COMP4021
Internet Computing

SVG Basics

David Rossiter & Gibson Lam

SVG Basics

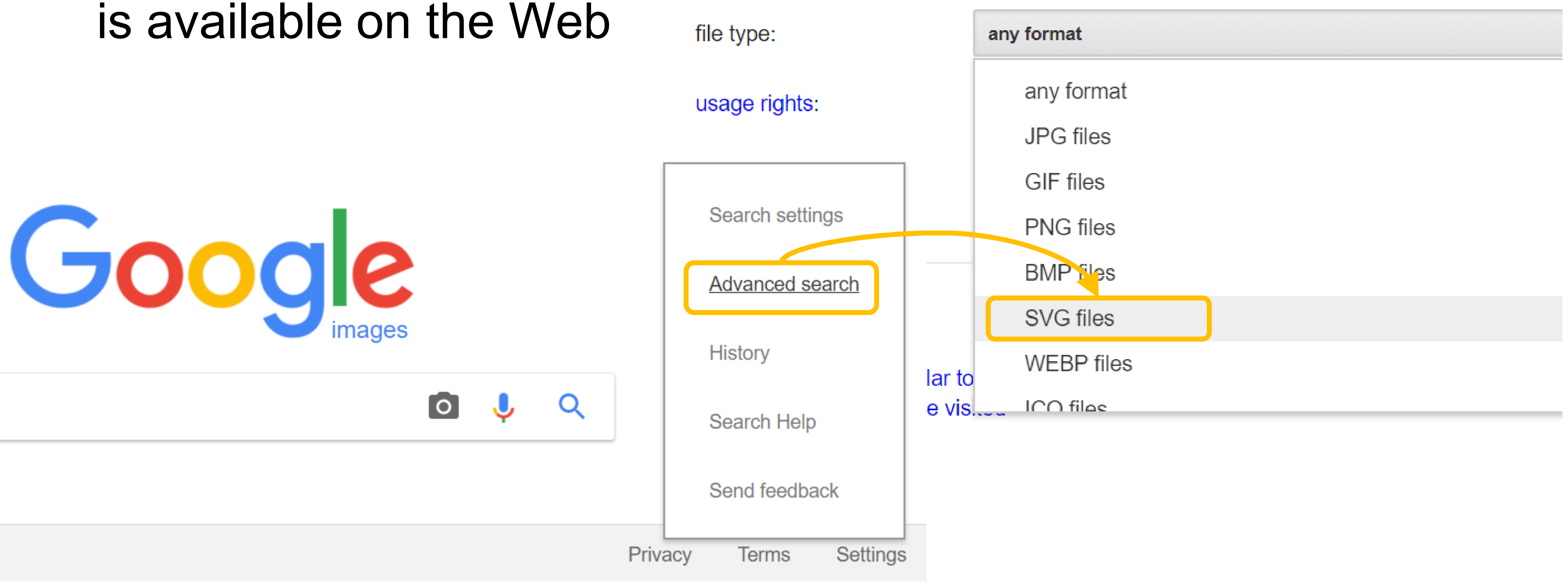
- SVG is a vector graphics language for web pages
- Logos, figures and charts are ‘easy’ for SVG to make

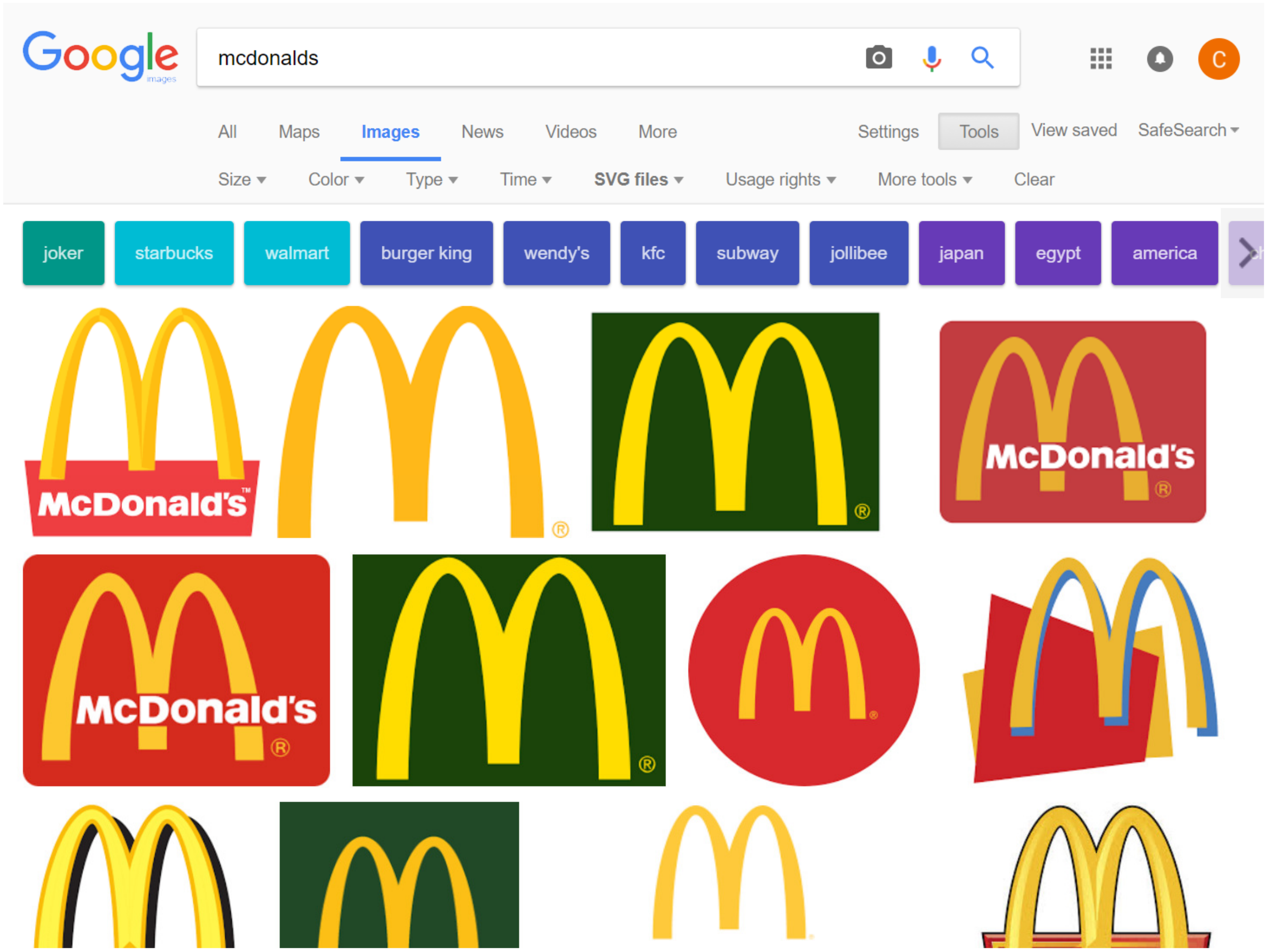
- In this presentation, we will look at how to create SVG content and the many different elements in SVG

SVG Images on the Web

- Before we look at how we create SVG, let's see what is available on the Web

- You can look for SVG images in Google by changing the search settings





mcdonalds

Google images
All
Maps
Images
News
Videos
More
Settings
Tools
View saved
SafeSearch
Size
Color
Type
Time
SVG files
Usage rights
More tools
Clear
joker
starbucks
walmart
burger king
wendy's
kfc
subway
jollibee
japan
egypt
america
McDonald's
McDonald's
McDonald's

Standalone or Embedded SVG

- SVG can be embedded inside an HTML file
  - ending in `.htm` or `.html`
- Or it can be in a standalone file
  - ending in `.svg`

- Most examples in this presentation are standalone SVG files

A Standalone SVG File – HKUST.svg

```
<?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 xmlns="http://www.w3.org/2000/svg"
     version="1.1"
     width="390px" height="600px"
     viewBox="0 0 390 600">
```

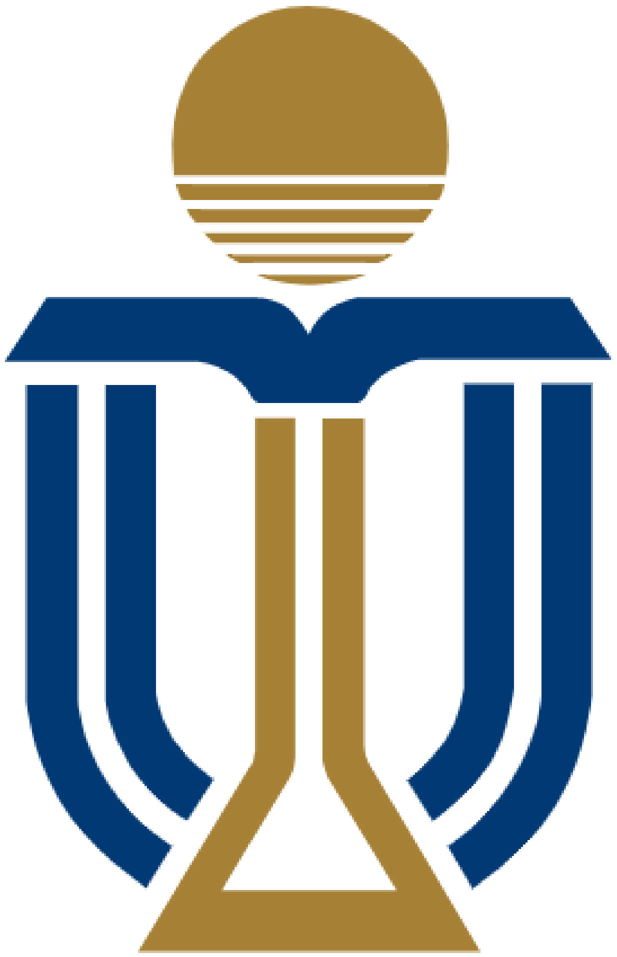
...SVG content...

```
</svg>
```

regardless of resolution, zooming in and out, and the viewing area, the svg element will look as good as it can be.

Embedding HKUST.svg in a Webpage

The SVG HKUST logo



```
<!DOCTYPE html>
<html xmlns='http://www.w3.org/1999/xhtml'>
<head>
    <title>HKUST Logo</title>
</head>
<body>
    <h1>The SVG HKUST logo</h1>
    <svg xmlns="http://www.w3.org/2000/svg"
         version="1.1"
         width="185px" height="300px"
         viewBox="0 0 390 600">

   ...SVG content...

   </svg>
</html>
```

Basic SVG Structure

- Here is a basic SVG structure:

```
<svg xmlns="http://www.w3.org/2000/svg"
     width="400" height="300">

...SVG content...

</svg>
```

An SVG area of 400 by 300 pixels

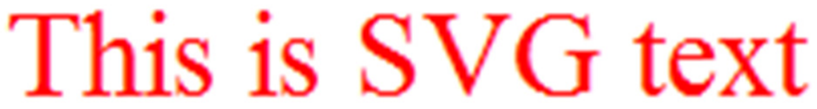
Simple SVG Text

- You can add text to an SVG area using the `text` tag:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <text x="10" y="300"
        style="font-size:60px;fill:red">
    This is SVG text
  </text>
</svg>
```

set the exact position of the text

You can leave out the size if you want SVG to automatically set the size for you


This is SVG text

About SVG Attributes

- Visual parameters can go inside or outside *style*, for example:

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

这是inline CSS的做法

has exactly the same meaning as:

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

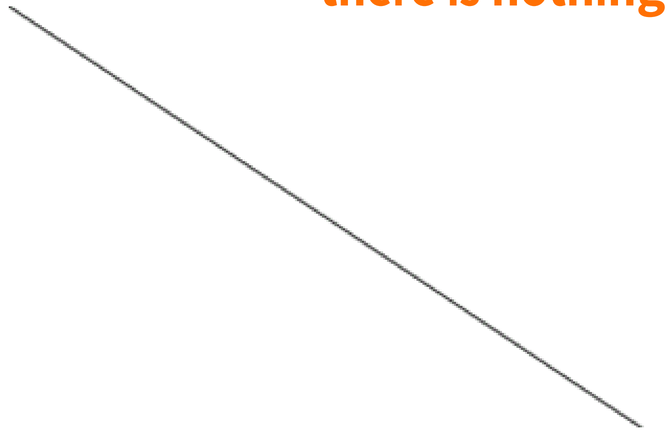
style parameters can be taken out

Lines

- Using the `line` tag:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <line x1="0" y1="0" x2="300" y2="200"
        style="stroke:black"/>
</svg>
```

no separate end tag, because there is nothing in the middle



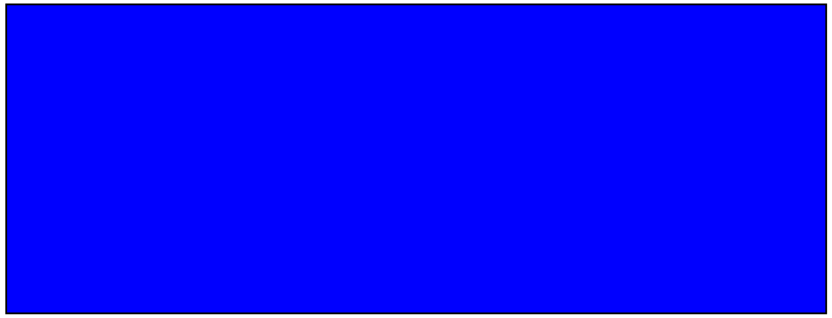
- The style property 'stroke' means 'line' (i.e. the line colour)

Rectangles

- Using the `rect` tag:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect width="700" height="100"
        x="0" y="200" style="fill:blue"/>
</svg>
```

no separate end tag



Closing Tags

- You may notice that the examples on the previous two slides do not have a closing tag for the `line` and `rect` tags
- If the tag does not have enclosing content, you should use a shortcut '/' at the end of the tag to close it

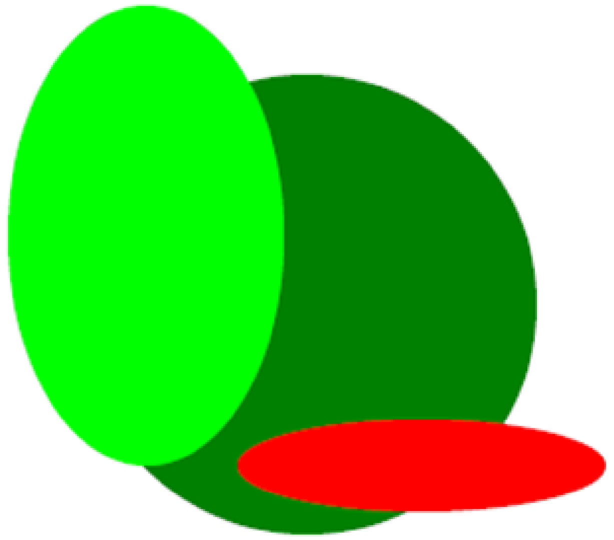
```
<line ...attributes... />
```

Close the tag

Circles and Ellipses

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

- Here we use the `circle` and `ellipse` tags to create a green circle covered by two ellipses



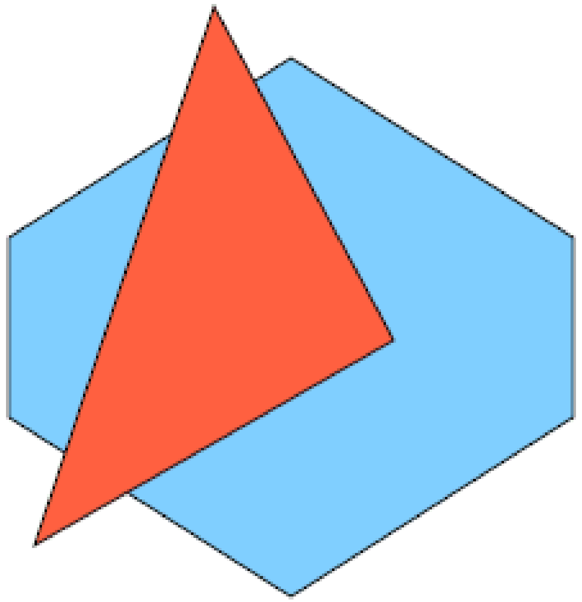
Polygons

- Using the polygon tag:

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

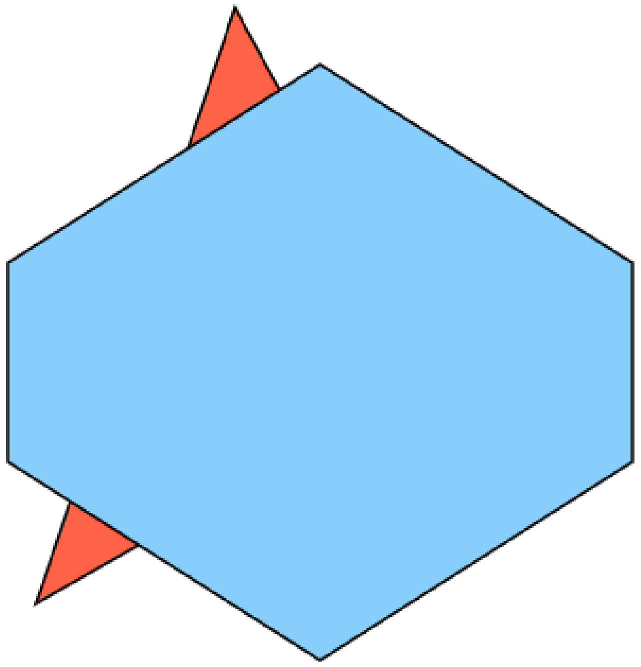
give a series of x,y

the content drawn later will appear on top of the previous things.



Drawing Order

- When there are graphic elements overlapping each other, the drawing order follows the order that they appear in the SVG file
- For example, if we swap the order of the two polygons in the previous example, the blue polygon will cover the red one

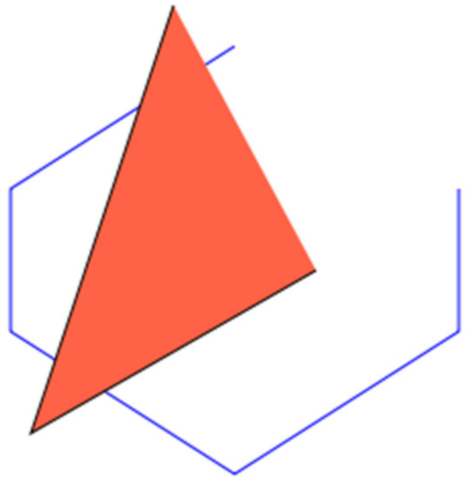


Polylines

to make it closed, we need to add a last point equal to the first point.

- Polylines are just polygons with an open end

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polyline points="150,40 40,110 40,180
                    150,250 260,180 260,110"
            style="fill:none;stroke:black"/>
  <polyline points="120,20 50,230 190,150"
            style="fill:tomato;
                   stroke:black"/>
</svg>
```



Using Bitmap Images

- You can use bitmap images inside SVG

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
  <image xlink:href="hong_kong.jpg"
         x="10" y="10" width="300" height="300"/>
</svg>
```

link to a bitmap image

You need to add this inside the svg tag in order to use a link

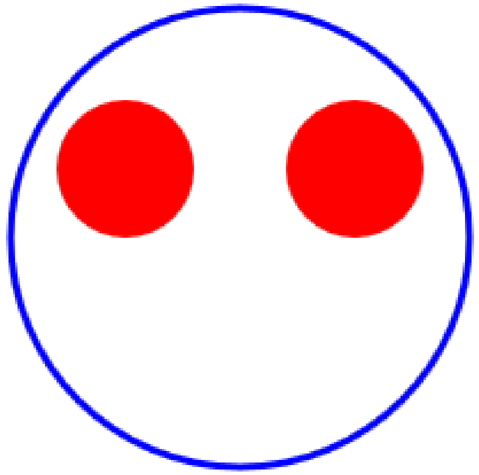
Grouping Things Together

- You can group any SVG things together using the `g` tag
- `g` means *a group of SVG things*
- When you group things together and 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

Creating a Group

- Here is a simple group with three circles:

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



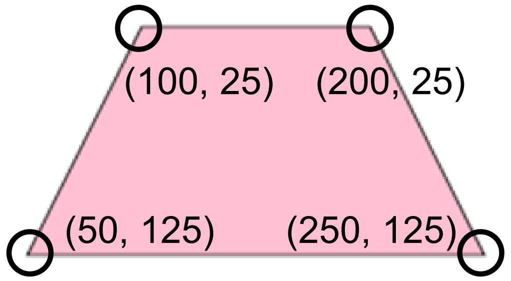
Creating Paths

- A path is a kind of drawing language in itself
- You can describe any shape/path using these:

`M`	Move to **Do not draw during moving. Just move the pen to a starting place.**
`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

Move to the starting point and draw a line during moving

Path Examples

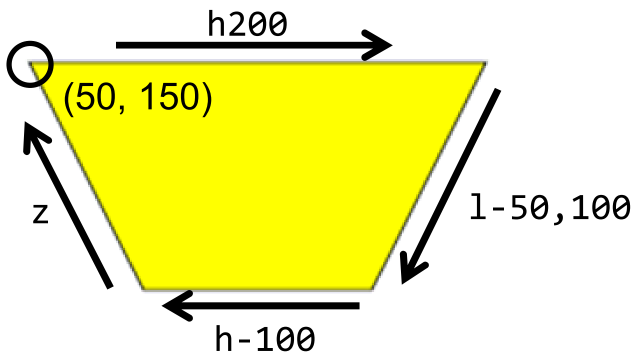


- For example, here is a path:

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

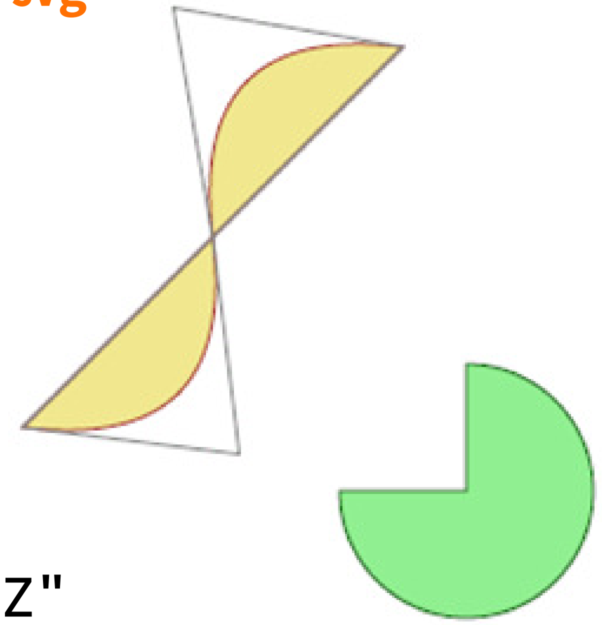
- You can change the command letters to small letters; in that case, the commands will use relative movement, like this:

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



most people won't type in these advanced commands
by themselves. They will use some software and generate the svg
code automatically.

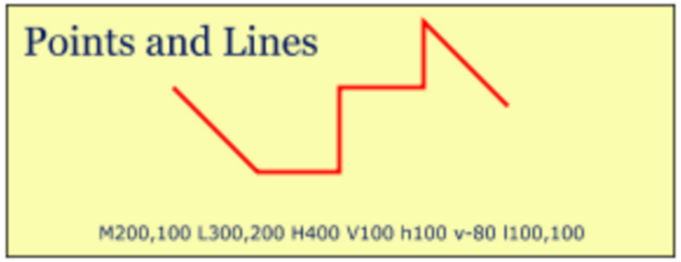
Curved Path Examples



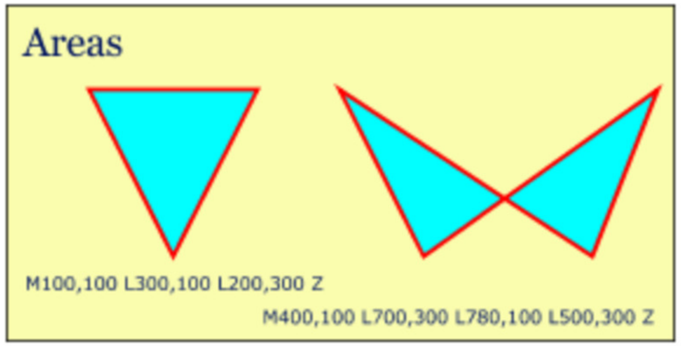
- Curves can be generated using quadratic/cubic Bezier or simple arc:

```
<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"/>
```

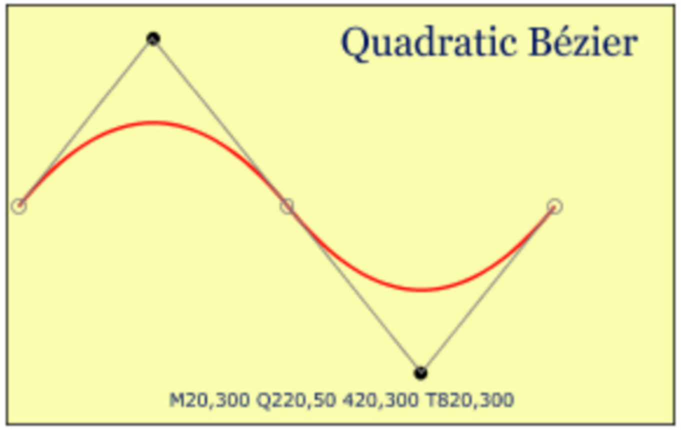
- Paths can be used to make any lines/shapes - if you have to make some complicated lines then the best thing is to use some software



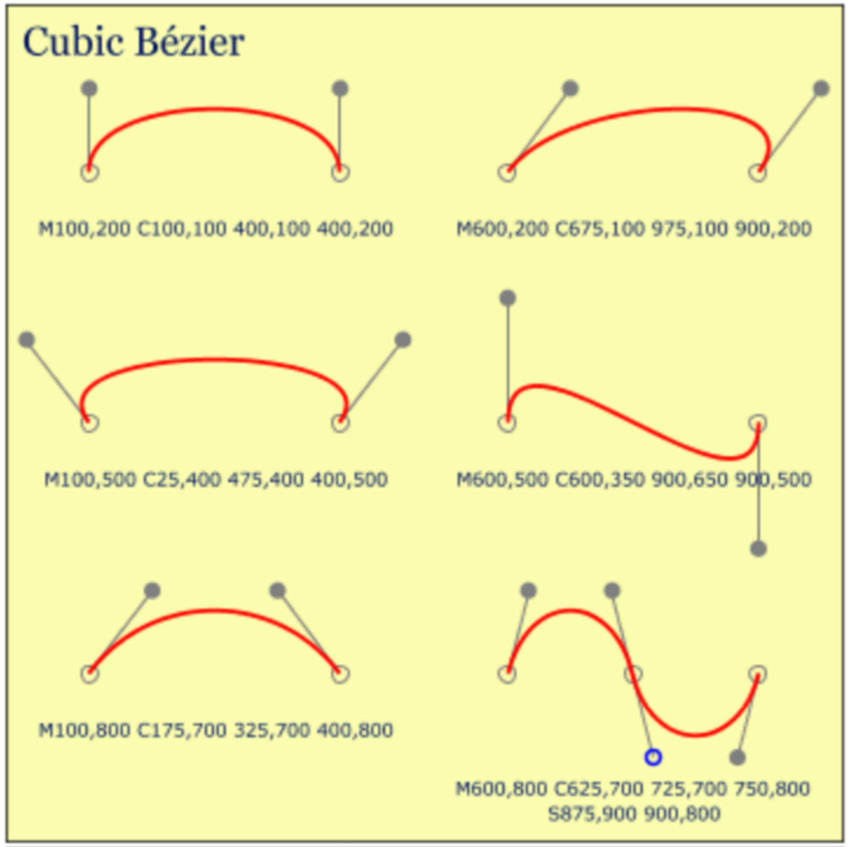
```
M200,100 L300,200 H400 V100 h100 v-80 l100,100
```



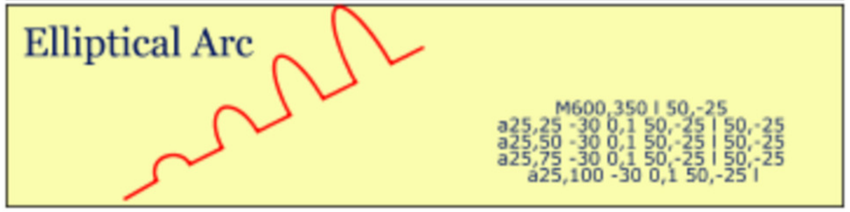
```
M100,100 L300,100 L200,300 Z
M400,100 L700,300 L780,100 L500,300 Z
```



```
M20,300 Q220,50 420,300 T820,300
```



```
M100,200 C100,100 400,100 400,200
M600,200 C675,100 975,100 900,200
M100,500 C25,400 475,400 400,500
M600,500 C600,350 900,650 900,500
M100,800 C175,700 325,700 400,800
M600,800 C625,700 725,700 750,800
S875,900 900,800
```



```
M600,350 l 50,-25
a25,25 -30 0,1 50,-25 l 50,-25
a25,50 -30 0,1 50,-25 l 50,-25
a25,75 -30 0,1 50,-25 l 50,-25
a25,100 -30 0,1 50,-25 l
```

Using Style

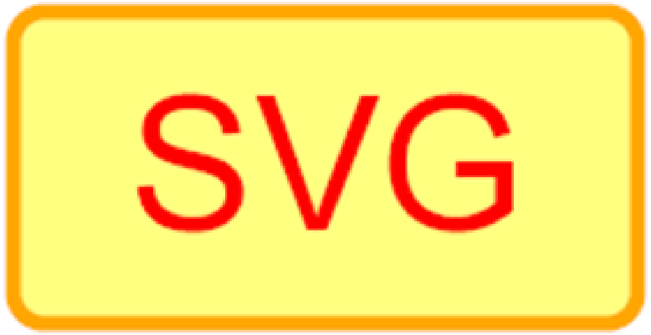
- You have previously seen ‘style’ used in a web page
- SVG can also use style

```
<svg xmlns="http://www.w3.org/2000/svg">
  <style type="text/css">
    rect {      selector: all rectangles
      fill: yellow;
      fill-opacity: 0.5;
      stroke: orange;
      stroke-width: 5;
    }
    text {
      fill: red;
      font-family: Arial;
      font-size: 60px;
      text-anchor: middle;
    }
  </style>
  ...
</svg>
```

Using the Style

- Then you can write simple SVG
  and the style rules will be applied:

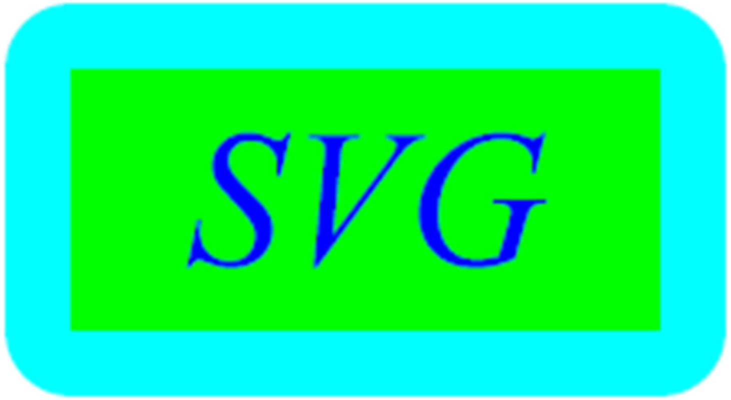
```
<svg xmlns="http://www.w3.org/2000/svg">

  ...the style section on the previous slide...

  <rect x="50" y="50"
        width="200" height="100"
        rx="10" ry="10"/> round corner
  <text x="150" y="120">SVG</text>

</svg>
```



Changing the Style

- If you don’t like the visual style you can simply change the style rules, like this:

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



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

Defining Things in SVG

- There is a ‘definitions’ area of SVG
- Basically, you define something (once) and then you can use in the rest of the SVG (as many times as you want)
- Here are some of the things you can define:
  - Gradients
  - Patterns
  - Clipping paths
  - Filters

We will briefly look at gradients in this discussion

The Defs Area

- You first define something you want to use in the `defs` area, using a specific id
- An SVG element can then use the defined thing by referring to that id

```
<svg ...>
  <defs>
    <... id="def_id">
       ...
    </...>
  </defs>

  <rect ...  #def_id  ... />
</svg>
```

SVG Gradients

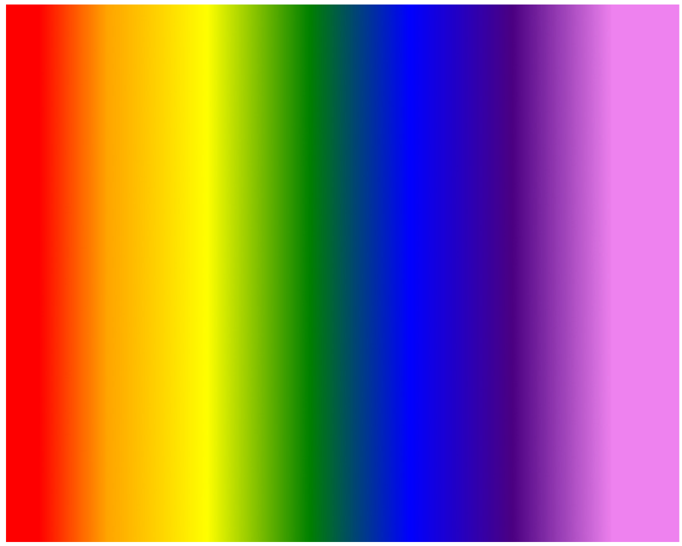
- The SVG `fill` attribute we have used so far is a solid flat colour, for example:

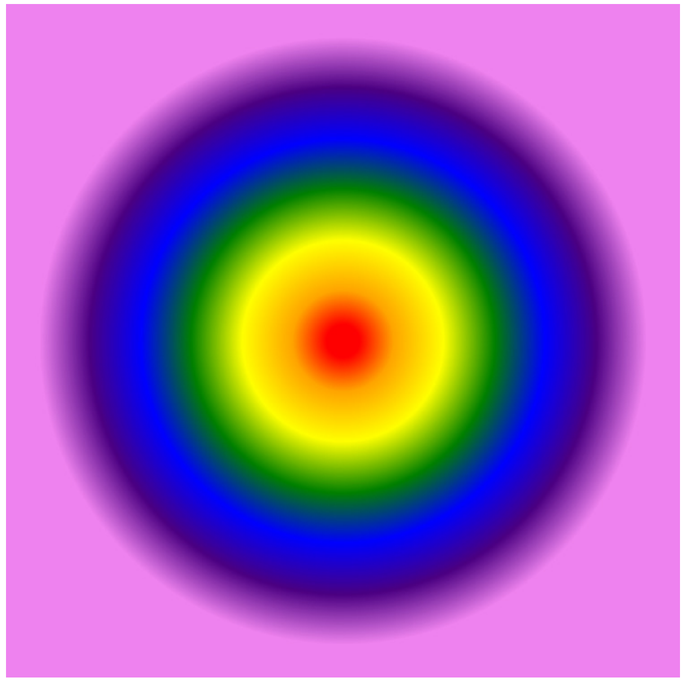
```
<rect width="700" height="100"
      x="0" y="200" fill="blue"/>
```

- Using SVG gradients you can create a smoothly changing gradient of colours in the colouring of SVG elements

Two Types of Gradient

- There are two types of SVG gradient:
  - *Linear gradients*, that change colour linearly along a line
  - *Radial gradients*, that change colour from the centre of a shape and radiate outwards





Using the Defs Area

```
<svg ...>
  <defs>
    <linearGradient id="my_gradient">
      ...gradient colours...
    </linearGradient>
  </defs>
  ...
  <rect fill="url(#my_gradient)" .../>
  ...
</svg>
```

- The gradients are defined in the `defs` area and then referred to using their ids

Gradient Colours

- The content of a gradient is a list of colours defined in different positions
- Each colour is specified by the `stop` tag with a particular `offset` and `stop-color`

不是0到100！！！

  – Offset is the positioning of the colour from 0 to 1
  – Stop colour is the colour used at that position
- For example, here is a red colour positioned at 50% of a gradient:

```
<stop offset="0.5" stop-color="red"/>
```

A Simple Linear Gradient

- Here is an example linear gradient with two colours:

```
...
<linearGradient id="gradient">
    <stop offset="0" stop-color="white"/>
    <stop offset="1" stop-color="black"/>
</linearGradient>
...
```

顺序：从左至右

```
<rect fill="url(#gradient)" .../>
```

- Applying the above gradient to a rectangle results in the fill shown on the right

A Simple Radial Gradient

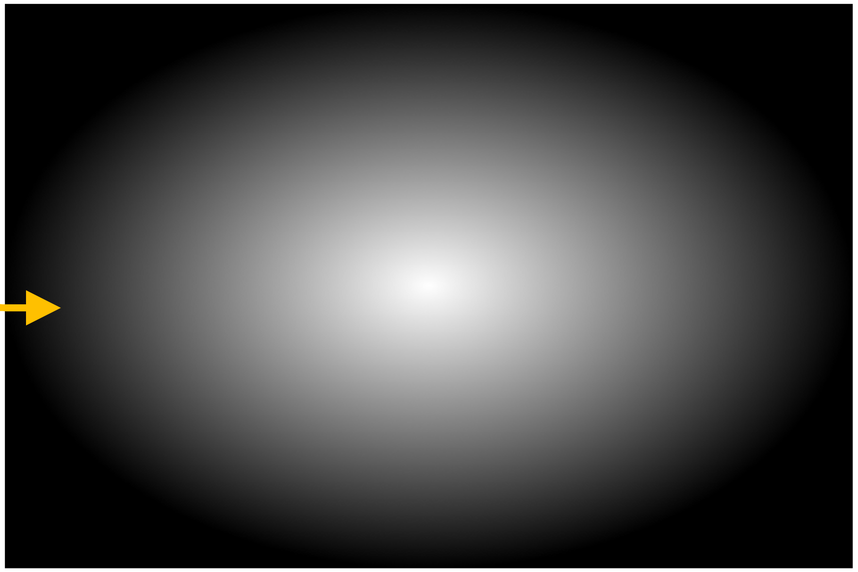
- The previous example can be easily changed to a radial gradient like this:

顺序：从中心到外

```
<radialGradient id="gradient">
    <stop offset="0" stop-color="white"/>
    <stop offset="1" stop-color="black"/>
</radialGradient>
```

```
<rect fill="url(#gradient)" .../>
```

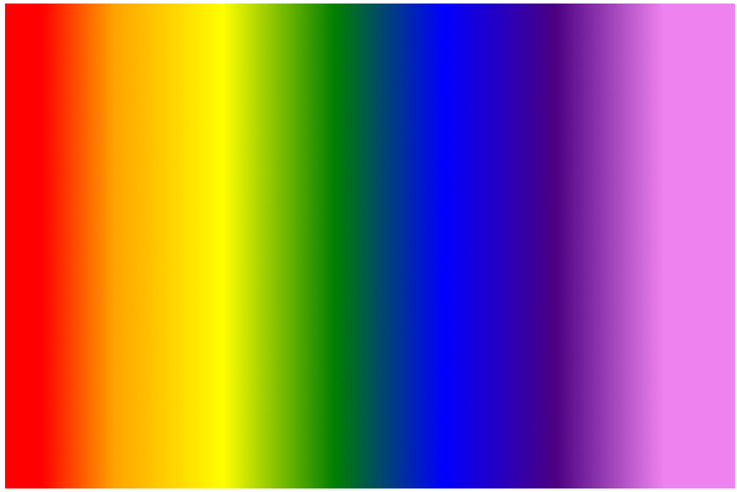
The radial gradient has an oval shape if the SVG element is a rectangle



A Rainbow Gradient

- With more colours you can create a highly varied gradient:

```
<linearGradient id="gradient">
    <stop offset="0.05" stop-color="red"/>
    <stop offset="0.15" stop-color="orange"/>
    <stop offset="0.3"  stop-color="yellow"/>
    <stop offset="0.45" stop-color="green"/>
    <stop offset="0.60" stop-color="blue"/>
    <stop offset="0.75"
          stop-color="indigo"/>
    <stop offset="0.9"
          stop-color="violet"/>
</linearGradient>
```

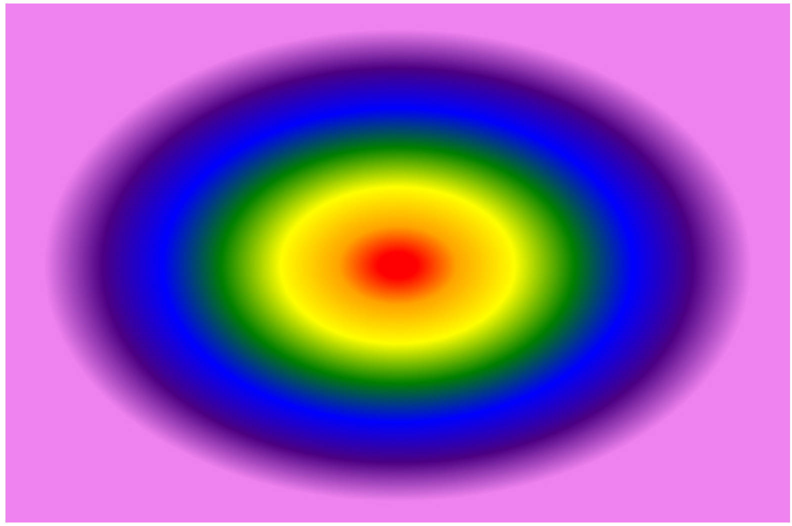


不一定非要从0和100写起，比如这个例子里面，browser will look for color definitions closest to 0 (100) as definition for 0 (100)

Rainbow Radial Gradient

- Similarly, the rainbow can become a radial gradient:

```
<radialGradient id="gradient">
    <stop offset="0.05" stop-color="red"/>
    <stop offset="0.15" stop-color="orange"/>
    <stop offset="0.3"  stop-color="yellow"/>
    <stop offset="0.45" stop-color="green"/>
    <stop offset="0.60" stop-color="blue"/>
    <stop offset="0.75"
          stop-color="indigo"/>
    <stop offset="0.9"
          stop-color="violet"/>
</radialGradient>
```



The Line in a Linear Gradient

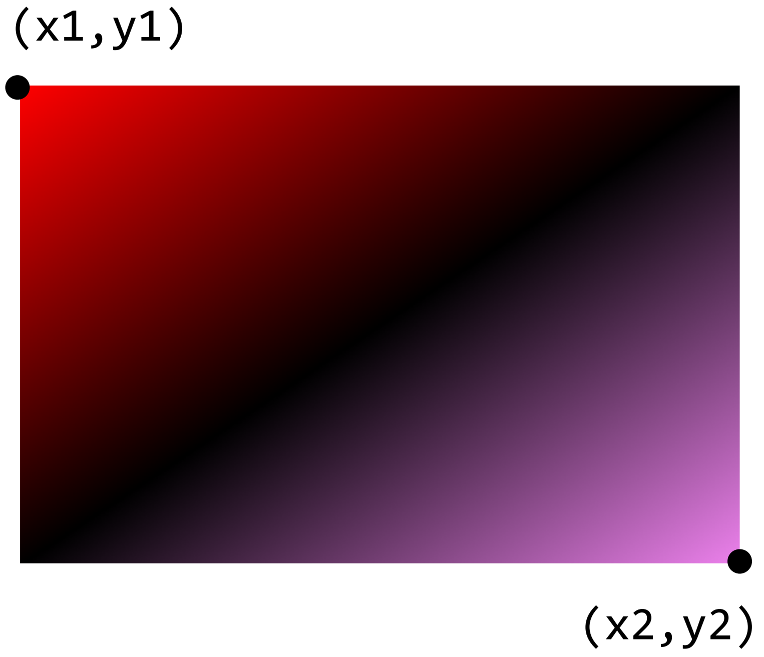
- A linear gradient, by default, changes its colour from left to right horizontally
- You can change the orientation of the line using a set of attributes: `x1`, `y1`, `x2`, `y2`
- These attributes specify a line within the gradient area and the change of colour then starts from `(x1,y1)` to `(x2,y2)`
- An example is shown in the next slide

Changing the Line

- In this example, the line inside the gradient starts from the top-left hand corner to the bottom-right hand corner of the area

注意加百分号！

```
<linearGradient id="gradient"
    x1="0%" y1="0%" x2="100%" y2="100%">
    <stop offset="0"   stop-color="red"/>
    <stop offset="0.5" stop-color="black"/>
    <stop offset="1"   stop-color="violet"/>
</linearGradient>
```



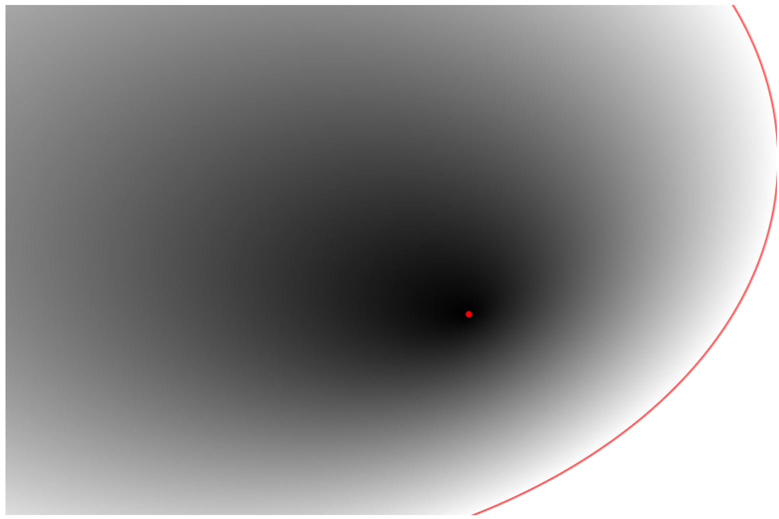
Changing a Radial Gradient

- You can similarly change the way a radial gradient works, as shown below:

```
<radialGradient id="gradient"
      fx="60%" fy="60%"
      cx="20%" cy="30%" r="80%" >
    <stop offset="0"
      stop-color="black"/>
    <stop offset="1"
      stop-color="white"/>
</radialGradient>
```

The starting point of the gradient

The target circular area



Applying a Gradient

- A gradient can be applied to anything!
- In most of the examples we apply them to a rectangle
- However, you could apply a gradient to a circle, an ellipse, a path… anything!
- And you can apply it more than once, as many times as you like