CSS

1. *What is the difference between inline and block elements in css?*

   Inline:-

   - The element does not start on a new line and only occupies just the width it requires. You can't set the width or height. Some examples are <span>, <strong>, <img>.
   - Respect left & right margins and padding, but not top & bottom.
   - Cannot have a width and height set
   - Allow other elements to sit to their left and right
   - An inline element has no line break before or after it, and it tolerates HTML elements next to it.

   Inline-Block:-

   - It's formatted just like the inline element, where it doesn't start on a new line. But you can set width and height values.
   - Allows other elements to sit to their left and right.
   - Respect top & bottom margins and padding
   - It respects height and width.
   - An inline element is placed as an inline element(on the same line as adjacent content), but it behaves as a block element

   Block:-

   - The element will start on a new line and occupy the full width available. And you can set width and height values.
   - It fills up the horizontal space left and right on the web page. Some of the block elements are <div> and <p> tags.
   - It acquires full-width if width is not defined.
   - A block element has some whitespace above and below it and does not tolerate any HTML elements next to it.

2. *What is the difference between margin and padding in css?*

   Padding:-

   - Padding is the space between the content and the border of an element. Padding is valuable in making additional space inside the element, keeping it at a set distance from other aspects of a website.
   - The inner space an element has around itself
   - Prevents text from appearing too close to a border.
   - It is not possible to set auto padding.
   - It is not possible to use negative values.
   - Can be impacted by the styling of other elements.

   Margin:-

- Margin is the space around the border of an element. The margin surrounding an element will inform the web browser being used of how much space should be left between independent elements and the external margin of the website's page. Margins can also be used to keep different elements an equal distance apart.
- The whitespace available surrounding an element.
- It's possible to use an auto setting for margins.
- It is possible to use negative values.
- Is not impacted by the styling of other elements.

<u>When to use padding:-</u>

Change the size of an element:- If you want to expand the space around an element, you can add or increase the padding surrounding it. This can be useful when working with interactive items, such as buttons or image-based links.

Add space between borders and content:- Using padding to add space between content and its corresponding border is one way to ensure the design aligns with other on-page elements. Doing so can help you increase the whitespace of your graphic or website

<u>When to use margins:-</u>

Adjust an element's positioning:- One of the most common reasons to use margins when designing and developing a site to change a specific element's position. Using margins can help you move an element based on whether you prefer it to be centered on your page or positioned to the right or left. You can also choose if the element is fixed and will scroll along the page or if it should remain in one place as a user scrolls.

Overlap elements:- If you want to overlap specific elements with one another, you can do so using margins. Using a negative margin value is one of the quickest ways to allow elements to overlap with one another.

Setting distance:- Setting the distance between elements is much easier once you're familiar with margins and how they work. Incorporating the right amount of white space can mean the difference between building an attractive website thriving with traffic and turning prospective customers away.

4. *What is the difference between absolute and relative positioning in CSS?*

CSS positioning has always been an essential part of creating website layouts. Even now when there are advanced techniques such as Flexbox and CSS Grid, positioning still has an important place in any web designer's bag of tricks.
Here are the types of CSS positioning properties.

- position:static
- position:fixed
- position:sticky
- position:relative
- Position:absolute

Static Position:-

- HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left and right properties.
- An element with position:static is not positioned in any special way, it is always positioned according to the normal flow of page

Relative Position:-

- Function:- Relative positioning moves an element to either left, right, top or bottom area with a document.
- When to use:- Relative positioning is used when the inner element of a specific page will be positioned absolutely.
- Setting the top, right, bottom and left properties of a relatively-positioned element will cause it to be adjusted away from normal position. Other content will not be adjusted to fit into any gap left by the element.

Position Fixed:-

- An element with *position:fixed* is positioned relative to the viewport, which means it is always 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.

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.

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.

5. *What is a pseudo-class in CSS?*
   - A css pseudo-class is a selector which specifies a specific state of the selected element. Pseudo classes are elements that are in a specific state. Pseudo classes are keywords that start with a colon.
   - CSS Pseudo classes are keywords for the selectors that set attributes when the selectors are in an extra special state. Basically, pseudo classes set the style of an element when the element is in a particular special state.

6. *How did you include CSS in an HTML document?*

CSS can be added to HTML documents in 3 ways.

- Inline - by using the style attribute inside HTML elements.
- Internal - by using a <style> element in the <head> section
- External - by using a <link> element to link an external CSS file.

7. *What is the difference between ID and class selectors in CSS?*

ID:-

- A selector in CSS that styles the element with a specified id
- Syntax is #id {css declarations}
- Used to apply styling to one specific element

Class:-

- A selector in CSS that styles the selected elements with a specified class.
- Syntax is .class
- Used to apply styling to multiple elements

8. *What is the difference between nth-child and nth-of type selectors in css?*

nth-child() Selector

- This selector is used to style only those elements that are in the nth number of child of its parent element
- It is used to select all elements that are the nth child.
- Its syntax is :*nth-child(n | even | odd | formula)*
- It does not consider type of the parent while selecting the elements.

nth-of-type() Selector

- This selector is used to style only those elements that are the nth number of child of its parent element.
- It is used to select all elements that are the nth child.
- It is syntax is :*nth-of-type(n | even | odd | formula)*
- It only consider a particular type of the parent

9. *What is the difference between a child and descendant selector in css?*

Child Selector:- Child Selector is used to match all the elements which are children of a specified element. It gives the relation between two elements. The element > element selector selects those elements which are the children of the specific parent. The operand on the left side of > is the parent and the operand on the right is the children element.
Syntax

```
element  > element
{
    //CSS property
}
```

Descendant Selector:- Descendant selector is used to select all the elements which are child of the element(not a specific element). It selects the elements inside elements i.e, it combines two selectors such that elements matched by the second selector are selected if they have an ancestor element matching the first selector.
Syntax

```
element > element
{
   //CSS property
}
```

10. *What is CSS specificity?*

CSS specificity is the set of rules applied to CSS selectors in order to determine which style is applied to an element. The more specific a CSS style is, the higher point value it accrues, and the likelier it is to be present on the element's style.

In CSS, specificity is a measurement of relevance based on the type and order of CSS selectors in a document. In cases when an HTML element or a group of elements is targeted by multiple CSS selectors, the rules of CSS specificity tell the web browser which CSS declarations should be applied.

At high level, there are three buckets of specificity your css will fall under

- Type selectors & pseudo-elements
- Class selectors, attribute selectors and pseudo classes
- ID selectors

11. *How do you clear floats in css?*

The float property in CSS is used to change the normal flow of an element. The float property defines where an element should be positioned (right or left) in a container.
We clear the float property to control the floating elements by preventing overlapping.

Techniques to clear float:-

- *Clear Property :-* The clear property is used to specify which side of floating elements are not allowed to float. It sets the position of the element concerning floating objects. The element can fit horizontally in the space next to another element that is floated. We have to apply the clear property to that element in the same direction as the float so that the element will move down below the floated element.

- ***clear:*** none | left | right | both | initial

none :- It is the default value of the clear property. After using this value the element will not be pushed left or right floated elements.
right :- This value pushes the element right below floated elements.
left :- This value pushes the element left below the floated elements.
both :- This value pushes the element left and right to the floated elements.
initial :- Change the properties to their default value.
inherit :- Inherit the floating properties to their parent element.

- ***The overflow method:-*** A clearfix is a way for an element to automatically clear or fix its elements so that it does not need to add additional markup. It is generally used in float layouts where elements are floated to be stacked horizontally. If the element is taller than the element containing it the use the overflow property of CSS by setting its value as *auto* to overcome and fix the problem. (*overflow:auto*)

12. *What is the difference between em and rem units in CSS?*

em:
- em units for the font-size property will be relative to the font-size of the parent element.
- em units on other properties than font-size will be relative to the font-size of the current element.
- Use em for media queries
- May lead to a compounding effect.

rem:-
- Rem unit sizes will always be relative to the font-size of the root *html* element.
- Use rems for sizes and spacing
- Does not lead to a compounding effect.

13. *What are media queries in CSS and how do you use them?*

A media query in a CSS3 feature that makes a webpage adapt its layout to different screen sizes. We have three types of media files.

- Screen
- Speech
- Print

@media only screen (min-width: 1281px) { //css properties }
The width breakpoints of almost all devices are given below.
- 320px - 480px: Mobile devices
- 481px - 768px: iPads, Tablets
- 769px - 1024px: Small screen laptops, ipad pros
- 1025px - 1280px: Desktops, laptops
- 1281px and more - Imacs, Big desktop monitor

➢ Default orientation is always portrait and we need not specify it
➢ Desktops do not have landscape orientation.

14. *How do you create a responsive design using CSS?*

A web page design that adapts to different screen sizes is called responsive.

Media Queries :-
- Media query is provided by CSS to achieve the concept of responsiveness. This is a way to conditionally apply CSS rules.
- @media rule indicates the media query using which different styles can be applied based on media types, screen sizes and orientation.

ViewPort :-
- The viewport is the actual area in the rendering surface where the web page is displayed. The width or height constraint specified in the media query refers to the viewport width or height usually on the browser.
- To make an HTML page to be responsive, the viewport meta tag has to be included.
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

15. *How do you create a gradient background in CSS?*

- CSS gradients let you display smooth transitions between two or more specified colors.
- CSS defines three types of gradients.
  - Linear Gradients (goes down/up/left/right/diagonally)
  - Radial Gradients (defined by their center)
  - Conic Gradients (rotated around a center point)

background-image: linear-gradient(direction, color1, color2, ...);

<u>Parameters:</u>

<u>*direction :*</u> Specify the direction of the transition. The default value is 180 deg (if not specified)

<u>*color 1 :*</u> Specify the first color

<u>*color 2 :*</u> Specify the second color

16. *What is the difference between display:none and visibility:hidden in CSS?*

<u>*display:none*</u>

- turns off the layout of the elements, so they are not rendered
- removes the element from the document, It does not take up any space

<u>*visibility:hidden*</u>

- Hides the elements without changing their layouts
- Hidel the element but it still takes up space in the layout

<u>*opacity:0*</u>

- Causes the elements to be very transparent but users can still interact with them.

<u>*Options for display:*</u>

- Block
- Inline
- Flex
- None

Options for visibility:

- Visible
- Hidden
- Collapse

17. What is the difference between pseudo-elements and pseudo-classes in CSS?

The English definition of the word pseudo is “fake” or “not real”. Pseudo classes and Pseudo elements are not manually written into HTML and don’t appear in the DOM but instead are created by CSS.

Pseudo-Classes

Pseudo classes allow you to select elements in CSS based on information outside of the HTML written on the page, such as user interaction or information stored in the browser. Pseudo-classes are accessed by a single colon(:) followed by the pseudo-class name.

You can use pseudo-classes to style an element based on its state. You might often see links you’ve visited on a page appear as a different color. This is achieved by targeting the :visited pseudo-class of an anchor tag (a) in CSS to style it. *a:visited {color:#c58af9}*

Pseudo-Element:-

Pseudo-element selectors allow you to style a specific part of a DOM element. Pseudo elements are accessed by a double colon(::) followed by the pseudo-element selector. Unlike pseudo-classes, pseudo-elements cannot be used to style an element according to state. *p::first-letter{font-size: 300%}*

18. How do you create a tooltip in css?

https://dev.to/isarisariver/how-to-create-a-custom-tooltip-with-pure-css-4a29

https://medium.com/front-end-weekly/creating-an-awesome-tooltip-in-css-9741407899d4

19. How do you center an element horizontally and vertically in CSS?

- Using flex
- Using grid
- Using CSS positioning and CSS transform
- Using CSS margin
- Using CSS Positioning
- Bonus method - Using CSS table

  https://dev.to/ruphaa/5-ways-to-center-a-div-vertically-and-horizontally-using-css-3i60

  https://www.geeksforgeeks.org/how-to-center-an-element-horizontally-and-vertically/

19. How do you position an element to the right or left of its container in CSS?

https://medium.com/12-developer-labors/css-all-the-ways-to-align-elements-left-and-right-52ecce4a4af9

20. How do you create a dropdown menu in CSS?

https://dev.to/monalishamondol/how-to-create-drop-down-menu-in-html-and-css-3e14

https://fireship.io/snippets/basic-css-dropdown-menu/

21. How do you create a slideshow using css?

https://dev.to/code_mystery/automatic-image-slideshow-using-html-css-3jg7

https://www.geeksforgeeks.org/programming-a-slideshow-with-html-and-css/

22. What is the difference between the :hover and :active pseudo-classes in css?

:hover: is CSS pseudo-class and it matches when the user interacts with an element with a pointing device, but does not necessarily activate it. It is generally triggered when the user hovers over(mouse over) an element with a cursor.

But you cannot detect when you hover something on mobile devices or devices with touch screen, hover is being used less and less now.

:active: is used to select and style the active link or button being clicked. When using a mouse, "activation" typically starts when the user presses down the primary mouse button.

:focus: is used to select the element that has focus. It is generally triggered when the user clicks or taps on an element or selects it with the keyboard's tab.

23. How do you apply CSS styles to only the first letter or line of an element?

*::first-lette*r

The ::first-letter CSS pseudo-element applies styles to the first letter of the first line of a block-level element, but only when not preceded by other content. (such as images or inline tables).

::first-line

The ::first-line CSS pseudo-element applies to the first line of a block-level element.

24. What is the CSS property box-sizing and how does it work?

https://dev.to/zachsnoek/understanding-box-sizing-in-css-28bk

25. How do you change the default cursor in CSS?

https://www.copycat.dev/blog/css-cursor/

https://medium.com/geekculture/changing-cursor-with-css-for-better-ux-407110d1ad76

26. What is the CSS property z-index and how does it work?

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

Syntax:- z-index : auto | number | initial | inherit

- auto : The stack order is equal to that of the parent(default)
- number : The stack order depends in the number
- initial : Set the property to its default value

- inherit : Inherits the property from the parent element.

Z-index works as a stack for the elements that are visible on screen. We can assign a number to the z-index property to assign it a position in the stack. Assigned number can be negative. Greater number assigns front position and lesser number assigns behind position.

27. What is CSS flexbox?

CSS flexbox is a one dimensional layout pattern that makes it easy to design flexible and effective layouts. Divide space between items and control their alignment in a given container flex layout. It also provides a lot of flexibility. With flexbox, we can organize items from left to right, top to bottom, and at the same time control the spacing and order of the items in the container.

28. How can we center the div using flexbox?

Here are the three lines of code we added to the container class like below.

display: flex;

justify-content: center;

align-items: center;

As expected , we begin with display: flex; which allows us to use Flexbox in CSS. We then used both the justify-content(horizontal alignment) and align-items (vertical alignment) properties to position the circle at the center.

Angular

4. *What is the CSS box model?*

When displaying the document on a browser, the elements are displayed as rectangular containers according to the standard CSS BOX model. Each box will consist of a content area and optionally padding, border and margin.

The box model is a fundamental concept in css that governs how elements are sized and positioned on a web page. In the box model, every HTML element is treated as a rectangular box, which is compared to four elements.

<u>Content:-</u> The area where the actual "content" goes. This is actual content of the element, such as text, images, or other media.

<u>Border:-</u> The line that wraps the content. This is a line that surrounds the element's padding and content.

<u>Padding:-</u> The space between the content area and the inner edge of the border

<u>Margin:-</u> The space between the element and its surroundings elements.

```
|            Margin              |
+--------------------------------+
|            Border              |
|                                |
|       +-------------------+    |
|       |      Padding      |    |
|       |                   |    |
|       |    Content        |    |
|       |                   |    |
|       +-------------------+    |
|                                |
+--------------------------------+
```

5. *What is the box sizing property?*

The box sizing property determines how the width and height of an elements are calculated. It accepts two values, content-box and border-box.

[Understanding box-sizing in CSS - DEV Community]

6. Merge Map vs Concat Map vs Exhaust Map vs Switch Map

Merge Map:- I am a hard worker, I can prepare multiple orders at the same time. But don't respect to orders sequence

Concat Map:- I respect order sequence! You will get your order as soon as I finish what I am currently doing.

Exhausted Map:- I'm exhausted. When I prepare an order, I won't listen to any other order.

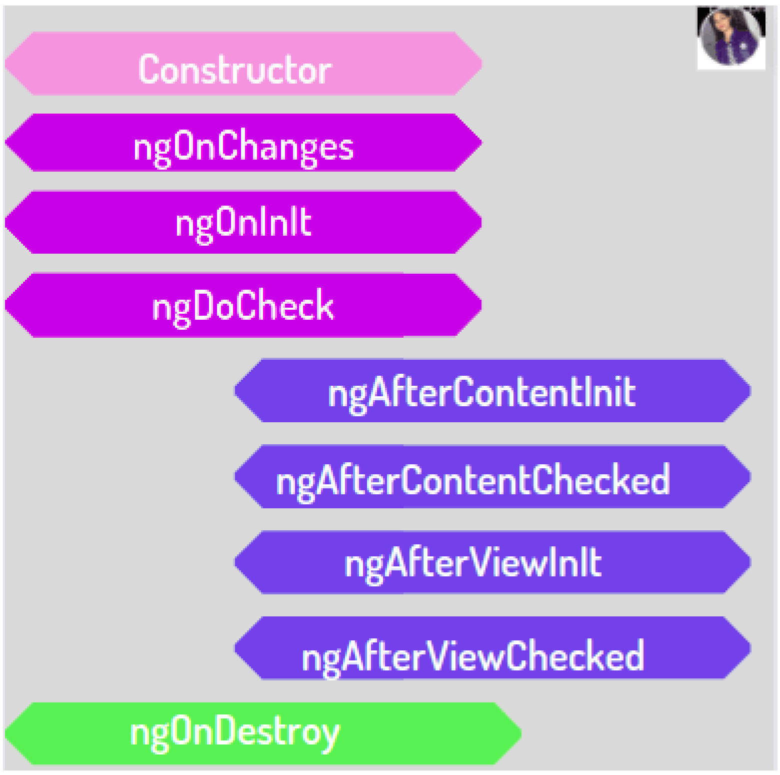
SwitchMap:- I'm mean! Your order will be in the trash If i receive a new one.

7. *What is the LifeCycle of Angular Components?*

Constructor :-

- It is called on component/directive instantiation.
- It also allows centralizing the dependency injections.
- You must avoid subscriptions in constructor

https://dev.to/deepachaurasia1/life-cycle-of-angular-components-3445



ngOnChanges:-

- It allows to set or reset data-bound input properties.
- You use it if you are using @input decorator in your component

ngOnInit:-

- It is called once after the first ngOnChanges()
- It allows initializing the component/directive
- Here we can do subscriptions.

ngDoCheck:-

- Detects and acts upon changes that Angular can't or won't detect on its own.
- It is called after every ngOnChanges and just after ngOnInit() first run.

ngAfterContentInit:-

- Respond after Angular projects external content into the component's view or into the view that a directive is in.
- It is called once after ngDoCheck()
- If you use @contentChild to get contentChild, they are initialized here.

ngAfterContentChecked:-

- Respond after Angular checks the content projected into the directive or component
- It is the same as ngDoCheck but for contentChild properties.

ngAfterViewInit:-

- Responds after Angular initializes the component's view and child views, or the view that contains the directive.
- It is called only once after the first ngAfterContentChecked.

ngAfterViewChecked:-

- Respond after Angular checks the component's view and child views, or the views that contains the directive.
- It is called after the ngAfterViewInit() and every subsequent ngAfterContentChecked().

ngOnDestroy:-

- Cleanup just before Angular destroys the directive or component. Unsubscribe Observables and detach event handlers to avoid memory leaks.
- It is called immediately before angular destroys the component/directives.

8. *What is the difference between constructor and ngoninit?*

    Constructor:-

    - It is a typescript feature with nothing to do.
    - Constructor is transformed into a function with the same name as the class created.
    - Called by the JS engine
    - Used to inject dependencies.
    - It is called when an object of the class is created.

    ngOnInit:-

    - It is one of the life cycle hook of an angular component
    - It is being added to the prototype of the class is created
    - Called by an angular
    - Used to write business logic
    - It is called when angular completes the creation of components.

9. *What is the difference between component and directive?*

    Directive:-

    - @Diretive annotations is used in order to create a new custom directive in the angular application

- Angular directive do not hold on to any particular view instead they can be used in any existing DOM to add behavior as per the requirement.
- Angular directives are not limited to be used only as Attribute inside any DOM element, Angular comes with different types of directives such as Attribute and structural directive.
- The directive should be used whenever you need to add/modify existing DOM behavior based on some action or event performed. This can be used to catch a click event or scroll on UI.
- @Directive is used to create reusable behavior.

Component:-

- @Component annotations is used in angular application in order to create a Component
- Inside the component annotations, a template field is used to provide the view of the particular angular component.
- Angular components can be used as many as any angular application requires. Only one angular component can be present for a particular view and this component can be used across angular applications using the selector tag.
- Always use angular components when you need to create your own view that is attached to its behavior. This component can be accessed by any other component within the application. Unlike directives only one component can be instantiated for a given element in a template.
- @componet is used to create reusable components.

10. *What is the difference between ElementRef, TemplateRef and ViewContainerRef?*

ElementRef:-

The elementref class is a simple wrapper for native elements, Which are usually dom elements in the browser. It provides a way to access the *nativeElement* object, exposing all the methods and properties of the associated native element.

Permitting direct access to the DOM can make your application more vulnerable to XSS attacks.

View - ViewRef:-

Views are the smallest grouping of displayed elements that can be created or destroyed together. Simply put, a component class and its template file define a view.

View Containers - ViewContainerRef:-

View containers are containers for dynamically creating and adding or removing views. A view container can contain

- ➢ Host views, which are created by instantiating a component with the created method.
- ➢ Embedded Views, Which are created by instantiating TemplateRef with the created embed view.

Template Ref:-

A template is a group of HTML elements that can be reused. Views defined in an <ng-tempalte> are called embedded views. Embedded views are not rendered until instantiated.

The TemplateRef class “represents an embedded template that can be used to instantiate embedded views” https://javascript.plainenglish.io/angular-elementref-templateref-viewcontainerref-8517b7ce3274

11. *What is the difference between ng-content, ng-template and ng-container?*

Ng-template:-

As the name suggests the <ng-template> is a template element that Angular uses with structural directives (*ngIf, *ngFor, [ngSwitch] and custom directives). These template elements only work in presence of structural directives which helps us to define a template that does not render anything itself, but conditionally render to the DOM. It helps us to create dynamic templates that can be customized and configured.

ng-Container:-

Ng-conatiner is an extremely simple directive that allows you to group elements in a template that does not interfere with styles or layouts because angular does not put it in the DOM. This is helpful if you don't want any extra div on DOM, you can simply use ng-container

ng-Content:-

Ng-content is used to project the content into angular components. You can use <ng-content></ng-content> tag as a placeholder for that dynamic content then when the template is parsed Angular will replace that placeholder tag with your content.

For example, you have two components as parent and child component and want to show some data in the child component from the parent component

12. *What is the difference between component view, host view and embedded view?*

Component View:-

Each component in angular is represented as a view with nodes. Most nodes in the view resemble component template structure and represent DOM nodes.

Host View:

The host view is a view that contains data for a component element and data for the component class. Angular compiler generates a host view for each component defined in Bootstrap or entrycomponents of a module. And each host view is responsible for creating a component view when you call factory.createComponent. The factories that are returned by the componentFactoryResolver are the host view factories.

Embedded View:

The embedded view is a view that is created for the view nodes specified in the ng-template. It is a component view but does not have a wrapper component element component data like injector etc. Basically it is still a valid view and is checking during detection as any other view.

13. *What is the difference between debounce time and throttle time?*

Debouncing and throttling are techniques in javascript to optimize the application and browser performance.

Debounce Function:-

Debounce function limits the execution of a function call and waits for a certain amount of time before running it again. (Ex- Search box).

Debouncing is a technique where we can monitor the time delay of user action and once that delay reaches our predetermined threshold we can make the function call.

Debounce postpone the execution until there is no input change for the delay period of time. If a change occurs, cancel the previous schedule execution and create a new schedule.

__Throttle Time:-__

Throttling is a technique to limit the execution of an event handler function even when this event triggers continuously due to user actions (Ex- browser resizing).

Throttling is a technique where we can make the function call in a predetermined time interval irrespective of continuous user actions.

Throttle allows execution immediately if the throttle flag is false. After the execution, this function will not be called until the delay period has elapsed.

14. What is the difference between forEach and Map?

15. What is the difference between ng-content and ng-templateoutlet?

__Ng-template:-__

- The Ng-template is basically an angular element. By the name we can say that it contains a part of the template of a component and used to display on a conditional basis.
- Under the hood Angular directive like ngIf, ngFor and ngSwitch uses the ng-template.

__Ng-container:-__

- Ng-container is a grouping element in angular, it doesn’t interfere with any of the styles or layout as the angular doesn’t put into the DOM.
- The other most important feature of ng-container is, it acts as a placeholder where we can inject the template dynamically with the help of ngTemplateOutlet.

__ngTemplateOutlet:-__

- ngTemplateOutlet is an angular directive which is used to instantiate the template automatically. We can have multiple ngTemplateOutlet in a component template to render the templates automatically.
    - ngTemplateOutletContext
    - ngTemplateOutlet

➔ ***ngTemplateOutletContext*** - This is of type Object which is used to pass the dynamic data into the template. The object keys will be available for the local variable can be defined using the let keyword to bind the values inside the template. Using the key $implicit in the context object will set its values as default.

➔ ***ngTemplateOutlet*** - This of type TemplateRef<any> which will be the reference of ng-template.

- Content projection is a way to import HTML content from outside the component and insert that content into the component’s template in a designated spot.
- The problem is that ‘nested’ components are immediately initialized and not destroyed when removed from DOM. This can cause unexpected overhead, if the code for initialize the nested component is heavy.
- This is by design. The lifecycle of a component is always tied to the place where the component was declared, not to the place where the is used. In this case use should use ***ngTemplateOutlet.**

16. What is the difference between forRoot and forChild?

- RouterModule.forRoot is known as a static method, and it is used to configure the modules.
- RouterModule.forChild is also a static method but it configures the routes of lazy-loaded modules.
- A pattern for singleton services that allows us to have control over providers through the applications cycle and modules.
- Both are used to declares the router information and manages the routes
- One difference is that forRoot is typically used for the entire application while forChild is used in feature modules.
- Another key difference is that forRoot registers the root services while forChild does not.
- forRoot creates a module that contains all the directives, the given routes and route service itself.
- forChild creates a module that contains all the directives and the given routes but does not include the router service. It registers the routers and uses the router service created at root level.

17. Why do we use pipe operators in RXJS? What is the use of it?

- A pipe takes in data as input and transforms it to the desired output.
- You can use pipes to link operators together, Pipes let you combine multiple functions into a single function.
- The pipe function takes as its arguments the functions you want to combine, and returns a new function that, when executed, runs the composed functions in sequence.
- The pipe is both a standalone function and a method on the Observable interface that can be used to combine multiple RXJS operators to compare asynchronous operations.

18. What is the difference between using Async pipe vs the subscribe function in the angular application?

- Async pipe is a pipe that subscribes to an observable or promise for you and returns the last value that was emitted.
- Every time a new value is emitted the async pipe will automatically mark your component to be checked for changes.
- If you subscribe() to an observable or Promise you will need to unsubscribe() at the end of your component’s life cycle to avoid memory leaks.
- Change detection works very well with the async pipe.
- We don’t have to unsubscribe manually because the AsyncPipe handles this for us.
- When you use an async pipe you will use ChangeDetectionStrategy in OnPush it will improve your performance. Async pipe it's better in most of the cases because it is in change to handle subscription and unsubscription and notify which receive part of the code is going to rendered.

20. What is the difference between promise and observable?

Both observables and promises help us work with asynchronous functionality in Javascript. Promises deal with one asynchronous event at a time, while observables handle a sequence of asynchronous events over a period of time.

Observables:-

- Emit multiple values over a period of time
- Are lazy; they are not executed until we subscribe to them using the subscribe() method.
- Have subscriptions that are cancellable using the unsubscribe() method, which stops the listener from receiving further values.
- Provide the map, filter, reduce, retry, retryWhen, and so many other RxJS operators, that makes it easy to deal with Observables.
- Deliver errors to the subscribers.

Promises:-

- Emit a single value at a time
- Are not lazy execute immediately after creation
- Are not cancellable.
- Don't provide any operations.
- Push errors to the child promises.

21. What is the difference between Event Emitter and Subjects?

Event Emitter:-

Event emitter is imported from the "@angular/core" package. It is used in directives and components to emit custom events synchronously or asynchronously and register handlers for those events by subscribing to an instance. Event Emitter is a generic type, it can take any additional information.

It uses @output and @input decorators.

Service:-

Services are a great way to share information among classes that don't know each other.

Subject:-

A subject is imported from the "rxjs" package. A subject is an active listener. It triggers with next(). A subject is a special kind of Observable. It maintains the stream of data which is maintained actively. A subject is on observer and observable. Event emitter built on top of the subject.

- Event emitter is built on top of the subject. A subject is better to use in case of cross-component communication.
- Use EventEmitter when transferring data from child component to parent component.
- Use Subject to transfer data from one component to another component.

22. What is the difference between Observable and Subject?

RXJS:-

- Rxjs is a library supporting reactive programming, very often used with an angular framework. It provides an Observable class that helps to compose asynchronous and event based programs.
- The most important concepts in RxJs for asynchronous event handling are Observables, Observers, Subjects, Subscriptions, Operators, and Schedulers.

Observable:-

- Observable is a new way of handling asynchronous requests, just like Promises or callbacks. Concerning push and pull methods, Observables is a push(data producer) collections of multiple values
- Observables pass four stages during their life cycle: creation, subscription, execution and destruction.
- Observable execution can provide three types of notifications.
    - ➢ Next, which sends a value
    - ➢ Error, which returns an error
    - ➢ Complete, which does not send a value.

Observer:-

- The observer is a consumer of values delivered by the Observable

Subscribe to Observable

- A subscription is an object that represents a disposable resource. When Observable is executed, the subscription gets new resources. Subscription has one important .unsubscribe() and it doesn't take any params. It just removes values kept in the Subscription object.

Subject

- The subject is another type of Observable, and it allows value to be consumed by many Observers, not like in the normal Observable just by one. This means that Subjects are multicast, and Observables are unicast.

Difference between Observables and Subjects

- First of all, Observables can't be data consumers, they are just data providers, but Subjects can be both consumers and providers.
- Observables are passive subscribers to the events, and they don't generate anything on their own, when Subjects can trigger new events with available methods like .next() or .complete()
- Another important difference is in firing events. When we have more than one subscriber on the channel, there are two ways of handling events. In one case, all subscribers get the **same event**, and its is the case of Subjects, but in Observables, we can get a **different result on each Observer,** because subscribers get another instance of the event.

23. What is the difference between Activated Route vs Activated route snapshot?

- Volatile state, use ActivatedRoute.
- Static state, use ActivatedRouteSnapshot

- ActivatedRoute requires that you subscribe. Which requires that you unsubscribe. Which requires that you implement OnDestroy. This is a lot of overhead for a static route.
- If you use ActivatedRouteSnapshot and have a parameter in your route definition like product/:id, then you will not get any new id's if the user changes them or your page does. Snapshot means that it was when OnInit ran, this was the state it was in at that point in time. So any changes will be ignored.

24. What is eager loading, Lazy loading, Pre-loading in Angular?

Eager Loading:-

- Feature modules under Eager Loading would be loaded before the application starts. This is the default module-loading strategy.
- Small size applications. In this case, it's not expensive to load all modules before application starts, and the application will be very faster and more responsive to process requests.
- Core modules and feature modules that are required to start the application. These modules could contain components of the initial page, interceptors(for authentication, authorization and error handling etc.), error response components, top-level routing and localization etc. We just have to eagerly load these modules to make the application function properly despite the application size.
- *It used to load core modules and feature modules that are required to start the application.*

Lazy Loading:-

- Feature Modules under lazy loading would be loaded **on demand** after the application starts. It helps to start applications faster.
- The scenario of applying Lazy Loading is relatively simple and straightforward. In a big-size web application, we can lazily load all other modules that are not required when the application starts.
- *It used to load specific feature modules that are very likely to be used soon after the application started*.

Pre Loading:-

- Feature Modules under Pre-Loading would be loaded automatically after the application starts. Compared with Eager Loading and Lazy Loading, Pre-Loading is not so much frequently used in web application development.
- Medium size applications. In this scenario, we can the application start faster since it will load all other modules later that are not required to run the application. And the application would be more responsive to process user's requests than applying Lazy Loading strategy since the application will load all these modules after the application started.
- Some specific modules that users are very likely to use after after the application started. In this scenario, we can pre-load these feature modules and still lazy load other modules.
- *All other modules could be lazily loaded on demand after the application started*.

25. What is Metadata?

- Metadata is used to decorate a class so that it can configure the expected behavior of the classes.

- When you configure a component, you are providing a metadata for that class that tells angular that you have a component, and that component has a specific configuration. Each decorator has a base configuration with some default values.
- There are four types of decorators in Angular
    - Class Decorators
    - Property Decorators
    - Method Decorators
    - Parameter Decorators

- **Class decorators** are the top level decorators that are used to define the purpose for the classes. They provide information to Angular that a particular class is a component or module.
- **Property Decorates** are used to decorate the specific within the classes. Take a look at @Input(). Imagine that you have a property within the class that you want to have an input binding. With decorators you can simply put the **@Input()** decorator above the property for which Angular's compiler will automatically create an input binding from the property name and link them.
- **Method Decorators** decorates specific methods within your class with functionality. A good example of this is **@HostListener**. This tells Angular that when an event on your host happens, you want the decorated method to be called with the event.
- Parameter Decorators are used to decorate parameters in your class constructors. For example **@Inject.** It tells Angular that what you want the parameter to be initiated with.

26. What is routelinkActive use for?

The RouteLinkActive is a directive for adding or removing classes from an HTML element that is bound to a RouterLink. Using this directive, we can toggle CSS classes for active Router Links based on the current RouterState. The main use case of this directive is to highlight which route is currently active. You can either make the font bold or apply some background color.
Example:-

- `<li><a [routerLink]="['home']" routerLinkActive="active">Home</a></li>`
- `<li><a [routerLink]="['product']" [routerLinkActive]="['active']">Product</a></li>`

27. Where do we use generics in angular?

In Typescript generics can be used to better describe the type in classes that are reusable with multiple derived types. Angular does not support instantiation of generic components, but this limitation can be worked around by creating a derived component for each derived type we want to use it with. This requires some extra code but we are rewarded with better type safety.

28. What is wild card route?

The Wild Card route is the last route because it matches any URL. Wild card routes are how we tell Angular how to handle invalid URLs and what to do with them. An invalid URL could be where the user mistypes the name of the path.

29. What is the difference between ngIf and hidden?

- The ngIf directive and the hidden property are both ways to control the visibility of an element in Angular. The ngIf directive is a structural directive which means that it changes the structure of the DOM by adding or removing elements. This means that when an element is removed from the DOM by ngIf, it is no longer available for interaction or event handling.
- The hidden property, on the other hand is a simple property that sets an element's display CSS property to none.

30. What is a router outlet?

- Router-Outlet is an angular directive from the router library that is used to insert the component matched by the routes to be displayed on the screen.
- The Router-Outlet is a directive that's available from the @angular/router package and is used by the router to mark where in a template, a matched component should be inserted.

31. What is the Router State?

- Angular RouterState is the state of the router as a tree of activated routes. It tells how the various components of an application are arranged on the screen to define what should be displayed on it.
- RouterState represents the state of the router as it keeps changing over time when users navigate from page to page.

32. What is an Active route?

A service that is provided to each route component that contains route specific information such as router parameters, static data, resolve data, global query params and the global fragment.

https://blog.briebug.com/blog/what-is-the-activated-route#:~:text=What%20is%20an%20activated%20route,params%20and%20the%20global%20fragment.

33. Explain different injections in angular?

- Dependency injection is a design pattern and mechanism for creating and delivering some parts of an application to other parts of an application that require them.
- In a loosely coupled, flexible and independent way.
- There are three types of dependency injection in angular, they are Constructor injection, Setter Injection, Interface injection.
- ***Constructor Injection*** :- Here, it provides the dependencies through a class constructor.
- ***Setter Injection***:- The client uses a setter method into which the injector injects the dependency.
- ***Interface Injection***:- The dependency provides an injector method that will inject the dependency into any client passed to it. On the other hand, the clients must implement an interface that exposes a setter method that accepts the dependency.

34. *What is the best way to implement translations in angular? - OPEN QUESTION*

35. Explain different routing params in Angular?

- Routing is one of the fundamental mechanisms in Angular. Its primary use is to provide a way to navigate through applications. Without it, we would be struck on the same page forever.
- Angular provides three different types of route parameters , those are
    - Required Parameters
    - Optional Parameters
    - Query Parameters

- ***Required Parameters:-*** As their name suggests , these parameters are required for the next routed component to function properly. Required Parameters must be defined as part of the route configuration.
  Example
  ```
  {
    path: 'pandas/:id', component: PandaDisplayComponent
  }
  ```
- ***Optional Parameters:-*** Unlike required parameters, optional parameters are not mandatory. Therefore, they are not part of the route configuration.
  Example
  ```
  <a  [routerLink]="['/pandas', { filterTerm, sortBy }]"> Pandas </a>
  ```

- ***Query Parameters:-*** Query Parameters, like optional parameters, are not specified as part of the route configuration.
  Example
  ```
  <a [routerLink]="['/pandas']"
    [queryParams]="{ filterTerm, sortBy }"> Pandas </a>
  ```

36. What is a virtual scroll in Angular?

There are 3 ways to handle to show more than thousands of elements at a time in a table or list.

- ***Pagination:-*** Paginate your list and show items as chunks. It's performant but you can't get the whole snapshot of your list at once and you also have to click back and forth between pages.
- ***Infinite Scroll:-*** Initially loads only few items and keep appending more items to the list as you scroll, it gives the complete snapshot of the list but as more and more items keep adding, list is going to slow down eventually as the number of nodes in the DOM are increasing and thus application will start slowing down.
- ***Virtual Scroll:-*** Display only small subset of the data that is in the viewport and keep changing the records as the user scrolls. It keep the number of DOM elements constant hence boosting the performance of the application.

37. What is the difference between route param vs query param?

- Query parameters allow you to pass optional parameters to a route such as pagination information.
- The key difference between query parameters and route parameters is that route parameters are essential to determining route, whereas query parameters are optional.

38. Explain different guards supported in Angular?

Angular route guards are interfaces provided by angular which when implemented allow us to control the accessibility of a route based on condition provided in class implementation of that interface.
Five types of route guards are provided by angular

- CanActivate
- CanActivateChild
- CanLoad
- CanDeactivate
- Resolve

CanActivate:- To implement route guard preventing access to specific route we use CanActivate interface.

CanActivateChild:- CanActivateChild is almost similar to CanActivate interface, the only difference is CanActivate is used to control the accessibility of the current route but CanActivateChild is used to prevent access to child routes of a given route, so by using this you don't need to add canActive on each child route, in other words, you just need to add canActivateChild to parent route and it will work for child route as well.

CanLoad:- Modules in angular can be loaded all at once or be lazy loaded. By default angular load all modules eagerly. To implement lazy loading we use *loadChildren* in route definition. The main advantage of lazy loading is that it reduces the loading time of application by downloading only the required modules.

CanDeactivate:- This route guard is used to keep the user from navigating away from a specific route. This guard can be useful when you want to prevent a user from accidentally navigating away from without saving or some other undone tasks.

ResolveGuard:- Complex angular applications involve data communication between components, sometimes data is so heavy that it is not possible to pass data through query params. To handle this situation angular has provided a Resolve Guard.

39. What is the best way to lazy load the component?
Lazy loading loads the resource as and when required,

- Faster page loads
- Better user experience
- Bandwidth conservation

Different ways of implements

- Import statements
- Async-await function
- Then method

40. What is the way we can display the app version in Angular? - OPEN QUESTION?

41. What are the generators in ES6?

- Generator (or Generator function) is the new concept introduced in ES6. It provides you a new way of working with iterators and functions.
- ES6 generator is a different kind of function that may be paused in the middle either one or many times and can be resumed later.
- A generator is a process that can be paused and resumed and can yield multiple values. A generator in Javascript consists of a generator function which returns an iterable Generator object.

42. What is bootstrapping in angular?

The process of loading the index html page, app-level module, and app-level component is called bootstrapping or loading the app.
Angular takes the following steps to bootstrap the application

- Load index.html
- Load Angular, Other Libraries, and App code
- Execute main.ts file
- Load App-Level Module
- Load App-Level Component
- Process template

43. What are Angular elements? Why do we use it?

Angular Elements are regular angular components that have been packaged as Custom Elements. Custom Elements are part of Web Components and allow us to create new HTML elements, that can be used directly on our page, or by using a polyfill if they are not supported by the browser.

There are two types of custom elements.

- **Customized built-in elements.** They inherit from other basic HTML elements. (<p>, <span>, etc)
- **Autonomous custom elements**, which are standalone and don’t inherit from other HTML elements

Why should I use Angular Elements

- You are migrating from AngularJS to Angular(allows a gradual migration)
- You have a static site, and you just want to embed a complex component to it.
- You have a large team working with different technologies, and you want to share some parts of your app
  .

46. What is the difference between Javascript and TypeScript? - OPEN QUESTION

47. What do you know about Closures? - OPEN QUESTIONS

48. What is the difference between Template Driven forms and Reactive Forms?

- Template-driven forms use FormsModule, while Reactive forms use the ReactiveFormsModule

- Template-driven forms based only on template directives, but Reactive forms are defined programmatically at the level of the component classes.

- Reactive forms are more powerful as they are easier and to use, therefore, forms are better default choice for new applications.

- The template driven approach is very familiar to AngularJS developers.

- The reactive approachable to removes validation logic from the template, keeping the templates cleaner

- Compared to template driven forms, reactive forms are more robust. They are more scalable, reusable, and testable.

- Template-driven forms are suitable for small or simple forms, whereas reactive forms are more scalable and suitable for complex forms.

- Reactive forms are easier to use in general and support better more advanced use case via its Observable-based API.

49. What are different kinds of binding possible in Angular?

Data binding is an important concept of Angular. It allows us to define the communication between component and view. Data Binding is passed from component to view and from view to component.

Types:-

- String Interpolation
- Property Binding
- Event Binding
- Two Way Data Binding

<u>String Interpolation:-</u> This data binding is one way in the sense that the Model's property can only be updated from the Model side and the View is updated or the Model's variables are kept up to date from the View. The data flow is in one direction either from Model-to-view or View-to-Model. One way data binding is unidirectional. *SYNTAX: {{propertyname}}*

<u>Property Data Binding:-</u> Interpolation is a special syntax that Angular converts into property binding *[pair of square bracket]*. It's an alternative to property binding.

<u>Event Data Binding:</u>- A user expects a UI to respond to her/his actions on the page. Every such action would trigger an event on the page and the page has to respond by listening to these events like clicks, keystrokes, change events, etc. Angular gives you a third type of binding of capture events raised on template in a component class.

Two way data Binding:- The combination of property binding and the event binding is called the two way data binding. Two way data binding, automatic synchronization of data happens between the Model and View. Whenever you make changes in the model it will be reflected immediately in the view component.

50. Which RxJS operators are used for transforming or manipulating data?

51. Which RxJS Operators do you use mostly to handle HTTP services?

52. What is the difference between Subject and BehaviourSubject?

Subject

- If you subscribe to a subject, we won't get the current value or initial value.
- For Subject, we don’t need to initial value.
- In Subject, the subscribers will only receive the upcoming value.
- A special type of Observable that allows values to be multicasted to many Observers

BehaviourSubject

- When you subscribe to a behaviourSubject, you will get the current value or initial value.
- Whenever we declare BehaviourSubject we need to initialize a default value.
- In BehaviourSubject, the subscribers will receive the previous value and also upcoming value.
- A subject that can ‘store’ a current value that new subscribers will receive.

ReplaySubject :- All previous values and upcoming values. A subject that can send old values to new subscribers.

AsyncSubject:- latest value when stream will close. ASYNC pipe will take care of all manual work of subscribing and unsubscribe to the component.

https://gist.github.com/rupeshtiwari/e890234dc15e06382de2875792b5211f

53. What is the difference between PURE and IMPURE pipe in angular?
PIPES:-

A pipe takes in data as input and transforms it to a desired output. You can chain pipes together in potentially useful combinations. You can write your own custom pipes. Angular comes with a stock of pipes as *DatePipe, UpperCasePipe, LowerCasePipe, CurrencyPipe and PercentPipe*.

Pure Pipe:-

- A pure pipe is called when angular detects a change in the value or the parameters passed to a pipe.

- Pure pipes are the pipes which are executed only when a “PURE CHANGE” the input value is detected.

<u>Impure Pipe:-</u>

- An impure pipe is called for every change detection cycle no matter whether the value or parameter(s) changes.
- Impure pipe executes every time irrespective of whether the source has changed or not which leads to bad performance.

54. What is multicasting?

- Multi-Casting is the practice of broadcasting to a list of multiple subscribers in a single execution.
- Multicast shares the source Observable by using a subject.
- Multicasting basically means that one Observable execution is shared among multiple subscribers.
- The main reason to use Subject is to multicast. An Observable by default is unicast. Unicasting means that each subscribed observer owns an independent execution of the Observable

57. Why IVY in Angular?

- IVY is Angular's rendering engine. By using IVY we can compile components more independently of each other. IVY promises smaller bundle size, easy and better debugging, mainly for boost in overall performance.
- IVY introduced in angular version of 8, but now in angular version 13 it’s 100% IVY and NO MORE SUPPORT for View Engine
- There are two main concept of IVY
    - Tree Shaking
    - Locality
- ***<u>Tree Shaking:-</u>*** Tree shaking is used to remove unused code during the bundling process. We can do this by using tools like RollUp and Uglify. At the time of build process, tree shaking tools use static analysis and eliminate the unused and unreferenced code. However, tree shaking tools also have limitations when the conditional code exists as static analysis depends on references.
- ***<u>Locality:-</u>*** The locality is used to compiling each component independently with its own local information that rebuilds faster by compiling partial changes and not the entire project files because of this increase the process of build.
- Angular IVY also known as “Angular next-generation compilation and rendering pipelines ”. The advantages are
    - Reduced bundled size
    - Improved testing
    - Better debugging
    - Improved CSS class and style binding
    - Improved type checking
    - Improved build errors
    - Faster build times, enabling AOT on by default
    - Improved Internationalization

58. What is angular universal?

- Angular Universal is simply the nickname for Server-Side-Rendering(SSR) with Angular. Technically the package is now found under *@angular/platform-server.*
- This approach has several benefits. One of the main benefits is that it allows search engines to crawl and index the content of the application, which can improve the initial load time of the application, as the browser does not have to execute any Javascript code before displaying the content.
- Another difference is that Angular Universal is an add-on package that can be added to an existing Angular project, while Angular is a standalone framework.

59. What is the difference between JIT vs AOT?

<u>JIT:-</u>

- JIT compiles our app in the browser at run-time
- Complies before running
- Compiles the entire application(On-demand) in the browser at the runtime. Default out of the box until Angular version 8.
- Compiles your application in the browser at runtime. This was the default until Angular 8.
- Each file is compiled separately.
- No need to build after changing our app code and it automatically reflects the changes in your browser page.
- Highly secure
- Very suitable for local development

<u>AOT:-</u>

- Compiles the entire application in the browser beforehand (During the build time). Default setting in and post Angular version 9.
- AOT compiled our app code at build time.
- Comiles while running.
- Compiled by the machine itself, via command line(Faster)
- All code compiled together, in-line HTML/CSS in the scripts.
- Highly secure
- Very suitable for production builds.

You can change the compiler option through the angular.json file. Based on your angular version, aot property is set to true by default. However, you can turn it to falsetto enable the JIT compiler.

60. What are new features from angular 8 to 14?

[How I Failed an Angular Developer Interview by Failing to Answer this Simple Question | by Atit Patel | Mar, 2023 | JavaScript in Plain English]

61. What method should we need to override while writing a custom pipe in Angular?

Every pipe will implement the PipeTransform interface. This interface provides a transform() method and we have to override it in our custom pipe class. transform() method will decide the input types, number of arguments and its types and output type of our custom pipe.

62. What is switchMap in angular?

SwitchMap:- SwitchMap is an operator that flattens a higher-order observable(an observable of observables) into a single observable by canceling the previous inner observable when a new inner observable is emitted. In other words, if a new event comes in, it switches to the new observable, ignoring the previous one.

MergeMap:- MergeMap is an operator that merges the output of multiple observables into a single observable, but does not cancel any of the inner observables. This means that all inner observables are subscribed to and their emissions are merged into the output observable.

ConcatMap:- ConcatMap is an operator that maps each value from the source observable to an observable and concatenates the output observables. This means that it subscribes to each inner observable sequentially, and does not start subscribing to the next inner observable until the previous one completed.

ExhaustedMap:- ExhaustedMap is an operator that maps each value from the source observable to an observable, and then ignores subsequent source values until the mapped observable completes. This means that if a new value comes in while an inner observable is still emitting values, the new value is ignored until the inner observable completes.

https://frontprostechspace.quora.com/Difference-between-SwitchMap-MergeMap-ConcatMap-and-exhaustMap-in-RxJS-Observable

63. How can we improve the application performance in Angular?

- Use trackBy in ngFor loops
- Use lazy loading
- Avoid using ngIf with complex expressions (instead of using a complex ngIf expression, use ngSwitch for better performance)
- Use OnPush change detection strategy
- Use immutable data structures
- Use AOT compilation
- Use Angular Universal for server-side rendering
- Use Rxjs for reactive programming
- Usage of web workers
- Using pure pipes

66. What is the difference between OF and FROM operators in RXJS?

From:- Emit the items one by one of array

Of:- Emit the whole array at once.

Of operator can behave as from operator with spread operator.

let fruits = ['orange','apple','banana']
from(fruits).subscribe(console.log) // 'orange','apple','banana'
of(fruits).subscribe(console.log) // ['orange','apple','banana']
of(...fruits).subscribe(console.log) // 'orange','apple','banana'

RECENTLY OBSERVED QUESTIONS

1. What is the difference between providers and view providers in angular?
2. How can we inject the service dynamically? Based on condition?
3. How can we get the child param routes?
4. What is skip and optional in Providers?
5. What is the alternative of async and await in Rxjs?
6. What is a signal?
7. What are the features of angular 16?
8. What is a template outlet?
9. How can we inject the content into ng-template?
10. How can we create multiple instances of service?
11. How can we debug the angular application?
12. What are the properties of flexbox?
13. How does the micro front end communicate?

https://netbasal.com/rxjs-six-operators-that-you-must-know-5ed3b6e238a0 - READ IT

DSA

Array/Hashing

1. https://leetcode.com/problems/contains-duplicate/
2. https://leetcode.com/problems/valid-anagram/description/
3. https://leetcode.com/problems/two-sum/description/
4. https://leetcode.com/problems/group-anagrams/description/
5. https://leetcode.com/problems/top-k-frequent-elements/description/
6. https://leetcode.com/problems/product-of-array-except-self/description/
7. https://leetcode.com/problems/valid-sudoku/description/
8. https://leetcode.com/problems/longest-consecutive-sequence/description/

Solutions:- https://levelup.gitconnected.com/dsa-javascript-arrays-hashing-map-or-set-72115e45de87

Two Pointers

1. https://leetcode.com/problems/valid-palindrome/description/
2. https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/description/
3. https://leetcode.com/problems/3sum/description/
4. https://leetcode.com/problems/container-with-most-water/description/
5. https://leetcode.com/problems/trapping-rain-water/description/
6. https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/

Solutions:- https://levelup.gitconnected.com/dsa-interview-preparation-two-pointer-89897d267553

DSA - Stack

1. https://leetcode.com/problems/valid-parentheses/description/
2. https://leetcode.com/problems/min-stack/description/
3. https://leetcode.com/problems/evaluate-reverse-polish-notation/description/
4. https://leetcode.com/problems/generate-parentheses/description/
5. https://leetcode.com/problems/daily-temperatures/description/
6. https://leetcode.com/problems/car-fleet/description/
7. https://leetcode.com/problems/largest-rectangle-in-histogram/editorial/ (Detail - Desc)

Solutions:-
https://sonikamaheshwari067.medium.com/dsa-javascript-stack-interview-preparation-eccea836f366

Binary Search

1. https://leetcode.com/problems/binary-search/description/

2. https://leetcode.com/problems/search-in-rotated-sorted-array/description/ (*Distinct Element*)
3. https://leetcode.com/problems/search-in-rotated-sorted-array-ii/description/ (Non Distinct Elem)
4. https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/description/
5. https://leetcode.com/problems/find-minimum-in-rotated-sorted-array-ii/description/
6. https://leetcode.com/problems/find-peak-element/description/
7. https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/description/
8. https://leetcode.com/problems/koko-eating-bananas/description/
9. https://leetcode.com/problems/search-a-2d-matrix/description/ (Whole 2D array is sorted)
10. https://leetcode.com/problems/search-a-2d-matrix-ii/description/ (Individual Row Col Sorted)
11. https://leetcode.com/problems/row-with-maximum-ones/description/
12. https://leetcode.com/problems/search-suggestions-system/description/
13. https://leetcode.com/problems/time-based-key-value-store/description/
14. https://leetcode.com/problems/median-of-two-sorted-arrays/description/

Solutions:- https://levelup.gitconnected.com/dsa-binary-search-ba7bbbedfb2b

Sliding Window

1. https://leetcode.com/problems/longest-substring-without-repeating-characters/description/
2. https://leetcode.com/problems/longest-repeating-character-replacement/description/
3. https://leetcode.com/problems/permutation-in-string/description/
4. https://leetcode.com/problems/minimum-window-substring/description/
5. https://leetcode.com/problems/sliding-window-maximum/description/
6. https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/

Solutions:- https://levelup.gitconnected.com/dsa-sliding-window-interview-preparation-92a5d6dcb9de

Greedy Problems

1. https://leetcode.com/problems/maximum-subarray/description/
2. https://leetcode.com/problems/jump-game/description/
3. https://leetcode.com/problems/jump-game-ii/description/
4. https://leetcode.com/problems/hand-of-straights/description/

Solutions:- https://levelup.gitconnected.com/javascript-dsa-greedy-problems-d558289239b8

1D Dynamic Programming

1. https://leetcode.com/problems/climbing-stairs/description/
2. https://leetcode.com/problems/min-cost-climbing-stairs/description/
3. https://leetcode.com/problems/house-robber/description/
4. https://leetcode.com/problems/house-robber-ii/description/
5. https://leetcode.com/problems/longest-palindromic-substring/description/
6. https://leetcode.com/problems/palindromic-substrings/description/

7. https://leetcode.com/problems/coin-change/description/
8. https://leetcode.com/problems/maximum-product-subarray/description/
9. https://leetcode.com/problems/word-break/description/
10. https://leetcode.com/problems/longest-increasing-subsequence/description/

Solutions:-
https://sonikamaheshwari067.medium.com/dynamic-programming-coding-interview-preparation-1d-array-ff996e865e2e

DSA Reference:- https://levelup.gitconnected.com/all-about-dsa-roadmap-javascript-4b4e2b13665e

2D Dynamic Programming

1. https://leetcode.com/problems/unique-paths/description/
2. https://leetcode.com/problems/unique-paths-ii/description/
3. https://leetcode.com/problems/minimum-path-sum/description/
4. https://leetcode.com/problems/triangle/description/

Solutions:-
https://sonikamaheshwari067.medium.com/dp-2d-or-grid-unique-paths-unique-paths-ii-minimum-path-sum-77fd466ac12f

Longest Common Sequence

1. https://leetcode.com/problems/longest-common-subsequence/description/
2. https://www.geeksforgeeks.org/longest-common-substring-dp-29/
3. https://leetcode.com/problems/longest-palindromic-subsequence/description/
4. https://leetcode.com/problems/minimum-insertion-steps-to-make-a-string-palindrome/
5. https://leetcode.com/problems/delete-operation-for-two-strings/description/
6. https://leetcode.com/problems/shortest-common-supersequence/description/
7. https://leetcode.com/problems/distinct-subsequences/description/
8. https://leetcode.com/problems/edit-distance/description/

Solutions:-
https://levelup.gitconnected.com/dp-2d-longest-common-palindromic-subsequence-or-substring-minimum-insertion-deletion-b506f35e99ec

Array/Matrix | Math & Geometry

1. https://leetcode.com/problems/contains-duplicate/description/
2. https://leetcode.com/problems/valid-anagram/description/
3. https://leetcode.com/problems/two-sum/description/
4. https://leetcode.com/problems/group-anagrams/description/
5. https://leetcode.com/problems/top-k-frequent-elements/description/
6. https://leetcode.com/problems/product-of-array-except-self/description/

7. https://leetcode.com/problems/valid-sudoku/description/
8. https://leetcode.com/problems/longest-consecutive-sequence/description/

Solutions:- https://levelup.gitconnected.com/javascript-dsa-greedy-problems-d558289239b8

https://github.com/sudheerj/javascript-interview-questions#what-are-the-possible-ways-to-create-objects-in-javascript https://github.com/sadanandpai/javascript-code-challenges

Top 100 Questions You Must Prepare For Your Next Angular Interview (1–45) - DEV Community

"Top 25 JavaScript Interview Questions and Answers for 5 Years of Experience (ES6 and Above) | Medium

https://dev.to/acharyaks90/intermediate-level-angular-interview-questions-377e

What is the difference between the :nth-child() and :nth-of-type() selectors in CSS?
What is the difference between a child and a descendant selector in CSS?
What is specificity in CSS?
How do you clear floats in CSS?
What is the difference between em and rem units in CSS?
What are media queries in CSS and how do you use them?
How do you create a responsive design using CSS?
How do you create a gradient background in CSS?
What is the difference between display:none and visibility:hidden in CSS?
What is the difference between pseudo-elements and pseudo-classes in CSS?
How do you create a tooltip in CSS?
How do you center an element horizontally and vertically in CSS?
How do you position an element to the right or left of its container in CSS?
How do you create a dropdown menu in CSS?
How do you create a slideshow using CSS?
What is the difference between the :hover and :active pseudo-classes in CSS?
How do you apply CSS styles to only the first letter or line of an element?
What is the CSS property box-sizing and how does it work?
How do you change the default cursor in CSS?
What is the CSS property z-index and how does it work?
How do you add a background image to an element in CSS?
How do you animate an element in CSS?
How do you create a border radius in CSS?

What is the difference between margin auto and margin 0 auto in CSS?
What is the CSS property overflow and how does it work?
How do you apply multiple classes to an element in CSS?
How do you create a responsive navigation menu in CSS?
What is the CSS property opacity and how does it work?
How do you create a sticky header in CSS?
How do you create a fixed sidebar in CSS?
What is the CSS property text-overflow and how does it work?
How do you change the font size and color in CSS?
How do you add a shadow to an element in CSS?
What is the CSS property transform and how does it work?
How do you create a background color gradient in CSS?
How do you create a border image in CSS?
How do you create a responsive image gallery in CSS?
What is the CSS property position and how does it work?
How do you create a responsive table in CSS?
How do you create a multi-column layout in CSS?
How do you create a sticky footer in CSS?
How do you create a responsive form in CSS?

JAVASCRIPT QUESTIONS

➡️ What are the different data types in JavaScript?
➡️ Explain the concept of hoisting in JavaScript.
➡️ How does prototypal inheritance work in JavaScript?

➡️ Explain the concept of event delegation in JavaScript.
➡️ How does the "this" keyword work in JavaScript?
➡️ What are higher-order functions in JavaScript?
➡️ Explain the difference between synchronous and asynchronous programming in JavaScript.
➡️ How do you handle errors in JavaScript? What is the purpose of try-catch blocks?
➡️ What are the different ways to create objects in JavaScript?
➡️ Explain the concept of callback functions in JavaScript.
➡️ What is the difference between let, const, and var in JavaScript?
➡️ How does the event loop work in JavaScript?
➡️ What are arrow functions in JavaScript? How do they differ from regular functions?
➡️ Explain the concept of closures and their practical uses.
➡️ What is the purpose of the bind, call, and apply methods in JavaScript?
➡️ How do you handle asynchronous operations in JavaScript? What are Promises and async/await?
➡️ Explain the concept of debouncing and throttling in JavaScript.
➡️ What are the different ways to manipulate the DOM in JavaScript?

****

```
var arrayInput = [7, 0, 0, 5, 0, 9, 3, 0, 1, 0, 8, 0];
var tempArray = [];
var zeroArray = [];
var outputArray = [];
for(let i=0; i<arrayInput.length; i++)
{
   console.log(arrayInput[i]);
   if(arrayInput[i] != 0)
   {
      tempArray.push(arrayInput[i]);
   }else {
      zeroArray.push(arrayInput[i]);
   }
}
outputArray = tempArray.concat(zeroArray);
console.log("output is..>", outputArray);
console.log("SORT.....", arrayInput.sort());
```

****

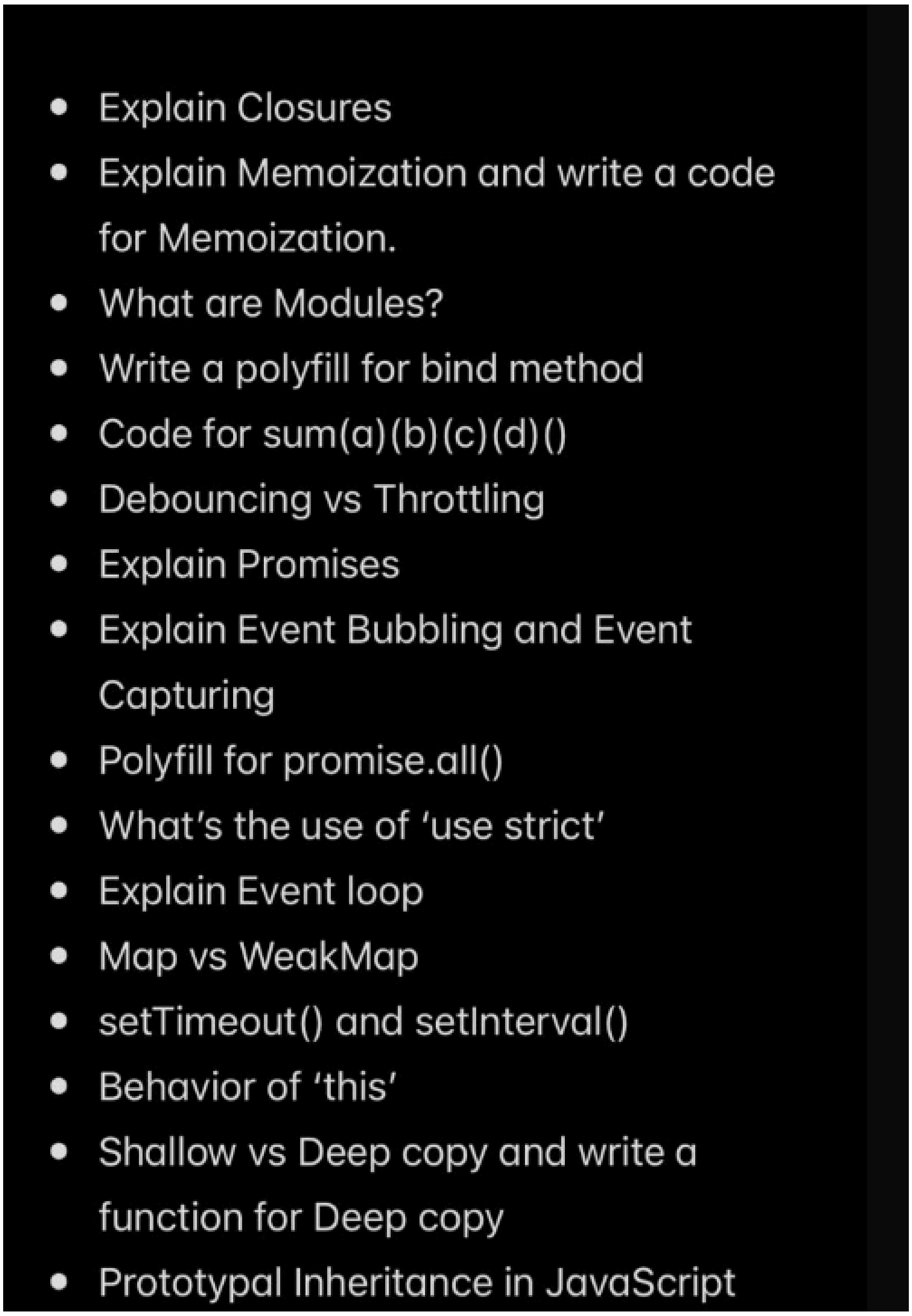
```
var newArray = [3, 2, 3, 5, 2, 7, 6, 4]
var target = 6;
//function of index of those numbers sum equal to be this target

var findPoisition = newArray.indexOf(6);
console.log("Position......>", findPoisition);
for(let i=0; i<newArray.length; i++)
{
   // console.log(newArray[i],".........new Array........", newArray[i+1]);
   if(i == 0)
   {
      if(newArray[i] && newArray[i+1])
     {
       var sum = findSum(newArray[i], newArray[i+1]);
     }
   }

   // console.log("SUM......", sum);
   if(sum == 6)
   {
      let indexOne = newArray.indexOf(newArray[i]);
```

```
        let indexTwo = newArray.indexOf(newArray[i+1]);
        console.log(indexOne,"..........INDEX.......", indexTwo);

    }
}


function findSum(a, b)
{
    return a + b;
}
```

****

1. for of and for in difference
2. What is spread and rest operation
3. Difference between router link and roter outlet
4. What are the different kinds of joins?
5. How can we change the column label with drop a table?
6. Find the second largest number in a array?
7. Write an example for Async and Await?
8. Minimum Number of Platforms Required for a Railway Station, Bloomberg Interview question

Recollection



- Explain Closures
- Explain Memoization and write a code for Memoization.
- What are Modules?
- Write a polyfill for bind method
- Code for sum(a)(b)(c)(d)()
- Debouncing vs Throttling
- Explain Promises
- Explain Event Bubbling and Event Capturing
- Polyfill for promise.all()
- What's the use of 'use strict'
- Explain Event loop
- Map vs WeakMap
- setTimeout() and setInterval()
- Behavior of 'this'
- Shallow vs Deep copy and write a function for Deep copy
- Prototypal Inheritance in JavaScript

Host Sites

1. https://javascript.plainenglish.io/40-portfolio-templates-free-for-web-design-d0611a372763
2. https://pages.github.com/
3. https://www.netlify.com/
4. https://docs.gitlab.com/ee/user/project/pages/
5. https://wordpress.com/
6. https://www.wix.com/
7. https://www.squarespace.com/

IMPORTANT LOOK

https://github.com/kamranahmedse/developer-roadmap

https://github.com/yangshun/tech-interview-handbook

https://github.com/EbookFoundation/free-programming-books

https://github.com/freeCodeCamp/freeCodeCamp

https://roadmap.sh/questions/javascript