CSS HANDBOOK

CSS

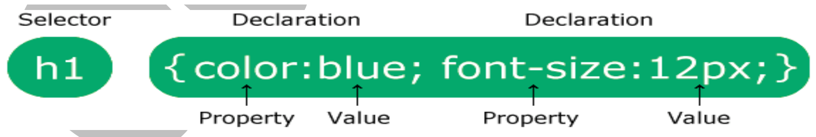
CSS stands for **Cascading Style Sheets**. It describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. **It can control the layout of multiple web pages all at once.** External style sheets are stored in CSS files. It was developed by W3C on 17 December 1996.

The advantages of CSS

i. CSS saves time – You can write CSS once and then reuse the same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
ii. Easy maintenance – To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
iii. Global web standards – Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible with future browsers.
iv. Platform Independence – The Script offer consistent platform independence and can support latest browsers as well.

CSS Rule/Syntax

A CSS rule consists of a selector and a declaration block.
CSS SYNTAX :-
It consists of selector & declaration block.



The different selectors of CSS

i. Simple selectors (select elements based on name, id, class):

Different types of simple selectors:

- **element selector** - The element selector selects HTML elements based on the element name.
  E.g. p { text-align: center; color: red; }
  For multiple elements (grouping selector):

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

E.g. h1, h2, p { text-align: center; color: red; }

- **id selector** - The id selector uses the id attribute of an HTML element 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.

  E.g. #para1 { text-align: center; color: red; }

  The CSS rule above will be applied to the HTML element with id="para1".

- **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.

  E.g. .center { text-align: center; color: red; }

  You can also specify that only specific HTML elements should be affected by a class.

  E.g. p.center { text-align: center; color: red; }

  In this example only <p> elements with class="center" will be red and center-aligned

  HTML elements can also refer to more than one class.

  E.g. <p class="center large">This paragraph refers to two classes. </p>

  In this example the <p> element will be styled according to class="center" and to class="large".

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

  E.g.* { text-align: center; color: blue; }

The differentiation between ID and Class selectors in CSS is presented in the table below:

Feature	ID Selector	Class Selector
Uniqueness	Must be unique within a single HTML document; only one element can have a specific ID.	Can be applied to multiple elements within a single HTML document.

Usage	Primarily used for targeting a specific, unique element that requires distinct styling or JavaScript interaction (e.g., a main content area, a unique button).	Used for applying the same styles to a group of elements that share common visual characteristics or functionality (e.g., all buttons, all navigation links).
CSS Syntax	Preceded by a hash symbol (#) in CSS (e.g., #myID).	Preceded by a dot (.) in CSS (e.g., .myClass).
HTML Attribute	Applied using the id attribute (e.g., <div id="header">).	Applied using the class attribute (e.g., <p class="text-highlight">).
Specificity	Has higher specificity than class selectors. Styles defined with an ID selector will generally override conflicting styles from a class selector on the same element.	Has lower specificity than ID selectors but higher than element selectors.
Multiple on Element	An element can only have one ID.	An element can have multiple classes (e.g., <button class="btn primary">).

ii. Combinator selectors:

A combinator is something that explains the relationship between the selectors. A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

- **descendant selector (space)** - The descendant selector matches all elements that are descendants of a specified element.
  E.g. div p { background-color: yellow;   }
  The above example selects all <p> elements inside <div> elements.
- **child selector (>)** - The child selector selects all elements that are the children of a specified element.
  E.g. div > p {   background-color: yellow; }
  The above example selects all <p> elements that are children of a <div> element.
- **adjacent sibling selector (+)** - The adjacent sibling selector is used to select an element that is directly after another specific element. Sibling elements must have the same parent element, and "adjacent" means "immediately following".

E.g. div + p {background-color: yellow;}
The above example selects the first <p> element that are placed immediately after <div> elements.

- **general sibling selector (~)** – The general sibling selector selects all elements that are next siblings of a specified element.
  E.g. div ~ p {background-color: yellow;}
  The above example selects all <p> elements that are next siblings of <div> elements.

iii. Pseudo-Class selectors:

A pseudo-class is used to define a special state of an element.
For example, it can be used to:

- Style an element when a user mouses over it.
- Style visited and unvisited links differently.
- Style an element when it gets focus.

Syntax: selector:pseudo-class { property: value; }
The most commonly used pseudo-classes:

- **Anchor Pseudo-classes** - Links can be displayed in different ways:
  E.g.
  ```
  /* unvisited link */
  a: link {color: #FF0000;}

  /* visited link */
  a: visited {color: #00FF00;}

  /* mouse over link */
  a: hover {color: #FF00FF;}

  /* selected link */
  a: active {color: #0000FF;}
  ```
  a: hover MUST come after a:link and a: visited in the CSS definition in order to be effective! a:active MUST come after a:hover in the CSS definition in order to be effective! Pseudo-class names are not case-sensitive.
- **:focus** – It is used to select an element which is currently focused by the user. It works on user input elements used in forms and is triggered as soon as the user clicks on it.
  E.g. Select and style an input field when it gets focus

input:focus {background-color: yellow; }

- **:first-child** - The :first-child selector is used to select the specified selector, only if it is the first child of its parent.
  E.g.
  Select and style every <p> element that is the first child of its parent:
  p:first-child { background-color: yellow; }
- **:lang** - The :lang() selector is used to select elements with a lang attribute with the specified value.
  Syntax: :lang(languagecode) { css declarations; }

  a. Pseudo-Classes and HTML Classes - Pseudo-classes can be combined with HTML classes.
     E.g. a.highlight:hover { color: #ff0000; }
     When you hover over the link in the above example, it will change color.
  b. hover on <div> - An example of using the :hover pseudo-class on a <div> element.
     E.g. div:hover { background-color: blue; }

iv. Pseudo-element selector:

A CSS pseudo-element is used to style specified parts of an element.
For example, it can be used to:

- Style the first letter, or line, of an element.
- Insert content before, or after, the content of an element.

Syntax: selector::pseudo-element { property: value; }

Different types of Pseudo-element:

- **::first-line** - The ::first-line pseudo-element is used to add a special style to the first line of a text.
  E.g. p::first-line { color: #ff0000; font-variant: small-caps }
  The above example formats the first line of the text in all <p> elements.
- **::first-letter** - The ::first-letter pseudo-element is used to add a special style to the first letter of a text.
  E.g. p::first-letter { color: #ff0000; font-size: xx-large; }
  The above example formats the first letter of the text in all <p> elements.
- **::before** - The ::before pseudo-element can be used to insert some content before the content of an element.
  E.g. h1::before { content: url(smiley.gif); }
  The above example inserts an image before the content of each <h1> element.

- **::after** - The ::after pseudo-element can be used to insert some content after the content of an element.
  E.g. h1::after { content: url(smiley.gif); }
  The above example inserts an image after the content of each <h1> element.
- **::marker** - The ::marker pseudo-element selects the markers of list items.
  E.g. ::marker { color: red; font-size: 23px; }
  The above example styles the markers of list items.
- **::selection** - The ::selection pseudo-element matches the portion of an element that is selected by a user. The following CSS properties can be applied to ::selection: color, background, cursor, and outline.
  E.g. ::selection { color: red; background: yellow; }
  The above example makes the selected text red on a yellow background.

v. Attribute selector:

The [attribute] selector is used to select elements with a specified attribute.
E.g. a[target] { background-color: yellow; }
The above example selects all <a> elements with a target attribute.
Different types of attribute selectors:

- **[attribute="value"]** - The [attribute="value"] selector is used to select elements with a specified attribute and value.
  E.g. a[target="_blank"] { background-color: yellow; }
  The above example selects all <a> elements with a target="_blank" attribute.
- **[attribute~="value"]** - The [attribute~="value"] selector is used to select elements with an attribute value containing a specified word.
  E.g. [title~="flower"] { border: 5px solid yellow; }
  The above example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower".
- **[attribute|="value"]** - The [attribute|="value"] selector is used to select elements with the specified attribute, whose value can be exactly the specified value, or the specified value followed by a hyphen (-).
  E.g. [class|="top"] { background: yellow; }
- **[attribute^="value"]** - The [attribute^="value"] selector is used to select elements with the specified attribute, whose value starts with the specified value.
  E.g. [class^="top"] { background: yellow; }
  The above example selects all elements with a class attribute value that starts with "top".

- **[attribute*=”value”]** - The [attribute*="value"] selector is used to select elements whose attribute value contains a specified value.
  E.g. [class*="te"] { background: yellow; }
  The above example selects all elements with a class attribute value that contains "te".

The different types of CSS

There are three types of CSS:-

- **Inline CSS** - An inline CSS is used to apply a unique style to a single HTML element. An inline CSS uses the style attribute of an HTML element.
  E.g.
  ```
  <h1 style="color:blue;">A Blue Heading</h1>
  <p style="color:red;">A red paragraph.</p>
  ```
  The above example sets the text color of the <h1> element to blue, and the text color of the <p> element to red.
- **Internal CSS** - An internal CSS is used to define a style for a single HTML page. An internal CSS is defined in the <head> section of an HTML page, within a <style> element.
  ```
  <!DOCTYPE html>
  <html>        <head>
  <style>
  body {background-color: powderblue;}   h1   {color: blue;} p   {color: red;}
  </style>       </head>
  <body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
  </body>       </html>
  ```
  The above example sets the text color of ALL the <h1> elements (on that page) to blue, and the text color of ALL the <p> elements to red. In addition, the page will be displayed with a "powderblue" background color.
- **External CSS** - An external style sheet is used to define the style for many HTML pages. To use an external style sheet, add a link to it in the <head> section of each HTML page.
  E.g.
  ```
  <!DOCTYPE html>
  <html>
  <head>
  ```

```
  <link rel="stylesheet" href="styles.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>        </html>
```

Note=(Priority of CSS: Inline > Internal > External.)

Background Image in CSS.

i. background-image:

The background-image property sets one or more background images for an element. By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally.

E.g.   body {background-image: url("img_tree.gif"), url("paper.gif"); background-color: #cccccc;}

The above example set two background images for the <body> element.

ii. background-origin:

The background-origin property specifies the origin position (the background positioning area) of a background image.

Different values used in background-origin property:

- padding-box - Default value. The background image starts from the upper left corner of the padding edge.
- border-box - The background image starts from the upper left corner of the border.
- content-box - The background image starts from the upper left corner of the content.
- Initial - Sets this property to its default value.
- inherit - Inherits this property from its parent element.

Syntax: background-origin: padding-box|border-box|content-box|initial|inherit;

iii. background-position:

The background-position property sets the starting position of a background image.

Different values used in background-position:

- left top, left center, left bottom, right top, right center, right bottom, center top, center center, center bottom.
  [If you only specify one keyword, the other value will be "center"]

- *x% y%* - The first value is the horizontal position and the second value is the vertical. The top left corner is 0% 0%. The right bottom corner is 100% 100%. If you only specify one value, the other value will be 50%. Default value is: 0% 0%.
- xpos ypos - The first value is the horizontal position and the second value is the vertical. The top left corner is 0 0. Units can be pixels (0px 0px) or any other CSS units. If you only specify one value, the other value will be 50%. You can mix % and positions.
- initial - Sets this property to its default value.
- inherit - Inherits this property from its parent element.

Syntax: background-position: value;

iv. background-repeat:

The background-repeat property sets if/how a background image will be repeated. By default, a background-image is repeated both vertically and horizontally.

Different values used in background-repeat property:

- **repeat -** The background image is repeated both vertically and horizontally. The last image will be clipped if it does not fit. This is default.
- **repeat-x -** The background image is repeated only horizontally.
- **repeat-y -** The background image is repeated only vertically.
- **no-**repeat - The background-image is not repeated. The image will only be shown once.
- **space -** The background-image is repeated as much as possible without clipping. The first and last image is pinned to either side of the element, and whitespace is distributed evenly between the images.
- **round -** The background-image is repeated and squished or stretched to fill the space (no gaps).

Syntax: background-repeat: repeat|repeat-x|repeat-y|no-repeat|initial|inherit;

v. background-size:

The background-size property specifies the size of the background images.

Different values in background-size property:

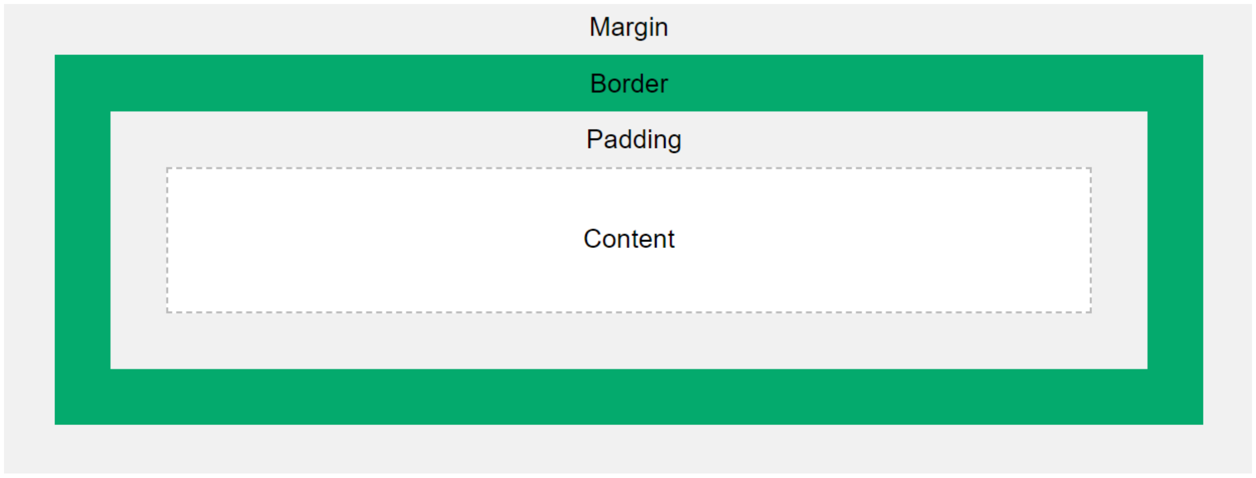
- **auto -** Default value. The background image is displayed in its original size.
- **length -** Sets the width and height of the background image. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto".
- **percentage -** Sets the width and height of the background image in percent of the parent element. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto".

- **cover -** Resize the background image to cover the entire container, even if it has to stretch the image or cut a little bit off one of the edges.
- **contain -** Resize the background image to make sure the image is fully visible.

Syntax: background-size: auto|length|cover|contain|initial|inherit;

CSS box model with the help of diagram.

In CSS, the term "box model" is used when talking about design and layout.
The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The box model allows us to add a border around elements, and to define space between elements.
The image below illustrates the box model:



Explanation of the different parts:

- **Content** - The content of the box, where text and images appear.
- **Padding** - Clears 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.

Example:
Demonstration of the box model:
div {width: 300px; border: 15px solid green; padding: 50px; margin: 20px; }

Border in CSS.

i. border-style:

The border-style property sets the style of an element's four borders.
This property can have from one to four values.

Examples:

- **<u>border-style: dotted solid double dashed;</u>**
  top border is dotted, right border is solid, bottom border is double, left border is dashed.
- **<u>border-style: dotted solid double;</u>**
  top border is dotted, right and left borders are solid, bottom border is double.
- **<u>border-style: dotted solid;</u>**
  top and bottom borders are dotted, right and left borders are solid.
- **<u>border-style: dotted;</u>**
  all four borders are dotted.

Different values used in border-style property:

- **none** - Default value. Specifies no border.
- **hidden** - The same as "none", except in border conflict resolution for table elements.
- **dotted** – Specifies a dotted border.
- **dashed** - Specifies a dashed border.
- **solid** - Specifies a solid border.
- **double** - Specifies a double border.
- **groove** - Specifies a 3D grooved border. The effect depends on the border-color value.
- **ridge** - Specifies a 3D ridged border. The effect depends on the border-color value.
- **inset** - Specifies a 3D inset border. The effect depends on the border-color value.
- **outset** - Specifies a 3D outset border. The effect depends on the border-color value.

Syntax: border-style: none|hidden|dotted|dashed|solid|double|groove|ridge|inset|outset|initial|inherit;

<u>ii. border-width:</u>

The border-width property sets the width of an element's four borders.
This property can have from one to four values.
Examples:

- border-width: thin medium thick 10px;
  top border is thin, right border is medium, bottom border is thick, eft border is 10px.
- border-width: thin medium thick;

top border is thin, right and left borders are medium, bottom border is thick.

- border-width: thin medium;
  top and bottom borders are thin, right and left borders are medium.
- border-width: thin;
  all four borders are thin.

Different values used in border-width property:

- medium – Specifies a medium border. This is default.
- thin - Specifies a thin border.
- thick - Specifies a thick border.
- length - Allows you to define the thickness of the border(px).

Syntax: border-width: medium|thin|thick|length|initial|inherit;

iii. border-color:

The border-color property sets the color of an element's four borders.
This property can have from one to four values.
Examples:

- border-color: red green blue pink;
  top border is red, right border is green, bottom border is blue, left border is pink.
- border-color: red green blue;
  top border is red, right and left borders are green, bottom border is blue.
- border-color: red green;
  top and bottom borders are red, right and left borders are green.
- border-color: red;
  all four borders are red.
  Different values used in border-color property:
- color - Specifies the border color. Look at CSS Color Values for a complete list of possible color values. Default color is the current color of the element.
- transparent – Specifies that the border color should be transparent.

Syntax: border-color: *color*|transparent|initial|inherit;

- **Difference between margin and padding**

Margin	Padding

The outer space of an element i.e. margin is the space outside the border.	The inner space of an element i.e.padding is space inside the element’s border.
It can be negative or any float number.	It does not allow negative values.
We can set the margin to auto.	We cannot set the padding to auto.
Styling of an element such as background color does not affect the margin.	Padding is affected by the styling of an element, such as background color.

Position property in CSS.

The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

- **There are five different position values:**

position:static; - HTML elements are positioned static by default. Static positioned elements are not affected by the top, bottom, left, and right properties. An element with **position: static;** is not positioned in any special way; it is always positioned according to the normal flow of the page.

position:relative; - An element with position: relative; is positioned relative to its normal position. Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

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. A fixed element does not leave a gap in the page where it would normally have been located.

position:absolute; - An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed). However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

position:sticky; - An element with position: sticky; is positioned based on the user's scroll position. A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

Syntax: position: static| absolute| fixed| relative| sticky| initial|inherit;

- **<u>Display property in CSS.</u>**

The display property specifies the display behavior (the type of rendering box) of an element.

Some values used in display property:

- **inline** - Displays an element as an inline element (like <span>). Any height and width properties will have no effect.
- **block** - Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width.
- **contents** - Makes the container disappear, making the child elements children of the element the next level up in the DOM.
- **flex** - Displays an element as a block-level flex container
- **grid** - Displays an element as a block-level grid container.
- **inline-block** - Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values.

  Syntax: display: value;

<u>Overflow property in CSS.</u>

The CSS overflow property controls what happens to content that is too big to fit into an area. It specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

<u>Different values used in overflow property:</u>

- **visible -** By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box.
- **hidden -** With the hidden value, the overflow is clipped, and the rest of the content is hidden.
- **Scroll -** Setting the value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it).
- **Auto -** The auto value is similar to scroll, but it adds scrollbars only when necessary.

- **x and y -** The overflow-x and overflow-y properties specifies whether to change the overflow of content just horizontally or vertically (or both):
  overflow-x - specifies what to do with the left/right edges of the content.
  overflow-y - specifies what to do with the top/bottom edges of the content.
  Syntax: **overflow:** visible| hidden| scroll| auto| initial| inherit;

<u>**Transition property in CSS.**</u>

i. **transition-property:** The transition-property property specifies the name of the CSS property the transition effect is for (the transition effect will start when the specified CSS property changes).

Different values used:

- **none -** No property will get a transition effect.
- **all -** Default value. All properties will get a transition effect.
- **property -** Defines a comma separated list of CSS property names the transition effect is for.

Syntax**: transition-property:** none| all| property| initial| inherit;

ii. **transition-duration:** The transition-duration property specifies how many seconds (s) or milliseconds (ms) a transition effect takes to complete.

Values used:

- **time -** Specifies how many seconds or milliseconds a transition effect takes to complete. Default value is 0s, meaning there will be no effect.

Syntax: transition-duration: time| initial| inherit;

iii. **transition-timing-function:** The transition-timing-function property specifies the speed curve of the transition effect.

This property allows a transition effect to change speed over its duration.

Different values used:

- **ease -** Default value. Specifies a transition effect with a slow start, then fast, then end slowly .
- **linear -** Specifies a transition effect with the same speed from start to end.
- **ease-in -** Specifies a transition effect with a slow start.
- **ease-out -** Specifies a transition effect with a slow end.
- **ease-in-out -** Specifies a transition effect with a slow start and end.

Syntax: transition-timing-function: linear|ease|ease-in|ease-out|ease-in-out|step-start|step-end|steps(int,start|end)|cubic-bezier(n,n,n,n)|initial|inherit;

iv. **transition-delay:** The transition-delay property specifies when the transition effect will start.Its value is defined in seconds (s) or milliseconds (ms).

Values used:

- **time -** Specifies the number of seconds or milliseconds to wait before the transition effect will start.

Syntax: transition-delay: time| initial| inherit;

{ transform: matrix(scaleX(); skewY(); skewX(), scaleY(), translateX(), translate());}

Transform property in CSS.

The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements.

Different values used in transform property (2D&3D):

Value	Description	Units & Example

1. Translate
• translate(x,y)
• translate3d(x.y,z)
• translateX|Y|Z(x|y|z)
	Defines a 2D translation
	Defines a 3D translation
	Defines a translation, using only the value for the X, Y, Z-axis respectively.
		In Pixels
		:translate(+- 25px,+- 25px);
		:translate3d(20px,20px,20px);
		:translateX(10px);
		(Can have both negative and positive values).

2. Scale
• scale(x,y)
• scale3d(x.y,z)
• scaleX|Y|Z(x|y|z)
 	
3. Rotate
• rotate(angle)
• rotate3d(x,y,z,angle)
• rotateX|Y|Z(angle)
	Defines a 2D scale transfor.
	Defines a 3D scale transfor.
	Defines a scale transformation, using only the value for the X, Y, Z-axis respectively.
	
	Defines a 2D rotation
	Defines a 3D rotation
	Defines a 3D rotation along the X, Y, Z-axis respectively.
		In numbers
		:scale(1.1,1.1);
		:scale3d(1.1,1.1,1.1);
		:scaleX(1.1);
		(Can have decimal values).
		
		In Degrees
		:rotate(40deg);
		:rotate3d(1,1,1,40deg);
		:rotateX(10deg);

4. Skew • skew(x-angle,y-angle) • skewX|Y(angle)		**In Degrees** :skew(10deg,10deg);

	Defines a 2D skew transfor. along the X-axis and Y-axis Defines a 2D skew transformation along the X-axis or Y-axis respectively.	:skewX(20deg);
5. **Matrix** • matrix(n,n,n,n,n,n) • matrix3d	Defines a 2D transformation, using a matrix of six values Defines a 3D transformation, using a 4x4 matrix of 16 values	**In Numbers** :matrix(0.586,0.8,-0.8, 0.586,40,10); (Can have decimal, positive and negative values).

<u>Transform-origin property in CSS.</u>

The transform-origin property allows you to change the position of transformed elements.

2D transformations can change the x- and y-axis of an element. 3D transformations can also change the z-axis of an element.

Different values used in transform-origin property:

- **x-axis -** Defines where the view is placed at the x-axis.

Possible values: (left,center,right,length,%).

- **y-axis -** Defines where the view is placed at the x-axis.

Possible values: (left,center,right,length,%).

- **z-axis -** Defines where the view is placed at the x-axis(for 3D transformations).

Possible values: (left,center,right,length,%).

Syntax: **transform-origin:** x-axis y-axis z-axis| initial| inherit;

Different CSS Properties.

a) **align-content:** The align-content property modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines.

Syntax: **align-content:** stretch|center|flex-start|flex-end|space-between|space-around|initial|inherit;

b) **align-items:** The align-items property specifies the default alignment for items inside the flexible container.
Syntax: **align-items:** stretch| center| flex-start| flex-end| baseline | initial| inherit;

c) **align-self:** The align-self property specifies the alignment for the selected item inside the flexible container.
Syntax: **align-self:** auto| stretch| center| flex-start| flex-end| baseline | initial| inherit;

d) **font-family:** The font-family property specifies the font for an element. The font-family property can hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font (Georgia, book antiqua, times new roman, arial, etc.).
Syntax: **font-family:** family-name| generic-family| initial| inherit;

e) **font-size:** The font-size property sets the size of a font.
Syntax: **font-size:** medium|xx-small|x-small|small|large|x-large|xx-large|smaller|larger|length|initial|inherit;

f) **font-style:** The font-style property specifies the font style for a text (normal, italic, oblique).
Syntax: **font-style:** normal|italic|oblique|initial|inherit;

g) **font-weight:** The font-weight property sets how thick or thin characters in text should be displayed (normal, bold, bolder, lighter, numeric {100, 200, etc}).
Syntax: font-weight: normal| bold| bolder| lighter| number| initial | inherit;

h) **opacity:** The opacity property sets the opacity level for an element.The opacity-level describes the transparency-level, where 1 is not transparent at all, 0.5 is 50% see-through, and 0 is completely transparent.
Syntax: **opacity:** number|initial|inherit;

i) **text-align:** The text-align property specifies the horizontal alignment of text in an element (left, right, center, justify).

Syntax: text-align: left|right|center|justify|initial|inherit;

j) **text-decoration:** The text-decoration property specifies the decoration added to text, and is a shorthand property for.
Syntax: **text-decoration:** text-decoration-line text-decoration color text-decoration-style|initial|inherit;

- **text-decoration-line -** The text-decoration-line property sets the kind of text decoration to use (like underline, overline, line-through).
  (none, underline, overline, line-through).
- **text-decoration-color -** The text-decoration-color property specifies the color of the text-decoration **(underlines, overlines, linethroughs).**
- **text-decoration-style -** The text-decoration-style property sets the style of the text decoration **(like solid, wavy, dotted, dashed, double).**

k) text-justify - The text-justify property specifies the justification method of text when text-align is set to "justify" **(auto, inter-word, inter-character, none).**
Syntax: text-justify: auto| inter-word| inter-character| none| initial |inherit;

l) **visibility -** The visibility property specifies whether or not an element is visible.
Syntax: **visibility:** visible| hidden| collapse| initial| inherit;