CSS (Selectors)

Cascading Style Sheets

There are 3 ways to use CSS for styling :-

1. External style sheets
2. Internal style sheet
3. Inline CSS for an element

Note: If in index.html page, we have both external & internal style sheets and we're applying different styles on the same html element, more of them takes precedence. The one that is encountered by the browser first is applied, and then if a different style is applied to the same element, then it is overridden (for only the part which is contradicting)

Eg:-

index.html :

```
<head>
    <link rel="stylesheet" href="style.css">
    <style>
        p {
            color: lightgreen;
        }
    </style>
</head>
```

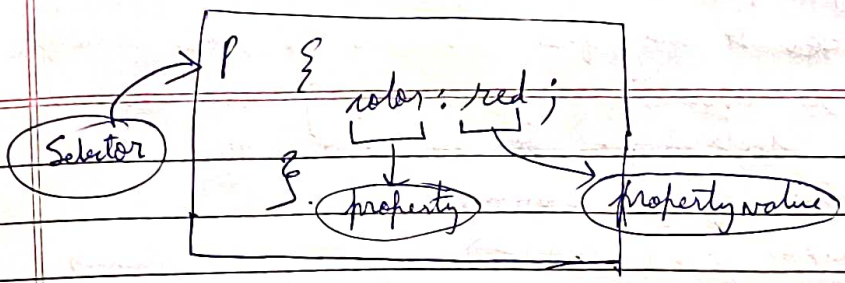
style.css :

```
p {
    color: red;
    font-size: 62px;
}
```

So browser will open index.html that the first style sheet it will encounter in the external one. So p's color will be red and size 62px. Next it will encounter the internal sheet. Here p's color will be override to lightgreen and font size will remain 62px. (Had the internal sheet be placed above external sheet, p's color would be red and 62px.)

Inline CSS has the highest priority always.

Anatomy of CSS ruleset:



↳ Among selectors, class selector will have higher priority than an element selector (like h2, p, span, etc.) if both target the same html tag.

↳ #id has more precedence than .class, if both target the same html tag.

Note: Try to avoid #id for the purpose of styling as much as possible. Use .class instead. Also if you're using one, remember no two elements can have the same id.

↳ Selectors can be grouped together to apply a common style.

Eg:-
```
h2, h3 {
    color: blue;
}
```

↳ Nested selectors: Suppose you have a span tag inside p tag which you want to style.

Eg-
```
p span {
    text-transform: uppercase;
}
```

Note: But this selector will apply the style to all span tags within p. So if you want to style a specific set of span tags, not all, then use a class. That's the better approach.

↳ Universal selector - will apply styling to entire page.

```
* {

}
```

CSS follows a top-down approach ie, whichever selector is written below, overrides the previous elements. But remember, specificity/precedance overrides this top-down approach ie, a class or id selector will override element selector, even if they are not declared at the end.

Note: If you have a huge code base, and at the end you apply say a p selector, but the styling does not get applied. It may feel daunting to skim through all the code to understand which styling has overriden your recent declaration. So to solve this, use 'Inspect' feature of your browser.
Open the webpage in a browser → Inspect → Elements → Styles → Select the html line your applied style to → You'll then see exactly which styling got applied and which one got overriden.

Inheritance in CSS.

So if you declare a font size in body selector, that will be inherited into something like a p selector. But certain things are not inherited (like the border property). If body selector has border property set to 1px x solid black, p selector will not inherit it (ie, individual paragraphs will not have borders).

Form elements dont inherit the font size from a body or other parent selector it is within. To make it inherit the font styling use:

```
button, input, textarea, select {
    font: inherit;
}
```

Specificity

Inline CSS > #id selector > .class selector (for the same html tag) > element selector.

If there are two same elements, the element selector declared at the bottom takes higher precedence (provided there is no inheritance).

However you can ignore this specificity by using "!important" (though not recommended to use).

Eg:- If p has a "class" attribute called "gray",

```
.gray {
    color: gray;
}

p {
    color: pink !important;
}
```

According to precedence, p's color should be gray. But since "!important" is used, color will be pink.

Specificity in inheritance.

If there are elements in CSS where one is parent, other one is child (eg:- body → parent ; p → child), then inheritance comes into play.

If body selector is defined with color: red, and p selector is defined with color: blue, even though both body & p have same specificity, the text inside p will be in blue.

Reason: Any CSS declaration that matches element directly will get priority over the property that is inherited from the element's parent. Here p selector directly matches p element in html, hence that takes precedence over the inherited color property value from parent element 'body'.

Reason: Child element selector > Parent element selector (specificity order)

if, both have different value for same property.

Example 1:

html
```
<h2> This should be black </h2>
<div class = "all_red_text">
    <h2> This should be in red </h2>.
</div>.
```

CSS.
```
h2 { color: black ; } .
.all_red_text { color: red }
```

Here O/P all text will be in black.

Note: class has higher precedence than element, but only if both the selectors target the same html tag. ie., had "div" been a selector and ".class of div" is another selector then .class > div. But in this case, inheritance comes into play. And <h2> becomes a child selector of div. Hence h2 styling gets precedence.

Example 3:

Note: If different types of selectors (ie, class selector, id selector, element selector) target the same html tag, then rule of Specificity comes into play. ie; Inline CSS > id > class > element.

But if Inheritance comes into play, then always child element selector > Parent element selector (irrespective of the type of selectors).

If there are two element selectors targeting different html tags and no inheritance is there, then both have same specificity, and whichever is declared later overrides the former.

Example 2:

html
```
<p> ..... </p>.
<p id="record" class="gray"> .... <span> ... </span> </p>
<p class="gray"> .... </p>.
```

CSS
```
p { color: purple; }. → (1)
#record { color: aqua; } → (2)
p span { color: gold; } } → (3)
.gray { color: gray; } → (4).
```

↳ All paragraphs will be purple due to (1)

↳ Second para will be aqua cause id higher priority than element selector for the same html tag.

↳ 2nd para span element will be gold cause, span is child element of p in 2nd para, hence inheritance comes into play. (and child element selector > Parent element selector)

↳ 3rd para becomes gray but 2nd para does not cause for html tag p, #record > .gray.

Final O/P: 1st para → purple.
2nd para → aqua, but span text in gold
3rd para → gray.

CSS (Colors)

```
body {
    font-size : 22px;
    font-family : Arial, Helvetica, sans-serif;
    line-height : 1.5;
    background-color : papayawhip;
    color : darkblue;
}
```

↳ background-color : is for background

↳ color : is for text

Can be mentioned in two ways: (1) color : color_name ; or,

(2) color : rgb (255, 0, 0) → min value.
→ max value.
↓
stands for red green blue.

∴ rgb (255, 0, 0) → means red.
red green blue.

rgb (255, 0, 0) → red.
rgb (0, 0, 0) → black
rgb (0, 255, 0) → green
rgb (0, 0, 255) → blue

↳ **rgba()** → defines color using the Red - green - blue - alpha model.

RGBA color values are an extension of RGB with an alpha channel - which specifies the opacity of the color.

Range (in Chrome): 0 to 1.
↓ transparent → opaque

↳ **HEX Colors** → A hexadecimal color is specified with #RRGGBB, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the color components. It takes values from 0 to 255 and 1 to F.
highest value. highest value.

#ff0000 → red.
#000000 → black
#ffffff → white

Gray is demonstrated by using equal values for all 3 colors.

↳ Short hand of HEX colors → 3 digit HEX value
Eg:- #ffffff can also be written as #fff.

→ **HSL Colors** → A color can also be specified using hue, saturation, and lightness (HSL).

- hue: 0 to 360
  (0 → red
  120 → green
  240 → blue).
- saturation: (0% → shade of gray
  100% → full color)
- lightness: (0% → black.
  50% → neither ~~black~~ light nor dark
  100% → full white)

Note: For getting HEX codes of different exciting colors,