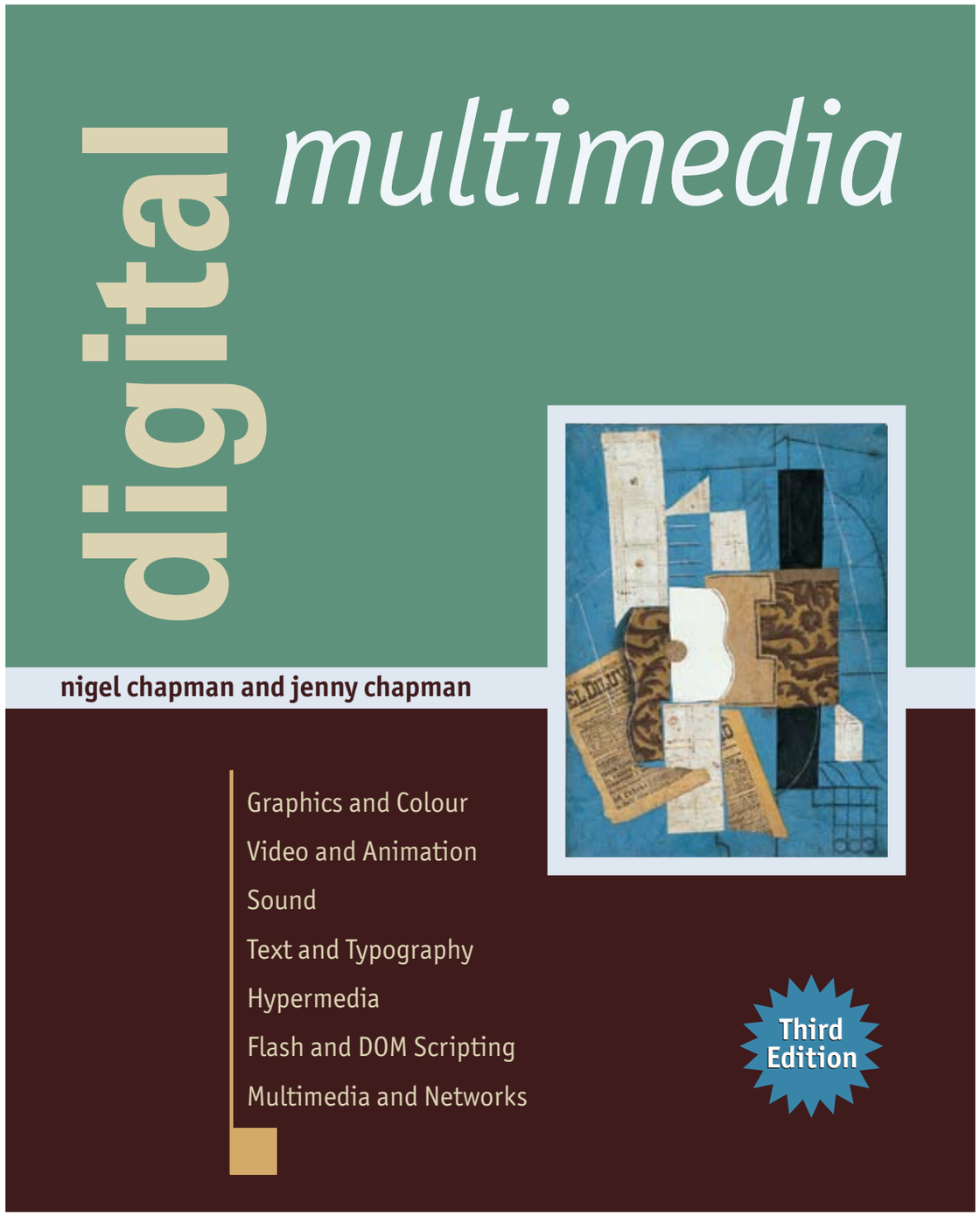


15

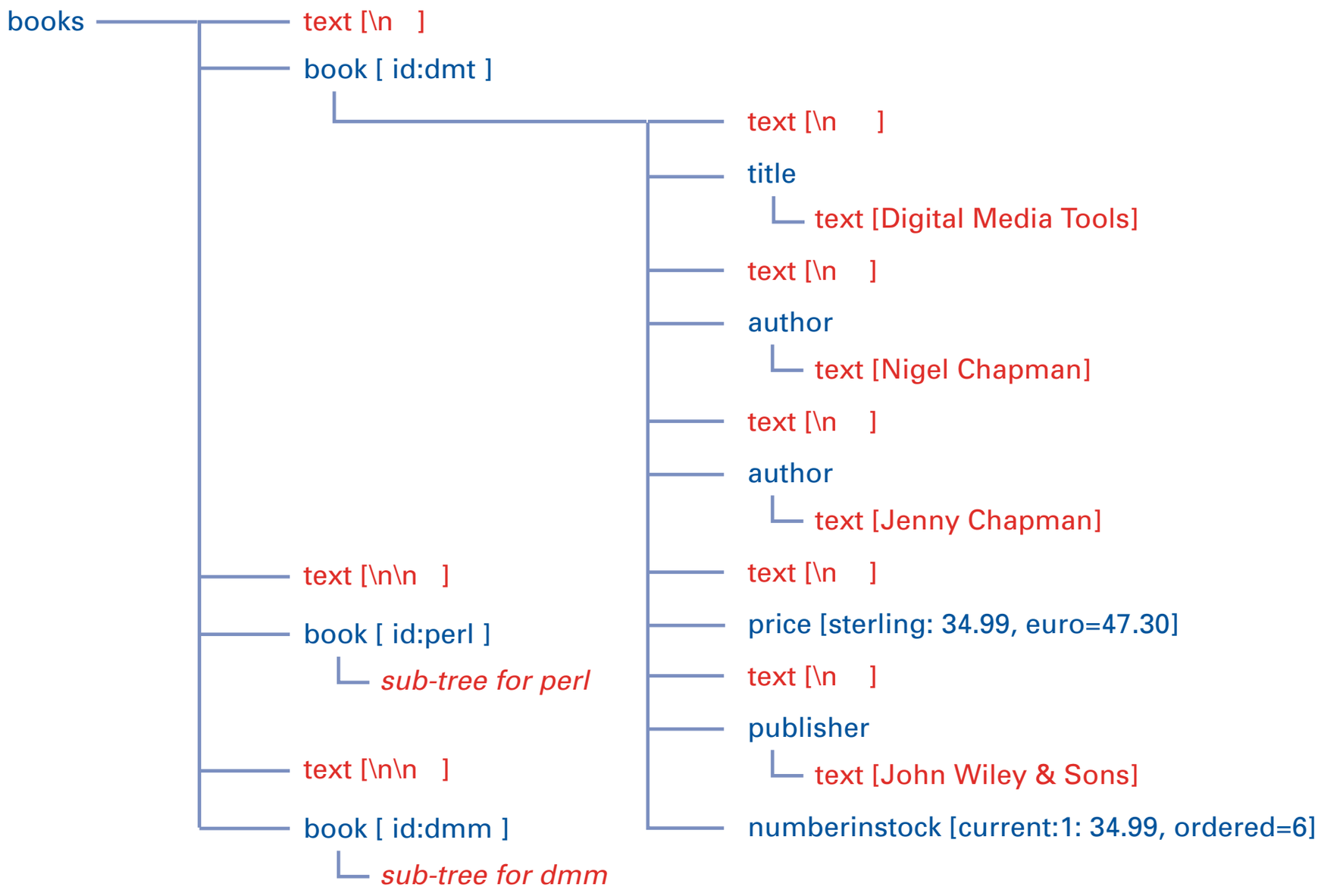
XML and Multimedia

Syntax and DTDs

XML-based languages all use the same syntax for tags, attributes and entities.

Documents that obey the basic syntax rules of XML are well-formed.

The tree structure of any well-formed XML document can be traversed and modified using the methods of the core DOM.



The structure model of an XML document

A Document Type Definition (DTD) provides a specification of a set of permitted elements, the attributes each may have and which elements they can contain.

Schemas are a more recently developed alternative to DTDs, which do not need any special syntax in the specifications.

A well-formed document is valid if it declares a DTD and conforms to it.

An XML document may begin with an XML declaration specifying the XML version and the character set used in the document.

The DOCTYPE declaration declares the name of the document element and the location of a DTD, by its public name and system identifier (URL).

A DTD consists of markup declarations, enclosed between <! and >, which provide the definitions of the set of elements and attributes.

An element declaration declares the name and content model of an element.

Content models include EMPTY, (#PCDATA)* and the names of other elements.

A list of content types separated by commas means that the corresponding elements, etc. must appear in the given order.

Postfix operators +, * and ? indicate one or more, zero or more, or zero or one occurrences. The | operator represents a choice between two alternatives.

Each element's attributes are listed in a separate attribute-list declaration.

An attribute-list declaration begins <!ATTLIST followed by the element name, and a list of attribute specifications, terminated by >.

Each attribute's specification consists of its name, its type and an indication of whether it is compulsory or optional.

Types include CDATA (characters), an enumerated list of values and ID, for identifiers which must be unique.

Compulsory attributes are specified as #REQUIRED, optional attributes as #IMPLICIT. A default value may be provided instead.

Namespaces

Namespaces are collections of element and attribute names, which are used to prevent name clashes when XML-based languages are combined.

Namespaces are identified by unique URLs.

A prefix is associated with a namespace's URL within an element by assigning the URL as the value of an attribute consisting of the string xmlns: followed by the namespace prefix in the element's start tag.

Assigning to xmlns with no prefix defines a default namespace.

A name with a prefix and colon at the beginning (e.g. px:elname) belongs to the namespace with which that prefix has been associated.

Names without prefixes belong to the default namespace.

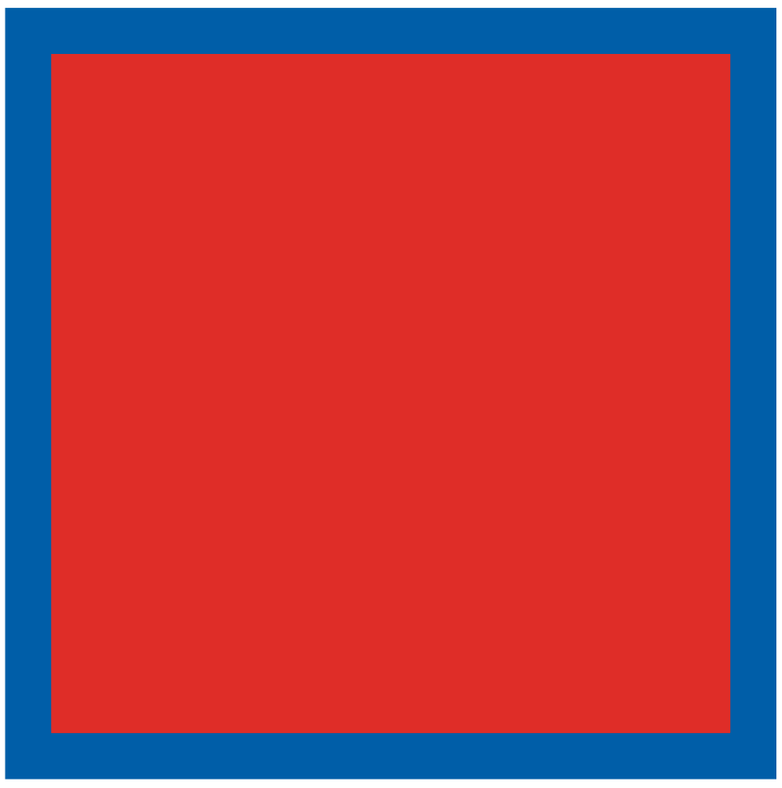
Attributes do not need to be in a namespace to avoid name clashes, but if an attribute name is used for the same purpose in more than one language, placing it in a namespace makes it easy to process in the same way in each language.

Namespaces can be used to classify the set of values an attribute may have.

RDFa allows a prefix to be used to identify values of the property attribute as belonging to some metadata standard, such as Dublin Core.

SVG

SVG (Scalable Vector Graphics) is an XML-based language for two-dimensional vector graphics.



```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1" xmlns="http://www.w3.org/2000/svg">
  <rect x="4" y="4" fill="#E53930" stroke="#0066B3" stroke-
    width="8"
    width="126" height="126"/>
</svg>
```

A simple SVG drawing

An SVG document begins with the usual XML and DOCTYPE declarations. The graphic content is contained in the svg element, which must declare the SVG namespace.

SVG fragments can be embedded in documents that use other XML-based languages, provided the document is treated as XML.

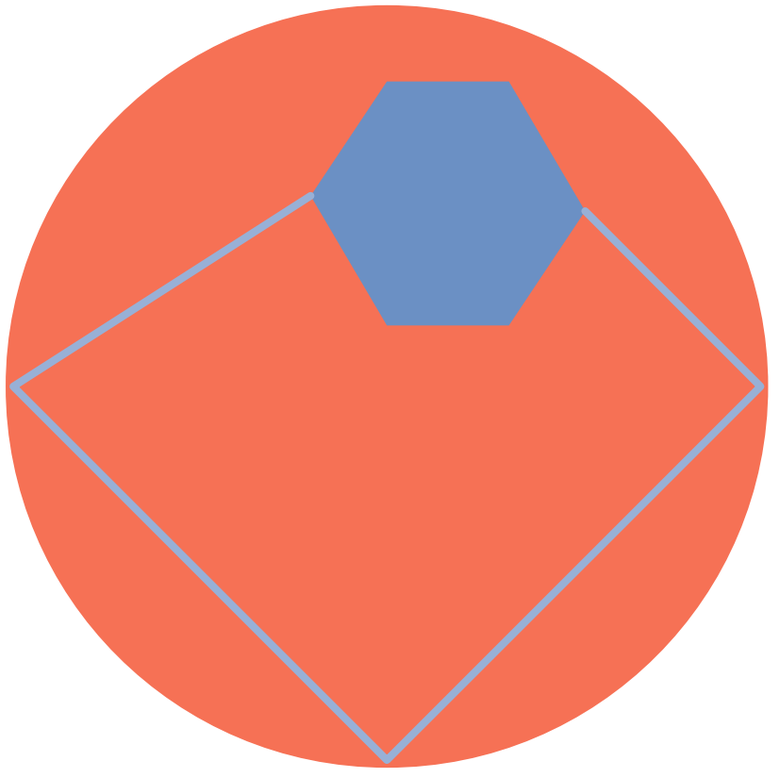
SVG Basic and SVG Tiny define SVG profiles suitable for mobile devices.

The rect, circle, ellipse, line, polyline and polygon elements represent the basic shapes.

Attributes define the geometry, position, stroke, fill and other properties of each shape.

Element Name	Attributes	Notes
rect	x	coordinates of top left corner
	y	
	width	
	height	
	rx	*x* and *y* radii of rounded corners
	ry	
circle	cx	coordinates of centre
	cy	
	r	radius
ellipse	cx	coordinates of centre
	cy	
	rx	*x* and *y* radii
	ry	
line	x1	coordinates of end points
	y1	
	x2	
	y2	
polyline	points	list of points – see text
polygon	points	

SVG shape elements



```
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
    width="110px" height="110px">
  <polygon fill="#6B90C4" points="50,10 66,10 76,27 66,42
50,42 40,25 "/>
  <polyline fill="none" stroke="#98B0D6"
    stroke-linecap="round" stroke-linejoin="round"
    points="40,25 1,50 50,99 99,50 76,27 "/>
</svg>
```

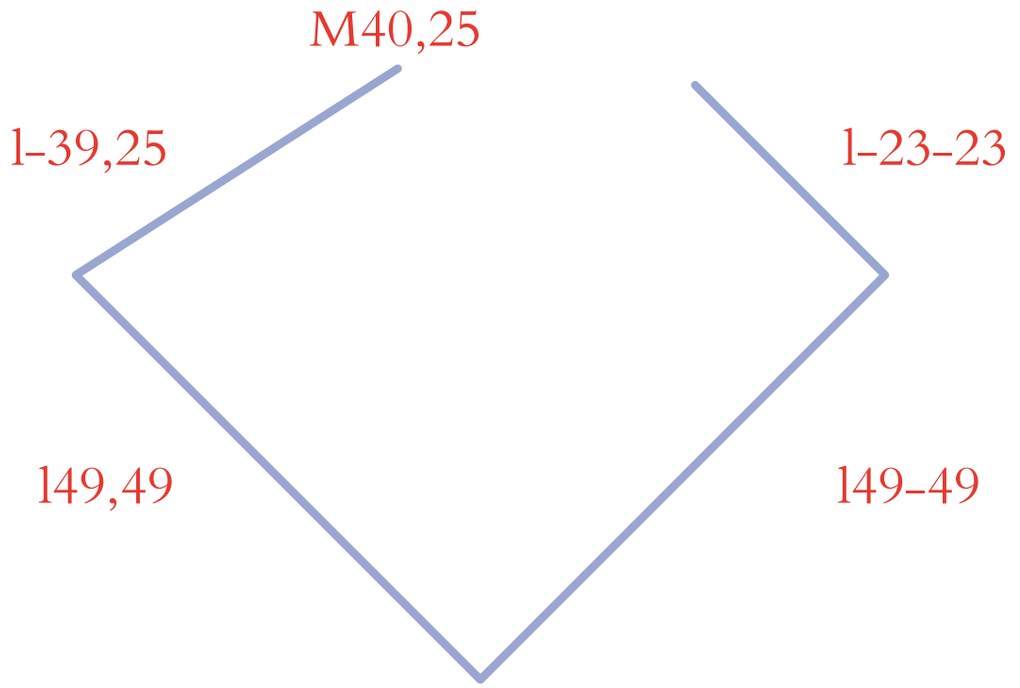
Shapes

The path element represents a sequence of lines and curves.

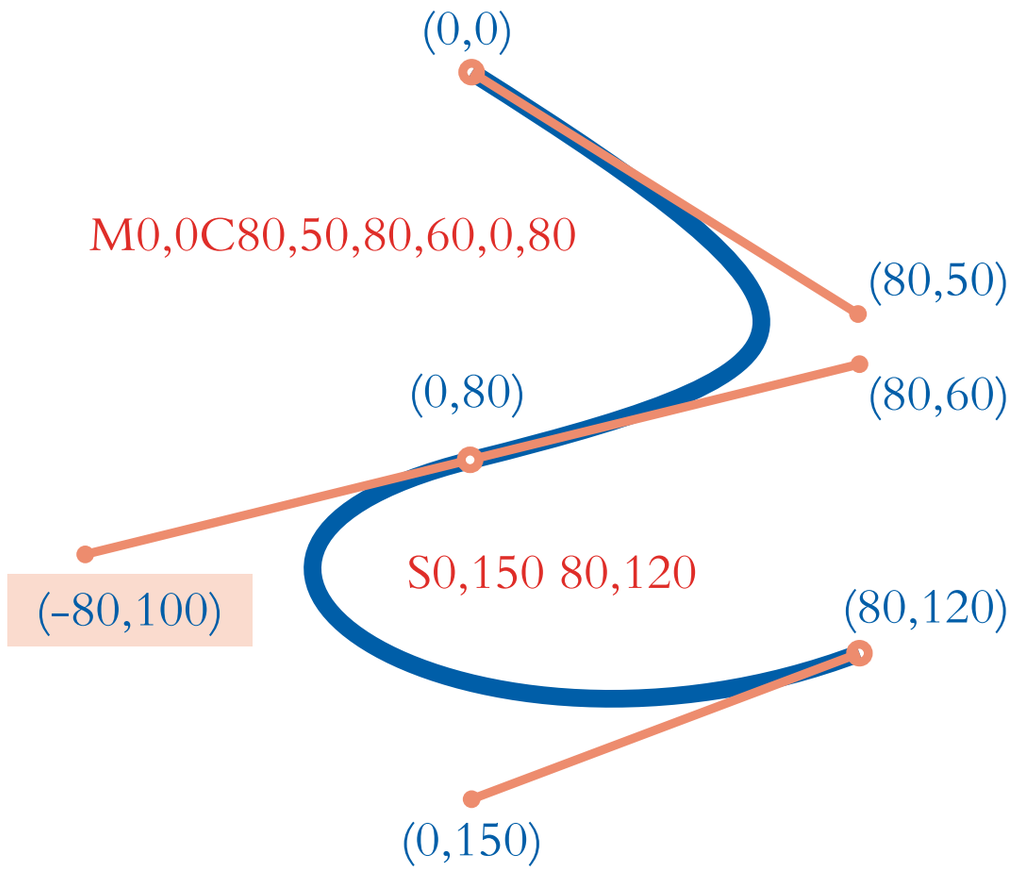
Its d attribute's value is a string containing a sequence of instructions for drawing the path.

Path instructions include M (move to), L (draw a line to an absolute position), l (draw a line to a relative position), H, h, V, v (draw horizontal and vertical lines), C, c (Bézier curves), S and s (curve segments).

Each instruction is followed by an appropriate number of pairs of values, to be interpreted as coordinates.



A polyline as a path



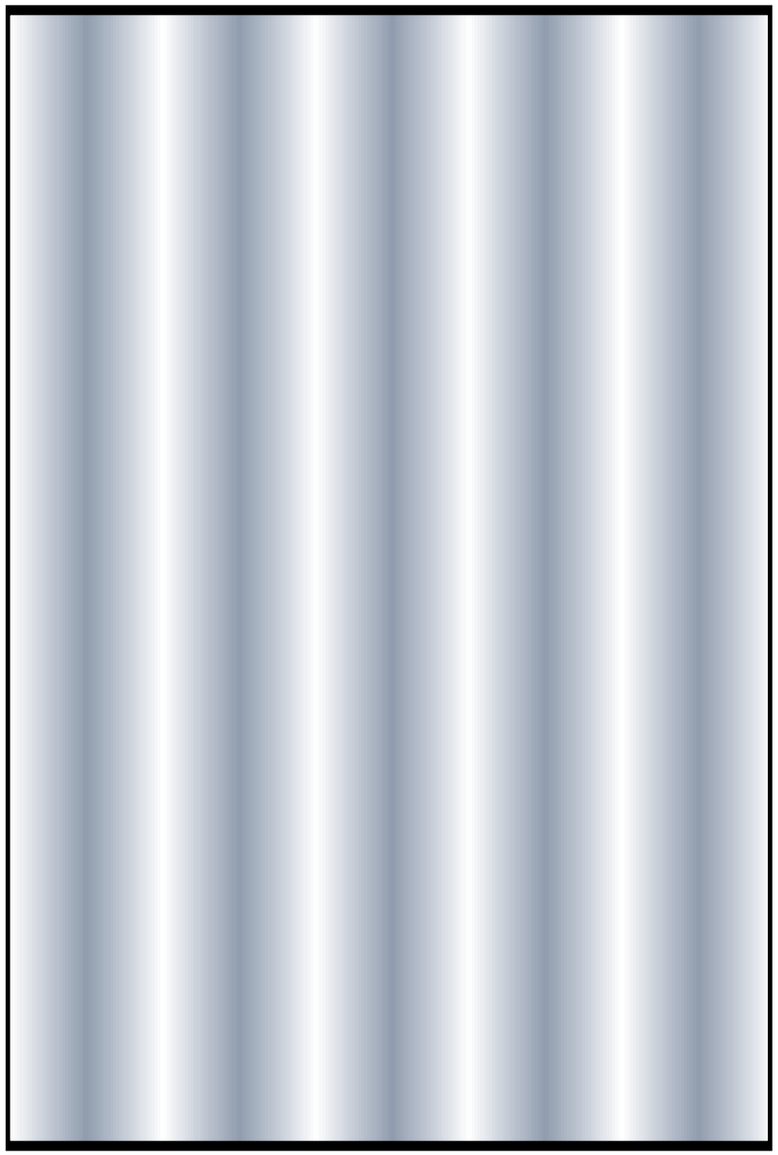
Smoothly joined Bézier curves

Stroke colour and width are specified by the stroke and stroke-width attributes.

Line joining and end styles can be specified with stroke-linejoin and stroke-linecap.

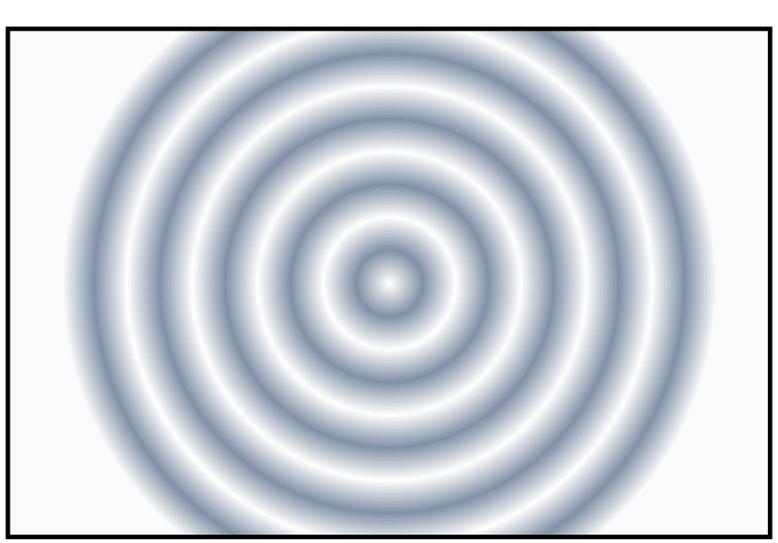
Gradients are defined by linearGradient and radialGradient elements, which may contain stop elements, each specifying the colour at an offset.

Gradients are applied by setting the fill attribute of a shape to the URL of a gradient element, usually just a fragment identifier referring to its id.



```
<linearGradient id="SVGID_1_">
   <stop  offset="0" stop-color="#FFFFFF"/>
   <stop  offset="0.1" stop-color="#8191A6"/>
   <stop  offset="0.2" stop-color="#FFFFFF"/>
   <stop  offset="0.3" stop-color="#8191A6"/>
   <stop  offset="0.4" stop-color="#FFFFFF"/>
   <stop  offset="0.5" stop-color="#8191A6"/>
   <stop  offset="0.6" stop-color="#FFFFFF"/>
   <stop  offset="0.7" stop-color="#8191A6"/>
   <stop  offset="0.8" stop-color="#FFFFFF"/>
   <stop  offset="0.9" stop-color="#8191A6"/>
   <stop  offset="1" stop-color="#FFFFFF"/>
</linearGradient>
```

Linear gradient fill



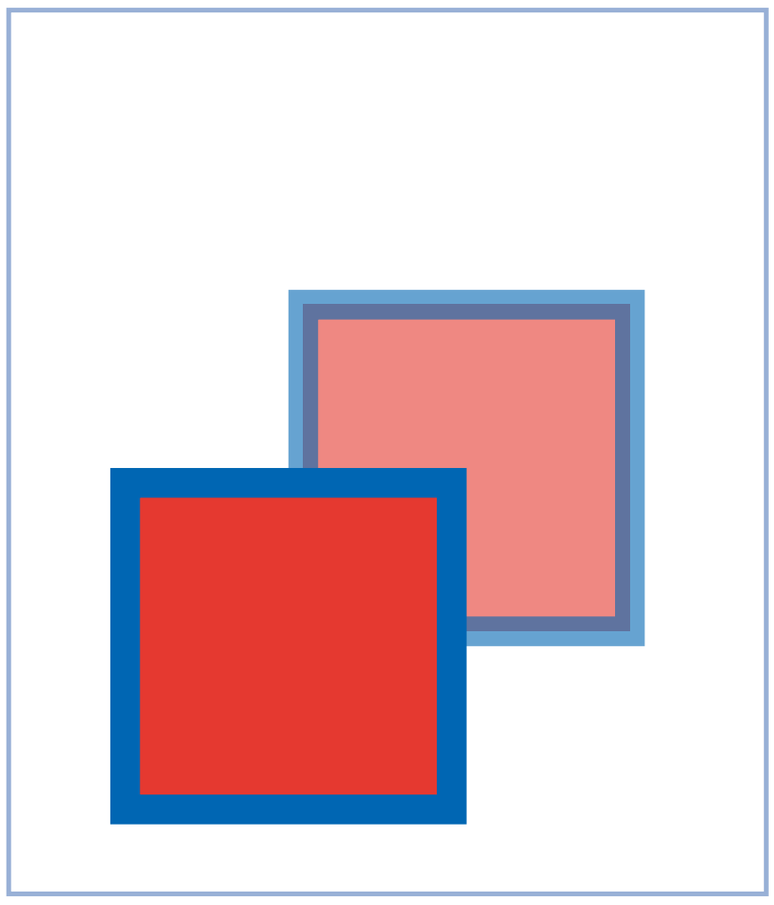
```
<radialGradient id="SVGID_2_">
   <stop  offset="0" stop-color="#FFFFFF"/>
   <stop  offset="0.1" stop-color="#8191A6"/>
   eight more stop elements, as before
   <stop  offset="1" stop-color="#FFFFFF"/>
</radialGradient>

<rect x="0.5" y="0.5" fill="url(#SVGID_2_)" stroke="#000000"
 width="170" height="113"/>
```

Radial gradient fill

Any element may have a transform attribute, whose value is a string of transformation specifications.

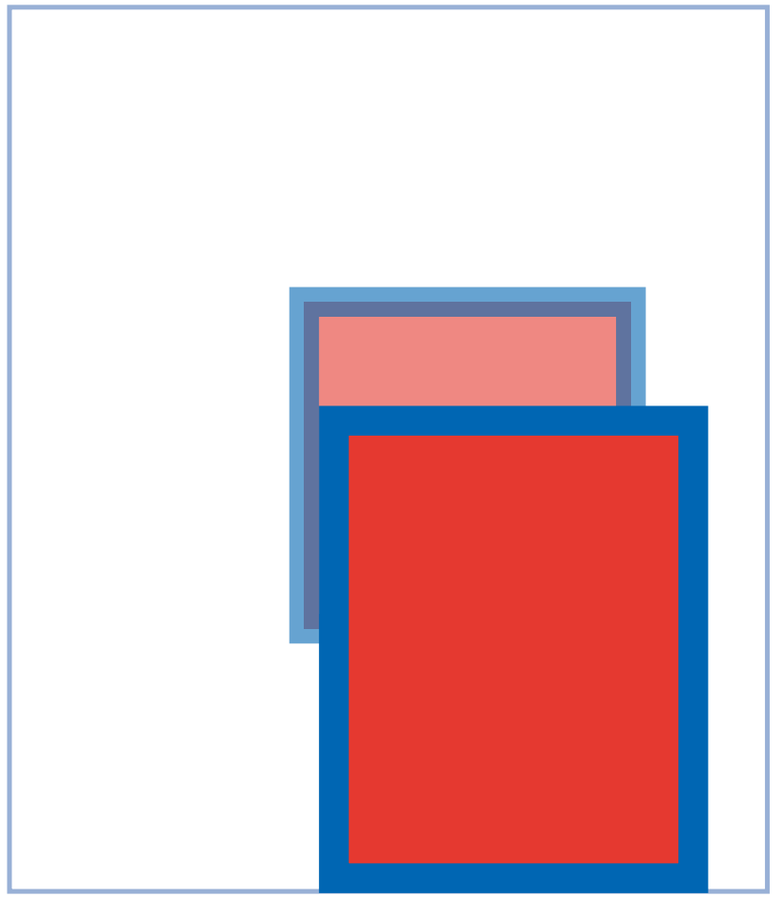
The available transformation names are translate, scale, rotate, skewX and skewY.

Appropriate arguments appear in brackets.



```
<rect x="50" y="50" fill="#E53930" stroke="#0066B3"
  stroke-width="5"
  width="55" height="55"
  transform="translate(-30,30)"/>
```

Translation

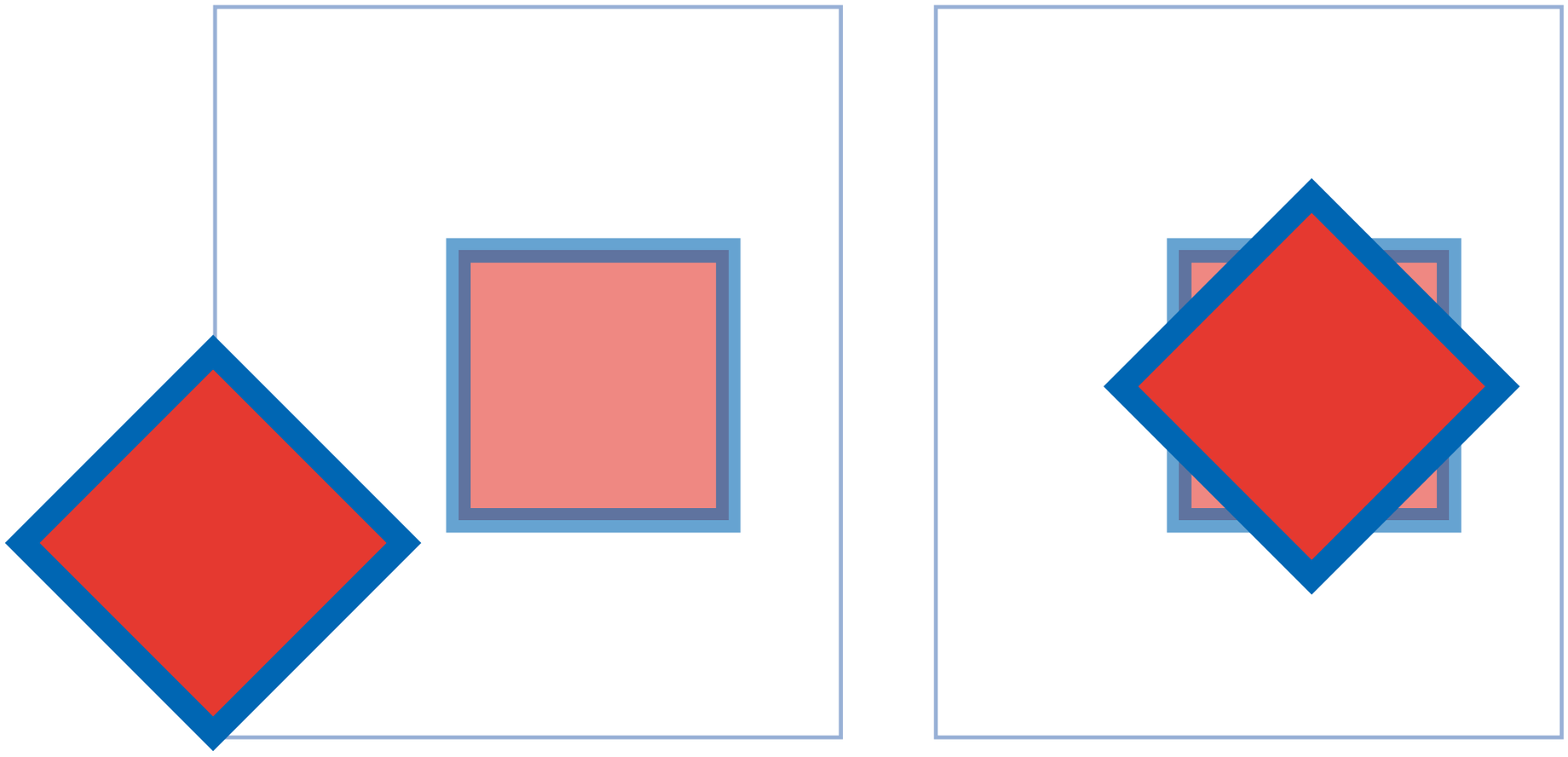


```
<rect x="50" y="50" fill="#E53930" stroke="#0066B3"
  stroke-width="5"  width="55" height="55"
  transform="scale(1.1,1.4)"/>
```

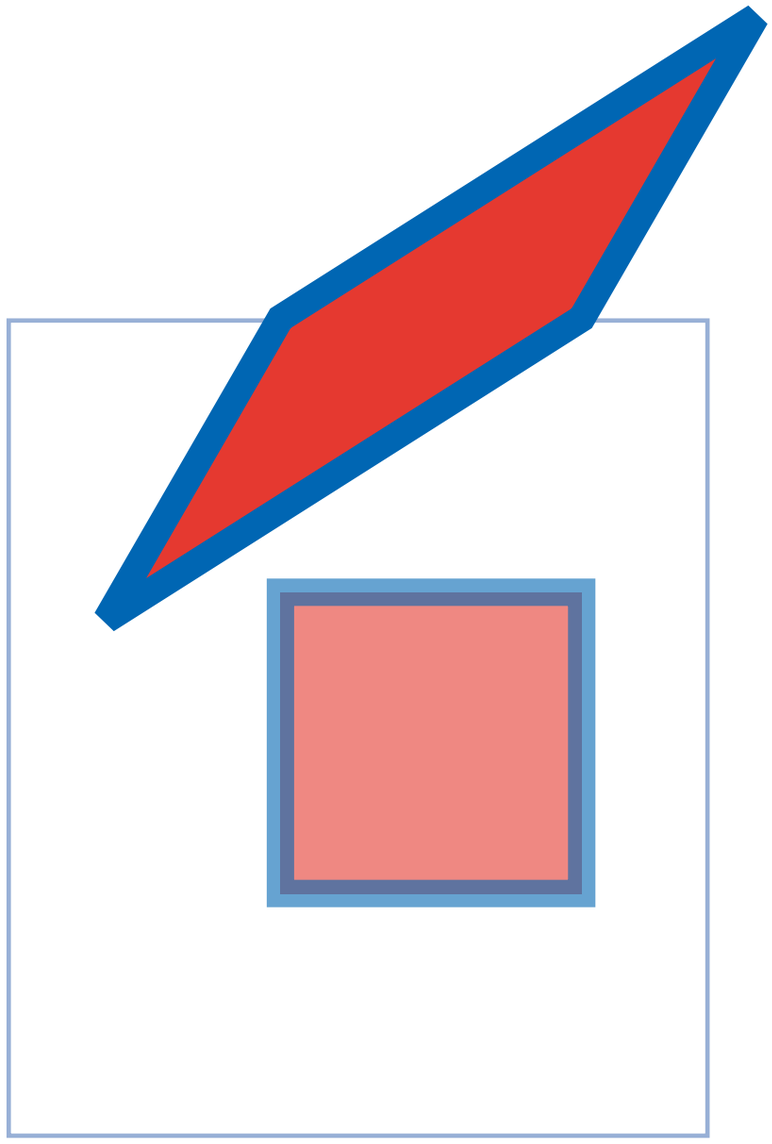
Scaling

```
<rect x="50" y="50" fill="#E53930"
  stroke="#0066B3"
  stroke-width="5"  width="55" height="55"
  transform="rotate(45)"/>
```

```
<rect x="50" y="50" fill="#E53930"
  stroke="#0066B3"
  stroke-width="5"  width="55" height="55"
  transform="rotate(45, 77, 77)"/>
```



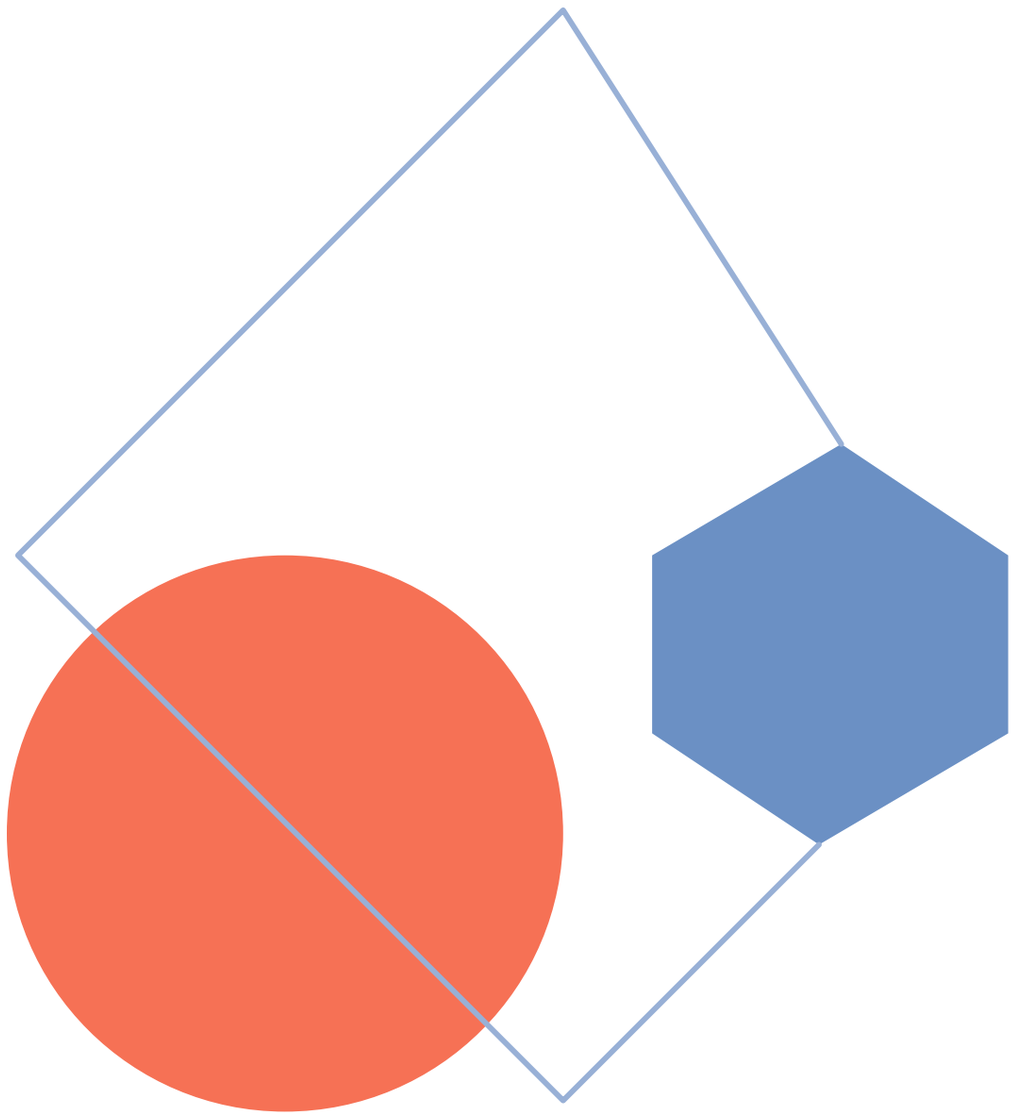
Rotation about the origin and about a point



```
<rect x="50" y="50" fill="#E53930" stroke="#0066B3"
  stroke-width="5"  width="55" height="55"
  transform="skewX(-30),skewY(-45)"/>
```

Skewing

The g element may be used to combine elements into a group, which can be transformed as a whole.



```
<g transform="scale(2), rotate(90),
  translate(-50,-100)">
  <polygon fill="#6B90C4"
    points="50,10 66,10 76,27 66,42 50,42 40,25 "/>
  <polyline fill="none" stroke="#98B0D6"
    stroke-linecap="round" stroke-linejoin="round"
    points="40,25 1,50 50,99 99,50 76,27 "/>
</g>
```

```
<g transform="scale(2)">
  <g transform="rotate(90)">
    <g transform="translate(-50,-100)">
      polygon and polyline elements
    </g>
  </g>
</g>
```

Transforming a group

Links (hot spots) can be created with the a element.

The XLink namespace must be declared so the xlink:href attribute is available.

The text element is used to hold text strings.

Text may be set on a path.

The SVG DOM makes it possible for scripts to perform dynamic drawing and programmed animation by operating on objects that represent SVG elements.