What is CSS?

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications CSS is designed to make style sheets for the web.

Cascading: Refers to how CSS applies rules based on hierarchy, importance, and specificity.
Style: Refers to defining the visual appearance like color, font, size, layout, etc.
Sheet: Refers to the external or internal file that holds these styling rules.

==========
==========

CSS History

CSS 1 -: ----------

The first CSS specification to become an official W3C Recommendation is CSS level 1, published on December 17, 1996. Hakon Wium Lie and Bert Bos are credited as the original developers.Among its capabilities are support for Font properties such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes such as spacing between words,

letters, and lines of text Alignment of text, images, tables and other elements Margin, border, padding, and positioning for most elements Unique identification and generic classification of groups of attributes The W3C no longer maintains the CSS 1 Recommendation.

CSS 2 -: ----------

CSS level 2 specification was developed by the W3C and published as a recommendation in May 1998. A superset of CSS 1, CSS 2 includes a number of new capabilities like absolute, relative, and fixed positioning of elements and z index, the concept of media types, support for aural style sheets (which were later replaced by the CSS 3 speech modules) and bidirectional text, and new font properties such as shadows.

The W3C no longer maintains the CSS 2 recommendation.

CSS 2.1 -: ----------

CSS level 2 revision 1, often referred to as "CSS 2.1", fixes errors in CSS 2, removes poorly supported or not fully interoperable features and adds already implemented browser extensions to the specification. To comply with the W3C Process for standardizing technical specifications, CSS 2.1 went back and forth between Working Draft status and Candidate Recommendation status for many years. CSS 2.1 first became a Candidate Recommendation on February 25, 2004, but it was reverted to a Working Draft on June 13, 2005 for further review. It returned to Candidate Recommendation on 19 July 2007 and then updated twice in

2009. However, because changes and clarifications were made, it again went back to Last Call Working Draft on 7 December 2010.

CSS 2.1 went to Proposed Recommendation on 12 April 2011. After being reviewed by the W3C Advisory Committee, it was finally published as a W3C

Recommendation on 7 June 2011.

CSS 2.1 was planned as the first and final revision of level 2—but low priority work on CSS 2.2 began in 2015.

CSS 3-: -----------

"CSS3" redirects here. For other uses, see CSS3 (disambiguation). Unlike CSS 2, which is a large single specification defining various features, CSS 3 is divided into several separate documents called "modules". Each module adds new capabilities or extends features defined in CSS 2, preserving backward compatibility. Work on CSS level 3 started around the time of publication of the original CSS 2 recommendation. The earliest CSS 3 drafts were published in June 1999.

What are the uses of CSS?

CSS is used for defining the styles for web pages. It describes the look and formatting of a document which is written in a markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces.

It is easier to make the web pages presentable using CSS. It is easy to learn and understand and used to control the presentation of an HTML document. CSS helps us to control the text color, font style, the spacing between paragraphs, sizing of columns, layout designs, and many more. It is independent of HTML, and we can use it with any XML-based markup language.It is recommended to use CSS because the HTML attributes are being deprecated. So, for making HTML pages compatible with future browsers, it is good to start using CSS in HTML pages.

There are several uses of CSS that are discussed as follows -:

1)Solves a big problem

Before CSS, tags like font, color, background style, element alignments, border, and size had to be repeated on every web page. This was a very long process.

For example: If we are making a large website where fonts and color information are required to add on every page, it will be a long process. CSS was created to solve this problem. It was a W3C recommendation.

2)Saves a lot of time

CSS style definitions are saved in external CSS files, so it is possible to change the entire website by changing just one file.

3)Provide more attributes

CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

4)Pages load faster

CSS does not require the writing of HTML tag attributes every time. There is the writing of rule just once for a tag, which can be applied to all the occurrences of the corresponding tag. So using CSS, there is less code, which means faster downloading.

5)Easier Website maintenance

CSS makes the maintenance of the website easier. It plays an essential role in website maintenance. If we require a global change in the file, it can be simply done by changing the style by which all the elements on the web page will update automatically. The CSS file provides a flexible look to the website, which can be altered in a convenient way. It also makes HTML formatting and modification of corresponding data elements easier.

6)Multiple device compatibility

CSS is compatible with the older language versions so that we can use CSS with the earlier language versions. Because of this,

if the CSS application is developed with the older programming language versions and if the developer combines the same with new improvements, then CSS can be easily implemented with the corresponding changes so the developer can update the existing code successfully. CSS allows the content to be optimized for more than one type of device.

How To Write CSS?

There are 3 ways to write CSS in our HTML file-:

1.Inline CSS

2.Internal CSS

3.External CSS

1] Inline CSS -:

Are those css which are used with style attribute inside html element. Before CSS this was the only way to apply styles

Not an efficient way to write as it has a lot of redundancy

Self-contained Uniquely applied on each element

The idea of separation of concerns was lost

Syntax-:

Hello,Welcome to

2] Internal /Embedded CSS -:

Are those css which are used with <style> inside an html document. With the help of style tag, we can apply styles within the HTML file Redundancy is removed But the idea of separation of concerns still lost Uniquely applied on a single document

Syntax-:

<style type="text/css">

p

{

color:red;

}

</style>

Write the above mentioned syntax in <head> tag.

3] External Css -:

Where we create external file having .css extension.

With the help of <link> tag in the head tag, we can apply styles

Reference is added File saved with .css extension Redundancy is removed The idea of separation of concerns is maintained Uniquely applied to each document

Syntax -:

p

{

color:red;

}

Save above the syntax in css file with .css extension & link css file inside HTML

using <link> tag.

--Priority order--

Inline > Internal > External

If you apply all of these files the 1st priority will be given to Inline CSS then

Internal CCC & then External CSS.

==========

CSS Selector

CSS selectors are used to select the content you want to style. Selectors are part of the CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

There are several different types of selectors in CSS.

1]CSS Element Selector

2]CSS Id Selector

3]CSS Class Selector

4]CSS Universal Selector

5]CSS Group Selector

6]CSS Descendent Selector

7]CSS Child Combinator

8]CSS Attribute Selector

1] CSS Element Selector -:

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

Syntax -:

```
<style>
p{
text-align: center;
```

color: blue;

}

</style>

2]CSS Id Selector -:

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

It is written with the hash character (#), followed by the id of the element.

Syntax -:

<style>

#para1 {

text-align: center;

color: blue;

}

</style>

3]CSS Class Selector-:

The class selector selects HTML elements with a specific class attribute. It is used with a (.) character . (full stop symbol) followed by the class name.

Syntax -:

<style>

.center {

text-align: center;

color: blue;

}

</style>

=>If you want to specify that only one specific HTML element should be affected then you should use the element name with the class selector.

<style>

p.center {

text-align: center;

color: blue;

}

</style>

4]CSS Universal Selector-:

The universal selector is used as a wildcard character. It selects all the elements on the pages.

Syntax-:

<style>

{

color: green;

font-size: 20px;

}

</style>

5]CSS Group Selector-:

The grouping selector is used to select all the elements with the same style definitions. Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

Syntax-:

h1,h2,p,#p1,.para {

text-align: center;

color: blue;

}

6]Descendent Selector

The descendant combinator matches all elements that are descendants of a specified element.

The following example selects all <p> elements inside elements:

Syntax-:

```
div p {
  background-color: yellow;
}
```

7]Child Combinator :-

The child combinator selects all elements that are the children of a specified element.

The following example selects all <p> elements that are children of a element:

Syntax:-

```
div > p {
  background-color: yellow;
}
```

8]Attribute Selector :-

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

The following example selects all <a> elements with a target attribute:

Syntax-:

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

Pseudo-classes-:

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 moves the mouse over it
- Style visited and unvisited links differently
- Style an element when it gets focus
- Style valid/invalid/required/optional form elements

Link-related pseudo class:

:link , :visited , :hover , :active

Input-related pseudo class:

:focus, :disabled, :checked, :required, :read-only, :valid,:invalid

Position/Number related pseudo class:

:root, :first-child, :last-child, :nth-child(), :empty

Pseudo-elements-:

Content-related:

::after (:after), ::before (:before)

Text related:

::first-letter, ::first-line, ::placeholder , ::selection

==========
==========

CSS Units and Measurements

When working with CSS, understanding the different types of units and measurements is essential for creating responsive and adaptable designs. Here's a breakdown of the types of units you'll encounter in CSS:

---

1. Absolute Units

Absolute units are fixed, meaning they don’t change based on the context in which they are used (such as screen size or device resolution). They are useful for print layouts or when exact measurements are needed.

•px (pixels): Represents a single dot on the screen. The most commonly used unit for web design.

•cm (centimeters): Measures length in centimeters. Generally used for print.

•mm (millimeters): Measures length in millimeters. Like cm, it's used for print but less common on the web.

•in (inches): Measures length in inches. Primarily used for print media.

When to use absolute units:

•When you need precise control over element dimensions, like printing or creating an element where exact pixel values are crucial.

---

2. Relative Units

Relative units change their size depending on other factors such as the parent element, viewport size, or font size. These are especially helpful for responsive designs.

•% (percentage): A percentage value is relative to the parent element’s size. For example, width: 50% means 50% of the parent’s width.

•em: Relative to the font size of the element’s closest parent (or the body if no parent has a specified font size). For example,

font-size: 2em means the font size is twice the size of the element's current font size.

•rem: Stands for "root em" and is relative to the font size of the root element (usually the <html> tag). For example, if the root element has font-size: 16px, then 2rem equals 32px.

•vw (viewport width): Represents a percentage of the viewport's width. 1vw equals 1% of the viewport's width.

•vh (viewport height): Represents a percentage of the viewport's height. 1vh equals 1% of the viewport's height.

When to use relative units:

•For flexible and scalable layouts that adjust according to screen size or the user's preferences.

•For typography that scales dynamically with the page or element size.

---

3. Responsive Units

Responsive units, such as %, vw, vh, em, and rem, are useful in creating designs that adapt to different screen sizes and orientations. These are especially important for responsive web design to ensure that elements look good on a variety of devices, from mobile phones to desktop monitors.

•vw/vh: Great for creating full-page elements that adjust based on the viewport size.

•%: Often used in fluid layouts to make sure elements grow and shrink with the parent container.

•em/rem: Helpful for flexible typography, allowing the text size to scale with the user's settings or the base font size.

When to use responsive units:

•When designing fluid layouts or scalable typography.

•When you need to create flexible grid systems, container sizes, and elements that adjust to various screen widths.

---

4. CSS Calculations (calc() function)

The calc() function in CSS allows you to perform calculations with different units, making it easier to create responsive designs and precise layouts without relying on pre-calculated values.

For example:

width: calc(100% - 50px);

This will set the width to 100% of the parent element, minus 50 pixels. You can use calc() with any combination of units like px, em, %, etc.

When to use calc():

•When you need to combine different units (like percentages and pixels).

•To create dynamic layouts where element dimensions adjust based on other factors.

Summary of When to Use Each Unit

•Absolute Units: When you need fixed, unchanging measurements (e.g., px, in).

•Relative Units: When you want flexibility and responsiveness (e.g., %, em, rem, vw, vh).

•Responsive Units: When designing for different screen sizes or creating adaptable layouts.

•calc(): When combining different units or performing dynamic calculations.

1]Text Properties-:

CSS has a lot of properties for formatting text.You can set the following text properties of an element.

Syntax-:

color: colorname | Hashcode | rgb() , rgba();

text-align: center | left | right | justify;

text-decoration: overline | underline | line-through | none ;

text-transform: capitalize | uppercase | lowercase ;

text-indent: 40px ;

letter-spacing: 3px ;

word-spacing: 10px ;

line-height: 20px ;

Text Shadow:-

The text-shadow property adds shadow to text.

This property accepts a comma-separated list of shadows to be applied to the text.

Syntax:-

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

Box Shadow:-

The boxshadow property adds shadow to box

This property accepts a comma-separated list of shadows to be applied to the box

Syntax:-

box-shadow: x-axis y-axis blur color;

==========
==========

2]Font Properties-:

Font Selection is Important. Choosing the right font has a huge impact on how the readers experience a website.The right font can create a strong identity for your brand.Using a font that it is easy to read is important. The font adds value to your text. It is also important to choose the correct color and text size for the font.

Generic Font Families

In CSS there are five generic font families:

1.Serif fonts have a small stroke at the edges of each letter. They create a sense

of formality and elegance.

2.Sans-serif fonts have clean lines (no small strokes attached). They create a

modern and minimalistic look.

3.Monospace fonts - here all the letters have the same fixed width. They create a

mechanical look.

4.Cursive fonts imitate human handwriting.

5.Fantasy fonts are decorative/playful fonts.

Syntax-:

font-family: Arial, Helvetica, sans-serif;

font-size: medium | large | x-large | xx-large | xx-small | x-small | small | 20px;

font-style: Normal | italic | oblique;

font-variant: Normal | small-caps;

font-weight: normal | lighter | bolder | bold | number (100-900);

Defines from thin to thick characters. 400 is the same as normal, and 700 is the

same as bold

==========

3]Background Properties-:

The background property in CSS allows you to control the background of any element (what paints underneath the content in that element).

It is a shorthand property, which means that it allows you to write what would be multiple CSS properties in one.

Syntax-:

Background : (shorthand property)

background-color : colorName | HashCode | rgb() | rgba();

rgba : RED, GREEN, BLUE & ALPHA [for transparancy]

background-image : url();

background-repeat : no-repeat | repeat-x | repeat-y;

background-size:100% 100%;

background-position: top | center | bottom | left | top right | top center | top

left | bottom right | bottom left | bottom center | center center ;

background-position: 25% 75%;

background-position: bottom 50px right 100px;

background-position: right 35% bottom 45%;

background-attachment: scroll | fixed;

BackgroundGradient-:

Syntax-:

background:linear-gradient(to bottom,red,orange,yellow);

background:radial-gradient(red,orange,yellow);

background:linear-gradient(to bottom,red,orange,yellow) , url();

==========

4]List Properties-:

There are various CSS properties that can be used to control lists. Lists can be classified as ordered lists and unordered lists. In ordered lists, marking of the list items is with alphabet and numbers, whereas in unordered lists, the list items are marked using bullets.

Syntax-:

list-style-type: disc | circle | square | lower-alpha | upper-alpha | upper-roman |

lower-roman | none ;

list-style-image: url(../Resources/close.png);

==========

5]Dimensions -:

The CSS height and width properties are used to set the height and width of an element.The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

Syntax-:

width: 20px | 100% | auto;

height: 20px | 100% | auto;

==========

6]Border-:

The CSS border is a shorthand property used to set the border on an element.

The CSS border properties are use to specify the style, color and size of the border of an element. The CSS border properties are given below.

Syntax-:

border: 1px solid black;

(shorthand property)

border-bottom

border-top

border-left

border-right

border-style : solid | dashed | double | dotted | groove | ridge | inset | outset |

hidden | none;

border-width: thin | medium | thick | 1px ;

border-color : red | rgb()| rgba() | hashcode ;

border-radius: 100%;

border-radius: top right bottom left;

CSS-3

Border-:

The border-radius property defines the radius of the element's corners.This

property allows you to add rounded corners to elements!

Syntax-:

border-radius : 1px 1px 1px 1px;

tl tr br bl ;

border-bottom-left-radius: 10px 70px;

border-bottom-right-radius: 50px 50px;

border-top-left-radius: 20px 200px;

border-top-right-radius: 20px 200px;

box-shadow:x axis y axis blur shadowColor;

==========

--Outline--

CSS outline is just like CSS border property. It facilitates you to draw an extra

border around an element to get visual attention.

It is as easy as to apply as a border.

Syntax-:

Outline: 1px solid black;

(shorthand property)

Outline-width:2px;

Outline-style: ; same as border given above

Outline-color:red | rgb() | hashcode ;

Outline-offset: 6px;

==========

7]Margin Properties -:

The CSS margin properties are used to create space around elements, outside of any defined borders.With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).Negative values are allowed.

All the margin properties can have the following values:

auto - the browser calculates the margin

length - specifies a margin in px, %, rem,em etc.

% - specifies a margin in % of the width of the containing element

Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

Syntax-:

margin-top

margin-right

margin-bottom

margin-left

(Shorthand Property)

margin : top right bottom left;

margin : top&bottom right&left;

==========

8] Padding Properties -:

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element

(top, right, bottom, and left).Negative values are not allowed.

All the margin properties can have the following values:

auto - the browser calculates the margin

length - specifies a margin in px, %, rem,em etc.

% - specifies a margin in % of the width of the containing element

Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

Syntax-:

padding-top

padding-right

padding-bottom

padding-left

(Shorthand Property)

padding : top right bottom left;

padding : top&bottom right&left;

==========
==========

BOX Model Concept of CSS :-

The CSS box model is a fundamental concept that defines how elements are rendered on a webpage.

It consists of four key parts that determine the size and spacing of an element:

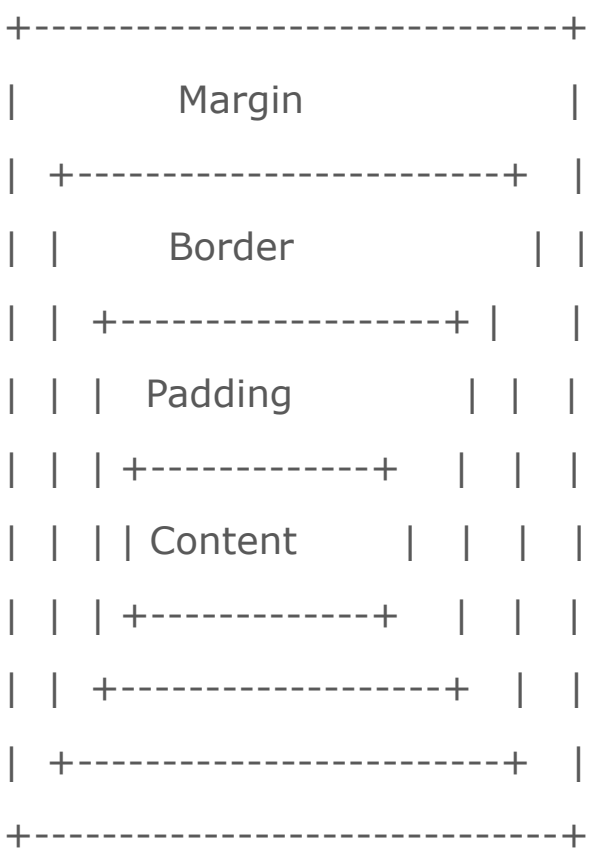
1.Content: This is where the actual content of the element (like text or images) is displayed. The size of the content area is defined by width and height.

2.Padding: Padding is the space between the content and the border. You can control the padding with properties like padding-top, padding-right, padding-bottom, and padding-left. Padding increases the size of the element's total area.

3.Border: The border wraps around the padding and content area. You can define its width, style (e.g., solid, dashed), and color. The border is placed between the padding and the margin.

4.Margin: Margin is the space outside the border, separating the element from other elements. Margins don't affect the size of the element itself, but they do affect the layout and spacing between elements. You can control margins with margin-top, margin-right, margin-bottom, and margin-left.

Visual Representation:



Box-Sizing Property:

By default, the width and height set on an element only account for the content area, so padding and borders are added outside the specified width and height, increasing the total size. However, you can change this behavior using the box-sizing property.

•box-sizing: content-box; (default) – The width and height are only for the content area. Padding and borders are added outside.

•box-sizing: border-box; – The width and height include the padding and border, so the overall size doesn't increase.

```
Example:div {
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 5px solid black;
  margin: 10px;
  box-sizing: border-box;
}
```

==========
==========

9]Display Properties -:

CSS display is the most important property of CSS which is used to control the layout of the element. It specifies how the element is displayed.

Every element has a default display value according to its nature.The default display value for most elements is block or inline.

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Inline Elements

The inline element takes the required width only. It doesn't force the line break so the flow of text doesn't break in inline.

Display: none;

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.

Syntax-:

Display: inline | block | inline-block | none;

The display property in CSS determines how an element is visually presented on the page.

1. block

•Behavior: The element takes up the full width available (by default) and starts on a new line.

•Examples: , <p>, <h1>, <section>

•Usage: Block elements will always begin on a new line, stacking vertically.

```
div {
    display: block;
}
```

---

2. inline

•Behavior: The element takes up only as much width as necessary, and it doesn't cause line breaks.

•Examples: , <a>,

•Usage: Inline elements flow within text, meaning they don't interrupt the flow of other elements, appearing on the same line if there's space.

```
span {
    display: inline;
```

}

---

3. inline-block

•Behavior: The element behaves like an inline element (stays on the same line as other inline elements) but can have block-level styling (e.g., width and height).

•Examples: <img>, <button>, <input>

•Usage: Useful when you want elements to align horizontally but still need to control their dimensions.

```
button {
    display: inline-block;
    width: 100px;
    height: 50px;
}
```

4. none

•Behavior: The element is not displayed at all. It takes up no space in the layout and is completely removed from the document's flow.

•Examples: Used for hiding elements (commonly in JavaScript or CSS transitions).

•Usage: Typically used when you want to hide an element without removing it from the DOM.

```
div {
    display: none;
}
```

---

5. flex

•Behavior: The element is treated as a flex container, and its children become flex items. The items are laid out according to the flexbox model, allowing for flexible and dynamic layouts.

•Examples: Used in layouts that need flexible and responsive alignment.

•Usage: Great for creating rows or columns of items that adjust based on the container size.

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

```
    justify-content: space-between;
}
```

---

6. grid

•Behavior: The element is treated as a grid container, and its children become grid items. This layout model allows for the creation of complex 2D layouts with rows and columns.

•Examples: Used for more complex layouts than flexbox, with both rows and columns.

•Usage: Ideal for creating more structured, multi-row, and multi-column designs.

```
.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
}
```

Display -:

This defines a flex container; inline or block depending on the given value. It enables a flex context for all its direct children.

Property for parent element -:

Syntax-:

display: flex;

i. flex-direction Property

The flex-direction property specifies the direction of the flexible items.

Syntax-:

flex-direction: row (default) |row-reverse|column|column-reverse;

row (default): left to right in ltr; right to left in rtl

row-reverse: right to left in ltr; left to right in rtl

column: same as row but top to bottom

column-reverse: same as row-reverse but bottom to top

---

ii. flex-wrap Property

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

Syntax-:

flex-wrap: nowrap (default) | wrap | wrap-reverse;

---

iii. justify-content Property

The justify-content property aligns the flexible container's items when the items

do not use all available space on the main-axis (horizontally).

Syntax-:

justify-content: flex-start (default) | flex-end | center | space-between |

space-around | space-evenly;

flex-start (default): items are packed toward the start of the flex-direction.

flex-end: items are packed toward the end of the flex-direction.

start: items are packed toward the start of the writing-mode direction.

center: items are centered along the line

space-between: items are evenly distributed in the line; first item is on the start line, last item on the end line space-around: items are evenly distributed in the line with equal space around them.

Note that visually the spaces aren't equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies.

space-evenly: items are distributed so that the spacing between any two items (and the space to the edges) is equal.

---

iv. align-items Property

The align-items property specifies the default alignment for items inside the flexible container.

Tip: Use the align-self property of each item to override the align-items property.

Syntax-:

align-items: stretch (default) | center | flex-start | flex-end | baseline;

Property for Child element -:

v.flex-grow property

The flex-grow property specifies how much the item will grow relative to the rest of the flexible items inside the same container.

Syntax-:

flex-grow: number;

Default Value is 0.

---

vi.order Property

The order property specifies the order of a flexible item relative to the rest of the flexible items inside the same container.

Syntax-:

order: number;

Default Value is 0.

---

vii. align-self Property

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

Syntax-:

align-self: auto (default) | stretch | center | flex-start | flex-end | baseline;

==========

Media Queries :-

Media Query is a CSS technique used to apply different styles to a webpage based on the device's characteristics, such as screen width, height, resolution, or orientation, enabling responsive design.

Syntax:

```
@media media-type and (condition) {
    /* CSS rules here */
}
```

Example:

```
@media screen and (min-width: 768px) {
    body {
        background-color: lightblue;
    }
}
```

##Common Breakpoints for Responsive Design:

Device Type	Width (px)
Extra Small (xs)	< 576px
Small (sm)	≥ 576px
Medium (md)	≥ 768px
Large (lg)	≥ 992px
Extra Large (xl)	≥ 1200px
XXL	≥ 1400px

==========

10]Position Properties -:

The CSS position property is used to set position for an element. It is also used to place an element behind another and also useful for scripted animation effects.

You can position an element using the top, bottom, left and right properties.

These properties can be used only after the position property is set first. A position element’s computed position property is relative, absolute, fixed or sticky.

There are five different position values:

1.static

2.relative

3.fixed

4.absolute

5.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.

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.

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

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 | relative | absolute | fixed | sticky ;

top:20px;

bottom:20px;

left:20px;

right:20px;

==========

11] Z-index-:

The z-index in CSS allows us to define a visual hierarchy on the 3-dimensional plane, i.e., the z-axis. It is used to specify the stacking order of the positioned elements (elements whose position value is either fixed, absolute, relative, or sticky). The stacking order means that the element's position along the z-axis, which is perpendicular to the screen.The z-index property specifies the stack order of an element.An element with greater stack order is always in front of an element with a lower stack order.

Note: z-index only works on positioned elements (position: absolute, position:

relative, position: fixed, or position: sticky).

number: It means that the element’s stack level is set to the given value. It allows negative values.

auto: It means that the order of the stack is equivalent to the parent, i.e.,

default.

Syntax-:

z-index: number | auto;

==========

12]Overflow-:

The CSS overflow property specifies how to handle the content when it overflows its block level container.We know that every single element on a page is a rectangular box and the size, positioning and behavior of these boxes are controlled via CSS.The overflow property specifies what should happen if content overflows an element’s box.

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

Let’s take an example: If you don’t set the height of the box, it will grow as large as the content. But if you set a specific height or width of the box and the content inside cannot fit then what will happen. The CSS overflow property is used to overcome this problem. It specifies whether to clip content, render scroll bars, or just display content. Visible It specifies that overflow is not clipped. it renders outside the element’s box.this is a default value. hidden It specifies that the overflow is clipped, and rest of

the content will be invisible. scroll It specifies that the overflow is clipped, and a scroll bar is used to see the rest of the content. auto It specifies that if overflow is clipped, a scroll bar is needed to see the rest of the content.inherit It inherits the property from its parent element. initial It is used to set the property to its initial value.

Syntax-:

overflow-x

overflow-y

overflow:visible | scroll | hidden | auto;

==========

13] Cursor-:

It is used to define the type of mouse cursor when the mouse pointer is on the element. It allows us to specify the cursor type, which will be displayed to the user. When a user hovers on the link, then by default, the cursor transforms into the hand from a pointer.

Syntax-:

cursor: alias | all-scroll | auto (default) | cell | context-menu | col-resize | copy

| crosshair | default | e-resize | ew-resize | grab | grabbing | help | move | n resize | ne-resize | nesw-resize

 | ns-resize | nw-resize | nwse-resize | no-drop |

none | not-allowed | pointer | progress | row-resize | s-resize | se-resize | sw resize | text| vertical-text |

w-resize | wait | zoom-in | zoom-out ;

==========

14]Clip-path-:

This CSS property is used to create a clipping region and specifies the element's area that should be visible. The area inside the region will be visible, while the outside area is hidden. Anything outside the clipping will be clipped by browsers, including borders, text-shadows, and many more.It allows us to define a particular region of the element to display, instead of displaying the entire area.

It makes it easier to clip the basic shapes by using ellipse, circle, polygon, or inset

keywords.

Used to draw different shapes with using div.

Syntax-:

clip-path: polygon(100% 0, 50% 50%, 100% 100%, 41% 100%, 0 100%, 0 0);

==========

15]ANIMATION Properties:-

1]Transition-:

CSS transitions allows you to change property values smoothly, over a given duration.

Syntax-:

transition-duration: 4s;

transition-timing-function: linear ,ease , ease-in ,ease-out , ease-in-out;

transition-delay: 1s;

==========

2]Transform-:

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

Syntax-:

translate() :It moves an object from one position to another

translateX() : Moves an object in x-axis

translateY(): Moves an object in y-axis

translate(x,y) : Moves an object in both axis

Note : values in pixel.

scale():It scales an element

scaleX()

scaleY()

scale(existing_width X value , existing_height Y value)

scale(x,y)

Note:Values In No.s.(Count).

---

rotate(): It rotates an object

rotateX()

rotateY()

rotateZ() : clockwise & anti clockwise rotates

Note:Values In Degree (40deg).

---

skew(x-axis, y-axis)

skewX(value in Degree)

skewY(value in Degree) : For Template

Short Hand for Transform Property:-

transform:rotateZ(90deg) translate(280px) scale(2);

==========

3]Animation-:

Syntax-:

animation-name : Name of the Animation

animation-duration :Duration of Animation

animation-delay : delay for that object

animation-iteration-count : number | infinite [loop through display or for particular time]

animation-direction:reverse|alternate|normal

animation [shorthand] Property:-

animation: name duration timing-function delay iteration-count direction fill-mode ;

To create a animation using CSS first we need to create Frames

To create frames use "@keyframes & keyframe name"

Ex:

@keyframes moveIt

{

from{css properties}

to{css properties}

}

@keyframes moveIt

{

0%{css properties}

100%{css properties}

}

==========

16]Filter-:

Syntax-:

filter: none | blur(5px) | brightness(100%) | contrast(200%) | drop-shadow(8px 8px 10px gray) | grayscale(100%) | hue-rotate(90deg) | invert(100%) | opacity(30%) | saturate(8) | sepia(100%) ;

The filter property defines visual effects (like blur and saturation) to an element

(often <img>).

==========

17]Opacity-:

Syntax-:

opacity: number;

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.

==========
==========

Thank You