CSIT5400
Computer Graphics

SVG Basics

David Rossiter

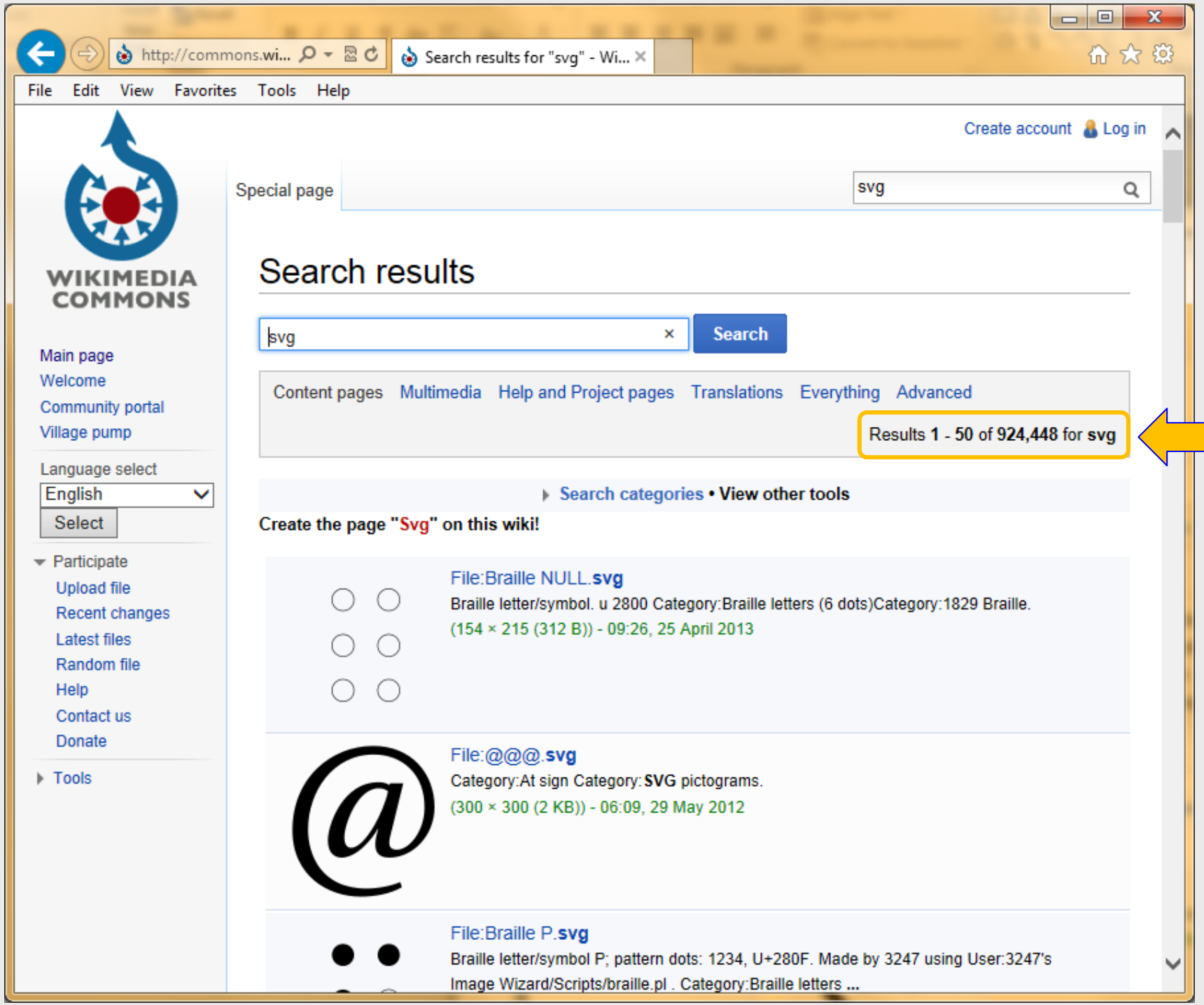
This Presentation

- In this presentation we look at the some basic elements of SVG:
  - Different versions
  - Embedded SVG
  - Text
  - Coding SVG
  - Line
  - Circle
  - Ellipse
  - Rectangle
  - Polygons
  - Polyline
  - Bitmap images
  - Grouping
  - Paths
  - Using *style*
  - Various stroke parameters

SVG on the Web

- SVG is a vector graphics language designed specifically for web pages
- It is commonly used when a vector format is a better choice than a bitmap format

- Example: all vector images in wikipedia use SVG


http://commons.wi...
Search results for "svg" - Wi...
File Edit View Favorites Tools Help
Create account
Log in
WIKIMEDIA COMMONS
Special page
svg
Search results
svg
Search
Content pages
Multimedia
Help and Project pages
Translations
Everything
Advanced
Results 1 - 50 of 924,448 for svg
Search categories • View other tools
Create the page "Svg" on this wiki!
Main page
Welcome
Community portal
Village pump
Language select
English
Select
Participate
Upload file
Recent changes
Latest files
Random file
Help
Contact us
Donate
Tools
File:Braille NULL.svg
Braille letter/symbol. u 2800 Category:Braille letters (6 dots)Category:1829 Braille.
(154 × 215 (312 B)) - 09:26, 25 April 2013
File:@@@.svg
Category:At sign Category:SVG pictograms.
(300 × 300 (2 KB)) - 06:09, 29 May 2012
File:Braille P.svg
Braille letter/symbol P; pattern dots: 1234, U+280F. Made by 3247 using User:3247's Image Wizard/Scripts/braille.pl . Category:Braille letters ...

Embedded SVG Vs. SVG Files

- SVG can be embedded inside an HTML file (ending in .htm or .html)
  or it can be in a stand-alone file (ending in .svg)
- Although SVG can be mixed with HTML, sometimes there are limitations (e.g. some browsers won't let JavaScript work with mixed HTML and SVG)
- All the 5400 examples use stand-alone SVG files
- You'll use that approach for the assignment also

SVG Text

- Using this header all the SVG in this PPT will be shown correctly in the latest browsers

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

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


</svg>
```

- If you want the SVG to have a particular width and height, add something like this in the header:

```
width="800" height="600"
```

File – 01_simple_text.svg

Coding SVG

- For any SVG code, the order of parameters is not important, for example:

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

has exactly the same meaning as:

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

- For any SVG code, visual parameters can go inside or outside *style*, for example:

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

has exactly the same meaning as:

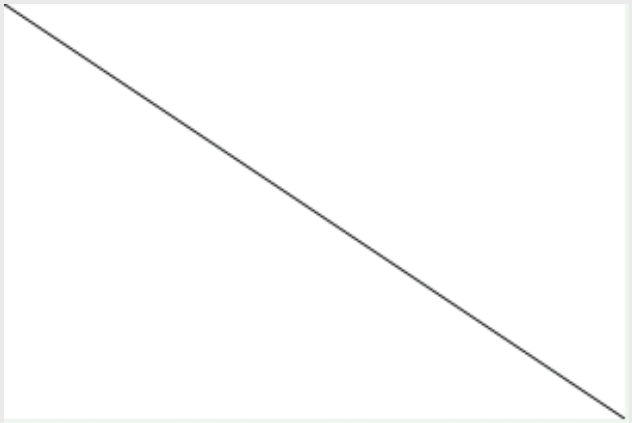
```
<text x="10" y="300" font-size="60px" fill="red" >
```

Line

- To save space in these PPT slides the header is shown only in the first example, a few slides ago

```
<svg>

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

</svg>
```

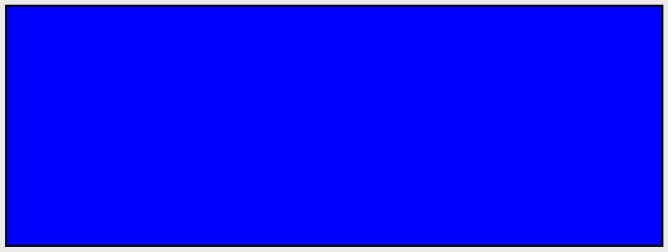
- ‘Stroke’ means ‘line’



File – 02_line.svg

Rectangle

```
<svg>

   <rect width="700" height="100" x="0" y="200"
       style="fill:blue"/>

</svg>
```

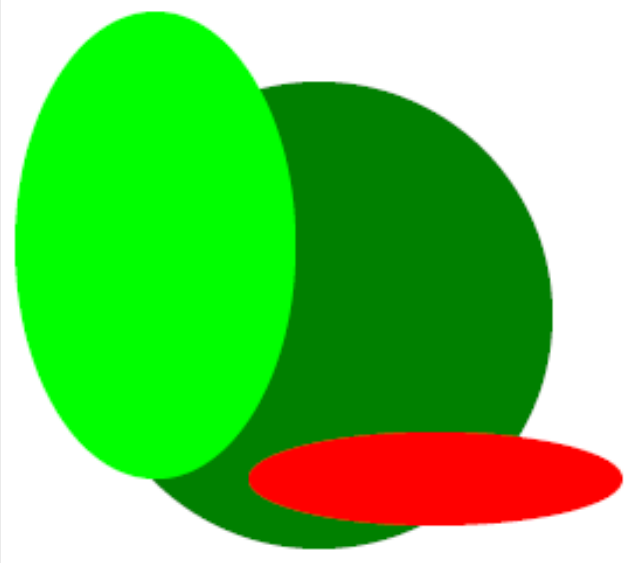


File – 03_blue_rectangle.svg

Circle and Ellipse

```
<!--
This example shows a green circle covered
by a bright green ellipse and a red ellipse
-->
<svg>

   <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>
```

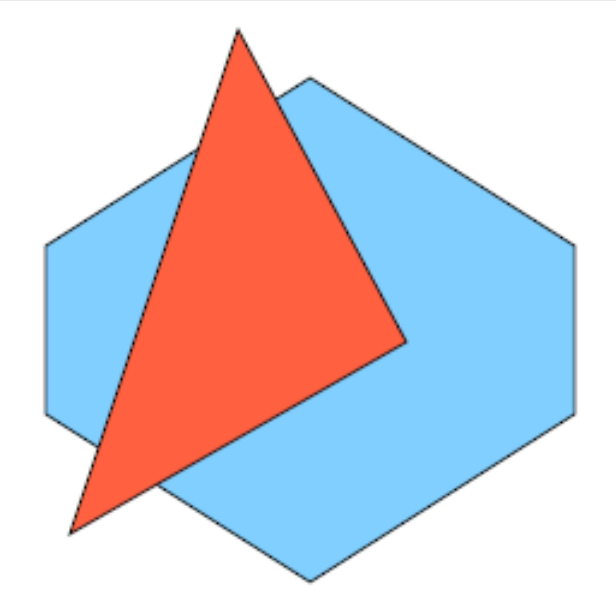
- An SVG comment



File – 04_circle_ellipse.svg

Polygons

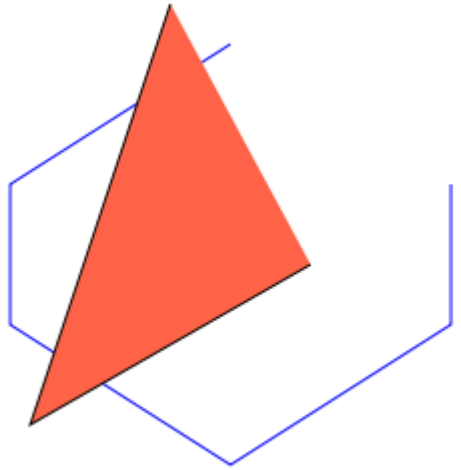
```
<!--
This example shows two polygons. One is a triangle and the
other is a polygon with 6 points.
-->
<svg>

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

   <polygon points="150,40 40,110 40,180
     150,250 260,180 260,110"
   style="fill:lightskyblue;stroke:black"/>

</svg>
```



File – 05_polygons.svg

PolyLine

```
<!--
This example shows two polylines. One is a triangle and
the other is a polygon with 6 points with open ends.
-->
<svg>

   <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"/>

</svg>
```

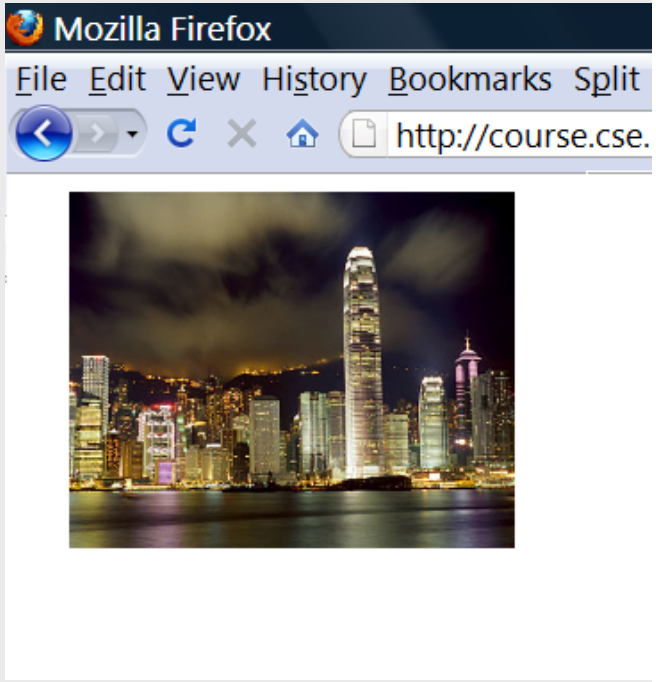


File – 06_polylines.svg

Bitmap Images

- You can use bitmap images inside SVG

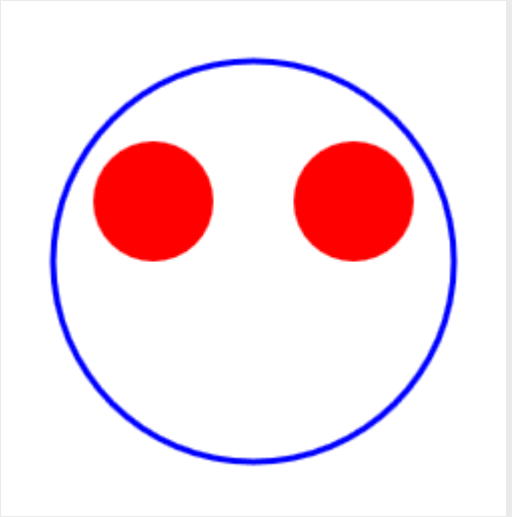
```
<svg>

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

</svg>
```



File – 07_image.svg

Grouping Things Together

- You can group any SVG things together
- When you group things together, they *look* the same
- However, if you give the group a name (=an id) then your JavaScript code can manipulate the whole group
- For example, you can move the whole group with 1 or 2 lines of JavaScript code



File – 08_group.svg

```
<svg>
  <g id="my_group_name">
      <circle cx="100" cy="120" r="30" style="fill:red"/>
      <circle cx="200" cy="120" r="30" style="fill:red"/>
      <circle cx="150" cy="150" r="100"
          style="fill:none;stroke:blue;stroke-width:3"/>
  </g>
</svg>
```

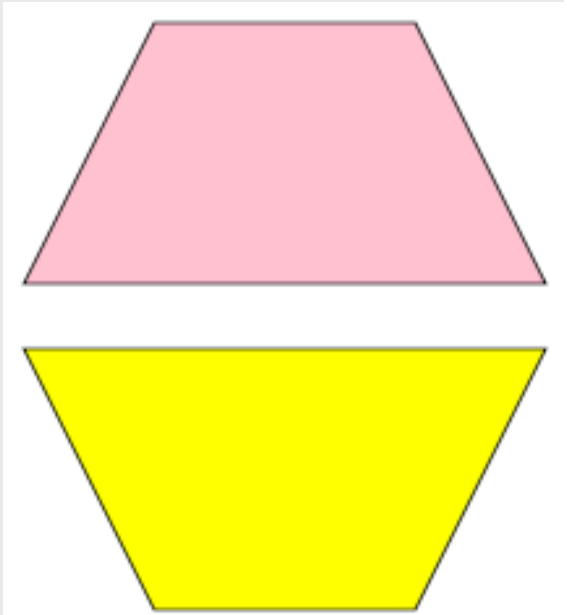
Creating Paths

- If the letter is lower case, it is a relative movement
- If the letter is upper case, it is an absolute position

- A *path* is a kind of drawing language in itself
- You can describe any shape/path using these:
  - 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 curve)
  - S = draw a smooth curve to
  - Q = quadratic Bezier curve
  - T = draw a smooth quadratic Bezier curve to
  - A = draw an arc to
  - Z = finish/ go back to the beginning

SVG Simple Path

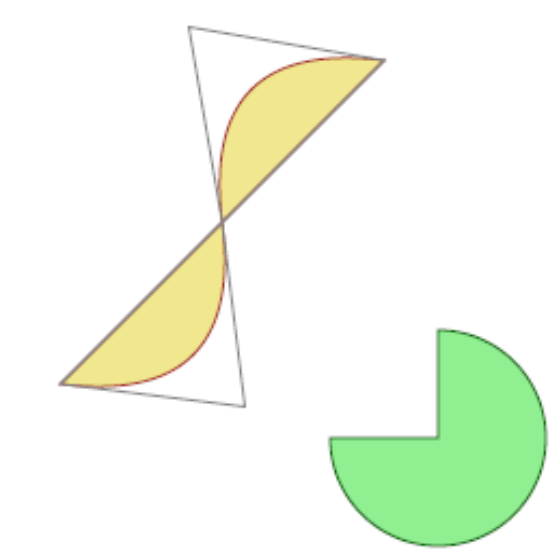
```
<!--
This example shows two simple paths.
Each time, a quadrilateral is built
using four points in the path.
-->
<svg>

   <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"/>

</svg>
```



File – 09_simple_path.svg

SVG Curved Path

```
<svg>

   <path d="M50,200 Q135.5,210.5 125,125 T200,50 z"
    style="fill:khaki;stroke:brown"/>

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

   <path d="M225,225 h-50 a50,50 0 1,0 50,-50 z"
   style="fill:lightgreen;stroke:darkgreen"/>

</svg>
```



File – 10_curved_path.svg

Using Style 1/2

- Sometimes you may see *style* used in a web page
- Here we look at using a style section containing lots of style instructions

```
<!--
   This example shows the use of a style sheet with SVG. The
   rectangle and the text are displayed with the style rules
   specified in the style instructions.
-->

<svg>

   <style type="text/css">
        rect {
                fill: yellow;
                fill-opacity: 0.5;
                stroke: orange;
                stroke-width: 5;
        }
```

- A *style sheet* is stored at the top of the file in a `<style>` … `</style>` structure
- Style sheets are optional, but in general they are a good idea
- They are not required for assignment 1

Using Style 2/2

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

</svg>
```

File – 11_css.svg

Different Style Parameters

- Don’t like it? Simply change the style information:

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

File – 12_css_altered.svg

- Note that the *content* and *structure* remains the same, just the visual display is changed

SVG Stroke 1/2

- Remember ‘stroke’ means ‘line’

```
<!--

This example shows the use of dasharray property in changing the pattern
   of a line. The pattern can be easily defined by a set of numbers.

-->

<svg>

   <style type="text/css">
        text {
                fill: darkslateblue;
        }

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

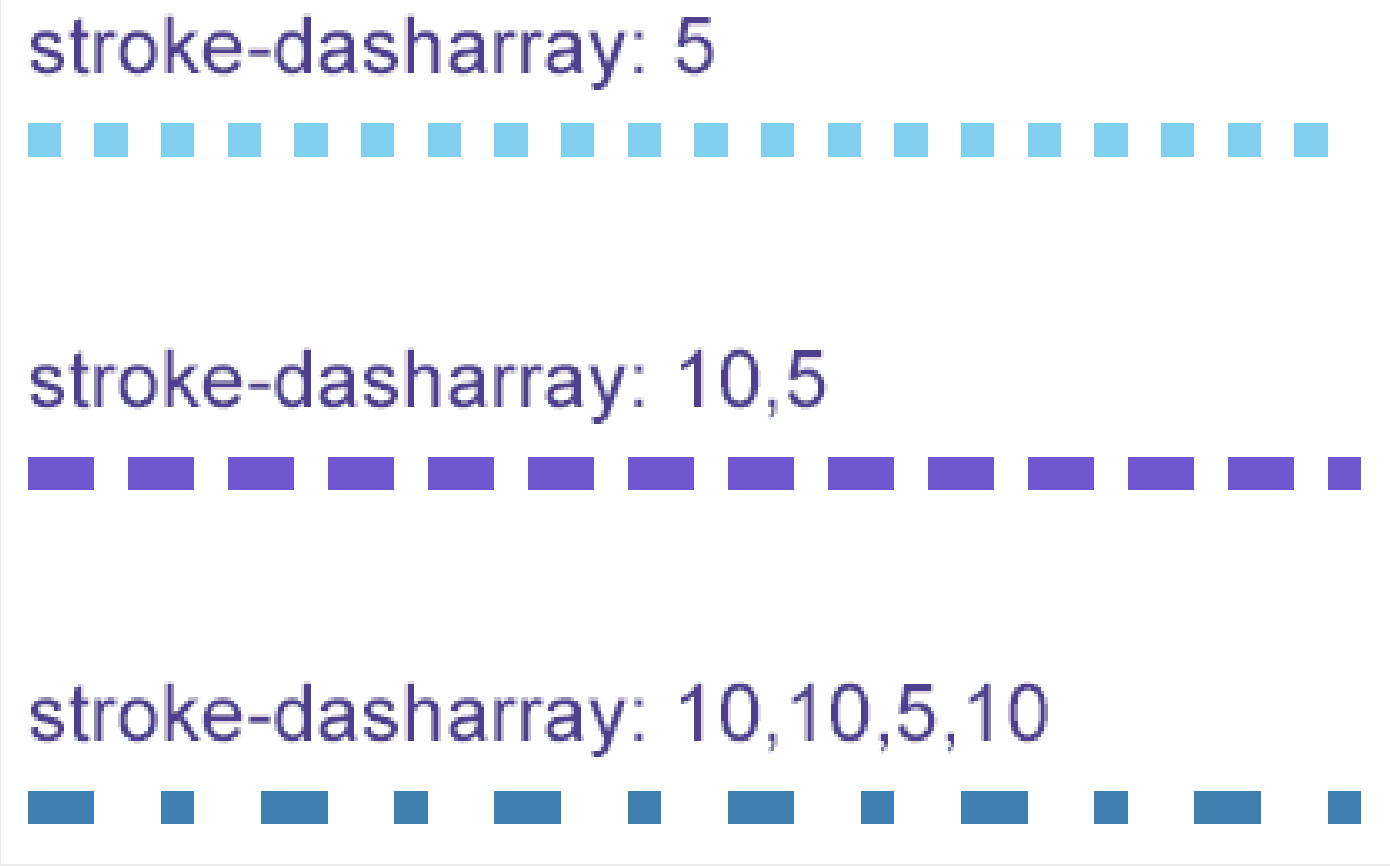
- There are many visual parameters you can play with in SVG, the next few slides give you an idea of some
- Here we focus on some of the things you can do with lines

```
.style2 {
        fill: none;
        stroke: slateblue;
        stroke-width: 5;
        stroke-dasharray: 10,5;
}
```

SVG Stroke 2/2

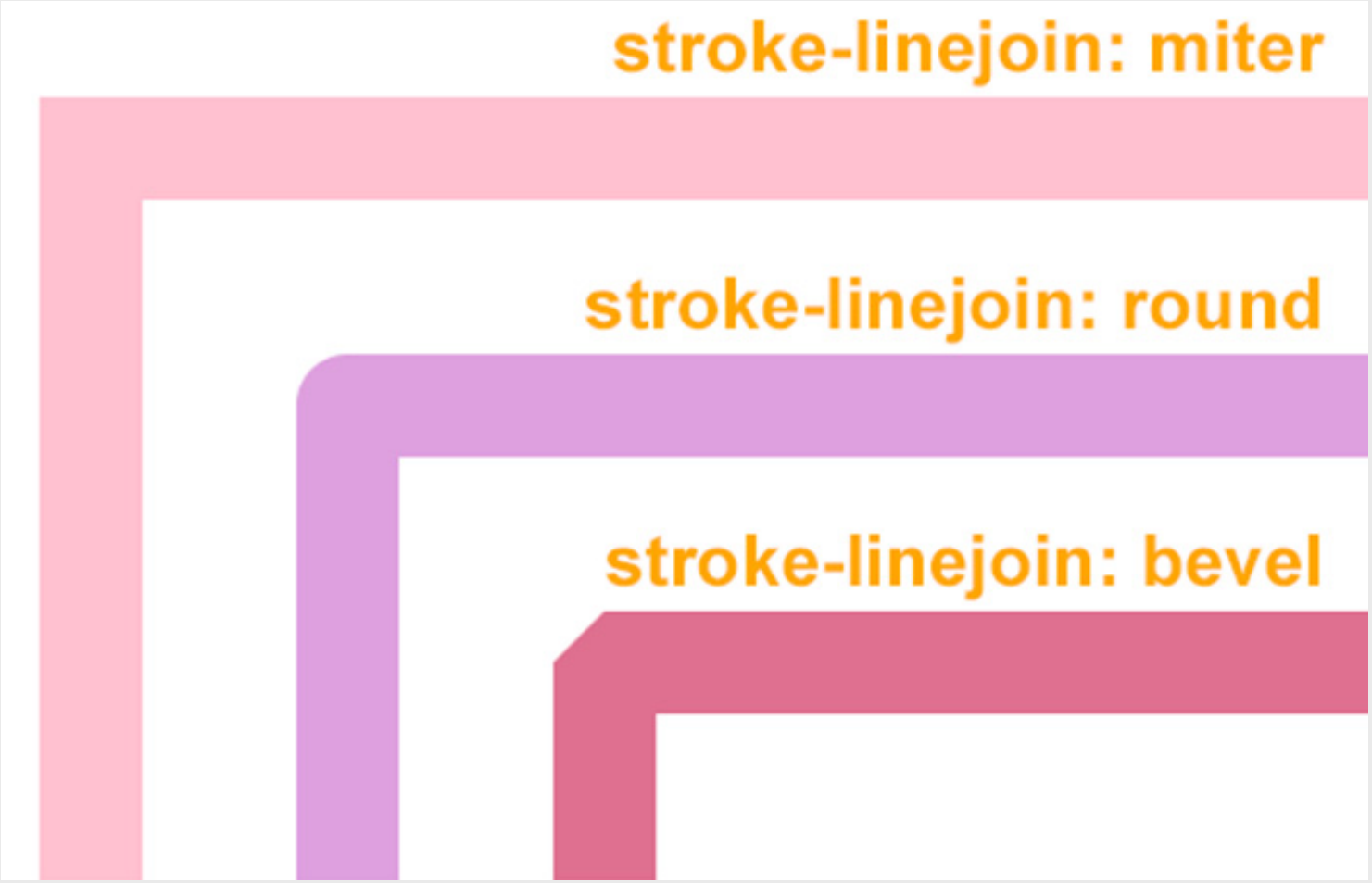
```
        .style3 {
                fill: none;
                stroke: steelblue;
                stroke-width: 5;
                stroke-dasharray: 10,10,5,10;
        }
    </style>

    <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>
```

The Result



File – 13_stroke_dasharray.svg

SVG Stroke Linejoin



File – 14_stroke_linejoin.svg