CASCADING STYLE SHEET (CSS)

CSS stands for **Cascading Style Sheets** and is used to control the look, feel and visual representation of HTML elements.

Relationship between HTML and CSS:

HTML is used to structure the content of a web page while CSS is used for styling and laying out content on the page.

Purposes Of CSS:

1. Styling and Layout
2. Responsiveness
3. Consistency

Basic CSS Syntax:

- **Selectors**: Target HTML elements to style.
- **Properties and Values**: Define the style for selected elements.
- **Declaration Block**: A set of rules that apply to the selected element.

Example:

```
h2  {
  font-family: “Brush Script MT”;
  color: blue;
}
```

h2 - Selector

{ } - Declaration block

color - property

blue - value

Types of CSS:

- **Inline CSS:**

  This is CSS code written directly in HTML tags. This is achieved by adding the **`style`** attribute within the html tag.

  Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS Styling Example</title>
</head>
<body>
    <div>
        <h1 style="color:blue">Style Web Page</h1>
        <p style="font-size: 16px;">This is a paragraph
with some basic styling.</p>
    </div>
</body>
</html>
```

- **Internal CSS**

  This is CSS code written in the **`<head>`** section of the html document. It is achieved by adding the **`<style>`** tag in the head section of the html document.

  Example:

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>CSS Styling Example</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        h1 {
            color: blue;
        }
        p {
            font-size: 16px;
            line-height: 1.5;
        }
        .container {
            width: 80%;
            margin: 0 auto;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Style Web Page</h1>
        <p>This is a paragraph with some basic styling.</p>
    </div>
</body>
</html>
```

- **External CSS**

  This is CSS code written in a blank document and called within the HTML document via the **<link>** tag in the head section of the HTML file.

  Example:

  mystyle.css

```
body {
    font-family: Arial, sans-serif;
}
p {
    font-size: 16px;
    line-height: 1.5;
}
.container {
    width: 80%;
    margin: 0 auto;
}
```

  index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS Styling Example</title>
    <link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>
    <div class="container">
        <h1>Style Web Page</h1>
        <p>This is a paragraph with some basic styling.</p>
    </div>
</body>
</html>
```

Selectors and Combinators:

Selectors are used to find the HTML elements that we want to style. They are a way of accessing elements on the web page and styling them in a way of choice.

Combinators on the other hand are objects that are used to define the relationship between two or more selectors.

Types of combinators:

- **Descendant combinator:** This is represented by a space **( )** character and is used to define styling for elements under a specific parent tag.
- **Child Combinator:** This is represented by the angle bracket **(>)** and is used to define styling for all elements that are direct children of the parent tag.
- **Next sibling combinator:** This is represented by the addition **(+)** and is used to define styling for an element that comes immediately after a specified tag
- **Subsequent sibling combinator:** This is represented by the tilde **(~)** and is used to define styling for all elements that are next siblings of a specified element

Types of selectors:

1. **Basic selectors (Simple selectors):**

   These are selectors that are used to style elements based on the element name, id, or class.

   Selection by name:

   This is used to style elements on a web page based on a given tag.

   Example:

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

The above declaration would be used to style all paragraphs on a web page.

Inorder to select all elements on the web page, we use an asterisk (*) symbol before the declaration block. E.g

```
* {
  text-align: center;
  color: red;
}
```

Selection by id:

This is used to style an element on a web page using its unique identifier created by the id attribute in the element tag . To select an element with a specific id, we would write a **hash (#)** character before the id of the element and then a declaration block.

Example:

```
<p id="first">Hello World!</p>

<p>My other content</p>
```

```
#first {
  text-align: center;
  color: red;
}
```

This would be used to style only the paragraph with the **id** as **`first`** on the web page.

Selection by class:

This is used to style elements on a web page that have corresponding values under the class attribute of the element tags. To select elements with the same class value, we would write a **dot (.)** character before the class name and then a declaration block.

Example:

```
<h1 class="center">Red and center-aligned heading</h1>

<p class="center">Red and center-aligned paragraph.</p>
```

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

Other types of selectors include:

2. **Grouping Selectors**:

   Combine multiple selectors to apply the same styles to different elements.

   ```
   h1, p, h2  {
     text-align: center;
     color: red;
   }
   ```

   Would be used to style h1 headers, h2 headers and paragraphs with the same style.

3. **Descendant Selectors**:

   These target elements within a specific parent. They use the descendant combinator to style elements e.g., `div p` targets all `<p>` inside `<div>`.

4. **Child Selectors**:

   These select direct children of an element, e.g., **`div > p`** targets only **`<p>`** directly inside **`<div>`**.

5. **Attribute Selectors**:

   These select elements based on their attributes, e.g., `[type="text"]` targets input fields of type text.

Basic CSS Styling:

1. **Styling Text:**

   The following are properties that can be used when styling text in a web page.

   - **`text-align:`** Used to set the horizontal alignment of text in a web page

     ```
     h2  {
       text-align: center;
     }
     ```

   - **`color:`** Used to set color of a text in a web page

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

   - **`font-family:`** This is used to set the font family of a text on the web page.

     ```
     h2  {
       font-family: “Times New Roman”;
     }
     ```

This is used to set the font family of an h2 header. However, not all fonts may be supported by a web browser. This is when we can use **fall back fonts.** These are necessary as they allow a text to be displayed in a different font incase the browser doesn't support the set font family

```
h2  {
  font-family: “Brush Script MT”, “Times New Roman”, Verdana;
}
```

This sets the font family of an h2 header to `Brush Script MT` but in case a browser can't easily display it, it will fall back to `Times New Roman` and in case it also can't be displayed, it will display the `Verdana font.`

`Note:` If a font family name is more than one word, it should be added to the CSS declaration block using double quotes ("`Times New Roman`") and can be omitted in case the font family name is a single word.

- **`font-size:`** This is used to set the size of text.

```
h2  {
 font-size: 20px;
}
```

This is used to set the text size of an h2 header to 20px.

`Note:` The font property can be added to a css declaration block as a shorthand property that can be used to cater for all properties dealing with fonts.

```
h2  {
 font: “Brush Script MT” 20px;
}
```

This is used to add a font family of `"Brush Script MT"` to an h2 header but also set its text size to 20px.

2. **Styling backgrounds:**

Backgrounds can be added to html elements using the following CSS properties

- **`background-color:`** This simply sets a color to the background of an html element or the web page

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

- **`background-image:`** This adds an image to the background of an html element or web page

```
div  {
     background-image: url(myimage.png);
}
```

`Note:` The background property can be added to a css declaration block as a shorthand property that can be used to cater for all properties dealing with backgrounds.

```
h2  {
 background: green;
}
```

3. Box Model:

The box model defines how an element is structured in terms of content, padding, border, and margin.

- **Content**: The actual content of the element (text, images, etc.).

- **Padding**: Space between the content and the border.
- **Border**: Surrounds the padding and content.
- **Margin**: Space outside the border, separating elements.

```
div {
    width: 300px;
    padding: 20px;
    border: 5px solid black;
    margin: 10px;
    box-sizing: border-box; /* Prevents the padding and border from affecting the width */
}
```

4. Layout techniques

Layouts define the structure, positioning and visibility of elements on a web page. Layouts can be achieved in different ways and below are some examples:

- **`display:`**

  This is used to change the default display of an element. All elements are either displayed as inline or block but the display property can be used to change the predefined display. There are other different types of display but these are mainly used:

  - `inline` - sets an html element to be inline (fill available width)
  - `block` - sets an html element to be block (fill maximum width)
  - `none` - Removes the object from being displayed

```
li {
  display: inline;
}
```

- **`position`**

  This specifies the positioning of an element on a web page. These can be implemented using the following values:

  - `static` - element position isn't affected at all
  - `relative` - element position is relative to its normal position
  - `fixed` - element position is relative to the viewport meaning it will stay fixed even when page is scrolled
  - `absolute` - positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
  - `sticky` - is positioned based on the user's scroll position. It toggles between relative and fixed

  `Note:` The position property is used alongside **`top, bottom, left`** and **`right`** attributes to affect the position property.

```
div {
  position: absolute;
  top: 80px;
  right: 0;
  }
```

- **`float`**

  This defines the direction in which an element should float relative to other information on the web page

  - `left` - The element floats to the left of its container
  - `right` - The element floats to the right of its container

- `none` - The element does not float (will be displayed just where it occurs in the text). This is default
- `inherit` - The element inherits the float value of its parent

```
img {
  float: right;
}
```

There are other CSS properties used to design layouts but some include

- **`flex:`** used to design 1 dimension rows and columns
- **`Grid:`** Used to design 2D rows and columns

5. Responsive CSS:

While working with different devices having different sizes and resolutions, CSS helps us make web pages responsive with the help of media queries.

Media queries are conditional CSS rules based on viewport size, device orientation, and resolution.

CSS Media Types

- `all -` Used for all media type devices
- `print -` Used for print preview mode
- `screen -` Used for computer screens, tablets, smart-phones etc.

CSS Media Features

- `orientation -` orientation of the view port (landscape or portrait orientation)
- `max-height -` maximum height of the view port
- `min-height -` minimum height of the view port
- `height -` Height of the viewport (including scrollbar)
- `max-width -` maximum width of the view port

- `min-width` - minimum width of the view port
- `width` - width of the viewport (including scrollbar)

Usage of Media Queries:

```
@media not|only mediatype and (media feature) and (media feature) {
  CSS-Code;
}
```

Meaning of keywords:

- `not:` This keyword inverts the meaning of an entire media query.
- `only:` This keyword prevents older browsers that do not support media queries from applying the specified styles. It has no effect on modern browsers.
- `and:` This keyword combines a media type and one or more media features.

Example:

```
@media screen and (min-width: 480px) {
  body {
    background-color: lightgreen;
  }
}
@media (min-width: 768px) {
    body {
        font-size: 18px;
    }
}
```

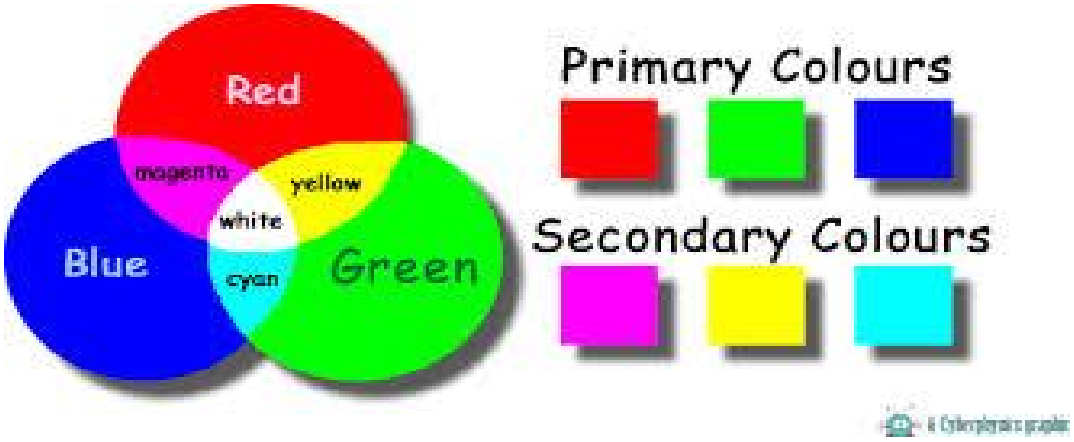
HTML COLORS:

HTML colors are specified with predefined color names, or with RGB, HEX, HSL, RGBA, or HSLA values.

Primary and secondary Colors:

Primary colors (red, green, and blue) are fundamental colors that cannot be created by mixing other colors. Primary colors are Red, Green and Blue

Secondary colors are created by combining two primary colors. Secondary colors are Yellow, Cyan and Magenta



1. **RGB Colors:**

   Colors are supported by all browsers and an rgb color is specified using:

   ```
   rgb( red, green, blue )
   ```

   Each parameter (red, green, and blue) defines the intensity of the color with a value between 0 and 255.

   Examples:

   - `rgb(255, 0, 0)` is displayed as **`red`**, because red is set to its highest value (255), and the other two (green and blue) are set to 0.
   - `rgb(0, 255, 0)` is displayed as **`green`**, because green is set to its highest value (255), and the other two (red and blue) are set to 0.

- To display **`black`**, set all color parameters to 0, like this: `rgb(0, 0, 0).`
- To display **`white`**, set all color parameters to 255, like this: `rgb(255, 255, 255).`

In CSS, the rgb colors are set as below:

```
div {
     background-color: rgb(0, 191, 255);
     color: rgb(255, 255, 255);
}
```

2. **RGBA Colors**

RGBA color values are an extension of RGB color values with an Alpha channel - which specifies the opacity for a color. An RGBA color value is specified with:

`rgba(`*`red`*`, `*`green`*`, `*`blue`*`, `*`alpha`*`)`

The **`alpha`** parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

```
div {
     background-color: rgba(0, 191, 255, 0.5);
     color: rgb(255, 255, 255);
}
```

3. HEX Colors

Hexadecimal color values are supported in all browsers. A hexadecimal color is specified with:

#rrggbb

Where `rr (red), gg (green) and bb (blue)` are hexadecimal integers between **00** and **ff**, specifying the intensity of the color.

Examples:

- **`#ff0000`** is displayed as **red**, because red is set to its highest value (ff), and the other two (green and blue) are set to 00.
- Another example, **`#00ff00`** is displayed as **green**, because green is set to its highest value (ff), and the other two (red and blue) are set to 00.
- To display **black**, set all color parameters to 00, like this: **`#000000`**.
- To display **white**, set all color parameters to ff, like this: `#ffffff.`

In CSS, the hex colors are set as below:

```
div {
      background-color: #00bfff;
      color: #ffffff;
}
```

4. HSL Colors

HSL stands for Hue, Saturation, and Lightness. This format however isn't acceptable in all browsers.
HSL color values are specified with:

`hsl(`***`hue`***`, `***`saturation`***`, `***`lightness`***`)`

`Hue -` is a degree on the color wheel from 0 to 360. 0 (or 360) is red, 120 is green, 240 is blue.

`Saturation` - is the intensity of a color. It is a percentage value from 0% to 100% with 100% as the full color, 50% as 50% gray and 0% as completely gray and can no longer see the color.

`Lightness -` is how much light you want to give the color. 0% means no light (dark), 50% means 50% light (neither dark nor light), and 100% means full light.

Examples:

- `hsl(0, 100%, 50%)` is displayed as **`red`**, because hue is set to 0, and the saturation to full color (100%), and lightness to neither white or black (50%).
- `rgb(120, 100%, 50%)` is displayed as **`green`**, because hue is set to 120, and the saturation to full color (100%), and lightness to neither white or black (50%).
- To display **`black`**, set the lightness to 0%, like this: `hsl(0, 0%, 0%).`
- To display **`white`**, set the lightness to 0%, like this: `hsl(0, 0%, 100%).`

In CSS, the hsl colors are set as below:

```
div {
     background-color: hsl(170, 50%, 50%);
     color: hsl(0, 50%, 50%);
}
```

5. **HSLA Color Values**

   HSLA color values are an extension of HSL color values, with an Alpha channel - which specifies the opacity for a color.

   An HSLA color value is specified with:

   hsla(*hue, saturation, lightness, alpha*)

   The **alpha** parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

```
div {
    background-color: hsla(170, 50%, 50%, 0.5);
    color: hsla(0, 50%, 50%, 0.9);
}
```

Trial Questions:

1. What will be the color code in hsl, rgb, rgba, hsla, and hex for the following colors
   - Red
   - Green
   - Yellow
   - Cyan
   - Magenta
2. Given the html piece of code, write a simple CSS code on how You can style the text Samuel below so that it's red in color and has a background color of green and font of Helvetica.

   <h2>My brother, Samuel Peter lives in Denmark</h2>

3. Create a website and make it responsive on different screens with media queries.