AIMERZ.ai
Aim.Act.Achieve

NOTES

Introduction to CSS

Introduction To CSS

What is CSS?

CSS (Cascading Style Sheets) is a language used to style and format web pages. It controls how HTML elements (like text, images, and buttons) look on a website.

Why use CSS?

CSS is used to make websites look visually appealing and organized. It separates the content (HTML) from the design, making it easier to maintain and update the look of a website without changing the core content.

What Can CSS Do?

- **Change appearance**: CSS can control colors, fonts, and the layout of a page.
- **Create layouts**: It helps design the structure of a webpage, like where to place headers, footers, sidebars, etc.
- **Add effects**: CSS can create animations, transitions, and hover effects to make a website interactive.
- **Responsive design**: CSS can make websites look good on different devices, like phones, tablets, and desktops.

CSS version

Cascading Style Sheets (CSS) is an older technology used for designing websites. Over time, it has gone through different versions, each bringing new features and improvements for web designers and developers.

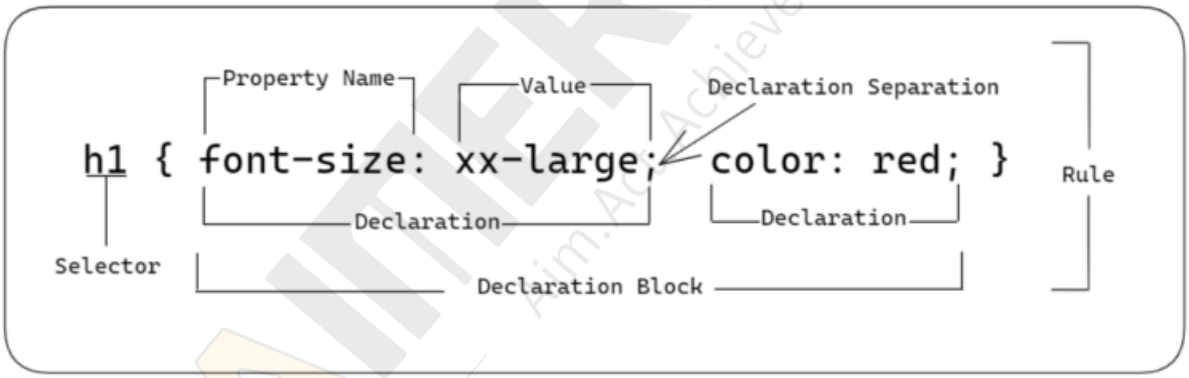
- **CSS1 (1996):** The first version of CSS introduced basic styling features like text formatting, lists, margins, borders, colors, and backgrounds. It worked in most browsers, but older browsers had some issues with lesser-used features like spacing and display settings.
- **CSS2:** This version focused on layout positioning and media types, such as styles for printing. However, certain features like

aural style sheets (for sound) were never widely adopted and were eventually removed.

- **CSS2.1 (2004):** This was an update to CSS2 that fixed issues and clarified rules but didn't bring any big new changes.
- **CSS3:** Unlike earlier versions, CSS3 is a collection of smaller modules released independently. It introduced powerful new features like advanced selectors (for targeting elements), gradients (for color transitions), shadows, animations, transitions, and much more. Modules include Selectors, Box Model, Backgrounds, Borders, Text Effects, and Transforms.

Anatomy Structure of the CSS

The anatomy of Cascading Style Sheets (CSS) refers to the different components or parts that make up CSS and how they work together to style web pages.



Here are the anatomy structure key components of the css:

Selector: CSS selectors define the pattern to select elements to which a set of CSS rules are then applied. CSS selectors can be grouped into the following categories based on the type of elements they can select. The categories can be Basic selectors, Group selectors, Combinators, Pseudo-classes, and pseudo-elements.

Properties: A property is a CSS attribute that defines the aspect of an HTML element to be styled. It could be the font size, color, margin, padding, or background

Value: value is the setting for a css property. It can be specific color, size, or other values that determine the appearance of the element.

Declaration: A declaration is a combination of a property and its corresponding value. It defines a specific style for an HTML element.

Rule - A css rule consists of a selector and its associated declaration block. It defines the styles to be applied to the selected elements.

How to Add CSS to HTML

There are three main ways to apply CSS to HTML documents:

1. **Inline CSS:** In Inline CSS, you directly add CSS styles to individual HTML elements using the style attribute.

```
JavaScript
<!DOCTYPE html>
<html>
<head>
    <title>Inline CSS Example</title>
</head>
<body>
    <h1 style="color: red;">This is a red heading</h1>
    <p style="font-size: 20px;">This text is 20px in size.</p>
</body>
</html>
```

In this example, the color of the **<h1>** is red, and the size of the <p> text is 20px, all specified directly within the HTML tags.

Pros of Inline CSS:

- **Simple and quick:** Easy to apply styles to a specific element.
- **Good for small tasks:** Useful for one-time changes.

Cons of Inline CSS:

- **Not reusable:** You need to repeat the styles for each element, making the code lengthy and harder to manage.
- **Difficult to maintain:** As the website grows, updating styles becomes inefficient since you would have to change it in every HTML tag.

- **Mixes content with styling:** This goes against the practice of separating content (HTML) and design (CSS).

2. **Internal CSS:** In Internal CSS, the styles are written inside a <style> tag within the <head> section of the HTML document. It allows you to define styles for multiple elements in one place.

```
JavaScript
<!DOCTYPE html>
<html>
<head>
    <title>Internal CSS Example</title>
    <style>
        h1 {
            color: blue;
        }
        p {
            font-size: 18px;
        }
    </style>
</head>
<body>
    <h1>This is a blue heading</h1>
    <p>This text is 18px in size.</p>
</body>
</html>
```

In this example, all <h1> elements will be blue, and all <p> elements will have a font size of 18px.

Pros of Internal CSS:

- **Centralized styling**: All styles are in one place within the HTML file, making it easier to manage compared to inline CSS.
- **Great for small websites**: Works well for single-page websites or small projects.
- **No need for external files**: It doesn't require linking to another file.

Cons of Internal CSS:

- **Not reusable across pages**: If you have multiple pages, you need to copy the same CSS code to every HTML file, leading to duplication.

- **Not ideal for large projects**: As the project grows, it becomes harder to maintain consistency and update styles.

3. **External CSS: External CSS** is the most recommended method. In this approach, the CSS is written in a separate file (with a `.css` extension) and linked to the HTML document using a `<link>` tag in the `<head>` section.

Ex:

index.html

```
JavaScript
<!DOCTYPE html>
<html>
<head>
    <title>External CSS Example</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>This is a green heading</h1>
    <p>This text is 16px in size.</p>
</body>
</html>
```

styles.css

```
JavaScript
h1 {
    color: green;
}
p {
    font-size: 16px;
}
```

Here, all styles are written in the external styles.css file, and the HTML file only contains the content and links to the CSS file.

Pros of External CSS:

- **Separation of content and design**: Keeps HTML clean and focuses purely on content, while CSS handles the design in a separate file.
- **Reusable**: The same CSS file can be used across multiple HTML pages, ensuring consistency.
- **Easier to maintain**: If you need to update the design, you only have to change the CSS file, and the updates will apply to all linked pages.
- **Optimized performance**: Since the browser caches the CSS file, it speeds up loading for large websites.

Cons of External CSS:

- **Requires multiple files**: You need to manage both the HTML file and the CSS file, which can be slightly more complex.
- **Won't work offline**: If the CSS file is not loaded (due to a broken link or network issue), the webpage may appear unstyled.

Cascading Order in CSS (Inline, Internal, and External)

The term **"cascading"** in CSS refers to how styles are applied in a specific order when there are multiple styles for the same element. The browser follows a set of rules to determine which style to apply, known as the cascading order. When multiple CSS rules target the same element, the browser resolves conflicts based on specificity, importance, and source order.

Here's the cascading order from the highest to the lowest specificity:

1. **Inline CSS (Highest Specificity)**
2. **Internal CSS**
3. **External CSS (Lowest Specificity)**

1. Inline CSS (Highest Specificity)

- Inline CSS is applied directly to individual HTML elements using the `style` attribute.
- **Specificity**: It has the **highest** priority, meaning it will override any styles defined in internal or external CSS for the same element.

Ex:

```
JavaScript
<p style="color: red;">This text will be red due to inline CSS.</p>
```

In this case, the text will always be red, even if internal or external styles define a different color for the <p> element.

2. Internal CSS

- Internal CSS (also called **embedded CSS**) is defined inside the `<style>` tag in the `<head>` section of the HTML document.
- **Specificity**: It has a **lower priority** than inline CSS but a **higher priority** than external CSS.
- **Where it is used**: Internal CSS applies styles to all matching elements within the same HTML document.

```
JavaScript
<head>
  <style>
    p {
      color: blue;
    }
  </style>
</head>
<body>
  <p>This text will be blue due to internal CSS unless overridden by inline CSS.</p>
</body>
```


External CSS (Lowest Specificity)

- External CSS is written in a separate `.css` file (like `styles.css`) and linked to the HTML document using a `<link>` tag.
- **Specificity**: It has the **lowest priority** compared to inline and internal CSS. If there are no inline or internal styles, external CSS will be applied.

- **Where it is used**: External CSS allows you to apply styles to multiple HTML documents, making it the **preferred** method for larger projects.

Note-
Out of the above three approaches the External is always preferred, some of the reason are given below -

- Modularity - Using external style confines conflict to specific stylesheet files, enabling focused conflict identification and resolution for each section.
- Isolated conflicts - Conflicts are isolated to designated external stylesheets, enabling localized conflict resolution without impacting other styles.
- Specificity Management precision - Utilizing separate CSS files enables effective specificity management, preventing unintended style overrides and ensuring intended styles.

Comments in CSS

In CSS, comments are used to explain the code, making it easier for anyone (including yourself) to understand when revisiting the code later. Comments are ignored by browsers and don't affect the styling of your webpage.

Purpose of Comments in CSS:

1. **Clarification:** To explain what a certain part of the code does.
2. **Organization:** To divide code into sections and make it more readable.
3. **Temporarily Disable Code:** To comment out parts of CSS code for testing or debugging.

Syntax:
CSS comments can be either single-line or multi-line.
Single-line Comment:

```javascript
/* This is a Single line comment */
```

Multi-line comment:

```javascript
/*
This is a multi-line comment.
It can span multiple lines.
*/
```

THANK YOU

AIMERZ.ai
Aim.Act.Achieve