INTRODUCTION TO CSS

1. What is CSS? (fundamentals & purpose)

CSS stands for Cascading style Sheets

why CSS

- HTML defines the structure & content of a webpage (headings, pargraphs, lists etc.).

- CSS defines the visual presentation - how those elements look: colors, size, fonts, layout, spacing

- without CSS, pages would look plain & unstyled (like old 90s websites)

Key Ideas

CSS helps separate content (HTML) from presentation (CSS) - this makes code easier to maintain & websites more scalable

2) Inline CSS

Inline CSS is when you apply style directly inside an HTML tag using style attribute

```
<h1 style ="color: blue; font-size : 36 px ;">Hello world</h
```

* Prop

- Quick changes to a single element
- Useful for testing or small tweaks

* Cons

- Hard to reuse style
- Becomes messy in large projects
- Not recommended for real projects

3 Internal Css

Internal Css is written inside a <style> block in the HTML <head>

```
<head>
   <style>
      h1{
         colors : green;
         text-align : center;
      }
   </style>
</head>
```

when to use

- Small pages
- when external stylesheet isn't needed
- useful for simple prototypes

⚠ Downside

still not as reusable as an external file - better for one - Page projects

4) External Css

Css is written in a separate .Css file & linked to the HTML

```
<link rel= "Stylesheet" href= "style.Css" >
```

Advantages

- One stylesheet can style an entire website.
- Keeps CSS & HTML cleanly separated.
- Best practice for real projects.

5 How to Debug CSS

Important debugging tips

- Use the browser's Dev tools (right click + inspect)
- Check if
  - CSS file is linked correctly.
  - CSS selectors match the HTML elements.
  - There are typos in property names or values
- use comments to temporariely disable rules without deleting them

```
/* color: red; */
```

Understanding the CasCade

- Rules can be overridden by:
  - Later rules in the stylesheet
  - Inline styles
  - More specific selectors.

6 The Anatomy of CSS Syntax

CSS rules follow a set pattern

```
selector {
    property : value;
}
```

→ Selector:
Selects which HTML element(s) to style

→ Property
Aspect of styling color, font-style or margin

→ Value
what the property is set to "red", 24px ; 10px

7 Css Selectors (Targetting Elements)

Selectors determine which part of the HTML the Styles apply to

- Element Selector:
  Targets HTML tags directly.

```
p {
  color: purple;
}
```

- Class selector:
  Targets elements with a class (prefixed with •);

```
.highlighter {
        background : yellow;
}
```

- Combining selectors
  You can target multiple selectors at once.

```
h1, h2, p {
      font-family : Arial;
}
```

Q Classes Vs IDs

→ Class (•)

- Can be used On many elements

```
<p class ="important">...</p>

.important { font-weight : bold; }
```

→ Id (#)

Must be unique per page

```
<div id="main - titde" >welcome </div>

#main - title { font - size: 48 px;}
```

* when to use what

- Class → reusable styles for multiple elements
- ID → unique styling when a single elements must be targeted.

⇒ Big pictures - Css Rules & Style priorities.

Specificity Hierarchy

More specific selectors override less specific ones:

1. Inline style (most specific)
2. ID selectors
3. Class, attributes selectors.
4. Elements selectors (least specific)

So:

```
#title { color: red;}
.title { color: blue;}
```

→ title will be red due to higher specificity

* Summary of key Concepts

Concepts	Purpose
Inline Css	Quick one-off style
internal Css	Style within a page
External css	Best practices for reusable & scalable styles.
selectors	Target elements to apply styles
class	Reusable style rule
ID	Unique style rule
Debugging	fix & verify CSS behavior

Next steps

1. Build a small HTML Pages
2. Styles heading, paragraphs, & lists with:
   - inline styles
   - Internal Styles
   - External Stylesheet
3. Practice applying classes Vs. IDs
4. use Dev tools to experiment with selectors & properties