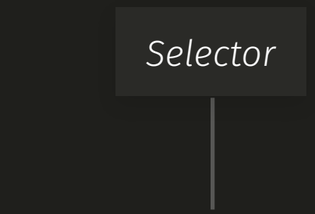
CSS Cheat Sheet



CSS “Rule” syntax

All CSS *Rules* start with a *selector*, followed by a pair of squiggly brackets ({ }).

The whitespace around a CSS rule is not required, but highly encouraged. We’ll see in a moment that these rules can get complex, quickly.

```
div {

}
```

CSS Rule matching all `div` *tags.*

This rule will apply all its contained style information (currently nothing) to all `div` tags in any document that includes it with the `link` tag.

CSS property-value syntax.

Rules contain pairs of *properties* and *values*.

Each should be written on its own line, and indented. Properties are separated from their value by a (`:`), and end with a semicolon (`;`).

```
div {
  background-color: red;
}
```

An example of the `background-color` *property set to* `red`.

Most properties take a single value.

This rule would make the background color of all `div` elements red.

```
div {
  border: 2px solid black;
}
```

CSS property-value syntax for a property that accepts multiple values.

The `border` property accepts a group of three properties, a `width`, `style` and `color`.

The Mozilla Developer Network documents all the valid values for each property. Reference it often!

/* Comment */

Comment syntax in CSS.

Comments are bits of code that aren't interpreted when the page loads. Use comments to take notes about your work, or explain to your future self why you made certain descisions. Humor is encouraged!

CSS Comments open with the slash followed by an asterisk (`/*`) and close with the inverse, an asterisk followed by a slash (`*/`).

```
/* Comment */

div {
  border: 2px solid black;
}

/*
  Comments can span multiple lines,
  if you're feeling chatty.
*/
```

A multi-line comment in CSS.

Comments can span multiple lines.

```
/*
div {
  border: 2px solid black;
}
*/
```

A comment that disables a CSS rule.

CSS rules aren’t interpreted when they’re inside a comment.

```
div {
  border: 2px solid black;
  /* font-size: 40px; */
}
```

A single CSS property commented out.

Single lines of CSS rules can be commented out, if you want to revert a change quickly, but not destroy all record of it.

Use `Command+/` to comment a single line in Sublime Text. With mutliple lines selected, you can use the same command to wrap it in one big comment. With your cursor inside a comment, do the same command to uncomment it.

```html
<ul>
  <li>Fruit</li>
  <li>Vegetables</li>
  <li>Meat</li>
</ul>

<ol>
  <li>Learn HTML</li>
  <li>Learn CSS</li>
  <li>Build a Website!</li>
</ol>
```

```css
ul, ol {
  background-color: #DDD;
  padding: 1em 1em 1em 2em;
}
```

Sometimes, we'll want to style multiple different tags similarly, like lists.

A grouped selector allows us to apply the same styles to multiple different selectors.

Notice the new color syntax, as well. We'll see more of that on page 21.

```html
<div class="square"></div>
```

```css
.square {
  background-color: red;
  width: 100px;
  height: 100px;
}
```

A CSS selector that matches all elements with the class of `square`

Adding a dot before the selector makes this rule apply to any elements that have the class of `square`.

This rule happens to give them a height and width of 100 pixels and a fill of red.

```
<blockquote class="fancy-text">
  <p>
     How would you expect this to be styled,
     if you could only see the HTML?
  </p>
</blockquote>
```

Use classes to be descriptive about the element's contents.

Here, we've left ourselves a hint about how the content will be styled.

```
.fancy-text {
  font-family: cursive;
}
```

A reasonable expected style declaration.

Sure! Maybe it’s cursive.

```html
<p>Normal paragraph.</p>

<p class="bigger">
  Paragraph with larger text.
</p>

<p>Normal paragraph.</p>
```

```css
p {
  font-size: 16px;
  margin-bottom: 20px;
}

.bigger {
  font-size: 32px;
}
```

The “Cascade”

Sometimes, we address the same element with multiple CSS rules.

Here, all three paragraphs will have the styles set in the `p` rule, but only the one with a class of `bigger` will receive styles from the `.bigger` rule. It still is given the `margin-bottom` from the more general `p` selector.

The `font-size` is overridden, but `margin-bottom` isn’t, because there isn’t a conflict.

```
div {
  width: 100px;
  height: 100px;
  color: #CCC;
}

div.blue {
  background-color: #00F;
}

div.yellow {
  background-color: #FF0;
}
```

CSS Combined Selectors

You can chain CSS selectors together to put additional constraints on

```
<p>Paragraph set at 16 pixels.</p>

<blockquote>
  <p>Paragraph set at 32 pixels.</p>
</blockquote>

<p>Another normal-sized paragraph.</p>
```



```
/* General Rule */

p {
  font-size: 16px;
}

/* Descendent Selector */

blockquote p {
  font-size: 32px;
}
```

The descendant selector

Using a space between selectors causes the style to only apply to elements found inside of other elements. Styles in the rule only apply to the last selector in the list— here, the rule can be read as “All paragraphs inside blockquotes.”

Descendent selectors allow us to specify contextual exceptions to more general rules.

```
a {
  color: #F00;
}
```



```
a:hover {
    color: #E44;
}
```

CSS pseudo-selectors.

A *pseudo-selector* styles elements that match certain other conditions, usually concerning user input or its context on the page.

Hover states— indicated in CSS by appending `:hover` to a selector— are partiularly useful when stlying links, to give users an additional queue that an element is interactive.

```
a:hover { ... }

li:last-of-type { ... }

h3:first-child { ... }

.navigation a:hover { ... }

.footer:hover a { ... }
```

A variety of pseudo-selectors, attached to different elements at different points in a CSS selector.

Pseudo-selectors may be attached to most any selector, in just about any situation.

```html
<a class="box" href="#">Doesn’t match!</a>

<div>Doesn’t match.</div>

<div class="box">I’m a square!</div>

<div class="blue">
  I’m not a box, nor am I blue!
</div>

<div class="box blue">Woo hoo!</div>
```



```css
div.box {
  width: 100px;
  height: 100px;
  color: #CCC;
}

.box.blue {
  background-color: #00F;
}

.box.yellow {
  background-color: #FF0;
}
```

CSS Combined Selectors

You can chain CSS selectors together to put additional constraints on which elements can use which classes and expect to be styled, as well as require that certain classes be used together. In this case, `.blue` could be used again to describe `a` tags in a different way, by specifying a rule like `a.blue`

```css
/* 1. Color Keyword */
.keyword {
  background-color: red;
}

/* 2. "Hex" Color */
.hex {
  background-color: #F00;
}

/* 3. "Hex" Color (Expanded) */
.hex-expanded {
  background-color: #FF0000;
}
```



```css
/* 4. RGB (Red, Green, Blue) */
.rgb {
  background-color: rgb(255,0,0);
}

/* 5. RGBa (RGB with opacity!) */
.rgba {
  background-color: rgba(255,0,0,0.5);
}
```

CSS color declaration

Colors can be specified in a number of ways in CSS.

We've used things like `#1` so far, but we can't name all the colors. In most cases, it'll be easiest to translate our designs using RGB color. All these examples (except `#5`) will display the same color.

The selectors were only chosen to describe the rule, and have no specific meaning, here.