptType.

O uDOM (svgudom.html)

- A focus API has been added for programatic modification of focus.
- fonts and animations cannot be modified by scripting once inserted.
- EventListenerInitializer2 inteface was added to the uDOM.
- The SVGElementInstance interface was added.
- '#text' is used to access the text content of a test node. The textContent attribute on the Node interface is another way to access this information.
- The ownerDocument attribute was added to the Node interface.
- The textContent attribute was added to the Node interface.
- The SVGVisualMediaElement interface was added.
- The ElementTimeControl interface was enhanced.
- GetAttributeNS was introduced and applies to any attribute on a non-svg element, any namespaced attribute on a non-svg element, and all the attributes in the trait table.
- createElementNS can now create all elements in the SVG namespace.
- getAttribute/getAttributeNS are now available in the uDOM, but have restrictions when used with elements from the SVG namespace.
- setAttributeNS is also available in SVG 1.2 Tiny.
- removeChild now removes elements with and without id's (earlier only without).
- Trait table has been extended.

O Implementation Requirements (implnote.html)

- The error handling defined in the specification has been modified better support extensibility and versioning.
