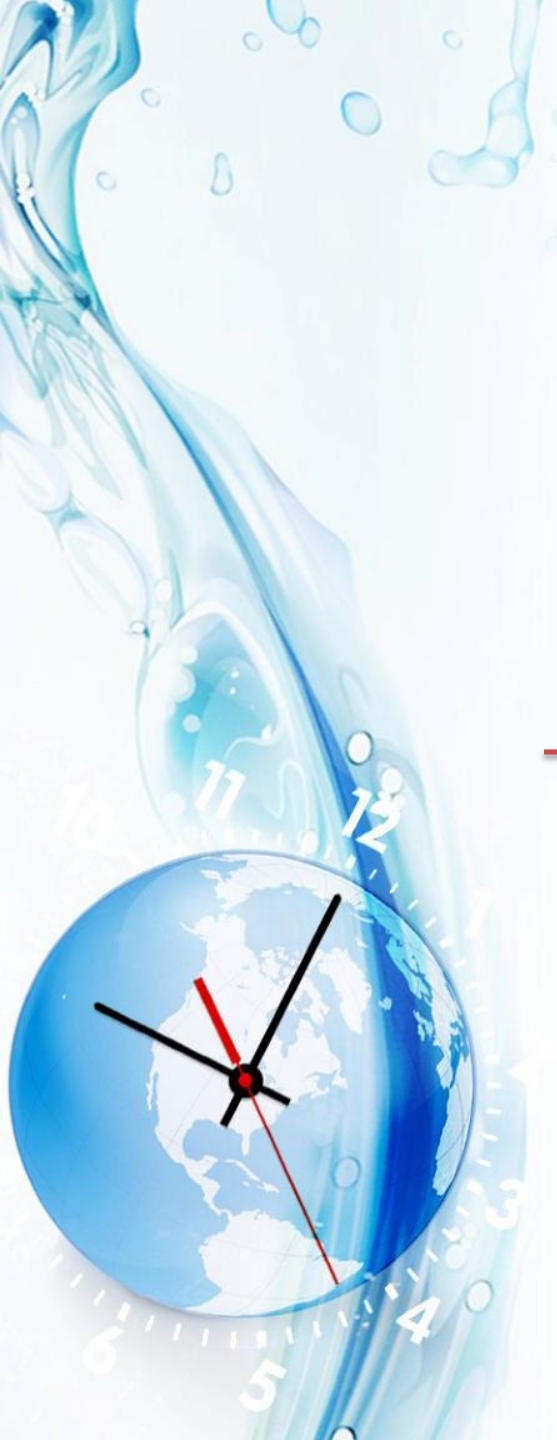
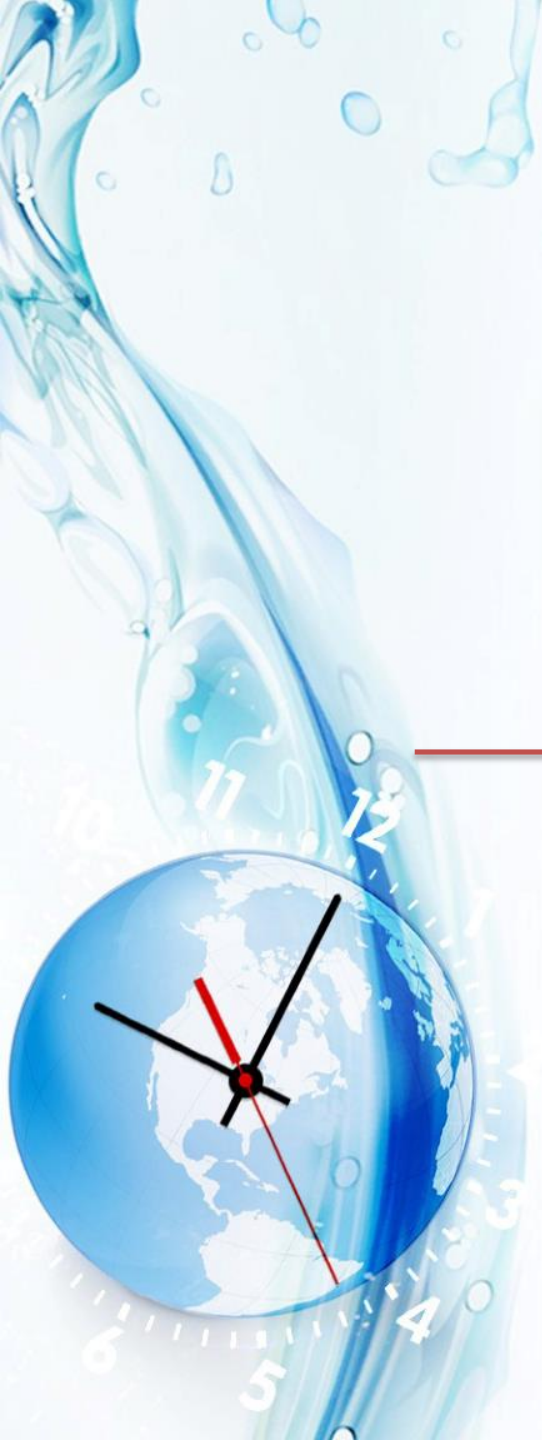


Cascading Style Sheets Ver 3 (CSS3)

National Telecommunication Institute - NTI



CSS Vendor Prefixes

Vendor prefixes are a way for browser makers to add support for new CSS features before those features are fully supported in all browsers.

The CSS browser prefixes that you can use (each of which is specific to a different browser) are:

- *-webkit-*

  Used for: Chrome, Safari, iOS Safari / iOS WebView, Android

- *-moz-*

  Used for: Firefox

- *-o-*

  Used for: Opera, Opera Mini

- *-ms-*

  Used for: Edge, Internet Explorer

In most cases, to use a brand new CSS style property, you take the standard CSS property and add the prefix for each browser. The prefixed versions would always come first (in any order you prefer) while the normal CSS property will come last.

Example

```
-webkit-border-radius:8px;
-moz-border-radius: 8px;
-o-border-radius: 8px;
-ms-border-radius: 8px;
border-radius: 8px;
```

CSS3 - Rounded Corners

Description:

Adds special rounded corners to an element by using the "border-radius" property.

Property Name:

border-radius: px or percent;

Example:

border-radius:15px;

➔ all corners

border-radius:15px 10px;

➔ TL,BR: 15px ➔ TR,BL: 10px

border-radius:15px 12px 10px 8px;

➔ TL:15px, TR:12px, BR:10px, BL:8px

CSS3 – Single Rounded Corner

Description:

Adds single rounded corner to an element using one of the following properties:

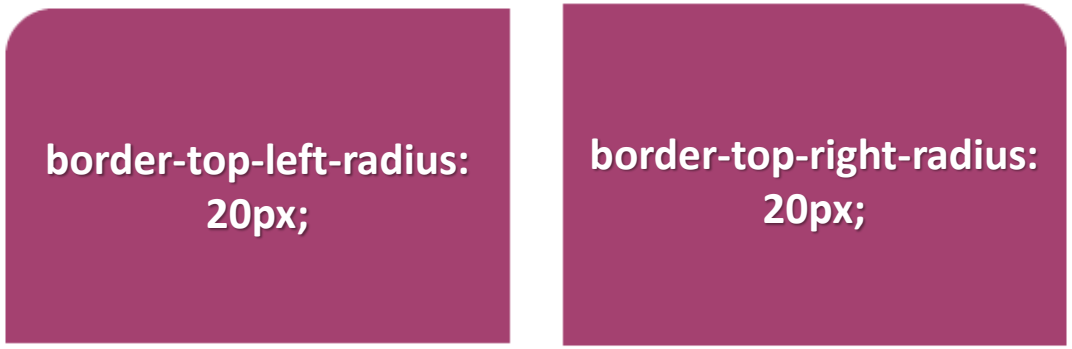
Properties:

border-top-left-radius: px or percent;

border-top-right-radius: px or percent;

border-bottom-right-radius: px or percent;

border-bottom-left-radius: px or percent;



CSS3 background-size

Description:

used to specify the size of the background images.

Property Name:

background-size: px | percent | cover | contain;

Example:

background-size: 120px 100px;

➔ background image width:120px, height:100px

background-size: 30% 10%;

➔ background image width: 30% of the element width, height: 10% of the element width.

background-size: cover;

➔ The bg image will be stretched to fill the element.

background-size: contain;

➔ The bg image will be scaled with the same aspect ratio to fit the element width and height (completely contained inside the element)

CSS3 background-origin

Description:

Specifies whether an element's background image (only) extends into the border or not.

Property:

background-origin: border-box | padding-box | content-box;

Example:

```
.box {
    width: 250px;
    height: 150px;
    padding: 10px;
    border: 6px dashed #333;
    background-size: contain;
    background-origin: content-box;
    background: url("images/sky.jpg") no-repeat;
}
```

CSS3 background-clip

Description:

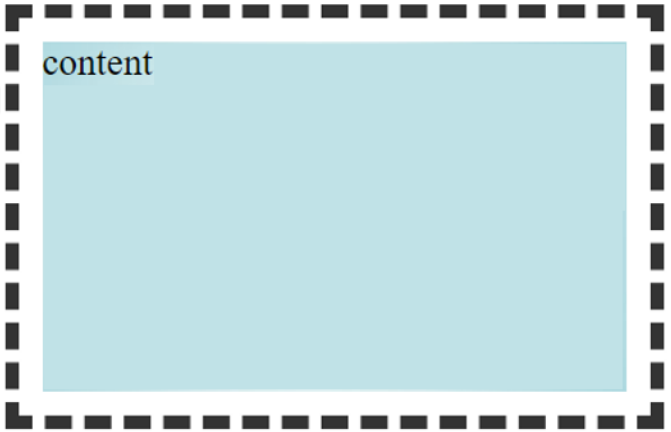
Specifies whether an element's background color extends into the border or not.

Property:

background-clip: border-box | padding-box | content-box;

Example:

```
.box {
    width: 250px;
    height: 150px;
    padding: 10px;
    border: 6px dashed #333;
    background-origin: content-box;
    background-color:#c0e2e7;
}
```



CSS3 Multiple Backgrounds

Description:

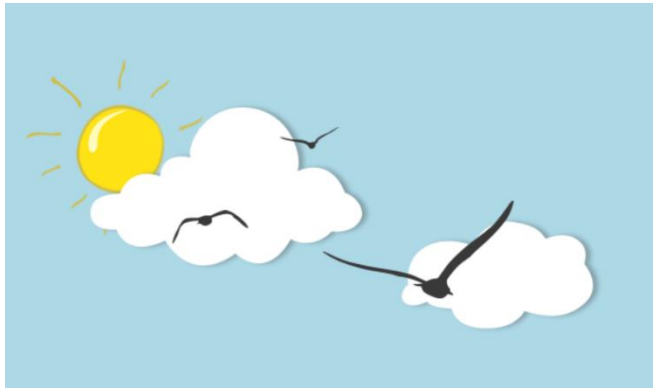
used to add multiple backgrounds to a single element. The backgrounds are layered on the top of one another. The number of layers is determined by the number of comma-separated.

Property Name:

background: all backgrounds values

Example:

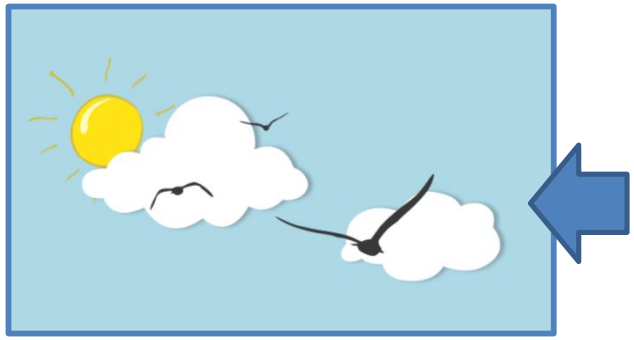
```
.box {
width: 100%;
height: 500px;
background:
   url("images/birds.png") no-repeat center,
   url("images/clouds.png")  no-repeat center,
   lightblue;
}
```

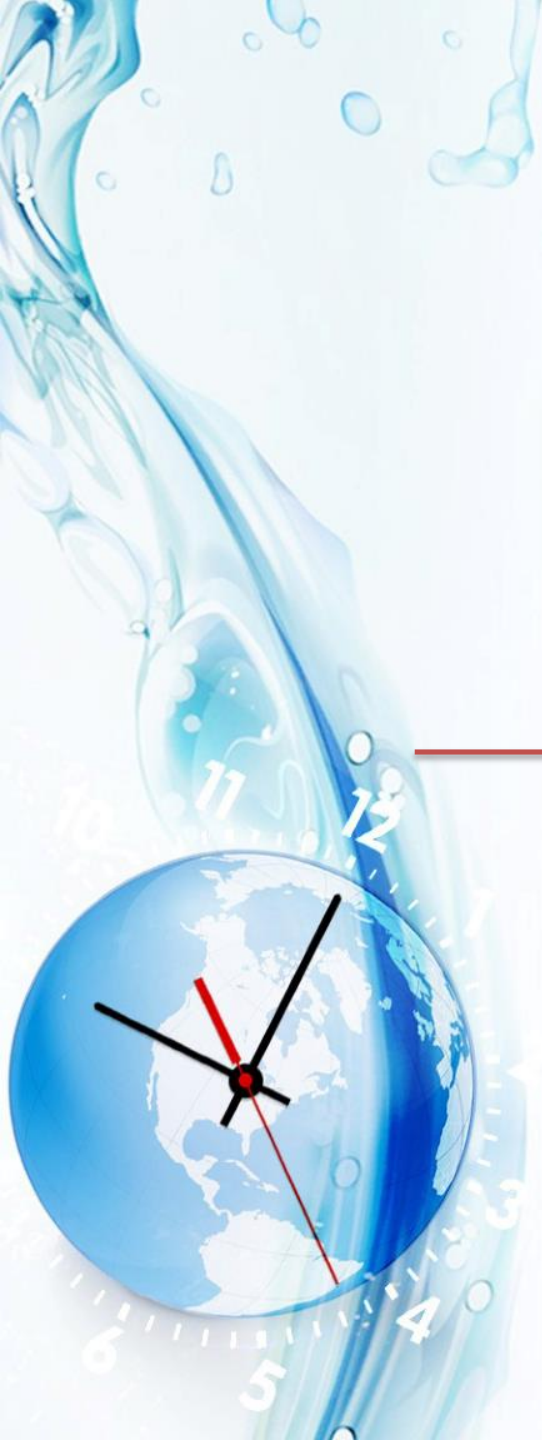


CSS3 Multiple Backgrounds

Example:

```
.box {
width: 100%;
height: 500px;
background:
    url("birds.png") no-repeat center,      ← 1
    url("clouds.png")  no-repeat center,    ← 2
    url("sun.png")  no-repeat 10% 30%,      ← 3
    lightblue; }                            ← 4
```





CSS3 Gradients

The CSS3 gradient feature allows to create a gradient from one color to another without using any images.

Gradients are available in two styles: linear and radial.

Description:

The linear gradient goes up/down/left/right and diagonally.

Property Name:

background: linear-gradient(direction, color1, color2, ...);

Example:

```
background: linear-gradient(to top, red, yellow);
background: linear-gradient(to bottom, #900, yellow);
background: linear-gradient(to left, rgb(255,0,0), yellow);
background: linear-gradient(to right, red, yellow);
background: linear-gradient(red, yellow);
```

➔ to bottom is the default direction

Using Vendor Prefixes:

Example 1 (left to right gradient):

```
background: -webkit-linear-gradient(left, red, green);
background: -o-linear-gradient(right, red, green);
background: -moz-linear-gradient(right, red, green);
background: linear-gradient(to right, red , green);
```

Example 2 (bottom to top gradient):

```
background: -webkit-linear-gradient(bottom, red, green);
background: -o-linear-gradient(top, red, green);
background: -moz-linear-gradient(top, red, green);
background: linear-gradient(to top, red , green);
```

Diagonal Linear Gradient

```
background: linear-gradient(to top right, red, yellow);
background: linear-gradient(to top left, #900, yellow);
background: linear-gradient(to bottom right, red, yellow);
background: linear-gradient(to bottom left, red, yellow);
```

Diagonal Linear Gradient using Vendor Prefixes

Example (to bottom right gradient):

```
background: -webkit-linear-gradient(left top, red , green);
background: -o-linear-gradient(bottom right, red, green);
background: -moz-linear-gradient(bottom right, red, green);
background: linear-gradient(to bottom right, red , green);
```

Setting Direction of Linear Gradients Using Angles

Ex1: background: linear-gradient(90deg, red, yellow);

Ex2: background: linear-gradient(50deg, #900, yellow);

Ex3: background: linear-gradient(-30deg, red, yellow);

CSS3 Radial Gradients

Description:

In a radial gradient color emerge from a single point and smoothly spread outward in a circular or elliptical shape.

Property Name:

background: radial-gradient(shape, color1, color2, ...);

Example:

background: radial-gradient(circle, red, yellow, green);

background: radial-gradient(ellipse, red, yellow, green);

CSS white-space

Description:

specifies how white-space such as spaces, tabs, and newline characters are handled inside the elements.

Property Name:

white-space: normal | pre | nowrap | pre-wrap

Example:

white-space:pre;

- ➔ *Respects the line breaks and the white spaces as pre-defined in the design time (act as the <pre> tag).*

white-space:nowrap;

- ➔ *Enforces the browser to display all the text content without line breaking.*

white-space:pre-line;

- ➔ *Respects the line breaks only as written in the code. (<pre> for line breaks only).*

white-space:pre-wrap;

- ➔ *Respects the line breaks and the white spaces as pre-defined in the design time and infocre the browser to wrap.*

CSS3 Text Overflow

Description:

Determines how the overflowed text content will be displayed when the value of "white-space:nowrap", and "overflow:hidden".

Property Name:

text-overflow: clip | ellipsis

Example:

text-overflow:clip;

➔ *clipped the overflowed text. (default value)*

text-overflow:ellipsis;

➔ *display '...' to represent clipped text.*

Example:

CSS

```
p {   width: 400px;
      overflow: hidden;
      white-space: nowrap;
      background: #cdcdcd;
      text-overflow: ellipsis; }
```

HTML

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
```

Output

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed…

CSS3 word-break

Description:

The word-break CSS property is used to specify if to break lines within words or not.

Property Name:

word-break: break-all | keep-all

Example:

word-break:break-all;

➔ *Breaks the word at any character.*

word-break:keep-all;

➔ *Breaks the word at and white spaces and hyphenates(-) only.*

CSS3 word-break

Example:

CSS

```
p {   width: 150px;
      padding: 10px;
      background: #90ee90; }
.line1 {
      word-break: break-all; }
.line2 {
      word-break: keep-all; }
```

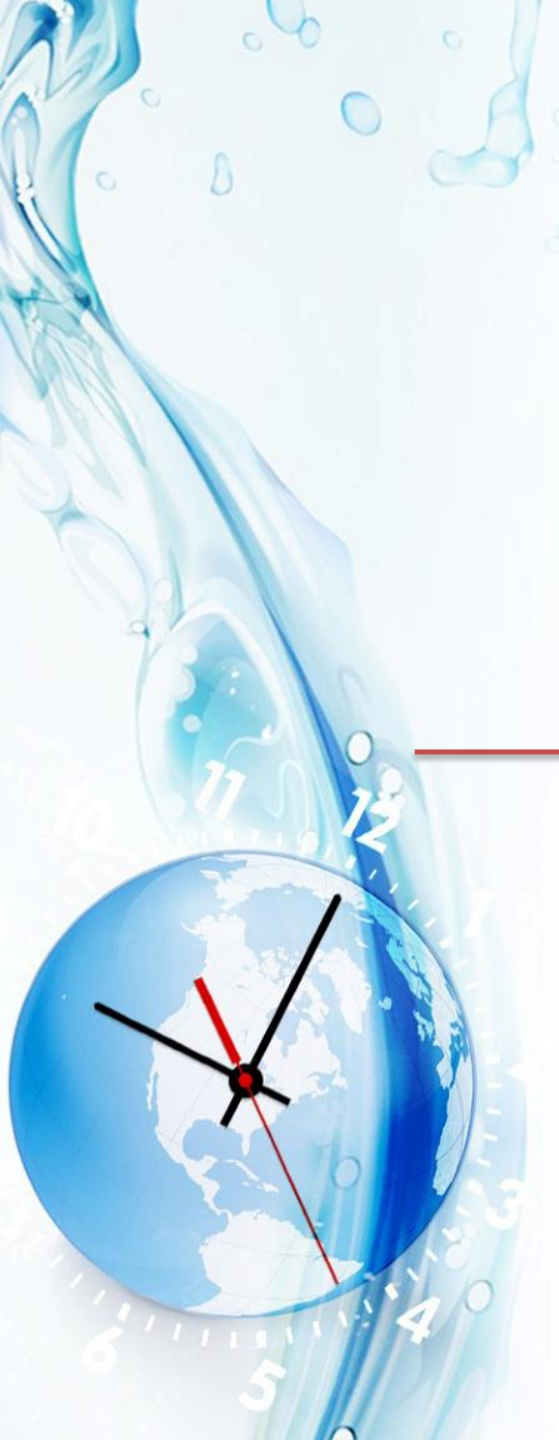
HTML

```
<p class="line1">The line of this paragraph may break at any character.</p>
<p class="line2">The line of this paragraph will-break-at-hyphenates.</p>
```

Output

The line of this paragra ph may break at any ch aracter.

The line of this paragraph will-break-at-hyphenates.



CSS3 Shadows

The CSS3 gives the ability to drop shadow effects to the elements like in Photoshop without using any images.

CSS3 box-shadow

Description:

used to add shadow to the element's boxes.

Property Name:

box-shadow: offset-x offset-y blur color;

offset-x ➔ Sets the horizontal offset of the shadow.

offset-y ➔ Sets the vertical offset of the shadow.

blur ➔ Sets the blur radius.

color ➔ Sets the color of the shadow.

Example:

```
box-shadow: 5px 5px 10px #000;
width: 200px;
height: 150px;
background: #900;
```

Adding multiple box-shadow:

Example1:

```
box-shadow: 5px 5px 10px red, 10px 10px 20px yellow;
width: 200px;
height: 150px;
background: #900;
```

Example2:

```
box-shadow: 5px 5px 10px red, -10px -10px 20px yellow;
width: 200px;
height: 150px;
background: #900;
```

CSS3 text-shadow

Description:

used to add shadow to the text.

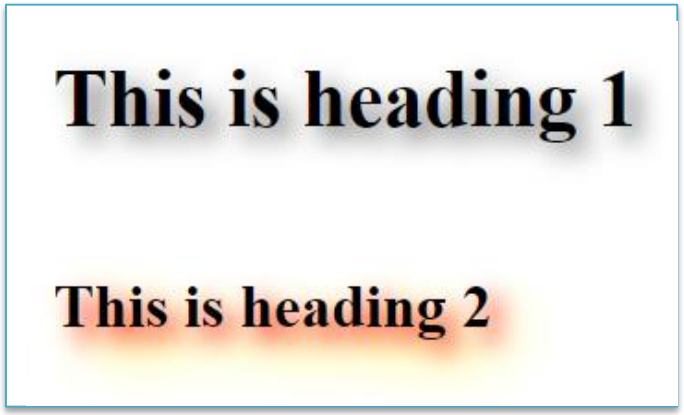
Property Name:

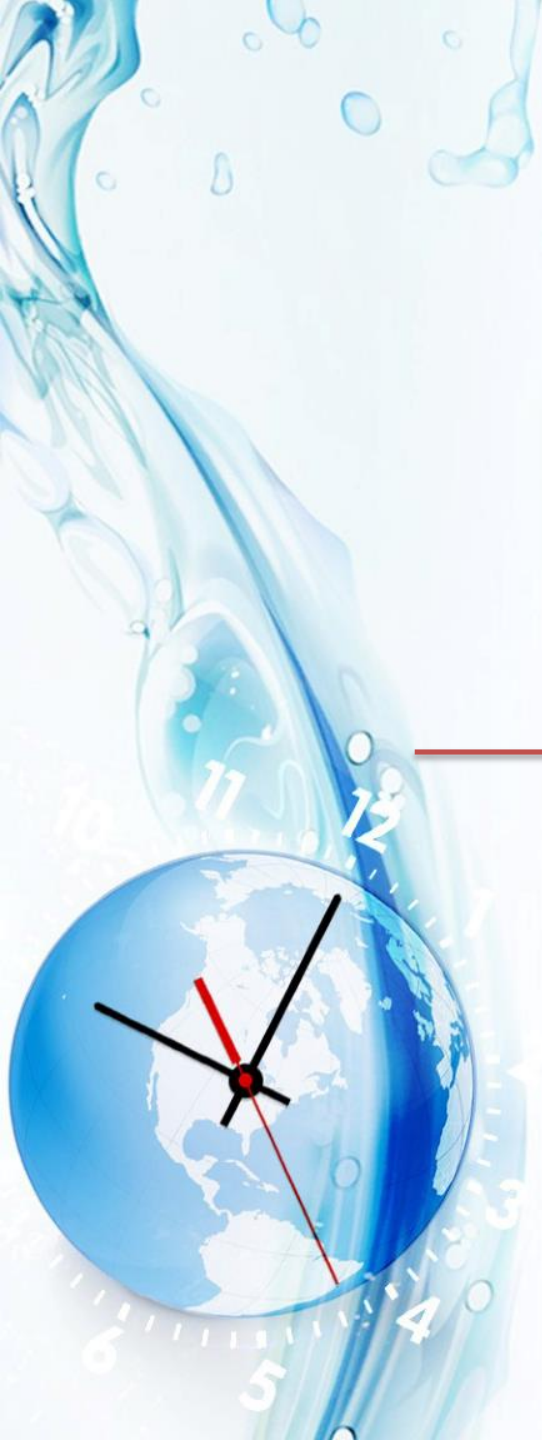
text-shadow: offset-x offset-y blur color;

Example:

```
h1 {
  text-shadow: 5px 5px 10px #666;
      }
h2 {
  text-shadow: 5px 5px 10px red,
  10px 10px 20px yellow;
      }
```

Output





CSS3 Transition

These transition properties allow elements to change values over a specified duration, animating the property changes, rather than having them occur immediately.

transition-duration

Description:

Used to define the duration of a specified transition; the length of time it will take for the targeted to transition between two defined states.

Property:

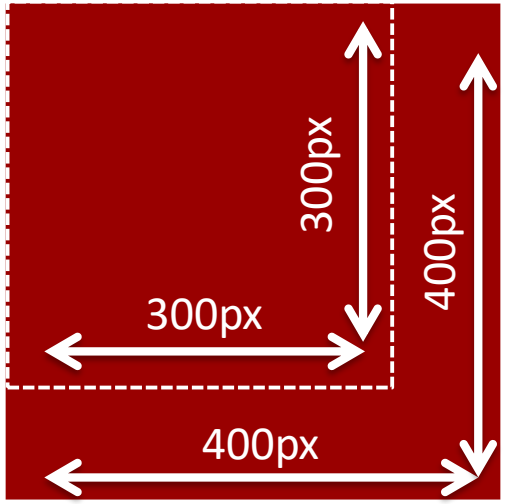
transition-duration: time in seconds (s) or milliseconds (ms)

Default Value:

transition-duration: 0s;

Example:

```
div { width:300px; height:300px;
      background color:#900;
      transition-duration: 2s;}
div:hover {
      width:200px; height:200px;}
```



Note:

Using the transition-duration will case all the changed CSS properties to be affected (affects the width and the height). To target one specific CSS property the “transition-property” property must be used.

transition-property

Description:

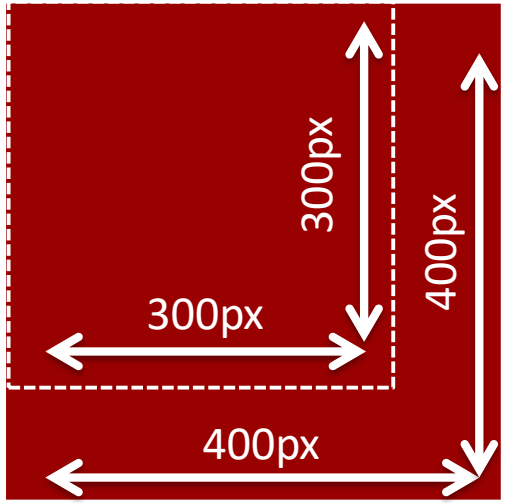
Used to define the CSS property, or properties, to apply a transition effect to, when a user hover over an element

Property:

transition-property: none | all | CSS Property

Default Value:

transition-property: all



Example:

```
div { width:300px; height:300px;
      background-color:#900;
      transition-duration: 2s; transition-property: width;}
div:hover {
      width:200px; height:200px;}
```

Note:

Using the transition-property will cause the transition effect to affect the CSS width property only, but the CSS height property will change without any transitions effects.

transition-timing-function

Description:

used to define a function that describes how a transition will proceed over its duration.

Property:

transition-timing-function: ease | linear | ease-in | ease-out | ease-in-out | step-start | step-end

Default Value:

transition-timing-function: ease

Values:

Linear: The Transition takes the same speed from start to end.

ease: The Transition starts slowly, then fast, then end slowly.

ease-in: The Transition starts slowly, then progressively accelerates until the final state.

ease-out: starts quickly then slow progressively down to its final state.

ease-in-out: starts slowly, then accelerates, then slows down to final state.

cubic-bezier(n,n,n,n):Defines a cubic Bezier curve custom timing function, Possible values are numeric values from 0 to 1. (see: www.cubic-bezier.com).

Transition Example

Example:

```
div { width:300px; height:300px;
        background-color:#900;
        transition-duration: 1s;          ➔ duration is one second
        transition-property: all;         ➔ transition applied for all properties
        transition-timing-function: linear;
}
div:hover {
        width:200px; height:200px;}
```

transition-delay

Description:

Used to define a length of time to delay the start of a transition.

Property:

transition-delay: time in seconds (s)or milliseconds (ms)

Default Value:

transition-delay: 0s

Example:

```
div { width:300px; height:300px; background-color:#900;
    transition-duration: 1s; transition-property: all;
    transition-timing-function: ease-in-out;
    transition-delay: 1s;}
div:hover {
    width:200px; height:200px;}
```

transition shorthand property

Description:

Used as a shorthand for "transition-property", "transition-duration", "transition-timing-function" and "transition-delay" properties.

Property:

transition: transition-property duration timing-function delay;

Default Value:

transition: all 0s ease 0s

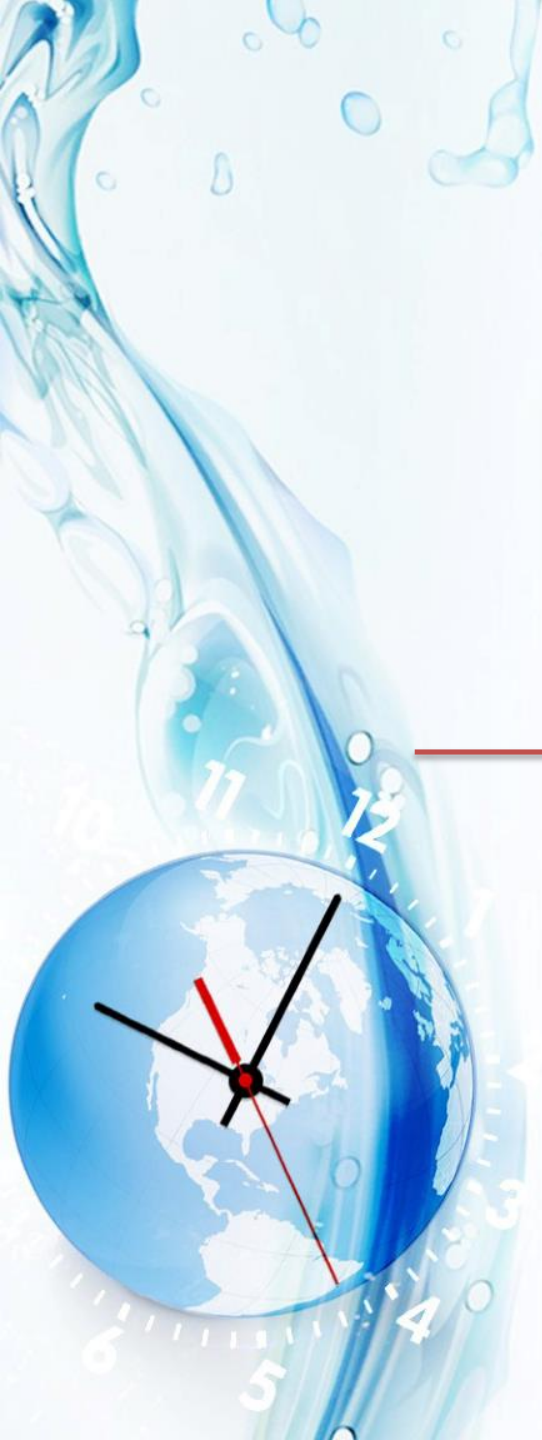
Example:

```
div { width:300px; height:300px; background-color:#900;
      transition: all 0.7s ease-in-out 100ms; }
div:hover {
      width:200px; height:200px;}
```

transition shorthand property

Example transition for 2 or more properties:

```
div { width:300px; height:300px; background-color:#900;
        transition: width 0.7s ease-in-out 100ms,
                    height 0.2s ease  80ms;
        }
div:hover {
        width:200px; height:200px;}
```



CSS3 Filters

used to perform visual effect operations like blur, balancing contrast or brightness, color saturation etc. on graphical elements like an image before it is drawn onto the web page.

Filter Property

Description:

performs visual effect operations on a graphical elements like an image before it is drawn onto the web page.

Property:

filter: none | blur(px) | brightness(%) | contrast(%) | grayscale(%) | drop-shadow() | hue-rotate(deg) | invert(%) | opacity(%) | sepia(%) | saturate(%)

Default value:

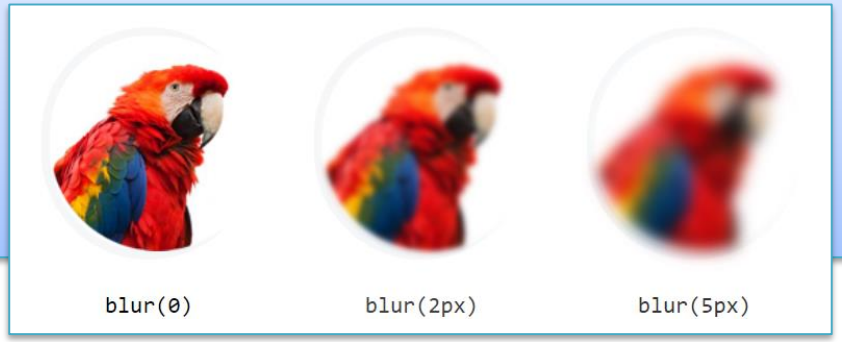
filter: none;

}

Filter Property

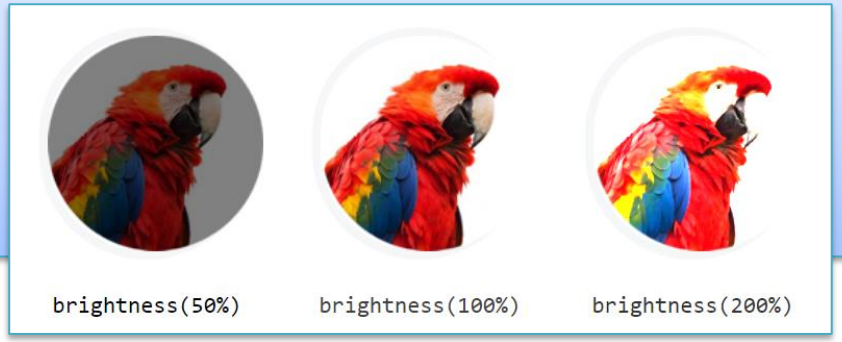
Example 1

```
img {
    filter: blur(5px);
}
```



Example 2

```
img {
    filter: brightness(50%);
}
```

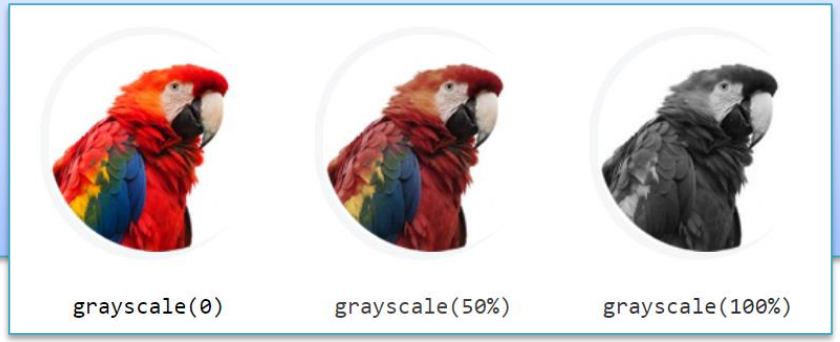


Example 3

```
img {
    filter: contrast(50%);
}
```

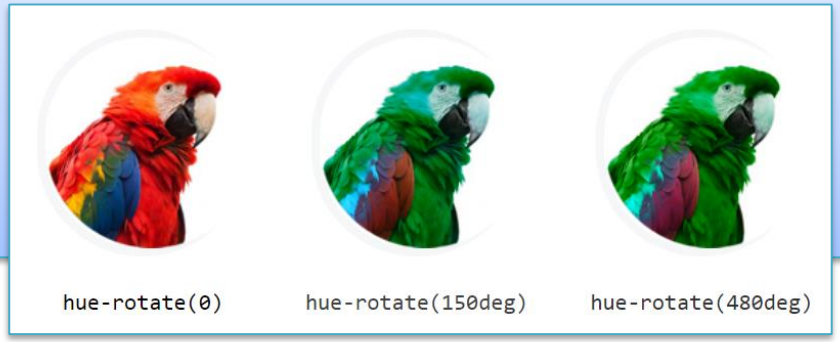
Example 4

```
img {
    filter: grayscale(100%);
}
```



Example 5

```
img {
    filter: hue-rotate(150deg);
}
```



Example 6

```
img {
    filter: invert(80%);
}
```

Description:

Used to alter the way sizing of an element occurs with respect to its "box model".

Property Name:

box-sizing: content-box | border-box

Values:

- *content-box*

  Width and height calculation represents "contents" only while padding, border and margin is left out. (default value)

- *border-box*

  Width and height calculation includes "content, padding and border". Only the margin is left out.

Resize

Description:

Specifies whether or not an element is resizable by the user, and if so, along which direction.

Property Name:

resize: none | both | horizontal | vertical | inherit

Values:

- none

  The element is not resizable.

- both

  The element is resizable in both directions (horizontal and vertical).

- horizontal

  The element can be resized horizontally.

- vertical

  The element can be resized vertically.

Overflow-x

Description:

Specifies whether the content of an element should be visible or not when the content overflows horizontally.

Property Name:

overflow-x: visible | hidden | scroll | auto

Values:

- visible

  The content rendered outside the left and/or right edges of the element and may overlap other elements next to it.

- hidden

  The content is clipped on the left and/or right edges and that no scroll bar should be provided.

- auto

  Adds a horizontal scroll bar when needed.

- scroll

  Adds the horizontal scroll bar in all cases.

Overflow-y

Description:

Specifies whether the content of an element should be visible or not when the content overflows vertically.

Property Name:

overflow-y: visible | hidden | scroll | auto

Values:

- *visible*

  The content rendered outside the top and/or bottom edges of the element and may overlap other elements next to it.

- *hidden*

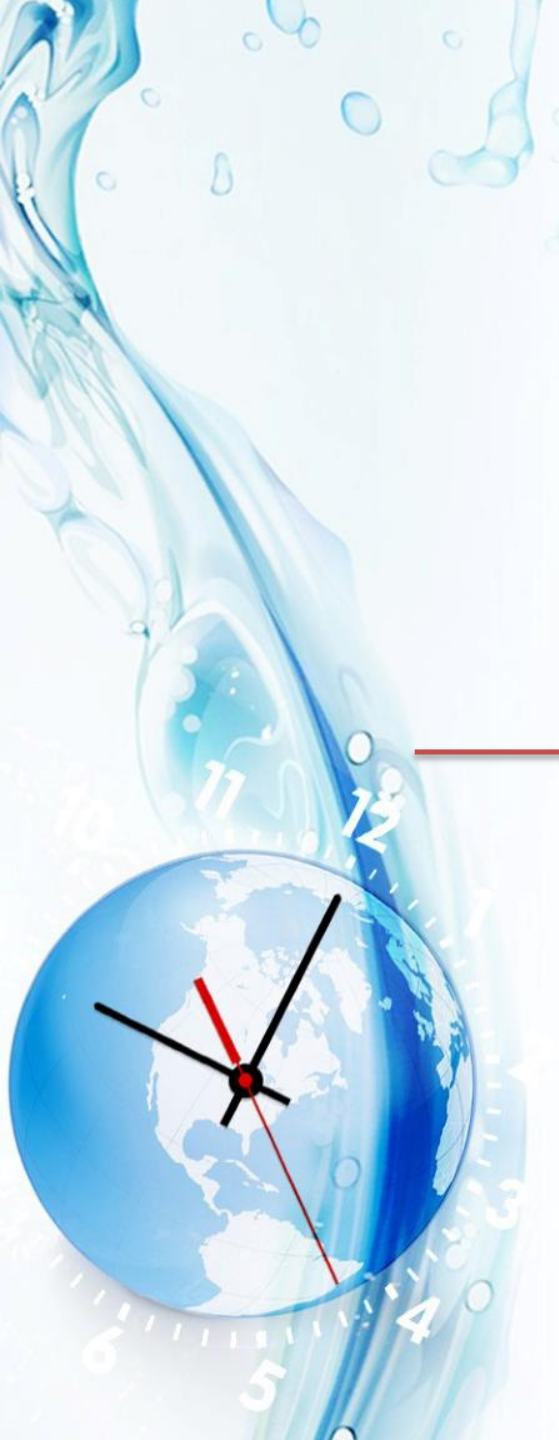
  The content is clipped on the top and/or bottom edges and that no scroll bar should be provided.

- *auto*

  Adds a vertical scroll bar when needed.

- *scroll*

  Adds the vertical scroll bar in all cases.



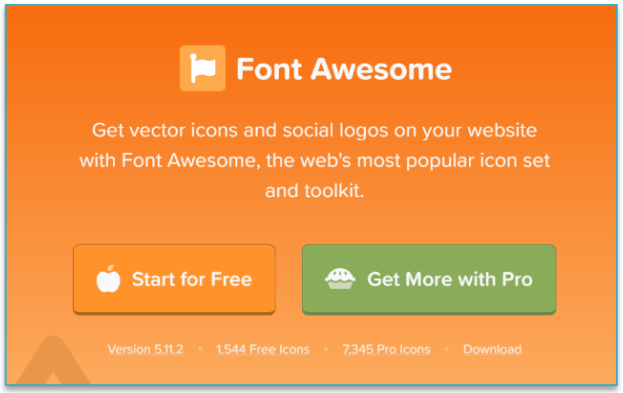
Adding Font Awesome Icons

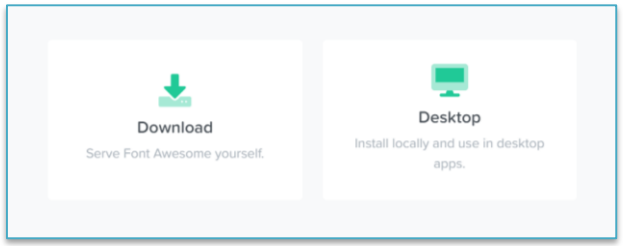
Used to Get vector icons and social logos for the web page.

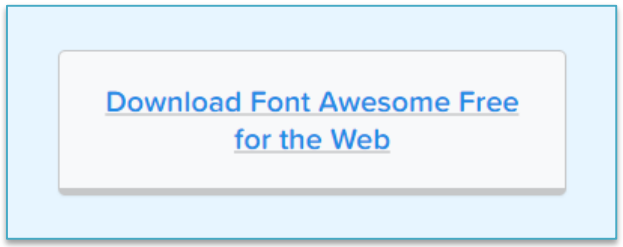
Adding Font Awesome Icons

- *Visit: https://fontawesome.com/*
- *Select "Start for free" button.*
- *Select "Download Serve Font Awesome yourself" button.*
- *Download "Font Awesome Free for the web" package.*
- *Unzip the package inside the CSS folder for your website.*
- *Add this tag to your html page:*

<link href="css/all.css" rel="stylesheet" type="text/css">

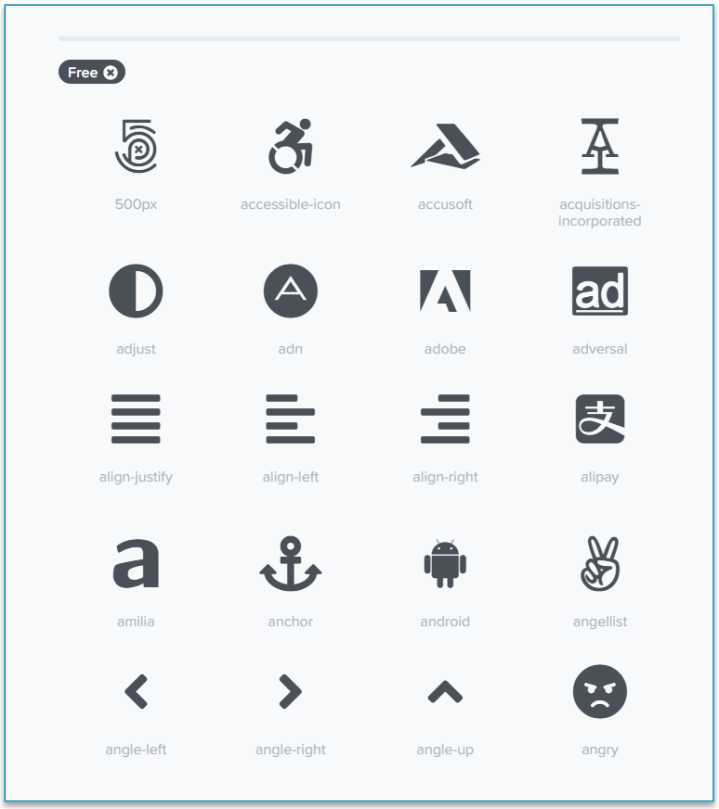


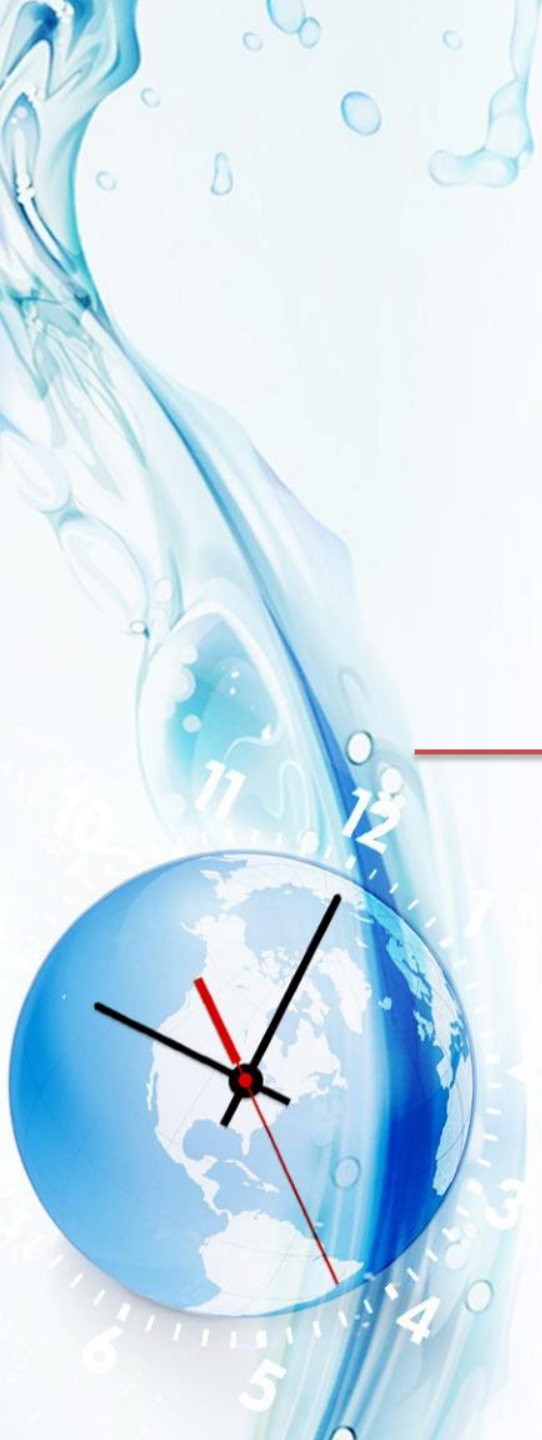




Adding Font Awesome Icons

- *To add an icon select "icon" from the top nav bar and "free" from the left nav bar.*
- *Select the icon from the icon list.*
- *Copy the icon code and paste it inside your HTML.*
- *Ex:*
  - **
  - **
  - **





CSS3 Animation

The animation in CSS is used to animate many CSS properties such as color, background-color, height, and width. Each animation needs to be defined with the @keyframes at-rule which is then called with the animation property.

@keyframes Rule

Description:

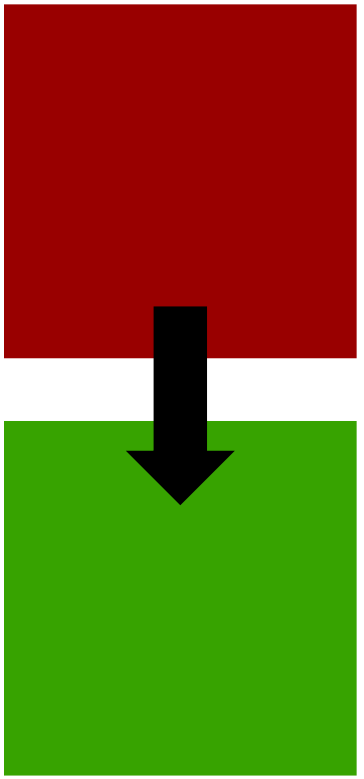
The @keyframes CSS at-rule is used to specify the intermediate steps during the cycle of an animation sequence by establishing the keyframes along the animation sequence.

A @keyframes rule consists of the keyword "@keyframes", followed by an identifier giving a name for the animation (which will be referenced using animation-name property), followed by a set of style rules (delimited by curly braces).

Also, The keyframe selector for a keyframe style rule starts with a percentage (%) or the keywords from (same as 0%) or to (same as 100%). The selector is used to specify where a keyframe is constructed along the duration of the animation.

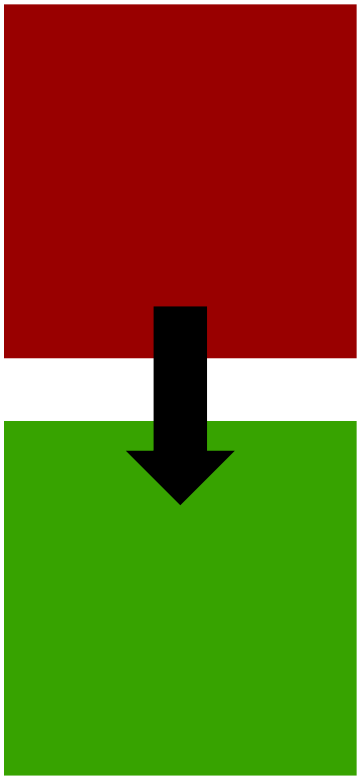
Example 1:

```
@keyframes animation1 {
        from {background-color: #900;}
        to {background-color: #0F0;}
}
div {
        width: 200px;
        height: 200px;
        background-color: #900;
        animation-name: animation1;
        animation-duration: 4s;
}
```



Example 2:

```
@keyframes animation1 {
        0% {background-color: #900;}
        100% {background-color: #0F0;}
}
div {
        width: 200px;
        height: 200px;
        background-color: #900;
        animation-name: animation1;
        animation-duration: 4s;
}
```



Example 2:

```
@keyframes animation1 {
        0% {background-color: #900;}
        50% {background-color: #0F0;}
        75% {background-color: #00F;}
        100% {background-color: #000;}
}
div {
        width: 200px;
        height: 200px;
        background-color: #900;
        animation-name: animation1;
        animation-duration: 4s;
}
```

Animation Sub-properties

animation-name

Description: declares the name of the @keyframes at-rule to manipulate.

Default value: none

Values: animation-name: keyframename | none;

animation-duration:

Description: the length of time it takes for an animation to complete one cycle.

Default value: 0

Values: time in seconds (s) or milliseconds (ms)

Animation Sub-properties

animation-timing-function:

Description: establishes preset acceleration curves such as ease or linear.

Default value: ease

Values: animation-timing-function: ease | linear | ease-in | ease-out | ease-in-out | step-start | step-end;

animation-delay:

Description: the time between the element being loaded and the start of the animation sequence.

Default value: 0s

Values: time in seconds (s) or milliseconds (ms);

animation-direction:

Description: sets the direction of the animation after the cycle.

Default value: normal

- *Values:*

  normal: The animation should play forward in each cycle

  reverse: The animation should play backward in each cycle

  Alternate: The animation is played forwards first, then backwards

  alternate-reverse: The animation is played backwards first, then forwards

Animation Sub-properties

animation-iteration-count:

Description: the number of times the animation should be performed.

Default value: 1

Values: number | infinite

animation-play-state:

Description: specifies whether an animation is playing or paused.

You can use the animation-play-state CSS property in combination with the JavaScript to pause an animation in the middle of a cycle.

Default value: running

Values: paused | running

animation-fill-mode:

Description: sets which values are applied after the animation. For example, you can set the last state of the animation to remain on screen, or you can set it to switch back to before when the animation began.

Default value: animation-fill-mode: none

Values:

none: delay ➔ first-keyframe ➔animation ➔last keyframe ➔default state

forwards : delay ➔ first-keyframe ➔animation ➔last keyframe

backwards: first-keyframe ➔ delay ➔animation ➔last keyframe ➔default state

both: first-keyframe ➔ delay ➔animation ➔last keyframe

Animation Shorthand:

Description: The animation property is a shorthand property for setting all the animation properties.

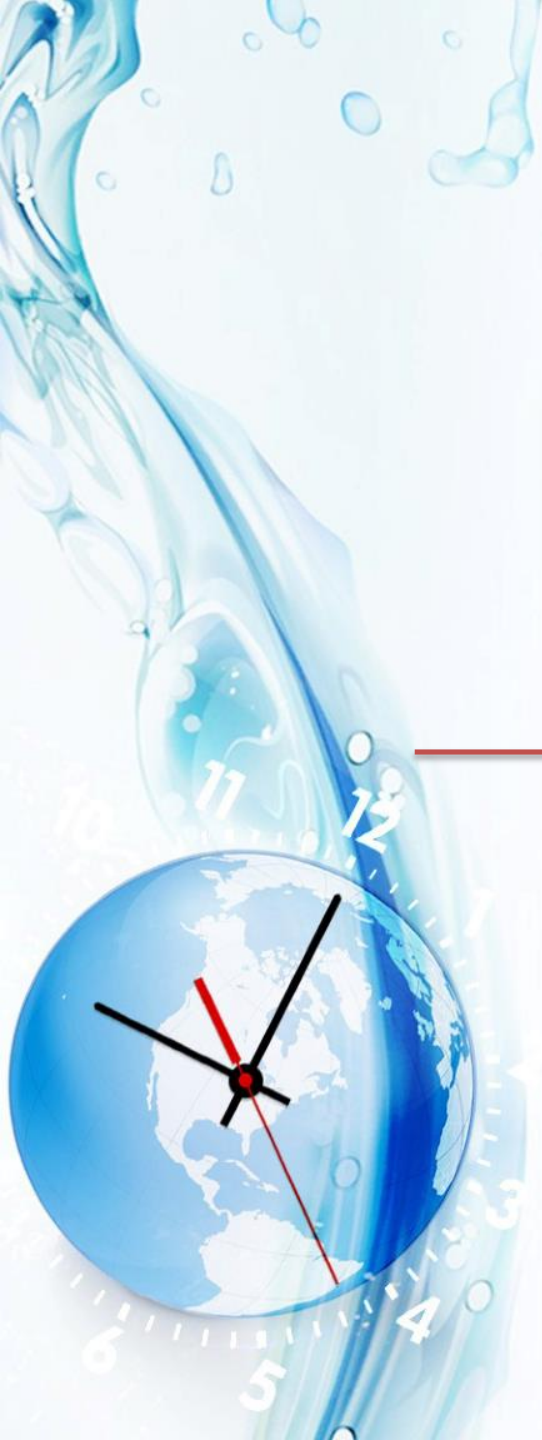
- Syntax:

  animation: [name] [duration] [timing-function] [delay] [iteration-count] [direction] [fill-mode] [play-state];

- two animation definitions:

  animation: [name] [duration] [timing-function] [delay] [iteration-count] [direction] [fill-mode] [play-state],

  [name] [duration] [timing-function] [delay] [iteration-count] [direction] [fill-mode] [play-state];



CSS3 2D Transforms

CSS3 2D transform performs a basic transform manipulations such as move, rotate, scale and skew on elements in a two-dimensional space.

A transformed element doesn't affect the surrounding elements, but can overlap them, just like the absolutely positioned elements.

CSS3 translate

Description:

Moves the element from its current position to a new position along the X and Y axes.

Property Name:

transform: translate(x, y);

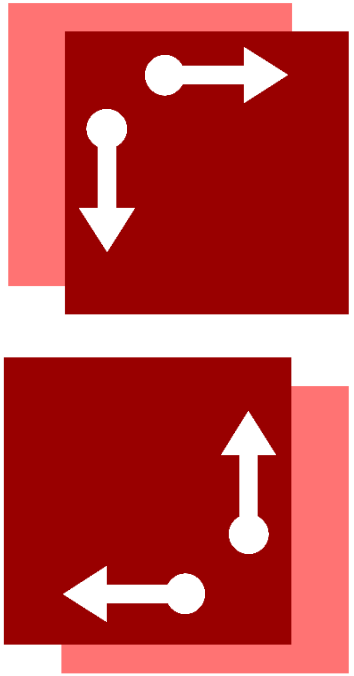
Examples:

transform: translate(40px, 20px);

- ➔ *Moves the element 40px horizontally to the left, and 20px vertically to the bottom.*

transform: translate(-40px, -20px);

- ➔ *Moves the element 40px horizontally to the right, and 20px vertically to the top.*



Description:

Rotates the element around its origin by a specified angle.

Property Name:

transform: rotate(angle);

Examples:

transform: rotate(20deg);

➔ *Rotates the element 20 degrees. (clockwise).*

transform: rotate(-20deg);

➔ *Rotates the element 20 degrees. (anti-clockwise).*

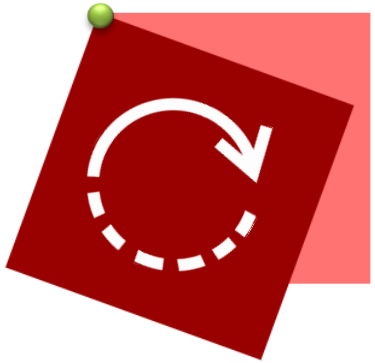
CSS3 transform-origin

Description:

Establishs the origin of transformation for an element.

Property Name:

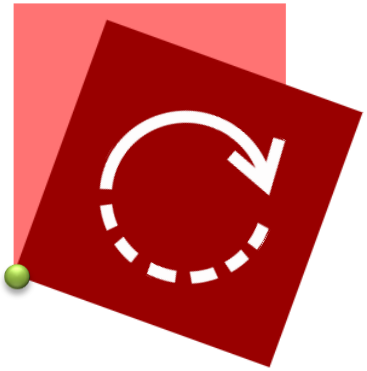
transform-origin: x-position | y-position;



Example1:

transform-origin: top left;

transform: rotate(20deg);



Example2:

transform-origin: bottom left;

transform: rotate(20deg);

Example2:

transform-origin: 30px 30px;

transform: rotate(20deg);

CSS3 scale

Description:

increases or decreases the size of the element using ratio to the original size.

Property Name:

transform: scale(width, height);

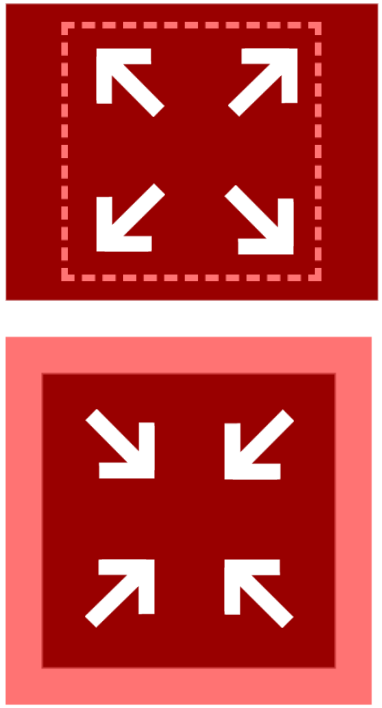
Example:

transform: scale(1.5, 1.2);

- ➔ *Increases the width to 150% and the height to 120%.*

transform: scale(0.8, 0.8);

- ➔ *Decreases the width to 80% and the height to 80%.*
- ➔ *Same as transform: scale(0.8);*



Description:

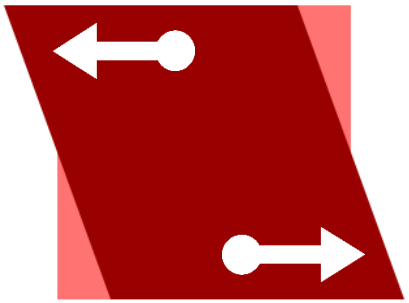
Skews an element along X and Y axes by a specified angles.

Property Name:

transform: skew(x, y);

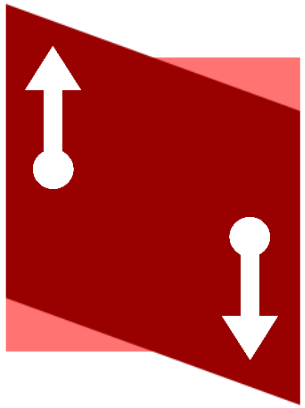
Example 1:

transform: skew(20deg, 0);



Example 2:

transform: skew(0, 20deg);



Other 2D transformation functions.

transform: translateX(100px)

➔ Moves the element by 100px along the X-axis.

transform: translateY(100px)

➔ Moves the element by 100px along the Y-axis.

transform: scaleX (1.5)

➔ Scale the width of the element to 150%.

transform: scaleY (1.2)

➔ Scale the height of the element to 120%.

transform: skewX (10deg)

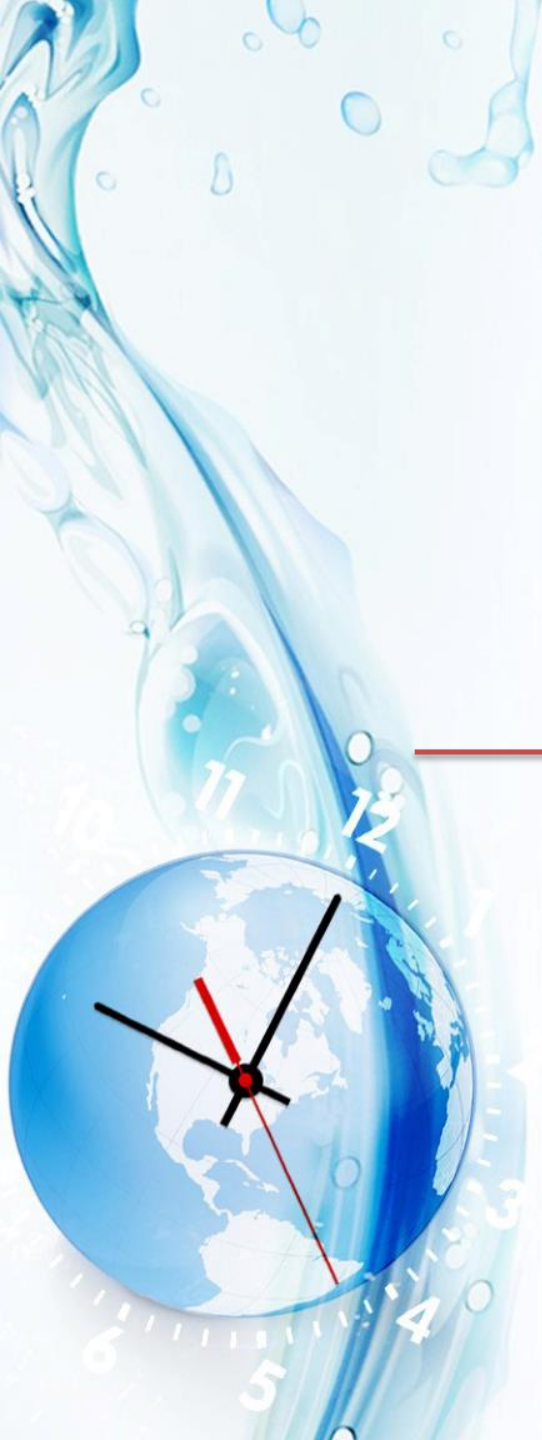
➔ Skews the element by 10 degrees along the X-axis.

transform: skewY (10deg)

➔ Skews the element by 10 degrees along the Y-axis.

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

➔ Shorthand for setting all 2D transform methods.



CSS3 3D Transforms

CSS3 3D transform can perform basic transform manipulations such as move, rotate and scale elements in a three-dimensional space.

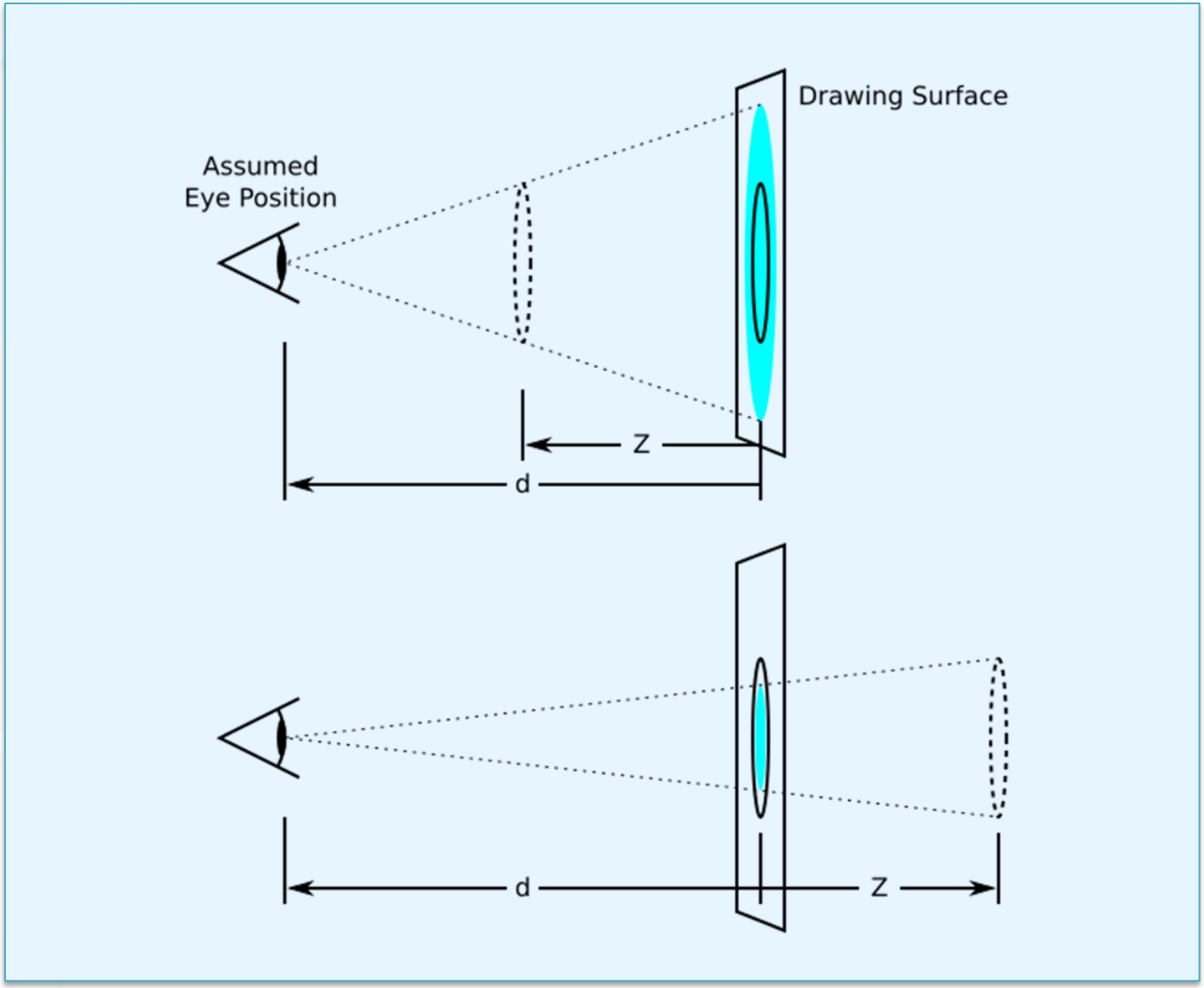
The transformed element doesn't affect the surrounding elements, but can overlap them.

Description:

- *This property is used to activate the three-dimensional space on an element so that its children can be positioned in that space.*
- *It allows to add a feeling of depth to a scene, by making elements higher on the z-axis (closer to the viewer) appear larger, and those further away to appear smaller.*
- *Without specifying a perspective, an element that is transformed using a three-dimensional transform function will look flat and two-dimensional.*

Property Name:

perspective: none | <length>


Drawing Surface
Assumed
Eye Position
Z
d
d
Z

Description:

Moves the element from its current position to a new position along the X,Y and Z axes.

Property Name:

- *transform: translateX(<number>);*

  Translates an element along the x-axis in three-dimensional space.
- *transform: translateY(<number>);*

  Translates an element along the Y-axis in three-dimensional space.
- *transform: translateZ(<number>);*

  Translates an element along the X-axis in three-dimensional space.
- *transform: translate3d(X, Y, Z);*

  Shorthand used to translate an element along the x-axis, y-axis and z-axis in three-dimensional space.

3D Translate

Examples:

transform: translate3d(40px, 20px, 10px);

➔ *Moves the element 40px horizontally to the left, 20px vertically to the bottom and 10px forwards.*

transform: translate3d(-40px, -20px, -10px);

➔ *Moves the element 40px horizontally to the right, 20px vertically to the top and 10px backwards.*

transform: translateZ(100px);

➔ *Moves the 100px forwards.*

transform: translateX(100px);

➔ *Moves the element 40px horizontally to the right*

transform: translateY(-30px);

➔ *Moves the element 30px vertically to the top*

Note: to use the third dimension "Z" the "perspective" property must be set to the parent.

3D Rotate

Description:

used to rotate an element in three-dimensional space.

Property Name:

- transform: rotateX(<angle>);

  used to rotate an element around the x-axis in three-dimensional space.

- transform: rotateY(<angle>);

  used to rotate an element around the y-axis in three-dimensional space.

- transform: rotateZ(<angle>);

  used to rotate an element around the z-axis in three-dimensional space.

- transform: rotate3d(X, Y, Z, <angle>);

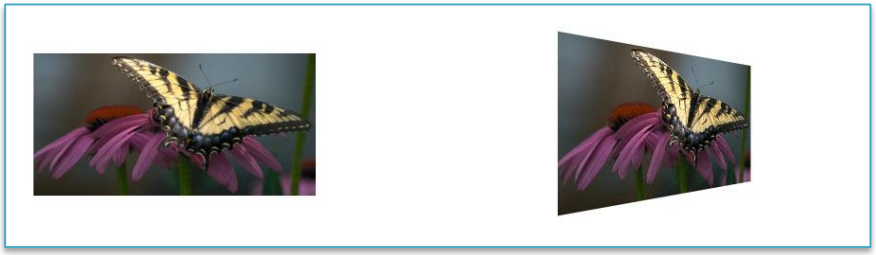
  Shorthand used to rotate an element around the x-axis and y-axis and z-axis in three-dimensional space.

3D Rotate

EX:

- *transform: rotateX(50deg);*

- *transform: rotateY(50deg);*



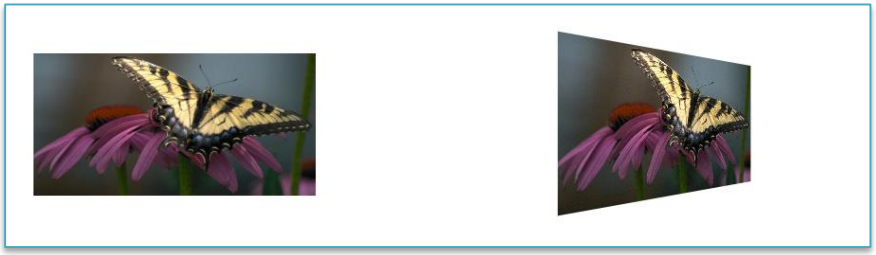
- *transform: rotateZ(50deg);*

3D Rotate

EX: same as

- *transform: rotate3d(1,0,0,50deg);*

- *transform: rotate3d(0,1,0,50deg);*



- *transform: rotate3d(0,0,1,50deg);*

3D Scale

Description:

Used to scale an element in three-dimensional space. It takes three numbers as a value, sx, sy, and sz, where "sx" will scale the element in the direction of the x-asis, "sy" will scale it in the direction of the y-axis, and "sz" will scale it in the direction of the z-axis.

Property Name:

transform: scale3d(sx,sy,sz);

Example:

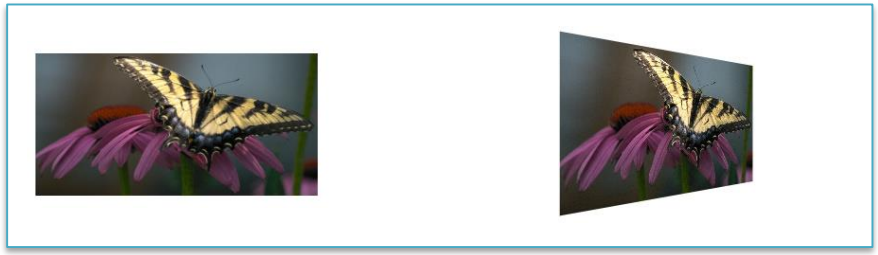
transform: scale3d(1.5, 1.2, 2);

- *You will notice that the "sx" and that "sy" values applied to the element, but the "sz" has no effect, why? Because nothing of the element is placed on the z-axis. So you need to combine another property as "rotate" with the "scale3d" property to force the element to pass through the z-axis.*
- *transform: scale3d(1.5, 1.2, 2) rotateY(60deg);*

3D Scale

- *transform:*
  scale3d(1.5, 1.5, 1.5);

- *transform:*
  rotateY(50deg);



- *transform:*
  scale3d(1.5, 1.5, 1.5)
  rotateY(50deg);

CSS3 transform-style

Description:

Determines whether the children of an element are positioned in three-dimensional space or flattened in the two-dimensional.

Property Name:

transform-style: flat | preserve-3d

Values:

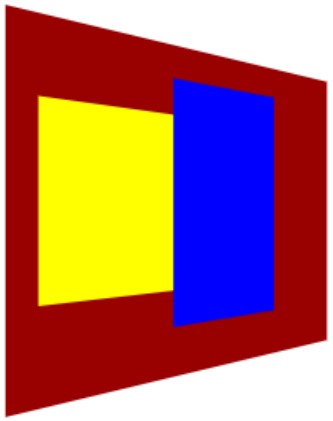
- *flat: (default) Flattens the children of the element so that they lie in the plane of the element itself.*
- *preserve-3d: Allows the children of the element to be positioned in the three-dimensional space.*

CSS3 transform-style

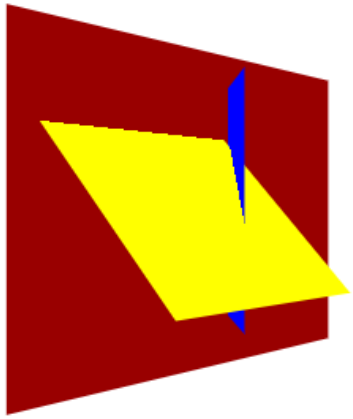
Example:

```
body { perspective: 500px; }
section.parent {
        width: 300px; height: 200px; margin: 150px auto;
        position: relative;    transition-duration: 0.8s;
        background: #900; transform: rotateY(50deg);
        transform-style: preserve-3d; }
div.div1, div.div2 {
        width: 150px; height: 150px;
        position: absolute; left: 20px; top:20px; }
div.div1 {
        background: yellow;
        transform: translateZ(40px) rotatex(45deg); }
div.div2 {
        background: #00f;
        transform: translatex(80px) rotateY(45deg);}
```

transform-style: flat



transform-style: preserve-3d



CSS3 backface-visibility

Description:

Determines whether or not the "back" side of a transformed element is visible when facing the viewer.

Property Name:

backface-visibility: visible | hidden

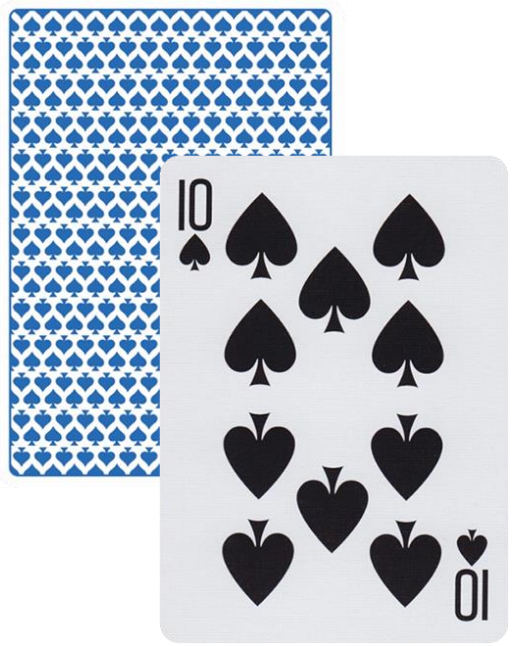
Values:

- *visible: Indicates that the back face of the element is visible, and the front face shows through it.*
- *hidden: Indicates that the back face is hidden and the front face does not show through it.*

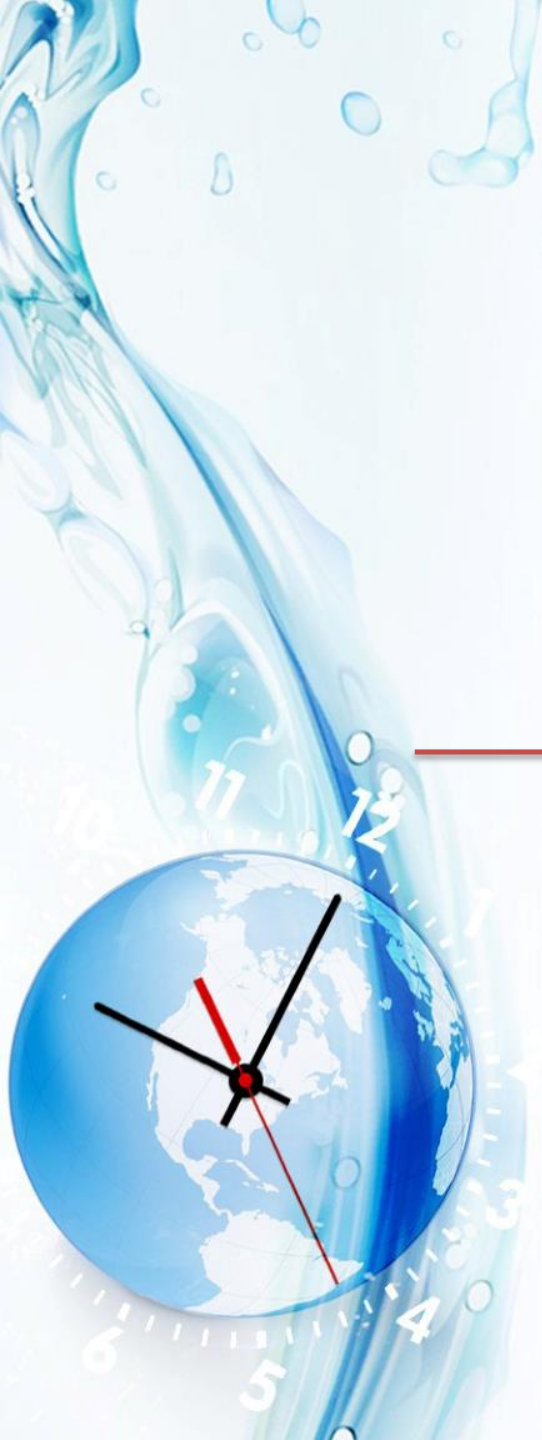
CSS3 transform-style

Example: Rotating Cards

```
body { padding-top: 100px; perspective: 1500px; }
section {
    width: 357px; height: 500px; margin: 0 auto;
    transform-style: preserve-3d;
    position: relative; transition: all 2s ease;}
section:hover { transform: rotateY(180deg); }
div {
    width: 357px;  height: 500px;
    position: absolute; left: 0; top:0; }
div.div1 {
    background: url("front.png") no-repeat;
    transform: rotateY(180deg); }
div.div2 {
    background: url("back.png") no-repeat;
    backface-visibility: hidden; }
```



```
<body>
    <section>
    <div class="div1"></div>
    <div class="div2"></div>
    </section>
</body>
```



Multiple Columns

With just a few CSS rules, you can create a print-inspired layout that has the flexibility of the web. It's like taking a newspaper, but as the paper gets smaller, the columns will adjust and balance automatically allowing the content to flow naturally.

column-count

Description:

Specifies the number of columns in a multi-column element.

Property:

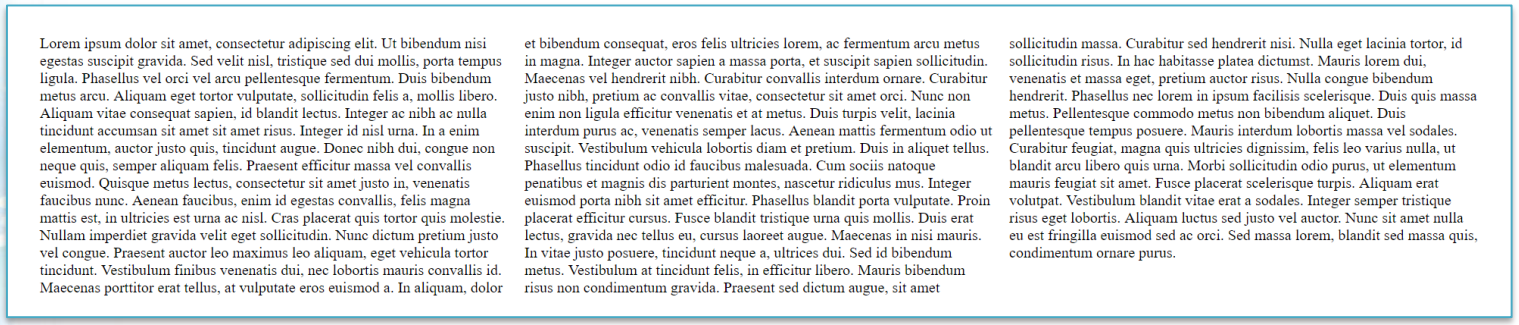
column-count: positive number | auto;

Default value:

column-count: auto;

Example:

```
div {
    column-count: 3;
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut bibendum nisi egestas suscipit gravida. Sed velit nisl, tristique sed dui mollis, porta tempus ligula. Phasellus vel orci vel arcu pellentesque fermentum. Duis bibendum metus arcu. Aliquam eget tortor vulputate, sollicitudin felis a, mollis libero. Aliquam vitae consequat sapien, id blandit lectus. Integer ac nibh ac nulla tincidunt accumsan sit amet sit amet risus. Integer id nisl urna. In a enim elementum, auctor justo quis, tincidunt augue. Donec nibh dui, congue non neque quis, semper aliquam felis. Praesent efficitur massa vel convallis euismod. Quisque metus lectus, consectetur sit amet justo in, venenatis faucibus nunc. Aenean faucibus, enim id egestas convallis, felis magna mattis est, in ultricies est urna ac nisl. Cras placerat quis tortor quis molestie. Nullam imperdiet gravida velit eget sollicitudin. Nunc dictum pretium justo vel congue. Praesent auctor leo maximus leo aliquam, eget vehicula tortor tincidunt. Vestibulum finibus venenatis dui, nec lobortis mauris convallis id. Maecenas porttitor erat tellus, at vulputate eros euismod a. In aliquam, dolor et bibendum consequat, eros felis ultricies lorem, ac fermentum arcu metus in magna. Integer auctor sapien a massa porta, et suscipit sapien sollicitudin. Maecenas vel hendrerit nibh. Curabitur convallis interdum ornare. Curabitur justo nibh, pretium ac convallis vitae, consectetur sit amet orci. Nunc non enim non ligula efficitur venenatis et at metus. Duis turpis velit, lacinia interdum purus ac, venenatis semper lacus. Aenean mattis fermentum odio ut suscipit. Vestibulum vehicula lobortis diam et pretium. Duis in aliquet tellus. Phasellus tincidunt odio id faucibus malesuada. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer euismod porta nibh sit amet efficitur. Phasellus blandit porta vulputate. Proin placerat efficitur cursus. Fusce blandit tristique urna quis mollis. Duis erat lectus, gravida nec tellus eu, cursus laoreet augue. Maecenas in nisi mauris. In vitae justo posuere, tincidunt neque a, ultrices dui. Sed id bibendum metus. Vestibulum at tincidunt felis, in efficitur libero. Mauris bibendum risus non condimentum gravida. Praesent sed dictum augue, sit amet sollicitudin massa. Curabitur sed hendrerit nisi. Nulla eget lacinia tortor, id sollicitudin risus. In hac habitasse platea dictumst. Mauris lorem dui, venenatis et massa eget, pretium auctor risus. Nulla congue bibendum hendrerit. Phasellus nec lorem in ipsum facilisis scelerisque. Duis quis massa metus. Pellentesque commodo metus non bibendum aliquet. Duis pellentesque tempus posuere. Mauris interdum lobortis massa vel sodales. Curabitur feugiat, magna quis ultricies dignissim, felis leo varius nulla, ut blandit arcu libero quis urna. Morbi sollicitudin odio purus, ut elementum mauris feugiat sit amet. Fusce placerat scelerisque turpis. Aliquam erat volutpat. Vestibulum blandit vitae erat a sodales. Integer semper tristique risus eget lobortis. Aliquam luctus sed justo vel auctor. Nunc sit amet nulla eu est fringilla euismod sed ac orci. Sed massa lorem, blandit sed massa quis, condimentum ornare purus.

column-gap

Description:

Specifies the gap between the columns in a multi-column element.

Property:

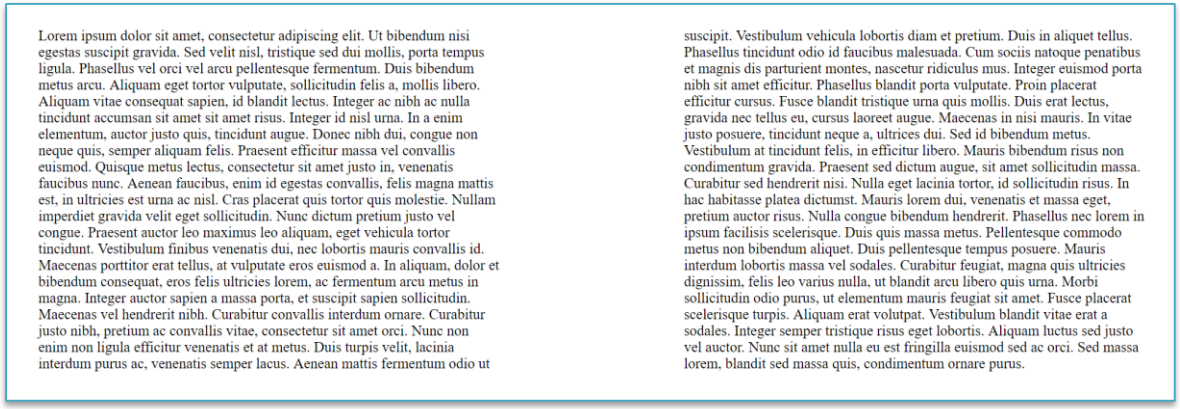
column-gap: length | normal

Default value:

column-gap: normal;

Example:

```
div {
    column-count: 2; column-gap: 200px;
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut bibendum nisi egestas suscipit gravida. Sed velit nisl, tristique sed dui mollis, porta tempus ligula. Phasellus vel orci vel arcu pellentesque fermentum. Duis bibendum metus arcu. Aliquam eget tortor vulputate, sollicitudin felis a, mollis libero. Aliquam vitae consequat sapien, id blandit lectus. Integer ac nibh ac nulla tincidunt accumsan sit amet sit amet risus. Integer id nisl urna. In a enim elementum, auctor justo quis, tincidunt augue. Donec nibh dui, congue non neque quis, semper aliquam felis. Praesent efficitur massa vel convallis euismod. Quisque metus lectus, consectetur sit amet justo in, venenatis faucibus nunc. Aenean faucibus, enim id egestas convallis, felis magna mattis est, in ultricies est urna ac nisl. Cras placerat quis tortor quis molestie. Nullam imperdiet gravida velit eget sollicitudin. Nunc dictum pretium justo vel congue. Praesent auctor leo maximus leo aliquam, eget vehicula tortor tincidunt. Vestibulum finibus venenatis dui, nec lobortis mauris convallis id. Maecenas porttitor erat tellus, at vulputate eros euismod a. In aliquam, dolor et bibendum consequat, eros felis ultricies lorem, ac fermentum arcu metus in magna. Integer auctor sapien a massa porta, et suscipit sapien sollicitudin. Maecenas vel hendrerit nibh. Curabitur convallis interdum ornare. Curabitur justo nibh, pretium ac convallis vitae, consectetur sit amet orci. Nunc non enim non ligula efficitur venenatis et at metus. Duis turpis velit, lacinia interdum purus ac, venenatis semper lacus. Aenean mattis fermentum odio ut suscipit. Vestibulum vehicula lobortis diam et pretium. Duis in aliquet tellus. Phasellus tincidunt odio id faucibus malesuada. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer euismod porta nibh sit amet efficitur. Phasellus blandit porta vulputate. Proin placerat efficitur cursus. Fusce blandit tristique urna quis mollis. Duis erat lectus, gravida nec tellus eu, cursus laoreet augue. Maecenas in nisi mauris. In vitae justo posuere, tincidunt neque a, ultrices dui. Sed id bibendum metus. Vestibulum at tincidunt felis, in efficitur libero. Mauris bibendum risus non condimentum gravida. Praesent sed dictum augue, sit amet sollicitudin massa. Curabitur sed hendrerit nisi. Nulla eget lacinia tortor, id sollicitudin risus. In hac habitasse platea dictumst. Mauris lorem dui, venenatis et massa eget, pretium auctor risus. Nulla congue bibendum hendrerit. Phasellus nec lorem in ipsum facilisis scelerisque. Duis quis massa metus. Pellentesque commodo metus non bibendum aliquet. Duis pellentesque tempus posuere. Mauris interdum lobortis massa vel sodales. Curabitur feugiat, magna quis ultricies dignissim, felis leo varius nulla, ut blandit arcu libero quis urna. Morbi sollicitudin odio purus, ut elementum mauris feugiat sit amet. Fusce placerat scelerisque turpis. Aliquam erat volutpat. Vestibulum blandit vitae erat a sodales. Integer semper tristique risus eget lobortis. Aliquam luctus sed justo vel auctor. Nunc sit amet nulla eu est fringilla euismod sed ac orci. Sed massa lorem, blandit sed massa quis, condimentum ornare purus.

column-span

Description:

Specifies whether an element can spans across (ignoring the columns and appears in all width) in a multi-column layout or not.

Property:

column-span: none | all ;

Example:

```
div { column-count: 3; }
h1 { column-span: all; }
```

This is the head line that must be spanned over all the column

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut bibendum nisi egestas suscipit gravida. Sed velit nisl, tristique sed dui mollis, porta tempus ligula. Phasellus vel orci vel arcu pellentesque fermentum. Duis bibendum metus arcu. Aliquam eget tortor vulputate, sollicitudin felis a, mollis libero. Aliquam vitae consequat sapien, id blandit lectus. Integer ac nibh ac nulla tincidunt accumsan sit amet sit amet risus. Integer id nisl urna. In a enim elementum, auctor justo quis, tincidunt augue. Donec nibh dui, congue non neque quis, semper aliquam felis. Praesent efficitur massa vel convallis euismod. Quisque metus lectus, consectetur sit amet justo in, venenatis faucibus nunc. Aenean faucibus, enim id egestas convallis, felis magna mattis est, in ultricies est urna ac nisl. Cras placerat quis tortor quis molestie. Nullam imperdiet gravida velit eget sollicitudin. Nunc dictum pretium justo vel congue.

Praesent auctor leo maximus leo aliquam, eget vehicula tortor tincidunt. Vestibulum finibus venenatis dui, nec lobortis mauris convallis id. Maecenas porttitor erat tellus, at vulputate eros euismod a. In aliquam, dolor et bibendum consequat, eros felis ultricies lorem, ac fermentum arcu metus in magna. Integer auctor sapien a massa porta, et suscipit sapien sollicitudin. Maecenas vel hendrerit nibh. Curabitur convallis interdum ornare. Curabitur justo nibh, pretium ac convallis vitae, consectetur sit amet orci. Nunc non enim non ligula efficitur venenatis et at metus. Duis turpis velit, lacinia interdum purus ac, venenatis semper lacus. Aenean mattis fermentum odio ut suscipit. Vestibulum vehicula lobortis diam et pretium. Duis in aliquet tellus. Phasellus tincidunt odio id faucibus malesuada. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer euismod porta nibh sit amet efficitur. Phasellus blandit porta vulputate. Proin placerat efficitur cursus. Fusce blandit tristique urna quis mollis. Duis erat lectus, gravida nec tellus eu, cursus laoreet augue. Maecenas in nisi mauris. In vitae justo posuere, tincidunt neque a, ultrices dui. Sed id bibendum metus. Vestibulum at tincidunt felis, in efficitur libero. Mauris bibendum risus non condimentum gravida. Praesent sed dictum augue, sit amet sollicitudin massa. Curabitur sed hendrerit nisi. Nulla eget lacinia tortor, id sollicitudin risus. In hac habitasse platea dictumst. Mauris lorem dui, venenatis et massa eget, pretium auctor risus. Nulla congue bibendum hendrerit. Phasellus nec lorem in ipsum facilisis scelerisque. Duis quis massa metus. Pellentesque commodo metus non bibendum aliquet. Duis pellentesque tempus posuere. Mauris interdum lobortis massa vel sodales.

Description:

Specifies how the column lengths (heights) are affected by the content flow.

Property:

column-span: balance | auto

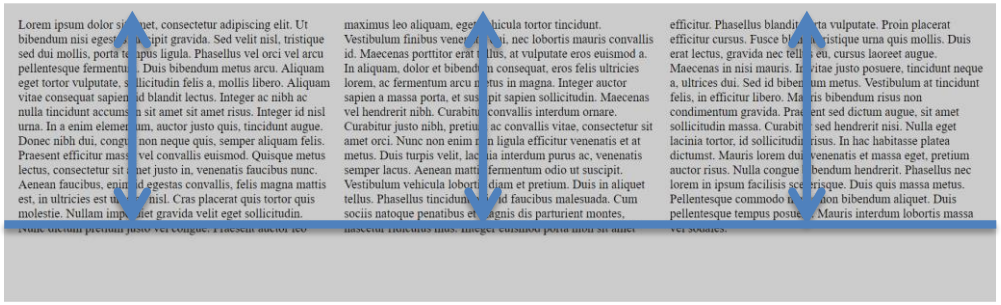
Values:

balance: default, fills all the columns with the same level with data, ignoring the container's height.

auto: fill the columns one by one to reach the container height respectively

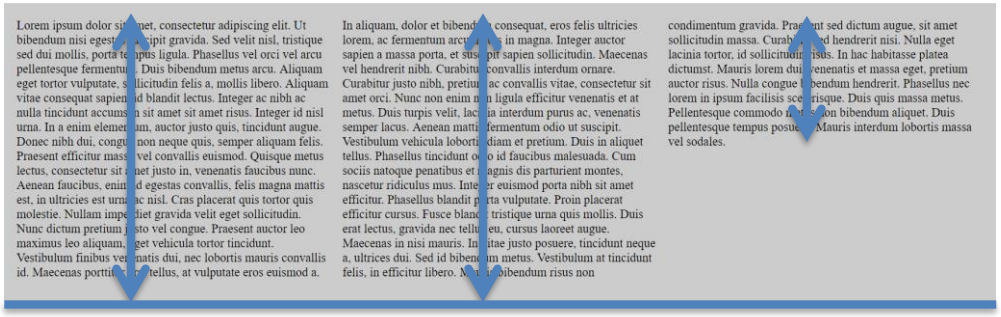
Example (column-fill: balance):

```
div { column-count: 3;
height: 350px;
background-color:#ccc;
column-fill: balance; }
h1 { column-span: all; }
```



Example (column-fill: auto):

```
div { column-count: 3;
height: 350px;
background-color:#ccc;
column-fill: auto; }
h1 { column-span: all; }
```



Description:

Determines the minimum column width, if the browser cannot fit at least two columns using this width then the columns will stop and drop into a single column.

Property:

column-width: auto | width in px

Example:

```
div {
    column-count: 3;
    column-width: 450px;
}
```

Description:

is a shorthand property for setting both the column-width and the column-count properties at the same time.

Property:

columns: column-width column-count

Default Vaule:

columns: auto auto;

Example:

```
div {
        columns: 450px 3;
}
```

column-rule-style

Description:

Sets the style of the rule (vertical line) drawn between the columns in a multi-column layout.

Property:

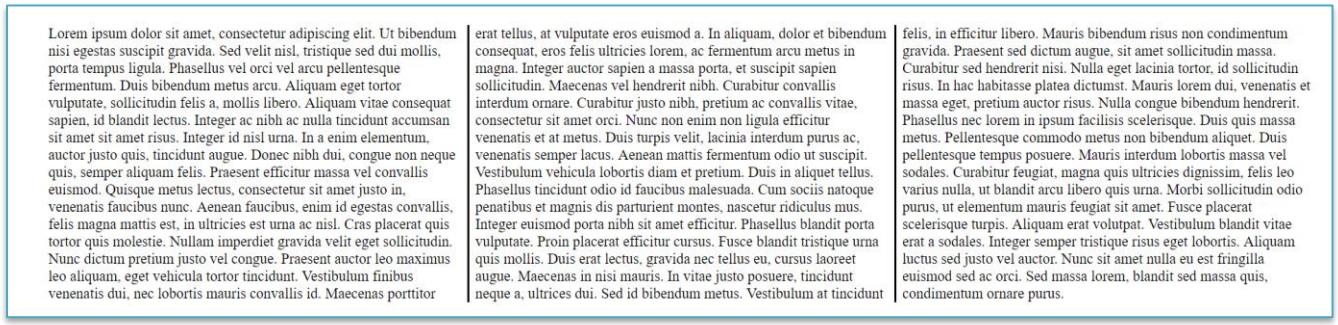
column-rule-style: none | hidden | dotted | dashed | solid | double

Default value:

column-rule-style: none

Example:

```
div {
    column-count: 3; column-rule-style: solid;
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut bibendum nisi egestas suscipit gravida. Sed velit nisl, tristique sed dui mollis, porta tempus ligula. Phasellus vel orci vel arcu pellentesque fermentum. Duis bibendum metus arcu. Aliquam eget tortor vulputate, sollicitudin felis a, mollis libero. Aliquam vitae consequat sapien, id blandit lectus. Integer ac nibh ac nulla tincidunt accumsan sit amet sit amet risus. Integer id nisl urna. In a enim elementum, auctor justo quis, tincidunt augue. Donec nibh dui, congue non neque quis, semper aliquam felis. Praesent efficitur massa vel convallis euismod. Quisque metus lectus, consectetur sit amet justo in, venenatis faucibus nunc. Aenean faucibus, enim id egestas convallis, felis magna mattis est, in ultricies est urna ac nisl. Cras placerat quis tortor quis molestie. Nullam imperdiet gravida velit eget sollicitudin. Nunc dictum pretium justo vel congue. Praesent auctor leo maximus leo aliquam, eget vehicula tortor tincidunt. Vestibulum finibus venenatis dui, nec lobortis mauris convallis id. Maecenas porttitor erat tellus, at vulputate eros euismod a. In aliquam, dolor et bibendum consequat, eros felis ultricies lorem, ac fermentum arcu metus in magna. Integer auctor sapien a massa porta, et suscipit sapien sollicitudin. Maecenas vel hendrerit nibh. Curabitur convallis interdum ornare. Curabitur justo nibh, pretium ac convallis vitae, consectetur sit amet orci. Nunc non enim non ligula efficitur venenatis et at metus. Duis turpis velit, lacinia interdum purus ac, venenatis semper lacus. Aenean mattis fermentum odio ut suscipit. Vestibulum vehicula lobortis diam et pretium. Duis in aliquet tellus. Phasellus tincidunt odio id faucibus malesuada. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer euismod porta nibh sit amet efficitur. Phasellus blandit porta vulputate. Proin placerat efficitur cursus. Fusce blandit tristique urna quis mollis. Duis erat lectus, gravida nec tellus eu, cursus laoreet augue. Maecenas in nisi mauris. In vitae justo posuere, tincidunt neque a, ultrices dui. Sed id bibendum metus. Vestibulum at tincidunt felis, in efficitur libero. Mauris bibendum risus non condimentum gravida. Praesent sed dictum augue, sit amet sollicitudin massa. Curabitur sed hendrerit nisi. Nulla eget lacinia tortor, id sollicitudin risus. In hac habitasse platea dictumst. Mauris lorem dui, venenatis et massa eget, pretium auctor risus. Nulla congue bibendum hendrerit. Phasellus nec lorem in ipsum facilisis scelerisque. Duis quis massa metus. Pellentesque commodo metus non bibendum aliquet. Duis pellentesque tempus posuere. Mauris interdum lobortis massa vel sodales. Curabitur feugiat, magna quis ultricies dignissim, felis leo varius nulla, ut blandit arcu libero quis urna. Morbi sollicitudin odio purus, ut elementum mauris feugiat sit amet. Fusce placerat scelerisque turpis. Aliquam erat volutpat. Vestibulum blandit vitae erat a sodales. Integer semper tristique risus eget lobortis. Aliquam luctus sed justo vel auctor. Nunc sit amet nulla eu est fringilla euismod sed ac orci. Sed massa lorem, blandit sed massa quis, condimentum ornare purus.

column-rule-width

Description:

Sets the width of the rule (vertical line) drawn between the columns in a multi-column layout.

Property:

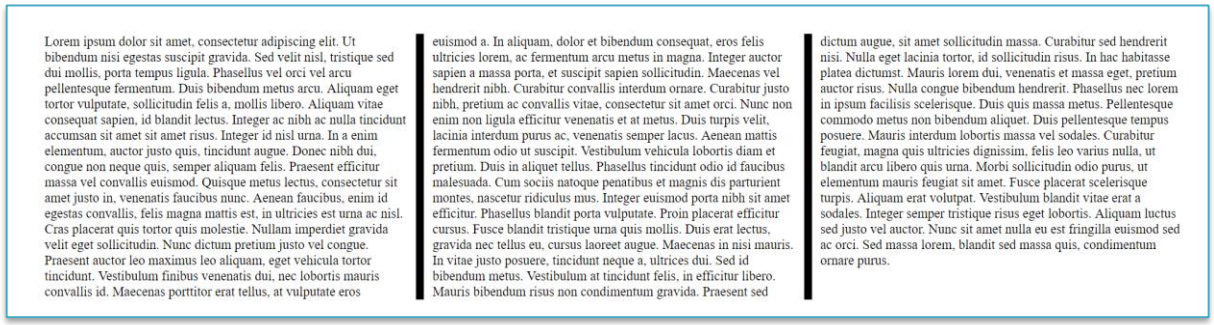
column-rule-width: length | medium | thin | thick;

Default value:

column-rule-width: medium;

Example:

```
div {
    column-count: 3; column-rule-width: 10px;
}
```



column-rule-color

Description:

Sets the color of all the rules drawn between columns in a multi-column layout.

Property:

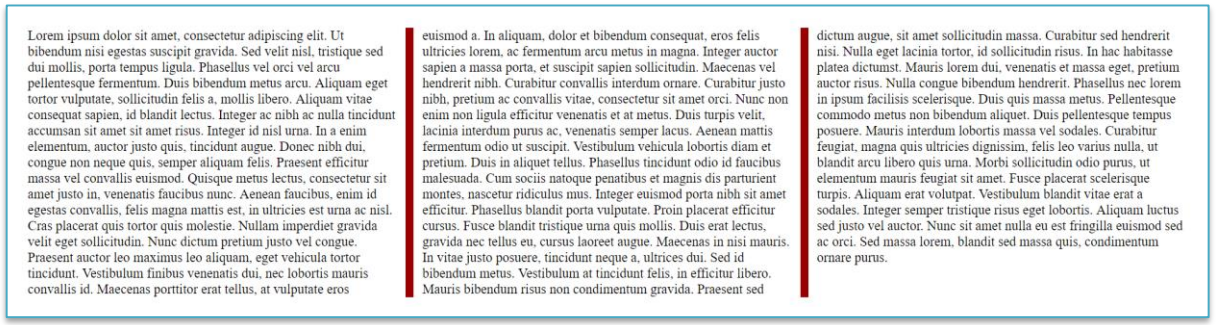
column-rule-color: color

Default value:

column-rule-color: current element color.

Example:

```
div {
    column-count: 3; column-rule-color:#900;
}
```



Description:

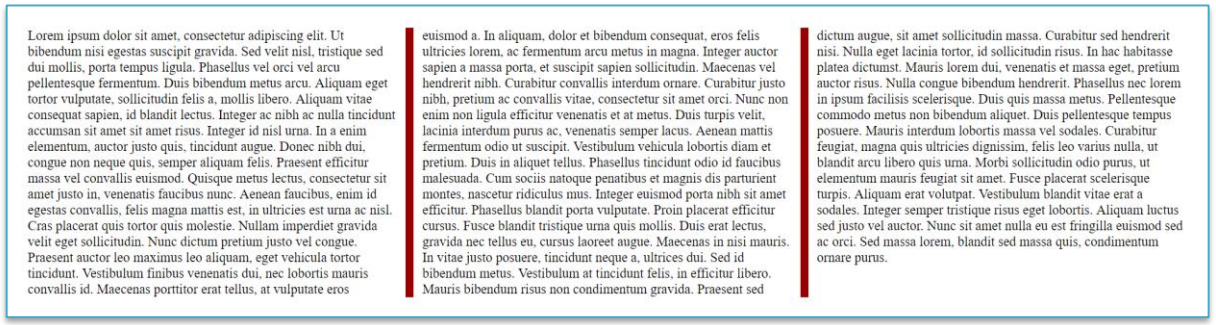
Shorthand property for setting “column-rule-width”, “column-rule-style” and “column-rule-color” at once.

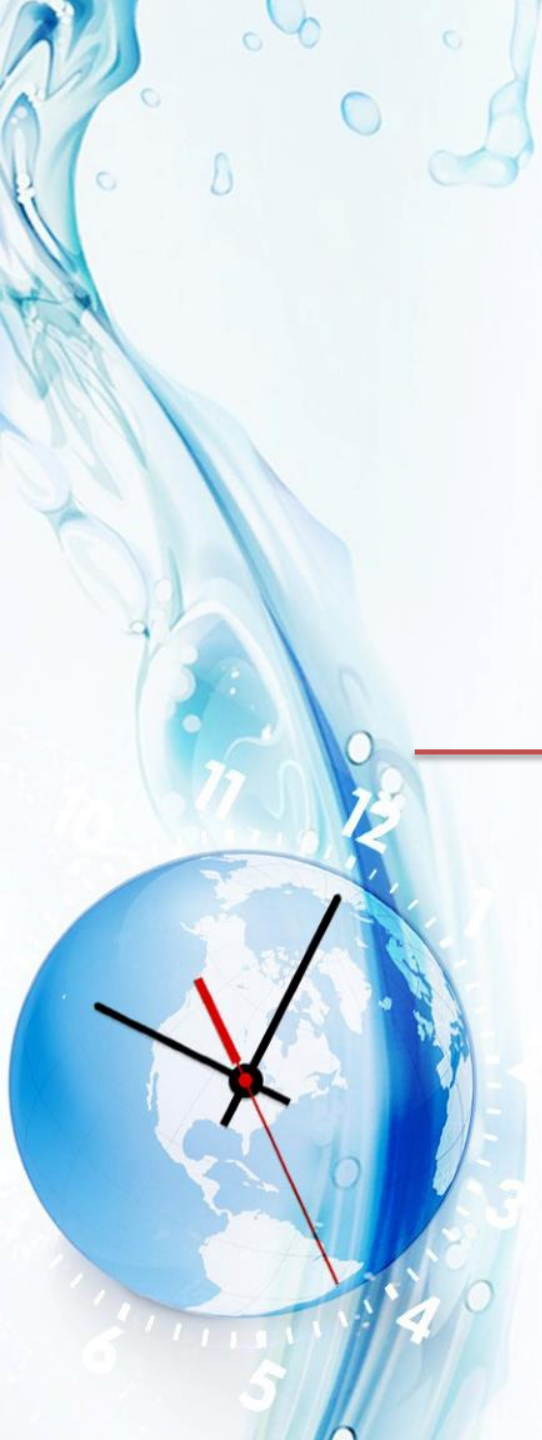
Property:

column-rule: column-rule-width column-rule-style column-rule-color

Example:

```
div {
    column-count: 3; column-rule: 10px solid #900;
}
```





CSS variables

Are entities defined by CSS that contain specific values to be reused throughout a document. They are set using custom property notation and are accessed using the var() function.

Declaring property

```
element {
    --variablename: value;
            }
```

Using property

```
:root {
            --mainbg: red;
            }
.red {
            background-color: var(--main-bg-color);
}
```

Custom property fallback values

Example 1

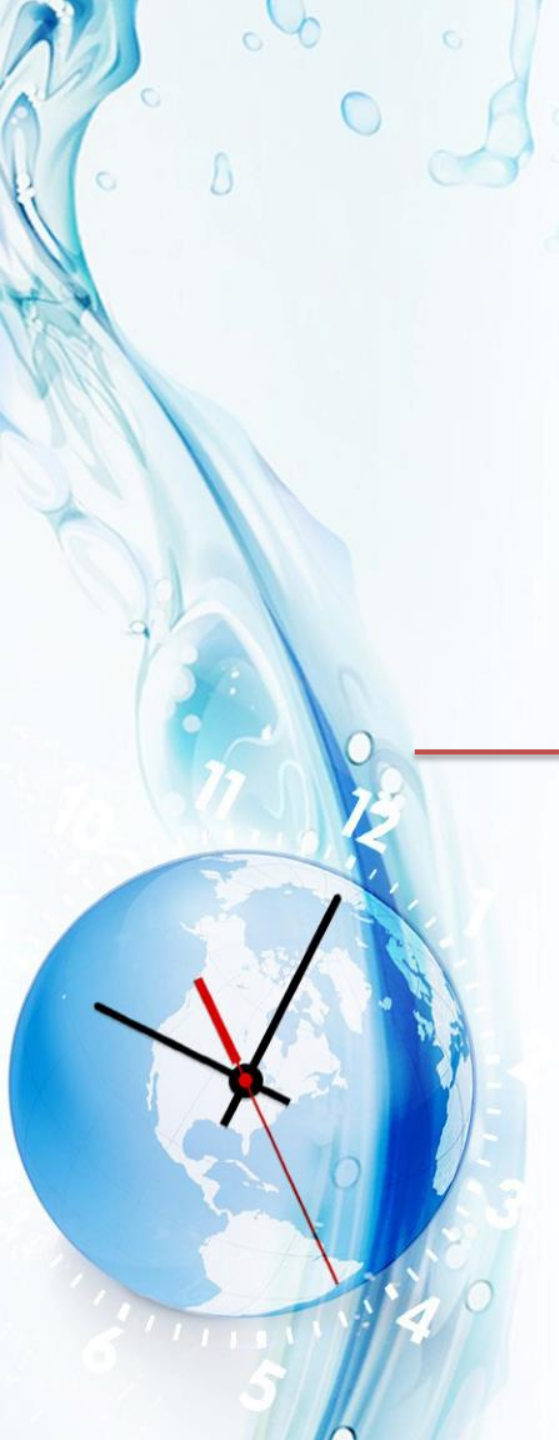
```
.red {
  color: var(--myvar, red);
}
```

➔ *Red if --myvar is not defined*

Example 2

```
.red {
  background-color: var(--myvar, var(--mybg, red));
}
```

➔ *Red if --myvar and –mybg is not defined*



CSS3 Flexbox

CSS3 Flexbox

used to make the elements behave predictably when they are used with different screen sizes, to provides more efficient way to layout, align and distribute space among items in the container.

The CSS3 flexbox contains:

Flex container: specifies the properties of the parent. It is declared by setting the display property of an element to either "flex" or "inline-flex".

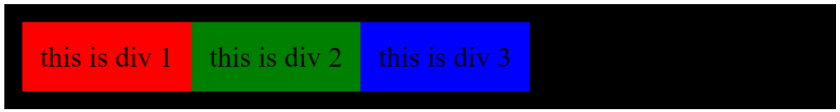
Flex items: specify properties of the children. There may be one or more flex items inside a flex container.

Flexbox Creation

Any container becomes flexible by setting the "display" property to: "flex" or "inline-flex".

Code:

```
<head>
<style>
    .box { display:flex; background-color:black; padding:10px; }
</style></head>
<body>
    <section class="box">
        <div style="background-color:red;"> this is div 1</div>
        <div style="background-color:green;"> this is div 2</div>
        <div style="background-color:blue;"> this is div 3</div>
    </section></body>
```



Flex Direction

Description:

The flex-direction property is used to set the direction of the flexible items inside the flex container. its default value is row (left-to-right, top-to-bottom).

Property:

flex-direction: row | column | row-reverse | column-reverse

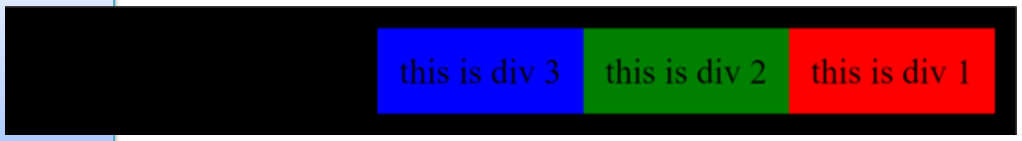
Default Value:

flex-direction:row

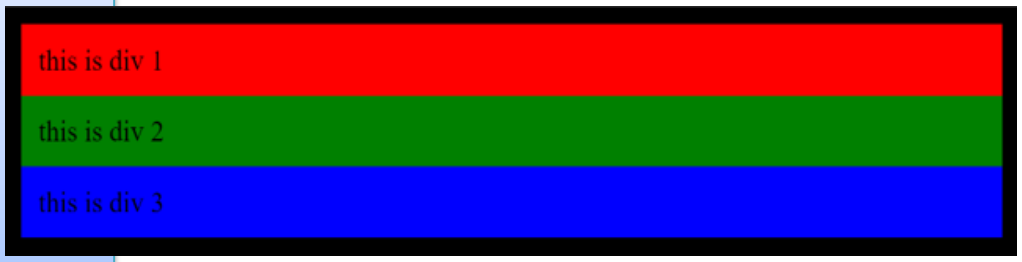
flex-direction: row

Flex Direction

flex-direction: row-reverse



flex-direction: column



flex-direction: column-reverse

Flex Wrap

Description:

It prevents the elements inside the flex box to wrap to a second row even if their total widths is greater than the flex box width.

Property:

flex-wrap: nowrap | wrap | wrap-reverse

Example 1 nowrap (default):

```
.box {
   background-color:black; padding:10px; width:500px;
   display:flex; flex-wrap: nowrap;
   }
.child1, .child2, .child3, {
   width:300px;
   }
```

Example 2 wrap:

```
.box {
        background-color:black; padding:10px; width:500px;
        display:flex; flex-wrap: wrap; }
.child1 {width:100px;}
.child2 {width:300px;}
.child3 {width:300px;}
```

Example 2 wrap-reverse:

```
.box {
        background-color:black; padding:10px; width:500px;
        display:flex; flex-wrap: wrap-reverse; }
.child1 {width:100px;}
.child2 {width:300px;}
.child3 {width:300px;}
```

Description:

The flex-flow property is a shorthand property for setting both the flex-direction and the flex-wrap properties.

Property:

flex-flow: flex-direction flex-wrap

Default value:

flex-flow: row nowrap;

Example:

```
.box {
    display: flex;
    flex-flow: row wrap;
}
```

Flexbox Justify-Content

Description:

The CSS3 justify-content property is used to define the horizontal alignment when the items are not using all the available space.

Property:

Justify-Content: flex-start | flex-end | Center | space-between | space-around

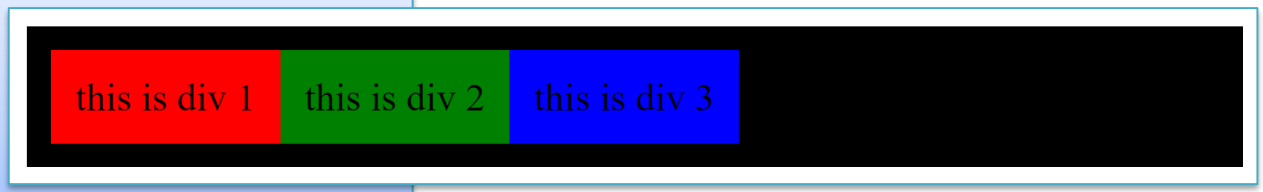
Values:

flex-start: default, sets items at the beginning of the container
flex-end: sets items at the end of the container.
Center: sets items at the center of the container.
space-between: sets items with space between them.
space-around: sets items with space before, between, and after them.

Example:

```
.box {
        background-color:black;
        padding:10px;
        width:500px;
        display:flex;}
```

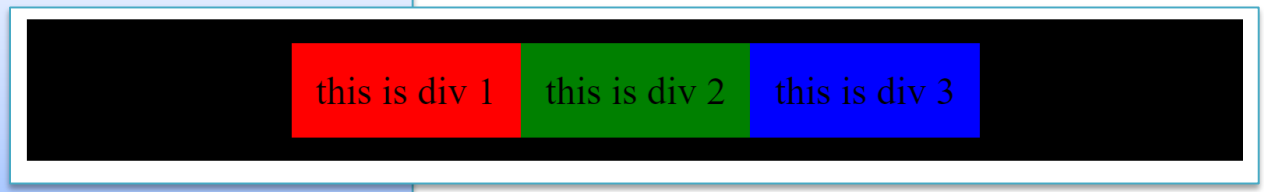
Justify-Content: flex-start;



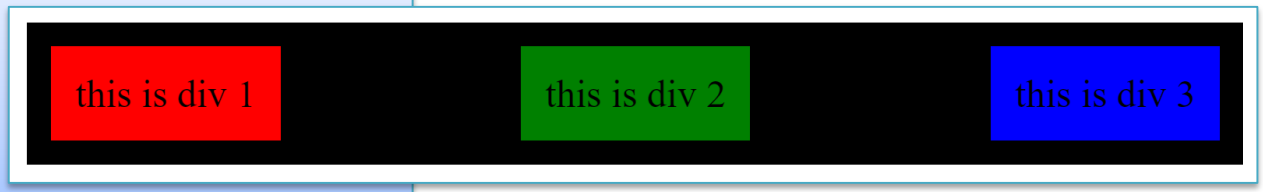
Justify-Content: flex-end;

Flexbox Justify-Content

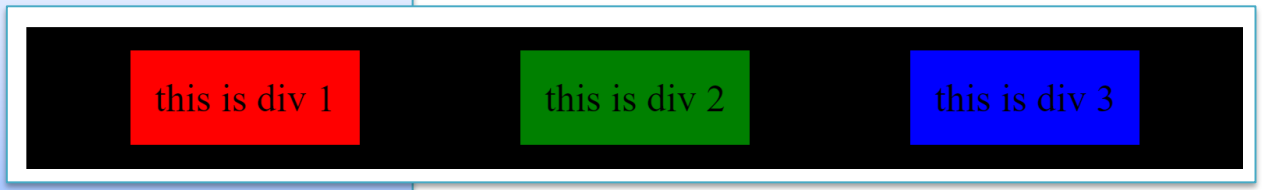
Justify-Content: Center;



Justify-Content: space-between;



Justify-Content: space-around;



Flexbox align-items

Description:

The CSS3 align-items property is used to set the flexbox items vertically align when the items are not using all the available space (vertically).

Property:

align-items: stretch | flex-start | flex-end | center

Values:

stretch: default, specifies that Items are stretched to fit the container.

flex-start: sets the items at the top of the container.

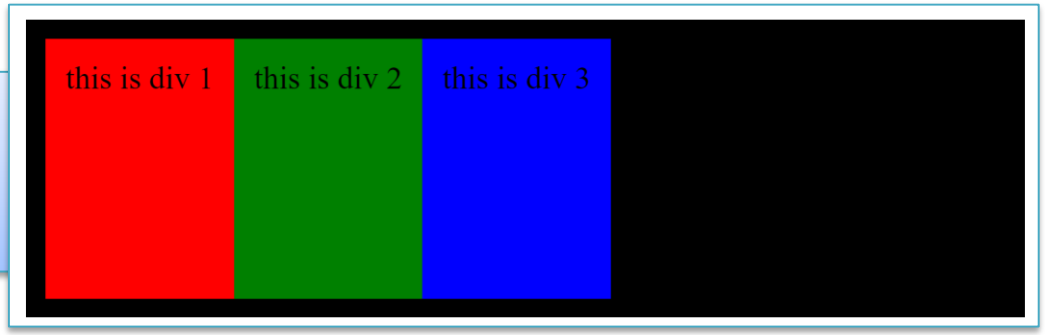
flex-end: sets the items at the bottom of the container.

center: sets the items at the middle of the container (vertically).

Example:

```
.box {
        background-color:black;
        padding:10px;
        width:500px;
        height: 150px;
        display:flex;}
```

align-items: stretch;

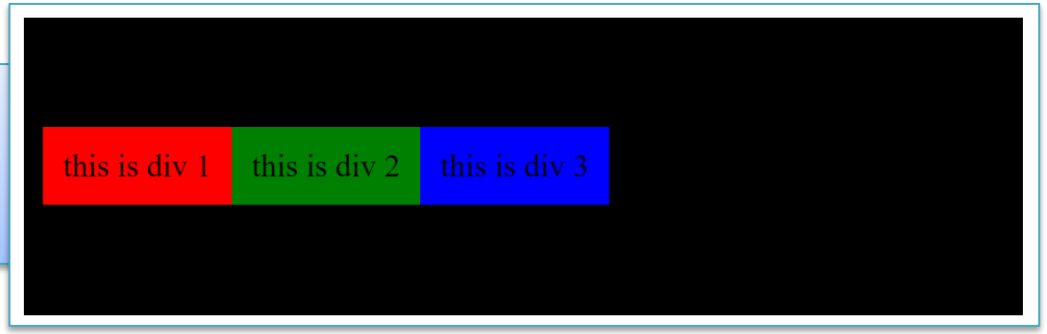


Flexbox align-items

Justify-Content: flex-start;

align-items: flex-end;

align-items: center;



Flexbox align-content

Description:

The CSS3 Flexbox align-content property is used to modify the behavior of the flex-wrap property. It is just like align-items, but it aligns flex lines instead of flex items.

Property:

align-content: stretch | flex-start | flex-end | Center | space-between | space-around

Values:

stretch: default, specifies lines stretch to take up the remaining space.

flex-start: lines are packed toward the start of the flex container.

flex-end: lines are packed toward the end of the flex container.

center: lines are packed toward the center of the flex container.

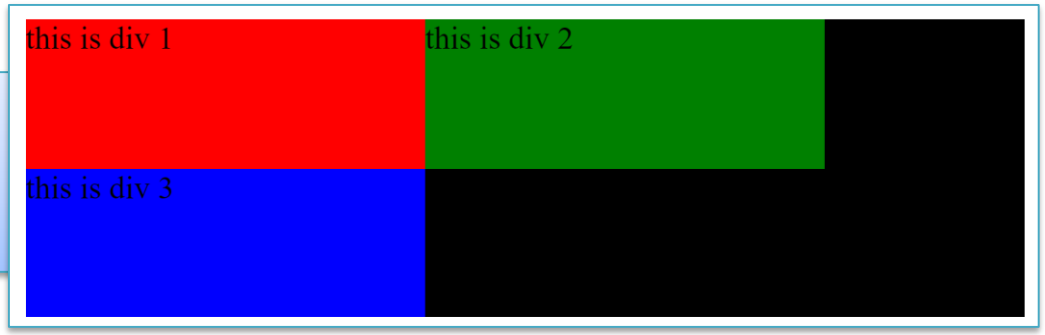
space-between: lines are evenly distributed in the flex container.

space-around: lines are evenly distributed in the flex container, with half-size spaces on either end.

Example:

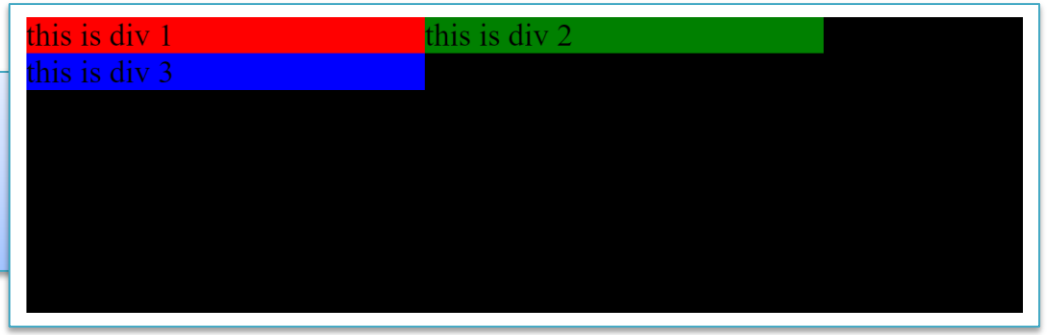
```
.parent {
    background-color:black;
    padding:10px;
    width:500px;
    height: 150px;
    display:flex;
    flex-wrap: wrap;
    }
```

align-content: stretch;

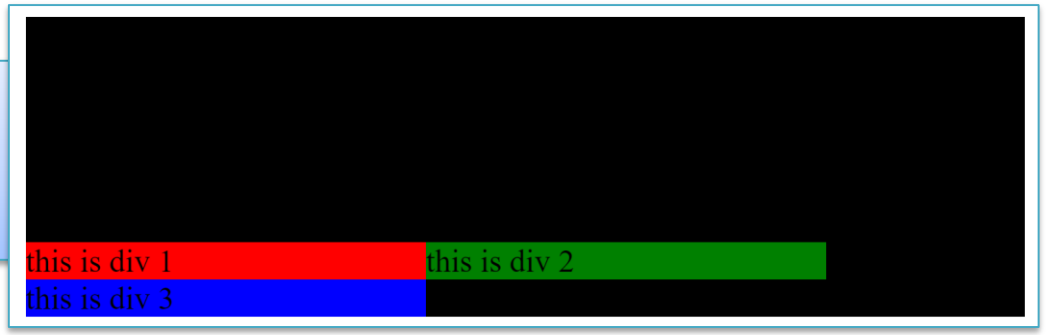


Flexbox align-content

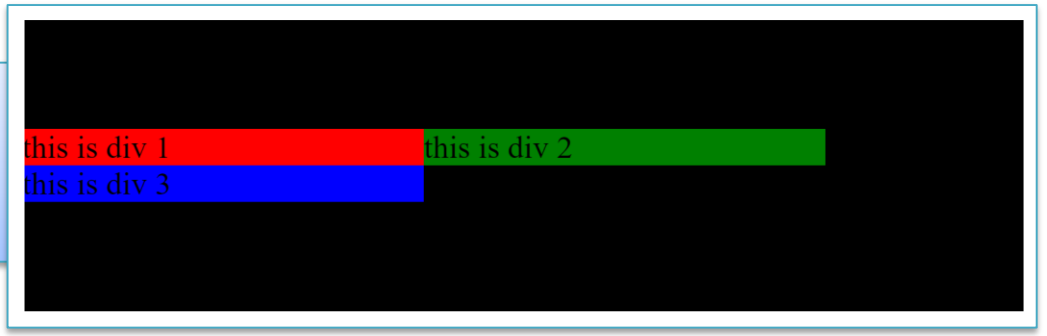
align-content:
flex-start;



align-content:
flex-end;

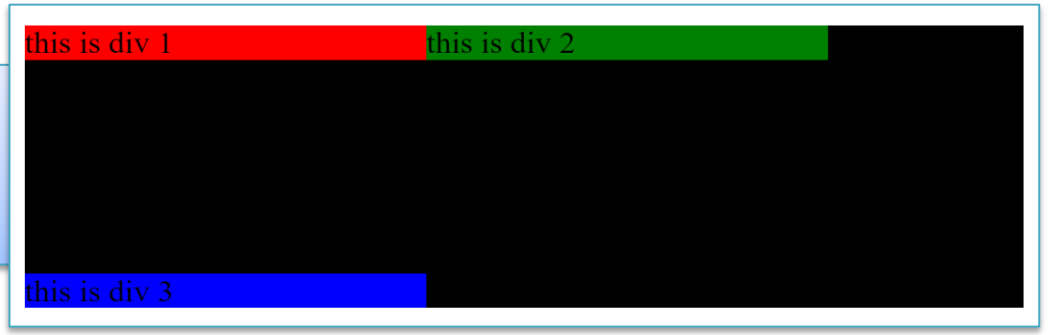


align-content:
center;

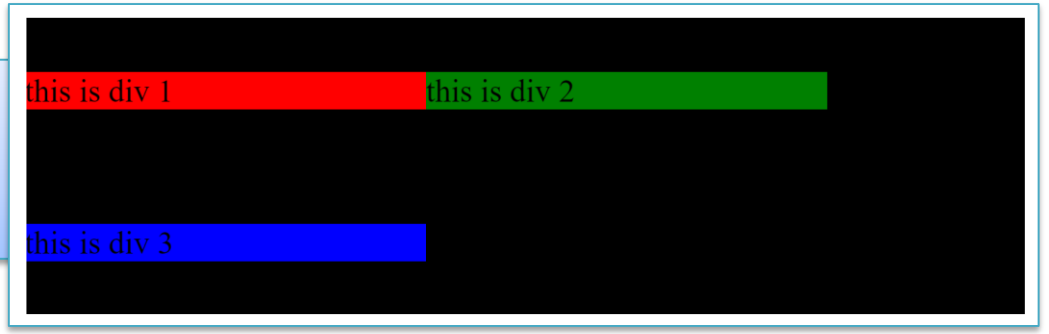


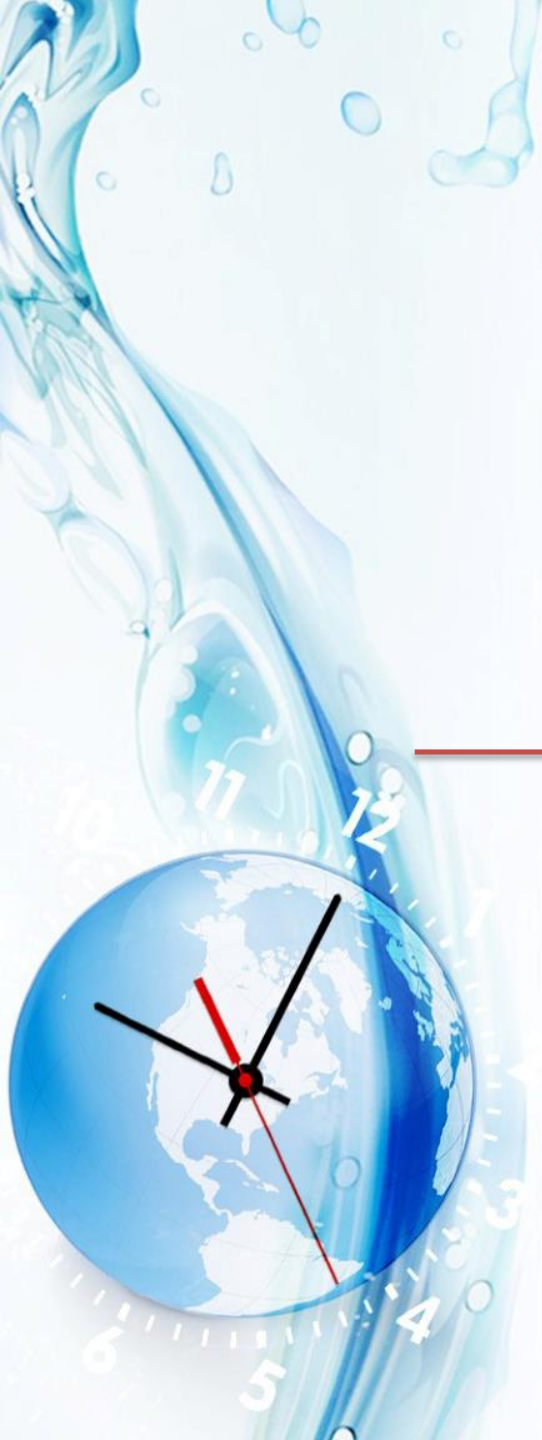
Flexbox align-content

align-content: space-between;



align-content: space-around;





CSS3 Flexbox Items

The direct child elements of a flex container automatically becomes flexible (flex) items.

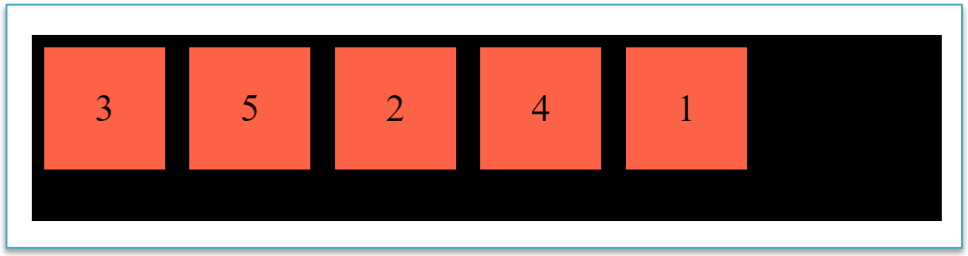
The flex item properties are: order, flex-grow, flex-shrink, flex-basis, flex, align-self

Description:

By default the flex items are displayed in the same order as they appear in the source document, The order property can be used to change this ordering.

Property:

order: 0 | number



Example:

```
<section>
    <div style="order: 5;">1</div>
    <div style="order: 3;">2</div>
    <div style="order: 1;">3</div>
    <div style="order: 4;">4</div>
    <div style="order: 2;">5</div>
</section>
```

flex-grow Property

Description:

It defines the ability for a flex item to grow if necessary. It dictates what amount of the available space inside the flex container the item should take up. (its share from the remaining space).

For example, if all items have flex-grow set to 1, every child will set to an equal size inside the container. If you were to give one of the children a value of 2, that child would take up twice as much space as the others.

Property:

flex-grow: 0 | number

Case:

if a container width is 300px and it has 2 Childs, each one has a width of 100px, the remainder white space will be 100px (300-(100+100)).

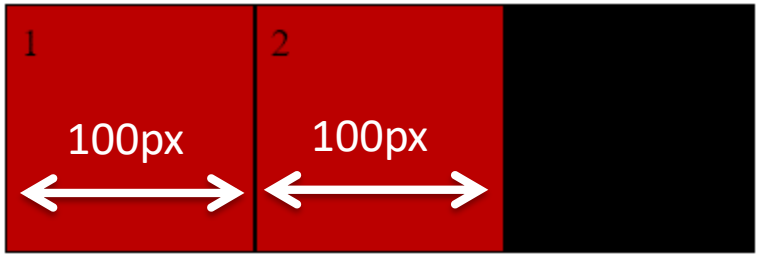
By default the white space will not be distributed. (default value for "flex-grow" for Childs is "0").

but if the first child "flex-grow" is set to "2" and the second child "flex-grow" is set to "1" then the remaining 100px will be distributed by ratio 2:1 (66.66px will be added to the first child width and 33.33 to the second).

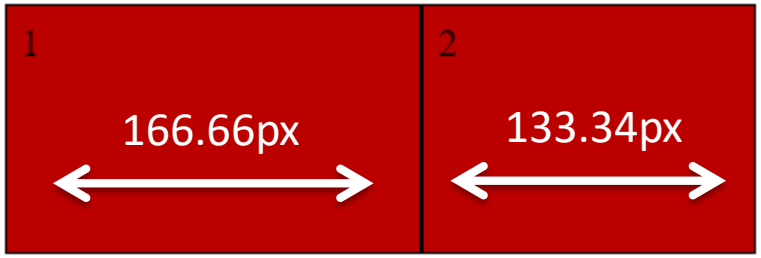
Example:

```
<head><style>
        section { background-color:black; width:300px;
                    height: 100px;  display:flex; }
        div {background: #b00; width:100; }
</style></head>
<body>
   <section>
     <div style="flex-grow:2;">1</div>
     <div style="flex-grow:1;">2</div>
  </section></body>
```

Flex-grow:0



Flex-grow:2 Flex-grow:1



flex-shrink Property

Description:

It specifies the "flex shrink factor", which determines how much the flex item will shrink relative to the rest of the flex items in the flex container when there isn't enough space on the row.

When omitted, it is set to 1 and the flex shrink factor is multiplied by the flex basis when distributing negative space.

Property:

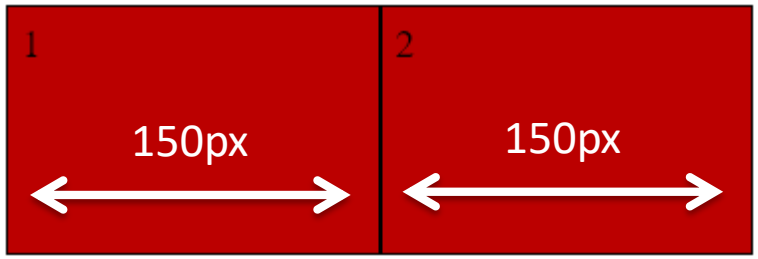
flex-shrink: 1 | number

flex-shrink Property

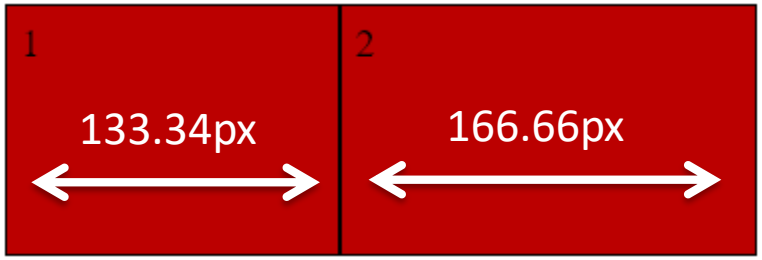
Example:

```
<head><style>
        section { background-color:black; width:300px;
                    height: 100px;  display:flex; }
        div {background: #b00; width:200}
</style></head>
<body>
   <section>
    <div style="flex-shrink:2;">1</div>
    <div style="flex-shrink:1;">2</div>
  </section></body>
```

Flex-shrink:1 - Default case



Flex-shrink:2 *Flex-shrink:1*



Description:

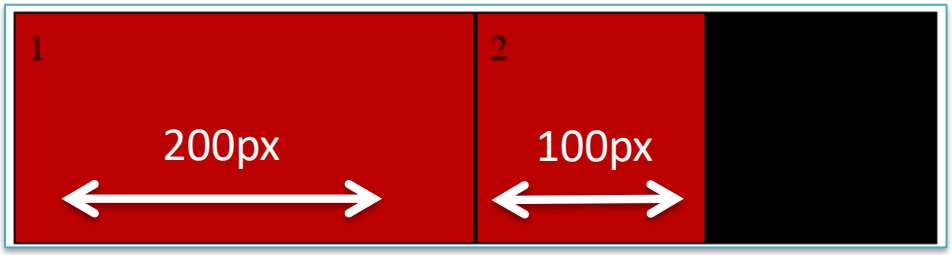
Specifies the initial size of the flex item, before any available space is distributed according to the flex factors.

Property:

Flex-basis: auto | width in px

Example:

```
section { background-color:black; width:400px;
          height: 100px;  display:flex; }
.child1 {flex-basis:200px;}
.child1 {flex-basis:200px;}
```



Description:

This is the shorthand for flex-grow, flex-shrink and flex-basis. The second and third parameters (flex-shrink and flex-basis) are optional.

Property:

flex: flex-grow flex-shrink flex-basis

Default Value:

flex: 0 1 auto;

Example:

```
section { background-color:black; width:400px;
          height: 100px;  display:flex; }
.child1 {flex: 0 0 200px;}
.child1 {flex: 0 1 100px}
```

Description:

Overrides the align-items value for specific flex items. (vertical alignment).

Property:

order: flex-start | flex-end | center | stretch

flex-start: margin edge of the item is placed on the cross-start line

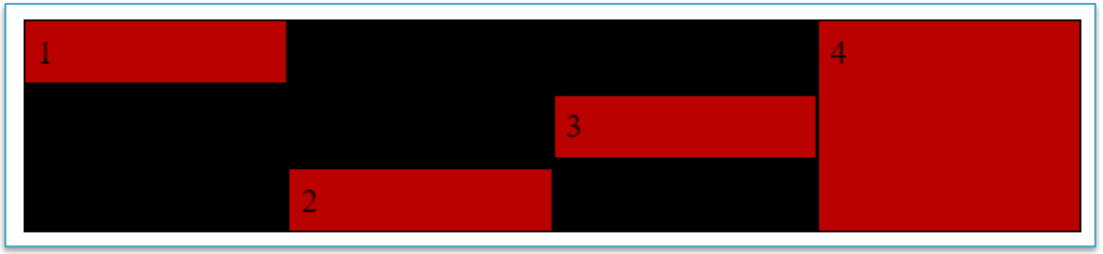
flex-end: margin edge of the item is placed on the cross-end line

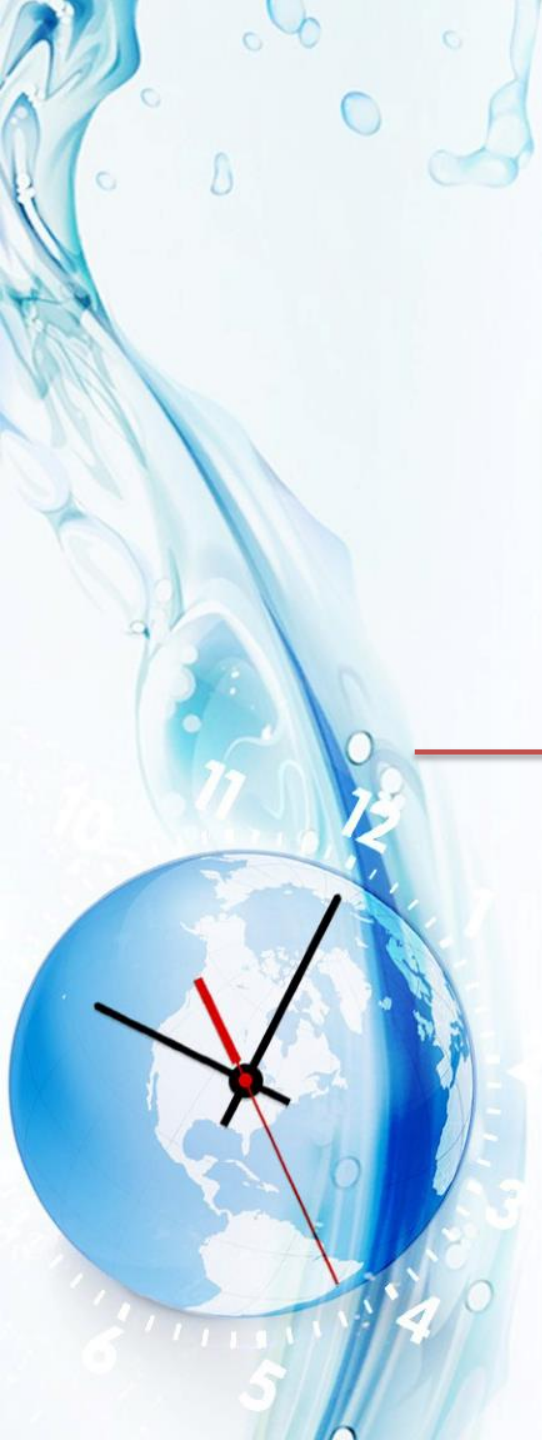
center: item is centered in the cross-axis

stretch (default): stretch to fill the container.

Example:

```
<head><style>
        section { background-color:black; width:500px;
                    height: 100px;  display:flex; }
        div {background: #b00;}
</style></head>
<body><section>
     <div style="flex-basis: 150px; align-self: flex-start;">1</div>
     <div style="flex-basis: 150px; align-self: flex-end;">2</div>
    <div style="flex-basis: 150px; align-self: center;">3</div>
    <div style="flex-basis: 150px; align-self: stretch;">4</div>
    </section></body>
```





CSS Advanced Selectors

Media Queries allows to have different styles for different devices/screen sizes. Their introduction in CSS3 has greatly eased the building of responsive webpages.

"Attribute Present" Selector

- *identifies an element based on an attribute, regardless of any actual value*

```
a[target] { color: red;}
```

```
<div>
    <a href="index.html" target="_blank">home</a>
    <a href="about.html">About Us</a>
</div>
```

“Attribute Equals” Selector

- *selects an element with a specific and exact matching attribute value*

```
a[href="http://google.com/"] { color: red;}
```

```
<div>
    <a href="index.html" target="_blank">home</a>
    <a href="http://google.com/">Google</a>
</div>
```

“Attribute Contains” Selector

- *selects an element based on part of an attribute value, but not an exact match.*

```
a[href*="dex"] { color: red;}
```

```
<div>
    <a href="index.html">home</a>
    <a href="http://google.com/">Google</a>
</div>
```

"Attribute Begins With" Selector

- *selects an element with an attribute value begins with a specific text.*

```
a[href^="http"] { color: red;}
```

```
<div>
    <a href="index.html">home</a>
    <a href="http://google.com/">Google</a>
</div>
```

"Attribute Ends With" Selector

- *selects an element with an attribute value ends with a specific text.*

```
a[href$="ml"] { color: red;}
```

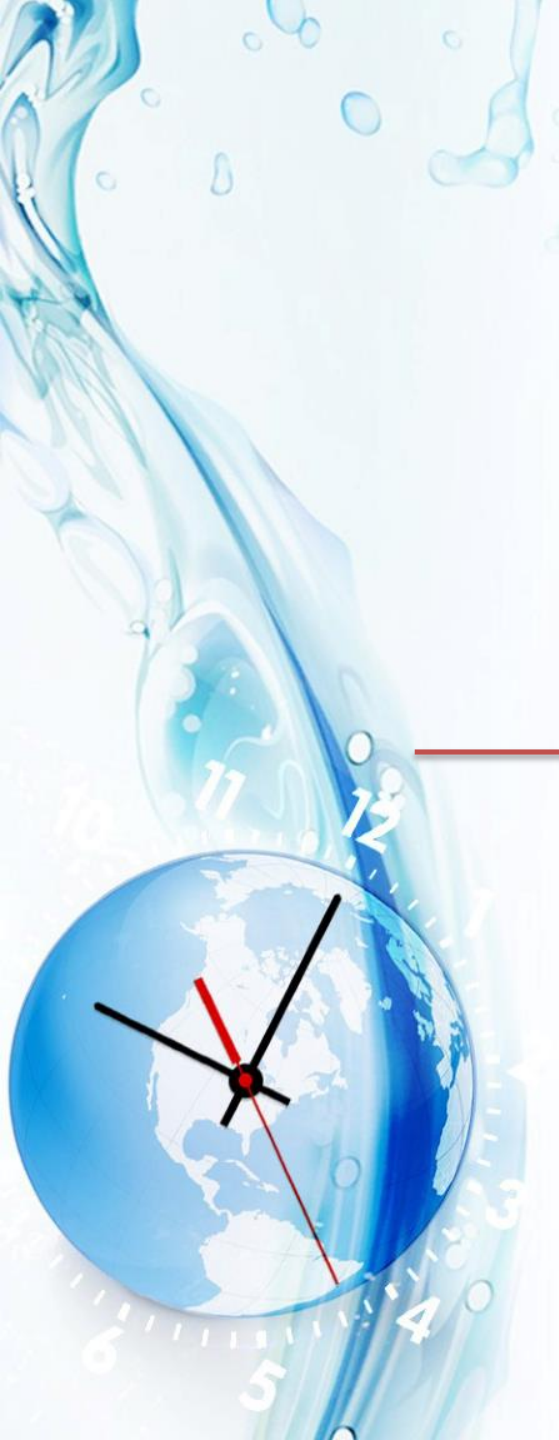
```
<div>
    <a href="index.html">home</a>
    <a href="http://google.com/">Google</a>
</div>
```

"Attribute Spaced" Selector

- *selects an element with an attribute value value that should be whitespace-separated.*

```
div[title~="nti"] { color: red;}
```

```
<div title="mcit nti">
        welcome to professional training program
</div>
```



Pseudo-classes

Link Pseudo-classes

- *a:link {...}*

  styles an anchor which has not been visited

- *a:visited {...}*

  styles links that a user has already visited based on their browsing history.

- *a:hover {...}*

  applied to an element when a user moves the cursor over the element

- *a:active {...}*

  applied to an element when a user clicks an element

- *a:focus {...}*

  applied to an element when a user has made an element the focus point of the page, often by using the keyboard to tab

User Interface State Pseudo-classes

- *input:enabled {...}*

  selects an input that is in the default state of enabled and available for use

- *input:disabled {...}*

  selects an input that has the disabled attribute tied to it.

- *input:checked {...}*

  selects checkboxes or radio buttons that are checked

- *input:valid {...}*

  selects elements with valid state

- *input[type='email']:invalid {...}*

  selects elements with in-valid state

- *:first-child*

  select an element if it's the first child within its parent

Example: ➔ li:first-child{…}

```
<ul>
    <li>selected</li>
    <li>not selected</li>
</ul>
<ul>
    <li>selected</li>
    <li>not selected</li>
</ul>
```

- *:last-child*

  select an element if it's the last child within its parent

Example: ➔ li:last-child{…}

```
<ul>
        <li> not selected</li>
        <li> selected</li>
</ul>
<ul>
        <li> not selected</li>
        <li>selected</li>
</ul>
```

- *:nth-child(n)*

  select elements based on their position within its parent

Example: ➔ li:nth-child(3){…}

```
<ul>
    <li> not selected</li>
    <li> not selected</li>
    <li>selected</li>
</ul>
```

- *:nth-last-child(n)*

  select elements based on their position within its parent, counting from the last child.

Example: ➔ li:nth-last-child(2) {…}

```
<ul>
    <li> not selected</li>
    <li> not selected</li>
    <li> selected</li>
    <li> not selected</li>
</ul>
```

- *:only-child*

  select an element if it's the last child within its parent

Example: ➔ li:only-child{…}

```
<ul>
  <li> not selected</li>
  <li> not selected</li>
</ul>
<ul>
    <li>selected</li>
</ul>
```

```
<ul>
  <li>not selected</li>
  <li>not selected</li>
</ul>
<ul>
  <li>not selected</li>
  <p> not selected</p>
</ul>
```

- *:first-of-type*

  select the first element of its type within a parent

Example: ➔ li:first-of-type{…}

```
<ul>
  <li>selected</li>
 <li> not selected</li>
</ul>
<ul>
    <li>selected</li>
</ul>
```

```
<ul>
 <li>selected</li>
 <li>not selected</li>
</ul>
<ul>
<p> not selected</p>
<li> selected</li>
</ul>
```

- *:last-of-type*

  select the last element of its type within a parent

Example: ➔ li:last-of-type{…}

```
<ul>
  <li> not selected</li>
  <li> selected </li>
</ul>
<ul>
  <li> not selected</li>
  <li> selected </li>
</ul>
```

- *:only-of-type*

  select an element if it is the only of its type within a parent.

Example: ➔ li:only-of-type {…}

```
<ul>
  <li>not selected</li>
  <li> not selected</li>
</ul>
<ul>
    <li>selected</li>
</ul>
```

```
<ul>
  <p> not selected</p>
  <li> selected</li>
  <p> not selected</p>
</ul>
```

- *:nth-of-type(n)*

  select elements based on their position, of a particular type, within its parent

Example: ➔ li:nth-of-type(3) {…}

```
<ul>
        <li> not selected</li>
        <li> not selected </li>
       <p> not selected </p>
       <li> selected</li>
       <li> not selected </li>
</ul>
```

- *:nth-last-of-type(n)*

  select elements based on their position, of a particular type, within its parent, counting from the last child

Example: ➔ li:nth-last-of-type(3) {…}

```
<ul>
    <li> not selected</li>
    <li> selected </li>
    <p> not selected </p>
    <li> not selected </li>
    <li> not selected </li>
</ul>
```

Empty Pseudo-class

- *:empty*

  selects elements that do not contain children or text nodes.

Example: ➔ div:empty {…}

```
<div>not selected</div>
<div><!-- selected --></div>
<div></div>
<div> </div> ➔ not selected
<div><strong></strong></div> ➔ not selected
```

- *:not(x)*

  The negation pseudo-class, :not(x), is a pseudo-class that takes an argument which is filtered out from the selection to be made.

Example: ➔ div:not(.red) {...}

selects all divs that are not have class="red".

Example: ➔ .red:not(div) {...}

select all elements with class="red" except the divs.

- *:first-letter*

  Select the first letter of text within an element.

  Example: ➔ p: first-letter {...}

  ➔ .red: first-letter {...}

- *:first-line*

  Select the first first line of text within an element.

  Example: ➔ p: first-line {...}

  ➔ .red: first-line {...}

Generated Content Pseudo-elements

creates new inline level pseudo-elements just inside the selected element. Using ":before" , ":after" and "contents" pseudo-elements

```
a:after {
        color: #9799a7;
        content: "click here";
        font-size: 11px;
}
```

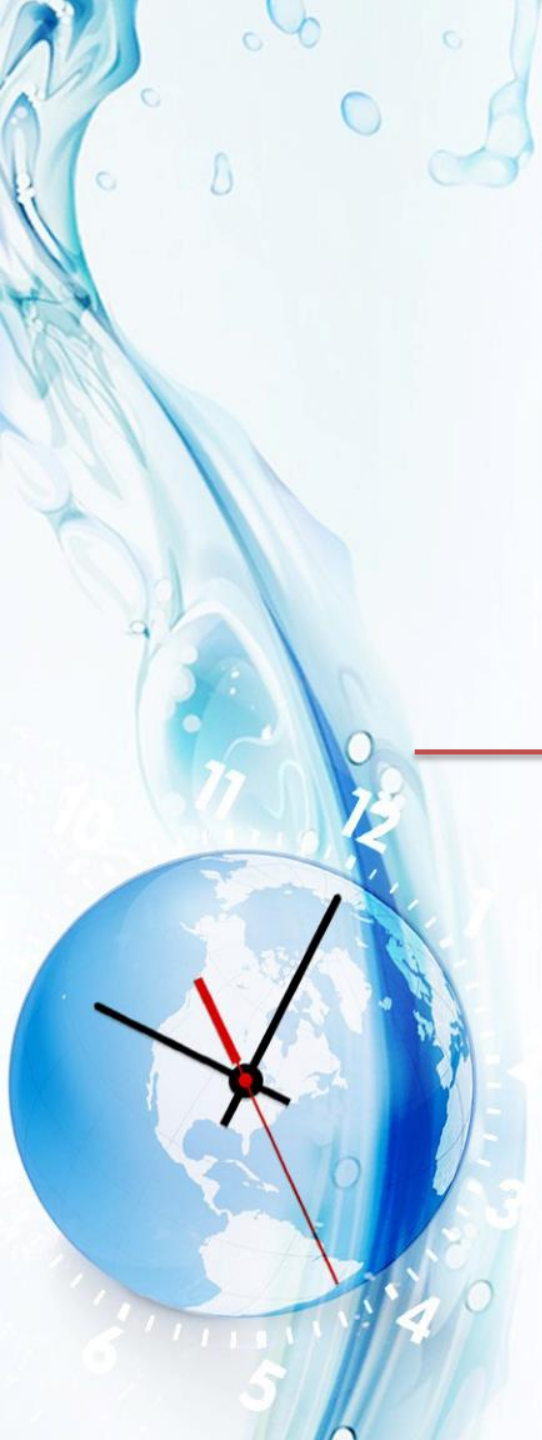
```
a:after {
    color: #9799a7;
    content: attr(href);
    font-size: 11px;
}
```

```
a:after {
        content: "*" attr(title) "*";
}
```

- *::selection*

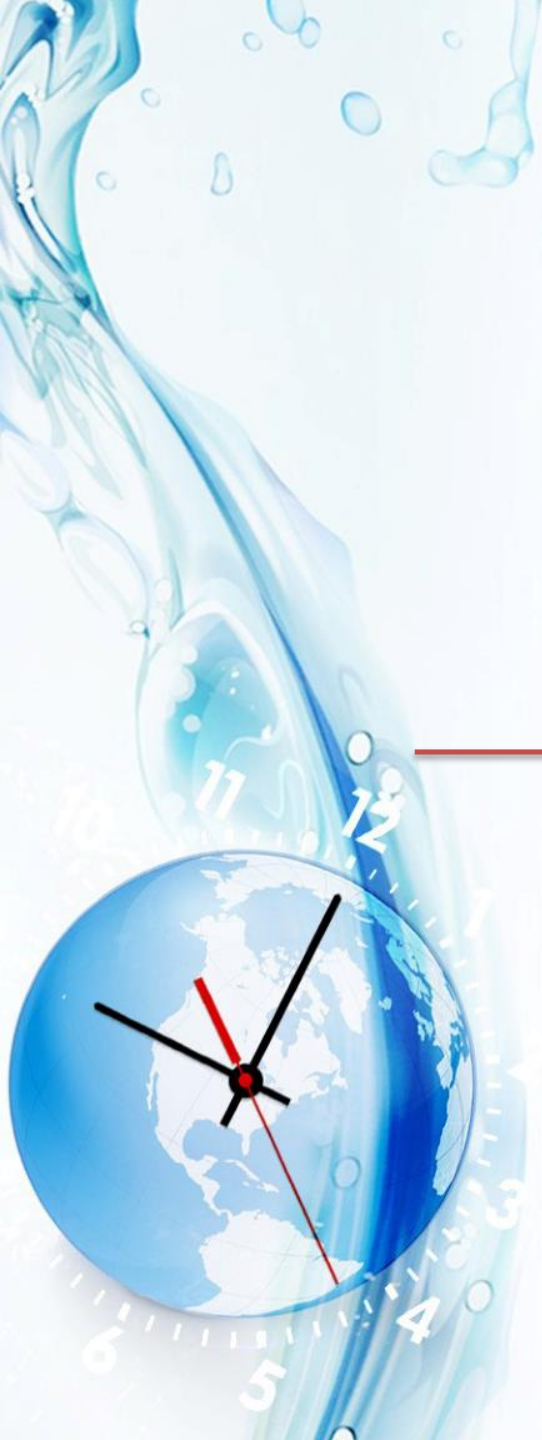
  selects a part of the document that has been selected, or highlighted, by a user's actions. The selection can be stylized only using the color, background, background-color, and text-shadow properties.

```
p::selection{
    background-color: red;
}
```



Responsive Web Design

Responsive web design is the practice of building a website suitable to work on every device and every screen size, no matter how large or small, mobile or desktop



CSS Media Queries

Media queries were built as an extension to media types commonly found when targeting and including styles. Media queries provide the ability to specify different styles for individual browser and device circumstances

Initializing Media Queries

There are two different ways to use media queries

- *Using the @media rule inside the existing style sheet.*
- *Using the @import rule to import a new style by linking to a separate style sheet.*

Use HTML

```
<link href="styles.css" rel="stylesheet"
media="all and (max-width: 1024px)">
```

Or Use CSS

```
/* @media Rule */
@media all and (max-width: 1024px) {...}
/* @import Rule */
@import url(styles.css) all and (max-width: 1024px) {...}
```

Height and Width Media Features

Used to determine the height or width for a device or browser viewport. Each of these media features may then also be prefixed with the min or max qualifiers, building a feature such as min-width or max-width.

Example

```
@media all and (min-width: 320px) and (max-width: 780px) {...}
```

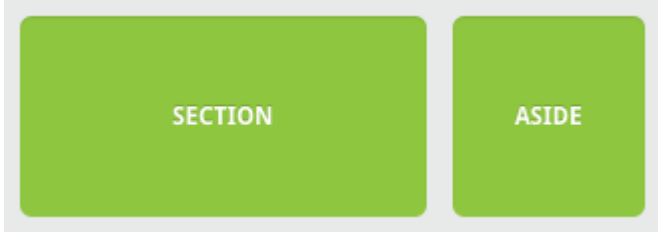
Media Queries Demo

Used to determine the height or width for a device or browser viewport. Each of these media features may then also be prefixed with the min or max qualifiers, building a feature such as min-width or max-width.

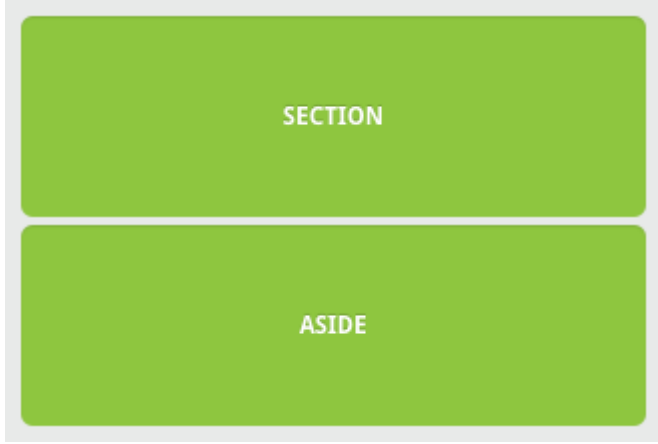
Example

```
@media all and (max-width: 420px) {
        section, aside {
          float: none;
          width: auto;
        }
}
```

No media queries



max-width: 420px



➢ Mobile First

- *Popular technique with using media queries. Means using styles targeted at smaller viewports as the default styles for a website, then use media queries to add styles as the viewport grows.*

- *The operating belief behind mobile first design is that a user on a mobile device, commonly using a smaller viewport, shouldn't have to load the styles for a desktop computer only to have them over written with mobile styles later.*

Mobile First Example

```
/* Default media */
body {
    background: #ddd;
}

/* Media for larger devices */
@media screen and (min-width: 800px) {
        body {
                background: #900;
        }
}
```