Scalable 2D Vector Graphics API

for J2ME™

Version 1.0

JSR-226 Expert Group
jsr-226-comments@jcp.org

Java Community Process
RESEARCH AND EVALUATION LICENSE

NOKIA CORPORATION IS WILLING TO LICENSE THIS SPECIFICATION TO YOU ONLY UPON THE TERMS CONTAINED IN THIS LICENSE (“LICENSE”). PLEASE READ THE TERMS AND CONDITIONS OF THIS LICENSE CAREFULLY. BY ACCESSING OR USING THE SPECIFICATION YOU WILL BE BOUND BY THE TERMS OF THIS LICENSE.

Version: 1.0
Status: Final Specification
Specification Lead: Nokia Corporation
Release: 2004-12-01

Copyright 2003-2004 Nokia Corporation. All rights reserved.

1. NOTICE; LIMITED LICENSE GRANTS
1.1 The Specification Lead hereby grants You a non-exclusive, non-transferable, worldwide, royalty-free, fully paid-up, limited license (without the right to sublicense) solely under intellectual property rights licensable by the Specification Lead to analyze and to use the Specification for research, evaluation, optimization and development purposes. In addition You may make a reasonable number of verbatim copies of this Specification in its entirety for Your private or internal use, as applicable, in accordance with the terms and conditions of this License.

1.2 No rights are granted under this License for internal deployment, the creation and/or distribution of implementations of the Specification for direct or indirect (including strategic) gain or advantage, the modification of the Specification (other than to the extent of Your fair use rights) or the distribution of the Specification or making the Specification available for 3rd parties.

1.3 Except as expressly set forth in this license, You acquire no right, title or interest in or to Specification or any other intellectual property licensable by the Specification Lead and no other rights are granted by implication, estoppel or otherwise. The Specification may only be used in accordance with the license terms set forth herein. This License will terminate immediately without notice from Specification Lead if You fail to comply with any provision of this License.

2 TRADEMARKS
2.1 Nokia is a registered trademark of Nokia Corporation. Nokia Corporation ’s product names are either trademarks or registered trademarks of Nokia Corporation. Your access to this Specification should not be construed as granting, by implication, estoppel or otherwise, any license or right to use any marks appearing in the Specification without the prior written consent of Nokia Corporation or Nokia’s licensors. No right, title, or interest in or to any trademarks, service marks, or trade names of any third parties, is granted hereunder.

2.2 You shall not be allowed to remove any of the copyright statements or disclaimers or other proprietary notices contained in the Specification and You are obliged to include the copyright statement and the disclaimers, if any, in any copies of the Specification You make.
3. DISCLAIMER OF WARRANTIES
3.1 SUBJECT TO ANY STATUTORY WARRANTIES OR CONDITIONS WHICH CAN NOT BE EXCLUDED, THE SPECIFICATION IS PROVIDED “AS IS” WITHOUT WARRANTY OR CONDITION OF ANY KIND EITHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, AND STATUTORY ARE HEREBY DISCLAIMED. THE ENTIRE RISK ARISING OUT OF OR RELATING TO THE USE OR PERFORMANCE OF THE SPECIFICATION REMAINS WITH YOU.

3.2 THE SPECIFICATION MAY INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION THEREIN; THESE CHANGES WILL BE INCORPORATED INTO NEW VERSIONS OF THE SPECIFICATION, IF ANY. SPECIFICATION LEAD MAY MAKE IMPROVEMENTS AND/OR CHANGES TO THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THE SPECIFICATION AT ANY TIME. Any use of such changes in the Specification will be governed by the then-current license for the applicable version of the Specification.

4. LIMITATION OF LIABILITY
4.1 TO THE FULLEST EXTENT PERMITTED BY LAW, IN NO EVENT WILL THE SPECIFICATION LEAD OR ITS SUPPLIERS BE LIABLE FOR ANY LOST PROFITS, LOST SAVINGS, LOST REVENUE, LOST DATA, PROCUREMENT OF SUBSTITUTE GOODS, OR FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, EVEN IF THE SPECIFICATION LEAD OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES OR DAMAGES. IN ADDITION THE SPECIFICATION LEAD AND ITS SUPPLIERS WILL NOT BE LIABLE FOR ANY DAMAGES CLAIMED BY YOU BASED ON ANY THIRD PARTY CLAIM.

4.2 Some jurisdictions do not allow the exclusion of implied warranties, or the limitation for consequential damages, so Section 4.1 may not apply to You in whole, but in such case Section 4.1 will apply to You to the maximum extent permitted by applicable law.

5. EXPORT CONTROL
5.1 You shall follow all export control laws and regulations relating to Specification.

6. RESTRICTED RIGHTS LEGEND
6.1 Note to U.S. Government Users. The Specification is a “Commercial Items”, as that term is defined at 48 C.F.R. 2. 101, consisting of “Commercial Computer Software” and “Commercial Computer Software Documentation”, as such terms are used in 48 C.F.R. 12.212 or 48 C.F.R. 227.7202, as applicable. Consistent with 48 C.F.R. 12.212 or 48 C.F.R. 227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software Documentation are being licensed to U.S. Government end users a) only as Commercial Items and b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States.
W3C SOFTWARE LICENSE NOTICE
Parts of this Specification are derived from the following W3C Specifications:

DOM Level 3 Core (http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/idl-definitions.html), licensed under terms as set forth in W3C Software Notice and License, dated December 31 2002.

W3C SVG Tiny 1.2 uDOM (http://www.w3.org/TR/SVG12/svgudom.html), licensed under terms as set forth in W3C Software Notice and License, dated December 31 2002.
Contents

RESEARCH AND EVALUATION LICENSE .................................................. iii

Overview .............................................................................................................. 1
  1. Introduction .................................................................................................. 1
  2. Use Cases ..................................................................................................... 3
  3. Requirements ............................................................................................... 3
  4. Status of this document ............................................................................... 5
  5. Example code .............................................................................................. 6
  6. Security Considerations ............................................................................. 10
  7. Data Types ................................................................................................ 10

javax.microedition.m2g ..................................................................................... 11
  ExternalResourceHandler ............................................................................... 12
  ScalableGraphics .......................................................................................... 14
  ScalableImage ............................................................................................... 18
  SVGAnimator ................................................................................................. 24
  SVGEVENTListener ....................................................................................... 30
  SVGImage ..................................................................................................... 34

org.w3c.dom .................................................................................................... 41
  Document ....................................................................................................... 42
  DOMException .............................................................................................. 45
  Element .......................................................................................................... 48
  Node ............................................................................................................... 49

org.w3c.dom.events .......................................................................................... 53
  Event ............................................................................................................... 54
  EventListener .............................................................................................. 55
  EventTarget ................................................................................................. 56

org.w3c.dom.svg ............................................................................................ 59
  SVGAnimationElement ............................................................................... 60
  SVGEElement .............................................................................................. 62
  SVGException ............................................................................................. 76
  SVGLocatableElement ............................................................................... 78
  SVGMatrix ................................................................................................. 83
  SVGPath ....................................................................................................... 86
  SVGPoint ..................................................................................................... 90
  SVGRect ..................................................................................................... 92
  SVGRGBColor ............................................................................................. 94
  SVGSVGEElement ................................................................................... 95

Constant Field Values ..................................................................................... 101
Index ............................................................................................................... 103

W3C® SOFTWARE NOTICE AND LICENSE ................................................. 109
CHAPTER 1

Overview

1. Introduction

This document is the final specification of JSR 226 Scalable 2D Vector Graphics API. This API is targeted for low-end mobile devices with constraints in memory, screen size, and computational power. The goal of this specification is to define an optional API package for rendering Scalable 2D vector images, including external images in SVG format. The main target use cases of this API are map visualization, scalable icons and other applications which require scalable, animated graphics.

Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11-Nov-2003</td>
<td>EG Initial Draft</td>
<td>Based on 2nd EG face-to-face meeting</td>
</tr>
<tr>
<td>09-Dec-2003</td>
<td>EG Draft 0.1</td>
<td>Revision based on regular EG discussions</td>
</tr>
<tr>
<td>12-Dec-2003</td>
<td>EG Draft 0.2</td>
<td>Community Review Draft for EG feedback</td>
</tr>
<tr>
<td>16-Dec-2003</td>
<td>EG Draft 0.3</td>
<td>Community Review Specification</td>
</tr>
<tr>
<td>16-Jan-2004</td>
<td>EG Draft 0.4</td>
<td>Revised based on further EG CR comments</td>
</tr>
<tr>
<td>10-Feb-2004</td>
<td>EG Draft 0.5</td>
<td>Early version of Public Review Draft</td>
</tr>
<tr>
<td>23-Feb-2004</td>
<td>EG Draft 0.6</td>
<td>Proposed Public Draft Specification</td>
</tr>
<tr>
<td>10-Mar-2004</td>
<td>EG Draft 0.7</td>
<td>Public Draft Specification</td>
</tr>
<tr>
<td>01-May-2004</td>
<td>EG Draft 0.8</td>
<td>Initial Proposed Final Draft Specification</td>
</tr>
<tr>
<td>07-May-2004</td>
<td>EG Draft 0.9</td>
<td>Revised Proposed Final Draft Specification, based on feedback from EG</td>
</tr>
<tr>
<td>22-June-2004</td>
<td>EG Draft 0.91</td>
<td>Internal revision of PFD</td>
</tr>
<tr>
<td>22-July-2004</td>
<td>EG Draft 0.92</td>
<td>Internal revision of PFD</td>
</tr>
<tr>
<td>22-Sep-2004</td>
<td>EG Draft 0.94</td>
<td>Internal revision of PFD</td>
</tr>
<tr>
<td>07-Oct-2004</td>
<td>EG Draft 0.95</td>
<td>Internal revision of PFD</td>
</tr>
<tr>
<td>13-Oct-2004</td>
<td>EG Draft 0.96</td>
<td>Internal revision of PFD</td>
</tr>
</tbody>
</table>
1. Introduction

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-Oct-2004</td>
<td>EG Draft 0.97</td>
<td>Internal revision of PFD</td>
</tr>
<tr>
<td>26-Oct-2004</td>
<td>EG Draft 0.98</td>
<td>Internal revision of PFD</td>
</tr>
<tr>
<td>16-Nov-2004</td>
<td>EG Draft 0.99</td>
<td>Internal revision of PFD</td>
</tr>
<tr>
<td>22-Nov-2004</td>
<td>EG Draft 1.0</td>
<td>Internal (final) revision of PFD</td>
</tr>
<tr>
<td>27-Nov-2004</td>
<td>Version 1.0</td>
<td>Final JSR226 Specification</td>
</tr>
</tbody>
</table>

Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUST</td>
<td>The associated definition is an absolute requirement of this specification.</td>
</tr>
<tr>
<td>MUST NOT</td>
<td>The definition is an absolute prohibition of this specification.</td>
</tr>
<tr>
<td>SHOULD</td>
<td>Indicates a recommended practice. There may exist valid reasons in particular circumstances to ignore this recommendation, but the full implications must be understood and carefully weighed before choosing a different course.</td>
</tr>
<tr>
<td>SHOULD NOT</td>
<td>Indicates a non-recommended practice. There may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.</td>
</tr>
<tr>
<td>MAY</td>
<td>Indicates that an item is truly optional.</td>
</tr>
</tbody>
</table>

Expert Group
The JSR-226 EG consists of the following representatives/companies.

- Dirk Ambras, Siemens AG
- Henric Axelsson, Ericsson Mobile Platforms
- Tolga K Capin, Nokia Corporation
- Peter Chen, iSolution Inc.
- Suresh Chitturi, Nokia Corporation (Editor)
- Christophe Gillette, Motorola
- David Girle, IBM
- Andrew Girow, TinyLine
- Vincent Hardy, Sun Microsystems, Inc.
- Jyri Huopaniemi, Nokia Corporation
- Thomas Landspurg, In-Fusio SA
- Michael Leone, Aplix Corporation
- Simon Lewis, Symbian Ltd
2. Use Cases

The primary use cases of this API include:

- Map Visualization
- Scalable Icons
- Animations (messaging)
- Technical Illustrations

3. Requirements

The JSR-226 specification is designed based on the following requirements.

3.1 General

GEN.1: The API must support SVG Tiny Profile

- Must be compatible with SVGT 1.1
- Should contain necessary extension mechanisms to allow support of future SVGT profiles
- May support additional complete SVG profiles

GEN.2: The API should require less than 30 KB footprint for wrapper (ROM) implementation on top of a native SVG component.

GEN.3: The API must not contain optional parts

GEN.4: Rendering hints may be allowed (text/shape/image quality, antialiasing).

GEN.5: The API must be designed to maximize interoperability on multiple implementations

GEN.6: The API must provide functionality to load and interact with some or all features within an SVG file
3. Requirements

3.2 Architecture
ARCH.1: The API must be efficiently implementable alongside CLDC/MIDP.
ARCH.2: The API must be efficiently implementable alongside CDC/PBP/PP/JSR209 and J2SE.
ARCH.3: The API must use CLDC 1.1 as the minimum configuration for floating point support.
ARCH.4: The API must not require more underlying XML features than the SVG Tiny profile supported by the implementation.

3.3 Functional

Rendering
FUNC.1: The API must support rendering SVG Tiny content.
FUNC.2: The API must allow layering and transparency/masking of multiple SVG images.
FUNC.3: The API must allow overlaying graphical objects over a raster/vector image.
FUNC.4: The API should support alpha blending of graphical objects and SVG images in arbitrary order.
FUNC.5: The API must allow zooming, panning, and rotation of SVG content.
FUNC.6: The API should support access to a specified future time within an animated SVG image.
FUNC.7: The API may be extendable to support other vector image formats.

Document Access
FUNC.8: The API must support access to SVG images in file level.

Tree Navigation (i.e., accessing individual nodes)
FUNC.9: This API must support a limited form of access to SVG document tree (e.g. accessing an element by its id).

Element creation
FUNC.10: This API must support creation of new SVG elements, but allow only basic shapes, path, group, image, text, and anchor elements.

Text Node creation
FUNC.11: The API must support creation of new text elements.

Element addition
FUNC.12: The API must support addition of new SVG elements with possible restrictions.

Element removal
FUNC.13: This API must support element removal, but only elements created by the application can be removed by the application.

Attribute access
FUNC.14: The API must support read access to attributes that are animatable.

Attribute modification
FUNC.15: The API must support write access to attributes that are animatable.
FUNC.16: The API should support type-based access to all the animatable attributes.
FUNC.17: The API may support string-based access to generic attributes.
Event dispatching
FUNC.18: The API must allow event dispatching on SVG document.

Event registration and removal
FUNC.19: The API must support event registration and removal.

Update notification, Animation support
FUNC.20: The API must allow the application to start/stop a particular animation.
FUNC.21: The API may allow SVG engine to optimize rendering of animations.

Thread safe access to the Document
FUNC.22: The API must allow synchronized access to the SVG Document/Image.

4. Status of this document

This document contains the Final JSR-226 Specification, as agreed by the JSR-226 Expert Group. It is based on the feedback received during the Proposed Final Draft period. This version of the specification is used as the basis for Reference Implementation (RI) and Technology Compatibility Kit (TCK) development.
5. Example code

Example One
This example illustrates rendering a simple static SVG Image. Before any rendering, the application must bind a rendering target to the ScalableGraphics context.

```java
import javax.microedition.m2d.*;

class MapApplication extends Canvas {
   private SVGImage map;
   private ScalableGraphics gc;

   public MapApplication() {
      // Load map file
      map = SVGImage.createImage(...);
      gc = ScalableGraphics.createInstance();
   }

   public void paint(Graphics g) {
      gc.bindTarget(g);  // Bind the ScalableGraphics context to MIDP Graphics object.
      gc.render(0, 0, map);  // Render the SVG Image.
      gc.releaseTarget();  // Release the target
   }
}
```

Example Two
The following example illustrates rendering of an animated SVG Image using an application/timer loop.

```java
import javax.microedition.m2d.*;
import javax.microedition.lcdui.Canvas;
import javax.microedition.m2d.*;

class MyAnimation extends MIDlet {
   private SVGImage cartoon;
   private MyCanvas myCanvas = null;
   private Timer myRefreshTimer = new Timer();
   private TimerTask myRefreshView = null;
   private ScalableGraphics gc;
   ...

   /**
    * MIDlet paint() method
    */
   public void paint(Graphics g) {
      long currentTime = System.currentTimeMillis();
      cartoon.incrementTime((currentTime - startTime)/1000.0f);  // startTime must be initialized in the code.
      startTime = currentTime;
      gc.bindTarget(g);
      gc.render(0, 0, cartoon);
      gc.releaseTarget();
   }
```
// Schedule an update to refresh, when necessary
myRefreshTimer.schedule(myRefreshView, 100);
}
/**
 * Inner class for refreshing the view.
 */
private class RefreshView extends TimerTask {
    public void run() {
        // Get the canvas to repaint itself.
        myCanvas.repaint();
    }
}
/**
 * Inner class for handling the canvas.
 */
private class MyCanvas extends Canvas {
    MyAnimation myAnimation;
    /**
     * Construct a new canvas
     */
    MyCanvas(MyAnimation animation) {
        myAnimation = animation;
    }
    /**
     * Ask MyAnimation to paint itself
     */
    protected void paint(Graphics g) {
        myAnimation.paint(g);
    }
}

Example Three
The following code sample illustrates how event handling can be supported by this JSR.

This 2 main elements involved in event mechanism are as follows:
- Registration and Removal of event listeners
- Event Creation and Dispatching

a) Registration and Removal of event listeners

... ...
//Load an SVG Image.
SVGImage mySvg = (SVGImage) ScalableImage.createImage(...);
Document cityMap = mySvg.getDocument();
SVGElem buttonControl = (SVGElem) cityMap.getElementById("buttonControl");
SVGElem highwayLayer = (SVGElem) cityMap.getElementById("highwayLayer");
//Create a new instance of MyLayerManager class that implements the EventListener.
MyLayerManager myLayerManager = new MyLayerManager();

// Registration of event listener with the buttonControl SVGElem for "DOMActivate" event
5. Example code

```java
buttonControl.addEventListener("DOMActivate", myLayerManager, false);
...
...
//Removal of event listener, when it is not necessary anymore
buttonControl.removeEventListener("DOMActivate", myLayerManager, false);
...
...
// MyLayerManager implements EventListener interface to handle JSR events. In this example,
// the highway layer in SVG content is turned "off" by setting the "display" property on
// highwaylayer SVGElement to "none".
class MyLayerManager implements EventListener {
    public void handleEvent(Event evt) {
        SVGElement targetElement = (SVGElement)evt.getCurrentTarget();
        if ("DOMActivate".equals(evt.getType()) && "buttonControl".equals(targetElement.getId())) {
            targetElement.setTrait("display", "none");
        }
    }
}
```

b) Event Creation and Dispatching

Based on the user event, the application notifies the SVGImage/document using the following command calls. It is the responsibility of the SVGImage to determine and set the target SVGElement for this event and also invoke the appropriate event listener registered with the SVG document.

```java
class MyApplication extends Canvas {
...
...
//Listen to user events from MIDP canvas
void keyPressed(int keyCode) {
    int gameAction = getGameAction(keyCode);
    // Logic to map the key code to the JSR-226 events is done here.
    switch(gameAction) {
        case FIRE:
            // Create a "DOMActivate" Event and dispatch
            mySvg.activate();
            break;
    }
}

// Similarly, create another JSR-226 Event (click) and dispatch it to the SVG document.
void pointerPressed(int x, int y) {
    mySvg.dispatchEventMouseEvent("click", x, y);
}
```

Example Four
This example shows rendering of SVG content using a “player” type API, also called the SVGAnimator. The SVGAnimator class handles automatic rendering of updates and animations in an SVGImage to a target user interface (UI) component.

```
// Create an SVGImage map = ...; // See the SVGImage documentation.
SVGImage map;
// Create an SVGAnimator and set it on the animator.
SVGAnimator svgAnimator = SVGAnimator.createAnimator(map);

// MIDPSVGEventListener listener = new MIDPSVGEventListener(map, animator);
MIDPSVGEventListener listener;

// SVGEventListener sample implementation. Adds a new circle every time a key is pressed.
// class MIDPSVGEventListener implements SVGEventListener, Runnable {
//     protected SVGDocument svgDocument;
//     protected SVGAnimator svgAnimator;
//     protected Vector addToTree = new Vector();
//
//     public MIDPSVGEventListener(SVGImage svgImage, SVGAnimator animator) {
//         this.svgDocument = svgImage.getDocument();
//         this.svgAnimator = svgAnimator;
//     }
//
//     public void keyPressed(int keyCode) {
//         SVGDocument circle = svgDocument.createElementNS(svgNS, "circle");
//         circle.setFloatTrait("cx", ...);
//         circle.setFloatTrait("cy", ...);
//         // ....
//         synchronized (addToTree) {
//             addToTree.addElement(circle);
//             svgAnimator.invokeLater(this);
//         }
//     }
//
//     public run() {
//         synchronized (addToTree) {
//             for (int i = 0; i < addToTree.size(); i++) {
//                 addToTree.removeElement((SVGElement) addToTree.elementAt(i));
//             }
//             addToTree.clear();
//         }
//     }
```
6. Security Considerations

JSR226 does not define any security mechanism. Rather, implementations of JSR226 API are subject to the security mechanisms provided by the underlying profile and configuration, e.g. OMA DRM, network security such as HTTPS, SSL, etc.

A few methods in this API are defined such that a SecurityException will be thrown when called without the appropriate security permissions from the caller. An implementation must guarantee that:

- the exception is thrown when the caller does not have the appropriate security permissions; and
- the method can be used when the appropriate permissions are granted.

7. Data Types

The user of the JSR226 API should be aware that SVG Tiny implementations are representing floats using fixed points and therefore the users must limit their input ranges to be within the range of -32,767.9999 to +32,767.9999 (or the scientific notation equivalent). This limitation applies to the values of the traits, but more importantly to any calculation performed internally by the SVG engine.

In addition, note that CSS units are only supported on the width and height attributes of the root SVG element, and for all the other traits only default (user) units are applied. For example, coordinates are expressed in the user space coordinate system and angles are always expressed in degrees. For more information, please visit SVG Mobile specification (http://w3c.org/TR/SVGMobile/#sec-data-types).
CHAPTER 2

Package
javax.microedition.m2g

Description
This package contains the core rendering part of this proposal. All the rendering is done using the classes/interfaces defined in this package.

Class Summary

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExternalResourceHandle</td>
<td>This interface is used to load external resources that are referenced within a ScalableImage, namely within an SVG document in the case of SVG.</td>
</tr>
<tr>
<td>SVGEventListener</td>
<td>The SVGEventListener is used to forward platform-specific events to an application.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScalableGraphics</td>
<td>This is the fundamental class for 2D rendering.</td>
</tr>
<tr>
<td>ScalableImage</td>
<td>This class models images in vector format, such as the Scalable Vector Graphics (SVG) image format.</td>
</tr>
<tr>
<td>SVGAnimator</td>
<td>The SVGAnimator class handles automatic rendering of updates and animations in an SVGImage to a target user interface (UI) component.</td>
</tr>
<tr>
<td>SVGImage</td>
<td>This class represents an SVG image conforming to the W3C SVG Tiny 1.1 Profile.</td>
</tr>
</tbody>
</table>
ExternalResourceHandler javax.microedition.m2g
requestResource(ScalableImage, String)

javax.microedition.m2g
ExternalResourceHandler

Declaration
public interface ExternalResourceHandler

Description
This interface is used to load external resources that are referenced within a ScalableImage, namely within an
SVG document in the case of SVG. In SVGT 1.1, these resources are not required for displaying the SVG
document. If no ExternalResourceHandler is registered when creating the image, the engine will retrieve these
resources using its own implementation of the requestResource method.

The default engine behaviour can be overwritten by passing to createImage an implementation of this
interface as parameter when the default implementation is insufficient. For example, if the SVGImage is loaded
from within a Jar file, the implementation will not be able to load images that have relative URIs (e.g., <image
xlink:href="myImage.png" />). All the external resources are requested by the engine at the time of the
createImage call. Note that SVGT 1.1 does not contain external references to SVG files, therefore an
external resource can not contain subsequent external resources.

See Also: ScalableImage

Member Summary

Methods

| void requestResource(ScalableImage image, java.lang.String URI)

Methods

requestResource(ScalableImage, String)

Declaration:
public void requestResource(javax.microedition.m2g.ScalableImage, image,
java.lang.String URI)

Description:
This method is invoked when an external resource is required by the underlying implementation. When the
request is completed by the implementation of this handler, a notification must be sent to the SVG engine
through the requestCompleted() method of ScalableImage. To get a synchronous behaviour,
requestCompleted() can be called in the implementation of requestResource. If called later, rendering the
document before completing all the requests will just display the currently available content. Once the
request is completed, the image will have to be redrawn to reflect the newly available data. For more details
on required resources, please refer to the externalResourcesRequired attribute description in the
SVG specification.

Parameters:
  image - image that originated the external data request
javax.microedition.m2g

**ExternalResourceHandler**

```java
requestResource(ScalableImage, String)
```

URI - the URI for the external resource.
ScalableGraphics javamicroedition.m2g

javamicroedition.m2g ScalableGraphics

Declaration
public class ScalableGraphics

java.lang.Object
| --javamicroedition.m2g.ScalableGraphics

Description
This is the fundamental class for 2D rendering. The ScalableGraphics context class provides and handles all the rendering capability within this package. In other words, the rendering can only be achieved through the render method provided in this class. Note that the ScalableGraphics instance must be bound to the rendering target prior to calling the render method. The implementation must clip to the viewport boundaries.

Member Summary

<table>
<thead>
<tr>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int RENDERING_QUALITY_HIGH 14</td>
</tr>
<tr>
<td>static int RENDERING_QUALITY_LOW 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>void bindTarget(java.lang.Object target) 15</td>
</tr>
<tr>
<td>static createInstance() 15</td>
</tr>
<tr>
<td>ScalableGraphics</td>
</tr>
<tr>
<td>void releaseTarget() 15</td>
</tr>
<tr>
<td>void render(int x, int y, ScalableImage image) 16</td>
</tr>
<tr>
<td>void setRenderingQuality(int mode) 16</td>
</tr>
<tr>
<td>void setTransparency(float alpha) 16</td>
</tr>
</tbody>
</table>

Inherited Member Summary

<table>
<thead>
<tr>
<th>Methods inherited from class Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone(), equals(Object), finalize(), getClass(), hashCode(), notify(), notifyAll(), toString(), wait(long, int), wait(long, int), wait(long, int)</td>
</tr>
</tbody>
</table>

Fields

RENDERING_QUALITY_HIGH

Declaration:
public static final int RENDERING_QUALITY_HIGH
javax.microedition.m2g

ScalableGraphics

RENDERING_QUALITY_LOW

Description:
Defines a high rendering quality level.

RENDERING_QUALITY_LOW

Declaration:
public static final int RENDERING_QUALITY_LOW

Description:
Defines a low rendering quality level.

Methods

bindTarget(Object)

Declaration:
public void bindTarget(java.lang.Object target)

Description:
Binds the given Graphics as the rendering target of this ScalableGraphics context. The type of the Graphics
object depends on the Java profile that this specification is implemented on, as follows:

• javax.microedition.lcdui.Graphics on profiles supporting LCDUI;
• java.awt.Graphics on profiles supporting AWT;
• either of the above on profiles supporting both AWT and LCDUI.

Parameters:
  target - the object (Graphics) to receive the rendered image.

Throws:
  java.lang.NullPointerException - if target is null.
  java.lang.IllegalArgumentException - if target is invalid.
  java.lang.IllegalStateException - if target is already bound.

createInstance()

Declaration:
public static javax.microedition.m2g.ScalableGraphics createInstance()

Description:
Retrieve a new instance of ScalableGraphics that can be associated to an application.

Returns: the newly created ScalableGraphics instance.

releaseTarget()

Declaration:
public void releaseTarget()

Description:
Flushes the rendered ScalableImage to the currently bound target and then releases the target. This ensures
that the ScalableImage is actually made visible on the target that was set in bindTarget. Otherwise, the
image may or may not become visible.
render(int, int, ScalableImage)

Throws:
java.lang.IllegalStateException - if target is not bound.

render(int, int, ScalableImage)

Declaration:
public void render(int x, int y, javax.microedition.m2g.ScalableImage image)

Description:
Renders the specified ScalableImage using the supplied anchor point. The anchor point given is relative to the upper left corner of the rendering surface. It is important to note that the content is made visible or flushed to the display only after a call is made to releaseTarget.

Parameters:
x - the X coordinate of the anchor point, in pixels.
y - the Y coordinate of the anchor point, in pixels.
image - the ScalableImage to be rendered.

Throws:
java.lang.NullPointerException - if image is null.
java.lang.IllegalStateException - if target is not bound.

See Also: releaseTarget()

setRenderingQuality(int)

Declaration:
public void setRenderingQuality(int mode)

Description:
Set the quality of rendering in the ScalableGraphics context. It can take one of the values, RENDERING_QUALITY_LOW or RENDERING_QUALITY_HIGH. Default=RENDERING_QUALITY_HIGH. The implementation of these quality levels is implementation dependent and should be mapped to definitions in SVG spec (shape, text, image and color rendering).

Parameters:
mode - this value indicates the quality of rendering required.

Throws:
java.lang.IllegalArgumentException - if the mode is invalid.

setTransparency(float)

Declaration:
public void setTransparency(float alpha)

Description:
Set the transparency in the ScalableGraphics context with the supplied alpha value. Alpha value must be a floating point number in the range [0.0, 1.0]. The source pixels are always combined with destination pixels using the Source Over Destination rule [Porter-Duff]. In this context, the Source Over Destination rule has the following properties: a fully opaque pixel in the source must replace the destination pixel, a fully transparent pixel in the source must leave the destination pixel unchanged, and a semitransparent pixel in the source must be alpha blended with the destination pixel using the supplied value. The default alpha value is 1.0 (fully opaque), when not specified.
ScalableGraphics

setTransparency(float)

javax.microedition.m2g

Parameters:
alpha - the constant alpha value to be used for rendering.

Throws:
java.lang.IllegalArgumentException - if alpha is out of range.
ScalableImage

javax.microedition.m2g

setTransparency(float)

javax.microedition.m2g
ScalableImage

Declaration
public abstract class ScalableImage

java.lang.Object
|--
   |-- javax.microedition.m2g.ScalableImage

Direct Known Subclasses: SVGImage

Description
This class models images in vector format, such as the Scalable Vector Graphics (SVG) image format. Therefore, it is required that all classes representing "scalable" images extend this class.

See Also: SVGImage

Member Summary

Constructors
protected ScalableImage()  

Methods
static ScalableImage createImage(java.io.InputStream stream, javax.microedition.m2g.ExternalResourceHandler handler)
static ScalableImage createImage(java.lang.String url, javax.microedition.m2g.ExternalResourceHandler handler)
abstract int getViewportHeight()
abstract int getViewportWidth()
abstract void requestCompleted(java.lang.String URI, java.io.InputStream resourceData)
abstract void setViewportHeight(int height)
abstract void setViewportWidth(int width)

Inherited Member Summary

Methods inherited from class java.lang.Object
clone(), equals(Object), finalize(), getClass(), hashCode(), notify(), notifyAll(), toString(), wait(long, int), wait(long, int), wait(long, int)
Constructors

ScalableImage()

Declaration:
protected ScalableImage()

Description:
Constructor

Methods

createImage(InputStream, ExternalResourceHandler)

Declaration:
public static javax.microedition.m2g.ScalableImage createImage(java.io.InputStream stream, javax.microedition.m2g.ExternalResourceHandler handler) throws IOException

Description:
This method creates and loads a ScalableImage (e.g. SVG) from the specified stream. A ScalableImage can only be rendered using the render() method in the ScalableGraphics context. The image size is determined by the content specification (e.g., width/height attributes on root SVG element). The default viewport size of 100-by-100 pixels is used when the size is unspecified. This method will throw an exception when the image is in error. For SVG document, the document is in error when:

- the SVG document does not conform to the XML 1.0 specification
- an element has an attribute or property value which is not permissible according to the SVG specification
- the required or mandatory attributes according to the SVG specification are missing
- the document contains circular references on the <use> element
- the document contains <image> element(s) with local references
- the document contains elements with duplicate Id's
- the document contains animation(s) in error

If a handler is specified, the engine will invoke it for any external resource referenced in the document. However, please note that data URIs (for ex: base64 encoded images like <image xlink:href="data:image/png;base64,/9j/4AAQ..."/> ) are required to be decoded by the engine, and therefore the handler will not be invoked in such cases. All these requests must occur before createImage returns, and the engine must only make one call if there exist multiple resources with the same URI. If the handler is set to null, the ScalableImage will try to load images automatically using the engine's default implementation, but might not be able to load all of them. For example, if the ScalableImage is loaded from a file in a Jar file, the implementation will not be able to load images that have relative URIs (ex: <image xlink:href="myImage.png"/> ). If the handler is not able to locate a requested resource, it informs the SVG engine by calling the requestCompleted method with null for resourceData.

Note that the handler is also called when the xlink:href attribute on <image> is set or changed by the application, but the call is made only when the element is hooked into the document tree i.e. when the ancestors go all the way up to the root <svg> element. There are two cases:
ScalableImage javax.microedition.m2g
createImage(String, ExternalResourceHandler)

- When changing the xlink:href attribute of an existing <image> element that is already hooked or part of the tree.
- When creating a new <image>, setting its xlink:href and hooking it to the document tree.

Parameters:
- stream - The stream from which the SVG content should be read.
- handler - implementation of the ExternalResourceHandler interface

Returns: the loaded ScalableImage.

Throws:
- java.io.IOException - if an error occurs while loading the content.
- java.lang.NullPointerException - if stream is null.

See Also: ExternalResourceHandler

createImage(String, ExternalResourceHandler)

Declaration:
public static javax.microedition.m2g.ScalableImage createImage(java.lang.String url, javax.microedition.m2g.ExternalResourceHandler handler)
throws IOException

Description:
This method creates and loads a ScalableImage (e.g. SVG) with the specified URI. A ScalableImage can only be rendered using the render() method in the ScalableGraphics context. The image size is determined by the content specification (e.g: width/height attributes on root SVG element). The default viewport size of 100-by-100 pixels is used when the size is unspecified. This method will throw an exception when the image is in error. For SVG document, the document is in error when:

- the SVG document does not conform to the XML 1.0 specification
- an element has an attribute or property value which is not permissible according to the SVG specification
- the required or mandatory attributes according to the SVG specification are missing
- the document contains circular references on the <use> element
- the document contains <image> element(s) with local references
- the document contains elements with duplicate Id’s
- the document contains animation(s) in error

If an handler is specified, the engine will invoke it for any external resource referenced in the document. However, please note that data URIs (for ex: base64 encoded images like <image xlink:href="data:image/png;base64,/9j/4AAQ..."/> are required to be decoded by the engine, and therefore the handler will not be invoked in such cases. All these requests must occur before createImage returns, and the engine must only make one call if there exist multiple resources with the same URI. If the handler is set to null, the ScalableImage will try to load images automatically using the engine’s default implementation, but might not be able to load all of them. For example, if the ScalableImage is loaded from a file in a Jar file, the implementation will not be able to load images that have relative URIs (ex: <image xlink:href="myImage.png" />). If the handler is not able to locate a requested resource, it informs the SVG engine by calling the requestCompleted method with null for resourceData.
Note that the `handler` is also called when the `xlink:href` attribute on `<image>` is set or changed by the application, but the call is made only when the element is hooked into the document tree i.e. when the ancestors go all the way up to the root `<svg>` element. There are two cases:

- When changing the `xlink:href` attribute of an existing `<image>` element that is already hooked or part of the tree.
- When creating a new `<image>`, setting its `xlink:href` and hooking it to the document tree.

If the platform implementation supports the JSR-75 FileConnection, then the file: URLs as used by JSR-75 shall be supported and the images are loaded from files identified by this locator. This can be used to load possible DRM protected image files from the file system.

**Parameters:**
- `url` - A string in URL syntax that identifies the image source.
- `handler` - implementation of the `ExternalResourceHandler` interface

**Returns:** the loaded ` ScalableImage`

**Throws:**
- `java.io.IOException` - if an error occurs while loading the SVG content.
- `java.lang.NullPointerException` - if `URL` is null.
- `java.lang.IllegalArgumentException` - if the specified locator is of unsupported type.
- `java.lang.SecurityException` - if the application does not have the privilege rights to access the contents of this resource.

**See Also:** `ExternalResourceHandler` 12

### getViewportHeight()

**Declaration:**
```
public abstract int getViewportHeight()
```

**Description:**
This method returns the ` ScalableImage`’s viewport height. The initial viewport height is taken from the “height” value specified in the ` Scalable Image`. The value returned is always in pixels. If the specified height is defined in percentages, the values are mapped to the default view port size of 100x100. If the viewport height is explicitly changed by the application, then the percentages are ignored and the content is made to fit to this new viewport height.

**Returns:** the current height of this ` ScalableImage`.

**See Also:** `setViewportHeight(int)` 22

### getViewportWidth()

**Declaration:**
```
public abstract int getViewportWidth()
```

**Description:**
This method returns the ` ScalableImage`’s viewport width. The initial viewport width is taken from the “width” value specified in the ` Scalable Image`. The value returned is always in pixels. If the specified width is defined in percentages, the values are mapped to the default view port size of 100x100. If the viewport width is explicitly changed by the application, then the percentages are ignored and the content is made to fit to this new viewport width.

**Returns:** the current width of this ` ScalableImage`. 
ScalableImage javax.microedition.m2g
requestCompleted(String, InputStream)

See Also: setViewportWidth(int) 22

requestCompleted(String, InputStream)

Declaration:
public abstract void requestCompleted(java.lang.String URI,
java.io.InputStream resourceData)
throws IOException

Description:
Once the requested external resource is available, the application forwards this information (resourceData) to the SVG engine. If this method is called a second time for a same URL of a same SVGImage, the engine will replace the current resource data with the new one. Note: Setting null for resourceData indicates that the requested resource could not be fetched by the ResourceHandler or application, and in this event the SVG engine will not make further attempts to load this resource.

Parameters:
- URI - URI that was requested through requestExternalResource
- resourceData - inputstream containing the external resource

Throws:
- java.io.IOException - if an error occurs while loading the resource.
- java.lang.NullPointerException - if URL is null.

See Also: ExternalResourceHandler 12

setViewportHeight(int)

Declaration:
public abstract void setViewportHeight(int height)

Description:
This method sets the new (viewport) height of this ScalableImage.

Parameters:
- height - the new height to be set.

Throws:
- java.lang.IllegalArgumentException - if height is negative.

See Also: getViewportHeight() 21

setViewportWidth(int)

Declaration:
public abstract void setViewportWidth(int width)

Description:
This method sets the new (viewport) width of this ScalableImage. The viewport is the area where the ScalableImage is rendered. Any parts of the viewport that lie outside the boundaries of the target clipping rectangle are clipped. The viewport upper left corner (x, y) is given relative to the upper left corner of the target rendering surface.

Parameters:
- width - the new width to be set.

Throws:
- java.lang.IllegalArgumentException - if width is negative.
See Also: `getViewportWidth()`
**Declaration**

```java
public abstract class SVGAnimator
```

java.lang.Object

```
|--javax.microedition.m2g.SVGAnimator
```

**Description**

The `SVGAnimator` class handles automatic rendering of updates and animations in an `SVGIImage` to a target user interface (UI) component. The target component type depends on the Java profile this specification is implemented on, as described in the `createAnimator` and `getTargetComponent` methods documentation. There are two types of rendering updates the `SVGAnimator` handles:

- **Animation Updates.** The `SVGAnimator` can run animations and automatically and periodically refreshes its rendering to reflect the effect of animations.

- **SVGIImage Updates.** The `SVGAnimator` updates its rendering following alterations of the associated `SVGIImage`, for example if the position of one of the graphical elements is modified by an API call.

An `SVGAnimator` instance can be in one of the following states:

- **Stopped.** This is the initial state. The `SVGAnimator` performs no rendering updates. Possible transitions:
  - To the `playing` state, with the `play` method.

- **Playing.** This is the typical state for an `SVGAnimator`. In that state, the `SVGAnimator` performs both Animation and SVGIImage updates. While there are active animations, the animator updates the rendering at a rate not faster than the one defined by the `setTimeIncrement` method. If SVGIImage updates are made (e.g., with calls to the `SVGElement setTrait` method, see examples), the rendering is updated at the time of the next animation rendering.

  When there are no active animations, the animator will update the rendering after each `Runnable` passed to the `invokeLater` or `invokeAndWait` methods has finished executing. Possible transitions:
  - To the `stopped` state, with the `stop` method.
  - To the `paused` state, with the `pause` method.

- **Paused.** When in that state, the `SVGAnimator` only performs SVGIImage updates rendering. The animator no longer automatically advances the SVG document's current time, so rendering reflects the animation at the document's current time. Note that a change to the document's current time while in the `paused` state will trigger a new rendering for the new current time. Possible transitions:
  - To the `stopped` state, with the `stop` method.
  - To the `playing` state, with the `play` method.

**Code example:**
// Create an SVGAnimator
SVGImage map = ...; // See the SVGImage documentation.
SVGAnimator svgAnimator = SVGAnimator.createAnimator(map);
// Display the associated SVGAnimator component.
// Depends on the platform.
// ================ AWT Example ================
Panel panel = ...;
panel.add((Component) svgAnimator.getTargetComponent());
...
// ================ MIDP Example ================
Canvas svgCanvas = (Canvas) svgAnimator.getTargetComponent();
...
// Start rendering animations.
svgAnimator.play();
....

class MapRunnable implements Runnable {
    public void run() {
        // Perform map updates based on current
        // traffic information.
        SVGElement statusRect = map.getDocument().getElementById("statusRect");
        // Reflect that traffic status.
        statusRect.setRGBTrait(...); // See setRGBTrait documentation.
    }
}
Runnable mapUpdates = new MapRunnable();
....
while (someLoopCondition) {
    if (hasMapUpdate) {
        svgAnimator.invokeAndWait(mapUpdates);
    }
}  

---

**Member Summary**

**Constructors**

- SVGAnimator() _26_

**Methods**

- static SVGAnimator createAnimator(SVGImage svgImage) _26_
- static SVGAnimator createAnimator(SVGImage svgImage, java.lang.String componentBaseClass) _26_
- abstract java.lang.Object getTargetComponent() _27_
- abstract float getTimeIncrement() _27_
- abstract void invokeAndWait(java.lang.Runnable runnable) _27_
- abstract void invokeLater(java.lang.Runnable runnable) _28_
- abstract void pause() _28_
- abstract void play() _28_
- abstract void setSVGEventListener(SVGEventListener svgEventListener) _28_
- abstract void setTimeIncrement(float timeIncrement) _29_
- abstract void stop() _29_

---

**Inherited Member Summary**

**Methods inherited from class Object**
Constructors

**SVGAnimator()**

Declaration:
```
public SVGAnimator()
```

Methods

**createAnimator(SVGImage)**

Declaration:
```
public static javax.microedition.m2g.SVGAnimator
    createAnimator(javax.microedition.m2g.SVGImage svgImage)
```

Description:
This method creates a new SVGAnimator for the specified SVGImage.

Parameters:
- `svgImage` - the SVGImage this animator should render.

Returns: a new SVGAnimator instance.

Throws:
- `java.lang.NullPointerException` - if `svgImage` is null.

**createAnimator(SVGImage, String)**

Declaration:
```
public static javax.microedition.m2g.SVGAnimator
    createAnimator(javax.microedition.m2g.SVGImage svgImage,
                    java.lang.String componentBaseClass)
```

Description:
This method creates a new SVGAnimator for the specified SVGImage. The following components types must be supported:

- “javax.microedition.lcdui.Canvas” on profiles supporting LCDUI
- “java.awt.Component” on profiles supporting AWT

On platforms that support the Swing UI component framework, the “javax.swing.JComponent” string may be used to request a Swing component.

Parameters:
- `svgImage` - the SVGImage this animator should render.
- `componentBaseClass` - the desired base class for the component associated with the animator.

This is used when the platform this specification is implemented on supports multiple UI component
frameworks. If componentBaseClass is null, this is equivalent to invoking the createAnimator method with the svgImage parameter only.

Returns: a new SVGAnimator instance.

Throws:
   java.lang.NullPointerException - if svgImage is null.
   java.lang.IllegalArgumentException - if the requested componentBaseClass is not supported by the implementation.

getTargetComponent()

Declaration: public abstract java.lang.Object getTargetComponent()

Description: The type of target component associated with the animator depends on the Java profile this specification is implemented on:
   • javax.microedition.lcdui.Canvas on profiles supporting LCDUI
   • java.awt.Component on profiles supporting AWT

Returns: target the target component associated with the animator.
See Also: createAnimator(SVGImage)

getTimeIncrement()

Declaration: public abstract float getTimeIncrement()

Description: Returns the current time increment used for animation rendering. The SVGAnimator increments the SVG document’s current time by this amount between each rendering.

Returns: the current time increment used for animation rendering. The default value is 0.1 (100 milliseconds)
See Also: setTimeIncrement(float)

invokeAndWait(Runnable)

Declaration: public abstract void invokeAndWait(java.lang.Runnable runnable) throws InterruptedException

Description: Invoke the input Runnable in the Document update thread and return after the Runnable has completed.

Parameters:
   runnable - the new Runnable to invoke.

Throws:
   java.lang.InterruptedException - if the current thread is waiting, sleeping, or otherwise paused for a long time and another thread interrupts it.
   java.lang.NullPointerException - if runnable is null.
   java.lang.IllegalStateException - if the animator is in the stopped state.
SVGAnimator javax.microedition.m2g
invokeLater(Runnable)

invokeLater(Runnable)

Declaration: public abstract void invokeLater(java.lang.Runnable runnable)

Description: Schedule the input Runnable for execution in the update thread at a later time.

Parameters: runnable - the new Runnable to execute in the Document’s update thread when time permits.

Throws: java.langNullPointerException - if runnable is null.
        java.lang.IllegalStateException - if the animator is in the stopped state.

pause()

Declaration: public abstract void pause()

Description: Transitions this SVGAnimator to the paused state. The SVGAnimator stops advancing the document’s current time automatically (see the SVGDocument’s setCurrentTime method). As a result, animations are paused until another call to the play method is made, at which points animations resume from the document’s current time. SVGImage updates (through API calls) trigger a rendering update while the SVGAnimator is in the paused state.

Throws: java.lang.IllegalStateException - if the animator is not in the playing state.

play()

Declaration: public abstract void play()

Description: Transitions this SVGAnimator to the playing state. While in the playing state, both Animation and SVGImage updates trigger rendering updates. Note that a change to the document’s current time while in the playing state will cause the animator to seek to the new time, and continue to play animations forward.

Throws: java.lang.IllegalStateException - if the animator is not currently in the stopped or paused state.

setSVGEventListener(SVGEventListener)

Declaration: public abstract void setSVGEventListener(javax.microedition.m2g.SVGEventListener30 svgEventListener)

Description: Sets the SVGEventListener associated with this SVGAnimate.

Parameters: svgEventListener - the new SVGEventListener which will receive events forwarded by this SVGAnimator. May be null, in which case events are not forwarded by the SVGAnimator.
**setTimeIncrement(float)**

**Declaration:**

```java
public abstract void setTimeIncrement(float timeIncrement)
```

**Description:**
Sets the time increment to use for animation rendering.

**Parameters:**
- `timeIncrement` - the minimal time that should elapse between frame rendering. In seconds.
  - Should be greater than zero.

**Throws:**
- `java.lang.IllegalArgumentException` - if `timeIncrement` is less than or equal to zero.

**See Also:** `getTimeIncrement()`

**stop()**

**Declaration:**

```java
public abstract void stop()
```

**Description:**
Transitions this `SVGAnimator` to the `stopped` state. This causes the `SVGAnimator` to stop advancing the document’s current time automatically, and no rendering updates are performed while in `stopped` state. A call to the `play` method will cause the animations to resume from the document’s current time.

**Throws:**
- `java.lang.IllegalStateException` - if the animator is not in the `playing` or `paused` state.
Declaration

public interface SVGEventListener

Description

The SVGEventListener is used to forward platform-specific events to an application. The application can implement this interface and, in response, dispatch SVG events on an SVGImage object.

Code example:

```java
// Create an SVGAnimator
SVGImage map = ...; // See the SVGImage documentation.
SVGAnimator svgAnimator = SVGAnimator.createAnimator(map);
// Create an SVGEventListener and set it on the animator.
MIDPSVGEventListener listener = new MIDPSVGEventListener(map, animator);
// SVGEventListener sample implementation. Adds a new circle every time a key is pressed.

class MIDPSVGEventListener implements SVGEventListener, Runnable {
    protected SVGDocument svgDocument;
    protected SVGAnimator svgAnimator;
    protected Vector addToTree = new Vector();
    public MIDPSVGEventListener(SVGImage svgImage, SVGAnimator animator) {
        this.svgDocument = svgImage.getDocument();
        this.svgAnimator = svgAnimator;
    }
    public void keyPressed(int keyCode) {
        SVGElement circle = svgDocument.createElementNS(svgNS, "circle");
        circle.setFloatTrait("cx", ...);
        circle.setFloatTrait("cy", ...);
        synchronized (addToTree) {
            addToTree.addElement(circle);
        }
        svgAnimator.invokeLater(this);
    }
    public run() {
        synchronized (addToTree) {
            for (int i = 0; i < addToTree.size(); i++) {
                svgDocument.getDocumentElement().appendChild((SVGElement) addToTree.elementAt(i));
            }
            addToTree.clear();
        }
    }
}
```
javax.microedition.m2g

SGEventListener

hideNotify()

Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>void hideNotify()</td>
</tr>
<tr>
<td>void keyPressed(int keyCode)</td>
</tr>
<tr>
<td>void keyReleased(int keyCode)</td>
</tr>
<tr>
<td>void pointerPressed(int x, int y)</td>
</tr>
<tr>
<td>void pointerReleased(int x, int y)</td>
</tr>
<tr>
<td>void showNotify()</td>
</tr>
<tr>
<td>void sizeChanged(int width, int height)</td>
</tr>
</tbody>
</table>

Methods

hideNotify()

Declaration:
public void hideNotify()

Description:
Invoked by the SVG implementation when the associated component is hidden. On MIDP, this method is invoked when the javax.microedition.lcdui.Canvas.hideNotify method is invoked. On AWT, this method is invoked when the java.awt.event.ComponentEvent with type java.awt.event.ComponentEvent.COMPONENT_HIDDEN is invoked on a java.awt.Component’s component listener.

keyPressed(int)

Declaration:
public void keyPressed(int keyCode)

Description:
Invoked by the SVG implementation when a key was pressed while the component associated with the SVGAnimator had focus. On MIDP, this method is invoked when the javax.microedition.lcdui.Canvas.keyPressed() method is invoked. On AWT, this method is invoked when the java.awt.event.KeyListener.keyPressed() method is invoked on a java.awt.Component’s key listener.

Parameters:
keyCode - the code of the key that was pressed. For MIDP, the code is the same as for the javax.microedition.lcdui.Canvas.keyPressed keyCode argument. For AWT, the code is the same as in the java.awt.event.KeyEvent.getKeyCode() method.

keyReleased(int)

Declaration:
public void keyReleased(int keyCode)

Description:
Invoked by the SVG implementation when a key was released while the component associated with the SVGAnimator had focus. On MIDP, this method is invoked when the javax.microedition.lcdui.Canvas.keyReleased() method is invoked. On AWT, this method is invoked when the java.awt.event.KeyListener.keyReleased() method is invoked on a java.awt.Component’s key listener.
SVGEventLister

**pointerPressed(int, int)**

*Parameters:*
- keyCode - the code of the key that was pressed. For MIDP, the code is the same as for the
  *javax.microedition.lcdui.Canvas.keyReleased.keyCode* argument. For AWT, the
code is the same as in the *java.awt.event.KeyEvent.getKeyCode()* method.

*Declaration:*

```java
public void pointerPressed(int x, int y)
```

*Description:*
Invoked by the SVG implementation when the pointer device (if any), is pressed over the component
associated with the SVGAnimator. On MIDP, this method is invoked when the
*javax.microedition.lcdui.Canvas.pointerPressed()* method is invoked. Note that
pointer events are only supported on MIDP if the platform supports them, as defined by the
Canvas.hasPointerEvents method. On AWT, this method is invoked when the
*java.awt.event.MouseListener.mousePressed()* method is invoked on a
java.awt.Component’s mouse listener.

*Parameters:*
- x - the x-axis coordinate, in the target component’s space (i.e., relative to the upper left corner of the
component associated with the SVGAnimator. On MIDP, this is the same value as passed to the
  *javax.microedition.midp.Canvas.pointerPressed()* method. On AWT, this is the
same value as returned from the *java.awt.event.MouseEvent.getX()* method.
- y - the y-axis coordinate, in the target component’s space (i.e., relative to the upper left corner of the
component associated with the SVGAnimator. On MIDP, this is the same value as passed to the
  *javax.microedition.midp.Canvas.pointerPressed()* method. On AWT, this is the
same value as returned from the *java.awt.event.MouseEvent.getY()* method.

**pointerReleased(int, int)**

*Declaration:*

```java
public void pointerReleased(int x, int y)
```

*Description:*
Invoked by the SVG implementation when the pointer device (if any), is released over the component
associated with the SVGAnimator. On MIDP, this method is invoked when the
*javax.microedition.lcdui.Canvas.pointerReleased()* method is invoked. Note that
pointer events are only supported on MIDP if the platform supports them, as defined by the
Canvas.hasPointerEvents method. On AWT, this method is invoked when the
*java.awt.event.MouseListener.mouseReleased()* method is invoked on a
java.awt.Component’s mouse listener.

*Parameters:*
- x - the x-axis coordinate, in the target component’s space (i.e., relative to the upper left corner of the
component associated with the SVGAnimator. On MIDP, this is the same value as passed to the
  *javax.microedition.midp.Canvas.pointerReleased()* method. On AWT, this is the
same value as returned from the *java.awt.event.MouseEvent.getX()* method.
- y - the y-axis coordinate, in the target component’s space (i.e., relative to the upper left corner of the
component associated with the SVGAnimator. On MIDP, this is the same value as passed to the
  *javax.microedition.midp.Canvas.pointerReleased()* method. On AWT, this is the
same value as returned from the *java.awt.event.MouseEvent.getY()* method.
showNotify()

**Declaration:**
```
public void showNotify()
```

**Description:**
Invoked by the SVG implementation when the associated component is shown. On MIDP, this method is invoked when the `javax.microedition.lcdui.Canvas.showNotify` method is invoked. On AWT, this method is invoked when the `java.awt.event.ComponentEvent` with type `java.awt.event.ComponentEvent.COMPONENT_SHOWN` is invoked on a `java.awt.Component`'s component listener.

sizeChanged(int, int)

**Declaration:**
```
public void sizeChanged(int width, int height)
```

**Description:**
Invoked by the SVG implementation when the associated component is resized. On MIDP, this method is invoked when the `javax.microedition.lcdui.Canvas.sizeChanged` method is invoked. On AWT, this method is invoked when the `java.awt.event.ComponentEvent` with type `java.awt.event.ComponentEvent.COMPONENT_RESIZED` is invoked on a `java.awt.Component`'s component listener.

**Parameters:**
- `width` - the new component width.
- `height` - the new component height.
**Declaration**

```java
public class SVGImage extends ScalableImage
```

The `SVGImage` class represents an SVG image conforming to the W3C SVG Tiny 1.1 Profile. Applications can access the Document associated with this SVGImage using the `getDocument` method. Using the `Document` class, it is possible to interact with the content and perform operations such as zoom, pan, and rotate apart from various other functions. The event dispatching methods as described in this class must be used for dispatching user specific events to the underlying SVG engine.

The width and height values specified in the SVGImage are only used to compute the initial viewport width/height. It is also important to note that according to the SVG 1.1 specification, if viewBox is not defined, then the `preserveAspectRatio` attribute is ignored. Therefore, the contents should not be scaled if there is no viewBox defined.

**Note:** If the application does not have the necessary privilege rights to access this (SVG) content, a `SecurityException` may be thrown by the underlying implementation. This is applicable to all the tree navigation (see `Node`) and trait accessor methods (see `SVGElement`). Features such as zooming, panning and playing of animations will not be affected.

Here is the list of events supported in this specification:

<table>
<thead>
<tr>
<th>Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“DOMFocusIn”</td>
<td>Occurs when the element comes into focus.</td>
</tr>
<tr>
<td>“DOMFocusOut”</td>
<td>Occurs when an element goes out of focus.</td>
</tr>
<tr>
<td>“DOMActivate”</td>
<td>Occurs when an element is activated, for instance, through a keypress (select)</td>
</tr>
<tr>
<td>“click”</td>
<td>Occurs when the pointing device button is clicked over the SVG content (or particular SVGElement)</td>
</tr>
</tbody>
</table>

**See Also:** `ScalableImage`, `org.w3c.dom.Document`
svg

Member Summary

Constructors

protected SVGImage()

Methods

void activate()

Inherited Member Summary

Methods inherited from class Object

clone(), equals(Object), finalize(), getClass(), hashCode(), notify(), notifyAll(), toString(), wait(long, int), wait(long, int), wait(long, int)

Methods inherited from class ScalableImage

createImage(String, ExternalResourceHandler), createImage(String, ExternalResourceHandler)

Constructors

SVGImage()

Declaration:

protected SVGImage()

Description:

Constructor

Methods

activate()

Declaration:

public void activate()
**createEmptyImage(ExternalResourceHandler)**

**Declaration:**
```
public static javax.microedition.m2g.SVGImage createEmptyImage(ExternalResourceHandler handler)
```

**Description:**
This method creates and loads an empty SVGImage (skeleton) that can be used to programatically construct a simple SVG image. Note: A root <svg> element with default viewport size of 100x100 is also created inside this method.

If an handler is specified, the engine will invoke it for any external resource referenced in the document. If the handler is set to null, the SVGImage will try to load images automatically using the engine default implementation, but might not be able to load all of them.

Note that the handler is also called when the xlink:href attribute on <image> is set or changed by the application, but the call is made only when the element is hooked into the document tree i.e. when the ancestors go all the way up to the root <svg> element. There are two cases:

- When changing the xlink:href attribute of an existing <image> element that is already hooked or part of the tree.
- When creating a new <image>, setting its xlink:href and hooking it to the document tree.

**Parameters:**
- **handler** - implementation of the ExternalResourceHandler interface

**Returns:**
an empty SVGImage

**See Also:** ExternalResourceHandler

**dispatchMouseEvent(String, int, int)**

**Declaration:**
```
public void dispatchMouseEvent(java.lang.String type, int x, int y)
    throws DOMException
```

**Description:**
This method is used to dispatch a mouse event of the specified type to the document. The mouse position is given as screen coordinates x, y. If the x, y values are outside the viewport area or no target is available for the x, y coordinates, the event is not dispatched. Note that when a “click” event is dispatched, a “DOMActivate” is automatically dispatched by the underlying implementation. The only required mouse event type is “click”. Therefore, if an unsupported type is specified, a DOMException with error code NOT_SUPPORTED_ERR is thrown.

**Parameters:**
- **type** - the type of mouse event.
- **x** - the x location of the mouse/pointer in viewport coordinate system.
- **y** - the y location of the mouse/pointer in viewport coordinate system.
focusOn(SVGElement)

Declaration:
public void focusOn(org.w3c.dom.svg.SVGElement element) throws DOMException

Description:
This method triggers a “DOMFocusIn” event with the specified element as the event target. That element becomes the element with focus. This method also triggers a “DOMFocusOut” event with the element which previous had focus as target. When the activate method is called, a “DOMActivate” event is triggered with the currently focused element as the target. The initial focus is always null and setting null will remove the current focus.

Parameters:
element - the element to set the focus on.

Throws:
org.w3c.dom.DOMException - with error code NOT_SUPPORTED_ERR: if the event type is not supported.
java.lang.NullPointerException - if type is null.
java.lang.IllegalArgumentException - if the x or y values are negative.

getDocument()

Declaration:
public org.w3c.dom.Document getDocument()

Description:
Returns the associated Document.

Returns: the associated Document.

getViewportHeight()

Declaration:
public int getViewportHeight()

Description:
This method returns the ScalableImage’s viewport height. The initial viewport height is taken from the “height” value specified in the Scalable Image. The value returned is always in pixels. If the specified height is defined in percentages, the values are mapped to the default view port size of 100x100. If the viewport height is explicitly changed by the application, then the percentages are ignored and the content is made to fit to this new viewport height.

Overrides: getViewportHeight in class ScalableImage

Returns: the current height of this ScalableImage.

See Also: setViewportHeight(int)
getViewportWidth()

Declaration:
public int getViewportWidth()

Description:
This method returns the ScalableImage’s viewport width. The initial viewport width is taken from the “width” value specified in the Scalable Image. The value returned is always in pixels. If the specified width is defined in percentages, the values are mapped to the default viewport size of 100x100. If the viewport width is explicitly changed by the application, then the percentages are ignored and the content is made to fit to this new viewport width.

Overrides: getViewportWidth in class ScalableImage

Returns: the current width of this ScalableImage.

See Also: setViewportWidth(int)

incrementTime(float)

Declaration:
public void incrementTime(float seconds)

Description:
Increments the animation or media timeline for this SVGImage (in seconds). As the name implies, this method is intended to move only forward in the timeline and typically should be used to animate SVG content when the SVGAnimator class is not used. Setting negative values will throw an Exception. It is important to note that setting large increments of time would result in skipping parts of the animation as per the SVG animation model.

Parameters:
seconds - the value of time to increment in seconds.

Throws:
java.lang.IllegalArgumentException - if the specified time is negative.

requestCompleted(String, InputStream)

Declaration:
public void requestCompleted(java.lang.String URI, java.io.InputStream resourceData) throws IOException

Description:
Once the requested external resource is available, the application forwards this information (resourceData) to the SVG engine. If this method is called a second time for a same URL of a same SVGImage, the engine will replace the current resource data with the new one. Note: Setting null for resourceData indicates that the requested resource could not be fetched by the ResourceHandler or application, and in this event the SVG engine will not make further attempts to load this resource.

Overrides: requestCompleted in class ScalableImage

Parameters:
URI - URI that was requested through requestExternalResource
resourceData - inputstream containing the external resource

Throws:
java.io.IOException - if an error occurs while loading the resource.
java.lang.NullPointerException - if URL is null.
See Also: ExternalResourceHandler

**setViewportHeight(int)**

Declaration:
```
public void setViewportHeight(int height)
```

Description:
This method sets the new (viewport) height of this ScalableImage.

Overrides: `setViewportHeight` in class `ScalableImage`

Parameters:
- `height`: the new height to be set.

Throws:
- `java.lang.IllegalArgumentException` - if `height` is negative.

See Also: `getViewportHeight()`

**setViewportWidth(int)**

Declaration:
```
public void setViewportWidth(int width)
```

Description:
This method sets the new (viewport) width of this ScalableImage. The viewport is the area where the ScalableImage is rendered. Any parts of the viewport that lie outside the boundaries of the target clipping rectangle are clipped. The viewport upper left corner (x, y) is given relative to the upper left corner of the target rendering surface.

Overrides: `setViewportWidth` in class `ScalableImage`

Parameters:
- `width`: the new width to be set.

Throws:
- `java.lang.IllegalArgumentException` - if `width` is negative.

See Also: `getViewportWidth()`
Package
org.w3c.dom

Description
This package is subsetted DOM Level 3 Core (http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/idl-definitions.html) APIs. DOM Level 3 Core is the normative specification for this package. The subset is designed to be appropriate for small footprint devices, yet retain compatibility to the full DOM Core implementations. This is achieved by following:

- Only two types of Node are supported: Element and Document and only a subset of methods is supported on each interface.
- All other types of nodes can be thought of still being in the tree, but not accessible through these subsetted APIs.
- Additional element tree traversal APIs are introduced on SVGElement interface

Note: The JSR 226 DOM APIs (in the org.w3c.dom, org.w3c.dom.events and org.w3c.dom.svg packages) are binary compatible with the SVG 1.2 Tiny DOM. This means that code compiled against the JSR 226 DOM APIs can run on a conformance implementation of the SVG 1.2 Tiny DOM.

Class Summary

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>This empty interface is defined to ensure compatibility with the DOM specification, and also to be used as a return type of some Document methods.</td>
</tr>
<tr>
<td>Node</td>
<td>The Node interface describes generic nodes in an SVG document tree.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exceptions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DOMException</td>
<td></td>
</tr>
</tbody>
</table>


Document

description

The Document interface represents an XML Document. This interface is a subset of the Document interface defined in the DOM Level 3 Core (http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/idl-definitions.html).

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

- getParentNode returns null
- appendChild throws HIERARCHY_REQUEST_ERR
- insertBefore throws HIERARCHY_REQUEST_ERR
- removeChild throws NOT_SUPPORTED_ERR

member summary

Methods

Element createElementNS(java.lang.String namespaceURI, java.lang.String qualifiedName)
Element getDocumentElement()
Element getElementById(java.lang.String id)

inherited member summary

Methods inherited from interface Node
appendChild(Node), getLocalName(), getNamespaceURI(), getParentNode(), insertBefore(Node, Node), removeChild(Node)
Methods

createElementNS(String, String)

Declaration:
public org.w3c.dom.Element createElementNS(String namespaceURI, String qualifiedName) throws DOMException

Description:
Create a new Element based on the specified (qualified) SVG tag name. This JSR does not require multiple namespaces and may throw a DOMException with a code of NOT_SUPPORTED_ERR if the URI is not the SVG namespace URI, or if the specified name is not a valid SVG Tiny element name. Only the following elements must be supported: &lt;rect&gt;, &lt;circle&gt;, &lt;ellipse&gt;, &lt;line&gt;, &lt;path&gt; &lt;use&gt; &lt;image&gt; &lt;text&gt;, &lt;a&gt; and &lt;g&gt;.

Parameters:
namespaceURI - the namespace uri for the newly created element. This should always be the SVG namespace URI “http://www.w3.org/2000/svg”.
qualifiedName - the qualified name for the newly created element (For example: “rect”, to create a &lt;rect&gt; element)

Returns: the newly created SVG Element.

Throws:
DOMException - NOT_SUPPORTED_ERR if the type of element is not supported by the implementation. JSR 226 only requires creation support for some of the SVG namespace elements and only for a limited number of local names in that namespace (see above documentation).Therefore, in a conformant JSR 226 implementation, trying to create elements with a namespace URIs other than the SVG namespace URI and with a qualified name not in the list of required qualified names may result in this exception being thrown.
java.lang.NullPointerException - if namespaceURI or qualifiedName is null.

getDocumentElement()

Declaration:
public org.w3c.dom.Element getDocumentElement()throws SecurityException

Description:
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.

Returns: the root Element associated with this document.

Throws:
java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

getElementById(String)

Declaration:
public org.w3c.dom.Element getElementsByTagName(String id)
Description:
Return the Element in the current document with the given unique ID. If no such element exists, this returns null.

Parameters:
id - the ID of the object to be retrieved.

Returns: the Element that matches with the given ID or null if the ID is not present.

Throws:
java.lang.NullPointerException - if id is null
java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.
## org.w3c.dom

### DOMException

**Declaration**

```java
public class DOMException extends java.lang.RuntimeException
```

**Tree Diagram**

```
java.lang.Object
    |-- java.lang.Throwable
    |    |-- java.lang.Exception
    |    |    |-- java.lang.RuntimeException
    |    |    `-- org.w3c.dom.DOMException
```

**All Implemented Interfaces:** java.io.Serializable

### Member Summary

<table>
<thead>
<tr>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>short code</td>
</tr>
<tr>
<td>static short HIERARCHY_REQUEST_ERR</td>
</tr>
<tr>
<td>static short INDEX_SIZE_ERR</td>
</tr>
<tr>
<td>static short INVALID_ACCESS_ERR</td>
</tr>
<tr>
<td>static short INVALID_MODIFICATION_ERR</td>
</tr>
<tr>
<td>static short INVALID_STATE_ERR</td>
</tr>
<tr>
<td>static short NO_MODIFICATION_ALLOWED_ERR</td>
</tr>
<tr>
<td>static short NOT_FOUND_ERR</td>
</tr>
<tr>
<td>static short NOT_SUPPORTED_ERR</td>
</tr>
<tr>
<td>static short TYPE_MISMATCH_ERR</td>
</tr>
<tr>
<td>static short WRONG_DOCUMENT_ERR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructors</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOMException(short code, java.lang.String message)</td>
</tr>
</tbody>
</table>

### Inherited Member Summary

**Methods inherited from class Object**

- clone(), equals(Object), finalize(), getClass(), hashCode(), notify(), notifyAll(), wait(long, int), wait(long, int), wait(long, int)

**Methods inherited from class Throwable**

- fillInStackTrace(), getCause(), getLocalizedMessage(), getMessage(), getStackTrace(), initCause(Throwable), printStackTrace(PrintWriter), printStackTrace(PrintWriter), printStackTrace(PrintWriter), setStackTrace(StackTraceElement[]), toString()
Fields

code

Declaration:
public short code

Description:
The member variable to store exception’s code, like INVALID_ACCESS_ERR.

HIERARCHY_REQUEST_ERR

Declaration:
public static final short HIERARCHY_REQUEST_ERR

Description:
If any Node is inserted somewhere it doesn’t belong.

INDEX_SIZE_ERR

Declaration:
public static final short INDEX_SIZE_ERR

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

INVALID_ACCESS_ERR

Declaration:
public static final short INVALID_ACCESS_ERR

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

INVALID_MODIFICATION_ERR

Declaration:
public static final short INVALID_MODIFICATION_ERR

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

INVALID_STATE_ERR

Declaration:
public static final short INVALID_STATE_ERR

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

NO_MODIFICATION_ALLOWED_ERR

Declaration:
public static final short NO_MODIFICATION_ALLOWED_ERR

Description:
If an attempt is made to modify an object where modifications are not allowed.
NOT_FOUND_ERR

Declaration:
public static final short NOT_FOUND_ERR

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

NOT_SUPPORTED_ERR

Declaration:
public static final short NOT_SUPPORTED_ERR

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

TYPE_MISMATCH_ERR

Declaration:
public static final short TYPE_MISMATCH_ERR

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

WRONG_DOCUMENT_ERR

Declaration:
public static final short WRONG_DOCUMENT_ERR

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

Constructors

DOMException(short, String)

Declaration:
public DOMException(short code, java.lang.String message)

Description:
Constructs a DOMException with a detailed message.

Parameters:
  code - the exception’s error code.
  message - the exception’s descriptive message.
org.w3c.dom

Element

Declaration
public interface Element extends Node

All Superinterfaces: Node

All Known Subinterfaces: org.w3c.dom.svg.SVGAnimationElement,
org.w3c.dom.svg.SVGELEMENT,
org.w3c.dom.svg.SVGLocatableElement,
org.w3c.dom.svg.SVGSVGELEMENT

Description
This empty interface is defined to ensure compatibility with the DOM specification, and also to be used as a
return type of some Document methods.

Inherited Member Summary

<table>
<thead>
<tr>
<th>Methods inherited from interface Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>appendChild(Node), getLocalName(), getNamespaceURI(), getParentNode(), insertBefore(Node, Node), removeChild(Node)</td>
</tr>
</tbody>
</table>
org.w3c.dom

Node

Declaration
public interface Node

All Known Subinterfaces: Document, Element,
org.w3c.dom.svg.SVGAElement,
org.w3c.dom.svg.SVGElement,
org.w3c.dom.svg.SVGLocatableElement,
org.w3c.dom.svg.SVGSVGElement

Description
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 (http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/idl-definitions.html).

Member Summary

Methods
Node appendChild(Node newChild)
java.lang.String getLocalName()
java.lang.String getNamespaceURI()
Node getParentNode()
Node insertBefore(Node newChild, Node refChild)
Node removeChild(Node oldChild)

Methods

appendChild(Node)

Declaration:
public org.w3c.dom.Node appendChild(org.w3c.dom.Node newChild)
throws DOMException

Description:
Appends a child to this Node.

Parameters:
newChild - the Node to be appended to this Node. This is equivalent to insertBefore(newChild,null)

Returns: the added Node.

Throws:
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.
getLocalName()  
**Declaration:**  
public java.lang.String getLocalName()  

**Description:**  
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.

**Returns:**  
the local part of the qualified name of this node.

**Throws:**  
java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

getNamespaceURI()  
**Declaration:**  
public java.lang.String getNamespaceURI()  

**Description:**  
Returns the namespace URI of the Node.

**Returns:**  
the namespace URI of the Node.

**Throws:**  
java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

getParentNode()  
**Declaration:**  
public org.w3c.dom.Node getParentNode()  

**Description:**  
Returns the parent Node of this Node.

**Returns:**  
the parent node or null if there is no parent (i.e. if a node has just been created and not yet added to the tree, or if it has been removed from the tree, this is null).

**Throws:**  
java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.
insertBefore(Node, Node)

Declaration:
public org.w3c.dom.Node insertBefore(Node newChild, Node refChild) throws DOMException

Description:
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.

Returns: the node being inserted.

Throws:
- 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, for ex: when the newChild contains a <use> element with an invalid xlink:href attribute.
- java.lang.NullPointerException - if newChild is null.
- java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

removeChild(Node)

Declaration:
public org.w3c.dom.Node removeChild(Node oldChild) throws DOMException

Description:
Removes the specified child associated with this Node. Elements that have ids cannot be removed from the tree.

Parameters:
- oldChild - the Node that is to be removed.

Returns: the node removed.

Throws:
- DOMException - with error code NOT_FOUND_ERR: Raised if oldChild is not a child of this node.
Node

org.w3c.dom

removeChild(Node)

DOMException 45 - 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 createElementNS.

DOMException 45 - with error code INVALID_ACCESS_ERR: if the element being removed or one of its descendants have non-null id.

java.lang.NullPointerException - if oldChild is null.

java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.
Package

org.w3c.dom.events

Description
This package contains the necessary event handling interfaces that are a subset of DOM Events Level 2 (http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/) APIs.

Note: The JSR 226 DOM APIs (in the org.w3c.dom, org.w3c.dom.events and org.w3c.dom.svg packages) are binary compatible with the SVG 1.2 Tiny DOM. This means that code compiled against the JSR 226 DOM APIs can run on a conformance implementation of the SVG 1.2 Tiny DOM.

Class Summary

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Event</code></td>
<td>The Event interface is used to provide contextual information about an event to the handler processing the event.</td>
</tr>
<tr>
<td><code>EventListener</code></td>
<td>This interface represents an event listener, and is a subset of the EventListener interface defined in the DOM Level 2 Event model (<a href="http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/idl-definitions.html">http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/idl-definitions.html</a>).</td>
</tr>
<tr>
<td><code>EventTarget</code></td>
<td>This interface represents an event target, and is a subset of the EventTarget interface defined in the DOM Level 2 Event model (<a href="http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/idl-definitions.html">http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/idl-definitions.html</a>).</td>
</tr>
</tbody>
</table>
Event

public interface Event

Description
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 handleEvent call. If an event target is an element instance (see SVGElementInstance (http://www.w3.org/TR/SVG11/struct.html#InterfaceSVGElementInstance)), the currentTarget is an implementation of EventTarget that does not implement the Node interface.

Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventTarget getCurrentTarget()</td>
</tr>
<tr>
<td>java.lang.String getType()</td>
</tr>
</tbody>
</table>

Methods

gCurrentTarget()

Declaration:
public org.w3c.dom.events.EventTarget getCurrentTarget()

Description:
This method is used to get the current target of this event. In SVG Tiny, this is always an object to which event listener was attached.

Returns: the event’s EventTarget.

gType()

Declaration:
public java.lang.String getType()

Description:
This method returns the event type information. The name of the event is case-sensitive. The following event types are supported:
- click, DOMActivate, DOMFocusIn, DOMFocusOut (defined DOM Events);

Returns: the event’s type.
EventListener

**Declaration**

```java
public interface EventListener
```

**Description**

This interface represents an event listener, and is a subset of the EventListener interface defined in the DOM Level 2 Event model (http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/idl-definitions.html).

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

**Member Summary**

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void handleEvent(Event evt)</code></td>
</tr>
</tbody>
</table>

**Methods**

**handleEvent( Event)  **

**Declaration:**

```java
public void handleEvent(org.w3c.dom.events.Event evt)
```

**Description:**

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.
EventTarget

org.w3c.dom.events

addEventListener(String, EventListener, boolean)

org.w3c.dom.events

EventTarget

Declaration

public interface EventTarget

All Known Subinterfaces:

org.w3c.dom.svg.SVGAnimationElement,
org.w3c.dom.svg.SVGEElement,
org.w3c.dom.svg.SVGLocatableElement,
org.w3c.dom.svg.SVGElement

Description

This interface represents an event target, and is a subset of the EventTarget interface defined in the DOM Level 2 Event model (http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/idl-definitions.html). This interface is implemented by an object (SVGElements) that can notify listeners about events and allows registration and removal of EventListener objects.

Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addEventListener(java.lang.String type, EventListener listener, boolean useCapture)</td>
</tr>
<tr>
<td>void removeEventListener(java.lang.String type, EventListener listener, boolean useCapture)</td>
</tr>
</tbody>
</table>

Methods

addEventListener(String, EventListener, boolean)

Declaration:

public void addEventListener(java.lang.String type, org.w3c.dom.events.EventListener listener, boolean useCapture)

Description:

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 removeEventListener method.

Parameters:

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 - If true, the listener will be called during the event flow capture phase. Otherwise, the listener will be called during the bubble phase. If the event’s target is this target, then the listener will be called during the ‘at target’ phase of event flow.

Throws:
org.w3c.dom.DOMException\_45 - with error code NOT\_SUPPORTED\_ERR if useCapture is true since capture phase is not supported in SVG Tiny.
java.lang.NullPointerException - if listener is null.
java.lang.NullPointerException - if type is null.

removeEventListener(String, EventListener, boolean)

Declaration:
public void removeEventListener(java.lang.String type,
org.w3c.dom.events.EventListener\_55 listener, boolean useCapture)

Description:
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 removeEventListener with arguments which do not identify any currently registered EventListener on the EventTarget has no effect.

Parameters:
type - The type of event that was listened to.
listener - The listener that was previously registered.
useCapture - If true, the listener was listening to events in the capture phase of event flow, otherwise the listener was listening to events in the bubble phase.

Throws:
org.w3c.dom.DOMException\_45 - with error code NOT\_SUPPORTED\_ERR if useCapture is true since capture phase is not supported in SVG Tiny.
java.lang.NullPointerException - if listener is null.
java.lang.NullPointerException - if type is null.
EventTarget
org.w3c.dom.events
removeEventListener(String, EventListener, boolean)
Package
org.w3c.dom.svg

Description
The SVG Tiny 1.1 DOM API defined in this specification is subset of W3C SVG Tiny 1.2 uDOM (http://www.w3.org/TR/SVG12/svgudom.html).

Note: The JSR 226 DOM APIs (in the org.w3c.dom, org.w3c.dom.events and org.w3c.dom.svg packages) are binary compatible with the SVG 1.2 Tiny DOM. This means that code compiled against the JSR 226 DOM APIs can run on a conformance implementation of the SVG 1.2 Tiny DOM.

### Class Summary

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVGAnimationElement</td>
<td>This interface represents an Animation element, which contains methods to control the timing of animations.</td>
</tr>
<tr>
<td>SVGElemetn</td>
<td>This interface represents an SVG element in the document tree.</td>
</tr>
<tr>
<td>SVGLocatableElement</td>
<td>This interface represents an SVGLocatableElement.</td>
</tr>
<tr>
<td>SVGMatrix</td>
<td>This interface represents an “SVGMatrix” datatype, identified by an affine transform.</td>
</tr>
<tr>
<td>SVGPath</td>
<td>This interface represents an “SVGPath” datatype used to define the path geometry.</td>
</tr>
<tr>
<td>SVGPoint</td>
<td>This interface represents an “SVGPoint” datatype, identified by its x and y components.</td>
</tr>
<tr>
<td>SVGRect</td>
<td>This interface represents an “SVGRect” datatype, consisting of a minimum X, minimum Y, width and height values.</td>
</tr>
<tr>
<td>SVGRGBColor</td>
<td>This interface represents an “SVGRGBColor” datatype made up of red, green, and blue components.</td>
</tr>
<tr>
<td>SVGSVGElement</td>
<td>This interface represents &lt;svg&gt; element in (SVG) document tree.</td>
</tr>
</tbody>
</table>

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVGException</td>
<td>An exception thrown for SVG-specific errors, such as noninvertable matrix in inverse.</td>
</tr>
</tbody>
</table>
**Declaration**

```java
public interface SVGAnimationElement extends SVGElement
```

**All Superinterfaces:** org.w3c.dom.Element, org.w3c.dom.events.EventTarget, org.w3c.dom.Node, SVGElement

**Description**

This interface represents an Animation element, which contains methods to control the timing of animations.

**Member Summary**

**Methods**

- `void beginElementAt(float offset)`
- `void endElementAt(float offset)`

**Inherited Member Summary**

**Methods inherited from interface EventTarget**

- `addEventListener(String, EventListener, boolean)`
- `removeEventListener(String, EventListener, boolean)`

**Methods inherited from interface Node**

- `appendChild(Node)`
- `getLocalName()`
- `getNamespaceURI()`
- `getParentNode()`
- `insertBefore(Node, Node)`
- `removeChild(Node)`

**Methods inherited from interface SVGElement**

- `getFirstElementChild()`
- `getFloatTrait(String)`
- `getId()`
- `getMatrixTrait(String)`
- `getNextElementSibling()`
- `getPathTrait(String)`
- `getRGBColorTrait(String)`
- `getRectTrait(String)`
- `getTrait(String)`
- `getTraitNS(String, String)`
- `setFloatTrait(String, float)`
- `setId(String)`
- `setMatrixTrait(String, SVGMatrix)`
- `setPathTrait(String, SVGPath)`
- `setRGBColorTrait(String, SVGRGBColor)`
- `setRectTrait(String, SVGRect)`
- `setTrait(String, String)`
- `setTraitNS(String, String, String)`
Methods

beginElementAt(float)

Declaration:
public void beginElementAt(float offset)

Description:
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 (http://www.w3.org/TR/2001/REC-smil20-20010807/ smil20.html#smil-timing-Timing-BeginEnd-InstanceTimesLists).

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

endElementAt(float)

Declaration:
public void endElementAt(float offset)

Description:
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 (http://www.w3.org/TR/2001/REC-smil20-20010807/ smil20.html#smil-timing-Timing-BeginEnd-InstanceTimesLists).

Parameters:
offset - The offset in seconds at which to end the element.
SVGElemment
endElementAt(float)

org.w3c.dom.svg

SVGElemment

Declaration
public interface SVGElemment extends org.w3c.dom.Element, org.w3c.dom.events.EventTarget

All Superinterfaces: org.w3c.dom.Element, org.w3c.dom.events.EventTarget,
org.w3c.dom.Node

All Known Subinterfaces: SVGAnimationElement, SVGLocatableElement,
SVGSVGElement

Description
This interface represents an SVG element in the document tree. Element’s id can be set only if it does not
already have an id. DOMException with error code NO_MODIFICATION_ALLOWED_ERR is raised if
an attempt is made to change an existing id. Elements with non-null id can be inserted, but cannot be removed
from the DOM tree (see removeChild). 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 SVGElemment. 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).

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
first situation is when the trait value is invalid with regards to its definition (for example, trying to set the
“stroke-linejoin” trait to “foo” would cause this exception). 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 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.
Traits supported in this specification, SVG Tiny 1.1 DOM

The table below shows the list of attributes and properties that SVG Tiny DOM 1.1 implementations must support. Each light gray section lists one or multiple elements for which the subsequent attributes or properties apply. Each attribute row lists the allowed getter and setter(s). The last column specifies the default values that must be used for each attribute or property.

**Note:** 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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Trait Getter [possible return value(s)]</th>
<th>Trait Setter [allowed value(s)]</th>
<th>Default Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>getRGBColorTrait [SVGRGBColor]</td>
<td>setTrait [inherit]</td>
<td>rgb(0,0,0)</td>
</tr>
<tr>
<td>display</td>
<td>getTrait [inline</td>
<td>none ]</td>
<td>setTrait [inline</td>
</tr>
<tr>
<td>fill</td>
<td>getRGBColorTrait [null, SVGRGBColor]</td>
<td>setRGBColorTrait [SVGRGBColor]</td>
<td>rgb(0,0,0)</td>
</tr>
<tr>
<td>fill-rule</td>
<td>getTrait [nonzero</td>
<td>evenodd]</td>
<td>setTrait [nonzero</td>
</tr>
<tr>
<td>stroke-dashoffset</td>
<td>getFloatTrait</td>
<td>setTrait [inherit]</td>
<td>0.0f</td>
</tr>
<tr>
<td>stroke-linecap</td>
<td>getTrait [butt</td>
<td>round</td>
<td>square]</td>
</tr>
<tr>
<td>stroke-linejoin</td>
<td>getTrait [miter</td>
<td>round</td>
<td>bevel ]</td>
</tr>
<tr>
<td>stroke-miterlimit</td>
<td>getFloatTrait [ value &gt;= 1]</td>
<td>setTrait [inherit]</td>
<td>4.0f</td>
</tr>
<tr>
<td>stroke-width</td>
<td>getFloatTrait [value &gt;= 0]</td>
<td>setTrait [inherit]</td>
<td>1.0f</td>
</tr>
<tr>
<td>visibility</td>
<td>getTrait [visible</td>
<td>hidden]</td>
<td>setTrait [visible</td>
</tr>
</tbody>
</table>
### Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Trait Getter</th>
<th>Trait Setter</th>
<th>Default Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>font-family</td>
<td>getTrait</td>
<td>setTrait</td>
<td>[single, computed font-family value] [same syntax as font-family attribute] User-Agent</td>
</tr>
<tr>
<td>font-size</td>
<td>getFloatTrait [value &gt;= 0]</td>
<td>setFloatTrait [value &gt;= 0]</td>
<td>setTrait [inherit] User-Agent</td>
</tr>
<tr>
<td>font-style</td>
<td>getTrait</td>
<td>setTrait</td>
<td>[normal</td>
</tr>
<tr>
<td>font-weight</td>
<td>getTrait</td>
<td>setTrait</td>
<td>[100</td>
</tr>
<tr>
<td>text-anchor</td>
<td>getTrait</td>
<td>setTrait</td>
<td>[start</td>
</tr>
<tr>
<td>transform</td>
<td>getMatrixTrait [SVGMatrix]</td>
<td>setMatrixTrait [SVGMatrix]</td>
<td>Identity matrix (1,0,0,1,0,0)</td>
</tr>
<tr>
<td>height</td>
<td>getFloatTrait [value &gt;= 0]</td>
<td>setFloatTrait [value &gt;= 0]</td>
<td>REQUIRED (0.0f)</td>
</tr>
<tr>
<td>width</td>
<td>getFloatTrait [value &gt;= 0]</td>
<td>setFloatTrait [value &gt;= 0]</td>
<td>REQUIRED (0.0f)</td>
</tr>
<tr>
<td>x</td>
<td>getFloatTrait</td>
<td>setFloatTrait</td>
<td>0.0f</td>
</tr>
<tr>
<td>y</td>
<td>getFloatTrait</td>
<td>setFloatTrait</td>
<td>0.0f</td>
</tr>
<tr>
<td>rx</td>
<td>getFloatTrait [value &gt;= 0]</td>
<td>setFloatTrait [value &gt;= 0]</td>
<td>0.0f</td>
</tr>
<tr>
<td>ry</td>
<td>getFloatTrait [value &gt;= 0]</td>
<td>setFloatTrait [value &gt;= 0]</td>
<td>0.0f</td>
</tr>
</tbody>
</table>
<circle>
<table>
<thead>
<tr>
<th>Trait</th>
<th>getFloatTrait</th>
<th>setFloatTrait</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cx</td>
<td></td>
<td></td>
<td>0.0f</td>
</tr>
<tr>
<td>cy</td>
<td></td>
<td></td>
<td>0.0f</td>
</tr>
<tr>
<td>r</td>
<td>getFloatTrait [value &gt;= 0]</td>
<td>setFloatTrait [value &gt;= 0]</td>
<td>REQUIRED (0.0f)</td>
</tr>
</tbody>
</table>

<ellipse>
<table>
<thead>
<tr>
<th>Trait</th>
<th>getFloatTrait</th>
<th>setFloatTrait</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cx</td>
<td></td>
<td></td>
<td>0.0f</td>
</tr>
<tr>
<td>cy</td>
<td></td>
<td></td>
<td>0.0f</td>
</tr>
<tr>
<td>rx</td>
<td>getFloatTrait [value &gt;= 0]</td>
<td>setFloatTrait [value &gt;= 0]</td>
<td>REQUIRED (0.0f)</td>
</tr>
<tr>
<td>ry</td>
<td>getFloatTrait [value &gt;= 0]</td>
<td>setFloatTrait [value &gt;= 0]</td>
<td>REQUIRED (0.0f)</td>
</tr>
</tbody>
</table>

<line>
<table>
<thead>
<tr>
<th>Trait</th>
<th>getFloatTrait</th>
<th>setFloatTrait</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td></td>
<td></td>
<td>0.0f</td>
</tr>
<tr>
<td>x2</td>
<td></td>
<td></td>
<td>0.0f</td>
</tr>
<tr>
<td>y1</td>
<td></td>
<td></td>
<td>0.0f</td>
</tr>
<tr>
<td>y2</td>
<td></td>
<td></td>
<td>0.0f</td>
</tr>
</tbody>
</table>

<path> (path-length is not supported)
<table>
<thead>
<tr>
<th>Trait</th>
<th>getPathTrait [SVGPath]</th>
<th>setPathTrait [SVGPath]</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td></td>
<td></td>
<td>REQUIRED (Empty SVGPath)</td>
</tr>
<tr>
<td>Element</td>
<td>Trait</td>
<td>Getter</td>
<td>Setter</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>image</td>
<td>x</td>
<td>getFloatTrait</td>
<td>setFloatTrait</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>getFloatTrait</td>
<td>setFloatTrait</td>
</tr>
<tr>
<td></td>
<td>width</td>
<td>getFloatTrait [value &gt;= 0]</td>
<td>setFloatTrait [value &gt;= 0]</td>
</tr>
<tr>
<td></td>
<td>height</td>
<td>getFloatTrait [value &gt;= 0]</td>
<td>setFloatTrait [value &gt;= 0]</td>
</tr>
<tr>
<td></td>
<td>xlink:href</td>
<td>getTrait NS[absolute URI]</td>
<td>setTraitNS [non local-URI value]</td>
</tr>
</tbody>
</table>

| use     | x     | getFloatTrait | setFloatTrait | 0.0f |
|         | y     | getFloatTrait | setFloatTrait | 0.0f |
|         | xlink:href | getTraitNS[absolute URI] | setTraitNS | "" |

| a       | target | getTrait | setTrait | "" |
|         | xlink:href | getTraitNS[absolute URI] | setTraitNS | "" |

| text     | x     | getFloatTrait | setFloatTrait | 0.0f |

(Notes: For 'x' and 'y', it is only possible to provide floating point scalar values; an array of x or y values is not supported. 'rotate' attribute is not supported.)
### org.w3c.dom.svg

<table>
<thead>
<tr>
<th>y</th>
<th>getFloatTrait</th>
<th>setFloatTrait</th>
<th>0.0f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>getTrait [not null]</td>
<td>setTrait [not null]</td>
<td>&quot;&quot;</td>
</tr>
</tbody>
</table>

```
<svg>

version | getTrait | Not available (readonly) | “1.1” |
baseProfile | getTrait | Not available (readonly) | “tiny” |
viewBox | getRectTrait [null, SVGRect] | setRectTrait [SVGRect] | null |
zoomAndPan | getTrait [disable | magnify] | setTrait [disable | magnify] | “magnify” |

```

### Member Summary

**Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.w3c.dom.Element</td>
<td>getFirstChild()</td>
</tr>
<tr>
<td>float</td>
<td>getFloatTrait(java.lang.String name)</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>getId()</td>
</tr>
<tr>
<td>SVGMatrix</td>
<td>getMatrixTrait(java.lang.String name)</td>
</tr>
<tr>
<td>org.w3c.dom.Element</td>
<td>getNextElementSibling()</td>
</tr>
<tr>
<td>SVGPath</td>
<td>getPathTrait(java.lang.String name)</td>
</tr>
<tr>
<td>SVGRect</td>
<td>getRectTrait(java.lang.String name)</td>
</tr>
<tr>
<td>SVGRGBColor</td>
<td>getRGBColorTrait(java.lang.String name)</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>getTrait(java.lang.String name)</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>getTraitNS(java.lang.String namespaceURI, java.lang.String name)</td>
</tr>
<tr>
<td>void</td>
<td>setFloatTrait(java.lang.String name, float value)</td>
</tr>
<tr>
<td>void</td>
<td>setId(java.lang.String Id)</td>
</tr>
<tr>
<td>void</td>
<td>setMatrixTrait(java.lang.String name, SVGMatrix matrix)</td>
</tr>
<tr>
<td>void</td>
<td>setPathTrait(java.lang.String name, SVGPath path)</td>
</tr>
<tr>
<td>void</td>
<td>setRectTrait(java.lang.String name, SVGRect rect)</td>
</tr>
<tr>
<td>void</td>
<td>setRGBColorTrait(java.lang.String name, SVGRGBColor color)</td>
</tr>
<tr>
<td>void</td>
<td>setTrait(java.lang.String name, java.lang.String value)</td>
</tr>
<tr>
<td>void</td>
<td>setTraitNS(java.lang.String namespaceURI, java.lang.String name, java.lang.String value)</td>
</tr>
</tbody>
</table>
**SVGElement**

**getFirstChild()**

Methods

**getFirstChild()**

Declarations:

```java
public org.w3c.dom.Element getFirstChild()
```

Description:

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

Returns:

the first child element node of this element.

**getFloatTrait(String)**

Declarations:

```java
public float getFloatTrait(String name) throws DOMException
```

Description:

Get the trait value as float.

Parameters:

- `name` - the name of the trait to retrieve.

Returns:

the trait value as float for the specified name.

Throws:

- `org.w3c.dom.DOMException` - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- `org.w3c.dom.DOMException` - with error code TYPE_MISMATCH_ERR if requested trait’s computed value cannot be converted to a float.
- `java.lang.SecurityException` - if the application does not have the necessary privilege rights to access this (SVG) content.

**getId()**

Declarations:

```java
public java.lang.String getId()
```

Description:

Returns the Element’s Id, null if no id specified.

Returns:

the Element’s Id.
getMatrixTrait(String)

Declaration:
public org.w3c.dom.svg.SVGMatrix getMatrixTrait(java.lang.String name) throws DOMException

Description:
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.

Returns: the trait value as SVGMatrix for the specified name.

Throws:
org.w3c.dom.DOMException - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
org.w3c.dom.DOMException - with error code TYPE_MISMATCH_ERR if requested trait’s computed value cannot be converted to SVGMatrix.
java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

getNextElementSibling()

Declaration:
public org.w3c.dom.Element getNextElementSibling()

Description:
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.

Returns: the next sibling element node of this element.

getPathTrait(String)

Declaration:
public org.w3c.dom.svg.SVGPath getPathTrait(java.lang.String name) throws DOMException

Description:
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.

Returns: the trait value as SVGPath for the specified name.

Throws:
org.w3c.dom.DOMException - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
org.w3c.dom.DOMException - with error code TYPE_MISMATCH_ERR if requested trait’s computed value cannot be converted to SVGPath.
java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.
getRectTrait(String)

Declaration:
public org.w3c.dom.svg.SVGRGBColor getRectTrait(java.lang.String name)

Description:
Returns the trait value as SVGRGBColor. 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.

Returns: the trait value as SVGRGBColor for the specified name.

Throws:
org.w3c.dom.DOMException - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
org.w3c.dom.DOMException - with error code TYPE_MISMATCH_ERR if requested trait’s computed value cannot be converted to SVGRGBColor.
java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

getRGBColorTrait(String)

Declaration:
public org.w3c.dom.svg.SVGRGBColor getRGBColorTrait(java.lang.String name)

Description:
Returns the trait value as SVGRGBColor. The returned object is a copy of the actual 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.

Returns: the trait value as SVGRGBColor for the specified name.

Throws:
org.w3c.dom.DOMException - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
org.w3c.dom.DOMException - with error code TYPE_MISMATCH_ERR if requested trait’s computed value cannot be converted to SVGRGBColor.
java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

getTrait(String)

Declaration:
public java.lang.String getTrait(java.lang.String name)

Throws: DOMException
Description:
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 `getTraitNS(null, String)` with namespaceURI set to null.

Parameters:
- name - the name of the trait to retrieve.

Returns: the trait value as String for the specified name.

Throws:
- `org.w3c.dom.DOMException` - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- `org.w3c.dom.DOMException` - with error code TYPE_MISMATCH_ERR if requested trait’s computed value cannot be converted to a String (SVG Tiny only).
- `java.lang.SecurityException` - if the application does not have the necessary privilege rights to access this (SVG) content.

get TraitNS(String, String)

Declaration:
```java
public java.lang.String getTraitNS(java.lang.String namespaceURI, java.lang.String name)
```

Description:
Same as `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.

Returns: the trait value as String for the specified name.

Throws:
- `org.w3c.dom.DOMException` - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- `org.w3c.dom.DOMException` - with error code TYPE_MISMATCH_ERR if requested trait’s computed value cannot be converted to a String (SVG Tiny only).
- `java.lang.SecurityException` - if the application does not have the necessary privilege rights to access this (SVG) content.

set Float Trait(String, float)

Declaration:
```java
public void setFloatTrait(java.lang.String name, float value)
```

Description:
Set the trait value as float.

Parameters:
- name - the name of the trait to set.
- value - the value of the trait to set as float.
**setId(String)**

**Declaration:**
```java
public void setId(java.lang.String Id)
```

**Description:**
Sets the Element’s id attribute.

**Parameters:**
- `Id` - the value of Id to be set for this Element.

**Throws:**
- `org.w3c.dom.DOMException45` - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element.
- `org.w3c.dom.DOMException45` - with error code TYPE_MISMATCH_ERR if the requested trait’s value cannot be specified as a float.
- `org.w3c.dom.DOMException45` - with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait.
- `java.lang.SecurityException` - if the application does not have the necessary privilege rights to access this (SVG) content.

**setMatrixTrait(String, SVGMatrix)**

**Declaration:**
```java
public void setMatrixTrait(java.lang.String name, org.w3c.dom.svg.SVGMatrix matrix)
```

**Description:**
Set the trait value as `SVGMatrix83`. 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.
- `matrix` - the value of the trait to be set as `SVGMatrix`.

**Throws:**
- `org.w3c.dom.DOMException45` - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- `org.w3c.dom.DOMException45` - with error code TYPE_MISMATCH_ERR if the requested trait’s value cannot be specified as an `SVGMatrix83`.
org.w3c.dom.svg

SVGElement

setPathTrait(String, SVGPath)

org.w3c.dom.DOMException - with error code INVALID_ACCESS_ERR if the input matrix value is null.

java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

setPathTrait(String, SVGPath)

Declaration:

public void setPathTrait(java.lang.String name, org.w3c.dom.svg.SVGPath path)

throws DOMException

Description:

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.

path - the value of the trait to be set as SVGPath.

Throws:

org.w3c.dom.DOMException - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.

org.w3c.dom.DOMException - with error code TYPE_MISMATCH_ERR if the requested trait’s value cannot be specified as an SVGPath.

org.w3c.dom.DOMException - with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait or null. SVGPath is invalid if it begins with any segment other than MOVE_TO segment. Note that an empty SVGPath is still a valid value.

java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

setRectTrait(String, SVGRect)

Declaration:

public void setRectTrait(java.lang.String name, org.w3c.dom.svg.SVGRect rect)

throws DOMException

Description:

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.

rect - the value of the trait to be set as SVGRect.

Throws:

org.w3c.dom.DOMException - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.

org.w3c.dom.DOMException - with error code TYPE_MISMATCH_ERR if the requested trait’s value cannot be specified as an SVGRect.

org.w3c.dom.DOMException - with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait or null. SVGRect is invalid if the width or height values are set to negative.
setRGBColorTrait(String, SVGRGBColor)

    java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

setRGBColorTrait(String, SVGRGBColor)

Declaration:
public void setRGBColorTrait(java.lang.String name, org.w3c.dom.svg.SVGRGBColor color) throws DOMException

Description:
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.

color - the value of the trait to be set as SVGRGBColor.

Throws:
org.w3c.dom.DOMException - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
org.w3c.dom.DOMException - with error code TYPE_MISMATCH_ERR if the requested trait’s value cannot be specified as an SVGRGBColor.
org.w3c.dom.DOMException - with error code INVALID_ACCESS_ERR if the input value is null.
java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

setTrait(String, String)

Declaration:
public void setTrait(java.lang.String name, java.lang.String value) throws DOMException

Description:
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 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.

Throws:
org.w3c.dom.DOMException - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
org.w3c.dom.DOMException - with error code TYPE_MISMATCH_ERR if the requested trait’s value cannot be specified as a String
org.w3c.dom.DOMException - with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait or null.
org.w3c.dom.DOMException - with error code NO_MODIFICATION_ALLOWED_ERR: if attempt is made to change readonly trait.
java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.

**setTraitNS(String, String, String)**

**Declaration:**
```
public void setTraitNS(java.lang.String namespaceURI, java.lang.String name, java.lang.String value)
throws DOMException
```

**Description:**
Same as 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 String.

**Throws:**
- org.w3c.dom.DOMException - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.
- org.w3c.dom.DOMException - with error code TYPE_MISMATCH_ERR if the requested trait’s value cannot be specified as a String.
- org.w3c.dom.DOMException - with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait or null. 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.
- org.w3c.dom.DOMException - with error code NO_MODIFICATION_ALLOWED_ERR: if attempt is made to change readonly trait.
- java.lang.SecurityException - if the application does not have the necessary privilege rights to access this (SVG) content.
SVGException

org.w3c.dom.svg

Declaration

public class SVGException extends java.lang.RuntimeException

java.lang.Object
   |-- java.lang.Throwable
   |   |-- java.lang.Exception
   |       |-- java.lang.RuntimeException
   |           |-- org.w3c.dom.svg.SVGException

All Implemented Interfaces: java.io.Serializable

Description

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

Member Summary

Fields

short code
static short SVG_INVALID_VALUE_ERR
static short SVG_MATRIX_NOT_INVERTABLE

Constructors

SVGException(short code, java.lang.String message)

Inherited Member Summary

Methods inherited from class Object
clone(), equals(Object), finalize(), getClass(), hashCode(), notify(), notifyAll(),
wait(long, int), wait(long, int), wait(long, int)

Methods inherited from class Throwable
fillInStackTrace(), getCause(), getLocalizedMessage(), getMessage(), getStackTrace(),
initCause(Throwable), printStackTrace(PrintWriter), printStackTrace(PrintWriter),
printStackTrace(PrintWriter), setStackTrace(StackTraceElement[]), toString()
Fields

code

Declaration:
public short code

Description:
An integer indicating the type of error generated.

SVG_INVALID_VALUE_ERR

Declaration:
public static final short SVG_INVALID_VALUE_ERR

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

SVG_MATRIX_NOT_INVERTABLE

Declaration:
public static final short SVG_MATRIX_NOT_INVERTABLE

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

Constructors

SVGException(short, String)

Declaration:
public SVGException(short code, java.lang.String message)

Description:
Constructs a SVGException with a detailed message.

Parameters:
  code - the exception's error code.
  message - the exception's descriptive message.
org.w3c.dom.svg
SVGLocatableElement

Declaration
public interface SVGLocatableElement extends SVGElement

All Superinterfaces: org.w3c.dom.Element, org.w3c.dom.events.EventTarget, org.w3c.dom.Node, SVGElement

All Known Subinterfaces: SVGSVGElement

Description
This interface represents an SVGLocatableElement. It is implemented by all drawable SVG elements in the document tree. Drawable elements are: <rect>, <circle>, <ellipse>, <line>, <path> <use> <image> <text>, <svg>, <a>, and <g>. Note that 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>
```

```
[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}
```

```
<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>
```

```
[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.

```
<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"/>
    <ellipse id="ellipse1" cx="20" cy="20" rx="0" ry="70"/>
  </g>
</svg>
```

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>
```

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>
  <g id="group2" transform="translate(10, 20)">
    <rect id="rect2" x="0" y="10" width="150" height="0"/>
  </g>
</svg>
```
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.

```xml
<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>
```

[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.

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

[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.

```xml
<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>
```

[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}

80
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.

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

[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.

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

[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.

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

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

- getFirstElementChild()
- getFloatTrait(String)
- getId()
- getMatrixTrait(String)
- getNextElementSibling()
- getPathTrait(String)
- getRGBColorTrait(String)
- getRectTrait(String)
- getTrait(String)
- getTraitNS(String, String)
- setFloatTrait(String, float)
- setId(String)
- setMatrixTrait(String, SVGMatrix)
- setPathTrait(String, SVGPath)
- setRGBColorTrait(String, SVGRGBColor)
- setRectTrait(String, SVGRect)
- setTrait(String, String)
- setTraitNS(String, String, String)

Methods

getBBox()

Declaration:
public org.w3c.dom.svg.SVGRect getBBox()

Description:
Returns the tight bounding box in current user coordinate space. 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 (http://www.w3.org/TR/SVG/coords.html#ObjectBoundingBox).

Returns: the tight bounding box in current user coordinate space.

getScreenBBox()

Declaration:
public org.w3c.dom.svg.SVGRect getScreenBBox()

Description:
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 getScreenCTM method. Note that null is returned if this element is not hooked into the document tree.

Returns: the tight bounding box in screen coordinate space.

getScreenCTM()

Declaration:
public org.w3c.dom.svg.SVGMatrix getScreenCTM()

Description:
Returns the transformation matrix from current user units (i.e., after application of the transform attribute, if any) 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.

Returns: the transformation matrix from current user units to the parent user agent’s notion of a “pixel”.

82
SVGMatrix

Declaration
public interface SVGMatrix

Description
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, 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:

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  a & c & e \\
  b & d & f \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} =
\begin{bmatrix}
a.x + c.y + e \\
b.x + d.y + f \\
1
\end{bmatrix}
\]

Methods

getComponent(int)

Declaration:
public float getComponent(int index)
throws DOMException

Description:
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.

Returns: the component for the specified index.

Throws:
org.w3c.dom.DOMException - INDEX_SIZE_ERR if the index is invalid.
inverse()  

Declaration:
public org.w3c.dom.svg.SVGMatrix\_inverse() 
     throws SVGException

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

Returns: the inverse of the current matrix.

Throws:
   SVGException\_76 -- SVG\_MATRIX\_NOT\_INVERTABLE when determinant of this matrix is zero.

mMultiply(SVGMatrix)

Declaration:
public org.w3c.dom.svg.SVGMatrix\_mMultiply\(org.w3c.dom.svg.SVGMatrix\_secondMatrix\)

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

Parameters:
   secondMatrix - the matrix to post-multiply with.

Returns: the resulting current matrix after post-multiplication.

Throws:
   java\_lang\_NullPointerException -- if secondMatrix is null.

mRotate(float)

Declaration:
public org.w3c.dom.svg.SVGMatrix\_mRotate\(float\ angle\)

Description:
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:

\[
\begin{bmatrix}
\cos(\text{angle}) & -\sin(\text{angle}) & 0 \\
\sin(\text{angle}) & \cos(\text{angle}) & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

Parameters:
   angle - the angle of rotation in degrees.

Returns: the resulting current matrix after post-multiplication.

mScale(float)

Declaration:
public org.w3c.dom.svg.SVGMatrix\_mScale\(float\ scaleFactor\)

Description:
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:
Parameters:

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

Returns: the resulting current matrix after post-multiplication.

mTranslate(float, float)

Declaration:
public org.w3c.dom.svg.SVGMatrix mTranslate(float x, float y)

Description:
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:

\[
\begin{bmatrix}
1 & 0 & x \\
0 & 1 & y \\
0 & 0 & 1
\end{bmatrix}
\]

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.

Returns: the resulting current matrix after post-multiplication.
**Declaration**

public interface **SVGPath**

**Description**

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.

---

**Member Summary**

<table>
<thead>
<tr>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>static short CLOSE 86</td>
</tr>
<tr>
<td>static short CURVE_TO 87</td>
</tr>
<tr>
<td>static short LINE_TO 87</td>
</tr>
<tr>
<td>static short MOVE_TO 87</td>
</tr>
<tr>
<td>static short QUAD_TO 87</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close() 87</td>
</tr>
<tr>
<td>void curveTo(float x1, float y1, float x2, float y2, float x3, float y3) 87</td>
</tr>
<tr>
<td>int getNumberOfSegments() 88</td>
</tr>
<tr>
<td>short getSegment(int cmdIndex) 88</td>
</tr>
<tr>
<td>float getSegmentParam(int cmdIndex, int paramIndex) 88</td>
</tr>
<tr>
<td>void lineTo(float x, float y) 88</td>
</tr>
<tr>
<td>void moveTo(float x, float y) 89</td>
</tr>
<tr>
<td>void quadTo(float x1, float y1, float x2, float y2) 89</td>
</tr>
</tbody>
</table>

---

**Fields**

**CLOSE**

**Declaration:**

public static final short CLOSE

**Description:**

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

Declaration:
public static final short CURVE_TO

Description:
Numeric value is ASCII code of the letter 'C'.

LINE_TO

Declaration:
public static final short LINE_TO

Description:
Numeric value is ASCII code of the letter 'L'.

MOVE_TO

Declaration:
public static final short MOVE_TO

Description:
Numeric value is ASCII code of the letter 'M'.

QUAD_TO

Declaration:
public static final short QUAD_TO

Description:
Numeric value is ASCII code of the letter 'Q'.

Methods

close()

Declaration:
public void close()

Description:
Appends 'Z' (close path) segment to the path

curveTo(float, float, float, float, float, float)

Declaration:
public void curveTo(float x1, float y1, float x2, float y2, float x3, float y3)

Description:
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.
getNumberOfSegments()

Declaration:
public int getNumberOfSegments()

Description:
Return number of segments in this path.

Returns: the number of segments in this path.

getSegment(int)

Declaration:
public short getSegment(int cmdIndex)
    throws DOMException

Description:
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.

Returns: the segment command for the specified cmdIndex.

Throws:
    org.w3c.dom.DOMException - with error code INDEX_SIZE_ERR if segment index out of bounds.

getSegmentParam(int, int)

Declaration:
public float getSegmentParam(int cmdIndex, int paramIndex)
    throws DOMException

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

Parameters:
    cmdIndex - the command index for the segment parameter to retrieve.
    paramIndex - the parameter index for the segment parameter to retrieve.

Returns: the segment parameter for the specified cmdIndex and paramIndex.

Throws:
    org.w3c.dom.DOMException - with error code INDEX_SIZE_ERR if segment index out of bounds or param index out of bounds for this segment’s type.

lineTo(float, float)

Declaration:
public void lineTo(float x, float y)

Description:
Appends ‘L’ (absolute line) segment to the path with the specified coordinates.
Parameters:
  x - the x-axis coordinate of the specified point.
  y - the y-axis coordinate of the specified point.

**moveTo(float, float)**

**Declaration:**
public void moveTo(float x, float y)

**Description:**
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.

**quadTo(float, float, float, float)**

**Declaration:**
public void quadTo(float x1, float y1, float x2, float y2)

**Description:**
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.
Declaration
public interface SVGPoint

Description
This interface represents an “SVGPoint” datatype, identified by its x and y components.

Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>float getX()</td>
</tr>
<tr>
<td>float getY()</td>
</tr>
<tr>
<td>void setX(float value)</td>
</tr>
<tr>
<td>void setY(float value)</td>
</tr>
</tbody>
</table>

Methods

g getX()

Declaration:
public float getX()

Description:
Returns the x component of the point.

Returns: the x component of the point.

g getY()

Declaration:
public float getY()

Description:
Returns the y component of the point.

Returns: the y component of the point.

setX(float)

Declaration:
public void setX(float value)

Description:
Sets the x component of the point to the specified float value.

Parameters:
value - the x component value
setY(float)

**Declaration:**
public void setY(float value)

**Description:**
Sets the y component of the point to the specified float value.

**Parameters:**
- value - the y component value
This interface represents an “SVGRect” datatype, consisting of a minimum X, minimum Y, width and height values.

Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>float getHeight()</td>
<td>Returns the height for this SVGRect.</td>
<td>the rectangle height.</td>
</tr>
<tr>
<td>float getWidth()</td>
<td>Returns the width for this SVGRect.</td>
<td>the rectangle width.</td>
</tr>
<tr>
<td>float getX()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>float getY()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void setHeight(float value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void setWidth(float value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void setX(float value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void setY(float value)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Methods

getHeight()

Declaration:
public float getHeight()

Description:
Returns the height for this SVGRect.

Returns: the rectangle height.

getWidth()

Declaration:
public float getWidth()

Description:
Returns the width for this SVGRect.

Returns: the rectangle width.

g getX()

Declaration:
public float getX()
**SVGRect**

**getY()**

*Declaration:*

```java
public float getY()
```

*Description:*

Returns the minimum Y value for this SVGRect.

*Returns:*

the minimum Y value.

**setHeight(float)**

*Declaration:*

```java
public void setHeight(float value)
```

*Description:*

Sets the height of this SVGRect to the specified value.

*Parameters:*

value - the rectangle height value.

**setWidth(float)**

*Declaration:*

```java
public void setWidth(float value)
```

*Description:*

Sets the width of this SVGRect to the specified value.

*Parameters:*

value - the rectangle width value.

**setX(float)**

*Declaration:*

```java
public void setX(float value)
```

*Description:*

Sets the minimum X value of this SVGRect to the specified value.

*Parameters:*

value - the minimum X value.

**setY(float)**

*Declaration:*

```java
public void setY(float value)
```

*Description:*

Sets the minimum Y value of this SVGRect to the specified value.

*Parameters:*

value - the minimum Y value.
SVGRGBColor

Declaration
public interface SVGRGBColor

Description
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 (getRGBColorTrait70) such as fill, stroke, and color.

Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getBlue()</td>
</tr>
<tr>
<td>int getGreen()</td>
</tr>
<tr>
<td>int getRed()</td>
</tr>
</tbody>
</table>

Methods

getBlue()

Declaration:
public int getBlue()

Description:
Returns the blue component of the SVGRGBColor.

Returns: the blue component.

green()

Declaration:
public int getGreen()

Description:
Returns the green component of the SVGRGBColor.

Returns: the green component.

green()

Declaration:
public int getRed()

Description:
Returns the red component of the SVGRGBColor.

Returns: the red component.
org.w3c.dom.svg

SVGSVGElement

Declaration
public interface SVGSVGElement extends SVGLocatableElement

All Superinterfaces: org.w3c.dom.Element, org.w3c.dom.events.EventTarget, org.w3c.dom.Node, SVGElement, SVGLocatableElement

Description
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 potentially be modified through user-agent specific UI. User agent transformation can be obtained by multiplying matrix

\[
\begin{bmatrix}
\text{currentScale} & 0 & \text{currentTranslate}.x \\
0 & \text{currentScale} & \text{currentTranslate}.y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos(\text{currentRotate}) & -\sin(\text{currentRotate}) & 0 \\
\sin(\text{currentRotate}) & \cos(\text{currentRotate}) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

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

Note: If the application does not have the necessary privilege rights to access this (SVG) content, a SecurityException may be thrown by the underlying implementation. This is applicable to all the Tree navigation and Trait accessor methods. Features such as zooming, panning and playing of animations will not be affected.

Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVGMatrix createSVGMatrixComponents(float a, float b, float c, float d, float e, float f)</td>
</tr>
<tr>
<td>SVGPath createSVGPath()</td>
</tr>
<tr>
<td>SVGRect createSVGRect()</td>
</tr>
<tr>
<td>SVGRGBColor createSVGRGBColor(int red, int green, int blue)</td>
</tr>
<tr>
<td>float getCurrentRotate()</td>
</tr>
<tr>
<td>float get_current_scale()</td>
</tr>
<tr>
<td>float get_current_time()</td>
</tr>
<tr>
<td>SVGPoint getCurrentTranslate()</td>
</tr>
<tr>
<td>void setCurrentRotate(float value)</td>
</tr>
<tr>
<td>void setCurrentScale(float value)</td>
</tr>
<tr>
<td>void setCurrentTime(float seconds)</td>
</tr>
</tbody>
</table>
**Inherited Member Summary**

**Methods inherited from interface EventTarget**

- addEventListener(String, EventListener, boolean)
- removeEventListener(String, EventListener, boolean)

**Methods inherited from interface Node**

- appendChild(Node)
- getLocalName()
- getNamespaceURI()
- getParentNode()
- insertBefore(Node, Node)
- removeChild(Node)

**Methods inherited from interface SVGElement**

- getFirstElementChild()
- getFloatTrait(String)
- getId()
- getMatrixTrait(String)
- getNextElementSibling()
- getPathTrait(String)
- getRGBColorTrait(String)
- getRectTrait(String)
- getTraitNS(String, String)
- setFloatTrait(String, float)
- setId(String)
- setMatrixTrait(String, SVGMatrix)
- setPathTrait(String, SVGPath)
- setRGBColorTrait(String, SVGRGBColor)
- setRectTrait(String, SVGRect)
- setTrait(String, String)
- setTraitNS(String, String, String)

**Methods inherited from interface SVGLocatableElement**

- getBBox()
- getScreenBBox()
- getScreenCTM()

**Methods**

`createSVGMatrixComponents(float, float, float, float, float, float)`

**Declaration:**

```java
def public org.w3c.dom.svg.SVGMatrix createSVGMatrixComponents(float a, float b, float c, float d, float e, float f)```

**Description:**

Creates a new `SVGMatrix` object. This object can be used to modify value of traits which are compatible with `SVGMatrix` type using `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.

**Returns:**

The newly created SVGMatrix object.

**See Also:** `SVGMatrix`
createSVGPath()

Declaration:
public org.w3c.dom.svg.SVGPath86 createSVGPath()

Description:
Creates new SVGPath86 object. This object can be used to modify value of traits which are compatible with SVGPath86 type using setPathTrait73 method.

Returns: the newly created SVGPath object with empty path commands.

createSVGRect()

Declaration:
public org.w3c.dom.svg.SVGRect92 createSVGRect()

Description:
Creates new SVGRect92 object. This object can be used to modify value of traits which are compatible with SVGRect92 type using setRectTrait73 method. The initial values for x, y, width, height of this new SVGRect are zero.

Returns: the newly created SVGRect object.

createSVGRGBColor(int, int, int)

Declaration:
public org.w3c.dom.svg.SVGRGBColor94 createSVGRGBColor(int red, int green, int blue) throws SVGException

Description:
Creates new SVGRGBColor94 object. This object can be used to modify value of traits which are compatible with SVGRGBColor94 type using setRGBColorTrait74 method.

Parameters:
red - the red component of SVGRGBColor object.
green - the green component of SVGRGBColor object.
blue - the blue component of SVGRGBColor object.

Returns: the newly created SVGRGBColor object with specified (r,g,b) values.

Throws:
SVGException76 - with error code SVG_INVALID_VALUE_ERR: if any of the parameters is not in the 0..255 range.

gGetCurrentRotate()

Declaration:
public float getGetCurrentRotate()

Description:
Returns current user agent rotation angle in degrees. The initial value for currentRotate is 0.

Returns: the current user agent rotation coefficient in degrees.

gGetCurrentScale()

Declaration:
public float getGetCurrentScale()
**get Current Time**

**Declaration:**
```
public float getcurrentTime()
```

**Description:**
Returns current animation timeline time in seconds.

**Returns:**
the current animation timeline time in seconds.

**get Current Translate**

**Declaration:**
```
public org.w3c.dom.svg.SVGPoint getCurrentTranslate()
```

**Description:**
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).

**Returns:**
returns the current user agent translation.

**set Current Rotate (float)**

**Declaration:**
```
public void setCurrentRotate(float value)
```

**Description:**
Sets current user agent rotate coefficient in degrees.

**Parameters:**
value - the value of user agent rotate coefficient to be set.

**set Current Scale (float)**

**Declaration:**
```
public void setCurrentScale(float value) throws DOMException
```

**Description:**
Sets current user agent scale (zoom) coefficient.

**Parameters:**
value - the value of user agent scale coefficient to be set.

**Throws:**
org.w3c.dom.DOMException - with error code INVALID_ACCESS_ERR if the scale value is set to zero.

**set Current Time (float)**

**Declaration:**
```
public void setCurrentTime(float seconds)
```
**Description:**
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.

**Parameters:**
- `seconds` - the value of time to be set in seconds.
SVGSVGElemnent

setcurrentTime(float)

org.w3c.dom.svg
# Constant Field Values

## Contents
- `javax.microedition.*`
- `org.w3c.*`

### javax.microedition.*

| static int | RENDERING_QUALITY_HIGH | 2 |
| static int | RENDERING_QUALITY_LOW  | 1 |

### org.w3c.*

| static short | HIERARCHY_REQUEST_ERR | 3 |
| static short | INDEX_SIZE_ERR         | 1 |
| static short | INVALID_ACCESS_ERR     | 15|
| static short | INVALID_MODIFICATION_ERR | 13 |
| static short | INVALID_STATE_ERR      | 11|
| static short | NO_MODIFICATION_ALLOWED_ERR | 7 |
| static short | NOT_FOUND_ERR          | 8 |
| static short | NOT_SUPPORTED_ERR      | 9 |
| static short | TYPE_MISMATCH_ERR      | 17|
| static short | WRONG_DOCUMENT_ERR     | 4 |
org.w3c.*

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>static short SVG_INVALID_VALUE_ERR</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>static short SVG_MATRIX_NOT_INVERTABLE</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Index

**A**
activate()
of javax.microedition.m2g.SVGImage 35
addEventListener(String, EventListener, boolean)
of org.w3c.dom.events.EventTarget 56
appendChild(Node)
of org.w3c.dom.Node 49

**B**
beginElementAt(float)
of org.w3c.dom.svg.SVGAnimationElement 61
bindTarget(Object)
of javax.microedition.m2g.ScalableGraphics 15

**C**
CLOSE
of org.w3c.dom.svg.SVGPath 86
close()
of org.w3c.dom.svg.SVGPath 87
code
of org.w3c.dom.DOMException 46
of org.w3c.dom.svg.SVGException 77
createAnimator(SVGImage)
of javax.microedition.m2g.SVGAnimator 26
createAnimator(SVGImage, String)
of javax.microedition.m2g.SVGAnimator 26
createElementNS(String, String)
of org.w3c.dom.Document 43
createEmptyImage(ExternalResourceHandler)
of javax.microedition.m2g.SVGImage 36
createImage(InputStream, ExternalResourceHandler)
of javax.microedition.m2g.ScalableImage 19
createImage(String, ExternalResourceHandler)
of javax.microedition.m2g.ScalableImage 20
createInstance()
of javax.microedition.m2g.ScalableGraphics 15
createSVGMatrixComponents(float, float, float, float, float, float)
of org.w3c.dom.svg.SVGSVGElement 96
createSVGPath()
of org.w3c.dom.svg.SVGSVGElement 97
createSVGRect()
of org.w3c.dom.svg.SVGSVGElement 97
createSVGRGBColor(int, int, int)
of org.w3c.dom.svg.SVGSVGElement 97
CURVE_TO
of org.w3c.dom.svg.SVGPath 87
curveTo(float, float, float, float, float, float)
of org.w3c.dom.svg.SVGPath 87

dispatchMouseEvent(String, int, int)
of javax.microedition.m2g.SVGImage 36
Document
of org.w3c.dom 42
DOMException
of org.w3c.dom 45
DOMException(short, String)
of org.w3c.dom.DOMException 47

element
of org.w3c.dom 48
endElementAt(float)
of org.w3c.dom.svg.SVGAnimationElement 61
Event
of org.w3c.dom.events 54
EventListener
of org.w3c.dom.events 55
EventTarget
of org.w3c.dom.events 56
ExternalResourceHandler
of javax.microedition.m2g 12

**F**
focusOn(SVGElement)
of javax.microedition.m2g.SVGImage 37

**G**
getBBox()
of org.w3c.dom.svg.SVGLocatableElement 82
getBlue()
of org.w3c.dom.svg.SVGRGBColor 94
insertBefore(Node, Node)
of org.w3c.dom.Node

INVALID_ACCESS_ERR
of org.w3c.dom.DOMException

INVALID_MODIFICATION_ERR
of org.w3c.dom.DOMException

INVALID_STATE_ERR
of org.w3c.dom.DOMException

inverse()
of org.w3c.dom.svg.SVGMatrix

invokeAndWait(Runnable)
of javax.microedition.m2g.SVGAnimator

invokeLater(Runnable)
of javax.microedition.m2g.SVGAnimator

K

keyPressed(int)
of javax.microedition.m2g.SVGEventListener

keyReleased(int)
of javax.microedition.m2g.SVGEventListener

L

LINE_TO
of org.w3c.dom.svg.SVGPath

lineTo(float, float)
of org.w3c.dom.svg.SVGPath

M

mMultiply(SVGMatrix)
of org.w3c.dom.svg.SVGMatrix

MOVE_TO
of org.w3c.dom.svg.SVGPath

moveTo(float, float)
of org.w3c.dom.svg.SVGPath

mRotate(float)
of org.w3c.dom.svg.SVGMatrix

mScale(float)
of org.w3c.dom.svg.SVGMatrix

mTranslate(float, float)
of org.w3c.dom.svg.SVGMatrix

N

NO_MODIFICATION_ALLOWED_ERR
of org.w3c.dom.DOMException

Node
of org.w3c.dom

NOT_FOUND_ERR
of org.w3c.dom.DOMException

NOT_SUPPORTED_ERR
of org.w3c.dom.DOMException

P

pause()
of javax.microedition.m2g.SVGAnimator

play()
of javax.microedition.m2g.SVGAnimator

pointerPressed(int, int)
of javax.microedition.m2g.SVGEventListener

pointerReleased(int, int)
of javax.microedition.m2g.SVGEventListener

Q

QUAD_TO
of org.w3c.dom.svg.SVGPath

quadTo(float, float, float)
of org.w3c.dom.svg.SVGPath

R

releaseTarget()
of javax.microedition.m2g.ScalableGraphics

removeChild(Node)
of org.w3c.dom.Node

removeEventListener(String, EventListener, boolean)
of org.w3c.dom.events.EventTarget

render(int, int, ScalableImage)
of javax.microedition.m2g.ScalableGraphics

RENDERING_QUALITY_HIGH
of javax.microedition.m2g.ScalableGraphics

RENDERING_QUALITY_LOW
of javax.microedition.m2g.ScalableGraphics

requestCompleted(String, InputStream)
of javax.microedition.m2g.ScalableImage

requestResource(ScalableImage, String)
of javax.microedition.m2g.ExternalResource-
Index

Handler 12

S
ScalableGraphics
  of javax.microedition.m2g 14
ScalableImage
  of javax.microedition.m2g 18
ScalableImage
  of javax.microedition.m2g 19
setCurrentRotate(float)
  of org.w3c.dom.svg.SVGElement 98
setCurrentScale(float)
  of org.w3c.dom.svg.SVGElement 98
setCurrentTime(float)
  of org.w3c.dom.svg.SVGElement 98
setFloatTrait(String, float)
  of org.w3c.dom.svg.SVGElement 71
setHeight(float)
  of org.w3c.dom.svg.SVGRect 93
setId(String)
  of org.w3c.dom.svg.SVGElement 72
setMatrixTrait(String, SVGMatrix)
  of org.w3c.dom.svg.SVGElement 72
setPathTrait(String, SVGPath)
  of org.w3c.dom.svg.SVGElement 73
setRectTrait(String, SVGRect)
  of org.w3c.dom.svg.SVGElement 73
setRenderingQuality(int)
  of javax.microedition.m2g.ScalableGraphics 16
setRGBColorTrait(String, SVGRGBColor)
  of org.w3c.dom.svg.SVGElement 74
setSVGEventListener(SVGEventListener)
  of javax.microedition.m2g.SVGAnimator 28
setTimeIncrement(float)
  of javax.microedition.m2g.SVGAnimator 29
setTrait(String, String)
  of org.w3c.dom.svg.SVGElement 74
setTraitNS(String, String, String)
  of org.w3c.dom.svg.SVGElement 75
setTransparency(float)
  of javax.microedition.m2g.ScalableGraphics 16
setWidth(float)
  of org.w3c.dom.svg.SVGRect 93
setX(float)
  of org.w3c.dom.svg.SVGPoint 90
  of org.w3c.dom.svg.SVGRect 93
setY(float)
  of org.w3c.dom.svg.SVGPoint 91
  of org.w3c.dom.svg.SVGRect 93
showNotify()
  of javax.microedition.m2g.SVGEventListener 33
sizeChanged(int, int)
  of javax.microedition.m2g.SVGEventListener 33
stop()
  of javax.microedition.m2g.SVGAnimator 29
SVG_INVALID_VALUE_ERR
  of org.w3c.dom.svg.SVGException 77
SVG_MATRIX_NOT_INVERTABLE
  of org.w3c.dom.svg.SVGException 77
SVGAnimationElement
  of org.w3c.dom.svg 60
SVGAnimator
  of javax.microedition.m2g 24
SVGAnimator()
  of javax.microedition.m2g 26
SVGEmployee
  of org.w3c.dom.svg 62
SVGEventListener
  of javax.microedition.m2g 30
SVGException
  of org.w3c.dom.svg 76
SVGException(short, String)
  of org.w3c.dom.svg.SVGException 77
SVGImage
  of javax.microedition.m2g 34
SVGImage()
  of javax.microedition.m2g 35
SVGLocatableElement
  of org.w3c.dom.svg 78
SVGMatrix
  of org.w3c.dom.svg 83
SVGPath
  of org.w3c.dom.svg 86
SVGPoint
  of org.w3c.dom.svg 90
SVGRect
  of org.w3c.dom.svg 92
SVGRGBColor
  of org.w3c.dom.svg 94
SVGSVGElement
  of org.w3c.dom.svg 95

T
TYPE_MISMATCH_ERR
  of org.w3c.dom.DOMException 47

W
WRONG_DOCUMENT_ERR
  of org.w3c.dom.DOMException 47
This work (and included software, documentation such as READMEs, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions.

Permission to copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications:

1. The full text of this NOTICE in a location viewable to users of the redistributed or derivative work.
2. Any pre-existing intellectual property disclaimers, notices, or terms and conditions. If none exist, the W3C Software Short Notice (http://www.w3.org/Consortium/Legal/2002/copyright-software-short-notice-20021231.html) should be included (hypertext is preferred, text is permitted) within the body of any redistributed or derivative code.
3. Notice of any changes or modifications to the files, including the date changes were made. (We recommend you provide URLs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED “AS IS,” AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

This formulation of W3C’s notice and license became active on December 31 2002. This version removes the copyright ownership notice such that this license can be used with materials other than those owned by the W3C, reflects that ERCIM is now a host of the W3C, includes references to this specific dated version of the license, and removes the ambiguous grant of “use”. Otherwise, this version is the same as the previous version (http://www.w3.org/Consortium/Legal/copyright-software-19980720) and is written so as to preserve the Free Software Foundation’s assessment of GPL compatibility (http://www.gnu.org/philosophy/license-list.html#GPLCompatibleLicenses) and OSI’s certification (http://www.opensource.org/licenses/W3C.php) under the Open Source Definition (http://www.opensource.org/docs/definition.php). Please see our Copyright FAQ (http://www.w3.org/Consortium/Legal/IRPR-FAQ) for common questions about using materials from our site, including specific terms and conditions for packages like libwww, Amaya, and Jigsaw. Other questions about this notice can be directed to site-policy@w3.org.

Joseph Reagle <mailto:site-policy@w3.org>
Last revised by Reagle SDate: 2003/01/16 15:01:10 $