



COMP 4021

Internet Computing

SVG Basics

Dr. Kenneth LEUNG

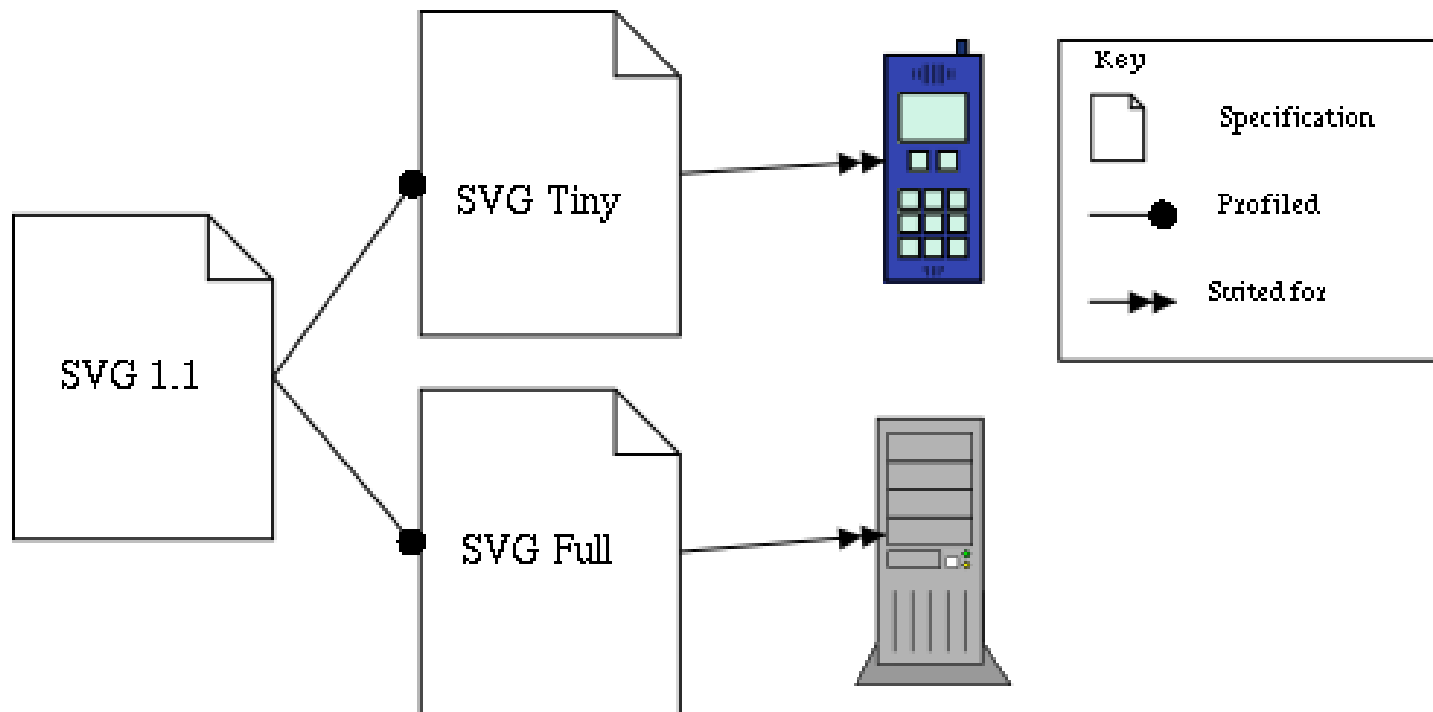
Slides created by Dr. David ROSSITER

This Presentation

- In this presentation we look at the some basic elements of SVG:
 - Text
 - Line
 - Circle
 - Ellipse
 - Rectangle
 - Polygons
 - Polyline
 - Using bitmap images in SVG
 - Paths
 - Using style
 - Opacity
 - Stroke parameters

Different Versions of SVG

- SVG has the full version and a 'reduced' version for mobile phones



SVG Text (01_simple_text.svg)

```
<svg xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink"
      version="1.1"
      baseProfile="full"
      width="800" height="600">
```

This line
enables all the
SVG's in this
PPT to be
shown correctly
in all major
browsers

```
<text x="10" y="300" style="font-size: 60px; fill: red" >
This is SVG text </text>
</svg>
```

Lower-left-corner of text
element = 10,300
Unlike other SVG elements!

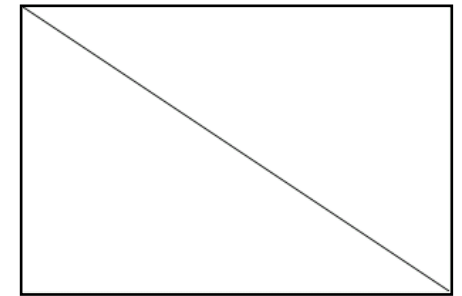
Line (02_line.svg)

```
<svg width="800" height="600">
```

```
  <line x1="0" y1="0"  
        x2="300" y2="200"  
        style="stroke:black"/>
```

```
</svg>
```

0,0



300,200

As with all W3C languages, the order of the parameters doesn't matter

Rectangle (03_blue_rectangle.svg)

```
<svg width="800" height="600">
```

```
  <rect width="700" height="100"
```

```
    x="0" y="200"
```

```
    style="fill:blue" />
```

```
</svg>
```



Circle and Ellipse (04_circle_ellipse.svg)

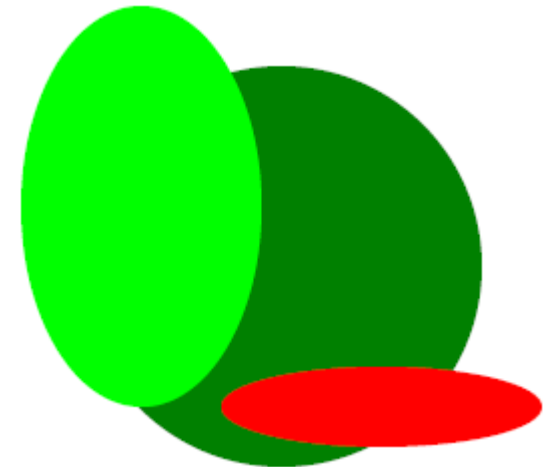
```
<svg width="300" height="300">
```

```
<circle cx="150" cy="150" r="100"  
  style="fill:green"/>
```

```
<ellipse cx="80" cy="120"  
  rx="60" ry="100" style="fill:lime"/>
```

```
<ellipse cx="200" cy="220"  
  rx="80" ry="20" style="fill:red"/>
```

```
</svg>
```



Later elements will be placed on top of previous ones

Polygons (05_polygons.svg)

```
<svg width="300" height="300">
```

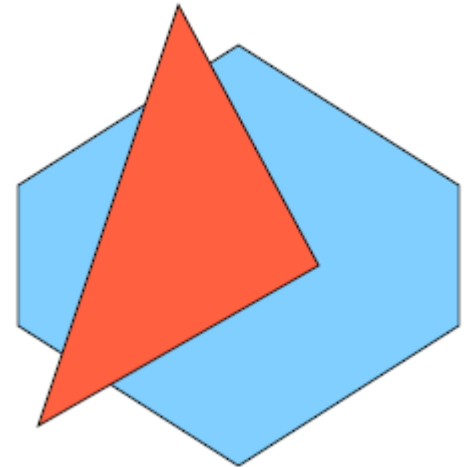
```
<polygon points="150,40 40,110 40,180  
150,250 260,180 260,110"
```

```
style="fill: lightskyblue; stroke: black"/>
```

```
<polygon points="120,20 50,230 190,150"
```

```
style="fill: tomato; stroke: black" />
```

```
</svg>
```

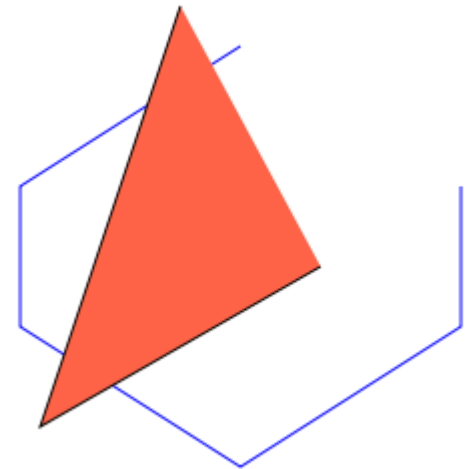


PolyLine (06_polylines.svg)

A polyline is a sequence of connected line segments
With “filled”, a polyline looks like a polygon: see the triangle

```
<polyline points="150,40 40,110  
40,180 150,250 260,180 260,110"  
style="fill:none;stroke:blue"/>
```

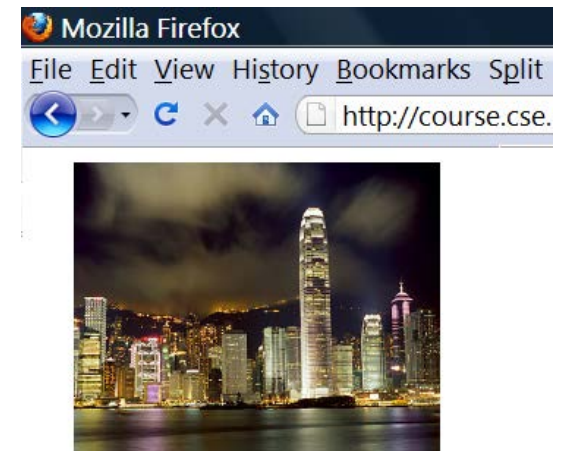
```
<polyline points="120,20 50,230, 190,150"  
style="fill:tomato;stroke:black"/>
```



Bitmap Images (07_image.svg)

You can use bitmap images inside SVG

```
<svg width="800" height="600">  
  <image xlink:href="hong_kong.jpg"  
    x="10" y="10"  
    width="300" height="300"/>  
</svg>
```



Creating Paths

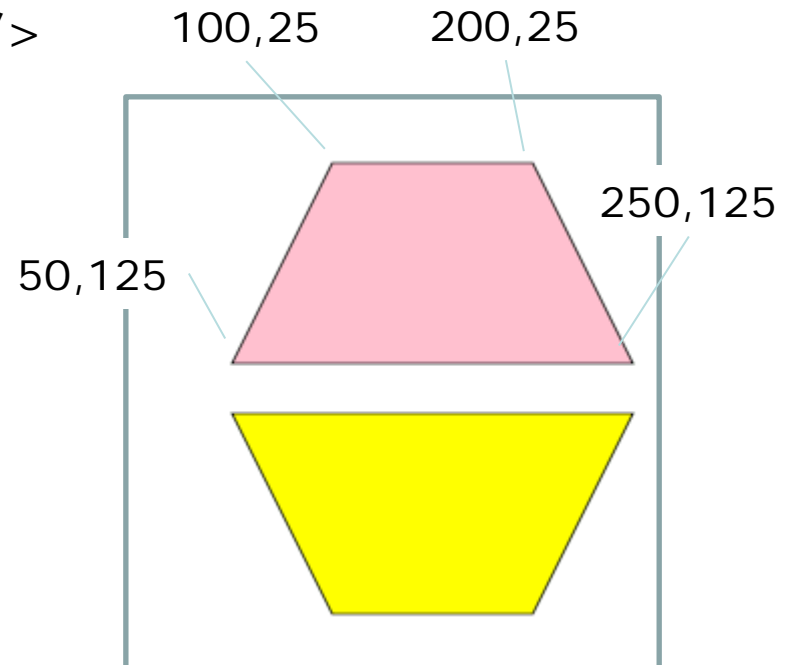
- A path is a general drawing language in itself in that it can describe any shape/path
 - M = move to
 - L = draw a straight line to
 - H = draw a horizontal line to
 - V = draw a vertical line to
 - C = draw a curve to (uses a cubic Bezier)
 - A = draw an arc to
 - Z = finish/ go back to the beginning

SVG Simple Path (08_simple_path.svg)

```
<path d="M100,25 L200,25  
        L250,125 L50,125 z"  
      style="fill: pink; stroke: black"/>
```

```
<path d="M50,150 h200 l-50,100 h-100 z"  
      style="fill: yellow; stroke: black"/>
```

- Draw polygons using closed paths
- Note the absolute and relative movements in the two quadrilaterals
- Find the coordinates of the second quadrilateral



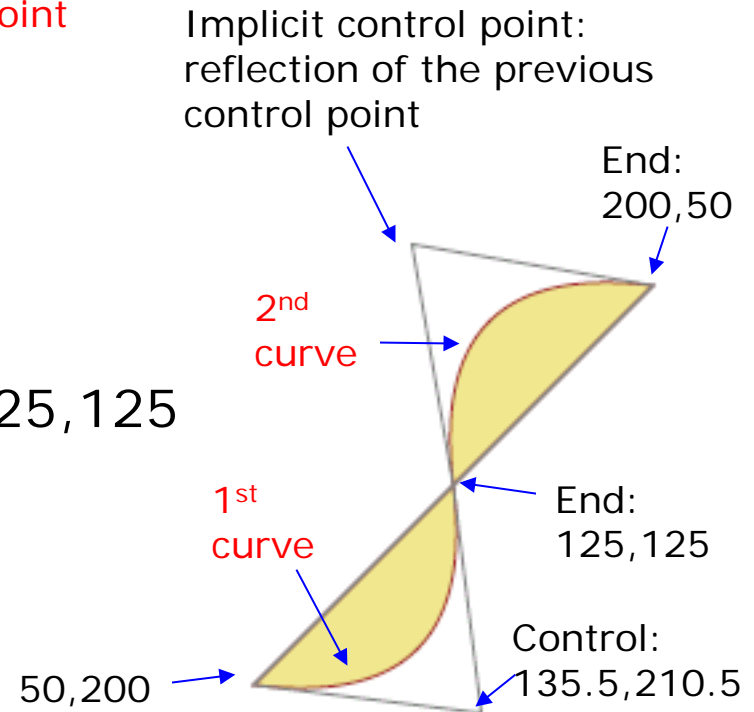
SVG Curved Path (09_curved_path.svg)

- Quadratic Bezier curve has one control point and start/end points; this example has two quadratic Bezier curves

`<path d="M50,200`
 `Q 135.5,210.5 125,125`
 `T`
 `200,50 z"`
 `style="fill: khaki;stroke: brown" />`

Start point → 50,200
Control point → 135.5,210.5
End point → 125,125
Control point → T
End point → 200,50
2nd curve

`<path d="M50,200 L135.5,210.5 L125,125`
 `L109.5,34.5 L200,50 z"`
 `style="fill: none;stroke: grey"/>`



SVG Curved Path (09_curved_path.svg)

```
<path d="M225,225 h-50
```

```
  a 50,50
```

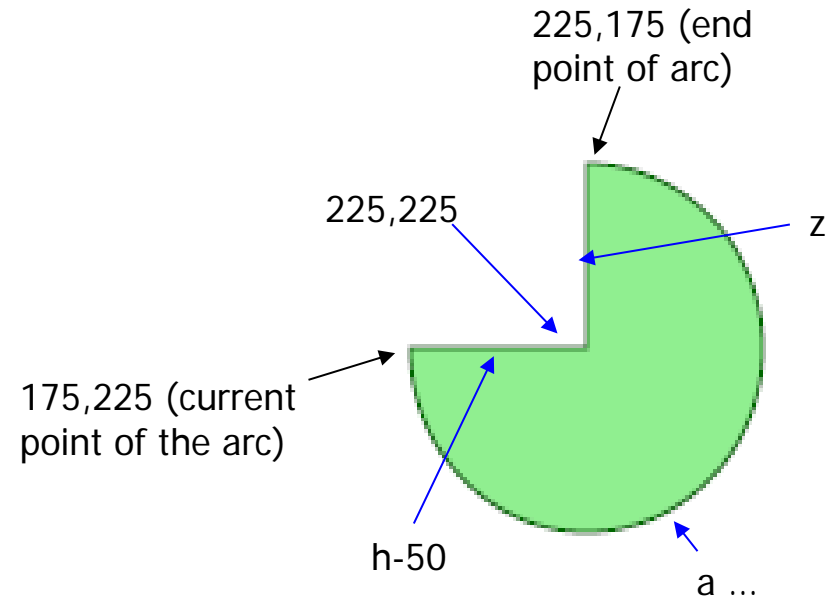
```
  0
```

```
  1,0
```

```
  50,-50 z"
```

End point x,y (relative)

```
style="fill: lightgreen; stroke: darkgreen" />
```

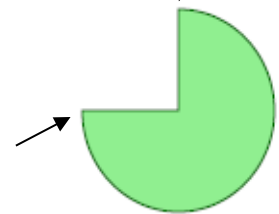


SVG Path “Elliptical Arc”

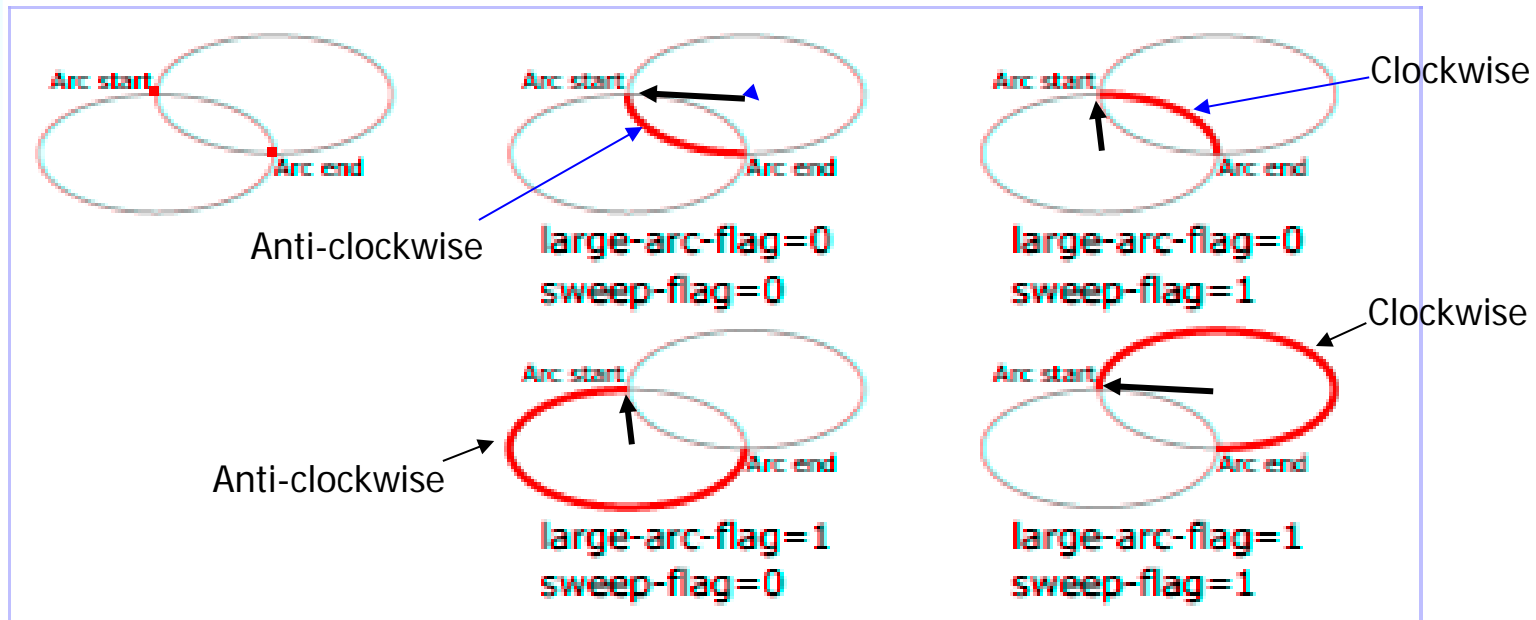
`<path d="a 50,50 0 1,0 50,-50 z"`
`style="fill:lightgreen;stroke:darkgreen"/>`

x and y radii → 50,50
 rotation=0 no rotation → 0
 large-arc-flag=1 → 1
 sweep-flag=0 (anti-clockwise) → 0
 End point x,y → 50,-50
 z" → Current (start) point (set by previous M and h operations)

End point



Current (start) point
(set by previous M
and h operations)



Using Style sheet with SVG (10_css.svg)

```
<svg width="300" height="200">
```

```
<style type="text/css">
```

```
  rect { fill: yellow;  
         fill-opacity: 0.5;  
         stroke: orange;  
         stroke-width: 5; }
```

```
  text { fill: red;  
         font-family: Arial;  
         font-size: 60px;  
         text-anchor: middle; }
```

```
</style>
```

```
<rect x="50" y="50"  
      width="200"  
      height="100"
```

```
      rx="10" ry="10" />  
Rounded corner ↗
```

```
<text x="150"  
      y="120">SVG</text>
```

```
</svg>
```



Result

Changing Style Parameters

(11_css_altered.svg)

- Redefine the style rules, and the same “rect” and “text” elements will be displayed differently

```
rect {  
    fill: lime;  
    stroke: cyan;  
    stroke-width: 20px;  
}  
text {  
    fill: blue;  
    font-family: Times;  
    font-style: italic;  
    font-size: 60px;  
    text-anchor: middle;  
}
```



Result

SVG Fill

- Many SVG elements can be filled, even things which you might not think of

```
<path style="fill: yellow">
```

values: `<color>` | `none` | `current-color`

- The "fill" property determines whether an element is filled or not, and if so, what color
- `current-color` will return the color value of the parent

SVG Fill Opacity - Transparency

```
<path style="fill-opacity: 0.25">
```

values: any value between 0 and 1

- The "fill-opacity" property determines whether an element is opaque or transparent
 - 1 is completely opaque
 - 0 is completely transparent (the element is practically invisible)
 - 0.5 means that you can see half of the element and half of the elements behind it

SVG Stroke 1/2

```
<svg width="300" height="200">
```

```
  <style type="text/css">
```

```
    text { fill: darkslateblue; }
```

```
    .style1 { fill: none;  
              stroke: skyblue;  
              stroke-width: 5;
```

```
              stroke-dasharray: 5; }
```

Style
classes

```
    .style2 { fill: none;  
              stroke: slateblue;  
              stroke-width: 5;
```

```
              stroke-dasharray: 10,5; }
```

- There are many visual parameters in SVG; the next few slides describe some of them
- Use of dasharray property in changing the pattern of a line. The pattern can be easily defined by a set of numbers

SVG Stroke 2/2

Style classes

```
.style3 { fill: none;
stroke: steelblue;
stroke-width: 5;
stroke-dasharray: 10,10,5,10; }
```

</style>

```
<text x="50" y="45">stroke-dasharray: 5</text>
<line class="style1" x1="50" y1="55" x2="250" y2="55"/>

<text x="50" y="95">stroke-dasharray: 10,5</text>
<line class="style2" x1="50" y1="105" x2="250" y2="105"/>

<text x="50" y="145">stroke-dasharray: 10,10,5,10</text>
<line class="style3" x1="50" y1="155" x2="250" y2="155"/>
</svg>
```

SVG Stroke - Output

stroke-dasharray: 5



stroke-dasharray: 10,5

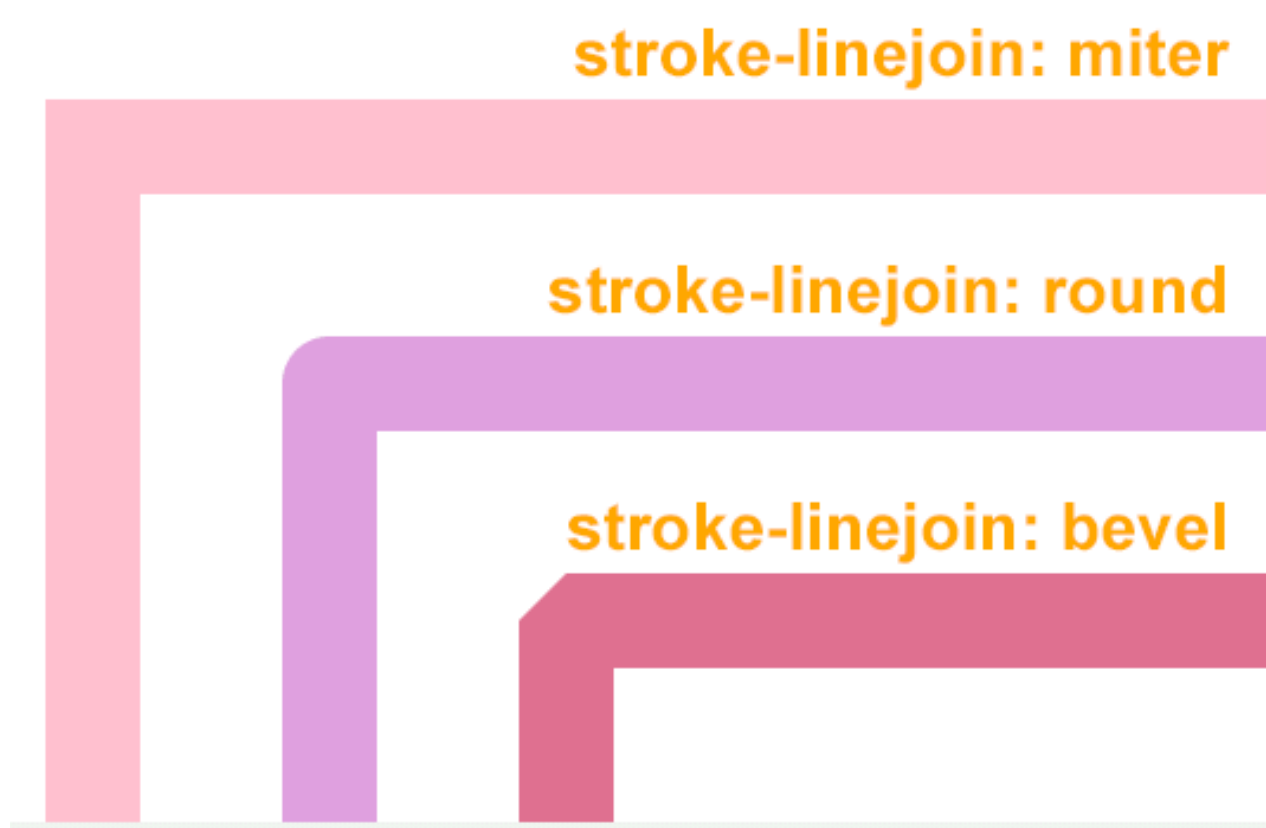


stroke-dasharray: 10,10,5,10



File – 12_stroke_dasharray.svg

SVG Stroke Linejoin



File – 13_stroke_linejoin.svg