5 – Color and images manipulation in CSS

5.1. Colors

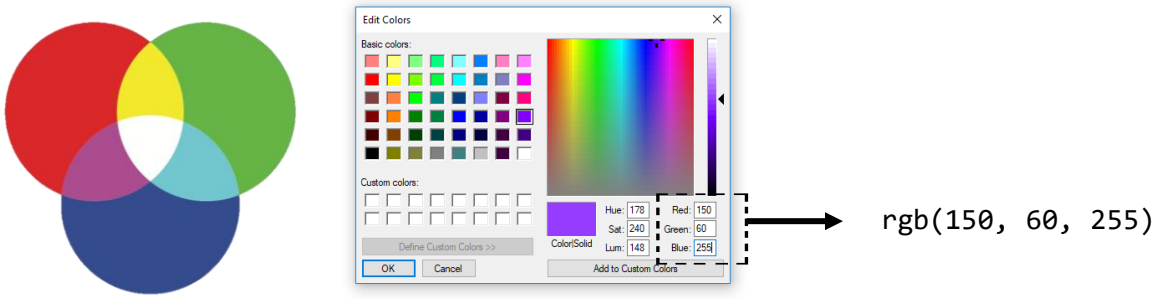
The color property allows us to specify the color of text inside an element. We can specify any color in CSS in one of three ways:

5.1.1. Color Names

There are 147 predefined color names that are recognized by browsers. For example: `darkcyan`

5.1.2. RGB Values

These express colors in terms of how much red, green and blue are used to make it up. Every color on a computer screen is created by mixing amounts of red, green, and blue. To find the color you want, we can use a color picker.



5.1.3. HEX Codes

These are six-digit codes that represent the amount of red, green and blue in a color, preceded by a pound or hash # sign. For example: `#ee3e80`

Exercise) Go to the following website: ***https://www.w3schools.com/cssref/css_colors.asp*** and select two colors using color name, hex code, and color value. Using the colors code, create paragraphs with different background colors using the three CSS color property.

HTML

```
<h3> Understanding Colors </h3>
    <h4>Color Using HEX code</h4>
      <p class="color1"> Color: HEX: #96ceb4</p>
      <p class="color2"> Color: HEX: #ffcc5c</p>
    <h4>Color Using rgb values</h4>
      <p class="color3"> Color: rgb(200,180,160)</p>
      <p class="color4"> Color: rgb(160,100,80)</p>
    <h4>Color Using color name</h4>
      <p class="color5"> Color: violet</p>
      <p class="color6"> Color: Dark Magenta</p>
```

CSS

```
h3{   background-color: lightblue;  padding: 10px; }
.color1{background-color: #96ceb4; padding: 12px; }
.color2{background-color: #ffcc5c;  padding: 12px; }
.color3{background-color: rgb(200,180,160);  padding: 12px; }
.color4{background-color: rgb(160,100,80);  padding: 12px;}
.color5{background-color:violet;  padding: 12px; }
.color6{background-color: darkmagenta;  padding: 12px; }
```

5.1.4. rgba( ) color

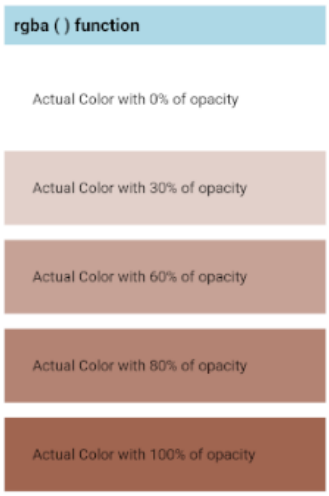
The rgba( ) function define colors using the red-green-blue-alpha (RGBA) model where alpha specifies the opacity of the color. Alpha color goes from 0 to 1, 0 means fully transparent (0% of opacity) and 1 means fully opaque (100% of opacity). The syntax of the code is:

rgba(*red, green, blue, alpha*)

For example, for color **rgb(150, 60, 255)**with 35% of opacity, we can write the code as:

rgba(150, 60, 255,0.35)

Exericse) Obtain a color using its rgb value and create paragraphs of the chosen color with 0%, 30%, 60%, 80%, and 100% of opacity. Add 30% of padding to each paragraph.



HTML

```
<h3>rgba ( ) function </h3>
    <p class="rgba1">Actual Color with 0% of opacity</p>
    <p class="rgba2">Actual Color with 30% of opacity</p>
    <p class="rgba3">Actual Color with 60% of opacity</p>
    <p class="rgba4">Actual Color with 80% of opacity</p>
    <p class="rgba5">Actual Color with 100% of opacity</p>
```

CSS

```
.rgba1{background-color:rgba(160,100,80,0);  padding:30px;}
.rgba2{background-color:rgba(160,100,80,0.3);  padding:30px;}
.rgba3{background-color:rgba(160,100,80,0.6);  padding:30px;}
.rgba4{background-color:rgba(160,100,80,0.8);  padding:30px;}
.rgba5{background-color:rgba(160,100,80,1);  padding: 30px;}
```

5.2. CSS linear-grading

CSS gradient display smooth transition between two or more specified colors. Those colors are the colors that we want to render smooth transition among the grading space. For example, we can create a **`linear-gradient`** of the red and blue color from top to bottom as the following:

HTML

```
<div class="gradient1"></div>
```

CSS

```
.grading1 {
  height: 200px;
  background-image: linear-gradient(red, blue);
}
```

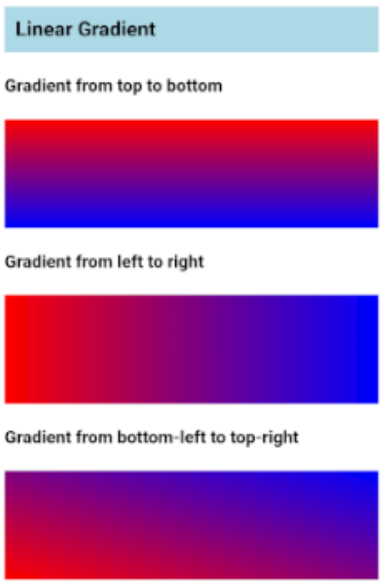
`linear-gradient` can set the color from left to right. For this, we can add **`to right`** value to the `background-image` property as the following:

```
background-image: linear-gradient(to right, red, blue);
```

We can also set the linear-gradient from bottom-left to top-right:

```
background-image: linear-gradient(to top right, red, blue);
```

The output will look as the following:



5.3. Position Property

One importation property of CSS layout is the `position` property. The `position` property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

position: static;

HTML elements are positioned static by default.
Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page.

position: relative;

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

position: absolute;

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: A "positioned" element is one whose position is anything except `static`.

Exercise) Create container with a back image with `position:relative;` and a message with `position:absolute;`

We can create a container for the back image using ** ** element

HTML

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="index.css" rel="stylesheet" type="text/css"/>
    <title>Activity 5 by Huixin Wu</title>
  </head>
  <body>
    <h3> Position relative and absolute </h3>

    <div class="image">
      <img src="image.jpg">
    </div>

  </body>
</html>
```

CSS

```
img{ width: 100%; height: 100%;}
.image{
  width: 80%;
  height: 450px;
  margin: auto;
  position: relative;
}
```

Position relative and absolute

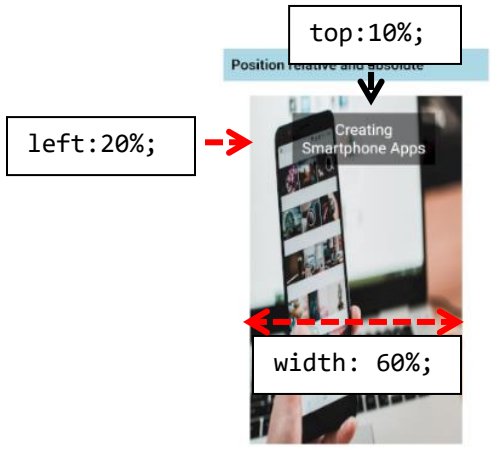
After it, we can add the text message using another `<div>` or `<p>` and use `position:absolute;`

HTML

```
<div class="message"> Creating Smartphone Apps</div>
```

CSS

```
.message{
  position: absolute;
  top: 10%;
  left: 20%;
  width: 60%;
  background-color: rgba(0, 0, 0, 0.3);
  color: white;
  text-align: center;
  padding: 10px;
  font-size: 20px;
}
```



position: sticky;

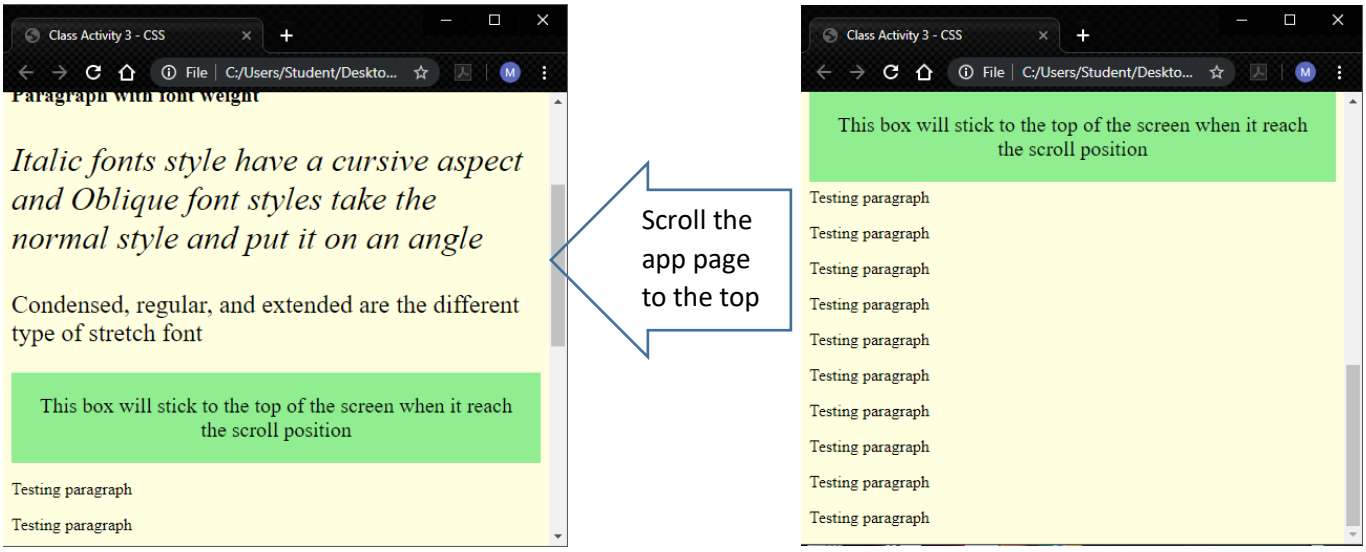
An element with `position: sticky;` is positioned based on the user's scroll position.

A sticky element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like `position:fixed`).

Exercise) Create a message box using `position:sticky;`

We can create a container for the sticky message using **`<div>`** element with `position:sticky;` and the position to stick to, in this case, we can set the message to stick on the top of the screen using `top:0;`

We can also add 16 testing paragraphs to see the sticky effect.



HTML

```
<h3>Position sticky </h3>
<p>Scroll down this page to see how  sticky positioning works.</p>

<div class="stickyBox">This box will stick to the top of the screen when it reach
the scroll position</div>

<p>Testing paragraph </p> <p>Testing paragraph </p>
<p>Testing paragraph </p> <p>Testing paragraph </p>
<p>Testing paragraph </p> <p>Testing paragraph </p>
<p>Testing paragraph </p> <p>Testing paragraph </p>
<p>Testing paragraph </p> <p>Testing paragraph </p>
<p>Testing paragraph </p> <p>Testing paragraph </p>
<p>Testing paragraph </p> <p>Testing paragraph </p>
<p>Testing paragraph </p> <p>Testing paragraph </p>
```

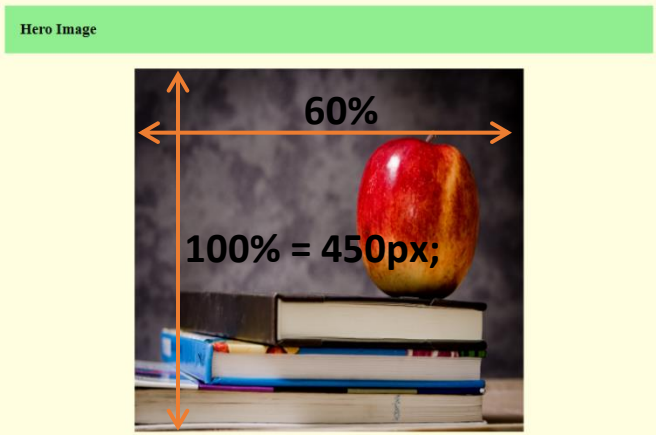
CSS

```
.stickyBox{
  font-size: 20px;
  text-align: center;
  padding: 20px;
  background-color: lightgreen;
  position: sticky;
  top:0;
}
```

5.4. Hero image

A hero image is a large image with text places on the top of it. For this, we can use the position absolute and relative.

Exercise) Create an ad of an image with a text places on top of it as the following:



First, we create a container with a background image:

HTML

```
<h3>Hero Image </h3>
  <div class="heroImage">

  </div>
```

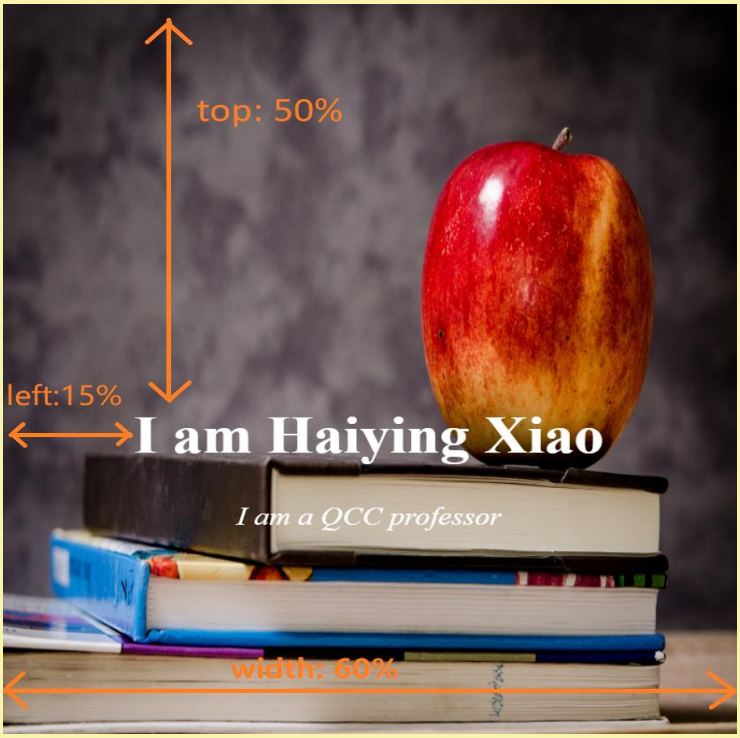
We can insert a background image from a CSS using **background-image** property. Also, we can align the background image using **background-position: center;** and adjust the width and height using **background-size: 60% 100%;** which 60% is the width and 100% is the height.

CSS

```
.heroImage{
  background-image: url("apple.jpg");
  background-repeat: no-repeat;
  background-position: center;
  background-size: 60% 100%;
  height: 450px;
  position: relative;
}
```

Once we have the container and the background image set, we can create the text that will sit on top of the background image. We can name the text container **heroText**

Hero image



HTML

```
<h3>Hero Image </h3>
  <div class="heroImage">
    <div class="heroText">
      <h1>I am Haiying Xiao </h1>
      <i>I am a QCC professor</i>
    </div>
  </div>
```

CSS

```
.heroText{
  position: absolute;
  top: 50%;
  left: 15%;
  width: 70%;
  color: white;
  text-align: center;
}
```

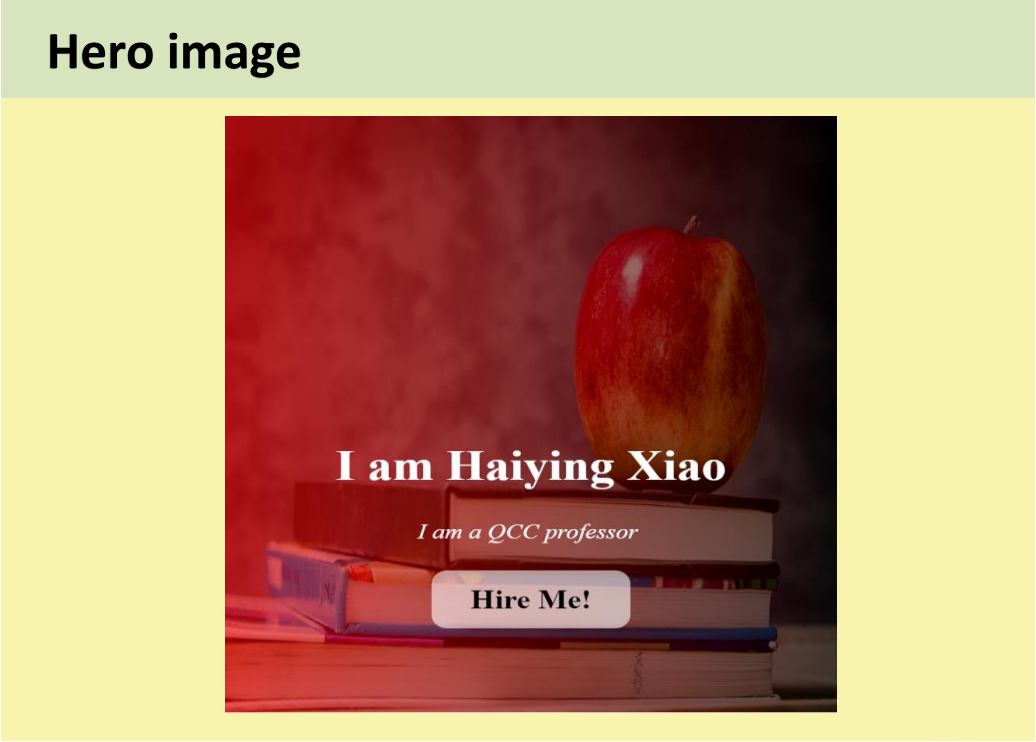
A text on top of an image makes the text information very hard to write, for this, we can add linear-gradient to the background image. We can set a red to black gradient with 60% of opacity from left to right:

Hero image

CSS

```
.heroImage{
  background-image: linear-gradient(to right, rgba(255,0,0,0.6),rgba(0,0,0,0.6)),
url("apple.jpg");
  background-position: center;
  background-repeat: no-repeat;
  background-size: 100% 100%;
  height: 450px;
  position: relative;
```

After it, we can add a “Hire Me!” link to the text.



HTML

```
<h3>Hero Image </h3>
  <div class="heroImage">
    <div class="heroText">
      <h1> I am Haiying Xiao < /h1>
      <i>I am a QCC professor</i>
        <br>
        <a href="#" class="submit">Hire Me!</a>
    </div>
  </div>
```

CSS

```
.submit{
  padding: 10px 25px;
  color: black;
  background-color: rgba(255,255,255,0.6);
  font-weight: 600;
  font-size: 20px;
  margin-top: 20px;
  text-align: center;
  border-radius: 10px;
  display: inline-block;}
a.submit{text-decoration: none;}
```

5.5. Filter images

Filter property adds visual effects to an element. Some of the filter attributes are: blur, contrast, sepia, grayscale, and invert.

blur(px)

blur applies a blur effect to an element. The value, in pixels *px*, between the parenthesis defines the blurriness of the element, a larger value will create more blur.

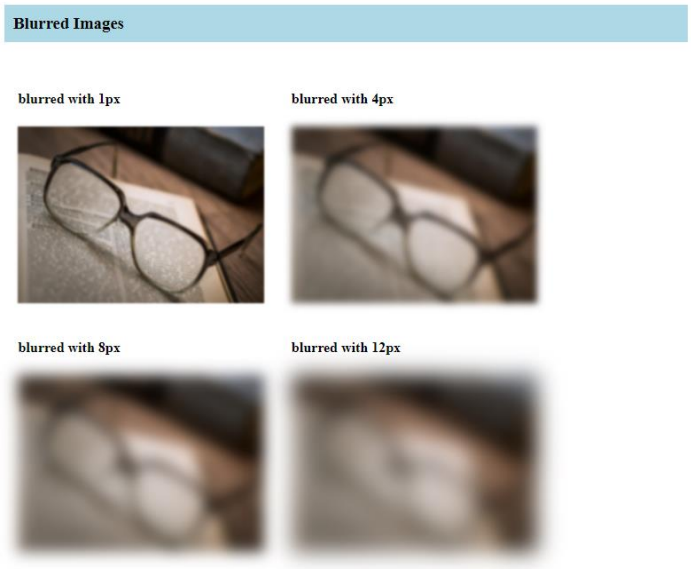
Activity) Create four different blur effects to an image with 1px, 4px, 8px, and 12px.

HTML

```
<h3>Blurred Images </h3>
  <div class="filterContainer">
       <div class="filter1">
              <h4>blurred with 1px</h4>
              <img src="book.jpg" style="filter:blur(1px);"/></div>
       <div class="filter2">
              <h4>blurred with 4px</h4>
              <img src="book.jpg" style="filter:blur(4px);"/></div>
       <div class="filter3">
              <h4>blurred with 8px</h4>
              <img src="book.jpg" style="filter:blur(8px);"/></div>
       <div class="filter4">
               <h4>blurred with 12px</h4>
              <img src="book.jpg" style="filter:blur(12px);"/></div>
  </div>
```

CSS

```
/* BLURRED IMAGES */
.filterContainer{height:700px;}
.filter1, .filter2, .filter3, .filter4{
       width: 40%;
       height: 40%;
       padding:2%;
       float:left;
}
```



contrast(%)

contrast adjusts the contrast of an element. If the element is used as the container for an image, then contrast will adjust the contrast of the image.

contrast(0%) will make the image completely black and **contrast(100%),** which is by default, will show the original image, and values over 100% will provide results with more contrast:

HTML

```
<h3>Images with Contrast</h3>
  <div class="filterContainer">
       <div class="filter1">
             <h4>Contrast 0%</h4>
             <img src="book.jpg" style="filter:contrast(0%);"/></div>
       <div class="filter2">
             <h4>Contrast 20%</h4>
             <img src="book.jpg" style="filter:contrast(20%);"/></div>
       <div class="filter3">
             <h4>Contrast 50%</h4>
             <img src="book.jpg" style="filter:contrast(50%);"/></div>
       <div class="filter4">
              <h4>Contrast 90%</h4>
             <img src="book.jpg" style="filter:contrast(90%);"/></div>
  </div>
```

Images with Contrast

Contrast 0%

Contrast 20%

Contrast 50%

Contrast 90%

sepia(%)

sepia converts the element to sepia color. **sepia(0%)** is default means that there is no sepia applies to the element, and **sepia(100%)** will make the image completely sepia. For this attributes, negative values are not allowed.

HTML

```
<h3>Sepia Images</h3>
  <div class="filterContainer">
      <div class="filter1">
            <h4>Sepia 0%</h4>
            <img src="book.jpg" style="filter:sepia(0%);"/></div>

      <div class="filter2">
            <h4>Sepia 20%</h4>
            <img src="book.jpg" style="filter:sepia(20%);"/></div>

      <div class="filter3">
            <h4>Sepia 50%</h4>
            <img src="book.jpg" style="filter:sepia(50%);"/></div>

      <div class="filter4">
             <h4>Sepia 90%</h4>
            <img src="book.jpg" style="filter:sepia(90%);"/></div>
  </div>
```

Sepia Images

Sepia 0%

Sepia 20%

Sepia 50%

Sepia 90%

grayscale(%)

grayscale converts the element to loose its black and white color. **grayscale(0%)** is default and it means that there isn't any changes in its grayscale. **grayscale(100%)** will make the image completely gray, black and white. This property does not allowed negative values.

HTML

```
<h3>Grayscale Images</h3>
  <div class="filterContainer">
       <div class="filter1">
              <h4>Grayscale 0%</h4>
              <img src="book.jpg" style="filter:grayscale(0%);"/></div>

       <div class="filter2">
              <h4>Grayscale 20%</h4>
              <img src="book.jpg" style="filter:grayscale(20%);"/></div>

       <div class="filter3">
              <h4>Grayscale 50%</h4>
              <img src="book.jpg" style="filter:grayscale(50%);"/></div>

       <div class="filter4">
               <h4>Grayscale 90%</h4>
              <img src="book.jpg" style="filter:grayscale(90%);"/></div>
  </div>
```

Grayscale Images

Grayscale 0%

Grayscale 20%

Grayscale 50%

Grayscale 90%

invert(%)

invert( ) inverts the sample color of the element. **invert(0%)** is default and means that there isn't any color inversion in the element. **invert(100%)** will make the image will invert completely its sample color.

HTML

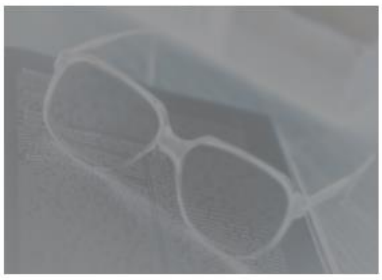
```
<h3>Invert Images</h3>
  <div class="filterContainer">
       <div class="filter1">
              <h4>Invert 0%</h4>
              <img src="book.jpg" style="filter:invert(0%);"/></div>

       <div class="filter2">
              <h4>Invert 20%</h4>
              <img src="book.jpg" style="filter:invert(20%);"/></div>

       <div class="filter3">
              <h4>Invert 60%</h4>
              <img src="book.jpg" style="filter:invert(60%);"/></div>

       <div class="filter4">
               <h4>Invert 90%</h4>
              <img src="book.jpg" style="filter:invert(90%);"/></div>
  </div>
```

Invert Images

Invert 0%

Invert 20%

Invert 60%



Invert 90%