1. Expand CSS. What is the use of CSS?

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation and visual appearance of HTML (Hypertext Markup Language)

The primary use of CSS is to separate the content of a web page from its presentation. Instead of embedding styling information directly into the HTML markup, CSS allows developers to create a separate CSS file or include CSS rules within the HTML document.

● CSS is the language we use to style a Web page.

● CSS describes how HTML elements are to be displayed on screen.

—————————————————————————————————————

2. What are the uses of <div> and <span> tags when used with CSS.

The **<div>** tag is a block-level element that is commonly used to create divisions or sections in a web page layout. It acts as a container that can group and organize other HTML elements.

The <div> tag is often used to define different sections of a webpage, such as headers, footers, sidebars, and content areas.

The **<span>** tag is an inline-level element that is used to apply styles to a specific portion of text or inline content within a larger block of text.

the <span> tag is typically used for smaller, more specific elements or text within a larger context.

—————————————————————————————————————

3. List some of the tags to avoid from HTML with reason.

<font> tag: The <font> tag was widely used in older versions of HTML to specify font styles, colors, and sizes.

<center> tag: The <center> tag was used to center-align content horizontally within its parent container.

<strike> tag: The <strike> tag was used to create a strikethrough effect on text.

<frame> and <frameset> tags: These tags were used to create frames within a webpage, allowing for the division of a page into multiple independent sections.

All the above actions can be replaced via CSS.

4. With a neat diagram specify the main parts of a CSS style.

CSS Selector
CSS Properties
CSS Values

CSS Selector: The CSS selector determines which HTML element(s) the style should be applied to. It is used to target specific elements based on their tag name, class, ID, attributes, or relationships with other elements.

CSS Properties: CSS properties define the visual appearance and behavior of the selected HTML element(s). Examples of CSS properties include color, font-size, padding, border, background, and many more.

CSS Values: CSS values are the assigned values for the CSS properties.For example, a value of red can be assigned to the color property to make the text appear in red,

———————————————————————————————————

5. What are the ways we can use style sheets in an HTML document?
Mention the tags used to work with them. (5marks Q.3)

———————————————————————————————————

6. What is the use of <link> tag? Mention its attributes with their purpose.
The **<link> tag i**s an HTML element used to link external resources to an HTML document. It is commonly used to link external stylesheets, but it can also be used to link other resources such as JavaScript files, icon files, or alternative versions of the same document.
The <link> tag has several attributes that serve different purposes:

rel (relationship): the relationship between the linked resource and the current document.For stylesheets, the value of rel should be set to "**stylesheet**"

href (hypertext reference):the URL or path to the external resource being linked

type: specifies the MIME type of the linked resource. For stylesheets, the value of type should be set to "**text/css**".

Example : **<link rel="stylesheet" href="styles.css" type="text/css">**

7. What is inline style? Give a code example.

Inline Style:

Inline styles are applied directly to an HTML element using the style attribute. It is useful when you want to apply a style to only one specific element. Here is an Example: <p style="color: red; font-size: 20px;"> This text is red and 20 pixels</p>

----

8. List the different types of selectors in CSS.

a) Tag Selectors	Eg : p { color: blue; }
b) Class Selectors :	Eg : .highlight { background-color: yellow; }
c) ID Selectors :	Eg : #header { font-size: 24px; }
d) Attribute Selector :	Eg : input[type="text"] { border: 1px solid gray; }
e) Universal Selector (*) :	Eg : .banner * { font-weight: bold; }
f) Child selectors (>) :	Eg : ul > li { color: blue; }

----

9. What is a universal selector? What is its use?

The universal selector in CSS is denoted by an asterisk **(*).** It selects and applies styles to all elements within the document. It matches any element type, regardless of its tag name, class, or ID.

Here are some use cases for the universal selector: It is useful for

a) Resetting styles

b) Defining global styles

c) Selective overrides

d) Debugging during development.

----

10. Write a note on the :not selector.

The ':not()' Selector : targets the elements that do not match a particular selector. It takes a single argument, which is the selector for the elements that you want to exclude. Here's an example:

img:not(.logo) { border: 1px solid black; }

This selector selects all img tags on a page, except for those with the class '.logo'

---

11. Write a note on the two CSS commands required to use web fonts.

To use web fonts in CSS, there are two essential commands that you need to include in your CSS stylesheet or <style> tag:

1. @font-face : The @font-face rule is used to specify a custom font and define its properties.

Example : @font-face {

```
        font-family: 'Font1';
        src: url('font-file.woff2') format('woff2');
    }
```

2. font-family : The font-family property is used to specify the font family or list of fonts to be applied to HTML elements.

Example : body {

```
        font-family: Font1, sans-serif;
    }
```

---

12. Specify the properties required to make a font bold and italic.

font-weight: This property is used to specify the weight or thickness of the font. To make the font bold, you can set the font-weight property to bold.

font-style: This property is used to specify the style of the font, such as italic or normal. To make the font italic, you can set the font-style property to italic

```
font-weight: bold;
font-style: italic;
```

—————————————————————————————————————

13. How to specify the color to font? Mention different ways.

Here are some of the common methods:

Color Names : CSS provides a set of predefined color keywords that you can use to specify the color of a font.

Examples: color: red;, color: blue;, color: green;

RGB and RGBA : RGB color values specify the intensity of red, green, and blue color channels using decimal numbers ranging from 0 to 255.

RGBA color values are similar to RGB, but with an additional alpha channel representing the opacity or transparency.

Examples: color: rgb(255, 0, 0);   color: rgba(255, 0, 0, 0.5);

Hexadecimal : Hexadecimal color values represent colors using a combination of six characters (digits and letters) preceded by a hash (#) symbol.

Examples: color: #FF0000;

HSL and HSLA : HSL stands for Hue, Saturation, and Lightness. It allows you to specify colors using those three components. HSLA is similar to HSL, HSLA color values also include an alpha channel to represent transparency.

Examples: color: hsl(0, 100%, 50%);   color: hsla(0, 100%, 50%, 0.5);

—————————————————————————————————————

14. Write a note on how to capitalise the text in CSS.

In CSS, there are various ways to capitalize text, allowing you to control the capitalization style of elements. Here are some common methods:

i.) text-transform property with uppercase value:

Setting the text-transform property to uppercase capitalizes **all letters** in the text.

<u>Example:</u> text-transform: uppercase;

ii.) text-transform property with capitalize value:

The text-transform property is used to transform the case of the text within an element. By setting its value to capitalize, the **first character** of each word in the text will be capitalized.

<u>Example:</u> text-transform: capitalize;

---

15. Write a note on text-decoration property.

The text-decoration property in CSS is used to add visual decorations to text, such as **underlines, overlines, line-throughs**, and more. It allows you to style and customize the appearance of text decorations. Here are some key aspects to note about the **text-decoration** property:

<u>Values:</u>

none: No text decoration is applied.

underline: Adds a line underneath the text.

overline: Adds a line above the text.

line-through: Draws a line through the middle of the text.

underline overline: Adds both underline and overline to the text.

inherit: Inherits the text decoration from its parent element.

---

16. Write the purpose of each value of text-shadow property.

The text-shadow property in CSS allows you to add shadow effects to text. It creates a shadow behind the text, enhancing its visual appearance and creating depth. The text-shadow property accepts a set of values that define the properties of the shadow.

Syntax : **text-shadow : H-Offset V-Offset blurr-radius color;**

Here are the purposes of each value:

Horizontal Offset :

The first value specifies the horizontal offset of the shadow from the text.A positive value moves the shadow to the right, while a negative value moves it to the left.

Vertical Offset :

The second value determines the vertical offset of the shadow from the text.A positive value moves the shadow downwards, while a negative value moves it upwards.

Blur Radius :

The third value sets the blur radius, which controls the blurriness of the shadow's edges. A larger value creates a more blurred shadow, while a smaller value produces a sharper shadow.

Shadow Color :

The fourth value specifies the color of the text shadow.
You can use color names, hexadecimal values, RGB, or RGBA values to define the shadow color.

Example : text-shadow : 2px 2px 5px rgba(0, 0, 0, 0.5);

----

17. Write a note on aligning text using CSS.

Aligning text using CSS is a fundamental aspect of controlling the layout and presentation of text within HTML elements. Here's a note on aligning text using CSS:

text-align property:

The text-align property is used to horizontally align text within its containing element. Common **values** for text-align include:

left: Aligns the text to the left edge of the container.

right: Aligns the text to the right edge of the container.

center: Centers the text horizontally within the container.

justify: Adjusts the spacing between words to align both the left and right edges of the text to the container's edges.

Example : text-align: center;

---

18. Write a note on margin and padding shorthand.

Using the margin and padding shorthand allows you to set multiple values with a single line of code, simplifying the CSS and improving readability.

Margin Shorthand :- **margin : top right bottom left ;**

The margin shorthand property allows you to set the margins around an element in a single declaration. You can specify values for the top, right, bottom, and left margins in clockwise order.

Padding Shorthand :- **padding: top right bottom left ;**

The padding shorthand property enables you to set the padding within an element in a single declaration. You can specify values for the top, right, bottom, and left paddings in clockwise order.

Example: margin: 10px 20px 15px 30px;

padding: 10px 20px 15px 30px;

19. Differentiate inline and block boxes.

Inline boxes and block boxes are two display types in CSS that determine how elements are rendered and how they interact with other elements on a web page. Here's a differentiation between inline and block boxes:

Inline Boxes :

Inline boxes are primarily used for text and inline-level elements.
They **do not break** the flow of content and are **rendered within the same line**.
Examples of inline elements include **<span>, <a>, <em>, <strong>, <i>, and <img>**
Inline boxes **do not start on a new line** and can appear side by side within the same line.

Block Boxes :

Block boxes are used for block-level elements such as **<div>, <p>, <h1> to <h6>, <ul>, <li>, and <section>.**
Block boxes create a new block-level formatting context and **start on a new line,** causing subsequent content to appear below them.

----

20. Write a note on border property shorthand.

The border shorthand property in CSS allows you to set multiple aspects of an element's border in a single declaration.

Syntax : **border: width style color** ;

The border shorthand property consists of three individual sub-properties:

- **Border Width :** Specifies the width or thickness of the border
- **Border Style** : Determines the style of the border.

Values include **solid** (a continuous line), **dashed** (a series of short dashes), **dotted** (a series of small dots), **double** (two parallel lines), and **none** (no border).

- **Border Color :** Sets the color of the border.

Example : **border : 1px solid #FF0000;**

21. Write a note on providing background color in CSS with appropriate property.

In CSS, you can set the background color of an element using the background-color property. It allows you to specify a color value that fills the background area of the element. Here's a note on providing background color in CSS: Syntax : **background-color: color-value;**

Color Value: You can set the background color using **color names, hexadecimal values, RGB, or RGBA values.**

- Color names refer to predefined color names in CSS, such as "red," "blue," "green,"
- Hexadecimal values represent colors using a combination of six characters (digits and letters) preceded by a hash (#) symbol, such as #FF0000 for red.
- RGB values specify the intensity of red, green, and blue color channels using decimal numbers ranging from 0 to 255. The RGB values are defined inside the rgb() function, such as rgb(255, 0, 0) for red.
- RGBA values are similar to RGB, but with an additional alpha channel representing the opacity or transparency.

Examples:-

Using color names:	background-color : red;
Using hexadecimal values:	background-color : #00FF00;
Using RGB values:	background-color : rgb(0, 0, 255);
Using RGBA values:	background-color : rgba(255, 0, 0, 0.5);

----

22. Write a note on the box-shadow property of CSS.

We can add shadow effects to the text to make it pop from the page. CSS3 includes the box-shadow property to add drop shadows to an element's bounding box. The simple example for the box-shadow property is as follows:

Syntax :

box-shadow : **h-offset v-offset blur-radius spread-radius color**;

Example :

box-shadow: 4px 6px 8px 2px red ;

4px horizontal offset , **6px** vertical offset , **8px** blur and **2px** spread ,shadow color **red**

---

23. Write a note on float property with its options.

The float property in CSS is used to control the positioning and alignment of elements, allowing them to float to the left or right of their containing element. Here's a note on the float property and its options:

<u>Syntax:</u> **float : left | right | none | initial | inherit ;**

left: Floats the element to the left side of its containing element. Other content will wrap around it on the right side.

right: Floats the element to the right side of its containing element. Other content will wrap around it on the left side.

none: Default value. The element does not float, and content flows around it as normal.

initial: Sets the float property to its initial value.

inherit: Inherits the float property from its parent element.

---

24. What is animation? What are start and end states?

An animation is nothing more than a visualization of something changing over a period of time.CSS animations allow you to create animations using CSS properties and keyframes.

Start State: The start state represents the initial appearance or behaviour of the element before the animation begins.

End State: The end state represents the desired appearance or behaviour of the element when the animation is complete.

——

25. What do you mean by keyframes ?

In CSS, keyframes are used to define a set of styles or property changes that occur during an animation. Keyframes allow you to specify how an element should appear at various points in time during the animation, creating smooth and controlled transitions between those states.

Keyframes are defined using the **@keyframes** rule in CSS. The @keyframes rule specifies the name of the animation and defines a series of keyframe blocks, each representing a specific point in the animation timeline. Inside each keyframe block, you can define the styles or property values that should be applied to the element at that particular point in the animation. Eg:

```
@keyframes Change_color {
  0%    { background-color: red;    }
  50%   { background-color: blue;   }
  100% { background-color: green; }
}
```

——

26. Write the purpose of animation-direction property with its possible values.
The animation-direction property in CSS is used to define the direction of an animation sequence. It determines whether the animation should play forward and then reverse, or simply repeat in a loop. Here's the purpose of the animation-direction property along with its possible values:

Syntax : **animation-direction: normal** | **reverse** | **alternate** | **alternate-reverse** | **initial** | **inherit** ;

- **normal**: Default value. The animation plays forward from the beginning to the end, and then restarts.
- **reverse**: The animation plays in reverse, starting from the end and going back to the beginning, and then restarts.
- **alternate**: The animation plays forward from the beginning to the end, then reverses and plays backward from the end to the beginning. This creates a back-and-forth effect.
- **alternate-reverse:** Similar to alternate, but it starts by playing in reverse from the end to the beginning, then forward from the beginning to the end.
- **initial:** Sets the animation-direction property to its initial value.
- **inherit:** Inherits the animation-direction property from its parent element.

——

27. Write the syntax of animation property shorthand form.
The animation property in CSS allows you to define multiple animation-related properties in a single shorthand declaration.

The syntax for the short-hand animation property :
animation : name duration timing-function delay iteration-count direction fill-mode play-state;

- **name**: Specifies the name of the animation previously defined using @keyframes. Multiple animation names can be provided, separated by commas.
- **duration**: Sets the length of time the animation takes to complete, usually in seconds (s) or milliseconds (ms).
- **timing-function:** Defines the timing function for the animation, controlling the pace and acceleration of the animation. Common values include **linear, ease, ease-in, ease-out, ease-in-out, step-start, step-end, and more**.
- **delay:** Specifies the amount of time to wait before starting the animation, typically in seconds (s) or milliseconds (ms).
- **iteration-count:** Determines the number of times the animation should repeat. You can use a specific number (e.g., 2, 3), or keywords such as **infinite** for infinite repetition or initial to set the default value.
- **direction**: Sets the direction of the animation, controlling whether it plays forward, in reverse, or alternates between forward and reverse. Possible values are **normal, reverse, alternate, and alternate-reverse.**
- **fill-mode**: Defines how the animation styles are applied before and after the animation. Common values include none, forwards, backwards and both.
- **play-state:** Controls the play state of the animation, allowing you to pause or resume it. Possible values are **running** and **paused**.

---

28. What are the properties a CSS transition defines?
CSS transition defines the properties that are affected by the transition and specifies how those properties change over time. It allows you to create smooth and gradual animations between different states of an element. Here are the properties that a CSS transition defines:

Transitions Longhand Syntax:

```
    transition-property :  property ;
    transition-duration  :  duration ;
    transition-timing-function :  timing-function ;
    transition-delay  :  delay ;
```

Transition Property:
The transition-property property specifies which CSS properties should be animated during the transition.

Transition Duration:
The transition-duration property determines the duration of the transition effect. Typically in seconds (s) or milliseconds (ms).

Transition Timing Function:
The transition-timing-function property defines the pace and acceleration of the transition.
Common timing function values include linear, ease, ease-in, ease-out, ease-in-out, step-start, step-end, and more.

Transition Delay:
The transition-delay property sets the delay before the transition starts. Typically in seconds (s) or milliseconds (ms).

Example :

```
    .box {
        transition-property: width, background-color;
        transition-duration: 0.5s;
        transition-timing-function: ease-in-out;
        transition-delay: 0.2s;
    }
```

———————————————————————————————————

29. List the values for the transition timing function.

The transition-timing-function property in CSS allows you to control the pace and acceleration of a transition. It determines how intermediate property values are calculated during the transition period. Here is a list of common values for the transition-timing-function property:

- **linear** : The transition occurs at a consistent pace from start to end.
- **ease** : Starts slowly, accelerates through the middle, and slows down again towards the end. It is the default timing function.
- **ease-in** : Starts slowly and gradually accelerates towards the end of the transition.
- **ease-out** : Starts quickly and gradually decelerates towards the end of the transition.
- **ease-in-out** : Starts slowly, accelerates through the middle, and then decelerates towards the end. It combines the effects of ease-in and ease-out.
- **step-start or start** : Transitions abruptly to the new state at the start of the transition, without any intermediate values. It behaves like a step function.
- **step-end or end** : Maintains the initial state until the end of the transition, then transitions abruptly to the new state without intermediate values.
- **steps(n)** : Specifies a specific number of intervals (n) during the transition. The intermediate values are evenly spaced, creating a stepped appearance.
- **cubic-bezier(n,n,n,n)** : Allows you to create custom timing functions using cubic Bezier curves. You can define the control points (n) to manipulate the timing and acceleration.

----

1. List and explain the advantages of CSS.

The primary use of CSS is to separate the content of a web page from its presentation. Instead of embedding styling information directly into the HTML markup, CSS allows developers to create a separate CSS file or include CSS rules within the HTML document.

CSS, in contrast, offers the following advantages:

Separation of Concerns: CSS separates the presentation (styling) from the structure (HTML), making it easier to manage and update the visual aspects of a website without affecting its content.

Reusability: Styles defined in CSS can be reused across multiple elements, reducing code duplication and making development and maintenance more efficient.

Browser Compatibility: CSS is supported by all modern web browsers, ensuring consistent rendering of styles across different platforms.

More Formatting : Style sheets offer far more formatting choices than HTML. With CSS, we can format almost all the HTML Tags

Maintainability: By keeping styles separate from the HTML structure, CSS promotes code organization and makes it easier to update and maintain the visual aspects of a website.

Layout Control: CSS provides precise control over the layout and positioning of elements on a web page, allowing for responsive designs that adapt to different screen sizes and devices.

Animations: Using CSS Animation and Transitions we can have animation effects on the HTML Pages.

——————————————————————————————————————

2. Explain all style elements of CSS.

In CSS (Cascading Style Sheets), there are various style elements that can be used to define the appearance and formatting of HTML elements. These style elements allow you to control aspects such as color, layout, typography, borders, backgrounds, and more. Here's an explanation of some common style elements in CSS:

1. Color : The color property is used to specify the color of text, borders, and backgrounds. It can be defined using named colors (e.g., "**red**"), hexadecimal values (e.g., "**#FF0000**"), RGB values (e.g., "**rgb(255, 0, 0)**"), or HSL values (e.g., "hsl(0, 100%, 50%)").

2. Typography : CSS provides several properties to control the typography of text, including **font-family** (specifies the font), **font-size** (sets the size of the font), **font-weight** (controls the thickness of the font), **font-style** (specifies italic or oblique style), and **text-decoration** (adds underline, overline, or line-through to text).

3. Box Model : The box model properties define the structure and spacing of elements. They include content (the actual element's content), **padding** (space between the content and the element's border), **border** (line surrounding the padding), and **margin** (space between the element and other elements).

4. Background : CSS provides properties to control the background of elements. You can set the background color using the **background-color** property, add an image with **background-image**, control the position and repetition of the background image using background-position and background-repeat, respectively.

5. Borders: The border properties allow you to define the style, width, and color of element borders. Some commonly used properties include **border-style** (e.g., solid, dashed), **border-width** (sets the thickness of the border), and **border-color** (defines the color of the border).

6. Transitions and Animations : CSS offers properties to create smooth transitions and animations between different states of an element. Transition properties like **transition-property, transition-duration**, **transition-timing-function, and transition-delay** control the timing and effect of the transition. Animation properties, such as **animation-name, animation-duration, animation-timing-function, and animation-delay,** enable the creation of more complex animations.

7. Positioning : CSS provides properties to precisely position elements on a page. **position** property determines the positioning method (e.g., **relative, absolute, fixed**), while top, bottom, left, and right properties specify the position offsets from the element's normal position.

——

3. Explain different ways of linking stylesheets in HTML5.

Here are the different ways to link stylesheets in HTML5:

1. Inline Style:

Inline styles are applied directly to an HTML element using the style attribute. It is useful when you want to apply a style to only one specific element. Here is an example: <p style="color: red; font-size: 20px;">This text is red and 20 pixels</p>

2. Internal or Embedded StyleSheet :

An internal stylesheet is a collection of styles that's part of the web page's code. It always appears between opening and closing HTML <style> tags in the page's <head> portion. Here's an example:

```
<head>  <style>
   p {   color: red;
    font-size: 18px;   }      </style>  </head>
```

3. External StyleSheets :

An external stylesheet is nothing more than a text file containing all your CSS rules. It never contains any HTML code. So it doesn't include the <style> tag. In addition, files will be saved with the extension '.css' . We can attach a style sheet to a web page two methods :

a) **Linking a StyleSheet by Using HTML <link> tag**
b) **Linking a StyleSheet by Using CSS @import directive**

a) Linking a StyleSheet by Using HTML : The most common method of adding an external stylesheet to a web page is to use the HTML <link> tag.

Eg : Using <link> tag : <link rel="stylesheet" href="css/styles.css">

● rel="stylesheet" indicates the type of link—in this case, a link to a style sheet.

● href points to the location of the external CSS file

b) Linking a StyleSheet by Using CSS : CSS includes a built-in way to add external stylesheets using **@import** directive. You add the directive inside of an HTML <style> tag .

Eg : <style>@import url(css/styles.css); @import url(css/form.css);
</style>

———

4. Explain the steps of creating an internal style sheet.

Creating an internal style sheet involves adding CSS rules directly within the <style> element in the <head> section of an HTML document. This method allows you to define styles specific to that particular HTML page. Here are the steps to create an internal style sheet:

Open an HTML document: Start by creating or opening an existing HTML document using a text editor or an HTML editor.

Insert the <style> element: In the <head> section of your HTML document, insert the <style> element. This element is used to contain your CSS rules.

Write CSS rules: Within the <style> element, you can write your CSS rules. These rules define how the HTML elements should be styled. For example, to set the color of all paragraphs to blue, you would write:

```
<head>
  <style>
    p {
      color: blue;
    }
  </style>
</head>
<p> Hello Internal CSS added paragraph </p>
```

——

5. Explain the steps of creating an external style sheet.

Creating an external style sheet involves creating a separate CSS file that contains all your CSS rules and linking it to your HTML document. This method allows you to reuse the same styles across multiple HTML pages. Here are the steps to create an external style sheet:

Create a new CSS file: Start by creating a new text file using a text editor or an HTML/CSS editor. Save the file with a .css extension, such as "**styles.css**". It is recommended to keep your CSS file separate from your HTML files for better organization.

Write CSS rules: Open the CSS file and start writing your CSS rules. Each rule should be written in the format **selector { property: value; }**. For example :

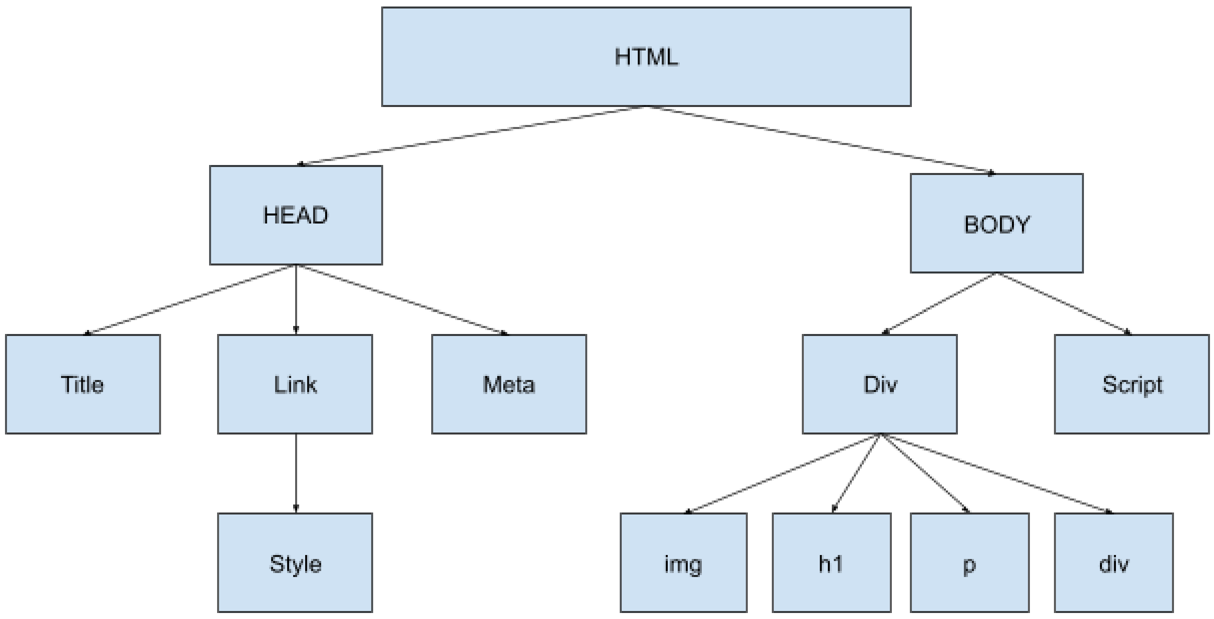
```
/* styles.css */
p {
  color: blue;
}
```

Link the CSS file to your HTML document: In your HTML document, in the <head> section, add a <link> element to link the external CSS file. The **<link>** element should have the rel attribute set to **"stylesheet"** and the href attribute pointing to the location of the CSS file. For example:

```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

——

6. Explain the relationships between HTML tags with the help of a representative diagram.



<u>Explanation of the relationships:</u>

- The **root element is <html>**, which encloses the entire HTML document.
- Within the <html> element, there are two main sections: **<head> and <body>.**
- The **<head>** element contains meta-information about the document, such as the page title (<title>), meta tags (<meta>), and external resources like stylesheets (<link>) and scripts (<script>).
- The **<body>** element contains the visible content of the webpage, including headings (<h1>), paragraphs (<p>), and other HTML elements.
- Additionally, the <head> and <body> elements can contain other elements, such as <style> for internal styles and <script> for embedded or external scripts.
- The **<style>** element is used to define CSS rules for styling the HTML elements within the document. The **<script>** element is used to embed or reference JavaScript code for interactive functionality.

——————————————————————————————————————

7. Explain the pseudo-classes used to style links.

Pseudo-classes in CSS are used to select and style elements based on their state or position within the document structure. When it comes to styling links, there are several pseudo-classes available that allow you to target and apply specific styles to different states of links.

Here's an explanation of the commonly used pseudo-classes for styling links:

1) :link:

The :link pseudo-class selects and styles the links that have not been visited.

<u>Example :</u> a:link { color: blue; }

2) :visited:

The :visited pseudo-class selects and styles the links that have been visited.

<u>Example :</u> a:visited { color: purple; }

3) :hover:

The :hover pseudo-class selects and styles the links when the mouse pointer hovers over them.

<u>Example :</u> a:hover { color: red; }

4) :active:

The :active pseudo-class selects and styles the links when they are being activated or clicked.

<u>Example :</u> a:active { color: green; }

5) :focus:

The :focus pseudo-class selects and styles the links when they receive focus.

<u>Example :</u> a:focus { outline: 2px solid yellow; }

8.Explain :focus, :before, :after ::selection pseudo-classes.

The :focus pseudo-class works much like the :hover pseudo-class. While :hover applies when a visitor mouses over a element, and focus is like when the element gets focus Ex:

```
input:focus { border: 2px solid blue; }
```

The :before pseudo-class, It lets you add content preceding a given element.

Ex :
```
h1::before { content: " This is added at the beginning "; }
```

Here it adds above content before all <h1> elements

The :after pseudo-class,Exactly like the :before selector, the :after pseudo-element adds generated content—but after the element, not before. You can use this selector, for example,

```
h1::after { content: " This is added after h1 tags content"; }
```

The ::selection pseudo-element, refers to items that a visitor has selected on a page. For example, when a visitor clicks and drags over text, the browser highlights that text, and the visitor can then copy the text.Normally, browsers add a blue background behind the text. We can change them as below

```
::selection { background-color: red; }
```

9.Explain attribute selectors with example code

Attribute selectors are a type of CSS selector that allow you to target HTML elements based on the presence or value of an attribute. They are denoted using square brackets [ ] containing the name of the attribute and optionally, a value or operator. Here are some examples of attribute selectors:

[attribute] : targets elements that have the specified attribute, regardless of its value.

```
img[alt] { border: 4px solid black; }
```

[attribute=value] : targets elements that have the specified attribute with the exact value.

```
input[type="submit"] { background-color: green; }
```

[attribute*=value] : targets elements that have the specified attribute with a value containing the specified substring.

```
Eg: a[href*="co.in"] { color: red; }
```

[attribute^=value] : targets tags that have the specified attribute with a value starting with the specified string.

```
Eg: a[href^="http://"] { font-weight: bold; }
```

[attribute$=value] : targets elements that have the specified attribute with a value ending with the specified string.

```
Eg: a[href$=".pdf"] { background-color: yellow; }
```

10.Explain child selectors with examples.

Child selectors are a type of CSS selector that allow you to target elements that are direct children of a specific parent element. They are denoted using the '>' symbol, which is placed between the parent element and the child element. Here is an example of a child selector:

Eg : ul > li { color: blue; }

CSS3 also includes some very specific pseudo-classes for selecting child elements :

:first-child: targets the first child element of its parent. (alternate ':first-of-type')

Eg : ul li:first-child { font-weight: bold; }

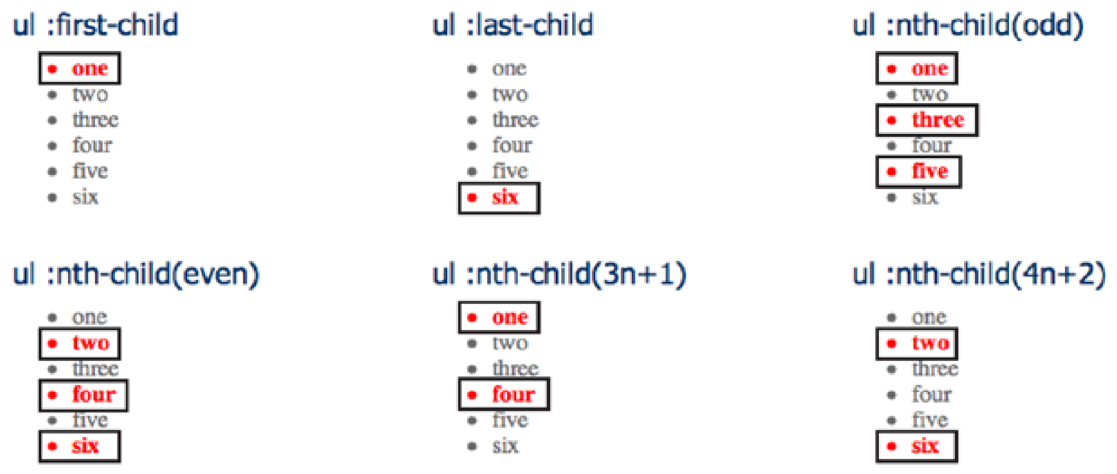
:last-child: targets the last child element of its parent. (alternate ':last-of-type')

Eg : ul li:last-child { color: red; }

:nth-child(n): targets the n-th child element of its parent, where n can be a number, a keyword, or a formula.(alternate ':nth-of-type')

```
Eg :    ul li:nth-child(odd) { background-color: lightgray; }
        ul li:nth-child(even) { background-color: red; }
        ul li:nth-child(3n+2) { background-color: green; }
```

Here, Odd means only odd li elements will be selected similarly Even will choose even li elements And 3n means every third element, while + 2 indicates which element to start at.



11. Explain child type selectors with code.

The child type selector is represented by the **> symbol** and is placed between the parent element and the child element.

Syntax: **parent > child**

For example, suppose we have the following HTML structure:

```
<style>
.parent > p {
  color : blue;
 }
</style>

<div class="parent">
        <h2>Heading</h2>
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
        <ul>
                <li>List item 1</li>
                <li>List item 2</li>
        </ul>
</div>
```

In this example, the **>** selector targets the **<p>** elements that are immediate children of the **.parent** element and applies the **color : blue;** style to them. The **<p>** elements inside the **<ul>** element are not affected by this selector.

Similarly, you can use child type selectors to target other types of elements, such as headings, lists, or any other elements that are direct children of a specific parent.

——

12. Explain how to use the @font-face directive.

The @font-face directive in CSS is used to define and use custom fonts on a web page.Here's an explanation of how to use the @font-face directive:

- Inside your CSS file, use the **@font-face** rule to declare the font and its properties.
- Start with the **@font-face keyword** followed by opening and closing curly braces **{ }**.
- Inside the curly braces, specify the font properties using the following descriptors:
    - ➢ **font-family :** Assign a name for the custom font.
    - ➢ **src :** Specify the path or URL of the font file(s) using the url() function. Provide multiple sources with different formats to ensure broader browser support.
    - ➢ **font-weight :** Set the font weight (e.g., normal, bold).
    - ➢ **font-style :** Set the font style (e.g., normal, italic).

<u>Example:</u>

```
@font-face {
        font-family : 'Font1';
        src : url('customfont.woff2') format('woff2');
        font-weight : normal;
        font-style : normal;
}
```

——

13. Explain filters used in searching google web fonts.

Google Fonts offers various filters to help users find and select the most suitable fonts for their projects. These filters allow you to refine your search based on specific criteria. Here are the key filters commonly used on Google Fonts:

Category : Fonts are classified into different categories such as serif, sans-serif, display, handwriting, monospace, and more. This filter helps you narrow down your search based on the style of the font you're looking for.

Language : Google Fonts supports a wide range of languages. By selecting a specific language or multiple languages, you can find fonts that include characters and glyphs needed for those languages.

Thickness : This filter allows you to adjust the weight or thickness of the fonts. You can choose from options like thin, light, regular, bold, and black to match the desired visual impact of your text.

Width : Fonts come in various widths, and this filter enables you to specify the desired width of the fonts, ranging from ultra-condensed to ultra-expanded.

Slant : Some fonts have an italic or oblique version available. You can use this filter to find fonts with specific slant styles.

Size : Google Fonts provides a size slider that allows you to adjust the visual size of the font preview on the website. This is particularly useful for comparing different fonts at different sizes.

Thickness/Slant Axes : Google Fonts also offers an advanced feature called "Variable Fonts," which allows you to adjust multiple parameters simultaneously. The thickness and slant axes enable you to customize the weight and slant of the fonts dynamically.

Sort : You can sort the search results based on different criteria, such as popularity, date added, trending, and alphabetical order. Sorting can help you discover popular or recently added fonts.

---

14. Explain the different possible ways to set color property value with proper examples.

Color is a property that allows you to control the color of text, backgrounds, and other elements on a web page. It offers various ways to specify colors, including predefined color names, hexadecimal codes, RGB values, and HSL values.

Named Colors : CSS provides a set of predefined color names that you can use directly. For example : color: red;   color: blue;

Hexadecimal Notation :

You can specify colors using a six-digit hexadecimal value preceded by a hash symbol (#). Each pair of digits represents the intensity of red, green, and blue (RGB) channels. For example : color: #FF0000; color: #0000FF;

RGB Notation :

RGB values allow you to specify the intensity of red, green, and blue channels using decimal values from 0 to 255.   For example : color: rgb(255, 0, 0);

RGBA Notation :

RGBA values are similar to RGB, but they include an additional parameter for opacity (alpha) expressed as a decimal between 0 and 1.

For example : color: rgba(255, 0, 0, 0.5);

HSL Notation :

HSL (Hue, Saturation, Lightness) values allow you to specify colors based on their hue, saturation, and lightness. Hue is represented as an angle from 0 to 360 degrees, while saturation and lightness are percentages.

For example : color: hsl(0, 100%, 50%);

HSLA Notation :

Similar to HSL, HSLA values include an additional parameter for opacity (alpha) expressed as a decimal between 0 and 1.

For example : color: hsla(0, 100%, 50%, 0.5);

---

15. Explain different ways to set font-size property value.

In CSS, the font-size property is used to specify the size of the font in an element. There are various ways to set the font-size property value. Here are the different methods:

Absolute Units:

Absolute units define the font size using fixed measurements like pixels (px), inches (in), centimeters (cm), millimeters (mm), or points (pt).

For example : font-size: 16px;
font-size: 1in;
font-size: 2cm;
font-size: 12pt;

Relative Units:

Relative units allow you to define the font size relative to other elements or the default font size of the browser. Relative units include em, rem, vw.

For example : font-size: 1em; // Relative to the parent element's font size
font-size: 2rem; // Relative to the root element's (html) font size
font-size: 50vw; // Relative to the viewport width

Percentage:

Font size can be specified as a percentage relative to the parent element's font size. For example : font-size: 150%;

Keyword Values:

CSS also provides keyword values for font size, such as small, medium, large, x-large, xx-large, smaller, and larger. The actual size of these keywords may vary depending on the user agent's default font size.

For Example: font-size: large;

---

16. Explain the styling of lists using CSS.

CSS list properties are used to style and customize the appearance of lists on a web page. Here are some commonly used CSS list properties along with examples:

1. list-style-type : Specifies the type of marker or bullet used for list items. It can be set to various values such as **none, disc, circle, square, decimal, lowercase letters, uppercase letters, and more.**

For example : ul { list-style-type: square; }

This sets the marker for unordered lists (<ul>) to square shapes.

2. list-style-position : Determines the position of the marker or bullet relative to the list item text. It can be set to inside or outside.

For example : ol { list-style-position: inside; }

This positions the numbering marker inside the list items of ordered lists (<ol>).

3. list-style-image : Allows you to use a custom image as the marker or bullet for list items. It specifies the URL of the image file.

For example : ul { list-style-image: url('bullet.png'); }

This sets a custom bullet image for unordered lists, using the image file "bullet.png".

4. list-style : A shorthand property that combines the three list properties (list-style-type, list-style-position, and list-style-image) into a single declaration.

For example : ul { list-style: square inside url('bullet.png'); }

This sets the marker for unordered lists to square shapes, positions it inside the list items, and uses a custom bullet image.

5. List Item Padding : You can adjust the spacing between the marker/bullet and the content of the list items using the padding property. This property applies to the entire list item.

For example : li { padding: 5px; }

Adds padding around list items

---

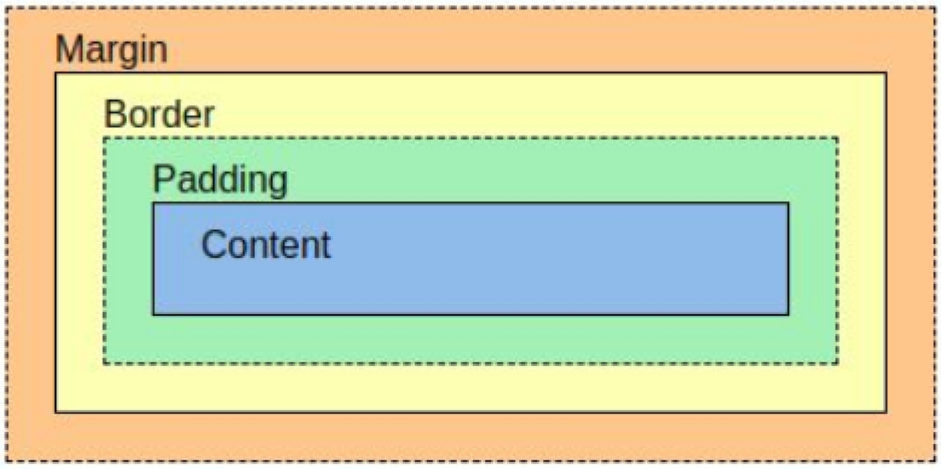
17. Explain the properties which make up a Box model.

To a browser, any tag is a box with something inside it—text, an image, or even other tags containing other things, Surrounding the content are different properties that make up the box:

a) padding : padding is the space between the content and the content's border. Padding is what separates a photo from the border that frames the photo. The properties that control the padding are: **padding-top, padding-right, padding-bottom, padding-left**. You can also use the shorthand padding property to set all four sides simultaneously.

b) border : border is the line that's drawn around each edge of the box. You can have a border around all four sides, on just a single side, or any combination of sides. It can have different styles, colors, and widths. The properties related to the border are : **border-width, border-style, border-color, border-radius.**

c) margin : margin is what separates one tag from another. The space that commonly appears between the tops and bottoms of paragraphs of text on a web page. It affects the positioning and spacing between elements. The margin properties are : **margin-top, margin-right, margin-bottom, margin-left.** You can also use the shorthand margin property to set all four sides simultaneously.



----

18. Explain about formatting individual borders with proper code examples.

In CSS, you can format individual borders of an element using specific properties that allow you to control the style, color, and width of each border independently. Here are the properties you can use to format individual borders:

i) border-top :
The border-top property allows you to set the style, color, and width of the top border of an element.

ii) border-right :
The border-right property sets the style, color, and width of the right border of an element.

iii) border-bottom :
The border-bottom property specifies the style, color, and width of the bottom border of an element.

iv) border-left :
The border-left property controls the style, color, and width of the left border of an element.

For Example :

```
border-top:    1px solid red;
border-right:  2px dashed blue;
border-bottom: 3px dotted green;
border-left:   4px double orange;
```

You can also use the **shorthand** border property to set all individual borders at once. The values for each border can be specified in **clockwise order** starting from the top border:

For Example :

```
border-width : 1px 2px 3px 4px;          // Top, Right, Bottom, Left
border-color : red blue green orange;    // Top, Right, Bottom, Left
```

---

19. Explain different ways of creating rounded corners in CSS with proper code examples.

To create rounded corners in CSS, there are multiple approaches you can take, each with its own set of properties and techniques. Here are three common ways of creating rounded corners in CSS with proper code examples:

border-radius Property:
The border-radius property allows you to specify the radius of the corners of an element. You can apply it to the element's border or background.
Here's an example:

```
.rounded-element {
        border-radius : 10px; // Applies rounded corners with a 10px radius
}
```

border-radius with Individual Values:
The border-radius property can also accept individual values for each corner. This allows you to create different radius sizes for each corner.

Example:
```
.rounded-element {
  border-radius :  10px  5px  20px   15px;
                        // top-left, top-right, bottom-right, bottom-left
}
```

border-radius with Percentage Values:
Instead of using pixels, you can also specify the border radius using percentage values relative to the element's size.

Example:
```
.rounded-element {
          border-radius : 50%;
     // Creates a perfect circle by setting the radius to 50%
}
```

---

20. Explain different options for box-sizing property in CSS3.

In CSS3, the box-sizing property allows you to control how the total width and height of an element are calculated, taking into account the content, padding, and border. The box-sizing property has three possible values:

content-box (Default) :
This is the default value, where the width and height of an element are calculated excluding the padding and border. In this mode, any padding and border are added to the specified width and height.

Example:
```
.element {
        box-sizing : content-box;
}
```

border-box :
With the border-box value, the width and height of an element include the content, padding, and border. In this mode, any padding and border are subtracted from the specified width and height. This makes it easier to create elements with fixed dimensions since you don't need to account for padding and border separately.

Example:
```
.element {
        box-sizing : border-box;
}
```

inherit :
The inherit value allows the box-sizing property to inherit the value from its parent element.

Example:
```
.element {
        box-sizing : inherit;
}
```

—————————————————————————————————————

21. Explain overflow property of CSS3 with all of its options.

The overflow property in CSS3 controls how content that exceeds the size of an element is displayed. It has several options that determine the behaviour of overflowing content. Here are the different options of the overflow property:

i) overflow : visible (Default)
This is the default value where the overflowing content is not clipped and extends beyond the boundaries of the element. It may overlap other elements.

ii) overflow : hidden
The hidden value hides any overflowing content and clips it within the boundaries of the element. The content that exceeds the dimensions of the element is not visible.

iii) overflow : scroll
The scroll value adds a scrollbar to the element, allowing users to scroll and view the overflowing content. The scrollbar appears regardless of whether there is content to scroll or not.

iv) overflow : auto
The auto value adds a scrollbar to the element only when the content overflows. If the content fits within the element, no scrollbar is displayed. It dynamically shows/hides the scrollbar based on the presence of overflowed content.

v) overflow : overlay
The overlay value is similar to auto, but it adds a scrollbar only when the content exceeds the dimensions of the element. It avoids taking up unnecessary space for the scrollbar when it's not needed.

vi) overflow : inherit
The inherit value allows the overflow property to inherit the value from its parent element.

———————————————————————————————————————

22. Explain clear properties of CSS3 with all its options.

In CSS3, the clear property is used to control the behaviour of floated elements and specifies whether an element should be placed next to or below the floated elements. It has several options that define how an element clears floated elements. Here are the different options of the clear property values:

a) none (Default) : The none value allows an element to be placed next to any floated elements. It does not clear any floated elements.

b) left : The left value clears any floated elements that are positioned to the left. It ensures that the element is placed below any floated elements on the left side.

c) right : The right value clears any floated elements that are positioned to the right. It ensures that the element is placed below any floated elements on the right side.

d) both : The both value clears any floated elements positioned to the left or right. It ensures that the element is placed below any floated elements on both sides.

e) inline-start : The inline-start value clears any floated elements positioned at the start of the block direction. It ensures that the element is placed below any floated elements on the start side (left in LTR, right in RTL).

f) inline-end : The inline-end value clears any floated elements positioned at the end of the block direction. It ensures that the element is placed below any floated elements on the end side (right in LTR, left in RTL).

The clear property is commonly **used in combination with floated elements** to control the layout and positioning of elements within a document flow. By using these options, you can ensure proper clearing of floated elements and achieve the desired layout behaviour.

---

23. Briefly explain the CSS animation with an example.

CSS Animations are the traditional animations that are on some sort of performance enhancing substance that makes them more awesome. With these kinds of animations, you can define not only the beginning and the end state but also any intermediate states commonly (and affectionately!) known as keyframes:

CSS animations allow you to create dynamic and visually appealing effects on web pages. They involve defining keyframes that specify intermediate states of an element's properties over a duration. CSS animations are controlled using animation properties and can be triggered by adding CSS classes or using JavaScript.

To create a simple CSS animation, you need to define keyframes that specify the intermediate states of an element's properties over time. Here are the steps to create a basic animation :

I. Define the Keyframes
II. Apply the Animation
III. Trigger the Animation

Example :

```
<!DOCTYPE html>
<html><head>
<style>
  @keyframes myAnimation {
     0%  { background-color: red; }
    50%  { background-color: yellow; }
    100% { background-color: blue; }
  }
  .box {
    width: 100px;
    height: 100px;
```

```
      background-color: red;
      animation-name: myAnimation;
      animation-duration: 3s;
      animation-iteration-count: infinite;
    }
  </style>
</head>
<body>
  <div class="box"></div>
</body>
</html>
```

In this example, we define a CSS animation called myAnimation using the @keyframes rule. The animation consists of three keyframes: at 0% (start), the background color is red; at 50%, it transitions to yellow, and at 100% (end), it becomes blue. Which will be done in 3 seconds and it repeats the same due to iteration count :infinite

—---

24. Explain all the animation related properties.

CSS provides several animation-related properties that allow you to create dynamic and interactive animations on elements.

The **syntax** for the short-hand animation property is as follows :

animation: animationName duration timing-function delay iteration-count direction fill-mode play-state;

Here is an explanation of the key animation-related properties:

1) animation-name :
Specifies the name of the animation defined using @keyframes. Multiple animations can be separated by commas.

2) animation-duration :
Sets the length of time that the animation takes to complete. You can specify the duration using seconds (s) or milliseconds (ms)

3) animation-timing-function :
Specifies the timing function that defines the pace of the animation. It can be linear, ease, ease-in, ease-out, ease-in-out, or a custom cubic-bezier function.

4) animation-delay :
Sets a delay before the animation starts. You can specify the delay using seconds (s) or milliseconds (ms).

5) animation-iteration-count :
Determines the number of times the animation should repeat. Values can be a positive integer, infinite, or alternate (which
reverses the animation on each iteration).

6) animation-direction :

Determines whether the animation should play forwards, backwards, or alternate between the two. It can be normal, reverse, alternate, or alternate-reverse.

7) animation-fill-mode :

Specifies how the element should be styled before and after the animation. It can be none, forwards, backwards, or both.

8) animation-play-state :

Determines whether the animation is running or paused.

This property allows you to toggle whether your animation plays or not by reacting to the running value or the paused value.

—--

25. Explain all the possible values for animation-fill-mode property.

The animation-fill-mode property in CSS specifies how an element should be **styled before and after an animation is played.** It controls the animation's effect on the element's styling. Here are the possible values for the animation-fill-mode property:

i) none (Default) :
This is the default value where no styles are applied to the element before or after the animation

ii) forwards :
The **forwards** value retains the styles of the last keyframe after the animation completes. The element's styles remain the same as the final state of the animation.

iii) backwards :
The **backwards** value applies the styles of the first keyframe before the animation starts. The element takes on the styles of the initial keyframe even before the animation begins.

iv) both :
The **both** value combines the effects of both forwards and backwards. It applies the styles of the first keyframe before the animation starts and retains the styles of the last keyframe after the animation completes.

The animation-fill-mode property allows you to control the styling of elements **before and after animations**, ensuring a desired appearance and behavior. By choosing the appropriate value, you can create smoother transitions and achieve the desired visual effects in your CSS animations.

---

26. Briefly explain the CSS Transition with an example.

CSS transitions provide a way to smoothly animate changes in CSS property values over a specified duration. They allow for gradual transitions between different property states, providing a smooth and visually appealing effect. Transitions can be applied to various CSS properties such as color, size, position, opacity, and more.

CSS Transitions: Syntax

Shorthand :-

```
    transition : property duration timing-function delay ;
```

Longhand :-

```
    transition-property : property ;
    transition-duration : duration ;
    transition-timing-function : timing-function ;
    transition-delay : delay ;
```

The following example shows a 100px * 100px red <div> element. The <div> element has also specified a transition effect for the width property, with a duration of 2 seconds:

```
    div {
        width: 100px;
        height: 100px;
        background: red;
        transition: width 2s; }
```

The transition effect will start when the specified CSS property (width) changes value. Now, let us specify a new value for the width property when a user mouses over the <div > element:

```
        div:hover {
            width: 300px;
        }
```

Notice that when the cursor mouses out of the element, it will gradually change back to its original style.

Refer Canvas Text Book PDF for the Same : Page 21 to 29

CANVAS CHEET SHEET

Drawing Methods

There are only 3 methods to draw directly on the canvas:

Method	Description
fillRect()	Draws a "filled" rectangle
strokeRect()	Draws a rectangle (with no fill)
clearRect()	Clears specified pixels within a rectangle

Path Methods

Method	Description
beginPath()	Begins a new path or resets the current path
closePath()	Adds a line to the path from the current point to the start
isPointInPath()	Returns true if the specified point is in the current path
moveTo()	Moves the path to a point in the canvas (without drawing)
lineTo()	Adds a line to the the path
fill()	Fills the current path
rect()	Adds a rectangle to the path
stroke()	Draws the current path
	Circles and Curves
bezierCurveTo()	Adds a cubic Bézier curve to the path
arc()	Adds an arc/curve (circle, or parts of a circle) to the path
arcTo()	Adds an arc/curve between two tangents to the path
quadraticCurveTo()	Adds a quadratic Bézier curve to the path

Text

Method/Prop	Description
direction	Sets or returns the direction used to draw text
fillText()	Draws "filled" text on the canvas
font	Sets or returns the font properties for text content
measureText()	Returns an object that contains the width of the specified text
strokeText()	Draws text on the canvas
textAlign	Sets or returns the alignment for text content
textBaseline	Sets or returns the text baseline used when drawing text

Colors, Styles, and Shadows

Method/Property	Description
addColorStop()	Specifies the colors and stop positions in a gradient object
createLinearGradient()	Creates a linear gradient (to use on canvas content)
createPattern()	Repeats a specified element in the specified direction
createRadialGradient()	Creates a radial/circular gradient (to use on canvas content)
fillStyle	Sets or returns the color, gradient, or pattern used to fill the drawing
lineCap	Sets or returns the style of the end caps for a line
lineJoin	Sets or returns the type of corner created, when two lines meet
lineWidth	Sets or returns the current line width
miterLimit	Sets or returns the maximum miter length
shadowBlur	Sets or returns the blur level for shadows
shadowColor	Sets or returns the color to use for shadows
shadowOffsetX	Sets or returns the horizontal distance of the shadow from the shape
shadowOffsetY	Sets or returns the vertical distance of the shadow from the shape
strokeStyle	Sets or returns the color, gradient, or pattern used for strokes

Transformations

Method	Description
scale()	Scales the current drawing bigger or smaller
rotate()	Rotates the current drawing
translate()	Remaps the (0,0) position on the canvas
transform()	Replaces the current transformation matrix for the drawing
setTransform()	Resets the current transform to the identity matrix. Then runs transform()

Compositing

Property	Description
globalAlpha	Sets or returns the current alpha or transparency value of the drawing
globalCompositeOperation	Sets or returns how a new image are drawn onto an existing image