SVG Syntax

basic syntax

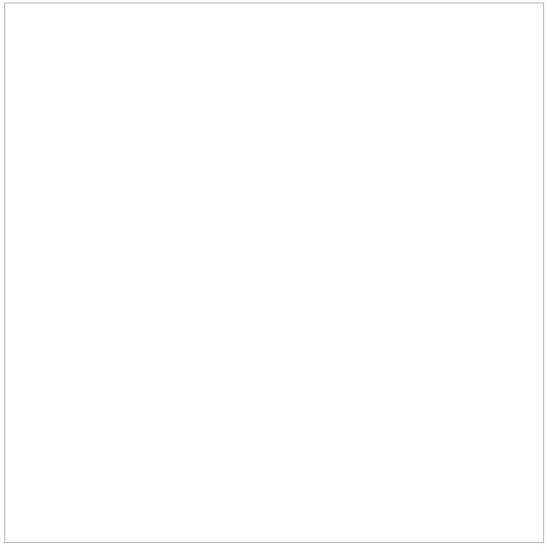
```
<svg>
</svg>
```

Let’s start with the <svg> element, which is the container for the svg code.

basic syntax

```
<svg width="500" height="500">
</svg>
```

Now we'll define the **viewport**, which is just the **width** and **height** of the svg on the page that contains it. If the viewport is not defined, the svg will scale to fit whatever contains it.



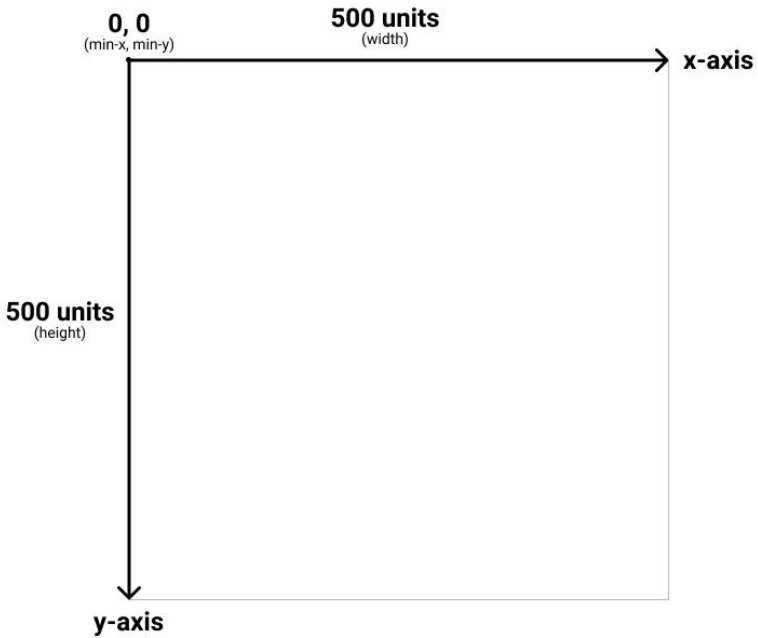
Note: I'm indicating the location of the svg using a grey outline.

basic syntax

```
<svg width="500" height="500" viewbox="0 0 500 500">
</svg>
```

Next we'll define the viewbox, which is the user-defined **coordinate system** of the viewport.
SVGs use a **coordinate system** for the placement of visual elements. The viewbox defines that coordinate system

The value for viewBox consists of four numbers:
min-x, min-y, width, and **height**

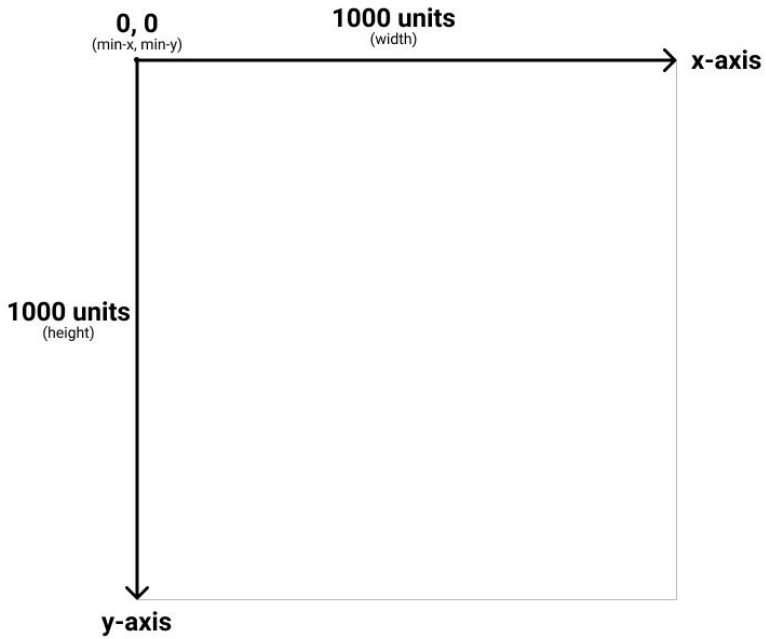


basic syntax

```
<svg width="500" height="500" viewbox="0 0 1000 1000">
</svg>
```

The important thing to remember about the width and height values of the viewBox is that they have **nothing to do** with the actual width and height of the svg.

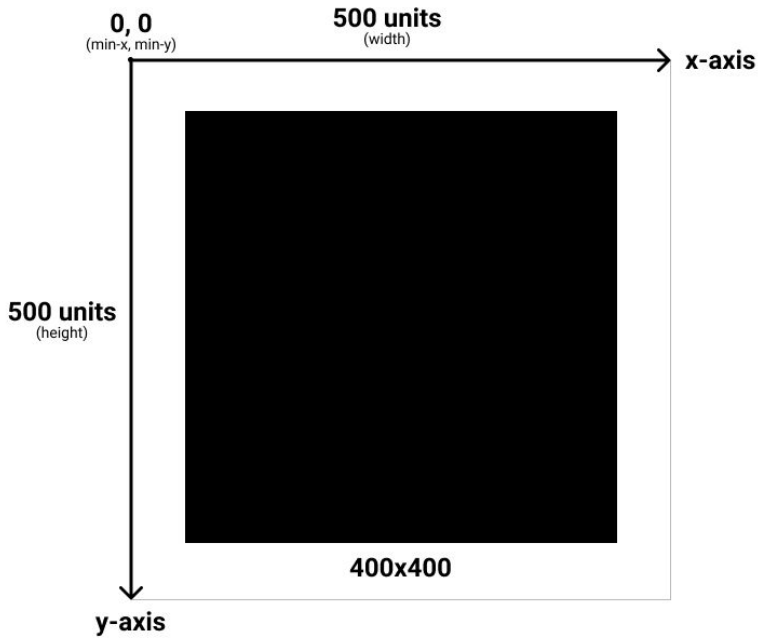
They only serve to define the **coordinate space** that will be used by the svg elements.



basic syntax

```
<svg
   width="500"
   height="500"
   viewbox="0 0 500 500">
   <rect x="50" y="50" width="400" height="400">
   </rect>
</svg>
```

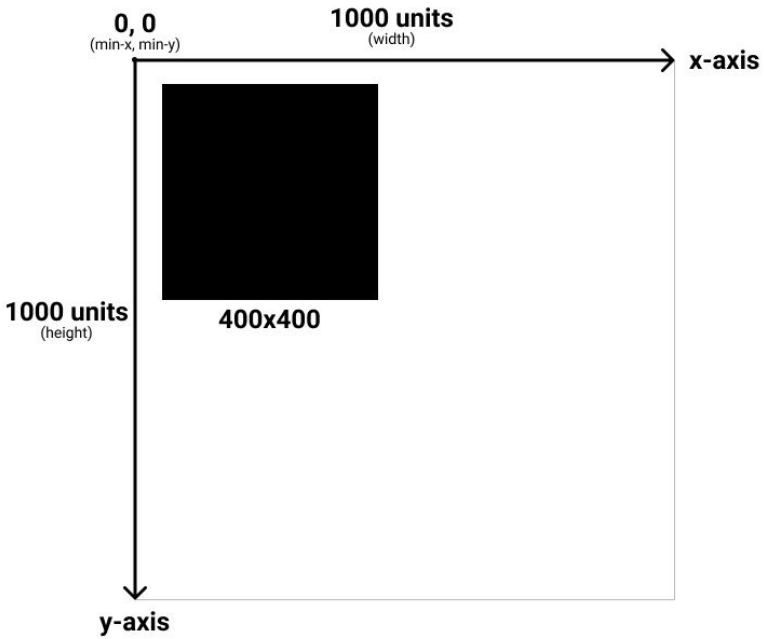
Example: here's a 400x400 rectangle in an svg with a viewBox size of 500x500



basic syntax

```
<svg
    width="500"
    height="500"
    viewbox="0 0 1000 1000">
    <rect x="50" y="50" width="400" height="400">
    </rect>
</svg>
```

Here’s the same 400x400 rectangle, except this time I’ve changed the viewBox size to 1000x1000



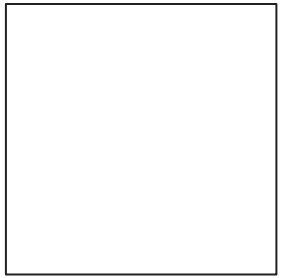
basic syntax

```
<svg
   width="500"
   height="500"
   viewbox="0 0 1000 500">
</svg>
```

But what if the aspect ratio of the viewbox is different than that of the viewport? For example, what if the viewport is a 500x500 square but the viewbox is a 1000x500 rectangle

That scenario is handled with the **preserveAspectRatio** svg attribute. It can be kind of confusing, so we'll leave that to a the **Creative Coding with SVGs - Aspect Ratio**. Right now we're going to try and keep the aspect ratio for the viewport and viewbox the same.

aspect ratio: 1x1



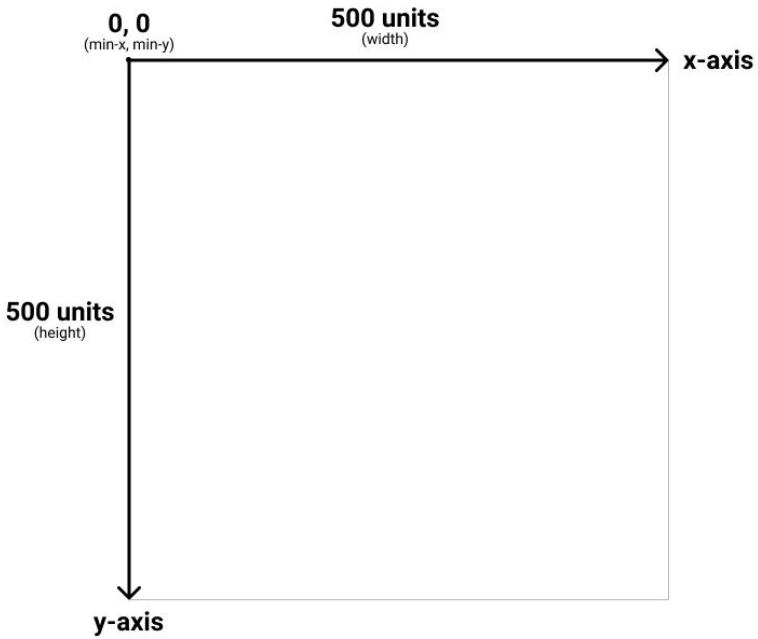
aspect ratio: 2x1

basic syntax

```
<svg
   width="500"
   height="500"
   viewbox="0 0 500 500">
</svg>
```

The **min-x** and **min-y** values are the minimum x and y values visible in the svg coordinate space.

mix-x and min-y represent the **upper left hand** corner of the svg. In this example, min-x min-y are 0 both 0.

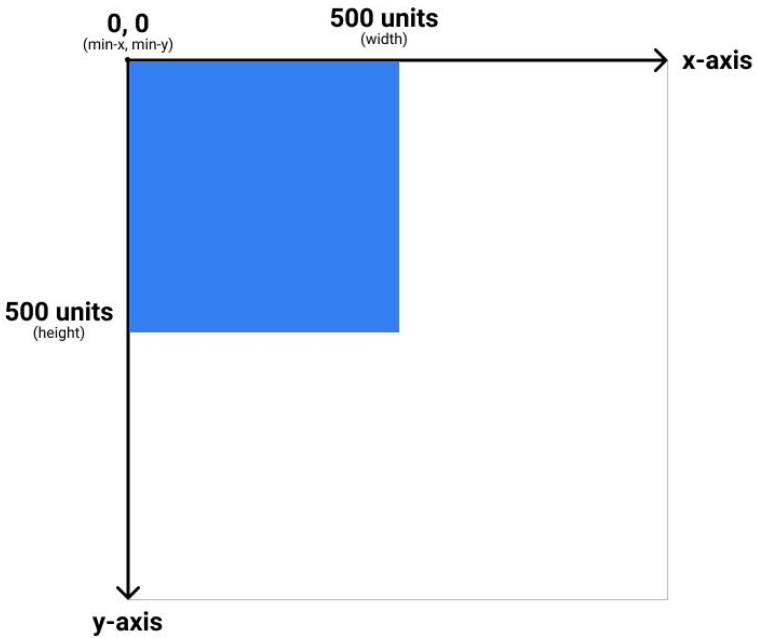


basic syntax

```
<svg
   width="500"
   height="500"
   viewbox="0 0 500 500">

   <rect x="0" y="0" width="250" height="250"></rect>
</svg>
```

Let’s draw a rectangle and put it at the 0,0 point.

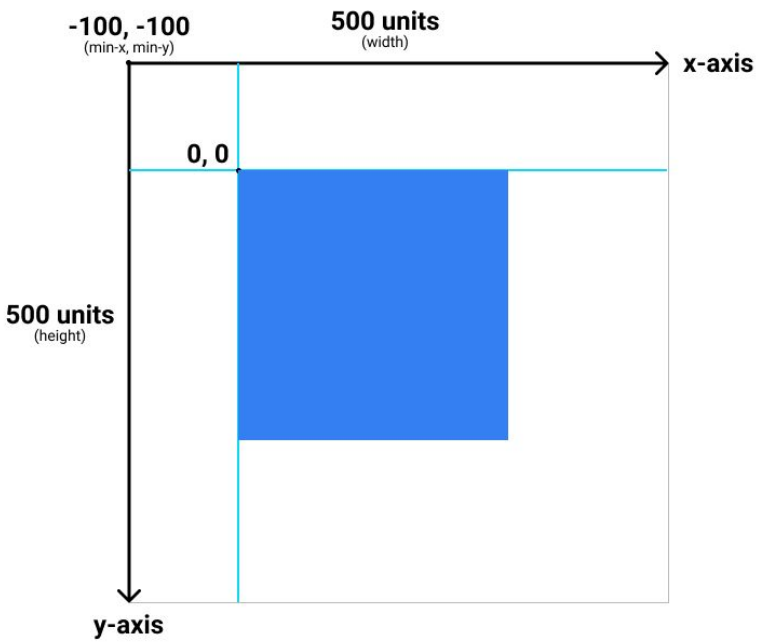


basic syntax

```
<svg
   width="500"
   height="500"
   viewbox="-100 -100 500 500">

   <rect x="0" y="0" width="250" height="250"></rect>
</svg>
```

Now lets update the min-x and min-y to:
-100, -100.
Note that we haven't changed the position of the rectangle at all. Only the **start position** of the viewBox coordinate space has changed.



basic syntax

```
<svg width="500" height="500" viewbox="-100 -100 500 500"
xmlns="http://www.w3.org/2000/svg">...
```

The final attribute we'll talk about is the namespace, which is only used with svg files, and not inline svgs. It essentially tells the browser that everything inside the file is an SVG, so that the browser knows what to do with it.

Again, it's only needed for svg **files**, and not inline svgs, so we won't be discussing this during the course.