**Q1 CSS Selectors:**

The Selectors in CSS are there to select elements from the documents either single using class or all using elements names

**There are three types of Selectors in CSS:**

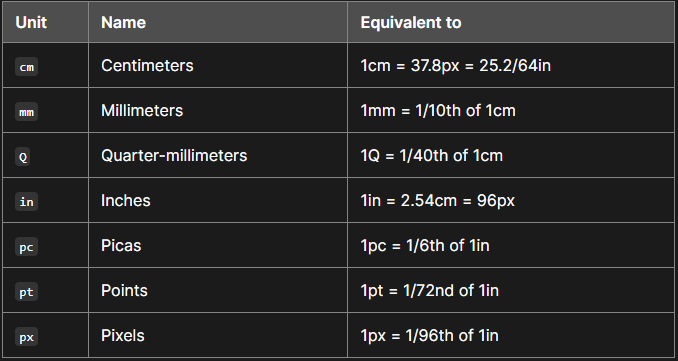
1. Select elements by using Directly Elements Names
2. Select elements by using Class Name (given to elements)
3. Select elements by using ID Name it is unique for every element

**Q2 Length in CSS:**

There are two types of lengths in CSS absolute and relative units

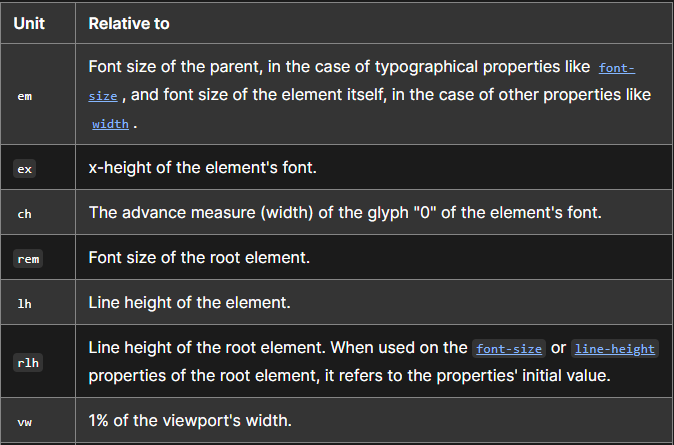
**Absolute length units**

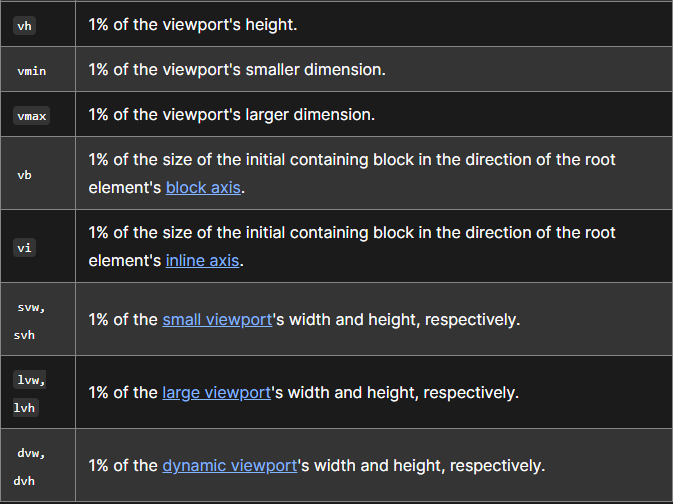
The following are all **absolute** length units — they are not relative to anything else and are generally considered to always be the same size.



#### Relative length units

Relative length units are relative to something else, perhaps the size of the parent element's font, or the size of the viewport. The benefit of using relative units is that with some careful planning you can make it so the size of text or other elements scales relative to everything else on the page. Some of the most useful units for web development are listed in the table below.





**The Difference Between PX, EM, REM, %, VW, and VH?**

Absolute Units

**PX**: Pixels (px) are considered absolute units, although they are relative to the DPI and resolution of the viewing device. But on the device itself, the PX unit is fixed and does not change based on any other element. Using PX can be problematic for responsive sites, but they are useful for maintaining consistent sizing for some elements. If you have elements that should not be resized, then using PX is a good choice.

Relative Units

1. **EM**: Relative to the parent element
2. **REM**: Relative to the root element (HTML tag)
3. **%:** Relative to the parent element
4. **VW**: Relative to the viewport’s width
5. **VH**: Relative to the viewport’s height

Unlike PX, relative units like %, EM, and REM are better suited to responsive design and also help [meet accessibility standards](https://elementor.com/blog/wordpress-accessibility-elementor/). Relative units scale better on different devices because they can scale up and down according to another element’s size.

1em = 16px (1 \* 16)

2em = 32px (2 \* 16)

.5em = 8px (.5 \* 16)

1rem = 16px

2rem = 32px

.5rem = 8px

100% = 16px

200% = 32px

50% = 8px

### What is the difference between EM and REM?

Looking at the chart above, it shows EM and REM looking exactly the same. So how do they differ?

Simply put, they differ based on inheritance. As mentioned, REM is based on the root element (HTML). Every child element that uses REM will then use the HTML root size as its calculation point, regardless of whether or not a parent element has any different sizes specified.

EM on the other hand, is based on the font size of the parent element. If a parent element has a different size than the root element, the EM calculation will be based off of the parent element, and not the root element. This means that nested elements that use EM for sizing can sometimes end up being a size that you didn’t anticipate. On the other hand, it does give you more fine-grained control if you need it to specify the size for a particular area of a page.

### So what about %, VW, and VH? What are they all about?

While PX, EM, and REM are primarily used for font sizing, %, VW, and VH are mostly used for margins, padding, spacing, and widths/heights.

To reiterate, VH stands for “viewport height”, which is the viewable screen’s height. 100VH would represent 100% of the viewport’s height or the full height of the screen. And of course, VW stands for “viewport width”, which is the viewable screen’s width. 100VW would represent 100% of the viewport’s width or the full width of the screen. % reflects a percentage of the parent element’s size, regardless of the viewport’s size.

Important Note ->>

**(Let’s look at some examples of where Element or gives %, VW, and VH options.)**

**Column Widths & Heights**: If you edit the layout of an Element or Column, you’ll notice that there is only one width sizing unit available – %. Column widths only work well and responsively when using (**percentages %)**, so no other option is given.

**Margins**: A section’s margins can be specified either in **PX** or **%.** Using % is usually preferable to ensure the margins don’t get larger than the content when scaling down for a mobile device for instance. By using a percentage of the width of the device, your margins will remain relative to the size of the content, which is almost always preferable.

**Padding**: A section’s padding can be specified either in **PX**, **EM**, or **%.** As with margins, it is often preferable to use either EM or % so the padding remains relative as the size of the page scales.

**Font Size**: If you edit the typography of an element, such as a Heading, you’ll see four choices: **PX**, **EM**, **REM**, and **VH**

**Q3 Padding & Margins:**

padding and margins top, right, left, bottom in single line (ex -> padding: top right and left bottom; or padding: top right left bottom ) for margin is same

ex - > padding: 1.5em 2em 2em;

padding 1.5em 2em 4em 2em

**Q4 Inherit in css:**

input element does not directly inherit form the parent to child but it can be done by using interit ex->

input, button{

font: interit;

}

**Q5 Inheritance in CSS:**

The child element will naturally inherit a CSS property with its value from the parent element if the CSS property is not specified.

CODE

<style>

#div1 {

color: red;

}

</style>

<div id="div1">

Parent Div

<div id="div1Child">Child Div 1</div>

<div id="div2Child">Child Div 2</div>

</div>

<style>

#div1 {

color: red;

}

</style>

<div id="div1">

Parent Div

<div id="div1Child">Child Div 1</div>

<div id="div2Child">Child Div 2</div>

</div>

## Which CSS properties are inherited?

Though not all CSS rules/properties are inherited, all font-\* properties are inherited. This includes:

* font-size
* font-family
* font-weight

The color property is also inherited.

Inheritance in CSS occurs when an inheritable property is not set on an element. It goes up in its parent chain to set the property value to its parent value.

CSS properties such as height, width, border, margin, padding, etc. are not inherited. We can enable inheritance on noninheritable CSS properties by using the inherit value.

## What is CSS inherit?

When you set inherit on a CSS property, the property takes the value from the element’s parent.

This applies not only to inheritable properties, but to all CSS properties.

Let’s say we have the following:

<style>

#div1 {

height: 100px;

color: red;

}

#div1Child {

height: inherit;

}

</style>

<div id="div1">

Parent Div

<div id="div1Child">Child Div 1</div>

<div id="div2Child">Child Div 2</div>

</div>

The div1 has a height set to 100px and a color set to red. The color will be inherited by the child elements. The height property is not inheritable, so the child elements won’t inherit it.

div1Child, on the other hand, has its height property set to inherit. This will make it inherit the value of its height from its parent element, div1. So the height of the div1Child will be 100px.

The inherit value enables inheritance on all CSS properties. With inherit, the specified element will take the value of the specified property from its parent element.

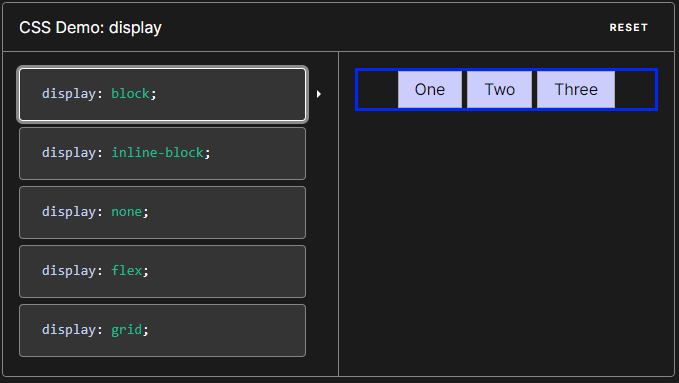
**Q5 Pseudo-Class:**

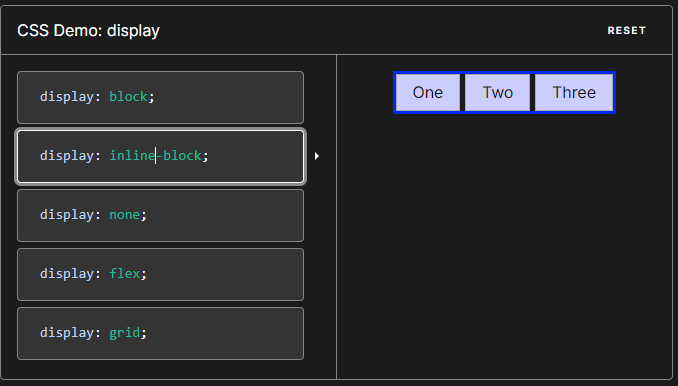
**A Pseudo class in CSS is used to define the special state of an element. It can be combined with a CSS selector to add an effect to existing elements based on their states. For Example, changing the style of an element when the user hovers over it, or when a link is visited. All of these can be done using Pseudo Classes in CSS.**

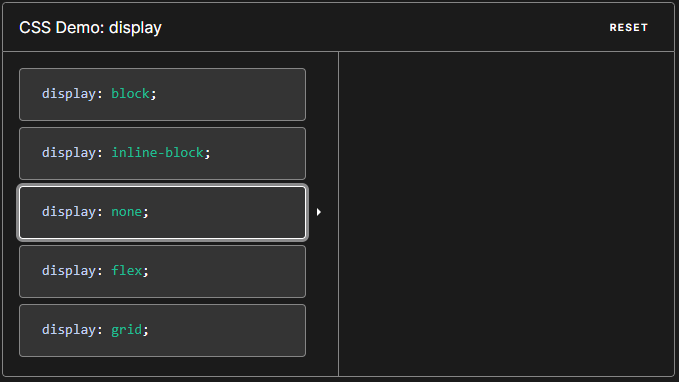
**Pseudo-elements defined by a set of CSS specifications include the following:**

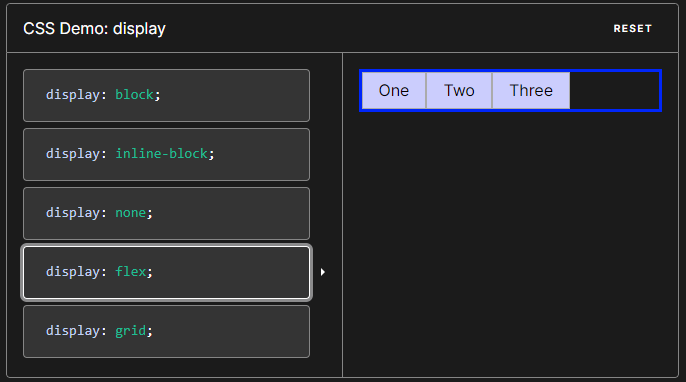
1. **::after**
2. **::backdrop**
3. **::before**
4. **::cue**
5. **::cue-region**
6. **::first-letter**
7. **::first-line**
8. **::file-selector-button**
9. **::grammar-error Experimental**
10. **::marker**
11. **::part()**
12. **::placeholder**
13. **::selection**
14. **::slotted()**
15. **::spelling-error Experimental**
16. **::target-text**
17. **:hover**
18. **:visited**

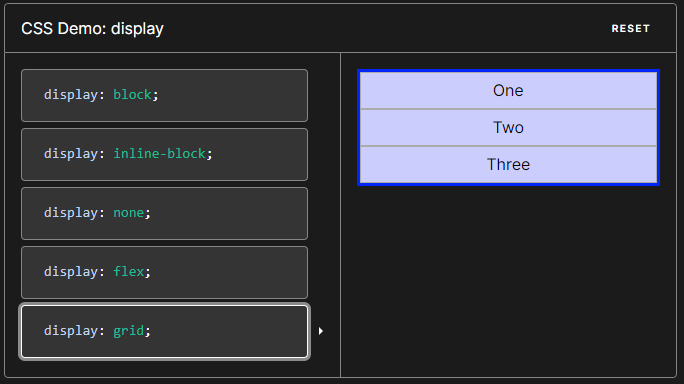
**Q6 Display Elements:**

****

****

****

****

****

**The CSS display property is specified using keyword values.**

**/\* precomposed values \*/**

**display: block;**

**display: inline;**

**display: inline-block;**

**display: flex;**

**display: inline-flex;**

**display: grid;**

**display: inline-grid;**

**display: flow-root;**

**/\* box generation \*/**

**display: none;**

**display: contents;**

**/\* multi-keyword syntax \*/**

**display: block flow;**

**display: inline flow;**

**display: inline flow-root;**

**display: block flex;**

**display: inline flex;**

**display: block grid;**

**display: inline grid;**

**display: block flow-root;**

**/\* other values \*/**

**display: table;**

**display: table-row; /\* all table elements have an equivalent CSS display value \*/**

**display: list-item;**

**/\* Global values \*/**

**display: inherit;**

**display: initial;**

**display: revert;**

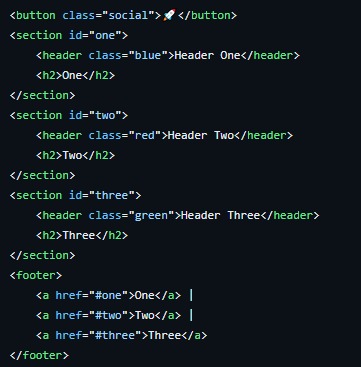
**display: revert-layer;**

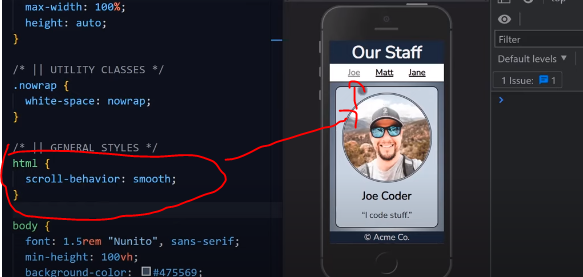
**display: unset;**

**Reference for deep down content:**[**https://developer.mozilla.org/en-US/docs/Web/CSS/display**](https://developer.mozilla.org/en-US/docs/Web/CSS/display)

**To Scroll Page via Given Link in the header:**

**Use element id to scroll the page**





For scrolling animation when the link is clicked

**Min width & Height**

Min-width:

The min-width property in CSS is used to set the **minimum width** of a specified element. The min-width property always overrides the width property whether followed before or after width in your declaration. Authors may use any of the [length values](https://css-tricks.com/the-lengths-of-css) as long as they are a positive value.

Code example

.wrapper {

width: 100%;

min-width: 20em; /\* Will be AT LEAST 20em wide \*/

}

Be mindful when assuming min-width is inherited as this property does not inherit from other parent elements. If authors define a width using an absolute value (px, pt, in, cm, mm), the min-width will not take affect as the width has been defined indefinitely. For example, if a value of 200px is used as a width length, your min-width value of 100px will not be neccessary as you have already specified an absolute value for the width (i.e. 200px). The best way to use min-width is to define a width value as a percentage and use an absolute value for the min-width property otherwise using a percentage value for for both min-width and width will not produce the expected result.

Example

<h1>min-width</h1>

<figure class="demo">

<p class="min min600">Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.</p>

<figcaption>Width = 80% / Minimum Width = 600px</figcaption>

</figure>

<figure class="demo">

<p class="min min320">Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.</p>

<figcaption>Width = 100% / Minimum Width = 320px</figcaption>

</figure>

<figure class="demo">

<p class="min min480">Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.</p>

<figcaption>Width = 600px / Minimum Width = 480px (won't work)</figcaption>

</figure>

<figure class="demo">

<p class="min min-percent">Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.</p>

<figcaption>Width = 100% / Minimum Width = 50% (won't work)</figcaption>

</figure>

<figure class="demo">

<p class="min em40">Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.</p>

<figcaption>Width = 100% / Minimum Width = 40em</figcaption>

</figure>

<figure class="demo">

<p class="min rem40">Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.</p>

<figcaption>Width = 100% / Minimum Width = 40rem</figcaption>

</figure>

<h3>Tables</h3>

<figure class="demo">

<table class="min mintable">

<thead>

<tr>

<th class="chrome"><span>Chrome</span></th>

<th class="safari"><span>Safari</span></th>

<th class="firefox"><span>Firefox</span></th>

<th class="opera"><span>Opera</span></th>

<th class="ie"><span>IE</span></th>

<th class="android"><span>Android</span></th>

<th class="iOS"><span>iOS</span></th>

</tr>

</thead>

<tbody>

<tr>

<td class="yep-nope">24+</td>

<td class="yep-nope">5.1+</td>

<td class="yep-nope">18+</td>

<td class="yep-nope">12.1+</td>

<td class="yep-nope">8+</td>

<td class="yep-nope">1.0+</td>

<td class="yep-nope">2.1+</td>

</tr>

</tbody>

</table>

<figcaption>Width = 100% / Minimum Width = 600px</figcaption>

</figure>

<footer>

<p><small>\*These examples use the entire viewport width in order to understand and grasp the concepts outlined. Squeeze your browser to witness the results.</small></p>

</footer>

@import "compass/css3";

//========================================

// $Global Styles

//========================================

html {

background: #444444;

color: #ffffff;

}

.demo {

background: #000;

}

//========================================

// $Minimum Width Demo

//========================================

.min {

background: #e78629;

}

// Min-Width Length Values

// ========================================

// minimum 600px

.min600 {

width: 80%;

min-width: 600px;

}

// minimum 320px

.min320 {

width: 100%;

min-width: 320px;

}

// minimum 480px

.min480 {

width: 600px;

min-width: 480px;

}

// minimum 50%

.min-percent {

width: 100%;

min-width: 50%;

}

// minimum 40em

.rem40 {

width: 100%;

min-width: 40em;

}

// minimum 40rem

.rem40 {

width: 100%;

min-width: 40rem;

}

// Min-Width TABLES

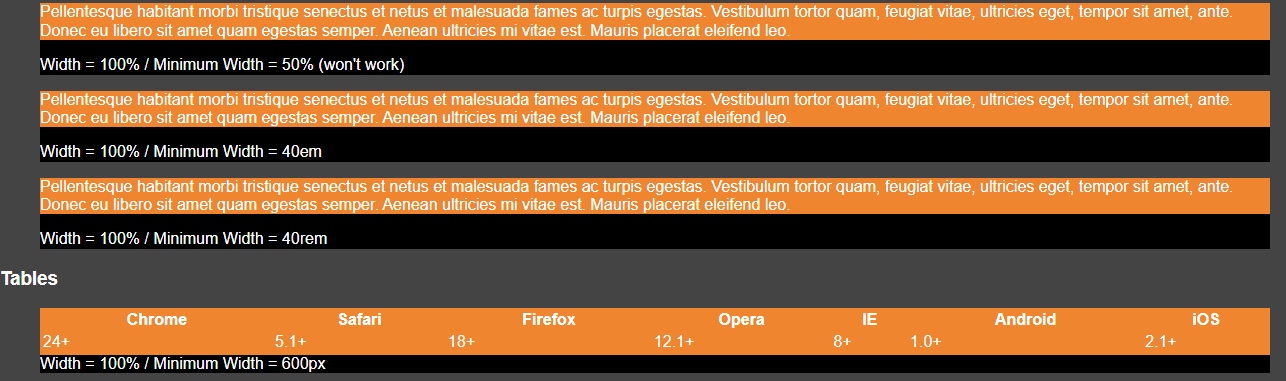
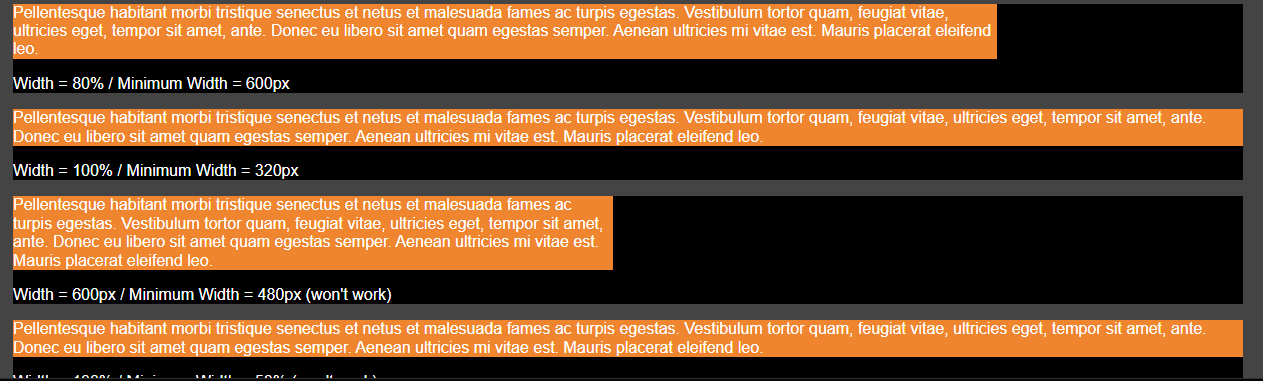
// ========================================

// minimum 30em

.mintable {

width: 100%;

min-width: 600px;

}

Min- height:

The min-height property in CSS is used to set the minimum height of a specified element. The min-height property always overrides both height and max-height. Authors may use any of the length values as long as they are a positive value

.wrapper {

height: 100%; /\* full height of the content box \*/

min-height: 20em; /\* Will be AT LEAST 20em tall \*/

}

.wrapper {

height: 600px;

min-height: 400px; /\* Will be AT LEAST 400px tall : overrides height \*/

}

.wrapper {

min-height: 400px; /\* overrides height and max-height \*/

height: 200px;

max-height: 300px;

}

Example:

<h1>Min-Height</h1>

<h3>Block Level Elements</h3>

<p class="min-height-px">Content in a paragraph : <code>height: 100% / min-height: 50px</code></p>

<p class="min-height-em">Content in a paragraph : <code>height: 100% / min-height: 8em</code></p>

<p class="min-height-percent">Content in a paragraph : <code>height: 100% / min-height: 50%</code></p>

<h3>Elements Nested</h3>

<div class="nesting example1">

<p class="min-heightnested">Example#1 : Containing element : <code>height: 100px</code>. Content in a paragraph : <code>min-height: 70%</code></p>

</div>

<div class="nesting example2">

<p class="min-heightnested">Example#2 : Containing element : <code>height: 100%</code>. Content in a paragraph : <code>height: auto</code></p>

</div>

<h3>Tables</h3>

<code>height: 100px; min-height: 500px;</code>

<table class="browser-support-table table-example1">

<thead>

<tr>

<th>Height</th>

<th class="chrome"><span>Chrome</span></th>

<th class="safari"><span>Safari</span></th>

<th class="firefox"><span>Firefox</span></th>

<th class="opera"><span>Opera</span></th>

<th class="ie"><span>IE</span></th>

<th class="android"><span>Android</span></th>

<th class="iOS"><span>iOS</span></th>

</tr>

</thead>

<tbody>

<tr>

<td>100px</td>

<td class="yep-nope">All</td>

<td class="yep-nope">All</td>

<td class="yep-nope">All</td>

<td class="yep-nope">All</td>

<td class="yep-nope">All</td>

<td class="yep-nope">All</td>

<td class="yep-nope">All</td>

</tr>

</tbody>

</table>

<code>height: 100%; min-height: 100px;</code>

<table class="browser-support-table table-example2">

<thead>

<tr>

<th>Height</th>

<th class="chrome"><span>Chrome</span></th>

<th class="safari"><span>Safari</span></th>

<th class="firefox"><span>Firefox</span></th>

<th class="opera"><span>Opera</span></th>

<th class="ie"><span>IE</span></th>

<th class="android"><span>Android</span></th>

<th class="iOS"><span>iOS</span></th>

</tr>

</thead>

<tbody>

<tr>

<td>100%</td>

<td class="yep">All</td>

<td class="yep">All</td>

<td class="yep">All</td>

<td class="yep">All</td>

<td class="yep">All</td>

<td class="yep">All</td>

<td class="yep">All</td>

</tr>

</tbody>

</table>

CSS

@import "compass/css3";

//========================================

// Globals

//========================================

html {

background: #e78629;

}

p {

background: #212121;

color: #FFF;

}

.nesting {

background: white;

}

table {

float: left;

margin-right: .5em;

background: white;

}

//========================================

// Min-Height Demo

//========================================

// $Block Level Elements

// =========================================

p.min-height-px {

height: 100%;

min-height: 50px;

}

p.min-height-em {

height: 100%;

min-height: 8em;

}

/\*\*

\* height value determined by

\* content and not these % values

\*/

p.min-height-percent {

height: 100%;

min-height: 50%;

}

// $Elements Nested

// =========================================

// Example#1

.nesting.example1 {

height: 100px;

.min-heightnested {

min-height: 70%; /\* works because height is an absolute length value \*/

}

}

// Example#2

.nesting.example2 {

height: 100%;

.min-heightnested {

height: auto;

}

}

// $Tables

// =========================================

// Example Left

.table-example1 {

min-height: 500px;

height: 100px;

}

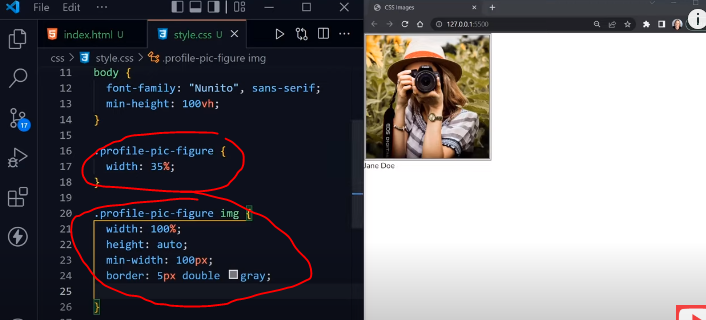
// Example Right

.table-example2 {

min-height: 100px;

height: 100%;

}



min-height and width provide the least minimum width, if the min-width provided 100px so that it cannot get smaller than 100px same apply for max-width and height

**max-width:**

The max-width CSS property sets the maximum width of an element. It prevents the used value of the width property from becoming larger than the value specified by max-width.

max-width overrides [width](https://developer.mozilla.org/en-US/docs/Web/CSS/width), but [min-width](https://developer.mozilla.org/en-US/docs/Web/CSS/min-width) overrides max-width.

## [Syntax](https://developer.mozilla.org/en-US/docs/Web/CSS/max-width#syntax)

/\* <length> value \*/

max-width: 3.5em;

/\* <percentage> value \*/

max-width: 75%;

/\* Keyword values \*/

max-width: none;

max-width: max-content;

max-width: min-content;

max-width: fit-content(20em);

/\* Global values \*/

max-width: inherit;

max-width: initial;

max-width: revert;

max-width: revert-layer;

max-width: unset;

The max-width property in CSS is used to define the maximum width of an element. The value of the width cannot be larger than the value by max-width. If the content is larger than the max-width then it will go to the next line and if the content is smaller than max-width then it has no effect.

Syntax:

max-width: none| length| initial| inherit;

**max-height:**

The max-height CSS property sets the maximum height of an element. It prevents the used value of the height property from becoming larger than the value specified for max-height.

max-height overrides [height](https://developer.mozilla.org/en-US/docs/Web/CSS/height), but [min-height](https://developer.mozilla.org/en-US/docs/Web/CSS/min-height) overrides max-height.

## [Syntax](https://developer.mozilla.org/en-US/docs/Web/CSS/max-height#syntax)

/\* <length> value \*/

max-height: 3.5em;

/\* <percentage> value \*/

max-height: 75%;

/\* Keyword values \*/

max-height: none;

max-height: max-content;

max-height: min-content;

max-height: fit-content(20em);

/\* Global values \*/

max-height: inherit;

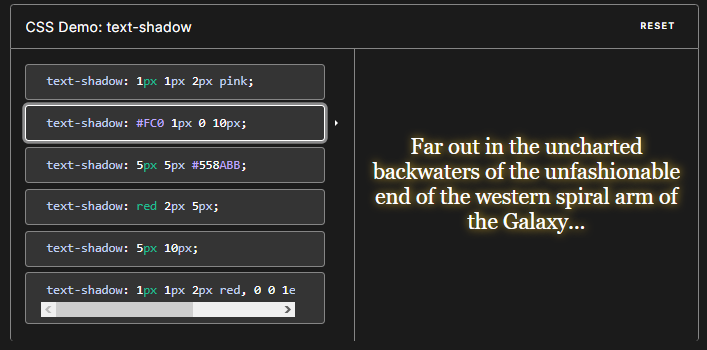
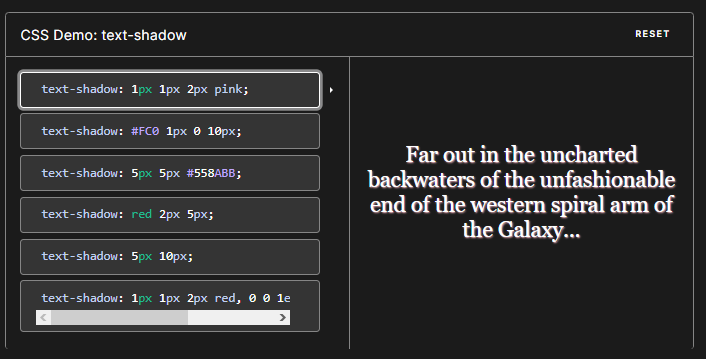
max-height: initial;

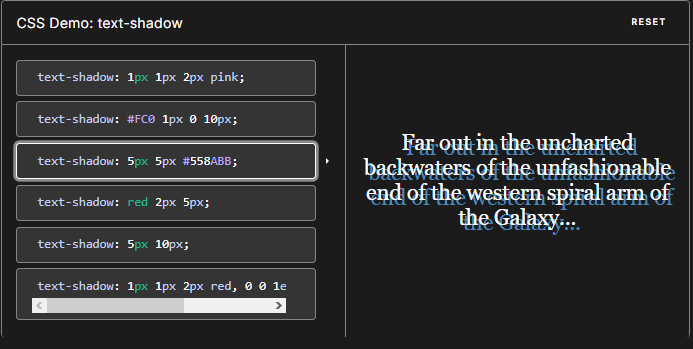
max-height: revert;

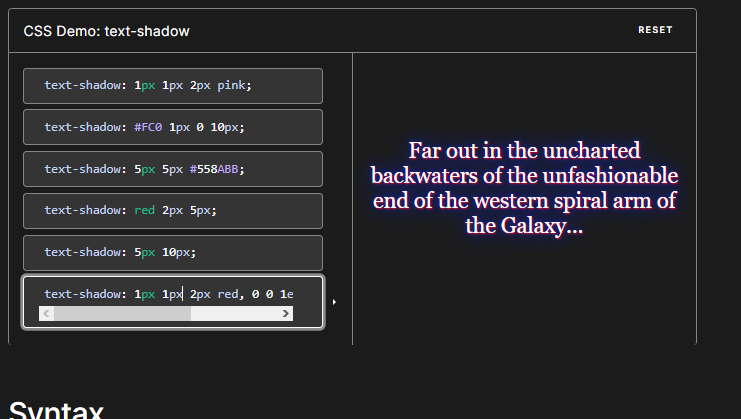
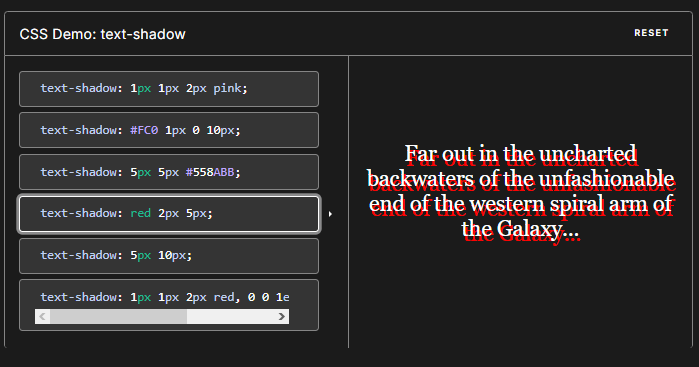
max-height: revert-layer;

max-height: unset;

**text-shadow element:**

The text-shadow CSS property adds shadows to text. It accepts a comma-separated list of shadows to be applied to the text and any of its decorations. Each shadow is described by some combination of X and Y offsets from the element, blur radius, and color.





text-shadow: 1px 1px 2px red, 0 0 1em blue, 0 0 0.2em blue;

Syntax

/\* offset-x | offset-y | blur-radius | color \*/

text-shadow: 1px 1px 2px black;

/\* color | offset-x | offset-y | blur-radius \*/

text-shadow: #fc0 1px 0 10px;

/\* offset-x | offset-y | color \*/

text-shadow: 5px 5px #558abb;

/\* color | offset-x | offset-y \*/

text-shadow: white 2px 5px;

/\* offset-x | offset-y

/\* Use defaults for color and blur-radius \*/

text-shadow: 5px 10px;

/\* Global values \*/

text-shadow: inherit;

text-shadow: initial;

text-shadow: revert;

text-shadow: revert-layer;

text-shadow: unset;

p {

text-shadow: 1px 1px 1px #000;

}

You can apply multiple text shadows by comma separating

p {

text-shadow: 1px 1px 1px #000,

3px 3px 5px blue;

}

The first two values specify the length of the shadow offset. The first value specifies the horizontal distance and the second specifies the vertical distance of the shadow. The third value specifies the blur radius and the last value describes the color of the shadow:

1. **value** = The X-coordinate  
2. **value** = The Y-coordinate  
3. **value** = The blur radius  
4. **value** = The color of the shadow

Using positive numbers as the first two values ends up with placing the shadow to the right of the text horizontally (first value) and placing the shadow below the text vertically (second value).

The third value, the blur radius, is an optional value which can be specified but don’t have to. It’s the amount of pixels the text is stretched which causes a blur effect. If you don’t use the third value it is treated as if you specified a blur radius of zero.

Also, remember you can use [RGBa](https://css-tricks.com/rgba-browser-support/) or [HSLa](https://css-tricks.com/examples/HSLaExplorer/) values for the color, for example, a 40% transparency of white:

p {

text-shadow: 0px 2px 2px rgba(255, 255, 255, 0.4);

}

h1.simple Monster Mash

h1.otto Slippery Slime

h1.relief Mummy Mommy

h1.close Graveyard Smash

h1.printers Skeleton Crew

h1.glow Smooth Zombie

h1.vamp Vampire Diaries

CSS

h1 {

text-align: center;

font: bold 80px Sans-Serif;

padding: 40px 0;

}

.simple {

background: #91877b;

text-shadow: 0 1px 0 rgba(255, 255, 255, 0.4);

}

.otto {

background: #0e8dbc;

color: white;

text-shadow: 0 1px 0 #ccc,

0 2px 0 #c9c9c9,

0 3px 0 #bbb,

0 4px 0 #b9b9b9,

0 5px 0 #aaa,

0 6px 1px rgba(0,0,0,.1),

0 0 5px rgba(0,0,0,.1),

0 1px 3px rgba(0,0,0,.3),

0 3px 5px rgba(0,0,0,.2),

0 5px 10px rgba(0,0,0,.25),

0 10px 10px rgba(0,0,0,.2),

0 20px 20px rgba(0,0,0,.15);

}

.relief {

background-color: #3a50d9;

color: #e0eff2;

font: italic bold 100px Georgia, Serif;

text-shadow: -4px 3px 0 #3a50d9, -14px 7px 0 #0a0e27;

}

.close {

background-color: #fff;

color: #202c2d;

text-shadow:

0 1px #808d93,

-1px 0 #cdd2d5,

-1px 2px #808d93,

-2px 1px #cdd2d5,

-2px 3px #808d93,

-3px 2px #cdd2d5,

-3px 4px #808d93,

-4px 3px #cdd2d5,

-4px 5px #808d93,

-5px 4px #cdd2d5,

-5px 6px #808d93,

-6px 5px #cdd2d5,

-6px 7px #808d93,

-7px 6px #cdd2d5,

-7px 8px #808d93,

-8px 7px #cdd2d5;

}

.printers {

background-color: #edde9c;

color: #bc2e1e;

text-shadow:

0 1px 0px #378ab4,

1px 0 0px #5dabcd,

1px 2px 1px #378ab4,

2px 1px 1px #5dabcd,

2px 3px 2px #378ab4,

3px 2px 2px #5dabcd,

3px 4px 2px #378ab4,

4px 3px 3px #5dabcd,

4px 5px 3px #378ab4,

5px 4px 2px #5dabcd,

5px 6px 2px #378ab4,

6px 5px 2px #5dabcd,

6px 7px 1px #378ab4,

7px 6px 1px #5dabcd,

7px 8px 0px #378ab4,

8px 7px 0px #5dabcd;

}

.glow {

color: #444;

text-shadow:

1px 0px 1px #ccc, 0px 1px 1px #eee,

2px 1px 1px #ccc, 1px 2px 1px #eee,

3px 2px 1px #ccc, 2px 3px 1px #eee,

4px 3px 1px #ccc, 3px 4px 1px #eee,

5px 4px 1px #ccc, 4px 5px 1px #eee,

6px 5px 1px #ccc, 5px 6px 1px #eee,

7px 6px 1px #ccc;

}

.vamp {

color: #92a5de;

background: red;

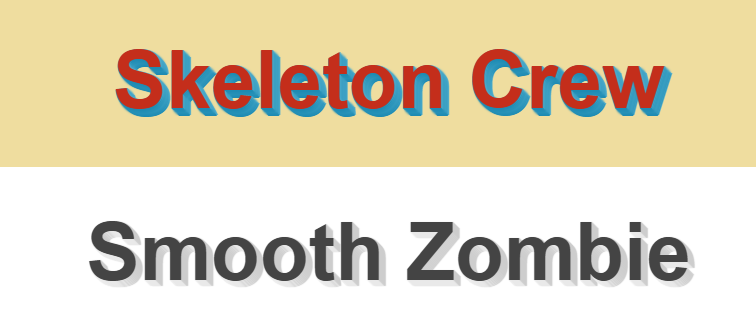
text-shadow:0px 0px 0 rgb(137,156,213),1px 1px 0 rgb(129,148,205),2px 2px 0 rgb(120,139,196),3px 3px 0 rgb(111,130,187),4px 4px 0 rgb(103,122,179),5px 5px 0 rgb(94,113,170),6px 6px 0 rgb(85,104,161),7px 7px 0 rgb(76,95,152),8px 8px 0 rgb(68,87,144),9px 9px 0 rgb(59,78,135),10px 10px 0 rgb(50,69,126),11px 11px 0 rgb(42,61,118),12px 12px 0 rgb(33,52,109),13px 13px 0 rgb(24,43,100),14px 14px 0 rgb(15,34,91),15px 15px 0 rgb(7,26,83),16px 16px 0 rgb(-2,17,74),17px 17px 0 rgb(-11,8,65),18px 18px 0 rgb(-19,0,57),19px 19px 0 rgb(-28,-9,48), 20px 20px 0 rgb(-37,-18,39),21px 21px 20px rgba(0,0,0,1),21px 21px 1px rgba(0,0,0,0.5),0px 0px 20px rgba(0,0,0,.2);

}

Outputs









**Text Warping:**

If you had written the text in between span elements if the screen size is changed it will wrap it but the single full name cannot warp in the web to do, so use the element **white-space to nowarp ex white-space:nowarp;**

Example:

Html:

<h1>My Name is Rizvi <span class=”nowarp”>Ahmed Abbas</span></h1>

CSS

.nowarp{

White-space:nowarp;

}

**Syntax**

/\* Keyword values \*/

white-space: normal;

white-space: nowrap;

white-space: pre;

white-space: pre-wrap;

white-space: pre-line;

white-space: break-spaces;

Values

**normal**

Sequences of white space are collapsed. Newline characters in the source are handled the same as other white space. Lines are broken as necessary to fill line boxes.

**nowrap**

Collapses white space as for normal, but suppresses line breaks (text wrapping) within the source.

**pre**

Sequences of white space are preserved. Lines are only broken at newline characters in the source and at <br> elements.

**pre-wrap**

Sequences of white space are preserved. Lines are broken at newline characters, at <br>, and as necessary to fill line boxes.

**pre-line**

Sequences of white space are collapsed. Lines are broken at newline characters, at <br>, and as necessary to fill line boxes.

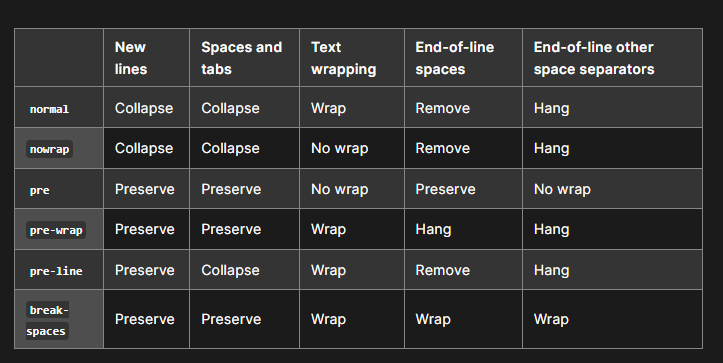
**break-spaces**

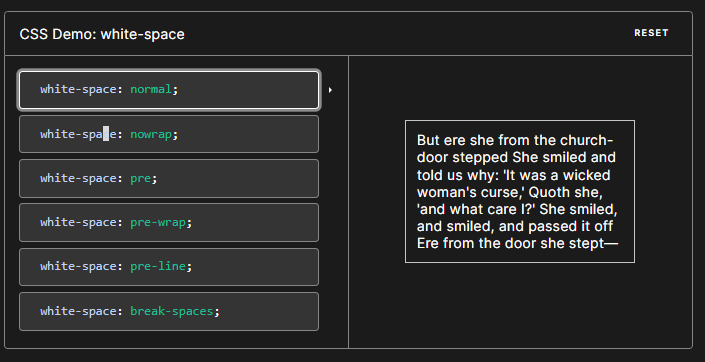
The behavior is identical to that of pre-wrap, except that:

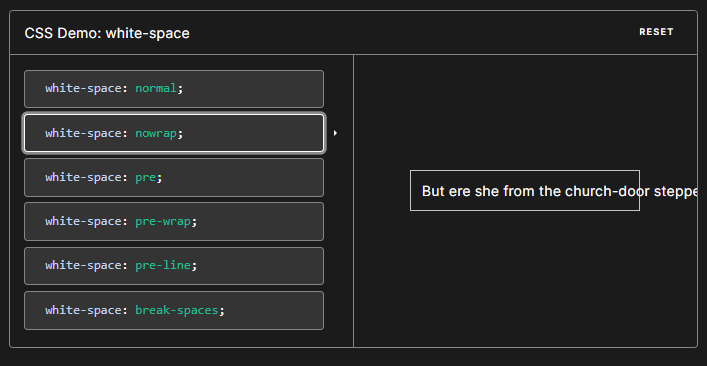
Any sequence of preserved white space always takes up space, including at the end of the line.

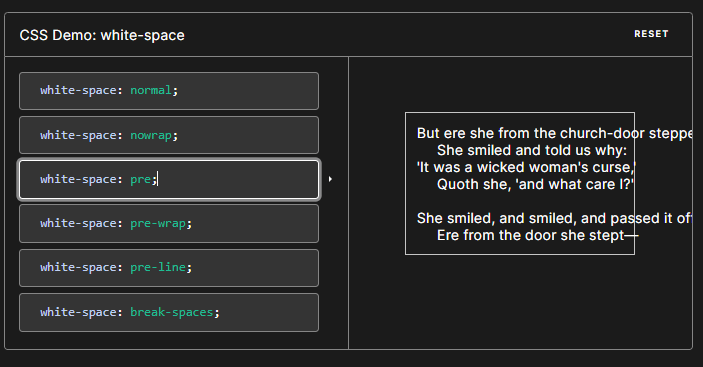
A line breaking opportunity exists after every preserved white space character, including between white space characters.

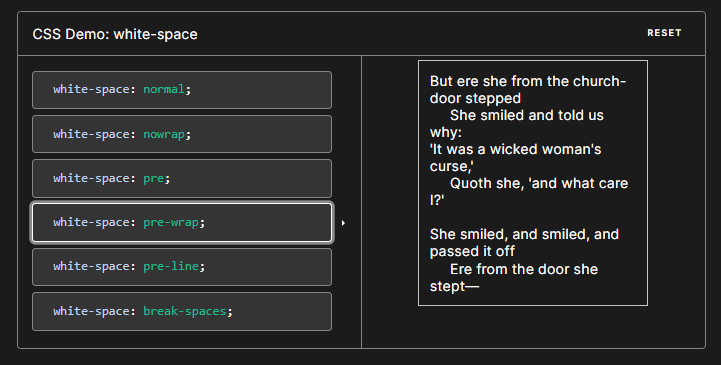
Such preserved spaces take up space and do not hang, and thus affect the box's intrinsic sizes (min-content size and max-content size).

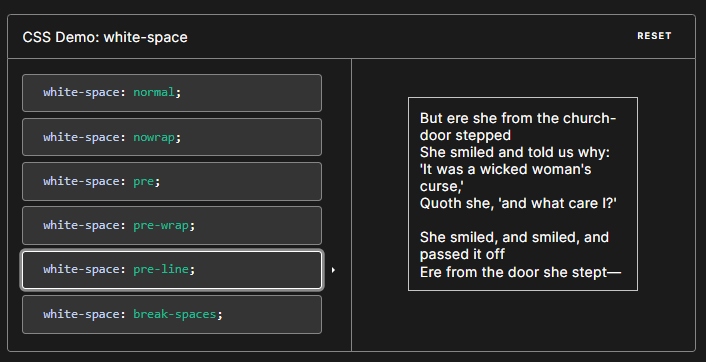


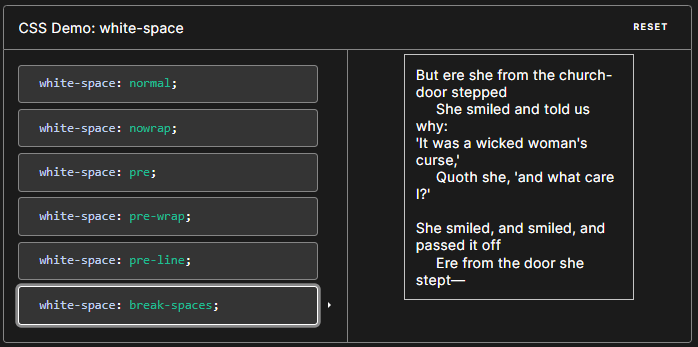












# box-shadow

The box-shadow CSS property adds shadow effects around an element's frame. You can set multiple effects separated by commas. A box shadow is described by X and Y offsets relative to the element, blur and spread radius, and color.

The box-shadow property enables you to cast a drop shadow from the frame of almost any element. If a border-radius is specified on the element with a box shadow, the box shadow takes on the same rounded corners. The z-ordering of multiple box shadows is the same as multiple text shadows (the first specified shadow is on top).

[**Syntax**](https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow#syntax)

/\* Keyword values \*/

box-shadow: none;

/\* offset-x | offset-y | color \*/

box-shadow: 60px -16px teal;

/\* offset-x | offset-y | blur-radius | color \*/

box-shadow: 10px 5px 5px black;

/\* offset-x | offset-y | blur-radius | spread-radius | color \*/

box-shadow: 2px 2px 2px 1px rgba(0, 0, 0, 0.2);

/\* inset | offset-x | offset-y | color \*/

box-shadow: inset 5em 1em gold;

/\* Any number of shadows, separated by commas \*/

box-shadow: 3px 3px red, -1em 0 0.4em olive;

/\* Global values \*/

box-shadow: inherit;

box-shadow: initial;

box-shadow: revert;

box-shadow: revert-layer;

box-shadow: unset;

Specify a single box-shadow using:

* Two, three, or four [<length>](https://developer.mozilla.org/en-US/docs/Web/CSS/length) values.
  + If only two values are given, they are interpreted as <offset-x> and <offset-y> values.
  + If a third value is given, it is interpreted as a <blur-radius>.
  + If a fourth value is given, it is interpreted as a <spread-radius>.
* Optionally, the inset keyword.
* Optionally, a [<color>](https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow#color) value.

To specify multiple shadows, provide a comma-separated list of shadows.

### [Values](https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow#values)

inset

If not specified (default), the shadow is assumed to be a drop shadow (as if the box were raised above the content). The presence of the inset keyword changes the shadow to one inside the frame (as if the content was debossed inside the box). Inset shadows are drawn inside the border (even transparent ones), above the background, but below content.

<offset-x>

The [<length>](https://developer.mozilla.org/en-US/docs/Web/CSS/length) value specifies the horizontal distance. Negative values place the shadow to the left of the element.

<offset-y>

The [<length>](https://developer.mozilla.org/en-US/docs/Web/CSS/length) values specifies the vertical distance. Negative values place the shadow above the element.

If both <offset-x> and <offset-y> are set to 0, the shadow is placed behind the element (and may generate a blur effect if <blur-radius> and/or <spread-radius> is set).

<blur-radius>

This is a third [<length>](https://developer.mozilla.org/en-US/docs/Web/CSS/length) value. The larger this value, the bigger the blur, so the shadow becomes bigger and lighter. Negative values are not allowed. If not specified, it will be 0 (the shadow's edge is sharp). The specification does not include an exact algorithm for how the blur radius should be calculated, however, it does elaborate as follows:

…for a long, straight shadow edge, this should create a color transition the length of the blur distance that is perpendicular to and centered on the shadow's edge, and that ranges from the full shadow color at the radius endpoint inside the shadow to fully transparent at the endpoint outside it.

<spread-radius>

This is a fourth [<length>](https://developer.mozilla.org/en-US/docs/Web/CSS/length) value. Positive values will cause the shadow to expand and grow bigger, negative values will cause the shadow to shrink. If not specified, it will be 0 (the shadow will be the same size as the element).

<color>

See [<color>](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value) values for possible keywords and notations. If not specified, it defaults to [currentcolor](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value#currentcolor_keyword).

### [Interpolation](https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow#interpolation)

When animating shadows, such as when multiple shadow values on a box transition to new values on hover, the values are interpolated. [Interpolation](https://developer.mozilla.org/en-US/docs/Glossary/Interpolation) determines intermediate values of properties, such as the blur radius, spread radius, and color, as shadows transition. For each shadow in a list of shadows, the color, x, y, blur, and spread transition; the color as [<color>](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value), and the other values as [<length>](https://developer.mozilla.org/en-US/docs/Web/CSS/length)s.

In interpolating multiple shadows between two comma-separated lists of multiple box shadows, the shadows are paired, in order, with interpolation occurring between paired shadows. If the lists of shadows have different lengths, then the shorter list is padded at the end with shadows whose color is transparent, and X, Y, and blur are 0, with the inset, or lack of inset, being set to match. If, in any pair of shadows, one has inset set and the other is does not, the entire shadow list is uninterpolated; the shadows will change to the new values without an animating effect.

EXAMPLE:

