

Unit-5 - Advance CSS

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>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
img{
    border: 1px solid blue;
    border-radius:5px;
    width:200px;
}
</style></head>
<body>
<h1>Rounded Image</h1>

</body></html>
```

Output:

Rounded Image



- **Thumbnail Image**

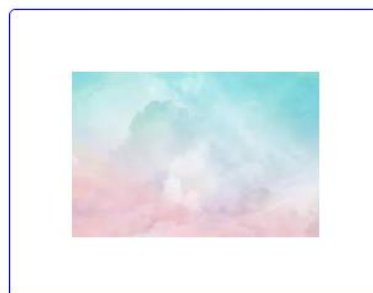
The border property is used to make a thumbnail image.

Example:

```
<html>
<head><title>CSS style images</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
img{
    border: 1px solid blue;
    border-radius:5px;
    width:200px;
    padding:50px;
}
</style></head>
<body>
    <h1>Rounded Image</h1>
    
</body>
</html>
```

Output:

Rounded Image



- **Responsive Image**

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

Example:

```
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS style images</title>
  <style>
    img
    {
      max-width:100%;
      height:auto;
    }
  </style>
</head>
<body>
  
</body>
</html>
<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 (Transparent) to 1.0 (Solid), respectively.

Example:

```
<html>
<head><title>CSS style images</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
img
{
    opacity:0.7;
}
</style></head>
<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.

Syntax:

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

Default Direction - Top to Bottom

Example (Direction top to Bottom):

```
<html>
<head><meta name="viewport" content="width=device-width, initial-scale=1.0">
<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:

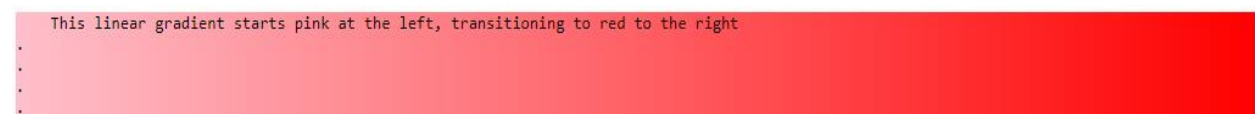
This linear gradient starts pink at the top, transitioning to red at the bottom



Example (Left to Right)

```
<html>
<head><title>GRADIENT</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<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:



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>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
.grad
{
    background-image:linear-gradient(to bottom right,pink,red);
}
</style>
</head>
<body>
<pre class="grad">
```

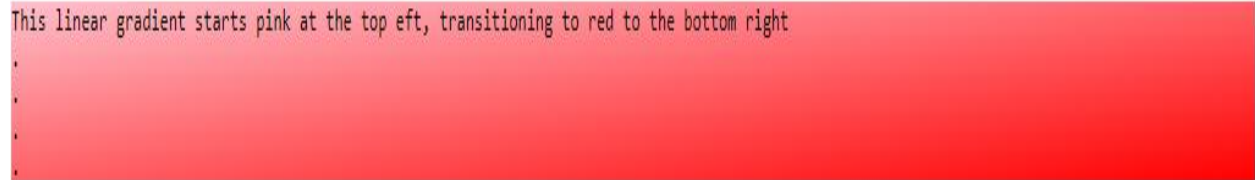
This linear gradient starts pink at the top left, transitioning to red to the bottom right

```
.
.
.
.
```

```
</pre>
```

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

Output:



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".

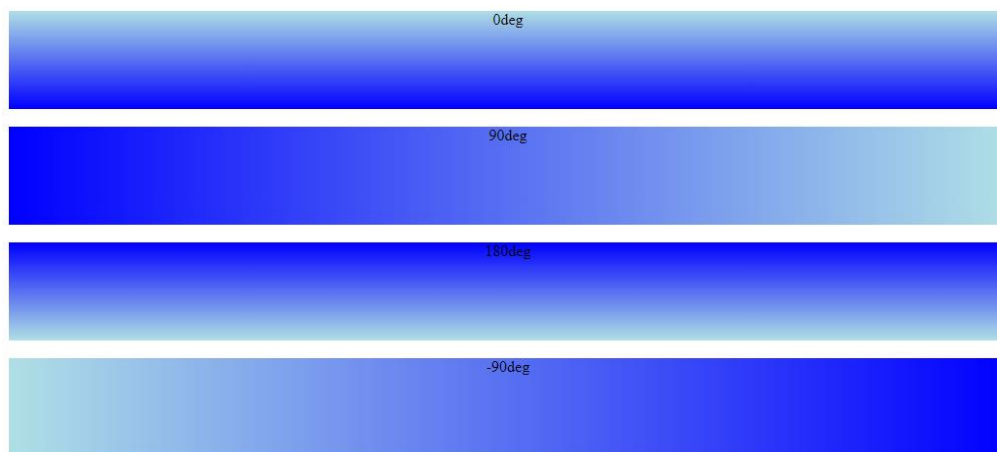
Syntax:

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

Example:

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<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>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<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><strong>Note:</strong> 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:**3 Color Stops (evenly spaced):****7 Color Stops (evenly spaced):****3 Color Stops (not evenly spaced):****Example (Rainbow):**

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<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 with ellipse):

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<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>
```

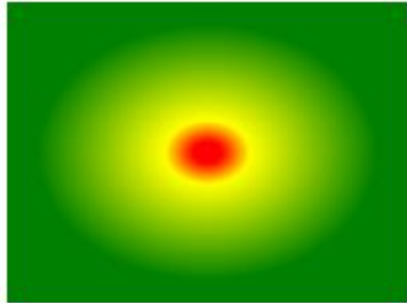
Output:



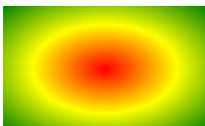
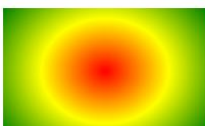
Example (Differently Spaced color stops)

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<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>
```

```
</style></head>
<body>
  <div id="grad1"></div>
</body></html>
```

Output:**Example (set shape)**

```
<html><head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<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

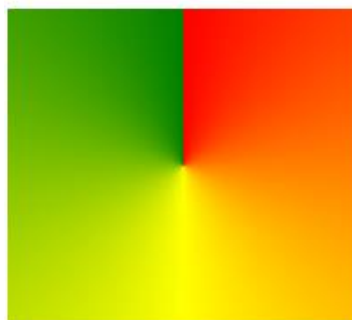
A conic gradient is a gradient with color transitions rotated around a center point. To create a conic gradient, you must define at least two colors.

Example:

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<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</h1>
    <div id="grad1"></div>
</body></html>
```

Output:

Conic Gradient

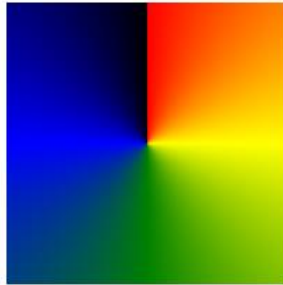
**Example with 5 color:**

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<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

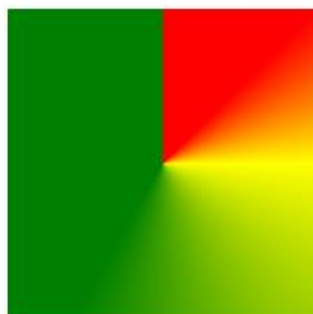


Example (3 colors and degrees)

```
<html>
<head><meta name="viewport" content="width=device-width, initial-scale=1.0">
<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</h1>
<div id="grad1"></div>
</body>
</html>
```

Output:

Conic Gradient



Example (Pie Charts)

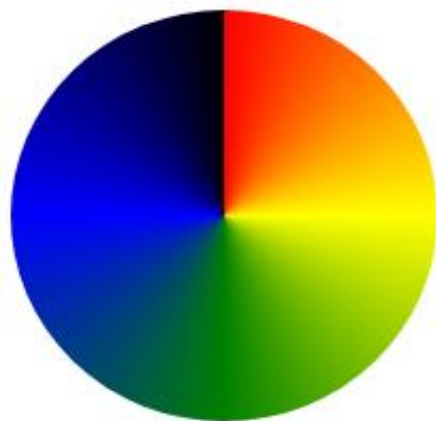
Just add border-radius: 50% to make the conic gradient look like a pie:

Conic Gradient - Pie Chart

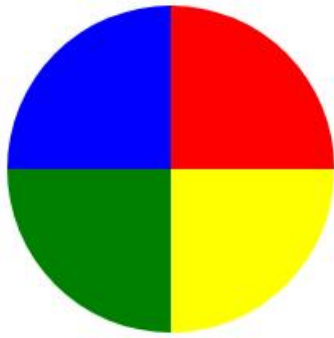
```
<html>
<head><meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
#grad1 {
  height: 200px;
  width: 200px;
  background-color: red; /* For browsers that do not support gradients */
  background-image: conic-gradient(red, yellow, green, blue, black);
  border-radius: 50%;
}
</style>
</head>
<body>
<h1>Conic Gradient - Pie Chart</h1>
<div id="grad1"></div>
</body>
</html>
```

Output:

Conic Gradient - Pie Chart



Task: Write a code to design image:



```
<html>
<head><meta name="viewport" content="width=device-width, initial-scale=1.0"><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%;
}
#grad2 {
    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>
    <div id="grad1"></div>
    <br>
    <div id="grad2"></div>
</body></html>
```

CSS Transitions

- CSS transitions allows you to change property values smoothly, over a given duration.
- Properties: **transition-property**, **transition-delay**, **transition-duration**.
- a new value for the width property when a user mouses over the <div> element:

- **transition-property:** This property allows you to select the CSS properties which you want to animate during the transition(change).

Syntax: transition-property: all | property | property1,property2, ..., propertyN;

Values: **all** is used to specify all the properties to be selected, though not all properties are animate-able, only the properties which are animate-able will be influenced.

We can specify a single **property** or a set of comma-separated properties **property1, property2, ..., propertyN**.

- **transition-duration:** This property allows you to determine how long it will take to complete the transition from one CSS property to the other.

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

Syntax:

transition-duration: time;

Here, **time** can be in seconds(s) or milliseconds(ms), you should use 's' or 'ms' after the number

- **transition-delay:** This property allows you to determine the amount of time to wait before the transition actually starts to take place.

Syntax:

transition-delay: time;

- **The Shorthand Property transition**

You can combine all the transition properties mentioned above, into one single shorthand property, according to the syntax given below.

Syntax:

transition: (property name) | (duration) | (delay);

transition: width 6s,height 9s;background-color 8s;

Example:

```
<html>
<head>
<style>
div {
    width: 100px;
    height: 100px;
    background: red;
    transition-property: width; transition-duration: 2s;
/* transition:width 2s ; can write in shorthand which cover above two effects; */
}
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**The transition Property**

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

during hover**The transition Property**

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



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:

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

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

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

Output:

The transition Property

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



The transition Property

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



Transition -duration

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

Example:

```
<html>
<head>
<style>
div {
width: 100px;
height: 100px;
background-color: red;
/* transition-property: background-color; */
transition-duration: 5s;
}
div:hover {
width: 300px;
/* background-color: pink; */
}
</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:

The transition Property

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



The transition Property

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



The Shorthand Property transition

You can combine all the transition properties mentioned above, into one single shorthand property, according to the syntax given below.

Syntax:

transition: (property name) | (duration) | (delay);

Transition example (form fields on hover):

```
<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

Password

Transform

CSS 2D Transforms Methods

With the CSS transform property you can use the following 2D transformation methods:

translate()	Moves an element from its current position (according to the parameters given for the X-axis and the Y-axis). Syntax: transform: translate(100px, 75px); Note: translateX(100px) - to translate in X direction translateY(75px) - to translate in Y direction
rotate()	rotates an element (+ve) clockwise or (-ve) counter-clockwise according to a given degree . (**Used to Rotate Image) Syntax: transform: rotate(45deg)
scale()	Increases or decreases the size of an element (according to the parameters given for the width and height). Syntax: transform: scale(2,3);
scaleX()	Increases or decreases the width of an element Syntax: transform: scaleX(2);
scaleY()	Increases or decreases the height of an element. Syntax: transform: scaleY(2);
skew()	Skews an element along the X and Y-axis by the given angles. Syntax: transform: skew(50deg, -15deg);
skewX()	skews an element along the X-axis by the given angle
skewY()	skews an element along the Y-axis by the given angle.

translate()

- The translate() Method The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Yaxis).
- The following example moves the <div> element 50 pixels to the right, and 100 pixels down from its current position.

EXAMPLE

```
<html><head><style>
  div {
    width: 300px;
    height: 100px;
    background-color: pink;
    border: 1px solid black;
    padding: 10px;
  }
  div:hover { transform: translate(50px,100px); }
</style></head>
<body>
  <h1>The translate() Method</h1>
  <p>The translate() method moves an element from its current position:</p>
  <div> This div element is moved 50 pixels to the right, and 100 pixels down from
its current position. </div>
</body></html>
```

The translate() Method

The translate() method moves an element from its current position:

This div element is moved 50 pixels to the right, and 100 pixels down from its current position.

The translate() Method

The translate() method moves an element from its current position:

This div element is moved 50 pixels to the right, and 100 pixels down from its current position.

- **translateX(50px):** It moves the element 50px along X-axis.
- **translateY(100px):** It moves the element 100px along Y-axis.

rotate()

- The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.
- For counter-clockwise give value in (-) negative.
- The following example rotates the one image clockwise with 20 degrees and the other image anticlockwise with -40deg.

EXAMPLE

```
<html>
<head>
<style>
.i1 {
width: 600px;
height: 300px;
border: 1px solid black;
}
.i1:hover
{ transform: rotate(20deg);
}
.i2 {
width: 600px;
height: 300px;
border: 1px solid black;
}
.i2:hover { transform: rotate(-40deg); }
</style></head>
<body>
<h1>The rotate() Method</h1>
<p>The rotate() method rotates an element clockwise or counter-clockwise.</p>


</body>
</html>
```



scale()

- The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).
- The following example increases the <div> element to be two times of its original width, and three times of its original height:

EXAMPLE

```
<html> <head> <style>
div { margin: 100px; width: 200px; height: 100px; background-color:yellow;
border: 1px solid black; }

div:hover { transform: scale(2,3); }
</style> </head>
<body>
<h1>The scale() Method</h1>
<p>The scale() method increases or decreases the size of an element.</p>
<div> This div element is two times of its original width, and 2.5 times of its
original height. </div>
</body>
</html>
```

The scale() Method

The scale() method increases or decreases the size of an element.

This div element is 2 times of its original width, and 2.5 times of its original height.

The scale() Method

The scale() method increases or decreases the size of an element.

This div element is 2 times of its original width, and 2.5 times of its original height.

scaleX()

- ✓ The scaleX() method increases or decreases the width of an element.
- ✓ The following example decreases width of the <div> element by half of its original width:
 - **div { transform: scaleX(0.5); }**
- ✓ The following example increases the <div> element to be 2 times of its original width:
 - **div { transform: scaleX(2); }**

scaleY()

- ✓ The scaleY() method increases or decreases the height of an element.
- ✓ The following example increases the <div> element to be three times of its original height:
 - **div { transform: scaleY(3); }**
- ✓ The following example decreases the <div> element to be half of its original height:
 - **div { transform: scaleY(0.5); }**

skew()

- The skew() method skews an element along the X and Y-axis by the given angles.
- The **skewX()** method skews an element along the X-axis by the given angle.
- The following example skews the <div> element 20 degrees along the X- axis:

EXAMPLE

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
.i1 {
  margin: 100px;
  width:300px;height:200px
}
.i1:hover {
  transform: skewX(20deg);
}
</style>
</head>
<body>
  <div class="flip-box-inner">
    
  </div>
</body>
</html>
```



Task: Transform Example(All in One on Hover):

```

<html> <head> <style>
div{
height:100px; width:200px; border:1px solid black; background:aqua; float:left; margin:20px
}
.p1:hover{
    transform:rotate(50deg);background: tomato;text-align: center;font-weight: bolder;
}
.p2:hover{
    transform:rotatex(180deg);background: tomato;text-align: center;font-weight: bolder;
}
.p3:hover{
transform:rotatey(180deg);background: tomato;text-align: center;font-weight: bolder;
}
.p4:hover{
transform:skew(20deg,0deg);background: tomato;text-align: center;font-weight: bolder;
}
.p5:hover{
transform:scale(2,0.5);background: tomato;text-align: center;font-weight: bolder;}
/* width height 2 means double 0.5 means half*/
.p6:hover{
transform:scalex(2);background: tomato;text-align: center;font-weight: bolder;}
/*only width */
.p7:hover{
transform:scaley(2);background: tomato;text-align: center;font-weight: bolder;}
/*only height */
.p8:hover{
transform:translate(50px,100px);background: tomato;text-align: center;font-weight: bolder;}
/* MOVE TO NEXT POINT*/
    </style> </head>
<body>
    <div class="p1">Rotate 50 deg</div>
    <div class="p2">Rotate x 180 deg</div>
    <div class="p3">Rotate y 180deg</div>
    <div class="p4">skew(20deg,0deg)</div>
    <div class="p5">scale(2,0.5)</div>
    <div class="p6">scalex(2)</div>
    <div class="p7">scaley(2)</div>
    <div class="p8">translate(50px,100px)</div>
</body></html>

```

Output:

Check mentioned effect according to given class on hovering

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) 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 rotate the image to 180 deg.
 - The transform: scaleX(-2) - double the mirror image horizontally.
 - The transform: scaleY(-2) - double the mirror image vertically.
 - **The transform: scaleY(-1) - mirror image vertically.**
- **Note:** The transform: scale(-1) property create a mirror image vertically.

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 hovering

Flip an Image

Move your mouse over the image.



After hovering

Flip an Image

Move your mouse over the image.



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

1. animation-name: keyframe name

(To get an animation to work, you must bind the animation to an element.)

Value	Description
Keyframe name	Specifies the name of the keyframe you want to bind to the selector

2. animation-delay: time;

The **animation-delay** property specifies a delay for the start of an animation. Negative values are also allowed. If using negative values, the animation will start as if it had already been playing for N seconds. (Delay will be one time at page load). (-ve value also allowed)

3. animation-direction: normal|reverse|alternate|alternate-reverse

normal	Default value. The animation is played as normal (forwards)
reverse	The animation is played in reverse direction (backwards)
alternate	The animation is played forwards first, then backward. Minimum 2 iteration count required to perform alternate animation.
alternate-reverse	The animation is played backwards first, then forwards. Minimum 2 iteration count required to perform alternate animation.

4. animation-duration: time

time: Specifies the length of time an animation should take to complete one cycle. This can be specified in seconds or milliseconds. Default value is 0, which means that no animation will occur.

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).

5. animation-iteration-count: number|infinite

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 forever.

number	A number that defines how many times an animation should be played. Default value is 1
infinite	Specifies that the animation should be played infinite times (for ever)

Shorthand

animation: name duration timing-function delay iteration-count direction fill-mode play-state;
Shorthand: animation: myfirst 5s 2s infinite alternate;

CSS @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

Syntax: @keyframes animationname {keyframes-selector {css-styles;}}

Value	Description
animationname	Required. Defines the name of the animation.
keyframes-selector	Required. Percentage of the animation duration. Legal values: 0-100% from (same as 0%) to (same as 100%) Note: You can have many keyframes-selectors in one animation.
css-styles	Required. One or more legal CSS style properties

Example:

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 "lightskyblue" to "blueviolet":

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)).

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: lightskyblue;
  animation-name: example;
  animation-duration: 4s;
  /* animation-delay: 2s;
  animation-iteration-count: 5;
  animation-direction: reverse; */
}

@keyframes example {
  from {background-color: lightskyblue;}
  to {background-color: blueviolet;}
}
</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:

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

Example on different Intervals:

```
<html>
<head>
<style>
div {
    width: 100px;
    height: 100px;
    background-color:lightblue;
    position: relative;
    animation-name: example;
    animation-duration: 4s;
}
@keyframes example {
    0% {background-color:lightblue; left:0px; top:0px;}
    25% {background-color:aqua; left:200px; top:0px;}
    50% {background-color:blue; left:200px; top:200px;}
    75% {background-color:aqua; left:0px; top:200px;}
    100% {background-color:lightblue; left:0px; top:0px;}
}
</style>
</head>
<body>
    <div></div>
</body>
</html>
```

Output:

It will rotate a div box clockwise and changes color simultaneously.

Opacity

The **opacity** property specifies the opacity/transparency of an element.

The opacity property can take a value from 0.0 - 1.0.

The lower the value, the more transparent

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.background {
    background-color: tomato;
    border: 2px solid black;
}
div.transbox {
    margin: 30px;
    background-color: #ffffff;
    border: 1px solid black;
    opacity: 0.6;
}
div.transbox p {
    margin: 5%;
    font-weight: bold;
    color: #000000;
}
</style></head>
<body>
<div class="background">
  <div class="transbox">
    <p>This is some text that is placed in the transparent box.</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:

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <style>
.p1{display:none;}
.p2{display:inline;color:tomato;}
.p3{display:block;color:tomato; }
.p4{display:inline-block;color:tomato;height:25px;}
</style></head>
<body>
<div> This is a none demo<p class="p1">THIS IS ME</p>"This is ME" hidden & No space
left</div>
<div> This is inline DEMO<p class="p2">THIS IS Inline </p>ME</div>
<div> This is block DEMO<p class="p3">THIS IS Block </p>ME</div>
<div> This is inline-block DEMO<p class="p4">THIS IS Inline-Block</p>ME</div>
<h2> Open Inspect and observe tomato color display block
</body>
```

Output:

This is a none demo"This is ME" hidden & No space left

This is inline DEMO THIS IS Inline ME

This is block DEMO

THIS IS Block

ME

This is inline-block DEMO THIS IS Inline-Block ME

Open Inspect and observe tomato color display block

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.

Let's understand the property values of the cursor.

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	It is used to indicate that something can be zoomed in.
Zoom-out	It is used to indicate 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; /* To avoid underline due to anchor tag */
    background-color:blue;
    border:none;
    color:white;
    padding:10px 3px;
    margin:7px 6px;
    text-align:center;
    display:inline-block;
    font-size:20px;
    cursor:pointer;
}
</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:



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
4. **column-rule-color:** Specifies the color of the rule between columns
5. **column-rule-width:** Specifies the width of the rule between columns

Shorthand: column-rule: width style color

example : column-rule: 1px dotted red

Example:

```
<head>
<style>
    .multi { column-gap:20px;column-rule-style:dashed;column-rule-color:pink;
            column-count: 3;}
</style>
</head>

<body>
    <div class="multi">
        Lorem data to be inserted*****
    </div>
</body>

</html>
```

Output:

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Placeat, dolorem autem amet, cum, molestias pariatur expedita dignissimos eum ratione quaerat accusamus tempore beatae non. Sapiente sint obcaecati nesciunt consequatur beatae?</p>	<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Facere sit eaque non, harum dicta officiis. Pariatur inventore omnis repellat, voluptas mollitia iure quasi id sequi voluptatem, culpa quisquam! Ratione, inventore. Lorem ipsum dolor sit</p>	<p>amet consectetur adipisicing elit. Magni earum voluptatum harum porro nam eaque! Sapiente in, qui ratione est neque distinctio ipsum? Illum numquam consequuntur, culpa quaerat voluptas ad!</p>
--	--	---

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.

Syntax:

Define variable globally in side :root{} class

```
:root{ --b: blue; }
```

Use of variable inside var() function

```
P{ color:var(--b) }
```

All the elements of the document can use the variable.

Define variable locally and use it

```
P{ --b: blue; color:var(--b) }
```

It has a scope only for the p element.

- A variable in CSS is defined by using the two dashes (--) at the beginning, followed by the name, which is case-sensitive.
- In the above syntax, the element indicates the selector that specifies the scope of the custom property. If we define the custom properties on the :root pseudo-class, then it will be globally applied to our HTML document. The names of the custom properties are case-sensitive.

Example:

```
<html>
<head><style>
:root      /*global variable*/
{ --col:blue;
--Bcol:lightgray;
}

h1 {
  --col:purple; /*local variable. local variable will get first priority over global*/
  color:var(--col);
}
```

```
P{
  color:var(--col);
  Background-color:var(--Bcol);
}
</style>
</head>
<body>
<h1>this is an example of local and globle variable</h1>
<p> The var() function in CSS is used to insert the custom property value. </p>
</body>
</html>
```

Output:

this is an example of local and globle variable

The var() function in CSS is used to insert the custom property value.

Pagination

Pagination is the process of dividing the document into pages and providing them with numbers. It 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

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

Output:

« 1 2 3 »

Active and Hover effect on 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;
    text-decoration:none; /*if not used none then it shows underline.*/
```

```

padding:20px; }
a.active {
    background-color: lightgreen;
    color: white; }
a:hover:not(.active) {background-color: lightblue;}

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

```

Output:

Note:

.pagination a:hover:not(.active) – This means hover pseudo class will not work on link with active class.

Example for

1. **Rounded Active & Hoverable Buttons:** Add the border-radius property if you want a rounded "active" and "hover" button.
2. **Bordered pagination:** Use the border property to add borders to the pagination.
3. **Pagination size:** Change the size of the pagination with the font-size property.

Example:

```

<html>
<head><style>
.pagination{
    display:inline-block; margin:15px;}
.pagination a {
    color:black;
    text-decoration:none; /*if not used none then it shows underline.*/
    padding:20px;
    border-radius: 10px; /*to create a rounded buttons*/
    border:1px solid gray; /*to apply border property to buttons*/

```



```
        font-size:12px;    /*to apply pagination size*/
    }
a.active {
    background-color: lightgreen;
    color: white; }
a:hover:not(.active) {background-color: lightblue;}
</style></head>
<body>
    <div class="pagination">
        <a href="#">&laquo</a>
        <a href="#">1</a>
        <a class="active" href="#">2</a>
        <a href="#">3</a>
        <a href="#">&raquo</a>
    </div>
</body></html>
```

Output: Before and After hover



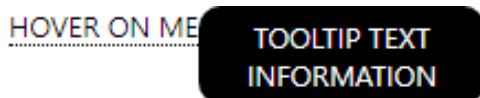
Tooltip

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

Example:

```
<html>
<head>
<style>
.tooltip{
    display:inline-block;
    border-bottom:1px dotted black;
}
.tooltiptxt {
    visibility:hidden; /* it leaves only empty space while content will be 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>
<div class="tooltip">
    HOVER ON ME
    <span class="tooltiptxt">TOOLTIP TEXT INFORMATION</span>
</div>
</body>
</html>
```

Output:



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.

Setting the viewport to make your website look good on all devices:

Must to Include

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Meta tag with all other content attribute values

```
<meta name="viewport" content="width=device-width,initial-scale=1.0,user-scalable=no,maximum-scale=0.5,minimum-scale=0.1"/>
```

minimum-scale[0.0-10.0] gives minimum size to allow to zoom-out.

Maximum-scale[0.0-10.0] gives maximum size to allow to zoom-in

Setting the Viewport

The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.

This gives the browser instructions on how to control the page's dimensions and scaling

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 (mediafeature) and|or (mediafeature)
{
CSS-Code;
}
```

Examples

```
@media only screen and (min-width:500px) and (max-width:700px)
```

```
@media(min-width:500px)
```

```
@media not (orientation:landscape)
```

```
@media screen and (max-width:500px)
```

not, only, and keywords:

- ✓ **not:** The not keyword inverts the meaning of an entire media query.
- ✓ **only:** The only keyword prevents older browsers that do not support media queries with media features from applying the specified styles. It has no effect on modern browsers.
- ✓ **and:** The and keyword combines a media feature with a media type or other media features.

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 screen readers that "reads" the page out loud

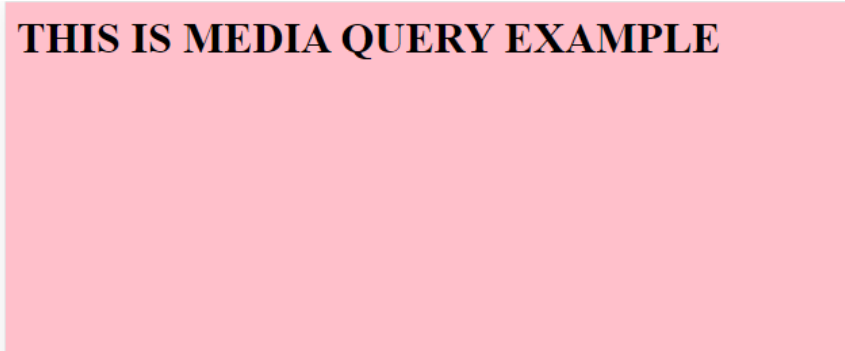
Example:

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
<style>
    body{
        background-color:lightgreen;
    }
    @media only screen and (orientation:landscape){
        body{background-color:pink;}
    }
</style>
</head>
<body>
    <h1>THIS IS MEDIA QUERY EXAMPLE</h1>
</body>
</html>
```

Output:

Open Inspect – Toggle device toolbar---select dimensions responsive – resize window to see the effect

Landscape mode



THIS IS MEDIA QUERY EXAMPLE

Portrait mode



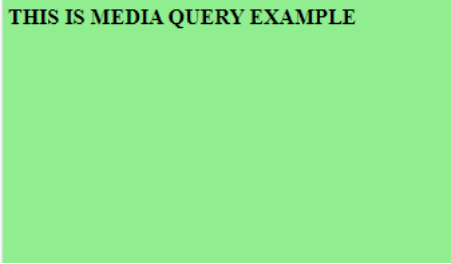
**THIS IS MEDIA QUERY
EXAMPLE**

Example 2(max-width)

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0 "/>
<style>
body
{
background-color:lightgreen;
}
@media only screen and (max-width:412px)
{
body{ background-color:lightblue;}
```


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

Output:



THIS IS MEDIA QUERY EXAMPLE

(After minimization)



THIS IS MEDIA QUERY
EXAMPLE