# SVG Filters

SVG filters allow for non-destructive image manipulation. They are a superset of the filters available directly in CSS

The basic container for the filters is the `filter` element. To use them you add the `filter` attribute to the image element containing the image you want to filter.

A barebones svg for filtering an image looks like this:

```
<svg width="800" height="600" viewBox="0 0 800 600">
    <filter id="myFilter">
        <filter id="myFilter"><!-- filter effects go in here -->
    </filter>
    <image xlink:href="..."
           width="100%" height="100%" x="0" y="0"
           filter="url(#myFilter)"></image>
</svg>
```

In the following sections we'll talk about filter components and other things we need to understand before making them work.

# Filter Primitives

In SVG filters, each filter element contains one or more filter primitives as its children. Each primitive performs a graphical operation on one or more inputs, producing a result that can be used as an input for subsequent elements on the chain. They all share the same prefix: **fe**, which is short for "filter effect" and use a name that references its purpose (for example `feGausianBlur` for the primitive tht creates a gausian blur).

Filter primitives take a source graphic as input and outputing another one as the result. You chain the output of one effect as the input of another one; this flexibility gives you an almost endless combination of filter primitives producing an endless number of effects

Each primitive can take one or two inputs and output only one result. The inputs of a filter primitive are defined in attributes called `in` and `in2`. The result of

an operation is defined in the `result` attribute. If you don't specify the result of a primitive, its result will automatically be used as input to the primitive that follows.

A filter primitive also accepts other types of inputs, the most important of which are:

- **SourceGraphic**: the actual element that we're applying the filter to, for example, an image or a piece of text
- **SourceAlpha**: the alpha channel for the element

```
<svg width="600" height="400" viewBox="0 0 600 400">
  <filter id="myFilter">

    <feFlood flood-color="navy" flood-opacity=".15" result="flood"></feFl

    <feMerge>
      <feMergeNode in="flood" />
      <feMergeNode in="SourceGraphic" />
    </feMerge>

  </filter>
  <image xlink:href="https://s3-us-west-2.amazonaws.com/s.cdpn.io/32795/oc
         width="100%" height="100%" x="0" y="0"
         filter="url(#myFilter)"></image>
</svg>
```

There are 17 filter primitives available in the Filter Effects Module Level 1.

# The Filter Region

In SVG, elements have "regions" whose boundaries are defined by the borders of the element's Bounding Box. The Bounding Box (also abbreviated "bbox") is the smallest fitting rectangle around an element.

If you apply a filter effect to an image or a piece of text, the effect will be restricted to its bounding box, and any filter result that lies beyond the boundaries of it will be clipped off. This may not be very useful because many filters will impact pixels outside the boundaries of the bounding box and, by default, those pixels will end up being cut off.

So how do we prevent that from happening? By extending the filter region. We can extend the region the filter is applied to by modifying the x, y, width and height attributes on the `filter` element.

By default, filters have regions extending 10% the width and height of the bounding box in all four directions. When working with filters we might want to create it using the following syntax:

```
<filter x="-10%" y="-10%" width="120%" height="120%"
        filterUnits="objectBoundingBox">
    <!-- filter operations here -->
</filter>
```

If you omit these attributes on the `filter` element, these values will be used by default. You can also override them to extend or shrink the region as you need.

Keep in mind that the units used in the x, y, width and height attributes are dependent on which filterUnits value is in use.

- **objectBoundingBox** (default units): When the filterUnits is `objectBoundingBox`, the values of the x, y, width and height attributes are percentages or fractions of the size of the element's bounding box
- **userSpaceOnUse**: when filterUnits is set to userSpaceOnUse the coordinates of the x, y, width and height attributes are set relative to the current coordinate system ussed in the SVG.

See Sara Soueidan's Understanding SVG Coordinate Systems and Transformations (Part 1) for a good introduction and objectBoundingBox versus userSpaceOnUse for an in-depth discussion on `ObjectBoundingBox` and `userSpaceOnUse` that is applicable to gradients, patterns, filters, masks, and clipping paths.