Fundamentals of CSS I

Introduction to CSS

History of CSS:

1. **Early Web Design:**
   - In the early days of the World Wide Web, web pages were primarily created using HTML (HyperText Markup Language).
   - HTML was responsible for defining the structure and content of web pages, but it could not control their visual presentation.
2. **Challenges with Styling:**
   - Web designers faced challenges in styling web pages uniformly across different browsers and devices.
   - HTML tables were often used for layout, leading to complex and cumbersome code that was difficult to maintain and customize.
3. **Emergence of CSS:**
   - In the late 1990s, the need for a separate language to control the presentation of web pages became apparent.
   - CSS (Cascading Style Sheets) emerged as a solution to this problem, providing a way to separate the structure (HTML) from the presentation (CSS) of web pages.
4. **First CSS Specification:**
   - CSS was first proposed by Håkon Wium Lie and Bert Bos in 1994.
   - The first CSS specification, CSS1, was published by the W3C (World Wide Web Consortium) in 1996, establishing a standard for styling web documents.

Why CSS Came into the Tech World:

1. **Separation of Concerns:**
   - CSS was introduced to address the need for separation of concerns in web development.

- By separating the structure (HTML) from the presentation (CSS), developers could create cleaner, more maintainable code.

2. **Flexibility and Consistency:**
    - CSS provided developers with greater flexibility and control over the visual presentation of web pages.
    - It allowed for consistent styling across multiple pages and facilitated easier updates and modifications.
3. **Accessibility and Usability:**
    - CSS enabled developers to create web pages that were more accessible and user-friendly.
    - It allowed for better organization of content, improved readability, and enhanced user experience.
4. **Adaptability to Different Devices:**
    - With the rise of mobile devices and varying screen sizes, CSS became essential for creating responsive and adaptive web designs.
    - Media queries and other CSS features allowed developers to tailor the layout and styling of web pages based on the device's characteristics.

Types of CSS

1. Inline CSS:

Definition: Inline CSS involves applying styles directly to individual HTML elements using the style attribute.
Syntax:
Example:

```
<p style="color: blue; font-size: 16px;">This is a blue paragraph with a font size of 16px.</p>
```

Pros:
- Quick application of styles to specific elements.
- Useful for small-scale styling adjustments.

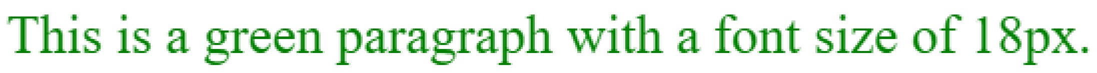
Cons:
- Lack of reusability: Styles are not shared across multiple elements or pages.
- Mixing HTML and CSS can make code less maintainable.

2. Internal CSS:

Definition: Internal CSS involves defining styles within the HTML document using the <style> tag in the document's <head> section.
Syntax:
Example:

```
<head>
 <style>
  p {
   color: green;
   font-size: 18px;
  }
 </style>
</head>
<body>
 <p>This is a green paragraph with a font size of 18px.</p>
</body>
```

Output:



```
This is a green paragraph with a font size of 18px.
```

Pros:

- Better organization: Styles are contained within the HTML file.
- Allows for styling multiple elements.

Cons:

- Styles are still not reusable across different HTML files.
- Separation of concerns is improved but not as clean as external CSS.

3. External CSS:

Definition: External CSS involves placing the styles in a separate CSS file and linking it to the HTML document.

Syntax (HTML):
Linking to the external CSS file in the <head> section:

```
<head>
```

```
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

Syntax (CSS - styles.css):
Example:

```
/* styles.css */
p {
  color: red;
  font-size: 20px;
}
```

Pros:

- Maximum reusability: Styles can be shared across multiple HTML files.
- Clean separation of concerns: HTML focuses on structure, and CSS focuses on presentation.

Cons:

- It requires an additional HTTP request to fetch the external CSS file.
- Slightly, more complex setup compared to inline and internal CSS.

What is the cascading order of the three types of CSS?

The cascading order in CSS refers to the priority and hierarchy applied when multiple styles conflict. For the three types of CSS (Inline, Internal, and external), the cascading order can be understood through the following principles:

Cascading Order:

1. **Inline CSS:**

   Specificity:

   - Highest specificity.
   - Styles are applied directly to individual elements using the style attribute.

   Example:

```html
<p style="color: blue; font-size: 16px;">This is an inline-styled paragraph.</p>
```

2. **Internal CSS:**

   Specificity:

   - Intermediate specificity.
   - Styles are defined within the <style> tag in the HTML document's <head> section.

   Example:

```html
<head>
  <style>
    p {
      color: green;
      font-size: 18px;
    }
  </style>
</head>
<body>
  <p>This is an internally-styled paragraph.</p>
</body>
```

3. **External CSS:**

   Specificity:

   - Lowest specificity.
   - Styles are defined in an external CSS file and linked to the HTML document.

   Example (HTML):

```html
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

   Example (CSS - styles.css):

```css
/* styles.css */
```

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

Specificity Rules:

1. **Inline > Internal > External:**
   If conflicting styles are applied, the inline style takes precedence over internal and external styles.
2. **Internal > External:**
   If both internal and external styles conflict, the internal style takes precedence.
3. **Specificity within Inline or Internal:**
   If multiple styles are applied inline or internally to the same element, the more specific style takes precedence.
4. **Importance of Order:**
   Styles defined later in the document or in an external stylesheet override conflicting styles defined earlier.

Selectors:

1. Type Selectors:

Definition:
Type selectors target HTML elements based on their tag names.
They apply styles to all instances of a specific HTML element.
Example:

```
p {
  color: blue;
  font-size: 16px;
}
```

Usage:

- Commonly used to define styles for standard HTML elements like paragraphs, headings, lists, etc.
- Affects all elements of the specified type on the page.

2. Class Selectors:

Definition:
Class selectors target HTML elements based on their class attributes.
They allow for the styling of multiple elements with the same class.
Example:

```
.highlight {
  background-color: yellow;
  font-weight: bold; }
```

Usage:

- Useful for applying styles to a group of elements with a shared characteristic.
- Can be applied to multiple elements of different types.

Syntax (HTML):

```
<p class="highlight">This paragraph has a special highlight class.</p>
<div class="highlight">This div also has the highlight class.</div>
```

3. ID Selectors:

Definition:
ID selectors target HTML elements based on their id attribute.
They provide a way to uniquely style a specific element on a page.
Example:

```
#header {
  background-color: #333;
  color: white;
}
```

Usage:

- Used for styling a single, unique element on a page.
- Each page should have only one element with a specific ID.

Syntax (HTML):

```
<div id="header">This is the header section.</div>
```

Specificity and Priority:

1. Specificity Hierarchy:

ID Selector > Class Selector > Type Selector:

- ID selectors have the highest specificity.
- Class selectors have medium specificity.
- Type selectors have the lowest specificity.

2. **Importance of Order:**
    - If two selectors have the same specificity, the one declared later takes precedence.
    - Order in the stylesheet matters when resolving conflicts.

3. **!important Rule:**

   Adding !important to a style declaration gives it the highest priority.
   Example:

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

   The color of all paragraphs will be blue, overriding other conflicting styles.

4. **Inline Styles:**

   Inline styles have the highest priority.
   Example:

```
<p style="color: green;">This paragraph has an inline style.</p>
```

   The inline style will override conflicting styles from external or internal stylesheets.

Text Styling Properties:

1. font-family:

Definition**:** Specifies the typeface or font family for the text content.
Example:

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

- It's crucial for maintaining consistency across different platforms and devices.
- Multiple fonts can be listed as fallbacks in case the preferred font is not available.

2. font-size:

Definition: Determines the size of the text characters.
Example:

```
h1 {
 font-size: 24px;
}
```

- Size can be specified in various units such as pixels (px), em units, or rem units.
- Establishing a harmonious font size hierarchy is essential for readability and visual appeal.

3. font-weight:

Definition: Controls the thickness or boldness of the text characters.

Example:

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

- Values include normal, bold, bolder, lighter, or numeric values (e.g., 400, 700).
- Boldness can be adjusted to emphasize or de-emphasize text content.

4. font-style:

Definition: Sets the style of the font (normal, italic, or oblique).
Example:

```
em {
font-style: italic;
}
```

- Italics or oblique styles can add emphasis or convey a different tone.
- Some fonts may not have a distinct italic style, and oblique is applied as a slanted version of the normal font.

5. line-height:

Definition: Specifies the height of a line of text, affecting the spacing between lines.
Example:

```
p {
line-height: 1.5;
}
```

- A proper line height improves readability and enhances the overall appearance of text.
- Can be set as a unitless number, percentage, or with units like px or em.

6. letter-spacing:

Definition: Adjusts the space between characters in the text.
Example:

```
h2 {
 letter-spacing: 2px;
}
```

- Allows fine-tuning of character spacing for aesthetic or design purposes.
- Negative values can be used for tighter character spacing.

7. text-align:

Definition: Determines the horizontal alignment of text within its container.
Example:

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

- Values include left, right, center, and justify.
- Affects the overall layout and presentation of text within a block or inline element.

8. text-decoration:

Definition: Adds visual styling to text, such as underlining, overlining, or strikethrough.
Example:

```
a {
  text-decoration: underline;
}
```

- Helps convey links or emphasize specific text elements.
- Multiple decorations can be combined for a single text element.

9. text-transform:

Definition: Controls the capitalization of text.
Example:

```
.uppercase {
  text-transform: uppercase;
}
```

- Values include uppercase, lowercase, and capitalize.
- Useful for maintaining a consistent text format across different elements.

10. color:

Definition: Sets the color of the text characters.
Example:

```
p {
color: #333;
}
```

- Color can be specified using color names, hexadecimal values, RGB, or HSL values.
- Contributes significantly to the overall design and aesthetic appeal of the webpage.

11. text-shadow:

Definition: Adds a shadow effect to the text characters.

- Creates a visual depth and emphasis, enhancing the readability of text.
- Parameters include horizontal and vertical offsets, blur radius, and shadow color.

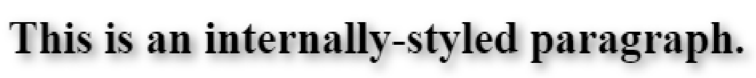
Example:

```
h3 {
  text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
}
```

- **Horizontal Offset (2px):** This value (2px) determines the horizontal distance the shadow extends from the text. In the example, the shadow will be displaced 2 pixels to the right of each character.
- **Vertical Offset (2px):** This value (2px) determines the vertical distance the shadow extends from the text. In the example, the shadow will be displaced 2 pixels below each character.
- **Blur Radius (4px):** This value (4px) controls the amount of blurring applied to the shadow. A larger value results in a more diffused and softer shadow. In the example, the shadow will have a slight blur effect.
- **Shadow Color (rgba(0, 0, 0, 0.5)):** This value specifies the color of the shadow. In the example, the shadow color is defined using RGBA values:
  - ☐ 0, 0, 0 represents the RGB values for black.

- ☐ 0.5 represents the alpha channel, controlling the shadow's transparency. It is set to 0.5, making the shadow semi-transparent.

Output:



This is an internally-styled paragraph.

CSS Units:

1. Pixels (px):

Definition: A pixel is the smallest unit on a digital display. It represents a single dot on the screen.
Example:

```
div {
width: 200px;
height: 150px;
}
```

Usage:
Commonly used for fixed-size elements, providing a precise measurement.

2. Points (pt), Picas (pc):

Definition: Points and picas are units commonly used in typography and print styles.
Example:

```
p {
  font-size: 12pt;
}
```

Usage:
Appropriate for specifying font sizes in print-related styles.

3. Relative Length Units:

- **Em (em):**
  Definition: Relative to the font-size of the parent element.
  Example:

```
p {
  font-size: 1.2em;
  margin-bottom: 1.5em;
}
```

Usage:
Excellent for creating scalable designs, especially for text-related properties.

- **Rem (rem):**
  Definition: Relative to the font-size of the root element (typically the <html> element).
  Example:

```
body {
 font-size: 16px;
}

p {
  font-size: 1.5rem;
}
```

Usage:
Useful for maintaining consistent and scalable designs across the entire document.

4. Viewport Percentage Units:

- **Viewport Width (vw), Viewport Height (vh):**
  Definition: Relative to the viewport's width or height.

  Example:

```css
section {
  width: 80vw;
  height: 50vh;
}
```

Usage:
Effective for creating responsive designs that adapt to the dimensions of the viewport.

- **Viewport Minimum (vmin), Viewport Maximum (vmax):**
  Definition: Relative to the smaller or larger dimension of the viewport (width or height).
  Example:

```css
div {
  width: 30vmin;
  height: 40vmax;
}
```

Usage:
Useful for maintaining proportions while considering the viewport's dimensions.

5. Percentage (%):

Definition: A percentage is relative to the property's parent element.
Example:

```css
.container {
  width: 80%;
  margin: 0 auto;}
```

Usage:

Commonly used for creating fluid and responsive layouts, particularly for widths and margins.

CSS Background Properties :

1. background-color:

Definition: Sets the background color of an element.
Example:

```
body {
  Background-color: blue;
}
```

Usage:
Provides a solid color for the background of an element.

Values:
Accepts color names, hexadecimal values, RGB, or HSL values.

2. background-image:

Definition: Specifies an image for the background.
Example:

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

Usage:
This allows you to set an image as the background, enhancing visual appeal.

Values:
url('path/to/image'): Specifies the path to the image file.

linear-gradient(): Creates a gradient background.

3. background-repeat:

Definition: Controls how a background image is repeated.
Example:

```
section {
  background-repeat: repeat-x;
}
```

Usage:
Adjusts the tiling behavior of background images.

Values:
- **repeat:** Default, repeats both horizontally and vertically.
- **repeat-x:** Repeats only horizontally.
- **repeat-y:** Repeats only vertically.
- **no-repeat:** Does not repeat.

4. background-position:

Definition: Sets the starting position of a background image.
Example:

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

Usage:
Determines where the background image starts, using keywords or percentage values.

Values:
- Coordinates (e.g., center, top, 50%, 25px).
- Keywords like center, top, bottom, left, and right.

5. background-size:

Definition: Specifies the size of a background image.
Example:

```
.banner {
  background-size: cover;
}
```

Usage:

- Values include auto, cover, contain, or specific dimensions.
- Controls how the background image is sized within the container.

Values:

- **auto:** Default, preserves the image's original size.
- **cover:** Scales the image to cover the entire container.
- **contain:** Scales the image to fit within the container.

6. background-attachment:

Definition: Determines if a background image is fixed or scrolls with the page.
Example:

```
body {
  background-attachment: fixed;
}
```

Usage:

- Values include scroll (default) or fixed.
- Determines if the background image moves with the content or stays fixed.

Values:

- **scroll:** Default, background scrolls with the content.
- **fixed:** Background remains fixed while content scrolls.

- **local:** Background scrolls with the element's content.

7. background:

Definition: A shorthand property to set multiple background properties at once.
Example:

```
header {
  background: #4CAF50 url('header-background.jpg') center/cover no-repeat;
}
```

Usage:
Consolidates multiple background properties into a single declaration for concise code.

CSS Colors:

1. Named Colors:

Definition: Named colors are predefined color keywords that represent specific colors.
Example:

```
h1 {
  color: red;
}
```

- Common named colors include red, blue, green, yellow, black, white, and others.
- Provides simplicity but offers a limited set of colors.

2. Hexadecimal Color Codes:

Definition: Hexadecimal color codes use a six-digit combination of numbers (0-9) and letters (A-F) to represent colors.
Example:

```
p {
  color: #ff0000; /* Red */
}
```

- Each pair of digits represents the intensity of red, green, and blue components in hexadecimal (e.g., #RRGGBB).

3. RGB Values:

Definition: RGB values represent colors using the intensity of red, green, and blue components on a scale of 0 to 255.
Example:

```
div {
  background-color: rgb(255, 0, 0); /* Red */
}
```

- Specified using the rgb() function with three parameters for red, green, and blue intensity.
- Values range from 0 (no intensity) to 255 (full intensity).

4. RGBA Values:

Definition: RGBA values are similar to RGB but include an additional alpha channel for transparency.
Example:

```
h2 {
  color: rgba(0, 128, 0, 0.5); /* Semi-transparent green */
}
```

- The alpha value ranges from 0 (completely transparent) to 1 (completely opaque).

5. HSL Values:

Definition: HSL values represent colors using three parameters: hue, saturation, and lightness.
Example:

```css
span {
  color: hsl(120, 100%, 50%); /* Pure green */
}
```

- Hue (0-360) represents the type of color.
- Saturation (0%-100%) represents the intensity or vividness.
- Lightness (0%-100%) represents the brightness.

6. HSLA Values:

Definition: HSLA values are similar to HSL but include an alpha channel for transparency.
Example:

```css
a {
  color: hsla(240, 100%, 50%, 0.7); /* Semi-transparent blue */
}
```

- The alpha value ranges from 0 (completely transparent) to 1 (completely opaque).

Conclusion:

In conclusion, the evolution of CSS has played a pivotal role in the advancement of web development, addressing the limitations and challenges faced during the early days of the World Wide Web. The introduction of CSS in the late 1990s brought about a significant shift by providing a separate language to control the presentation of web pages, thereby promoting the separation of concerns in web development.

CSS has proven to be a powerful tool, offering developers flexibility, consistency, and improved accessibility in crafting visually appealing and user-friendly websites. The adoption of CSS has enabled responsive and adaptive web designs, accommodating the diverse landscape of devices and screen sizes.

The discussion on the types of CSS, including Inline, Internal, and External, illustrates the various approaches to styling web pages, each with its advantages and limitations. Understanding the cascading order and specificity rules helps developers manage conflicting styles effectively.

Moreover, the exploration of CSS selectors, text styling properties, units, and background properties equips developers with a comprehensive set of tools to tailor the appearance of web elements. The emphasis on color representation, including Hexadecimal, RGB, HSL, and CurrentColor, adds another layer of depth to the design possibilities.

In essence, the fundamentals of CSS outlined here serve as a foundation for web developers to create visually engaging, responsive, and accessible websites, ushering in a new era of dynamic and aesthetically pleasing online experiences. As technology continues to evolve, the principles and techniques of CSS remain integral to the art and science of web development.

References:

1. https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/How_CSS_is_structured
2. https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Type_Class_and_ID_Selectors
3. https://developer.mozilla.org/en-US/docs/Learn/CSS/Styling_text/Fundamentals
4. https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units
5. https://developer.mozilla.org/en-US/docs/Web/CSS/background