3.5. Color and images manipulation in CSS

3.5.1. Colors

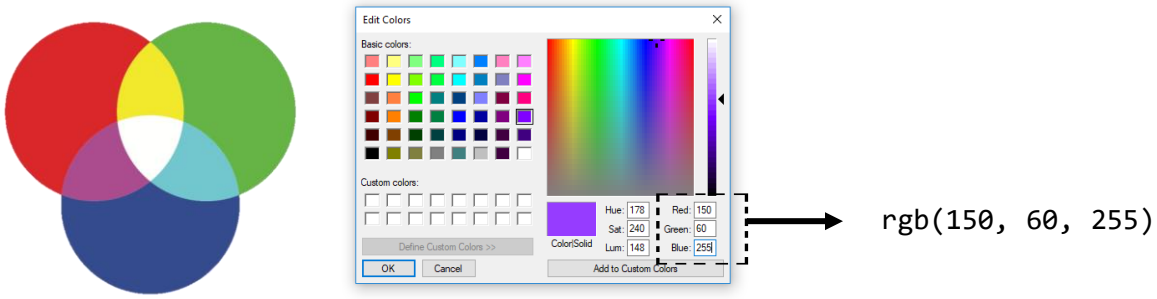
The color property allows us to specify the color of text inside an element. We can specify any color in CSS in one of three ways:

Color Names

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

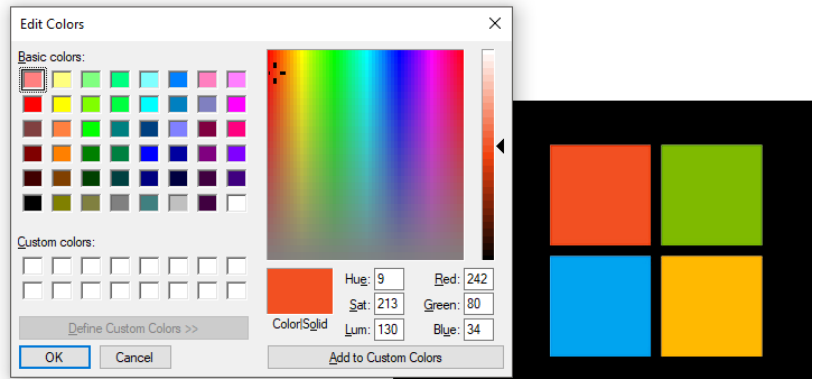
RGB Values

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.

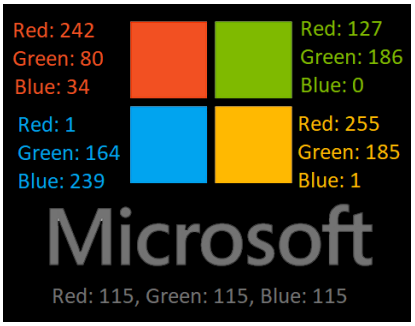


Example) find each RGB color the Microsoft logo:

To find a specific color, we can use a color picker tool from any graphic app. One simple graphic app that comes with Windows is *Paint.*



Using Paint, we can the four colors of the Microsoft logo as:



Using the RGB in CSS will look as following:

HTML code

```
  <section class="rgb">
      <div class="red"></div>
      <div class="green"></div>
      <div class="blue"></div>
      <div class="orange"></div>
      <div class="logo">Microsoft</div>
    </section>
```

CSS code

```
.rgb{margin: 2em; display: inline-block;}
.red,.green,.blue,.orange{
      width:100px;
      height:100px;
      margin: 20px;
      float: left;
}
.red{background-color:rgb(242,80,34)}
.green{background-color:rgb(127,186,0)}
.blue{background-color:rgb(1,164,239)}
.orange{background-color:rgb(255,185,1)}
.logo{
      color:rgb(115,115,115);
      font-size:5em;
      font-family:"Arial Black"
}
```

The display in the browser will be as:

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)
```

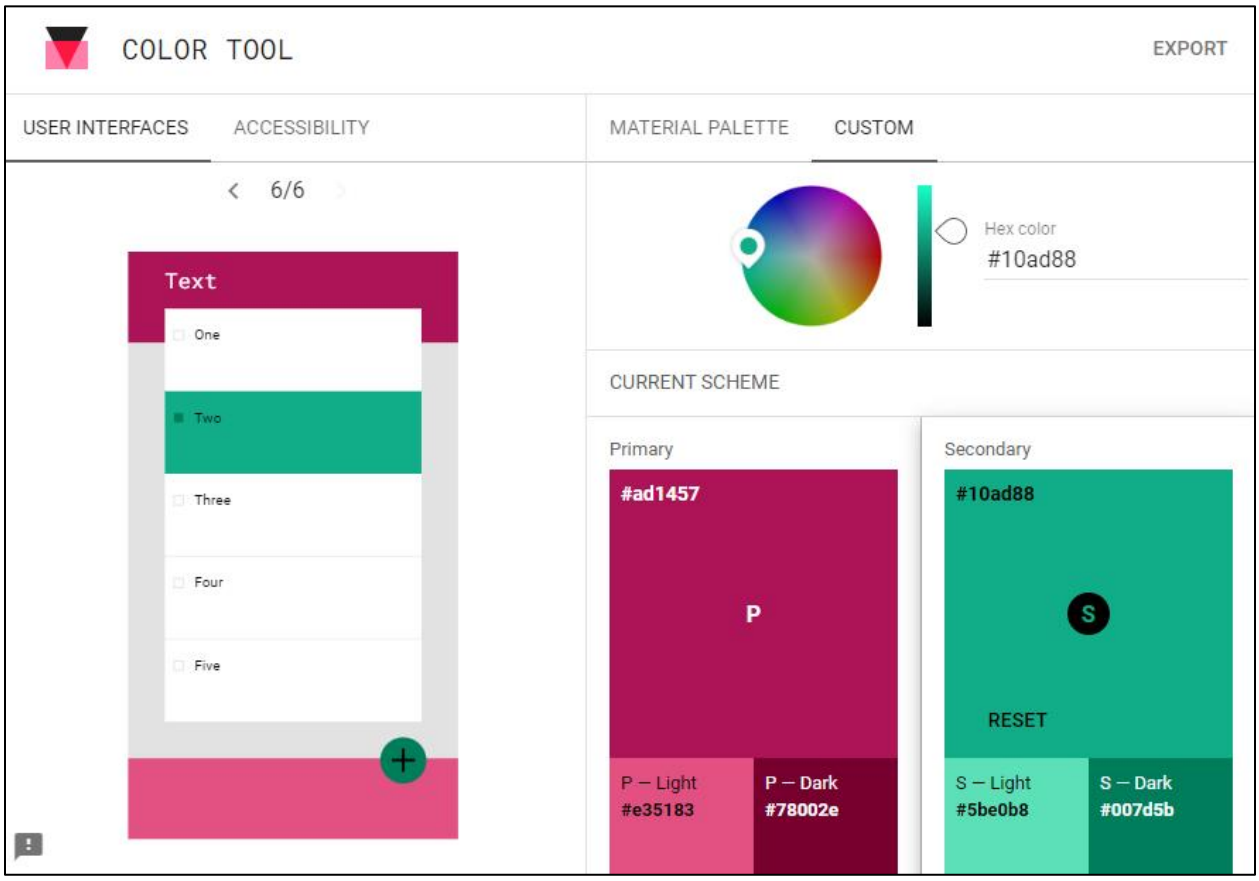
For example, for color **`rgb(150, 60, 255)`** with 35% of opacity, we can write the code as:

```
rgba(150, 60, 255,0.35)
```

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`

We can find different HEX color picker tools online. One of the website to explore the HEX color picker as a primary or and secondary color is *material.io*. material.io is a website dedicates to the UI design of smartphone. Some visitors of material.io likes to use its *Color Tools* to explore the combination of a primary and secondary color. https://m2.material.io/resources/color/#!/?view.left=0&view.right=0



Position Property

The position CSS property sets how an element is positioned in a document. For position: relative, absolute, or sticky, the top, right, botton, and left properties determines the final location of positioned elements.

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

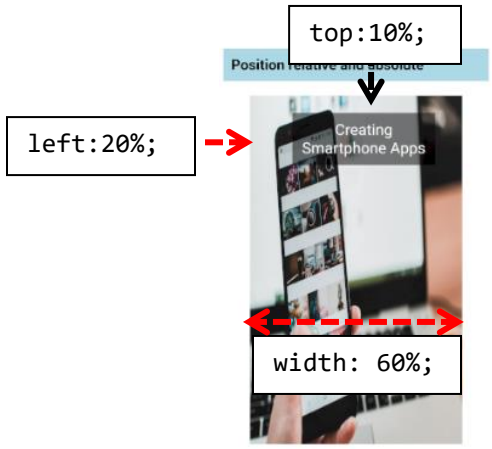
After it, we can add the text message using another or <p> and use position:absolute;

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;

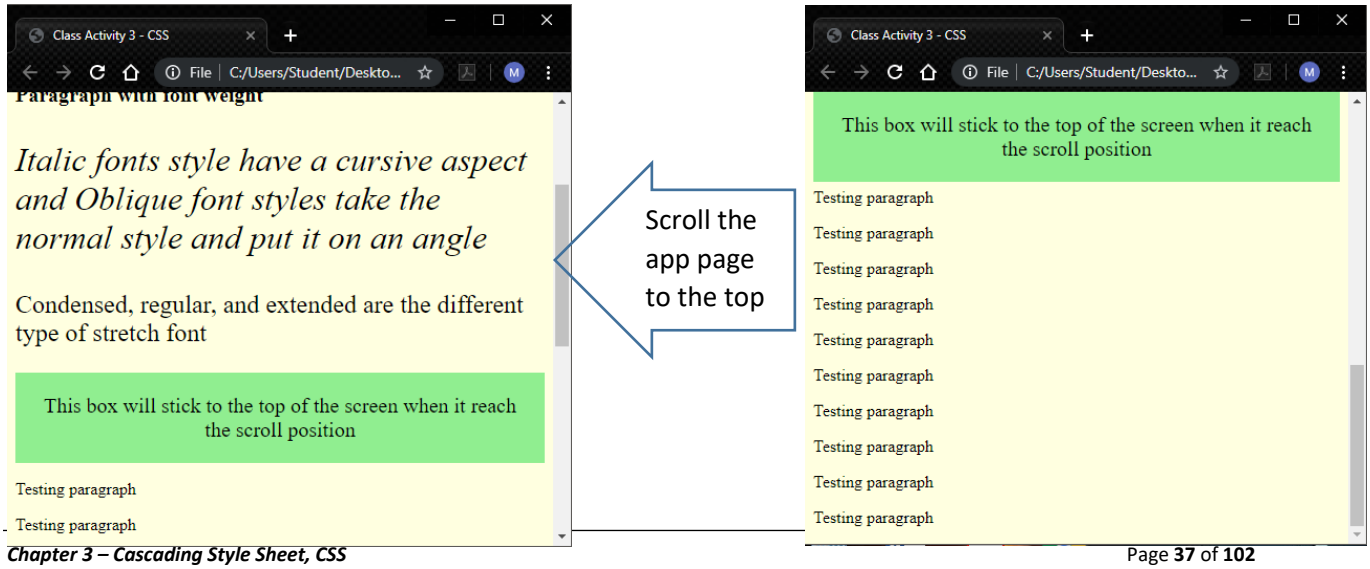
An element with `position: sticky;` is positioned based on the user's scroll position.

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 ** ** 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;
}
```

Fixed navigation bar

Fixed navigation bar happens when the navigation bar remain at the top or bottom of the app, even when the user moves the page. For this, you can use the CSS property `position: sticky;` to the element that contains the hyperlinks:

```
position: fixed;
top: 0;
```

`position:fixed;` and **`top: 0;`** will make the element to have a position fixed to the top of the app.

Exercise) Create a navigation bar that will stick on the top of the app.

We can create the hyperlink using **`<div>`**

```
<nav class="nav_top">
  <a href="#home">Home</a>
  <a href="#nav">Tabs</a>
  <a href="#flex">Flex-Wrap</a>
  <a href="#social">Social Media</a>
</nav>
```

Once we have the HTML code set, in CSS, we can set the hyperlink container with background-color, the width, the height, the font-size, and the position:

```
.nav_top{
    background-color: orange;
    font-size: 1.2em;
    height: 3em;
    position: sticky;
    top: 0;
}
```

After it, we can set hyperlink with left border, height, font color, center the text, and add the padding. We also set the hyperlink to float from the left to right without underline:

```
.nav_top a {
  float: left;
  color: white;
  text-align: center;
  text-decoration: none;
  padding: 1em 2em;
}
```

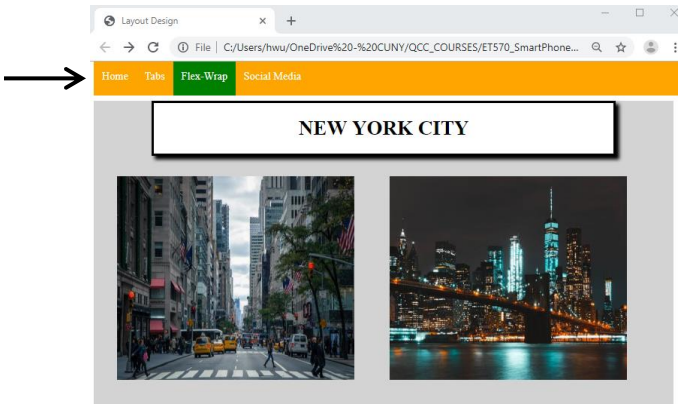
After it, we can change the background color of the links when the user clicks on them:

```
.nav_top a:hover{
  background-color: olive;
  font-weight: bold;
}
```

The HTML and CSS code looks as the following:

html file

```
<nav class="nav_top">
  <a href="#home">Home</a>
  <a href="#nav">Tabs</a>
  <a href="#flex">Flex-Wrap</a>
  <a href="#social">Social Media</a>
</nav>
```



css file

```
.nav_top{
   background-color: orange;
   font-size: 1.2em;
   height: 3em;
   position: sticky;
   top: 0;
}
.nav_top a {
   float: left;
   color: white;
   text-align: center;
   text-decoration: none;
   padding: 1em 2em;
}
.nav_top a:hover{
   background-color: olive;
   font-weight: bold;
}
```

If you need the navigation bar stick in the bottom, you can change **`top: 0;`** to **`bottom: 0;`**

Display properties

The `display` property specifies the display behavior (the type of rendering box) of an element.

In HTML, the default display property value is taken from the HTML specifications or from the browser/user default style sheet. The default value in XML is inline, including SVG elements.

Note: The values "flex" and "inline-flex" requires the -webkit- prefix to work in Safari.

Note: "display: contents" does not work in Edge prior version 79.

Some of the display properties are:

inline	Displays an element as an inline element (like). Any height and width properties will have no effect
block	Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width
contents	Makes the container disappear, making the child elements children of the element the next level up in the DOM
flex	Displays an element as a block-level flex container
grid	Displays an element as a block-level grid container
inline-block	Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values
inline-flex	Displays an element as an inline-level flex container
inline-grid	Displays an element as an inline-level grid container
inline-table	The element is displayed as an inline-level table
list-item	Let the element behave like a element

Background properties

Background images and linear gradient

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);
}
```

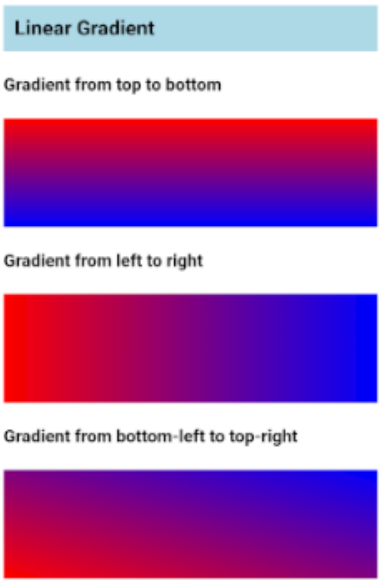
`linear-gradient` can set the color from left to right. For this, we can add **`to right`** value to the `background-image` property as the following:

```
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);
```

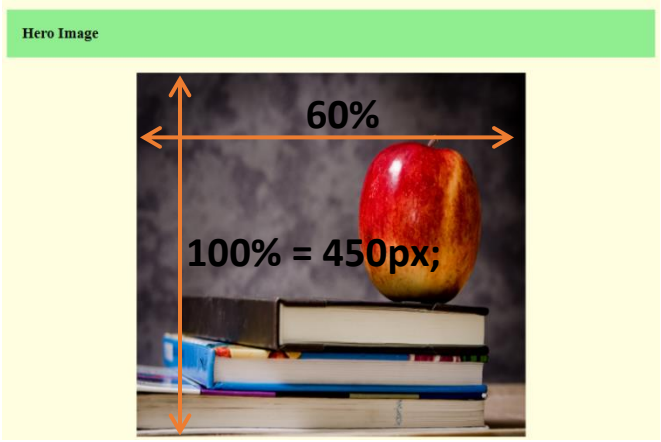
The output will look as the following:



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

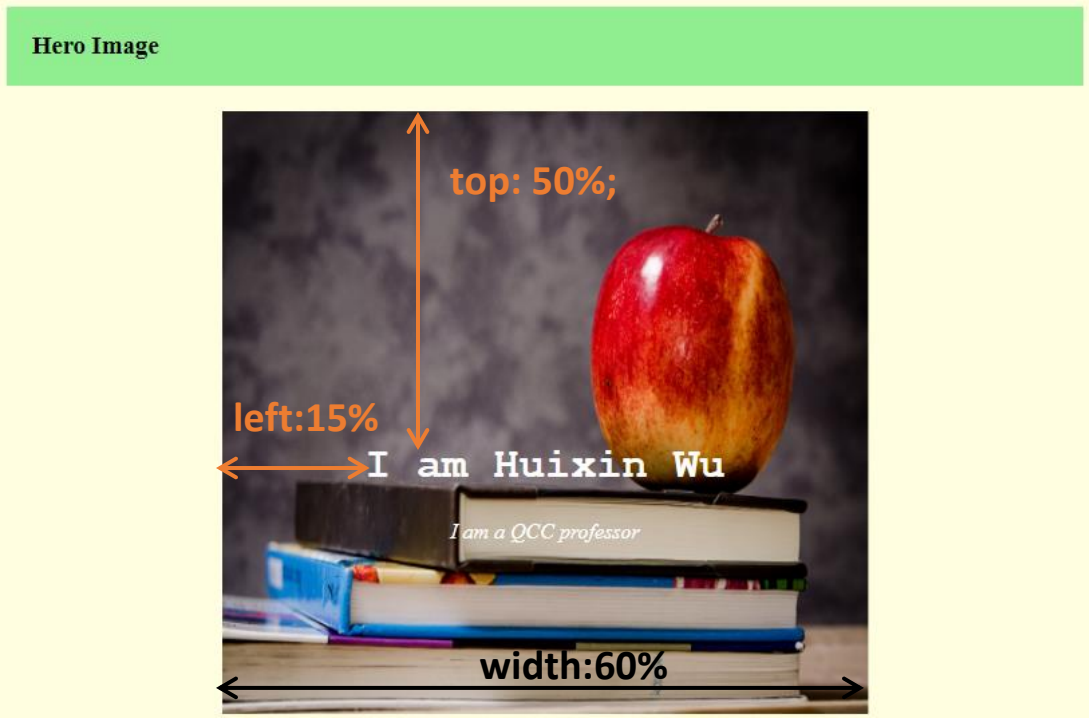
```
  <h3>Hero Image </h3>
    <div class="heroImage">

    </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**



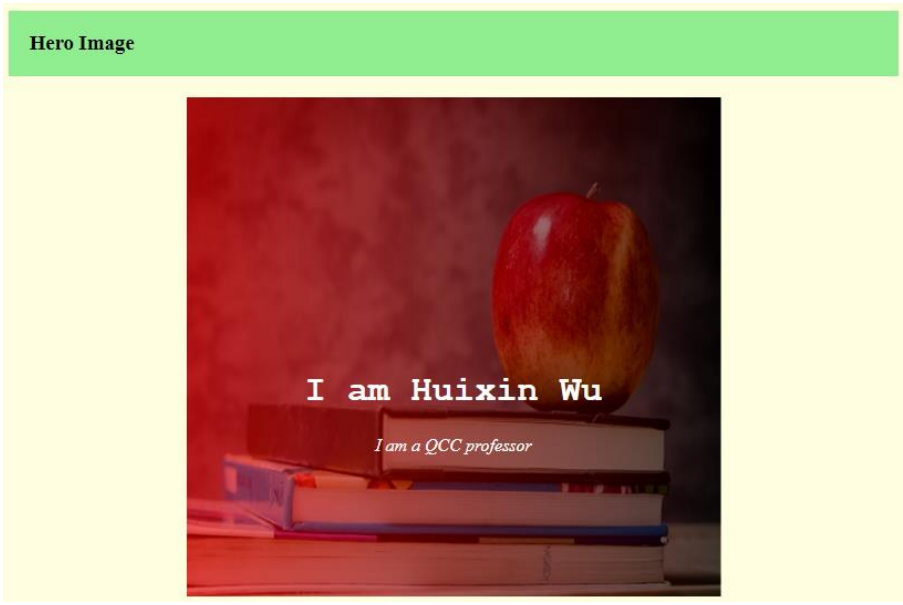
HTML

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

CSS

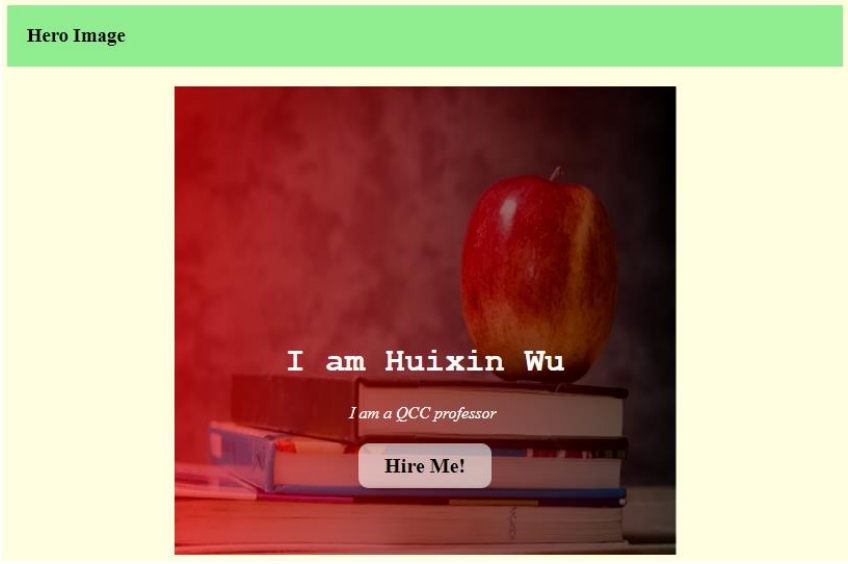
```
.heroText{
  position: absolute;
  top: 50%;
  left: 15%;
  width: 70%;
  color: white;
  text-align: center;
}
```

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:



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;
```



HTML

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

      <a href="#" class="submit">Hire Me!</a>
    </div>
  </div>
```

CSS

```
.submit{
  display:block;
  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;
  text-decoration: none;}
  }
```

z-index property

The z-index property specifies the stack order of an element.

An element with greater stack order is always in front of an element with a lower stack order.

Note: z-index only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky) and flex items (elements that are direct children of display:flex elements).

Note: If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

Example) Use z-index on two overlap images and one text

To do so, we first use a <figure> element to hold two images and one paragraph

HTML

```
<figure>
    <img src="images/nyc4.jpg" alt="New York City" class="nyc1">
    <img src="images/nyc1.jpg" alt="New York City" class="nyc2">
    <p class="nyc_txt">New York City</p>
  </figure>
```

Once we have the HTML layout, in the CSS file, we can use z-index to set the order of how the images and paragraph display on the <figure> element. For example, we can set the first image, with `class="nyc1"`, to the back of the other image in the paragraph using z-index: -1. After it, the second image can have z-index:0 and the paragraph with z-index: 1.

CSS

```
figure{width: 500px; height: 400px; position: relative;}
.nyc1{ width:100%;position: absolute; z-index: 0;}
.nyc2{ width:50%;position: absolute; border: ridge white 1em; z-index:1; }
.nyc_txt{font-size: 4em; position: absolute; z-index:2; }
```

The output in the browser looks as:

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.

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

HTML

```
<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;
}
```

Blurred Images

blurred with 1px

blurred with 4px

blurred with 8px

blurred with 12px

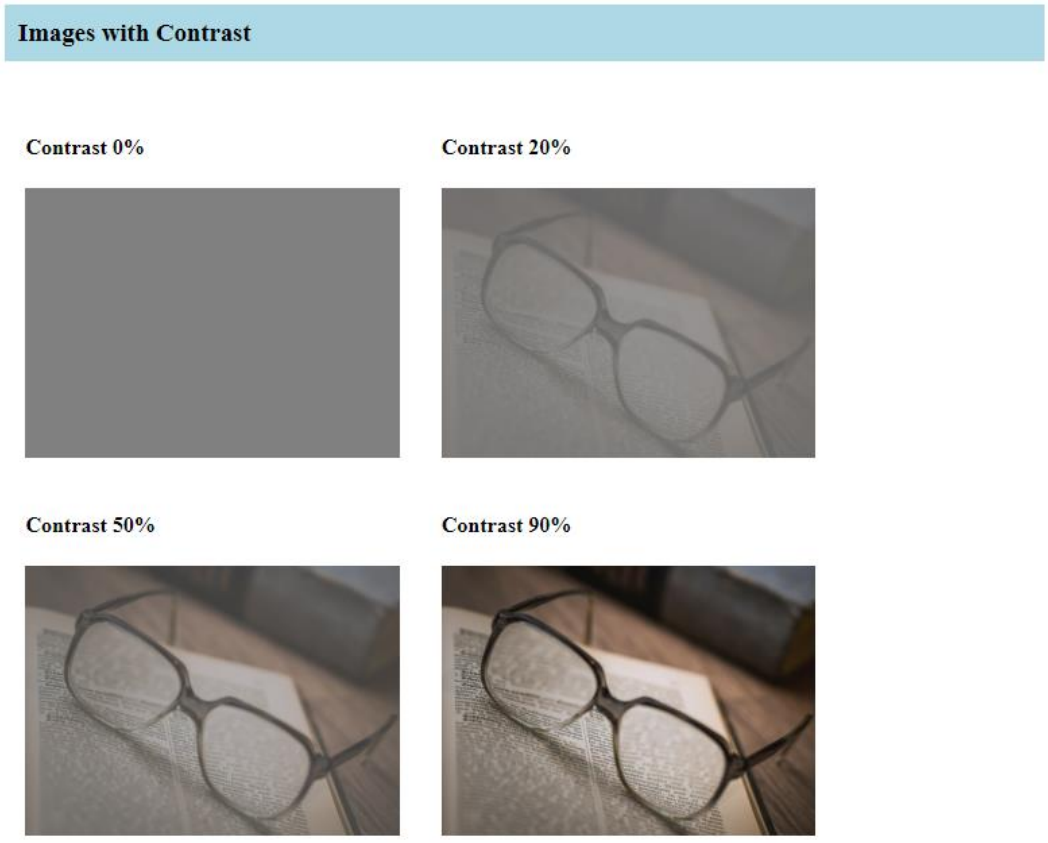
contrast(%)

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>
```

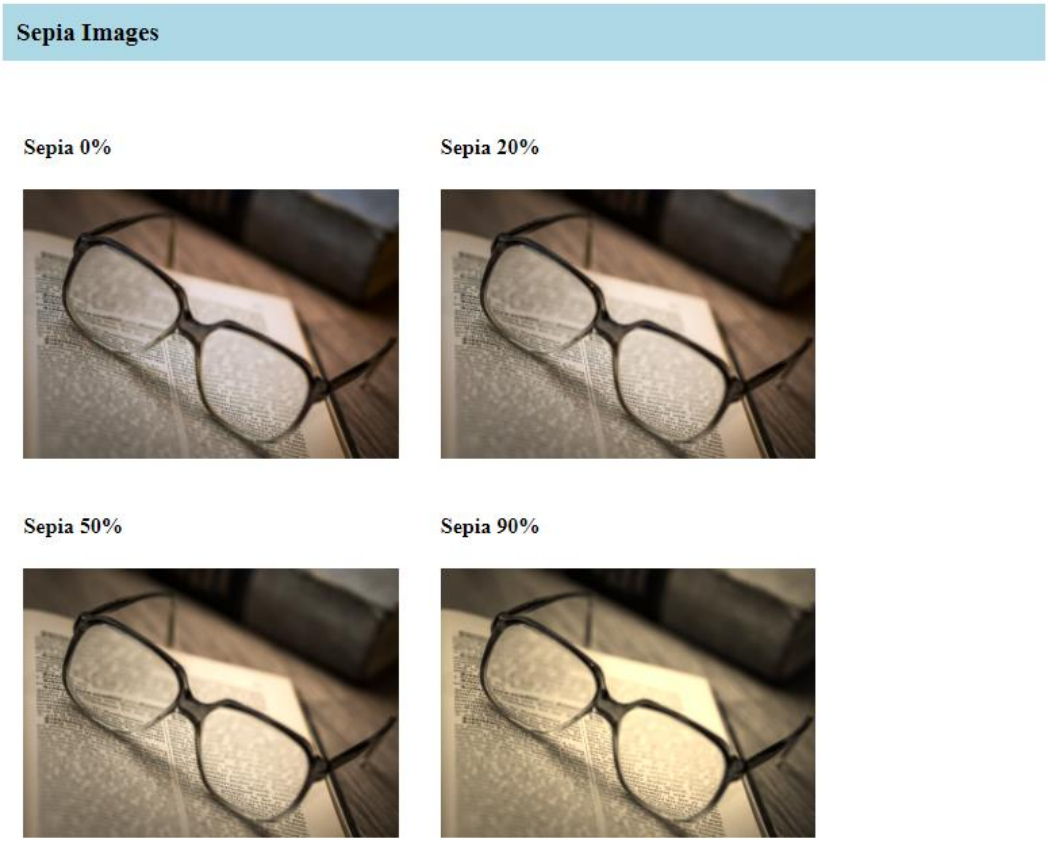


sepia(%)

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>
```



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>
```