LTI MINDTREE:

**1.HOW REACT IS DIFFERENT FROM FRAMEWORKS?**

**A Virtual DOM**: React uses a virtual DOM (Document Object Model) instead of directly interacting with the real DOM. This allows React to efficiently update only the parts of the page that need to be changed, instead of re-rendering the entire page every time a change occurs. Other frameworks, such as Angular and Vue, also use a virtual DOM approach.

**Component-Based**: React is a component-based framework, meaning that the UI is broken down into smaller, reusable components that can be combined to create complex UIs. This makes it easier to maintain and test large codebases, as well as promote code reusability. Other frameworks, such as Angular and Vue, also use a component-based approach.

**JSX:** React uses JSX, a syntax extension to JavaScript, to define components. JSX allows developers to write HTML-like syntax within JavaScript, which makes it easier to reason about and visualize the structure of the UI. Other frameworks, such as Vue, also support a similar templating system.

**Unopinionated**: React is unopinionated, meaning that it does not impose a specific architecture or project structure on developers. This gives developers more flexibility and freedom to choose how they want to structure their codebase. Other frameworks, such as Angular, have a more opinionated approach.

**One-way Data Binding**: React uses one-way data binding, which means that data flows in one direction from the parent component to its child components. This makes it easier to reason about the flow of data within the application, and reduces the risk of data inconsistencies. Other frameworks, such as Angular, use two-way data binding, which can be more complex to manage.

**CONSLUSION:**

Overall, React's unique combination of virtual DOM, component-based architecture, JSX syntax, unopinionated approach, and one-way data binding sets it apart from other web frameworks and makes it a popular choice for building dynamic and complex web applications.

**2.WHAT IS VIRTUAL DOM?**

A. the **Virtual DOM** in React is a lightweight representation of the real DOM that helps to improve the performance of web applications by reducing the number of updates required to the real DOM.

**3.HOOKS AND USESTATE AND USEEFFECT?**

**Hooks** are functions that allow developers to use state and other React features within functional components.

**useState** is a built-in hook in React that allows developers to add state to functional components. It returns an array with two elements: the current state value, and a function to update the state. useState is typically used to manage simple state values such as booleans, strings, or numbers.

**import React, { useState } from 'react';**

**function Example() {**

**const [count, setCount] = useState(0);**

**return (**

**<div>**

**<p>You clicked {count} times</p>**

**<button onClick={() => setCount(count + 1)}>**

**Click me**

**</button>**

**</div>**

**);**

**}**

**useEffect** is another built-in hook in React that allows developers to perform side effects within functional components. Side effects are any operations that affect something outside of the component, such as updating the document title, fetching data from an API etc

**import React, { useState, useEffect } from 'react';**

**function Example() {**

**const [count, setCount] = useState(0);**

**useEffect(() => {**

**document.title = `You clicked ${count} times`;**

**});**

**return (**

**<div>**

**<p>You clicked {count} times</p>**

**<button onClick={() => setCount(count + 1)}>**

**Click me**

**</button>**

**</div>**

**);**

**}**

**4.useMemo and useCallback**

The **useMemo** hook is used in the functional component of React to return a memoized value. **Purpose of useMemo** In computer science, memoization is a concept used in general when we can save re-compilation time by returning the cached result. In the functional component of React.

The **useCallback** hook is used when a child component is rerendering over and over again without any need. **Purpose of useCallback** By using useCallback, you can prevent unnecessarily re-rendering components by returning the same instance of the function that is passed instead of creating a new one each time.

**5.How functional component is good as compared to class component?**

Functional components are generally considered to be better than class components for several reasons:

Simplicity: Functional components are simpler and easier to understand than class components. They have a simpler syntax and are easier to write and maintain. They also tend to have fewer bugs and errors.

Performance: Functional components are generally faster than class components because they don't have the overhead of creating an instance of a class. Functional components also have a smaller memory footprint.

Hooks: Functional components can use Hooks, which allow you to use state and other React features without using classes. Hooks make it easier to reuse code and simplify complex logic.

Testing: Functional components are easier to test because they have fewer moving parts. Since they are simpler and have fewer dependencies, it is easier to create unit tests for functional components.

Hooks: Hooks are available in functional components only. It provides a way to reuse component logic without changing component hierarchy.

Overall, functional components are more straightforward to understand, write, and test. They also perform better and can take advantage of Hooks to simplify complex logic. However, there are some situations where class components may still be necessary, such as when you need to use lifecycle methods.

**6.what is HOC and why we need this component in reactJs?**

HOC stands for Higher-Order Component, which is a function that takes a component and returns a new component with extended functionality. The main purpose of an HOC is to reuse component logic across multiple components.

Here are some reasons why you might use an HOC in your React application:

Code reuse: HOCs allow you to reuse component logic across multiple components. For example, if you have two components that share some functionality, you can create an HOC that provides that functionality and then use it in both components.

Cross-cutting concerns: HOCs are useful for adding cross-cutting concerns, such as authentication or logging, to components. By wrapping a component in an HOC, you can add this functionality without modifying the component itself.

Composition: HOCs allow you to compose components together. For example, you can create an HOC that renders a spinner while data is being fetched, and then use that HOC with any component that needs to fetch data.

Flexibility: HOCs are very flexible and can be used for a wide variety of use cases. They allow you to add functionality to a component without modifying its code, which can be useful when you don't have access to the original component's source code.

To create an HOC, **you create a function that takes a component as its argument and returns a new component that wraps the original component. The new component can then provide additional props, state, or other functionality to the original component.**

**7.How to manage data?**

In React, there are several ways to manage data depending on the complexity of your application and your preferences. Here are some common approaches:

State: You can manage data in a component's state using the **useState** hook. This is best for managing data that is specific to a single component and doesn't need to be shared with other components.

Props: You can pass data down to child components through props. This is best for managing data that needs to be shared between components.

Context: You can use the **useContext** hook to share data between components that are not directly connected in the component tree. This is best for managing global or application-level data.

Redux: You can use a library like Redux to manage data in a centralized store that can be accessed by any component. This is best for managing complex data that needs to be shared across many components.

When choosing a data management approach, consider the complexity of your application, the size of your data, and the level of reusability you want to achieve. It's also a good idea to keep your data management strategy as simple as possible, so that it's easy to understand and maintain over time.

**8.Redux?**

Redux is a popular state management library for JavaScript applications, including React. It provides a centralized store for managing state that can be accessed by any component in the application.

Here are some reasons why you might want to use Redux:

Centralized state management: Redux provides a single store for managing all of the state in your application, making it easy to manage and debug.

Predictable state updates: Redux uses a strict unidirectional data flow, making it easy to predict how state will change in response to actions.

Easy debugging: With Redux, you can log all state changes and actions, making it easier to debug and trace issues in your application.

Time travel debugging: Redux allows you to step back and forth through the state changes in your application, making it easier to debug complex issues.

Cross-platform compatibility: Redux can be used with a wide variety of frameworks and libraries, making it a versatile choice for many applications.

However, there are also some potential drawbacks to using Redux:

Complexity: Redux can add complexity to your application, especially for simple applications where simpler solutions might be sufficient.

Learning curve: Redux has a steep learning curve, and it can take some time to fully understand how to use it effectively.

Boilerplate code: Redux can require a lot of boilerplate code, especially when setting up actions and reducers.

Performance: In some cases, Redux can add overhead to your application, leading to slower performance.

Overall, **Redux is a powerful and flexible tool for managing state in JavaScript applications. It can be a great choice for large or complex applications, but may not be necessary for simpler applications. It's important to weigh the advantages and disadvantages carefully and choose the approach that works best for your specific needs.**

**9.SSR?**

**Server-side rendering (SSR) is a technique for rendering web pages on the server before sending them to the browser. With SSR, the server generates the HTML and CSS for a web page and sends it to the browser as a complete page, instead of sending just a basic HTML structure and waiting for the browser to request and render the page's content.**

This approach has several benefits, including improved performance, search engine optimization (SEO), and better support for users with slower internet connections or devices with limited processing power. By generating the entire page on the server, the browser can display content more quickly, and search engines can more easily crawl and index the page's content.

SSR is often used in conjunction with client-side rendering (CSR), where JavaScript is used to enhance the user experience after the initial page load. This combination of SSR and CSR is often referred to as "universal" or "isomorphic" rendering, as it provides the benefits of both approaches.

**10.Scss?**

SCSS (Sassy CSS) is a preprocessor for CSS that adds several features to the standard CSS language, including variables, nesting, inheritance, and modularization. SCSS files are processed by a preprocessor to generate standard CSS files that can be used in web development.

Advantages of using SCSS:

Variables: With SCSS, you can define variables to hold values that you can reuse throughout your stylesheets. This can help reduce code repetition and make your stylesheets more maintainable.

Nesting: You can nest your CSS rules in SCSS, which makes it easier to understand the relationship between different parts of your page.

Inheritance: Inheritance in SCSS allows you to create a hierarchy of styles, where certain styles are inherited by other elements on the page. This can save you time and make your stylesheets more organized.

Modularity: SCSS supports modularization, which means you can split your stylesheets into smaller, more manageable files. This can make it easier to work on large projects with many developers.

Code Reusability: SCSS makes it easy to reuse code, as you can define mixins and functions to reuse throughout your stylesheets.

**Disadvantages of using SCSS**:

Learning Curve: SCSS requires some additional learning compared to plain CSS, so there can be a learning curve for developers who are new to it.

Extra Build Step: Because SCSS files need to be processed by a preprocessor before they can be used in web development, this can add an extra build step to your development process.

Complexity: While SCSS can make your code more maintainable, it can also add complexity to your stylesheets, especially if you don't use it effectively.

In summary, **SCSS is a powerful tool for web development that can make your CSS code more maintainable and efficient. However, it does require some additional learning and can add complexity to your development process**.

**11. What is code splitting?**

Code splitting is a technique used in web development to improve website performance by breaking down a large JavaScript file into smaller, more manageable pieces. This allows the browser to only download the JavaScript code that is required for a particular page, rather than downloading the entire JavaScript file upfront.

The benefits of code splitting are:

Faster page load times: By reducing the amount of JavaScript that needs to be downloaded, the page load times can be significantly improved.

Reduced bandwidth usage: Smaller JavaScript files mean less data that needs to be transferred over the network, which can result in lower bandwidth usage and costs.

Better user experience: Faster page load times can lead to a better user experience, as users are less likely to get frustrated with slow-loading pages.

There are different techniques for implementing code splitting in web development. One common approach is to use a bundler like Webpack, which can analyze the dependencies between modules and generate multiple smaller JavaScript files that can be loaded as needed. Another approach is to use dynamic imports, which allow you to load JavaScript code asynchronously when it is needed, rather than upfront.

In summary, **code splitting is a technique used in web development to improve website performance by breaking down a large JavaScript file into smaller, more manageable pieces. This can result in faster page load times, reduced bandwidth usage, and a better user experience.**

**12.Unit testing (Jest)?**

Jest is a popular JavaScript testing framework that is used for unit testing. It provides a simple and easy-to-use interface for writing and running tests. Here are some key aspects of Jest that make it a popular choice for unit testing:

Easy Setup: Jest comes pre-packaged with everything you need to get started with testing your JavaScript code. You can install it with a single command, and it comes with all the necessary dependencies.

Built-in Mocking: Jest provides built-in support for mocking and stubbing functions, which can be extremely useful for testing code that has dependencies on external resources.

Snapshot Testing: Jest also provides snapshot testing, which is a feature that allows you to capture a snapshot of your component or function output and compare it to previous snapshots to detect any unexpected changes.

Parallel Test Execution: Jest can execute tests in parallel, which can significantly speed up the testing process for large codebases.

Extensibility: Jest is highly extensible and allows you to customize its behavior and add additional functionality using plugins and custom configurations.

**In summary, Jest is a powerful and easy-to-use JavaScript testing framework that makes it simple to write and run unit tests for your code. Its built-in mocking and snapshot testing features, along with its extensibility and parallel test execution, make it a popular choice for unit testing in the JavaScript ecosystem.**

**13 What is browser side storage?**

Browser side storage, also known as client-side storage, refers to the storage of data within a user's web browser. This data can be stored temporarily or permanently and can be accessed and modified by JavaScript code running on the client-side.

There are two main types of browser side storage: cookies and web storage.

**Cookies:** Cookies are small text files that are created by a website and stored on the user's browser. They are often used to store user preferences or login information, and can be accessed by the website that created them. Cookies can be either persistent or session-based, meaning they can either remain on the user's browser after the session ends or be deleted once the user closes their browser.

**Web storage**: Web storage, also known as HTML5 storage, is a newer form of client-side storage that allows websites to store larger amounts of data than cookies. Web storage comes in two flavors: local storage and session storage. Local storage allows websites to store data on a user's browser permanently, while session storage allows websites to store data temporarily, and it is cleared when the user closes their browser.

Browser side storage is a useful tool for web developers to create more dynamic and personalized experiences for users. However, it is important to use browser side storage responsibly, as too much storage can slow down a website's performance, and users may be concerned about their privacy if too much data is being stored on their browser.

**14.Vaidate the form using HTML?**

Required attribute: You can use the "required" attribute to mark a form field as required, which means the user must fill in that field before submitting the form. The browser will prevent form submission if a required field is left blank.

Pattern attribute: You can use the "pattern" attribute to specify a regular expression that the input value must match. For example, you can use the pattern attribute to validate email addresses or phone numbers.

Input types: HTML5 introduces several new input types that can be used for form validation, such as "email", "url", "number", and "date". These input types provide browser-specific validation to ensure that the user enters the correct type of data.

Min and max attributes: You can use the "min" and "max" attributes to specify the minimum and maximum values for number and date inputs.

**NAME: Required , type**

<input id="name" type="text" required>

**AGE: min, max**

<input id="age" type="number" name="age" required min="18" max="36">

**URL: pattern**

<input id="url" type="url" name="url" required pattern="https?://.+">

**PASSWORD:**

<input id="password" type="password" name="password" required pattern="^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)[a-zA-Z\d]{8,25}$">

**USERNAME: title**

<input id="username" type="text" name="username" required pattern="^[a-z0-9]{3,15}$" title="Password may only contain letters and numbers">

**15.what is W3C validator?**

The W3C validator is an important tool for web developers and designers, as it helps ensure that web documents are accessible, interoperable, and optimized for search engines. By using the validator, web developers can identify and fix errors and warnings in their code, which can improve the quality and performance of their web pages.

**16. What is media queries?**

It helps to create responsive web designs that adapt to the screen size and device characteristics of the user, making your website more accessible and user-friendly.

**@media screen and (max-width: 768px) {**

**}**

**17. Box model?**

The CSS box model is a fundamental concept in CSS that describes how HTML elements are rendered as rectangular boxes on a web page. **Every element on a web page is considered a box, and the box model describes the properties of that box, including its dimensions, padding, borders, and margin.**

The CSS box model consists of the following parts:

**Content**: This is the actual content of the HTML element, such as text, images, or other elements.

**Padding**: This is the space between the content and the border. Padding can be specified using the **padding** property in CSS.

**Border**: This is the edge of the box that separates the content from the padding and margin. Borders can be styled using the **border** property in CSS.

**Margin**: This is the space between the border and the surrounding elements. Margins can be specified using the **margin** property in CSS.

The total width of a box is calculated by adding the content width, padding width, and border width together. The height of the box is calculated in the same way. The margin is then added to the total width and height to determine the final size of the box.

**margin is the space around an element On the other hand, padding is space inside the border of an element**

**17.Position in css?**

## **1.position: static;** HTML elements are positioned static by default.

**elements are not affected by the top, bottom, left, and right properties**

## **2.position: relative;** elementis positioned relative to its normal position.

## **3.position: fixed;** An element is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.

## **4. position: absolute ;**An element is positioned relative to the nearest positioned ancestor .

## **5.position: sticky;** An element is positioned based on the user's scroll position. top:10px

**18.NFR?**

**NFRs are essential in the development and testing process of any software or system to ensure that it meets the necessary quality and performance standards.**

**Usability**: The application should be easy to use and navigate for all users, regardless of their technical expertise or familiarity with the system.

**Performance**: The application should be fast and responsive, with acceptable load times, response times, and processing times.

**Security:** The application should be secure and protect sensitive data and resources from unauthorized access or attacks.

**Availability**: The application should be available and accessible to users at all times, with minimal downtime or disruptions.

**Maintainability**: The application should be easy to maintain, update, and modify over time, with clear documentation and coding standards.

**Scalability:** The application should be able to handle increasing amounts of traffic or usage over time, without degradation in performance or reliability.

**19.Web Accessibility?**

**the practice of designing and developing websites and web applications that can be accessed and used by everyone, including people with disabilities. The goal of web accessibility is to ensure that people with visual, auditory, motor, or cognitive impairments can perceive, understand, navigate, interact with, and contribute to the web content and functionality.**

* **Providing alternative text for images**
* **semantic markup and clear headings to make the content structure**
* **Providing captions and transcripts for audio and video content,**
* **Using color contrast and alternative text for links and buttons to make them more visible and distinguishable to people with color blindness or low vision.**
* **Designing forms and navigation that are easy to use with a keyboard**

**20. A,AA,AAA?**

The A, AA, and AAA conformance levels help to establish a common understanding of what it means to make web content accessible, and provide a way for organizations to measure and report on their accessibility efforts.

A, AA, and AAA are conforimtry levels for web accessibility standards

Level A (or Single-A): This is the lowest level of conformance, which includes the most basic requirements for web accessibility. Meeting the Level A criteria means that some accessibility barriers have been removed, but there may still be significant issues that prevent some users with disabilities from accessing the content.

Level AA (or Double-A): This is the intermediate level of conformance, which includes additional requirements beyond Level A to improve the accessibility of web content. Meeting the Level AA criteria means that most accessibility barriers have been removed, and the content is accessible to a wider range of users with disabilities.

Level AAA (or Triple-A): This is the highest level of conformance, which includes the most rigorous requirements for web accessibility. Meeting the Level AAA criteria means that all significant accessibility barriers have been removed, and the content is highly accessible to almost all users with disabilities.

**21.Progressve WEB APP?**

A Progressive Web App (PWA) is a type of web application that uses modern web technologies to provide users with an app-like experience on the web. PWAs are designed to work on any device or platform, including desktops, smartphones, and tablets, and they can be accessed through a web browser, without the need to download or install an app from an app store.

PWAs use technologies such as Service Workers, Web App Manifests, and HTTPS to provide users with features and capabilities that were previously only available in native mobile apps, such as offline access, push notifications, home screen installation, and access to device hardware like cameras and microphones.

Some of the key features and benefits of PWAs include:

Responsiveness: PWAs are designed to be responsive to different screen sizes and device capabilities, providing a consistent experience across all devices.

Offline access: PWAs can be designed to work offline, using cached data and resources to provide users with a seamless experience even when they are not connected to the internet.

Push notifications: PWAs can send push notifications to users, just like native mobile apps, to keep them engaged and informed about updates and new content.

App-like experience: PWAs can be designed to look and feel like native mobile apps, with features such as full-screen mode, home screen installation, and app-like navigation.

Security: PWAs are required to use HTTPS, which provides secure connections and protects user data from interception and tampering.

Discoverability: PWAs can be discovered and indexed by search engines, just like traditional websites, making them easier to find and access.

PWAs are becoming increasingly popular as a way for businesses and organizations to provide their users with a seamless and engaging experience on the web, without the need for a native mobile app.

**22.Lighthouse?**

Lighthouse is an open-source tool developed by Google that helps developers and site owners improve the quality and performance of their web applications. It provides a suite of audits, diagnostics, and best practices for web development, covering areas such as performance, accessibility, SEO, and more.

**23.SEO?**

SEO (Search Engine Optimization) is the practice of optimizing a website or web content to increase its visibility and ranking in search engine results pages (SERPs). The goal of SEO is to make it easier for search engines like Google, Bing, and Yahoo to find, crawl, and index your website, so that it appears higher in search results for relevant keywords and phrases.

There are several key factors that contribute to SEO, including:

On-page optimization: This involves optimizing individual web pages and content for specific keywords and phrases, by including them in page titles, headings, meta descriptions, and content. It also involves optimizing images, videos, and other multimedia elements on the page.

Off-page optimization: This involves building links to your website from other high-quality, relevant websites. The quality and relevance of the linking sites, as well as the anchor text used in the links, can have a significant impact on your search engine rankings.

23. How is react or framework better than js?

React and other front-end frameworks are built on top of JavaScript and are designed to make it easier and more efficient to build complex web applications. Here are a few reasons why React and other frameworks are often preferred over plain JavaScript:

1. Component-based architecture: React and other frameworks are based on a component-based architecture, where different parts of the user interface are broken down into individual components. This makes it easier to organize and maintain code, and allows for greater reusability and modularity.
2. Improved performance: Frameworks like React use a virtual DOM to optimize updates and minimize the amount of DOM manipulation required. This can result in faster and smoother performance, especially for complex applications.
3. Consistent code: Frameworks like React come with their own syntax and best practices, which can help ensure that code is written consistently and is easier to read and understand.
4. Ecosystem of tools and libraries: React and other frameworks have large and active ecosystems of tools and libraries that can help with everything from state management to testing and debugging. This can save developers a lot of time and effort, and allow them to build more advanced applications.
5. Community support: Because React and other frameworks are widely used, there is a large and active community of developers who contribute to their development and provide support through forums, documentation, and other resources.

While plain JavaScript can be used to build web applications, frameworks like React can provide many benefits that can make development faster, more efficient, and more organized.

24. How browser renders a webpage?

When a web page is loaded in a browser, the browser follows a series of steps to render the page on the screen. Here's a high-level overview of how this process works:

1. Parsing HTML: The browser parses the HTML code of the web page and creates a Document Object Model (DOM) tree that represents the structure of the page.
2. Parsing CSS: The browser parses the CSS code of the web page and creates a CSS Object Model (CSSOM) tree that represents the style rules of the page.
3. Combining the DOM and CSSOM: The browser combines the DOM and CSSOM trees to create a Render Tree, which represents the visual elements of the page and their styles.
4. Layout: The browser calculates the layout of the elements in the Render Tree, determining their size, position, and relationship to one another.
5. Painting: The browser paints the final layout of the elements onto the screen, rendering the page for the user to see.

Throughout this process, the browser may also download additional resources such as images, videos, and scripts. The timing and order of these steps can affect the performance of the page, which is why web developers often use techniques like minification, caching, and lazy loading to optimize the loading and rendering process.

Additionally, modern browsers use techniques like caching and preloading to speed up the rendering process and improve user experience. For example, some browsers will pre-fetch resources that are likely to be needed in the future, or use a "painting" technique called "progressive rendering" to display parts of the page as soon as they become available.

25. How to make a webpage responsive without media queries?

Media queries are a powerful tool for making webpages responsive, but they are not the only way to achieve this. Here are a few ways to make a webpage responsive without using media queries:

1. Use relative units: Instead of using fixed units like pixels or points, use relative units like percentages or ems. This will allow elements to scale based on the size of the viewport.
2. Use flexible layouts: Use CSS flexbox or CSS grid to create flexible layouts that can adapt to different screen sizes.
3. Use fluid images: Use the **max-width** property on images to ensure that they don't exceed the width of their container. This will allow images to scale down as the viewport size decreases.
4. Avoid absolute positioning: Avoid using absolute positioning for elements that need to be responsive, as this can cause them to overlap or become hidden on smaller screens.
5. Prioritize content: Prioritize the most important content on your page so that it appears first on smaller screens. This will ensure that users can access the most important information without having to scroll too much.

While media queries are a powerful tool for creating responsive webpages, these techniques can be used in conjunction with media queries or as a fallback for browsers that do not support them.

**26.** pseudo selectors(:hover) vs elements(::before)

Pseudo-selectors and pseudo-elements are both used in CSS to select and style specific parts of an HTML document, but they serve different purposes.

Pseudo-selectors are used to select and style elements based on their state, such as when the user hovers over or clicks on them. Examples of pseudo-selectors include **:hover**, **:active**, **:focus**, and **:visited**. Pseudo-selectors are denoted with a single colon (**:**) before the selector name.

Pseudo-elements, on the other hand, are used to add content or style specific parts of an element, such as its first letter or the text before or after it. Examples of pseudo-elements include **::before**, **::after**, **::first-letter**, and **::first-line**. Pseudo-elements are denoted with two colons (**::**) before the selector name.

Here are some differences between pseudo-selectors and pseudo-elements:

1. Purpose: Pseudo-selectors are used to select and style elements based on their state, while pseudo-elements are used to add content or style specific parts of an element.
2. Syntax: Pseudo-selectors are denoted with a single colon (**:**) before the selector name, while pseudo-elements are denoted with two colons (**::**) before the selector name.
3. Applicability: Pseudo-selectors can be applied to any HTML element, while some pseudo-elements can only be applied to specific types of elements. For example, **::before** and **::after** can be applied to any element, while **::first-letter** can only be applied to block-level elements.

Overall, both pseudo-selectors and pseudo-elements are useful tools for styling HTML documents in CSS, but they serve different purposes and have different syntax and applicability.

**27.** display none vs opacity 0 vs visibility: hidden?

**display: none**, **opacity: 0**, and **visibility: hidden** are all CSS properties used to hide elements on a webpage, but they work in different ways.

1. **display: none** removes the element from the page completely. The element is not rendered and does not take up any space on the page. This means that any child elements or content within the element will also be removed from the page.
2. **opacity: 0** makes the element transparent, but it still takes up space on the page. The element and its contents are still rendered on the page, but they cannot be seen. This means that other elements on the page may be visible through the transparent element.
3. **visibility: hidden** hides the element, but it still takes up space on the page. The element and its contents are still rendered on the page, but they cannot be seen. However, unlike **opacity: 0**, the hidden element still affects the layout of the page and may affect the positioning of other elements.

Which of these properties to use depends on the specific use case. If you want to completely remove an element from the page, such as in a dropdown menu that should only appear when clicked, use **display: none**. If you want to keep the element on the page but make it invisible, such as in an animation that should fade in and out, use **opacity: 0**. If you want to hide an element but still have it affect the layout of the page, such as in a form where certain fields should only appear based on user input, use **visibility: hidden**.

**28** inline vs inline block - in inline, you can’t set height, width and top bottom padding?

Inline and inline-block are two display values in CSS that are used to control the layout of HTML elements. Inline elements are displayed inline with the text, while inline-block elements are displayed inline but can have a width, height, padding, and margin applied to them.

In an inline layout, elements are positioned on the same line as the text, with no line breaks between them. Inline elements do not have a width, height, margin, or padding applied by default, and their size is determined by their content.

On the other hand, inline-block elements can have a width, height, margin, and padding applied to them, just like block-level elements. This means that inline-block elements can be used to create more complex layouts than inline elements.

It's worth noting that while you can set a height, width, and padding on an inline-block element, you cannot set a top or bottom padding. This is because inline-block elements are still part of the text flow, and adding top or bottom padding would affect the spacing between lines of text.

In general, inline elements are best used for small, simple elements like links or buttons, while inline-block elements are better suited for more complex elements like images or blocks of text that need to be styled with a width, height, margin, or padding.

**30.** Above the fold vs below the fold?

"Above the fold" and "below the fold" are terms used in web design to describe the portion of a webpage that is visible without scrolling.

"Above the fold" refers to the portion of a webpage that is visible to a user when they first land on the page, without having to scroll down. Historically, this term came from the world of print journalism, where newspapers were folded in half and the top half of the front page was the most important area for capturing a reader's attention.

In web design, "above the fold" content typically includes the most important information and calls-to-action, such as the page title, headline, and primary navigation. This content should be designed to immediately capture a user's attention and encourage them to explore the rest of the page.

"Below the fold" refers to the portion of a webpage that is not visible when a user first lands on the page and requires scrolling to view. While "above the fold" content is important for capturing a user's attention, "below the fold" content is also important for providing additional information and details, such as product features or additional navigation options.

It's important for web designers to consider both "above the fold" and "below the fold" content when designing webpages. By designing a clear and effective layout that balances both types of content, designers can create webpages that are both engaging and informative for users.

**31.** SASS/SCSS benefits over css?

SASS (Syntactically Awesome Style Sheets) and its newer syntax SCSS (Sassy CSS) are extensions of CSS that offer several benefits over traditional CSS:

1. Variables: SASS/SCSS allows you to define and use variables to store values such as colors, font sizes, and margins. This makes it easy to maintain consistency across your project and quickly update values throughout your code.
2. Nesting: SASS/SCSS allows you to nest selectors inside one another, making it easier to read and organize your CSS code. This can also help to reduce the amount of repetition in your code.
3. Mixins: SASS/SCSS allows you to define and reuse code snippets called mixins. Mixins can be used to encapsulate common CSS patterns such as vendor prefixes or complex animations.
4. Inheritance: SASS/SCSS supports inheritance, which allows you to define a set of styles that can be inherited by other elements. This can help to reduce the amount of repetition in your code and make it easier to maintain.
5. Modularity: SASS/SCSS supports modularity, which allows you to break your CSS code into separate files and import them as needed. This can help to keep your code organized and make it easier to manage.

Overall, SASS/SCSS can help you write cleaner, more organized, and more efficient CSS code. By using these features, you can reduce the amount of repetition in your code, make it easier to read and maintain, and save time and effort in the long run.

**32.** CSS Specificity?

CSS specificity refers to the set of rules that determine which CSS styles are applied to an element when there are conflicting styles. When multiple CSS rules target the same element and conflict with each other, the browser needs to determine which rule takes precedence.

The specificity of a CSS rule is calculated based on the selectors used in the rule. In general, the more specific a selector is, the higher its specificity. The specificity is determined by counting the number of ID selectors, class selectors, and element selectors in the selector chain.

For example, consider the following CSS rules:

**#header {**

**background-color: blue;**

**}**

**.header {**

**background-color: red;**

**}**

**h1 {**

**background-color: green;**

**}**

If we have an element with the class "header" and an **id** of "header", the background color will be blue because the **id** selector is more specific than the class selector.

The specificity of a rule is important to understand when designing a website or application, as it can help to avoid unexpected styling conflicts and ensure that styles are applied consistently. It is also important to be aware of the different types of selectors and how they affect specificity in order to write efficient and effective CSS code.