

Full Stack Development Using Javascript-1

Unit-5 Advance CSS

5.1 Style Images

- Images are an important part of any web application.
- Including a lot of images in a web application is generally not recommended, but it is important to use the images wherever they are required.
- CSS helps us to control the display of images in web applications.
- The styling of an image in CSS is similar to the styling of an element by using the borders and margins.
- There are multiple CSS properties such as **border** property, **height** property, **width** property, etc. that help us to style an image.

Rounded Images

- The border-radius property sets the radius of the bordered image. It is used to create the rounded images. The possible values for the rounded corners are given as follows:
 1. **border-radius**: It sets all of the four border-radius property.
 2. **border-top-right-radius**: It sets the border of the top-right corner.
 3. **border-top-left-radius**: It sets the border of the top-left corner.
 4. **border-bottom-right-radius**: It sets the border of the bottom-right corner.
 5. **border-bottom-left-radius**: It sets the border of the bottom-left corner.

Example:

```
<html>
  <head><title>CSS style images</title></head>
  <style>
    img
    {
      border: 1px solid blue;
      border-radius:5px;
      width:200px;
    }

  </style>
  <body>
    <h1>Rounded Image</h1>
    
  </body>
</html>
```

Output:

Rounded Image



Thumbnail:

- The border property is used to make a thumbnail image.

Example:

```
<html>
  <head><title>CSS style images</title></head>
  <style>
    img
    {
      border: 1px solid red;
      border-radius:5px;
      padding:50px;
      width:200px;
    }

  </style>
  <body>
    <h1>Thumbanail Image</h1>
    
  </body>
</html>
```

Output:

Thumbnail Image



Responsive Images:

- It automatically adjusts to fit on the screen size. It is used to adjust the image to the specified box automatically.

Example:

```
<html>
  <head><title>CSS style images</title></head>
  <style>
    img
    {
      max-width:auto;
      height:auto;
    }
  </style>
  <body>
    <h1>Responsive Image</h1>
    
  </body>
</html>
```

Output:

Responsive Image



Transparent Images:

- To make an image transparent, we have to use the **opacity** property. The value of this property lies between 0.0 to 1.0, respectively.

Example:

```
<html>
  <head><title>CSS style images</title></head>
  <style>
    img
    {
      opacity:0.2;
    }
  </style>
  <body>
    <h1>Transparent Image</h1>
    
  </body>
</html>
```

Output:

Transparent Image



CSS Gradients

- CSS gradients let you display smooth transitions between two or more specified colors.
- CSS defines three types of gradients:
 1. **Linear Gradients** (goes down/up/left/right/diagonally)
 2. **Radial Gradients** (defined by their center)
 3. **Conic Gradients** (rotated around a center point)

1. CSS Linear Gradients:

- To create a linear gradient you must define at least two color stops.
- Color stops are the colors you want to render smooth transitions among.
- You can also set a starting point and a direction (or an angle) along with the gradient effect.
- **Direction - Top to Bottom (this is default)**

Syntax:

background-image: linear-gradient(*direction*, *color-stop1*, *color-stop2*, ...);

Example (Direction top to Bottom):

```
<html>
  <head><title>GRADIENT</title>
  <style>
    .grad
    {
      background-image:linear-gradient(pink,red);
    }
  </style>
</head>
<body>
  <pre class="grad">
```

This linear gradient starts pink at the top, transitioning to red at the bottom

.
.
.
.

```
    </pre>
  </body>
</html>
```

Output:

This linear gradient starts pink at the top, transitioning to red at the bottom

.
.
.
.

Example (Left to Right)

```
<html>
  <head><title>GRADIENT</title>
  <style>
    .grad
    {
      background-image:linear-gradient(to right,pink,red);
    }
  </style>
</head>
<body>
  <pre class="grad">
```

This linear gradient starts pink at the left, transitioning to red to the right

.
.
.

```

    </pre>
  </body>
</html>

```

Output:

This linear gradient starts pink at the left, transitioning to red to the right

```

.
.
.
.

```

Example (Diagonal)

- You can make a gradient diagonally by specifying both the horizontal and vertical starting positions.
- The following example shows a linear gradient that starts at top left (and goes to bottom right). It starts red, transitioning to yellow:

```

<html>
  <head><title>GRADIENT</title>
  <style>
    .grad
    {
      background-image:linear-gradient(to bottom right,pink,red);
    }
  </style>
</head>
<body>
  <pre class="grad">

```

This linear gradient starts pink at the top eft, transitioning to red to the bottom right

```

.
.
.
.

```

```

    </pre>
  </body>
</html>

```

Output:

This linear gradient starts pink at the top eft, transitioning to red to the bottom right

```

.
.
.
.

```

Example (Using angle)

- If you want more control over the direction of the gradient, you can define an angle, instead of the predefined directions (to bottom, to top, to right, to left, to bottom right, etc.).
- A value of 0deg is equivalent to "to top". A value of 90deg is equivalent to "to right". A value of 180deg is equivalent to "to bottom".

Syntax:

background-image: linear-gradient(*angle*, *color-stop1*, *color-stop2*);

Example:

```
<html>
<head>
<style>
#grad1 {
  height: 100px;
  background-image: linear-gradient(0deg, blue, powderblue);
}

#grad2 {
  height: 100px;
  background-image: linear-gradient(90deg, blue, powderblue);
}

#grad3 {
  height: 100px;
  background-image: linear-gradient(180deg, blue, powderblue);
}

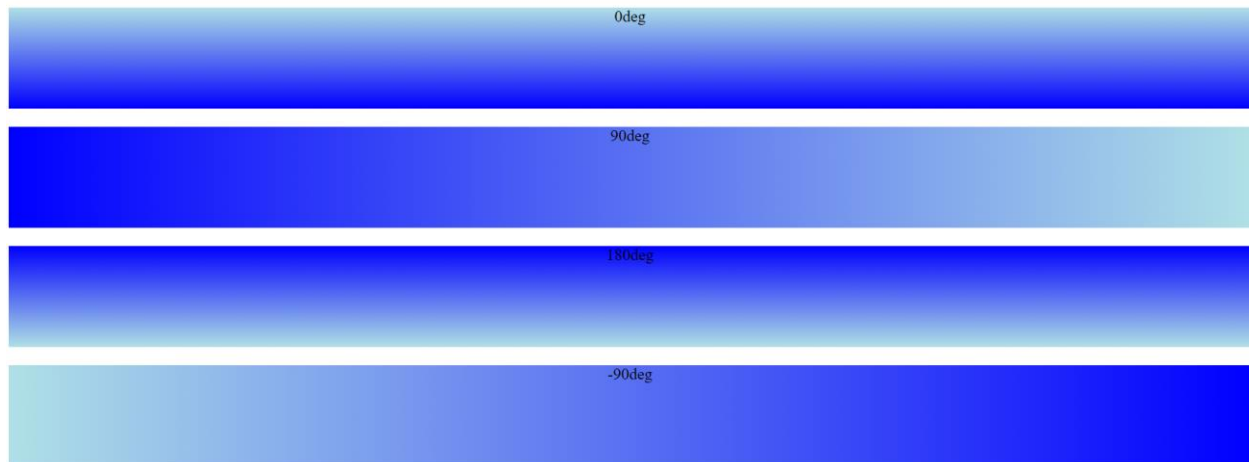
#grad4 {
  height: 100px;
  background-image: linear-gradient(-90deg, blue, powderblue);
}
</style>
</head>
<body>

<h1>Linear Gradients - Using Different Angles</h1>
<div id="grad1" style="text-align:center;">0deg</div><br>
<div id="grad2" style="text-align:center;">90deg</div><br>
<div id="grad3" style="text-align:center;">180deg</div><br>
<div id="grad4" style="text-align:center;">-90deg</div>

</body>
</html>
```


Output:

Linear Gradients - Using Different Angles



Example (Using Multiple colors)

```
<html>
<head>
<style>
#grad1 {
  height: 100px;
  background-image: linear-gradient(red, yellow, green);
}

#grad2 {
  height: 100px;
  background-image: linear-gradient(red, orange, yellow, green, blue, indigo, violet);
}

#grad3 {
  height: 100px;
  background-image: linear-gradient(red 20%, green 30%, blue 50%);
}
</style>
</head>
<body>
```

<p>Note: Color stops are spaced evenly when no percents are specified.</p>

<h2>3 Color Stops (evenly spaced):</h2>

<div id="grad1"></div>

<h2>7 Color Stops (evenly spaced):</h2>

```
<div id="grad2"></div>
```

```
<h2>3 Color Stops (not evenly spaced):</h2>
```

```
<div id="grad3"></div>
```

```
</body>
```

```
</html>
```

Output:

Note: Color stops are spaced evenly when no percents are specified.

3 Color Stops (evenly spaced):



7 Color Stops (evenly spaced):



3 Color Stops (not evenly spaced):



Example (Rainbow):

```
<html>
<head>
<style>
#grad1 {
  height: 50px;
  background-image: linear-gradient(to right, red, orange, yellow, green, blue, indigo, violet);
}
</style>
</head>
<body>
```

```
<div id="grad1" style="text-align:center;margin:auto;color:#FFFFFF;font-size:40px;font-
weight:bold">
```

Rainbow Background

```
</div>  
</body>  
</html>
```

Output:



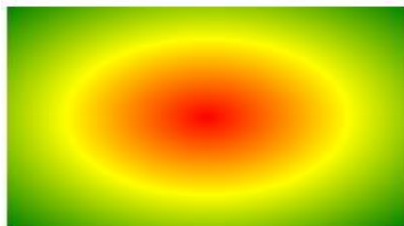
2.CSS Radial Gradients

- The radial-gradient() function sets a radial gradient as the background image.
- A radial gradient is defined by its center.
- To create a radial gradient you must define at least two color stops.

Example (Evenly Spaced color stops by default):

```
<html>  
<head>  
<style>  
#grad1 {  
  height: 150px;  
  width: 200px;  
  background-color: red; /* For browsers that do not support gradients */  
  background-image: radial-gradient(red, yellow, green);  
}  
</style>  
</head>  
<body>  
<div id="grad1"></div>  
</body>  
</html>
```

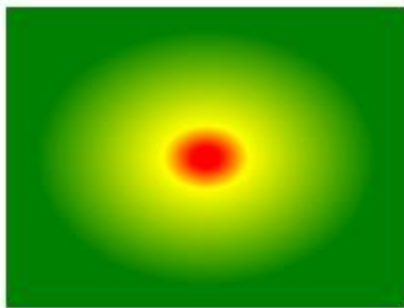
Output:



Example (Differently Spaced color stops)

```
<html>
<head>
<style>
#grad1 {
  height: 150px;
  width: 200px;
  background-color: red; /* For browsers that do not support gradients */
  background-image: radial-gradient(red 5%, yellow 15%, green 60%);
}
</style>
</head>
<body>
<div id="grad1"></div>
</body>
</html>
```

Output:



Example (set shape)

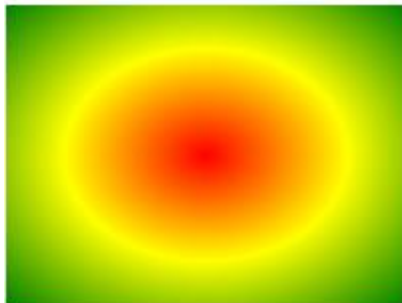
```
<html>
<head>
<style>
#grad1 {
  height: 150px;
  width: 200px;
  background-color: red; /* For browsers that do not support gradients */
  background-image: radial-gradient(red, yellow, green);
}

#grad2 {
  height: 150px;
  width: 200px;
  background-color: red; /* For browsers that do not support gradients */
  background-image: radial-gradient(circle, red, yellow, green);
}
```

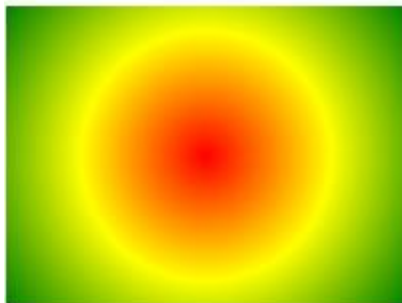
```
}  
</style>  
</head>  
<body>  
  
<h2>Ellipse (this is default):</h2>  
<div id="grad1"></div>  
  
<h2><strong>Circle:</strong></h2>  
<div id="grad2"></div>  
  
</body>  
</html>
```

Output:

Ellipse (this is default):



Circle:



3. CSS Conic Gradients

- A conic gradient is a gradient with color transitions rotated around a center point.
- To create a conic gradient you must define at least two colors.

Example:

```
<html>
<head>
<style>
#grad1 {
  height: 200px;
  width: 200px;
  background-color: red; /* For browsers that do not support gradients */
  background-image: conic-gradient(red, yellow, green);
}
</style>
</head>
<body>

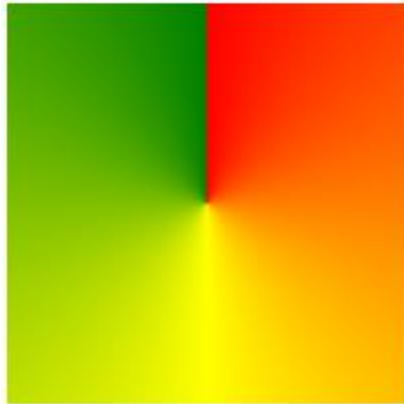
<h1>Conic Gradient - Three Colors</h1>

<div id="grad1"></div>

</body>
</html>
```

Output:

Conic Gradient - Three Colors



Example:

```
<html>
<head>
<style>
#grad1 {
  height: 200px;
```

```

width: 200px;
background-color: red; /* For browsers that do not support gradients */
background-image: conic-gradient(red, yellow, green, blue, black);
}
</style>
</head>
<body>

<h1>Conic Gradient - Five Colors</h1>

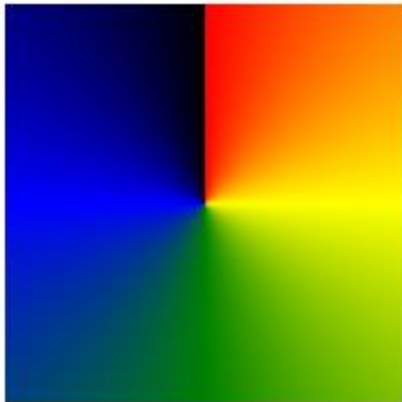
<div id="grad1"></div>

</body>
</html>

```

Output:

Conic Gradient - Five Colors



Example (3 colors and degrees)

```

<html>
<head>
<style>
#grad1 {
height: 200px;
width: 200px;
background-color: red; /* For browsers that do not support gradients */
background-image: conic-gradient(red 45deg, yellow 90deg, green 210deg);
}
</style>

```

```
</head>
<body>

<h1>Conic Gradient - Defined degree for each color</h1>

<div id="grad1"></div>

</body>
</html>
```

Output:

Conic Gradient - Defined degree for each color



Example (Pie Charts)

- Just add border-radius: 50% to make the conic gradient look like a pie:

```
<html>
<head>
<style>
#grad1 {
  height: 200px;
  width: 200px;
  background-color: red; /* For browsers that do not support gradients */
  background-image: conic-gradient(red, yellow, green, blue, black);
  border-radius: 50%;
}
</style>
</head>
<body>
```



```
<h1>Conic Gradient - Pie Chart</h1>
```

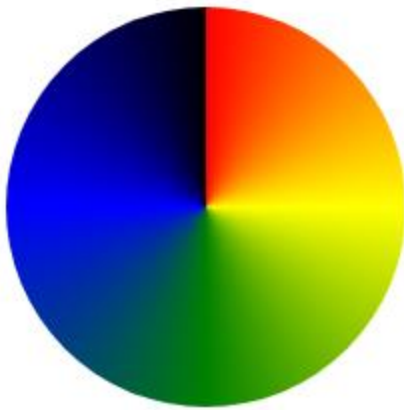
```
<div id="grad1"></div>
```

```
</body>
```

```
</html>
```

Output:

Conic Gradient - Pie Chart



Example:

```
<html>
```

```
<head>
```

```
<style>
```

```
#grad1 {
```

```
  height: 200px;
```

```
  width: 200px;
```

```
  background-color: red; /* For browsers that do not support gradients */
```

```
  background-image: conic-gradient(red 0deg, red 90deg, yellow 90deg, yellow 180deg, green 180deg, green 270deg, blue 270deg);
```

```
  border-radius: 50%;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1>Conic Gradient - Pie Chart</h1>
```

```
<div id="grad1"></div>
```

```
</body>
</html>
```

Output:

Conic Gradient - Pie Chart



Example (conic gradient with specified from angle):

```
<html>
<head>
<style>
#grad1 {
  height: 200px;
  width: 200px;
  background-color: red; /* For browsers that do not support gradients */
  background-image: conic-gradient(from 90deg, red, yellow, green);
  border-radius: 50%;
}
</style>
</head>
<body>

<h1>Conic Gradient - With a from angle</h1>

<div id="grad1"></div>

</body>
</html>
```

Output:

Conic Gradient - With a from angle



CSS Transitions

- CSS transitions allows you to change property values smoothly, over a given duration.
- Properties: **transition**, **transition-delay**, **transition-duration**.

transition

- The transition effect will start when the specified CSS property (width) changes value.
- A shorthand property for setting the four transition properties into a single property
- To create a transition effect, you must specify two things:
 1. the CSS property you want to add an effect to
 2. the duration of the effect
- Now, let us specify a new value for the width property when a user mouses over the <div> element:

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

Example:

```
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
```

```
}  
  
div:hover {  
  width: 300px;  
}  
</style>  
</head>  
<body>  
  
<h1>The transition Property</h1>  
  
<p>Hover over the div element below, to see the transition effect:</p>  
<div></div>  
  
</body>  
</html>
```

Output (Before hover):

Hover over the div element below, to see the transition effect:



Output (After hover)



transition-delay

- The transition-delay property specifies a delay (in seconds) for the transition effect.
- The following example has a 1 second delay before starting:

Example:

```
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 3s;
  transition-delay: 1s;
}

div:hover {
  width: 300px;
}
</style>
</head>
<body>
```

<p>Hover over the div element below, to see the transition effect:</p>

<div></div>

<p>Note: The transition effect has a 1 second delay before starting.</p>

```
</body>
</html>
```

Output (before hover):

Hover over the div element below, to see the transition effect:



Note: The transition effect has a 1 second delay before starting.

Output (after hover):

Hover over the div element below, to see the transition effect:



Note: The transition effect has a 1 second delay before starting.

transition -duration

- Specifies how many seconds or milliseconds a transition effect takes to complete

Example:

```
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition-property: width;
  transition-duration: 2s;
  transition-delay: 1s;
}

div:hover {
  width: 300px;
}
</style>
</head>
<body>
<p>Hover over the div element below, to see the transition effect:</p>

<div></div>

<p><b>Note:</b> The transition effect has a 1 second delay before starting.</p>

</body>
</html>
```

Output (before hover):

Hover over the div element below, to see the transition effect:



Note: The transition effect has a 1 second delay before starting.

Output (after hover):

Hover over the div element below, to see the transition effect:



Note: The transition effect has a 1 second delay before starting.

transition example (form fields):

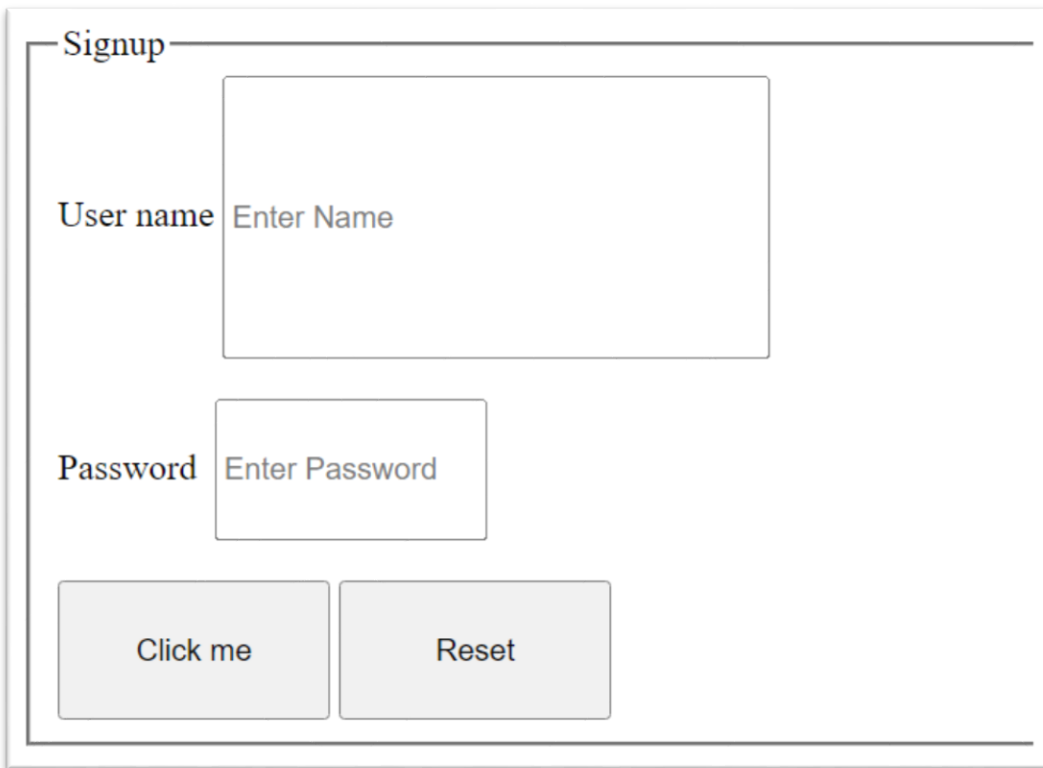
```
<html>
  <head>
    <style>
      input
      {
        width:10%;
        height:10%;
        transition:height 4s,width 4s;
        transition-delay:2s;
      }
      input:hover
      {
        height:30%;
        width:30%;
      }
    </style>
  </head>
  <body>
    <form method="post" action="demo.html">
```

```
<fieldset><legend>Signup</legend>

User name
<input type="text" maxlength="10" placeholder="Enter Name"/>
<br/><br/>
Password
  &nbsp;<input type="password" maxlength="8" placeholder="Enter Password">
<br/>
<br/>
<input type="submit" value="Click me"/>
<input type="reset" value="Reset"/>

</fieldset>
</form>
</body>
</html>
```

Output:



Signup

User name Enter Name

Password Enter Password

Click me Reset

CSS animation

- CSS allows animation of HTML elements without using JavaScript or Flash!
- An animation lets an element gradually change from one style to another.
- You can change as many CSS properties you want, as many times as you want.
- To use CSS animation, you must first specify some keyframes for the animation.
- Keyframes hold what styles the element will have at certain times.

Properties

- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction

@keyframes

- When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.
- To get an animation to work, you must bind the animation to an element.
- The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":
- **Note:** The animation-duration property defines how long an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).
- In the below example we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).
- It is also possible to use percent. By using percent, you can add as many style changes as you like.

Example:

```
<html>
<head>
<style>
div {
  animation-name: example;
  width: 100px;
  height: 100px;
  background-color: red;
```

```
    animation-duration: 4s;
}
```

```
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
</style>
</head>
<body>
```

```
<div></div>
```

```
<p><b>Note:</b> When an animation is finished, it goes back to its original style.</p>
```

```
</body>
</html>
```

Output (before animation):



Note: When an animation is finished, it goes back to its original style.

Output (after animation)



Note: When an animation is finished, it goes back to its original style.

Example (Using percentage)

```
<html>
<head>
```

```
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

```
@keyframes example {
  0% {background-color: red;}
  25% {background-color: yellow;}
  50% {background-color: blue;}
  100% {background-color: green;}
}
</style>
</head>
<body>
```

```
<div></div>
```

<p>Note: When an animation is finished, it goes back to its original style.</p>

```
</body>
</html>
```

Output:



Note: When an animation is finished, it goes back to its original style.

Output (after animation):



Note: When an animation is finished, it goes back to its original style.

Example:

- The following example will change both the background-color and the position of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
}

@keyframes example {
  0% {background-color:red; left:0px; top:0px;}
  25% {background-color:yellow; left:200px; top:0px;}
  50% {background-color:blue; left:200px; top:200px;}
  75% {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>

<div></div>

</body>
</html>
```

animation-delay

- The animation-delay property specifies a delay for the start of an animation.
- The following example has a 2 seconds delay before starting the animation:

Example:

```
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
  animation-delay: 2s;
}

@keyframes example {
  0% {background-color:red; left:0px; top:0px;}
  25% {background-color:yellow; left:200px; top:0px;}
  50% {background-color:blue; left:200px; top:200px;}
  75% {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>

<div></div>

</body>
</html>
```

Example (Negative animation-delay value)

- Negative values are also allowed. If using negative values, the animation will start as if it had already been playing for N seconds.
- In the following example, the animation will start as if it had already been playing for 2 seconds:

```
<html>
<head>
<style>
```

```
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
  animation-delay: -2s;
}
```

```
@keyframes example {
  0% {background-color:red; left:0px; top:0px;}
  25% {background-color:yellow; left:200px; top:0px;}
  50% {background-color:blue; left:200px; top:200px;}
  75% {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>
```

<p>Using negative values in the animation-delay property: Here, the animation will start as if it had already been playing for 2 seconds:</p>

```
<div></div>
```

```
</body>
</html>
```

animation-iteration-count

- The animation-iteration-count property specifies the number of times an animation should run.
- The animation-iteration-count property can be set to infinite to let the animation run for ever:
- The following example will run the animation 3 times before it stops:

Example:

```
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
```

```
background-color: red;
position: relative;
animation-name: example;
animation-duration: 4s;
animation-iteration-count: 3;
}
```

```
@keyframes example {
  0% {background-color:red; left:0px; top:0px;}
  25% {background-color:yellow; left:200px; top:0px;}
  50% {background-color:blue; left:200px; top:200px;}
  75% {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>
```

```
<h1>CSS Animation</h1>
```

```
<p>The animation-iteration-count property can be set to infinite to let the animation run for ever:</p>
```

```
<div></div>
```

```
</body>
</html>
```

animation-direction

- The animation-direction property specifies whether an animation should be played forwards, backwards or in alternate cycles.
- **Shorthand: animation: myfirst 5s 2s infinite alternate;**
- The animation-direction property can have the following values:
 1. **normal** - The animation is played as normal (forwards). This is default
 2. **reverse** - The animation is played in reverse direction (backwards)
 3. **alternate** - The animation is played forwards first, then backwards
 4. **alternate-reverse** - The animation is played backwards first, then forwards

Example:

```
<html>
<head>
<style>
div {
  width: 100px;
```

```
height: 100px;
background-color: red;
position: relative;
animation-name: example;
animation-duration: 4s;
animation-direction: reverse;
}
```

```
@keyframes example {
  0% {background-color:red; left:0px; top:0px;}
  25% {background-color:yellow; left:200px; top:0px;}
  50% {background-color:blue; left:200px; top:200px;}
  75% {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>
```

<p>The animation-direction property specifies whether an animation should be played forwards, backwards or in alternate cycles. The following example will run the animation in reverse direction (backwards):</p>

```
<div></div>
```

```
</body>
</html>
```

CSS Display Property:

- The display property specifies the display behavior (the type of rendering box) of an element.
- Possible values are:
 1. **Inline** : Displays an element as an inline element (like). Any height and width properties will have no effect
 2. **Block** : Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width
 3. **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
 4. **none**: The element is completely removed

Example:

```
<html>
<head>
<style>
.p1
{
display:none;
}
.p2
{
display:inline;
color:red;
}
.p3
{
display:block;
color:red;
}
.p4
{
display:inline-block;
color:red;
}
</style>
</head>
<body>
<div> This is a demo<p class="p1">THIS IS ME</p>sfsdfgdh</div><br/><br/>
<div> This is ABC DEMo<p class="p2">THIS IS ABC</p>fhfghfjh</div><br/><br/>
<div> This is XYZ DEMo<p class="p3">THIS IS XYZ</p>fghfgjghj</div><br/><br/>
<div> This is PQR DEMO<p class="p4">THIS IS PQR</p>fghfgj</div><br/><br/>
</body>
</html>
```

Output:

This is a demofsdfgdh

This is ABC DEMoTHIS IS ABCfhfghfjh

This is XYZ DEMo

THIS IS XYZ

fghfgjghj

This is PQR DEMOTHIS IS PQRfghfgj

CSS Tooltip

- A tooltip is often used to specify extra information about something when the user moves the mouse pointer over an element
- Properties are: visibility, width, background-color, color, text-align.

Example:

```
<html>
<head>
  <style>
    .tooltip
    {
      display:inline-block;
      border-bottom:1px dotted black;
    }
    .tooltip .tooltiptxt
    {
      visibility:hidden;
      width:120px;
      background-color:black;
      color:white;
      text-align:center;
      border-radius:10px;
      padding:5px;
      position:absolute;
    }

    .tooltip:hover .tooltiptxt
    {
      visibility:visible;
    }
  </style>
</head>
<body>
  <h1>Tooltip Demo</h1>
  <div class="tooltip">
    HOVER ON ME
    <span class="tooltiptxt">TOOLTIP TEXT INFORMATION</span>
  </div>

</body>
</html>
```

Output (before hover):

Tooltip Demo

HOVER ON ME

Output (after hover):

Tooltip Demo

HOVER ON ME

TOOLTIP TEXT
INFORMATION

Flip an Image

- The image is flipped to create a mirror image. Flip an image means rotating the image horizontally or vertically.
- The transform: scaleX(-1) property is used to flip the image horizontally.
- The transform property is used to rotate the image and scaleX(-1) rotates the image to axial symmetry. Hence the original image is flipped to its mirror image.
- The transform: rotate (180 deg) will flip the image vertically.
- The transform: scaleX(-2) property double the mirror image horizontally.

Example:

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
img:hover {

    transform: scaleX(-1);
}
</style>
</head>
<body>

<h2>Flip an Image</h2>
<p>Move your mouse over the image.</p>


</body>
</html>
```

Output (before hover):

Flip an Image

Move your mouse over the image.



Output (after hover):

Flip an Image

Move your mouse over the image.



CSS Buttons

- In HTML, we use the button tag to create a button, but by using CSS properties, we can style the buttons.
- Buttons help us to create user interaction and event processing.
- They are one of the widely used elements of web pages.
- Properties: background-color, border, color, padding, text-align, display, font-size, **cursor**.

Cursor property:

Value	Description
pointer	The cursor is a pointer and indicates a link
alias	The cursor indicates an alias of something is to be created
auto	Default. The browser sets a cursor
zoom-in	The cursor indicates that something can be zoomed in
zoom-out	The cursor indicates that something can be zoomed out
wait	The cursor indicates that the program is busy
progress	The cursor indicates that the program is busy (in progress)
not-allowed	The cursor indicates that the requested action will not be executed
grab	The cursor indicates that something can be grabbed
none	No cursor is rendered for the element

Example:

```
<html>
<head>
  <style>
    .button
    {
      text-decoration:none;
      background-color:blue;
      border:none;
      color:white;
      padding:10px 3px;
      margin:7px 6px;
      text-align:center;
      display:inline-block;
      font-size:20px;
      cursor:not-allowed;
    }
  </style>
</head>
<body>
```

```
<button>CLICK ME</button>
<a href="#" class="button">ENTER</a>
<button class="button">SUBMIT</button>
<input type="button" class="button" value="BUTTON"/>
</body>
</html>
```

Output:



Hoverable Buttons

- Use the :hover selector to change the style of a button when you move the mouse over it.
- **Tip:** Use the transition-duration property to determine the speed of the "hover" effect:

Example:

```
<html>
<head>
  <style>
    .b2
    {
      text-decoration:none;
      background-color:white;
      border:none;
      color:lightgreen;
      padding:10px 3px;
      margin:7px 6px;
      text-align:center;
      display:inline-block;
      font-size:20px;
      cursor:progress;
      transition-duration:0.5s;
      border:1px solid lightgreen;
    }
    .b2:hover
    {
      background-color:lightgreen;
      color:white;
    }
  </style>
```

```
</head>
<body>
  <button class="b2">HOVER</button>
</body>
</html>
```

Output (before hover):



Output (after hover):



CSS Multiple Column

- The CSS multi-column layout allows easy definition of multiple columns of text - just like in newspapers:
- Properties are:
 1. **column-count**: Specifies the number of columns an element should be divided into
 2. **column-gap**: Specifies the gap between the columns
 3. **column-rule-style**: Specifies the style of the rule between columns

Example:

```
<html>
  <head>
    <title>CSS MULTIPLE COLUMNS</title>
    <style>
      .news
      {
        column-count:3;
        column-gap:20px;
        column-rule-style:solid;
        column-rule-color:pink;
        column-rule-width:20px;
      }
    </style>
  </head>
</html>
```


CSS Pagination

- CSS pagination is a very useful technique for indexing different pages of a website on the homepage.
- If your website has lots of pages, you have to add some sort of pagination to each page.

Simple pagination

Example:

```
<html>
<head>
<style>

.pagination
{
display:inline-block;
}
.pagination a
{
color:black;
text-decoration:none;    // if not used none then it shows underline.
padding:20px;
}

</style>
</head>
<body>
<div class="pagination">
<a href="#">&laquo;</a>
<a href="#">1</a>
<a href="#">2</a>
<a href="#">3</a>
<a href="#">&raquo;</a>

</body>
</html>
```

Output:

Simple Pagination

« 1 2 3 4 5 6 »

Active and Hoverable pagination:

- Highlight the current page with an .active class, and use the :hover selector to change the color of each page link when moving the mouse over them:

Example:

```
<html>
<head>
<style>
.pagination {
  display: inline-block;
}

.pagination a {
  color: black;
  float: left;
  padding: 8px 16px;
  text-decoration: none;
}

.pagination a.active {
  background-color: #4CAF50;
  color: white;
}

.pagination a:hover:not(.active) {background-color: #ddd;}
</style>
</head>
<body>

<h2>Active and Hoverable Pagination</h2>

<p>Move the mouse over the numbers.</p>
<div class="pagination">
  <a href="#">&laquo;</a>
  <a href="#">1</a>
  <a class="active" href="#">2</a>
  <a href="#">3</a>
  <a href="#">4</a>
  <a href="#">5</a>
  <a href="#">6</a>
  <a href="#">&raquo;</a>
</div>
</body>
</html>
```

Output (before hover):

Active and Hoverable Pagination

Move the mouse over the numbers.

« 1 2 3 4 5 6 »

Output (after hover):

Active and Hoverable Pagination

Move the mouse over the numbers.

« 1 2 3 4 5 6 »

Rounded Active & Hoverable Buttons:

- Add the border-radius property if you want a rounded "active" and "hover" button:

Example:

```
<html>
<head>
<style>
.pagination {
  display: inline-block;
}

.pagination a {
  color: black;
  float: left;
  padding: 8px 16px;
  text-decoration: none;
}

.pagination a.active {
  background-color: #4CAF50;
  color: white;
  border-radius: 5px;
}
```

```
.pagination a:hover:not(.active) {
  background-color: #ddd;
  border-radius: 5px;
}
</style>
</head>
<body>
```

<h2>Rounded Active and Hover Buttons</h2>

```
<div class="pagination">
  <a href="#">&laquo;</a>
  <a href="#">1</a>
  <a href="#" class="active">2</a>
  <a href="#">3</a>
  <a href="#">4</a>
  <a href="#">5</a>
  <a href="#">6</a>
  <a href="#">&raquo;</a>
</div>

</body>
</html>
```

Output (before hover):

Rounded Active and Hover Buttons

« 1 2 3 4 5 6 »

Output (after hover):

Rounded Active and Hover Buttons

« 1 2 3 4 5 6 »

Bordered pagination:

- Use the border property to add borders to the pagination:

Example:

```
<html>
<head>
<style>
.pagination {
  display: inline-block;
}

.pagination a {
  color: black;
  float: left;
  padding: 8px 16px;
  text-decoration: none;
  border: 1px solid #ddd;
}

.pagination a.active {
  background-color: #4CAF50;
  color: white;
  border: 1px solid #4CAF50;
}

.pagination a:hover:not(.active) {background-color: #ddd;}
</style>
</head>
<body>

<h2>Pagination with Borders</h2>

<div class="pagination">
  <a href="#">&laquo;</a>
  <a href="#">1</a>
  <a href="#" class="active">2</a>
  <a href="#">3</a>
  <a href="#">4</a>
  <a href="#">5</a>
  <a href="#">6</a>
  <a href="#">&raquo;</a>
</div>

</body>
</html>
```

Output:

Pagination with Borders

«	1	2	3	4	5	6	»
---	---	---	---	---	---	---	---

Pagination size:

- Change the size of the pagination with the font-size property:

Example:

```
<html>
<head>
<style>
.pagination {
  display: inline-block;
}

.pagination a {
  color: black;
  float: left;
  padding: 8px 16px;
  text-decoration: none;
  transition: background-color .3s;
  border: 1px solid #ddd;
  font-size: 22px;
}

.pagination a.active {
  background-color: #4CAF50;
  color: white;
  border: 1px solid #4CAF50;
}

.pagination a:hover:not(.active) {background-color: #ddd;}
</style>
</head>
<body>

<h2>Pagination Size</h2>
<p>Change the font-size property to make the pagination smaller or bigger.</p>
```

```

<div class="pagination">
  <a href="#">&laquo;</a>
  <a href="#">1</a>
  <a href="#" class="active">2</a>
  <a href="#">3</a>
  <a href="#">4</a>
  <a href="#">5</a>
  <a href="#">6</a>
  <a href="#">&raquo;</a>
</div>
</body>
</html>

```

Output:

Pagination Size

Change the font-size property to make the pagination smaller or bigger.



CSS Variable:

- The var() function is used to insert the value of a CSS variable.
- CSS variables have access to the DOM, which means that you can create variables with local or global scope, change the variables with JavaScript, and change the variables based on media queries.
- A good way to use CSS variables is when it comes to the colors of your design. Instead of copy and paste the same colors over and over again, you can place them in variables.
- CSS variables can have a global or local scope.
- Global variables can be accessed/used through the entire document, while local variables can be used only inside the selector where it is declared.
- To create a variable with global scope, declare it inside the :root selector. The :root selector matches the document's root element.
- To create a variable with local scope, declare it inside the selector that is going to use it.

Example:

```

<html>
<head>
<style>
:root
{

```

```

--blue:#1234EE;
--white:#FFDDCC;
--xyz:center;
}
h1
{
--blue:red;
color:var(--blue);
}
p
{
color:var(--white);
background-color:var(--blue);
text-align:var(--xyz);
}
</style>
</head>
<body>

<h1 >Hello</h1>
<p>Good Morning!</p>
</body>
</html>

```

Output:

Hello

Good Morning!

Example:

```

<html>
<head>
<style>
:root
{
--blue:#ffffff;
--white:#FFDDCC;
}
h1
{
--blue:#000011;
color:var(--blue);

```



```
text-align:left;
border-bottom:5px solid var(--blue);
margin:20px;
}

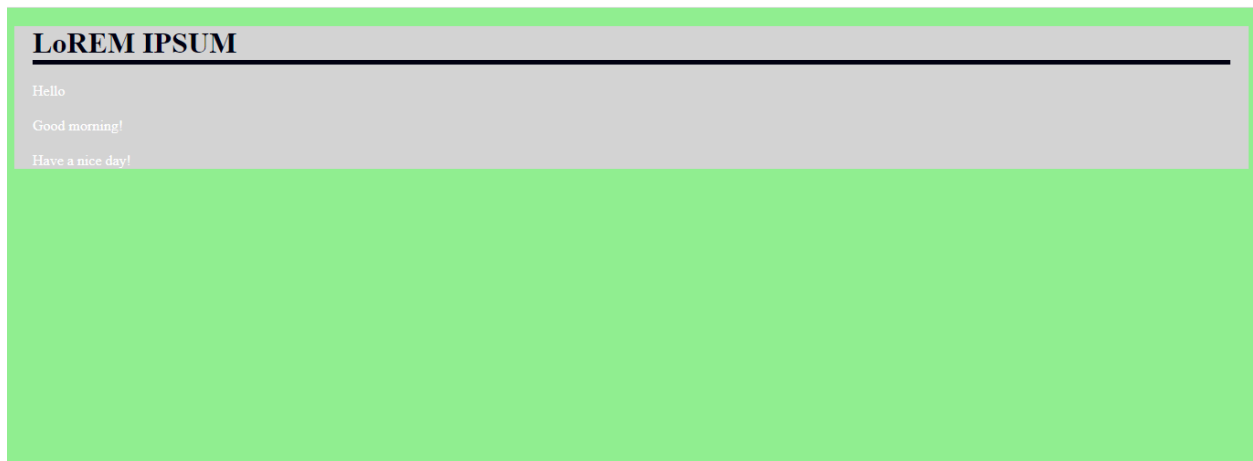
p
{
color:var(--blue);
text-align:var(--xyz);
margin:20px;
}

</style>
</head>
<body bgcolor="lightgreen">
<div style="background-color:lightgray;">

<h1 >LoREM IPSUM</h1>
<p>Hello</p>
<p>Good morning!</p>
<p>Have a nice day!</p>

<div>
</body>
</html>
```

Output:



2D Transformation

CSS transforms allow you to move, rotate, scale, and skew elements. With the CSS transform property you can use the following 2D transformation methods:

- translate()
- rotate()
- skew()
- skewX()
- skewY()
- scaleX()
- scaleY()
- scale()

translate() :

The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

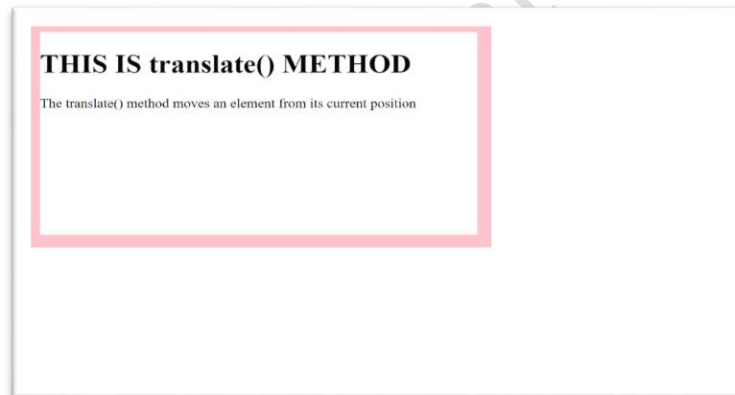
Example:

```
<html>
  <head>
    <style>
      div
      {
        border:20px solid pink;
        width:50%;
        height:50%;
        font-size:20px;
      }
      div:hover
      {
        transform:translate(180px,180px);
      }
    </style>
```

```
</head>
<body>
  <div>
    <h1>THIS IS translate() METHOD</h1>
    <p>The translate() method moves an element from its current position</p>
  </div>
</body>
</html>
```

Output:

Before hovering



After hovering



skew():

The skew() method skews an element along the X and Y-axis by the given angles.

Example:

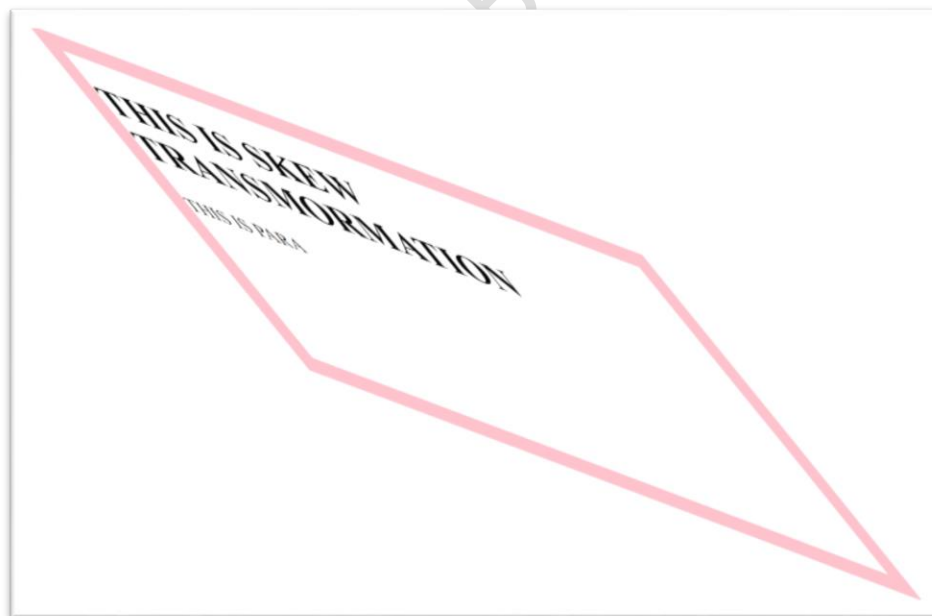
```
<html>
  <head>
    <style>
      div
      {
        margin:150px;
        border:20px solid pink;
        width:50%;
        height:50%;
        font-size:20px;
      }
      div:hover
      {
        transform:skew(40deg, 20deg);
      }
    </style>
  </head>
  <body>
    <div>
      <h1>THIS IS SKEW TRANSFORMATION</h1>
      <p>THIS IS PARA</p>
    </div>
  </body>
</html>
```

Output:

Before hovering



After hovering



skewX():

The skewX() method skews an element along the X-axis by the given angle..

Example:

```
<html>
  <head>
    <style>
      div
      {
        margin:150px;
        border:20px solid pink;
        width:50%;
        height:50%;
        font-size:20px;
      }
      div:hover
      {
        transform:skew(40deg, 0deg);
      }
    </style>
  </head>
  <body>
    <div>
      <h1>THIS IS SKEW TRANSFORMATION</h1>
      <p>THIS IS PARA</p>
    </div>
  </body>
</html>
```

Output:

Before hovering



After hovering



skewY():

The skewY() method skews an element along the Y-axis by the given angle..

Example:

```
<html>
  <head>
    <style>
      div
      {
        margin:150px;
        border:20px solid pink;
        width:50%;
        height:50%;
        font-size:20px;
      }
      div:hover
      {
        transform:skew(0deg, 20deg);
      }
    </style>
  </head>
  <body>
    <div>
      <h1>THIS IS SKEW TRANSFORMATION</h1>
      <p>THIS IS PARA</p>
    </div>
  </body>
</html>
```


Output:

Before hovering



After hovering



rotate() :

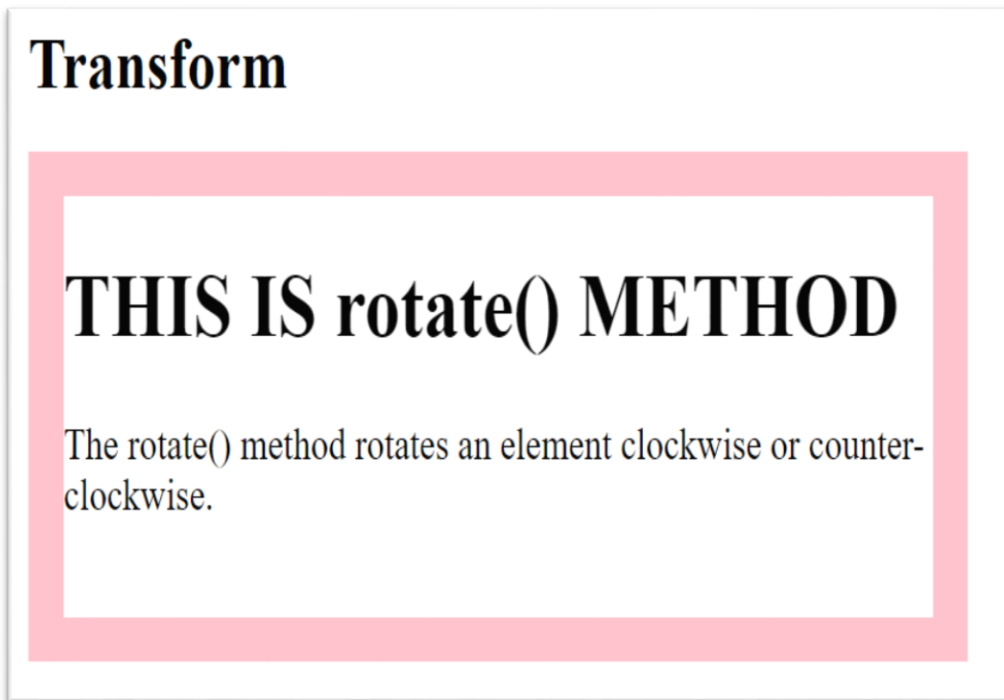
The rotate() method rotates an element clockwise or counter-clockwise according to a given degree. Using negative values will rotate the element counter-clockwise.

Example(Positive Value-clockwise):

```
<html>
  <head>
    <style>
      div
      {
        border:20px solid pink;
        width:40%;
        height:30%;
        font-size:20px;
      }
      div:hover
      {
        transform:rotate(20deg);
      }
    </style>
  </head>
  <body>
    <h1>Transform</h1>
    <div>
      <h1>THIS IS rotate() METHOD</h1>
      <p>The translate() method moves an element from its current position</p>
    </div>
  </body>
</html>
```

Output:

Before hovering



After hovering

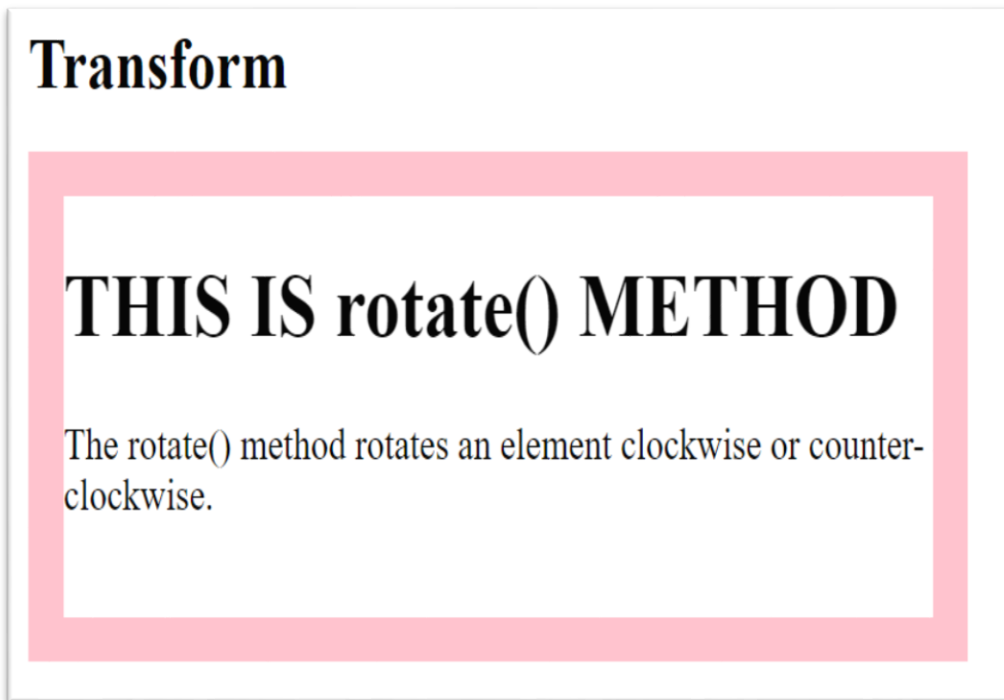


Example(Negative Value-counter clockwise):

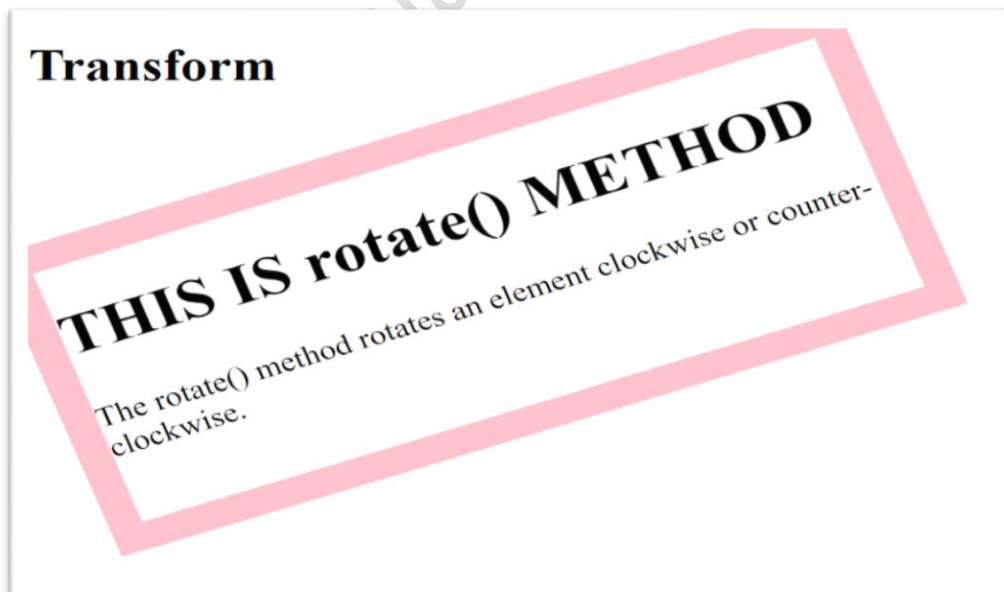
```
<html>
  <head>
    <style>
      div
      {
        border:20px solid pink;
        width:40%;
        height:30%;
        font-size:20px;
      }
      div:hover
      {
        transform:rotate(-20deg);
      }
    </style>
  </head>
  <body>
    <h1>Transform</h1>
    <div>
      <h1>THIS IS rotate() METHOD</h1>
      <p>The translate() method moves an element from its current position</p>
    </div>
  </body>
</html>
```

Output:

Before hovering



After hovering



scaleX()

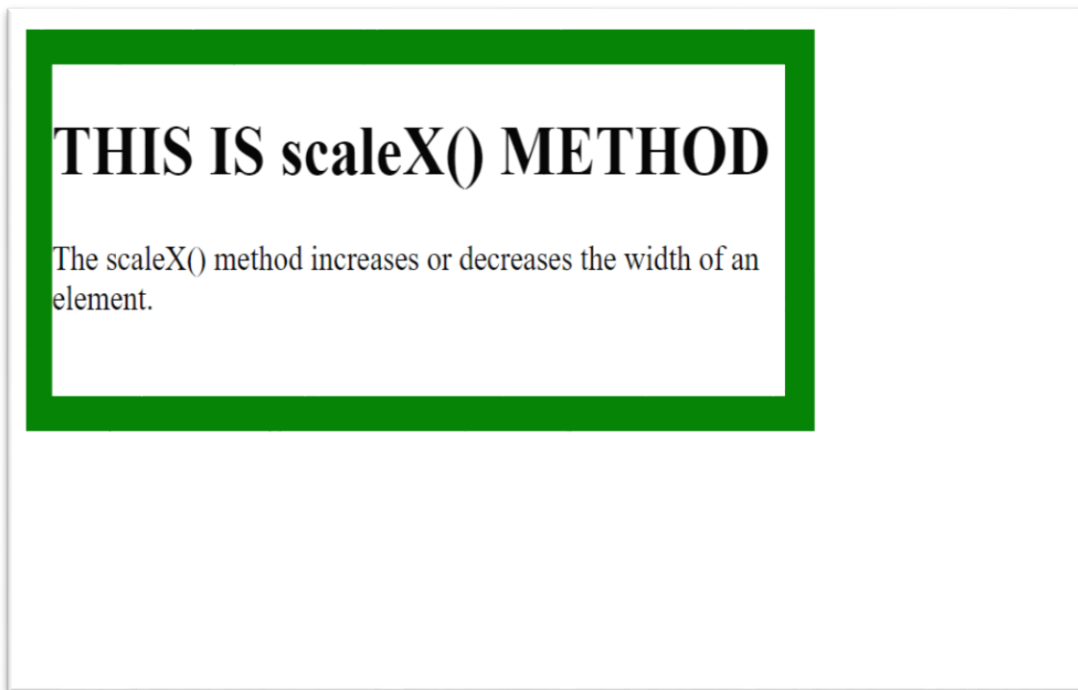
The scaleX() method increases or decreases the width of an element.

Example:

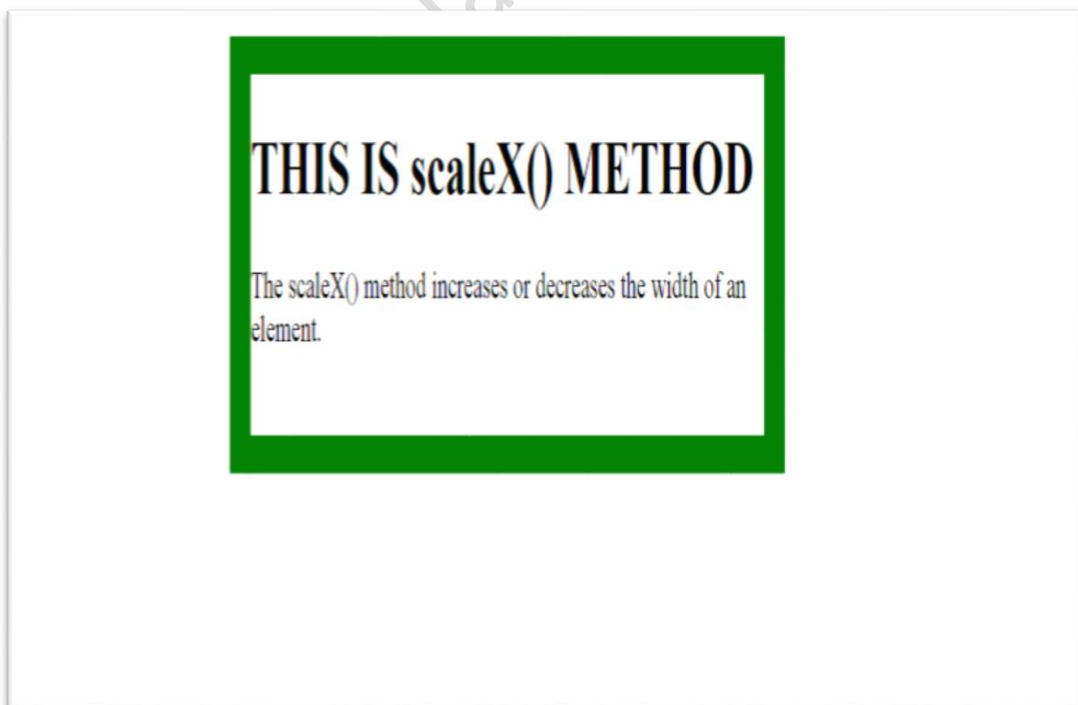
```
<html>
  <head>
    <style>
      div
      {
        border:20px solid green;
        width:40%;
        height:30%;
        font-size:20px;
      }
      div:hover
      {
        transform:scaleX(0.5);
      }
    </style>
  </head>
  <body>
    <h1>Transform</h1>
    <div>
      <h1>THIS IS scaleX() METHOD</h1>
      <p>The scaleX() method increases or decreases the width of an
element.</p>
    </div>
  </body>
</html>
```

Output:

Before hovering



After hovering



scaleY()

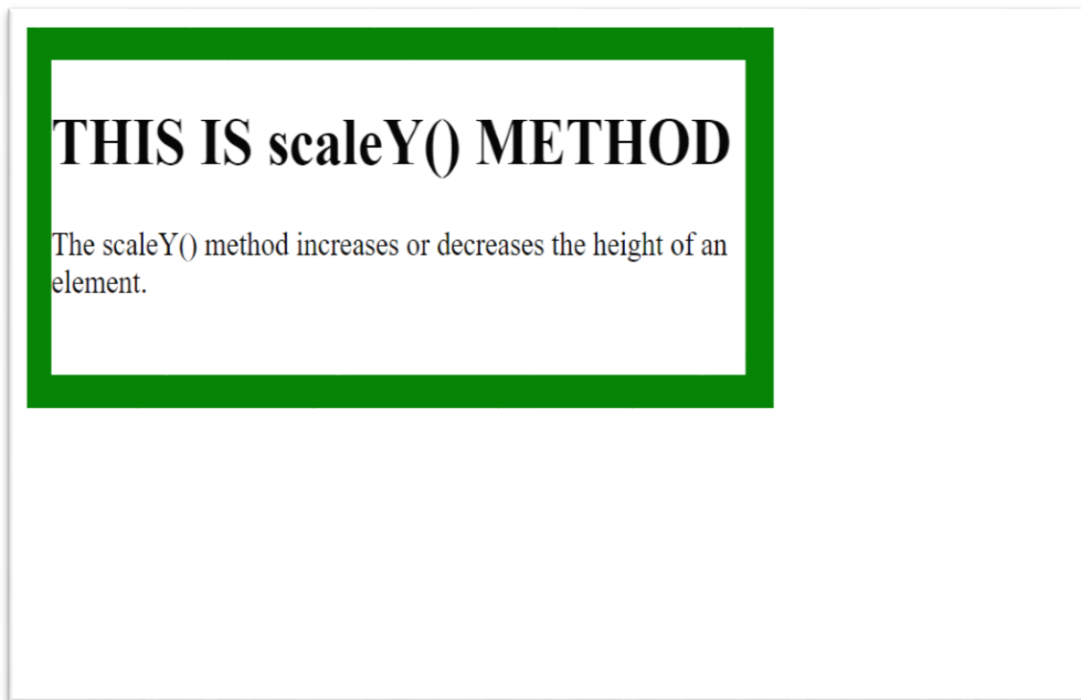
The scaleY() method increases or decreases the height of an element.

Example:

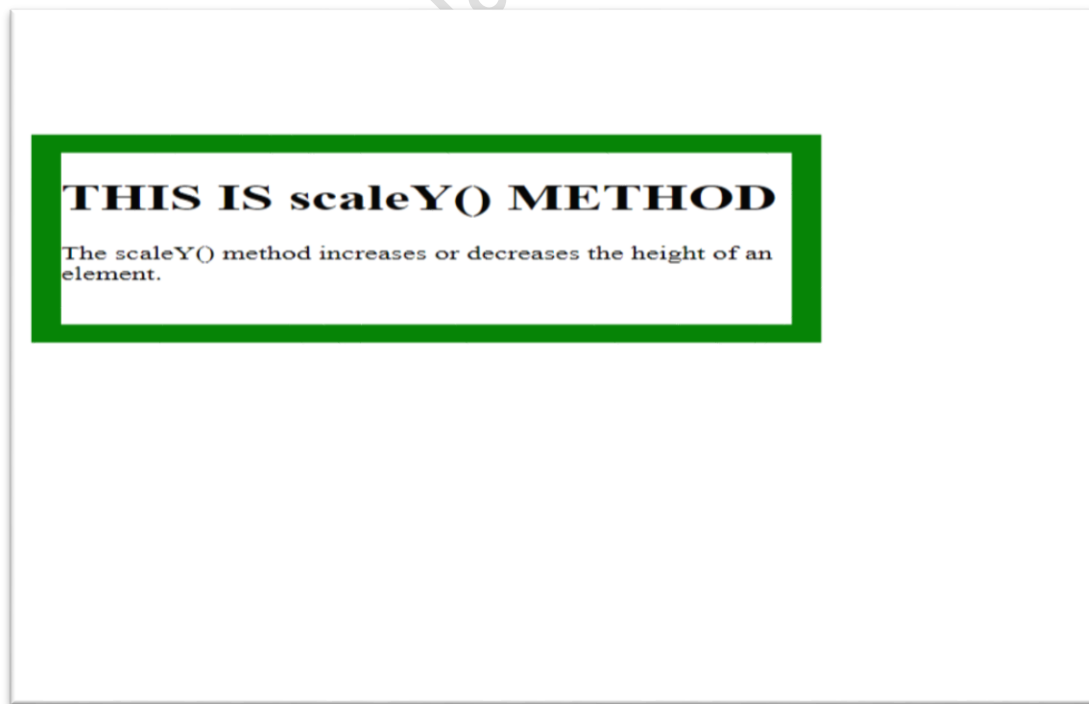
```
<html>
  <head>
    <style>
      div
      {
        border:20px solid green;
        width:40%;
        height:30%;
        font-size:20px;
      }
      div:hover
      {
        transform:scaleX(0.5);
      }
    </style>
  </head>
  <body>
    <h1>Transform</h1>
    <div>
      <h1>THIS IS scaleX() METHOD</h1>
      <p>The scaleX() method increases or decreases the width of an
element.</p>
    </div>
  </body>
</html>
```


Output:

Before hovering



After hovering



scale()

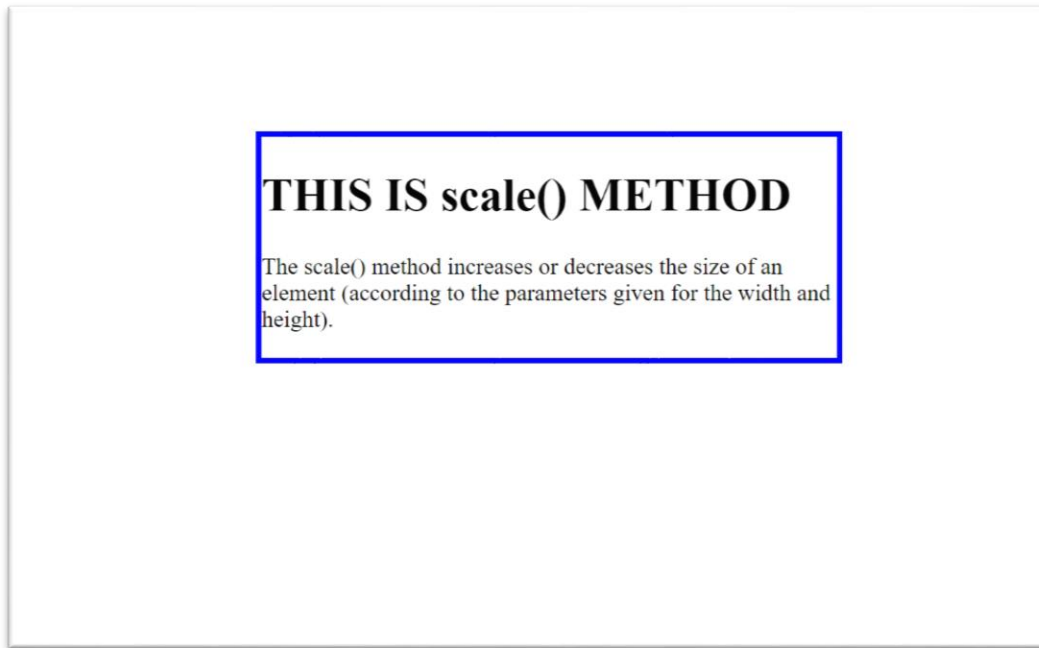
The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

Example:

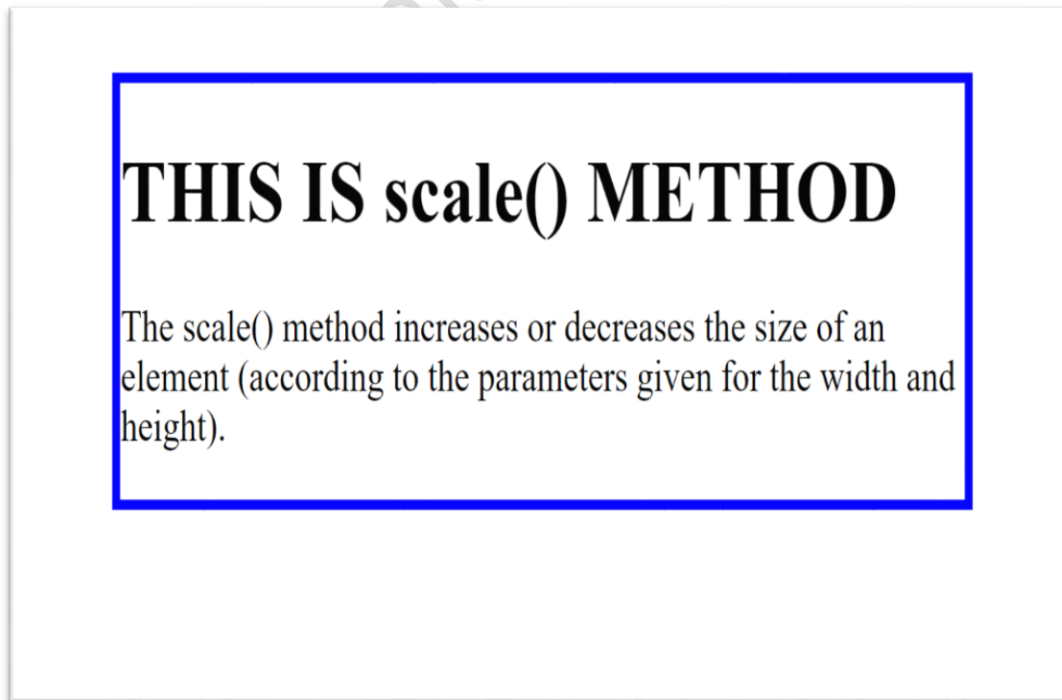
```
<html>
  <head>
    <style>
      div
      {
        border:5px solid blue;
        width:40%;
        height:30%;
        font-size:20px;
        margin-left:200px;
        margin-top:100px;
      }
      div:hover
      {
        transform:scale(1.5,1.5);
      }
    </style>
  </head>
  <body>
    <div>
      <h1>THIS IS scale() METHOD</h1>
      <p>The scale() method increases or decreases the size of an element
      (according to the parameters given for the width and height).</p>
    </div>
  </body>
</html>
```

Output:

Before hovering



After hovering



Media Queries

The media query in CSS is used to create a responsive web design to make a user-friendly website. It means that the view of web pages differs from system to system based on screen or media types.

Media is allowing us to reshape and design the user view page of the website for specific devices like Tablets, Desktops, Mobile phones, etc.

Media queries can be used to check many things like the following

- Width and height of the viewport
- Width and height of the device
- Orientation (Landscape, Portrait)
- Resolution

Syntax

@media not | only mediatype and (expression)

```
{  
    // Code content  
}
```

The result of the query is true if the specified media type matches the type of device the document is being displayed on and all expressions in the media query are true.

When a media query is true, the corresponding style sheet or style rules are applied, following the normal cascading rules.

Unless you use the not or only operators, the media type is optional and the all type will be implied.

Media Types

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

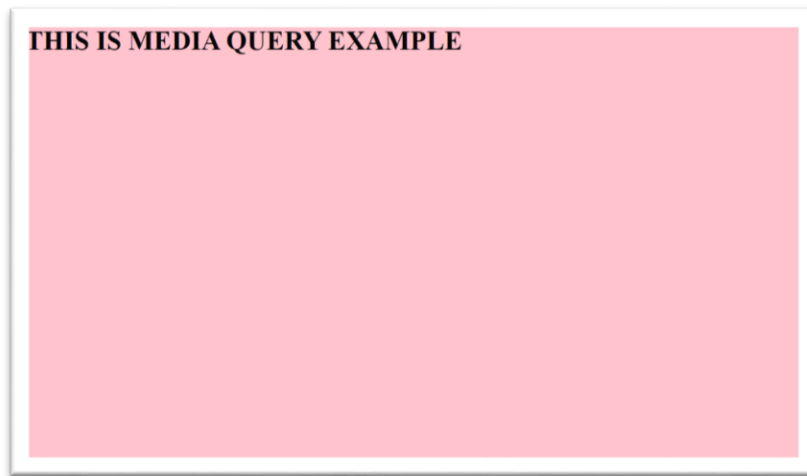
Example(Portrait, Landscape):

```
<html>
  <head>
    <meta name="viewport" content="width=device-width,initial-scale=1.0,user-
scalable=yes, minimum-scale=0.5, maximum-scale=5.0"/>
    <style>
      body
      {
        background-color:lightgreen;
      }
      @media only screen and (orientation:landscape)
      {
        body{background-color:pink;}
      }
    </style>
  </head>
</html>
```

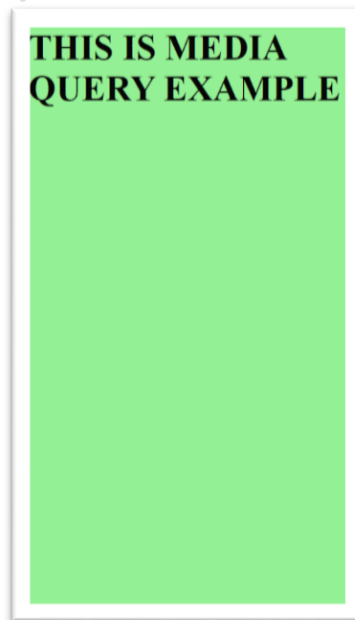
```
        </style>
    </head>
    <body>
        <h1>THIS IS MEDIA QUERY EXAMPLE</h1>
    </body>
</html>
```

Output:

Landscape mode



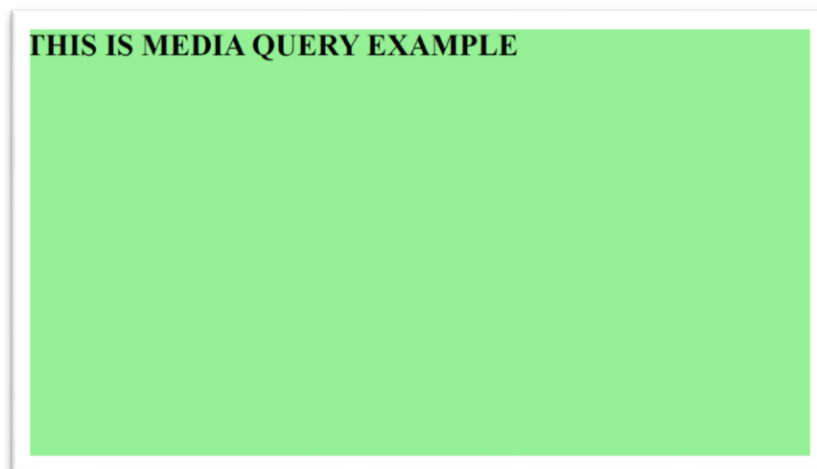
Portrait mode



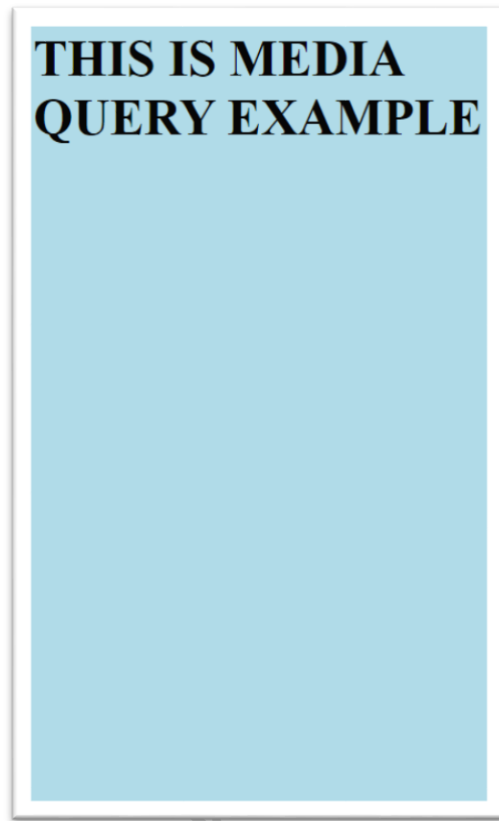
Example(max-width):

```
<html>
  <head>
    <meta name="viewport" content="width=device-width,initial-scale=1.0,user-
scalable=yes, minimum-scale=0.5, maximum-scale=5.0"/>
    <style>
      body
      {
        background-color:lightgreen;
      }
      @media only screen and (max-width:500px)
      {
        body{background-color:lightblue;}
      }
    </style>
  </head>
  <body>
    <h1>THIS IS MEDIA QUERY EXAMPLE</h1>
  </body>
</html>
```

Output:



After minimization



Zalakov