SVG

This is the vector graphics "drawing" technology with its technical and creative beauty

SVG  Inkscape  vectors
SVG = Scalable Vector Graphics is an integrated standard for drawing

Along with other standards, it replaces plugin technologies like Flash

There is an open source drawing tool called Inkscape, which outputs SVG, and which can be used instead of proprietary tools like Illustrator
Inkscape

The leading open source tool for editing SVG drawings is *Inkscape* (or possibly online tools such as svg-edit)

You can also use an editor with XML support for really simple drawings

The tiger is a famous SVG drawing

SVG concentrates on drawing, but also has moderate support for text, images, filters and animation
Status of SVG

See [wikipedia: SVG](https://en.wikipedia.org/wiki/SVG) for overview, news on status, links

See [the 1.1 standard](https://www.w3.org/TR/SVG11/) and the MDN tutorial

The SVG standard is fairly stable, with version 1.1 being the main current version, with version 2 under way

There is reasonable support in modern browsers, with several fallback possibilities for old browsers.

For animation, JS is used to change the SVG objects (SVG has its own built-in animation, but it is not fully programmable and obsolescent)
Inkscape was developed as a graphic designer's drawing tool, and is excellent in many ways, but not perfect. SVG is its internal format, but it doesn't support all of SVG, and you need to choose "plain SVG" export to avoid mess.

On the other hand, it provides an XML tree-view of the source which allows you to add some things which Inkscape itself doesn't yet support.

It can crash, and has some pitfalls for beginners, so save your work often.
The object tag

You can use `img` but then JS can't get at it, so use `object`:

```html
<object data="demo.svg" type="image/svg+xml">
... text for non-SVG browsers ...
</object>
```

An SVG file can be compressed into `.svgz` format
Another possibility is to include the SVG text directly in the page:

```html
<p>
<svg xmlns="http://www.w3.org/2000/svg"...>
...
</svg>
</p>
```
It is important to understand the SVG source format, in order to make the best use of the tools, check standards compliance, write animation scripts, or write programs which generate or manipulate SVG.

So if you are using Inkscape:

Use Inkscape's XML view (<x> button), maybe edit the XML tree, export as plain SVG, maybe edit the result.
The `xmlns` attribute is necessary to specify what tags are allowed, and `width`, `height`, `viewBox` may need to be set manually in Inkscape.
Coordinates

You can set width and height in any absolute units, the default being browser pixels, or you can set them to 100% to mean that the drawing will scale to whatever size you give it on the web page.

If you want to allow stretching, you can add the `preserveAspectRatio="none"` attribute.

The viewBox coordinates specify what units to use within the SVG file, and these are arbitrary floating point units, not integer pixel units, so choose them for your convenience.
Compositing

The shapes inside the `<svg>` element are drawn in the order they appear, on top of one another.

Shapes have colour and opacity, and are combined by compositing using the formulas from the PNG chapter, or their pre-multiplied versions.

The default background is "transparent black", so the browser background shows through.

To prevent that, make the first shape a full-size opaque rectangle.
A *vector* is a line between two points, but it also has a direction, which you can think of as the direction which it is drawn in.

Press the button to see the direction.
A *path* is a sequence of vectors

Press the button to see the path
A path is mathematically precise, and infinitely thin. To make it visible, you have to *stroke* it, meaning draw along it with a given thickness.

Press the button
A second operation to make a path visible is to \textit{fill} it.

Press the button
Splines

Curves made out of straight vectors are inefficient and not scalable, so *splines* are used.

A spline was originally a bendy piece of wood held down with weights called ducks(!)
Bézier Splines

Bézier curves are simple and quick for online use.

Other kinds, B-splines and NURBS, may be better for offline modelling and 3-D work.

Think of a spline as the simplest curve joining two points, where you control the directions at each end (and how persistent those directions are).
Joining Splines

Controlling the directions of the ends means you can make a smooth path out of splines; it is traditional to illustrate with a duck.
Special Shapes

SVG has tags for special shapes, namely `<rect>`, `<circle>`, `<ellipse>`, `<line>`, `<polyline>`, `<polygon>`

These add nothing because they are all drawn with paths (possibly made curvy with splines)

Using special tags helps to preserve the intention of the author and/or improve editing and/or reduce download time

Beware that tools may not preserve them, but may lossily transform them into paths
Crossing

As far as possible, SVG puts no limitations on shapes, so that many different operations succeed.

For example, a path can cross itself arbitrarily, and one of the rules for filling is "evenodd"
The vectors of a single path need not be joined up

Press the button

(One vector is invisible and instant)
Boolean Operations

Disjoint paths mean that shapes can be combined by union, difference, intersection, exclusion
Because everything is made of paths, a compact representation is needed, so an SVG 100x100 square can be written:

```html
<path d="M 0 0 L 100 0 L 100 100 L 0 100 L 0 0"/>
```

Here M stands for "Move to" (without drawing), L for "draw a Line to" etc, and no commas etc are needed.

This helps keep download times small, even without compression.
Groups

Shapes can be grouped using the `<g>` tag

Leaving the group information in place rather than replacing the shapes by a single combined shape helps with editing, applying transformations both to a group and to a single shape within a group, etc.

Groups are an object-oriented and much more convenient version of layers, and layers are easily simulated using groups (as Inkscape does)

There is also a facility for defining shapes as re-usable symbols or templates
Shapes can be translated, rotated, scaled and skewed using the usual $2\times3$ matrices.

The original and the transform(s) are usually stored separately, to prevent cumulative rounding errors.
Text is represented using paths, but the original text is kept to allow search.

The text can be transformed, or written along a curved path, but it can still be searched, copied and pasted, and indexed by Google.

**Problem:** platform fonts are used by default but this is rarely correct, proper SVG fonts are difficult, so Web fonts are usually best.
Anti-aliasing

Shapes, including text, are turned into pixels at the last moment, so any aliasing problems are only ever at the true device resolution.

SVG viewers then optionally do a final anti-aliasing step on all sharp edges.
Images

Pixel based images in PNG or JPEG format can be included in or referenced from an SVG drawing.

The usual transformations can be applied, but the raw data is kept to avoid loss of precision.

To cope with the mismatch between the pixel matrix of the transformed image and the device, re-sampling is used to keep the quality high.

Images can be composited, and used in clipping and masking in the same way as all other shapes.
Filters

SVG supports the application of painting-style filters to all shapes.

Inkscape supports some of these filters, and tolerates more, so you can still use Inkscape in an edit-and-preview style.

The filters can be used for standard effects such as pseudo-3D, drop shadows and so on, which make a lot of difference to the perceived interest of drawings.
SVG supports style, e.g. via CSS, as with other XML languages, and most properties can be represented equally by direct attributes or using style attributes:

```xml
<rect fill="#ff0000".../>
<rect style="fill:#ff0000;..."/>
```

The first form is preferred for properties which are part of the drawing, or for animation etc.

The second form is supposed to be used only for applying a changeable site style to all drawings on a site, e.g. background colour
There are two ways to animate SVG: with JavaScript, or with SVG's own declarative animation (not fully programmable, obsolescent) - for the latter, start with:

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

This tells XML browsers that you may want to use XML compatible references of the type `xlink:href="x"` from one element to another with `id="x"`