**Colors Element**

**rgb()**

The rgb() functional notation expresses a color according to its red, green, and blue components. An optional alpha component represents the color's transparency.

**Syntax**

rgb(255 255 255)

rgb(255 255 255 / .5)

RGB Value

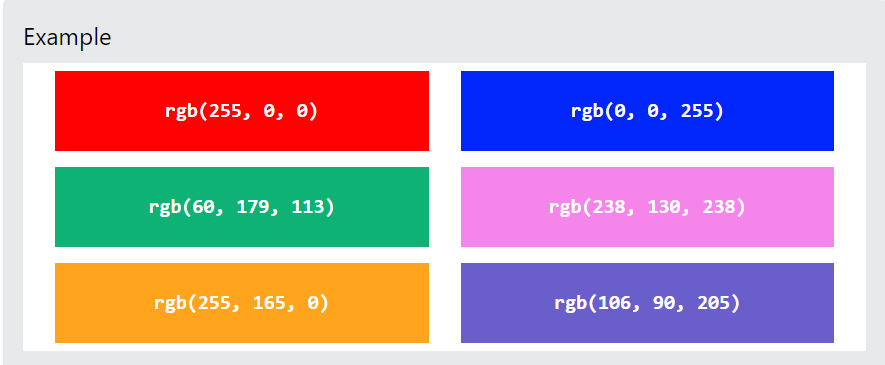
In CSS, a color can be specified as an RGB value, using this formula:

**rgb(*red,* *green*, *blue*)**

Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.

1. To display black, set all color parameters to 0, like this: rgb(0, 0, 0).
2. To display white, set all color parameters to 255, like this: rgb(255, 255, 255).



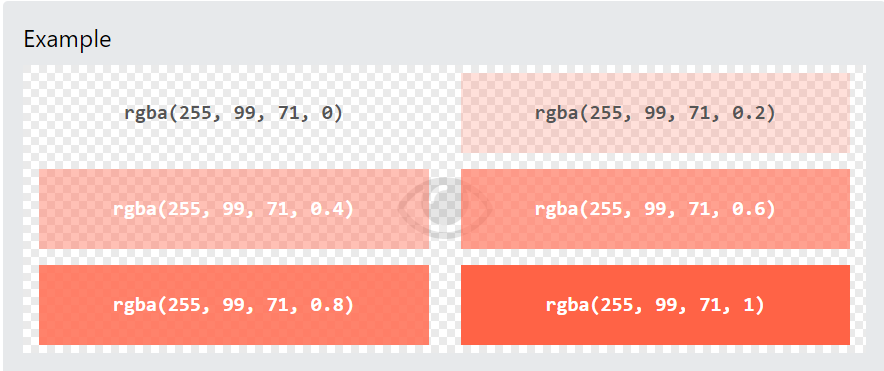
## RGBA Value

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 (not transparent at all):



HSL:

The hsl() functional notation expresses an sRGB color according to its hue, saturation, and lightness components. An optional alpha component represents the color's transparency.

Values

Functional notation: hsl(H S L[ / A])

1. H - A <number> or an <angle> representing the hue angle. More details on this type can be found on the <hue> reference.
2. S - A <percentage> representing saturation, where 100% is completely saturated, while 0% is completely unsaturated (gray).
3. L - A <percentage> representing lightness, where 100% is white, 0% is black, and 50% is "normal".
4. A Optional - An <alpha-value>, where the number 1 corresponds to 100% (full opacity).

For reference: <https://www.smashingmagazine.com/2021/07/hsl-colors-css/>

**HSLA**:

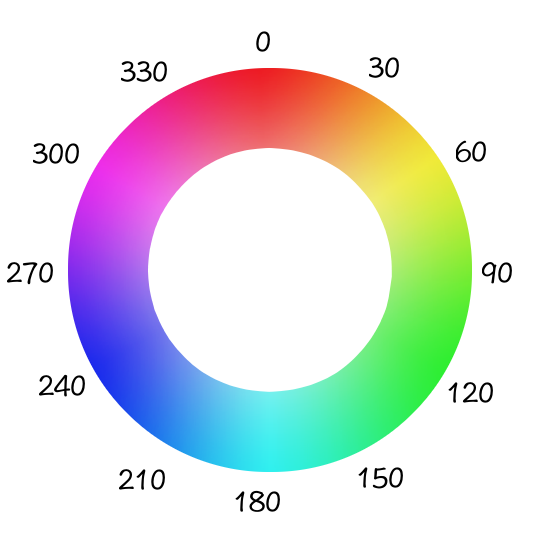
he hsla() function is based on the HSL color model. HSL (which stands for Hue Saturation Lightness) is a hue-based representation of the RGB color space of computer graphics.

The hsla() function accepts the HSLA value as a parameter. The HSLA value is provided as a comma-separated list of four values — the three HSL values (which provide the hue, saturation, and light components respectively), and a fourth value, which provides the alpha channel.

Example:

hsla(30, 100%, 50%, 0.5);

The first of the four values — the hue component — represents an angle of the color circle.

[](https://www.quackit.com/pix/stock/color_wheel_hsl.png)

You can specify the value as an angle in degrees (e.g. 180deg) or simply as a number (e.g. 180). For example, if you look at the color circle, blue is at 240 degrees, so it could be written as either 240deg or 240.

The second value is expressed as a percentage. It represents the amount of saturation in the color. For example, 100% is fully saturated (more colorful and intense), while 0 is a fully-unsaturated gray.

The third value is also expressed as a percentage. It represents the amount of light in the color. For lightness, 50% is the "normal" setting, while 100% is white and 0% is black.

The fourth value is the alpha value. It determines how transparent the color is. A value of 1 is fully opaque, while a value of 0 is fully transparent. A value of 0.5 is semi-transparent.

**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)**

**Linear Gradient:**

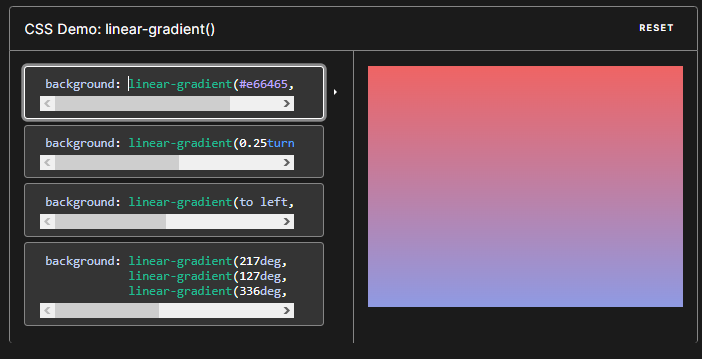
[**https://cssgradient.io**](https://cssgradient.io) **(Color Picker)**

The linear-gradient() CSS function creates an image consisting of a progressive transition between two or more colors along a straight line. Its result is an object of the <gradient> data type, which is a special kind of <image>.

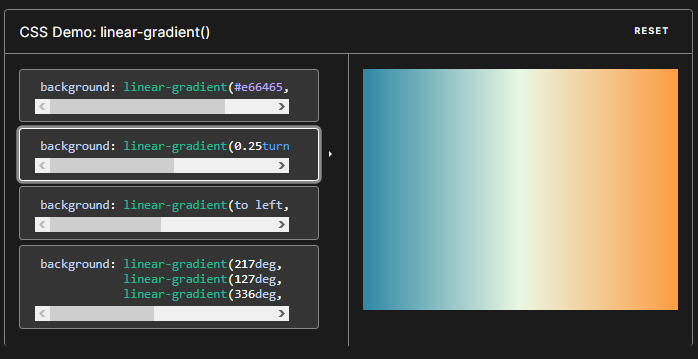
To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax:

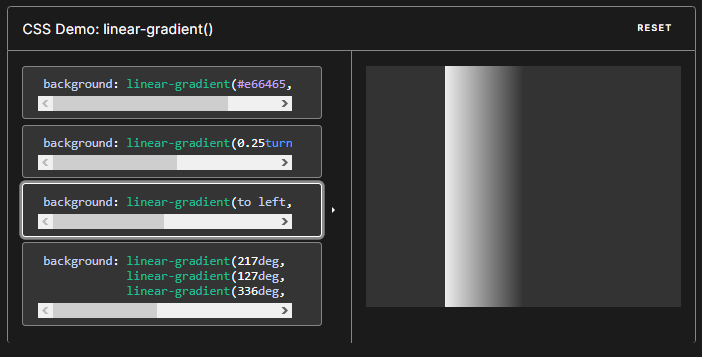
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);



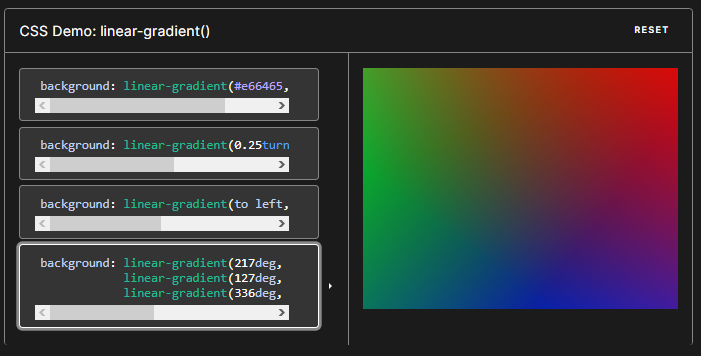
background: linear-gradient(#e66465, #9198e5);



background: linear-gradient(0.25turn, #3f87a6, #ebf8e1, #f69d3c);



background: linear-gradient(to left, #333, #333 50%, #eee 75%, #333 75%);



background: linear-gradient(217deg, rgba(255,0,0,.8), rgba(255,0,0,0) 70.71%), linear-gradient(127deg, rgba(0,255,0,.8), rgba(0,255,0,0) 70.71%), linear-gradient(336deg, rgba(0,0,255,.8), rgba(0,0,255,0) 70.71%);

Syntax

/\* A gradient tilted 45 degrees,

starting blue and finishing red \*/

linear-gradient(45deg, blue, red);

/\* A gradient going from the bottom right to the top left corner,

starting blue and finishing red \*/

linear-gradient(to left top, blue, red);

/\* Color stop: A gradient going from the bottom to top,

starting blue, turning green at 40% of its length,

and finishing red \*/

linear-gradient(0deg, blue, green 40%, red);

/\* Color hint: A gradient going from the left to right,

starting red, getting to the midpoint color

10% of the way across the length of the gradient,

taking the rest of the 90% of the length to change to blue \*/

linear-gradient(.25turn, red, 10%, blue);

/\* Multi-position color stop: A gradient tilted 45 degrees,

with a red bottom-left half and a blue top-right half,

with a hard line where the gradient changes from red to blue \*/

linear-gradient(45deg, red 0 50%, blue 50% 100%);

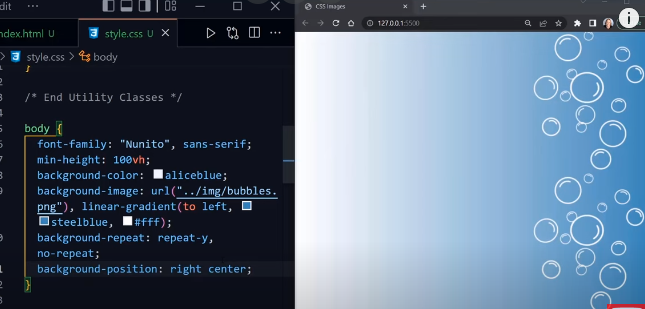
Customizing Gradients

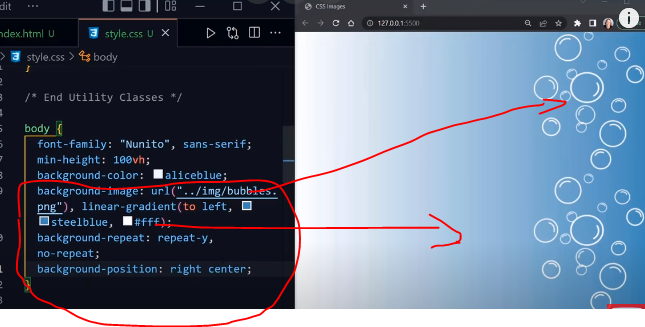
By adding more color-stop points on the gradient line, you can create a highly customized transition between multiple colors. A color-stop's position can be explicitly defined by using a <length> or a <percentage>. If you don't specify the location of a color, it is placed halfway between the one that precedes it and the one that follows it. The following two gradients are equivalent.

linear-gradient(red, orange, yellow, green, blue);

linear-gradient(red 0%, orange 25%, yellow 50%, green 75%, blue 100%);

**Linear-gradient and background image on top:**





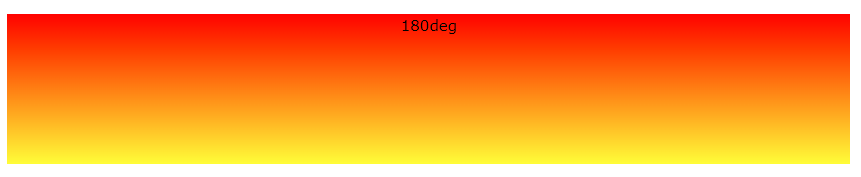
Example of linear-gradient:

#grad {  
  background-image: linear-gradient(to right, red , yellow);  
}

#grad {  
  background-image: linear-gradient(to bottom right, red, yellow);  
}

## Using Angles

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



## Using Multiple Color Stops

## #grad {   background-image: linear-gradient(red, yellow, green); }



#grad {  
  background-image: linear-gradient(to right, red,orange,yellow,green,blue,indigo,violet);  
}

## Using Transparency

CSS gradients also support transparency, which can be used to create fading effects.

To add transparency, we use the rgba() function to define the color stops. The last parameter in the rgba() function can be a value from 0 to 1, and it defines the transparency of the color: 0 indicates full transparency, 1 indicates full color (no transparency).

The following example shows a linear gradient that starts from the left. It starts fully transparent, transitioning to full color red:

#grad {  
  background-image: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));  
}



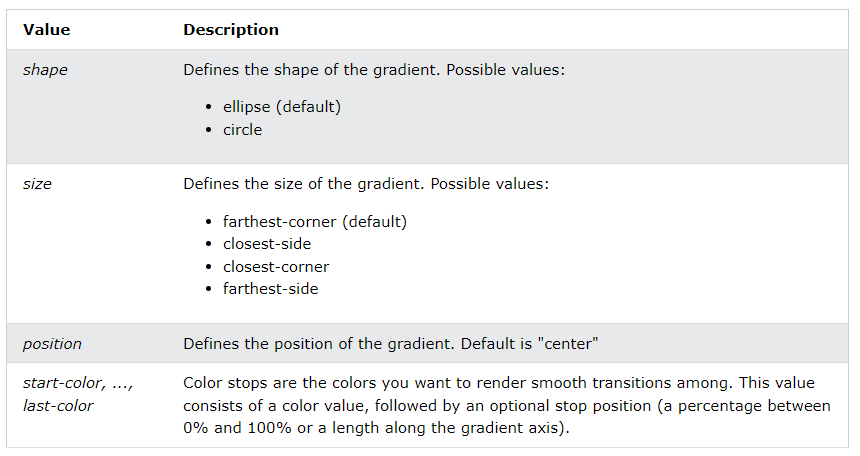
**Radial Gradient:**

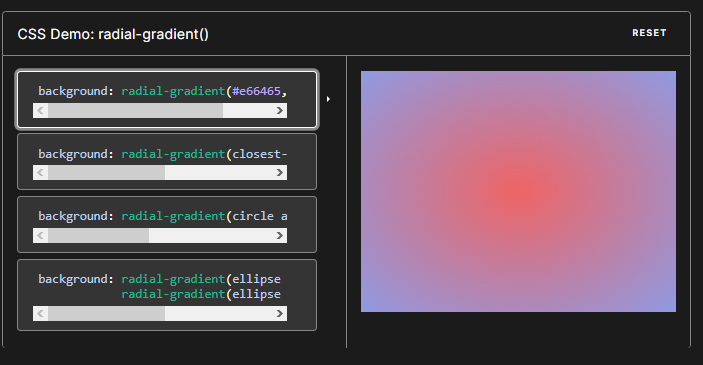
**radial-gradient()**

The radial-gradient() CSS function creates an image consisting of a progressive transition between two or more colors that radiate from an origin. Its shape may be a circle or an ellipse. The function's result is an object of the <gradient> data type, which is a special kind of <image>.

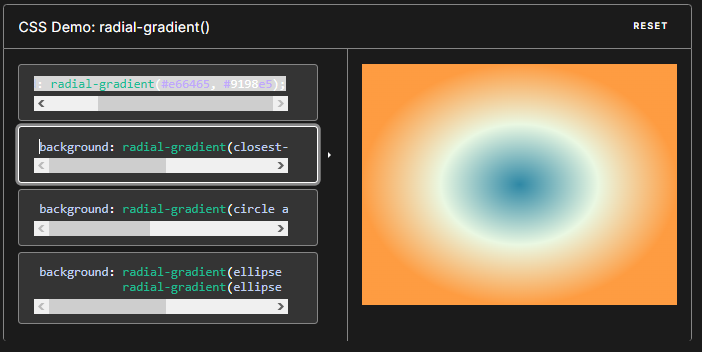
Syntax:

background-image: radial-gradient(shape size at position, start-color, ..., last-color);

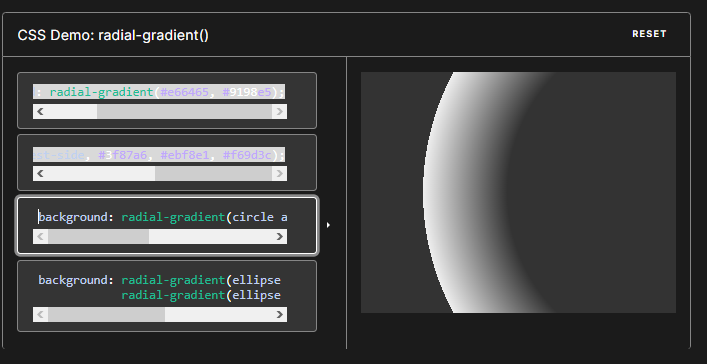




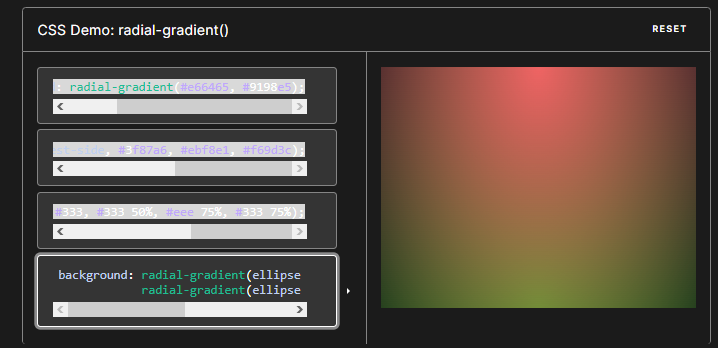
background: radial-gradient(#e66465, #9198e5);



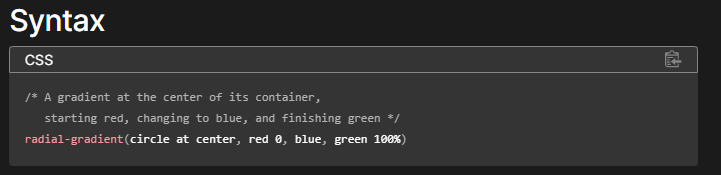
background: radial-gradient(closest-side, #3f87a6, #ebf8e1, #f69d3c);



background: radial-gradient(circle at 100%, #333, #333 50%, #eee 75%, #333 75%);



background: radial-gradient(ellipse at top, #e66465, transparent), radial-gradient(ellipse at bottom, #4d9f0c, transparent);



A radial gradient is specified by indicating the center of the gradient (where the 0% ellipse will be) and the size and shape of the ending shape (the 100% ellipse).

Values

<position>

The position of the gradient, interpreted in the same way as background-position or transform-origin. If unspecified, it defaults to center.

<ending-shape>

The gradient's ending-shape. The value can be circle (meaning that the gradient's shape is a circle with a constant radius) or ellipse (meaning that the shape is an axis-aligned ellipse). If unspecified, it defaults to ellipse.

<size>

Determines the size of the gradient's ending shape. If omitted it defaults to farthest-corner. It can be given explicitly or by keyword. For the purpose of the keyword definitions, consider the gradient box edges as extending infinitely in both directions, rather than being finite line segments.

Both circle and ellipse gradients accept the following keywords for their <size>:

Keyword Description

closest-side The gradient's ending shape meets the side of the box closest to its center (for circles) or meets both the vertical and horizontal sides closest to the center (for ellipses).

closest-corner The gradient's ending shape is sized so that it exactly meets the closest corner of the box from its center.

farthest-side Similar to closest-side, except the ending shape is sized to meet the side of the box farthest from its center (or vertical and horizontal sides).

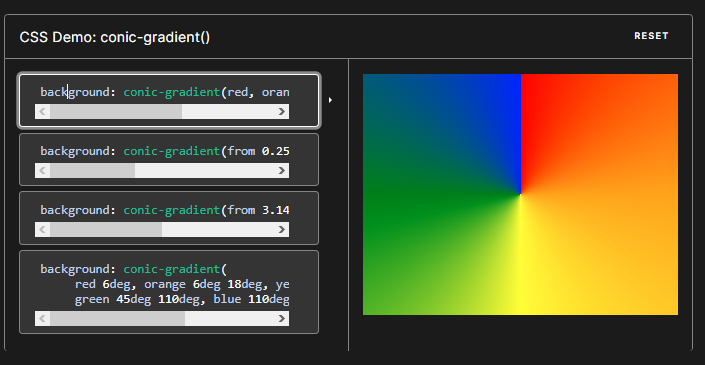
farthest-corner The default value, the gradient's ending shape is sized so that it exactly meets the farthest corner of the box from its center.

**Conic Gradients**

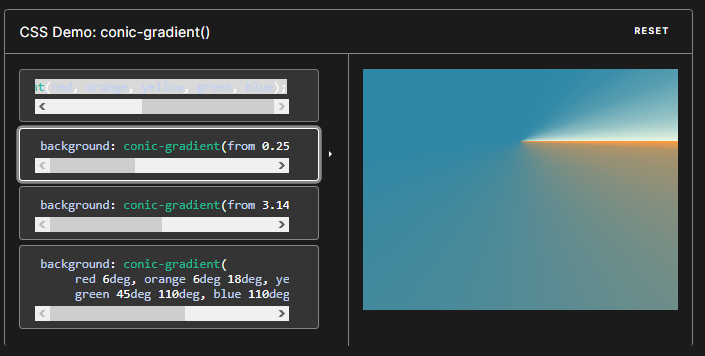
The conic-gradient() CSS function creates an image consisting of a gradient with color transitions rotated around a center point (rather than radiating from the center). Example conic gradients include pie charts and color wheels. The result of the conic-gradient() function is an object of the <gradient> data type, which is a special kind of <image>.

Syntax:

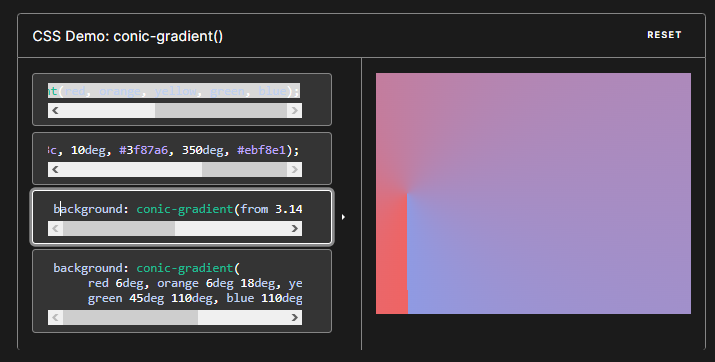
background-image: conic-gradient([from angle] [at position,] color [degree], color [degree], ...);



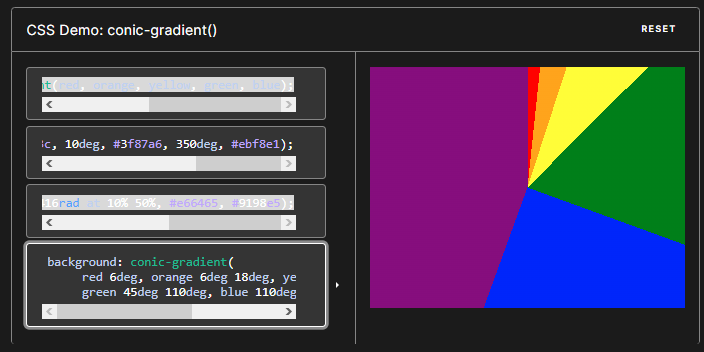
background: conic-gradient(red, orange, yellow, green, blue);



background: conic-gradient(from 0.25turn at 50% 30%, #f69d3c, 10deg, #3f87a6, 350deg, #ebf8e1);



background: conic-gradient(from 3.1416rad at 10% 50%, #e66465, #9198e5);

 background: conic-gradient( red 6deg, orange 6deg 18deg, yellow 18deg 45deg, green 45deg 110deg, blue 110deg 200deg, purple 200deg);