Chapter – 0 (Introduction to CSS)

HTML is just a skeletal layout of a website. We need CSS to design a website, add styles to it and make it look beautiful.

What is CSS?

CSS stands for cascading style sheets
CSS is optional but it converts an off looking HTML page into a beautiful & responsive website

Installing VS Code

We will use Microsoft Visual Code as a tool to edit our code. It is very powerful, free, and customizable.

What Learn CSS?

CSS is a much-demanded skill in the world of web development If you are successfully able to master CSS, you can customize your website as per your liking.

Your first line of CSS

Create a .css file inside your directory and add it to your HTML. Add the following line to your CSS

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

This will make your page background red.

HTML Refresher

HTML is a bunch of tags used to lay the structure of a page.

Chapter – 1 (Creating our first CSS Website)

We will create our first CSS website in this section.

What is DOM?

DOM stands for document object model. When a page is loaded, the browser creates a DOM of the page which is constructed as a tree of objects.

HTML id and class attributes

When an HTML element is given an id, **it serves as a unique identifier for that element.**

On the other hand, when an HTML element is given a class, it now belongs to that class. More than one element can belong to a single class but every element must have a unique id (if assigned).

We can add multiple classes to an element like this,

```
<div id = 'first' class = 'C1 C2 C3'>
    ...
</div>
```

- first is the unique id
- C1, C2 and C3 are the multiple classes followed by spaces

Three ways to add CSS to HTML

There are 3 ways to add CSS to HTML:

1. <style> tag : Adding <style> ... </style> to HTML
2. **Inline CSS : Adding CSS using style attribute**
3. External CSS : Adding a stylesheet(.css) to HTML using <link> tag

CSS Selectors

A CSS Selector is used to select an HTML element(s) for styling

```
body {
        color: red;             #Declaration (property: value)
        background: pink;       #Declaration
}
```

- **Body is the selector**

Element Selector

It is used to select an element based on the tag name
For example:

```
H2 {
        color: blue;
}
```

Id Selector

It is used to select an element with a given id
For example:

```
#first {
        color: white;
        background: black;
}
```

 # is used to target by id

Class Selector
It is used to select an element with a given class
For example:

```
.red {
        background: red;
}
```

Important Notes:

- We can group selectors like this:

```
h1,h2,h3,div {
        color:blue;          /*h1,h2,h3 and div will be blue*/
}
```

- We can use element class as a selector like this:

```
p.red {
        color: red;          /*all paragraphs will get color of red*/
}
```

- * Can be used as a universal selector to select all the elements

```
* {
        margin: 0;
        padding: 0;
}
```

- An inline style will override external and internal styles

Comments in CSS

Comments in CSS are the text which is not parsed and is thus ignored.

Chapter – 1 (Practice Set)

1. Create a website with a class red div which has a background color of the red and color white.
2. Create an element with id head and verify that background color works on it as inline, external as well as using style tag CSS.
3. Create a CSS class one and verify that it works on multiple elements.
4. Create multiple CSS classes and verify that all of these work on the same element.
5. Have a look at the MDN CSS reference and try to play around with few key-value CSS rules.

Chapter – 2 (Colors & Backgrounds)

CSS rules are simple **key-value pairs with a selector**. We can write CSS rules to change color and set backgrounds.

The color property

The **CSS color property can be used to set the text color inside an element**

```
p{
        color: red;          /*Text color will be changed to red*/
}
```

Similarly, we can set color for different elements

Types of color values
Following are the most commonly used color values in CSS

1. RGB: Specify color using Red, Green, Blue values. E.g. RGB (200,98,70)
2. HEX Code: Specify color using hex code. E.g. #ff7180
3. HSL: Specify the color using hsl values. E.g. hsl(8,90%,63%)

HSL stands for Hue, saturation, and lightness
The value of the color or background color is provided as any one of these values.

Note: We also have RGBA and HSLA values for color but they are rarely used by beginners.

The background-color property
The CSS background-color property specifies the background color of a container.
For e.g.

```
.brown {
        background-color: brown;
}
```

The background-image property

Used to set an image as the background

```
body {
background-image: url(“harry.jpg”)
}
```

The image is by default repeated in X & Y directions

The background-repeat property

Can be any of:

- repeat-x : repeat in the horizontal direction
- repeat-y : repeat in the vertical direction
- **no-repeat : image not repeat**

See more possible values at MDN docs

The background-size property

Can be following:

- cover : fits & no empty space remains
- contain : fits & image is fully visible
- auto : display in original size
- {{width}} : set width & height will be set automatically
- {{width}} {{height}} : set width & height

Note: Always check the MDN docs to dissect a given CSS property. Remember, practice will make you perfect

The background-position property

Sets the starting position of a background image

```
.div1{
        background-position: left top;
}
```

The background-attachment property

Defines a scrollable/non-scrollable character of a background image

```
.div2{
        background-attachment: fixed
}
```

The background shorthand

A single property to set multiple background properties

```
.div3{
        background: red url('img.png') no-repeat fixed
right top;
}
```

One of the properties can be missing given the others are in order.

- {{width}} {{height}} : set width & height

Note: Always check the MDN docs to dissect a given CSS property. Remember, practice will make you perfect

The background-position property:
Sets the starting position of a background image

```
.div1{
        background-position: left top;
}
```

The background-attachment property
Defines a scrollable/non-scrollable character of a background image.

```
.div2{
        background-attachment: fixed;
}
```

The background shorthand
A single property to set multiple background properties

```
.div3{
        background: red url('img.png') no-repeat fixed right top;
```

```
}
```

Copy

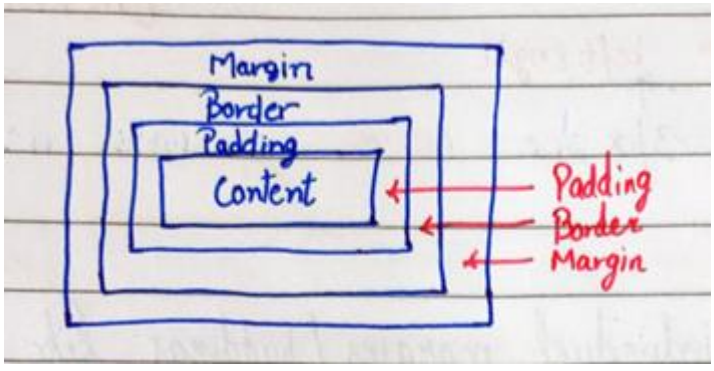
One of the properties can be missing given the others are in order.

Chapter – 2 (Practice Set)

1. Create a dark blue navigation bar with light color items.
2. Change the color of the main container on your page to dark red.
3. Create a div and add a background image with a given width and height.
4. Create a vertical box and add a fixed non-scrolling background to it.
5. Verify that the background shorthand property works with some of the values skipped.

Chapter – 3 (CSS Box Model)

The CSS box model looks at all the HTML elements as boxes



Setting Width & Height

We can set width and height in CSS as follows

```
#box {
        height: 70px;
        width: 70px;
}
```

Note that the total width/height is calculated as follows:
Total height = height + top/bottom padding + top/bottom border + top/bottom margin

Setting Margin & Padding

We can set margin and padding as follows:

```
.box{
        margin: 3px;        /* Sets top, bottom, left & right values*/
        padding: 4px;       /* Sets top, bottom, left & right values*/
}
```

```
.boxMain{
        margin: 7px 0px 2px 11px;      /*top, right, bottom, left*/
}
```

```
.boxLast{
        margin: 7px 3px;          /*(top & bottom) (left & right)*/
}
```

We can also set individual margins/padding like this:
margin-top: 70px
margin-bottom: 3px
margin-left: 8px
margin-right: 9px
Same goes with padding also

Setting Borders
We can set the border as follows

```
.bx{
        border-width: 2px;
        border-style: solid;
        border-color: red;
}
```

Shorthand for above codes,

```
set border: 2px solid red;
```

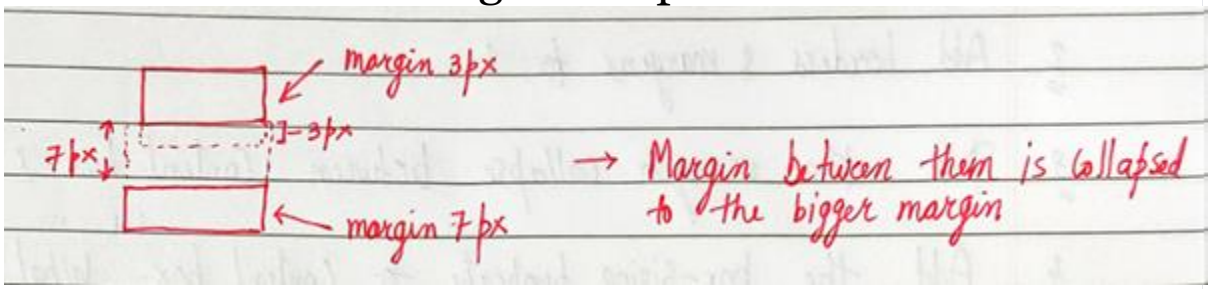
Border Radius

We can set border-radius to create rounded borders

```
.div2{
        border-radius: 7px;
}
```

Margin Collapse

When two margins from different elements overlap, the equivalent margin is the greater of the two. This is called margin collapse.



Box Sizing

Determines what out of padding and border is included in elements width and height

Can be content-box or border-box

- Include only content in width/height

```
.div1{
        box-sizing: border-box;
}
```

The content width and height include, content + padding + border

Chapter – 3 (Practice Set)

1. Create a website layout. Add a header box, one content box, and one footer.
2. Add borders and margins to question 1 (website layout)
3. Did the margin collapse between the content box and footer?
4. Add the box-sizing property to the content box. What changes did you notice?

Chapter – 4 (Fonts & Display)

The display property

The CSS display property is used to determine whether an element is treated as a block/inline element & the layout used for its children (flexbox/grid/etc.)

display: inline

Takes only the space required by the element. No line breaks before and after. Setting width/height (or margin/padding) not allowed.

display: block

Takes full space available in width and leaves a newline before and after the element

display: inline-block
Similar to inline but setting height, width, margin, and padding is allowed. Elements can sit next to each other

display: none vs visibility: hidden
With display: none, the element is removed from the document flow. Its space is not blocked.
With visibility: hidden, the element is hidden but its space is reserved.

text-align property
Used to set the horizontal alignment of a text

```
.div1{
        text-align: center;
}
```

text-decoration property
Used to decorate the text
Can be overline, line-through, underline, none

text-transform property
Used to specify uppercase and lowercase letters in a text

```
p.uppercase{
        text-transform: uppercase;
}
```

line-height property
Used to specify the space between lines

```
.Small{
        line-height: 0.7;
}
```

Font
Font plays a very important role in the look and feel of a website

Font-family
Font family specifies the font of a text.
It can hold multiple values as a “fallback” system

```
p{
        font-family: “Times new Roman”, monospace;
}
```

Always follow the above technique to ensure the correct font of your choice is rendered

Web Safe Fonts
These fonts are universally installed across browsers

How to add Google Fonts
In order to use custom, google fonts, go to google fonts then select a style, and finally paste it to the style.css of your page.

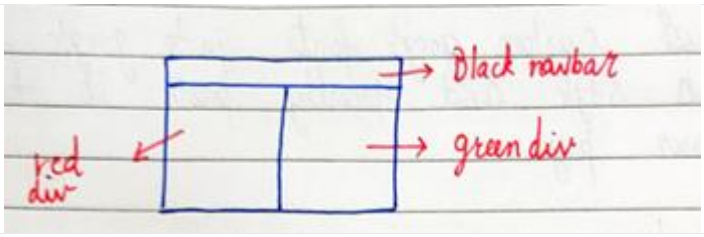
Other Font Properties

Some of the other font properties are listed below:
font-size: Sets the size of the font
font-style: Sets the font style
font-variant: Sets whether the text is displayed in small-caps
font-weight: sets the weight of the font

Generic Families
A broad class of similar fonts e.g. Serif, Sans-Serif
Just like when we say fruit, it can be any fruit
When we say Serif it can be any Serif font
font-family – Specific
Generic family - Generic

Chapter – 4 (Practice Set)

1. Create the following website layout,



1. Add a footer with Google Font “Ballu Bhai” to question 1.
2. Remove the underlines from links in question 1.
3. Demonstrate the difference between display: none and visibility: hidden using a div.
4. Change the footer to all uppercase in question 1.

Chapter – 5 (Size, Position & Lists)

There are more units for describing size other than 'px'
There are rem, em, vw, vh, percentages, etc.

What's wrong with pixels?

Pixels (px) are relative to the viewing device.
For a device with the size 1920x1080, 1px is 1unit out of 1080/1920.

Relative lengths

These units are relative to the other length property.
Following are some of the most commonly used relative lengths,

1. **em** – unit relative to the parent font size

em means "my parent element's font-size"

2. **rem** – unit relative to the root font size (<html> tag)
3. **vw** – unit relative to 1% viewport width
4. **vh** – unit relative to 1% viewport height
5. **%** - unit relative to the parent element

Min/max- height/width property

CSS has a min-height, max-height, and min-width, max-width property.

If the content is smaller than the minimum height, minimum height will be applied.

Similar is the case with other related properties.

The position property

Used to manipulate the location of an element
Following are the possible values:

- **static:** The default position. top/bottom/left/right/z-index has no effect

- **relative:** The top/bottom/left/right/z-index will now work. Otherwise, the element is in the flow of the document like static.

- **absolute:** The element is removed from the flow and is relatively positioned to its first non-static ancestor. top/bottom etc. works

- **fixed:** Just like absolute except the element is positioned relative to the browser window

- **sticky:** The element is positioned based on the user's scroll position

list-style property

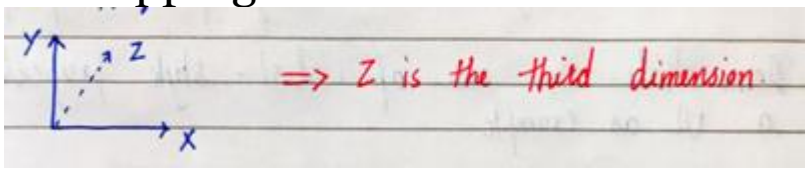
The list-style property is a shorthand for type, position, and image

```
ul{
        list-style: square inside url('harry.jpg')
}
```

- square in the above code is the list-style-type, 'inside' is the list-style-position and 'harry.jpg' is the list-style-image.

z-index property

The z-index property specifies the stack order of an element.
It defines which layer will be above which in case of overlapping elements.



Chapter – 5 (Practice Set)

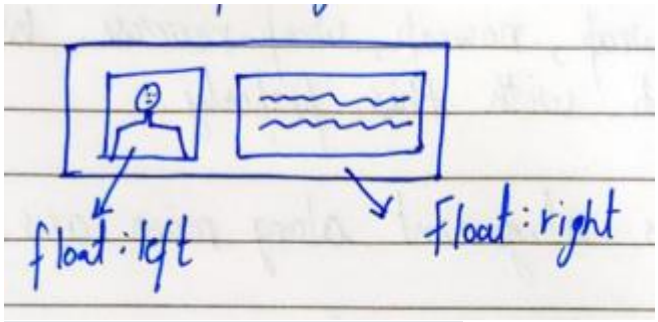
1. Create a responsive navbar using relative lengths.
2. Create a sticky navbar using the position property.
3. Demonstrate the use of list-style property using a ul as an example.
4. Demonstrate the use of z-index using an example.

Chapter – 6 (Flexbox)

Before we look into the CSS flexbox, we will look into float and clear properties.

The float property

float property is simple. It just flows the element towards left/right



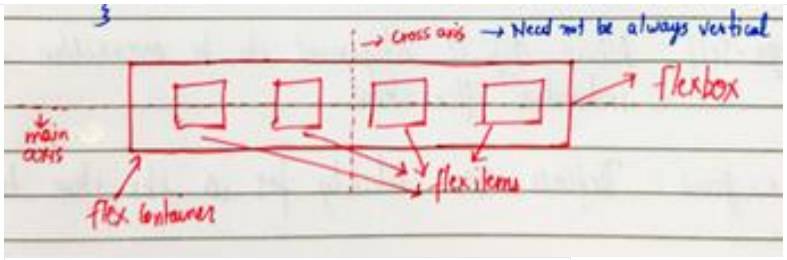
The clear property

Used to clear the float. It specifies what elements can float beside a given element

The CSS Flexbox

Aims at providing a better way to layout, align, and distribute space among items in a container.

```
.container{
        display: flex;                  /*Initialize a flexbox*/
}
```



flex-direction property

Defines the direction towards which items are laid. Can be row (default), row-reverse, column and column-reverse

Flex properties for parent (flex container)
Following are the properties for the flex parent:

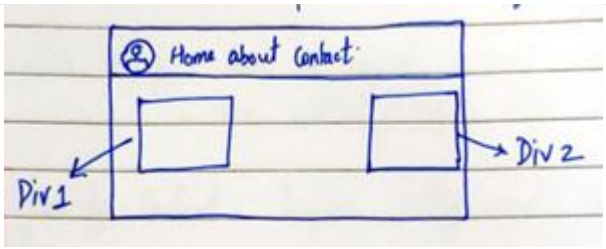
1. **flex-wrap:** Can be wrap, nowrap, wrap-reverse. Wrap items as needed with this property
2. **justify-content:** Defines alignment along the main axis
3. **align-items:** Defines alignment along the cross axis
4. **align-content:** Aligns a flex container's lines when there is extra space in the cross axis

Flex properties for the children (flex items)
Following are the properties for the flex children:

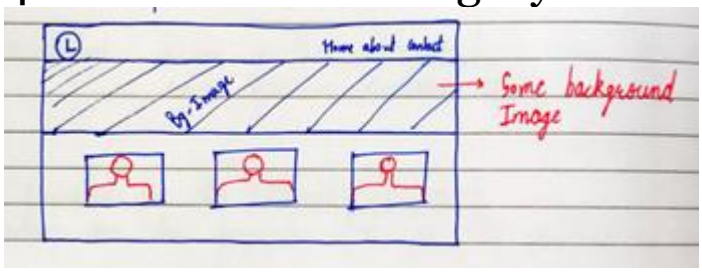
1. **order:** Controls the order in which the items appear in the flex container
2. **align-self:** Allows default alignment to be overridden for the individual flex items
3. **flex-grow:** Defines the ability for a flex item to grow
4. **flex-shrink:** Specifies how much a flex item will shrink relative to the rest of the flex items

Chapter – 6 (Practice Set)

1. Create a layout of your choice using float.
2. Create the same layout in question 1 using flexbox.
3. Create the following navigation bar using flexbox



4. Create the following layout using flexbox,



Chapter – 7 (CSS Grid & Media Queries)

A CSS grid can be initialized using:

```
.container {
        display: grid;
}
```

All direct children automatically become grid items

The grid-column-gap property
Used to adjust the space between the columns of a CSS grid

The grid-row-gap property

Used to adjust the space between the rows of a CSS grid

The grid-gap property
Shorthand property for grid-row-gap & grid-column-gap

```
.container {
        display: grid;
        grid-gap: 40px 100px;          /*40px for row and 100px for column*/
}
```

Note: For a single value of grid-gap, both row and column gaps can be set in one value.

Following are the properties for grid container:

1. The grid-template-columns property can be used to specify the width of columns

```
.container {
        display: grid;
        grid-template-columns: 80px 120px auto;
}
```

2. The grid-template-rows property can be used to specify the height of each row

```
.container {
        display: grid;
```

```
        grid-template-rows: 70px 150px;
}
```

3. The justify-content property is used to align the whole grid inside the container.
4. The align-content property is used to vertically align the whole grid inside the container.

Following are the properties for grid item:

1. The grid-column property defines how many columns an items will span.

```
.grid-item{
        grid-column: 1/5;
}
```

2. The grid-row property defines how many rows an item will span.
3. We can make an item to start on column 1 and space 3 columns like this:

```
.item{
        grid-column: 1/span 3;
}
```

CSS Media Queries

Used to apply CSS only when a certain condition is true.
Syntax:

```
@media only screen and (max-width: 800px) {
        body{
            background: red;
        }
    }
```

Chapter – 7 (Practice Set)

1. Create a header with content using CSS grid.
2. Create the layouts created in chapter-6 practice set using CSS grid.
3. Create a webpage that is green on large devices, red on medium, and yellow on small devices.

Chapter – 8 (Transforms, Transitions & Animations)

Transforms are used to rotate, move, skew or scale elements. They are used to create a 3-D effect.

The transform property

Used to apply a 2-D or 3-D transformation to an element

The transform-origin property

Allows to change the position of transformed elements
2D transforms – can change x & y-axis
3D transforms – can change Z-axis as well

CSS 2D transform methods

You can use the following 2-D transforms in CSS:

1. translate()
2. rotate()
3. scaleX()
4. scaleY()
5. skew()
6. matrix()
7. scale()

CSS 3D transform methods

1. rotateX()
2. rotateY()
3. rotateZ()

CSS Transitions

Used to change property values smoothly, over a given duration.

The transition property

The transition property is used to add a transition in CSS.
Following are the properties used for CSS transition:

1. **transition-property:** The property you want to transition
2. **transition-duration:** Time for which you want the transition to apply
3. **transition-timing-function:** How you want the property to transition
4. **transition-delay:** Specifies the delay for the transition

All these properties can be set using a single shorthand property
Syntax:

```
transition: property duration timing-function delay;
transition: width 35 ease-in 25;
```

Transitioning multiple properties
We can transition multiple properties as follows:

```
transition: opacity 15 ease-out 15, transform 25 ease-in;
```

CSS Animations
Used to animate CSS properties with more control.
We can use the @keyframes rule to change the animation from a given style to a new style.

```
@keyframes harry {
        from { width: 20px; }            /*Can change multiple properties*/
        to { width: 31px; }
}
```

Properties to add Animations

Following are the properties used to set animation in CSS:

1. **animation-name:** name of the animation
2. **animation-duration:** how long does the animation run?
3. **animation-timing-function:** determines speed curve of the animation
4. **animation-delay:** delay for the start of an animation
5. **animation-iteration-count:** number of times an animation should run
6. **animation-direction:** specifies the direction of the animation

Animation Shorthand

All the animation properties from 1-6 can be applied like this:

```
animation: harry 65 linear 15 infinite reverse;
```

Using percentage value states with animation

We can use % values to indicate what should happen when a certain percent of animation is completed

```
@keyframes harry {
        0% {
                width: 20px;
                }
        50% {
                width: 80px;
                }
        100% {
                width: 200px;
                }
        }
```

- Can add as many intermediate properties as possible

Chapter – 8 (Practice Set)

1. Create a thin progress bar animation for a website.
2. Create the same progress bar using transition.
3. Create a rotating image animation using CSS.
4. Create a slider with 3 images using CSS.