

CSS 3

This is a quick reference cheat sheet for CSS goodness, listing selector syntax, properties, units and other useful bits of information.

Getting Started

CSS is rich in capabilities and is more than simply laying out pages.

External stylesheet

```
<link href=".path/to/stylesheet/style.css" rel="stylesheet" type="text/css">
```

Internal stylesheet

```
<style>
body {
    background-color: linen;
}
</style>
```

Inline styles

```
<h2 style="text-align: center;">Centered text</h2>
```

```
<p style="color: blue; font-size: 18px;">Blue, 18-point text</p>
```

Add class

```
<div class="classname"></div>
<div class="class1 ... classn"></div>
```

Support multiple classes on one element.

!important

```
.post-title {
    color: blue !important;
}
```

Overrides all previous styling rules.

Selector

```
h1 { }
#job-title { }
div.hero { }
div > p { }
```

See: [Selectors](#)

Text color

```
color: #2a2aff;  
color: green;  
color: rgb(34, 12, 64, 0.6);  
color: hsla(30 100% 50% / 0.6);
```

See: [Colors](#)

Background

```
background-color: blue;  
background-image: url("nyan-cat.gif");  
background-image: url("../image.png");
```

See: [Backgrounds](#)

Font

```
.page-title {  
    font-weight: bold;  
    font-size: 30px;  
    font-family: "Courier New";  
}
```

See: [Fonts](#)

Position

```
.box {  
    position: relative;  
    top: 20px;  
    left: 20px;  
}
```

See also: [Position](#)

Animation

```
animation: 300ms linear 0s infinite;  
  
animation: bounce 300ms linear infinite;
```

See: [Animation](#)

Comment

```
/* This is a single line comment */  
  
/* This is a  
   multi-line comment */
```

Flex layout

```
div {  
  display: flex;  
  justify-content: center;  
}  
div {  
  display: flex;  
  justify-content: flex-start;  
}
```

See: [Flexbox](#) | [Flex Tricks](#)

Grid layout

```
#container {  
  display: grid;  
  grid: repeat(2, 60px) / auto-flow 80px;  
}  
  
#container > div {  
  background-color: #8ca0ff;  
  width: 50px;  
  height: 50px;  
}
```

See: [Grid Layout](#)

Variable & Counter

```
counter-set: subsection;  
counter-increment: subsection;  
counter-reset: subsection 0;  
  
:root {  
  --bg-color: brown;  
}  
element {
```

```
background-color: var(--bg-color);  
}
```

See: Dynamic content

CSS Selectors

Examples

Groups Selector

```
h1, h2 {  
  color: red;  
}
```

Chaining Selector

```
h3.section-heading {  
  color: blue;  
}
```

Attribute Selector

```
div[attribute="SomeValue"] {  
  background-color: red;  
}
```

First Child Selector

```
p:first-child {  
  font-weight: bold;  
}
```

No Children Selector

```
.box:empty {  
  background: lime;  
  height: 80px;  
  width: 80px;  
}
```

Basic

*

All elements

<code>div</code>	All div tags
<code>.classname</code>	All elements with class
<code>#idname</code>	Element with ID
<code>div, p</code>	All divs and paragraphs
<code>#idname *</code>	All elements inside #idname

See also: Type / Class / ID / Universal selectors

Combinators

<code>div.classname</code>	Div with certain classname
<code>div#idname</code>	Div with certain ID
<code>div p</code>	Paragraphs inside divs
<code>div > p</code>	All p tags one level deep in div
<code>div + p</code>	P tags immediately after div
<code>div ~ p</code>	P tags preceded by div

See also: Adjacent / Sibling / Child selectors

Attribute selectors

<code>a[target]</code>	With a target attribute
<code>a[target="_blank"]</code>	Open in new tab
<code>a[href^="/index"]</code>	Starts with /index
<code>[class = "chair"]</code>	Starts with chair
<code>[class*= "chair"]</code>	containing chair
<code>[title~= "chair"]</code>	Contains the word chair
<code>a[href\$=".doc"]</code>	Ends with .doc
<code>[type= "button"]</code>	Specified type

See also: Attribute selectors

User action pseudo classes

a:link	Link in normal state
a:active	Link in clicked state
a:hover	Link with mouse over it
a:visited	Visited link

Pseudo classes

p::after	Add content after p
p::before	Add content before p
p::first-letter	First letter in p
p::first-line	First line in p
::selection	Selected by user
::placeholder	Placeholder attribute
:root	Documents root element
:target	Highlight active anchor
div:empty	Element with no children
p:lang(en)	P with en language attribute
:not(span)	Element that's not a span

Input pseudo classes

input:checked	Checked inputs
input:disabled	Disabled inputs
input:enabled	Enabled inputs
input:focus	Input has focus
input:in-range	Value in range
input:out-of-range	Input value out of range
input:valid	Input with valid value
input:invalid	Input with invalid value

<code>input:optional</code>	No required attribute
<code>input:required</code>	Input with required attribute
<code>input:read-only</code>	With readonly attribute
<code>input:read-write</code>	No readonly attribute
<code>input:indeterminate</code>	With <code>indeterminate</code> state

Structural pseudo classes

<code>p:first-child</code>	First child
<code>p:last-child</code>	Last child
<code>p:first-of-type</code>	First of some type
<code>p:last-of-type</code>	Last of some type
<code>p:nth-child(2)</code>	Second child of its parent
<code>p:nth-child(3n+2)</code>	Nth-child (an + b) formula
<code>p:nth-last-child(2)</code>	Second child from behind
<code>p:nth-of-type(2)</code>	Second p of its parent
<code>p:nth-last-of-type(2)</code>	...from behind
<code>p:only-of-type</code>	Unique of its parent
<code>p:only-child</code>	Only child of its parent

CSS Fonts

Properties

<code>font-family:</code>	<code></code>
<code>font-size:</code>	<code><size></code>
<code>letter-spacing:</code>	<code><size></code>
<code>line-height:</code>	<code><number></code>

font-weight: <number> / bold / normal

font-style: italic / normal

text-decoration: underline / none

text-align: left / right
center / justify

text-transform: capitalize / uppercase / lowercase

See also: [Font](#)

Shorthand

font:	italic	400	14px	/	1.5	sans-serif
style	weight	size (required)		line-height		family (required)

Example

```
font-family: Arial, sans-serif;  
font-size: 12pt;  
letter-spacing: 0.02em;
```

Case

```
/* Hello */  
text-transform: capitalize;  
  
/* HELLO */  
text-transform: uppercase;  
  
/* hello */  
text-transform: lowercase;
```

@font-face

```
@font-face {  
    font-family: 'Glegoo';  
    src: url('../Glegoo.woff');  
}
```

CSS Colors

Named color

```
color: red;  
color: orange;  
color: tan;  
color: rebeccapurple;
```

Hexadecimal color

```
color: #090;  
color: #009900;  
color: #090a;  
color: #009900aa;
```

rgb() Colors

```
color: rgb(34, 12, 64, 0.6);  
color: rgba(34, 12, 64, 0.6);  
color: rgb(34 12 64 / 0.6);  
color: rgba(34 12 64 / 0.3);  
color: rgb(34.0 12 64 / 60%);  
color: rgba(34.6 12 64 / 30%);
```

HSL Colors

```
color: hsl(30, 100%, 50%, 0.6);  
color: hsla(30, 100%, 50%, 0.6);  
color: hsl(30 100% 50% / 0.6);  
color: hsla(30 100% 50% / 0.6);  
color: hsl(30.0 100% 50% / 60%);  
color: hsla(30.2 100% 50% / 60%);
```

Other

```
color: inherit;  
color: initial;  
color: unset;  
color: transparent;  
  
color: currentcolor; /* keyword */
```

CSS Backgrounds

Properties

background:	(Shorthand)
background-color:	See: Colors
background-image:	url(...)
background-position:	left/center/right top/center/bottom
background-size:	cover X Y
background-clip:	border-box padding-box content-box
background-repeat:	no-repeat repeat-x repeat-y
background-attachment:	scroll/fixed/local

Shorthand

background:	#ff0	url(a.jpg)	left	top	/	100px auto	no- repeat	fixed;
background:	#abc	url(b.png)	center	center	/	cover	repeat- x	local;
	color	image	posX	posY		size	repeat	attach..

Examples

```
background: url(img_man.jpg) no-repeat center;
```

```
background: url(img_flwr.gif) right bottom no-repeat, url(paper.gif) left top  
repeat;
```

```
background: rgb(2,0,36);  
background: linear-gradient(90deg, rgba(2,0,36,1) 0%, rgba(13,232,230,1) 35%,  
rgba(0,212,255,1) 100%);
```

CSS The Box Model

Maximums/Minimums

```
.column {  
    max-width: 200px;  
    width: 500px;  
}
```

See also: [max-width](#) / [min-width](#) / [max-height](#) / [min-height](#)

Margin / Padding

```
.block-one {  
    margin: 20px;  
    padding: 10px;  
}
```

See also: [Margin](#) / [Padding](#)

Box-sizing

```
.container {  
    box-sizing: border-box;  
}
```

See also: [Box-sizing](#)

Visibility

```
.invisible-elements {  
    visibility: hidden;  
}
```

See also: [Visibility](#)

Auto keyword

```
div {  
    margin: auto;  
}
```

See also: Margin

Overflow

```
.small-block {  
    overflow: scroll;  
}
```

See also: [Overflow](#)

CSS Animation

Shorthand

animation:	bounce	300ms	linear	100ms	infinite	alternate-reverse	both
	name	duration	timing-function	delay	count	direction	fill-mode

Properties

animation:	(shorthand)
animation-name:	<name>
animation-duration:	<time>ms
animation-timing-function:	ease / linear / ease-in / ease-out / ease-in-out
animation-delay:	<time>ms
animation-iteration-count:	infinite / <number>
animation-direction:	normal / reverse / alternate / alternate-reverse
animation-fill-mode:	none / forwards / backwards / both / initial / inherit
animation-play-state:	normal / reverse / alternate / alternate-reverse

See also: [Animation](#)

Example

```
/* @keyframes duration | timing-function | delay |
   iteration-count | direction | fill-mode | play-state | name */
animation: 3s ease-in 1s 2 reverse both paused slidein;

/* @keyframes duration | timing-function | delay | name */
animation: 3s linear 1s slidein;

/* @keyframes duration | name */
animation: 3s slidein;

animation: 4s linear 0s infinite alternate move_eye;
animation: bounce 300ms linear 0s infinite normal;
animation: bounce 300ms linear infinite;
animation: bounce 300ms linear infinite alternate-reverse;
```

Javascript Event

```
.one('webkitAnimationEnd oanimationend msAnimationEnd animationend')
```

CSS Flexbox

Simple example

```
.container {
  display: flex;
}

.container > div {
  flex: 1 1 auto;
}
```

Container

```
.container {

  display: flex;
  display: inline-flex;

  flex-direction: row;          /* ltr - default */
  flex-direction: row-reverse;  /* rtl */
```

```
flex-direction: column;           /* top-bottom */
flex-direction: column-reverse; /* bottom-top */

flex-wrap: nowrap; /* one-line */
flex-wrap: wrap;   /* multi-line */

align-items: flex-start; /* vertical-align to top */
align-items: flex-end;  /* vertical-align to bottom */
align-items: center;   /* vertical-align to center */
align-items: stretch;  /* same height on all (default) */

justify-content: flex-start; /* [xxx] */
justify-content: center;    /* [xxx] */
justify-content: flex-end;  /* [xxx] */
justify-content: space-between; /* [x x x] */
justify-content: space-around; /* [ x x x ] */
justify-content: space-evenly; /* [ x x x ] */
```

Child

```
.container > div {

/* This: */
flex: 1 0 auto;

/* Is equivalent to this: */
flex-grow: 1;
flex-shrink: 0;
flex-basis: auto;

order: 1;

align-self: flex-start; /* left */
margin-left: auto;      /* right */

}
```

CSS Flexbox Tricks

Vertical center

```
.container {  
  display: flex;  
}  
  
.container > div {  
  width: 100px;  
  height: 100px;  
  margin: auto;  
}
```

Vertical center (2)

```
.container {  
  display: flex;  
  
/* vertical */  
align-items: center;  
  
/* horizontal */  
justify-content: center;  
}
```

Reordering

```
.container > .top {  
  order: 1;  
}  
  
.container > .bottom {  
  order: 2;  
}
```

Mobile layout

```
.container {  
  display: flex;  
  flex-direction: column;  
}  
  
.container > .top {  
  flex: 0 0 100px;  
}  
  
.container > .content {
```

```
    flex: 1 0 auto;  
}
```

Table-like

```
.container {  
  display: flex;  
}  
  
/* the 'px' values here are just suggested percentages */  
.container > .checkbox { flex: 1 0 20px; }  
.container > .subject { flex: 1 0 400px; }  
.container > .date { flex: 1 0 120px; }
```

This creates columns that have different widths, but size accordingly according to the circumstances.

Vertical

```
.container {  
  align-items: center;  
}
```

Vertically-center all items.

Left and right

```
.menu > .left { align-self: flex-start; }  
.menu > .right { align-self: flex-end; }
```

CSS Grid Layout

Grid Template Columns

```
#grid-container {  
  display: grid;  
  width: 100px;  
  grid-template-columns: 20px 20% 60%;  
}
```

fr Relative Unit

```
.grid {  
  display: grid;  
  width: 100px;  
  grid-template-columns: 1fr 60px 1fr;  
}
```

Grid Gap

```
/*The distance between rows is 20px*/  
/*The distance between columns is 10px*/  
#grid-container {  
  display: grid;  
  grid-gap: 20px 10px;  
}
```

CSS Block Level Grid

```
#grid-container {  
  display: block;  
}
```

CSS grid-row

CSS syntax:

- grid-row: grid-row-start / grid-row-end;

Example

```
.item {  
  grid-row: 1 / span 2;  
}
```

CSS Inline Level Grid

```
#grid-container {  
  display: inline-grid;  
}
```

minmax() Function

```
.grid {  
  display: grid;
```

```
grid-template-columns: 100px minmax(100px, 500px) 100px;  
}
```

grid-row-start & grid-row-end

CSS syntax:

- grid-row-start: auto|row-line;
- grid-row-end: auto|row-line|span n;

Example

```
grid-row-start: 2;  
grid-row-end: span 2;
```

CSS grid-row-gap

```
grid-row-gap: length;
```

Any legal length value, like px or %. 0 is the default value

CSS grid-area

```
.item1 {  
  grid-area: 2 / 1 / span 2 / span 3;  
}
```

Justify Items

```
#container {  
  display: grid;  
  justify-items: center;  
  grid-template-columns: 1fr;  
  grid-template-rows: 1fr 1fr 1fr;  
  grid-gap: 10px;  
}
```

CSS grid-template-areas

```
.item {  
  grid-area: nav;  
}  
.grid-container {  
  display: grid;
```

```
grid-template-areas:  
  'nav nav . .'  
  'nav nav . .';  
}
```

Justify Self

```
#grid-container {  
  display: grid;  
  justify-items: start;  
}  
  
.grid-items {  
  justify-self: end;  
}
```

The grid items are positioned to the right (end) of the row.

Align Items

```
#container {  
  display: grid;  
  align-items: start;  
  grid-template-columns: 1fr;  
  grid-template-rows: 1fr 1fr 1fr;  
  grid-gap: 10px;  
}
```

CSS Dynamic Content

Variable

Define CSS Variable

```
:root {  
  --first-color: #16f;  
  --second-color: #ff7;  
}
```

Variable Usage

```
#firstParagraph {  
  background-color: var(--first-color);  
  color: var(--second-color);  
}
```

Counter

```
/* Set "my-counter" to 0 */  
counter-set: my-counter;  
  
/* Increment "my-counter" by 1 */  
counter-increment: my-counter;  
  
/* Decrement "my-counter" by 1 */  
counter-increment: my-counter -1;  
  
/* Reset "my-counter" to 0 */  
counter-reset: my-counter;
```

See also: [Counter set](#)

Using counters

```
body { counter-reset: section; }  
  
h3::before {  
  counter-increment: section;  
  content: "Section." counter(section);  
}  
  
ol {  
  counter-reset: section;  
  list-marker-type: none;  
}  
  
li::before {  
  counter-increment: section;  
  content: counters(section, ".") " ";  
}
```

Css 3 tricks

Scrollbar smooth

```
html {  
  scroll-behavior: smooth;  
}
```

[Click me, the page will scroll smoothly to Getting started](#)

Also see

[CSS selectors cheatsheet](#) (frontend30.com)
[MDN: Using CSS flexbox](#)
[Ultimate flexbox cheatsheet](#)
[GRID: A simple visual cheatsheet](#)
[CSS Tricks: A Complete Guide to Grid](#)
[Browser support](#)

Related Cheatsheet

[HTML Cheatsheet](#)

Quick Reference

[JavaScript Cheatsheet](#)

Quick Reference

[jQuery Cheatsheet](#)

Quick Reference

[Laravel Cheatsheet](#)

Quick Reference

Recent Cheatsheet

[Remote Work Revolution Cheatsheet](#)

Quick Reference

[Homebrew Cheatsheet](#)

Quick Reference

[PyTorch Cheatsheet](#)

[Taskset Cheatsheet](#)

© 2023 QuickRef.ME, All rights reserved.