CSS Introduction

What is CSS?

* CSS stands for Cascading Style sheets
* CSS describes how HTML elements are to be displayed on screen.
* It can control layout of multiple pages all at ones.
* External style sheets are stored in css files.

Why use CSS ?

It is used to define styles of your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS Syntax

A css rule consists of a selector and a declaration block.

```
h1 {color : blue;}
```

selector — property — value

Selectors in CSS

A CSS selector selects the HTML element(s) you want to style.

1- Simple Selector

   1. Element Selector
   2. Class Selector
   3. ID Selector

2. Pseudo-class Selector

3. Multiple Selector

Element Selector

CSS element selector selects HTML elements based on the element name.

Ex-
```
p {
    text-align: center;
    color: red;
}
```

Class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period(.) character, followed by the class name.

Ex
```
.center {
    color: red;
}
```

ID Selector

⇒ The ID selector uses the id attribute of an HTML elements to select a specific element.

⇒ The id of an element is unique within a page, so the id selector is used to select one unique element!

⇒ To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Ex
```
#para1 {
    color: red;
}
```

Psuedo-classes Selector

A pseudo-class is a keyword added to a selector that specifies a special state of the selected element(s).

For example:

* Style an element when a user hovers the cursor over it.
* Style visited and unvisided links differently.

Syntax

```
selector: pseudo-class {
    property: value;
}
```

Multiple Selector / Grouping Selector

A grouping selector selects all the HTML elements with the same style definitions.

⇒ It will be better to group the selectors, to minimize the code.

⇒ To group selectors, seperate each selector with a (,) comma.

Ex
```
h1, h2, p{
    color: red;
}
```

Exploration Time (HW)

* Universal Selector
* Nested Selector
* Attribute Selector

Universal Selector

The universal selector (*) selects all HTML elements on the page.

Ex
```
* {
  color : red;
}
```

Nested Selector

Just like in HTML where you can have elements nested inside other elements, the same can be done in CSS.

Syntax

```
class1_sele class2_sele id_sele {
    property: value;
}
```

Ex
```
table tr th {
    background-color: red;
}
```

Attribute Selector

The [attribute] selector is used to select elements with a specified attribute.

Ex
```
a[target] {
    background-color: yellow;
}
```

How to add styling in HTML?

* Inline CSS
* Internal CSS
* External CSS

Inline CSS

⇒ To style an HTML element, you can add the style attribute directly to the opening tag.

⇒ To use inline styles, add the style attribute to the relevent ~~ems~~ element.

⇒ Inline styles should be avoided at all costs because it makes it impossible to alter styles from an external stylesheet.

Ex `<p style="color:red;">` I am Rishabh. `</p>`

Internal CSS / Style Tag

An internal style sheet may be used if one single HTML page has a unique style.

HTML allows us to write CSS code inside the `<style>` element, inside the head section.

Ex
```
<head>
    <style>
        h1 {
            color : red;
        }
```

```
</style>
</head>
```

External CSS

⇒ When the HTML & CSS code are in seperate files, the must be linked.

⇒ You can use the <link> element to link HTML and CSS files together. The <link> element must be placed within the head of the HTML file.

Ex `<link rel="stylesheet" href="style.css">`

Specificity

If there are two or more CSS rules that point to the same element, the selector with highest specificity value will "win", and its style declaration will be applied to that HTML element.

Specificity Hierarchy

Every css selector has its place in the specificity hierarchy.

There are four categories which define the specifity level of a selector:

* Inline styles
* IDs
* Classes, pseudo-classes, attribute selectors
* Elements and pseudo-elements

!important Rule

⇒ The !important rule in CSS is used to add more importance to a property/value than normal.

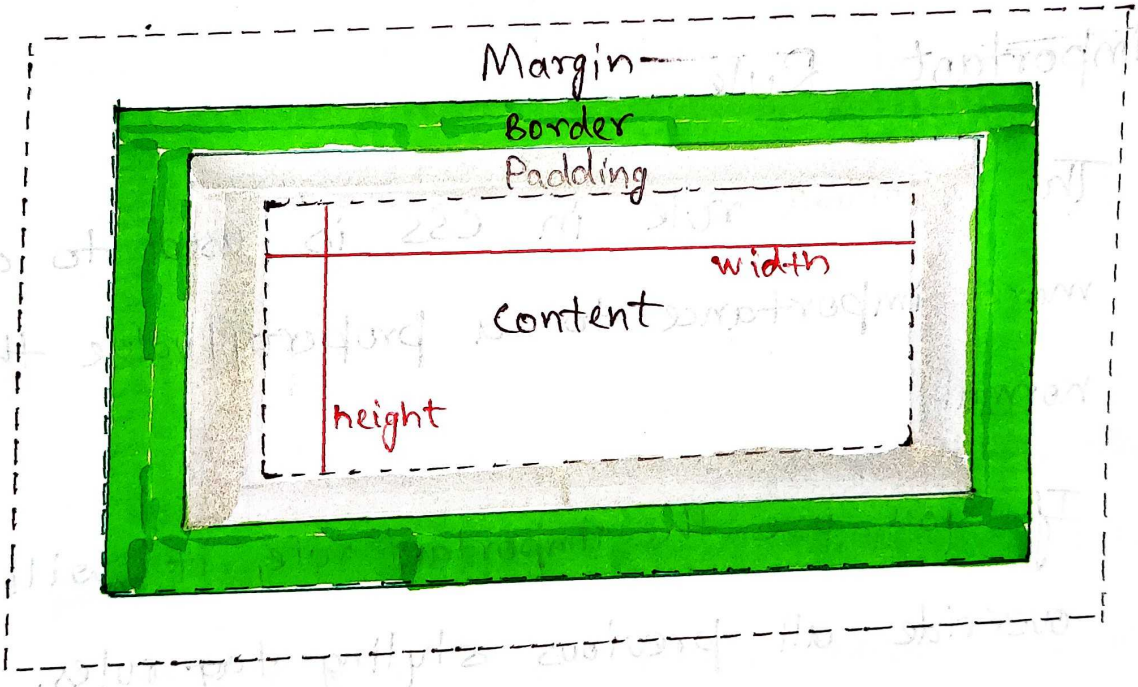
⇒ If you use the !important rule, it will override all previous styling ~~tag~~ rules.

⇒ It is good to know about the !important rule. However, do not use it unless you absolutely have to.

Ex
```
p {
  color: red !important;
}
```

Box Model in CSS

* The box model is the basic building block of CSS.
* According to the box model concept every element on a page is a rectangular box and may have width, height, padding, borders and margins.



Explaination

* Content - The content of the box, where text and images appear
* Padding - Clrears an area around the content. The padding is transparent.

* Border - A border that goes around the padding and content

* Margin - Clears an area outside the border. The margin is transparent.

Colors in CSS

Colors in CSS can be specified by the following methods:

* Hexadecimal colors
* RGB colors
* Predefined / Cross-browser color names
* RGBA colors
* HSL colors
* HSLA colors

Hexadecimal Colors

A hexadecimal color is specified with: #RRGGBB where RR(Red), GG(Green) and BB(Blue).

All values must be b/w 00 and FF, where 00 means lowest value and FF means highest value.

Black = #000000 White = #ffffff

RGB Colors

An RGB color values represent RED, GREEN and BLUE light sources.

rgb(red, green, blue)

Each parameter defines the intensity of the color and can be an integer b/w 0 and 255 or a percentage value from 0% to 100%.

Ex rgb(255, 99, 71)

Predefined / Cross-browser Color Names

140 color names are predefined in the HTML and CSS color specification.

RGBA Colors

It is an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

The alpha parameter is a number b/w 0.0 (fully transparent) and 1.0 (not transparent at all).

rgba(255, 99, 71, 0.8)

HSL colors

hsl (hue, saturation, lightness)

Hue is a degree on the color wheel from 0 to 360. 0 is Red, 120 is Green and 240 is blue.

Saturation is a percentage value. 0% means a shade of grey, and 100% is the full color.

Lightness is also a percentage. 0% is black, 50% is neither light or dark, 100% is white.

HSLA Colors

HSLA is an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

The alpha parameter is a number b/w 0.0 ~~to~~ and 1.0.

hsla (0, 100%, 50%, 1)

Explore Time

* Font:
    * Font-family
    * Font-weight
    * Font-style
    * Emphasis & Important
    * How to add external fonts?

Font

The word font refers to a set of printable or displayable typography or text characters in a specific style and size.

Font-family

A font family is a set of fonts that have a common design.

Font-weight

Weight is the overall thickness of a typeface's stroke in any given font.

Font-style

The font-style property is mostly used to specify italic text.

External Fonts

```
@font-face {
    font-family: myFont;
    src: url(sansation_light.woff);
}
```

Units in CSS

1- Absolute unit

2- Percentage unit

3- Relative unit

   1) Relative to font size

   2) Related to Document

Absolute Unit

* mm
* cm
* in
* px] fixed (1/96 inch)

Percentage Unit

* div { width: 10%; }

  => 10% of parent div

Relative Unit to Font size

* em => Related to parent element
* rem => Related to root element

Relative Unit to ViewPort

* vw => $\frac{1}{100}$ x width of viewport
* vh => $\frac{1}{100}$ x height of viewport

CSS Gradients

CSS gradients let you display smooth transition b/w two or more specified colors.

* CSS defines three types of gradients:
  - ⇒ Linear Gradients (goes down/up/left/right/diagonally)
  - ⇒ Radial Gradients (defined by their center)
  - ⇒ Conic Gradients (rotated around a center point)

Linear Gradient

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transition among.

You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax

```
background-image: linear-gradient(direction, color1,
color2, ---);
```

Directions:

* Top to Bottom (Default)

  Linear gradient that starts at the top and ends at the bottom.

* Left to Right

  It starts at the left and ends at the bottom.

* Diagonal

  You can make a gradient diagonally by specifying both the horizontal and vertical starting positions.

  It starts at the top left and goes to bottom right.

* Using Angles

  You can also define the angles, instead of predefined direction.

Syntax

```
background-image: linear-gradient(angle, color1, color2);
```

Radial Gradient

A radial gradient is defined by its center.

To create a radial gradient you must also define at least two color stops.

Syntax

```
background-image: radial-gradient(shape size at position,
    color1, color2 ---);
```

⇒ By default shape is ellipse.

Repeating a Radial Gradient

By using this you can also repeat the radial gradient.

Syntax

```
background-image: repeating-radial-gradient (color1, color2);
```

Conic Gradient [explore time]

A conic gradient is a gradient with color transition rotated around a center point.

To create a conic gradient you must define at least two colors.

Syntax

background-image: conic-gradient(color1, color2--);

⇒ By default, angle is ~~0g~~ 0deg and position is center.

⇒ If no degree is specified, the colors will be spread equally around the center point.

CSS Shadow Effects

With css you can add shadow to text and to elements.

* text-shadow
* box-shadow

Text shadow

The css text-shadow property applies shadow to text.

Syntax

text-shadow : ~~2px 2px~~ value1 value2 value3;

- value1 — horizontal shift
- value2 — Vertical shift
- value3 — Blur Effect

Ex
```
p {
    text-shadow: 2px 2px red;
}
```

Hello

⇒ By default shadow color is same as text color.

Box Shadow

The CSS box-shadow property is used to apply one or more shadow to an element.

Syntax

```
box-shadow: value1 value2 value3;
```

value1 — horizontal shift
value2 — vertical shift
value3 — Blur effect

⇒ By defaut shadow color is same as text color.

Q. How can we add border using shadows?

We can add border using shadows by putting horizontal shift value and vertical shift value as 1px.

⇒ Multiple shadows can be added using comma.

⇒ Color of the shadow can be changed.

⇒ Spread radius can be changed.

CSS Dimension Properties:

* Height

  The height property is used to set the height of the box.

* Width

  The width property is used to set the width of a box.

* Max-height

  It is used to set maximum height that can a box attain.

* Min-height

  It is used to set minimum height that a box can be.

* Max-width

  It is used to set maximum width that a box can be.

* Min-width

  It is used to set minimum width that a box can be.

NOTE - Do practical for better understanding.

Overflow Property

The overflow property specifies what should happen if content overflows an element's box.

This property specifies whether to clip content or to add scrollbar when an element's content is too big to fit in a specified area.

NOTE: The overflow property only works for block elements with a specified height.

Syntax

```
overflow: visible|hidden|clip|scroll|auto;
```

Values in Overflow Property

* Visible
* Hidden
* Clip
* Scroll
* Auto

CSS Position Property

The position property specifies the type of positioning method used for an element.

* Static (Default value)

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of page.

* Relative

An element with position: relative; is position relative to its normal position.

Setting the top, right, bottom and left properties of a relative-positioned element will cause it to be adjusted away from its normal position.

* Fixed

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.

The top, right, bottom and left properties are used to position the element.

* Absolute

An element with position: absolute; is positioned relative to the nearest positioned ascestor.

However, if an absolute positioned element has no positioned ancestors, it uses the document body, and move along with the page scroll.

NOTE: Absolute positioned elements are removed from the normal flow, and can overlap elements.

* Sticky

The element is positioned based on the user's scroll position.

A sticky element toggles b/w relative & fixed, depending on the scroll position.

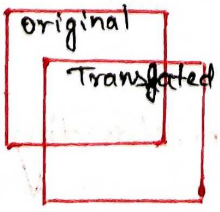
IMP ⇒ Do practical for better understanding.

2D Transform

CSS transform allow you to move, rotate, scale, and skew elements.

* With transform property you can use following methods.
  * translate()
  * rotate()
  * scaleX()
  * scaleY()
  * scale()
  * skewX()
  * skewY()
  * skew()
  * matrix()

* The translate() method

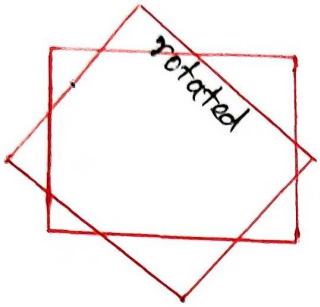


The translate() method moves an element from its current position according to its given parameters

Syntax

```
transform: translate(value1, value2);
```

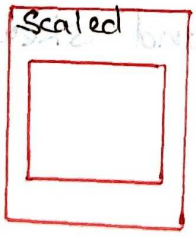
* The rotate() Method



The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

Ex
```
div{
  transform: rotate(20deg);
}
```

* The scale() Method



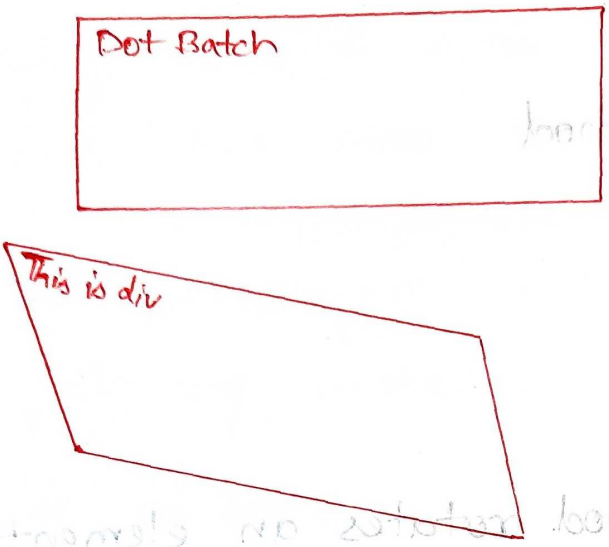
The scale() method increases or decreases the size of an element (according to the given parameters for the width and height).

Ex
```
div {
  transform: scale(2, 3);
}
```

scalex scaley

We can also use scalex and scaley seperately.

* The skew() Method



The skew() method skews an element along with x and Y-axis by the given angles.

Ex
```
div {
    transform : skew(20deg, 10deg);
}
```
skewX skewY

We can also use skewX and skewY seperately.

* The matrix() Method

The matrix() method combines all the 2D transform methods into one.

The parameters are as follows: matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())

```
div {
    transform : matrix(1, -0.3, 0, 1, 0, 0);
}
```

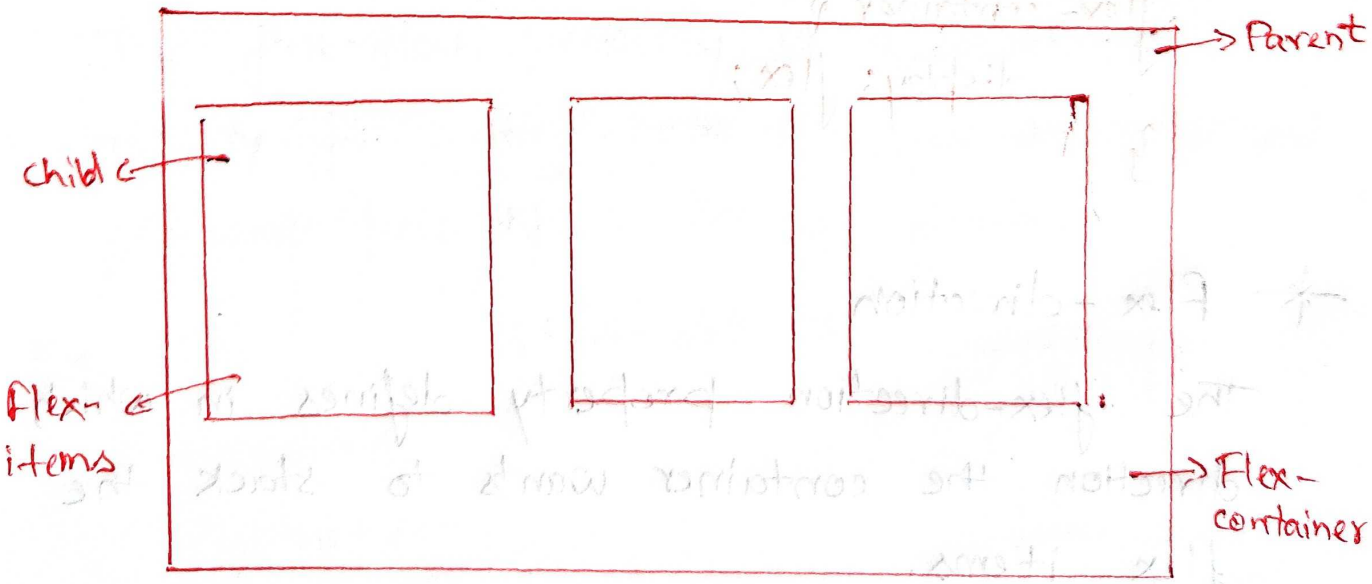
3D Transforms (Refer to any video explaination)
It works on the Z-axis like 2D transforms which works on X-axis and Yaxis. Not possible to explain in 2D Notebook.

CSS FlexBox

Flexbox is a great way to get more flexibility in your layouts and to simplify responsive layout design.

* The first step is to set display: flex on a "container" element. The "children" to the flex container becomes flex item.

  A set of properties can be applied to flex container, and have an effect to all the items as a whole. and a different set of properties can be applied to flex items and have their affect on the targeted items.



Flex-container Properties

Flex-container is a parent container.
The flex container becomes flexible by setting the display property to flex.

The flex container properties are:

* flex-direction
* flex-wrap
* flex-flow
* justify-content
* align-items
* align-content

Example

```
.flex-container {
    display: flex;
}
```

* Flex-direction

The flex-direction property defines in which direction the container wants to stack the flex items.

Ex

```
.flex-direction {
    display: flex;
    flex-direction: row| column| row-reverse| column-reverse;
}
```

* Flex-wrap

The flex-wrap property specifies whether the flex items should wrap or not.

Example

```
.flex-container {
    display: flex;
    flex-wrap: wrap | no-wrap | wrap-reverse;
}
```

Default

* Flex-flow

The flex-flow property is a shorthand property for setting both the flex-direction and flex-wrap property.

Ex

```
.flex-container {
    display: flex;
    flex-flow: row  wrap;
}
```

* Justify-content

Justify-content property is used to align the flex items.

Imp It is responsible for the alignment in horizontal axis.

Ex Center

The center value aligns the flex items at the center of the container.

```
.flex-container {
    display: flex;
    justify-content: center; }
```

Flex-start (Default)

The flex-start value aligns the flex items at the beginning of the ~~center~~ container.

```
justify-content: flex-start;
```

Flex-end

The flex-end value aligns the flex items at the end of the container.

justify-content: flex-end;

Space-around

* The space-around value displays the flex items with space before, b/w, and after the lines.
* The space b/w ~~with be~~ the boxes will be same, the space of the start and end will be same but the space b/w boxes and of start or end will not be same.

justify-content: space-around;

Space-between

* The space-between value displays the flex items with space between the boxes.
* Equal space distribution b/w the boxes will happen and start/end boxes will move to their extreme positions; This doesn't happen in space-around

justify-content: space-between;

Space-evenly

The space distribution at the start, between and end will be equal.

justify-content: space-evenly;

~~Baseline~~

* Align-items

The align-items property is used to align the flex items. Used for single row only.

Imp It is responsible for the alignment of the flex items in the vertical axis.

Ex Center

The center value aligns the flex items in the middle of the container.

```
.flex-container {
    display: flex;
    height: 200px;
    align-items: center;
}
```

Flex-start

The flex-start value aligns the flex items at the top of the container.

```
align-items: flex-start;
```

Flex-end

The flex-end value aligns the flex items at the bottom of the container.

```
align-items: flex-end;
```

Stretch (Default)

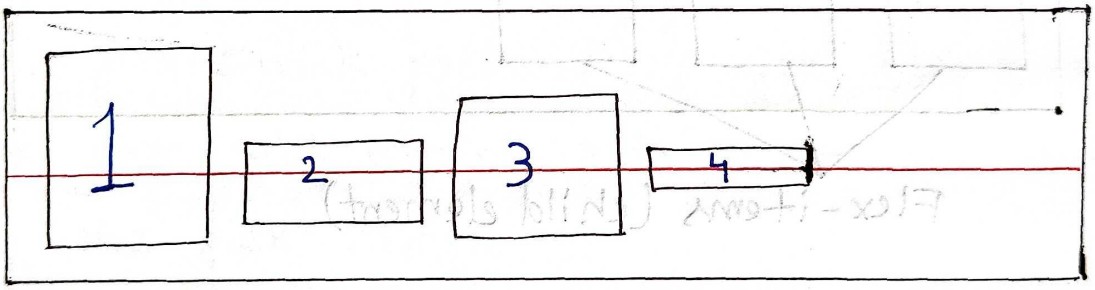
The stretch value stretches the flex items to fill the container.

```
align-items: stretch;
```

Baseline

The baseline value aligns the flex items such as their baselines aligns.

```
align-items : baseline;
```



* Align-content

It is same as align-items but it is used when our content is of more than one rows.

Properties

* space-between
* stretch
* Flex-start
* space-around
* Center
* Flex-end

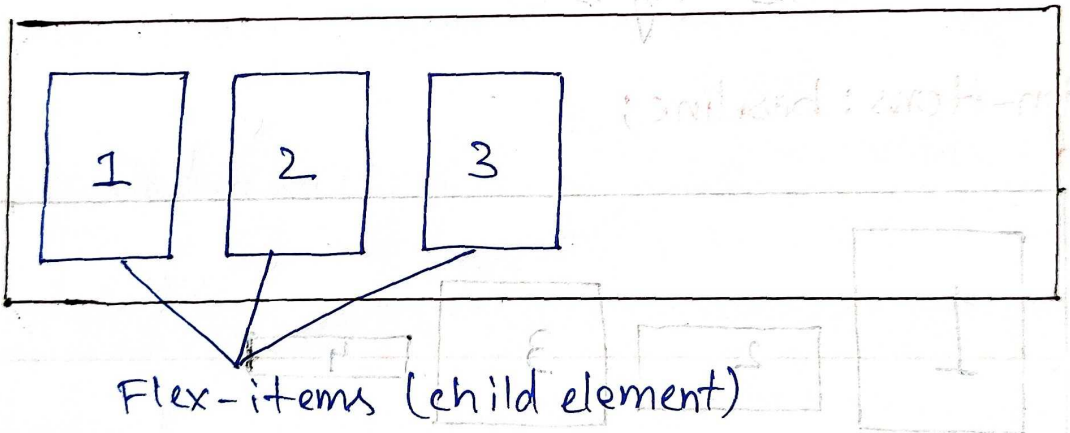
Q How to perfectly center a flex item / div?

Set both the justify-content and align-items properties to center, and the flex item / div will be perfectly center.

CSS Flex Items

Child Element (Items)

A direct child elements of a flex container autometically becomes flexible (flex) items.



The flex item properties are:

* order
* flex-grow
* flex-shrink
* flex-basis
* flex
* align-self

* Flex-grow

The flex-grow property specifies how much a flex item will grow relative to the rest of the flex items.

The value must be a number, default value is 0.

It can be applied to the individual flex-items.

Ex

```
<div style="flex-grow: 1"> 2 </div>
```

* Flex-shrink

The flex-shrink property specifies how much a flex item will shrink relative to the rest of the flex items.

The value must be a number, default value is 1.

Ex

```
<div> 1 </div>
<div style="flex-shrink : 0"> 3 </div>
```

* Flex-basis

The flex-basis property specifies the initial length of a flex item.

Ex 2
3

Q. Difference b/w Width and Flex-basis.

Width

* Visualy you won't notice any difference when the content is less.
* When the content is more than the width of the flex, then the additional content will be trimmed out.
* The overflow will be hidden.
* ~~Webpage~~ will be no longer responsive. That particular flex.

Flex-basis

* Visually looks same when content is less.
* If the content is more than the flex width, then the width will be increased.
* Content will not be trimmed out.
* That particular flex will be responsive.

* Flex

The flex properly is a shorthand property for the flex-grow, flex-shrink and flex-basis properties.

Syntax

```
flex : 0 0 200px;
```

flex-grow, flex-shrink, flex-basis

~~Self~~ Align-self

⇒ The align-self property specifies the alignment for the selected item inside the flex container.

⇒ The align-self property overrides the default alignment set by the container's align-items property.

Syntax

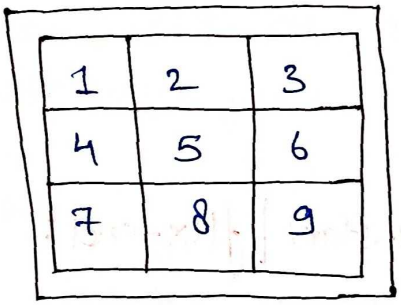
```
align-self : center | flex-start | flex-end;
```

CSS Grid

The CSS Grid Layout Module offers a grid-based layout system, with rows and colums, making it easier to design webpages without having to use floats and positioning.

Grid Elements

A grid layout consists of a parent element, with one or more child elements.

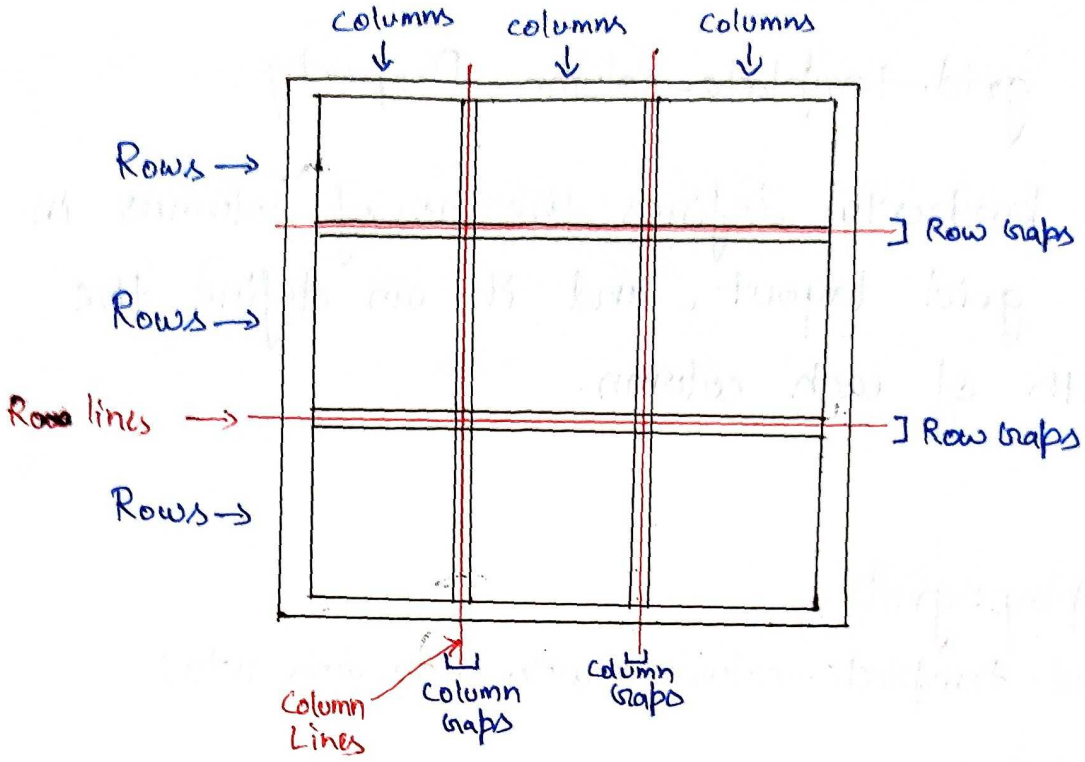


Display Property

An HTML element becomes a grid container when its display property is set to grid or inline-grid.

Ex

```
.grid-container {
    display: grid | inline-grid;
}
```



Examples

Column Gaps

```
column-gaps: 50px;
```

Row Gaps

```
row-gaps: 50px;
```

Shorthand property for gaps

```
gap: 50px 100px;
```

50px → row gap

100px → column gap

Grid Container

Grid container is a parent container that consists of grid items (child container) in row and column format.

* The grid-template-column Property

This property defines the no. of columns in your grid layout, and it can define the width of each column.

Ex

```
display : grid;
grid-template-column : auto auto auto auto;
```

* The grid-template-row Property

This property defines the height of each row.

Ex

```
display : grid;
grid-template-rows : 80px 200px;
```

* The justify-content Property

* space-evenly
* space-around
* space-between
* center
* start
* end

* The align-content Property

* space-evenly
* space-around
* space-between
* center
* start
* end

Grid Item

* grid-column Property

The grid-column property defines on which column(s) to place an item. It is a shorthand property.

Ex => Make "item1" start on column 1 and end before column 5.

```
.item1 {
    grid-column: 1 / 5;
}
```

=> Make grid "item1" start on column 1 and span 3 columns:

```
.item1 {
    grid-column: 1 / span 3;
}
```

* grid-row Property

This property defines on which row to place an item. It is a shorthand property.

Ex Make "item1" start on row-line 1 and end on row-line 4;

```
.item1{
    grid-row: 1 / 4;
}
```

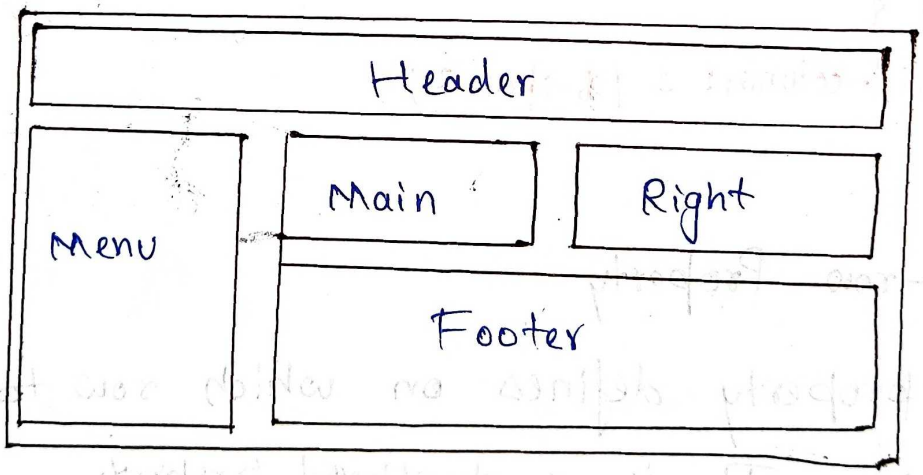
* grid-area Property

This property can be used as a shorthand property for the grid-row-start, grid-column-start, grid-row-end and the grid-column-end property.

Ex Make "item8" start on row-line 1 and column-line 2, and end on row-line 5 and column-line 6:

```
.item8 {
  grid-area: 1 / 2 / 5 / 6;
}
```

* Naming Grid Items



Named grid items can be referred to by the grid-template-areas property of the grid container.

Ex- .grid-template-areas: 'myArea myArea myArea myArea myArea';

Each row is defined by apostrophes (').