# Web Technology Notes

---

## Introduction

**Web:**

Web is a collection of information. i.e in the form of text / web pages / websites.

**Webpage:**

It is a simple document written in html / saved with html extension.

**Website:**

Website is a collection of webpages.

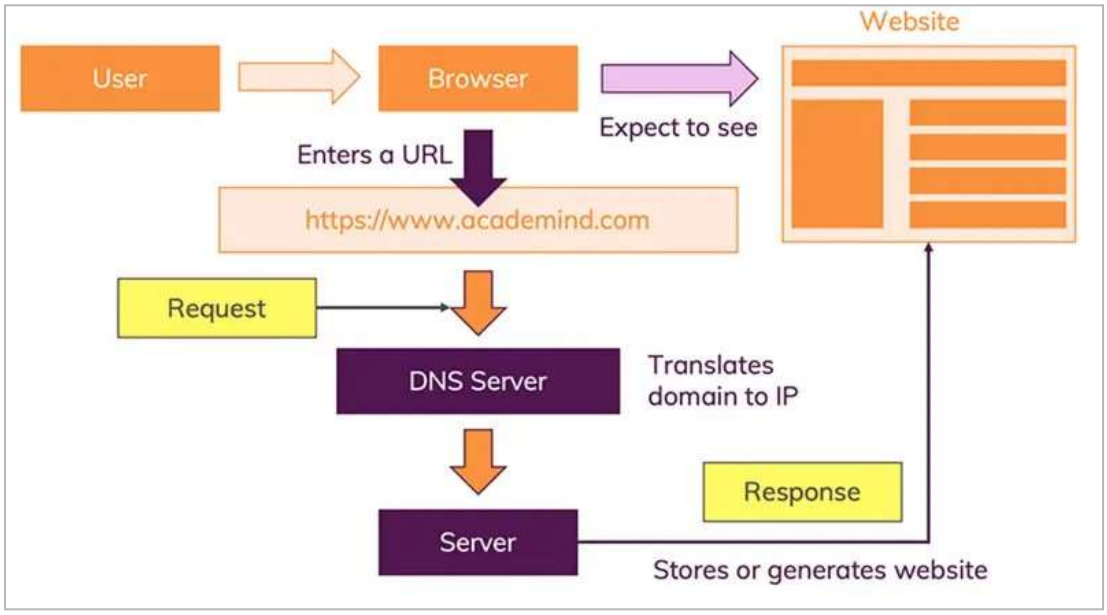
**Network:**

Two or more computers that are connected with one another for the purpose of communicating data electronically.

**Internet:**

Internet is a global network, we are connecting through www.

**How web works:**



**Browser:**

Browser is a client side application which is used to send requests and get back the responses from the server.

**URL:**

URL stands for uniform resource locator.

When we search for anything in the browser, It will generate an URL.

Ex: https://www.instagram.com

**DNS:**

DNS stands for Domain Name Server

It is responsible for converting URLs into IP addresses.

**Server:**

It is the place where all the websites are hoisted.

Ip address of all data were stored here.

**Http:**

Http stands for HyperText Transfer Protocol.

It transfers the information as plain text.

**Https:**

Https stands for HyperText Transfer Protocol Secure.

It transfers the information in encrypted format.

**Static Webpage:**

These are the webpages which are common for everyone

Or

These are the webpages which will display the same information for all users.

**Ex:** wikipedia, javaTpoint, tutorialspoint

**Dynamic Webpage:**

These web pages display different information for users.

**Ex:** Instagram, youtube, linkedIn

**Single Page Application:**

These websites consist of only one web page.

All operations will be performed in only one page.

Most of the single page applications are dynamic web pages.

**Multi Page Application:**

These websites consist of many web pages.

All web pages linked together.

Most of the multi page applications are static web pages.

---

# HTML

## HTML:

Html stands for Hyper Text Markup Language.
It helps us to create web pages.
The current version of html is html5.

## Structure of HTML:

```
<!DOCTYPE html>
<html>
      <head>
            <title>Document</title>
      </head>
      <body>

      </body>
</html>
```

**Doctype html:**

It is used to declare the version of html as html5.

**Html:**

It is the root element which contains the head section and body section.

**Head:**

It holds the meta data.

**Title:**

It will provide the title to the webpage.

**Body:**

This is the place where we have to write all the code which has to be displayed on UI.

## Tags

Tags are the predefined keywords which are enclosed in angular braces.
The task of each tag has been already declared.

**Syntax:**

<tagname> content </tagname>

**Ex:**

 Hello world 
 Hello world 

**Types of Tags:**

HTML tags were categorized into 2 ways.
They are:

1. Paired Tags
2. UnPaired Tags

**Paired Tags:**

A tag which is required of both opening and closing tags are called paired tags.
Ex:  Hello Web Dev 

**Unpaired Tags:**

A Tag which does not require a closing tag is called unpaired tag.
Ex: <br> , <hr> , <input>

## Heading Tags

Heading Tags are basically used to provide title or subtitle to the content in the webpage.
We have a set of heading tags from h1 to h6.
Where h1 is highest in font size and h6 is lowest in font size.

**Default Sizes:**

h1 → 2 em

h2 → 1.5 em

h3 → 1.17 em

h4 → 1 em

h5 → 0.83 em

h6 → 0.67 em

**Note:** 1 em = 16px

## Formatting Tags

Formatting Tags are used to display the content in different formats.

Formatting Tags are:

- **b** → It will display the content in bold format.
- **strong** → It will display the content in bold format and the browser will understand that information is very important.
- **i** → It will display the content in Italic format.
- **em** → It will display the content in Italic format and it is an alternative to the <i> tag.
- **u** → It will provide an underline to the content.
- **ins** → It will provide an underline to the content and it is an alternative to <u> tag.
- **strike** → It will strike off the content.
- **del** → It will strike off the content and it is the alternative to <strike> tag.
- **q** → It will provide quotations to the content.
- **mark** → It will highlight the content with yellow background color.
- **big** → It will display the content with a big font-size.
- **small** → It will display the content with a small font-size
- **sup** → It will display the content as superscript.
- **sub** → It will display the content as subscript.
- **code** → It will display in the font-family "monospace".
- **pre** → It will consider space also. It will also be known as a pre structured tag.

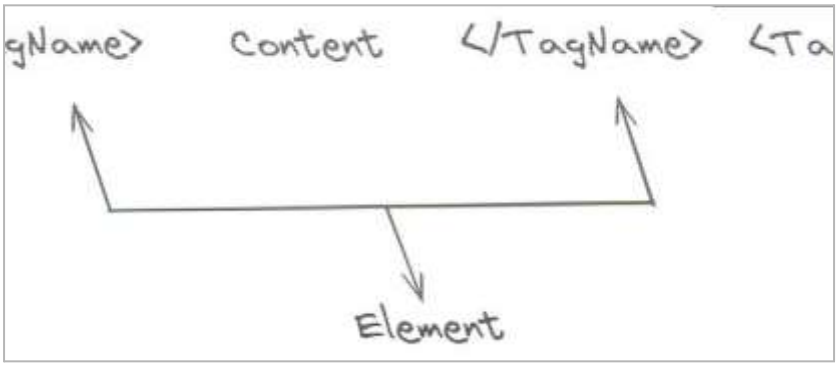
## Deprecated Tags:

These are the tags which will be removed in further versions.

Vs code gives suggestions by displaying in red color.

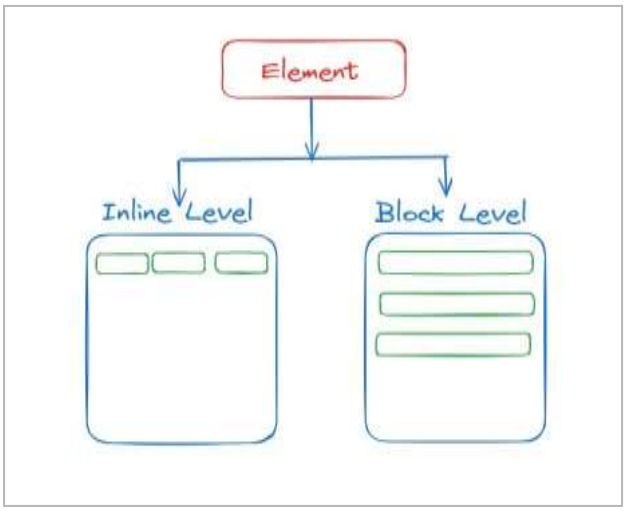
## Element

An element is a combination of opening tag + content + closing tag.




Html Elements were classified into 2 ways

1. Inline Level
2. Block Level




**Inline Level:**

- These elements will occupy only content space.
- These elements will display in the same line.
- We cannot assign height and width properties for the inline elements.
- Ex: <b> , <i> ,  , etc…

**Block Level:**

- These elements will occupy the whole viewport width.
- These Elements will display in the next line.

- We can assign height and width properties for the block elements.
- Ex: <h1> ,  , <p> , etc…

## Attributes

Attributes are also known as properties.

It will provide additional information to the element.

Ex: style , src, href , id , class , etc …

Attributes are case sensitive.

## Image Tag

Img tag helps us to use images in web pages.

Img tag is a self closing tag.

It is an inline-block element.

**Syntax:** <img src="path address" alt=" ">

**Attributes of image Tag:**

- Src → It will specify the path address of an image
- Alt → It will provide an alternate name to the image.
- Height → It helps to modify the height of an image.
- Width → It helps to modify the Width of an image.

## Anchor Tag

Anchor tag helps us to navigate from one web page to another webpage.

By using anchor tags, we can create hyperlinks in a webpage.

Syntax:

<a href="pathAddress"> content </a>

Whatever we give in the content space that becomes a hyperlink.

Content can be text, image, button, icon, etc.

To mail a person we have to use mailto:

To call a person we have to use tel:

**Attributes:**

1. href → It will specify the path address to where we have to navigate.
2. download → It indicates to download the file.

Default colors:

1. Visited link → purple.
2. Unvisited link → blue.
3. Active link → red.

## Marquee Tag

Marquee tag implements scrollable content in our webpages.

**Syntax:**

```
<marquee behavior="" direction=""> Content </marquee>
```

Whatever we provide in the content place will scroll, content like text / images / anchor tag / icons / buttons etc…

**Attributes:**

- direction → It specifies the direction of scrolling content.

  Values = left (default) / right / up / down
- Behavior → It specifies the behavior of marquee element.

  Values = scroll (default) / slide / alternate
- Scrollamount → It specifies the speed of an element

  Values = Numbers (default is 6)
- Loop → It specifies how many times the content has to scroll.
- Height → It Specifies the height of an element.
- Width → It Specifies the width of an element.

## Lists

HTML Lists allows to group a set of related items together.

HTML lists were classified into 3 ways.

They are:

1. Ordered List
2. Unordered List
3. Description List

### 1. Ordered List:

- Ordered list group a set of related items in a sequential format.
- Ordered List is also known as Number List.
- **Tags:**
  - ol → It indicates that we are creating an ordered List.
  - li → It indicates List Items.
- **Attributes:**
  - Type → Specifies the type of sequence.

    Values = 1 (default) , A , a , I , i
  - Start → Specifies the Starting Position.

    Values = 1 (default) , Any Number
  - Reversed → It will reverse the sequence.

### 2. Unordered List:

- Unordered List is used to display a set of related items with special symbols.
- Unordered List is also known as Bulleted List.
- **Tags:**
  - ul → It indicates that we are creating an Unordered List.
  - li → It indicates List Items.
- **Attributes:**
  - Type → Specifies the type of symbol.

    Values = disc (default) , circle , square , none

### 3. Description List:

- Description List is used to display the description data along with description terms.
- **Tags:**
  - dl → Indicates Description List
  - dt → Indicates Description term
  - dd → Indicates Description Definition.

## Tables

HTML Tables allow us to store the data in the form of rows and columns.

### Tags:

- **table** → It defines the creation of a table.
- **tr** → It defines table rows.
- **td** → It defines table columns.
- **th** → It defines table heading. (bold & aligns to center)
- **thead** → It combines all the header cells together.
- **tbody** → It combines all the body cells together.
- **tfoot** → It combines all the footer cells together.
- **caption** → It provides a title to the table tag.

### Attributes:

- **border & cellspacing & cellpadding:**
  - These 3 attributes we have to use in a **table** tag.
  - By default, the table will not display the border.
  - To get the border to the table, we have to use **border** attribute.
  - **Cellspacing** → It will provide the space between the cells.
  - **Cellpadding** → It will provide the space between the content and cell border.
- **rowspan & colspan:**
  - These 2 attributes we have to use in **th** and **td** tags.

- **Rowspan** → It will merge the cells over the rows.
- **Colspan** → It will merge the cells over the columns.

## Forms

Forms are used to collect user information.

Form is a block level element and most of the form elements are inline level.

Syntax: <form action=""> </form>

### Tags of Form:

1. form
2. label
3. input
4. select
5. optgroup
6. option
7. datalist
8. fieldset
9. legend
10. textarea
11. button

### Form:

Form is a semantic element which indicates the of it.

It will contain all the form elements.

Ex:

```
<form>
    .
    .
    form elements
    .
    .
```

```
    </form>
```

**Label:**

It specifies the purpose of the input field.

**Input:**

It helps to receive the user input.

To link label tag and input tag we have to use **for** and **id** attributes.

Ex: example for both label and input

```
<label for="userName">Name:</label>
<input type="text" id="userName">

<label for="userEmail">Email:</label>
<input type="email" id="userEmail">
```

**Select:**

It will create a drop down list with a set of options.

By using the select tag, we can choose options.

**Optgroup:**

It is used to group the set of options together.

**Option:**

It used to define the options in the drop down list.

Ex:

```
<label>Skills:</label>

<select name="userSkills" multiple>
    <optgroup label="frontend">
        <option value="React">React</option>
        <option value="Angular">Angular</option>
    </optgroup>
```

```
    <optgroup label="database">
        <option value="SQL">SQL</option>
        <option value="MongoDB">MongoDB</option>
    </optgroup>

    <optgroup label="backend">
        <option value="Python">Python</option>
        <option value="Java">Java</option>
        <option value="NodeJs">NodeJs</option>
    </optgroup>
</select>
```

## Datalist:

Datalist is also used to create a dropdown list just like select tag.

But, here we can sort the options based on initial letters.

Here input tag is mandatory.

Value attribute and content should be the same.

Ex:

```
<label>Skills:</label>
<input list="skillsList">
<datalist id="skillsList">
        <option value="React">React</option>
        <option value="Angular">Angular</option>
        <option value="SQL">SQL</option>
        <option value="MongoDB">MongoDB</option>
        <option value="Python">Python</option>
        <option value="Java">Java</option>
        <option value="NodeJs">NodeJs</option>
</datalist>
```

## Fieldset:

This tag is used to group the set of fields together that will display a border to those fields.

## Legend:

It will provide a name to the fieldset border.

Ex:

```
<fieldset>
    <legend>Personal Details</legend>
    .
    .
    form elements
    .
    .
</fieldset>
```

## Textarea:

It is used to take multiline input based on rows and cols.

rows and cols attributes specify the visible height and width of textarea.

Ex:

```
<textarea name="comment" rows="10" cols="30"></textarea>
```

## Button:

It will create a button based on type attribute.

Type attribute will specify the purpose of the button.

Ex:

```
<button type="reset">Reset</button>
<button type="submit">Submit</button>
```

## Attribute of Input Tag:

1. type
2. id
3. name
4. value
5. list
6. multiple

7. readonly
8. disabled
9. required
10. placeholder
11. min
12. max
13. minlength
14. maxlength
15. step
16. size
17. autofocus
18. autocomplete
19. pattern

**Values of type attribute:**

1. text
2. email
3. password
4. number
5. tel
6. radio
7. checkbox
8. button
9. submit
10. reset
11. search
12. date
13. time

14. datetime-local
15. week
16. month
17. color
18. image
19. file
20. url
21. range

## Attributes of form tag:

1. **Action:**

   It specifies where the data has to be stored.

2. **Method:**

   It specifies the way of receiving the data.

   We can assign the values as get and post.

   Get:

   It receives the on the url.

   Data is not secure.

   Post:

   It sends the data directly to the server side.

   Data is secure.

3. **Novalidate:**

   By default, form will validate the data before submitting.

   If we want to submit without checking then we can use novalidate.

4. **Auto complete:**

   It will accept the values as on / off.

   If we pass "ON", it will show suggestions.

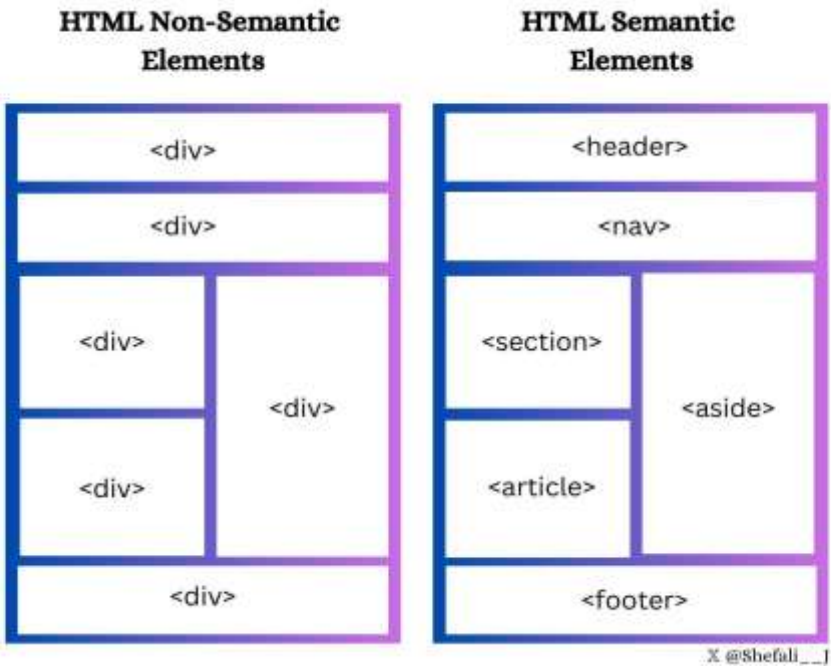
   Else it will not show any suggestions.

## Semantic Tags

Semantic tags are the tags which will say the purpose of it.

For example, header , footer , nav , section , etc..

Semantic tags were introduced in the HTML5 version.




Example:

Header , nav , section , article , aside , main , table , form , audio , video , etc..

## Audio Tag

It helps to implement audible content in a web page.

Syntax: <audio src="path address"> </audio>

**Attributes:**

1. **src** → It will specify the path address
2. **controls** → It will provide the access to control the audio file.
3. **autoplay** → It will play the song automatically

4. **loop** → It will repeat the song
5. **muted** → It will mute the song.

## Video Tag

It helps to implement video content in a web page.

Syntax: <video src="path address"> </video>

**Attributes:**

1. **src** → It will specify the path address
2. **controls** → It will provide the access to control the audio file.
3. **autoplay** → It will play the song automatically
4. **loop** → It will repeat the song
5. **muted** → It will mute the song.
6. **poster** → It will provide a thumbnail to the video.
7. **height** → It will specify the height of the poster & video.
8. **width** → It will specify the width of the poster & video.

## Iframe Tag

Iframe tag is used to embed another web page into the current web page.

We can embed from internal web pages as well as external web pages like youtube , google maps etc…

**Attributes:**

1. **src** → It will specify the path address.
2. **frameborder** → It will specify iframe should have a border or not.
3. **height** → It will specify the height of the iframe tag.
4. **width** → It will specify the width of the iframe tag.

---

# CSS

## Introduction

- CSS stands for cascading style sheets.
- The current version of css is CSS3.
- CSS was introduced by Hakon Wium Lie in 1996.

### CSS Rule:

```
selector{
    propertyName1: value1;
    propertyName2: value2;
    .
    .
    .
    propertyNameN: valueN;
}
```

- Each property consists of a key & value pair separated by colon.
- Each property is separated by a semicolon.

## Types of CSS

We can achieve css in 3 ways .
They are:

1. Inline CSS
2. Internal CSS
3. External CSS

### 1. Inline CSS:

It is a way of applying the CSS in the same line.
By using style **attribute**.
**Ex:**

```
<h1 style="color: white;background-color: blue;">Let's
```

```
dive into CSS</h1>
```

## 2. Internal CSS:

It is  a way of adding the CSS in the same html file itself.

By using Style **Tag**.

We can use style tag anywhere in the html file. It's recommended to use in a head tag.

**Ex:**

```
<html>
  <head>
    <title>Document</title>
    <style>
      h1 {
        color: white;
        background-color: blue;
      }
    </style>
  </head>
  <body>
    <h1>Let's dive into CSS</h1>
  </body>
</html>
```

## 3. External CSS:

It is a way of writing all the css code in one file which is saved with .css extension.

That css file we have to link to the html file by using the link tag.

```
<!-- index.html -->
<html>
  <head>
    <title>Document</title>
    <link rel="stylesheet" href="./style.css">
  </head>
  <body>
```

```
    <h1>Let's dive into CSS</h1>
  </body>
</html>
```

```
/* style.css */
h1 {
  color: white;
  background-color: blue;
}
```

Note:

1. The first will always be taken by inline CSS. because of the most recent update.
2. Internal vs External depends on code flow which is written at last.

## CSS Selectors

To apply the css Properties, we can target the html elements in many ways.
They are:

1. Simple Selector
2. Combinator Selector
3. Pseudo Class Selector
4. Pseudo Element Selector
5. Attribute Selector

| Selectors | | | | |
|---|---|---|---|---|
| Simple | Combinator | Pseudo Class | Pseudo Element | Attribute |
| ~~TagName~~ | ~~Descendent( )~~ | ~~:hover~~ | ~~::before~~ | ~~Tag[attribute]~~ |
| Grouping (,) | Direct Child(>) | :first-child | ::after | Tag[attribute="value"] |
| Universal (*) | Adjacent | :last-child | ::first-letter | |
| Id Name (#) | Sibling(+) | :nth-child() | ::first-line | |
| Class Name (.) | General SIbling(~) | :first-of-type | ::selection | |
| | | :last-of-type | ::marker | |
| | | :nth-of-type() | | |
| | | :checked | | |

## 1. Simple Selectors:

To target the element based on classes / id's / tagname we have basic selectors. They are:

1. TagName
2. Grouping
3. Universal
4. Id Name
5. Class Name

- **TagName:**
  - To target the element based on tagname itself we have to use tagName selector.
  - The symbol was the tagname itself.
- **Grouping:**
  - To target multiple elements at a time we have to use a grouping selector.
  - Whenever we need to pass similar properties for multiple elements we can use a grouping selector.
  - The symbol used to combine all elements is comma (,)
- **Universal:**
  - It will target all the elements in the document including body tag too.
  - The symbol used is asterisk (*).
- **Id Name:**
  - To target the elements uniquely we have to use an id name.
  - In CSS id name can be duplicated also there is no problem,
  - But once we moved to advanced languages.
  - Repetition of id name will not work. So it is highly recommended not to use it from now on.
  - The Symbol used is hash (#).
- **Class Name:**
  - To target the specific elements on the basis of class name we have to use class name.
  - Class attribute allows multiple identifier names in the same attribute.
  - Class names can be repetitive also.
  - The symbol used is dot (.).

## 2. Combinator Selectors:

It is a combination of 2 simple selectors.

Based on the relation b/w 2 elements the css will target the elements.

They are:

1. Descendent Selector
2. Direct Child Selector
3. Adjacent Sibling Selector
4. General Sibling Selector

- **Descendent Selector:**
  - It will target all the children, grandchildren , great grandchildren and so on.
  - The symbol used is space ( ).
- **Direct Child Selector:**
  - It will target only the children but not grandchildren , great grandchildren and so on.
  - The symbol used is greater than ( > ).
- **Adjacent Sibling Selector:**
  - It will target only the first sibling of the element.
  - The symbol used is plus ( + ).
- **General Sibling Selector:**
  - It will target all the first siblings of the element.
  - The symbol used is plus ( + ).

## 3. Pseudo Class Selectors

These are the special selectors.

The symbol used is single solon (:).

They are:

- **:hover** → It will apply the properties whenever we move the cursor on the element.
- **:first-child** → It will target only the first child.
- **:last-child** → It will target only the last child.
- **:nth-child (even / odd / 2n / 2n+1 )** → It will target the specific child based on the parameter.
- **:first-of-type** → It will target the first child of each type
- **:last-of-type** → It will target the last child of each type

- **:nth-of-type()** → It will target the nth child of each type.
- **:checked** → It will apply the css properties once after checked.

**4. Pseudo Element Selectors** These are the special selectors. The symbol used is single solon (:).

They are:

- **::first-letter** → It will target only the first letter of content.
- **::first-line** → It will target only the first line of content.
- **::before** → It will add the content before the element.
- **::after** → It will add the content before the element.
- **::selection** → while selecting the content by default it will show color: "white" & background-color: "blue". To change the default behaviour we can use this.
- **::marker** → It will target the type attribute of lists.

**5. Attribute Selectors**

It will target the elements based on attribute name.

Instead of creating new id names and class names we can use existing attributes.

1. TagName[attribute]
2. TagName[attribute="Value"]

## Text Properties

Text Properties are basically used to apply CSS effects on text content.

Text properties are:

1. color
2. text-align
3. text-transform
4. text-decoration
5. text-indent
6. text-wrap
7. letter-spacing
8. word-spacing
9. line-height
10. text-shadow

### 1. color:

It will change the color of the text.

**Syntax:** color : value;

**Values** = Any Color.

### 2. text-align:

It is used to align the content based on the value.

**Syntax:** text-align : value;

**Values** = left / start , center , right / end , justify

### 3. text-transform:

It will convert the text into other formats.

**Syntax:** text-transform : value;

**Values** = lowercase , uppercase , capitalize.

### 4. text-decoration:

It will decorate the text by providing underlines or overlines with special styles.

It is a shorthand property of text-decoration-line , text-decoration-style & text-decoration-color.

**Syntax:** text-decoration: text-decoration-line text-decoration-style text-decoration-color

**Values**=

text-decoration-line ⇒ underline , overline , line-through

text-decoration-style ⇒ solid , dashed , dotted , double , wavy.

text-decoration-color ⇒ any color.

**Note**: out of 3 properties text-decoration-line is mandatory.

### 5. text-indent:

It will provide the space at the initial position of the content.

**Syntax:** text-indent : value;

**Values** = any number.

### 6. text-wrap:

It will indicate whether to wrap the content into the next line or not.

**Syntax:** text-wrap : value;

**Values** = wrap , nowrap.

### 7. letter-spacing;

It will specify the space between the letters.

**Syntax:** text-spacing : value;

**Values** = any number.

### 8. word-spacing:

It will specify the space between the words.

**Syntax:** text-spacing : value;

**Values** = any number.

### 9. line-height:

It will specify the height of the line.

**Syntax:** line-height : value;

**Values** = any number.

### 10. text-shadow:

It will specify the shadow for text content.

**Syntax:** text-shadow : x-axis y-axis blur color;

Default values: text-shadow : 0px 0px 0px black;

**Ex**: text-shadow : 2px 2px 3px blue;

## Font Properties

Font properties are used to apply CSS effects of size , boldness , styles , etc.

Font properties are:

1. font-size
2. font-weight
3. font-style
4. font-family
5. font-variant

### 1. font-size:

It will specify the size of text content.

**Syntax:** font-size: value;

**Values** = xx-small , x-small , small , medium , large , x-large , xx-large , any number.

### 2. font-weight:

It will specify the boldness of the content.

**Syntax:** font-weight: value;

**Values** = 100 - 900 , lighter , normal , bold / bolder.

### 3. font-style:

It will specify the style of text like it should be italic or normal.

**Syntax:** font-style : value;

**Values** = normal , italic / oblique.

### 4. font-family:

It will specify the font family of text content.

**Syntax:** font-family : value;

**Values** = 1. From vs code 2. Google fonts.

### 5. font-variant:

It will be displayed in upper camel case formats.

**Syntax:** font-variant : value;

**Values** = normal , small-caps

## Background Properties

Background properties are used to apply the background effects to an element's background.

Background is a shorthand property of background-color , background-image , background-repeat , background-position , background-attachment.

### 1. background:

It is a shorthand property of multiple background properties.

Instead of writing all properties we can pass all values over here.

### 2. background-color:

It will specify the background color of an element.

**Syntax:** background-color : value;

**Values** = Any color.

### 3. background-image:

It will specify the background image of an element.

**Syntax:** background-image : value;

**Values** = url("pathaddress").

### 4. background-repeat:

By default,the background image will repeat.

To prevent repetition of images we can use this property.

**Syntax:** background-repeat: value;

**Values** = repeat , no-repeat , repeat-x , repeat-y

### 5. background-position:

It will specify the position of the background image.

It is a shorthand property of background-position-x and background-position-y.

**Syntax:** background-position : background-position-x background-position-y

**Values** = background-position-x ⇒ left , center , right

Background-position-y ⇒ top , center , bottom

### 6. background-size:

It will specify how to fit the image into a container.

**Syntax:** background-size : value;

**Values** = auto , contain , cover.

### 7. background-attachment:

It will decide the behavior of the background image. That has to scroll or fixed.

**Syntax:** background-attachment : values;

**Values** = scroll , fixed.

## Border Properties

Border properties are used to apply the CSS effects to the border of an element.

### 1. border:

It is a shorthand property of border-width , border-style and border-color.

Syntax: border: border-width border-style border-color

**2. border-width:**

It will specify the width of the border.

Syntax: border-width: any number;

**3. border-style:**

It will specify the style of the border.

Syntax: border-style:Values;

Values: solid , dashed , dotted , double , groove , ridge , none.

**4. border-color:**

It will specify the color of the border.

Syntax: border-style: any color;

**5. border-top/bottom/left/right:**

If we need to specify the border only at one side we can use border - top / border - bottom / border - left / border-right.

**6. border-collapse:**

This is a property to apply effects on tables.

It will decide whether the table cells have to separate or not.

Syntax: border-collapse: values;

Values: separate , collapse.

**7. border-spacing:**

It is an alternative to the cellspacing attribute of a table tag.

It will specify the space b/w cells in a table.

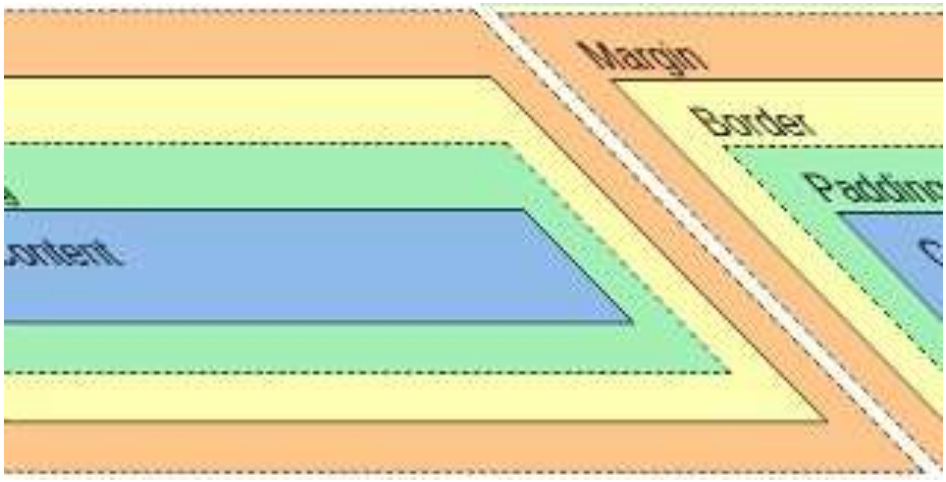
Syntax: border-spacing : any number;

## Box Model

By default, each and every element has default padding and margin.

These default values will disturb our websites when we are layouting the webpages.

To come out of these problems mainly we use this box model concept.




This model contains mainly 4 boxes.

1. Content box
2. Padding box
3. Border box
4. Margin box

**1. Content:**

It specifies the content area of an element.

It includes content height and width (not element height & width).

**2. Padding:**

It specifies the space between border and content of an element.

Ex: padding: 5px;

padding: 10px;

**3. Border:**

It specifies the border to an element.

Ex: border : 2px solid blue;

**4. Margin:**

It specifies the space around the element.

Ex: margin: 5px;

margin: 10px;

**Note:** margin is a shorthand property of margin-top , margin-bottom , margin-left , margin-right.

We can pass

**Margin : 10px 20px 30px 40px;**
T R B L

**Margin : 10px 20px 30px;**
T L/R B

**Margin : 10px 20px;**
T/B L/R

**Margin : 10px;**
T/B/L/R

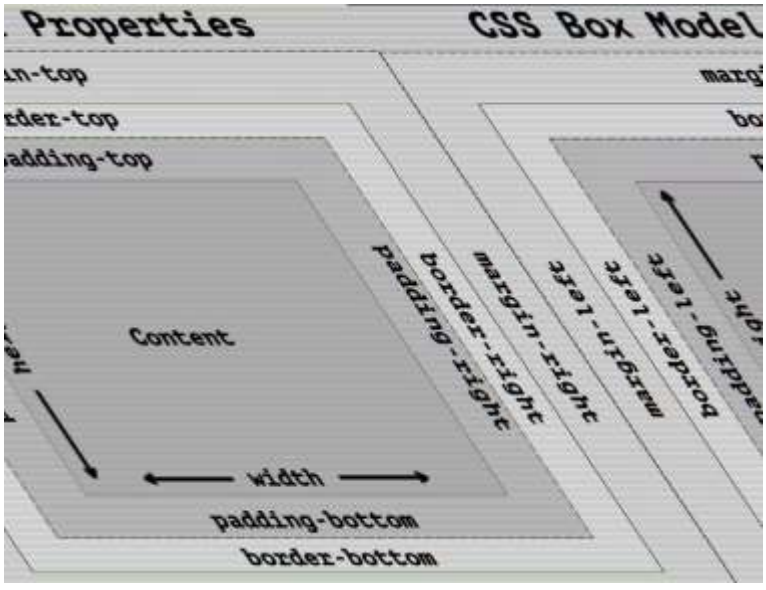
Same as for padding property also.

**Height Calculation:**

Total height of element = margin-top + border-top + padding-top + content-height + padding-bottom + border-bottom + margin-bottom

**Width Calculation:**

Total width of element = margin-left + border-left + padding-left + content-width + padding-right + border-right + margin-right



The main point over here is that content height is different from the rest of properties.

For example if i specified height: 100px; padding:10px; so the total height will be 120px.

But if we need fix the size of the element we have to use **box-sizing:border;**

And To remove the default margin and padding of each element we have to use **margin:0; padding:0;**

So we have to use

```
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

**Box-shadow:**

It will specify the shadow of an element.

Syntax: box-shadow: x-axis y-axis blur spread color inset;

Ex: box-shadow: 2px 2px 2px 3px blue;

Ex: box-shadow: 2px 2px 2px 3px blue inset;

## Positions

Positions are used to arrange the elements in a web page.

It will change the position based on position behavior.

They are:

1. Static
2. Relative
3. Absolute
4. Fixed
5. Sticky

To deal with positions we have to use helping properties.

They are:

1. top
2. bottom
3. left

4. right

**1. Static:**

a. It is the default value.
b. It will not perform any changes to the element.
c. Helping properties was not required.

**2. Relative:**

a. It will change the position of an element with respect to the original position.
b. It will leave an empty space. It doesn't allow other elements to occupy its place.

**3. Absolute:**

a. It will change the position of an element with respect to the parent.
b. By default the parent is a viewport, if we need to change we have to pass **position:relative** for parent element.
c. It will **not** leave any empty space. It **allows** other elements to occupy its place.
d. If we provide position:absolute for all the elements , all elements will float on each other.
e. To specify which element has to float top or bottom , we have to use the **z-index** property.

**4. Fixed:**

a. It will fix the position of an element.
b. To fix the position we have to mention the helping properties.

**5. Sticky:**

a. Initially these elements will scroll until it reaches to some point.
b. Then it will stick at that position.

**Note:**

If we specify the position as relative / absolute / fixed / sticky elements will float on the z-axis.
All of the above need the help properties.
We should not use top and bottom / left and right at a time.

## Display

Display property allows to change the behavior of an element.

It accepts the values like

- None
- Inline
- Block
- Inline-block
- Flex
- Grid

**None:**

- This will not display anything in the browser.
- As well as it will not occupy any space for the element.

**Inline:**

- It will convert the element into inline-level.
- So then the element will occupy only content height and width.
- We cannot assign height & width properties.

**Block:**

- It will convert the elements into block level.
- So then the element will occupy viewport width.
- We can assign height & width properties.

**Inline-Block:**

- It will convert the elements into inline-block level.
- So then it will have features of both inline & block level.
- Because of inline → It will display in the same line.
- Because of block → we can assign height & width properties.

**Flex:**

- It will layouts the webpages in one dimensional format.

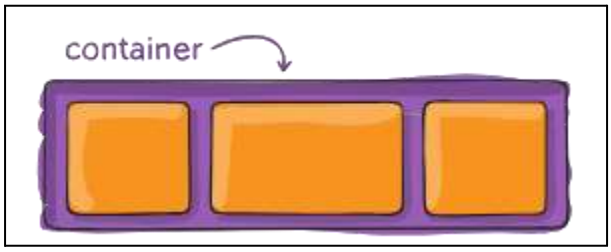
**Grid:**

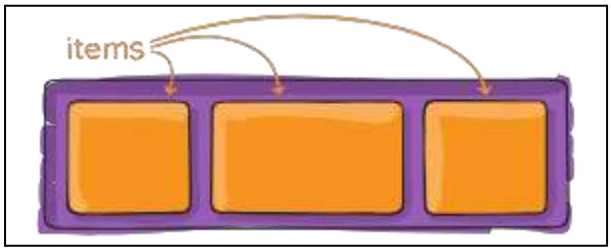
- It will layouts the webpages in two dimensional format.


## Flex

Flex will layout the webpages in one dimensional format.

To work with flex boxes, We need a flex container and all the direct children will become the flex-items. (ONLY DIRECT CHILDREN).




There are many properties which we can apply only to the container as well as only to the items.

**Flex-container Properties** are:

1. display:flex
2. justify-content
3. align-items
4. flex-direction
5. flex-wrap
6. align-content
7. column-gap
8. row-gap
9. gap

### 1. display: flex:

This is the main property we must use to work with flex boxes.

By assigning this property we are able to use all flex properties.

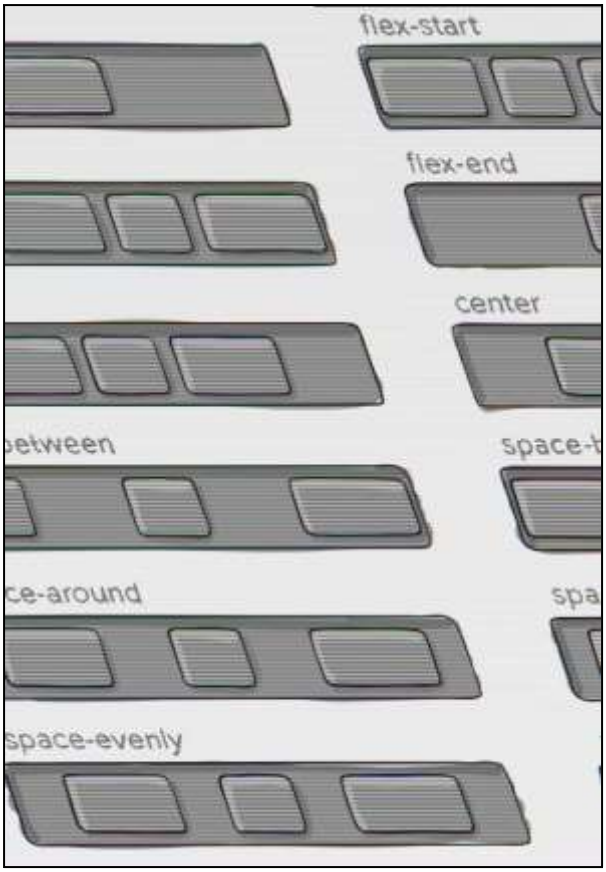
### 2. Justify-content:

It will align the flex-items on the main axis.

**Syntax:** justify-content: values;

**Values** = start , center , end , space-between , space-evenly , space-around.

The default value is start.



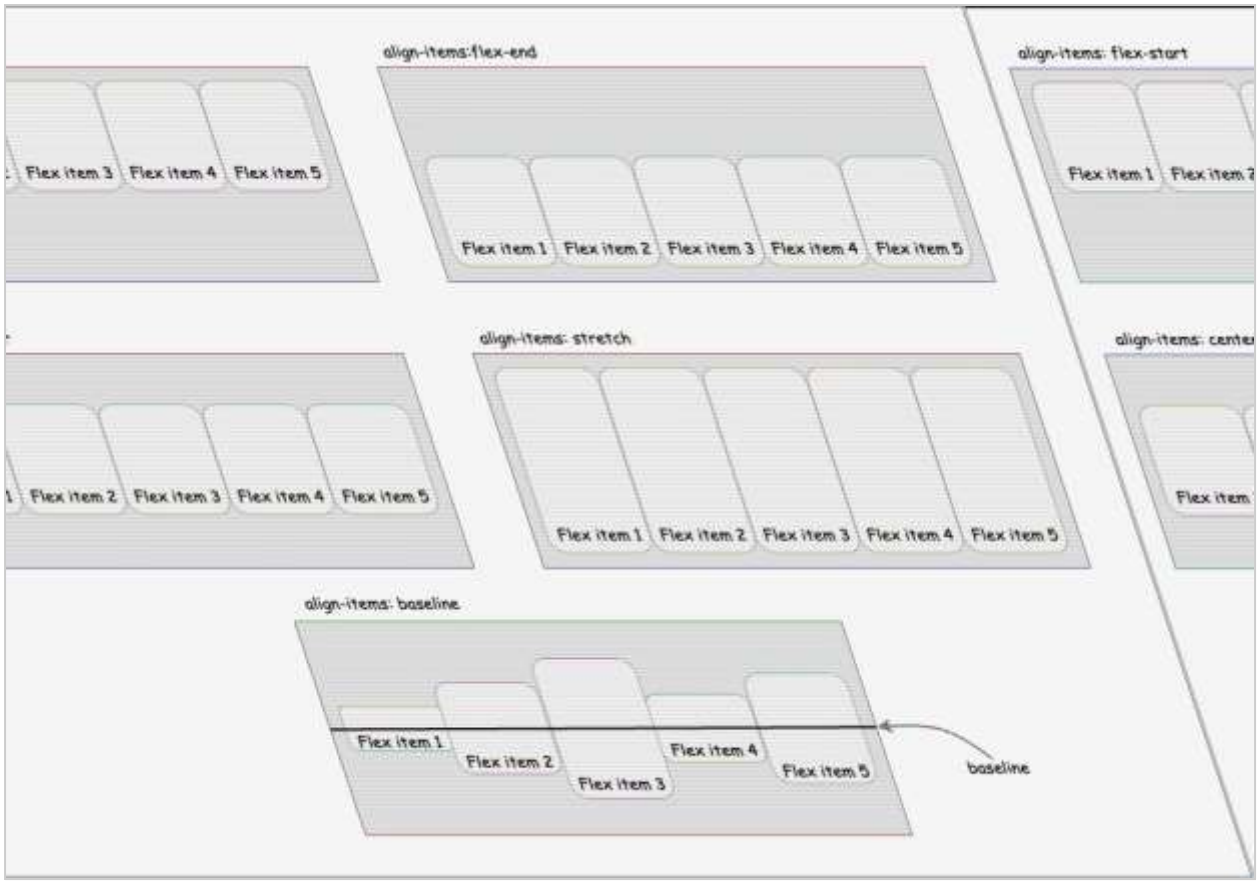

### 3. align-Items:

It will align the flex-items on the cross axis.

**Syntax**: align-items: values;

**Values** = start , center , end , stretch , baseline.

The default value is start.




## 4. flex-direction:

It will specify the direction of flex-items.

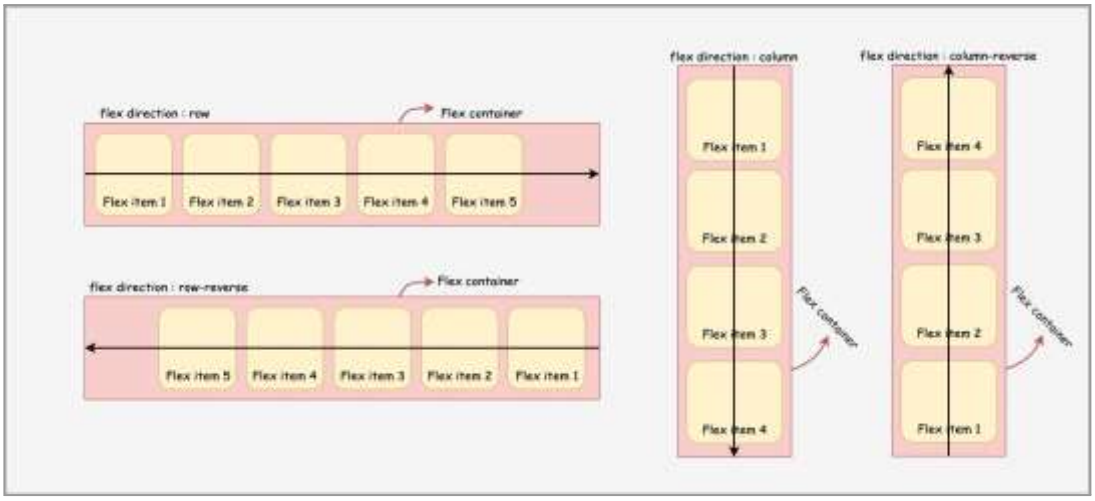
Whether to display in row format or column format.

It will decide the main axis.

**Syntax**: flex-direction: values;

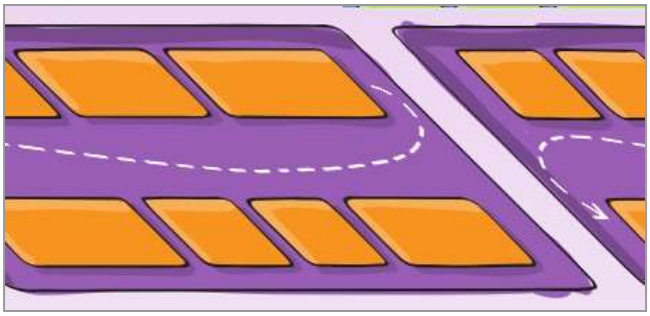
**Values** = row , column , row-reverse , column-reverse.

The default value is row.




## 5. flex-wrap:

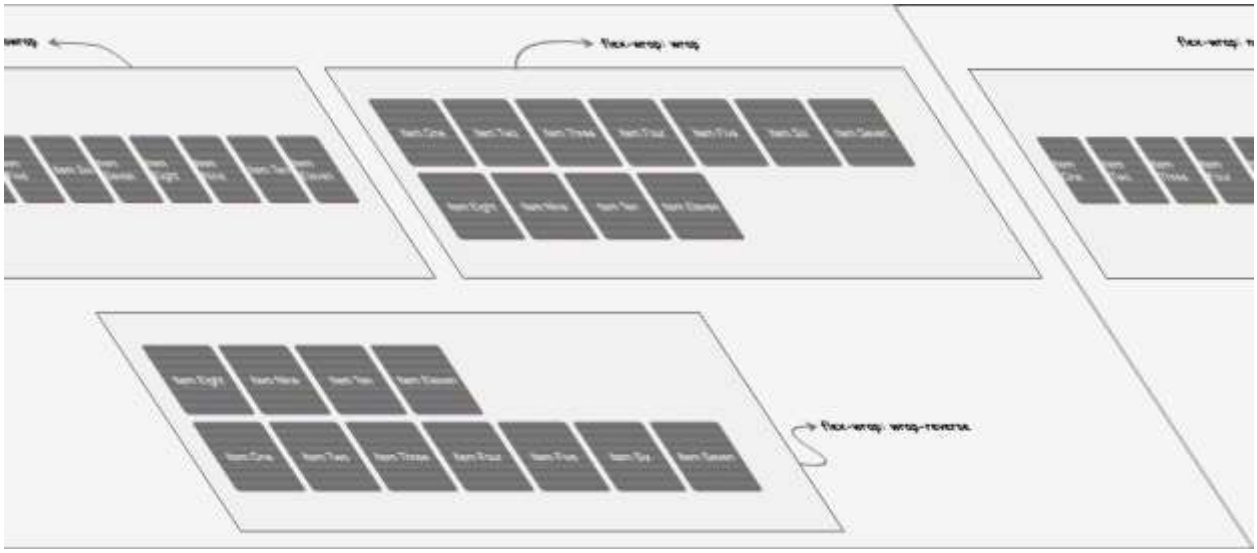
It will specify whether to wrap the flex-items into the next line or not.



**Syntax**: flex-wrap: values;

**Values** = nowrap , wrap , wrap-reverse.

The default value is nowrap.



## 6. align-content:

It will specify the position of wrapped flex items.

**Syntax**: align-content: values;

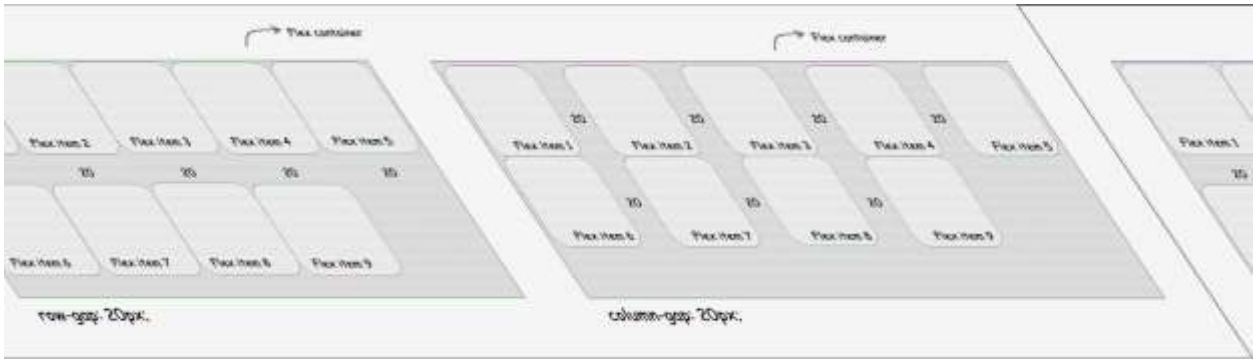
**Values** = start , center , end , space-between , space-around , stretch

## 7. gap:

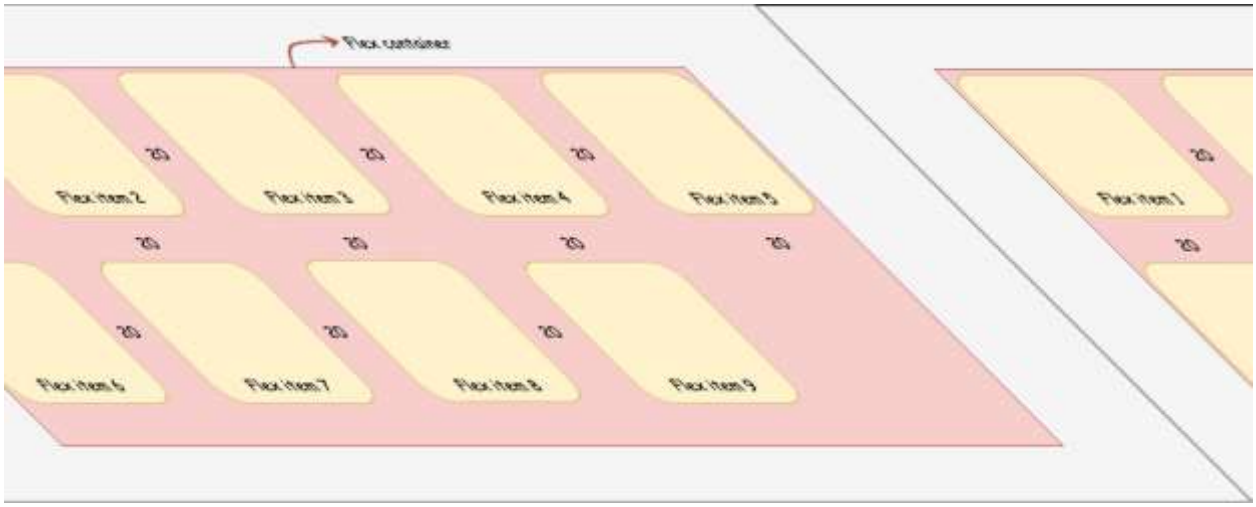
It is a shorthand property of row-gap & column-gap.

It will specify the space between the flex items.

Ex: row-gap:20px & column-gap:20px;  / gap: 20px 20px;



Ex: gap:20px;



**Flex-items Properties** are:

1. order
2. align-self
3. flex-grow
4. flex-basis
5. flex-shrink
6. flex

### 1. Order:

It will arrange the flex-items in an order.

Ex:

```
.item1 {
    order: 4;
}
.item2 {
    order: 1;
}
.item3 {
    order: 2;
}
.item4 {
    order: 3;
}
```


## 2. align-self:

By using this property, we can align the flex-items individually.

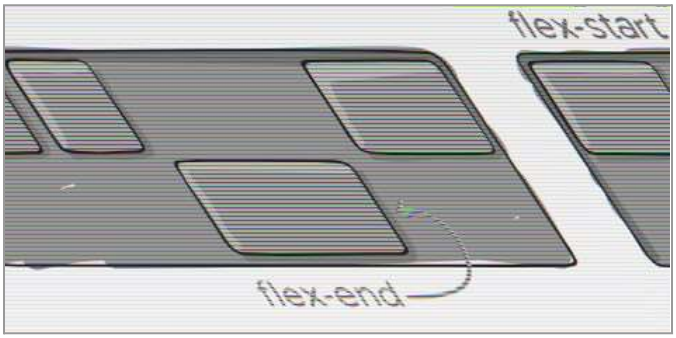
It will override the default alignment which is specified by align-items.

It will accept all the values which can be acceptable by align-items property.

The only difference is

i. **align-items** for **container** for all flex-items.

ii. **align-self** for **items** for individual flex-item.




## 3. Flex-grow:

It will make sure the flex-items have to fit to the flex-container.

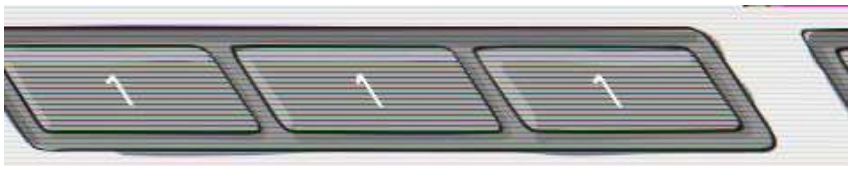
It will increase the size of the flex-items based on remaining space.

The remaining space in a container will be divided into units and it will be added to each flex-item based on flex-grow values.

Ex:

```
.item1 {
    flex-grow: 0;
}
.item2 {
    flex-grow: 0;
}
.item3 {
    flex-grow: 0;
}
```

```
.item1 {
    flex-grow: 1;
}
.item2 {
    flex-grow: 1;
}
.item3 {
    flex-grow: 1;
}
```




```
.item1 {
    flex-grow: 1;
}
.item2 {
    flex-grow: 2;
}
.item3 {
    flex-grow: 1;
}
```


## 4. flex-basis

It is equivalent to the width property.

Ex:

```
.item1 {
```

```
    width: 200px;
}
.item2 {
    width: 100px;
}
.item3 {
    width: 200px;
}
```

```
.item1 {
    flex-basis: 200px;
}
.item2 {
    flex-basis: 100px;
}
.item3 {
    flex-basis: 200px;
}
```

## 5. flex-shrink:

It will shrink the size of the flex-items.

Based on the extra space required in a container to fit all the flex-items.

If there is no enough space in a container to fit all the flex-items then all flex-items will shrink(decreases) equally and fit's in a container.

**Ex:**

For example:
The width of container is 800px;
And with provided to items is 300px;
So to fit 300 + 300 + 300 = 900 px into an 800px container all items will

shrink equally.


But our need is that we may shrink item2 and item3 but item1 must take 300px.
So, we target item2 & item3 and specify the flex-shrink value.


### 6. flex:

It is a shorthand property of flex-grow , flex-shrink and flex-basis.
Ex: flex: 1
If we specify flex 1 , it will increase , decrease , and width equally.

## Grid

Grid will layout the webpages in 2 dimensional format.

To work with a grid, We need a grid container and all the direct children will become grid-items. (ONLY DIRECT CHILDREN).

**Grid Container Properties** are:

1. display:grid
2. grid-template-columns
3. grid-template-rows
4. column-gap
5. row-gap
6. gap

### 1. display:grid

a. It specifies the container as a grid container.
b. So that we can use all the grid properties.

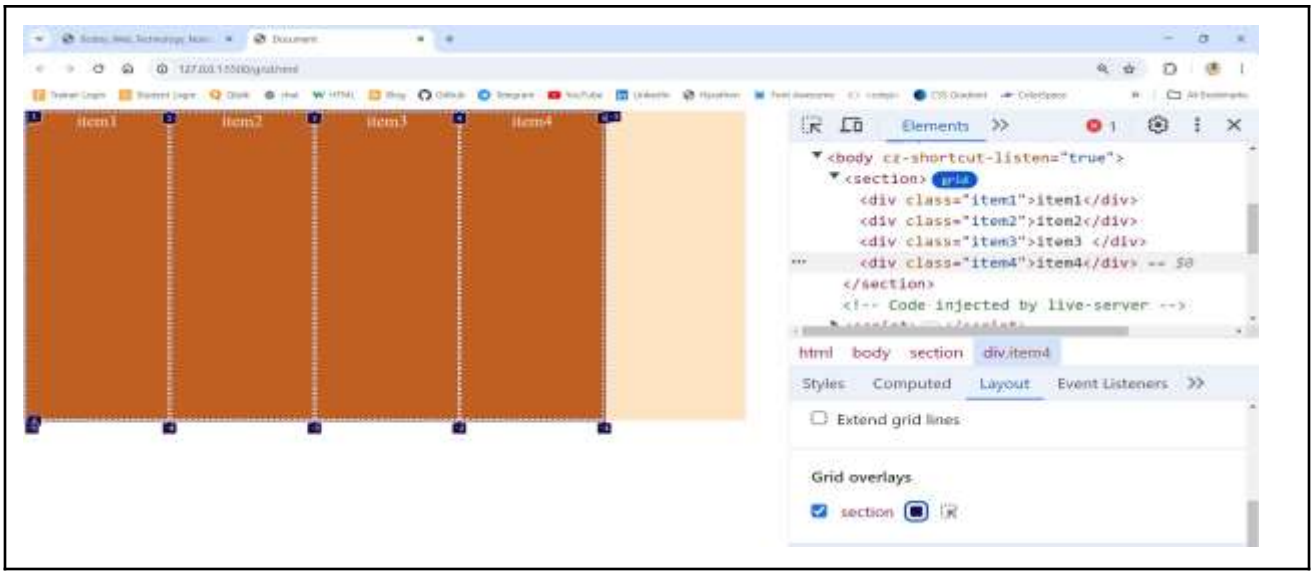
### 2. grid-template-columns:

a. It will specify no.of columns along with the width of each column.

b. As the width of the cell is defined here, It is not required to mention again for the grid item.

```html
<!-- html code -->
<html>
  <head>
    <title>Document</title>
    <link rel="stylesheet"
href="./grid.css" />
  </head>
  <body>
    <section>
      <div class="item1">item1</div>
      <div class="item2">item2</div>
      <div class="item3">item3</div>
      <div class="item4">item4</div>
    </section>
  </body>
</html>
```

```css
/* CSS Code */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

section {
  height: 500px;
  width: 1000px;
  background-color: bisque;
  border: solid;
  color: white;
  font-size: 25px;
  display: grid;
  grid-template-columns: 200px
200px 200px 200px;
/* It will consider as 4 columns
    each with 200px width */
}

div {
  background-color: chocolate;
  border: dashed;
  text-align: center;
}
```



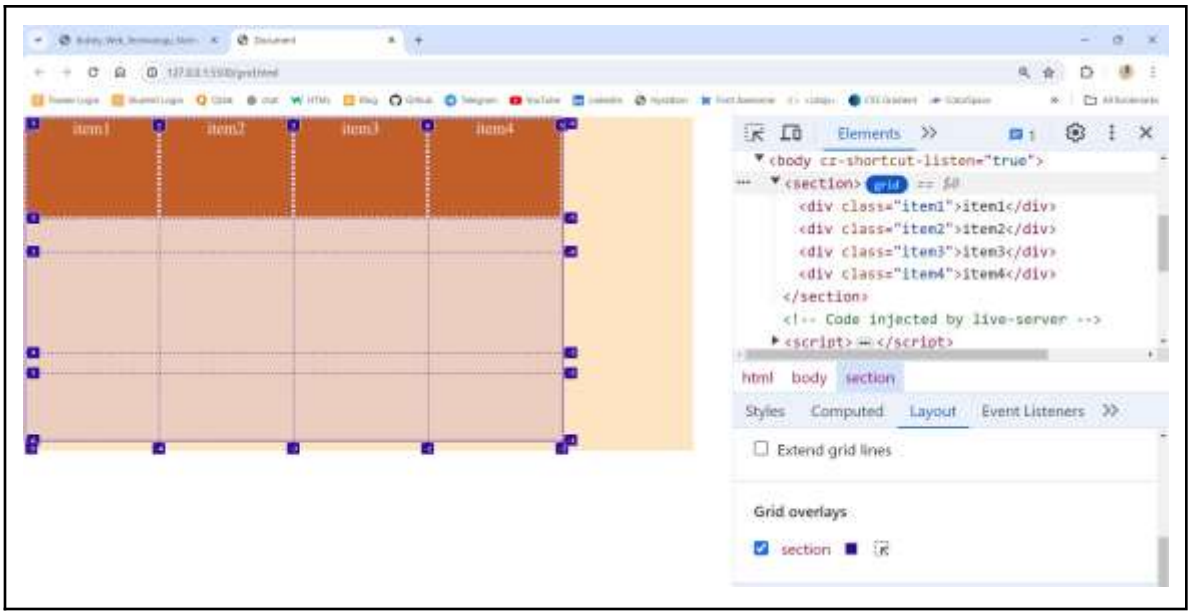

### 3. grid-template-rows:

a. It will specify no.of rows along with the height of each column.

b. As the height of the cell is defined here, It is not required to mention again for the grid item.

Ex:

```
/* CSS Code */
section {
  display: grid;
  grid-template-columns: 200px 200px 200px 200px;
  /* It will consider as 4 columns each with 200px width */
  grid-template-rows: 150px 50px 150px 30px 100px;
  /*  It will consider as 5 rows with widths => 1st - 150px , 2nd
- 50px , 3rd - 150px , 4th - 30px , 5th - 100px*/
}
```




We can pass the values for grid-template-rows and grid-template-columns in multiple ways.

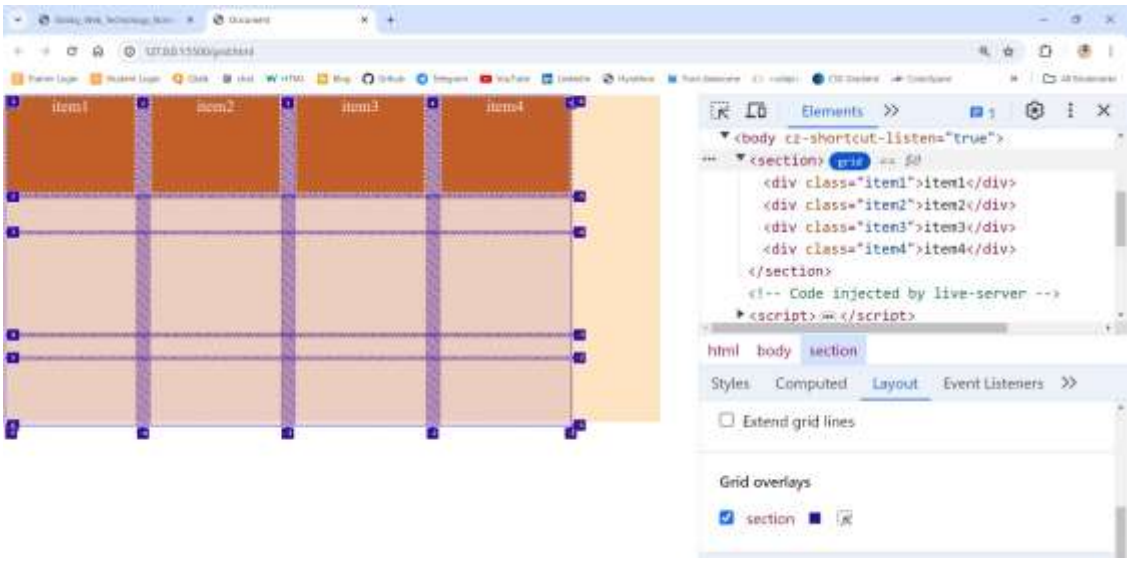
```
grid-template-columns: 200px 100px 150px 100px  80px 20px;
grid-template-columns: 200px 200px 200px 200px;
grid-template-columns: 150px 150px 150px 150px;
grid-template-columns: repeat(4, 180px);
grid-template-columns: repeat(4, auto);
grid-template-columns: 1fr 3fr 1fr;
grid-template-columns: repeat(4,1fr);

grid-template-rows: 150px 150px 100px;
grid-template-rows: 1fr 1fr 1fr;
grid-template-rows: 1fr 5fr 1fr;
grid-template-rows: 0.5fr 0.2fr 1fr 0.2fr;
```

**4. gap:**

a. It will specify  the space between the grid items.
b. It is the shorthand property of row-gap & column-gap.

```
/* CSS Code */
```

```
section {
  row-gap: 5px;
  column-gap: 20px;
  ===== or ======
  gap: 5px 20px;
}
```




**Grid Items Properties** are:

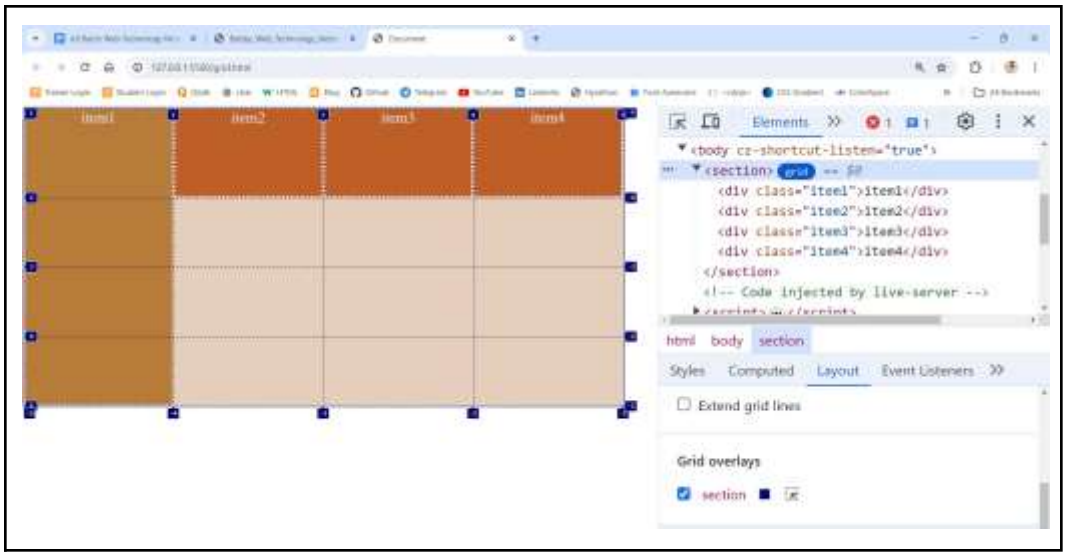
1. grid-row-start
2. grid-row-end
3. grid-column-start
4. grid-column-end
5. grid-row
6. grid-column
7. grid-area

**1. grid-row:**

a. It is a shorthand property of grid-row-start and grid-row-end.
b. Grid-row-start will specify where to start the row.
c. Grid-row-end will specify where to end the row.
d. Syntax: grid-row: grid-row-start/grid-row-end
e. Ex: grid-row: 1/5;
f. Ex:

g.

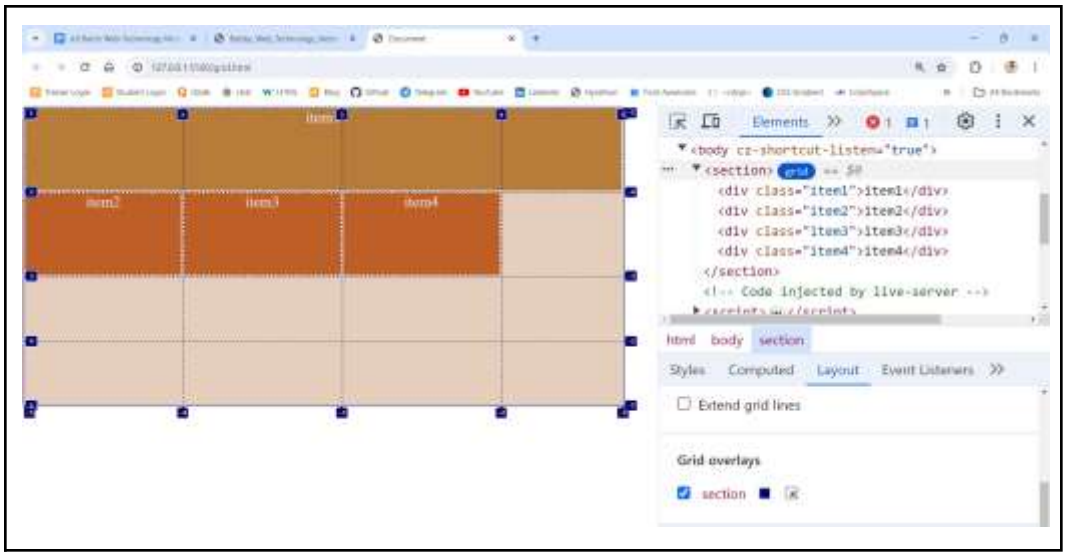
```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

section {
  height: 500px;
  width: 1000px;
  background-color: bisque;
  border: solid;
  color: white;
  font-size: 25px;
  display: grid;
  grid-template-columns: repeat(4, auto);
  grid-template-rows: repeat(4, auto);
}

div {
  background-color: chocolate;
  border: dashed;
  text-align: center;
}

.item1 {
  background-color: rgb(204, 138, 52);
  grid-row-start: 1;
  grid-row-end: 5;
  ======= or =======
   grid-row: 1/5;
}
```




## 2. grid-column:

a. It is a shorthand property of grid-column-start and grid-column-end.
b. Grid-column-start will specify where to start the column.
c. Grid-column-end will specify where to end the column.
d. Syntax: grid-column: grid-column-start/grid-column-end
e. Ex: grid-column: 1/5;
f. Ex:
g.

```
.item1 {
  background-color: rgb(204, 138, 52);
  grid-column-start: 1;
  grid-column-end: 5;
  ======= or =======
   grid-column: 1/5;
}
```



### 3. grid-area:

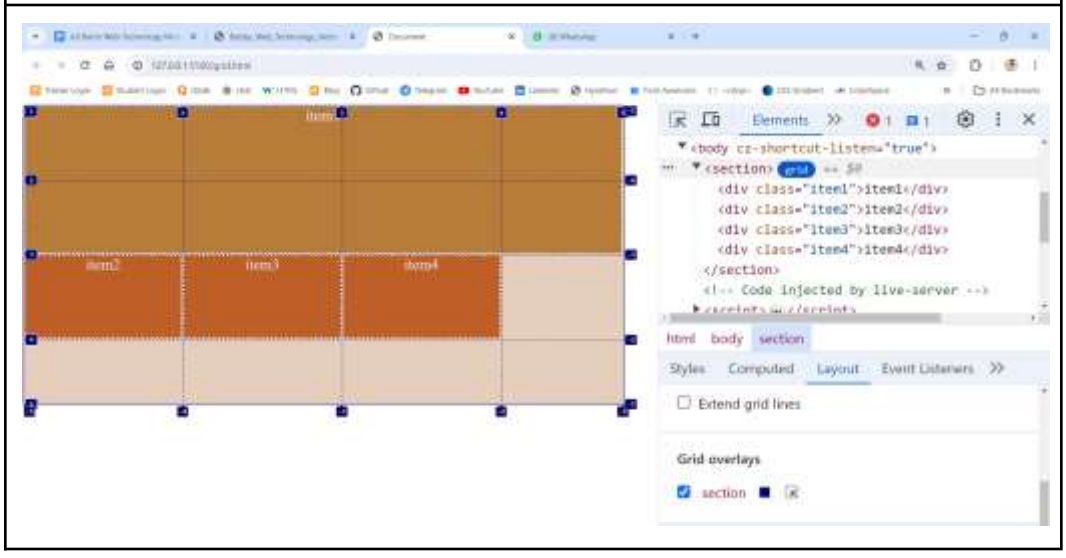
a. It is the shorthand property of grid-row-start , grid-column-start , grid-row-end and grid-column-end.

b. Syntax: grid-row-start/grid-column-start/grid-row-end/grid-column-end.

c. Ex:

d.

```
.item1 {
  background-color: rgb(204, 138, 52);
  grid-area: 1/1/3/5;
}
```



## Transform

Transform will change the state of an element like changing position , applying rotations , altering sizes, etc…

======== or ==========

It will transform the elements original behavior with new behaviors like changing positions , applying rotations , altering sizes , etc..

Syntax: transform: value;

Values: translateX() translateY() translate()
scaleX() scaleY() scale()
skewX() skexY() skew()
rotateX() rotateY() rotate()

### Translate():

It will change the position of an element according to x-axis and y-axis.

It is a shorthand property of

translateX() → which will according to x-axis

And

translateY() → which will according to y-axis

Ex: transform: translateX(200px);
transform: translateY(400px);
transform: translate(200px , 400px);

### Scale():

It will alter the sizes of an element based on the scale value provided either to increase or decrease.

By default, the scale is 1.

Ex: transform: scaleX(2); // It will increase the width into double.
transform: scaleY(0.5); // It will decrease the height into half.
transform: scale(2,0.5); // It will increase the width & decrease the height.

### Skew():

It will slant the elements in x and y axises.

Ex: transform: skewX(40deg);
transform: skewY(20deg);
transform: skew(40deg , 20deg);

**Rotate():**

It will rotate the element in clockwise and anticlockwise directions.

Ex: transform: rotateX(90deg);

transform: rotateY(180deg);

transform: rotate(90deg); // clockwise

transform: rotate(-90deg); // anticlockwise

## Transition

Transition will apply the hover effects smoothly.

Without hover this property will not work.

It is a shorthand property of

Transition-property

Transition-duration

Transition-delay

Transition-timing-function

**Transition-property:**

It will specify which properties have to be targeted. (In hover we will have many properties , for which property transition have to apply).

If we are not targeted specifically, It will target all.

**Transition-duration:**

It specifies how long the transition has to occur.

The default is 0 secs.

**Transition-delay:**

It specifies when the transition has to apply.

Basically, it provides a delay time. So transition will start after that time.

The default is 0 secs.

**Transition-timing-function:**

It specifies the speed format of transition.

We can assign the values like

ease-in → slow start

ease-out → slow end

ease-in-out → slow start and slow end
ease → slow start , fast , slow end
linear → same speed

## Animation

Animations will change the style of an element from one style to another style to another style and so on.

Animation is a shorthand property of

Animation-name
Animation-duration
Animation-delay
Animation-timing-function
Animation-direction
Animation-iteration-count

As we said, animation will change from one style to another and so on. To store all these styles we have to use @keyframes rule.

Syntax:

```
@keyframes identifier {

  from{
    // code
  }
  to{
    // code
  }

}
```

**Animation-name:**

It specifies the animation-name which is nothing the @keyframes rule identifier name.

We can create @keyframes rules with different styles.

Which animation you want to use, you can specify that identifier name.

**Animation-duration:**

It specifies how the animation has to apply.

**Animation-delay:**

It specifies when the animation has to start.

**Animation-timing-function:**

It specifies the speed format of animation.

We can assign the values like

ease-in → slow start

ease-out → slow end

ease-in-out → slow start and slow end

ease → slow start , fast , slow end

linear → same speed

**Animation-direction:**

It specifies the direction of animation.

We can assign the values normal , reverse , alternate and alternate-reverse.

**Animation-iteration-count:**

It specifies how many times the animation has to apply.

We can assign any number or infinite.

## CSS Units

- In CSS, Units will specify the length of an element / size of content.
- Units were categorized into 2 Ways
- They are:
    1. Absolute Units
    2. Relative Units

**Absolute Units:**

These units are static, It will not depend on any other values.

Absolute Units are:

1. mm
2. cm
3. in
4. Px

**Relative Units:**

These units are dynamic, It will depend on some other values.

Relative Units are:

1. vw → depends on viewport width
2. vh → depends on viewport height
3. % → depends on parent element
4. rem → depends on root element
5. em → depends on parent element

## CSS Responsiveness

To achieve responsiveness, mainly we have to use @media rule.

### @media:

Basically, this @media rule will provide some features.

Features like mediaType, mediaOperator and mediaFeatures which will work as conditions. If these conditions are satisfied then a particular block of code will be executed.

**Syntax:**

```
@media mediaType mediaOperator mediaFeatures{

    // code

}
```

**MediaType:**

It will specify the type of view mode.

It will accept the values like all, screen , print.

The default value is all.

**MediaOperator:**

This will combine multiple media features.

It will accept the values like and , or , not.

And → it will check if all the conditions are true or not.

Or → it will check if any of the conditions are true or not.

**MediaFeatures:**

It will provide some breakpoints based on screen sizes.

It will accept the values like:

1. min-width
2. max-width
3. width
4. min-height
5. max-height
6. height
7. orientation: landscape / portrait

**Ex:**

```
@media screen and (min-width:0px) and (max-width:400px){

    h1::after{
        content: "Mobile View";
    }
    body{
        background-color: teal;
    }

}

@media screen and (min-width:400px) and
(max-width:800px){

    h1::after{
        content: "Tablet View";
    }
    body{
        background-color: cadetblue;
    }

}
```

**@import:**

It will import the css files into a css file based on media conditions

**Ex:**

```
@import url(./mobileCodePathAddress) (min-width:0px) and (min-width:400px);
```

```
@import url(./printCode.css) print;
```

---

## JavaScript