ASSIGNMENT-2

Q.1 Define CSS. Explain different types of CSS with example? Explain Advantages of CSS?

Solution: CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces.CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

There are three types of CSS. They are:

1. Inline CSS
2. Internal CSS
3. External CSS

Inline CSS:

The inline CSS in HTML is added to the HTML element with the help of a style tag followed by the CSS that you want to style the element with.

- Inline CSS in HTML is one of the techniques used to style an HTML page.
- They are applied only to a specific element.
- You cannot use the selectors in the inline CSS.
- These inline styles are not reusable which adds a disadvantage to it.
- They have the highest order of precedence in all the techniques used to style an HTML page.

For Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Inline Css</title>
</head>
```

```
    <body>
        <h1 style="color:green; text-align:center; margin-
top:20px;">Welcome to the page.</h1>
    </body>
</html>
```

Internal CSS: An internal CSS is used to define a style for a single HTML page. An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

For Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: yellow;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

External CSS:

An external style sheet is used to define the style for many HTML pages. To use an external style sheet, add a link to it in the <head> section of each HTML page.

For Example:
Index.html File

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Style.css File

```
body {
  background-color: yellow;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

The external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.

The advantages of using CSS (Cascading Style Sheets) are following:

1.Separation of Concerns: CSS allows separation of the structure (HTML) and presentation (CSS) of a web page. This separation makes code cleaner, easier to maintain, and more modular.

2.Consistency: CSS enables consistent styling across multiple web pages. By defining styles in a centralized CSS file, you can ensure uniformity in appearance throughout your website.

3.Ease of Maintenance: With CSS, you can make global style changes quickly by modifying a few lines of code in the CSS file. This saves time and effort compared to updating styles individually on each HTML page.

4.Reusability: CSS styles can be reused across multiple elements or pages. You can define classes, IDs, or other selectors once and apply them to multiple elements as needed, reducing redundancy in code.

5.Flexibility: CSS offers flexibility in styling elements, allowing you to control layout, typography, colors, spacing, and more. It supports various layout techniques like floats, flexbox, and grid layout, enabling you to create diverse and complex designs.

6.Responsive Design: CSS supports responsive design, allowing web pages to adapt to different screen sizes and devices. Media queries enable you to apply different styles based on viewport width, height, orientation, and resolution, ensuring optimal user experience across devices.

7.Accessibility: CSS provides accessibility features to improve the usability of web pages for all users, including those with disabilities. You can use CSS to enhance readability, provide high contrast for better visibility, and improve navigation for screen readers.

8.Faster Page Loading: By separating style information into external CSS files, you can reduce the size of HTML files, resulting in faster page loading times. Browser caching of CSS files further improves performance by storing styles locally, reducing the need for repeated downloads.

9.SEO Benefits: CSS can indirectly improve search engine optimization (SEO) by making web pages more accessible, readable, and user-friendly. Well-structured and semantically meaningful HTML, styled with CSS, contributes to better indexing and ranking by search engines.

10.Cross-Browser Compatibility: CSS helps ensure consistent rendering of web pages across different web browsers and platforms. Browser-specific CSS hacks and vendor prefixes can be used to address compatibility issues and ensure a consistent user experience.

Q2. Make use of CSS selectors to narrow down the element selection?

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Styled HTML Elements</title>
    <style>
        /* Targeting specific elements using element selectors */
        h1 {
            color: #333; /* Text color */
            text-align: center; /* Center alignment */
            font-size: 28px; /* Font size */
            margin-top: 20px; /* Top margin */
        }

        p {
            color: #666; /* Text color */
            font-size: 16px; /* Font size */
            line-height: 1.6; /* Line height */
            margin-bottom: 20px; /* Bottom margin */
        }

        /* Targeting specific elements using class selectors */
        .button {
            padding: 10px 20px; /* Padding */
            background-color: #007bff; /* Background color */
            color: #fff; /* Text color */
            border: none; /* Remove border */
            border-radius: 5px; /* Border radius */
            cursor: pointer; /* Cursor style */
            display: inline-block; /* Make it inline-block to adjust width */
```

```
        text-align: center; /* Center alignment */
        text-decoration: none; /* Remove any default text decoration */
    }

    /* Hover effect for button */
    .button:hover {
        background-color: #0056b3; /* Background color on hover */
    }
    /* Targeting specific elements using ID selectors */
    #container {
        max-width: 600px; /* Maximum width */
        margin: 0 auto; /* Center align */
        padding: 20px; /* Padding */
        background-color: #f9f9f9; /* Background color */
        border-radius: 5px; /* Border radius */
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Box shadow */
    }
  </style>
</head>
<body>
<div id="container">
  <h1>Welcome to Our Website</h1>
  <p>This is a simple and useful example demonstrating styled HTML elements. You can customize these styles to fit your project needs.</p>
  <p>For more information, visit our <a href="#">About Us</a> page.</p>
  <a href="#" class="button">Get Started</a>
</div>
</body>
</html>
```

Q.3 Apply the CSS properties to design different HTML elements

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

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Styled HTML Elements</title>
    <style>
        /* Styles for heading */
        h1 {
            color: #333; /* Text color */
            text-align: center; /* Center alignment */
            font-size: 28px; /* Font size */
            margin-top: 20px; /* Top margin */
        }
        /* Styles for paragraph */
        p {
            color: #666; /* Text color */
            font-size: 16px; /* Font size */
            line-height: 1.6; /* Line height */
            margin-bottom: 20px; /* Bottom margin */
        }
        /* Styles for link */
        a {
            color: #007bff; /* Link color */
            text-decoration: none; /* Remove underline */
        }

        /* Styles for button */
        .button {
            padding: 10px 20px; /* Padding */
            background-color: #007bff; /* Background color */
            color: #fff; /* Text color */
            border: none; /* Remove border */
            border-radius: 5px; /* Border radius */
            cursor: pointer; /* Cursor style */
```

```
            display: inline-block; /* Make it inline-block to adjust width */
            text-align: center; /* Center alignment */
            text-decoration: none; /* Remove any default text decoration */
        }

        /* Hover effect for button */
        .button:hover {
            background-color: #0056b3; /* Background color on hover */
        }

        /* Styles for container */
        .container {
            max-width: 600px; /* Maximum width */
            margin: 0 auto; /* Center align */
            padding: 20px; /* Padding */
            background-color: #f9f9f9; /* Background color */
            border-radius: 5px; /* Border radius */
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Box shadow */
        }
    </style>
</head>
<body>

<div class="container">
    <h1>Welcome to Our Website</h1>
    <p>This is a simple and useful example demonstrating styled HTML elements. You can customize these styles to fit your project needs.</p>
    <p>For more information, visit our <a href="#">About Us</a> page.</p>
    <a href="#" class="button">Get Started</a>
</div>

</body>
</html>
```

Q.4 Apply CSS properties to create simple page layout

In HTML,layout consists of:

- A header with a title.
- A navigation bar with links.
- A main content area.
- A footer with copyright information.

Example of create simple page layout using CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Page Layout</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }

        header {
            background-color: #333;
            color: #fff;
            padding: 20px;
            text-align: center;
        }

        nav {
            background-color: #444;
            color: #fff;
```

```
            padding: 10px;
            text-align: center;
        }

        nav ul {
            list-style-type: none;
            padding: 0;
            margin: 0;
        }

        nav ul li {
            display: inline;
            margin-right: 20px;
        }

        main {
            padding: 20px;
        }

        footer {
            background-color: #333;
            color: #fff;
            padding: 10px;
            text-align: center;
            position: fixed;
            bottom: 0;
            width: 100%;
        }
    </style>
</head>
<body>

<header>
```

```
    <h1>Simple Page Layout</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>
    <h2>Main Content Area</h2>
    <p>This is the main content of our simple page layout.</p>
</main>

<footer>
    <p>© 2024 Simple Page Layout. All rights reserved.</p>
</footer>

</body>
</html>
```

Q.5 Explain CSS BOX MODEL with example?

The CSS Box Model is a fundamental concept in web design that describes how elements are structured and displayed in HTML. Each HTML element is considered a rectangular box, and the box model defines the properties of this box, including its content area, padding, border, and margin.

Components of the Box Model:

1. **Content:** This is the innermost part of the box and holds the actual content, such as text, images, or other HTML elements.
2. **Padding:** The padding is the space between the content and the border. It provides cushioning around the content and helps separate it from the border.
3. **Border:** The border surrounds the padding and content of the element. It can have a specific width, style, and color, defining the visual boundary of the element.
4. **Margin**: The margin is the space outside the border of the element. It creates separation between adjacent elements and affects the layout of the page.

Example of demonstrating Box model in CSS.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>CSS Box Model Example</title>
<style>
.box {
   width: 200px;
   height: 220px;
   padding: 20px;
   border: 2px solid black;
   margin: 30px;
   border-radius:12px;
}
h1,p{
text-align:center;
}
</style>
</head>
<body>
```

```
  <div class="box">
<h1>BOX MODEL</h1>
      <p>The box model is important for understanding how elements are positioned and how their sizes are determined on a web page</p>
  </div>
</body>
</html>
```

6.Creating Layouts with display, position and float property

1.Using display Property:

The display property in CSS defines how an element should be displayed. It has various values like block, inline, inline-block, flex, grid, etc. Here's an example of using the display property with the grid value to create a simple grid layout:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Display Property - Grid Layout</title>
   <style>
      .container {
         display: grid;
         grid-template-columns: 1fr 1fr 1fr; /* Three equal-width columns */
         gap: 20px; /* Gap between grid items */
      }

      .item {
         background-color: #f0f0f0;
         padding: 20px;
      }
```

```
    </style>
</head>
<body>
<div class="container">
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
    <!-- Add more items here -->
</div>
</body>
</html>
```

In this example:

- The .container div is set to display: grid, making it a grid container.
- The grid-template-columns property divides the grid container into three equal-width columns.
- Each .item div represents a grid item.

2.Using position Property:

The position property in CSS specifies the type of positioning method used for an element. It has values like static, relative, absolute, and fixed. Here's an example of using the position property with absolute value to position an element relative to its closest positioned ancestor:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Position Property - Absolute Positioning</title>
    <style>
        .container {
            position: relative; /* Establish positioning context for absolute child */
```

```
            width: 300px;
            height: 200px;
            background-color: #f0f0f0;
        }
        .absolute-element {
            position: absolute;
            top: 50%; /* Position 50% from top */
            left: 50%; /* Position 50% from left */
            transform: translate(-50%, -50%); /* Center element */
            background-color: #fff;
            padding: 20px;
        }
    </style>
</head>
<body>
<div class="container">
    <div class="absolute-element">
        This is an absolutely positioned element.
    </div>
</div>
</body>
</html>
```

In this example:

- The .container div is set to position: relative, creating a positioning context for its absolutely positioned child.
- The .absolute-element div is set to position: absolute, positioning it relative to its closest positioned ancestor (container in this case).

3.Using float Property:

The float property in CSS is used to float an element to the left or right of its container, allowing content to flow around it. Here's an example of using the float property to create a simple float-based layout with a sidebar and main content area:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>Float Property - Sidebar Layout</title>
   <style>
      .sidebar {
         float: left;
         width: 30%; /* Width of sidebar */
         background-color: #f0f0f0;
         padding: 20px;
      }
      .main-content {
         float: right;
         width: 70%; /* Width of main content */
         background-color: #fff;
         padding: 20px;
      }
      /* Clearfix to prevent collapsing margins */
      .clearfix::after {
         content: "";
         display: table;
         clear: both;
      }
   </style>
</head>
<body>
<div class="clearfix">
   <div class="sidebar">
      This is the sidebar.
```

```
    </div>
    <div class="main-content">
        This is the main content.
    </div>
</div>
</body>
</html>
```

In this example:

- The .sidebar div is floated left, and the .main-content div is floated right, creating a two-column layout.
- The .clearfix class is added to the container to clear the floats and prevent layout issues caused by floating elements.

7.Explain different types of css layouts / layout techniques with example

1.Float Layout:

Float layout is a traditional method used for creating multi-column layouts. Elements are floated left or right within their container, allowing content to flow around them. It's commonly used for creating simple grid-based layouts.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Float Layout</title>
```

```
  <style>
    .column {
      float: left;
      width: 50%; /* Two columns */
      padding: 10px;
    }

    /* Clearfix to prevent collapsing margins */
    .clearfix::after {
      content: "";
      display: table;
      clear: both;
    }
  </style>
</head>
<body>

<div class="clearfix">
  <div class="column" style="background-color: #f0f0f0;">
    <h2>Column 1</h2>
    <p>This is the content of column 1.</p>
  </div>
  <div class="column" style="background-color: #ccc;">
    <h2>Column 2</h2>
    <p>This is the content of column 2.</p>
  </div>
</div>

</body>
</html>
```

2.Flexbox Layout:

Flexbox is a one-dimensional layout model used for arranging elements in a row or column.

It provides powerful alignment and distribution capabilities, making it ideal for creating responsive designs.

Flexbox simplifies complex layouts and offers better control over spacing and alignment.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Flexbox Layout</title>
   <style>
      .container {
         display: flex;
         justify-content: space-between; /* Distribute items evenly */
         padding: 20px;
         background-color: #f0f0f0;
      }

      .item {
         flex: 1; /* Equal width */
         padding: 10px;
         background-color: #ccc;
      }
   </style>
</head>
<body>
<div class="container">
```

```
    <div class="item">
        <h2>Item 1</h2>
        <p>This is the content of item 1.</p>
    </div>
    <div class="item">
        <h2>Item 2</h2>
        <p>This is the content of item 2.</p>
    </div>
    <div class="item">
        <h2>Item 3</h2>
        <p>This is the content of item 3.</p>
    </div>
</div>
</body>
</html>
```

3.Grid Layout:

CSS Grid Layout is a two-dimensional layout system used for creating grid-based layouts with rows and columns.

It offers precise control over the placement and alignment of elements within the grid.

Grid Layout is highly versatile and efficient for designing complex and responsive layouts.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial scale=1.0">
    <title>Grid Layout</title>
    <style>
```

```
      .container {
         display: grid;
         grid-template-columns: repeat(3, 1fr); /* Three equal-width
columns */
         gap: 20px; /* Gap between grid items */
         padding: 20px;
         background-color: #f0f0f0;
      }
      .item {
         background-color: #ccc;
         padding: 10px;
      }
   </style>
</head>
<body>
<div class="container">
   <div class="item">
      <h2>Item 1</h2>
      <p>This is the content of item 1.</p>
   </div>
   <div class="item">
      <h2>Item 2</h2>
      <p>This is the content of item 2.</p>
   </div>
   <div class="item">
      <h2>Item 3</h2>
      <p>This is the content of item 3.</p>
   </div>
</div>

</body>
</html>
```