---

1. What "Cascading" Means

- CSS stands for *Cascading Style Sheets* because styles *cascade* or "flow down" when multiple rules conflict.
- When several rules target the same element, the browser decides which one *wins* based on hierarchy.
- Think of it like a waterfall: the lower levels overwrite styles from the higher ones.

---

2. Example of the Cascade

- If all <li> items are set to **green** in the external CSS, they'll all appear green.
- But if one <li> has an **inline style** setting color: red;, that particular item will appear **red**.
- Why? Inline styles override external or internal CSS — that's the cascade at work.

---

3. Four Factors That Decide Which Style Wins

The **cascade hierarchy** depends on four main factors:

a) Position

- If two CSS rules target the same element and property, **the rule written later** in the file takes precedence.
- li { color: red; }
- li { color: blue; } /* This wins */
- The lower the rule appears in your CSS file, the *stronger* it is.

b) Specificity

Specificity is how *precisely* a selector targets an element.
Order of specificity (from weakest to strongest):

1. **Element selector** (e.g., li)
2. **Class selector** (e.g., .first-class)
3. **Attribute selector** (e.g., [draggable])
4. **ID selector** (e.g., #first-id)

- The **ID selector** has the highest specificity since an ID is unique per page.
- Example outcome:
    - li { color: green; }
    - .first-class { color: purple; }

- #first-id { color: orange; }
  → Final color: **orange**

---

c) Type (Origin of CSS)

The source of the CSS rule also affects its priority:

1. **External stylesheet** (linked file)
2. **Internal stylesheet** (inside <style> tags)
3. **Inline styles** (inside the HTML element itself)

Order of power (from weakest to strongest):
External < Internal < Inline

- Inline CSS overrides internal or external CSS.

---

d) Importance

- Adding !important after a value makes that rule *outrank everything else*.
- h1 { color: red !important; }
- Even if an inline style or ID rule exists, !important still wins.
- Use this carefully; it can cause conflicts later.

---

4. The Master Cascade Summary

Order of evaluation:

1. Position (lower down = higher priority)
2. Specificity (ID > Attribute > Class > Element)
3. Type (Inline > Internal > External)
4. Importance (!important beats all)

If all else is equal, the browser reads from **top to bottom**, applying the *last valid rule* it finds.

Why It Matters

- Understanding the cascade helps you debug “why my CSS isn’t working.”
- Always analyze from *top to bottom*:
    1. Is there a more specific selector?
    2. Is there an inline style overriding it?
    3. Is !important being used somewhere?

Once you understand this flow, CSS stops feeling random — every visual result has a reason behind it.