Day 1: CSS Fundamentals

---

1. Introduction to CSS

What is CSS?

- CSS stands for **Cascading Style Sheets**.
- It is used to **style and layout** web pages, including:
    - Colors.
    - Fonts.
    - Spacing.
    - Layouts.
    - Animations.
- It separates **content (HTML)** from **presentation (CSS)**, making it easier to maintain and update.

How CSS Works with HTML

- HTML defines the **structure** of the web page.
- CSS defines the **style** of the elements in the HTML.
- Example:
- This is a paragraph.
- <style>
- p {
- color: blue;
- font-size: 16px;
- }
- </style>

The tag in HTML defines the paragraph, and CSS styles it to have blue text and a font size of 16px.

Different Ways to Write CSS

1. **Inline CSS**:
    - CSS written directly inside the HTML element using the style attribute.
    - Example:
    - This is an inline styled paragraph.
    - Pros: Quick for small changes.
    - Cons: Hard to maintain for large projects.
2. **Internal CSS**:
    - CSS written inside a <style> tag in the <head> section of the HTML document.
    - Example:
    - <head>
    - <style>
    - p {
    - color: green;
    - }
    - </style>
    - </head>
    - Pros: Useful for single-page styles.
    - Cons: Not reusable across multiple pages.
3. **External CSS**:
    - CSS written in a separate .css file and linked to the HTML file using a <link> tag.
    - Example: HTML:
    - <link rel="stylesheet" href="styles.css">

CSS (styles.css):

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

- Pros: Centralized, reusable, and best practice.
- Cons: Requires proper file management.

---

2. Basic Selectors

Element Selectors

- Target HTML elements directly by their tag name.
- Example:
- p {
- color: blue;
- }

Class Selectors

- Target elements with a specific class attribute.
- Syntax: .<className>
- Example: HTML:
- This is a class selector example.

CSS:

```
.box {
  background-color: yellow;
}
```

ID Selectors

- Target a single element with a specific id attribute.
- Syntax: #idName
- Example: HTML:
- This is an ID selector example.

CSS:

```
#header {
  background-color: lightblue;
```

```
}
```

---

3. Basic CSS Properties

Text Styling

1. **color**: Sets the color of text.

```
p {
  color: red;
}
```

5. **font-family**: Specifies the font for the text.

```
p {
  font-family: Arial, sans-serif;
}
```

9. **font-size**: Sets the size of the text.

```
p {
  font-size: 20px;
}
```

13. **font-weight**: Defines the thickness of the text.

```
p {
  font-weight: bold;
}
```

17. **text-align**: Aligns text horizontally.

```
p {
  text-align: center;
}
```

21. **text-decoration**: Adds or removes text decorations (e.g., underline).

```
p {
  text-decoration: underline;
```

```
}
```

Backgrounds

1. **background-color**: Sets the background color of an element.

```
div {
  background-color: lightgrey;
}
```

5. **background-image**: Adds an image as the background.

```
div {
  background-image: url('background.jpg');
}
```

9. **background-position**: Specifies the position of a background image.

```
div {
  background-position: center;
}
```

Dimensions

1. **width and height**: Define the size of an element.

```
div {
  width: 200px;
  height: 100px;
}
```

6. **margin**: Creates space around an element.

```
div {
  margin: 10px;
}
```

10. **padding**: Creates space between the content and the border.

```
div {
  padding: 10px;
```

```
}
```

14. **border**: Defines the border of an element.

```
div {
  border: 2px solid black;
}
```

---

4. Display and Positioning

display Property

- Specifies the type of rendering box for an element.

1. block: Starts on a new line and takes up the full width.

```
div {
  display: block;
}
```

5. inline: Does not start on a new line and only takes as much width as necessary.

```
span {
  display: inline;
}
```

9. inline-block: Like inline but allows setting width and height.

```
div {
  display: inline-block;
}
```

Basic Positioning

1. static (default): Element is positioned according to the normal flow of the document.

```
div {
  position: static;
}
```

5. relative: Position relative to its normal position.

```
6. div {
7.   position: relative;
8.   top: 10px;
9.   left: 20px;
10.}
```

---

5. Working with Colors

Color Formats

1. **Color Names**: Use predefined color names.

```
2. p {
3.   color: red;
4. }
```

5. **Hexadecimal Codes**: Specify colors using #RRGGBB format.

```
6. p {
7.   color: #ff5733;
8. }
```

9. **RGB Values**: Use rgb(red, green, blue) format.

```
10.p {
11.  color: rgb(255, 87, 51);
12.}
```

13. **RGBA Values**: Add transparency with rgba(red, green, blue, alpha) where alpha is between 0 (transparent) and 1 (opaque).

```
14.p {
15.  color: rgba(255, 87, 51, 0.5);
16.}
```

Summary

- **CSS** enhances the look and feel of your webpage.
- **Selectors** allow targeting specific elements.
- **Properties** like color, font-size, and background define the appearance of elements.
- **Positioning and display** control layout and element behavior.
- **Colors** can be specified using names, hex codes, or RGB values.

provides a strong starting point for creating visually appealing webpages.