CSS.PROGRAMMERS.IM

Which rule wins?

When two CSS rules clash, there is an official way to decide.

MDN's lesson on handling conflicts says the first word in Cascading Style Sheets — *cascading* — is key. The cascade, and the closely related idea of specificity, "are mechanisms that control which rule applies when such a conflict occurs."[1]

Stylesheets cascade

"At a very simple level, this means that the origin and the order of CSS rules matter. When two rules both have equal specificity, the one that is defined last in the stylesheet is the one that will be used."[1]

MDN's official example has two rules that could apply to an `h1`. Both come from the same source and have the same selector, so they carry the same specificity. The last one in the source order wins, and the heading ends up blue:[1]

```
h1 { color: red; }
h1 { color: blue; }
```

The second rule is defined last, so the text is blue.

The same lesson also covers inheritance: some CSS properties inherit values set on the parent by default, and some don't, which can cause unexpected behaviour. Cascade layers are mentioned, but not covered in detail.[1]

Three factors, in order of importance

When more than one style block sets the same property on the same element, MDN lists three factors, "listed here in increasing order of importance. Later ones overrule earlier ones":[1]

Factor	What it means
1. Source order	If rules have exactly the same weight, "the one that comes last in the CSS will win." Source order only matters when specificity is the same.[1]
2. Specificity	A later rule can still lose if an earlier one is more specific. A class selector has more weight than an element selector. Only the properties declared in more than one place are overwritten, not the whole rule.[1]
3. Importance	Inline styles (a style attribute) take precedence over all normal styles, no matter the specificity. The `!important` flag can overrule even those, including normal inline styles. MDN recommends you never use it unless you absolutely have to.[1]

How specificity is scored

The amount of specificity a selector has is measured using three values, "which can be thought of as ID, CLASS, and ELEMENT columns worth hundreds, tens, and ones, respectively." IDs score in the first column (100 points each), classes, attribute selectors and pseudo-classes in the second (10 points), and element selectors or pseudo-elements in the third (1 point). The universal selector, combinators, and `:where()` have no effect on specificity.[1]

Selector	IDs	Classes	Elements	Total
`h1`	0	0	1	0-0-1
`h1 + p::first-letter`	0	0	3	0-0-3
`li > a[href*="en-US"] > .inline-warning`	0	2	2	0-2-2
`#identifier`	1	0	0	1-0-0

Quick quiz

1. Two rules target the same element, with equal specificity. Which one is used?
2. What two official mechanisms decide a conflict between style declarations?
3. Do all CSS properties inherit from the parent element by default?

Answers. 1. The one defined last in the stylesheet.[1] 2. The cascade, and specificity.[1] 3. No — some inherit by default and some don't.[1]

Source, checked 17 November 2026. All wording, the red-then-blue `h1` example, the three factors, and the specificity table are from MDN Learn: Handling conflicts. https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Styling_basics/Handling_conflicts

ReallyByte — Fun. Educational. Useless.™ · css.programmers.im · Free educational fact sheet. Not affiliated with MDN or Mozilla.