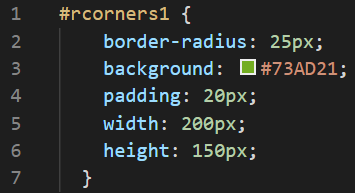
# CSS Rounded Corners

With the CSS border-radius property, you can give any element "rounded corners".

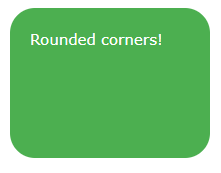
## CSS border-radius Property

The CSS border-radius property defines the radius of an element's corners.

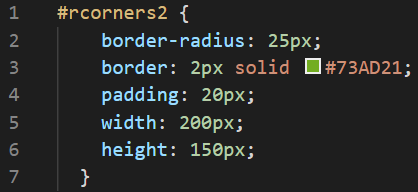
1. Consider the following example with rounded corners for an element with a specified background color:



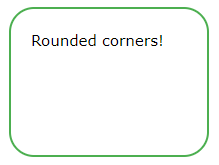
The output for the above code is as follows:



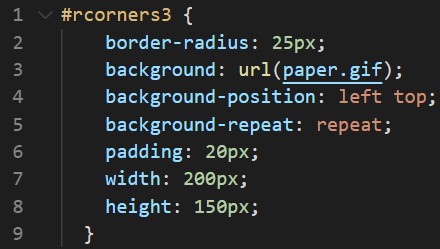
1. Consider the following example with rounded corners for an element with a border:



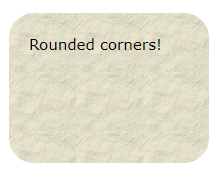
The output for the above code is as follows:



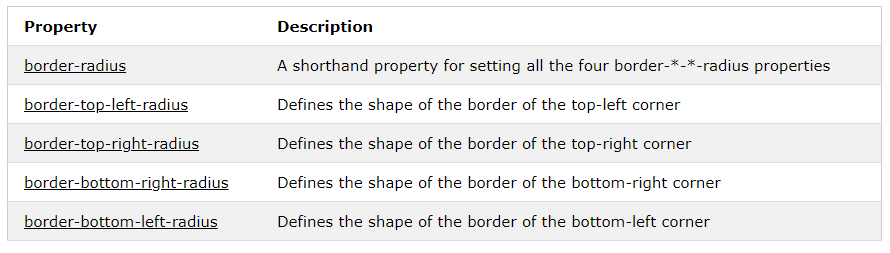
1. Consider the following example with rounded corners for an element with a background image:



The output for the above code is as follows:



## CSS Rounded Corners Properties



# CSS Border Images

With the CSS border-image property, you can set an image to be used as the border around an element.

## CSS border-image Property

The CSS border-image property allows you to specify an image to be used instead of the normal border around an element.

The property has three parts:

1. The image to use as the border
2. Where to slice the image
3. Define whether the middle sections should be repeated or stretched

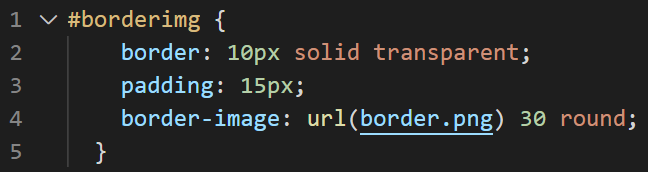
We will use the following image (called "border.png"):



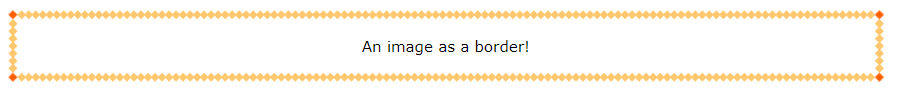
The border-image property takes the image and slices it into nine sections, like a tic-tac-toe board. It then places the corners at the corners, and the middle sections are repeated or stretched as you specify.

Note: For border-image to work, the element also needs the border property set!

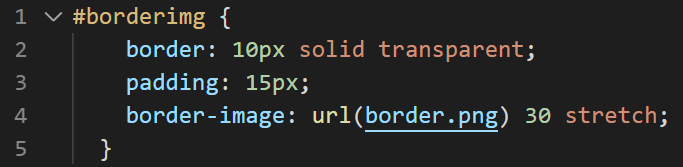
1. Consider the following example where the middle sections of the image are repeated to create the border:



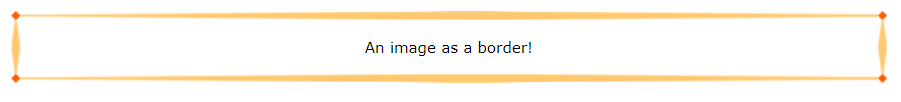
The output for the above is as follows:



1. Consider the following example where the middle sections of the image are stretched to create the border:



The output of the above code is as follows:

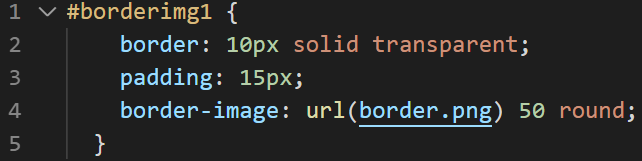


## CSS border-image - Different Slice Values

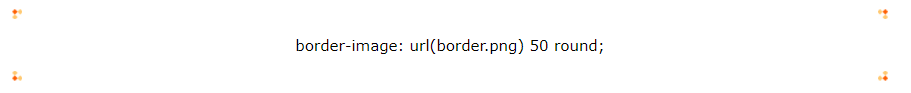
Different slice values completely changes the look of the border:

Consider the following examples for the better understanding:

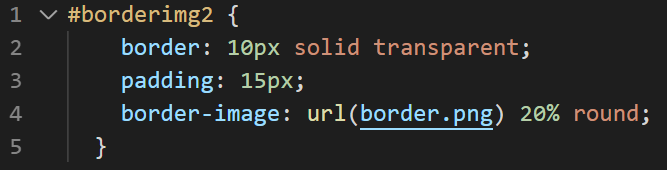
Example 1:



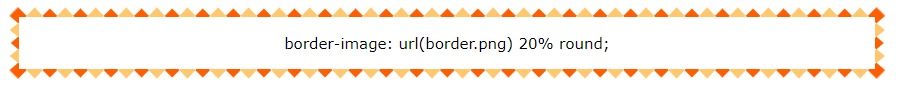
The output of the above code is as follows:



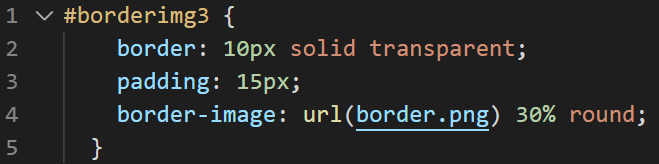
Example 2:



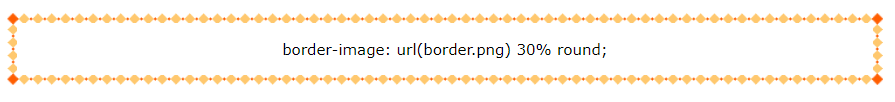
The output of the above code is as follows:



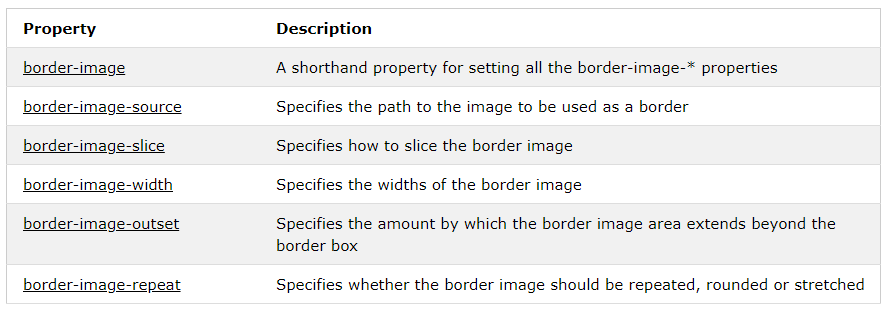
Example 3:



The output of the above code is as follows:



## CSS Border Image Properties

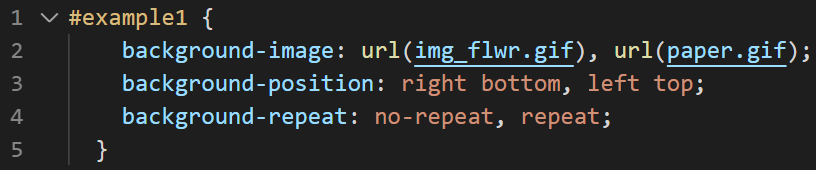


# CSS Multiple Backgrounds

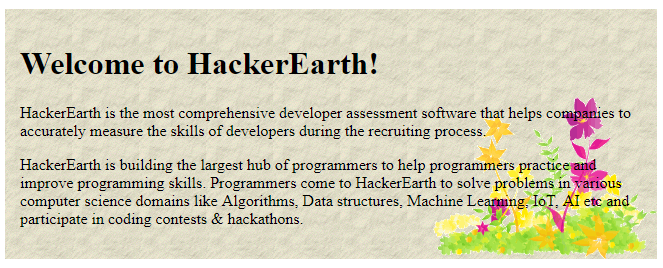
CSS allows you to add multiple background images for an element, through the background-image property.

The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.

The following example has two background images, the first image is a flower (aligned to the bottom and right) and the second image is a paper background (aligned to the top-left corner):



The output of the above code is as follows:

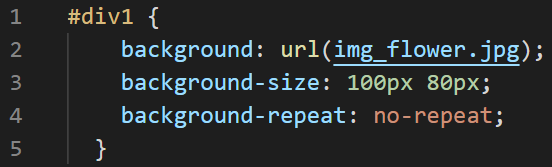


## CSS Background Size

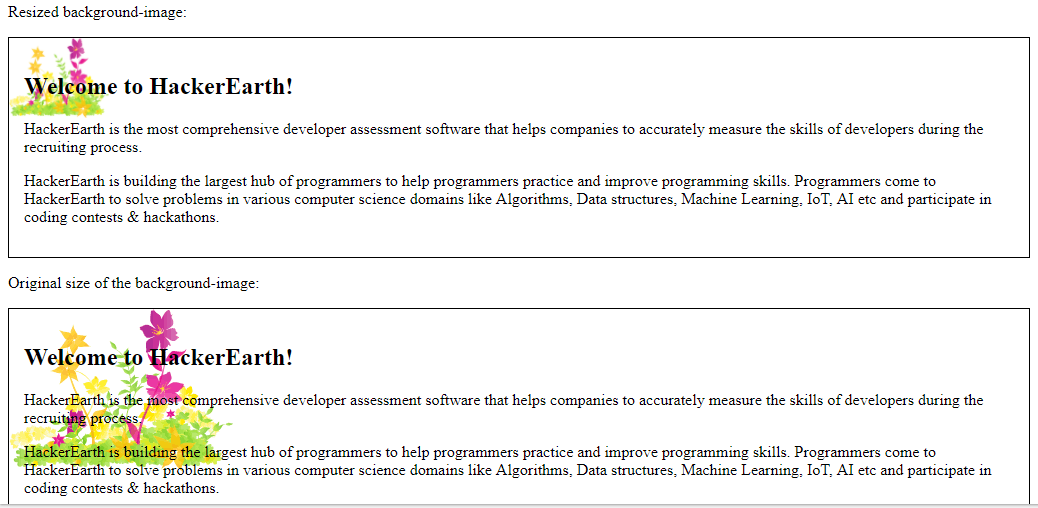
The CSS background-size property allows you to specify the size of background images.

The size can be specified in lengths, percentages, or by using one of the two keywords: contain or cover.

The following example resizes a background image to much smaller than the original image (using pixels):



The output of the above code is as follows:

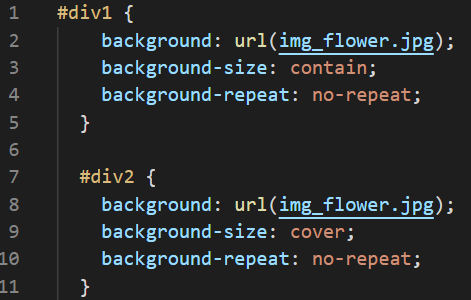


The two other possible values for background-size are contain and cover.

The contain keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area). As such, depending on the proportions of the background image and the background positioning area, there may be some areas of the background which are not covered by the background image.

The cover keyword scales the background image so that the content area is completely covered by the background image (both its width and height are equal to or exceed the content area). As such, some parts of the background image may not be visible in the background positioning area.

The following example illustrates the use of contain and cover:



The output of the above code is as follows:



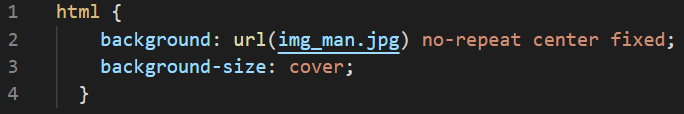
## Full Size Background Image

Now we want to have a background image on a website that covers the entire browser window at all times.

The requirements are as follows:

1. Fill the entire page with the image (no white space)
2. Scale image as needed
3. Center image on page
4. Do not cause scrollbars

The following example shows how to do it; Use the <html> element (the <html> element is always at least the height of the browser window). Then set a fixed and centered background on it. Then adjust its size with the background-size property:



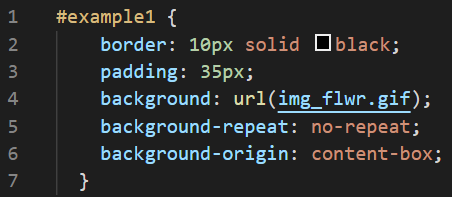
## CSS background-origin Property

The CSS background-origin property specifies where the background image is positioned.

The property takes three different values:

1. border-box - the background image starts from the upper left corner of the border
2. padding-box - (default) the background image starts from the upper left corner of the padding edge
3. content-box - the background image starts from the upper left corner of the content

The following example illustrates the background-origin property:



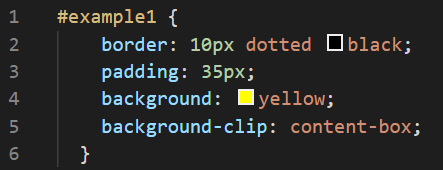
## CSS background-clip Property

The CSS background-clip property specifies the painting area of the background.

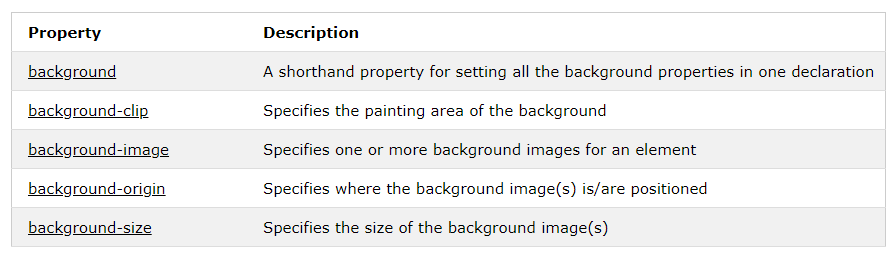
The property takes three different values:

1. border-box - (default) the background is painted to the outside edge of the border
2. padding-box - the background is painted to the outside edge of the padding
3. content-box - the background is painted within the content box

The following example illustrates the background-clip property:



## CSS Advanced Background Properties



# CSS Colors

## RGBA Colors

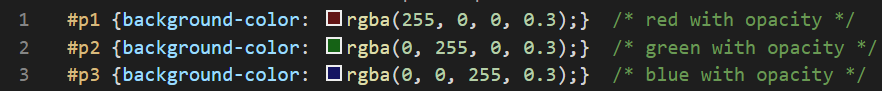
RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with: rgba(red, green, blue, alpha). The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Here we can see the different opacity levels with the color RED:



The following example defines different RGBA colors:



The output of the above code is as follows:



## HSL Colors

HSL stands for Hue, Saturation and Lightness.

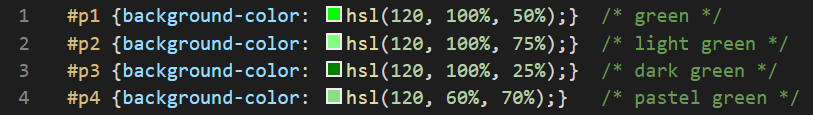
An HSL color value is specified with: hsl(hue, saturation, lightness).

1. Hue is a degree on the color wheel (from 0 to 360):
   * 0 (or 360) is red
   * 120 is green
   * 240 is blue
2. Saturation is a percentage value: 100% is the full color.
3. Lightness is also a percentage; 0% is dark (black) and 100% is white.

Here we can see the different opacity levels with the color RED:



The following example defines different HSL colors:



The output of the above code is as follows:

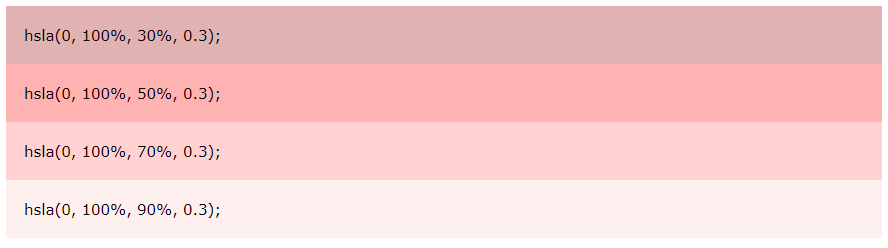


## HSLA Colors

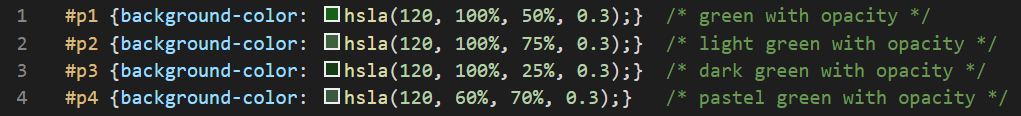
HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with: hsla(hue, saturation, lightness, alpha), where the alpha parameter defines the opacity. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

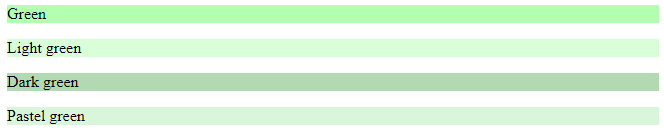
Here we can see the different opacity levels with the color RED:



The following example defines different HSLA colors:



The output of the following code is as follows:



# CSS Gradients

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines two types of gradients:

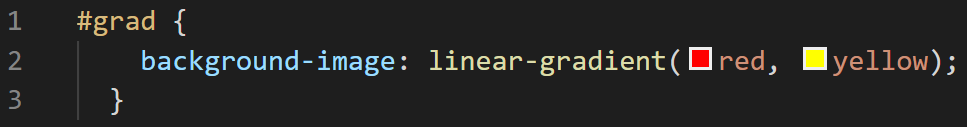
1. Linear Gradients (goes down/up/left/right/diagonally)
2. Radial Gradients (defined by their center)

## 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)

The following example shows a linear gradient that starts at the top. It starts red, transitioning to yellow:

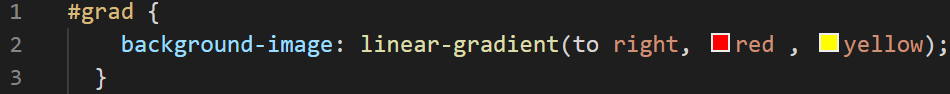


The output of the above code is as follows:



Direction - Left to Right

The following example shows a linear gradient that starts from the left. It starts red, transitioning to yellow:



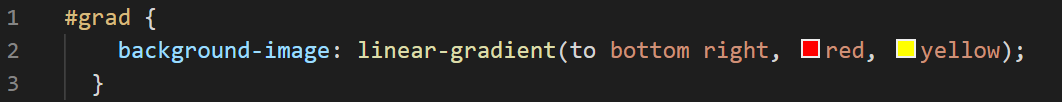
The output of the above code is as follows:



Direction - 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:

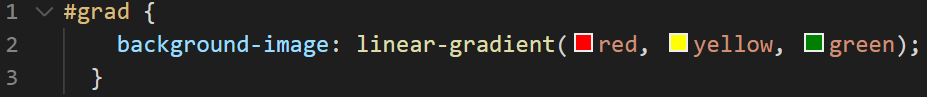


The output of the following code is as follows:



## Using Multiple Color Stops

The following example shows a linear gradient (from top to bottom) with multiple color stops:



The output of the above code is as follows:



## CSS Radial Gradients

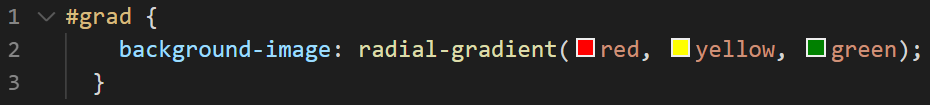
A radial gradient is defined by its center.

To create a radial gradient you must also define at least two color stops.

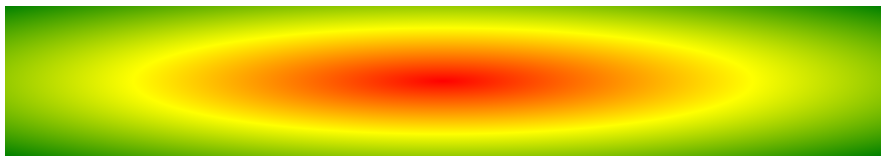
By default, shape is ellipse, size is farthest-corner, and position is center.

Radial Gradient - Evenly Spaced Color Stops (this is default)

The following example shows a radial gradient with evenly spaced color stops



The output of the above code is as follows:



## CSS Shadow Effects

With CSS you can add shadow to text and to elements.

In these chapters you will learn about the following properties:

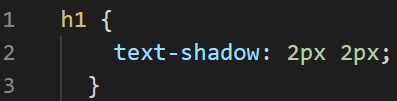
1. text-shadow
2. Box-shadow

CSS Text Shadow

The CSS text-shadow property applies shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px)

Example,



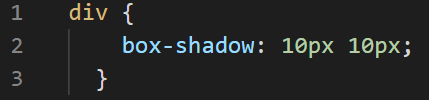
The output of the above code is as follows:



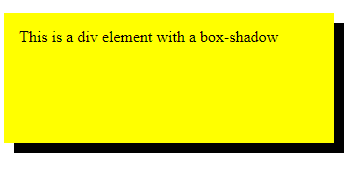
## CSS box-shadow Property

The CSS box-shadow property applies shadow to elements.

In its simplest use, you only specify the horizontal shadow and the vertical shadow:



The output of the above code is as follows:



# CSS Text Effects

The following are the properties of CSS Text Effects

1. text-overflow
2. word-wrap
3. word-break
4. writing-mode

CSS Text Overflow

The CSS text-overflow property specifies how overflowed content that is not displayed should be signaled to the user.

## CSS Word Wrapping

The CSS word-wrap property allows long words to be able to be broken and wrap onto the next line.

If a word is too long to fit within an area, it expands outside.

## CSS Word Breaking

The CSS word-break property specifies line breaking rules.

## CSS Writing Mode

The CSS writing-mode property specifies whether lines of text are laid out horizontally or vertically.

## CSS 2D Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

## CSS 2D Transforms Methods

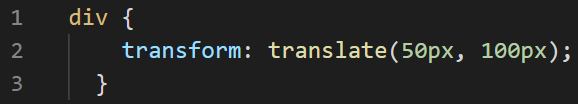
With the CSS transform property you can use the following 2D transformation methods:

* translate()
* rotate()
* scaleX()
* scaleY()
* scale()
* skewX()
* skewY()
* skew()
* matrix()

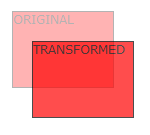
## The translate() Method

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

The following example moves the <div> element 50 pixels to the right, and 100 pixels down from its current position:



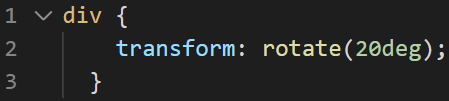
The output of the above code is as follows:



## The rotate() Method

The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

The following example rotates the <div> element clockwise with 20 degrees:



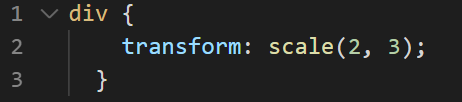
The output of the above code is as follows:



## The scale() Method

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:

****

The output of the above code is as follows:



## The scaleX() Method

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

## The scaleY() Method

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

## The skewX() Method

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

## The skewY() Method

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

## The matrix() Method

The matrix() method combines all the 2D transform methods into one.

The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.

## CSS 3D Transforms

CSS also supports 3D transformations.

## CSS 3D Transforms Methods

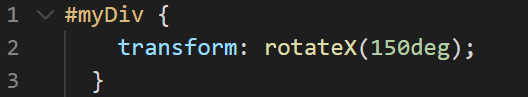
With the CSS transform property you can use the following 3D transformation methods:

1. rotateX()
2. rotateY()
3. rotateZ()

## The rotateX() Method

The rotateX() method rotates an element around its X-axis at a given degree

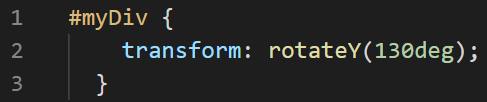
Example:



## The rotateY() Method

The rotateY() method rotates an element around its Y-axis at a given degree.

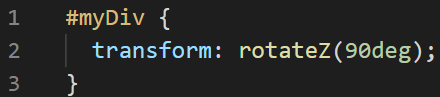
Example:



## The rotateZ() Method

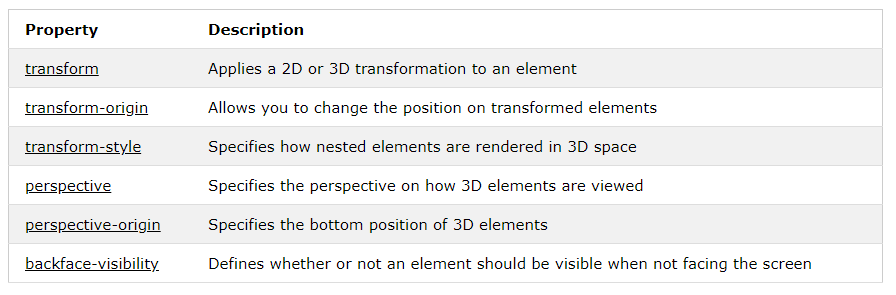
The rotateZ() method rotates an element around its Z-axis at a given degree.

Example:



## CSS Transform Properties

The following table lists all the 3D transform properties.



-------------------------------------------------End of Doc-----------------------------------------------------------

------------------------Reference Credits: W3SCHOOLS,Tutorialspoint---------------------------