G1 CSS Interview Question

Sr.No	Topic	Number of Questions
1	Basic CSS	10
2	Selectors	4
3	CSS Box Model	7
4	Layout	8
5	Typography	7
6	Colors and Backgrounds	G
7	CSS Transition	5
8	CSS Animation	5
G	CSS Transform	10
10	CSS Flex	8
11	Grid Layout	10
12	Responsive Design	8
	Total	G1

Here are 99 CSS questions that cover a wide range of topics, including theory, concepts, and practical questions. These questions will help you prepare effectively for an interview.

Basic CSS

Q.1] What does CSS stand for?

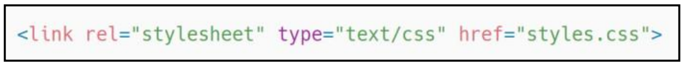
- CSS stands for Cascading Style Sheets.

Q.2] What is the purpose of CSS?

- CSS is used to style and format the appearance of web pages written in HTML and XML.
- It allows web developers to control aspects like **layout**, **colors**, **fonts**, and **spacing** of HTML elements.

Q.3] How do you link a CSS file to an HTML document?

- You can link a CSS file to an HTML document using the `<link>` element within the `<head>` section of the HTML document.

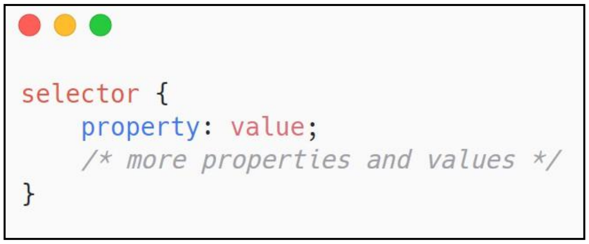


```
<link rel="stylesheet" type="text/css" href="styles.css">
```

- **`rel`** – It is used to indicate different relationships between the current document and the linked resource. Some common values are
  - stylesheet
  - icon
  - preload
  - alternate
- **`type`** – The `type` attribute specifies the MIME type of the linked resource.

Q.4] What is the syntax of a CSS rule?

- A CSS rule consists of a **selector** and a **declaration block.**



```
selector {
    property: value;
    /* more properties and values */
}
```

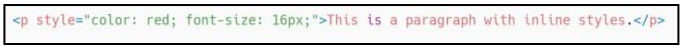
Q.5] What are the different ways to apply CSS to a web page?

CSS can be applied in three ways:

1. **Inline styles**: Directly within an HTML element.
2. **Internal stylesheets**: Within a <style> element in the <head> section of an HTML document.
3. **External stylesheets**: inked to an HTML document using the <link> element.

Q.6] What is an inline style?

- Inline styles are **CSS** styles applied directly to an individual HTML element using the style attribute



```
<p style="color: red; font-size: 16px;">This is a paragraph with inline styles.</p>
```

Q.7] What is an internal stylesheet?

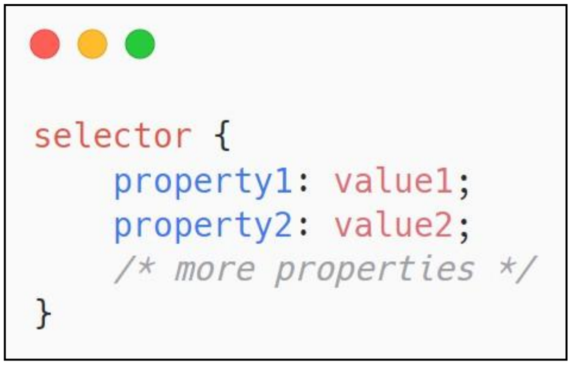
- An internal stylesheet is CSS code written within a `<style>` element in the `<head>` section of an HTML document.
- It applies styles to that particular HTML document only.

Q.8] What is an external stylesheet?

- An external stylesheet is a separate CSS file linked to an HTML document using the `<link>` element.
- It allows styles to be applied across multiple HTML documents.

Q.G] How do you apply multiple styles to a single element?

- Multiple styles can be applied to a single element by separating each style declaration with a semicolon within the style attribute (for inline styles) or within the CSS rule:

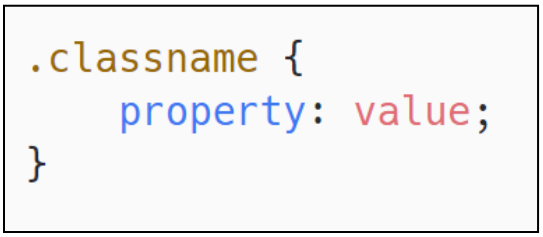


```
selector {
    property1: value1;
    property2: value2;
    /* more properties */
}
```

Q.10] What is the difference between a class selector and an ID selector?

Class selector

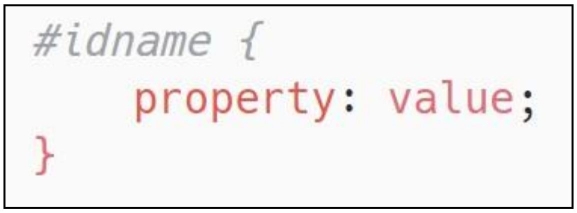
- Defined using a dot (.) followed by a class name.
- Classes can be applied to multiple elements, and one element can have multiple classes.



```
.classname {
    property: value;
}
```

ID selector

- Defined using a hash (#) followed by an ID name.
- IDs are unique within a document and should only be applied to one element.



```
#idname {
    property: value;
}
```

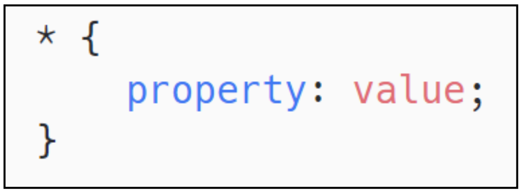
Selectors

Q.11] What is a CSS selector?

- A CSS selector is a pattern that is used to select and style HTML elements based on various criteria such as **element types**, **IDs, classes**, attributes, and **relationships with other elements**

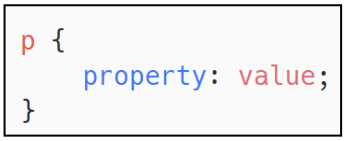
Q.12] What are the types of CSS Selectors

1. **Universal Selector (*)**
    - ✓ Selects all elements on the page



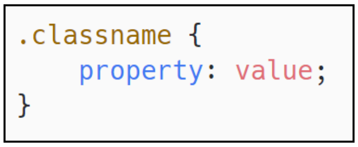
```
* {
    property: value;
}
```

2. **Type Selector**
    - ✓ Selects all elements of a specific type



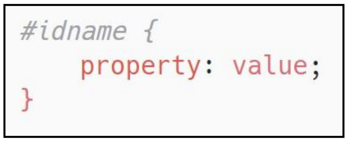
```
p {
    property: value;
}
```

3. **Class Selector (.)**
    - ✓ Selects elements with a specific class attribute



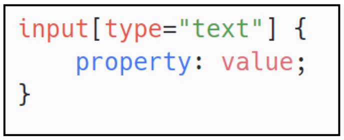
```
.classname {
    property: value;
}
```

4. **ID Selector (#)**
    - ✓ Selects a single element with a specific ID attribute



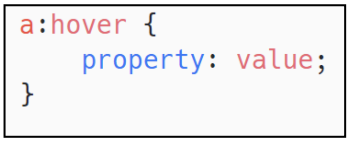
```
#idname {
    property: value;
}
```

5. **Attribute Selector**
    - ✓ Selects elements based on the presence or value of an attribute.



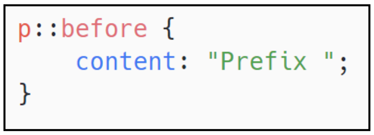
```
input[type="text"] {
    property: value;
}
```

6. **Pseudo-classes**
    - ✓ Selects elements based on their special state



```
a:hover {
    property: value;
}
```

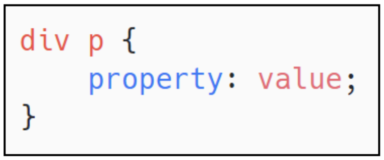
7. **Pseudo-elements**
    - ✓ Selects specific parts of an element's content



```
p::before {
    content: "Prefix ";
}
```

8. Descendant Selector (space)

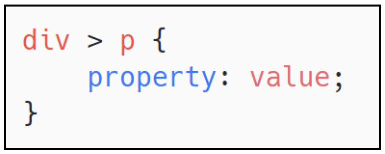
- ✓ Selects elements that are descendants of another element.



```
div p {
    property: value;
}
```

G. Child Selector (>)

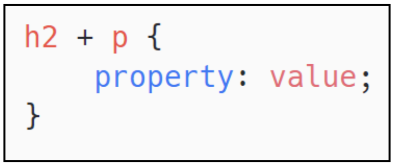
- ✓ Selects elements that are direct children of another element.



```
div > p {
    property: value;
}
```

10. Adjacent Sibling Selector (+)

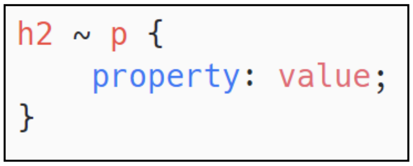
- ✓ Selects the first **element** that immediately follows a specified element, and both elements must have the same parent.



```
h2 + p {
    property: value;
}
```

11. General Sibling Selector (~)

- ✓ Selects **elements** that are siblings of a specified element.

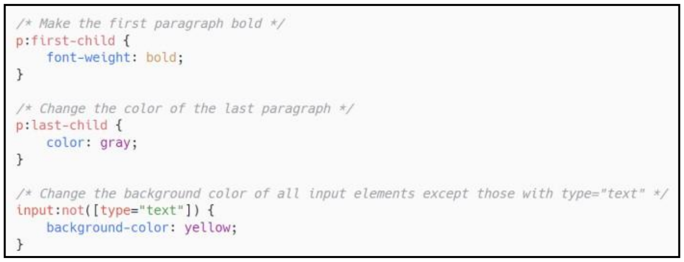


```
h2 ~ p {
    property: value;
}
```

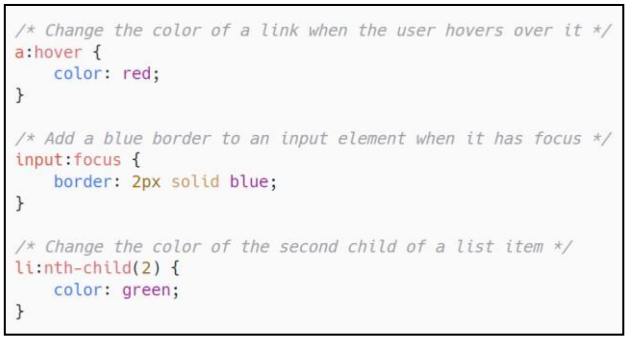
Q.13] What is a pseudo-class?

- A pseudo-class is a way to style an element in CSS **based on its state or position** without needing extra HTML code.
- Pseudo-classes are particularly useful for creating interactive and dynamic styles without needing to manipulate the DOM directly with JavaScript

Q.14] What is a pseudo-element?



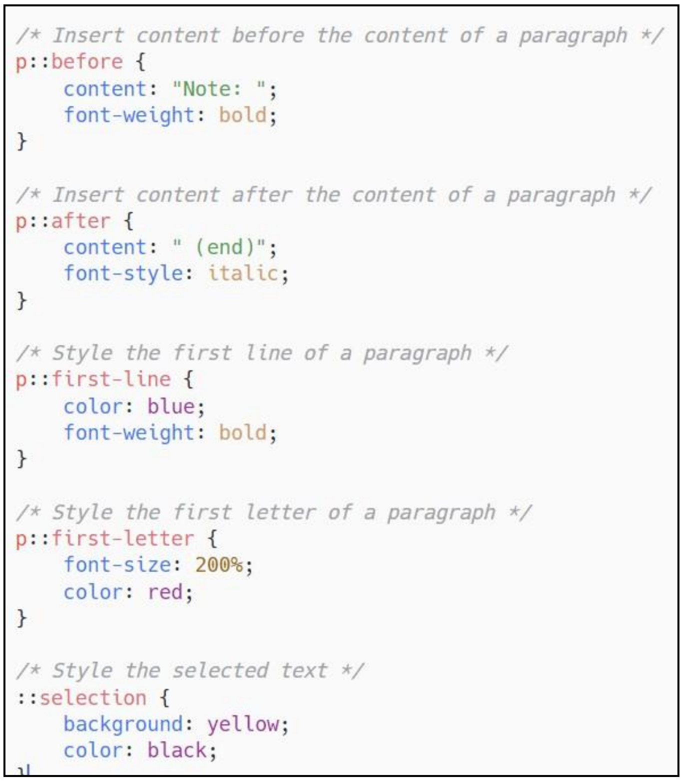
```
/* Make the first paragraph bold */
p:first-child {
    font-weight: bold;
}

/* Change the color of the last paragraph */
p:last-child {
    color: gray;
}

/* Change the background color of all input elements except those with type="text" */
input:not([type="text"]) {
    background-color: yellow;
}
```



```
/* Change the color of a link when the user hovers over it */
a:hover {
    color: red;
}

/* Add a blue border to an input element when it has focus */
input:focus {
    border: 2px solid blue;
}

/* Change the color of the second child of a list item */
li:nth-child(2) {
    color: green;
}
```

- It is a keyword added to the selectors which will allow to style the specific parts of an element's content.
- Pseudo classes - targets the entire element
- Pseudo elements – targets the specific part of an element
- Pseudo-elements are written with a double colon (::)

Here are some common pseudo-elements

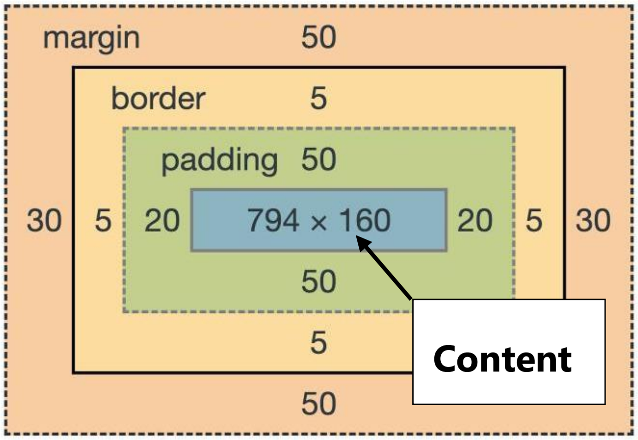


```
/* Insert content before the content of a paragraph */
p::before {
    content: "Note: ";
    font-weight: bold;
}

/* Insert content after the content of a paragraph */
p::after {
    content: " (end)";
    font-style: italic;
}

/* Style the first line of a paragraph */
p::first-line {
    color: blue;
    font-weight: bold;
}

/* Style the first letter of a paragraph */
p::first-letter {
    font-size: 200%;
    color: red;
}

/* Style the selected text */
::selection {
    background: yellow;
    color: black;
```

CSS BOX MODEL

Q.15] What is the CSS box model?

- The **CSS** box model is a fundamental concept that describes the rectangular boxes generated for elements in a web page, which includes the **element's content**, **padding**, **border**, and **margin**



Q.16] What are the components of the box model?

- **Content**: The actual content of the box, where text and images appear.
- **Padding**: The space between the content and the border.
- **Border**: The border surrounding the padding (if any) and content.
- **Margin**: The space outside the border, separating the element from other elements.

Q.17] What is the difference between margin and padding?

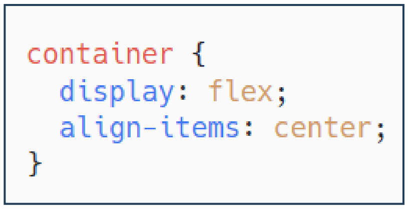
- **Margin**: Space outside the element's border, creating space between elements.
- **Padding**: Space inside the element's border, creating space between the content and the border.

Q.18] How do you center an element horizontally?

- To center an element horizontally, you can use the `margin: auto`; property for **block-level elements**

Q.1G] How do you center an element vertically?

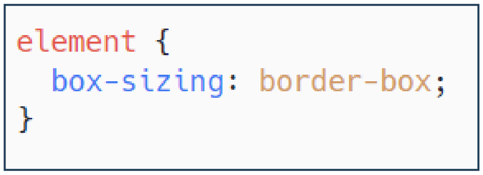
- To center an element vertically, you can use Flexbox



```
container {
  display: flex;
  align-items: center;
}
```

Q.20] What is the box-sizing property?

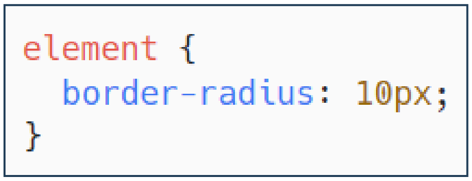
- The `box-sizing` property defines how the total width and height of an element are calculated.
- With `box-sizing: border-box;` the `padding` and `border` are included in the element's total width and height.



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

Q.20] How do you create a rounded border?

- Use the `border-radius` property to create rounded borders



```
element {
  border-radius: 10px;
}
```

Q.22] How do you create a border around an element?

- Use the **`border-radius`** property to create rounded borders

```css
element {
  border: 2px solid black;
}
```

LAYOUT

Q.23] What is the position property in CSS?

- The **`position`** property in CSS determines how an element is positioned in a document.
- It can take several values: `static`, `relative`, `absolute`, `fixed`, and `sticky`.

Q.24] Different Values of the position Property

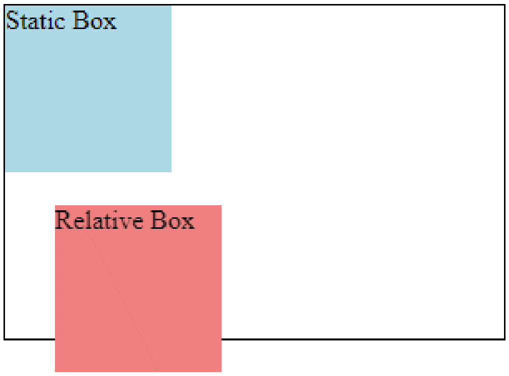
`static` Refer a codepen

- Default positioning
- Elements are positioned according to the normal flow of the document.
- We can't use offsets like `top`, `right`, `bottom`, and `left`

`relative` Refer a codepen

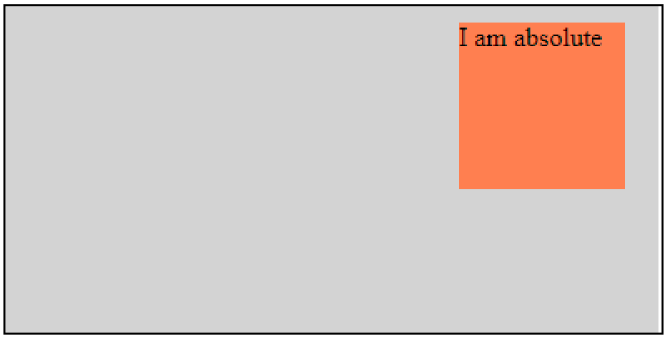
- Positioned relative to its normal position
- We can use offsets like like `top`, `right`, `bottom`, and `left`

Here is the diagram which illustrates the **static** and **relative** position



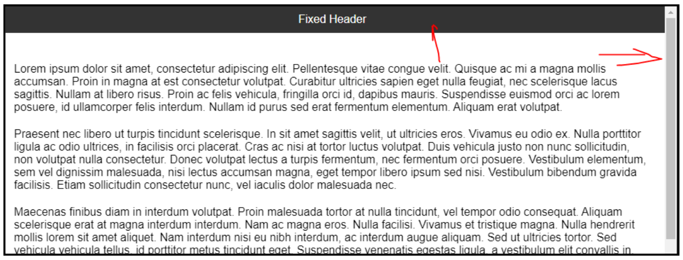
`absolute` Refer codepen

- Positioned relative to the nearest positioned ancestor
- If there is no positioned ancestor, the element is positioned relative to the initial containing block (usually the **<html>** element).
- The ancestor can have **relative**, **absolute**, **fixed**, or **sticky** positioning
- It does not work with position **static**



`fixed` Refer a codepen

- Positioned relative to the viewport, meaning it stays in the same place even when the page is scrolled



Fixed Header

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque vitae congue velit. Quisque ac mi a magna mollis accumsan. Proin in magna at est consectetur volutpat. Curabitur ultricies sapien eget nulla feugiat, nec scelerisque lacus sagittis. Nullam at libero risus. Proin ac felis vehicula, fringilla orci id, dapibus mauris. Suspendisse euismod orci ac lorem posuere, id ullamcorper felis interdum. Nullam id purus sed erat fermentum elementum. Aliquam erat volutpat.

Praesent nec libero ut turpis tincidunt scelerisque. In sit amet sagittis velit, ut ultricies eros. Vivamus eu odio ex. Nulla porttitor ligula ac odio ultrices, in facilisis orci placerat. Cras ac nisi at tortor luctus volutpat. Duis vehicula justo non nunc sollicitudin, non volutpat nulla consectetur. Donec volutpat lectus a turpis fermentum, nec fermentum orci posuere. Vestibulum elementum, sem vel dignissim malesuada, nisi lectus accumsan magna, eget tempor libero ipsum sed nisi. Vestibulum bibendum gravida facilisis. Etiam sollicitudin consectetur nunc, vel iaculis dolor malesuada nec.

Maecenas finibus diam in interdum volutpat. Proin malesuada tortor at nulla tincidunt, vel tempor odio consequat. Aliquam scelerisque erat at magna interdum interdum. Nam ac magna eros. Nulla facilisi. Vivamus et tristique magna. Nulla hendrerit mollis lorem sit amet aliquet. Nam interdum nisi eu nibh interdum, ac interdum augue aliquam. Sed ut ultricies tortor. Sed vehicula vehicula tellus, id porttitor metus tincidunt eget. Suspendisse venenatis egestas ligula, a vestibulum elit convallis in.

`Sticky` Refer a codepen

- Acts like relative until the element reaches a specified scroll point, then it "**sticks**" in place.

Q.25] What is the `float` property?

- Used to position an element to the left or right within its container , allowing text and inline elements to wrap around it.
  Refer a codepen

Q.26] How do you clear floats in CSS?

- To clear floats and ensure elements below a floated element are not affected, you can use the clear property
  Refer a codepen

Q.27] What are the different values of the display property?

- **`Block`** - Elements take up the full width available and start on a new line
- **`Inline`** - Elements only take up as much width as necessary and do not start on a new line.
- **`inline-block`** - Behaves like inline elements but can have a width and height.
- **`Flex`** - Turns the element into a flex container
- **`grid`** -Turns the element into a grid container

Q.28] How do you create a flexbox layout?

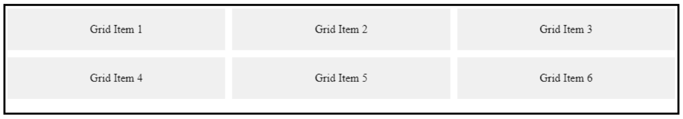
- To create a flexbox layout, you set `display: flex;` on the parent container and use various properties like `flex-direction`, `justify-content`, `align-items`, etc., on the child elements to control their layout.

Q.2G] What is the `flex` property?

- The **`flex`** property is a shorthand for **`flex-grow`**, **`flex-shrink`**, and **`flex-basis`** combined, defining how a flex item will grow or shrink to fit the available space.

Q.30] How do you create a grid layout in CSS?

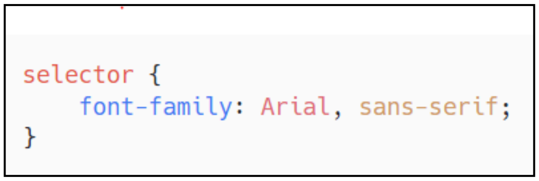
- To create a grid layout, you set `display: grid;` on the container and define its columns and rows using `grid-template-columns`, `grid-template-rows`, and place items within the grid using `grid-column` and `grid-row` properties.
- Refer codepen



TYPOGRAPHY

Q.31] How do you change the font of an element?

- Using the `font-family` property in CSS.



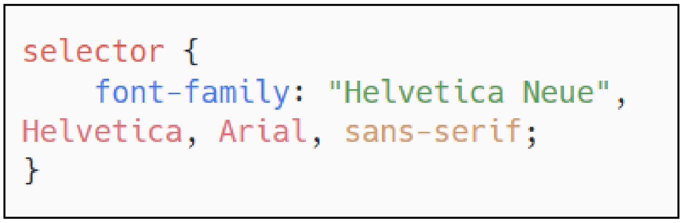
```
selector {
    font-family: Arial, sans-serif;
}
```

Q.32] What is the `font-family` property?

- The **font-family** property specifies the font family for text.
- It allows you to define a prioritized list of fonts to use, **separated by commas**.
- If the first font is not available on the user's system, it falls back to the next one in the list.

Q.33] What is the `font-family` property?

- You specify a fallback font by listing multiple fonts in the font-family property, separated by commas.



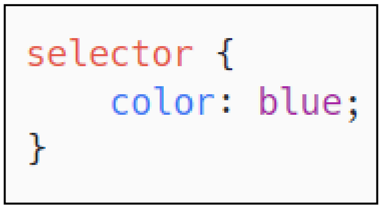
```
selector {
    font-family: "Helvetica Neue",
Helvetica, Arial, sans-serif;
}
```

Q.34] What is the `font-size` property?

- The **font-size** property in CSS specifies the size of the font.
- It can be set to various units like pixels (**px**), **ems** (**em**), or **percentages** (%).

Q.35] How do you change the color of the text?

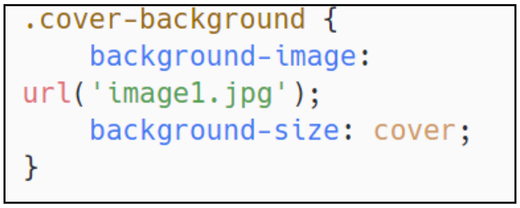
- Text color is changed using the `color` property.



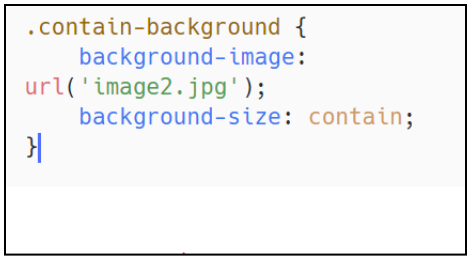
```
selector {
    color: blue;
}
```

Q.36] How do you change the color of the text?

- Text alignment can be controlled using the `text-align` property.



```
.cover-background {
    background-image:
url('image1.jpg');
    background-size: cover;
}
```

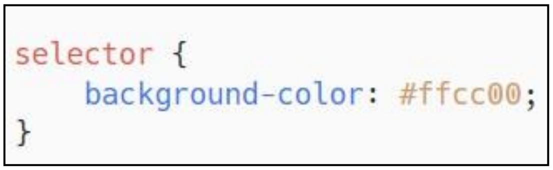


```
.contain-background {
    background-image:
url('image2.jpg');
    background-size: contain;
}
```

Colors s Backgrounds

Q.37] How do you change the color of the text?

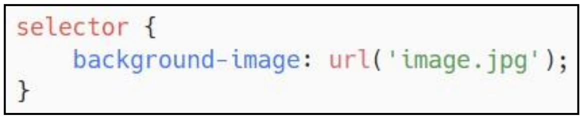
- You can set the background color of an element using the **`background-color`** property.



```
selector {
    background-color: #ffcc00;
}
```

Q.38] How do you set a background image?

- You can set a background image using the `background-image` property.



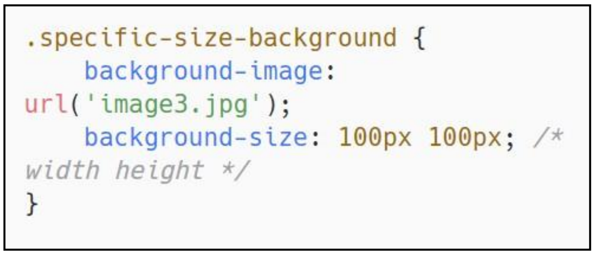
```
selector {
    background-image: url('image.jpg');
}
```

Q.3G] How do you set the size of a background image?

- To set the size of a background image, you can use the background-size property in CSS.

Here are some common ways to use it

1. **Cover**: Scales the background image to cover the entire element, possibly cropping it to fit.

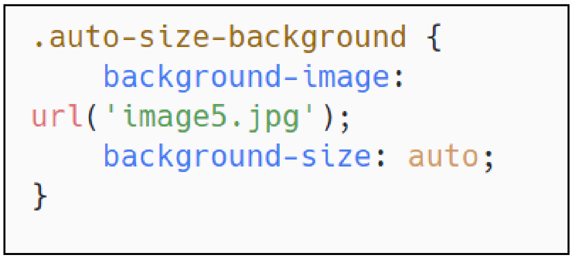


```
.specific-size-background {
    background-image:
url('image3.jpg');
    background-size: 100px 100px; /*
width height */
}
```

2. **Contain**: Scales the background image to be fully visible within the element, maintaining its aspect ratio.

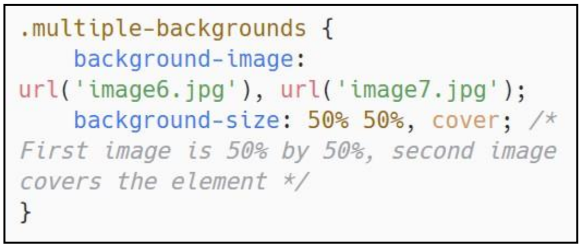
3. **Specific Size**: You can set specific dimensions for the background image using units like pixels, percentages, or any other CSS length units.

4. **Auto**: Maintains the original size of the background image.



```
.auto-size-background {
    background-image:
url('image5.jpg');
    background-size: auto;
}
```

5. **Multiple Background Sizes**: If you have multiple background images, you can specify sizes for each one.

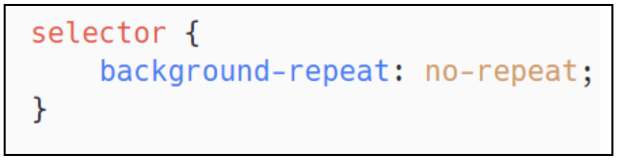


```
.multiple-backgrounds {
    background-image:
url('image6.jpg'), url('image7.jpg');
    background-size: 50% 50%, cover; /*
First image is 50% by 50%, second image
covers the element */
}
```

Q.40] How do you repeat a background image?

- You can control the repetition of a background image using the **background-repeat** property.
- This prevents the background image from repeating.

- Other values include **repeat**, **repeat-x**, and **repeat-y**.

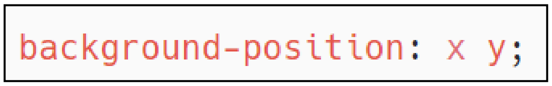


```
selector {
    background-repeat: no-repeat;
}
```

Q.41] What is the background-position property?

- Is used to specify the initial position of a background image within its container
- You can set the position using keywords, length values, or percentages
- This property helps in aligning the background image as desired within the element.

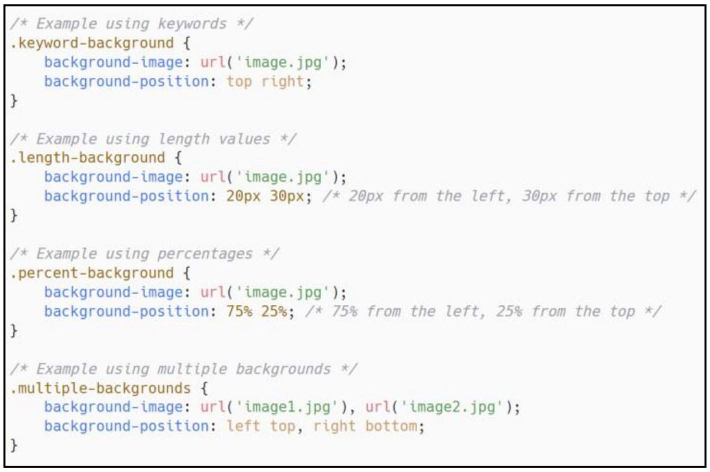
Syntax



```
background-position: x y;
```

- **x** represents the horizontal position.
- **y** represents the vertical position.

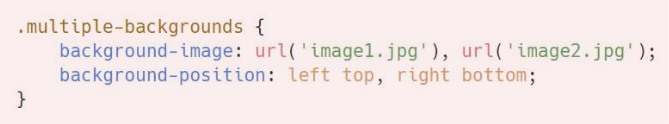
Common Values



```
/* Example using keywords */
.keyword-background {
    background-image: url('image.jpg');
    background-position: top right;
}

/* Example using length values */
.length-background {
    background-image: url('image.jpg');
    background-position: 20px 30px; /* 20px from the left, 30px from the top */
}

/* Example using percentages */
.percent-background {
    background-image: url('image.jpg');
    background-position: 75% 25%; /* 75% from the left, 25% from the top */
}

/* Example using multiple backgrounds */
.multiple-backgrounds {
    background-image: url('image1.jpg'), url('image2.jpg');
    background-position: left top, right bottom;
}
```

REMEMBER

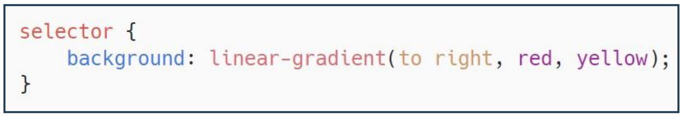
- You can use multiple background images in an element and position each one as needed using the background-position property.
- This allows for creative and flexible design options.



```
.multiple-backgrounds {
    background-image: url('image1.jpg'), url('image2.jpg');
    background-position: left top, right bottom;
}
```

Q.42] How do you create a gradient background?

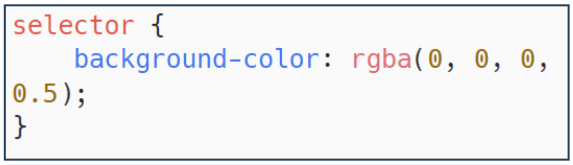
- You can create a gradient background using the `background` property with a gradient function.



```
selector {
    background: linear-gradient(to right, red, yellow);
}
```

Q.43] How do you create a semi-transparent background?

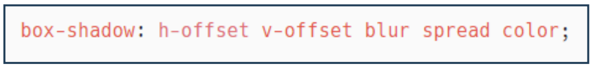
- You can create a semi-transparent background using the **rgba()** function in the background-color property.



```
selector {
    background-color: rgba(0, 0, 0,
0.5);
}
```

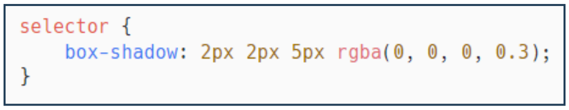
Q.44] How do you create a shadow effect?

- You can create a shadow effect using the `box-shadow` property for element shadows or `text-shadow` for text shadows



```
box-shadow: h-offset v-offset blur spread color;
```

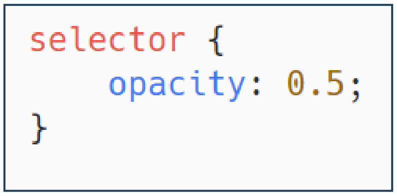
- **h-offset** : The horizontal offset of the shadow
- **v-offset**: The vertical offset of the shadow
- **blur** (optional): The blur radius
- **spread** (optional): The spread radius
- **Color** (optional): The color of the shadow



```
selector {
    box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.3);
}
```

Q.45] How do you change the opacity of an element?

- We can change the opacity of an element using the opacity property



```
selector {
    opacity: 0.5;
}
```

CSS TRANSITION

Q.46] What is a CSS transition?

- A CSS transition allows you to change property values smoothly (over a given duration) instead of instantly.

Q.47] How do you create a transition effect?

- You create a transition effect using the transition property.
- It specifies which properties to animate, the **duration of the animation**, the **timing function**, and any **delay** before the animation starts.

Q.48] What is the transition-duration property?

- The `transition-duration` property specifies how long the transition should take to complete.
- It can be set in seconds (s) or milliseconds (ms).

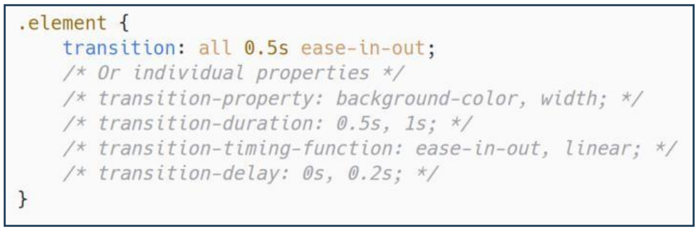
Q.4G] What is the transition-timing-function property?

- The **transition-timing-function** property describes how the intermediate values of the transition are calculated.
- Common values include **linear, ease**, **ease-in**, **ease-out**, and **ease-in-out**.

Q.50] What is the transition-delay property?

- The **transition-delay** property specifies the amount of time to wait before starting the transition.
- It can be set in seconds (**s**) or milliseconds (**ms**).

Example of CSS Transition

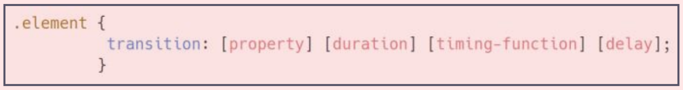


```
.element {
    transition: all 0.5s ease-in-out;
    /* Or individual properties */
    /* transition-property: background-color, width; */
    /* transition-duration: 0.5s, 1s; */
    /* transition-timing-function: ease-in-out, linear; */
    /* transition-delay: 0s, 0.2s; */
}
```

[Refer Advance CSS Animation and Transition git hub repository]

Remember

The syntax for the transition property is as follows



```
.element {
        transition: [property] [duration] [timing-function] [delay];
      }
```

CSS ANIMATION

Q.51] What is the transition-delay property?

- A CSS animation allows you to animate changes to CSS properties over time, using keyframes to define the changes.

Q.52] How do you create a CSS animation?

- You create a **CSS** animation by defining keyframes with the **@keyframes** rule and then applying the animation to an element using the animation property.

Q.53] What is the @keyframes rule?

- The **@keyframes** rule specifies the animation code.
- It defines the styles for the element at various points during the animation sequence.

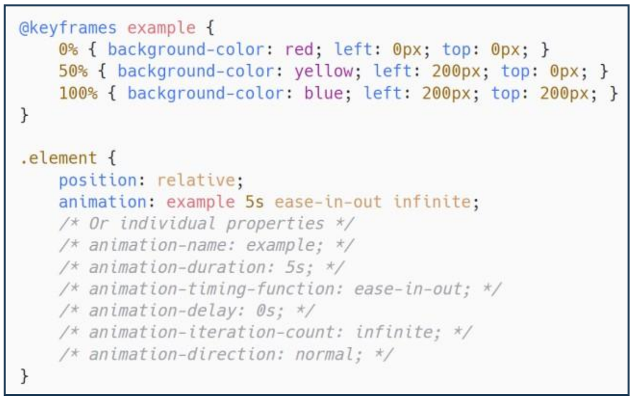
Q.54] What is the animation-duration property?

- The **animation-duration** property specifies how long an animation should take to complete one cycle.
- It can be set in seconds (s) or milliseconds (ms).

Q.55] What is the animation-timing-function property?

- The **animation-timing-function** property specifies the speed curve of the animation.

- Common values include **linear**, **ease**, **ease-in**, **ease-out**, and **ease-in-out**.



```css
@keyframes example {
    0% { background-color: red; left: 0px; top: 0px; }
    50% { background-color: yellow; left: 200px; top: 0px; }
    100% { background-color: blue; left: 200px; top: 200px; }
}

.element {
    position: relative;
    animation: example 5s ease-in-out infinite;
    /* Or individual properties */
    /* animation-name: example; */
    /* animation-duration: 5s; */
    /* animation-timing-function: ease-in-out; */
    /* animation-delay: 0s; */
    /* animation-iteration-count: infinite; */
    /* animation-direction: normal; */
}
```

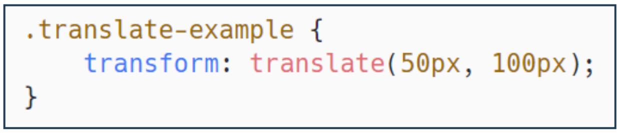
CSS TRANSFORM

Q.56] What is the `animation-timing-function` property?

- The `transform` property applies a 2D or 3D transformation to an element.
- This property allows you to **rotate**, **scale**, **skew**, or **translate** an element.

Q.57] How do you translate an element using CSS?

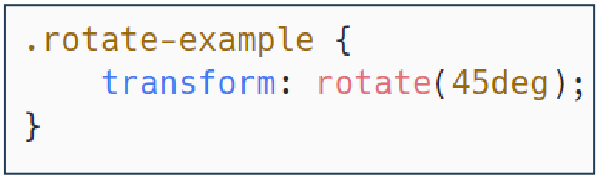
- You can translate an element using the **translate()** function.



```css
.translate-example {
    transform: translate(50px, 100px);
}
```

- This moves the element 50 pixels to the right and 100 pixels down.

Q.58] How do you rotate an element using CSS?

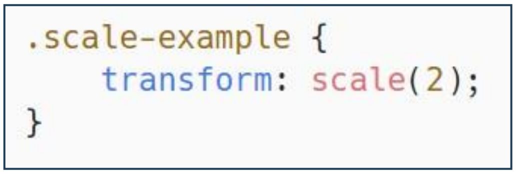


```css
.rotate-example {
    transform: rotate(45deg);
}
```

- You can rotate an element using the **rotate()** function.
- This rotates the element 45 degrees clockwise.

Q.5G] How do you scale an element using CSS?

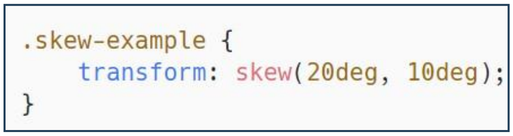
- You can scale an element using the **scale()** function.



```css
.scale-example {
    transform: scale(2);
}
```

Q.60] How do you skew an element using CSS?

- You can skew an element using the **skew()** function.

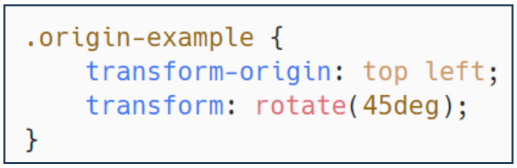


```css
.skew-example {
    transform: skew(20deg, 10deg);
}
```

- This skews the element **20 degrees** along the X-axis and **10 degrees** along the Y-axis.

Q.61] What is the transform-origin property?

- The **transform-origin** property specifies the point around which a transformation is applied.

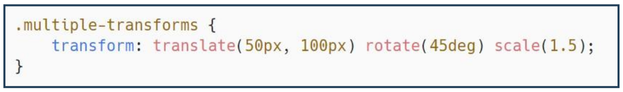


```css
.origin-example {
    transform-origin: top left;
    transform: rotate(45deg);
}
```

- This rotates the element 45 degrees around its top-left corner. Refer codepen

Q.62] How do you apply multiple transforms to an element?

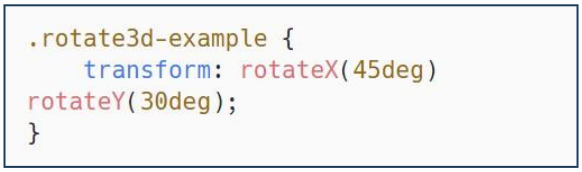
- You can apply multiple transforms by chaining them together.



```css
.multiple-transforms {
    transform: translate(50px, 100px) rotate(45deg) scale(1.5);
}
```

Q.63] What are 3D transforms, and how do you use them?

- 3D transforms allow you to manipulate elements in three-dimensional space.
- Functions like **rotateX**(), **rotateY(), translateZ**(), and **scaleZ()**

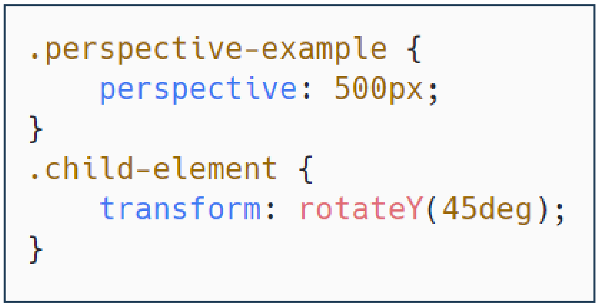


```
.rotate3d-example {
    transform: rotateX(45deg)
rotateY(30deg);
}
```

- This rotates the element 45 degrees around the X-axis and 30 degrees around the Y-axis.

Q.64] What is the perspective property, and how is it used?

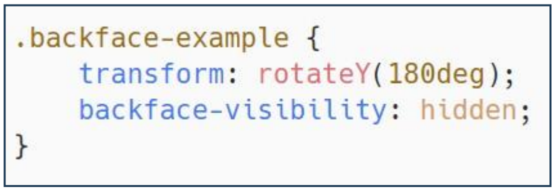
- The perspective property defines the perspective from which a 3D element is viewed.
- It gives depth to 3D transformed elements.



```
.perspective-example {
    perspective: 500px;
}
.child-element {
    transform: rotateY(45deg);
}
```

Q.65] What is the `backface-visibility` property?

- The `backface-visibility` property determines whether the back face of an element is visible when it is rotated



```
.backface-example {
    transform: rotateY(180deg);
    backface-visibility: hidden;
}
```

CSS FLEX

Q.66] What is Flexbox?

- Flexbox, or the Flexible Box Layout, is a CSS layout module that provides an efficient way to **align and distribute space among items** in a container, even when their **size is unknown** or dynamic
- The main idea behind the flex layout is to give the container the **ability to alter its items' width, height, and order to best fill the available space**.

Q.67] What is the main axis and cross axis in Flexbox?

- In Flexbox, the **main axis** and **cross axis** are fundamental concepts that determine the layout and alignment of flex items within a flex container.

Main Axis

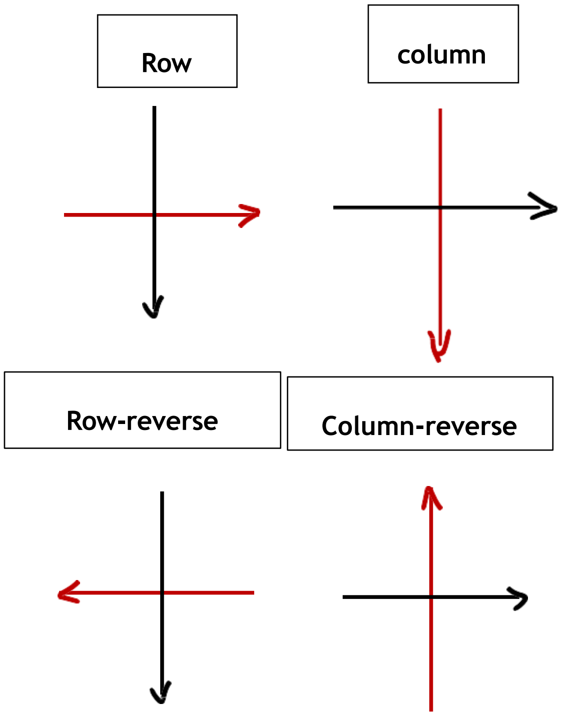
- The main axis is the primary axis along which flex items are laid out.
- It is defined by the **flex-direction** property
- Directions
    - **row (default)**: The main axis runs horizontally from left to right.
    - **row-reverse**: The main axis runs horizontally from right to left.
    - **column**: The main axis runs vertically from top to bottom.
    - **column-reverse**: The main axis runs vertically from bottom to top.

Cross Axis

- The cross axis is perpendicular to the main axis.
- It runs in the opposite direction of the main axis
- It is used for alignment and spacing of flex items.
- Directions
    - When **flex-direction** is **row** or **row-reverse**, the cross axis runs vertically.
    - When **flex-direction** is **column** or **column-reverse**, the cross axis runs horizontally.

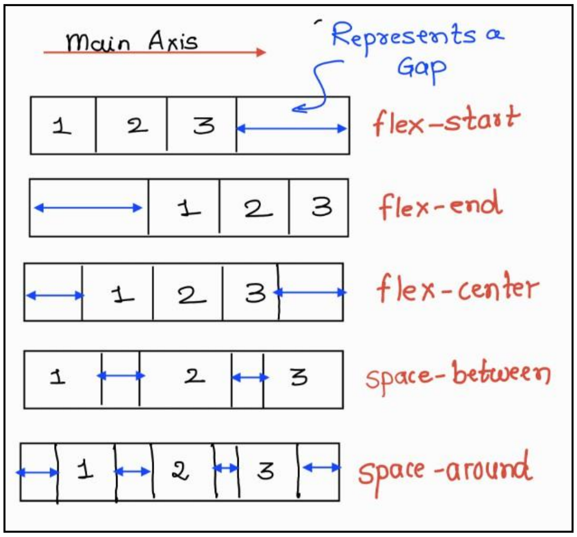
Refer the below figure

- **Red Arrow --- Main Axis**
- **Black Arrow ---Cross Axis**


Row
column
Row-reverse
Column-reverse

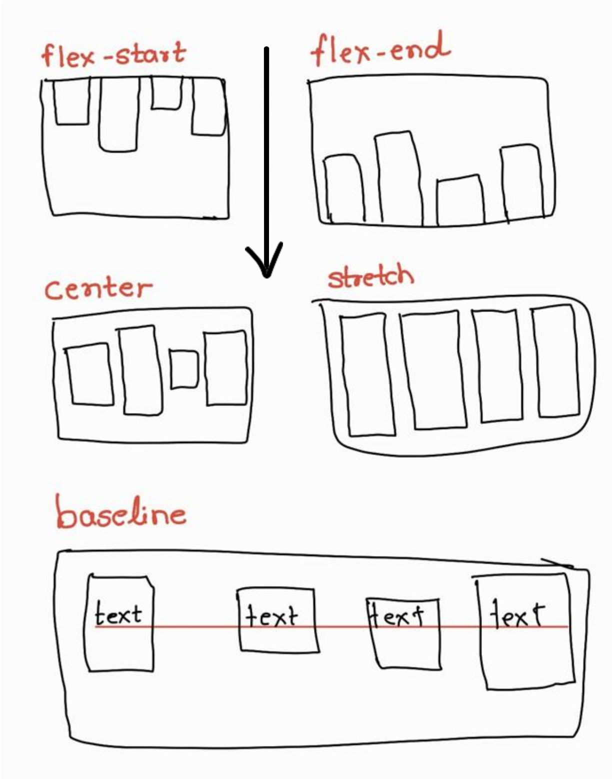
Q.68] What is the justify-content property?

- The justify-content property ***aligns flex items along the main axis of the container***.
- It distributes space between and around content items, with possible values:
    - `flex-start`
    - `flex-end`
    - `space-between`
    - `space-around`
    - `space-evenly`


Main Axis
Represents a Gap
1
2
3
flex-start
flex-end
flex-center
space-between
space-around

Q.6G] What is the align-items property?

- The **align-items** property **aligns flex items along the cross axis**.
- It can be used to set the default alignment for all flex item
    - `Stretch`
    - `flex-start`
    - `flex-end`
    - `center`
    - `baseline`


flex-start
flex-end
center
stretch
baseline
text
text
text
text

Q.70] What is the flex-direction property?

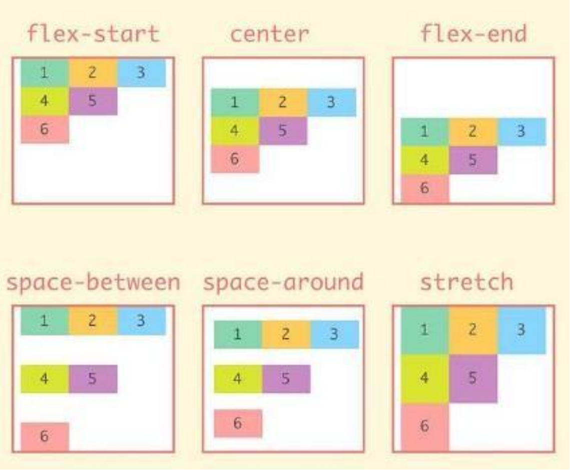
- The flex-direction property specifies the direction of the main axis
    - ✓ `Row`
    - ✓ `row-reverse`
    - ✓ `column`
    - ✓ `column-reverse`

Q.71] What is the flex-wrap property?

- ✓ The **flex-wrap** property determines whether flex items are forced into a single line or can wrap onto multiple lines
- ✓ Common values are
    - `Nowrap` : Wrap in single line
    - `Wrap` : Multiple line wrapping
    - `wrap-reverse`: Wrap in Multiple line from bottom to top

Q.72] What is the align-content property?

- The **align-content** property aligns flex lines (not individual items) along the cross axis when there is extra space in the container
- Use align-content when your flex container has more than one line of items.
- Values used are
  - ✓ `Stretch`
  - ✓ `flex-start`
  - ✓ `flex-end`
  - ✓ `center`
  - ✓ `space-between`
  - ✓ `space-around`
  - ✓ `space-evenly`



Source of this image: samanthaming.com

Q.73] What is the order property in Flexbox?

- The `order` property defines the order in which flex items are displayed within the flex container.
- By default, items have an order of **0**, but this can be changed to rearrange the items

Note:

- ✓ **Lower values** of order appear first, **higher values** appear later.
- ✓ **Negative values** can be used to move items before those with default order.

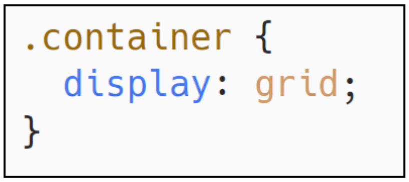
Grid Layout

Q.74] What is CSS Grid Layout?

- CSS Grid Layout is a two-dimensional layout system for the web
- It allows you to create complex, responsive web layouts more easily and consistently across browsers.
- Grid Layout enables you to align elements into rows and columns, providing more control over the design and the position of items

Q.75] How do you create a grid container?

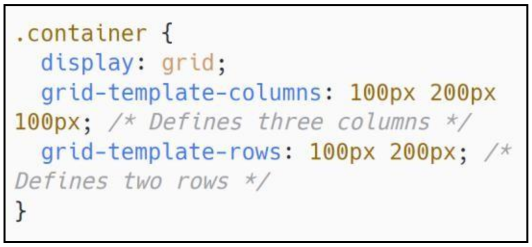
- To create a grid container, you use the **display** property with a value of **grid** or **inline-grid** on the container element.



```
.container {
  display: grid;
}
```

Q.76] How do you define grid columns and rows?

- You define grid columns and rows using the **grid-template-columns** and **grid-template-rows** properties.
- These properties specify the size and number of columns and rows in the grid.

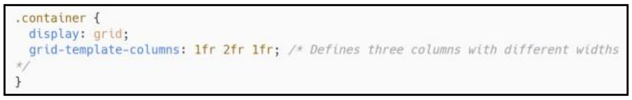


```
.container {
  display: grid;
  grid-template-columns: 100px 200px
100px; /* Defines three columns */
  grid-template-rows: 100px 200px; /*
Defines two rows */
}
```

Q.77] What is the grid-template-columns property?

- The **grid-template-columns** property defines the number and width of columns in the grid.

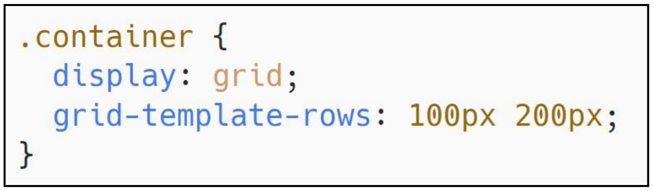
- You can specify the width using various units like **pixels** (px), **percentages** (%), **fractions** (fr), or any other CSS units.



```
.container {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr; /* Defines three columns with different widths
*/
}
```

Q.78] What is the grid-template-rows property?

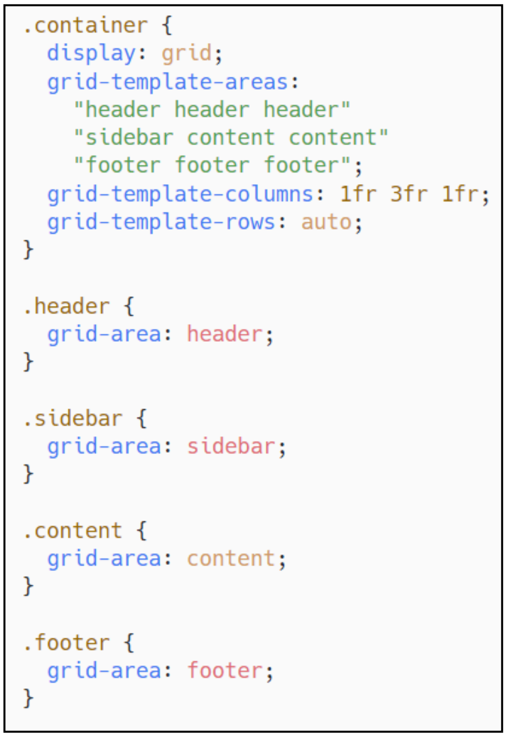
- The **grid-template-rows** property defines the number and height of rows in the grid.
- Similar to **grid-template-columns**, you can specify the height using various units.



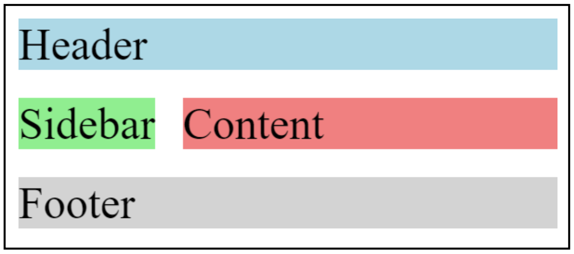
```
.container {
  display: grid;
  grid-template-rows: 100px 200px;
}
```

Q.7G] How do you create grid areas?

- Grid areas are defined using **the grid-template-areas** property.
- You name areas in the grid and then assign elements to those areas using the grid-area property. [Refer Codepen]

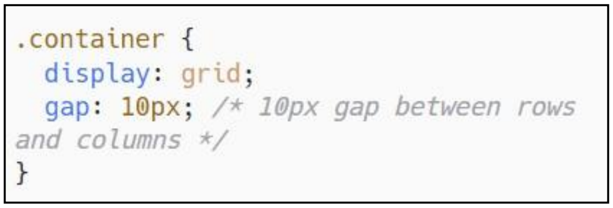


```
.container {
  display: grid;
  grid-template-areas:
    "header header header"
    "sidebar content content"
    "footer footer footer";
  grid-template-columns: 1fr 3fr 1fr;
  grid-template-rows: auto;
}

.header {
  grid-area: header;
}

.sidebar {
  grid-area: sidebar;
}

.content {
  grid-area: content;
}

.footer {
  grid-area: footer;
}
```



Q.80] What is the `grid-gap` property

- The **grid-gap** property (now commonly referred to as gap) defines the spacing between rows and columns in the grid.



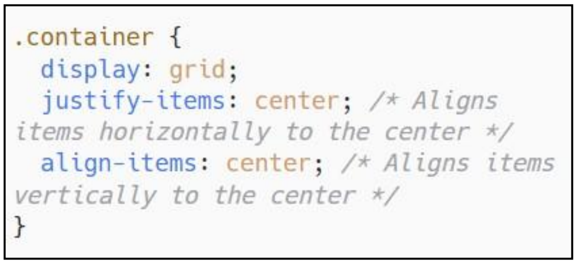
```
.container {
  display: grid;
  gap: 10px; /* 10px gap between rows
and columns */
}
```

- It can take one or two values: one for **both row** and column gap, or two values where the first is the row gap and the second is the column gap.

Q.81] How do you align items in a grid?

You can align items in a grid using the following properties:

- `justify-items`: Aligns items horizontally within their grid area.
- `align-items`: Aligns items vertically within their grid area.
- `place-items`: A shorthand for setting both justify-items and align-items.



```
.container {
  display: grid;
  justify-items: center; /* Aligns
items horizontally to the center */
  align-items: center; /* Aligns items
vertically to the center */
}
```

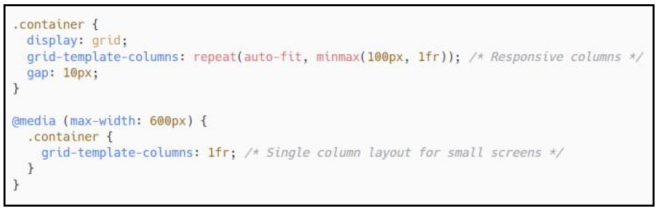
Q.82] What is the grid-auto-flow property?

- The **grid-auto-flow** property controls how the auto-placement algorithm works, specifying how auto-placed items are inserted in the grid.
- It can take the following values:
  - ✓ **row** (default): Items are placed by row.

- ✓ **column**: Items are placed by column.
- ✓ **dense**: Items are placed to fill in gaps.

Q.83] How do you create a responsive grid layout?

- To create a responsive grid layout, you can use media queries and relative units like percentages (%) or fractions (**fr**).
- Additionally, the **repeat()** function and the **minmax()** function can help create flexible grid structures.



```css
.container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(100px, 1fr)); /* Responsive columns */
  gap: 10px;
}

@media (max-width: 600px) {
  .container {
    grid-template-columns: 1fr; /* Single column layout for small screens */
  }
}
```

- **auto-fit :** Automatically fit as many columns as possible in the available space.
- **minmax(150px, 1fr) :** Each column should be at least 150px wide but can grow to fill available space (1fr)

RESPONSIVE DESIGN

Q.84] What is Responsive Design?

- Responsive design is an approach to web design that ensures web pages render well on a variety of devices and window or screen sizes

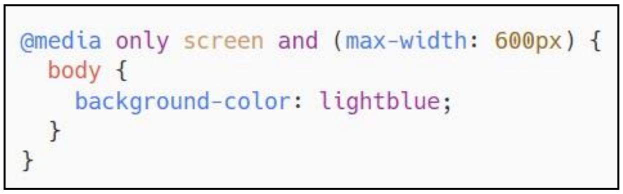
Q.85] How do you create a responsive layout using CSS?

- Use percentage-based widths for layout elements rather than fixed pixel values
- Ensure images can scale within their containing elements by using max-width: 100%
- Apply different styles based on the screen size or device characteristics

Q.86] What Are Media Queries?

- Media queries are a CSS feature that allows content rendering to adapt to different conditions such as **screen size**, **resolution**, or **orientation**.
- They can be used to apply styles based on these conditions.

Q.87] How to Write a Media Query in CSS



```css
@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

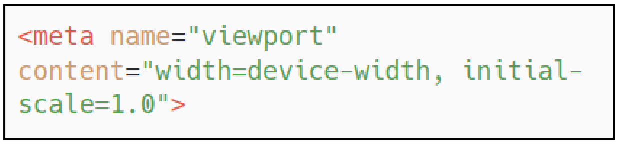
- This example changes the background color to light blue on devices with a screen width of 600 pixels or less.

Q.88] What is the @media Rule?

- The `@media` rule in CSS is used to define a block of CSS rules that will apply only if certain conditions are true, such as a minimum or maximum width of the viewport.

Q.8G] What is the Viewport Meta Tag, and Why is it Important?

- The viewport meta tag is used to control the layout on mobile browsers.
- It ensures that the web page is rendered correctly on all devices by setting the viewport width to the device width.



```html
<meta name="viewport"
content="width=device-width, initial-
scale=1.0">
```

Q.G0] Explain `@media only screen and (max-width: 600px)`

- **Only**: This keyword is used to hide stylesheets from older browsers that do not support media queries. It's a way to ensure that the media query will only apply in browsers that understand the media query syntax.
- **Screen:** This is a media type that targets screens, such as computer monitors, tablets, smartphones, etc.
- **(max-width: 600px):** This condition applies the styles within the media query if the viewport width is 600 pixels or less.

REMEMBER

Media Types

Media types specify the general category of the device or rendering context

- `all`
- `print`
- `screen`
- `speech`

Media Features

Media features describe specific characteristics of the user agent, output device, or environment:
Here are the few of them

- `width`
- `min-width`
- `max-width`
- `height`
- `min-height`
- `max-height`
- `orientation`

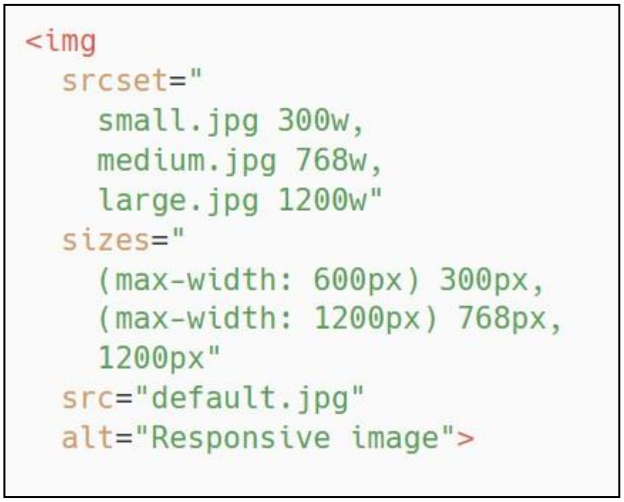
Combining Media Queries

Media queries can be combined using logical operators

- **and**: Combines multiple media features.
- **not**: Negates a media query.
- **only**: Applies styles only if the entire query matches

Q.GG1] Explain how you can make images responsive using HTML and CSS. Provide an example demonstrating different techniques such as the `srcset` attribute in HTML and CSS properties.

- The `srcset` attribute allows you to specify different images for different device widths, improving performance by serving the most appropriate image size
- In this example
  - `srcset` specifies different image files for various widths (300w, 768w, 1200w)
  - sizes defines the image display sizes based on the viewport width.
  - `src` provides a fallback image for browsers that do not support `srcset`.



```
<img
  srcset="
    small.jpg 300w,
    medium.jpg 768w,
    large.jpg 1200w"
  sizes="
    (max-width: 600px) 300px,
    (max-width: 1200px) 768px,
    1200px"
  src="default.jpg"
  alt="Responsive image">
```