Thursday - 9/5/2024

CSS

Introduction

- CSS stands for Cascading Style Sheets and is used for styling and laying out web pages.
- CSS works in conjunction with HTML to create visually appealing and user-friendly websites.
- CSS allows you to control the layout, colors, fonts, and other design elements of your website, separate from the content (HTML).

CSS Selectors

- CSS selectors are used to select HTML elements that you want to style.
- There are several types of selectors, including:
    1. Element Selectors: Select elements based on the HTML tag name. Example: p selector will select all <p> elements.
    2. Class Selectors: Select elements based on a class attribute. Example: .my-class selector will select all elements with class my-class.
    3. ID Selectors: Select elements based on an id attribute. Example: #my-id selector will select the element with id my-id.
    4. Attribute Selectors: Select elements based on an attribute and its value. Example: a[href="https://www.example.com"] selector will select all <a> elements with a href attribute equal to https://www.example.com.
    5. Pseudo-class Selectors: Select elements based on a special state. Example: a:hover selector will select an <a> element when the mouse is hovering over it.

CSS Properties

- CSS properties are used to control the visual presentation of HTML elements.
- There are several types of properties, including:
    1. Text Properties: Control the text appearance, such as font size, font color, etc.
    2. Box Properties: Control the box model, such as width, height, padding, margin, etc.
    3. Background Properties: Control the background appearance, such as background color, background image, etc.

4. Border Properties: Control the border appearance, such as border size, border color, etc.
5. Layout Properties: Control the layout and position of elements, such as display, float, position, etc.

HTML & CSS

Three ways to add CSS to HTML

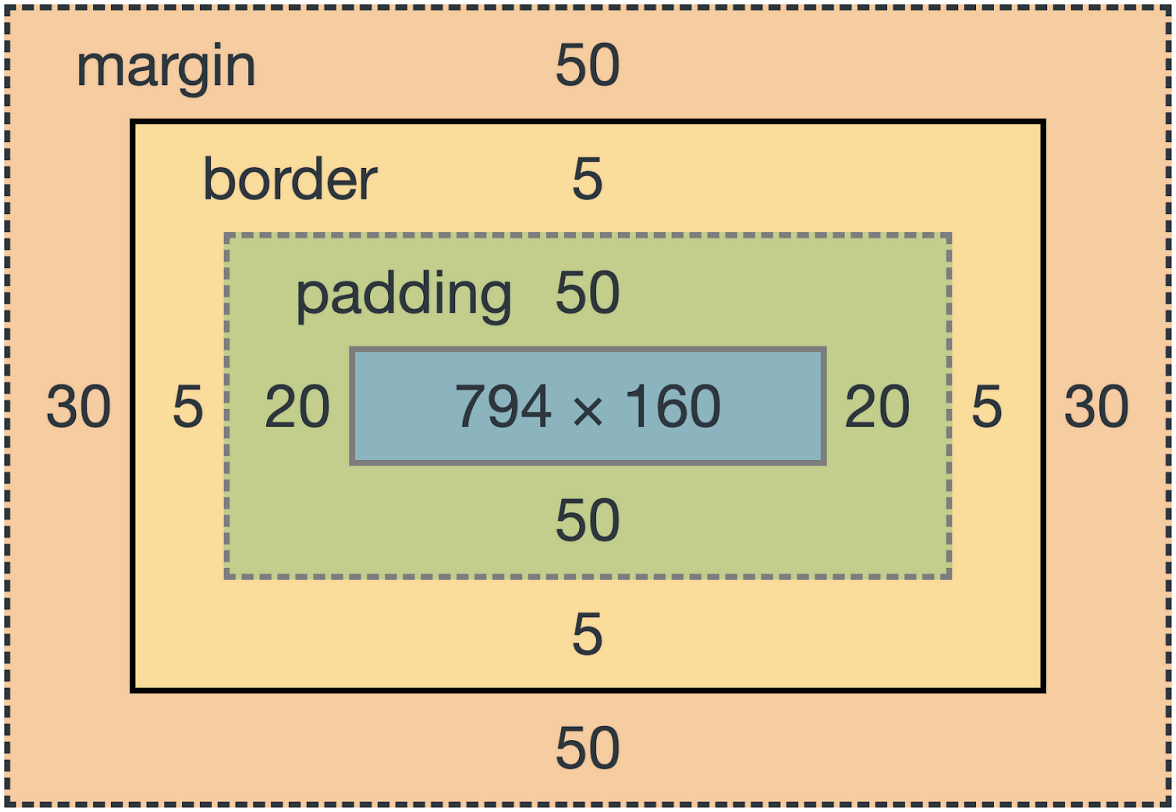
1. Inline CSS - by using the *style* attribute inside HTML elements
2. Internal CSS - by using a <*style*> element in the <*head*> section
3. External CSS - by using a <*link*> element to link to an external CSS file

```
<!DOCTYPE html>
<html>
 <head>
  <title>My HTML & CSS Document</title>
  <style>
    .container {
      width: 60%;
      margin: 0 auto;
    }

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

    p {
      font-size: 20px;
      line-height: 2;
      font-family: Georgia, 'Times New Roman', Times, serif;
    }
```

```
  </style>
 </head>
 <body>
  <div class="container">
   <h1>Welcome to my HTML & CSS Web Page</h1>
   <p>This is a simple paragraph with styles applied.</p>
   <p>I will be discussing about fashion and modelling</p>
  </div>
 </body>
</html>
```

CSS Box Model

The CSS box model is a box that wraps around every HTML element. It consists of content, padding, borders, and margins.
The following diagram illustrates the box model.



Properties of the Box Model

There are several properties in the CSS box model. They are as mentioned below:

Content

The content area consists of content like images, text, or other forms of media content. The height and width properties help to modify the box dimensions.

Padding

The padding area is the space around the content area and within the border-box. It can be applied to all sides of the box or to the specific, selected side(s) - top, right, bottom, and/or left.

Border

The border area surrounds the padding and the content and can be applied to all the sides of the box or selected side(s) - top, right, bottom, and/or left.

Margin

The margin area consists of space between the border and the margin. The margin does not possess its background color and is completely transparent. It shows the background color of the element, like the body element.

CSS Inheritance

- CSS inheritance is a mechanism where properties are passed from parent elements to child elements.
- Child elements inherit the styles of their parent elements by default.
- You can override inherited styles by applying styles directly to the child elements.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Inheritance</title>
    <style>
    body {
      font-size: 20px;
```

```
      color: green;
    }

    .my-element {
      font-size: 30px;
    }
  </style>
</head>
<body>
    <p>This is a paragraph.</p>
    <div class="my-element">
        <p>This is a paragraph inside a div.</p>
    </div>



</body>
</html>
```

In this example, the body has a font size of 20px and a color green. The child element .my-element has a font size of 30px, which overrides the font size inherited from the body. The color green is still inherited by the child element.

CSS Specificity

- CSS specificity refers to the priority of styles applied to an element.
- When multiple styles are applied to an element, the style with the highest specificity wins and is applied to the element.
- Specificity is determined by the type, class, and id selectors used.

Example 1

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Specificity</title>
  <style>
     p {
       color: blue;
     }

     .my-element {
       color: green;
     }

     #my-id {
       color: red;
     }
   </style>
</head>
<body>
  <body>
     <p class="my-element" id="my-id">CSS specificity refers to the priority of styles applied to an element.</p>
   </body>



</body>
</html>
```

In this example, the p selector has the color blue, the .my-element selector has the color green, and the #my-id selector has the color red. The #my-id selector has the highest specificity, so the color of the paragraph will be red.

Example 2

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Specificity Example</title>
    <style>
        /* Style 1 */
        p {
            color: blue;
        }

        /* Style 2 */
        .highlight {
            color: red;
        }

        /* Style 3 */
        #content p {
            color: green;
        }

        /* Style 4 */
        p.special {
            color: orange;
        }
    </style>
</head>
```

```
<body>
  <p>Paragraph 1</p> <!-- Applies Style 1 -->
  <p class="highlight">Paragraph 2</p> <!-- Applies Style 1 and Style 2 -->
  <div id="content">
    <p>Paragraph 3</p> <!-- Applies Style 1 and Style 3 -->
  </div>
  <p class="highlight special">Paragraph 4</p> <!-- Applies Style 1, Style 2, and Style 4 -->
</body>
</html>
```

Exercise

1. Research on how Specifity is calculated in a web browser

CSS Classes and IDs

CSS classes and IDs are used to target specific elements in the HTML document and apply styles to them. The difference between classes and IDs is that a class can be applied to multiple elements in the HTML document, while an ID can only be used once in a document.

CSS Classes

```
<!DOCTYPE html>
<html>
 <head>
  <title>My HTML & CSS Document</title>
  <style>
    .highlight {
      background-color: yellow;
    }
  </style>
 </head>
 <body>
  <p class="highlight">This is a highlighted paragraph.</p>
  <p class="highlight">This is another highlighted paragraph.</p>
```

```
 </body>
</html>
```

Class Ids

```
<!DOCTYPE html>
<html>
 <head>
  <title>My HTML & CSS Document</title>
  <style>
    #header {
      background-color: blue;
      color: white;
      text-align: center;
    }
  </style>
 </head>
 <body>
  <header id="header">This is the header of the page.</header>
 </body>
</html>
```

Responsive Web Design with CSS

FlexBox

CSS Flexbox is a layout model that allows you to design flexible and responsive layouts with ease. It provides a powerful way to distribute space and align-items within a container, regardless of their size or order. Below is an explanation of CSS Flexbox concepts along with some code examples:

Flex Container

To create a flex container, set the display property to flex or inline-flex.

Any direct child elements of a flex container become flex items.

```
.container {
```

```
  display: flex;
  /* or display: inline-flex; */
}
```

Flex Direction

Determines the main axis along which flex items are laid out.

Options: row (default), row-reverse, column, column-reverse.

```
.container {
  flex-direction: row; /* or row-reverse, column, column-reverse */
}
```

Flex Wrap

Controls whether flex items are forced onto a single line or can wrap onto multiple lines.

Options: nowrap (default), wrap, wrap-reverse

```
.container {
  flex-wrap: nowrap; /* or wrap, wrap-reverse */
}
```

Flex Justify Content

Aligns flex items along the main axis of the flex container.

Options: flex-start (default), flex-end, center, space-between, space-around, space-evenly.

```
.container {
  justify-content: flex-start; /* or flex-end, center, space-between, space-around, space-evenly */
}
```

Flex Align Items

Aligns flex items along the cross axis of the flex container.

Options: stretch (default), flex-start, flex-end, center, baseline.

```
.container {
  align-items: stretch; /* or flex-start, flex-end, center, baseline */
}
```

Flex Align Content

Aligns flex lines within the flex container when there is extra space on the cross-axis.

Options: stretch (default), flex-start, flex-end, center, space-between, space-around.

```
.container {
  align-content: stretch; /* or flex-start, flex-end, center, space-between, space-around */
}
```

Example

flexbox.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Flexbox Example</title>
 <link rel="stylesheet" href="flexbox.css">
</head>
<body>
 <header>
   <h1>Flexbox Example</h1>
 </header>
 <nav>
   <ul>
     <li><a href="#">Home</a></li>
     <li><a href="#">About</a></li>
     <li><a href="#">Services</a></li>
     <li><a href="#">Contact</a></li>
   </ul>
 </nav>
```

```
<main>
  <section>
    <h2>Welcome to our website</h2>
    <p>This is a simple example of a webpage layout using Flexbox.</p>
  </section>
  <aside>
    <h3>Latest News</h3>
    <ul>
      <li>News item 1</li>
      <li>News item 2</li>
      <li>News item 3</li>
    </ul>
  </aside>
</main>
<footer>
  <p>&copy; 2024 Flexbox Example</p>
</footer>
</body>
</html>
```

flexbox.css

```
body {
   font-family: Arial, sans-serif;
   margin: 0;
   padding: 0;
 }
  header, nav, main, footer {
   padding: 10px;
 }
  header {
   background-color: #333;
   color: #fff;
   text-align: center;
 }
  nav ul {
   list-style-type: none;
   padding: 0;
   margin: 0;
```

```
  display: flex;
  justify-content: center;
}
 nav li {
  margin: 0 10px;
}
 nav a {
  text-decoration: none;
  color: #333;
}
 main {
  display: flex;
}
 section {
  flex: 2;
  padding-right: 20px;
}
 aside {
  flex: 1;
  background-color: #f4f4f4;
  padding: 10px;
}
 footer {
  background-color: #333;
  color: #fff;
  text-align: center;
  padding: 10px 0;
  left: 0;
  bottom: 0;
  width: 100%;
  position: fixed;
}
```

- The <header>, <nav>, <main>, and <footer> elements are flex containers.
- The navigation links (<nav>) are horizontally centered using justify-content: center;.
- The main content area (<main>) is divided into two sections using Flexbox: the main content area (<section>) and a sidebar (<aside>).
- The <footer> is positioned at the bottom of the page using position: fixed; and bottom: 0;.

CSS Grid

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

grid.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>CSS Grid Example</title>
 <link rel="stylesheet" href="grid.css">
</head>
<body>
 <header>Header</header>
 <nav>Navigation</nav>
 <main>
   <section class="left">Left Sidebar</section>
   <section class="content">Content Area</section>
   <section class="right">Right Sidebar</section>
 </main>
 <footer>2024 Grid Example</footer>
</body>
</html>
```

grid.css

```
body {
   font-family: Arial, sans-serif;
   margin: 0;
   padding: 0;
 }
```

```
 header, nav, main, footer {
  padding: 20px;
  text-align: center;
}
 header, footer {
  background-color: #333;
  color: #fff;
}
 nav {
  background-color: #666;
  color: #fff;
}
 main {
  display: grid;
  grid-template-columns: 200px 1fr 200px;
  grid-gap: 20px;
}
 .left, .right {
  background-color: #f4f4f4;
  padding: 10px;
}
 .content {
  background-color: #ddd;
  padding: 10px;
}
footer{
  bottom: 0;
  left: 0;
  position: fixed;
  width: 100%;
}
```

Introduction to Media Queries

- Media Queries are a CSS technique used to change the style and layout of a website depending on the device or viewport size.
- It allows you to apply different styles based on different conditions, such as screen size, device orientation, and screen resolution.
- The basic syntax of a media query is as follows:

```
@media screen and (min-width: 700px) {
  /* styles here */
}
```

- In this example, the styles inside the media query will only be applied when the viewport is 700 pixels or wider.
- Media queries can be used in conjunction with CSS styles to create a responsive design that adapts to the device or viewport size.