ScopeTreasure Courses

CSS COURSES · LESSON 2.2 · CODE SCOPE SERIES

Selectors (element, class, id, attribute)

CSS finds elements by address: element, .class, #id, [attribute], groups and descendants — and the specificity ladder that decides conflicts.

Introduction

Before CSS can style anything, it must **find** it. A selector is the address you write at the start of a rule — it tells the browser exactly which elements the style belongs to. Choose the address well and one small rule can style a hundred elements; choose it badly and you will fight your own page. This lesson covers the five everyday selectors and the **specificity ladder** — the weight system that settles every argument between rules. *(Roman Urdu: Selector woh address hai jis se CSS sahi element tak pohanchti hai.)*

Key points

- **Element selector** — just the tag name: `p { ... }`. It styles **every** element of that type on the page. Broad and blunt.
- **Class selector** — a dot plus the name you invented: `.note { ... }`, used in HTML as `class="note"`. One class can be shared by **many** elements, and one element can carry several classes (`class="card big"`). This is the workhorse of CSS.
  (Roman Urdu: Class aik group ka naam hai — jis par lagaya, wohi style mila.)
- **ID selector** — a hash plus the name: `#header { ... }`, used as `id="header"`. An id must be **unique on the page** — like a CNIC number, one per person. Use ids rarely in CSS; they are heavy (see the ladder below).
- **Attribute selector** — square brackets target elements by an attribute: `input[type="text"] { ... }` styles only text inputs, not checkboxes.
- **Grouping** — one rule, many selectors, separated by commas: `h1, h2, h3 { font-family: Georgia, serif; }`.
- **Descendant** — a space means "inside": `nav a { ... }` styles links that live inside a `nav`, and no others. `.card p` = paragraphs inside cards.
- **Universal selector** — `* { margin: 0; padding: 0; }` touches everything; famous as the classic reset line.
- **Specificity — the weight ladder:** element = **1**, class (and attribute) = **10**, id = **100**, inline style = above all. When two rules style the same element, add up each selector's weight — the heavier rule wins, whatever order the file is in. Equal weight? The **later** rule wins.

Code examples

Example 1 — Element, class and id on one page

```
p { color: #444444; }
.highlight { background-color: #fff3b0; }
#main-title { color: #0b6e4f; font-size: 36px; }
```

```
<h1 id="main-title">My Notes</h1>
<p>First paragraph.</p>
<p class="highlight">Important paragraph.</p>
<p>Last paragraph.</p>
```

What it does: The element rule colours **all three** paragraphs grey. The class rule gives only the middle one a pale yellow background. The id rule styles the one heading that owns `id="main-title"`.

What you will see: a green 36px heading, three grey paragraphs, and a yellow highlight behind the middle one. One class, reused anywhere; one id, used once.

Example 2 — Attribute, grouping and descendant

```
h1, h2, h3 { font-family: Georgia, serif; }
nav a { color: white; text-decoration: none; }
input[type="text"] { border: 2px solid #0b6e4f; }
input[type="checkbox"] { width: 18px; height: 18px; }
```

What it does: The group rule gives all three heading levels one font in a single line. `nav a` reaches only links inside the nav — a link in a paragraph keeps the default blue. The two attribute rules treat the two input types differently: text fields get a green border, checkboxes just get a size.

What you will see: serif headings, white nav links, green-bordered text boxes and normal small checkboxes. Change `nav a` to just `a` and **every** link on the page turns white — that is the power (and danger) of a broad address.

Example 3 — The specificity battle

```
p { color: blue; }          /* weight 1   */
.note { color: green; }     /* weight 10  */
#special { color: red; }    /* weight 100 */
```

```
<p class="note" id="special">Which colour am I?</p>
```

What it does: All three rules match this one paragraph. The browser adds the weights: the id rule (100) beats the class rule (10), which beats the element rule (1).

What you will see: the text is **red**. Now add `style="color: orange;"` to the tag and it turns **orange** — inline style sits above the whole ladder. Open DevTools and hover the rules: the losing rules appear crossed out, with the weights decided for you.

Quick recap

- Addresses: **element** (every tag), **.class** (reusable group), **#id** (unique, heavy), **[attribute]** (by attribute value), **grouping** (commas), **descendant** (space = inside), ***** (everything).
- Specificity ladder: element **1** · class/attribute **10** · id **100** · inline **top**. Heavier wins; equal weight → later rule wins.

Common mistakes

1. **Using an id for styling many elements** — an id is one-per-page. The moment you copy that element, the HTML is invalid and the style turns unpredictable. Groups are classes' job.
2. **Forgetting the dot or the hash** — `note { }` looks for a `<note>` element that does not exist; `.note` finds the class, `#note` finds the id. The style "doing nothing" is usually this one character.
3. **Fighting with heavier and heavier selectors** — if a class rule will not apply, the fix is not an id or `!important`; something heavier is already winning. Check in DevTools, then write a rule of the right weight.

Practice

1. Build a small marks-table page and style it using each selector type exactly once: element for the table, a class for the topper row, an id for the title, an attribute selector for `input[type="number"]`, and a descendant rule for cells inside the table body.
2. Specificity prediction drill: write three rules that all target one paragraph (element, class, id). Predict the winning colour before opening the page — then add an inline style and predict again.

Practice quiz

1. What does a class selector like .note style?

- One single element only
- Every element that carries that class
- Only the first element on the page
- Only elements inside a nav

Answer: Every element that carries that class

2. What is the rule for an id selector like #header?

- It can be reused on many elements
- It must be unique, one element per page

- It must be written with a dot
- It only works on headings

Answer: It must be unique, one element per page

3. Three rules match one paragraph: p (weight 1), .note (weight 10), #special (weight 100). Which colour wins?

- The p rule, because it comes first
- The .note rule, because classes beat elements
- The #special rule, because 100 beats 10 and 1
- None, the browser picks at random

Answer: The #special rule, because 100 beats 10 and 1

Watch the video lesson: https://youtu.be/zhVCktQoQW8

Read online: https://scopetreasure.com/notes/css-selectors · More free tools and notes at https://scopetreasure.com