WSA | Forward looking IT finishing school

Effects

Cascading Style Sheets (CSS3)

Table of Content

- Gradients
- Shadows
- Transforms
- Transitions
- Animations

Gradients

(Cascading Style Sheets 3)

Gradients

- Gradients let you display smooth transitions between two or more specified colors
- There are two types of gradients
  - Linear Gradients
  - Radial Gradients

Gradients
(Linear)

- Linear gradient is created with linear-gradient() function
- To create a linear gradient you must pass at least two color stops to linear-gradient function
- Color stops are the colors among which transition to be applied
- You can also set a starting point and a direction (down | up | left | right | or an angle) along with the gradient effect
- By default, linear gradients run from top to bottom

Gradients
(Linear - Syntax)

Syntax :

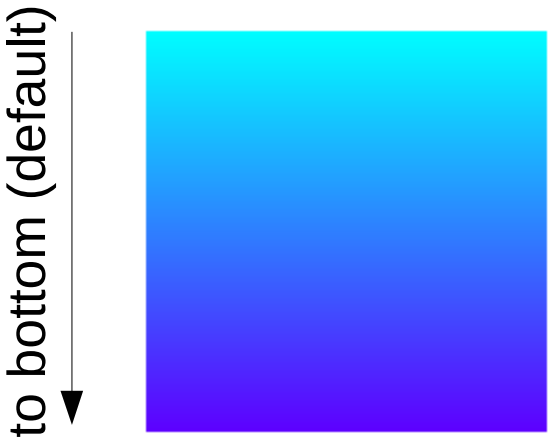
background: linear-gradient(direction, color-stop1, color-stop2, ...);

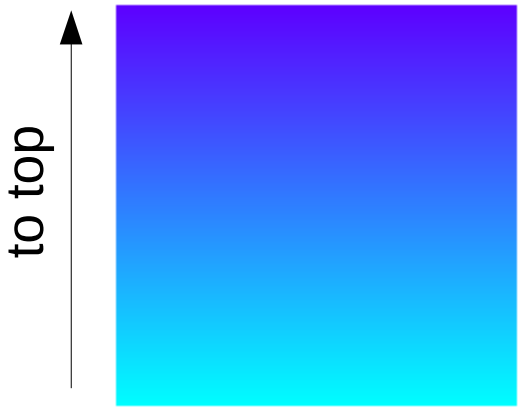
Example :

background: linear-gradient(cyan, blue);

background: linear-gradient(to left, red, green);

Gradients
(Linear - Direction)



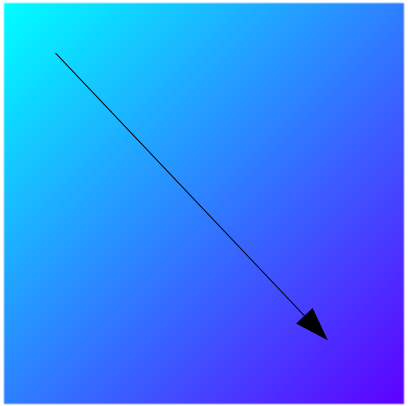


Gradients
(Linear - Direction)

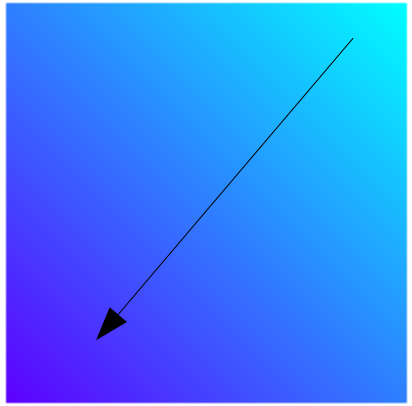
to left

to right

Gradients
(Linear – Direction - Diagonal)

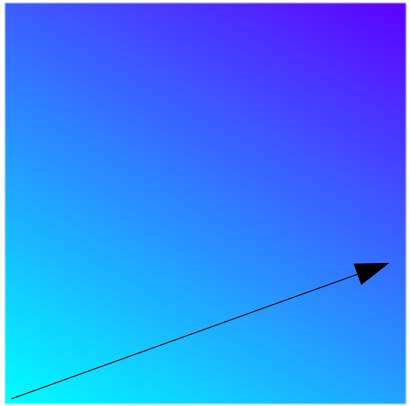


Diagonal
(to right bottom)



Diagonal
(to left bottom)

Gradients
(Linear – Direction - Angled)



Angled
(30deg)

Angled
(180deg)

Angled
(0deg)

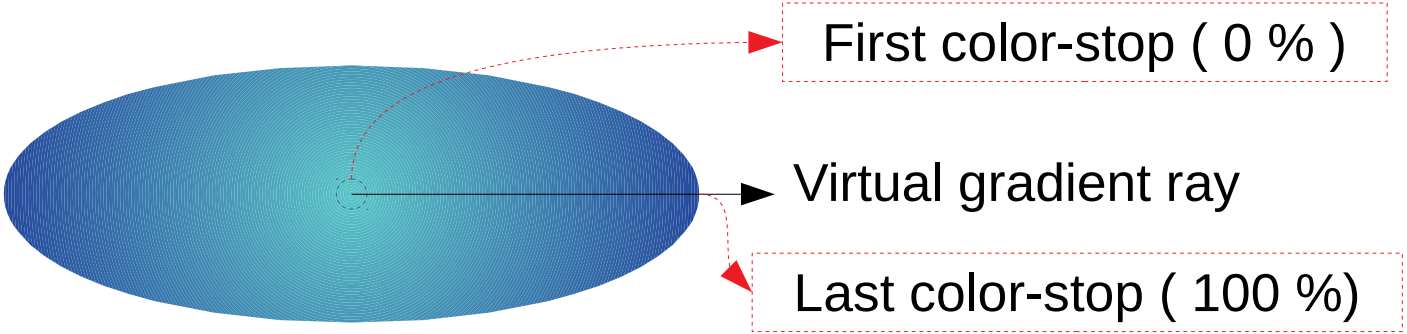
Gradients
(Linear - Multi-color)

Rainbow

Tri-color

Gradients
(Radial)

- Radial gradient is created with radial-gradient() function
- It is defined by radial center



Gradients
(Radial - Syntax)

Syntax :

background: radial-gradient(color-stop-1, ... , color-stop-n);

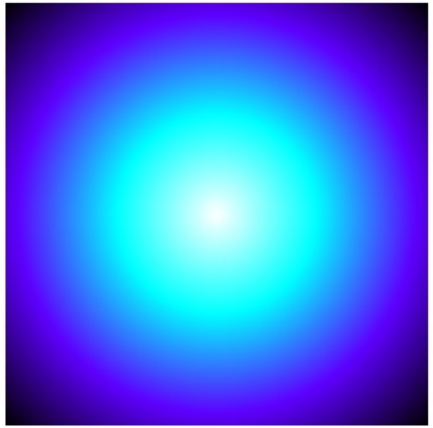
or

background: radial-gradient(shape size at position, color-stop-1, ... , color-stop-n);

Gradients
(Radial - Example)

Example : */* default position center center */*

background: radial-gradient(white, aqua, blue, black);



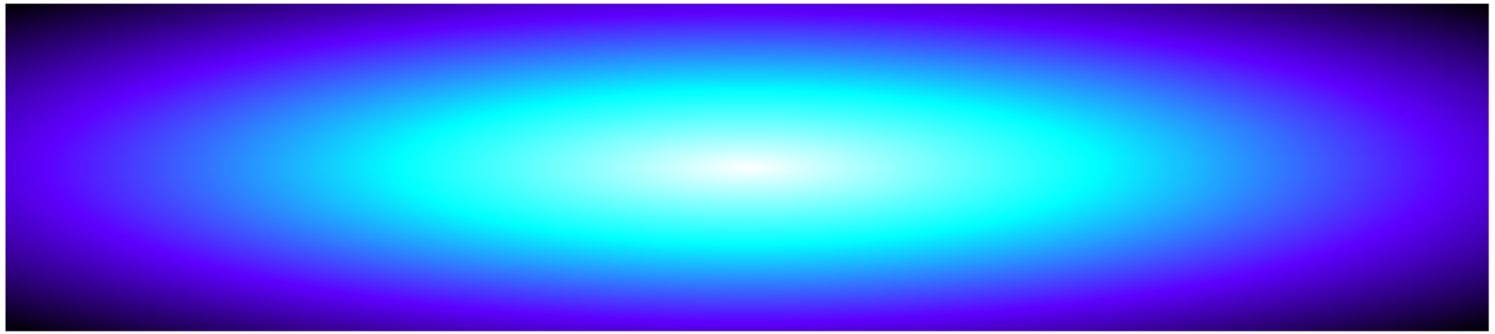
Gradients
(Radial - shape)

- The value can be
  - circle (shape is a circle with constant radius)
  - ellipse (shape is an axis-aligned ellipse)
- If unspecified, it defaults to ellipse

Example :

```
background: radial-gradient(circle, white, aqua, blue, black);
```

Gradients
(Radial - shape)



ellipse

circle

Gradients
(Radial - position)

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

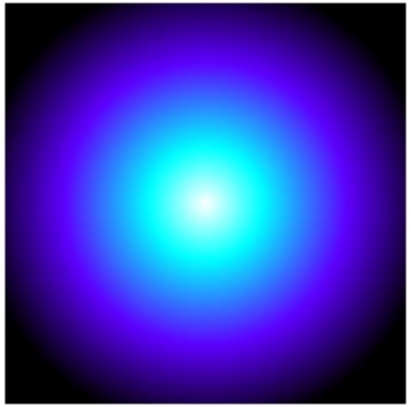
Example 1 :

```
background: radial-gradient(white, aqua 20%, blue 50%, black 80%);
```

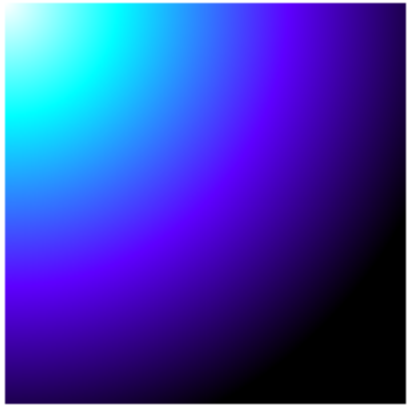
Gradients
(Radial - position)

Example 2 :

```
background: radial-gradient(circle at top left, white, aqua 20%, blue 50%, black 80%);
```



Example 1

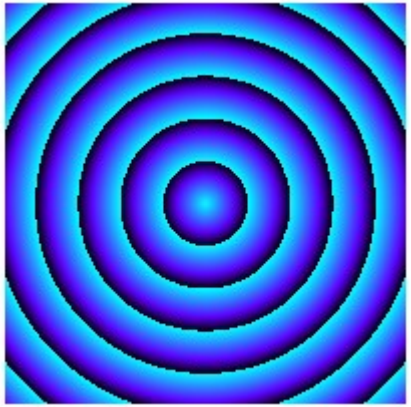


Example 2

Gradients
(Radial - repeat)

Example :

background: repeating-radial-gradient(aqua, blue 10%, black 15%);

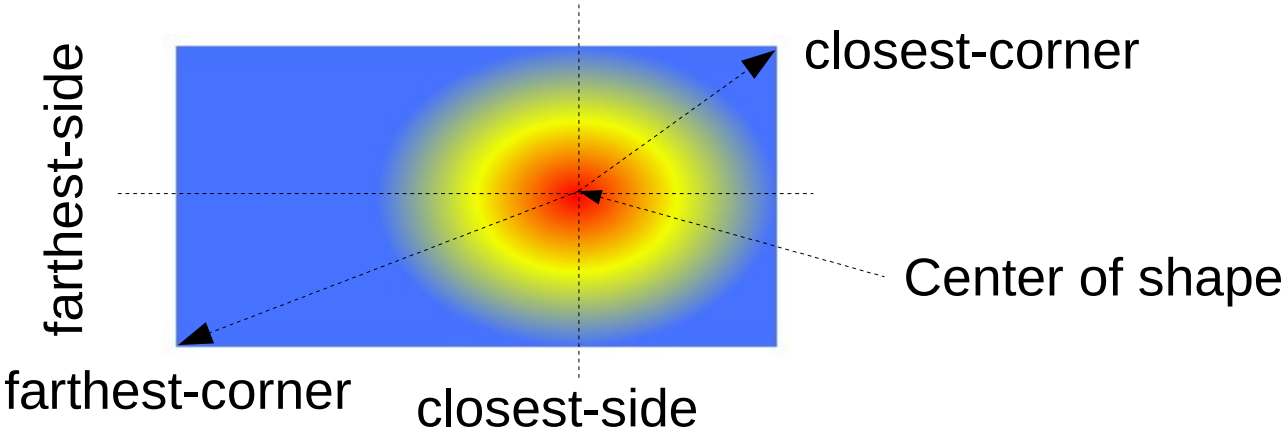


Gradients
(Radial - size)

- Defines the size of the gradient
- Possible values
  - ‘closest-side’
  - ‘closest-corner’
  - ‘farthest-side’
  - ‘farthest-corner’ (default)

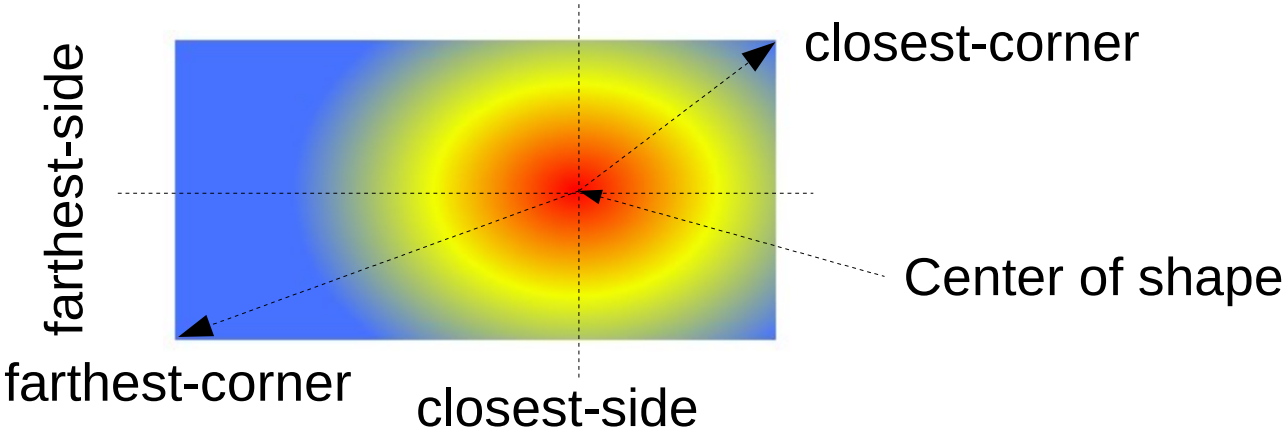
Gradients
(Radial - size)

- ‘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)



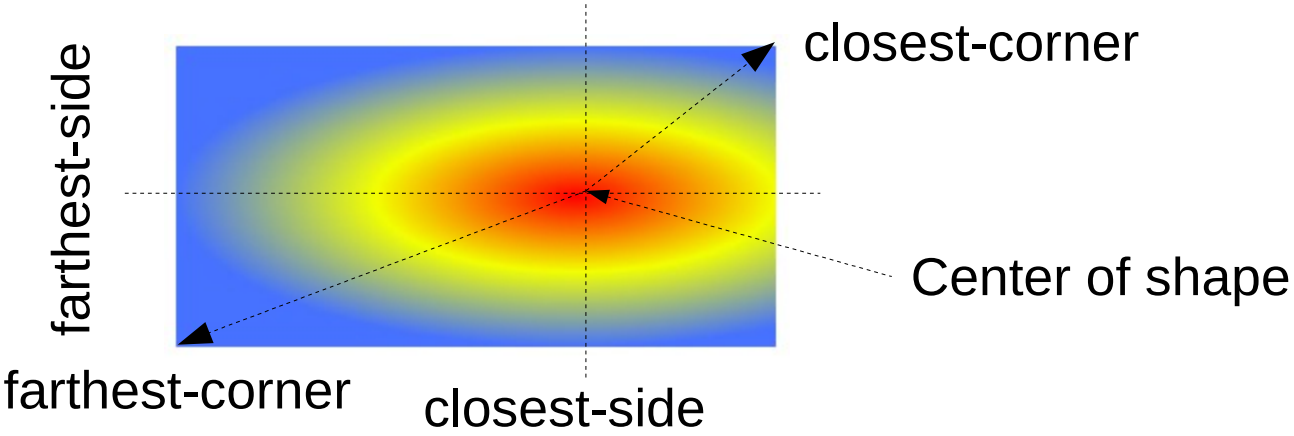
Gradients
(Radial - size)

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



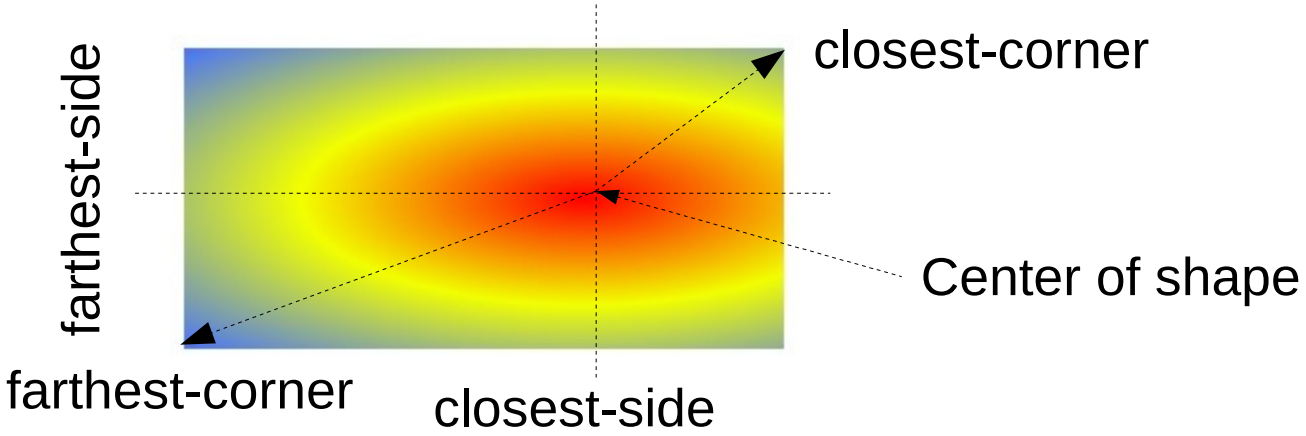
Gradients
(Radial - size)

- ‘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)



Gradients
(Radial - size)

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



Shadows

(Cascading Style Sheets 3)

Shadows

Webstack Academy

- CSS allow developers to add shadow to text and elements
- Properties used to get the shadow effect
  - text-shadow
  - box-shadow

Text Shadow

- ‘text-shadow’ property applies shadow to text

Syntax :

```
selector {
    text-shadow: none | h-shadow v-shadow blur-radius color |
initial | inherit;
}
```

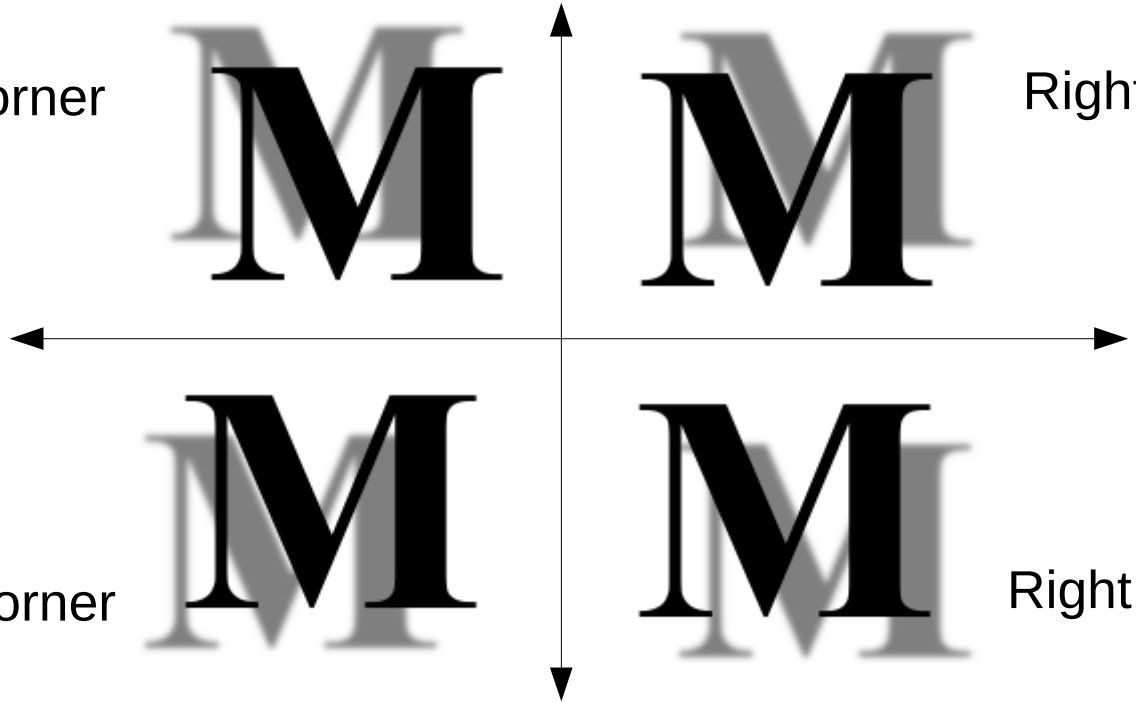
Text Shadow

Value	Field	Description
none	-	Default – No shadow
h-shadow	Required	• Position of horizontal shadow • Negative values are allowed
v-shadow	Required	• Position of vertical shadow • Negative values are allowed
blur-radius	Optional	The blur radius
color	Optional	Hex, rgb, rgba, hsl, hsla or ‘color name’
initial	-	Sets this property to its default value
inherit	-	Inherits this property from its parent element

Text Shadow

Left top corner
- , -

Right top corner
+ , -



Left bottom corner
- , +

Right bottom corner
+ , +

Text Shadow - Blur

No-blur

blur-radius 8px

blur-radius 16px

Box Shadow

- ‘box-shadow’ property applies shadow to elements

Syntax :

```
selector {
    box-shadow: none | h-offset v-offset blur spread color | inset | initial | inherit;
}
```

Box Shadow

Value	Field	Description
none	-	Default – No shadow
h-offset	Required	• The horizontal offset of the shadow • Positive value puts the shadow on right side of the box • Negative value puts the shadow on left side of the box
v-offset	Required	• The vertical offset of the shadow • Positive value puts the shadow below the box • Negative value puts the shadow above the box
blur	Optional	• The blur radius • Higher the number, more blurred the shadow will be

Box Shadow

Value	Field	Description
spread	Optional	• The spread radius • Positive value increases the size of the shadow • Negative value decreases the size of the shadow
color	Optional	• The color of the shadow • The default value is the text color • Unit could be hex, rgb, rgba, hsl, hsla or named color
inset	Optional	Changes the shadow from an outer shadow (outset) to an inner shadow
initial	-	Sets this property to its default value
inherit	-	Inherits this property from its parent element

Box Shadow - Blur

Example - no blur, no-spread

Example - blur 20px

Box Shadow - Spread

Spread 20px

Spread 40px

Transform

(Cascading Style Sheets 3)

Transform

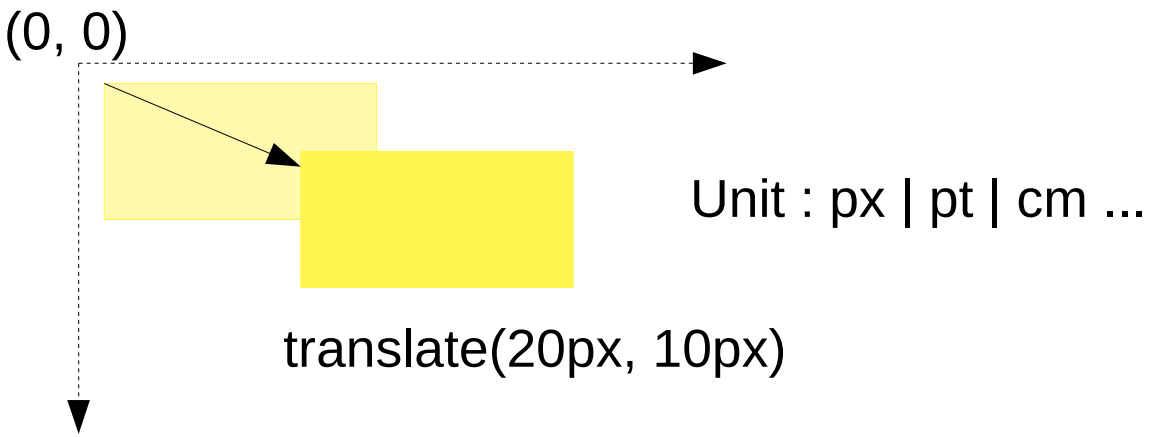
- A transformation is an effect that lets an element change shape, size and position
- CSS transform allow you to translate, rotate, scale, and skew elements
- CSS supports 2D and 3D transformations

2D Transform

- The following methods are provided in CSS for 2D transformations
  - translate()
  - rotate()
  - scale()
  - skewX()
  - skewY()
  - matrix()

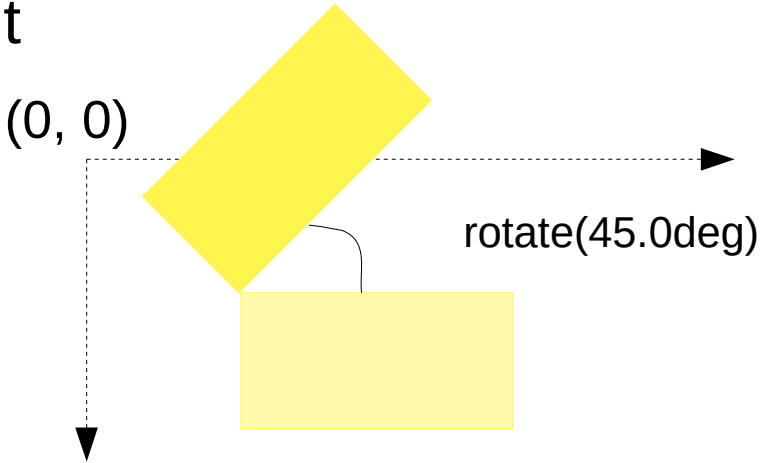
2D Transform
(translate())

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



2D Transform
(rotate())

- The rotate() function defines a transformation that rotates an element around a fixed point on the 2D plane, without deforming it



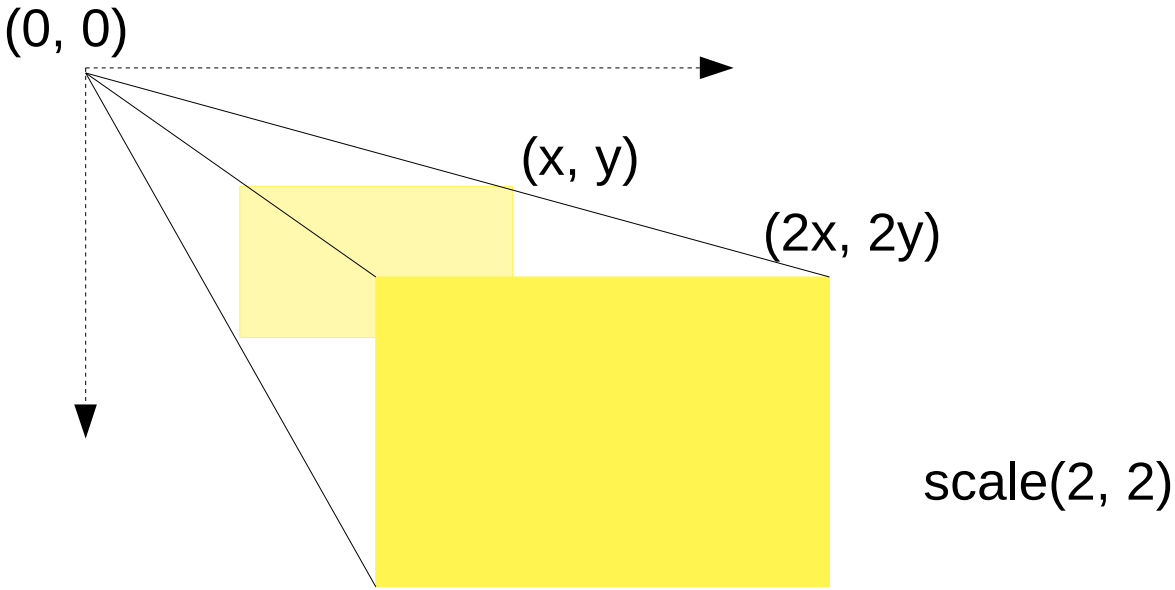
*Transformation takes place with respect to ‘transform-origin’ (default center)

2D Transform
(scale())

- The scale() CSS function defines a transformation that resizes an element on 2D plane
- Because the amount of scaling is defined by a vector, it can resize the horizontal and vertical dimensions at different scales

*Transformation takes place with respect to ‘transform-origin’ (default center)

2D Transform
(scale())



2D Transform
(scale())

Syntax :

scale (sx); */* sx is a number */*

scale (sx, sy); */* sx and sy are numbers */*

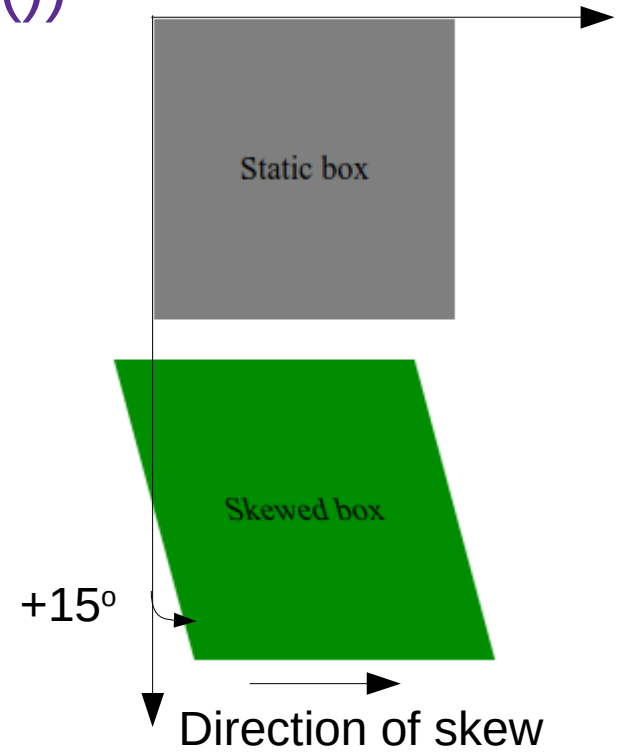
- sx : Represents the abscissa (X-axis) of scaling vector
- sy : Represents the ordinate (Y-axis) of scaling vector; If not defined, its default value is sx, resulting in a uniform scaling that preserves the element's aspect ratio

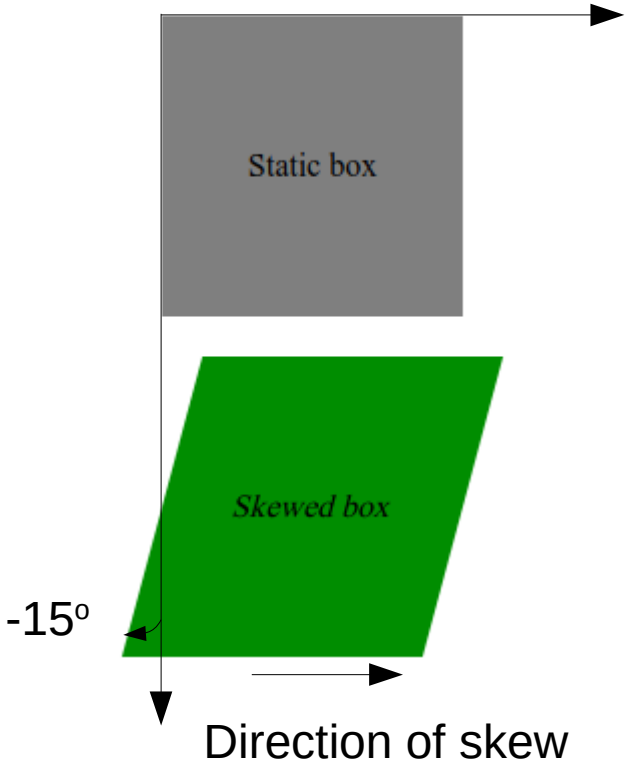
2D Transform
(skew())

- The skew() CSS function defines a transformation that skews an element on the 2D plane
- The skew() function is specified with either one or two values, which represent the amount of skewing (in angle) to be applied in each direction
- The unit of angle value
  - Degree (Example : 15deg)
  - Radian (Example : 0.325rad)
  - Turn (Example : -1.5turn)

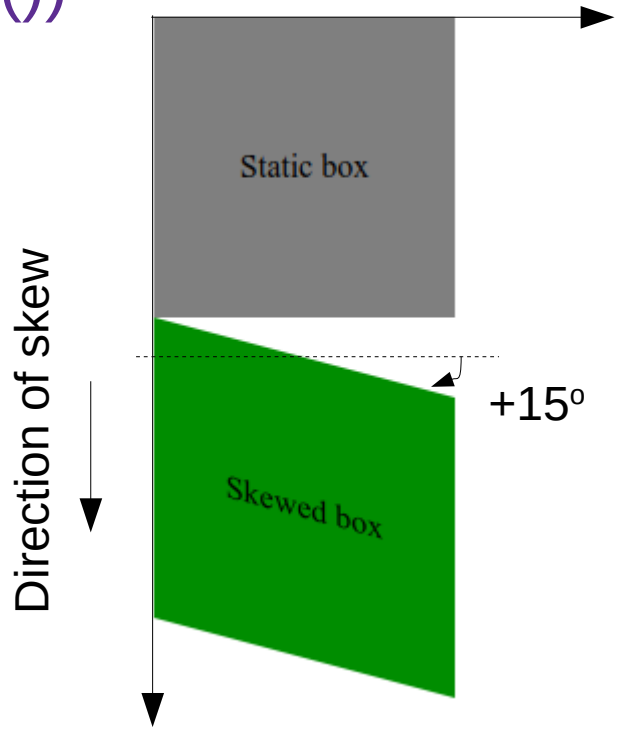
*Transformation takes place with respect to ‘transform-origin’ (default center)

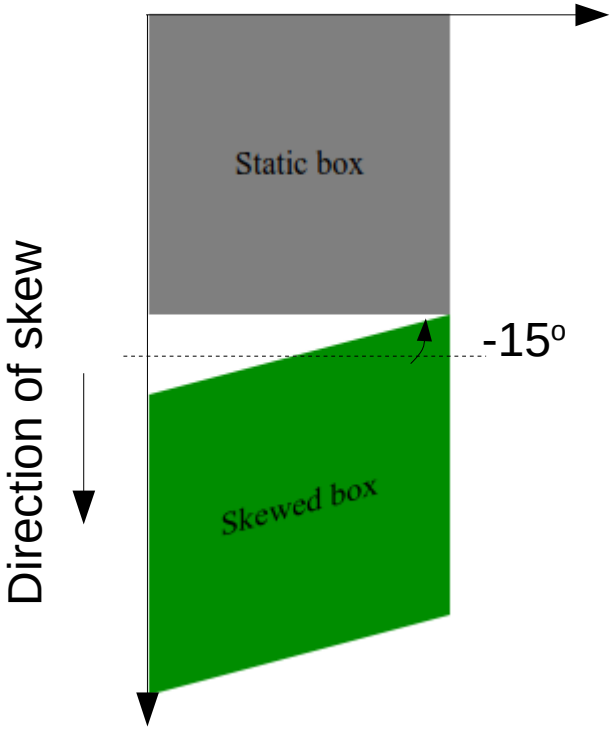
2D Transform
(skewX())





2D Transform
(skewY())





2D Transform
(skew())

Syntax :

skew (ax); */* ax is an angle */*

skew (ax, ay); */* ax and ay are angles */*

- ax : Represents the angle to be used to distort the element along the X-axis (abscissa)
- ay : Represents the angle to be used to distort the element along the Y-axis (ordinate)
  - If not defined, its default value is 0, resulting in a purely horizontal skewing

2D Transform
(skewX(), skewY())

Syntax :

skewX (ax); */* ax is an angle */*

skewY (ay); */* ay is an angle */*

- The skewX() and skewY() are CSS functions defines a transformation that skews an element in horizontal and vertical directions on the 2D plane
- ax and ay : Represents the angle to use to distort the element along the X-axis (abscissa) and Y-axis (ordinate)

2D Transform
(matrix())

- The matrix() method combines all the 2D transform methods into one
- The matrix() function is specified with six values

matrix (scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())

2D Transform
(matrix())

Syntax :

matrix(a, b, c, d, tx, ty);

- a, b, c, d : Are numbers describing the linear transformation
- tx, ty : Are numbers describing the translation to apply

2D Transform
(matrix())

Example :

transform-origin : left; */* default transform origin is center */*

transform: matrix(2, 0, 0, 1, 0, 0);

- ScaleX = 2 : Scale by a factor of 2 in X direction
- SkewY = 0 : No skew in Y direction
- SkewX = 0 : No skew in X direction
- ScaleY = 1 : No scaling in Y direction
- TranslateX = 0 : No movement in X direction
- TranslateY = 0 : No movement in Y direction

3D Transform

- 3D transform functions deform or move the element along X, Y and Z axes
- You should define a container to provide the 3D space to child element(s)
- The sense of 3D depth is defined by perspective
- The transformation takes place with respect to perspective-origin

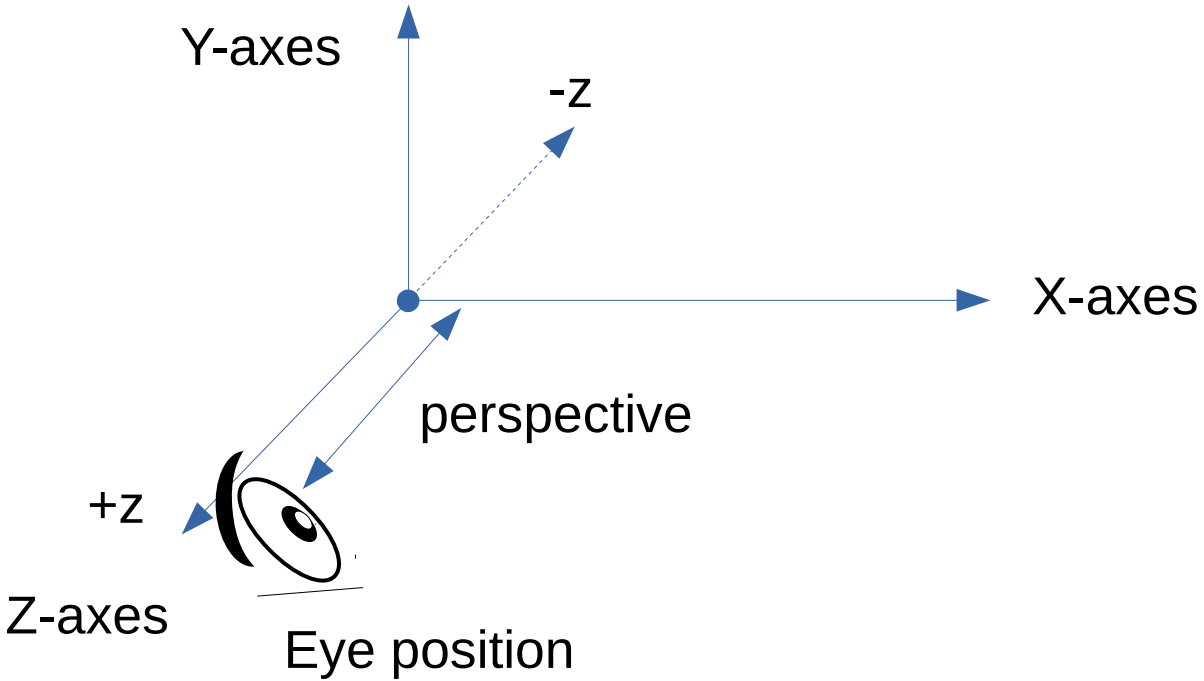
3D Transform properties

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements
transform-style	Specifies how nested elements are rendered in 3D space
perspective	Specifies the perspective on how 3D elements are viewed
perspective-origin	Specifies the bottom position of 3D elements
backface-visibility	Defines whether or not an element should be visible when not facing the screen

3D Transform properties
(perspective)

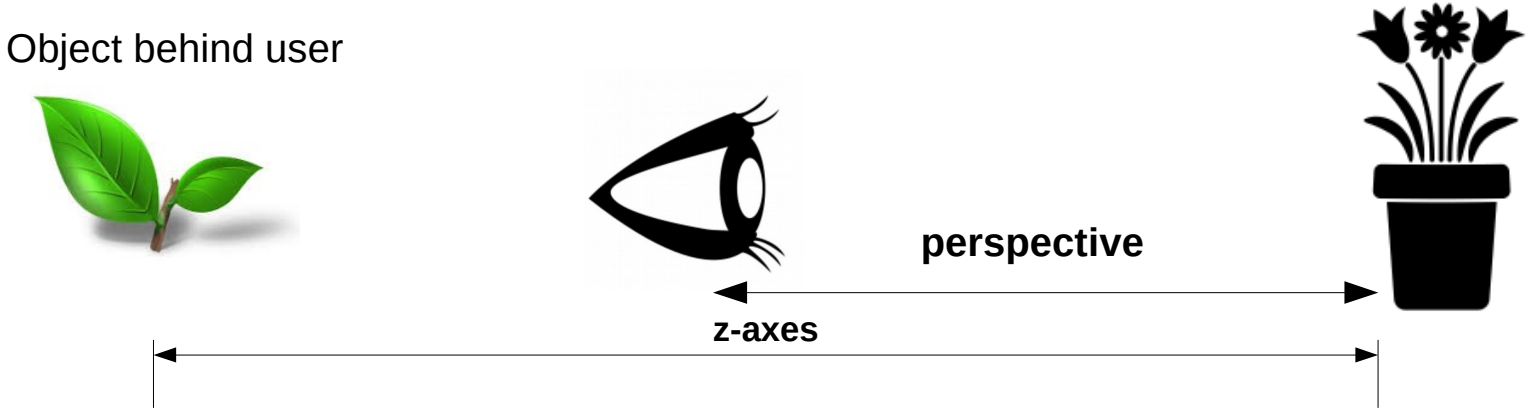
- “perspective” property determines the distance between the z=0 plane and user in order to give a 3D-positioned elemen
- Each 3D element with z>0 becomes larger; each 3D-element with z<0 becomes smaller
- Smaller the perspective larger the object seen by user

3D Transform properties
(perspective)



3D Transform properties
(perspective)

- The parts of the 3D elements that are behind the user — i.e. their z-axis coordinates are greater than the value of the perspective CSS property — are not drawn



3D Transform properties
(perspective)

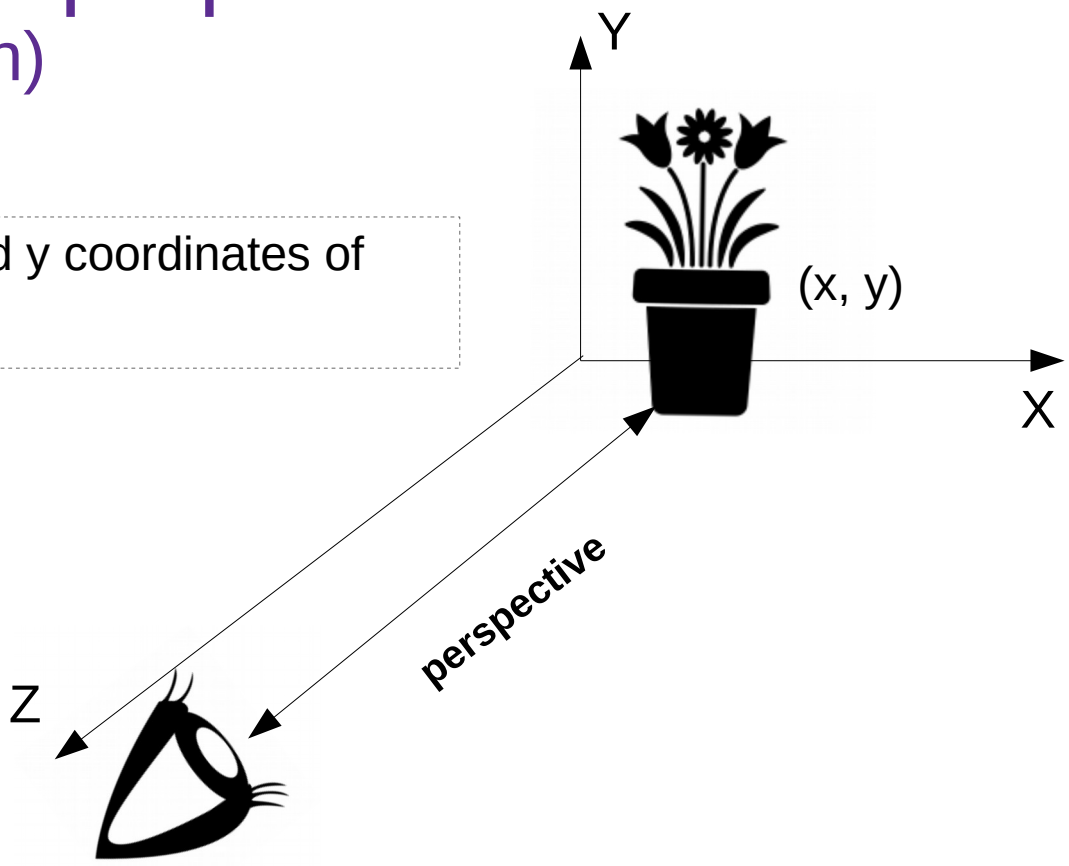
- The vanishing point is by default placed at the center of the element, but its position can be changed using the perspective-origin property
- Perspective length -
  - Giving the distance from the user to the z=0 plane
  - It is used to apply a perspective transform to the element and its content
  - If it is 0 or a negative value, no perspective transform is applied

3D Transform properties
(perspective-origin)

- The perspective-origin property determines the position at which the viewer is looking
- It is used as the vanishing point by the perspective property
- Vanishing point = perspective-origin = x and y position where viewer is looking at

3D Transform properties
(perspective-origin)

Vanishing point is x and y coordinates of center of flower pot



3D Transform properties
(perspective-origin)

Left top

Center

Top

Right bottom

3D Transform properties
(perspective-origin)

Syntax :

perspective-origin : x-axis y-axis | initial | inherit;

x-axes values : left | right | center | length | % /* default 50% */

y-axes values : top | bottom | center | length | % /* default 50% */

3D Transform properties
(transform-style)

Syntax :

transform-style : flat | preserve-3d | initial | inherit;

- “flat” - Specifies that child elements will NOT preserve its 3D position (default)
- “preserve-3d” - Specifies that child elements will preserve its 3D position
- This property must be used together with the transform property

3D Transform methods
(translate)

- 3D translate functions move the element by given amount along the given axis
  - translateX(x), translateY(y), translateZ(z)
- Short hand translate function
  - translate3d(x, y, z)
- Unit of x, y, z can be px, pt, cm, mm etc.

3D Transform methods
(rotate)

- 3D rotate functions rotates the element by the given angle (deg, rad or turn) around given axis
  - rotateX(angle), rotateY(angle), rotateZ(angle)
- Shorthand rotate function
  - rotate3d(x, y, z, angle)
  - Value of x, y and z could be between 0 and 1
  - if x, y, z are positive the movement will be clockwise
  - if x, y, z are negative the movement will be anti-clockwise

3D Transform methods
(scale)

- 3D scale functions scales the element by given amount along the given axis
  - scaleX(sz), scaleY(sy), scaleZ(sz)
- Shorthand scale function
  - scale3d(sx, sy, sz)

3D Transform methods
(matrix)

- “matrix3d()” specifies a 3D transformation in the form of a 4×4 transformation matrix of 16 values
  - Matrix3d(n, n, n, n, n, n, n, n, n, n, n, n, n, n, n, n)
- “perspective(length)” Defines a perspective view for a 3D transformed element
- As the value of perspective function increases, the element will appear further away from the viewer

Transition

(Cascading Style Sheets 3)

Transition

- CSS transitions allows you to change property values smoothly (control transition speed from one value to another) over a given duration
- Instead of having property changes take effect immediately, transition allow the changes in a property to take place over a period of time
- Following points are important to create a transition effect
  - CSS property (to which effect to be applied)
  - Duration of the effect (non-zero value)

Transition

- Not all properties can be used to apply transition effect
- The set of properties that can be animated is changing as the specification develops
- Default value of duration is zero, therefore, the transition will have no effect if not specified
- Specification recommends not animating from and to auto value

Transition properties

- “**transition-property**” - The name of property on which transition to be applied
- “**transition-duration**” - How long transition shall last (higher the duration slower the transition)
- “**transition-timing-function**” - Used to define a function that describes how a transition will proceed over its duration, allowing a transition to change speed during its course
- “**transition-delay**” - When to trigger commencement of transition (zero delay means start transition immediately) from occurrence of event

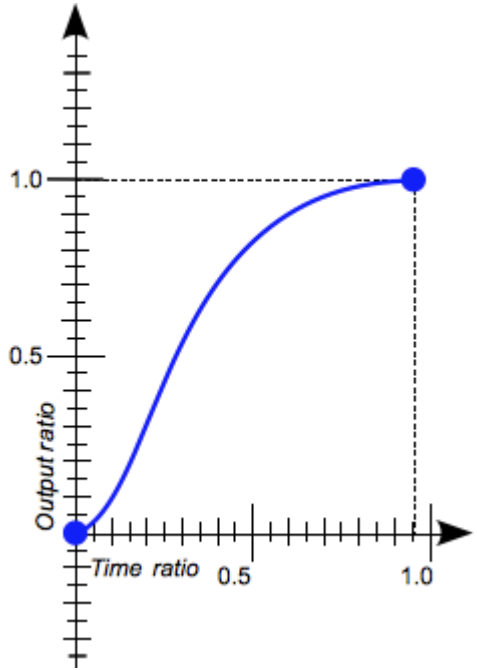
Transition properties
(transition-timing-functions)

Name	Description
ease (default)	• Transition effect starts slow, accelerates sharply, then ends slowly • Equivalent to cubic-bezier(0.25, 0.1, 0.25, 1.0)
linear	• Transition effect remains at the same speed from start to end • Equivalent to cubic-bezier(0.0, 0.0, 1.0, 1.0)
ease-in	• Transition effect starts slow, progressively speeds up and ends abruptly • Equivalent to cubic-bezier(0.42, 0.0, 1.0, 1.0)
ease-out	• Transition effect starts abruptly and progressively slows down towards the ends • Equivalent to cubic-bezier(0.0, 0.0, 0.58, 1.0)
ease-in-out	• Transition effect starts slow, speeds up and ends slowly • Equivalent to cubic-bezier(0.42, 0.0, 0.58, 1.0)

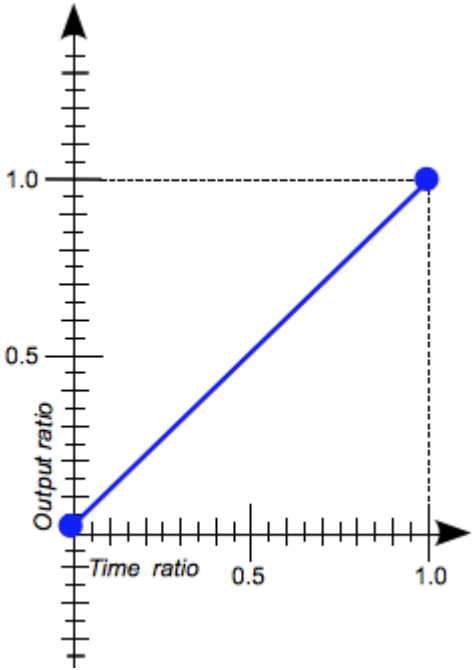
Transition properties
(transition-timing-functions)

Name	Description
step-start	Is equivalent to steps(1, start)
step-end	Is equivalent to steps(1, end)
steps()	• Called step of staircase function • Allows you to specify intervals for the timing function • It takes one or two parameters, separated by a comma (a positive integer and an optional value of either start or end) • If no second parameter is included, it will default to end
cubic-bezier()	Lets you define your own values in a cubic-bezier function

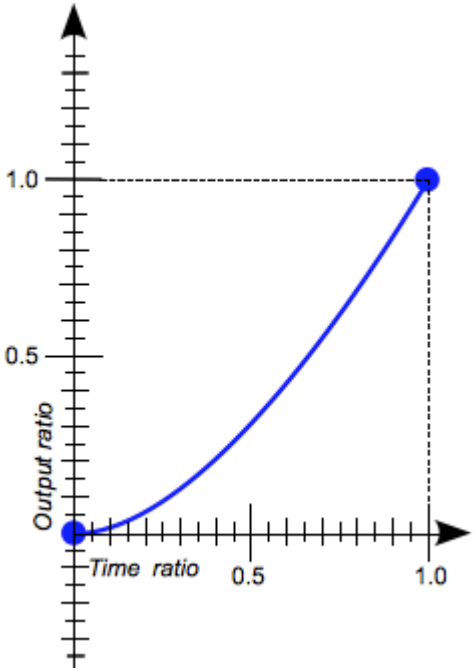
Transition properties
(transition-timing-functions - Ease)



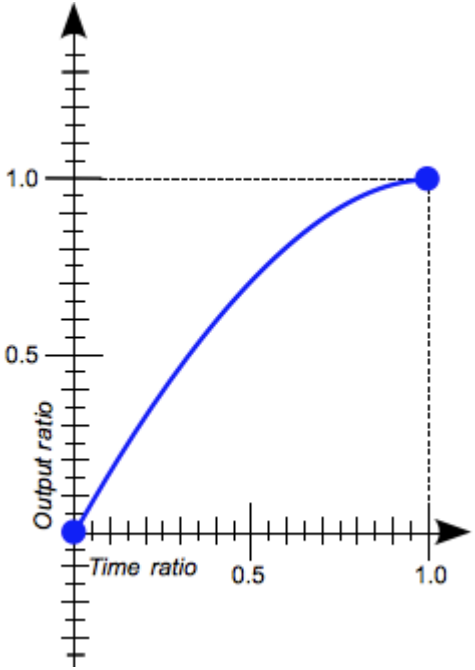
Transition properties
(transition-timing-functions - Linear)



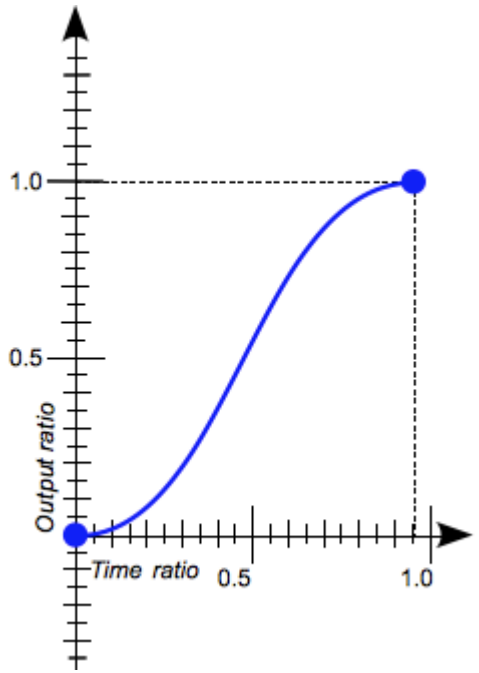
Transition properties
(transition-timing-functions - Ease-in)



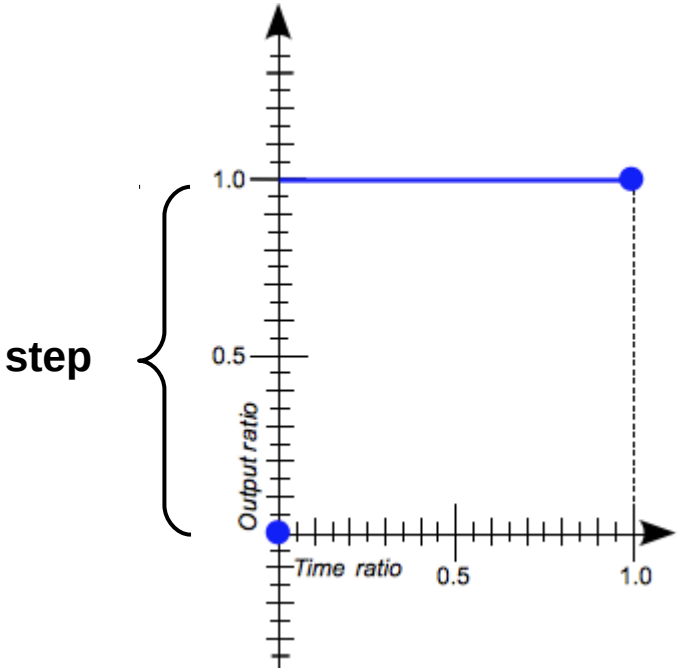
Transition properties
(transition-timing-functions - Ease-out)



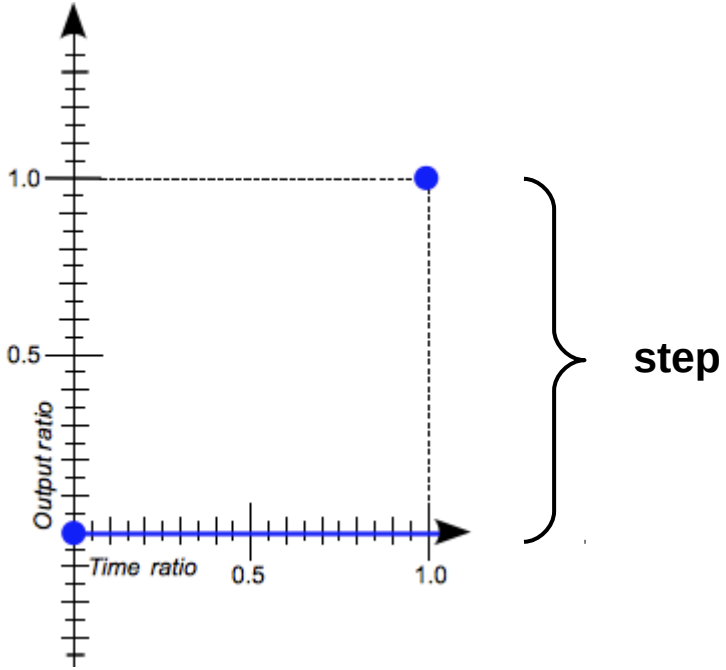
Transition properties
(transition-timing-functions - Ease-in-out)



Transition properties
(transition-timing-functions - step-start)



Transition properties
(transition-timing-functions - step-end)



Transition properties
(steps ())

Syntax :

steps (number_of_steps, direction);

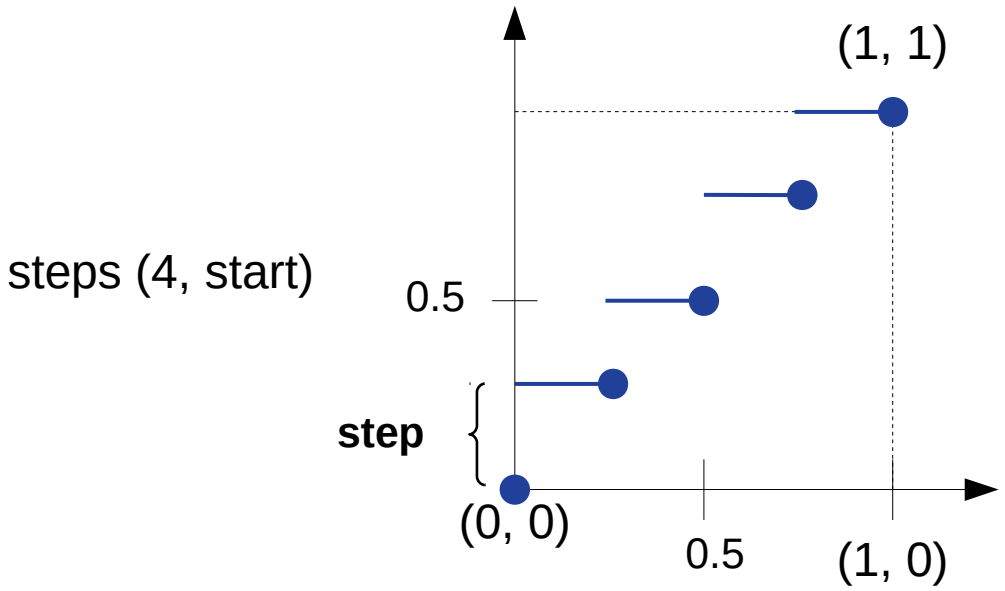
Example :

steps (4, start); */* 4 steps stair-case, first one is happening in the beginning */*

- number_of_steps : shall be strictly > 0
- direction : “start” or “end”

Transition properties
(transition-timing-functions)

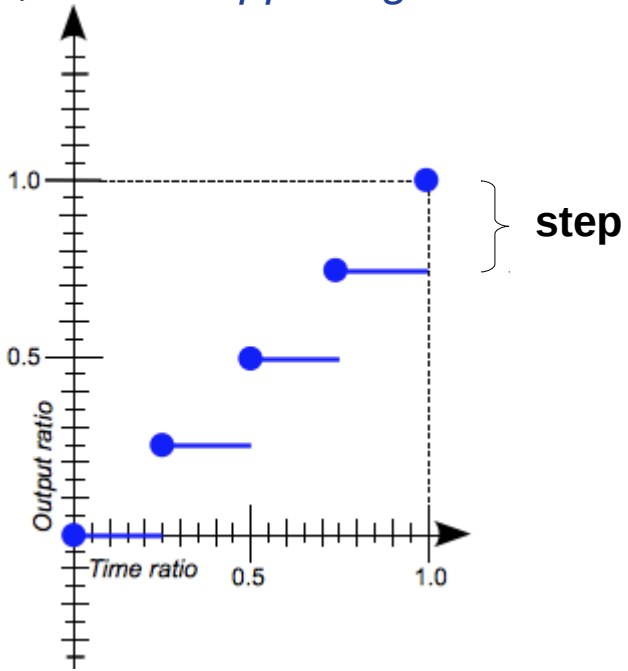
4 steps stair-case, first one is happening in the beginning



Transition properties
(transition-timing-functions)

4 steps stair-case, one is happening in the end

steps (4, end)



Transition properties
(cubic-bezier ())

Syntax :

cubic-bezier (x1, y1, x2, y2);

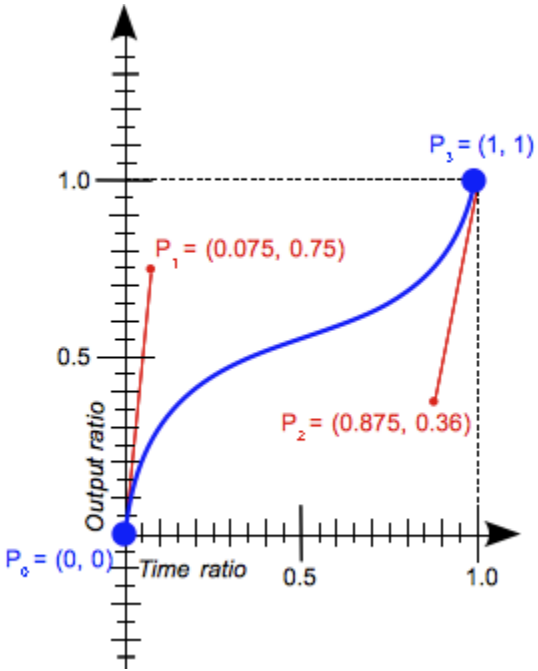
Example :

Cubic-bezier (0.0, 1.0, 0.7, 4);

- X1, y1, x2, y2 can be integer of float numbers
- x1 and x2 must be in the range [0, 1] or the value is invalid
- y1 and y2 outside [0, 1] range may generate bouncing effects

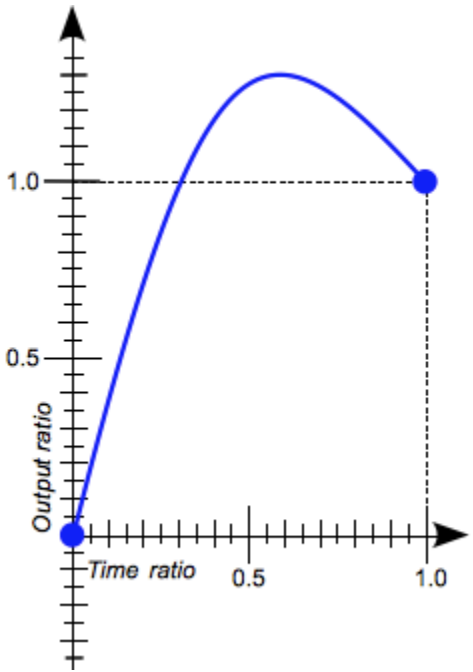
Transition properties
(cubic-bezier ())

- A cubic Bezier curve is defined by four points P0, P1, P2, and P3
- P0 and P3 are the start and the end of the curve



Transition properties
(cubic-bezier ())

- A cubic Bezier curve which may create bouncing effect



Transition properties
(Syntax)

```
Syntax :
    selector {
      transition-property: <property-name>;
      transition-duration: <duration>;
      transition-timing-function: <timing-function>
      transition-delay: <delay>;
    }
```

Transition Example

Example :

```
h1 {
  transition-property: font-size;
  transition-duration: 3s;
  transition-delay: 0;
}
```

Transition properties
(shorthand property)

Syntax :

```
selector {
  transition: <property> <duration> <timing-function> <delay>;
}
```

Transition properties
(vendor prefixes)

- Along with normal transition properties, you may have to use vendor prefixed properties for different browsers
  - Safari : -webkit-
  - Firefox : -moz-
  - Opera : -o-
  - IE9 or less : -ms-

Transition Example

Example :

```
h1 {
  -webkit-transition-property: font-size;
  -webkit-transition-duration: 3s;
  -webkit-transition-delay: 0;

  transition-property: font-size;
  transition-duration: 3s;
  transition-delay: 0;
}
```

Animation

(Cascading Style Sheets 3)

Animation

- CSS animations allows animation of most HTML elements without using JavaScript or Flash
- An animation lets an element gradually change from one style to another
- Animations consist of two components
  - Properties describing the CSS animation
  - Set of keyframes that indicate start and end states of the animation’s style, as well as possible intermediate waypoints

Animation

- There are three key advantages to CSS animations over traditional script-driven animation techniques
  - Easy to use for simple animations
  - The animations run well, even under moderate system load
  - Letting the browser control the animation sequence, hence, allow browser to optimize performance and efficiency

Animation properties

Property	Description
animation-delay	Delay for the start of an animation
animation-direction	Animation playing direction forwards, backwards or in alternate cycles
animation-duration	How long an animation should take to complete one cycle
animation-fill-mode	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
animation-iteration-count	The number of times an animation should be played
animation-name	The name of the @keyframes animation

Animation properties

Property	Description
animation-play-state	State whether animation is running or paused
animation-timing-function	The speed curve of the animation
animation	Shorthand property

Animation properties

Syntax :

```
selector {
    animation-name: keyframe_name;
    animation-duration: <duration>;
    animation-timing-function: <function name>;
    animation-delay: <delay>; /* +ve or -ve value in seconds or milliseconds */
    animation-iteration-count: <count>; /* number | infinite , default is 1 */
    animation-direction: <direction>; /* normal | reverse | alternate | alternate-reverse */
    animation-fill-mode: <fill-mode>; /* none | forward | backward | both */
    animation-play-state: <play-state>; /* running | paused */
}
```

Animation properties

Syntax :

```
selector {
    animation: <name> <duration> <timing-function> <delay> <iteration-count> <direction> <fill-mode> <play-state>;
}
```

Animation Keyframes

- The keyframes are defined using the @keyframes at-rule
- Stages of the animations are represented as a percentage
- 0% represents the beginning state of the animation
- 100% represents the ending state of the animation
- Many intermediate states can be added in between

Animation Keyframes

Syntax :

```
@keyframes keyframe_name {
    0% { . . . CSS code . . . }  /* start state */
    25% { . . . CSS code . . . } /* intermediate states */
    50% { . . . CSS code . . . }
    75% { . . . CSS code . . . }
    100% { . . . CSS code . . . } /* end state */
}
```

Animation direction

- “**normal**” (default) -
  - The animation plays forward
  - On each cycle the animation resets to the beginning state (0%)
  - And plays forward again (to 100%).
- “**reverse**” -
  - The animation plays backwards
  - On each cycle the animation resets to the end state (100%)
  - And plays backwards (to 0%)

Animation direction

- “**alternate**” -
  - The animation reverses direction every cycle
  - On each odd cycle, the animation plays forward (0% to 100%)
  - On each even cycle, the animation plays backwards (100% to 0%)
- “**alternate-reverse**” -
  - The animation reverses direction every cycle
  - On each odd cycle, the animation plays in reverse (100% to 0%)
  - On each even cycle, the animation plays forward (0% or 100%)

Animation fill-mode

- “**backwards**” -
  - Before the animation (during the animation delay), the styles of the initial keyframe (0%) are applied to the element
- “**forwards**” -
  - After the animation is finished, the styles defined in the final keyframe (100%) are retained by the element

Animation fill-mode

- “**both**” - The animation will follow the rules for both forwards and backwards, extending the animation properties before and after the animation
- “**normal**” (default) - The animation does not apply any styles to the element, before or after the animation

WSA | Forward looking IT finishing school

Thank you

Web Stack Academy (P) Ltd

#83, Farah Towers,

1st floor,MG Road,

Bangalore – 560001

M: +91-80-4128 9576

T: +91-98862 69112

E: info@www.webstackacademy.com