CSS

* Introduction.

- It stands for Cascading Style Sheets.
- It is used to style & layout web pages.
- CSS is code that style web content
- It is not programming or markup long.
- It is style sheet longuage
- It is used to describe the style of doc.

* Syntax :

Selector

```
h1 {

    Color : red ;
}
```

Property Value

Declaration.

* Selector.

- This is HTML element name at the start of ruleset
- It defines the elements to be styled.

* Declaration.

- It specifies which of the elements properties you want to style.

* Properties :
- There are ways in which you can style an HTML element.

* Property value.
- To the right of the property - after the colon - there is the property value.

* Different types of selector.

(i) Element selector.
- All HTML elements of the specified type.
- eg. <p>

(ii) ID selector.
- The element of page with the specified ID
- eg. #my-id.

(iii) Class selector.
- specified class.
- eg. .my-class

Note :
& - ampersand
* - asterisk

(iv) Attribute selector.
- with specified attribute.
- eg. <img src = " myimage.png " >
→ img [src]

(v) Pseudo-class selector.
- The specified elements but only when in specified state
- eg. a : hover (when cursor hover over a link.)

(vi) Universal selector - all - * { }

* Inline style.
Including.

- Inline (written in html body tag)

<h1 style = " color : red " > APna dg </h1>

- <style> Tag. (written in Html head tag)

```
<style>
    h1 {
        color : red;
    }
</style>
```

- External stylesheet.
- Writing CSS in a seperate doc. & linking it with HTML file

* Color Property (level 1)
- Used to set the color of foreground

color : red ;

* Background Color Property.
- Used to set the color of background.

background - color : red ;

- Color system. (color picker used to find no. of RGB)

(i) RGB color : rgb (255, 0, 0) - Red
color : rgb (0, 0, 255) - blue

- Hex (Hexadecimal) (color palette & color hunt website used)

color : #ff0000 ; (red)
color : #0000ff ; (blue)

* Text Properties.

* text-align

text-align : left / right / center ;

* text-decoration.

text-decoration : underline / overline / line-through ;

* font-weight

font-weight : normal / bold / bolder / lighter ;

lightest - 100
boldest - 900

* font-family.

font-family : arial ;

- Generic font families.

- Generic font families are determined by font family properties

eg. 5 Generic font families
- Serif
- Sans-serif
- Cursive
- Fantasy
- Monospace.

- Specific font families.

- Specific font families are specific fonts with different styles within the one font family name. such as Arial, Times New Roman, & Tahoma.

* Units in CSS (It specify a fixed length value)

Absolute.

pixels (px)

96 px = 1 inch = 2.54 cm

font-size : 2px ;

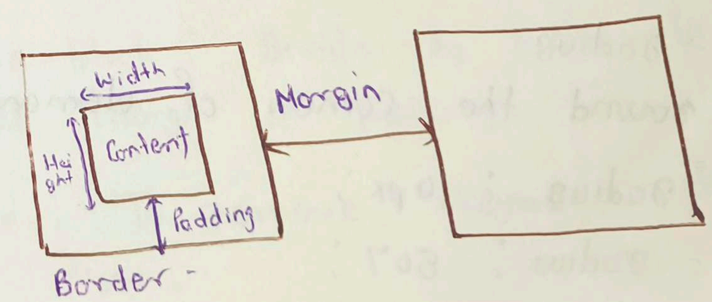
* line-height

line-height : 2px ;
line-height : normal ;

* text-transform.

text-transform : uppercase / lowercase / capitalize / none ;

* Box Model in CSS. (Level 2)



- Height.

- By default, it sets the content area height of the element.

```
div {
    height : 50px ;
}
```

- Width.

- By default, it sets the content area width of element.

```
div {
    width : 50px ;
}
```

- Border

- Used to set an element's border.

```
border - width : 2px ;
border - style : solid / dotted / dashed ;
border - color : black ;
```

- Shorthand :

```
border : 2px solid black ;
```

- Border - radius :

- Used to round the corners of element

```
border - radius : 10 px ;
border - radius : 50% ;
```

- Padding.

- This property is used to create space around the element, inside any defined border.

```
Padding : left / right / top / bottom ;
```

- Shorthand

```
padding : 50 px
padding : 1px 2px 3px 4px
          top right bottom left.
```

- Margin.

- This property is used to create space around the element.

```
Margin - right / left / top / bottom ;
```

- Display Property.

```
display : inline / block / inline - block / none.
```

- inline : Takes only the space required by the element (no margin / padding).
- block : Takes full space available in width.
- inline - block : Similar to inline but we can set margin & padding.
- none : To remove element from document flow.

* Visibility.

Visibility : visible / hidden / collapse.

- Collapse : It is used only for table elements

* Alpha Channel.

(opacity 0 to 1)

- RGBA

Color : rgba (255, 0, 0, 0.5)
R G B A

* Relative Units in CSS. (level-3)

- Relative length units specify a length relative to another length property

%.

em

rem

• Percentage (%)

- It is used to define a size as relative to an elements parent object.

width : 30%;

margin-left : 50%;

• Em :

- Relative to the font-size of element (2em means 2 times of size of current font).

• Rem (Root em).

- Relative to the font size of root element

• Vh

- relative to 1% viewport height.

• Vw

- relative to 1% viewport width

* Position.

- It sets how an element is positioned in document.

position : static / relative / absolute / fixed ;

• Static : defaulut position.

• relative : element is relative to itself. (The top, right, bottom, left & z-index will work).

• absolute : positioned relative to its closest positioned ancestor. (removed from the flow)

• fixed : positioned relative to browser (removed from flow)

• Sticky : positioned based on users scroll position.

• Z- index.

- It decides the stack level of elements
- Overlapping elements with larger z-index cover those with smaller one.

z-index : auto (0) / 1 / 2 / -1 / -2 . .

* Background image.

Used to set an image as background

background - image : url (" image.jpeg ");

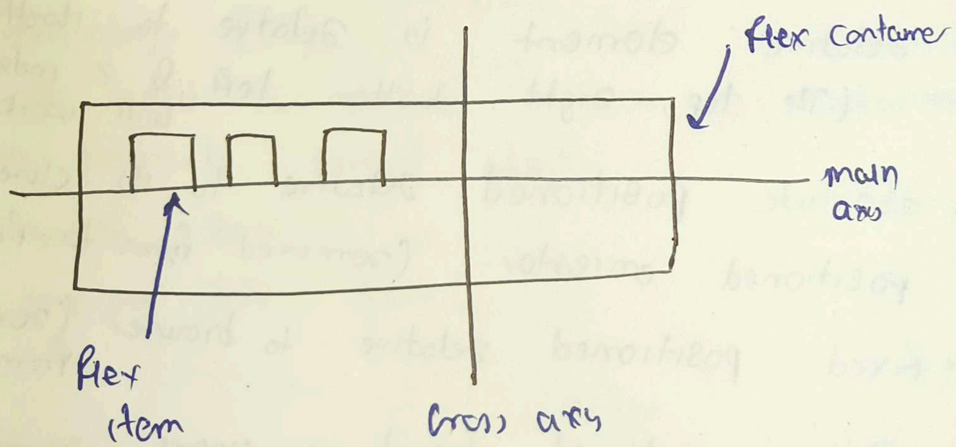
* Background Size.

background - size : cover / contain / auto ;

* Flexbox in CSS (level 4)

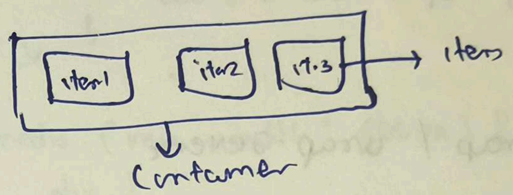
(Flexible Box Layout)

- It is one - dimensional layout method for arranging items in rows or colums



• Flex Container : The Parent "div" which contains various divisions is called a flex container.

• Flex items. : The items inside the container "div" are flex items.



• Main Axis.

- By default, the main axis runs from left to right.

- Main Start
- Main Size
- Main End

• Cross Axis :

- The cross axis will be perpendicular to the main Axis.

- Cross start / size / End.

* Flexbox direction.

- It sets how flex items are placed in the flex container, along with axis & direction

flexbox - direction : row (default) / row-reverse / column / column - reverse ;

* Flex Properties for flex container

- justify - content :
alignment along the main axis
flex-start / flex-end / center / space-evenly / space-betn.

- flex-wrap :
nowrap / wrap / wrap-reverse

- align-items :
alignment along the cross axis

- align-content :
alignment of space betn & around the content along cross-axis.

- align-self :
alignment of individual along the cross axis

- flex-grow :
- How much a flex item will grow relative to the rest of flex items if space is available.

- flex-shrink :
- How much a flex item will shrink - - - - available

* Media Queries
- Help create a responsive website.

```
@media (width : 600px) {
    div {
        background-color : red }
}

@media (min-width : 600px) {
    div {
    }
}

@media (min-width : 200px) & (min-width : 300px)
{
    div {
    }
}
```

Transitions (level 5).

- Transitions enable you to define the transition betn two states of an element.

- transition-property : property you want to transition (font-size, width, etc)

- transition-duration : 2s / 4ms

- transition-timing-function : ease-in / ease-out / linear / steps ..

- transition-delay : 2s / 4ms ...

- Pseudoclass

```
div:hover {
    background-color : red;
    color : white ;
}

div : active {
    background-color : pink;
}
```

Note : Transition Shorthand

property name / duration | timing-funct^ | delay.

transition : font-size 2s ease-in-out 0.2s ;

* CSS Transform

- Used to apply 2D & 3D transformation to an element

- rotate

transform : rotate (45 deg) ;

rotate : 45 deg ;

rotate X : 45 deg ;

rotate Y : 45 deg ;

rotate Z : 45 deg ;

- Scale

transform : scale (2) | (0.5) / (1, 2) ;

transform : scale X (0.5) / scale Y (0.5) ;

- translate

transform : translate (20px) / translate (20px, 50px) ;

transform : translate X (20px) / translate Y (20px) ;

- Skew

transform : skew (30 deg) ;

Animation

- To animate CSS elements.

```
@keyframe myName {
    from { font-size : 20px ; }
    to { font-size : 40px ; }
}
```

* Animation - Properties.

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

- Animation Shorthand

fx delay iteration count

animation: myName 2s linear 3s infinite normal;

name duration direction

- % in Animation

```
@keyframe myName {
    0%  {font-size: 20px;}
    50% {font-size: 30px;}
    100% {font-size: 40px;}
}
```

- Grid.
- 3D Animation.
- Media Queries.
- CSS float & clear
- CSS obj-fit & obj-cover
- CSS filters

* Figma. Website / App.

- Figma is the leading collaborative design tool for building meaningful products (design of products)

CSS Properties

Date:- / /
Page No.:-

* More Text properties.

- text decoration - line
- text - decoration - color
- text - indent - It is used to indent the first line of paragraph & the size can be in px, cm, pt.
- text - justify - Used to set the text - align to justify.
- text - shadow - h1 { text-shadow : 2px 2px red ; }
- line - height
- Word - Spacing
- White - space - Used to handle white space in content
- Google Font style used with link
  eg. neon effect.

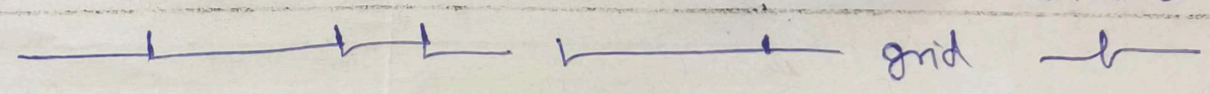
* Box Model Properties.

- outline - style : dotted / dashed / solid / double / etc
- outline - color :
- outline - width :
- outline - offset :

* More Display Properties

display : inline / block / inline-block / none / contents / flex / grid / table.

flex - Displays an element as block - level flex container

grid - 

Date:- / /
Page No.:-

- box shadow :

box - shadow : 10px 15 px 50 px red ;

x - axis y - axis blur color

box - shadow : in set 10 px 15 px red ; //inner shadow

- Text Shadow :

text - shadow : 10 px 15 px 50 px red ;

Date:- / /
Page No.:-

* CSS Grid

- CSS grid layout is a two-dimensional layout system for the web.
- It is used organize content into rows & columns
- It will typically have columns, rows & then gap betn rows & coln.

Properties :

- Column-gap
- gap
- grid
- grid-area
- grid-auto-columns
- grid-auto-flow
- grid-auto-rows
- grid-column
- grid-column-start
- grid-column-end
- grid-column-gap
- grid-gap
- grid-row
- grid-row-start
- grid-row-end
- grid-row-gap
- grid-template

* Read about all CSS units (eg. fractional unit

14 June

Date:- / /
Page No.:-

- CSS Sizing Units

(i) Pixel - Px. eg. 100px, 200px.

(ii) vw - viewport. width (used for responsiveness)

(iii) vh - viewport height.

(iv) em -

(v) rem — relative to font size of parent & root element

(vi) in - Inches = 1in = 96 px = 2.56 cm.

(viii) vmin - Relative to 1% of viewports smaller dimension

(ix) vmax - Relative to 1% of viewports larger dimension

(x) % Relative to parent element

CSS Interview Questions...

Q-1 Explain the CSS Box Model & its components

→ The box model is a fundamental concept in CSS that describes how elements are positioned, structured & sized in web page.

Comp.: Content, padding, border, Margin.

Q-2 CSS Specificity.

→ When more than one set of CSS rules applies on same element, then the browser will have to decide which specific set will be applied to element.

CSS Specificity Rules:

- Inline CSS - Internal CSS - External CSS

Q-3 CSS Preprocessor.

→ - It is scripting languages that extend the default capabilities of CSS.

- They enable us to use logic in our CSS codes, such as variable, nesting, inheritance, mixing, Funct^n & maths operators.

Q-4 OOCSS - Object oriented CSS

- The focuses on seperating structure (obj) from visual styles by seperating this. OOCSS promotes code reusability & maintainability.

Q-5 Explain the concept of reflow & repaint in relⁿ to browser rendering & performance optimizatⁿ.

→ Reflow:
- It is the process of calculating the position & size of elements on web page.

Repaint:
- It is the process of updating the visual representatⁿ of elements on the screen.

Q-6 Key techniques used in responsive web design

→ (i) Fluid Grid
(ii) Flexible images
(iii) Media Queries
(iv) Breakpoints
(v) Mobile-first approach

Q-7 What is CSS?

Q-8 Different way to apply CSS

→ (i) inline CSS (ii) internal CSS (iii) External

Q-9 difference between "padding" & "Margin".

Page No.
Date

Q-10 CSS Selectors

→ This are patterns used to select & style elements in HTML docs.

Q-11 diff betn.

(i) display - block

It takes full width available & start on new line

(ii) display - Inline

It takes necessary width & not start on new line

(iii) display : inline - block.

Combine features.

elements are laid out as inline elements but can have padding, margin & width & height prop. like block - level ele.

Q-12 Z - index in CSS.

It controls the stacking order of elements that overlap.

Q-13 CSS frameworks

- CSS frameworks are pre-prepared libraries that are meant to be used as a basis for building websites.
- They provides pre-written code in files & folders

ex. Bootstrap, Foundation, Bulma & Tailwind CSS.

Q-14 Advantages of CSS.

→ • Easy to maintain
• effectively used
• Used to represent neatly.

Q-15 limitations of CSS

→ • Browser Compatibility
• Some selector behave diff^n in diff^n browser
• There is no parent selector

Q-16 Different types of selector

Q-17 Reset CSS: Used to remove all built-in browser styling

Normalize CSS: Used to make built-in browser styling

Q-18 Pseudo elements: allows to create items that do not normally exist in the doc. tree.

:: before
:: after
:: first-letter
:: first-line
:: selection.

```
p :: first-line {
    color : red ;
}
```

Q-19. Pseudo Classes : select regular elements but under certain conditions like when the user is hovering over the link

: link

: visited

: hover

: active

: focus.

a: hover {

color : red

}

Q-20 Meaning of Cascading in CSS

→ It is the process by which styles are applied to HTML elements

Q-21 Responsive Design : ~~Use~~ It adjust its content & width according to device

Adaptive Design : According to device, it loads the content of web page that was already designed

Responsive eg : Github, Shopify etc

Adaptive eg : Apple, Amazon

Q.22 Float Property in CSS

→ Used for positioning the HTML elements horizontally either towards the left or right of container.

```
float-demo {
    float : right ;
}
```

Q.23 CSS grid : two dimensional system

Flexbox : one

Q.24 Four type of @media Properties

(i) All : default property
used for all media-type devices.

(ii) Screen : Used for comp, mobile screen

(iii) Print : Used for printers

(iv) Speech : Used for screen readers.

Q.25 Calc work

→ Perform mathematical operations on property values

```
Foo {
    width : calc (100px + 50px)
}
```

Q.26 CSS variables vs Preprocessor variables.

→ - CSS variables can be used without the need for a Preprocessor.

- CSS variables cascade but Preprocessor variable don't cascade

- CSS variables can be accessed & manipulated in JS.

Q-27 Progressive rendering

→ Used to improve the performance of a webpages

Q-28 nth child

→ Used for matching elements based on the no. that represents the position of an element based on siblings.

Q-29 CSS sprites

→ Used for combining multiple images in a single large image

Q-30 tweening in CSS

→ It is the process of filling the gaps betn the key sequences (key frames)

11:00 302.3 KB/s 5G 97%
geeksforgeeks.org/cs
CSS Tutorial CSS Exercises CSS Interview Questions CSS Selectors CSS Properties CSS Functions CSS Examples C Sign In

Our CSS Tutorial.

CSS Interview Questions For Freshers

1. What is CSS?

Cascading **S**tyle **S**heets fondly referred to as **CSS**, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages.

2. Why do we use CSS?

We use CSS because of the following reasons:

- **CSS Saves Time:** You can write CSS once and reuse the same sheet on multiple HTML pages.
- **Easy Maintenance:** To make a global change simply change the style, and all elements in all the webpages will be updated automatically.
- **Search Engines:** CSS is considered a clean coding technique, which means search engines won't have to struggle to "read" its content.

3. What are the advantages of CSS?

- CSS plays an important role, by using CSS you simply got to specify a repeated style for an element once & use it multiple times because CSS will automatically apply the required styles.
- The main advantage of CSS is that style is applied consistently across a variety of sites. One instruction can control several areas which are advantageous.
- Web designers need to use a few lines of programming for every page improving site speed.
- Cascading sheet not only simplifies website development but also simplifies maintenance as a change of one line of code affects the whole website and maintenance time.

4. What are the disadvantages of CSS?

- CSS, CSS 1 up to CSS3, result in creating confusion among web browsers.
- With CSS, what works with one browser might not always work with another. The web developers need to test for compatibility, running the program across multiple browsers.
- There exists a scarcity of security.
- After making the changes we need to confirm the compatibility if they appear. A similar change affects all the browsers.

5. What is the current version of CSS?

CSS3 is the latest version of CSS.

6. How is CSS different from CSS 3?

CSS	CSS3
CSS is capable of positioning texts and objects. CSS is somehow backward compatible with CSS3.	On the other hand, CSS3 is capable of making the web page more attractive and takes less time to create. If you write CSS3 code in CSS, it will be invalid.
Responsive designing is not supported in CSS	CSS3 is the latest version, hence it supports responsive design.
CSS cannot be split into modules.	Whereas, whereas CSS3 can be breakdown into modules.
Using CSS, we cannot build 3D animation and transformation.	But in CSS3 we can perform all kinds of animation and transformations as it supports animation and 3D transformations.
CSS is very slow as compared to CSS3	Whereas, CSS3 is faster than CSS.

To know more about the topic refer to difference between CSS 3 and CSS.

7. Suggest Some Best CSS Frameworks?

The best CSS frameworks are:

- Bootstrap

Open In App

11:00 Vo) LTE 210 KB/s 5G 97%
geeksforgeeks.org/cs
CSS Tutorial CSS Exercises CSS Interview Questions CSS Selectors CSS Properties CSS Functions CSS Examples C Sign In

7. Suggest Some Best CSS Frameworks?

The best CSS frameworks are:

- Bootstrap
- Foundation
- Bulma
- UIKit
- Semantic UI
- Materialize
- Pure
- Tailwind CSS

8. What is the syntax for CSS?

A CSS style rule consists of a selector, property, and its value. The selector points to the HTML element where CSS style is to be applied. The CSS property is separated by semicolons.

Related searches
Html Css and Javascript Interview Questions
Html and Css Interview Questions and Answe

Syntax:

```
selector {
    Property: value;
}
```

9. What are the different ways to apply CSS to a webpage?

There are three ways we can add add CSS to a webpage:

- **Inline CSS:**Inline CSS contains the CSS property in the body section attached with the element known as inline CSS. This kind of style is specified within an HTML tag using the style attribute.
- **Internal CSS:**Internal CSS can be used when a single HTML document must be styled uniquely. The CSS ruleset should be within the HTML file in the head section i.e the CSS is embedded within the HTML file.
- **External CSS:**External CSS contains a separate CSS file which contains only style property with the help of tag attributes (For example class, id, heading, ... etc). CSS property is written in a separate file with .css extension and should be linked to the HTML document using the **link** tag.

10. Which type of CSS holds the highest priority?

Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority. Multiple style sheets can be defined on one page. If for an HTML tag, styles are defined in multiple style sheets then the below order will be followed.

- As **Inline** has the highest priority, any styles that are defined in the internal and external style sheets are overridden by Inline styles.
- **Internal or Embedded** stands second in the priority list and overrides the styles in the external style sheet.
- **External style** sheets have the least priority. If there are no styles defined either in the inline or internal style sheet then external style sheet rules are applied for the HTML tags.

11. What are CSS Selectors?

CSS Selectors: CSS Selectors are used to selecting HTML elements based on their element name, id, attributes, etc. It can select one or more elements simultaneously.

Element Selectors: The element selector in CSS is used to select HTML elements which are required to be styled. In a selector declaration, there is the name of the HTML element, and the CSS properties which are to be applied to that element is written inside the brackets {}.

Syntax:

```
element_name {
    // CSS Property
}
```

id selector: The #id selector is used to set the style of the given id. The id attribute is the unique identifier in HTML document. There is only one id attribute

Open In App

- **External style** sheets have the least priority. If there are no styles defined either in the inline or internal style sheet then external style sheet rules are applied for the HTML tags.

11. What are CSS Selectors?

CSS Selectors: CSS Selectors are used to selecting HTML elements based on their element name, id, attributes, etc. It can select one or more elements simultaneously.

Element Selectors: The element selector in CSS is used to select HTML elements which are required to be styled. In a selector declaration, there is the name of the HTML element, and the CSS properties which are to be applied to that element is written inside the brackets {}.

Syntax:

```
element_name {
    // CSS Property
}
```

id selector: The #id selector is used to set the style of the given id. The id attribute is the unique identifier in an HTML document. The id selector is used with a # character.

Syntax:

```
#id_name {
    // CSS Property
}
```

class selector: The .class selector is used to select all elements which belong to a particular class attribute. To select the elements with a particular class, use the (.) character with specifying the class name. The class name is mostly used to set the CSS property to the given class.

Syntax:

```
.class_name {
    // CSS Property
}
```

12. How can we add comments in CSS?

Comments are the statements in your code that are ignored by the compiler and are not executed. Comments are used to explain the code. They make the program more readable and understandable.

Syntax:

```
/* content */
```

Comments can be single-line or multi-line.

13. What does the 'a' in rgba mean?

RGBA contains A (Alpha) which specifies the transparency of elements. The value of alpha lies between 0.0 to 1.0 where 0.0. represents fully transparent and 1.0 represents not transparent.
Syntax:

```
h1 {
    color:rgba(R, G, B, A);
}
```

14. What are CSS HSL Colors?

HSL: HSL stands for Hue, Saturation, and Lightness respectively. This format uses the cylindrical coordinate system.

- **Hue:** Hue is the degree of the color wheel. Its value lies between 0 to 360 where 0 represents red, 120 represents green and 240 represents a blue color.
- **Saturation:** It takes a percentage value, where 100% represents completely saturated, while 0% represents completely unsaturated (gray).
- **Lightness:** It takes a percentage value, where 100% represents white, while 0% represents black.

Syntax:

```
h1 {
    color:hsl(H, S, L);
}
```

14. What are CSS HSL Colors?

HSL: HSL stands for Hue, Saturation, and Lightness respectively. This format uses the cylindrical coordinate system.

- **Hue:** Hue is the degree of the color wheel. Its value lies between 0 to 360 where 0 represents red, 120 represents green and 240 represents a blue color.
- **Saturation:** It takes a percentage value, where 100% represents completely saturated, while 0% represents completely unsaturated (gray).
- **Lightness:** It takes a percentage value, where 100% represents white, while 0% represents black.

Syntax:

```
h1 {
    color:hsl(H, S, L);
}
```

Example:

```
<html>
    <head>
        <title>CSS hsl color property</title>
        <style>
            h1{
                color:hsl(120, 100%, 30%);
                text-align:center;
            }
        </style>
    </head>
    <body>
        <h1>
            GeeksforGeeks
        </h1>
    </body>
</html>
```

Output:

GeeksforGeeks

15. What are CSS backgrounds, list the properties?

The CSS background properties are used to define the background effects for elements.

CSS background properties are as follows:

1. **background-color:** This property specifies the background color of an element.
2. **background-image:** This property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element.
3. **background-repeat:** By default, the background image property repeats the image both horizontally and vertically.
4. **background-attachment:** This property is used to fix the background ground image. The image will not scroll with the page.
5. **background-position:** This property is used to set the image to a particular position.

16. What are the different CSS border properties?

CSS border properties allow us to set the style, color, and width of the border.

- **Border Style:** The border-style property specifies the type of border. None of the other border properties will work without setting the border style.
- **Border Width:** Border width sets the width of the border. The width of the border can be in px, pt, cm or thin, medium and thick.
- **Border Color:** This property is used to set the color of the border. Color can be set using the color name, hex value, or RGB value. If the color is not specified border inherits the color of the element itself.

17. What does margin: 40px 100px 120px 80px signify?

CSS margins are used to create space around the element. We can set the different sizes of margins for individual sides (top, right, bottom, left).

Margin properties can have the following values:

1. Length in cm, px, pt, etc.
2. Width % of the element.
3. Margin calculated by the browser, auto.

11:00 363.6 KB/s 5G 97%

color name, hex value, or RGB value. If the color is not specified border inherits the color of the element itself.

17. What does margin: 40px 100px 120px 80px signify?

CSS margins are used to create space around the element. We can set the different sizes of margins for individual sides (top, right, bottom, left).

Margin properties can have the following values:

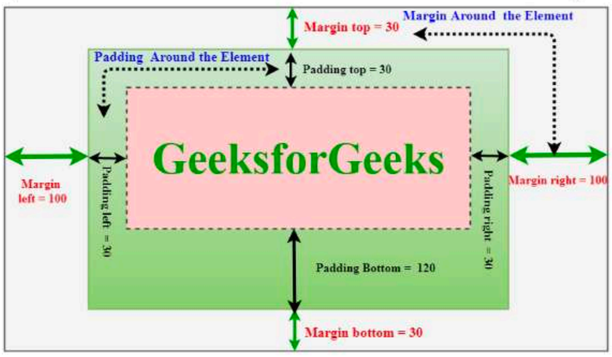
1. Length in cm, px, pt, etc.
2. Width % of the element.
3. Margin calculated by the browser: auto.

Therefore, margin: 40px 100px 120px 80px signifies:

1. top = 40px
2. right = 100px
3. bottom = 120px
4. left = 80px

18. What is the difference between margin and padding?

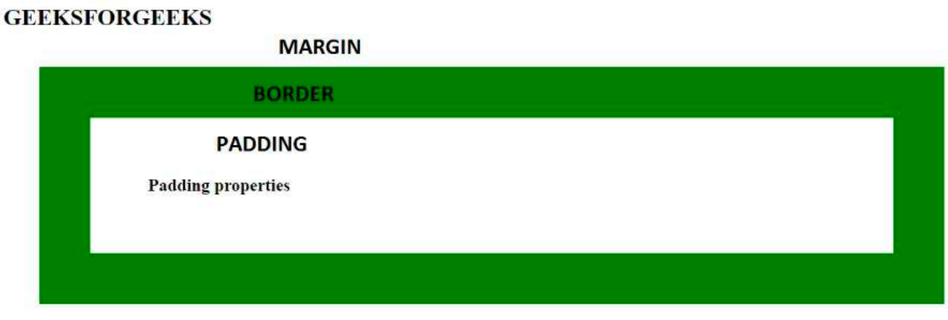
- Margin is used to create space around elements and padding is used to create space around elements inside the border.
- We can set the margin property to auto but we cannot set the padding property to auto.
- In Margin property we can allow negative or float number but in padding we cannot allow negative values.
- Margin and padding target all the 4 sides of the element. Margin and padding will work without the border property also. The difference will be more clear with the following example.



Example:

```
<!DOCTYPE html>
<html>
<head>
    <style>
        h2 {
            margin:50px;
            border:70px solid green;
            padding:80px;
        }
    </style>
</head>

<body>
    <h1>GEEKSFORGEEKS</h1>
    <h2>
        Padding properties
    </h2>
</body>
</html>
```

Output:



19. What is CSS Box Model?

The CSS box model is a container that contains the properties including borders, margin,

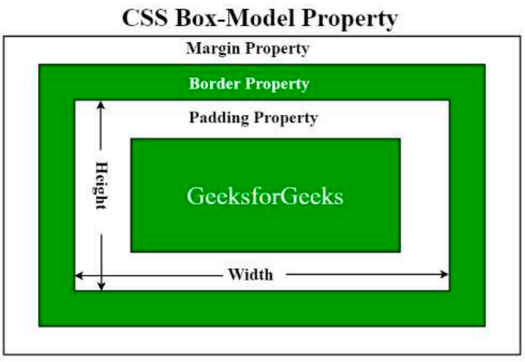
Open In App

19. What is CSS Box Model?

The CSS box model is a container that contains multiple properties including borders, margin, padding, and the content itself. It is used to create the design and layout of web pages. It can be used as a toolkit for customizing the layout of different elements. The web browser renders every element as a rectangular box according to the CSS box model.
Box-Model has multiple properties in CSS. Some of them are given below:

- borders
- margins
- padding
- Content

The following figure illustrates the box model.



- **Border Area:** It is the area between the box's padding and margin. Its dimensions are given by the width and height of the border.
- **Margin Area:** This area consists of space between border and margin. The dimensions of the Margin area are the margin-box width and the margin-box height. It is useful to separate the element from its neighbors.
- **Padding Area:** It includes the element's padding. This area is actually the space around the content area and within the border box. Its dimensions are given by the width of the padding-box and the height of the padding-box.
- **Content Area:** This area consists of content like text, images, or other media content. It is bounded by the content edge and its dimensions are given by content box width and height.

20. What is the difference between CSS border and outline?

- **CSS border** properties allow us to set the style, color, and width of the border.
- **CSS outline** property allows us to draw a line around the element, outside the border.

Example:

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <style>
        p {
            outline: 5px solid #ddd;
            border: 1px solid #000;
        }
    </style>
</head>

<body>
    <p>This is a paragraph.</p>
</body>

</html>
```

Output:

This is a paragraph.

Differences:

- Unlike borders, outlines don't allow us to set each edge to a different width, or set different colors and styles for each edge. An outline is the same on all sides.
- Outlines cannot be circular.
- Outlines do not take up space, because they are always placed on top of the box of the element.

21. How can we format text in CSS?

CSS text formatting properties are used to format and style text.
CSS text formatting includes the following properties:

Open In App

- Outlines cannot be circular.
- Outlines do not take up space, because they are always placed on top of the box of the element.

21. How can we format text in CSS?

CSS text formatting properties are used to format text and style text.
CSS text formatting includes the following properties:

1. Text-color
2. Text-alignment
3. Text-decoration
4. Text-transformation
5. Text-indentation
6. Letter spacing
7. Line height
8. Text-direction
9. Text-shadow
10. Word spacing

22. What are the different CSS link states?

A link is a connection from one web page to another web page. CSS property can be used to style the links in various different ways.

States of Link: Before discussing CSS properties, it is important to know the states of a link. Links can exist in different states and they can be styled using pseudo-classes.
There are four states of links given below:

- **a:link:** This is a normal, unvisited link.
- **a:visited:** This is a link visited by a user at least once
- **a:hover:** This is a link when the mouse hovers over it
- **a:active:** This is a link that is just clicked.

23. Can we add an image as a list item marker?

To add an image as the list-item marker in a list, we use the list-style-image property in CSS.

Syntax:

```
list-style-image: none | url | initial | inherit;
```

24. How can we hide an element in CSS?

The style **display property** is used to hide and show the content of HTML DOM by accessing the DOM element using JavaScript/jQuery.

To hide an element, set the style display property to “none”.

```
display: "none";
```

To show an element, set the style display property to “block”.

```
display:"block";
```

Example:

```
<!DOCTYPE html>
<html>

<head>
    <style>
        .visible {
            display: block;
        }

        .hidden {
            display: none;
        }
    </style>
</head>

<body>
    <h1 class="visible">visible heading</h1>
    <h1 class="hidden">hidden heading</h1>
    <p>
        Note: The h1 element with display: none;
        does not take up any space.
    </p>
</body>
```

```
list-style-image: none | url | initial | inherit;
```

24. How can we hide an element in CSS?

The style **display property** is used to hide and show the content of HTML DOM by accessing the DOM element using JavaScript/jQuery.

To hide an element, set the style display property to “none”.

```
display: "none";
```

To show an element, set the style display property to “block”.

```
display:"block";
```

Example:

```html
<!DOCTYPE html>
<html>

<head>
    <style>
        .visible {
            display: block;
        }

        .hidden {
            display: none;
        }
    </style>
</head>

<body>
    <h1 class="visible">visible heading</h1>
    <h1 class="hidden">hidden heading</h1>
    <p>
        Note: The h1 element with display: none;
        does not take up any space.
    </p>
</body>

</html>
```

Output:

visible heading

Note: The h1 element with display: none; does not take up any space.

The **visibility property** is used to hide or show the content of HTML elements. The visibility property specifies that the element is currently visible on the page. The ‘hidden’ value can be used to hide the element. This hides the element but does not remove the space taken by the element, unlike the display property.

Syntax:

```
visibility : 'hidden';
visibility :'visible';
```

Example:

```html
<!DOCTYPE html>
<html>

<head>
    <style>
        .visible {
            visibility: visible;
        }

        .hidden {
            visibility: hidden;
        }
    </style>
</head>

<body>
    <h2 class="visible">This heading is visible</h2>
    <h2 class="hidden">This heading is hidden</h2>
    <p>Note: The hidden element still takes up space.</p>
</body>
```

Open In App

to hide the element. This hides the element but does not remove the space taken by the element, unlike the display property.

Syntax:

```
visibility : 'hidden';
visibility :'visible';
```

Example:

```
<!DOCTYPE html>
<html>

<head>
    <style>
        .visible {
            visibility: visible;
        }

        .hidden {
            visibility: hidden;
        }
    </style>
</head>

<body>
    <h2 class="visible">This heading is visible</h2>
    <h2 class="hidden">This heading is hidden</h2>
    <p>Note: The hidden element still takes up space.</p>
</body>

</html>
```

Output:

This heading is visible

Note: The hidden element still takes up space.

25. What is the difference between display: none and visibility: hidden?

Both of the property is quite useful in CSS. The **visibility: "hidden";** property is used to specify whether an element is visible or not in a web document but the hidden elements take up space in the web document. The **visibility** is a property in CSS that specifies the visibility behavior of an element and **display: "none"** property is used to specify whether an element is exist or not on the website.

Syntax:

- Visibility property:

```
visibility: visible| hidden | collapse | initial | inherit;
```

- Display property:

```
display: none |  inline | block | inline-block;
```

So, the difference between **display: "none";** and **visibility: "hidden";**, right from the name itself we can tell the difference as **display: "none",** completely gets rids of the tag, as it had never existed in the HTML page whereas **visibility: "hidden";**, just makes the tag invisible it will still be on the HTML page occupying space it's just invisible.

CSS Intermediate Interview Questions

26. Can we overlap elements in CSS?

Creating an overlay effect simply means putting two div together at the same place but both the div appear when needed i.e while hovering or while clicking on one of the div to make the second one appear. Overlays are very clean and give the webpage a tidy look. It looks sophisticated and is simple to design. Overlays can create using two simple CSS properties:

The z-index propertyis used to displace elements on the z-axis i.e in or out of the screen. It is used to define the order of elements if they overlap with each other.

Syntax:

```
z-index: auto | number | initial | inherit;
```


The *position*property in CSS tells about the [illegible] positioning for an element or an HTML

can tell the difference as **display: "none"**, completely gets rids of the tag, as it had never existed in the HTML page whereas **visibility: "hidden";**, just makes the tag invisible it will still be on the HTML page occupying space it's just invisible.

CSS Intermediate Interview Questions

26. Can we overlap elements in CSS?

Creating an overlay effect simply means putting two div together at the same place but both the div appear when needed i.e while hovering or while clicking on one of the div to make the second one appear. Overlays are very clean and give the webpage a tidy look. It looks sophisticated and is simple to design. Overlays can create using two simple CSS properties:

The z-index propertyis used to displace elements on the z-axis i.e in or out of the screen. It is used to define the order of elements if they overlap with each other.

Syntax:

```
z-index: auto | number | initial | inherit;
```

The ***position***property in CSS tells about the method of positioning for an element or an HTML entity.

27. What are the various positioning properties in CSS?

The *position* property in CSS tells about the method of positioning for an element or an HTML entity. There are five different types of position properties available in CSS:

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

- **Fixed:** Any HTML element with **position: fixed** property will be positioned relative to the viewport. An element with fixed positioning allows it to remain at the same position even as we scroll the page. We can set the position of the element using the top, right, bottom, and left.
- **Static:** This method of positioning is set by default. If we don't mention the method of positioning for any element, the element has the **position: static** method by default. By defining Static, the top, right, bottom and left will not have any control over the element. The element will be positioned with the normal flow of the page.
- **Relative:** An element with **position: relative** is positioned relatively with the other elements which are sitting at top of it. If we set its top, right, bottom, or left, other elements will not fill up the gap left by this element.
- **Absolute:** An element with **position: absolute** will be positioned with respect to its parent. The positioning of this element does not depend upon its siblings or the elements which are at the same level.
- **Sticky:** Element with **position: sticky** and **top: 0** played a role between **fixed & relative** based on the position where it is placed. If the element is placed in the middle of the document then when the user scrolls the document, the sticky element starts scrolling until it touches the top. When it touches the top, it will be fixed at that place in spite of further scrolling. We can stick the element at the bottom, with the **bottom** property.

28. What is CSS overflow?

- The CSS overflow controls the big content. It tells whether to clip content or to add scroll bars. The overflow contains the following property:
- **Visible:** The content is not clipped and is visible outside the element box.
- **Hidden:** The overflow is clipped and the rest of the content is invisible.
- **Scroll:** The overflow is clipped but a scrollbar is added to see the rest of the content. The scrollbar can be horizontal or vertical.
- **Auto:** It automatically adds a scrollbar whenever it is required.
- **Overflow-x and Overflow-y:** This property specifies how to change the overflow of elements. x deals with horizontal edges and y deals with vertical edges.

29. What does the CSS float property do?

Float is a CSS property written in a CSS file or directly in the style of an element. The float property defines the flow of content. Below are the types of floating properties:

Float type	Usage
float: left	Element floats on the left side of the container
float: right	Element floats on the right side of the container

Open In App

- **Overflow-x and Overflow-y:** This property specifies how to change the overflow of elements. x deals with horizontal edges and y deals with vertical edges.

29. What does the CSS float property do?

Float is a CSS property written in a CSS file or directly in the style of an element. The float property defines the flow of content. Below are the types of floating properties:

Float type	Usage
float: left	Element floats on the left side of the container
float: right	Element floats on the right side of the container
float: inherit	The element inherits the floating property of its parent (div, table, etc...)
float: none	Element is displayed as it is (Default).

30. What does display:inline-block do?

Inline-block: This feature uses both properties: block and inline. So, this property aligns the div inline but the difference is it can edit the height and the width of the block. Basically, this will align the div both in the block and inline fashion.

Example:

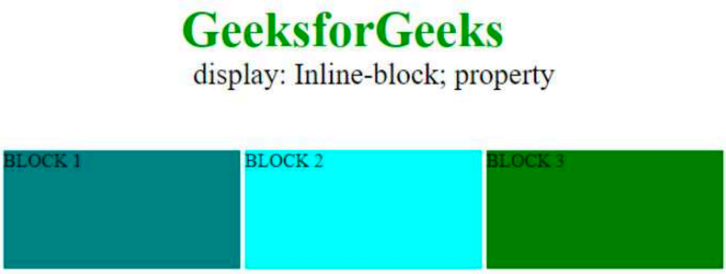
```html
<!DOCTYPE html>
<html>
    <head>
        <title>CSS | Display property</title>
        <style>
        #main{
                height: 100px;
                width: 200px;
                background: teal;
                display: inline-block;

            }
            #main1{
                height: 100px;
                width: 200px;
                background: cyan;
                display: inline-block;

            }
            #main2{
                height: 100px;
                width: 200px;
                background: green;
                display: inline-block;
            }
            .gfg {
                margin-left:200px;
                font-size:42px;
                font-weight:bold;
                color:#009900;
            }
            .geeks {
                font-size:25px;
                margin-left:210px;
            }
            .main {
                margin:50px;
            }
        </style>
    </head>
    <body>
        <div class = "gfg">GeeksforGeeks</div>
        <div class = "geeks">display: Inline-block; property</div>
        <div class = "main">
            <div id="main"> BLOCK 1 </div>
            <div id="main1"> BLOCK 2</div>
            <div id="main2">BLOCK 3 </div>
        </div>
    </body>
</html>
```

Output:

GeeksforGeeks
display: Inline-block; property

```
    </body>
</html>
```

Output:



31. How can we vertically center a text in CSS?

This solution will work for a single line and multiple lines of text, but it still requires a fixed height container:

```
div {
    height: 200px;
    line-height: 200px;
    text-align: center;
    border: 2px dashed #f69c55;
}
span {
    display: inline-block;
    vertical-align: middle;
    line-height: normal;
}
div{
    GeeksforGeeks
}
```

32. How can we center an image in CSS?

Given an image and the task is to set the image to align to center (vertically and horizontally) inside a bigger div. It can be done by using the position property of the element.

Example: This example uses the **position property** to make the image align to the center.

```
<!DOCTYPE html>
<html>

<head>
    <title>
        Horizontal and Vertical alignment
    </title>

    <!-- Style to set horizontal and
        vertical alignment -->
    <style>
        #Outer {
            border: 2px solid black;
            height: 300px;
            position: relative;
        }
        img {
            position: absolute;
            margin: auto;
            top: 0;
            left: 0;
            right: 0;
            bottom: 0;
        }
    </style>
</head>

<body>
    <div id = "Outer">
        <img src=
"https://media.geeksforgeeks.org/wp-content/uploads/gfgbg.png"/>
    </div>
</body>

</html>
```

Output:

Open In App

```
</body>

</html>
```

Output:

33. What are CSS Combinators?

CSS combinators are explaining the relationship between two selectors. CSS selectors are the patterns used to select the elements for style purposes. A CSS selector can be a simple selector or a complex selector consisting of more than one selector connected using combinators.
There are four types of combinators available in CSS which are discussed below:

- **General Sibling selector:** The general sibling selector is used to select the element that follows the first selector element and also shares the same parent as the first selector element. This can be used to select a group of elements that share the same parent element.
- **Adjacent Sibling selector:** The Adjacent sibling selector is used to select the element that is adjacent or the element that is next to the specified selector tag. This combinator selects only one tag that is just next to the specified tag.
- **Child Selector:** This selector is used to select the element that is the immediate child of the specified tag. This combinator is stricter than the descendant selector because it selects only the second selector if it has the first selector element as its parent.
- **Descendant selector:** This selector is used to select all the child elements of the specified tag. The tags can be the direct child of the specified tag or can be very deep in the specified tag. This combinator combines the two selectors such that selected elements have an ancestor same as the first selector element.

34. What are pseudo-classes in CSS?

A Pseudo class in CSS is used to define the special state of an element. It can be combined with a CSS selector to add an effect to existing elements based on their states. For Example, changing the style of an element when the user hovers over it, or when a link is visited. All of these can be done using Pseudo Classes in CSS.

Note that pseudo-class names are not case-sensitive.

Syntax:

```
selector: pseudo-class{
     property: value;
}
```

There are many Pseudo-classes in CSS but the ones which are most commonly used are as follows:

- **:hover Pseudo-class:** This pseudo-class is used to add a special effect to an element when our mouse pointer is over it. The below example demonstrates that when your mouse enters the box area, its background color changes from yellow to orange.
- **:active Pseudo-class:** This pseudo-class is used to select an element that is activated when the user clicks on it. The following example demonstrates that when you click on the box, its background color changes for a moment.
- **:focus Pseudo-class:** This pseudo-class is used to select an element that is currently focussed by the user. It works on user input elements used in forms and is triggered as soon as the user clicks on it. In the following example, the background color of the input field which is currently focused changes.
- **:visited Pseudo-class:** This pseudo-class is used to select the links which have been already visited by the user. In the following example, the color of the link changes once it is visited.

35. What are pseudo-elements in CSS?

Pseudo Elements: Pseudo-element in CSS is used to add style to specified parts of an element. Example: Using style before or after an element.

Syntax:

```
selector::pseudo-element {
     property:value;
```

clicks on it. In the following example, the background color of the input field which is currently focused changes.

- **:visited Pseudo-class:** This pseudo-class is used to select the links which have been already visited by the user. In the following example, the color of the link changes once it is visited.

35. What are pseudo-elements in CSS?

Pseudo Elements: Pseudo-element in CSS is used to add style to specified parts of an element. Example: Using style before or after an element.

Syntax:

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

Use of Pseudo-Element: Below is some examples to describe the use of pseudo-element.

- **::before Pseudo-element:** It is used to add some CSS property before an element when that element is called.
- **::after Pseudo-element:** It is used to add some CSS property after an element when that element is called.
- **::first-letter Pseudo-element:** It is used to make changes to the first letter of an element.
- **::first-line Pseudo-element:** It is used to make changes to the first line of an element.

36. How can we add gradients in CSS?

There are two types of Gradients:

1. Linear Gradients: It includes the smooth color transitions to going up, down, left, right, and diagonally. A minimum of two colors are required to create a linear gradient. More than two color elements can be possible in linear gradients. The starting point and the direction are needed for the gradient effect.

Syntax:

```
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
```

2. CSS Radial Gradients: A radial gradient differs from a linear gradient. It starts at a single point and emanates outward. By default, the first color starts at the center position of the element and then fades to the end color towards the edge of the element. Fade happens at an equal rate until specified.

Syntax:

```
background-image: radial-gradient(shape size at position, start-color, ..., l
```

37. Can we add 2D transformations to our project using CSS?

Yes, we can, a transformation modifies an element by its shape, size, and position. It transforms the elements along the X-axis and Y-axis.
There are six main types of 2D transformations which are listed below:

- translate()
- rotate()
- scale()
- skewX()
- skewY()
- matrix()

38. Can we add 3D transformations to our project using CSS?

Yes, it allows changing elements using 3D transformations. In 3D transformation, the elements are rotated along the X-axis, Y-axis, and Z-axis.

There are three main types of transformation which are listed below:

- rotateX()
- rotateY()
- rotateZ()

39. What are CSS transitions?

Transitions in CSS allow us to control the way in which transition takes place between the two states of the element.

The transition allows us to determine how the change in color takes place. We can use the transitions to animate the changes and make the changes visually appealing to the user and hence giving a better user experience and interactivity. In this article, we will show you how to animate

Open In App

11:00 Vo) LTE 0.75 KB/s 5G 97%

- rotateX()
- rotateY()
- rotateZ()

39. What are CSS transitions?

Transitions in CSS allow us to control the way in which transition takes place between the two states of the element.

The transition allows us to determine how the change in color takes place. We can use the transitions to animate the changes and make the changes visually appealing to the user and hence, giving a better user experience and interactivity. In this article, we will show you how to animate the transition between the CSS properties.

There are four CSS properties that you should use, all or in part (at least two, transition-property and transition-duration, is a must), to animate the transition. All these properties must be placed along with other CSS properties of the initial state of the element:

- **transition-property:** This property allows you to select the CSS properties which you want to animate during the transition(change).

Syntax:

```
transition-property: none | all | property | property1,
property2, ..., propertyN;
```

- **transition-duration:** This property allows you to determine how long it will take to complete the transition from one CSS property to the other.

Syntax:

```
transition-duration: time;
```

Here, **time** can be in seconds(s) or milliseconds(ms), you should use 's' or 'ms' after the number (without quotes).

- **transition-timing-function:** This property allows you to determine the speed of change and the manner of change, during the transition. Like, the change should be fast at the beginning and slow at the end, etc.

Syntax:

```
transition-timing-function: ease|ease-in|ease-out|ease-in-out|linear|
step-start|step-end;
```

- **transition-delay:** This property allows you to determine the amount of time to wait before the transition actually starts to take place.

Syntax:

```
transition-delay: time;
```

Here, again, **time** can be in seconds(s) or milliseconds(ms), and you should use 's' or 'ms' after the number (without quotes).

- **The Shorthand Property** You can combine all the four transition properties mentioned above, into one single shorthand property, according to the syntax given below. This saves us from writing long codes and prevents us from getting messy. Note the ordering of property, it has significance.

Syntax:

```
transition: (property name) | (duration) | (timing function) | (delay);
```

40. How can we animate using CSS?

CSS Animations is a technique to change the appearance and behavior of various elements in web pages. It is used to control the elements by changing their motions or display. It has two parts, one which contains the CSS properties which describe the animation of the elements and the other contains certain keyframes which indicate the animation properties of the element and the specific time intervals at which those have to occur.

The @keyframes rule: Keyframes are the foundations with the help of which CSS Animations works. They define the display of the animation at the respective stages of its whole duration. For example: In the following code, the paragraph c[redacted]ges its color with time. At 0% completion, it is red, at 50% completion it is of orange color [redacted]mpletion i.e. at 100%, it is brown.

Open In App

11:00 0.72 KB/s 5G 97%


```
transition: (property name) | (duration) | (timing function) | (delay);
```

40. How can we animate using CSS?

CSS Animations is a technique to change the appearance and behavior of various elements in web pages. It is used to control the elements by changing their motions or display. It has two parts, one which contains the CSS properties which describe the animation of the elements and the other contains certain keyframes which indicate the animation properties of the element and the specific time intervals at which those have to occur.

The @keyframes rule: Keyframes are the foundations with the help of which CSS Animations works. They define the display of the animation at the respective stages of its whole duration. For example: In the following code, the paragraph changes its color with time. At 0% completion, it is red, at 50% completion it is of orange color and at full completion i.e. at 100%, it is brown.

Example:

```html
<!DOCTYPE html>
<html>
    <head>
        <style>
            #gfg {
                animation-name: color;
                animation-duration: 25s;
                padding-top:30px;
                padding-bottom:30px;
                font-family:Times New Roman;
            }
            #geeks {
                font-size: 40px;
                text-align:center;
                font-weight:bold;
                color:#090;
                padding-bottom:5px;
            }
            #geeks1 {
                font-size:17px;
                font-weight:bold;
                text-align:center;
            }
            @keyframes color {
                0% {
                    background-color: red;
                }
                50% {
                    background-color: orange;
                }
                100% {
                    background-color: brown;
                }
            }
        </style>
    </head>
    <body>
        <div id = "gfg">
            <div id = "geeks">GeeksforGeeks</div>
            <div id = "geeks1">
                A computer science portal for geeks
            </div>
        </div>
    </body>
</html>
```

Output:

CSS Interview Questions for Advanced

41. What does the CSS box-sizing property do?

The **box-sizing** CSS property defines how the user should calculate the total width and height of an element i.e. padding and borders, are to be included or not.

Syntax:

```
box-sizing: content-box|border-box;
```

Property Values:

- **content-box:** This is the default value of the box-sizing property. In this mode, the width and height properties include only the content. Borders and padding are not included in it i.e. if we set an element's width to 200 pixels, then the element's content box will be 200 pixels wide, and

Open In App

11:00 Vo)) LTE 0.88 KB/s 5G 97%

CSS Interview Questions for Advanced

41. What does the CSS box-sizing property do?

The **box-sizing** CSS property defines how the user should calculate the total width and height of an element i.e. padding and borders, are to be included or not.

Syntax:

```
box-sizing: content-box|border-box;
```

Property Values:

- **content-box:** This is the default value of the box-sizing property. In this mode, the width and height properties include only the content. Border and padding are not included in it i.e if we set an element's width to 200 pixels, then the element's content box will be 200 pixels wide, and the width of any border or padding will be added to the final rendered width.
- **border-box:** In this mode, the width and height properties include content, padding, and borders i.e if we set an element's width to 200 pixels, that 200 pixels will include any border or padding we added, and the content box will shrink to absorb that extra width. This typically makes it much easier to size elements.

42. How can we make our website responsive using CSS?

Media query is used to create a responsive web design. It means that the view of a web page differs from system to system based on screen or media types.

Media queries can be used to check many things:

- width and height of the viewport
- width and height of the device
- Orientation
- Resolution

A media query consist of a media type that can contain one or more expression which can be either true or false. The result of the query is true if the specified media matches the type of device the document is displayed on. If the media query is true then a style sheet is applied.

Syntax:

```
@media not | only mediatype and (expression) {
    // Code content
}
```

43. What is CSS flexbox?

It is also called a flexible box model. It is basically a layout model that provides an easy and clean way to arrange items within a container. Flexbox is different from the block model which is vertically biased and the inline which is horizontally biased.

Flexbox was created for small-scale layouts and there's another standard called grids which are geared more towards larger-scale layouts, It works similar to the way to Twitter bootstrap grid system works. Flexbox is responsive and mobile-friendly. To start with flexbox firstly create a flex container. To create a flex container set the display property to flex.

Syntax:

```
.main-container {
    display: flex;
}
```

Flex Properties:

- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

44. What is CSS Grid?

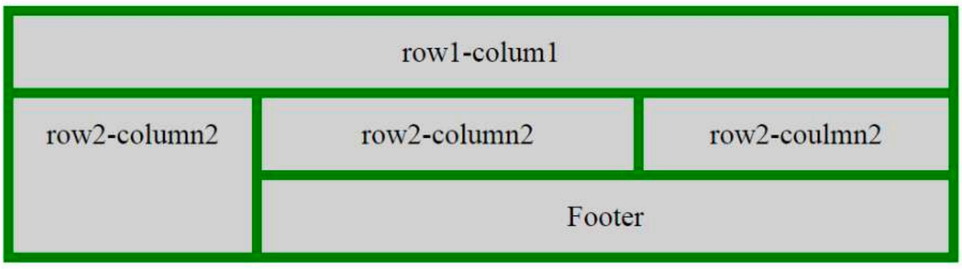
It is a CSS property that offers a grid-based layout system, with rows and columns, making it easier to design web pages without floats and positioning.

Open In App

- justify-content
- align-items
- align-content

44. What is CSS Grid?

It is a CSS property that offers a grid-based layout system, with rows and columns, making it easier to design web pages without floats and positioning.



Syntax:

```
grid: none|grid-template-rows / grid-template-columns|grid-template-areas|
grid-template-rows / [grid-auto-flow] grid-auto-columns|[grid-auto-flow]
grid-auto-rows / grid-template-columns|initial|inherit;
```

45. What is the difference between flexbox and grid?

1. Dimensionality and Flexibility:

- Flexbox offers greater control over alignment and space distribution between items. Being one-dimensional, Flexbox only deals with either columns or rows.
- The grid has two-dimension layout capabilities which allow flexible widths as a unit of length. This compensates for the limitations in Flex.

2. Alignment:

- Flex Direction allows developers to align elements vertically or horizontally, which is used when developers create and reverse rows or columns.
- CSS Grid deploys fractional measure units for grid fluidity and auto-keyword functionality to automatically adjust columns or rows.

3. Item Management

- Flex Container is the parent element while Flex Item represents the children. The Flex Container can ensure balanced representation by adjusting item dimensions. This allows developers to design for fluctuating screen sizes.
- Grid supports both implicit and explicit content placement. Its inbuilt automation allows it to automatically extend line items and copy values into the new creation from the preceding item.

Property	Grid	Flexbox
Dimension	Two – Dimensional	One – Dimensional
Features	Can flex combination of items through space-occupying Features	Can push content element to extreme alignment
Support Type	Layout First	Content First

46. What is the best way to include a CSS file? Why use @import?

The External Style Sheet (using HTML <link> Tag) is the best method that is used to link the element. Maintaining and re-using the CSS file across different pages is easy and efficient. The <link> tag is placed in the HTML <head> element. To specify a media *type="text/css"* for a Cascading Style Sheet <type> attribute which is used to ignore style sheet types that are not supported in a browser.

@import rule: The @import rule is used to import one style sheet into another style sheet. This rule also supports media queries so that the user can import the media-dependent style sheet. The @import rule must be declared at the top of the document after any @charset declaration.

Characteristics of @import:

- The @import at-rule is used to import a style sheet into an HTML page or another style sheet.
- The @import at-rule is also used to add media queries, therefore import is media-dependent.
- It is always to be declared at the top of the document.

type

46. What is the best way to include a CSS file? Why use @import?

The External Style Sheet (using HTML <link> Tag) is the best method that is used to link the element. Maintaining and re-using the CSS file across different pages is easy and efficient. The <link> tag is placed in the HTML <head> element. To specify a media *type="text/css"* for a Cascading Style Sheet <type> attribute which is used to ignore style sheet types that are not supported in a browser.

@import rule: The @import rule is used to import one style sheet into another style sheet. This rule also supports media queries so that the user can import the media-dependent style sheet. The @import rule must be declared at the top of the document after any @charset declaration.

Characteristics of @import:

- The @import at-rule is used to import a style sheet into an HTML page or another style sheet.
- The @import at-rule is also used to add media queries, therefore import is media-dependent.
- It is always to be declared at the top of the document.

Syntax:

```
@import url|string list-of-mediaqueries;
```

47. How case-sensitive is CSS?

All CSS style sheets are case-insensitive, except for portions that are not under the control of CSS. For example, the case sensitivity due to values of the HTML attributes "id" and "class", font names, and URIs lies outside the scope of this specification.

48. What does CSS Animations allow?

CSS allows the animation of HTML elements without using JavaScript. An animation lets an element systematically and with proper timing, change from one style to another. You can change whatever CSS properties you want, and end a number of times, as you want it. To use CSS animation, you must first specify some @keyframes for the animation. @keyframes will describe which styles that element will have at specific times. We will be using a basic example such as the animation of a battery charging.

The @keyframes property has the option to divide the animation time into parts/percentage and perform an activity that is specified for that part of the whole duration of the animation. The @keyframes property is given to each animation according to the name of that animation. It allows you to run the animation infinitely as well.

49. What is @keframes used for?

Keyframes are the foundations with the help of which CSS Animations works. They define the display of the animation at the respective stages of its whole duration. For example: In the following code, the paragraph changes its color with time. At 0% completion, it is red, at 50% completion it is of orange color and at full completion i.e. at 100%, it is brown.

Example:

```html
<!DOCTYPE html>
<html>

<head>
    <style>
        div {
            width: 200px;
            height: 200px;
            margin: 200px;
            border-radius: 100px;
            background-color: red;
            animation: circle 8s infinite;
        }

        @keyframes circle {
            0% {
                background-color: red;
            }

            25% {
                background-color: yellow;
            }

            50% {
                background-color: blue;
```

you to run the animation infinitely as well.

49. What is @keframes used for?

Keyframes are the foundations with the help of which CSS Animations works. They define the display of the animation at the respective stages of its whole duration. For example: In the following code, the paragraph changes its color with time. At 0% completion, it is red, at 50% completion it is of orange color and at full completion i.e. at 100%, it is brown.

Example:

```html
<!DOCTYPE html>
<html>

<head>
    <style>
        div {
            width: 200px;
            height: 200px;
            margin: 200px;
            border-radius: 100px;
            background-color: red;
            animation: circle 8s infinite;
        }

        @keyframes circle {
            0% {
                background-color: red;
            }

            25% {
                background-color: yellow;
            }

            50% {
                background-color: blue;
            }

            100% {
                background-color: green;
            }
        }
    </style>
</head>

<body>
    <div></div>
</body>

</html>
```

Output:

50. What are CSS counters?

Counters in CSS are basically variables that can be used for numbering and values of CSS counters may be incremented by CSS rules. For example, CSS counters can be used to increment the numbering of the headings automatically. In HTML, the <ol> tag is used to give the ordered numbers to list items but CSS contains a counter to give order elements in some other fashion.

CSS counters properties: CSS counters contains the following properties:

- **counter-reset:** It is used to reset a counter.
- **counter-increment:** It basically increments a counter value.
- **content:** It is used to generate content.
- **counter() or counters() function:** The value of a counter can be displayed using either the counter() or counters() function in a content property. These two functions basically used to add the value of a counter to the element.

Initialization of the CSS Counter: To use the CSS counter property firstly it must be created with the counter-reset property and the first step is resetting the counter. The counter by default initialized to a value 0(zero) with the counter-reset property.

Syntax:

50. What are CSS counters?

Counters in CSS are basically variables that can be used for numbering and values of CSS counters may be incremented by CSS rules. For example, CSS counters can be used to increment the numbering of the headings automatically. In HTML, the <ol> tag is used to give the ordered numbers to list items but CSS contains a counter to give order elements in some other fashion.

CSS counters properties: CSS counters contains the following properties:

- **counter-reset:** It is used to reset a counter.
- **counter-increment:** It basically increments a counter value.
- **content:** It is used to generate content.
- **counter() or counters() function:** The value of a counter can be displayed using either the counter() or counters() function in a content property. These two functions basically used to add the value of a counter to the element.

Initialization of the CSS Counter: To use the CSS counter property firstly it must be created with the counter-reset property and the first step is resetting the counter. The counter by default initialized to a value 0(zero) with the counter-reset property.

Syntax:

```
counter-reset: myCounter;
```

Incrementation and Use of CSS Counter: To increment the counter use the CSS counter-increment property.

Syntax:

```
counter-increment: myCounter;
```

The counter() or counters() function in content is used to display the content in a particular order.

Syntax:

```
content: counter(myCounter);
```

51. What is meant by universal selector?

The *** selector in CSS** is used to select all the elements in an HTML document. It also selects all elements which are inside under another element. It is also called the universal selector.

Syntax:

```
* {
    // CSS property
}
```

To know more about the topic refer to Universal selector.

52. What is RWD?

Responsive Web Design comprises two words i.e., responsive and web design. Responsive means to respond and web design means to design a website. Therefore, responsive web design generally means the website that responds to or resizes or adjusts itself depending upon the screen size it is being seen through.

It automatically adjusts to fit the user's screen whether it's desktop, laptop, mobile, tablet, etc. It only uses one layout for a web page and it can be done either using CSS and HTML or CSS3 and HTML5.

To know more about the topic refer to RWD .

53. What is thedifference between class and id selector?

Id selector("#"): The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element. It is written with the hash character (#), followed by the id of the element.

- **Syntax:**

```
#element_id_name{
    // CSS properties
}
```

53. What is thedifference between class and id selector?

Id selector("#"): The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element. It is written with the hash character (#), followed by the id of the element.

- **Syntax:**

```
#element_id_name{
    // CSS properties
}
```

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

- **Syntax:**

```
.element_class_name{
    // CSS properties
}
```

Class "."	Id "#"
The class selector "." is used to represent class="class_name" in HTML element.	The id selector "#" is used to represent id="id_name" in HTML element.
Each elements can contain more that one "." selector means that elements is containing more than one class which is separated by space, they will be selected by multiple dots like .class1 .class2 .... and so on.	Each element can contain only one "#" selector, not more than one unlike class selectors.
The "." selectors are not unique, same selectors can applicable on multiple elements, if the HTML elements holds the same class property like a list of elements can contains the same class.	

54. How can we use pagination in CSS?

Pagination is the process of dividing the document into pages and providing them with numbers.

Types of Pagination: There are many types of pagination in CSS. Some of them are given below:

1. Simple Pagination
2. Active and Hoverable Pagination
3. Rounded Active and Hoverable Buttons
4. Hoverable Transition Effect
5. Bordered Pagination
6. Rounded Border Pagination
7. Centered Pagination
8. Space between Pagination
9. Pagination Size

Simple Pagination: This is the basic form of pagination.

Syntax:

```
.pagination {
    display:type
}
.pagination body {
    color:colorname
    decoration:type
}
```

55. What is CSS Image reflection?

The box-reflect property is used to create an image reflection.

Attributes:

- **below:** to create a reflection below the original image
- **above:** to create a reflection above the original image
- **left:** to create a reflection on the left side of the original image
- **right:** to create a reflection on the right side of the original image

Open In App

Simple Pagination: This is the basic form of pagination.

Syntax:

```
.pagination {
    display:type
}
.pagination body {
    color:colorname
    decoration:type
}
```

55. What is CSS Image reflection?

The box-reflect property is used to create an image reflection.

Attributes:

- **below:** to create a reflection below the original image
- **above:** to create a reflection above the original image
- **left:** to create a reflection on the left side of the original image
- **right:** to create a reflection on the right side of the original image

Example:

```html
<!DOCTYPE html>
<html>

<head>
    <style>
        img {
            -webkit-box-reflect: right;
        }
    </style>
</head>

<body>
    <h1>CSS Image Reflection</h1>
    <p>Shows the reflection of the image on right side:</p>
    <img src=
"https://media.geeksforgeeks.org/wp-content/uploads/20210322123023/gfg2.png">
</body>

</html>
```

Output:

56. How can we create multiple columns of text-like newspapers using CSS?

The multiple columns are used to create column layouts on the web pages. There are many column properties in CSS which are listed below:

- column-count
- column-gap
- column-rule-style
- column-rule-width
- column-rule-color
- column-rule
- column-span
- column-width

Example:

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Column-count property</title>
        <style>
        .geeks_content {
            -webkit-column-count: 3;
```

Open In App

56. How can we create multiple columns of text-like newspapers using CSS?

The multiple columns are used to create column layouts on the web pages. There are many column properties in CSS which are listed below:

- column-count
- column-gap
- column-rule-style
- column-rule-width
- column-rule-color
- column-rule
- column-span
- column-width

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Column-count property</title>
        <style>
        .geeks_content {
            -webkit-column-count: 3;
            -moz-column-count: 3;
            column-count: 3;
            padding-top:35px;
            text-align:justify;
        }
        .gfg {
            text-align:center;
            font-size:40px;
            font-weight:bold;
            color:green;
        }
        .geeks {
            text-align:center;
        }
        </style>
    </head>
    <body>
        <div class="gfg">GeeksforGeeks</div>
        <div class = "geeks">A computer science portal for geeks</div>
        <div class="geeks_content">
            Sudo Placement: Prepare for the Recruitment
            drive of product based companies like Microsoft,
            Amazon, Adobe etc with a free online placement
            preparation course. The course focuses on
            various MCQ's & Coding question likely to be
            asked in the interviews & make your upcoming
            placement season efficient and successful.
            Placement preparation solely depends on the
            company for which you are preparing. There
            are basically three different categories
            into which we can divide the companies visiting
            campuses for placements based on their recruitment
            process. Mass Recruiters, Tech Giants, Others / Start-ups
            Companies belonging to the above categories have
            their own recruitment process. In this course, we
            will try to cover every possible detail required
            to know for cracking interview of the companies
            falling in each of the above categories.
        </div>
    </body>
</html>
```

Output:

57. How can we give a shadow effect to our text in CSS?

The approach of this article is to add a shadow using the *text-shadow* property in CSS. This property accepts a list of a comma-separated [illegible] allows to be applied to the text. The default value of the *text-shadow* property is "none".

Open In App

course focuses on various MCQ's & Coding question likely to be asked in the interviews & make your upcoming placement season we can divide the companies visiting campuses for placements based on their recruitment process. Mass Recruiters, Tech required to know for cracking interview of the companies falling in each of the above categories.

57. How can we give a shadow effect to our text in CSS?

The approach of this article is to add a shadow using the *text-shadow* property in CSS. This property accepts a list of a comma-separated list of shadows to be applied to the text. The default value of the *text-shadow* property is "none".

Syntax:

```
text-shadow: h-shadow v-shadow blur-radius color|none|initial|
```

58. What is !important?

The !important property in CSS is used to provide more weight (importance) than normal property. In CSS, **the !important means that "this is important", ignore all the subsequent rules, and apply !important rule and the !important keyword must be placed at the end of the line,** immediately before the semicolon.

- In other words, it adds importance to all the sub-properties that the shorthand property represents.
- In normal use, a rule defined in an external style sheet which is overruled by a style defined in the head of the document, which in turn, is overruled by an inline style within the element itself (assuming equal specificity of the selectors).
- Defining a rule with the !important attribute that discards the normal concerns as regards the later rule overriding the earlier ones.
- So, it is used for overriding the styles that are previously declared in other style sources, in order to achieve a certain design.

Syntax:

```
element {
    color: blue  !important;
    font-size: 14px !important;
    ...
}
```

59. What is specificity in CSS?

When more than one set of CSS rules applies to the same element, the browser will have to decide which specific set will be applied to the element. The rules the browser follows are collectively called **Specificity**

Specificity Rules include:

- CSS style applied by referencing external stylesheet has the lowest precedence and is overridden by Internal and inline CSS.
- Internal CSS is overridden by inline CSS.
- Inline CSS has the highest priority and overrides all other selectors.

Specificity Hierarchy: Every element selector has a position in the Hierarchy.

1. **Inline style:** Inline style has the highest priority.
2. **Identifiers(ID):** ID has the second-highest priority.
3. **Classes, pseudo-classes, and attributes:** Classes, pseudo-classes, and attributes have come next.
4. **Elements and pseudo-elements:** Elements and pseudo-elements have the lowest priority.

60. What are the attribute selectors?

The CSS Attribute Selector is used to select an element with some specific attribute or attribute value. It is an excellent way to style the HTML elements by grouping them based on some specific attributes and the attribute selector will select those elements with similar attributes.
There are several types of attribute selectors which are discussed below:

- **[attribute] Selector:** This type of attribute selector is used to select all the elements that have the specified attribute and applies the CSS property to that attribute. For example, the selector [class] will select all the elements with the style attribute.
- **[attribute = "value"] Selector:** This selector is used to select all the elements whose attribute has the value exactly the same as the specified value.
- **[attribute~="value"] Selector:** This selector is used to select all the elements whose attribute value is a list of space-separated values, one of which is exactly equal to the specified value.
- **[attribute|="value"] Selector:** This selector is used to select all the elements whose attribute has a hyphen-separated list of values beginning with the specified value. The value has to be a whole word either alone or followed by a hyphen.

Open In App

11:01 19.9 KB/s 5G 97%
geeksforgeeks.org/cs
CSS Tutorial CSS Exercises CSS Interview Questions CSS Selectors CSS Properties CSS Functions CSS Examples C Sign In

2. **Identifiers(ID):** ID has the second-highest priority.
3. **Classes, pseudo-classes, and attributes:** Classes, pseudo-classes, and attributes have come next.
4. **Elements and pseudo-elements:** Elements and pseudo-elements have the lowest priority.

60. What are the attribute selectors?

The CSS Attribute Selector is used to select an element with some specific attribute or attribute value. It is an excellent way to style the HTML elements by grouping them based on some specific attributes and the attribute selector will select those elements with similar attributes.
There are several types of attribute selectors which are discussed below:

- **[attribute] Selector:** This type of attribute selector is used to select all the elements that have the specified attribute and applies the CSS property to that attribute. For example, the selector [class] will select all the elements with the style attribute.
- **[attribute = "value"] Selector:** This selector is used to select all the elements whose attribute has the value exactly the same as the specified value.
- **[attribute~="value"] Selector:** This selector is used to select all the elements whose attribute value is a list of space-separated values, one of which is exactly equal to the specified value.
- **[attribute|="value"] Selector:** This selector is used to select all the elements whose attribute has a hyphen-separated list of values beginning with the specified value. The value has to be a whole word either alone or followed by a hyphen.
- **[attribute^="value"] Selector:** This selector is used to select all the elements whose attribute value begins with the specified value. The value doesn't need to be a whole word.
- **[attribute$="value"] Selector:** This selector is used to select all the elements whose attribute value ends with the specified value. The value doesn't need to be a whole word.
- **[attribute*="value"] Selector:** This selector selects all the elements whose attribute value contains the specified value present anywhere. The value doesn't need to be a whole word.

Commonly Asked CSS Interview Questions

1. **What is CSS, and why is it used?**
2. **What are the different ways to apply CSS to a webpage?**
3. **What are the differences between `class` and `id` selectors in CSS?**
4. **What are the CSS box model properties?**
5. **What is the difference between `display: none` and `visibility: hidden`?**
6. **What is Flexbox, and how does it work?**
7. **What are media queries in CSS?**
8. **What is the difference between `absolute`, `relative`, `fixed`, and `sticky` positioning in CSS?**
9. **What is the z-index in CSS?**
10. **What is the difference between the Flexbox and Grid?**

Conclusion

If you're preparing for roles such as a Web Designer that primarily require proficiency in CSS, the interview questions provided here will be instrumental in your preparation. These questions encompass fundamental aspects of CSS, ensuring you're well-prepared for such positions.

For those targeting roles like Web Developer or other technical positions that demand a more comprehensive understanding of web technologies, it's advisable to delve into additional interview questions covering HTML, JavaScript, and other relevant areas. A broader preparation will enhance your readiness for a wider array of opportunities in the tech industry.

Comment More info Campus Training Program

Next Article
JavaScript Interview Questions and Answers

Similar Reads

1. Accenture Interview Questions
2. TCS Digital Interview Questions
3. TCS Interview Questions
4. Software Developer Interview Questions
5. Standard Chartered Interview Questions
6. JCS Digital Interview Experiences and Interview Questions

Open In App

11:01 95.4 KB/s 5G 97%
interviewbit.com/css-i

This article covers the most frequently asked CSS and CSS3 interview questions for freshers and experienced candidates.

CSS Interview Questions For Freshers

1. What are the advantages of using CSS?

The main advantages of CSS are given below:

- **Separation of content from presentation -** CSS provides a way to present the same content in multiple presentation formats in mobile or desktop or laptop.
- **Easy to maintain -** CSS, built effectively can be used to change the look and feel complete by making small changes. To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Bandwidth -** Used effectively, the style sheets will be stored in the browser cache and they can be used on multiple pages, without having to download again.

2. How do you specify units in the CSS?. What are the different ways to do it?

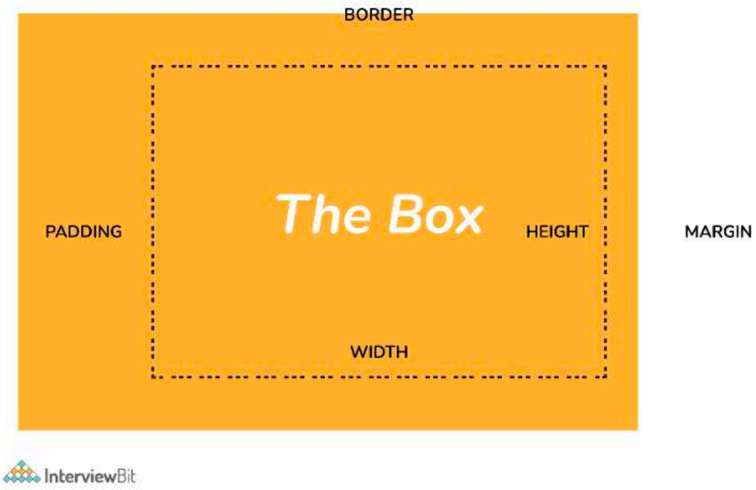
There are different ways to specify units in CSS like px, em, pt, percentage (%). px(Pixel) gives fine-grained control and maintains alignment because 1 px or multiple of 1 px is guaranteed to look sharp. px is not cascade. em maintains relative size. you can have responsive fonts. Em, will cascade 1em is equal to the current font-size of the element or the browser default. If u sent font-size to 16px then 1em = 16px. The common practice is to set default body font-size to 62.5% (equal to 10px).

pt(point) are traditionally used in print. 1pt = 1/72 inch and it is a fixed-size unit.

%(percentage) sets font-size relative to the font size of the body. Hence, you have to set the font-size of the body to a reasonable size.

3. What is the Box model in CSS? Which CSS properties are a part of it?

A rectangle box is wrapped around every HTML element. The box model is used to determine the height and width of the rectangular box. The CSS Box consists of Width and height (or in the absence of that, default values and the content inside), padding, borders, margin.



- **Content:** Actual Content of the box where the text or image is placed.
- **Padding:** Area surrounding the content (Space between the border and content).
- **Border:** Area surrounding the padding.
- **Margin:** Area surrounding the border.

4. What are the limitations of CSS?

Disadvantages of CSS are given below:

- **Browser Compatibility:** Some style selectors are supported and some are not. We have to determine which style is supported or not using the @support selector).
- **Cross Browser issue:** Some selectors behave differently in a different browser).
- **There is no parent selector:** Currently, Using CSS, you can't select a parent tag.

5. How to include CSS in the webpage?

There are different ways to include a CSS in a webpage,

1 - External Style Sheet: An external file linked to your HTML document: Using link tag, we can link the style sheet to the HTML page.

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

2 - Embed CSS with a style tag: A set of CSS styles included within your HTML page

```
<style type="text/css">
```

Get Ready with Free Mock Coding Interview

- **Cross Browser Issue:** Some selectors behave differently in a different browser).
- **There is no parent selector:** Currently, Using CSS, you can't select a parent tag.

5. How to include CSS in the webpage?

There are different ways to include a CSS in a webpage,

1 - External Style Sheet: An external file linked to your HTML document: Using link tag, we can link the style sheet to the HTML page.

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

2 - Embed CSS with a style tag: A set of CSS styles included within your HTML page.

```
<style type="text/css">

/*Add style rules here*/

</style>
```

Add your CSS rules between the opening and closing style tags and write your CSS exactly the same way as you do in stand-alone stylesheet files.

3 - Add inline styles to HTML elements(CSS rules applied directly within an HTML tag.): Style can be added directly to the HTML element using a style tag.

```
<h2 style="color:red;background:black">Inline Style</h2>
```

4 - Import a stylesheet file (An external file imported into another CSS file): Another way to add CSS is by using the @import rule. This is to add a new CSS file within CSS itself.

```
@import "path/to/style.css";
```

6. What are the different types of Selectors in CSS?

A CSS selector is the part of a CSS ruleset that actually selects the content you want to style. Different types of selectors are listed below.

Universal Selector: The universal selector works like a wildcard character, selecting all elements on a page. In the given example, the provided styles will get applied to all the elements on the page.

```
* {
  color: "green";
  font-size: 20px;
  line-height: 25px;
}
```

Element Type Selector: This selector matches one or more HTML elements of the same name. In the given example, the provided styles will get applied to all the ul elements on the page.

```
ul {
  line-style: none;
  border: solid 1px #ccc;
}
```

ID Selector: This selector matches any HTML element that has an ID attribute with the same value as that of the selector. In the given example, the provided styles will get applied to all the elements having ID as a container on the page.

```
#container {
  width: 960px;
  margin: 0 auto;
}

<div id="container"></div>
```

Class Selector: The class selector also matches all elements on the page that have their class attribute set to the same value as the class. In the given example, the provided styles will get applied to all the elements having ID as the box on the page.

```
.box {
  padding: 10px;
  margin: 10px;
  width: 240px;
}

<div class="box"></div>
```

Descendant Combinator: The descendant selector or, more accurately, the descendant combinator lets you combine two or more selectors so you can be more specific in your selection method.

```
#container .box {
        float: left;
        padding-bottom: 15px;
}

<div id="container">
        <div class="box"></div>

        <div class="box-2"></div>
</div>

<div class="box"></div>
```

Get Ready with Free Mock Coding Interview

Class Selector: The class selector also matches all elements on the page that have their class attribute set to the same value as the class. In the given example, the provided styles will get applied to all the elements having ID as the box on the page.

```
.box {
  padding: 10px;
  margin: 10px;
  width: 240px;
}

<div class="box"></div>
```

Descendant Combinator: The descendant selector or, more accurately, the descendant combinator lets you combine two or more selectors so you can be more specific in your selection method.

```
#container .box {
        float: left;
        padding-bottom: 15px;
}

<div id="container">
        <div class="box"></div>

        <div class="box-2"></div>
</div>

<div class="box”></div>
```

This declaration block will apply to all elements that have a class of box that is inside an element with an ID of the container. It's worth noting that the `.box` element doesn't have to be an immediate child: there could be another element wrapping `.box`, and the styles would still apply.

Child Combinator: A selector that uses the child combinator is similar to a selector that uses a descendant combinator, except it only targets immediate child elements.

```
#container> .box {
        float: left;
        padding-bottom: 15px;
}

<div id="container">
        <div class="box"></div>

        <div>
                <div class="box"></div>
        </div>
</div>
```

The selector will match all elements that have a class of `box` and that are immediate children of the `#container` element. That means, unlike the descendant combinator, there can't be another element wrapping `.box` it has to be a direct child element.

General Sibling Combinator: A selector that uses a general sibling combinator to match elements based on sibling relationships. The selected elements are beside each other in the HTML.

```
h2 ~ p {
        margin-bottom: 20px;
}

<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<div class="box”>
        <p>Paragraph example.</p>
</div>
```

In this example, all paragraph elements (<p>) will be styled with the specified rules, but only if they are siblings of `<h2>` elements. There could be other elements in between the `<h2>` and `<p>`, and the styles would still apply.

Adjacent Sibling Combinator: A selector that uses the adjacent sibling combinator uses the plus symbol (+), and is almost the same as the general sibling selector. The difference is that the targeted element must be an immediate sibling, not just a general sibling.

```
p + p {
        text-indent: 1.Sem;
        margin-bottom: 0;
}

<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>

<div class="box”>
        <p>Paragraph example.</p>
        <p>Paragraph example.</p>
</div>
```

The above example will apply the specified styles only to paragraph elements that [illegible]
This means the first paragraph element on a page would not receive these styles. [illegible]
paragraphs, the second paragraph of the two wouldn't have the styles applied.

Get Ready with Free Mock Coding Interview

11:01 6.76 KB/s 5G 97%
interviewbit.com/css-
InterviewBit Practice Contests Login Sign up Looking to hireWe can help

```
        <div>
                <div class="box"></div>
        </div>
</div>
```

The selector will match all elements that have a class of `box` and that are immediate children of the `#container` element. That means, unlike the descendant combinator, there can't be another element wrapping `.box` it has to be a direct child element.

General Sibling Combinator: A selector that uses a general sibling combinator to match elements based on sibling relationships. The selected elements are beside each other in the HTML.

```
h2 ~ p {
        margin-bottom: 20px;
}

<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<div class="box">
        <p>Paragraph example.</p>
</div>
```

In this example, all paragraph elements (<p>) will be styled with the specified rules, but only if they are siblings of `<h2>` elements. There could be other elements in between the `<h2>` and `<p>`, and the styles would still apply.

Adjacent Sibling Combinator: A selector that uses the adjacent sibling combinator uses the plus symbol (+), and is almost the same as the general sibling selector. The difference is that the targeted element must be an immediate sibling, not just a general sibling.

```
p + p {
        text-indent: 1.5em;
        margin-bottom: 0;
}

<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>

<div class="box">
        <p>Paragraph example.</p>
        <p>Paragraph example.</p>
</div>
```

The above example will apply the specified styles only to paragraph elements that immediately follow other paragraph elements. This means the first paragraph element on a page would not receive these styles. Also, if another element appeared between two paragraphs, the second paragraph of the two wouldn't have the styles applied.

Attribute Selector: The attribute selector targets elements based on the presence and/or value of HTML attributes, and is declared using square brackets.

```
input [type="text"] {
        background-color: #444;
        width: 200px;
}

<input type="text">
```

7. What is a CSS Preprocessor? What are Sass, Less, and Stylus? Why do people use them?

A CSS Preprocessor is a tool used to extend the basic functionality of default vanilla CSS through its own scripting language. It helps us to use complex logical syntax like – variables, functions, mixins, code nesting, and inheritance to name a few, supercharging your vanilla CSS.

SASS: Sass is the acronym for "Syntactically Awesome Style Sheets". SASS can be written in two different syntaxes using SASS or SCSS

SASS vs SCSS

- SASS is based on indentation and SCSS(Sassy CSS) is not.
- SASS uses .sass extension while SCSS uses .scss extension.
- SASS doesn't use curly brackets or semicolons. SCSS uses it, just like the CSS.

SASS Syntax

```
$font-color: #fff
$bg-color: #00f

#box
        color: $font-color
        background: $bg-color
```

Get Ready with Free Mock Coding Interview

```
        width: 200px;
}

<input type="text">
```

7. What is a CSS Preprocessor? What are Sass, Less, and Stylus? Why do people use them?

A CSS Preprocessor is a tool used to extend the basic functionality of default vanilla CSS through its own scripting language. It helps us to use complex logical syntax like – variables, functions, mixins, code nesting, and inheritance to name a few, supercharging your vanilla CSS.

SASS: Sass is the acronym for "Syntactically Awesome Style Sheets". SASS can be written in two different syntaxes using SASS or SCSS

SASS vs SCSS

- SASS is based on indentation and SCSS(Sassy CSS) is not.
- SASS uses .sass extension while SCSS uses .scss extension.
- SASS doesn't use curly brackets or semicolons. SCSS uses it, just like the CSS.

SASS Syntax

```
$font-color: #fff
$bg-color: #00f

#box
        color: $font-color
        background: $bg-color
```

SCSS Syntax

```
$font-color: #fff;
$bg-color: #00f;

#box{
        color: $font-color;
        background: $bg-color;
}
```

LESS: LESS is an acronym for "Leaner Stylesheets". LESS is easy to add to any javascript projects by using NPM or less.js file. It uses the extension .less.

LESS syntax is the same as the SCSS with some exceptions. LESS uses @ to define the variables.

```
@font-color: #fff;
@bg-color: #00f

#box{
        color: @font-color;
        background: @bg-color;
}
```

Stylus: Stylus offers a great deal of flexibility in writing syntax, supports native CSS as well as allows omission of brackets, colons, and semicolons. It doesn't use @ or $ for defining variables.

```
/* STYLUS SYNTAX WRITTEN LIKE NATIVE CSS */
font-color= #fff;
bg-color = #00f;

#box {
        color: font-color;
        background: bg-color;
}

/* OR */

/* STYLUS SYNTAX WITHOUT CURLY BRACES */
font-color= #fff;
bg-color = #00f;

#box
        color: font-color;
        background: bg-color;
```

8. What is VH/VW (viewport height/ viewport width) in CSS?

It's a CSS unit used to measure the height and width in percentage with respect to the viewport. It is used mainly in responsive design techniques. The measure VH is equal to 1/100 of the height of the viewport. If the height of the browser is 1000px, 1vh is equal to 10px. Similarly, if the width is 1000px, then 1 vw is equal to 10px.

9. Difference between reset vs normalize CSS?. How do they differ?

Reset CSS: CSS resets aim to remove all built-in browser styling. For example margins, paddings, font-sizes of all elements are reset to be the same.

Normalize CSS: Normalize CSS aims to make built-in browser styling consistent a
browser dependencies.

Get Ready with Free Mock Coding Interview

design techniques. The measure VH is equal to 1/100 of the height of the viewport. If the height of the browser is 1000px, 1vh is equal to 10px. Similarly, if the width is 1000px, then 1 vw is equal to 10px.

9. Difference between reset vs normalize CSS?. How do they differ?

Reset CSS: CSS resets aim to remove all built-in browser styling. For example margins, paddings, font-sizes of all elements are reset to be the same.

Normalize CSS: Normalize CSS aims to make built-in browser styling consistent across browsers. It also corrects bugs for common browser dependencies.

10. What is the difference between inline, inline-block, and block?

Block Element: The block elements always start on a new line. They will also take space for an entire row or width. List of block elements are , <p>.

Inline Elements: Inline elements don't start on a new line, they appear on the same line as the content and tags beside them. Some examples of inline elements are <a>, , , and <img> tags.

Inline Block Elements: Inline-block elements are similar to inline elements, except they can have padding and margins and set height and width values.

11. Is it important to test the webpage in different browsers?

It's most important to test a website in different browsers when you're first designing it, or when making major changes. However, it's also important to repeat these tests periodically, since browsers go through a lot of updates and changes.

12. What are Pseudo elements and Pseudo classes?

Pseudo-elements allows us to create items that do not normally exist in the document tree, for example ::after.

- ::before
- ::after
- ::first-letter
- ::first-line
- ::selection

In the below example, the color will appear only on the first line of the paragraph.

```
p: :first-line {
        color: #ff0000;
        font-variant: small-caps;
}
```

Pseudo-classes select regular elements but under certain conditions like when the user is hovering over the link.

- :link
- :visited
- :hover
- :active
- :focus

Example of the pseudo-class, In the below example, the color applies to the anchor tag when it's hovered.

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

13. Does margin-top or margin-bottom have an effect on inline elements?

No, it doesn't affect the inline elements. Inline elements flow with the contents of the page.

14. What is cascading in CSS?

"Cascading" refers to the process of going through the style declarations and defining weight or importance to the styling rules that help the browser to select what rules have to be applied in times of conflict. The conflict here refers to multiple rules that are applicable to a particular HTML element. In such cases, we need to let the browser know what style needs to be applied to the element. This is done by cascading down the list of style declarations elements.

For example, if we have the below style:

```
p{
    color:white;
}
```

and we also have the following declaration below it or in another stylesheet that has been linked to the page:

```
p{
    color: black;
}
```

We have a conflict in color property here for the paragraph elements. Here, the browser just cascades down to identify what is the most recent and most specific style and applies that. Since we have the `color:black;` as the most specific declaration, the color black is applied to the paragraph elements. Now if you want to ensure color white is applied to the paragraph, we can define weight to that style by adding `!important` as shown below:

Get Ready with Free Mock Coding Interview

13. Does margin-top or margin-bottom have an effect on inline elements?

No, it doesn't affect the inline elements. Inline elements flow with the contents of the page.

14. What is cascading in CSS?

"Cascading" refers to the process of going through the style declarations and defining weight or importance to the styling rules that help the browser to select what rules have to be applied in times of conflict. The conflict here refers to multiple rules that are applicable to a particular HTML element. In such cases, we need to let the browser know what style needs to be applied to the element. This is done by cascading down the list of style declarations elements.

For example, if we have the below style:

```
p{
    color:white;
}
```

and we also have the following declaration below it or in another stylesheet that has been linked to the page:

```
p{
    color: black;
}
```

We have a conflict in color property here for the paragraph elements. Here, the browser just cascades down to identify what is the most recent and most specific style and applies that. Since we have the `color:black;` as the most specific declaration, the color black is applied to the paragraph elements. Now if you want to ensure color white is applied to the paragraph, we can define weight to that style by adding `!important` as shown below:

```
p{
    color:white !important;
}
```

`!important` ensures that the property has the maximum weight in presence of other conflicting properties.

15. What property is used for changing the font face?

We can use the font-family property for achieving this. The `font-family` property is used for specifying what font needs to be applied on the targetted DOM element. It can hold several font names as part of "fallback" mechanism in case the browser does not support the fonts. For example, we can use:

```
 p {
  font-family: "Times New Roman", Times, serif;
}
```

In the above piece of code, we are applying font-family property to the paragraph element.

- It tells the browser to look for "Times New Roman" font and apply it.
- If the "Times New Roman" font is not installed or supported, then it asks the browser to use Times font.
- If both "Times New Roman" and Times are not supported, then it asks the browser to use any supported generic font belonging to serif.

If you do not want the font-face of the paragraph element to be Times New Roman/Times/serif font, and you want to use the Arial/Helvetica/sans-serif font, then we can just update the CSS property of paragraph element as:

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

16. What are the differences between adaptive design and responsive design?

Adaptive Design	Responsive Design
Adaptive design focuses on developing websites based on multiple fixed layout sizes.	Responsive design focuses on showing content on the basis of available browser space.
When a website developed using adaptive design is opened on the desktop browser, first the available space is detected and then the layout with most appropriate sizes are picked and used for the display of contents. Resizing of browser window has no affect on the design.	When a website developed using responsive design is opened on a desktop browser and when we try to resize the browser window, the content of the website is dynamically and optimally rearranged to accomodate the window.
Usually, adaptive designs use six standard screen widths - 320 px, 480 px, 760 px, 960 px, 1200 px, 1600 px. These sizes are detected and appropriate layouts are loaded.	This design makes use of CSS media queries for changing styles depending on the target devices properties for adapting to different screens.
It takes a lot of time and effort to first examine the options and realities of the end users and then design best possible adaptive solutions them.	Generally, Responsive design takes much less work to build and design fluid websites that can accomodate content from screen depending on the screen size.
Gives a lot of control over the design to develop sites for specific screens.	No much control over the design is offered here.

17. How are the CSS selectors matched against the elements by the browser?

The order of matching selectors goes from **right to left** of the selector expression. The elements in the DOM are filtered by browsers based on the key selectors and are then traversed up to the parent elements for determining the matches. The speed of determining the elements depends on the length of the chain of selectors. Consider an example:

```
 p span{
```

Get Ready with Free Mock Coding Interview

specific screens.

17. How are the CSS selectors matched against the elements by the browser?

The order of matching selectors goes from **right to left** of the selector expression. The elements in the DOM are filtered by browsers based on the key selectors and are then traversed up to the parent elements for determining the matches. The speed of determining the elements depends on the length of the chain of selectors. Consider an example:

```
 p span{
    color: black;
}
```

Here, the browser first finds all `span` elements in the DOM and then it traverses to each of its parent elements to check if they are the paragraph `p` elements.

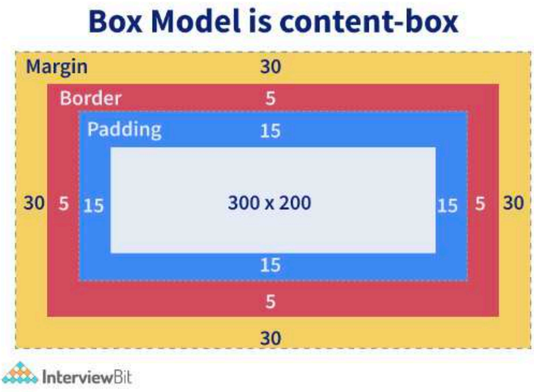
Once the browser finds **all** matching span tags having paragraph elements as parent and applies the color of black to the content, the matching process is stopped.

18. How is border-box different from content-box?

`content-box` is the default value box-sizing property. The height and the width properties consist only of the content by excluding the border and padding. Consider an example as shown:

```
div{
    width:300px;
    height:200px;
    padding:15px;
    border: 5px solid grey;
    margin:30px;
    -moz-box-sizing:content-box;
    -webkit-box-sizing:content-box;
    box-sizing:content-box;
}
```

Here, the box-sizing for the div element is given as content-box. That means, the height and width considered for the div content exclude the padding and border. We will get full height and width parameters specified for the content as shown in the below image.

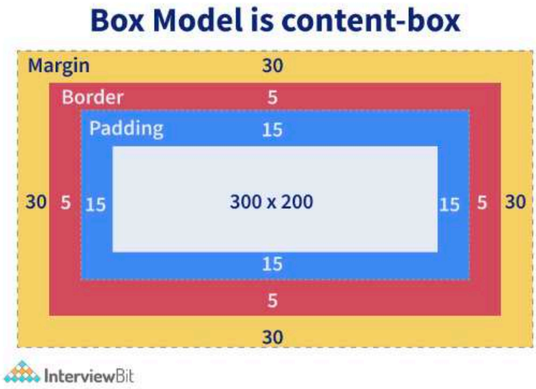


The `border-box` property includes the content, padding and border in the height and width properties. Consider an example as shown:

```
div{
    width:300px;
    height:200px;
    padding:15px;
    border: 5px solid grey;
    margin:30px;
    -moz-box-sizing:border-box;
    -webkit-box-sizing:border-box;
```

Get Ready with Free Mock Coding Interview

Here, the box-sizing for the div element is given as content-box. That means, the height and width considered for the div content exclude the padding and border. We will get full height and width parameters specified for the content as shown in the below image.



The `border-box` property includes the content, padding and border in the height and width properties. Consider an example as shown:

```
div{
    width:300px;
    height:200px;
    padding:15px;
    border: 5px solid grey;
    margin:30px;
    -moz-box-sizing:border-box;
    -webkit-box-sizing:border-box;
    box-sizing:border-box;
}
```

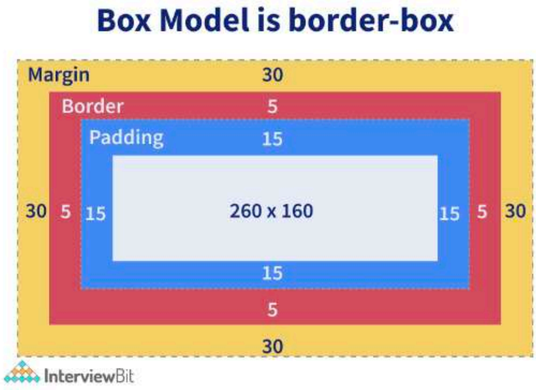
Here, the box-sizing for the div element is given as border-box. That means the height and width considered for the div content will also include the padding and border. This means that the actual height of the div content will be:

```
actual height = height -
                padding on top and bottom -
                border on top and bottom
              = 200 - (15*2) - (5*2)
              = 160 px
```

and the actual width of the div content would be:

```
actual width  = width -
                padding on right and left -
                border on right and left
              = 300 - (15*2) - (5*2)
              = 260 px
```

This is represented in the image below:



19. How is opacity specified in CSS3?

Opacity refers to the degree to which the content is transparent or opaque. We can use the property named `opacity` which takes the values ranging from 0 to 1. 0 specifies that the element is completely transparent where 1 means that the element is completely opaque. We can use the opacity property as follows:

```
 div {
    opacity: 0.6;
}
```

In the above example, an opacity of 60% is applied to the div section. The opacity

Get Ready with Free Mock Coding Interview

19. How is opacity specified in CSS3?

Opacity refers to the degree to which the content is transparent or opaque. We can use the property named `opacity` which takes the values ranging from 0 to 1. 0 specifies that the element is completely transparent where 1 means that the element is completely opaque. We can use the opacity property as follows:

```
div {
    opacity: 0.6;
}
```

In the above example, an opacity of 60% is applied to the div section. The opacity property is not supported by the internet explorer browser. To make it work there, we need to use filter property as polyfill as shown in the example below.

```
div {
    opacity: 0.6;
    filter: alpha(opacity=60);
}
```

20. Why should we use float property in CSS?

The float property is used for positioning the HTML elements horizontally either towards the left or right of the container. For instance,

```
float-demo {
    float: right;
}
```

Here, the element to which the class is applied ensures that the element is positioned on the right of the container. If you specify the value of float as left, then the element will be placed on the left side of the container.

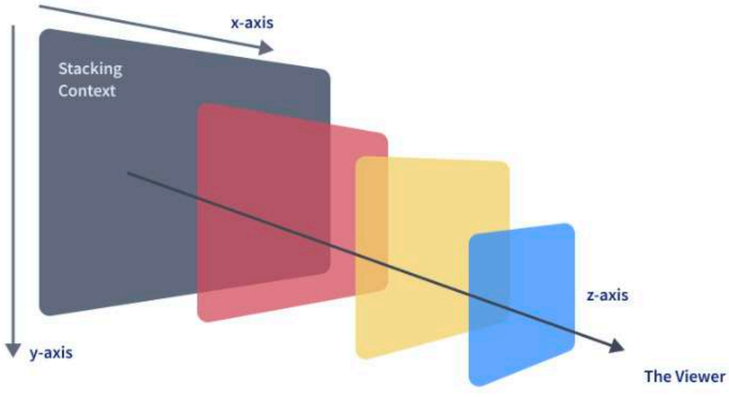
21. What is a z-index, how does it function?

z-index is used for specifying the vertical stacking of the overlapping elements that occur at the time of its positioning. It specifies the vertical stack order of the elements positioned that helps to define how the display of elements should happen in cases of overlapping.

The default value of this property is 0 and can be either positive or negative. Apart from 0, the values of the z-index can be:

- Auto: The stack order will be set equal to the parent.
- Number: The number can be positive or negative. It defines the stack order.
- Initial: The default value of 0 is set to the property.
- Inherit: The properties are inherited from the parent.

The elements having a lesser value of z-index is stacked lower than the ones with a higher z-index.



The z-index property determines the order of the element on the z-axis of the stacking cont

Get Ready with Free Mock Coding Interview

The z-index property determines the order of the element on the z-axis of the stacking contect.

InterviewBit

From the above figure, we can see that as the value of the z-index increases along the z-axis, the order of stacking would be towards the top of other elements along the vertical axis.

22. What do the following CSS selectors mean?

1. **div, p**
2. **div p**
3. **div ~ p**
4. **div + p**
5. **div > p**

The meaning of the given list of selectors goes as follows:

- `div, p`: This selector implies selecting all div elements and all p elements.

Consider an example below:

```
<h1>Heading 1</h1>
  <div>
      Division 1
      <p> paragraph 1</p> <!-- Will be selected -->
  </div>
  <p> paragraph 2</p>
  <p> paragraph 3</p>
  <div>
      Division 2
  </div>
  <span> Span 1 </span>
```

Here, **all** the div elements and the p elements would be selected by the browser irrespective of their parents or where they are placed. The remaining tags like h1 and span are ignored.

- `div p` : This selector tells to select all p elements that are inside div elements. Consider an example below:

```
  <h1>Heading 1</h1>
<div>
    Division 1
    <p> paragraph 1</p> <!-- Will be selected -->
    <div>
        <p> Inner Div Paragraph </p> <!-- Will be selected -->
    </div>
</div>
<p> paragraph 2</p>
<p> paragraph 3</p>
<div>
    Division 2
</div>
<span> Span 1 </span>
```

Here, `<p> paragraph 1</p>` and `<p> Inner Div Paragraph </p>` would be selected by the browser and the properties are applied. The rest of the paragraph tags are not selected.

- `div ~ p` : This selector tells to select all p elements that have div elements preceeded anywhere. Consider an example,

```
<h1>Heading 1</h1>
<div>
   Division 1
   <p> paragraph 1</p>
</div>
<p> paragraph 2</p> <!-- Will be selected -->
<p> paragraph 3</p> <!-- Will be selected -->
<div>
   Division 2
</div>
<span> Span 1 </span>
```

Here, paragraph 2 and paragraph 3 elements would be selected as marked in the code above.

- `div + p` : This selector says to select all p elements placed immediately after the div element. Consider an example in this case:

```
 <h1>Heading 1</h1>
<div>
    Division 1
    <p> paragraph 1</p>
</div>
<p> paragraph 2</p> <!-- Will be selected -->
<p> paragraph 3</p>
<div>
    Division 2
</div>
```

Get Ready with Free Mock Coding Interview

```
   <p> paragraph 1</p>
</div>
<p> paragraph 2</p> <!-- Will be selected -->
<p> paragraph 3</p> <!-- Will be selected -->
<div>
   Division 2
</div>
<span> Span 1 </span>
```

Here, paragraph 2 and paragraph 3 elements would be selected as marked in the code above.

- `div + p` : This selector says to select all p elements placed immediately after the div element. Consider an example in this case:

```
 <h1>Heading 1</h1>
<div>
    Division 1
    <p> paragraph 1</p>
</div>
<p> paragraph 2</p> <!-- Will be selected -->
<p> paragraph 3</p>
<div>
    Division 2
</div>
<span> Span 1 </span>
```

In this case, we have paragraph 2 element immediately after the div tag. Hence, only that element will be selected.

- `div > p` : This selector says to select all p elements which has div as an immediate parent. In the same example below:

```
 <h1>Heading 1</h1>
<div>
    Division 1
    <p> paragraph 1</p> <!-- Will be selected -->
</div>
<p> paragraph 2</p>
<p> paragraph 3</p>
<div>
    Division 2
</div>
<span> Span 1 </span>
```

Only `<p> paragraph 1</p>` will be selected in this case because it has immediate div as the parent.

23. What are the properties of flexbox?

Flexbox stands for flexible box and it was introduced around 2017 in CSS with the purpose of providing an efficient way to handle layouts, align elements within them and distribute spaces amongst the items in dynamic/responsive conditions. It provides an enhanced ability to alter the dimensions of the items and make use of the available space in the container efficiently. In order to achieve this, CSS3 provides some properties.

The properties of flexbox are as follows:

- **flex-direction**: This property helps in defining the direction the container should stack the items targetted for flex. The values of this property can be
  - row: Stacks items horizontally from left to right in the flex container.
  - column: Stacks items vertically from top to bottom in the flex container.
  - row-reverse: Stacks items horizontally from right to left in the flex container.
  - column-reverse: Stacks items vertically from bottom to top in the flex container.
- **flex-wrap**: This property specifies of the flex items should be wrapped or not. Possible values are:
  - wrap: The flex items would be wrapped if needed.
  - nowrap: This is the default value that says the items won't be wrapped.
  - wrap-reverse: This specifies that the items will be wrapped if needed but in reverse order.
- **flex-flow**: This property is used for setting both flex-direction and flex-wrap properties in one statement.
- **justify-content**: Used for aligning the flex items. Possible values are:
  - center: It means that all the flex items are present at the center of the container.
  - flex-start: This value states that the items are aligned at the start of the container. This is the default value.
  - flex-end: This value ensures the items are aligned at the end of the container.
  - space-around: This value displays the items having space between, before, around the items.
  - space-between: This value displays items with spaces between the lines.
- **align-items**: This is used for aligning flex items.
- **align-content**: This is used for aligning the flex lines.

CSS Interview Questions for Experienced

24. Difference between CSS grid vs flexbox?

- CSS Grid Layout is a two-dimensional system, meaning it can handle both columns and rows. Grid layout is intended for larger-scale layouts which aren't linear in design.
- Flexbox is largely a one-dimensional system (either in a column or a row). Flexbox layout is most appropriate to the components of an application.

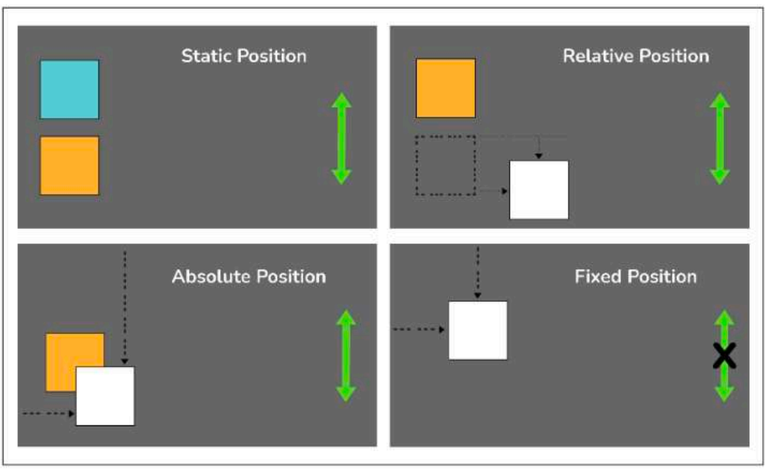
25. Explain CSS position property?

Get Ready with Free Mock Coding Interview

- Flexbox is largely a one-dimensional system (either in a column or a row). Flexbox layout is most appropriate to the components of an application.

25. Explain CSS position property?

- **Absolute:** To place an element exactly where you want to place it. absolute position is actually set relative to the element's parent. if no parent is available then the relative place to the page itself (it will default all the way back up to the element).
- **Relative:** "Relative to itself". Setting position: relative; on an element and no other positioning attributes, it will no effect on its positioning. It allows the use of z-index on the element and it limits the scope of absolutely positioned child elements. Any child element will be absolutely positioned within that block.
- **Fixed:** The element is positioned relative to the viewport or the browser window itself. viewport doesn't change if you scroll and hence the fixed element will stay right in the same position.
- **Static:** Static default for every single page element. The only reason you would ever set an element to position: static is to forcefully remove some positioning that got applied to an element outside of your control.
- **Sticky:** Sticky positioning is a hybrid of relative and fixed positioning. The element is treated as relative positioned until it crosses a specified threshold, at which point it is treated as fixed positioned.



InterviewBit

26. When does DOM reflow occur?

Reflow is the name of the web browser process for re-calculating the positions and geometries of elements in the document, for the purpose of re-rendering part or all of the document.

Reflow occurs when:

- Insert, remove or update an element in the DOM.
- Modify content on the page, e.g. the text in an input box.
- Move a DOM element.
- Animate a DOM element.
- Take measurements of an element such as offsetHeight or getComputedStyle.
- Change a CSS style.

27. Different Box Sizing Property?

The box-sizing CSS property sets how the total width and height of an element are calculated.

- **Content-box:** The default width and height values apply to the element's content only. The padding and border are added to the outside of the box.
- **Padding-box:** Width and height values apply to the element's content and its padding. The border is added to the outside of the box. Currently, only Firefox supports the padding-box value.
- **Border-box:** Width and height values apply to the content, padding, and border.

28. How to center align a div inside another div?

- **Centering with Table:**

HTML:

```
<div class="cn"><div class="inner">your content</div></div>
```

CSS:

```
.cn {
        display: table-cell;
        width: 500px;
        height: 500px;
        vertical-align: middle;
        text-align: center;
}

.inner {
        display: inline-block;
        width: 200px; height: 200px;
}
```

Get Ready with Free Mock Coding Interview

- **Centering with Transform**

- **Padding-box:** Width and height values apply to the element's content and its padding. The border is added to the outside of the box. Currently, only Firefox supports the padding-box value.
- **Border-box:** Width and height values apply to the content, padding, and border.

28. How to center align a div inside another div?

- **Centering with Table:**

HTML:

```
<div class="cn"><div class="inner">your content</div></div>
```

CSS:

```
.cn {
        display: table-cell;
        width: 500px;
        height: 500px;
        vertical-align: middle;
        text-align: center;
}

.inner {
        display: inline-block;
        width: 200px; height: 200px;
}
```

- **Centering with Transform**

HTML:

```
<div class="cn"><div class="inner">your content</div></div>
```

CSS:

```
.cn {
        position: relative;
        width: 500px;
        height: 500px;
}

.inner {
        position: absolute;
        top: 50%; left: 50%;
        transform: translate(-50%,-50%);
        width: 200px;
        height: 200px;
}
```

- **Centering with Flexbox**

HTML:

```
<div class="cn"><div class="inner">your content</div></div>
```

CSS:

```
.cn {
        display: flex;
        justify-content: center;
        align-items: center;
}
```

- **Centering with Grid**

HTML:

```
<div class="wrap_grid">
        <div id="container">vertical aligned text<br />some more text here
        </div>
</div>
```

CSS:

```
.wrap-grid {
        display: grid;
        place-content: center;
}
```

29. Can you name the four types of @media properties?

The four types of @media properties are:

1. All → It's the default property. Used for all media-type devices.
2. Screen → Used for computer screen, mobile screen.
3. Print → Used for printers.
4. Speech → Used for screen readers.

30. What is the grid system?

Get Ready with Free Mock Coding Interview

3. Print → Used for printers.
4. Speech → Used for screen readers.

30. What is the grid system?

CSS Grid Layout is the most powerful layout system available in CSS. It is said to be a 2-dimensional system, meaning it can handle both columns and rows, unlike flexbox which is largely a 1-dimensional system.

31. What are the different ways to hide the element using CSS?

- Using display property(**display: none**). It's not available for screen readers. The element will not exist in the DOM if display: none is used.
- Using visibility property(**visibility: hidden**), will take up the space of the element. It will be available to screen reader users. The element will actually be present in the DOM, but not shown on the screen.
- Using position property (**position: absolute**). Make it available outside the screen.

32. What does the :root pseudo-class refer to?

The :root selector allows you to target the highest-level "parent" element in the DOM, or document tree. It is defined in the CSS Selectors Level 3 specification.

33. What does Accessibility (a11y) mean?

Accessibility refers to how software or hardware combinations are designed to make a system accessible to persons with disabilities, such as visual impairment, hearing loss, or limited dexterity.

For example, a website developed with accessibility in mind might have text-to-speech capabilities. In the USA public websites have to have accessible compliance. It's defined in 508 compliance. It gives the guidelines and best practices for all website users that should be met with key areas of accessibility.

34. How do I restore the default value of a property?

The keyword initial can be used to reset it to its default value.

35. How does Calc work?

The CSS3 calc() function allows us to perform mathematical operations on property values. Instead of declaring, for example, static pixel values for an element's width, we can use calc() to specify that the width is the result of the addition of two or more numeric values.

```
.foo {
        Width: calc(100px + 50px)
}
```

36. What do CSS Custom properties variables mean?

Custom properties (sometimes referred to as CSS variables or cascading variables) are defined by users that contain specific values to be reused throughout a document. The value is set using -- notion. And the values are accessed using the var() function.

```
:root {
        --main-bg-color: brown
}

.one {
        color: white;
        background-color: var (--main-bg-color);
        margin: 10px,
        width: 50px,
        height: 50px;
        display: inline-block;
}
```

37. What is the difference between CSS variables and preprocessor(SASS, LESS, Stylus) variables?

- CSS variables can be used without the need for a preprocessor. Currently, all the major browsers support the CSS variables.
- CSS variable cascade. But the preprocessor variables don't cascade.
- CSS variable can be accessed and manipulated in javascript.

38. What does * { box-sizing: border-box; } do? What are its advantages?

- It makes every element in the document include the padding and border in the element's inner dimension for the height and width computation.
- In box-sizing: border-box, The height of an element is now calculated by the content's height + vertical padding + vertical border width.
- The width of an element is now calculated by the content's width + horizontal padding + horizontal border width.

39. What does !important mean in CSS?

The style is having the important will have the highest precedence and it overrides the cascaded property.

```
p {
        color: red !important;
}
#thing {
        color: green;
}
```

Get Ready with Free Mock Coding Interview

- The width of an element is now calculated by the content's width + horizontal padding + horizontal border width.

39. What does !important mean in CSS?

The style is having the important will have the highest precedence and it overrides the cascaded property.

```
p {
        color: red !important;
}
#thing {
        color: green;
}
<p id="thing">Will be RED.</p>
```

40. What is specificity? How to calculate specificity?

A process of determining which CSS rule will be applied to an element. It actually determines which rules will take precedence. Inline style usually wins then ID then the class value (or pseudo-class or attribute selector), the universal selector (*) has no specificity. ID selectors have a higher specificity than attribute selectors.

41. What is progressive rendering? How do you implement progressive rendering in the website?. What are the advantages of it?

Progressive rendering is the name given to techniques used to improve the performance of a webpage (in particular, improve perceived load time) to render content for display as quickly as possible.

We can implement the progressive rendering of the page by loading the lazy loading of the images. We can use Intersection Observer API to lazy load the image. The API makes it simple to detect when an element enters the viewport and take an action when it does. Once the image enters the viewport, we will start loading the images.

A sample snippet is given below.

```
<img class="lazy"
src="placeholder-image.jpg"
data-src="image-to-lazy-load-1x.jpg"
data-srcset="image-to-lazy-load-2x.jpg 2x, image-to-lazy-load-1x.jpg 1x"
alt="I'm an image!">

document.addEventListener("DOMContentLoaded", function() {
  var lazyImages = [].slice.call(document.querySelectorAll("img.lazy"));

  if ("IntersectionObserver" in window) {
    let lazyImageObserver = new IntersectionObserver(function(entries, observer) {
      entries.forEach(function(entry) {
        if (entry.isIntersecting) {
          let lazyImage = entry.target;
          lazyImage.src = lazyImage.dataset.src;
          lazyImage.srcset = lazyImage.dataset.srcset;
          lazyImage.classList.remove("lazy");
          lazyImageObserver.unobserve(lazyImage);
        }
      });
    });

    lazyImages.forEach(function(lazyImage) {
      lazyImageObserver.observe(lazyImage);
    });
  } else {
    // Possibly fall back to event handlers here
  }
});
```

42. What are the advantages of using translate() instead of absolute position?

Translate() does not cause the browser to trigger repaint and layout and instead only acts on the compositor. The absolute position triggers the repaint or DOM reflow. So, translate() gives better performance.

43. Does style1.css have to be downloaded and parsed before style2.css can be fetched?

```
<head>
        <link h ref=" stylel. css" rel=" stylesheet">
        <link href="style2.css" rel="stylesheet">
</head>
```

No, the browsers will download the CSS in the order of its appearance on the HTML page.

44. How to determine if the browser supports a certain feature?

The @support in CSS can be very useful to scan if the current browser has support for a certain feature.

```
@supports (display: grid) {
        div {
                display: grid;
        }
}
```

Get Ready with Free Mock Coding Interview

45. How will you fix browser-specific styling issues?

```
</head>
```

No, the browsers will download the CSS in the order of its appearance on the HTML page.

44. How to determine if the browser supports a certain feature?

The @support in CSS can be very useful to scan if the current browser has support for a certain feature.

```
@supports (display: grid) {
        div {
                display: grid;
        }
}
```

45. How will you fix browser-specific styling issues?

Different ways to fix browser-specific issues.

- We can write browser-specific styles separately in different sheets and load that only when the specific browser is used. This makes use of the server-side rendering technique.
- We can use auto-prefix for automatically adding vendor prefixes in the code.
- We can also use normalize.css or reset CSS techniques.

There are some ways for avoiding browser compatibility issues too. They are as follows:

- **Validate HTML and CSS:** We know that the code will be read, interpreted and handled differently by different browsers. We need to validate our HTML and CSS files for the missing closing tags, or missing semicolons in the syntaxes because there are chances that the old browsers will throw errors while rendering the code. We can avoid those errors by:
  - Maintaining well-aligned code that helps in easy readability.
  - Inserting comments at necessary places.
  - Make use of validation tools like Jigsaw CSS validator, W3C HTML Validators to identify syntax issues in the code.
- **Maintain Cross-Browser Compatibility in the Layouts:** Cross-Browser compatibility is a must while developing web applications. We expect our application to be responsive across all devices, browsers and platforms. Some of the effects of layout incompatibilities are unresponsiveness of the layouts in mobile devices, the difference in layout rendering between modern and old browsers, etc. These incompatibilities can be avoided by using:
  - CSS Multi-Column layouts - For maintaining proper layouts w.r.t columns and containers.
  - HTML viewport metatag – For ensuring content is properly spanned across mobile devices.
  - CSS Flexbox and Grids - To layout child elements depending on the content and available space.
  - CSS resets stylesheets - For reducing browser inconsistencies in default line heights, font sizes, margins etc.
- **Check JavaScript Library issues:** Ensure the libraries are used judiciously and the ones used are supported by the browsers.
- **Check DOCTYPE tag keyword:** The DOCTYPE keyword is meant for defining rules of what needs to be used in the code. Older browser versions check for DOCTYPE tag at the beginning and if not found, the application rendering won't be proper.
- **Test on real devices:** Although applications can be tested on virtual environments, it would be more beneficial if the testing is carried out on real devices and platforms. We can use tools like Testsigma for this purpose that enables us to test in real devices parallelly.

Conclusion

CSS plays the most important role in the field of web development. This is because CSS helps in achieving beautiful, responsive or adaptive websites depending on the business requirements. CSS helps in building lighter and flexible layouts that help in loading pages faster and making the content visually appealing. CSS is continuously evolving and is becoming more powerful thereby making it the most sought-after technology by various companies to develop websites. In this article, we have seen the most commonly asked interview questions in CSS, more particularly CSS3.

Useful Resources

- CSS Guides
- HTML
- Coding Practice
- JavaScript
- SASS Vs SCSS
- HTML/CSS Books
- CSS Vs HTML

46. How does this property work overflow: hidden?

The overflow property in CSS is used for specifying whether the content has to be clipped or the scrollbars have to be added to the content area when the content size exceeds the specified container size where the content is enclosed. If the value of overflow is hidden, the content gets clipped post the size of the container thereby making the content invisible. For example,

```
div {
    width: 150px;
    height: 50px;
    overflow: hidden;
}
```

If the content of the div is very large and exceeds the height of 50px, the content gets clipped post 50px and the rest of the content is not made visible.

47. How will you align content inside the p tag at the exact center inside the div?

We can add the `text-align: center` property inside the parent div for aligning the contents horizontally. But it will not align the contents vertically. We can align the content vertically by making the parent element have relative positioning and the child element have absolute positioning. The child element should have the values of top, botto
Then we need to set the margin as auto. It is assumed that both the child and mo

Get Ready with Free Mock Coding Interview

```
}
```

If the content of the div is very large and exceeds the height of 50px, the content gets clipped post 50px and the rest of the content is not made visible.

47. How will you align content inside the p tag at the exact center inside the div?

We can add the `text-align: center` property inside the parent div for aligning the contents horizontally. But it will not align the contents vertically. We can align the content vertically by making the parent element have relative positioning and the child element have absolute positioning. The child element should have the values of top, bottom, right, left as 0 to center it in the middle vertically. Then we need to set the margin as auto. It is assumed that both the child and mother elements will have height and width values.

Consider we have a div element of height and width taking 20% of the screen size, and we have a paragraph element taking the height of 1.2em and width of 20%. If we want to align the paragraph element at the center (vertically and horizontally), we write the following styles:

```
div {
    position : relative;   // Make position relative
    height : 20%;
    width : 20%;
    text-align : center; //Align to center horizontally
}
p {
    position : absolute; // Make position absolute
    top:0;                 // Give values of top, bottom,left, right to 0
    bottom:0;
    left:0;
    right:0;
    margin : auto;         // Set margin as auto
    height : 1.2 em;
    width : 20%;
}
```

48. How is margin different from padding in CSS?

Margin property using which we can create space around the elements. We can also create space for borders defined at the exteriors. We have the following properties for defining the margin:

- margin-top
- margin-right
- margin-bottom
- margin-left
  margin property by itself has the values as:
- auto – The browser auto-calculates the margin while we use this.
- length – The value of this property can be in px, pt, cm, em etc. The values can be positive or negative.
- % – We can also give percentage value as margin to the element.
- inherit – Using this property, the margin properties can be inherited from the parent elements.

The padding property is used for generating the space around the element's content and inside any known border. The padding also has sub-properties like:

- padding-top
- padding-right
- padding-bottom
- padding-left

It is to be noted that the padding does not allow negative values.

From the below image, we can see that the Margin is the outermost entity of the CSS Box Model that lies outside of the borders whereas the padding lies within the borders.

49. What do you have to do to automatically number the heading values of sections and categories?

We can use the concept of CSS counters. This lets us adjust the appearance of th
While using this, we need to first initialize the value of the counter-reset property

Get Ready with Free Mock Coding Interview

InterviewBit

49. What do you have to do to automatically number the heading values of sections and categories?

We can use the concept of CSS counters. This lets us adjust the appearance of the content based on the location in a document. While using this, we need to first initialize the value of the counter-reset property which is 0 by default. The same property is also used for changing the value to any number that we need. Post initialization, the counter's value can be incremented or decremented by using the counter-increment property. The name of the counter cannot be CSS keywords like "none", "initial", "inherit" etc. If the CSS keywords are used, then the declaration would be ignored.

Consider an example as shown below:

```
body {
  counter-reset: header;     /* define counter named 'header' whose initial value is 0 by default */
}

h2::before {
  counter-increment: header;     /* The value of header counter by 1.*/
  content: "Header " counter(header) ": ";    /* To display word Header and the value of the counter with colon before it.*/
}
```

Here, we are trying to achieve auto count increment and display feature for the h2 tag. Wherever we use h2 tag, the content will be prefixed by "Header 1 : " , "Header 2 : ", "Header 3 : " etc.

50. How is the nth-child() different from nth of type selectors?

Both are pseudo-classes (Pseudo-classes are those keywords that specifies the special state of the selected element). The `nth-child()` pseudo-class is used for matching elements based on the number that represents the position of an element based on the siblings. The number is used to match an element on the basis of the element's position amongst its siblings.

For example, in the below piece of code, if we give nth-child(4) for the example class, then the 4th child of the example class is selected irrespective of the element type. Here, the fourth child of the example class is the div element. The element is selected and a background of black is added to it.

```
.example:nth-child(4) {
    background: black;
}
<div class="example">
        <p>This is a paragraph.</p>
        <p>This is a paragraph.</p>
        <p>This is a paragraph.</p>
        <div>This is a div.</div> <!-- 4th Element to select and apply style-->
        <div>This is a div.</div>
        <p>This is a paragraph.</p>
        <p>This is a paragraph.</p>
        <div>This is a div.</div>
</div>
```

The `nth-of-type()` pseudo-class is similar to the nth-child but it helps in matching the selector based on a number that represents the position of the element within the elements that are the siblings of its same type. The number can also be given as a function or give keywords like odd or even.

For example, in the below piece of code, if we give `p:nth-of-type(even)` for the example class, then all the even paragraph tags are selected within the example class and the style of background black is applied to them. The selected elements are marked in comments in the below code:

```
.example p:nth-of-type(even) {
    background: black;
}
<div class="example">
        <p>This is a paragraph.</p>
        <p>This is a paragraph.</p> <!-- Select this and apply style-->
        <p>This is a paragraph.</p>
        <div>This is a div.</div>
        <div>This is a div.</div>
        <p>This is a paragraph.</p> <!-- Select this and apply style-->
        <p>This is a paragraph.</p>
        <div>This is a div.</div>
        <p>This is a paragraph.</p> <!-- Select this and apply style-->
        <div>This is a div.</div>
</div>
```

51. What is the importance of CSS Sprites?

CSS sprites are used for combining multiple images in a single larger image. They are commonly used for representing icons that are used in the user interfaces. The main advantages of using sprites are:

- It reduces the number of HTTP requests to get data of multiple images as they are acquired only by sending a single request.
- It helps in downloading assets in advance that help display icons or images upon hover or other pseudo-states.
- When there are multiple images, the browser makes separate calls to get the image for each of them. Using sprites, the images are combined in one and we can just call for that image using one call.

Get Ready with Free Mock Coding Interview

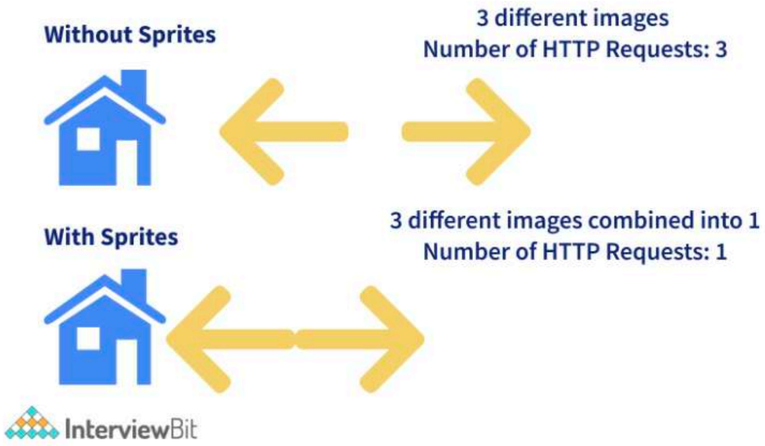
```
        <div>This is a div.</div>
</div>
```

51. What is the importance of CSS Sprites?

CSS sprites are used for combining multiple images in a single larger image. They are commonly used for representing icons that are used in the user interfaces. The main advantages of using sprites are:

- It reduces the number of HTTP requests to get data of multiple images as they are acquired only by sending a single request.
- It helps in downloading assets in advance that help display icons or images upon hover or other pseudo-states.
- When there are multiple images, the browser makes separate calls to get the image for each of them. Using sprites, the images are combined in one and we can just call for that image using one call.

Consider an example where our application requires 3 images as shown below (Without Sprites Section). If we are trying to load the images independently, we require 3 different HTTP Requests to get the data. But if we have CSS Sprites where all 3 images are combines into 1 separated by some spaces, then we require only 1 HTTP Request.



We can access each image from the sprite by accessing the positioning properties as shown in the below code:

```
<!DOCTYPE html>
<html>
<head>
<style>
#home-icon {
  left: 0px;
  width: 46px;
  background: url('spriteFile.gif') 0 0;
}

#prev-icon {
  left: 63px;
  width: 43px;
  background: url('spriteFile.gif') -47px 0;
}

#next-icon {
  left: 129px;
  width: 43px;
  background: url('spriteFile.gif') -91px 0;
}
</style>
</head>
<body>

<img id="home-icon" src="spriteFile.gif" width="1" height="1">      <!-- To display home icon here -->
<img id="next-icon" src="spriteFile.gif" width="1" height="1">      <!-- To display next icon icon here -->
<img id="prev-icon" src="spriteFile.gif" width="1" height="1">      <!-- To display previous icon icon here -->

</body>
</html>
```

In the above code, we are trying to access each element - house, previous and next icon - from the sprite file by using the left, width properties. The image is displayed in the img section by means of the background property. Do note that the source of the image (src attribute of the img tag) is just one file which is the `spriteFile.gif` and depending on the rules specified in the id selectors, the images are loaded accordingly.

52. What do you understand by tweening in CSS?

Tweening is the process of filling the gaps between the key sequences, i.e between the keyframes that are already created. Keyframes are those frames that represent start and end point of animation action. Tweening involves generating intermediate keyframes between two images that give the impression that the first one has evolved smoothly to the second image. For this purpose, we use properties like transforms - matrix, translate, scale, rotate etc.

In the below example, we are generating intermediate frames of paragraph elements to slide through from the start to the right edge of the browser.

Get Ready with Free Mock Coding Interview

```
p {
```

In the above code, we are trying to access each element - house, previous and next icon - from the sprite file by using the left, width properties. The image is displayed in the img section by means of the background property. Do note that the source of the image (src attribute of the img tag) is just one file which is the `spriteFile.gif` and depending on the rules specified in the id selectors, the images are loaded accordingly.

52. What do you understand by tweening in CSS?

Tweening is the process of filling the gaps between the key sequences, i.e between the keyframes that are already created. Keyframes are those frames that represent start and end point of animation action. Tweening involves generating intermediate keyframes between two images that give the impression that the first one has evolved smoothly to the second image. For this purpose, we use properties like transforms - matrix, translate, scale, rotate etc.

In the below example, we are generating intermediate frames of paragraph elements to slide through from the start to the right edge of the browser.

```
p {
  animation-duration: 2s;
  animation-name: slidethrough;
}

@keyframes slidethrough {
  from {
    margin-left: 100%;
    width: 300%;
  }

  to {
    margin-left: 0%;
    width: 100%;
  }
}
```

Here, the paragraph element specifies that the animation process should take 2 seconds for execution from start to the finish. This is done by using the `animation-duration` property. The animation-name of the `@keyframes` is defined by using the property `animation-name`. The intermediate keyframes are defined by using `@keyframes` rule. In the example, we have just 2 keyframes. The first keyframe starts at 0% and runs till the left margin of 100% which is the rightmost edge of the containing element. The second keyframe starts at 100% where the left margin is set as 0% and the width to be set as 100% which results in finishing the animation flush against the left edge of the container area.

53. Why do we need to use clear property along with floats in CSS?

The clear property along with floats is used for specifying which side of floating elements is not supposed to float. An element having clear property ensures that the element does not move up adjacent to the float. But the element will be moved down past the float.

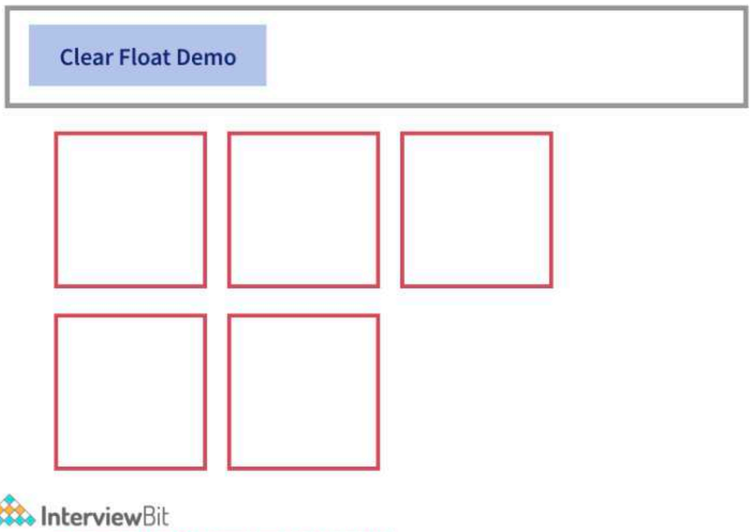
Let us understand this with the help of an example. We know that the floated objects do not add to the height of the objects where they reside. Consider we have a div element with class “floated_div” within another div element with id “main_div”.

```
<html>
    <head>
    <style>
   #main_div {
      width: 400px;
      margin: 10px auto;
      border: 4px solid #cccccc;
      padding: 5px;
   }

   .floated_div {
      float: left;
      width: 50px;
      height: 50px;
      border: 2px solid #990000;
      margin: 10px;
   }
  </style>
    </head>
    <body>
        <div id="main_div">
            <p>Clear Float Demo</p>
            <div class="floated_div"></div>
            <div class="floated_div"></div>
            <div class="floated_div"></div>
            <div class="floated_div"></div>
            <div class="floated_div"></div>
        </div>
    </body>
</html>
```

The result of this code would be as shown below. We see that the squares that are expected to be within dev are not within the main parent div. How do we fix this?

```
            <p>Clear Float Demo</p>
            <div class="floated_div"></div>
            <div class="floated_div"></div>
            <div class="floated_div"></div>
            <div class="floated_div"></div>
            <div class="floated_div"></div>
        </div>
    </body>
</html>
```

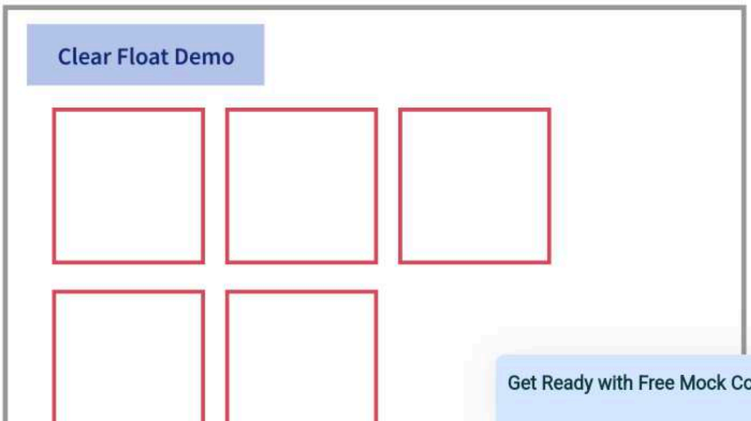
The result of this code would be as shown below. We see that the squares that are expected to be within dev are not within the main parent div. How do we fix this?



We can do it just by adding `<div style="clear:both"></div>` line at the end of the last floated element so that the floated elements are fit in properly within the main div container.

```
<html>
    <head>
    <style>
        #main_div {
            width: 400px;
            margin: 10px auto;
            border: 4px solid #cccccc;
            padding: 5px;
        }

        .floated_div {
            float: left;
            width: 50px;
            height: 50px;
            border: 2px solid #990000;
            margin: 10px;
        }
    </style>
    </head>
    <body>
        <div id="main_div">
            <p>Clear Float Demo</p>

            <div class="floated_div"></div>
            <div class="floated_div"></div>
            <div class="floated_div"></div>
            <div class="floated_div"></div>
            <div class="floated_div"></div>
            <div style="clear:both"></div>    <!-- Adding this fixed the issue -->
        </div>
    </body>
</html>
```

The result of this will be:

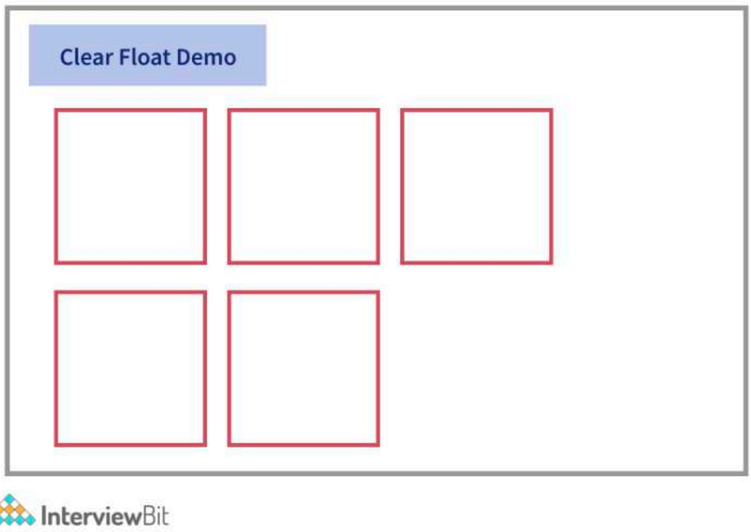


Get Ready with Free Mock Coding Interview

11:02 Vo) LTE 1.49 KB/s 5G 97%
interviewbit.com/css-i


```
    </head>
    <body>
        <div id="main_div">
             <p>Clear Float Demo</p>

             <div class="floated_div"></div>
             <div class="floated_div"></div>
             <div class="floated_div"></div>
             <div class="floated_div"></div>
             <div class="floated_div"></div>
             <div style="clear:both"></div>    <!-- Adding this fixed the issue -->
        </div>
    </body>
</html>
```

The result of this will be:



54. How does the absolute positioning work?

Absolute positioning is a very powerful positioning mechanism that allows users to place any element wherever they want in an exact location. The CSS properties right, left, top, bottom and define the exact locations where you need to place the element. In absolute positioning, the following points need to be considered:

- The element to which the absolute positioning is applied is removed from the normal workflow of the HTML document.
  - The HTML layout does not create any space for that element in its page layout.
- The element is positioned relative to the closest positioned ancestor. If no such ancestor is present, then the element is placed relative to the initial container block.
- The final position of the element is determined based on values provided to the top, right, left, bottom.

CSS MCQ

1.What would be the color of text “I am awesome” for the following rules?

```
<ul class="shopping-list" id="awesome">
<li><span>Milk</span></li>
<li class="favorite" id=”must-buy”>
<span class=”highlight”>I am awesome</span>
</li>
 </ul>
<style>
ul#awesome {
color: red;
}
Ul.shopping-list li.favorite span {
color: blue;
}
</style>
```

- Blue
- Red
- Green
- Yellow

2.Which of the following is NOT a valid CSS length unit?

Get Ready with Free Mock Coding Interview

11:02 Vo) LTE 0.39 KB/s 5G 97%
chatgpt.com/c/683dc

Here's a comprehensive list of CSS interview questions and coding challenges for 2025. This includes conceptual questions, scenario-based challenges, and practical coding problems that are frequently asked during frontend developer interviews.

🔹 CSS Theoretical Interview Questions

🧠 Basic Level

1. What is the difference between classes and IDs in CSS?
2. What are the different types of CSS? (Inline, Internal, External)
3. What is the Box Model in CSS?
4. What is specificity in CSS?
5. How does inheritance work in CSS?
6. What is the difference between `em`, `rem`, `%`, `px` units?
7. What's the difference between `visibility: hidden` and `display: none`?

⚙️ Intermediate Level

8. Explain the difference between relative, absolute, fixed, and sticky positioning.
9. What is a pseudo-class? Name a few.
10. What is the difference between `nth-child()` and `nth-of-type()`?
11. How do media queries work in responsive design?
12. What are flexbox and grid? When would you use one over the other?
13. What is the difference between `min-width`, `max-width`, and `width` in media queries?
14. How can you center a div horizontally and vertically in modern CSS?

🚀 Advanced Level

15. Explain stacking context and z-index in CSS.
16. What is the difference between `auto-fill` and `auto-fit` in CSS Grid?
17. How would you create a custom scrollbar using CSS?
18. What is `contain: layout` used for in CSS?
19. How can you improve the performance of CSS in a large project?
20. How does `will-change` help with rendering performance?
21. Explain the concept of CSS-in-JS.

🔸 CSS Coding Questions (With Example Challenges)

1. Center a div

html Copy Edit

```html
<!-- Create a div centered both horizontally and vertically using Flexbox.
```

2. Create a responsive card grid

html Copy Edit

```html
<!-- Design a grid layout with cards that adjust based on screen size using
```

Unlock more with Plus
ChatGPT Plus gives you higher limits, smarter models, and Sora
Get Plus

11:02 Vo) LTE 0 KB/s 5G 97%
chatgpt.com/c/683dc

CSS Coding Questions (With Example Challenges)

1. **Center a div**

```html
<!-- Create a div centered both horizontally and vertically using Flexbox.
```

2. **Create a responsive card grid**

```html
<!-- Design a grid layout with cards that adjust based on screen size usin
```

3. **Build a pure CSS tooltip**

```html
<!-- Show a tooltip above an element on hover using only CSS. -->
```

4. **Create a dropdown menu using only CSS**

```html
<!-- Style a dropdown that opens on hover without JavaScript. -->
```

5. **Responsive Navigation Bar**

```html
<!-- Build a nav bar that collapses into a hamburger icon on mobile (bonus
```

6. **Animated Loader**

```html
<!-- Design a spinning loader or bouncing dots using CSS animations. -->
```

7. **Glassmorphism card UI**

```html
<!-- Design a card with background blur and transparency effect. -->
```

8. **Responsive image gallery**

```html
<!-- Build an image grid that reflows based on screen size using media que
```

9. **Accordion UI using only CSS**

```html
<!-- Implement collapsible content sections using input and label tags wit
```

10. **Create a sticky header**

```html
<!-- Implement a header that sticks to the top while scrolling using `posi
```

Unlock more with Plus
ChatGPT Plus gives you higher limits, smarter models, and Sora for video.
Get Plus

Ask anything

11:02 7.42 KB/s 5G 97%
chatgpt.com/c/683dc

```html
<!-- Show a tooltip above an element on hover using only CSS. -->
```

4. **Create a dropdown menu using only CSS**

```html
<!-- Style a dropdown that opens on hover without JavaScript. -->
```

5. **Responsive Navigation Bar**

```html
<!-- Build a nav bar that collapses into a hamburger icon on mobile (bonus
```

6. **Animated Loader**

```html
<!-- Design a spinning loader or bouncing dots using CSS animations. -->
```

7. **Glassmorphism card UI**

```html
<!-- Design a card with background blur and transparency effect. -->
```

8. **Responsive image gallery**

```html
<!-- Build an image grid that reflows based on screen size using media que
```

9. **Accordion UI using only CSS**

```html
<!-- Implement collapsible content sections using input and label tags wit
```

10. **Create a sticky header**

```html
<!-- Implement a header that sticks to the top while scrolling using `posi
```

📘 Preparation Tips for CSS Interviews in 2025

- Focus on modern layout techniques (Flexbox, Grid).
- Be confident in responsive design and media queries.
- Practice pure CSS UIs and avoid JS when possible in challenges.
- Explore real-world applications: custom scrollbars, modals, animations.
- Get familiar with Tailwind CSS or Bootstrap for bonus framework knowledge.

Would you like sample code for any of the coding questions above or a mock interview setup?

Unlock more with Plus
ChatGPT Plus gives you higher limits, smarter models, and Sora for video.
Get Plus

Ask anything