**CSS**

* Styles are additional attributes used to make HTML more interactive and responsive
* Styles provide a CSSOM to make elements effective. [Cascade Style Sheet Object Model]
* Styles can be defined in 3 ways
  + Inline Styles
  + Embedded Styles
  + External Style Sheet
* Inline Styles:
  + Styles are defined for every element individually by using "style" attribute.
  + It is faster in rendering
  + It difficult to re-use.
  + Syntax:

<h2 style = “attribute : value”><h2>

* Embedded Styles: -
  + Styles are defined in head or body section by using <style> tag.
  + You can re-use the styles.
  + Syntax :

<style>

h2{

}

</style>

**FAQ: What is difference between styles in <head> or <body>?**

Styles in head section are intended to load into browser memory first; from there you can apply to elements.

Styles in body section are intended to load and apply to elements directly.

**FAQ: Can we define styles in both sections?**

Ans: Yes.

**Note:** You can keep styles outside HTML.

**FAQ: What is the MIME type for Styles?**

Ans: **text/css**

syntax:

<style type="text/css"> </style>

**FAQ: What is the media type for styles?**

Ans: **screen, print, speech**

syntax:

<style type="text/css" media="print"> </style>

* You can't access embedded styles from another page.
* External Style Sheets [Cascading Style Sheets]
  + Styles are maintained in a separate stylesheet.
  + Clean separation of markup and styles.
  + Reusability across page.
  + Using an external style sheet will increase the number of requests for page.
  + If number of requests for page increases then page load time will increase.
  + Example

1. Create a new style sheet "effects.css"

h1{

background-color: tomato;

color:white;

text-align: center;

}

1. Link the style sheet to any page

<link rel="stylesheet" href="../src/styles/effects.css" type="text/css">

**FAQ: What is Minification?**

Ans: It is a technique used to **reduce the file size.** It is not ZIP or RAR file.

effects.css Development

effects.min.css Production

Ex:

1. visit [www.cssminifier.com](http://www.cssminifier.com).

2. paste your actual css code.

3. click "minify" button.

4. copy minified code.

5. add a new file into project "effects.min.css".

6. link minified file to your page.

**FAQ: Can we modify minified file directly?**

Ans : **Yes,** But you should not disturb the minified logic.

**FAQ: What is CDN?**

Ans: Content Distribution Network

We keep our resources in a server location.

We access and use in various projects.

We will not download the files, we will use direct server links.

It save the memory space for your project.

**Stylesheet Rules**

* Q1. If we define all 3 types of styles in a page, which one will work?

a) Inline

b) Embedded

c) External File

A. If you are writing same attributes in all 3 locations then the priority

1st Inline

2nd Embedded

3rd External

if you are writing different attributes in all 3 locations then all will apply.

* Q2. Can we link multiple stylesheet to a page?

1. Yes

* Q3. If all stylesheet have same effect then which one will apply?

A. The latest will apply.

<link href="a.css">

<link href="b.css"> // b effects will apply

If all stylesheets have different effects, then all will apply.

**Writing Styles for Element**

1. If you are writing styles inline

<h1 style="attribute:value; attribute:value"> </h1>

1. If you are writing styles embedded or external style sheet

selector{

attribute:value;

attribute:value;

}

**CSS Selectors**

1. Primary Selectors
   1. Type Selector
   2. ID Selector
   3. Class Selector
2. Type Selector:
   * It defines the element name, to which element you want the styles to apply.
   * You can’t ignore effects for any specific occurrence.

h2{ }

p { }

li {}

1. ID Selector :

* It uses an ID reference for element.
* Every element can refer to one ID only.
* You can’t apply multiple categories of styles.

<h2 id = “heading”></h2>

#heading{ }

<p id = “para”></p>

#para{ }

1. Class Selector:

* It uses class reference for elements.
* Class allows to implement multiple categories of styles

.className{ }

<div class = “class1 class2 class3”>

**Selector Rules**

**FAQ: If we are writing effects with all selectors which one will apply?**

Ans: If you are writing same attributes with all 3 selectors then

1st Priority ID

2nd Priority Class

3rd Priorty Type

If you are writing different attributes then all will apply.

**Grouping Primary Selector**

* You can group the selector using “,” separator

h1,p {} both heading and para

h2, #heading, .head { }

1. Rational Selectors and Combinators

* These selectors are used for elements having parent and child hierarchy or siblings.
  1. Child Selector parent child { }

It applies effect at any level of child.

* 1. Decendent Selector parent > child {}

It applies only to direct child.

* 1. Adjacent Sibling element + sibling { }

It applies effect to adjacent element.

* 1. General Sibling element sibling {}

It applies to all siblings

1. Dynamic Pseudo Classes

* Dynamic : It changes according to situation.
* Pseudo : It is not what it is showing.
* Class : It is a program template with data and logic.
* Inheritance : It is a technique of reusing and extending code. you inherit class by using ":" operator.

:hover specifies actions on mouse over

:link specifies effect for anchor in normal state

:visited specifies effect for anchor in visited state

:active specifies actions for anchor in active state

* You can control the hover timing by using “transition”.

element : hover{

effects;

Transition : 4s;

}

1. Element state pseudo classes

:disabled

:enabled

:read-only

:checked

1. Element Validation State Pseudo Classes

:required It verifies the required attribute set for input.

:optional It defines effects when required is not defined.

:valid It verifies any validation defined in input

:invalid It returns effects when any validation returns invalid.

:in-range It is for number validation [min]

: out-of-range It is for number validation [max]

**required and optional are used only to verify the attribute not validate.**

1. Attribute Selectors:

* It allows applying effects based on the element attribute instead of tag directly.
* Syntax :

element[attribute=value]{ .... }

element[attribute]{ .... }

* Attribute selectors have conditions

attribute="value" Equal to

attribute^="value" Starts with

attribute$="value" Ends with

attribute~="value" Any location with space

attributes="value" with - separator and start with.

1. Structural Selectors (Pseudo class]

* It depends on hierarchy parent-child
* It is to handle various occurrences in hierarchy.

:first-child

:last-child

:nth-child()

:nth-of-type()

:nth-last-of-type()

* Syntax:

:nth-child(2)

:nth-child(even)

:nth-child(odd)

:nth-of-type(2) only 2nd

nth-of-type(2n) for every 2nd occurance

:nth-of-type (2n+1) for every 2nd starting with 1st from top

:nth-last-of-type(2n+1) starting from bottom

1. Element Classes and Behavioural

* First : inheritance (logic)
* Second : implements(Rules)

::placeholder

::before

::after

::selection

1. Root Selector

:root{ .... }

1. Universal Selector

\*{ .... }

1. Language Selector

:lang { .... }

Synatax :

p:lang(en){

color : red;

}

**CSS Unit**

* CSS units are classified into 2 types
  + Absolute Units
  + Relative Units
* Absolute units are individual for element.
  + px pixels
  + % percentage
  + cm centimeter
  + mm milli meter
  + in inch
* Relative Units change according to parent.
  + em element
  + rem root element

Syntax:

parent{

font-size : 20px;

}

child{

font-size : 0.9em;

}

**CSS Inheritance**

* Inheritance is a mechanism of adapting the styles of parent.
* You can control inheritance by using
  + inherit
  + initial
  + unset

Syntax:

parent {

border:'';

padding:'';

color:'';

}

child{ child{ child{

border : inherit; all : inherit all : unset;

padding : inherit; } }

color : initial;

}

**CSS Colors**

* Color Name
* Hexadecimal code
  + #RGB (0,1,2,3,4,5,6,7,8,9, a,b,c,d,e,f)
  + #RRGGBB
  + rgb(r,g,b)
    - r = 0-255
    - g = 0-255
    - b = 0-255
  + rgba(r,g,b,a)
    - a = 0 to 1
  + hsl(hue, staturation, lightness)
  + hsla(h, s, I, alpha)

**CSS Attributes**

1. **Box Model Attributes**
   * 1. width
     2. height
     3. margin
        + margin : all direction
        + margin-left
        + margin-right
        + margin-top
        + margin-bottom
     4. padding
        + padding : all direction
        + padding-left
        + padding-right
        + padding-top
        + padding-bottom
     5. border
        + - border : all effects
          - border-style : dotted, solid, dashed
          - border-color : color name
          - border-size : size

syntax :

border : 2px solid red; (border shorthand operator)

border-left : 2px solid blue

border-right : ...

border-top : ....

border-bottom : ...

* border-image : You can set image as border

Rules:

* You have to define border
  + border : 20px solid transparent;
* You have to define suitable image
  + border-image: url()
* You have to defines display style
  + space
  + stretch
* You have to define outset percent
  + 1 to 100
* border-radius all direction
* border-top-left-radius
* border-bottom-left-radius
* border-top-right-radius

1. **Position Attributes**

* Allow to change the position of element in HTML page.
* "position" uses following values

1. static
2. absolute
3. relative
4. fixed
5. sticky
   1. Static Position

* It is the default position.
* It will keep element according to normal flow of document.
* It will not allow you to change position with left, right, top or bottom attributes.
* Syntax:

element {

position: static;

}

* 1. Absolute Position
* It removes element from normal flow of document.
* It can be controlled with left, right, top and bottom attributes.
* But it is fixed with the relative content.
* It will be scrollable along with the content.
* Syntax:

element {

position: absolute;

right: 10px;

top:20px;

}

* 1. Fixed Position
* It removes element from normal flow of document.
* It can be controlled with left, right, top and bottom.
* But it is fixed with relative to page not to the content.
* It will not scroll along with content.
* Syntax:

element {

position:fixed;

right:10px;

top:20px;

}

* 1. Sticky Position
* It will keep element according to normal flow of document.
* But it can fix relative to page after reaching specific position, which can be bottom, top, right, left.
* Syntax:

element {

position:sticky;

top:0px;

}

* 1. Relative Position:
* It keeps element according to normal flow of document.
* It can change the position only with relative to its parent.
* Relative position will change according to parent.
* Syntax:

element {

position:relative;

top:10px;

left:10px;

}

1. **Float Attributes**

* Float allows to keep any element to absolute left or right.
* Float will continue for next elements.
* You have to clear float for next element.
* Syntax

element {

float : left

}

nextElement {

clear: left, right, both;

}

1. **Display Attributes**

* They are used to control the display of elements in page.

display : none

display : block display : inline

display : grid display : flex

* none : To hide any element in page.
* block : To show element in block style. [below existing]
* inline : To show element beside existing
* grid:
* It allows to arrange elements into specified row or column.
* You can dynamically change the position.
* Display Grid uses
* grid-template-columns : 12 fractions maximum (1200 pixels maximum)

3fr

300px

* Display grid used following attributes to arrange into rows and columns

grid-row

grid-column

grid-gap

**FAQ: What is difference between Display : Grid and Columns?**

- Grid allows to arrange content into specified row and column.

- Columns will place continuous content.

* Display Flex
* Inline is used for individual elements
* Flex is used for container that contains elements.
* Flexible display
* It can Warp contents, set space between contents, align content to top, centre, bottom [vertically] or left, centre, right. [horizontally]
* Attributes:

display:flex

flex:wrap

flex-direction:row ]

flex-direction:row-reverse ] direction

flex-direction:column ]

flex-direciton:column-reverse ]

1. **Background Attributes**

* background : short hand for all options.
* background-color : It sets a color as background.
* background-image : It sets a background image

body {

background-image: url("path");

}

* background-size : width, height

contain - according content - fits to content

cover - fits to window.

* background-position : x-axis, y-axis

**Ex: Transparent Background**

<!DOCTYPE html>

<html>

<head>

<title>Background</title>

<style>

body {

background-image: url("../public/images/netflixback.jpg");

background-size: cover;

}

.body-cover {

width: 110%;

height: 700px;

background-color: rgba(0,0,0,0.6);

margin-top: -30px;

margin-left: -30px;

}

</style>

</head>

<body>

<div class="body-cover">

</div>

</body>

</html>

top-left, top-center, top-right

center-left, center-center, center-right

bottom-left, center-bottom, bottom-right

* background-attachment

: lock scrolling, scrolling, fixed

* background-repeat

: repeat, repeat-x, repeat-y, no-repeat

**FAQ: Can we make background image transparent?**

Ans: No. You can't set transparent image.

You have to configure background

with a cover that can be transparent using "rgba()".

**FAQ: Can we define multiple background colors?**

Ans : Yes. By using Gradient Colors

Background multiple colors are applied by using

"background-image"

Gradiants are 2 types

a) linear gradient horizontal and vertical

b) radial gradient from center

<style>

@media screen and (orientation:landscape){

body{

background-image: url("../public/images/netflixback.jpg");

background-size: cover;

}

}

@media screen and (orientation:portrait){

body{

background-image: url("../public/images/netflixportrait.jfif");

background-size: cover;

}

}

</style>

Syntax:

body {

background-image:

linear-gradient(to right, red, yellow);

}

**FAQ: Can we change background image**

**and color according to situation?**

Ans: Yes. By using "@media()" queries

**FAQ: Can we display multiple background images?**

Ans: Yes.

<style>

body {

background-image: url("../public/images/shoe.jpg"), url("../public/images/netflixback.jpg");

background-repeat: no-repeat, repeat;

background-size: 200px 200px, cover;

background-position: center center;

height: 500px;

}

</style>

1. **CSS Text Attributes**

* font-family : It defines the font name. sans-serif, serif, monospace
* font-weight : It defines bold
* font-style : It defines italic
* font-size : It defines char size.
* font-variant : It changes the capitalization
* text-decoration : It can set text decoration as underline, overline, line through etc..

text-decoration : style, color, size

**FAQ: How to remove underline for HyperLink?**

Ans : text-decoration:none

* text-align : Alignment left, center, right, justify
* color : Text color
* text-shadow : Sets shadow for text
* text-transform : Converts into lower case, uppercase, titlecase etc..
* word-spacing : sets space between words
* letter-spacing : sets space between letters
* line-height : sets space between lines
* word-break : It controls the larger words at right margin.
* white-space : It controls wrapping of text in container wrap, no-wrap.
* text-overflow : It is used to display overflowing text with clip or ellipsis.
* overflow : It controls overflowing content [auto, hidden, scroll]
* Text Direction – Orientation : Changes the orientation of text in container

text-orientation

writing-mode

**FAQ: How to design a drop cap?**

Ans: By using

Selector ::first-letter

Float : left

Font-size

1. **CSS List Attributes**

* List attributes are used for displaying ordered list and unordered list.
* list-style : short hand for list items none - to remove numbering or bullets.

Syntax:

ol, ul {

list-style: none;

}

* list-style-type : numbered, lower-alpha, disc, etc..
* list-style-image : custom image for list item.
* list-style-position : Inside or outside

1. **CSS Z-Index**

* z-index : It is used to bring forward or send backward.

Syntax:

z-index : 0 - bottom [behind]

1. **CSS 2D**

* 2D is 2 Dimensional Effects Along X and Y axis.
* It allows to change position, size, orientation of text.
* We have to use "transform" attribute for 2D.
* Transform attribute have browser dependency issues, hence we have to use with the help of browser plugins

transform : effect;

webkit-transform chrome, safari, edge, opera

moz-transform firefox

o-transform opera

ms-transform internet explorer

* CSS 2D Transforms are

translate()

scale()

rotate()

skew()

matrix()

* CSS Translate:

It is used to change the position of element in page.

You can move element along x and y axis.

translate(200px) x 200, y 200

translate(200px, 300px)

translateX(200px)

translateY(300px)

You can control the transform time by using "transition".

* Scale()

It is used to change the size of element

It uses the width and height

It used the value 1 = 100

scale() width and height

scaleX() width

scaleY() height

* Rotate()

It rotate element by specified angle. [value in deg]

You can give -ve angle for counter-clock-wise

roate()

rotateX()

rotateY()

* Skew()

It is used to tilt element by specified angle.

skew()

skewX()

skewY()

* matrix()

Can apply multiple transforms

matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())

Values can be in interger or fraction.

1 = 100

1.5 = 150

2 = 20deg [Skew]

2 = 200 [translate, scale]

1. **CSS 3D**

* You can transform in 3 dimensions using

translate3D()

translateX()

translateY()

translateZ()

scale3D()

rotate3D()

skew3D()

matriz3D()

1. **CSS Transition Attributes**

transition

transition-duration : total time interval

<html>

<head>

<title>Keyframes</title>

<style>

.container {

overflow: hidden;

}

@keyframes zoomin {

from {

font-size: 100px;

}

to {

font-size: 20px;

}

}

h2 {

animation-name: zoomin;

animation-duration: 2s;

animation-iteration-count: infinite;

animation-direction: alternate;

text-align: center;

}

</style>

</head>

<body>

<div class="container">

<h2>Welcome</h2>

<img src="../public/images/jacket.jpg"

width="100" height="100">

</div>

</body>

</html>

transition-delay : start time

transition-property : properties to used transition

transition-timing-function : ready made functions [animations]

1. **@keyframes**

* Animations are designed by using frames.
* In every frame object moves, resize or rotate etc.
* CSS uses keyframe with tweening animation.

Syntax:

@keyframes animationName

{

from { }

50% { }

70% { }

to { }

}

* To apply keyframes you need animation attributes

animation-name

animation-duration

animation-delay

animation-iteration-count

animation-direction

animation-timing-function

1. **Responsive Design**

* Page should fit according to screen and device.
* To create responsive design you need

a) Fluid Images

b) Fluid Containers

c) Display Grid | Flex | Flex Wrap | Flex Direction

d) Columns

e) Media Queries

Fluid Images: Size in percentage

<img width="40%" height="40%">

Fluid Container: Container width, height in percent

Position relative

Margins and padding in percent.