Frontend development Interview Questions

by,

@maybeshalinii

Q) How does the CSS position property work?

The CSS position property is used to control the positioning of an element within its containing element.

There are primarily 4 types:
- Static
- Relative
- Absolute
- Fixed

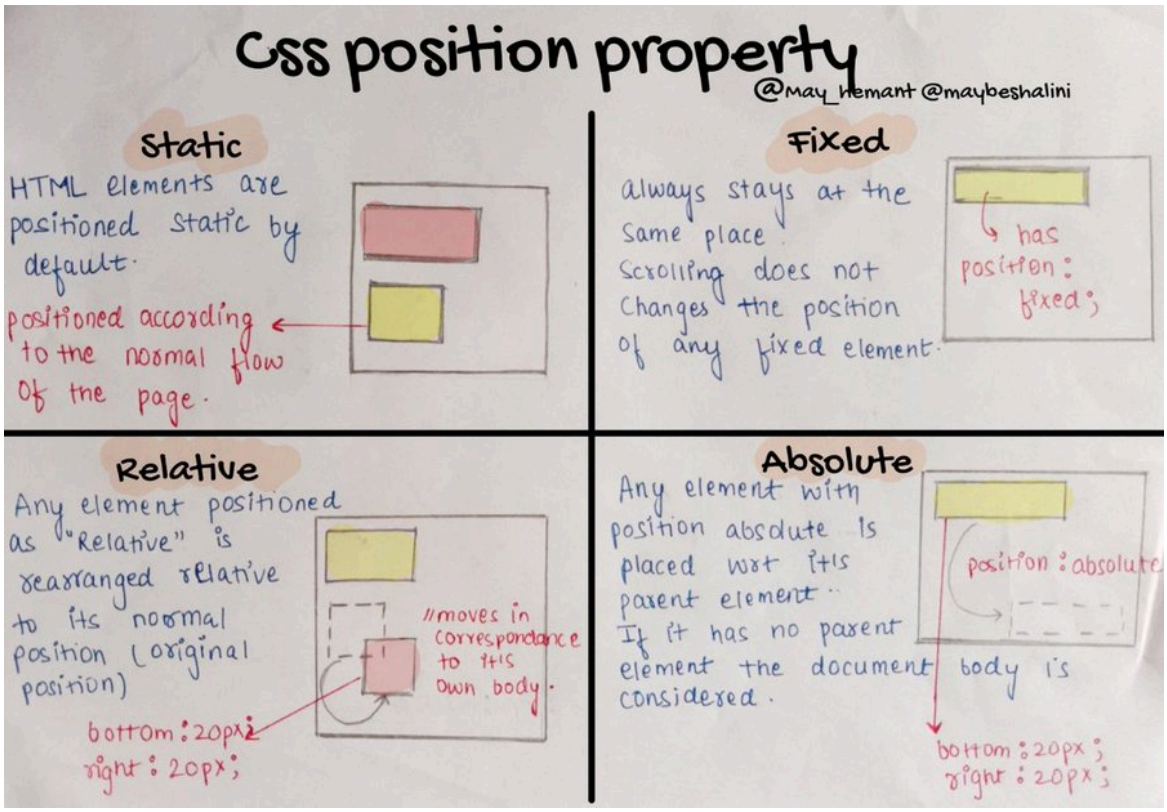
Static
This is the default positioning. Elements are positioned according to the normal flow of the document.
The top, right, bottom, and left properties have no effect.

Relative:
Elements are positioned relative to their normal position in the document flow.
The element stays in the normal flow but can be nudged around from its original position using top, right, bottom, or left.

Absolute:

Elements are removed from the normal document flow and positioned relative to the nearest positioned ancestor. If no positioned ancestor is found, it is positioned relative to the initial containing block (usually the viewport).
It won't affect other elements' positions, and you can use top, right, bottom, or left to precisely place it.

Fixed:
The element is removed from the normal document flow and positioned relative to the initial containing block.
It remains fixed at its position even when the page is scrolled. Handy for things like fixed navigation bars.



Q) How do you declare variables in JavaScript, and what's the difference between let and const?

In JavaScript, we can declare variables using three keywords:
- var
- let
- const

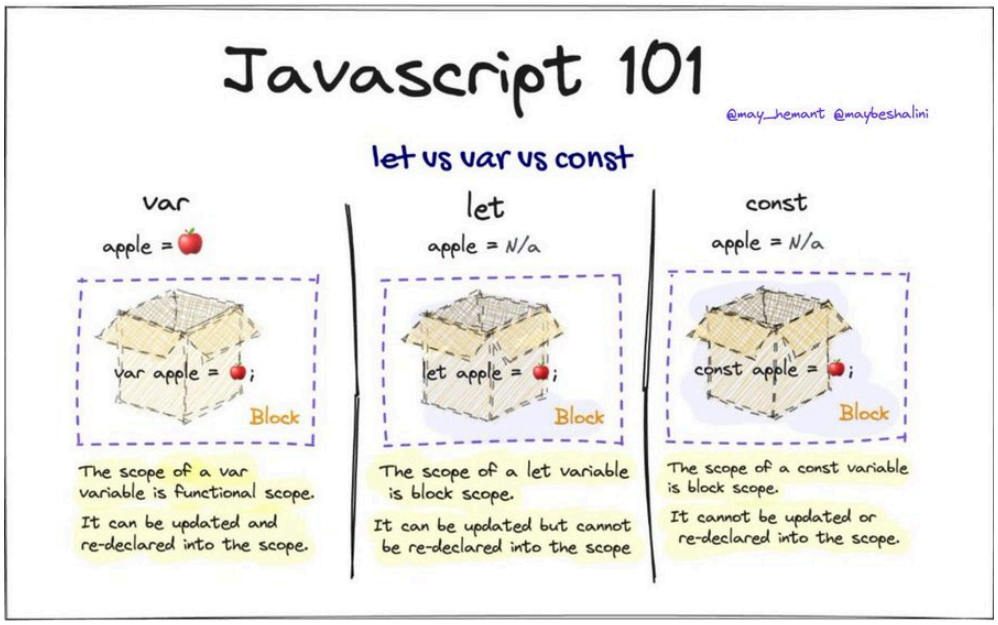
var:
- var is the oldest way to declare variables in JavaScript.
- Variables declared with var are function-scoped, which means they are only accessible within the function where they are declared.
- They can also be accessed outside the function, but their value will be undefined until the point in the code where they are declared.

let:
- let was introduced in ECMAScript 6 (ES6) and provides a block-scoping mechanism.
- Variables declared with let are limited in scope to the block, statement, or expression where they are defined.
- let allows you to reassign values to the variable.

const:

- Like let, const was also introduced in ES6.
- Variables declared with const are block-scoped, just like let.
- The key difference is that the value of a const variable cannot be reassigned once it has been assigned. It is a constant.

Difference between let and const:
- let allows you to reassign values to the variable, whereas const does not. Once a value is assigned to a const variable, it cannot be changed.
- It's important to note that while the value of a const variable is immutable, the variable itself is not. This means that if a const variable holds an object or array, the properties or elements of that object or array can still be modified.



Q) What is the difference between synchronous and asynchronous JavaScript?

Synchronous and asynchronous in JavaScript refer to how code is executed and how tasks are handled in relation to the event loop.

- Synchronous JavaScript:
• In synchronous code execution, tasks are performed one at a time, in sequence.

• Each task must complete before the next one starts, blocking the execution of code until the current task is finished.

• It follows a predictable and straightforward flow, making it easier to understand and debug.

Asynchronous JavaScript:
• Asynchronous code allows tasks to be executed concurrently without waiting for the completion of each task.

• It doesn't block the execution of code; instead, it uses mechanisms like callbacks, promises, or async/await to handle tasks in the background.

• Asynchronous operations are often used for tasks that may take some time to complete, such as fetching data from a server or reading a file.

Q) FlexBox vs Grid Layout (in CSS) What would you use and why?

Both Flexbox and CSS Grid are powerful layout systems in CSS, but they have different use cases and are designed to solve different layout challenges.
My choice between Flexbox and Grid will depend on the specific layout requirements project.

About flexbox:
Flexbox is primarily designed for one-dimensional layouts, either in a row or a column. It's great for arranging items within a container along a single axis, which makes it suitable for scenarios like navigation menus, lists and simple card layouts.

When To use FlexBox:
1. Simple Row/Column Layouts:

If your layout requires arranging items in a single row or column.

2. Alignment and Centering

3. Unequal Heights or Widths:
When dealing with items of varying heights or widths, Flexbox can help align them neatly.

About Grid:
CSS Grid Layout is a two-dimensional system designed for creating complex grid-based layouts. It allows us to define both rows and columns and place items at precise locations within this grid. It's particularly useful for creating complex designs.

When to Use CSS Grid:
1. Complex Layouts:
If your layout requires a multi-dimensional grid structure, such as magazine layouts or intricate card designs.

2. Multi-Column and Multi-Row Forms:
When designing forms with multiple columns or rows.

3. Responsive Designs with Complex Requirements: For layouts that need to change significantly across different screen sizes, CSS Grid can handle the complexity better than Flexbox.

Q) What is useState() in React?

useState() is a React Hook that allows functional components in React to manage and update their own local state. It's like a special function that gives you a state variable and a function to update that variable.

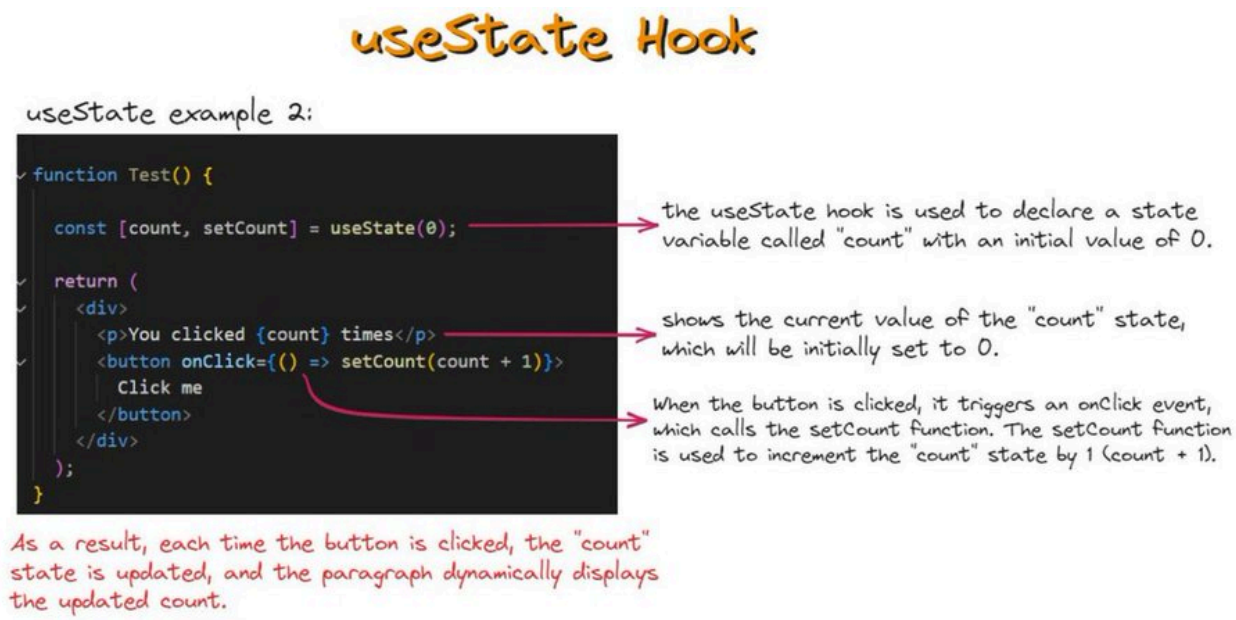
It helps functional components remember things and re-render when those things change. For example, you might use useState() to keep track of a number of likes on a post, and when the user clicks a "like" button, it updates that number and causes the component to re-render, reflecting the new state. It's a way for React components to have their own memory, making them more dynamic and interactive.

Syntax:
const [state, setState] = useState(initialState);

state: This is the current state value, similar to this.state in class components.

setState: This is a function that allows you to update the state. When setState is called, React will re-render the component, and the new state will be reflected.

initialState: This is the initial value of the state. It can be a primitive value (e.g., number, string) or an object.



Q) Explain React Lifecycle Methods vs. Effects

React Lifecycle Methods:

Imagine a React component as a machine with different parts that move and work together. In the past, React used something called "lifecycle methods" to control how this machine behaved. These methods were like levers you could pull or buttons you could push at specific times during the machine's life.

For example, there was a method called componentDidMount that would be triggered when the machine was "born," or when it was first put on the web page. You could use this method to do things like load data from a server. Another method, componentDidUpdate, was like a button you could press whenever something about the machine changed. You could use it to respond to changes in the machine's state or props.

useEffect Hook:

Now, React has a new tool called the useEffect hook. It's like a more versatile and flexible way of controlling the machine's behavior. Instead of having different levers and buttons for different situations, you have one tool that can do it all.

With useEffect, you can say, "Hey, React, I want to do something when certain conditions are met." You tell React what conditions to watch, like when a specific piece of data changes or when the component is first created. For example, you can use useEffect to say, "When this component is born (componentDidMount), do this thing." Or, "Whenever this piece of data changes (componentDidUpdate), do this other thing."

To make it easier:
Lifecycle methods are like separate buttons and levers for controlling your component's behavior at specific moments in its life. useEffect is like a more flexible tool that you can use to specify when and how you want to do something with your component, whether it's when it's born, when it changes, or in other situations.
In modern React development, the useEffect hook is generally preferred because it offers more control and simplifies how you manage side effects (like data fetching, subscriptions, or DOM manipulation) in your components.

Q) What is responsive web design, and how does it differ from mobile-first design?

Responsive web design is an approach to web development that aims to ensure a seamless and consistent user experience across various devices and screen sizes. It involves designing and coding a website so that it automatically adapts and responds to different viewport sizes, such as those of desktops, tablets, and smartphones. Responsive web design uses flexible grids, media queries, and flexible images to achieve this adaptability.

On the other hand, mobile-first design is a specific strategy within responsive design. It involves initially designing and developing a website for mobile devices before progressively enhancing it for larger screens. The idea is to prioritize the mobile experience, ensuring that the site is optimized for smaller screens, and then scaling up the design and functionality for larger devices.

While responsive design focuses on creating a flexible and adaptive layout for various devices, mobile-first design specifically starts with the mobile user experience in mind, often resulting in a more streamlined and efficient design.

Both approaches contribute to a better overall user experience, but they differ in their initial design priorities.

Conclude it like:
Responsive web design is like making a website that's smart and can adjust itself to look good on any device, like a computer, tablet, or phone.
Mobile-first design is a special way of doing this. It means you first design the website to work really well on small screens, like phones, and then make it bigger for computers.
So, responsiveness is about making websites flexible for all devices, and mobile-first is a cool strategy to start by making it awesome for phones.

Q) What is the difference between React & React Native?

React (React.js):

- Used for building websites and web applications. It uses regular web components like and <input>.

- Styling is typically done with CSS. Development happens in web browsers with web development tools. Apps are deployed to web servers and accessed through web browsers.

React Native:

- Used for creating mobile apps for iOS and Android. It uses mobile-specific components like <View> and <Text>.

- Styling is done using JavaScript-like styles. Development requires mobile development tools, simulators, or physical devices. Apps are distributed through app stores (Apple App Store, Google Play).

Code Reusability:
- In React, code isn't directly reusable in React Native. - In React Native, you can share some code and components between iOS and Android apps.

Performance:
- React apps perform well in web browsers, using the virtual DOM. React Native apps also perform well due to native components, but complex animations might require native code.

Q) How do you handle errors in javascript?

Handling errors in JavaScript is crucial for writing robust and reliable code. There are several mechanisms to deal with errors in JavaScript:

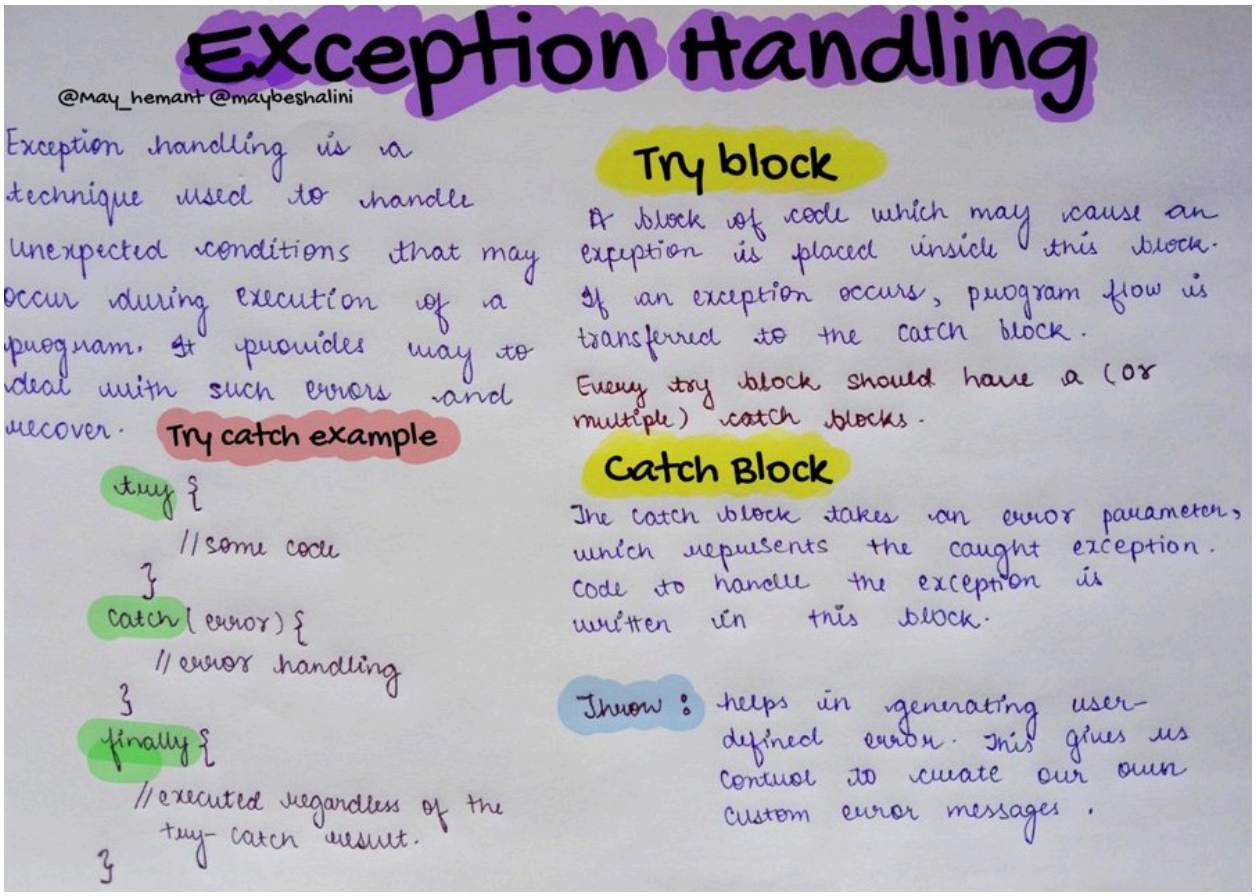
- Global Error Handling:
• Use window.onerror or window.addEventListener('error') to capture unhandled errors globally.

- Try-Catch Blocks:
• JavaScript has a built-in mechanism for handling errors using try, catch, and finally blocks.
• We can wrap the potentially error-prone code in a try block, and if an exception occurs, it is caught and handled in the corresponding catch block.

- Throwing Custom Errors:
• We can throw custom errors using the throw statement, allowing you to create meaningful error messages and handle them accordingly.

-Async/Await Error Handling:

• When working with asynchronous code and async/await, use try-catch blocks to handle errors in a synchronous manner.



Q) How is React Router different from Conventional Routing?

React Router:

React Router is a powerful navigation library tailored for single-page applications (SPAs) in React.
Employing a declarative and component-based approach, React Router seamlessly integrates with the React ecosystem.
Routes are defined as components, facilitating dynamic navigation and enabling developers to efficiently manage UI states.
With client-side navigation, React Router ensures a smooth and responsive user experience by updating views without requiring server requests.

Conventional Routing:
Conventional routing, often associated with multi-page applications (MPAs), relies on an imperative model.
Navigation is typically tightly coupled with templates or files, and changes may involve manual updates across multiple pages.
Unlike React Router's component-based structure, conventional routing may lack the flexibility and ease of managing dynamic UI states.
Additionally, client-side navigation is less common, resulting in full page reloads and potentially slower performance due to server requests for each navigation event.

Q) What is JSX in React, and how is it different from HTML?

First of all, let's understand, What is JSX?
JSX is a way to write code in React that looks a lot like HTML. It lets you describe how things should look in your web app. Even though it looks like HTML, it's actually a special kind of JavaScript.

How is it different from HTML?

- *Mixing with JavaScript:*
In JSX, you can easily include pieces of JavaScript code right alongside your HTML-like code. This helps to make your web app more dynamic.

- *Attribute Names:*
Some words that HTML uses for special things (like class for defining styles) need to be a bit different in JSX. For example, class becomes className.

- *Styles:*
If you want to add styles, in JSX, you use a JavaScript object instead of a regular CSS string.

Conclude it by saying, JSX is like a more powerful version of HTML that plays really well with JavaScript, especially in React apps. It helps make your web pages interactive and dynamic.

Q) Discuss the advantages and disadvantages of using CSS frameworks.

Advantages:
- Faster Building Blocks: CSS frameworks provide pre-made building blocks (like buttons, forms, and layouts) that save time. Instead of starting from scratch, developers can use these ready-to-go pieces.

- Consistent Look: Frameworks help ensure that your website looks consistent. They provide a set of rules and styles, so every part of your site has a similar appearance, making it look more professional.

- Works on Different Devices: Many frameworks make it easier to build websites that look good on both big computer screens and small phone screens. This is crucial because people use all sorts of devices to browse the internet.

- Less Browser Headache: Frameworks are tested on different web browsers, so you're less likely to run into problems where your site looks great in one browser but terrible in another.

Disadvantages:

- Takes Time to Learn: Learning how to use a framework can take some time. If you're working on a small project, the time spent learning might not be worth it.

- Can Make Your Site Slower: Frameworks come with a lot of code, and sometimes you might end up with more code than you actually need. This can slow down your website's loading time.

- Less Freedom to be Unique: Using a framework might limit your ability to make your website look exactly the way you want. Your site might end up looking like many other sites that also use the same framework.

- Dependent on Updates: If the framework gets updated, it could change how things work. This means you might need to spend time fixing your website when you update the framework.

- Might Not Fit Every Project: Some frameworks have a specific style or way of doing things. If your project doesn't fit well with the framework's style, it might be more trouble than it's worth.

Q) Discuss the purpose of the Document Object Model (DOM) in web development.

The Document Object Model (DOM) in web development serves as a programming interface for web documents. It represents the structure of a document as a tree of objects, where each object corresponds to a part of the document, such as elements, attributes, and text.

It is like a digital blueprint that web browsers use to understand and organize the different parts of a webpage. It creates a tree-like structure where each branch represents an element, like paragraphs or images. Web developers use the DOM to change what's on a webpage without refreshing the whole thing. For instance, it helps in updating scores in a game, loading new posts on social media, or showing and hiding elements when you click a button. In simpler terms, the DOM is the web developer's tool to make websites interactive and dynamic, allowing them to respond to your actions in real-time.

The DOM plays a crucial role in web development for several key purposes:
- Dynamic Content Manipulation
- Responsive User Interfaces
- Event Handling
- AJAX (Asynchronous JavaScript and XML) Requests
- Cross-Browser Compatibility
- Structural Representation
- Document Navigation and Modification
- Integration with Other Web Technologies

Q) Describe the concept of scope in JavaScript.

In JavaScript, scope is like a set of rules that determines where in your code you can use or change a variable.
There are two main types of scope:

- Global Scope:
Variables declared outside of any function or block have a global scope. They can be accessed from any part of the code, including inside functions.
Imagine you have a toy that you can play with anywhere in your house. That toy is like a variable in the global scope, you can use it anywhere in your code.

- Local Scope:
Variables declared inside a function or block have a local scope. They are only accessible within that specific function or block.
Now, think of a toy that you can only play with inside your house. This toy is like a variable in a local scope you can only use it in a specific part of your code, like inside a function or a specific area.

Scope Chain:
When you want to use a variable, JavaScript checks if it's available in the current place you're working. If not, it looks in the outer areas, moving up like climbing stairs until it finds the variable or reaches the top level (global scope).

Block Scope (ES6 and later):
In newer JavaScript, some variables stay limited to the area where they are created. It's like having a special toy that only works within a certain playroom and doesn't affect the rest of the house.
Understanding scope helps prevent confusion and ensures that your variables work where you expect them to in your code.

Q) Explain the concept of event delegation in JavaScript.

Event delegation is a programming pattern in JavaScript where a single event listener is used to manage all occurrences of a particular event type for multiple elements, usually within a common parent container.

How it works:
Instead of attaching an event listener to each individual child element, you attach a single event listener to a common

ancestor (typically a parent element) that contains all the child elements.

Advantages:

- Efficiency:
It reduces the number of event listeners, making the code more efficient, especially in scenarios with a large number of elements.

- Dynamic Elements:
It allows for handling events on dynamically added elements without needing to attach new listeners each time.

Event delegation is particularly beneficial in scenarios where the number of elements or dynamic changes to the DOM structure can make attaching individual event listeners impractical.

Q) Discuss the concept of the CSS box model and its components. How does it impact the layout of elements on a webpage?

The CSS box model is a fundamental concept that describes the layout of elements on a webpage. It consists of four main components:

1. Content:
- Represents the actual content of the box, such as text, images, or other media.
- Its size is determined by properties like width and height.

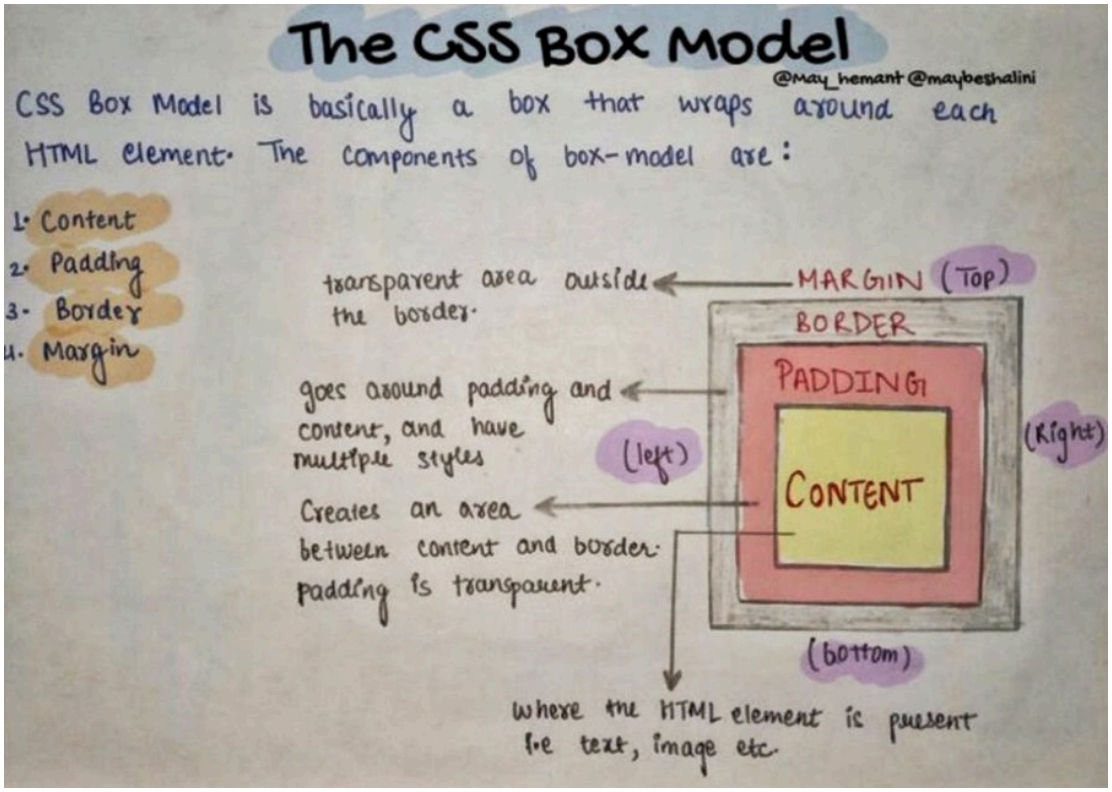
2. Padding:
- A transparent area surrounding the content, inside the border.
- It adds space between the content and the border.
- Controlled by properties like padding-top, padding-right, padding-bottom, and padding-left.

3. Border:
- A border surrounding the padding (and content) of the box.

- It's defined by properties like border-width, border-style, and border-color.

4. Margin:
- A transparent area outside the border, providing space between this box and its neighboring elements.
- Controlled by properties like margin-top, margin-right, margin-bottom, and margin-left.

Impact on Layout:
- The total width or height of an element is calculated as follows: width/height + padding + border.
- The margin adds spacing between elements, preventing them from being too close to each other.



Q) What is the difference between null and undefined?

In JavaScript, null and undefined are both special values that represent the absence of a meaningful value, but they are used in slightly different contexts.

- undefined: When a variable is declared but not initialized, it automatically gets assigned the value undefined. If you try to access an object property that doesn't exist, you get undefined.

- null: It is a value that represents the intentional absence of any object value. You can explicitly assign a variable or

object property the value null to indicate no value or no object. It is often used to indicate that a variable or object property should have no value or that the value is unknown or irrelevant. Conclusion is, undefined is typically a default value that indicates the absence of an assigned value, while null is a value that needs to be explicitly assigned to indicate a deliberate absence of a meaningful value.