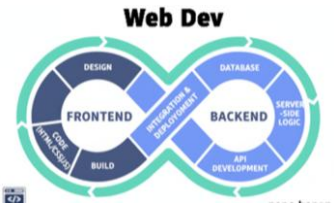
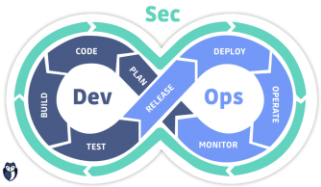




Day 18

"Web Development + Security"

CSS Specificity & Cascade

A very simple example of using without any CSS properties:

```
index.html > html > body > div
  <html lang="en">
  <head>
  </head>
  <body>
      <div> Aditya is a boy</div>
  </body>
  </html>
```

Aditya is a boy

Now, we applied many properties to one , we can see that only the last property is applied. But why?

```
index.html > html
  <html lang="en">
  <head>
      <style>
          div{
              color: aqua;
          }
          .yellow{
              color: yellow;
          }
          .cred{
              color: red;
          }
          .purple{
              color: purple;
          }
      </style>
  </head>
  <body>
      <div class="yellow cred purple"> Aditya is a boy</div>
  </body>
  </html>
```

Aditya is a boy

The above example is an example of CSS cascade.

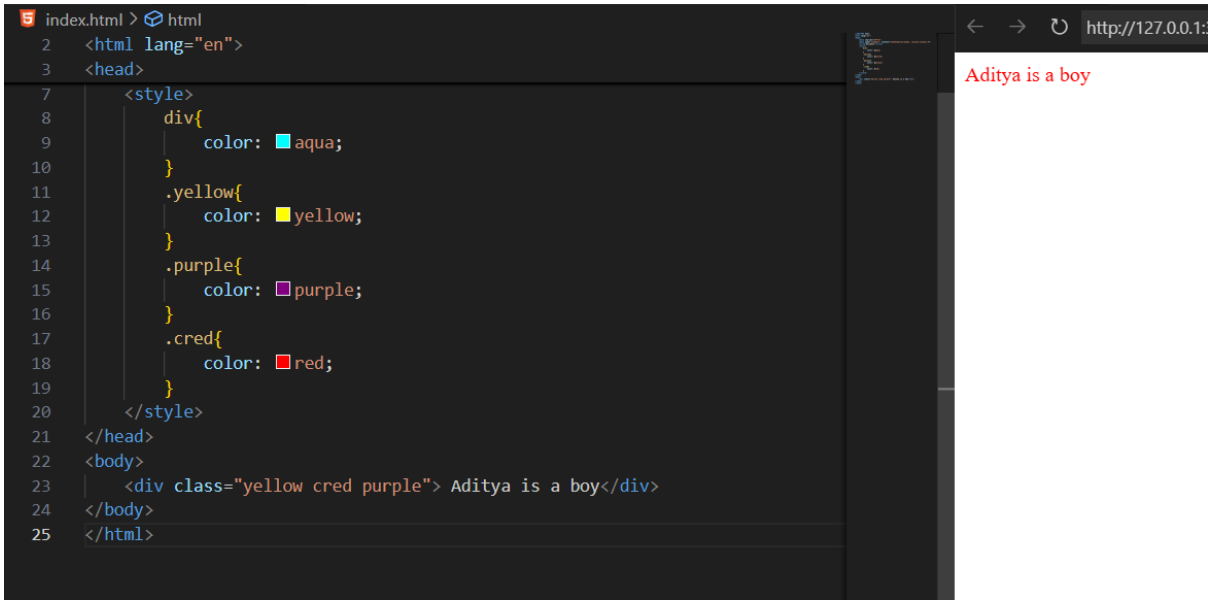
CSS cascade:

What is CSS cascade?

The CSS Cascade is the process the browser uses to decide which CSS style rule is applied to an element when there are multiple conflicting rules.

Think of it like layers of priority — the browser looks at *who said it last and how important it was*.

Example: just change the position of the .purple above the .cred:



Clearly, the properties changed completely. Now, output is of red colour. It means what we wrote at later about that property, it get followed only, and all above it get ignored.

But how this is decided?

The 3 Core Principles of the Cascade

1. **Origin / Importance**
    - CSS comes from different sources:
        - User agent (browser) – default styles
        - User styles – user-defined accessibility settings
        - Author styles – your CSS (highest priority)
    - You can override normal rules with:
        - !important → highest priority (use sparingly)
2. **Specificity**
    - Determines which selector wins if multiple rules apply.
    - Specificity is calculated by selector type:

Selector Type	Example	Specificity Value
Inline style		1000
ID selector	#id	100
Class, attribute, pseudo-class	.class, [attr], :hover	10
Element, pseudo-element	p, ::before	1

    - The rule with the **higher specificity** wins.
3. **Source Order (Last Rule Wins)**

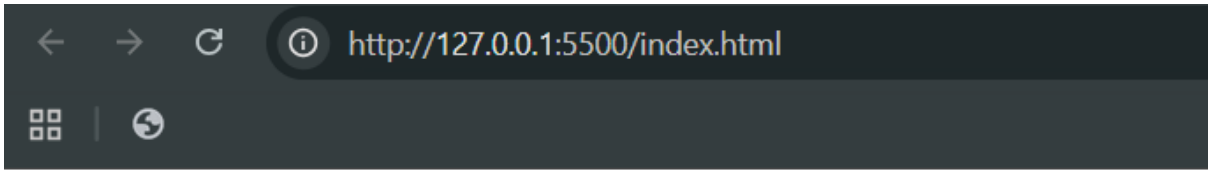
- If two rules have the **same importance & specificity**, the one **written last** in the CSS file wins. (as shown just above)

Example: using the specificity

index.html > ...

```
28  <html lang="en">
29  <head>
33  <style>
34    /* 1 Element selector → specificity = 1 */
35    p {
36      color: blue;
37    }
38
39    /* 2 Class selector → specificity = 10 */
40    .highlight {
41      color: green;
42    }
43
44    /* 3 ID selector → specificity = 100 */
45    #special {
46      color: red;
47    }
48
49    /* 4 Inline style → specificity = 1000 */
50  </style>
51  </head>
52  <body>
53
54    <p>This is a normal paragraph (blue).</p>
55
56    <p class="highlight">This paragraph has a class (green).</p>
57
58    <p id="special" class="highlight">This one has both class and ID (red wins).</p>
59
60    <p id="special2" class="highlight" style="color: orange;">
61      This one has inline style (orange wins).
62    </p>
63
64  </body>
65  </html>
```

Output:



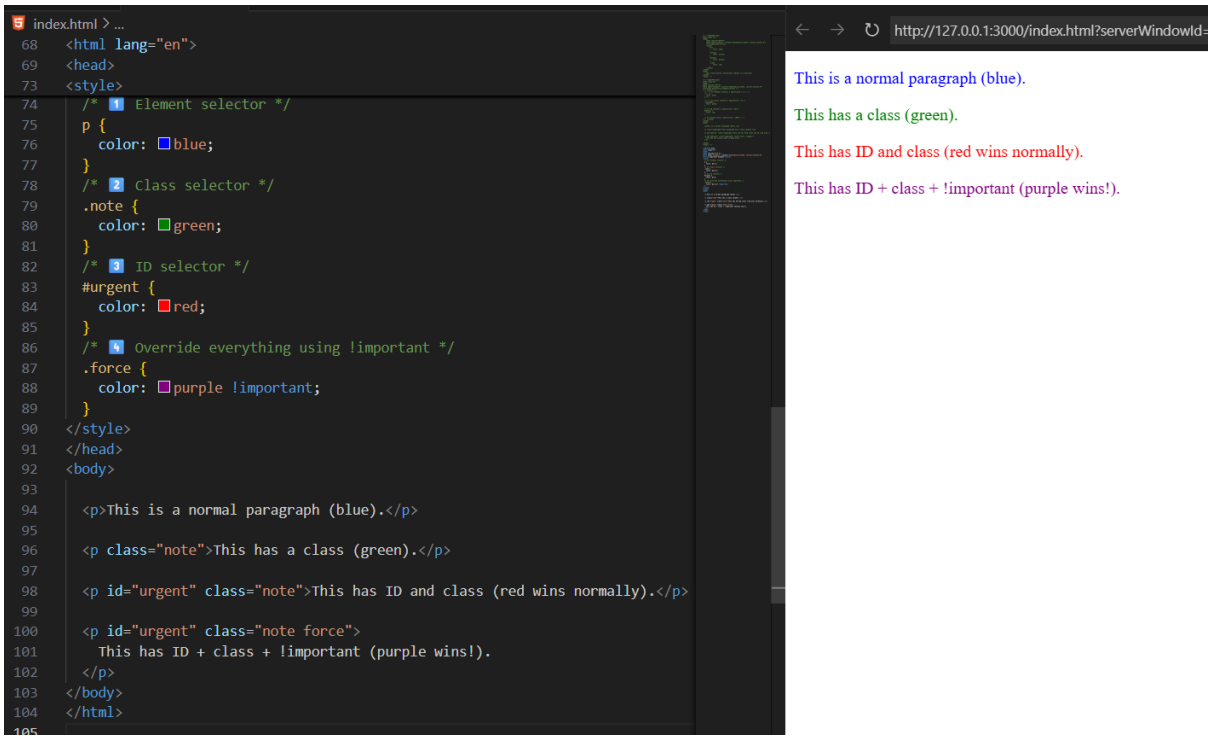
This is a normal paragraph (blue).

This paragraph has a class (green).

This one has both class and ID (red wins).

This one has inline style (orange wins).

Example: !important usage



--The End--