**Css Interview Questions with Answers**

1.What is CSS? Explain different versions

2.What are the advantages of CSS?

3.What are the CSS frameworks?

4.What is a CSS selector?

5.What is pseudo-elements vs psudo pseudo-class?

6.What is the difference between class selectors and id selectors?

7.What are the advantages of External Style Sheets?

8.What is RWD?

9.What are the benefits of CSS sprites?

10.What is the CSS Box model and what are its elements?

11.What is the float property of CSS?

12.What is the purpose of the z-index and how is it used?

13.Explain the difference between visibility: hidden and display: none?

14.What is tweening?

15.What are the advantages of using CSS & What are the disadvantages of using CSS?

16.What are the different units used in CSS?

17.What is the overflow property in CSS used for?

18.What is the property that is used for controlling image-scroll

19.What are the various font-related attributes in CSS?

20.What is CSS flexbox?

21.What’s the difference between a relative, fixed, absolute and statically positioned element?

22.What are Vendor-Prefixes?

23.Give an example of how you would use counter-increment and counter-reset in CSS to create automatic numbering within a webpage.

24.Explain about breakpoints?

25.Define ‘important’ declarations used in CSS.

26.What is css grid?

27.How many ways to align center div

28.Explain about Image sprite?

29.What is word-wrapping in CSS3?

30.How to add '..." for sentance?

**1.What is CSS? Explain different versions?**

* CSS stands for Cascading Style Sheet. It is a popular styling language which is used with HTML to design websites.
* It is a language by which we can set the behavior of an HTML webpage. It describes how the HTML content will be shown on screen.
* CSS was first developed in 1997 as a way for web developers to define the visual appearance of the web pages that they were creating.

CSS is developed By Hakon Wium Lie; Bert Bos;

CSS1

CSS2->1998

CSS2.1

CSS3->1999

CSS4

CSS was originally released in 1996 and consists of properties for adding font properties such as typeface and emphasis color of text, backgrounds, and other elements.

CSS2 was released in 1998 with added styles for other media types so that it can be used for page layout designing.

CSS3 was released in 1999 and presentation-style properties were added in it that allows you to build a presentation from documents.

Unlike CSS2, which was comprised of a single document, CSS3 has its specifications divided into many individual modules, which makes CSS3 a whole lot easier to handle.

With CSS3, the designers can now use special fonts, like those available in Google Fonts and Typecast.

Earlier, with CSS and CSS2, designers could only use “web-safe fonts” for being 100% sure to use fonts that would always display the same on every machine.

While CSS and CSS2 had ‘simple selectors’, CSS3 calls the components as ‘a sequence of simple selectors’.

CSS3 came up with some key web design considerations like rounded borders that help in rounding up the borders without any hassle.

This turned out to be a huge plus point for developers who were struggling with initial versions of CSS borders.

CSS3 has the capability to split text sections into multiple columns so that it can be read like a newspaper.

In CSS2, the developers had difficulty because the standard was not equipped with automatically breaking the text so that it fits within a box.

**There are three ways that you could integrate a certain CSS style:**

* You can integrate your style using the style-tags in the head section of your HTML page.
* You can integrate your style using inline-styling.
* You can write your CSS in a separate file and add it to your HTML page using the link tag.

**There are several modules in CSS as stated below:**

* Selectors
* Box Model
* Backgrounds and Borders
* Text Effects
* 2D/3D Transformations
* Animations
* Multiple Column Layout
* User Interface.

**2.What are the advantages of CSS?**

Following are the advantages of using CSS

**1. CSS saves time:**

You can write CSS once and then reuse same sheet in multiple HTML pages.

You can define a style for each HTML element and apply it to as many Web pages as you want.

**2.Pages load faster:**

If you are using CSS, you do not need to write HTML tag attributes every time.

Just write one CSS rule of a tag and apply it to all the occurrences of that tag.

So less code means faster download times.

**3. Easy maintenance:**

To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.

**4.Multiple Device Compatibility:**

Style sheets allow content to be optimized for more than one type of device.

By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.

**5.Global web standards**:

Now HTML attributes are being deprecated and it is being recommended to use CSS. So it’s a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

**6.Offline Browsing:**

CSS can store web applications locally with the help of an offline cache.

Using of this, we can view offline websites. The cache also ensures faster loading and better overall performance of the website.

**7.Platform Independence:**

The Script offer consistent platform independence and can support latest browsers as well.

**3.What are the CSS frameworks?**

CSS frameworks provide a basic structure for designing consistent solutions to tackle common recurring issues across front end web development. They provide generic functionality which can be overridden for specific scenarios and applications. This greatly decreases the time needed to start creating applications and websites.

This way developers do not have to start from scratch every time when building applications. They can reuse the basic foundation from earlier applications and get straight into working on the crux of the website or application rather than coding in every little details again and again for each application.

**Ex:**

Bootstrap ,Foundation,Pure

**4.What is a CSS selector?**

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

There are several different types of selectors in CSS.

**CSS Element Selector**

**CSS Id Selector**

**CSS Class Selector**

**CSS Universal Selector**

**CSS Group Selector**

**1) CSS Element Selector**

The element selector selects the HTML element by name.

<!DOCTYPE html>

<html>

<head>

<style>

p{

text-align: center;

color: blue;

}

</style>

</head>

<body>

<p>This style will be applied on every paragraph.</p>

<p id="para1">Me too!</p>

<p>And me!</p>

</body>

</html>

**2) CSS Id Selector**

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

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

Let?s take an example with the id "para1".

<!DOCTYPE html>

<html>

<head>

<style>

#para1 {

text-align: center;

color: blue;

}

</style>

</head>

<body>

<p id="para1">Hello Javatpoint.com</p>

<p>This paragraph will not be affected.</p>

</body>

</html>

**3) CSS 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.

Note: A class name should not be started with a number.

Let's take an example with a class "center".

<!DOCTYPE html>

<html>

<head>

<style>

.center {

text-align: center;

color: blue;

}

</style>

</head>

<body>

<h1 class="center">This heading is blue and center-aligned.</h1>

<p class="center">This paragraph is blue and center-aligned.</p>

</body>

</html>

**CSS Class Selector for specific element**

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

**example.**

<!DOCTYPE html>

<html>

<head>

<style>

p.center {

text-align: center;

color: blue;

}

</style>

</head>

<body>

<h1 class="center">This heading is not affected</h1>

<p class="center">This paragraph is blue and center-aligned.</p>

</body>

</html>

**4) CSS Universal Selector**

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

<!DOCTYPE html>

<html>

<head>

<style>

\* {

color: green;

font-size: 20px;

}

</style>

</head>

<body>

<h2>This is heading</h2>

<p>This style will be applied on every paragraph.</p>

<p id="para1">Me too!</p>

<p>And me!</p>

</body>

</html>

**5) CSS Group Selector**

The grouping selector is used to select all the elements with the same style definitions.

Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

Let's see the CSS code without group selector.

h1 {

text-align: center;

color: blue;

}

h2 {

text-align: center;

color: blue;

}

p {

text-align: center;

color: blue;

}

As you can see, you need to define CSS properties for all the elements. It can be grouped in following ways:

h1,h2,p {

text-align: center;

color: blue;

}

Let's see the full example of CSS group selector.

<!DOCTYPE html>

<html>

<head>

<style>

h1, h2, p {

text-align: center;

color: blue;

}

</style>

</head>

<body>

<h1>Hello Javatpoint.com</h1>

<h2>Hello Javatpoint.com (In smaller font)</h2>

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

</body>

</html>

**5.What is Pseudo-elements vs Pseudo-class?**

**Pseudo-Elements:**

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

Style the first letter, or line, of an element

Insert content before, or after, the content of an element

selector::pseudo-element {

property: value;

}

p::first-line {

color: #ff0000;

font-variant: small-caps;

**}**

**Pseudo-class:**

A pseudo-class is used to define a special state of an element.

/\* unvisited link \*/

a:link {

color: red;

}

/\* visited link \*/

a:visited {

color: green;

}

/\* mouse over link \*/

a:hover {

color: hotpink;

}

/\* selected link \*/

a:active {

color: blue;

}

</style>

</head>

<body>

<p><b><a href="default.asp" target="\_blank">This is a link</a></b></p>

**6.What is the difference between class selectors and id selectors?**

**The CSS id Selector:**

The id selector uses the id attribute of an HTML element to select a specific element.

#para1 {

text-align: center;

color: red;

}

<p id="para1">Hello World!</p>

<p>This paragraph is not affected by the style.</p>

**The CSS class Selector:**

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

.center {

text-align: center;

color: red;

}

<h1 class="center">Red and center-aligned heading</h1>

<p >Red and center-aligned paragraph.</p>

**7.Advantages of External Stylesheets?**

* You can use the same .css file for multiple pages.
* Since the CSS code is in a separate document, your HTML files will have a cleaner structure and are smaller in size

**8.What is RWD?**

Responsive web design makes your web page look good on all devices.

**Viewport:**

* The viewport is the user's visible area of a web page.
* The viewport varies with the device, and will be smaller on a mobile phone than on aComputer screen.
* HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

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

This gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

**Media queries:**

Media query is a CSS technique introduced in CSS3.

We can add a breakpoint where certain parts of the design will behave differently on each side of the breakpoint.

**Break Points:**

/\* Extra small devices (phones, 600px and down) \*/

@media only screen and (max-width: 600px) {...}

/\* Small devices (portrait tablets and large phones, 600px and up) \*/

@media only screen and (min-width: 600px) {...}

/\* Medium devices (landscape tablets, 768px and up) \*/

@media only screen and (min-width: 768px) {...}

/\* Large devices (laptops/desktops, 992px and up) \*/

@media only screen and (min-width: 992px) {...}

/\* Extra large devices (large laptops and desktops, 1200px and up) \*/

@media only screen and (min-width: 1200px) {...}

@media (min-width: 600px) and (max-width: 992px) { ... }

Orientation: Portrait / Landscape:

Media queries can also be used to change layout of a page depending on the orientation of the browser.

@media only screen and (orientation: landscape) {

body {

background-color: lightblue }}

**9.What are the benefits of CSS sprites?**

CSS Sprites are a means of combining multiple images into a single image file for use on a website

CSS Sprites is technique which you can use to reduce the server load.

**Example:**

<img id="home" src="img\_trans.gif" width="1" height="1"><br><br>

<img id="next" src="img\_trans.gif" width="1" height="1">

#home {

width: 46px;

height: 44px;

background: url(img\_navsprites.gif) 0 0;

}

#next {

width: 43px;

height: 44px;

background: url(img\_navsprites.gif) -91px 0;

}

**10. What is the CSS Box model and what are its elements?**

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.

**Content box:** The area where your content is displayed, which can be sized using properties like width and height.

**Padding box**: The padding sits around the content as white space; its size can be controlled using padding and related properties.

**Border box:** The border box wraps the content and any padding. Its size and style can be controlled using border and related properties.

**Margin box:** The margin is the outermost layer, wrapping the content, padding and border as whitespace between this box and other elements. Its size can be controlled using margin and related properties.

**11.What is the float property of CSS?**

The float property specifies how an element should float.

Note: Absolutely positioned elements ignore the float property!

Note: Elements after a floating element will flow around it.

To avoid this, use the clear property or the clearfix hack.

**Values for float:**

1.left

The element must float on the left side of its containing block.

2.right

The element must float on the right side of its containing block.

3.none

The element must not float.

4.inline-start

The element must float on the start side of its containing block. That is the left side with ltr scripts, and the right side with rtl scripts.

5.inline-end

The element must float on the end side of its containing block. That is the right side with ltr scripts, and the left side with rtl scripts.

**Example:**

**example:**

css:

<style>

section {

border: 1px solid blue;

width: 100%;

float: left;

}

div {

margin: 5px;

width: 50px;

height: 150px;

}

.left {

float: left;

background: pink;

}

.right {

float: right;

background: cyan;

}

</style>

html:

<section>

<div class="left">1</div>

<div class="left">2</div>

<div class="right">3</div>

<p>Lorem ipsum dolor sit amet</p>

</section>

**12.What is the purpose of the z-index and how is it used?**

The z-index property specifies the stack order of an element.

An element with greater stack order is always in front of an element with a lower stack order.

Note: z-index only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky).

z-index:999999(value always number positive or negative)

**Value and Description:**

**auto:**Sets the stack order equal to its parents. This is default

**number:**Sets the stack order of the element. Negative numbers are allowed

**initial:** Sets this property to its default value. Read about initial

**html code:**

<div class="dashed-box">Dashed box

<span class="gold-box">Gold box</span>

<span class="green-box">Green box</span>

</div>

**css code:**

.dashed-box {

position: relative;

z-index: 1;

border: dashed;

height: 8em;

margin-bottom: 1em;

margin-top: 2em;

}

.gold-box {

position: absolute;

z-index: 3; /\* put .gold-box above .green-box and .dashed-box \*/

background: gold;

width: 80%;

left: 60px;

top: 3em;

}

.green-box {

position: absolute;

z-index: 2; /\* put .green-box above .dashed-box \*/

background: lightgreen;

width: 20%;

left: 65%;

top: -25px;

height: 7em;

opacity: 0.9;

}

**13.Explain the difference between visibility: hidden and display: none?**

display:none means that the tag in question will not appear on the page at all (although you can still interact with it through the dom).

There will be no space allocated for it between the other tags.

**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

<style>

#p2{

display: none;

}

</style>

</head>

<body>

<p id="p1">Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>

<p id="p2">Lorem ipsum dolor sit amet consectetur, adipisicing elit.</p>

<p id="p3">Lorem ipsum dolor sit amet, consectetur adipisicing elit.p>

</body>

</html>

Visibility:hidden means that unlike display:none, the tag is not visible, but space is allocated for it on the page.

The tag is rendered, it just isn't seen on the page.

example:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

<style>

#p2{

visibility: hidden;

}

</style>

</head>

<body>

<p id="p1">Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>

<p id="p2">Lorem ipsum dolor sit amet consectetur, adipisicing elit.</p>

<p id="p3">Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>

</body>

</html>

**14.What is tweening?**

It is a process of creating intermediary frames between two images to provide the appearance that the first image develops efficiently into the second image it is a key process which is used in all types of animations.

**15.What are the advantages of using CSS & What are the disadvantages of using CSS?**

**List of Advantages of Cascading Style Sheets**

**1. Consistency**

The main benefit of CSS is that style is applied consistently across a number of web pages. One command line can control several areas at one time, which is quite advantageous if there are changes that need to be made. You only need to alter one thing and the rest will follow. Because you don’t have to change each page one at a time, web designers can be very efficient in creating and changing a website with only a few lines of code.

**2. Improved website speed**

Web designers only need to use a small amount of lines of programming for each page. And if there are less code, there are fewer lines to read, resulting in a faster load time for every page. With online users not willing to wait for a website to load, improving site speed will be most advantageous. Owners who bank on website performance to improve search engine rankings and customer base will benefit from CSS.

**3. Easy to maintain**

Cascading style sheet not only simplifies website development, but also maintenance. All the codes are placed on one page, which means making improvements or changing a few lines will not involve going through several pages. And since a change with a single line of code is applied across the website, maintenance time and effort are significantly reduced.

**List of Disadvantages of Cascading Style Sheets**

**1. Come in different levels**

There’s CSS, CSS 1 up to CSS3, which has resulted in confusion among developers and web browsers. One type of CSS should be enough. It would be preferable than having to choose which CSS level to use.

**2. Fragmentation**

With CSS, what works with one browser may not always work with another. This is why web developers have to test for compatibility, running the program across multiple browsers before a website is set live. If only people use Mozilla or Chrome, but they don’t.

**3. Lack of security**

Because it is an open text-based system, CSS doesn’t have the built-in security that will protect it from being overridden. Anyone who has a read/write access to a website can change the CSS file, alter the links or disrupt the formatting, whether by accident or design.

**16.What are the different units used in CSS?**

There are various units in CSS to express the measurement and length. A CSS unit is used to determine the property size, which we set for an element or its content. The units in CSS are required to define the measurement such as margin: 20px; in which the px (or pixel) is the CSS unit. They are used to set margin, padding, lengths, and so on.

We cannot apply the whitespace between the number and the unit. The unit can be omitted for the value 0. Some properties of CSS allow the negative values of length.

**The length unit in CSS is of two types:**

* **Absolute length.**
* **Relative length.**

**Absolute lengths**

These are the fixed-length units, and the length expressed using the absolute units will appear as exactly that size. It is not recommended to use on-screen, because the size of the screen varies too much. So, the absolute units should be used when the medium of output is known, such as the print layout.

Absolute units are useful when the responsiveness is not considered in a project. They are less favorable for the responsive sites because they do not scale when the screen changes.

Generally, absolute lengths are considered to be the same size always. The absolute length units are tabulated as follows:

**Unit Name Explanation**

cm Centimeters It is used to define the measurement in centimeters.

mm Millimeters It is used to define the measurement in millimeters.

in Inches It is used to define the measurement in inches.1in = 96px 2.54cm

pt Points It is used to define the measurement in points.1pt = 1/72 of 1inch.

pc Picas It is used to define the measurement in picas.1pc = 12pt so, there 6 picas is equivalent to 1 inch.

px Pixels It is used to define the measurement in pixels.1px = 1/96th of inch

**Relative lengths**

Relative units are good to style the responsive site because they scale relative to the window size or the parent. They specify the length, which is relative to another length property.

Depending on the device, if the size of the screen varies too much, then the relative length units are the best because they scale better between the different rendering mediums. We can use the relative units as the default for the responsive units. It helps us to avoid update styles for different screen sizes.

The relative length units are tabulated as follows:

**Unit Name**

em It is relative to the font-size of the element.

ex It is relative to the x-height of the font of the element. It is rarely used. The x-height is determined by the height of the lowercase letter 'x'.

ch It is similar to the unit ex, but instead of using the height of the letter x, it measures the width of the integer "0" (zero).

rem It is the font-size of the root element

vh It is relative to the height of the viewport.1vh = 1% or 1/100 of the height of the viewport.

vw It is relative to the width of the viewport.1vw = 1% or 1/100 of the width of viewport

vmin It is relative to the smaller dimension of the viewport.1vmin = 1% or 1/100 of the viewport's smaller dimension.

vmax It is relative to the larger dimension of the viewport.1vmax = 1% or 1/100 of the viewport's larger dimension.

% It is used to define the measurement as a percentage that is relative to another value.

**CSS units: Time**

Some animation properties require values to express in time.

**Unit Explanation**

s It is the duration of time in seconds.

ms It is the duration of time in milliseconds.

1ms = 1/100 of a second

**CSS units: Angles**

The transform properties in CSS require values to express in angles

**Unit Explanation**

deg It expresses the angles in degrees.

grad It expresses the angles in gradians, i.e., 1/400 of a turn.

turn It expresses the angles in turns, i.e., 360 degrees.

**17.What is the overflow property in CSS used for?**

The overflow property specifies what should happen if content overflows an element's box.

**overflow:** visible|hidden|scroll|auto|initial|inherit;

**visible:** he overflow is not clipped. It renders outside the element's box. This is default

**hidden:** The overflow is clipped, and the rest of the content will be invisible

**scroll:** The overflow is clipped, but a scroll-bar is added to see the rest of the content

**auto:** If overflow is clipped, a scroll-bar should be added to see the rest of the content

**initial:** Sets this property to its default value.

**Inherit:** Inherits this property from its parent element

[https://www.w3schools.com/cssref/playit.asp?filename=playcss\_overflow HYPERLINK "https://www.w3schools.com/cssref/playit.asp?filename=playcss\_overflow&preval=hidden" HYPERLINK "https://www.w3schools.com/cssref/playit.asp?filename=playcss\_overflow HYPERLINK "https://www.w3schools.com/cssref/playit.asp?filename=playcss\_overflow&preval=hidden"& HYPERLINK "https://www.w3schools.com/cssref/playit.asp?filename=playcss\_overflow&preval=hidden"preval=hidden" HYPERLINK "https://www.w3schools.com/cssref/playit.asp?filename=playcss\_overflow&preval=hidden"& HYPERLINK "https://www.w3schools.com/cssref/playit.asp?filename=playcss\_overflow&preval=hidden" HYPERLINK "https://www.w3schools.com/cssref/playit.asp?filename=playcss\_overflow HYPERLINK "https://www.w3schools.com/cssref/playit.asp?filename=playcss\_overflow&preval=hidden"& HYPERLINK "https://www.w3schools.com/cssref/playit.asp?filename=playcss\_overflow&preval=hidden"preval=hidden" HYPERLINK "https://www.w3schools.com/cssref/playit.asp?filename=playcss\_overflow&preval=hidden"preval=hidden](https://www.w3schools.com/cssref/playit.asp?filename=playcss_overflow&preval=hidden)

**18.What is the property that is used for controlling image-scroll?**

 To set the scrolling of an image in the background, use the *background-attachment* property.

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

body, html {

  height: 100%;

  margin: 0;

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

}

\* {

  box-sizing: border-box;

}

.bg-image {

  /\* Full height \*/

  height: 50%;

  /\* Center and scale the image nicely \*/

  background-position: center;

  background-repeat: no-repeat;

  background-size: cover;

}

/\* Images used \*/

.img1 { background-image: url("img\_snow.jpg"); }

.img2 { background-image: url("img\_girl.jpg"); }

.img3 { background-image: url("img\_lights.jpg"); }

.img4 { background-image: url("img\_nature.jpg"); }

.img5 { background-image: url("img\_forest.jpg"); }

.img6 { background-image: url("img\_woods.jpg"); }

/\* Position text in the middle of the page/image \*/

.bg-text {

  background-color: rgb(0,0,0); /\* Fallback color \*/

  background-color: rgba(0,0,0, 0.4); /\* Black w/opacity/see-through \*/

  color: white;

  font-weight: bold;

  font-size: 80px;

  border: 10px solid #f1f1f1;

  position: fixed;

  top: 50%;

  left: 50%;

  transform: translate(-50%, -50%);

  z-index: 2;

  width: 300px;

  padding: 20px;

 text-align: center;

}

</style>

</head>

<body>

<div class="bg-image img1"></div>

<div class="bg-image img2"></div>

<div class="bg-image img3"></div>

<div class="bg-image img4"></div>

<div class="bg-image img5"></div>

<div class="bg-image img6"></div>

<div class="bg-text">TEXT</div>

</body>

</html>

**19.What are the various font-related attributes in CSS?**

The font property is a shorthand property for:

**1.font-style:**

**Syntax:**

font-style: normal|italic|oblique|initial|inherit;

**2.font-variant:**

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

p.normal {

font-variant: normal;

}

p.small {

font-variant: small-caps;

}

</style>

</head>

<body>

<h1>The font-variant Property</h1>

<p class="normal">My name is Hege Refsnes.</p>

<p class="small">My name is Hege Refsnes.</p>

</body>

</html>

Output:My name is Hege Refsnes.

My name is Hege Refsnes.// this will show like small letters will be in uppercase letters but appears in a smaller font size than the original uppercase letters in the text .

**3.font-weight:**

**Example:**

<html>

<head>

<style>

p.normal {

font-weight: normal;-------------------->1

}

p.light {

font-weight: lighter;-------------------->2

}

p.thick {

font-weight: bold;----------------------->3

}

p.thicker {

font-weight: 800;----------------------->4

}

</style>

</head>

<body>

<h1>The font-weight Property</h1>

<p class="normal">This is a paragraph.</p>

<p class="light">This is a paragraph.</p>

<p class="thick">This is a paragraph.</p>

<p class="thicker">This is a paragraph.</p>

</body>

</html>

**4.font-size/line-height:**

Example:

<html>

<head>

<style>

div.a {

font-size: 15px;---------------->1

}

div.b {

font-size: large;---------------->2

}

div.c {

font-size: 150%;----------------->3

}

</style>

</head>

<body>

<h1>The font-size Property</h1>

<div class="a">This is some text.</div>

<div class="b">This is some text.</div>

<div class="c">This is some text.</div>

</body>

</html>

**5.font-family:**

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

p.a {

font-family: "Times New Roman", Times, serif;

}

p.b {

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

}

</style>

</head>

<body>

<h1>The font-family Property</h1>

<p class="a">This is a paragraph, shown in the Times New Roman font.</p>

<p class="b">This is a paragraph, shown in the Arial font.</p>

</body>

</html>

**20.What is CSS flexbox?**

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

To start using the Flexbox model, you need to first define a flex container.

Example:

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

margin: 10px;

padding: 20px;

font-size: 30px;

}

</style>

</head>

<body>

<div class="flex-container">--------------------------------------------> flex container

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

**The flex container properties are:**

**1.flex-direction:** The flex-direction property defines in which direction the container wants to stack the flex items.

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

flex-direction: column/column-reverse/row/row-reverse;---------------------------------->flex-direction

background-color: DodgerBlue;

**}**

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

**2.flex-wrap:** The flex-wrap property specifies whether the flex items should wrap or not.

Example:

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

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

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

</body>

</html>

**3.flex-flow:** The flex-flow property is a shorthand property for setting both the flex-direction and flex-wrap properties.

**Example:**follow the above example in place of flex-wrap we have to add flex-flow and the value is "row wrap"

**4.justify-content:** The justify-content property is used to align the flex items:

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

justify-content: center/flex-start/flex-end/space-around/space-between;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

font-size: 30px;

}

</style>

</head>

<body>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

**5.align-items:** The align-items property is used to align the flex items.

**Example:** <!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 200px;

align-items: center/flex-start/

flex-end/stretch/baseline;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>The align-items Property</h1>

<p>The "align-items: center;" aligns the flex items in the middle of the container:</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

**6.align-content:** The align-content property is used to align the flex lines.

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 600px;

flex-wrap: wrap;

align-content: space-between/space- around/stretch/center/flex-start/flex-end;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>The align-content Property</h1>

<p>The "align-content: space-between;" displays the flex lines with equal space between them:</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

</body>

</html>

**21.What’s the difference between a relative, fixed, absolute and statically positioned element?**

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

Types:

There are five different position values:

• static

• relative

• fixed

• absolute

• sticky

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

**position: static**

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

**Syntax:**

div.static {

position: static;

border: 3px solid #73AD21;

}

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

div.static {

position: static;

border: 3px solid #73AD21;

}

</style>

</head>

<body>

<h2>position: static;</h2>

<p>An element with position: static; is not positioned in any special way; it is

always positioned according to the normal flow of the page:</p>

<div class="static">

This div element has position: static;

</div>

</body>

</html>

**OUTPUT:**

<https://www.w3schools.com/css/tryit.asp?filename=trycss_position_static>

**position: relative**

An element with position: relative; is positioned relative to its normal position.

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

Syntax:

div.relative {

position: relative;

left: 30px;

border: 3px solid #73AD21;

}

Example:

<!DOCTYPE html>

<html>

<head>

<style>

div.relative {

position: relative;

left: 30px;

border: 3px solid #73AD21;

}

</style>

</head>

<body>

<h2>position: relative;</h2>

<p>An element with position: relative; is positioned relative to its normal position:</p>

<div class="relative">

This div element has position: relative;

</div>

</body>

</html>

**position: fixed**

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

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

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

Syntax:

div.fixed {

position: fixed;

bottom: 0;

right: 0;

width: 300px;

border: 3px solid #73AD21;

}

Example:

<style>

div.fixed {

position: fixed;

bottom: 0;

right: 0;

width: 300px;

border: 3px solid #73AD21;

}

</style>

</head>

<body>

<h2>position: fixed;</h2>

<p>An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:</p>

<div class="fixed">

This div element has position: fixed;

</div>

**position: absolute**

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: A "positioned" element is one whose position is anything except static.

Here is a simple example:

This <div> element has position: relative;

This <div> element has position: absolute;

Here is the CSS that is used:

Syntax:

div.relative {

position: relative;

width: 400px;

height: 200px;

border: 3px solid #73AD21;

}

div.absolute {

position: absolute;

top: 80px;

right: 0;

width: 200px;

height: 100px;

border: 3px solid #73AD21;

}

Example:

<style>

div.relative {

position: relative;

width: 400px;

height: 200px;

border: 3px solid #73AD21;

}

div.absolute {

position: absolute;

top: 80px;

right: 0;

width: 200px;

height: 100px;

border: 3px solid #73AD21;

}

</style>

</head>

<body>

<h2>position: absolute;</h2>

<p>An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):</p>

<div class="relative">This div element has position: relative;

<div class="absolute">This div element has position: absolute;</div>

</div>

**position: sticky**

An element with position: sticky; is positioned based on the user's scroll position.

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

Syntax:

div.sticky {

position: -webkit-sticky; /\* Safari \*/

position: sticky;

top: 0;

background-color: green;

border: 2px solid #4CAF50;

}

Example:

<style>

div.sticky {

position: -webkit-sticky;

position: sticky;

top: 0;

padding: 5px;

background-color: #cae8ca;

border: 2px solid #4CAF50;

}

</style>

</head>

<body>

<p>Try to <b>scroll</b> inside this frame to understand how sticky positioning works.</p>

<p>Note: IE/Edge 15 and earlier versions do not support sticky position.</p>

<div class="sticky">I am sticky!</div>

<div style="padding-bottom:2000px">

<p>In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.</p>

<p>Scroll back up to remove the stickiness.</p>

<p>Some text to enable scrolling.. Lorem ipsum dolor sit amet.</p>

<p>Some text to enable scrolling.. Lorem ipsum dolor sit amet</p>

</div>

</body>

</html>

**22.What are Vendor-Prefixes?**

Simply put, vendor prefixes are a way for your browser to support new CSS features before they become fully supported in all browsers. When CSS3 became popular, all sorts of new features started appearing.

When CSS3 became popular, all sorts of new features started appearing. Unfortunately, not all of them were supported across all browsers. Vendor prefixes helped developers use those new features, and have them supported instantly without having to wait for each of them to become available for every browser.

Vendor prefixes are not a hack, and you should feel free to use them.

The Prefixes

Major browsers use the following prefixes:

• -webkit- Chrome, Safari, newer versions of Opera, almost all iOS browsers.

• -moz- Firefox.

• -o- Old versions of Opera.

• -ms- Microsoft Edge and Internet Explorer.

How Should You Use Them?

You can easily use vendor prefixes, simply by adding them before the property, like this:

.element {

-webkit-transform: rotate(60deg);

-ms-transform: rotate(60deg);

-o-transform: rotate(60deg);

transform: rotate(60deg);

}

**23.Give an example of how you would use counter-increment and counter-reset in CSS to create automatic numbering within a webpage.**

<!DOCTYPE html>

<html>

<head>

<style>

body {

/\* Set "my-sec-counter" to 0 \*/

counter-reset: my-sec-counter;

}

h2::before {

/\* Increment "my-sec-counter" by 1 \*/

counter-increment: my-sec-counter;

content: "Section " counter(my-sec-counter) ". ";

}

</style>

</head>

<body>

<h2>HTML Tutorial</h2>

<h2>CSS Tutorial</h2>

<h2>JavaScript Tutorial</h2>

<h2>Bootstrap Tutorial</h2>

<h2>SQL Tutorial</h2>

<h2>PHP Tutorial</h2>

</body>

</html>

**24.Explain about breakpoints?**

By executing code one line or one function at a time, you can observe changes in the data and in the page to understand exactly what is happening. You can also modify data values used by the script, and you can even modify the script itself.

Why is this variable value 20 instead of 30? Why doesn't that line of code seem to have any effect? Why is this flag true when it should be false? Every developer faces these questions, and steps through code to find out.

After setting breakpoints, return to the page and use it normally until a breakpoint is reached. This pauses all JavaScript on the page, focus shifts to the DevTools Sources panel, and the breakpoint is highlighted. You can now selectively execute code and examine its data, step by step.

All step options are represented through clickable icons Breakpoints button bar in the sidebar,

but can also be triggered via shortcut. Here's the rundown:

Resume: Resumes execution up to the next breakpoint. If no breakpoint is encountered, normal execution is resumed.

Long Resume :Resumes execution with breakpoints disabled for 500ms. Convenient for momentarily skipping breakpoints that would otherwise continually pause the code, e.g., a breakpoint inside a loop.

Click and hold Resume until expands to show the action.

Step Over : Executes whatever happens on the next line and jumps to the next line.

Step Into : If the next line contains a function call, Step Into will jump to and pause that function at its first line.

Step Out : Executes the remainder of the current function and then pauses at the next statement after the function call.

Deactivate breakpoints Temporarily disables all breakpoints. Use to resume full execution without actually removing your breakpoints. Click it again to reactivate the breakpoints.

Pause on exceptions Automatically pauses the code when an exception occurs.

<https://developers.google.com/web/tools/chrome-devtools/javascript/breakpoints>

**25. Define ‘important’ declarations used in CSS?**

The !important statement is used for breaking this order and adding weight to a

specific declaration to put it in effect, ignoring the others.

It is used for overriding the styles that are previously declared in other style sources

**Example:**

**HTML code:**

<p id="example">

Hello World

</p>

**CSS code:**

#example {​​​​​​​

color:green !important;

}​​​​​​​

#example {​​​​​​​

color: yellow;

}​​​​​​​

**26.what is CSS grid?**

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns,

making it easier to design web pages without having to use floats and positioning.

A grid layout consists of a parent element, with one or more child elements.

**Example:**

**HTML code:**

<div id="container">

<header>Header</header>

<aside>Aside</aside>

<main>Main</main>

<footer>Footer</footer>

</div>

**CSS code:**

#container {

width: 100%;

height: 100vh;

background: yellow;

display:grid;

grid-template-rows: 80px 550px ;

grid-template-columns: 1fr 1fr;

}

aside {

background: green;

}

main {

background: pink;

}

footer {

background: yellow;

}

header {

grid-column: 1 / 3;

background: blue;

}

/\* grid-column-gap: 10px; \*/

**27.** **How many ways to align center div?**

**i) Using Flex layout:**

Flex layout is quickest way to to center multiple elements in a single div.

Also it can be used on both inline and block element.

The advantage of flex display is you don’t need to specify the width and height of the div at all (ex. div with just text)

HTML:

<div id="container">

<p>Using Flex to align center</p>

</div>

CSS:

#container

{

display: flex;

justify-content: center;

}

**ii) margin: auto**

First, we can only use this for element that have width defined.

The element must have block or table display and must not have fixed or absolute positioned. And, obviously, you can’t use it for vertical align.

So it’s not quite flexible and should be used for a certain scenario.

HTML:

<div id="container1">

<p>Using margin 0 Center</p>

</div>

**CSS code:**

#container1

{

margin: auto;

width: 30%;

}

**iii) text-align and inline-block:**

This will force our div to behave like inline element and

therefore subjected text-align center.

The difference between this method and margin: 0 auto is we don’t need the width to be specified.

HTML:

<div class="parent-div">

<p>Parent Div to align Center</p>

<div class="example-div">

<p>Example Div to align center</p>

</div>

</div>

CSS:

.example-div {

display: inline-block;

}

.parent-div {

text-align: center;

}

**iv) 2D Transform:**

we need to set the element position to absolute and top/left to 50%.

This will the div to the center of the screen.

Then add minus 50% translate for both x and y to offset the element size.

**HTML code:**

<div class="transform">

<p>2D transform Div to align Center</p>

</div>

**CSS code:**

.transform {

top: 50%;

left: 50%;

transform: translate(-50% , -50%);

position: absolute;

}

**28. Explain about Image sprite?**

An image sprite is a collection of images and put into a single image.

**Example:** Instead of using three separate images, we use this single image ("img\_navsprites.gif"):



**29. What is word-wrapping in CSS3?**

A Word or Text to be carried over to a new line automatically when the margin is reached(It breaks the text to next line).



**CSS Syntax:**

word-wrap: normal,break-word,initial,inherit;

**30. How to add '..." for sentance?**

**CSS text-overflow** Property is used to specify that some text has overflown and hidden from the view. The **white-space property** must be set to nowrap and **overflow property** must be hidden.

**ellipsis:** Text is clipped and the clipped text is represented as ‘…’ .

**Syntax:**

text-overflow: ellipsis;

<!DOCTYPE html>

<html>

<head>

<style>

div.a {

white-space: nowrap;

width: 200px;

font-size :50px;

overflow: hidden;

text-overflow: ellipsis;

border: 1px solid #000000;

}

</style>

</head>

<body>

<h1>The text-overflow Property</h1>

<div class="a">This is some long text that will not fit in the box.</div>

</body>

</html>