MILESTONE 4

- What and Why CSS

- SASS and SCSS

- Syntax

- Basic Properties

- Height, Width

- Margins, Paddings, Border

- Background

- Text, Font, Image

- Box Model, Display, Flex

1. **What and why CSS**
    - What is CSS - Cascading Style Sheets (CSS) is a stylesheet language used to describe presentation of a document written in HTML or XML.
    - Why CSS – CSS is used to define styles of our web pages, including design, layout, variation, display for different devices and screen size**.**
2. **SASS and SCSS**
    - SCSS and SASS are syntaxes of SASS pre-processor with advanced CSS feature.
    - Extension of SCSS is .scss and SASS is .sass
    - SCSS
        - Sassy CSS offers more CSS-like syntax.
        - It includes variables, nesting, mixins, inheritance with standard CSS.
        - Ex - $bgcolor, $textcolor etc for styling. They are used to set background color, color etc
        - Mixins – create reusable chunk of styles, to avoid repetitive code.
            - Ex - @mixin important-text {
              color: red;
              font-size: 25px;
              font-weight: bold;
              border: 1px solid blue; }

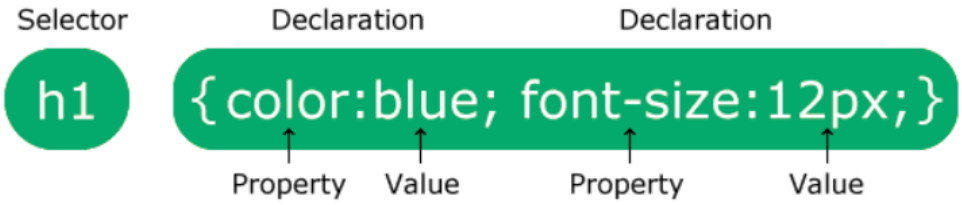
```
.danger {
  @include important-text;
  background-color: green;
}
```

- SASS
    - Syntactically Awesome Stylesheets is a CSS pre-processor that extends CSS with features
- Difference between SCSS and SASS
    - SCSS – It has braces and semicolons
    - SASS – It is based on indentations without braces or semicolon.
- SASS reduces repetition of CSS and saves time.
- EX - $primary_1: #a2b9bc;

```
.header {
    background-color: $primary_1;
}
```

    - Here if we want to change the HEX value, it can be changed in one place instead of changing it everywhere like in CSS.
- Browser doesn't understand SASS code. So transpiler converts SASS code to css. This process is called transpilling.

3. **HTML, CSS Syntax**
    - CSS Syntax –



- HTML Syntax – refers to set of rules and syntax to create a valid HTML document.
    a. It should have opening<> and closing tag</>.
    b. Every HTML documentation should begin with <!DOCTYPE html> declaration.

4. **Basic properties**

a. **Height, width**
    i. Height and width properties are used to set height and width of an element.
    ii. Max-width : o set the maximum width of an element. It has specified values like px, cm, %
    iii. Min-width: sets minimum width of an element.
    iv. Max-height: sets maximum height of an element.
    v. Min-height: sets minimum height of an element.
b. **Margin**
    i. Margin is used to create space around elements, outside any defined borders.
        1. Margin-top
        2. Margin-right
        3. Margin-bottom
        4. Margin-left
    ii. It can have following values – auto, length (px, pt, cm), %, inherit (margin should be inherited from parent element).
    iii. It is a shorthand property. Ex: margin: 20px 21px 22px 23px –
        1. 20px – top margin
        2. 21px – right margin
        3. 22px – bottom margin
        4. 23px – left mrgin
    iv. If it has 3 values, Ex: margin: 20px 21px 22px,
        1. 20px – top margin
        2. 21px – right and left margin
        3. 22px – bottom margin
    v. If margin: 25px 50px
        1. 1. 25px – top and bottom margin
        2. 50px – right and left margin
    vi. It margin: 25px, then all four sides have 25px
    vii. By using **margin: auto**, it horizontally aligns to centre within its container.
    viii. By using **margin: inherit,** It inherits the property from the property.
c. **Padding**
    i. Padding is used to create space around an element, inside any defined borders.
    ii. It has same properties as margin, like adding-top, padding-right, padding-bottom, padding-left.
    iii. It is also a short-hand property.

d. Background

i. It is a short-hand property. Ex: background: lightblue url("img_tree.gif") no-repeat fixed center; }
   1. Above example comprises of background-color, background-image, background-position, background-size, background-reeat, background-origin, background-clip, background-attachment.
   2. Its values can be left top, left center, left bottom, right top, right center, right bottom, center top, center bottom.
   3. It can also be x% y% - where 1st value is horizontal position, 2nd value is vertical position. Top left corner is 0% 0%, right bottom corner is 100% 100%. Default value is 0% 0%
   4. It can also be mentioned as xpos ypos.

ii. Background-position-x property sets position of background image on x-axis. Same goes for background-position-y

iii. Background-repeat : It says how the image is to be repeated. It has values like repeat, repeat-x, repeat-y, no-repeat, space, initial, inherit.

iv. Background-size. It has values like auto, cover, contain, initial, inherit. It can also have 2 values, where 1st value represents width of image. 2nd value represents height of the image.

v. Box-shadow: **EX:** box-shadow: 0 0 2px 1px rgba(0, 140, 186, 0.5);
   1. 1st 0 – horizontal offset
   2. 2nd 0 – vertical offset
   3. 2px – blur radius
   4. 1px – spread radius

e. Font

i. Choosing right font has a huge impact on how readers experience a website.

ii. Serif – small stroke at edges.

iii. Sans-serif: have clean lines.

iv. Font-family has **fallback** system. So start with font-family what we want, end with generic family. If starting font is not available, then generic font will be picked by the browser.

f. Text

i. Text color is specified by color: red.

ii. Background colour is specified by background-color: red

iii. Text-align is used to set horizontal alignment of a text.
   1. It has 4 values – center, left, right, justify
   2. Text-align last: specifies how to align last line of a text.

iv. Text-decoration: used to add decoration to text.

v. It is also one of the shorthand property.
   1. Ex: text-decoration: underline red double 5px

a. Text-decoration-line: underline (req)
b. Text-decoration-color: red (optional)
c. Text-decoration-style: double (optional)
d. Text-decoration-thickness: 5px (optional)

vi. Text-transform: uppercase/ lowercase/ capitalize (capitalizes 1st letter of the word).

vii. Text spacing
1. *Text-indent:* used to specify indentation of the first line of a text. **EX:** text-indent: 10px
2. *Letter-spacing*: used to specify space between characters in a text. **Ex**: letter-spacing: 5px
3. *Line-height:* used to specify the space between lines. **EX:** line-height: 0.8
4. Word-spacing: used to specify space between words in a text. **EX:** word-spacing: 10px

viii. White-space: specifies how white space inside an element is handled. **EX:** white-space: no-wrap (doesn't goes to next line)

ix. Text-shadow: adds shadow to text **EX:** text-shadow: 2px 3px black
1. Here 2px – adds horizontal shadow
2. 3px – adds vertical shadow
3. Black – color of shadow
More than 1 property can be given.

g. Images

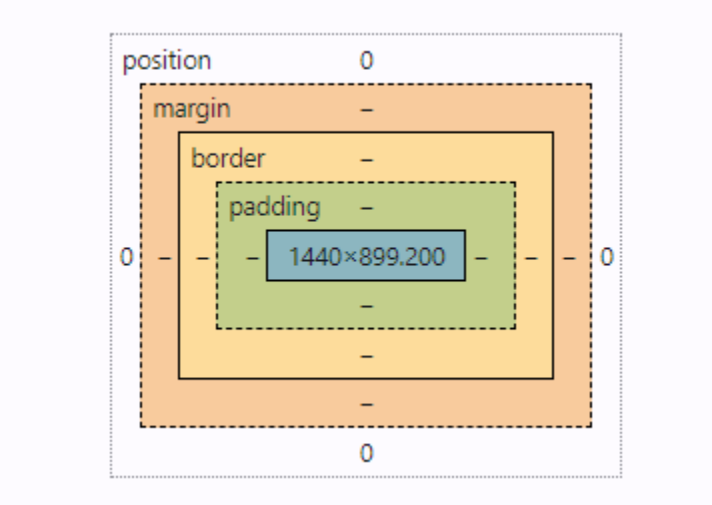
i. Border-radius: 50% → creates circled image

ii. Border-radius: 10px → creates rounded-image

iii. Transparent images can be created using **opacity.** It has value between 0.0-1.0

iv. **Filter** property is used to add sharp/ saturation to the image.
1. **EX:** {filter: blur(4px);}
{filter: brightness(250%);}
{filter: contrast(180%);}
{filter: grayscale(100%);}
{filter: hue-rotate(180deg);}
{filter: invert(100%);}
{filter: opacity(50%);}
{filter: saturate(7);}
{filter: sepia(100%);}
{filter: drop-shadow(8px 8px 10px green);}

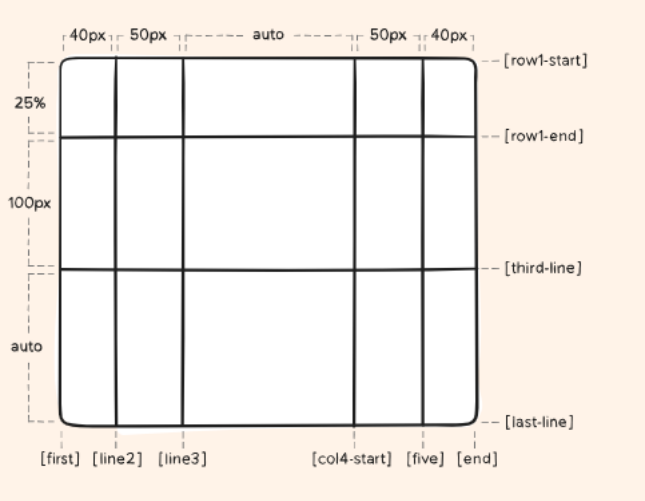
5. **CSS Box Model**
   - It is used to design and layout.
   - It is a box that wraps around every HTML element. It consists of content, padding, borders, margin. Below diagram refers to box model



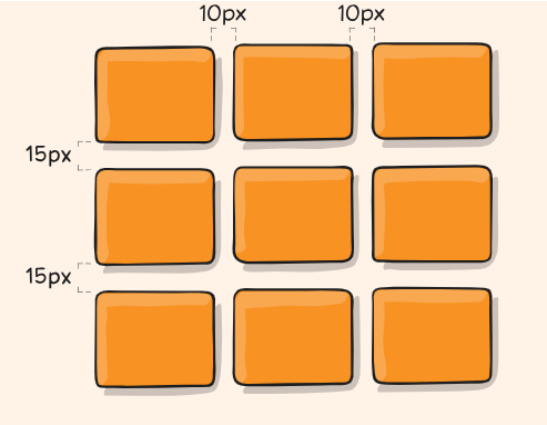
6. **Display**
   a. Display has many values like
      i. Inline – it will accept margin, padding. But it will not accept height, width. Margin and padding will push elements horizontally and not vertically. It doesn't break the flow of text.
      ii. Inline-block – Same as inline but here it is possible to set width, height.
      iii. Block - A block element always starts on a new line and takes up the full width available, from left to right, unless you specify a width.
      iv. Flex – easy to arrange and align items inside a container. It helps to place elements side by side, space them out, center them, organize them into rows and columns
         1. It contains additional property: **justify-content (flex/ grid)**
            a. Center: Aligns flex-items at center of the container.
            b. Flex-start: Aligns flex-items at starting of the container.
            c. Flex-end: Aligns flex-items at end of the container.
            d. Space-between: Aligns flex-items with space between the lines.
            e. Space-around: Aligns flex-items with space before, between, after the lines.
            f. Space-evenly: Aligns flex-items with space evenly the lines.

2. **Align-items:** used to align items inside flex/ grid
   align-items: stretch;
   align-items: center;
   align-items: flex-start;
   align-items: flex-end;
   align-items: baseline;

v. Grid - Defines the element as a grid container and establishes a new grid formatting context for its contents.
   1. Grid-template-columns, grid-template-rows: defines columns and rows of grid with space-separated list of values.
      a. **EX**:
      .container {
      grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];
      grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];
      }



      b. Grid-template-areas work with respect to grid-areas
      c. Column-gap, row-gap: specifies size of grid lines
         column-gap: 10px;
         row-gap: 15px;



d. Gap is a short-hand property for row-gap, column-gap
gap: 15px 10px; → 15px (row gap); 10px (col gap)

7. **Flex:**
   a. Flex-grow: it controls how a flex item should stretch more than others.
      i. .box1{

         Flex-grow: 1;}

         .box2{

         Flex-grow: 2;}

         Here box2 will have more space than box1

   b. Flex-shrink: tells how much a box should shrink, when space gets tight. If we want one box to shrink more than others, big number has to be given
   c. Flex-basis: defines initial size of a flex item before it starts growing or shrinking
   d. Flex is a shorthand property of flex-grow, flex-shrink, flex-basis.

Syntax: `flex: flex-grow flex-shrink flex-basis |auto|initial|inherit;`

   e. Flex-flow is a shorthand property of flex-direction, flex-wrap.
   f. Flex-direction: used to specify how flex-items are displayed. Its vales are
      i. Row, row-reverse, column, column-reverse, initial, inherit.
   g. Flex-wrap: controls where the flex items should stay in one line or wrap into many lines if there is not enough space in container.
      i. Flex-wrap: wrap – elements will move into new row when there's not any enough space in that row.

ii. Flex-wrap: nowrap – default value. Specifies that the flexible items will not wrap.
iii. Flex-wrap: wrap-reverse – items will wrap, if needed in reverse order.

- Justify-content: to align items in a single line
- Align-content: Controls the spacing and alignment of lines when items wrap
- Justify-items: Aligns items horizontally within their individual grid cells in a Grid container. Applies to a single cell/ item
- Align-items: Aligns items vertically within the container in Flexbox or Grid layouts. Applies to entire container.
- Justify- self: Aligns an individual grid item horizontally within its grid area.

HTML

```
<div class="grid-container">
    <div class="item1">1</div>
    <div class="item2">2</div>
    <div class="item3">3</div>
  </div>
```

CSS

```
.grid-container {  display: grid;
 grid-template-columns: 100%
} .item1 { justify-self: start;  }
.item2 {
  justify-self: center;  }
.item3 {
  justify-self: end; }
```

- Align-self: Aligns an individual flex or grid item vertically (in a row) or horizontally (in a column) within its container, overriding the container's align-items setting.