Introduction to CSS

Abdulla Ebrahim Subah

University of Bahrain

September 11, 2024

What is CSS?

- **CSS** stands for *Cascading Style Sheets*.
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
- CSS saves a lot of work by controlling the layout of multiple web pages all at once.

Why Use CSS?

- **Separation of concerns**: HTML for structure, CSS for presentation.
- **Consistency**: Style can be applied uniformly across multiple pages.
- **Flexibility**: Easily change the look and feel of a website.
- **Accessibility**: Improve access for users with disabilities via better design.

History of CSS

- **CSS** was first proposed by Håkon Wium Lie in **1994**.
- The first official specification, **CSS1**, was released by the **W3C** in **December 1996**.
- **CSS2** was released in **May 1998** and introduced features like absolute, relative, and fixed positioning.
- **CSS3** was modularized and introduced in **1999**, allowing independent development of different modules (e.g., selectors, flexbox, transitions).
- Modern CSS continues to evolve with new features such as **CSS Grid**, **custom properties (CSS variables)**, and **media queries** for responsive design.
- Latest CSS Standard: Link

CSS Syntax

A CSS rule consists of a **selector**, followed by a **property** and a **value** inside curly braces.

Selector { property: value; }

- **Selector**: Specifies which HTML element(s) to style.
- **Property**: The style attribute you want to change.
- **Value**: The specific attribute value.

Example

```
h1 {
  color: blue;
  text-align: center;
}
```

Types of CSS

- **Inline CSS**: Styles applied directly within HTML elements via the `style` attribute.
- **Internal CSS**: Styles written within a `<style>` tag in the `<head>` section of an HTML document.
- **External CSS**: Styles written in an external `.css` file and linked to the HTML document.

Inline CSS Examples

- Inline CSS is applied directly using the `style` attribute within an HTML tag.

Example 1

```
<h1 style="color: red; text-align: center;">Hello, Inline CSS!</h1>
```

This will make the heading red and center-align it.

Example 2

```
<p style="font-size: 14px; color: blue;">This is a blue paragraph with 14px font size.</p>
```

This will make the paragraph blue with a font size of 14px.

- Inline CSS is useful for quick, one-off customizations but is generally discouraged for large-scale styling.

Internal CSS

Internal CSS is defined within a <style> tag inside the <head> of an HTML document. It is used to style a single web page.

HTML example

```
<head>
  <style>
    h1 { color: green; text-align: center; }
    p { font-size: 14px; color: black; }
  </style>
</head>
<body>
  <h1>Welcome to My Page</h1>
  <p>This paragraph is styled internally.</p>
</body>
```

In this example, the CSS is written directly inside the HTML file within the <style> tag.

External CSS

External CSS is written in a separate `.css` file and linked to the HTML document with a `<link>` tag in the `<head>` section.

HTML example:

```
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <h1>Hello, External CSS!</h1>
  <p>This is a paragraph styled using an external stylesheet.</p>
</body>
```

In the external file (`styles.css`):

```
h1 { color: blue; text-align: center; }
p { font-size: 16px; color: gray; }
```

This method allows you to apply the same styles to multiple HTML pages, making it easier to maintain consistency across a website.

Element Selector

- **Element selector**: Targets elements by their name.

Example

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

HTML Example

```
<p>This is a paragraph.</p>
```

This will make the paragraph red.

Class Selector

- **Class selector**: Targets elements of a specific class. ### Example

```
.myClass { font-size: 14px; }
```

HTML Example

```
<p class="myClass">This text will be 14px.</p>
```

This will make the paragraph text 14px.

ID Selector

- **ID selector**: Targets a specific element by its ID.

Example

```
#myId { margin: 10px; }
```

HTML Example

```
<div id="myId">This div has a 10px margin.</div>
```

This will apply a 10px margin to the

with the ID `myId`.

CSS Combinators

- **Combinators** are used to define relationships between selectors.
- CSS supports four types of combinators:
  - Descendant combinator (space)
  - Child combinator (>)
  - Adjacent sibling combinator (+)
  - General sibling combinator (~)

Descendant Combinator

- The **descendant combinator** is represented by a space between two selectors.
- It selects all elements that are descendants of a specified element.

Example

```
div p {
  color: blue;
}
```

This will select all <p> elements that are inside any element.

Child Combinator

- The **child combinator** is represented by a > symbol.
- It selects only the direct children of a specified element.

Example

```
div > p {
  color: green;
}
```

This will select all <p> elements that are direct children of a element.

Adjacent Sibling Combinator

- The **adjacent sibling combinator** is represented by a + symbol.
- It selects an element that is directly adjacent (next to) another specified element.

Example

```
h1 + p {
  font-size: 18px;
}
```

This will select the first <p> element that immediately follows an <h1> element.

General Sibling Combinator

- The **general sibling combinator** is represented by the ~ symbol.
- It selects all sibling elements that follow a specified element.

Example

```
h1 ~ p {
  color: purple;
}
```

This will select all <p> elements that are siblings of an <h1> element, not just the one immediately following.

CSS Box Model

- Every HTML element is considered as a box.
- The CSS **box model** describes the space an element occupies.
  - **Content**: The actual content (text, image, etc.).
  - **Padding**: Clears an area around the content.
  - **Border**: A border that goes around the padding and content.
  - **Margin**: Clears an area outside the border.

Example of Box Model

```
div {
  width: 200px;
  padding: 10px;
  border: 5px solid black;
  margin: 20px;
}
```

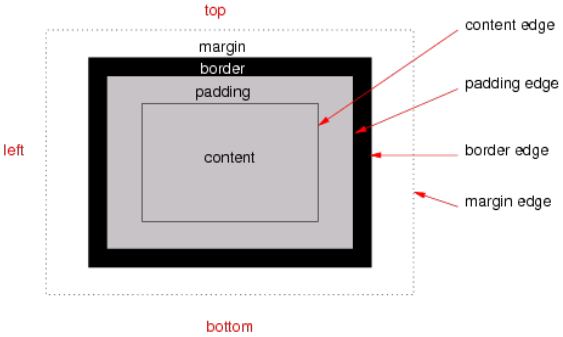

Figure 1: The various areas and edges of a typical CSS box. Source

Common CSS Attributes

A non comprehensive list of CSS attributes with examples. ## Color and Text

- **color**: Sets the text color.
  - Example: `p { color: red; }` – Makes text inside <p> elements red.
- **font-size**: Adjusts the font size.
  - Example: `h1 { font-size: 32px; }` – Makes <h1> text 32px.
- **font-family**: Specifies the font for an element.
  - Example: body `{ font-family: Arial, sans-serif; }` – Sets the body text to Arial or a similar sans-serif font.

Background and Borders

- **background-color**: Sets the background color of an element.
  - Example: `div { background-color: lightgrey; }` – Sets a light grey background for `<div>` elements.
- **border**: Defines the border of an element.
  - Example: `p { border: 1px solid black; }` – Adds a black border around `<p>` elements.
- **border-radius**: Rounds the corners of an element's border.
  - Example: `.box { border-radius: 10px; }` – Makes the corners of elements with class `box` rounded.

Spacing

- **margin**: Adds space outside an element.
  - Example: `h1 { margin: 20px; }` – Adds 20px space outside all sides of `<h1>` elements.
- **padding**: Adds space inside an element, between the content and the border.
  - Example: `div { padding: 10px; }` – Adds 10px of space inside the `<div>` elements.
- **line-height**: Adjusts the vertical space between lines of text.
  - Example: `p { line-height: 1.5; }` – Increases line spacing for paragraphs.

Text Alignment

- **text-align**: Aligns text within an element.
  - Example 1: `p { text-align: left; }` – Aligns the text to the left.
  - Example 2: `h1 { text-align: center; }` – Centers the heading text.
  - Example 3: `div { text-align: right; }` – Aligns the text inside the `<div>` to the right.

Vertical Alignment

- **vertical-align**: Aligns an inline or table-cell element vertically.
  - Example: `img { vertical-align: middle; }` – Vertically aligns an image in the middle of surrounding text.
  - Example: `td { vertical-align: top; }` – Ensures table cell content is aligned to the top.

Flexbox Alignment

- **Flexbox** provides powerful alignment for container elements.
  - **justify-content**: Aligns items horizontally.
  - **align-items**: Aligns items vertically.

Example

```
.container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 200px;
}
```

This will center any content inside the `.container` both horizontally and vertically.

Grid Alignment

- **CSS Grid** allows precise control of layout alignment.
  - **justify-items**: Aligns grid items horizontally within their grid area.
  - **align-items**: Aligns grid items vertically within their grid area.

Example

```
.grid-container {
  display: grid;
  justify-items: center;
  align-items: center;
}
```

This aligns grid items to the center both horizontally and vertically in the grid container.

Media Queries in CSS

- **Media queries** allow you to apply CSS rules depending on the size of the viewport or device.
- This is commonly used to create responsive designs that adapt to different screen sizes.

Example

```
@media (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

In this example, if the screen width is less than or equal to 600px, the background color of the body will change to light blue.

Media Queries Font Sizes Example

```css
/* Small screens (mobile) */
@media (max-width: 600px) {
  body {
    font-size: 14px;
  }
}

/* Medium screens (tablets) */
@media (min-width: 601px) and (max-width: 900px) {
  body {
    font-size: 16px;
  }
}

/* Large screens (desktop) */
@media (min-width: 901px) {
  body {
    font-size: 18px;
  }
}
```

SCSS

- **SCSS** is a preprocessor that extends regular CSS by introducing more advanced features like variables, nesting, and mixins.
- SCSS files end in `.scss` and must be compiled into regular CSS before they can be used in web pages.

Features of SCSS:

- **Variables**: Store reusable values for colors, fonts, etc.
- **Nesting**: Allows you to nest CSS selectors inside one another, reflecting the HTML structure.
- **Mixins**: Reusable chunks of code that can be included in other selectors.

Example of SCSS

```scss
$primary-color: blue;

body {
  font-family: Arial, sans-serif;
  color: $primary-color;

  h1 {
    font-size: 24px;
    text-align: center;
  }

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

Conclusion

- CSS is a powerful tool for designing web pages.
- It provides flexibility and control over the layout and appearance of websites.
- The separation of HTML and CSS allows for cleaner code and easier maintenance.