100_Days_Of_Frontend_Interview_Questions

100 Days Of Frontend Interview Questions

This repo will contain 500 frontend interview questions which will have HTML, CSS, Javascript & React questions. I will add not any question in which we have to write code, just theory questions.

I am myself learning 5 questions a day and will add those 5 questions a day in this repo till day 100 days. I'll be posting those 5 questions on social media too so if you want to see the 5 questions daily in your social media feed, connect with me on Twitter and LinkedIn.

I've also added some tables, no need to memorize them, just take a look at them few times and you'll easily be able to answer those concepts if asked in interview.

`Source of the questions: Google, ChatGPT, Github repos, etc`

Tables of content

Techs	No. of Questions	Range	Day Range
HTML	50	0-50	Day 1 - 10
CSS	100	51-150	Day 11 - 30
Javascript	180	151-330	Day 31 - 66
React	170	331-465	Day 67 - 93
Typescript	35	466-500	Day 94 - 100

HTML

Day 1

1. ### What is HTML and what are it’s basic components?

   HTML (Hyper Text Markup Language) is the standard markup language for creating web pages. It’s basics components includes element, tags and attribute.

2. ### What is the purpose of Doctype in HTML?

   The Doctype declaration specifies the type of document being used and tell the web browser how to interpret the pages content. It is located at the top of the HTML document.

3. ### What is semantic HTML?

   Semantic HTML uses specific HTML elements to provide additional information about the structure and content of the page, making it more accessible and easy to read.

4. ### What are different types of list in HTML?

   There are 3 types of list in HTML. They are Ordered list, unordered list and definition list.

5. ### What is the difference between div element and span element?

   The div element is a block-level element that is used to group and organize other HTML elements while span element is an inline-element that is used to apply style or attribute to a specific part of a block-level element.

Day 2

1. ### What is the difference between <b> and <strong> tags in HTML?

   The <B> tag is used to apply bold formatting to text, while the <strong> tag is used to indicate that the text is important and carries stronger semantic meaning than <b>.

2. ### What are the different types of input field in HTML?

   There are several types of input field in HTML including text, password, checkbox, radio, button, submit, file and reset.

3. ### What is the difference between a GET request and POST request?

   The GET request is used to retrieve data from a web server, while a POST request is used to submit data to a web server. GET requests are less secure and have limit on the amount of data that can be sent while POST request have no limit and are more secure.

4. ### How do you create a hyperlink in HTML?

   Hyperlinks are created using <a>(anchor) tag, which contains the URL of the link and the text to be displayed.

5. ### What is the purpose of the alt attribute in HTML?

   The alt attribute is used to provide a text description of an image for users who are unable in to see the image, such as those who use screen readers.

Day 3

1. ### What are void elements in HTML?

   Void elements are those elements in HTML which do not have closing tag or do not need to be closed. Example: <br />, <img />, etc.

2. ### Can we display web page inside a webpage or is nesting of webpage possible?

   Yes, we can display a web page inside another HTML web page. HTML provides <iframe> tag using which we can achieve this functionality.

3. ### What is the difference between <link> tag and <a> tag?

   The anchor tag <a> is used to create a hyperlink to another web page or to a certain part of a web page and these links are clickable, whereas, link tag <link> defines a link between a document and external resource and these are not clickable.

4. ### What are some of the advantages of HTML5 over its previous versions?

   Some advantages of HTML5 are:

   i. It has multi media support
   ii. Improved performance (faster loading)
   iii. Better accessibility

5. ### What is the canvas element in HTML5?

   The <canvas> element is a container that is used to draw graphics on the web page using scripting language like javascript.

Day 4

1. **What is an HTML form?**

   HTML form is a section of a web page that allows users to input and submit data to the server.

2. **What is the difference between HTML form's "action" and "method" attributes?**

   The "action" attribute specifies the URL to which the form data is submitted, while the "method" attribute specifies the HTTP method when submitting the form data(either "GET" or "POST").

3. **What would happen if there is no text between the HTML tags?**

   There would be nothing to format if there is not text present between the tags. Therefore, nothing will appear on the screen. Tags without closing tag like <img> do not require any text between them.

4. **What is the purpose of <head> tag?**

   The <head> tag in HTML is used to provide metadata about document, such as title of page, links and other information that is not directly displayed on the web page. This information is used by browser and search engine to understand the content and structure of the web page.

5. **What is the purpose of the <meta> tag in HTML?**

   The <meta> tag is used to provide additional information about the webpage, such as author, keywords, description which is used by search engines to understand the content of the page.

Day 5

1. **What is the purpose of the <script> tag in HTML?**

   The <script> tag is either used to embed client-side script(Javascript), or it points to an external script file through the src attribute.

2. **What is the purpose of the viewport meta tag?**

   Viewport meta tag lets us control the width and scaling of the viewport so that it's sized correctly on all devices.

3. **What is the purpose of the <noscript> tag in HTML?**

   The <noscript> tag in used to display text for those browsers which does not support script tag or the browser disabled the script by user.

4. **What is the purpose of the <fieldset> tag in HTML?**

   Fieldset is used to group related elements in a form. The <fieldset> tag draws a box around the related elements.

5. **what is the difference between HTML tag and HTML element?**

   HTML tag is just opening or closing entity. For example <p> and </p> are called HTML tags. HTML element consists of opening tag, closing tag and content(optional for content-less tags). For example: <p> This is the content </p>, this complete thing is called HTML element.

Day 6

1. **How do you change the color of bullets?**

   The color of the bullet is always the color of the first text of the list. So, if we can change the color of the bullets by changing the color of the text.

2. **What will happen if we don’t put <!Doctype HTML> tag?**

   If we don’t put <!Doctype HTML> tag, the browser will not be able to identify that it is an HTML document and HTML5 tags will not function properly.

3. **What is HTML5 web storage?**

   HTML5 Web Storage is a mechanism in modern web browsers that allows web applications to store data locally within the user’s browser. It provides two storage options: Local Storage and Session Storage.

4. **What is the use of “target” attribute in HTML?**

   The “target” attribute is used to specify where to open the linked document when the user clicks on the hyperlink.

5. **What is the use of “em” tag?**

The “em” tag is a semantic tag which is used to apply emphasis to a word or phrase within a paragraph.
Example: The sentence “I really enjoyed cycling” could be written using the “em” tag to emphasize the word “really”: “I <em>really </em> enjoyed the concert last night.”

Day 7

1. **What is the purpose of <header> tag?**

   The <header> tag is used to defined the header section of a webpage which includes site logo, navigation menu, and other introductory content.

2. **What is the purpose of <nav> tag?**

   The <nav> tag used to define a section of a webpage that contains links to other pages or sections of the current page.

3. **What is the purpose of <main> tag?**

   The <main> tag is used to define the main content of a webpage, where the most important information is displayed.

4. **What is the purpose of <article> tag?**

   The <article> tag is used to define a self-contained piece of content, such as a blog post, news article, or product review.

5. **What is the purpose of <section> tag?**

   The <section> tag is used to group together related content and can be thought of as a container for content.

Day 8

1. **What is the purpose of <aside> tag?**

   The <aside> tag is used to define content that is related to the main content of a webpage, but is not directly part of it. It is often used to provide additional information to the main content.

2. **What is the purpose of <footer> tag?**

The <footer> tag is used to define the footer section of a webpage which contains copyright information, contact details, and other legal content.

3. **What is the purpose of <figure> tag?**

   The <figure> tag is used to group together media content, such as images or video with their associated captions or explanatory text.

4. **What is the purpose of <figcaption> tag?**

   The <figcaption> is used to define the caption or description of a media that is contained within a <figure> tag.

5. **What is the purpose of <blockquote> tag?**

   The <blockquote> tag is used to indicate that a section of text is being quoted from another source. It is used to highlight a particularly important quote.

Day 9

1. **What is <datalist> tag?**

   The <datalist> tag is new addition to HTML5, and it is used to provide a list of pre-defined options for an input field. It allows us to create dropdown list of options.

2. **What is the use of <option> tag?**

   The <option> tag is used to define an option in a dropdown list. The 'value' attribute in <option> tag is specifies the value of the option that will be submitted when the form is submitted.

3. **How can you make an image clickable in html?**

   To make an image clickable, we have to use anchor tag with the image tag nested inside it and specify where we want to be directed to in "href" attribute when image in click.

4. **Details about all tags used in HTML table.**

Tag Name	Definition
<table>	Defines a table

Tag Name	Definition
<caption>	Defines a title or caption for a table
<thead>	Defines the header of a table
<tbody>	Defines the body of a table
<tfoot>	Defines the footer of a table
<tr>	Defines a row in a table
<th>	Defines a header cell in a table
<td>	Defines a standard cell in a table
<colgroup>	Defines a group of columns in a table
<col>	Specifies column properties for each column within <colgroup>
<caption>	Defines a title or caption for a table

5. **HTML table attributes.**

| Attribute | Description | | ——————— |

——————————————————————— | | `border` | Specifies the width of the border around the table. | | `cellpadding` | Specifies the amount of padding to be added to the cells in the table. | | `cellspacing` | Specifies the amount of space to be added between cells in the table. | | `width` | Specifies the width of the table. | | `height` | Specifies the height of the table. | | `align` | Specifies the horizontal alignment of the table within its containing element. Possible values are `left` , `center` , `right` , and `justify` . | | `valign` | Specifies the vertical alignment of the table within its containing element. Possible values are `top` , `middle` , `bottom` , and `baseline` . | | `bgcolor` | Specifies the background color of the table. | | `bordercolor` | Specifies the color of the border around the table. | | `border-collapse` | Specifies whether the borders of the cells in the table should be collapsed into a single border or not. Possible values are `collapse` and `separate` . Default is `separate` . | | `border-spacing` | Specifies the amount of space to be added between the cells in the table when the `border-collapse` property is set to `separate` . | | `caption` | Specifies the title or caption of the table. | | `caption-side` | Specifies the side of the table on which to place the caption. Possible values are `top` , `bottom` , `left` , and `right` . Default is `top` . | | `empty-cells` | Specifies whether or not to display borders around empty cells in the table. Possible values are `show` and `hide` . Default is `show` . | | `frame` | Specifies which sides of the table should have borders. Possible values are `void` , `above` , `below` , `hsides` , `vsides` , `lhs` , `rhs` , and `box` . | | `rules` | Specifies which parts of the table's border should be shown. Possible values are `none` , `groups` , `rows` , `cols` , and `all` . Default is `none` . | | `summary`

| Specifies a summary of the contents of the table. | | `dir` | Specifies the direction of the table's text. Possible values are `ltr` (left-to-right) and `rtl` (right-to-left). |

Day 10

1. **What is the purpose of the "data-*" attribute in HTML?**

   The data=* attribute allows us to store additional information about an HTML element that is not otherwise displayed on the page, but may be useful to scripts that interact with the page.

2. **What is the purpose of <legend> tag?**

   The <legend> tag provides semantic caption or title for a <fieldset> element. It is optional but better for better to have for accessibility.

3. **What are HTML entities?**

   HTML entities are codes used to represent special characters in HTML that cannot be easily typed or displayed. Example: `&amp` represents (`&`)

4. **What is the purpose of the <picture> element in HTML5, and how is it used to optimize images?**

   The <picture> element in HTML5 is used to provide multiple versions of an image at different resolutions or sizes, and allows the browser to choose the best version to display based on the user's device and viewport size.

5. **HTML ARIA attributes.**

   | Attribute | Description | Example | | ——————— |

   ————————— | ———- | | `aria-label` | Provides a short, descriptive label for an element that isn't already provided by its text content or associated label element. | `<button aria-label="Search">🔍</button>` | | `aria-describedby` | References another element that provides additional information about the current element, such as instructions or help text. | `<input type="text" aria-describedby="name-help"> <div id="name-help">Enter your full name</div>` | | `aria-required` | Indicates which form fields are mandatory and must be filled out before the form can be submitted. | `<input type="text" aria-required="true">` | | `aria-expanded` | Indicates whether a collapsible element like an accordion or dropdown menu is currently expanded or collapsed. | `<button aria-expanded="true" aria-controls="menu">Menu</button> <div id="menu">...</div>`

|| `aria-disabled` | Indicates whether an element is currently disabled or not, such as a disabled button. | `<button aria-disabled="true">Submit</button>` || `aria-checked` | Indicates whether a checkbox or radio button is currently checked or not. | `<input type="checkbox" aria-checked="true">` || `aria-haspopup` | Indicates that an element has a pop-up or dropdown menu associated with it. | `<button aria-haspopup="true" aria-controls="menu">Menu</button> <div id="menu">...</div>` || `aria-selected` | Indicates that an element is currently selected or highlighted, such as a selected item in a list | `<li aria-selected="true">Item 1</li>` || `aria-hidden` | Indicates that an element is currently hidden or not visible to users, such as content that is only revealed when certain conditions are met. | `<div aria-hidden="true">Hidden content</div>` |

CSS

Day 11

1. **What is CSS?**

   CSS stands for Cascading Style Sheets. CSS is used to define styles for web pages, including the design, layout and variations in display for different devices and screen sizes.

2. **What are the possible ways to apply CSS styles to a web page?**

   There are 3 ways to apply CSS styles to a web page. They are:

   i. Inline CSS
   ii. Internal CSS
   iii. External CSS

3. **What are some new features in CSS3?**

   Some new features of CSS3 are:

   i. New selectors.
   ii. Pseudo-classes
   iii. Rounded Corners(Border-radius)
   iv. Text shadow

v. Gradient, etc

4. **What are the css selectors?**

   CSS selector is a part of css rule which is used to apply styles to a target specific HTML element or group of elements.

5. **What is universal selector in css?**

   The universal selector is a css selector that can be used to apply styles to all elements on a page or to reset styles for all elements to their default values

Day 12

1. **Explain the difference between classes and IDs in CSS.**

   Classes are used to group together elements with similar styles, while IDs are used to target specific elements on a page. IDs must be unique, while classes can be used multiple times on a page.

2. **Explain box model in CSS.**

   The box model in CSS is a way of representing elements as boxes with content, padding, borders, and margins. The content area is the actual content of the element, the padding is the space between the content and the border, the border is a line around the element, and the margin is the space between the border and the surrounding elements.

3. **What is the difference between margin and padding in CSS?**

   Margin is the space between the border of an element and the surrounding elements, while padding is the space between the content of an element and its border.

4. **Explain the CSS display property and its values.**

   The CSS display property controls how an element is displayed on a page. The possible values are block, inline, inline-block, none, and others.

5. **What is the CSS position property and its values?**

   The CSS position property controls how an element is positioned on a page. The possible values are static, relative, absolute, fixed, and sticky

Day 13

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

   Relative positioning moves an element relative to its current position without affecting the position of other elements, while absolute positioning positions an element relative to its closest parent element, potentially affecting the position of other elements.

2. **Explain the CSS float property and its values.**

   The CSS float property controls the alignment of an element in a page layout. The possible values include left, right, none, and inherit. When an element is floated left or right, other elements will flow around it.

3. **What is the CSS clear property?**

   The CSS clear property controls whether an element is allowed to float next to another element or not. The possible values include left, right, both, and none. When an element is cleared, it will be moved below any floated elements.

4. **What is the CSS z-index property?**

   The CSS z-index property controls the stacking order of elements on a page. Elements with a higher z-index value are displayed on top of elements with a lower z-index value. The default value of z-index is auto.

5. **What are CSS pseudo-classes? Give some examples.**

   CSS pseudo-classes are selectors that target elements based on their state or position in the document. Some examples include :hover, :active, :focus, :first-child, :last-child, and :nth-child.

Day 14

1. **What is CSS animation?**

   CSS animation is a way to add movement and dynamism to the HTML element on a web pages using CSS. It is used to enhance users experience.

2. **How do you define a CSS animation?**

We can define CSS animation by creating a set of keyframes that specify the start and end states of an animation. We can apply the animation to an element using the animation property in css.

3. **How do you trigger a CSS animation to start and stop?**

   You can trigger a CSS animation to start and stop using various CSS selectors and events. For example, you can use the :hover pseudo-class to start an animation when a user hovers over an element.

4. **How do you create a responsive CSS animation?**

   We can create responsive CSS animation by using relative units like percentages instead of fixed units like pixels. We can also use media queries to adjust the animation properties based on the size of the viewport.

5. **How do you optimize CSS animations for performance?**

   We can optimize CSS animation for performance by using techniques like reducing the number of elements being animated, minimizing the use of box-shadow and text-shadow, and using transform and opacity instead of top, left, bottom, and right.

Day 15

1. **How do you debug CSS animations and troubleshoot common issues, such as animations not playing or playing too quickly?**

   We can debug CSS animations by checking for typos or syntax errors, checking browser's compatibility if it supports the property or not, checking for code that may override the animation.

2. **What is transition in CSS?**

   A transition in CSS is a way to create animation effects when a property of an HTML element changes, allowing developers to smoothly animate changes to an element's style over a specified duration.

3. **Can you specify multiple CSS transitions for an element?**

   Yes, you can specify multiple CSS transitions for an element by separating them with commas in the "transition" property.

4. ### What are gradients in CSS?

   It is a property of CSS which allows you to display a smooth transformation between two or more than two specified colors. There are two types of gradients in CSS. They are: Linear gradient and radial gradient.

5. ### What is the difference between linear and radial gradient in CSS?

   A linear gradient creates a smooth transition between two or more colors in a straight line while the radial gradient creates a smooth transition between two or more colors in a circular shape.

Day 16

1. ### What is flexbox?

   Flexbox is a CSS3 layout module which provides more flexible and efficient way to align arrange and align the elements within the container. It is widely supported by the modern web browser.

2. ### What is the difference between flex-grow and flex-shrink properties in Flexbox?

   `flex-grow` is a property in Flexbox that specifies how much a flex item should grow relative to the other items in the container, while `flex-shrink` specifies how much a flex item should shrink relative to the other items when there is not enough space available in the container.

3. ### What is the difference between flex-wrap: nowrap and overflow: hidden in CSS?

   `flex-wrap: nowrap` prevents flex items from wrapping to the next line when they overflow the container, while `overflow: hidden` hides any content that overflows the container’s boundaries.

4. ### What is the `gap` property in Flexbox, and how is it used to add spacing between flex items?

   The `gap` property in Flexbox sets the spacing between flex items, both horizontally and vertically. It is a shorthand for `row-gap` and `column-gap` .

5. ### How do you center an element using flexbox?

   To center an element using flexbox, we need to set the parent container’s display property to `flex` and use the `justify-content` & `align-items` properties with the value of `center` .

Day 17

1. **What is CSS grid?**

   CSS grid is a layout system in CSS that allows you to create complex, multi-dimensional grid layouts for web pages.

2. **How do you use the grid-template-columns and grid-template-rows properties to define the size and number of columns and rows in a grid?**

   You can use the grid-template-columns and grid-template-rows properties to define the size and number of columns and rows in a grid. For example, you might define a grid with three columns, where the first column is 100 pixels wide, the second column is 50% of the available width, and the third column takes up the remaining space, with the following CSS:

   ```
   grid-template-columns: 100px 50% 1fr;
   ```

3. **How do you specify the number of columns in a grid using CSS?**

   You can specify the number of columns in a grid by using the `grid-template-columns` property. For example, `grid-template-columns: repeat(3, 1fr)` will create a grid with three columns that are each one fraction of the available space.

4. **How do you align items within a grid using CSS?**

   You can align items within a grid using the `justify-items` and `align-items` properties. justify-items aligns items horizontally within a grid, while `align-items` aligns items vertically.

5. **What is the difference between grid-row and grid-column in CSS Grid?**

   `grid-row` is used to position an item within a row, while `grid-column` is used to position an item within a column. Both properties can take a value of either a number or a named grid line.

Day 18

1. **What is a CSS transition?**

   A CSS transition is a way to add a gradual animation effect to an element when its CSS properties change.

2. ### What is the purpose of the `transition-property` property in CSS transitions?

   The `transition-property` property specifies which CSS property or properties should be transitioned when a change occurs.

3. ### What is the function of the `transition-timing-function` property in CSS transitions?

   The `transition-timing-function` property specifies the rate of change of the transition over time. This can be used to control the speed of the transition and how it accelerates and decelerates. Example: `ease-in` , `ease` , `ease-out` .

4. ### How do you delay the start of a CSS transition?

   You can delay the start of a CSS transition using the transition-delay property. This property specifies the amount of time to wait before starting the transition, in seconds or milliseconds. Example: `transition-delay: 1s;`

5. ### How do you apply a transition to multiple properties at once?

   You can apply a transition to multiple properties at once by separating the property names with commas in the `transition-property` property. example, `transition-property: background-color, color, opacity;`

Day 19

1. ### What are CSS 2D Transforms?

   CSS 2D Transforms are a set of properties that allow you to transform the position, rotation, scaling, and skewing of an element in two dimensions, without affecting the surrounding elements.

2. ### Can you use negative values with CSS 2D transforms? If so, what effect do they have?

   Yes, you can use negative values with CSS 2D transforms. The effect of negative values depends on the function being used. For example, a negative value for the `rotate()` function will rotate the element counterclockwise instead of clockwise.

3. ### What are the properties of CSS 2D transform?

The properties of CSS 2D Transforms are:

i. transform: specifies the transformation functions to be applied to an element
ii. transform-origin: specifies the point around which the transformation should occur
iii. translate: moves an element along the x-axis and/or y-axis
iv. rotate: rotates an element clockwise or counterclockwise around a given point
v. scale: increases or decreases the size of an element
vi. skew: skews an element along the x-axis and/or y-axis

4. Can you apply multiple transformations to the same element using CSS 2D transforms?

Yes, you can apply multiple transformations to the same element using CSS 2D transforms. You can do this by separating each transformation function with a space in the `transform` property.

5. What is the difference between the `translate()` and `rotate()` functions?

The `translate()` function is used to move an element along the x-axis and/or y-axis, while the `rotate()` function is used to rotate an element clockwise or counterclockwise around a given point.

Day 20

1. What are the properties of CSS 3D transform?

The properties of CSS 3D Transforms are:

i. transform-style: determines whether an element's children are transformed in 3D space
ii. perspective: determines the distance between the viewer and the element, affecting the appearance of 3D transforms
iii. perspective-origin: specifies the origin point of the perspective
iv. transform: applies a 3D transformation to an element, such as rotateX(), rotateY(), rotateZ(), translateX(), translateY(),translateZ(), scale(), and skew()
v. transform-origin: specifies the origin point of the transformation
vi. backface-visibility: determines whether or not the back face of an element should be visible when the element is rotated.

2. What is the difference between RGB and RGBA colors?

RGBA is similar to RGB, but it includes an additional alpha value that represents the opacity of the color. The alpha value is a number between 0 and 1, with 0 being completely transparent and 1 being completely opaque.

3. **What is color contrast?**

   Color contrast is how much the colors of the text and the background of a website stand out from each other.

4. **How does color contrast applies to accessibility in web design?**

   It is important because people with visual impairments or color blindness may have trouble seeing things if there is not enough contrast. There are guidelines for making sure there is enough contrast which makes the website content more accessible to the people.

5. **what is css filter?**

   CSS filters are a set of visual effects that can be applied to HTML elements using CSS. Filters can be used to adjust the appearance of an element in various ways, such as changing its color, blurring or sharpening it, adjusting its brightness or contrast, and more.

Day 21

1. **What are some important considerations when using text effects?**

   Important considerations include making sure the text remains readable and legible, ensuring the effect is appropriate for the overall design, and avoiding overuse of effects.

2. **What is a CSS sprite?**

   A CSS sprite is a technique used to combine multiple images into a single image file, reducing the number of HTTP requests required to load the page and improving loading times.

3. **What is the :not() pseudo-class used for in CSS?**

   The :not() pseudo-class is used to select all elements that do not match the specified selector.

4. **How do you target all even or odd elements using a pseudo-selector in CSS?**

The :nth-child() pseudo-selector can be used with the “even” or “odd” keyword to target all even or odd elements of a parent element.

5. **What is the :visited pseudo-class used for in CSS?**

   The :visited pseudo-class is used to target a link that has been visited by the user.

Day 22

1. **How do you target the first letter of a text element using a pseudo-selector in CSS?**

   The ::first-letter pseudo-element is used to target the first letter of a text element.

2. **What is the :active pseudo-class used for in CSS?**

   The :active pseudo-class is used to target an element when it is in an active state, such as when a user clicks on it.

3. **What is the :lang() pseudo-class used for in CSS?**

   The lang() pseudo-class is used to target elements based on the language attribute of the HTML document. For example, you can use :lang(en) to target all elements in English, or :lang(fr) to target all elements in French.

4. **What is the :checked pseudo-class used for in CSS?**

   The :checked pseudo-class is used to target form elements that have been selected by the user, such as checkboxes or radio buttons. This can be used to change the appearance or behavior of the selected element.

5. **What is the @media rule in CSS?**

   The @media rule in CSS allows developers to apply styles to a web page based on the size of the device or screen being used to view it, making it more responsive.

Day 23

1. **What is the difference between fluid and fixed layouts in CSS?**

A fluid layout in CSS adjusts its width and height based on the size of the screen, while a fixed layout has a set width and height. Fluid layouts use percentages to set their dimensions, while fixed layouts use pixels.

2. ### How do you make images responsive in CSS?

   To make images responsive in CSS, you can use the max-width: 100% property, which will make the image scale down proportionally to fit the width of its container while maintaining its aspect ratio.

3. ### What is the difference between min-width and max-width in CSS media queries?

   In CSS media queries, min-width sets the minimum screen width at which a set of styles will be applied, while max-width sets the maximum screen width at which a set of styles will be applied. For example, if you use min-width: 768px, the styles will only be applied to screens that are 768 pixels wide or larger, while if you use max-width: 768px, the styles will only be applied to screens that are 768 pixels wide or smaller.

4. ### What is the difference between responsive and adaptive design in CSS?

   Responsive design in CSS adapts to different screen sizes and devices by using flexible grids, fluid images, and media queries to adjust the layout and content of the website. Adaptive design in CSS, on the other hand, uses predefined layout sizes and breakpoints to adjust the layout and content based on the screen size and device being used.

5. ### How do you optimize responsive images for faster loading in CSS?

   To optimize responsive images for faster loading in CSS, you can use smaller file formats like JPEG and PNG, reduce the image size and resolution, and use lazy loading to only load images when they are needed.

Day 24

1. ### How does calc() work in css?

   The CSS3 calc() function allows us to perform mathematical operations on property values. Example: `div{width: calc(100px + 50px)}`

2. ### What is the overflow property in CSS used for?

The overflow property specifies what should happen if content overflows an element's box. It's possible values are: auto, none, scroll, visible.

3. **What is the difference between `visibility:hidden` and `display:none` ?**

   `visibility:hidden` means the tag is not visible, but the space is allocated for it on the page. `display:none` means the tag will not appear at all and there will be no space allocated for it between the other tags.

4. **Are quotes mandatory in URL's?**

   Quotes are optional in URL's, and it can be single or double.

5. **Explain what are web-safe fonts and fallback fonts.**

   Web-safe fonts are fonts that are commonly installed on most devices and web browsers. Fallback fonts are alternative fonts specified in case the primary font is not available on the user's device.

Day 25

1. **What is the purpose of CSS content property?**

   The CSS content property is used to insert content before or after an HTML element using the ::before and ::after pseudo-elements.

2. **How can we create custom cursor in CSS?**

   To create a custom cursor in CSS, you can use the "cursor" property and set it to "url" with the path to the image file that you want to use as the cursor.

3. **What is the "line-height" property in CSS?**

   The "line-height" property in CSS is used to control the spacing between lines of text within an element.

4. **What is specificity in CSS?**

   Specificity in CSS is a way of determining which CSS rule applies to an element. It is based on the number of selectors and their types in a CSS rule. Specificity is calculated using a formula:

inline styles have the highest specificity, followed by IDs, classes, and then elements.

5. **Which property is used to control the scrolling of an image in the background?**

   The `background-attachment` property is used to control the scrolling of an image in the background.

Day 26

1. **Which CSS property is used to capitalize text or convert text to uppercase or lowercase letters?**

   The text-transform property is used to capitalize text or convert text to uppercase or lowercase letters.

2. **What is word-wrap property in CSS3?**

   The word-wrap property allows long words to be able to be broken in order to prevent overflow and wrap onto the next line.

3. **Describe 'rule set' in CSS?**

   It is an instruction that tells browser on how to render a specific element on the HTML page. It consists of a selector with a declaration block that follows.

4. **How can you create a CSS-only dropdown menu?**

   A CSS-only dropdown menu can be created by using the "hover" pseudo-class and the "display" property. When the user hovers over a parent element, the "display" property of the child element can be set to "block" or "inline-block" to reveal the dropdown menu.

5. **What are the potential drawbacks of using CSS frameworks such as Bootstrap?**

   Using CSS frameworks like Bootstrap can lead to bloated code, difficulties in customizing the design, and unoriginal or generic looks

Day 27

1. **What is Tailwind CSS?**

   TailwindCSS is a utility-first CSS framework that provides pre-defined CSS classes that can be used to rapidly build custom user interfaces.

2. **How do you customize TailwindCSS to match a specific design system or brand guidelines?**

   TailwindCSS provides a configuration file that can be customized to match a specific design system or brand guidelines. This file includes variables for colors, fonts, spacing, and more, which can be adjusted to match the project’s needs.

3. **Can you explain the difference between utility classes and component classes in TailwindCSS?**

   Utility classes in TailwindCSS are small, single-purpose classes that provide a specific styling utility, such as padding, margin, or text alignment. Component classes, on the other hand, are larger classes that provide a collection of styles for a specific component, such as a button or card.

4. **How do you optimize the file size of TailwindCSS in a production environment?**

   TailwindCSS provides a purge option that removes any unused classes from the final CSS file, reducing its size. This option should be enabled in a production environment to minimize the CSS file size.

5. **What are some common performance issues with TailwindCSS, and how do you optimize performance in your projects?**

   Common performance issues with TailwindCSS include the size of the CSS file and the number of classes being generated. To optimize performance, it is important to enable the purge option in a production environment, use a caching mechanism to speed up builds, and avoid generating unnecessary classes.

Day 28

1. **What is CSS preprocessor?**

   A CSS preprocessor is a scripting language that extends the capabilities of CSS which makes it easier and more efficient to write CSS code.

2. **What is the difference between a CSS preprocessor and a post-processor?**

   A CSS preprocessor generates CSS code from source code written in a higher-level scripting language, whereas a post-processor takes existing CSS code and applies transformations or optimizations to it. In other words, a preprocessor is used during development, while a post-processor is used after development to optimize performance.

3. **What is SASS?**

   Sass is a CSS preprocessor that adds functionality to CSS, such as variables, nesting, and more. It allows us to write more efficient code and simplifies task like browser compatibility.

4. **what is the difference between sass and scss?**

   Sass and SCSS are both CSS pre-processors and are very similar, but they have different syntax. Sass has a more concise and less verbose syntax, with no curly braces and no semicolon whereas SCSS has a syntax that is almost identical to standard CSS, with curly braces and semicolons

5. **Can you describe a situation where you would choose not to use Sass?**

   A developer might choose not to use Sass if they prefer to stick with standard CSS for simplicity or if they are working on a small project where the added features of Sass may not be necessary.

Day 29

1. **What is SASS nesting?**

   Sass nesting is a feature that allows us to write CSS selectors that are nested within one another which makes our code easier to read and understand.

2. **What are variables in Sass?**

   Variables in Sass allow you to assign values to a variable name, which can then be used throughout your stylesheet.

3. **What are mixins in Sass?**

   A mixin is a feature in Sass that allows you to define a set of CSS styles that can be reused throughout your stylesheet. Example:

```
@mixin my-text-style {
 font-size: 16px;
 font-weight: bold;
}

/* we can use the whole style like this now */

h1 {
 @include my-text-style;
}
```

4. **What is Sass inheritance?**

   Inheritance allows us to define a set of styles in one selector, called a “parent”, and then extend those styles to another selector, called a “child”. Example:

```
@mixin button-style($bg-color, $text-color) {
    background-color: $bg-color;
    color: $text-color;
    display: inline-block;
    padding: 8px 16px;
}

// Use the mixin to create different button styles

.button-primary {
    @include button-style(#007bff, #fff);
}

.button-secondary {
    @include button-style(#6c757d, #fff);
}
```

5. **How do you use ‘if’ statements in SCSS?**

   In SCSS, you can use the @if directive to add conditional logic to your styles. Example:

```
//declaring variable
$background-color: #333;

body {

@if $background-color == #333 {
background-color: $background-color;
} @else {

background-color: #fff;
    }
}
```

Day 30

1. **What are some common mistakes that developers make when writing CSS, and how do you avoid them?**

   Common mistakes in CSS include over-reliance on frameworks, lack of organization, and using non-semantic HTML

2. **How do you balance the need for visual aesthetics with the need for website or application functionality?**

   I balance the need for visual aesthetics with the need for functionality by designing with the user in mind, testing designs with real users, and incorporating feedback and data into the design process.

3. **How do you ensure that your CSS is optimized for search engine optimization (SEO)?**

   We can ensure CSS is optimized for SEO by minimizing code bloat to improve load time, use relevant class names, avoiding inline styles, etc.

4. **How do you ensure that your CSS is scalable and maintainable for large projects?**

   We can ensure that our CSS is scalable and maintainable for large project by:

   i. Using proper naming convection for ID and classes.
   ii. Using preprocessor like sass, less, etc.
   iii. Using performance enhancing techniques like lazy-loading, etc.

5. **How do you use CSS variables to create more flexible and dynamic designs, and what are some use cases where variables are particularly useful?**

CSS variables can improve maintainability and reduce repetition. Variables are particularly useful for theming, creating responsive designs, and making changes to global styles.

Javascript

Day 31

1. **What is javascript?**

   JavaScript is a programming language used to create interactive and dynamic web pages, as well as to create more complex applications on the client and server side.

2. **What is the difference between null and undefined in JavaScript?**

   Null represents a deliberate non-value or absence of any object value, while undefined represents a lack of value or an uninitialized variable.

3. **What is the difference between == and === in JavaScript?**

   The double equals (==) compares the value of two variables, while the triple equals (===) compares both the value and the data type of two variables.

4. **What is the difference between let, const, and var in JavaScript?**

   The var keyword is used for variable declaration in older versions of JavaScript, while let and const were introduced in ES6. Var has a function-level scope, while let and const have block-level scope. Additionally, const variables cannot be reassigned after being declared, while let variables can be.

5. **How do you convert a string to a number in JavaScript?**

   You can use the `Number()` or `parseFloat()` functions to convert a string to a number in JavaScript.

Day 32

1. **What is the purpose of the array slice method?**

The `slice()` method returns the selected elements in an array as a new array object. It selects the elements starting at the given start argument, and ends at the given optional end argument without including the last element.

2. **What is the purpose of the array splice method?**

   The `splice()` is an array method in JavaScript that allows you to modify an array by adding, removing, or replacing elements. It takes two required parameters: the index at which to start making changes to the array, and the number of elements to remove. It also has an optional parameter for adding one or more elements to the array.

3. **What is the difference between slice and splice?**

slice	splice
Doesn't modify the original array(immutable)	Modifies the original array(mutable)
Returns the subset of original array	Returns the deleted elements as array
Used to pick the elements from array	Used to insert or delete elements to/from array

4. **What are arrow function?**

   Arrow functions are a shorthand syntax for writing function expressions in JavaScript. They use the `=>` syntax to separate the function parameters from the function body and have a concise syntax that makes them ideal for writing short, one-liner functions.

5. **What are first class function?**

   First-class functions means when functions in that language are treated like any other variable. This means that functions can be assigned to variables, passed as arguments to other functions, and returned from functions.

Day 33

1. **What is a pure function?**

   A pure function is a function that, given the same input, will always return the same output and does not have any observable side effect.

2. What is the scope chain in JavaScript?

The scope chain is how Javascript looks for variables. When looking for variables through the nested scope, the inner scope first looks at its own scope.

3. What is a higher order function?

A higher-order function is a function that takes one or more functions as arguments and/or returns a function as its result.

4. What is hoisting?

Hoisting is a behavior in JavaScript where variable and function declarations are moved to the top of their respective scopes during compilation or interpretation, before the code is actually executed. This means that you can use a variable or function before it has been declared, but only if it is declared using the `var` or `function` keywords. However, only the declarations themselves are hoisted, not their values or assignments.

5. What are modules in javascript?

Modules are a way of organizing code into separate files or components that can be reused in different parts of an application. Modules allow you to encapsulate data and functionality, making your code more organized and easier to maintain

Day 34

1. What is a closure in JavaScript?

In JavaScript, a closure is created when a function is defined inside another function and the inner function is returned from the outer function. The inner function has access to the variables in the outer function, even after the outer function has returned.

```
function outer() {
  var name = "John";
  function inner() {
    console.log("Hello " + name);
  }
  return inner;
}
```

```
  var greeting = outer();
  greeting(); // Output: "Hello John"
```

2. ### What is callback in JavaScript?

   A callback is a function that is passed as an argument to another function and is intended to be called when the first function has completed its task. The primary use of callbacks in JavaScript is to handle asynchronous operations, such as making an AJAX request or waiting for a user to click a button.

3. ### What is a callback hell in javascript?

   Callback hell is a term used to describe a situation where multiple callbacks are nested within one another, making the code difficult to read, debug, and maintain. It often arises when dealing with asynchronous operations, such as making HTTP requests or working with databases.

4. ### What is memoization?

   Memoization is a technique used in computer science to speed up the execution of functions by caching the results of expensive function calls and returning the cached result when the same inputs occur again.

5. ### What is the purpose of the “use strict” statement in JavaScript?

   The “use strict” statement is used to enable strict mode in JavaScript, which helps to prevent common errors and make the code more secure. It prevents things like use of undeclared variable, use of keywords as variable name, using duplicate property names in objects, etc.

Day 35

1. ### What is a cookie in javascript?

   A cookie is a small data file that a website stores on a user’s computer or device. Cookies are commonly used to remember user preferences and login information, and to track user activity on a website. Cookies can also have an expiration date, after which they are automatically deleted.

2. ### What are the differences between cookie, local storage and session storage?

Feature	Cookie	Local storage	Session storage
Accessed on client or server side	Both server-side & client-side	client-side only	client-side only
Lifetime	As configured using Expires option	until deleted	until tab is closed
SSL support	Supported	Not supported	Not supported
Maximum data size	4KB	5 MB	5MB

3. **What is AJAX?**

   AJAX (Asynchronous JavaScript and XML) is a technique to create more dynamic and interactive web pages. It allows a web page to update content without requiring the page to reload. With AJAX, data is sent to and from the server in the background, using JavaScript and other data formats like JSON. This makes web applications more seamless and responsive, providing users with a faster and more engaging browsing experience.

4. **What is the difference between synchronous and asynchronous code in JavaScript?**

   Synchronous code executes tasks in sequence and waits for each task to complete before moving on, while asynchronous code can execute multiple tasks simultaneously and doesn’t wait for them to complete before moving on to the next task.

5. **What are promises in JavaScript?**

   Promises in JavaScript are a way of handling async operations. They help us write async code that looks and behaves like sync code, making it easier to read and maintain. Promises have three states: pending, fulfilled, and rejected.

Day 36

1. **Who created Javascript?**

   JavaScript was created by Brendan Eich in 1995.

2. **What is the difference between async/await and promises in JavaScript?**

Both async/await and Promises are used to handle asynchronous operations in JavaScript. However, async/await is built on top of Promises which makes asynchronous code more readable, easier to write and reason about.

3. **How do you handle errors in javascript?**

   In JavaScript, errors can be handled using try-catch blocks. The code that might generate an error is enclosed in a try block, and if an error occurs, the catch block is executed. The catch block can then handle the error, such as by logging it to the console or displaying an error message to the user.

4. **What is NaN in javascript?**

   NaN (Not A Number) is a special value in JavaScript that represents a situation where a value is not a valid number. One important thing to note is that NaN is not equal to any value, including itself. We can use the `isNaN()` function to check whether a value is NaN or not.

5. **How do you find operating system details in javascript?**

   In JavaScript, you can find the operating system details of the user by accessing the `navigator` object, which contains information about the user's browser and environment.

Day 37

1. **What is the Document Object Model (DOM)?**

   The DOM (Document Object Model) is a programming interface that represents the structure and content of an HTML document as a tree-like structure of nodes. It allows developers to access and manipulate the content and structure of a web page using programming languages like JavaScript.

2. **What is the difference between the DOM and HTML?**

   HTML is a markup language used to define the structure and content of a web page, while the DOM is an interface that represents that structure and content as a tree-like structure. The DOM provides a way to access and manipulate the content and structure of a web page, while HTML is simply a static markup language.

3. **What is the difference between the DOMContentLoaded event and the load event?**

The DOMContentLoaded event is fired when the initial HTML document has been completely loaded and parsed, while the load event is fired when all resources on the page, including images and scripts, have finished loading.

4. **What is the difference between innerHTML and innerText?**

   The main difference between `innerText` and `innerHTML` in the DOM is that `innerText` returns only the visible text content of an element, excluding any HTML tags, while `innerHTML` returns the complete HTML content of an element, including any nested elements and tags.

5. **What is the role of the Window object in the DOM?**

   The Window object in the DOM represents the browser window or tab that displays the web page. It provides methods and properties for controlling and manipulating the browser window.

Day 38

1. **What is a DOM node in JavaScript?**

   A node in the DOM is a fundamental unit that represents an element, attribute, or text content in a web page. Every node has a relationship with other nodes, such as a parent, child, or sibling. The parent node contains the child nodes, and the child nodes can have siblings that share the same parent

2. **How do you prevent default behavior of an event in the DOM using JavaScript?**

   To prevent the default behavior of an event in the DOM using JavaScript, you can use the preventDefault() method. This method is called on the event object that is passed to the event handler function

3. **What is event propagation?**

   Event propagation in the DOM refers to how events move or flow through different elements on a webpage. When an event happens on an element, like a click, it can travel to its parent elements and eventually to the whole document. This is called event bubbling. Alternatively, events can also travel from the document to the element that triggered the event, which is called event capturing.

4. **What is call stack in javascript?**

The call stack in JavaScript is a data structure that stores information about the currently executing functions. When a function is called, a new frame is added to the top of the stack, and when the function completes, its frame is removed from the stack. This helps the JavaScript engine keep track of where it is in the execution of a script and manage the order in which functions are called.

5. **What is an event loop?**

   The event loop in JavaScript handles asynchronous operations by queuing them up and processing them one by one in a non-blocking way. It checks the event queue continuously and processes the oldest operation first. When an operation is completed, its callback function is executed.

Day 39

1. **What is BOM?**

   BOM stands for Browser Object Model. It is a set of APIs provided by the browser that allow JavaScript to interact with the browser window.

2. **What is the use of `setTimeOut()` in javascript?**

   `setTimeout()` is a built-in function in JavaScript that allows you to schedule a function to be executed after a specified amount of time has elapsed.

3. **What is the use of `setInterval()` in javascript?**

   `setInterval()` is a function in JavaScript that allows you to repeatedly execute a given function at a specified interval. It works by calling the function repeatedly with a specified time delay between each call, until the interval is cancelled.

4. **What is the purpose of clearTimeout method and clearInterval?**

   The clearTimeout method is a built-in function in JavaScript that is used to cancel a timer created by the setTimeout function and clearInterval method is a built-in function in JavaScript that is used to cancel a recurring timer created by the setInterval function.

5. **How do you redirect new page in javascript?**

   To redirect to a new page using JavaScript, you can use the `window.location` object's `assign` or `replace` methods. Example:

```
// Redirect to a new page
window.location.assign("https://www.example.com");

// Redirect to a new page and replace the current page in the browser history
window.location.replace("https://www.example.com");
```

Day 40

1. **What is a JavaScript object?**

   JavaScript object is a non-primitive data-type that allows you to store multiple collections of data. It is a container of key-value pairs in which value may be a variable, function or object itself.

2. **What is the difference between dot notation and bracket notation when accessing properties of an object?**

   Dot Notation only allows static keys while Bracket Notation accepts dynamic keys. Static key here means that the key is typed directly, while Dynamic key here means that the key is evaluated from an expression.

3. **What is an object literal in javascript?**

   Object literal is a syntax for creating object in javascript in which property and method are inside of curly braces separated by comma. We assign a variable to an object in object literal.

4. **How would you clone an object in JavaScript?**

   There are four ways to clone an object in javascript. They are:

   i. Use the spread operator.
   ii. Call the Object.assign() function.
   iii. Use JSON parsing.
   iv. Use the structuredClone() function.

   ```
   const data = { name: "Alice", age: 26 }
   // 1
   const copy1 = { ...data }
   // 2
   const copy2 = Object.assign({}, data)
   // 3
   ```

```
const copy3 = JSON.parse(JSON.stringify(data))
// 4
const copy4 = structuredClone(data)
```

5. **What is a JSON?**

   JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications .

Day 41

1. **What is Class in JavaScript?**

   Class is a template that can be used to create objects that share the same properties and methods. When an object is created from a class, it is called an instance of that class. Class was introduced in ECMAScript(ES6).

2. **What is a constructor in javascript?**

   In JavaScript, a constructor is a special method that is used to create and initialize objects that are based on a class. It's like a blueprint for creating new objects.The constructor method is called automatically when a new object is created from a class, using the new keyword. It's used to set the initial state of the object by assigning values to its properties.

3. **What is the difference between a static method and an instance method in a class?**

   An instance method is a method that can be called on an instance of a class, and it can access and modify instance-specific data, like properties of the object. A static method, on the other hand, is a method that belongs to the class itself, not to any instance of the class. It can only access class-level data and can be called on the class itself, rather than on an instance of the class.

4. **What is "this" in javascript Classes?**

   In JavaScript classes, `this` refers to the current object that is being worked on. It's like a placeholder for the object. For example, if you have a class that creates `Person` objects, and you want to give each person a `name` , you can use `this.name` to refer to the `name` property of the current `Person` object that is being created or accessed. So, `this` is just a way to access the current object's properties and methods inside a class.

5. **What are the benefits of using classes in JavaScript?**

   There are several benefits of using classes in javascript. Some of them are:

   i. Encapsulation: Classes allow you to encapsulate related data and behavior into a single object, making it easier to manage and organize your code.

   ii. Inheritance: Classes support inheritance, which allows you to create subclasses that inherit properties and methods from a parent class. This can help you avoid duplicating code and make your code more modular and reusable.

   iii. Code Reusability: With classes, you can create objects that share common properties and behavior. This can help you avoid writing the same code over and over again, making your code more efficient and easier to maintain.

   iv. Readability: Classes provide a clean and organized way to structure your code, which can make it easier to read and understand.

Day 42

1. **Can you explain the concept of method overriding in a class in JavaScript?**

   Method overriding is a concept in JavaScript where a subclass can provide its own implementation of a method that is already defined in the parent class. To override a method in a subclass, you simply define a method with the same name as the method in the parent class

2. **What is composition in classes in JavaScript?**

   Composition in class JS is a technique of building complex classes by combining smaller, more focused classes that represent specific behaviors or properties.Composition is a flexible and powerful technique for creating modular, reusable code in JavaScript.

3. **What is inheritance in classes in javascript?**

   Inheritance in classes in JavaScript is the ability to create a new class based on an existing class. The new class inherits all the properties and methods of the existing class, and can also add new properties and methods or override existing ones.

4. **What is the `extends` keyword in JavaScript, and how is it used for inheritance?**

The `extends` keyword is used in JavaScript to create a new class that inherits from an existing class. It is used in the class declaration syntax, like this:

```
class ChildClass extends ParentClass {
 // ChildClass methods and properties
}
```

5. **what is the purpose of `super()` in javascript classes?**

   The `super()` keyword in JavaScript is used to call a method or constructor of a parent class from within a subclass. It allows a subclass to inherit and use functionality from the parent class, while also adding its own functionality.

Day 43

1. **What is a private class field in JavaScript?**

   A private class field in JavaScript refers to a class field that is only accessible within the class in which it is defined. It cannot be accessed or modified from outside the class, not even by instances of the class.

2. **Can you explain the concept of encapsulation in JavaScript classes?**

   Encapsulation is a concept in object-oriented programming that refers to bundling data and methods within a single unit, such as a class, and hiding the internal details of the class from the outside world. This makes the code more secure and maintainable.

3. **What is `get` keyword in javascript classes?**

   The `get` keyword is used to define a method that retrieves the value of a property. When the property is accessed, the `get` method is automatically called, and its return value is used as the property’s value.

4. **What is `set` keyword in javascript classes?**

   The `set` keyword is used to define a method that sets the value of a property. When the property is assigned a value, the `set` method is automatically called, and it can perform any necessary validation or processing before setting the property’s value.

5. **Can you explain the concept of instance variables in a class in JavaScript?**

In JavaScript, instance variables are properties of an object that are specific to an instance of a class. When we create a new instance of a class using the `new` keyword, each instance has its own set of instance variables that are separate from other instances.

Day 44

1. **What is the difference between a class and a function in JavaScript?**

   functions and classes are both important tools in JavaScript for defining reusable code, but they serve different purposes. Functions are used to encapsulate logic and perform specific tasks, while classes are used to create objects with shared properties and methods. Knowing when to use each one depends on the specific problem being solved and the design of the application.

2. **What is abstract class in javascript?**

   In JavaScript, an abstract class is a blueprint for creating other classes that share some common properties and methods. However, unlike regular classes, abstract classes cannot be directly instantiated. Instead, they are meant to be extended or subclassed by other classes.

3. **How would you convert an object to a JSON string in JavaScript, and vice versa?**

   In JavaScript, you can convert an object to a JSON string using the `JSON.stringify()` method, and you can convert a JSON string back to an object using the `JSON.parse()` method.

4. **What is the difference between a class and an interface in JavaScript?**

   Classes and interfaces are both used in JavaScript to define object types, but serve different purposes. A class defines a blueprint for creating objects that have properties and methods, while an interface describes the shape of an object and enforces a contract between different parts of a program. Classes define what an object is, while interfaces define what an object can do.

5. **Can you explain the concept of polymorphism in classes in JavaScript?**

   Polymorphism in JavaScript classes means that different objects can share the same methods, even if they belong to different classes. This allows us to reuse code across multiple classes and write more flexible, maintainable code.

Day 45

1. What is prototype in javascript?

In JavaScript, a prototype is an object that contains properties and methods that can be shared by all objects created with the same constructor function. It helps to reduce code duplication and makes your code more efficient.

2. What is prototype chain?

Every object in JavaScript has a built-in property, which is called its prototype. The prototype is itself an object, so the prototype will have its own prototype, making what’s called a prototype chain. The chain ends when we reach a prototype that has null for its own prototype.

3. How does prototypal inheritance work in JavaScript?

Prototypal inheritance allows objects to inherit properties and methods from their parent objects. When an object is created with a constructor function, its prototype is automatically set to the prototype object associated with that constructor function. Any properties or methods defined in the prototype object are shared by all objects created with that constructor function. When an object tries to access a property or method, JavaScript first looks for it in the object itself. If it’s not found, it looks up the prototype chain until it finds the property or method.

4. What is the difference between prototypal inheritance and classical inheritance?

The main difference between prototypal and classical inheritance is that prototypal inheritance allows objects to inherit properties and methods directly from other objects, without the need for classes or constructors. This makes the code more flexible and easier to maintain. Classical inheritance relies on classes and constructors to define the inheritance hierarchy, which can provide better organization and structure but is more rigid and requires more upfront planning.

5. What is the difference between an object’s prototype and its constructor function?

| Prototype | Constructor Function | |
——————————————————————————— |
————————————————————————- | | An object that is shared by all instances created by the constructor function | A function that is used to create new objects | | Used to define properties and methods that are shared by all instances | Used to define properties and methods that are unique to each instance | | Accessed using the `prototype` property of the constructor function | Accessed using the `new` keyword followed by the constructor function | | Modifying the prototype affects all instances created by the constructor function | Modifying the constructor function does not affect existing instances |

Day 46

1. **How do you add properties and methods to an object's prototype in JavaScript?**

   We can add properties and methods to an object's prototype by using the constructor function's prototype property. To add a property, simply assign a value to a property on the prototype object. To add a method, define a function and assign it to a property on the prototype object.

   ```
   MyConstructor.prototype.myProperty = "some value";
   ```

2. **What is the difference between `Object.prototype` and `Object.__proto__` in JavaScript?**

   In other words, `Object.prototype` is the object that provides default properties and methods that all objects in JavaScript inherit from. On the other hand, `Object.__proto__` is the object that the `Object` constructor itself inherits from, and it provides the properties and methods that are specific to the `Object` constructor.

3. **How do you check if an object inherits from a specific prototype in JavaScript?**

   We can check if an object inherits from a specific prototype by using the `isPrototypeOf()` method. This method can be called on a prototype object to check if it appears anywhere in the prototype chain of another object. If the prototype object does appear in the prototype chain of the other object, `isPrototypeOf()` will return `true`. Otherwise, it will return `false`.

   ```
   // Check if person inherits from the Object.prototype
   console.log(Object.prototype.isPrototypeOf(person)); // Outputs true if it inherits or e
   ```

4. **How do you override a method in an object's prototype in JavaScript?**

   We can override a method in an object's prototype by redefining the method on the prototype. To do this, you simply assign a new function to the existing property on the prototype. When you do this, any objects that were created using the constructor function whose prototype you are modifying will now have the new version of the method available to them.

5. What is the difference between Object.create() and new Object() in JavaScript?

The main difference between `new Object()` and `Object.create()` is that `new Object()` creates a new object from scratch, while `Object.create()` creates a new object that inherits from an existing object.

Day 47

1. What is a regular expression(regex)?

A regular expression, or regex for short, is a set of characters that form a pattern. This pattern is used to search for and match specific sequences of text.

2. What is the difference between a regular expression and a string?

A regular expression is a pattern used to match against a string. A string is simply a sequence of characters.

3. What is the syntax for creating a regular expression pattern?

The syntax for creating a regular expression pattern consists of a combination of characters, special characters, and operators that define the pattern to match.

```
// Define the regex pattern
const pattern = /\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b/;
```

4. What is a character class in regular expressions?

A character class in regular expressions is a set of characters that can be matched in a single position in the text. It is denoted by enclosing the characters in square brackets [].

```
// Define the regex pattern using a character class
const pattern = /[aeiou]/;
```

5. What is the difference between the asterisk () and the plus sign (+) in regular expressions?

In regular expressions, the asterisk (*) matches zero or more occurrences of the preceding character, while the plus sign (+) matches one or more occurrences of the preceding character.

For example, if we want to match the letter “a” followed by zero or more “b” characters, we would use the asterisk in our regular expression like this: /ab*/. This would match strings like “a”, “ab”, “abb”, “abbb”, and so on.

On the other hand, if we want to match the letter “a” followed by one or more “b” characters, we would use the plus sign in our regular expression like this: /ab+/

Day 48

1. **How do you use the question mark (?) in regular expressions?**

   The question mark (?) is a metacharacter used in regular expressions to indicate that the preceding character or group of characters is optional. It means that the preceding character or group of characters may appear zero or one time. For example, the regular expression “colou?r” will match both “color” and “colour”.

2. **What is the purpose of backslashes () in regular expressions?**

   Backslashes () are used in regular expressions to indicate that the following character has a special meaning. For example, the regular expression “\d” matches any digit character, while the regular expression “\s” matches any whitespace character. If you want to match a literal backslash character, you need to escape it by using two backslashes ().

3. **How do you specify a range of characters in a character class?**

   In a character class, you can specify a range of characters by using a hyphen (-) between two characters. For example, the regular expression “[a-z]” matches any lowercase letter from “a” to “z”. Similarly, the regular expression “[0-9]” matches any digit character from “0” to “9”. Note that the range is inclusive, so the characters at both ends are included in the match.

4. **What is the difference between a greedy and a non-greedy match in regular expressions?**

   In regular expressions, a greedy match will match as much as possible while still allowing the overall pattern to match. A non-greedy match, on the other hand, will match as little as possible while still allowing the overall pattern to match. Greedy matching is the default behavior in most regex engines. To make a match non-greedy, you can use the question mark (?) after the quantifier. For example, the regular expression “.*?” will match as few characters as possible.

5. **How do you use the pipe (|) operator in regular expressions?**

The pipe (|) operator is used in regular expressions to match either one pattern or another. For example, the regular expression “cat|dog” will match either “cat” or “dog”. You can also use parentheses to group patterns together when using the pipe operator. For example, the regular expression “(red|green|blue) car” will match “red car”, “green car”, or “blue car”.

Day 49

1. **What is the purpose of the caret (^) and dollar sign ($) characters in regular expressions?**

   The caret (^) and dollar sign ($) characters in regular expressions are used to match the beginning and end of a string, respectively.

2. **What are some common use cases for regular expressions?**

   Common use cases for regular expressions include text parsing, search and replace operations, and input validation.

3. **How do you match a specific number of characters in a regular expression?**

   To match a specific number of characters in a regular expression, you can use quantifiers such as {n} to match exactly n occurrences of a pattern, or {n,m} to match between n and m occurrences.

4. **How do you match a specific character that has a special meaning in a regular expression?**

   To match a specific character that has a special meaning in a regular expression, you can use an escape character () before the special character.

5. **How do you use lookarounds in regular expressions?**

   Lookarounds in regular expressions allow you to look ahead or behind the current position in the string without including the matched text in the result. Positive lookaheads (?=) and negative lookaheads (?!), as well as positive lookbehinds (?<=) and negative lookbehinds (?<!), are the four types of lookarounds that can be used.

Day 50

1. What is the `window.location` object in JavaScript?

The `window.location` object is a built-in object in JavaScript that contains information about the current URL of the webpage. It is a property of the global `window` object and provides several properties and methods to work with URLs.

2. What are the properties of the `window.location` object?

Some of the properties of the window.location object are:

`href` : returns the entire URL of the current page
`protocol` : returns the protocol of the URL (http:, https:, etc.)
`host` : returns the hostname and port number of the URL
`hostname` : returns the hostname of the URL
`port` : returns the port number of the URL
`pathname` : returns the path and filename of the URL
`search` : returns the query string of the URL
`hash` : returns the anchor part of the URL

3. How do you redirect to another page using JavaScript's `window.location` object?

We can redirect to another page using the `assign()` method of the `window.location` object.

```
      window.location.assign("https://www.google.com");
```

4. How do you reload the current page using JavaScript's `window.location` object?

You can reload the current page using the reload() method of the window.location object.

```
window.location.reload();
```

5. How do you get the value of a query parameter from the URL using JavaScript's `window.location` object?

You can get the value of a query parameter from the URL using the `searchParams` property of the `window.location` object. For example, to get the value of a query parameter named `id` .

```
const id = new URLSearchParams(window.location.search).get("id");
```

Day 51

1. ### What is the Date object in JavaScript?

   The Date object in JavaScript represents a date and time value, which can be used to perform various operations on dates and times.

2. ### How do you format a date in JavaScript?

   You can format a date in JavaScript using the toLocaleDateString() method of the Date object, which returns a string representation of the date in the specified locale.

   ```
   let date = new Date();
   let formattedDate = date.toLocaleDateString('en-US', { month: '2-digit', day: '2-digit',
   console.log(formattedDate); // Output: "05/14/2023"
   ```

3. ### How do you compare two dates in JavaScript?

   You can compare two dates in JavaScript using the <, >, <=, >=, ==, and != operators, which compare the numeric values of the Date objects (i.e., their timestamps).

   ```
   let date1 = new Date('2023-05-14');
   let date2 = new Date('2023-05-15');

   if (date1 < date2) {
     console.log('date1 is earlier than date2');
   } else {
     console.log('date1 and date2 are equal');
   }
   ```

4. ### How do you get the current timestamp in JavaScript?

   You can get the current timestamp in JavaScript using the `getTime()` method of the Date object, which returns the number of milliseconds since January 1, 1970, 00:00:00 UTC.

5. ### How do you add or subtract days to a date in JavaScript?

You can add or subtract days to a date in JavaScript using the `setDate()` method of the Date object, which allows you to set the day of the month for a given date.

```
let date = new Date();
date.setDate(date.getDate() + 3);
console.log(date); // Output: the date 3 days from now
```

Day 52

1. **what is iterator in javascript?**

   In JavaScript, an iterator is an object that provides a way to access elements of a collection or a custom data structure in a sequential manner. It allows you to loop over the elements one at a time, retrieving them on demand.

   The most important method is next(), which is responsible for returning the next element in the sequence. When you call next() on an iterator, it returns an object with two properties: value, representing the current element, and done, indicating whether there are more elements or if the iteration is complete.

2. **What is decorator in javascript?**

   In JavaScript, a decorator is a design pattern that allows you to modify the behavior of an object or a function by wrapping it with another function. It provides a way to add new functionality or modify existing functionality dynamically, without changing the original code.

3. **What is babel?**

   Babel is a tool that helps you write modern JavaScript code while ensuring compatibility with older environments by transforming or transpiling that code into an older JavaScript version. It allows developers to take advantage of new language features without worrying about browser compatibility issues.

4. **What is optional chaining?**

   Optional chaining is a js feature by which using the `?.` operator, you can directly access nested properties or call nested functions, and if any part of the chain is null or undefined, the expression short-circuits and returns undefined instead of throwing an error.

5. **What is throttling?**

Throttling is a technique used in JavaScript to control the rate at which a particular function or code block is executed. It ensures that the function is called at a maximum frequency or a specified interval, preventing it from being invoked too frequently.

Day 53

1. **What is debouncing?**

   Debouncing in JavaScript is a way to control how often a function gets called when there are frequent events happening, like typing or scrolling. This can be useful for optimizing performance and avoiding unnecessary function calls.

2. **What is global execution context?**

   The global execution context is the default or first execution context that is created by the JavaScript engine before any code is executed(i.e, when the file first loads in the browser). All the global code that is not inside a function or object will be executed inside this global execution context. Since JS engine is single threaded there will be only one global environment and there will be only one global execution context.

3. **What is function execution context?**

   In JavaScript, the function execution context refers to the environment in which a function is executed or called. Each time a function is invoked, a new execution context is created specifically for that function. It consists of two main components: the variable environment and the scope chain.

4. **What does the variable environment in JavaScript’s function execution context contain, and what is its purpose?**

   The variable environment contains all the variables, function declarations, and function arguments specific to that function. It keeps track of the function’s local variables and parameters, allowing the function to access and manipulate them during its execution.

5. **What is a scope chain in JavaScript’s function execution context?**

   The scope chain is a list of all the variable environments that are accessible to the function. It is used to resolve variable references during the function’s execution. When a variable is not found in the current variable environment, JavaScript looks up the scope chain to find the variable in outer environments until it reaches the global execution context.

Day 54

1. **What is Minification?**

   Minification is the process of removing all unnecessary characters(empty spaces are removed) and variables will be renamed without changing it’s functionality.

2. **How do I modify the url without reloading the page?**

   The `window.location.href` property will be helpful to modify the url but it reloads the page. HTML5 introduced the `history.pushState()` and `history.replaceState()` methods, which allow you to add and modify history entries, respectively. Example:

   ```
   window.history.pushState("page2", "Title", "/page2.html");
   ```

3. **What are dynamic imports in javascript?**

   Dynamic imports in JavaScript allow you to load modules or scripts dynamically at runtime, instead of including them statically in the initial page load.

   ```
       import('module.js')
   .then(module => {
     // Use the imported module
     // ...
   })
   .catch(error => {
     // Handle any import errors
     // ...
   });
   ```

4. **What paradigm is Javascript?**

   JavaScript is a multi-paradigm language, supporting imperative/procedural programming, Object-Oriented Programming and functional programming. JavaScript supports Object-Oriented Programming with prototypical inheritance.

5. **How do you empty an array?**

   You can empty an array quickly by setting the array length to zero.

```
let cities = ["Singapore", "Delhi", "London"];
cities.length = 0; // cities becomes []
```

Day 55

1. **What is a lambda function?**

   A lambda function, also known as an arrow function in JavaScript, is a concise and shorthand way of defining a function. It uses the => arrow syntax to indicate a function, allowing for shorter and more readable code.

2. **What is variable shadowing in javascript?**

   If there is a variable in the global scope, and you'd like to create a variable with the same name in a function. The variable in the inner scope will temporarily shadow the variable in the outer scope. It is called variable shadowing.

3. **How do you assign default values to variables?**

You can use the logical or operator		in an assignment expression to provide a default value. The syntax looks like as below,

```
let a = b || c;
```

4. **What is a rest operator in javascript?**

   The rest operator in JavaScript is a special syntax that allows you to pass an indefinite number of arguments to a function. It is represented by three dots (...).

```
function partyGuests(...names) {
console.log(names);
}

partyGuests('Alice', 'Bob', 'Charlie');
```

5. **What is a spread operator?**

The spread operator (`...`) is used to spread elements from an array or object into another array, function call, or object literal. It allows you to unpack or expand the individual items or properties from the source into a destination.

```
const fruits = ['apple', 'banana', 'orange'];
const fruitSalad = ['kiwi', 'grape', ...fruits, 'melon'];
//fruitSalad = ["kiwi", "grape", "apple", "banana", "orange", "melon"]
```

Day 56

1. **Why would you use something like the load event? Does this event have disadvantages? Do you know any alternatives, and why would you use those?**

   The `load` event fires at the end of the document loading process. At this point, all of the objects in the document are in the DOM, and all the images, scripts, links and sub-frames have finished loading.

   The DOM event `DOMContentLoaded` will fire after the DOM for the page has been constructed, but do not wait for other resources to finish loading. This is preferred in certain cases when you do not need the full page to be loaded before initializing.

2. **What’s the difference between Native objects and Host objects?**

   Native objects are objects that are part of the JavaScript language defined by the ECMAScript specification, such as String, Math, RegExp, Object, Function, etc on the other hand, the host objects are provided by the runtime environment (browser or Node), such as window, XMLHTTPRequest, etc.

3. **What’s a typical use case for anonymous functions?**

   Anonymous functions are functions that are not bound to a name. They are often used as inline functions, or as arguments to other functions. One typical use case for anonymous functions is as callback functions.

4. **What is negative Infinity?**

   Negative Infinity is a number in JavaScript which can be derived by dividing negative number by zero.

5. **What is the data type of variables in JavaScript?**

   All variables in JavaScript are object data types.

Day 57

1. **What is the difference between a prototype and an instance?**

   A prototype is a blueprint for creating objects. An instance is an object that is created from a prototype. Instances inherit properties and methods from their prototypes.

2. **What is a function expression?**

   A function expression in JavaScript is a way to define a function by assigning it to a variable. Instead of using the traditional function declaration syntax, a function expression involves creating an anonymous function that can be stored in a variable.

3. **What is the difference between a module and a library?**

   A module is a self-contained unit of code that can be imported into another program. A library is a collection of modules that can be used to perform a specific task. Modules are typically used to organize code and make it easier to reuse, while libraries are typically used to provide functionality that is not available in the core language.

4. **How are JavaScript and ECMA Script related?**

   ECMA Script is like rules and guidelines, while Javascript is a scripting language used for web development.

5. **What are the different ways to delete a variable in JavaScript?**

   We can delete a variable and remove it from memory in the following ways:

   i. Using the `delete` keyword:

   ```
   let x = 10;
   console.log(x); // Output: 10

   delete x;
   console.log(x); // Output: 10 (variable still exists but with no value)
   ```

   ii. Setting the variable to `undefined` or `null` :

```
let x = 10;
console.log(x); // Output: 10

x = undefined;
console.log(x); // Output: undefined
```

iii. Using a block scope with `let` or `const` :

```
//Variables declared with let or const within a block scope will automatically be re
// from memory once the block is exited.
{
let x = 10;
console.log(x); // Output: 10
}

console.log(x); // Output: ReferenceError: x is not defined
```

Day 58

1. **What are the different types of errors in JavaScript?**

   There are three types of errors:

   i. Load time errors: Errors that come up when loading a web page, like improper syntax errors, are known as Load time errors and generate the errors dynamically.
   ii. Runtime errors: Errors that come due to misuse of the command inside the HTML language.
   iii. Logical errors: These are the errors that occur due to the bad logic performed on a function with a different operation.

2. **What is the use of the blur function?**

   In JavaScript, the `blur()` function is used to remove the focus from a specific element on a web page. When an element has focus, it typically means that it is selected or ready to receive user input, such as when a user clicks on an input field or a button.

3. **What is the difference between an alert box and a confirmation box?**

   An alert box is a simple message box with an OK button for displaying information, while a confirmation box allows users to confirm or cancel an action with OK and Cancel buttons respectively. The alert box is non-interactive and pauses code execution until closed, whereas the confirmation box returns a boolean value indicating the user's choice and doesn't halt code execution.

4. ### What is `prompt()` in javascript?

   `prompt()` is a built-in JavaScript function that displays a dialog box to the user with a message, an input field, and OK/Cancel buttons. It allows the user to input data, which can then be captured and used in JavaScript code. The `prompt()` function halts the code execution until the user enters a value and clicks OK or cancels the dialog. If the user clicks OK, the entered value is returned as a string. If the user cancels or closes the dialog, the function returns null.

5. ### When to Use Internal and External JavaScript Code?

   If you have only a few lines of code that is specific to a particular webpage. In that case, it is better to keep your JavaScript code internal within your HTML document. On the other hand, if your JavaScript code is used in many web pages, you should consider keeping your code in a separate file.If your code is too long, it is better to keep it in a separate file. This helps in easy debugging.

Day 59

1. ### What is npm?

   NPM(Node Package Manager) is a helpful tool for developers that makes working with JavaScript easier. It's like a big library where they can find ready-to-use code and easily add it to their own projects. It is also the name of the command line package manager used to interact with npm.

2. ### What is the name of the file which npm uses to identify the project and its dependencies?

   The file that npm uses to identify the project and its dependencies is called "package.json". It serves as a configuration file where developers can specify information about their project, such as its name, version, and dependencies on external packages.

3. ### What is the difference between dependencies and devDependencies?

   Both are defined in the `package.json`. dependencies lists the packages that the project is dependent on. devDependencies lists the dependencies which are only required during testing and development.

4. ### What is a non-blocking function?

   A non-blocking function, also known as an asynchronous function, is a type of function that does not block the execution of other code while it is running. Instead of waiting for the function to complete before moving on to the next task, non-blocking functions allow the program to continue executing other tasks while it performs its operation in the background.

5. ### What is a blocking function?

   A blocking function is a type of function that halts the execution of the program until it completes its task or receives a response.

Day 60

1. ### What is Function Composition?

   Function composition in JavaScript refers to the process of combining two or more functions to create a new function. To compose functions, you typically create a higher-order function that takes two functions as arguments and returns a new function. This new function executes the functions in order, passing the result of one function as the argument to the next function.

2. ### What is heap?

   In JavaScript, the “heap” refers to a part of the memory where dynamic memory allocation takes place. It is the memory region used for storing objects and variables that are created during the execution of a program. The heap is an essential component of the JavaScript runtime environment. It is responsible for managing and allocating memory for objects that are created dynamically at runtime, such as objects created using the new keyword, arrays, and function closures.

3. ### What is typed array in javascript?

   Typed Arrays are specialized array-like objects that allow you to work with binary data in a structured and efficient manner. They come in different types, such as numbers and bytes, and provide optimized operations for reading, writing, and manipulating binary data. They offer better performance and memory efficiency compared to regular arrays.

4. ### What is nodejs?

   Node.js is a JavaScript runtime for server-side applications. It lets you run JavaScript outside of web browsers, handle network requests, access databases, and build scalable apps efficiently.

It's widely used for creating web servers, APIs, real-time apps, and command-line tools.

5. **What is a PWA?**

   PWAs (Progressive Web Apps) are web applications that use JavaScript, HTML, and CSS to provide a mobile app-like experience. They work offline, send push notifications, and can be installed on devices. PWAs combine the best of web and app technologies, allowing users to access them directly through web browsers without the need for app store downloads.

Day 61

1. **What is nullish coalescing operator (??)?**

   It is a logical operator that returns its right-hand side operand when its left-hand side operand is null or undefined, and otherwise returns its left-hand side operand. This can be contrasted with the logical OR (||) operator, which returns the right-hand side operand if the left operand is any falsy value, not only null or undefined.

```
console.log(null ?? true); // true
```

2. **How do you get the status of a checkbox with javascript?**

   You can apply the checked property on the selected checkbox in the DOM. If the value is true it means the checkbox is checked, otherwise it is unchecked. For example, the below HTML checkbox element can be access using javascript as below:

```
console.log(document.getElementById(‘checkbox_name’).checked); // true or false
```

3. **What is the purpose of double tilde operator?**

   The double tilde operator(~~) is known as double NOT bitwise operator. This operator is a slightly quicker substitute for Math.floor().

4. **What are the different ways to debug JavaScript code?**

   To debug JavaScript code, you can use console.log() statements to print values and messages to the console, browser developer tools for breakpoints, stepping through code, and variable inspection, the debugger statement to trigger breakpoints, exception handling to catch and log errors, linters and code analyzers to detect potential issues, and remote debugging for debugging code running in a different environment.

5. **What are the different ways to optimize JavaScript code?**

   To optimize JavaScript code, you can combine and minify files, minimize global variables, optimize loops and conditionals, use efficient data structures and algorithms, cache data, leverage asynchronous programming, and optimize DOM manipulation.

Day 62

1. **What is the difference between undeclared & undefined?**

   Undeclared variables are those that do not exist in a program and are not declared. If the program tries to read the value of an undeclared variable, then a runtime error is encountered. Undefined variables are those that are declared in the program but have not been given any value. If the program tries to read the value of an undefined variable, an undefined value is returned.

2. **What would be the result of 2+5+"3"?**

   Since 2 and 5 are integers, they will be added numerically. And since 3 is a string, its concatenation will be done. So the result would be 73. The " " makes all the difference here and represents 3 as a string and not a number.

3. **What is statically typed and dynamically typed language and is javascript a statically typed or a dynamically typed language?**

   Dynamically-typed languages perform type checking at runtime, while statically typed languages perform type checking at compile time.Javascript is a dynamically typed language.

4. **What is function currying?**

   Function currying is a process in which we convert a function with multiple parameters to a chain of functions with a single parameter.

```
      // Normal function
  function sum(a, b) {
          return a + b;
  }
```

```
// Curried function
function currySum(a) {
        return function (b) {
                return a + b;
  }
}
```

5. **What does delete do in JavaScript?**

   The `delete` operator is used for deleting an object's property or key.

Day 63

1. **What is control flow function?**

   A control flow function in JavaScript refers to a function that controls the flow of execution within a program, particularly when dealing with asynchronous operations.

2. **How does control flow function in js play a role in asynchronous operation in javascript?**

   Control flow functions provide mechanisms to coordinate and handle asynchronous operations, ensuring that certain actions occur before or after others. They help to maintain the desired order and synchronization in asynchronous code.

3. **Can you access DOM in nodejs?**

   No, you cannot directly access the DOM in Node.js. Node.js is a runtime environment for running JavaScript outside of web browsers, and it does not have a built-in DOM implementation.

4. **How can you share code between files?**

   In the client-side/browser environment, if variables and functions are declared in the global scope (window), they can be accessed and shared by all scripts on the page. This is often referred to as the global scope or global namespace.

5. **What does the `instanceof` operator do?**

   The `instanceof` operator checks whether the prototype property of a constructor appears anywhere in the prototype chain of an object. In other words, the `instanceof` operator checks if the object is an instance of a class or not at run time.

```
class Person {
  constructor(name) {
    this.name = name;
  }
}

const john = new Person("John");

console.log(john instanceof Person); // Output: true
console.log(john instanceof Object); // Output: true (all objects inherit from Object)

const str = "Hello";
console.log(str instanceof String); // Output: false (str is a primitive string,
not an instance of the String constructor)
```

Day 64

1. **What are the drawbacks of prototypal inheritance?**

   Prototypal inheritance also has some drawbacks. First, it can be more difficult to understand than class-based inheritance. Second, it can be more difficult to debug. Third, it can be more difficult to test.

2. **What are some common problems that you encounter when using regular expressions?**

   When using regular expressions, common problems include incorrect pattern matching, performance issues with large data or complex patterns, difficulty in readability and maintenance, and the risk of overfitting the problem at hand.

3. **What are the limitations of using the “var” keyword in JavaScript?**

   The “var” keyword in JavaScript has limitations. Variables declared with “var” are function-scoped, accessible throughout the entire function. It lacks block-level scoping, leading to confusion and unintended side effects. Additionally, “var” doesn’t prevent variable redeclaration within the same scope, risking inadvertent overwriting.

4. **What are the limitations of using the “this” keyword in JavaScript?**

The “this” keyword in JavaScript has limitations. Its value depends on the function invocation, leading to confusion and unexpected behavior. In arrow functions, “this” behaves differently, being lexically scoped. When using “this” in a constructor function without the “new” keyword, it refers to the global object instead of creating a new instance.

5. **What are the drawbacks of using the “delete” operator in JavaScript?**

   The “delete” operator in JavaScript has drawbacks. It can be slow and impact performance when deleting object properties. It does not affect the prototype chain, leading to unexpected behavior. It also cannot delete variables or functions declared with “var” or “function” keywords.

Day 65

1. **What are the advantages of using closures in JavaScript?**

   Closures in JavaScript allow for encapsulation, data privacy, and the creation of private variables and functions that are inaccessible from the outside scope.

2. **How does the concept of prototypal inheritance benefit JavaScript developers?**

   Prototypal inheritance in JavaScript allows objects to inherit properties and methods from other objects, promoting code reuse and reducing memory consumption.

3. **How does using template literals in JavaScript improve string manipulation and concatenation compared to traditional methods?**

   Template literals in JavaScript allow for easier string interpolation, multiline strings, and dynamic content embedding, enhancing readability and reducing string manipulation complexities.

4. **What are the advantages of using the fetch API over traditional XMLHttpRequest for making HTTP requests in JavaScript?**

   The fetch API offers a simpler and more modern way to make asynchronous HTTP requests, providing better error handling, support for promises, and the ability to handle various data formats.

5. **Can you explain the benefits of using the “this” keyword and how it differs in different contexts in JavaScript?**

The “this” keyword in JavaScript allows objects to refer to their own properties and methods dynamically, adapting to different execution contexts and enabling code reusability.

Day 66

1. **What resources or techniques do you use to stay up to date with the latest developments in JavaScript?**

   To stay updated with the latest developments in JavaScript, I also make use of online learning platforms like Udemy and Coursera to enroll in courses focused on JavaScript and attend conferences or meetups whenever possible. Additionally, I find it helpful to contribute to open-source projects on GitHub, as it exposes me to different coding styles and practices.

2. **How do you ensure the code you write is maintainable, readable, and follows best practices?**

   To ensure maintainable, readable, and best practice-following code, I follow coding conventions and style guidelines. I also write clear comments, use descriptive variable and function names, and modularize the code to make it easier to understand and maintain.

3. **How do you handle working on a JavaScript project with a large codebase or multiple developers?**

   Clear communication, organization, and collaboration are essential for JavaScript projects with a large codebase or multiple developers. Using Git for version control enables simultaneous work and easy code merging. Breaking down the code into smaller modules, documenting, following coding standards, conducting code reviews, and implementing testing ensure code quality and facilitate collaboration.

4. **How does JavaScript handle memory management and garbage collection?**

   JavaScript handles memory management through automatic garbage collection. The JavaScript engine keeps track of all objects created during the program execution. When an object is no longer reachable or referenced by any part of the program, it becomes eligible for garbage collection. The garbage collector then frees up the memory occupied by those unreferenced objects, making it available for future use.

5. **According to javascript array, what comes first the chicken or egg?**

According to the JavaScript array, “chicken” comes before “egg” because “chicken” is alphabetically sorted before “egg”.

```
const arr = ["egg", "chicken"];
const sortedArr = arr.sort();

console.log(sortedArr) // ["chicken", "egg"]

//Hence proved, chicken comes first than egg
```

ReactJS

Day 67

1. **What is ReactJS?**

   React is a free and open-source front-end JavaScript library for building user interfaces based on components. It is maintained by Meta and a community of individual developers and companies

2. **What are the key features of ReactJS?**

   The key features of ReactJS are its component-based architecture, virtual DOM implementation for efficient rendering, code reusability through reusable components, declarative syntax and JSX for easier UI development, unidirectional data flow for predictable updates, and performance optimization by minimizing direct DOM manipulation.

3. **What is the difference between ReactJS and other JavaScript frameworks/libraries?**

   ReactJS differs from other JavaScript frameworks/libraries in several ways. It focuses solely on the view layer, allowing developers to integrate it easily with other libraries or frameworks. React also introduces the concept of a virtual DOM, which enhances performance by minimizing direct manipulation of the actual DOM.

4. **What is JSX in ReactJS?**

   JSX stands for JavaScript syntax extension. It is a JavaScript extension that allows us to describe React’s object tree using a syntax that resembles that of an HTML template. It is just an XML-like extension that allows us to write JavaScript that looks like markup and have it returned from a component.

5. **Explain the concept of virtual DOM in ReactJS.**

   Virtual DOM is a lightweight copy of the actual DOM maintained by React. It allows React to efficiently update and render components by calculating the optimal changes needed and applying them to the real DOM.

Day 68

1. **What are the components in ReactJS?**

   Components in ReactJS are the building blocks of the user interface. They can be thought of as reusable, self-contained pieces of code that encapsulate a specific functionality or UI element.

2. **What is the significance of state in ReactJS?**

   The state in ReactJS is a JavaScript object that stores data specific to a component. It represents the current state of the component and can be updated over time. The significance of state is that it allows components to manage and display dynamic data, enabling interactivity in the user interface.

3. **What is the difference between props and state?**

   Props (short for properties) and state are both used in ReactJS to manage data, but they serve different purposes. Props are used to pass data from a parent component to a child component, while state is used to manage and update data within a component itself.

4. **What is the purpose of the “render” method in ReactJS?**

   The “render” method in ReactJS is a crucial part of a component. It is responsible for returning the JSX (JavaScript XML) code that defines the structure and content of the component’s UI. The render method is called automatically whenever there is a change in the component’s state or props.

5. **What are the lifecycle methods in ReactJS and how do they work?**

   Lifecycle methods in ReactJS are special methods that are invoked at different stages of a component’s lifecycle. They allow developers to perform specific actions at certain points, such as initializing state, updating the UI, or cleaning up resources. Examples of lifecycle methods include “componentDidMount,” “componentDidUpdate,” and “componentWillUnmount.” They provide hooks to execute code at specific moments during the component’s lifespan.

Day 69

1. **What is a higher-order component (HOC) in ReactJS?**

   A Higher-Order Component (HOC) in ReactJS is a function that takes a component as input and returns a new enhanced component. They act as wrappers around components, enabling you to inject props, modify behavior, and reuse common logic across multiple components.

2. **What is the purpose of the “key” prop in ReactJS?**

   The “key” prop in ReactJS is used to uniquely identify elements in a list of components or elements rendered by a loop. It helps React efficiently update and re-render only the necessary components when the list changes.

3. **Explain the concept of controlled and uncontrolled components in ReactJS.**

   Controlled components in ReactJS are form inputs whose state is managed by React. The input value is bound to a state variable, and changes are handled through event handlers. Uncontrolled components, on the other hand, have their state managed by the DOM, and the input value is accessed directly. Controlled components provide more control and validation options, while uncontrolled components are simpler for basic scenarios.

4. **What is the purpose of the “setState” method in ReactJS and how does it work?**

   The “setState” method in ReactJS is used to update the state of a component. It is a built-in method provided by React’s Component class. When called, “setState” triggers a re-render of the component, updating its UI based on the new state.

5. **What is the significance of the “React.Fragment” component?**

   “React.Fragment” is a component in ReactJS that allows you to group multiple elements without adding extra DOM elements. It helps keep the rendered output clean and is particularly useful when you don’t want to introduce additional wrappers or nesting levels.

Day 70

1. **How does React handle event handling?**

   React handles event handling by using synthetic events, which are cross-browser wrappers around native browser events. When an event is triggered, React creates a synthetic event object and passes it to the event handler function.

2. **What are the differences between class components and functional components in ReactJS?**

   The main difference between class components and functional components in ReactJS is the syntax and the way they manage state. Class components are defined using ES6 classes and have lifecycle methods, while functional components are defined as JavaScript functions. Class components have their own internal state and can use lifecycle methods like componentDidMount and componentDidUpdate, while functional components use React Hooks to manage state and handle side effects.

3. **How can you optimize the performance of React applications?**

   There are several ways to optimize the performance of React applications. Some techniques include: using a production build of React, minimizing the number of re-renders by using shouldComponentUpdate or React.memo, lazy loading components or data using code splitting, implementing virtualization techniques for long lists, and optimizing network requests by caching data or using techniques like memoization.

4. **What is React Router and how does it work?**

   React Router is a popular library in React for handling routing in a web application. It allows developers to create single-page applications with multiple views or pages. React Router works by defining routes that map to different components. When a user navigates to a specific route, the corresponding component is rendered

5. **Explain the concept of React Hooks and their benefits.**

   React Hooks provide a way to reuse stateful logic without writing a class component. Hooks enable developers to manage component state, handle side effects, and tap into the React lifecycle in a more straightforward and flexible manner. They offer benefits like improved code readability, reusability, and easier testing. Some commonly used hooks include useState, useEffect, and useContext.

Day 71

1. **Can web browsers read JSX directly?**

   Web browsers cannot read JSX directly. This is because they are built to only read regular JS objects and JSX is not a regular JavaScript object. For a web browser to read a JSX file, the file needs to be transformed into a regular JavaScript object. For this, we use Babel

2. **"In React, everything is a component." Explain.**

   Components are the building blocks of a React application's UI. These components split up the entire UI into small independent and reusable pieces. Then it renders each of these components independent of each other without affecting the rest of the UI.

3. **What are Pure Components?**

   Pure components are the simplest and fastest components which can be written. They can replace any component which only has a render(). These components enhance the simplicity of the code and performance of the application.

4. **Why is it necessary to start component names with a capital letter?**

   In React, it is necessary to start component names with a capital letter. If we start the component name with lower case, it will throw an error as an unrecognized tag. It is because, in JSX, lower case tag names are considered as HTML tags.

5. **When do we prefer to use a class component over a function component?**

   If a component needs state or lifecycle methods, we should use the class component; otherwise, use the function component. However, after React 16.8, with the addition of Hooks, you could use state, lifecycle methods, and other features that were only available in the class component right in your function component.

Day 72

1. **How do you handle forms in React?**

   In React, you can handle forms by using controlled components. This means that the form inputs' values are controlled by React state. You create state variables to store the values of the form inputs and update the state using the onChange event. You can access and process the form data from the state when the form is submitted.

2. **Explain the Flux architectural pattern.**

   The Flux architectural pattern is a design pattern used in React applications for managing data flow. It consists of four key components: the view, actions, dispatcher, and stores. Actions trigger changes, which are dispatched by the dispatcher to the appropriate stores. The stores contain the application state and update the views. This unidirectional data flow ensures predictability and maintainability.

3. **Explain the concept of context in React. How does it work?**

   Context in React is a way to share data between components without explicitly passing it through props at each level. It allows you to create a context object that holds the shared data, which can be accessed by any component within its subtree. The context provider sets the value, and the consuming components can access it using the useContext hook.

4. **What are portals in React? When would you use them?**

   Portals in React provide a way to render children components into a different DOM node, outside the current component's DOM hierarchy. They allow you to render components at a different location in the DOM, which is useful for scenarios like modals, popovers, or tooltips, where the component needs to be positioned relative to a specific DOM element outside its parent.

5. **Explain the concept of lazy loading in React.**

   Lazy loading in React is a technique used to optimize performance by loading components or resources only when they are needed. Instead of loading all components upfront, you can dynamically import them using the React.lazy function and render them when required. This helps reduce the initial bundle size and improves the loading speed of your application.

Day 73

1. **What is useEffect hook?**

   The useEffect hook is utilized for handling side effects in React components. It enables you to perform tasks like data fetching, subscriptions, or DOM manipulation.

2. **What is the role of the useReducer hook in React? How does it differ from the useState hook?**

   The useReducer hook serves as an alternative to useState for managing complex state logic. It takes a reducer function and an initial state, returning the current state value and a dispatch function. By dispatching actions to the reducer, state updates can be controlled and organized in a more structured manner.

3. **Describe the purpose of the useRef hook.**

   The useRef hook is employed to create mutable values that persist across renders in a React component. It is commonly used for accessing DOM elements or storing any mutable value that doesn't need to trigger a re-render. useRef allows components to retain a reference to a specific value or element throughout their lifecycle.

4. **What is the purpose of the useMemo hook? How does it help optimize performance in React?**

   The useMemo hook is designed for memoizing expensive calculations or function calls in React. It takes a function and a dependency array, returning a memoized value. By specifying the dependencies, the memoized value is recalculated only when the dependencies change, helping to optimize performance by avoiding unnecessary computations.

5. **When would you use the useCallback hook in React? How does it differ from the useMemo hook?**

   The useCallback hook in React is like useMemo, but it memoizes functions instead of values. It's handy when passing callbacks to child components because it avoids unnecessary re-renders. By memoizing the function, it ensures that the same function instance is returned as long as the dependencies (inputs) remain unchanged. This optimization can improve performance in certain situations by preventing unnecessary function recreations.

Day 74

1. **What happens when you call setState?**

   The state property is updated in a React component with the object passed into setState, and this is done asynchronously. It tells React that this component and its children need to be re-rendered, but React may not do this immediately.

2. What is `children` prop in React?

The `children` prop in React allows you to pass content or components as a nested element to another component, making it flexible and reusable. It enables you to include and render dynamic content within a component by placing it between the opening and closing tags of that component.

3. What are stateless components?

If the behaviour of a component is independent of its state then it can be a stateless component. You can use either a function or a class for creating stateless components. But unless you need to use a lifecycle hook in your components, you should go for function components.

4. What are stateful components?

If the behaviour of a component is dependent on the state of the component then it can be termed as stateful component. These stateful components are either function components with hooks or class components.

5. What is the impact of indexes as keys?

Using indexes as keys in React can cause problems. When components are rendered using indexes as keys, React may not properly update or reorder them when the order changes. This can result in incorrect rendering, loss of component state, and slower performance. It’s better to use unique and stable identifiers as keys to avoid these issues and ensure that components are updated correctly.

Day 75

1. What is prop drilling in react?

Prop drilling in React refers to the process of passing down props (properties) from a parent component to a deeply nested child component, even if the intermediate components do not need or use those props. This can lead to a cluttered and less maintainable codebase, as props need to be passed through multiple layers of components, making it harder to understand and modify the code.

2. How can we avoid prop drilling?

Prop drilling can be avoided by using React’s Context API or state management libraries like Redux, which allow for more efficient and organized sharing of data across components.

3. **How would you prevent a component from rendering in React?**

   To prevent a component from rendering in React, you can use conditional rendering. You can wrap the component’s JSX code inside an if statement or a ternary operator, where you specify a condition. If the condition evaluates to true, the component will render; otherwise, it won’t. This allows you to control when and under what circumstances the component should be displayed. By dynamically adjusting the condition, you can easily show or hide the component based on certain logic or user interactions.

4. **What do these three dots (…) in React do?**

   The three dots (…) in React, called the spread operator, can be used to make copies of objects or arrays. With objects, it lets you create a new object by copying properties from an existing object and adding or changing specific ones. For arrays, it allows you to create a new array by combining elements from different arrays. The spread operator is handy for manipulating data in React and simplifies tasks like updating component state or merging objects with new properties.

5. **Why React uses className over class attribute?**

   React uses `className` instead of `class` in HTML because `class` is a reserved keyword in JavaScript. Using `class` would cause conflicts with JavaScript syntax. So, to avoid issues, React uses `className` for specifying CSS classes on elements. This allows developers to apply CSS classes to React components without running into conflicts or JavaScript errors.

Day 76

1. **Why we should not update state directly?**

   Updating the state directly in ReactJS is not recommended because it can cause unexpected issues and make it hard to track changes. React provides the setState() method for updating the state, which ensures proper handling of updates, triggers component re-rendering, and maintains state management integrity.

2. **What is StrictMode in React?**

   StrictMode in React is a component that helps catch potential problems in your code by performing extra checks during rendering, highlighting issues and encouraging best practices.

3. **What’s wrong with using Context in React?**

   Context in React is not inherently bad, but it should be used carefully. Using Context extensively can make your code harder to understand and maintain. It may also cause unnecessary re-rendering and may not be the best choice for managing global state in complex applications. So we should consider other state management options like Redux for larger projects.

4. **Does React re-render all components and sub components every time setState is called?**

   No, React does not re-render all components and subcomponents every time setState is called. It only re-renders the components that are affected by the state change, optimizing performance by avoiding unnecessary updates.

5. **Explain why and when would you use `useMemo()` ?**

   You would use `useMemo()` in React when you want to optimize performance by memoizing the result of a function or computation. It is used to avoid unnecessary calculations or expensive operations by caching the computed value until its dependencies change. By using `useMemo()` , you can ensure that the computation is only performed when necessary, reducing the workload on the application and improving efficiency.

Day 77

1. **When shall we use useReducer hook in ReactJS?**

   The useReducer hook in ReactJS is typically used when you have complex state logic that involves multiple related values or when the state transitions are more intricate. It is a good choice when you find that managing state with the useState hook becomes cumbersome and leads to complex and nested code.

2. **How does React renderer work exactly when we call setState?**

   When you use setState in React, it triggers a process called reconciliation. React compares the old and new state, figures out what changed, and updates only the necessary parts of the user interface. It does this by creating a virtual copy of the user interface called the virtual DOM and then efficiently applying the required changes to the actual web page. This helps React make updates quickly and keeps your app running smoothly.

3. **what are styled component?**

Styled components are a way to style React components by writing CSS directly in JavaScript. They allow you to create reusable styled elements and easily manage styles within your components. It improves code organization and makes styling more readable and maintainable.

4. **What are the advantages of styled components?**

   Styled components in React offer advantages such as component-based styling, improved readability, scoped styles, support for dynamic styles, reusability, theme support, and better performance. They enhance code organization, make styles more maintainable, and provide a seamless integration between styling and component logic.

5. **What are the advantages of styled components?**

   Styled components in React can be more complex and have a higher learning curve compared to traditional CSS stylesheets. Defining styles within JavaScript may require developers to adapt their workflow and understand CSS-in-JS concepts. Additionally, the generated class names for styled components can be less readable, making debugging more challenging

Day 78

1. **Why props cannot be updated in ReactJS?**

   props cannot be updated in ReactJS as due to React core philosophy focused on making the props as immutable and top-down thus, a parent can send any prop values to a child but child cannot change the received prop.

2. **What is a dispatcher?**

   It is one of the components required for building apps according to Flux architecture. In this pattern dispatcher is the main point which manages data exchanging inside the app.

3. **What are Default Props?**

   In React, default props are used to define default values for props in a component. Default props ensure that if a parent component doesn't provide a value for a certain prop, the component will still have a default value to work with.

4. **What are inline conditional expressions?**

   In JSX, you can include any JavaScript expression within curly braces and use it with conditional (ternary) operators, making inline conditional expressions an interesting topic for react interview questions for freshers.

5. What are the differences between React and React Native?

React is a JavaScript library used for building web user interfaces, while React Native is a framework for developing native mobile applications. React renders components to the web browser using virtual DOM, while React Native renders components directly to native UI elements for iOS and Android platforms, resulting in a more native-like experience.

Day 79

1. Is React a library or a Framework and why?

React is a JavaScript library for UI building, not a framework. It focuses on the view layer, offering efficient ways to create UI components and manage state. Unlike frameworks, React doesn’t provide a full set of development tools, but it excels at making interactive web applications with its declarative approach and efficient UI updates.

2. What are nested component in react?

In React, nested components refer to the idea of rendering components within other components. It allows for building complex user interfaces by composing smaller components together.

3. Can a nested child component access the state of its sibling component?

No, a nested child component cannot directly access the state of its sibling component. If sibling components need to communicate or share data, it is recommended to lift the shared state up to their common parent component and pass it down as props to both siblings.

4. What are error boundaries in react?

Error boundaries in React are components that prevent the entire application from crashing when an error occurs within their child components. They act as safety nets by catching and handling errors, allowing developers to display fallback UI and maintain a smoother user experience.

5. What is React Dev Tool?

React DevTools is a browser extension that helps developers debug and inspect React applications. It allows you to see the component structure, check and modify component data, and analyze performance. It's a useful tool for understanding how React components work and finding and fixing issues in your code.

Day 80

1. **What is "React Node" in react?**

   In React, a "React Node" is a term used to describe any content that can be rendered within a React component. It could be HTML elements, other React components, text, numbers, or fragments.

2. **Why do we need to React Router?**

   It maintains consistent structure and behavior and is used to develop single-page web applications. Enables multiple views in a single application by defining multiple routes in the React application.

3. **Explain the role of Reducer.**

   Reducers are pure functions which specify how the application's state changes in response to an ACTION. Reducers work by taking in the previous state and action, and then it returns a new state. It determines what sort of update needs to be done based on the type of the action, and then returns new values. It returns the previous state as it is, if no work needs to be done.

4. **What is the use of React.cloneElement?**

   React.cloneElement is a method provided by React that allows you to clone and modify a React element. It is typically used when you want to add or override props of a React element while maintaining its original type and key. This is useful when working with components that accept and modify their children's elements.

5. **Can you explain React's "lifting state up" concept and why it is important?**

   "Lifting state up" is a concept in React where you move the shared state of multiple components to their common parent component. By doing this, you establish a single source of truth for the shared state, allowing you to pass down the state and related functions as props to child components. It enables better data flow, state management, and enhances performance by reducing unnecessary re-rendering.

Day 81

1. ### What is a ref in React and what is its purpose?

   A ref in React is a way to access and interact with the real DOM elements or React components directly. Its purpose is to provide a means of reading or modifying the properties, state, or behavior of a component outside of the typical React data flow. In simple terms, refs allow you to directly manipulate elements or components, like getting their values, focusing input fields, or triggering methods, bypassing the usual data flow of React.

2. ### How does using refs differ from traditional DOM manipulation?

   Refs in React provide a convenient and efficient way to work with elements and components, allowing you to easily access and modify them. It’s like a special tool that React provides, making it easier to interact with the user interface. Traditional DOM manipulation, on the other hand, involves more manual and cumbersome approaches to achieve the same results.

3. ### What is the main difference between “string refs” and “callback refs” in React.

   The main difference is that string refs rely on a string identifier, while callback refs use a function to capture the reference. Callback refs are more flexible and allow for better control and manipulation of the reference, making them the preferred method in React whereas using string refs approach is considered legacy and is not recommended in newer versions of React.

4. ### Explain the concept of “forwarding refs” in React.

   “Forwarding refs” in React allows a parent component to pass a ref to its child component. This way, the parent can access and control the child’s DOM element or component. It’s like giving a special power to the parent component to interact with its child component’s internals.

5. ### How do refs affect the component lifecycle in React?

   Refs in React do not directly impact the component lifecycle. They provide a way to access and interact with elements or components separately from lifecycle methods. Refs allow you to work with specific parts of the component without affecting its lifecycle or rendering.

Day 82

1. ### What are the benefits of using callback refs over string refs?

   The benefits of using callback refs over string refs in React are that callback refs provide more flexibility and control when accessing and manipulating DOM elements. With callback refs, you have direct access to the actual DOM node, allowing you to perform more advanced operations. On the other hand, string refs are limited to identifying elements using a string name, which can be less versatile and may not provide the same level of control.

2. **Can we use refs with functional components in React? If yes, how?**

   Yes, you can use refs with functional components in React. The `useRef` hook allows you to create a ref and use it within a functional component. By calling `useRef()` and assigning the result to a variable, you can access and manipulate the ref's current value, which can be used to reference elements in the component's JSX.

3. **What are the limitations or caveats of using refs in React?**

   While refs can be useful, there are limitations and caveats to consider when using them in React. One limitation is that refs bypass the typical data flow in React, which can make it harder to track and understand changes in your component's state. Additionally, overusing refs can lead to code that is harder to maintain and test, as it may introduce dependencies between different parts of your application.

4. **How do you clean up or release the resources associated with a ref in React?**

   To clean up or release resources associated with a ref in React, you can leverage the `useEffect` hook. Within the `useEffect` hook, you can return a cleanup function that will be called when the component unmounts or when the ref value changes. This allows you to handle any necessary cleanup operations, such as removing event listeners or cancelling timers, ensuring that your application's resources are properly managed.

5. **How do you access the DOM node using refs in React?**

   Accessing the DOM node using refs in React is straightforward. After creating a ref using the `useRef` hook or the `React.createRef()` function, you can attach the ref to a JSX element using the `ref` attribute. Once the component has rendered, you can access the DOM node by referring to the current property of the ref.

Day 83

1. **What is binding in React and why is it important?**

   Binding in React ensures that a function has the correct reference to the component it belongs to. It's important because it allows the function to access the component's data and methods.

2. **What happens if you don’t bind a function in React? Explain the potential issues that can arise.**

   If you don’t bind a function in React, it may lose access to the component’s instance and its associated data. This can result in errors or unexpected behavior when the function is called.

3. **How can you bind a function in a React class component? Describe the different methods available.**

   You can bind a function in a React class component by either using the constructor to bind the function explicitly or by using arrow function syntax, which automatically binds the function to the component instance.

4. **What is the recommended approach for binding functions in React components?**

   The recommended approach for binding functions in React components is to use arrow function syntax or bind the functions in the constructor. Arrow function syntax simplifies the code and eliminates the need for explicit binding.

5. **Explain the difference between binding in class components and functional components in React.**

   Binding functions in class components is done using the constructor or arrow function syntax, while functional components don’t require explicit binding as they automatically bind functions to the component instance.

Day 84

1. **What is the difference between Element and Component in React?**

   An element is a basic building block or specific part of a system or interface, like a button or text input field. A component, on the other hand, is a larger, self-contained entity made up of one or more elements, such as a navigation bar or login form. Components are reusable and provide specific functionality within a system. In essence, elements are smaller parts, while components are larger, composed entities.

2. **How to apply validation on props in React?**

   To apply validation on props in React, you can use the PropTypes library. Install it, import it into your component file, and define the expected props and their types using PropTypes. React will display warnings in the console if the provided props don't match the specified types or if any required props are missing.

3. **Why is a React component declarative?**

   A React component is declarative because it tells React what the UI should look like based on the current data. Instead of giving step-by-step instructions, we describe the desired UI state and let React handle the rest. It simplifies the process of building interfaces by focusing on "what" we want, rather than worrying about the nitty-gritty of "how" to achieve it.

4. **What is ReactDOM package?**

   The ReactDOM package in React is responsible for rendering the React components to the browser's DOM (Document Object Model). It provides methods and functionality for efficiently updating and manipulating the DOM based on changes in the React component tree. ReactDOM acts as the bridge between React's virtual representation of the UI and the actual HTML elements on the webpage, ensuring that any changes in the component hierarchy are reflected in the rendered UI.

5. **how is React different from Angular?**

   ReactJS is a JavaScript library focused on building user interfaces using reusable components, while Angular is a comprehensive framework following the MVC/MVVM patterns. React uses JavaScript and JSX, whereas Angular is based on TypeScript. React has a simpler learning curve, a large community, and a lightweight nature, while Angular offers a complete solution with a steeper learning curve, a robust ecosystem provided by Google, and built-in features. The choice depends on project needs, team preferences, and scalability requirements.

Day 85

1. **Explain the use of CSS modules in React.**

   CSS modules in React allow you to locally scope CSS styles to specific components, preventing class name collisions and promoting a more modular approach to styling. By generating unique class names for each component, CSS modules ensure that styles defined in one component won't interfere with styles in another

2. **Can you explain what custom hooks are in React and how they differ from regular hooks?**

Custom hooks in React are reusable JavaScript functions that allow you to extract logic and stateful behavior from components. While regular hooks like useState and useEffect are provided by React, custom hooks are created by developers to encapsulate specific logic that can be shared across multiple components.

3. **What are the benefits of using custom hooks in React? Can you provide an example of a scenario where a custom hook would be useful?**

   The benefits of using custom hooks in React are improved code organization, reusability, and separation of concerns. They enable you to extract complex logic into a reusable hook, reducing duplication and promoting cleaner, more maintainable code. For example, a custom hook for handling form validation can be reused in multiple forms throughout the application.

4. **How do you create a custom hook in React?**

   To create a custom hook in React, you start by defining a function with the "use" prefix, such as "useCustomHook." Inside the function, you can use existing React hooks and additional logic to encapsulate the desired state or behavior. Return any values or functions that need to be accessed outside the custom hook. Once defined, you can import and use the custom hook in any component, enabling code reuse and modularity across your React application.

5. **Are there any limitations or considerations when using custom hooks in React? Are there any performance implications or potential pitfalls that developers should be aware of?**

   Custom hooks in React don't have built-in limitations or performance issues. Just keep an eye on any dependencies and side effects in your custom hooks. Also, remember to follow the rules of hooks like calling them at the top level and avoiding conditional usage. By being aware of these considerations, you can effectively use custom hooks in your React applications.

Day 86

1. **What is Formik, and why would you use it in a React application?**

   Formik is a powerful form management library in React that simplifies the process of handling forms. It provides an easier way to manage form state, handle form validation, and manage form submission. With Formik, you can streamline the development of forms by reducing boilerplate code and handling complex form-related tasks efficiently.

2. **What are the advantages of using Formik over traditional form handling in React?**

   Formik offers several advantages over traditional form handling in React. It simplifies the process of managing form state and reduces the need for manual state management. Formik's built-in validation capabilities make it easier to handle form validation without relying on additional libraries. Overall, Formik significantly improves the developer experience when working with forms in React.

3. **What is Axios, and what problem does it solve in JavaScript or React development?**

   Axios is a JavaScript library used for making HTTP requests in browsers or Node.js. It simplifies the process of sending and receiving data from servers, making it easier to handle network requests in JavaScript or React applications.

4. **What are the advantages of using Axios over the built-in fetch API in JavaScript?**

   Axios offers advantages over the built-in fetch API by providing a more convenient and consistent API, better error handling, support for request cancellation, and the ability to intercept requests and responses.

5. **What is Material-UI, and what is its purpose in React development?**

   Material-UI is a React component library that implements Google's Material Design guidelines. It offers pre-designed and customizable UI components, making it easier to build visually appealing and user-friendly interfaces. With Material-UI, developers can focus on functionality while simplifying the process of styling and creating responsive layouts.

Day 87

1. **What is render hijacking in react?**

   Render hijacking is a technique in React that allows you to control what a component outputs from another component. You do this by wrapping the component in a Higher-Order Component (HOC). By wrapping the component, you can inject additional props or make other changes, which can cause the component to render differently.

2. **What are Keyed Fragments?**

Keyed fragments in React group multiple elements without a wrapper element, improving performance. By assigning a unique key to each child element, efficient reordering and updating are ensured during rendering. The general use case is mapping a collection to an array of fragment

3. **What is suspense component?**

   If the module containing the dynamic import is not yet loaded by the time parent component renders, you must show some fallback content while you're waiting for it to load using a loading indicator. This can be done using Suspense component.

4. **Is it possible to use react without JSX?**

   Yes, it's possible to use React without JSX. JSX is a syntax extension for JavaScript used to write HTML-like code in React. However, you can still create React elements using the `createElement` function provided by React, which takes the element type, props, and children as arguments.

```
const MyComponent = () => {
return React.createElement("div", { className: "my-class" }, "Hello, World!");
};
```

5. **Why does strict mode render twice in React?**

   Strict mode in React renders components twice to provide additional checks and useful warnings. It acts as a safeguard that catches possible issues in your code. By rendering twice, React can detect and alert you about problems like unintended side effects or outdated practices. This way, you can fix those issues before they cause any trouble in your application, making it more reliable and sturdy.

Day 88

1. **What is React Fiber?**

   React Fiber is a new reconciliation algorithm introduced in React 16. It is responsible for the efficient rendering and updating of components in React applications. React Fiber breaks the rendering work into smaller units called "fibers" and manages them in a priority-based manner.

2. **How does React Fiber improve the overall performance of React applications?**

React Fiber improves performance by allowing more control over scheduling and rendering. It enables asynchronous rendering, which avoids blocking the main thread, resulting in a smoother user experience.

3. **What is Babel in React js?**

   Babel is a popular JavaScript compiler widely used in the React ecosystem. It allows developers to write modern JavaScript code (including the latest ECMAScript features) and transforms it into backward-compatible versions that can run in older browsers and environments that don't support those features(ES6, ES7 into plain old ES5).

4. **What is a wrapper component in react?**

   A wrapper component in React is like a parent container that wraps around other components. It adds extra features, behavior, or styling to the wrapped components. It acts as a higher-level component, enhancing the functionality of the components it contains.

5. **Explain DOM diffing?**

   DOM diffing, or reconciliation in React, is the process of comparing the previous and new versions of the user interface. It identifies the changes and updates needed for the actual web page. React determines what has changed, adds or removes elements, and updates only the necessary parts of the user interface. This makes the updates faster and more efficient by avoiding unnecessary re-rendering and manipulation of the web page.

Day 89

1. **What are the dependencies in the dependency array of the useEffect hook? How do they affect the behavior of the hook?**

   Dependencies in the dependency array of the useEffect hook determine when the effect should be executed. If any of the dependencies change, the effect will run again. If the dependency array is empty, the effect will only run once.

2. **What are some common use cases for the useEffect hook?**

   Common use cases for the useEffect hook include fetching data from an API, subscribing to external data sources, setting up event listeners, updating the document title, and performing any other side effects that need to occur when the component renders or updates.

3. **Can you explain the concept of multiple useEffect hooks in a single component? How do they interact with each other?**

   Multiple useEffect hooks in a single component allow you to separate concerns and organize code. Each useEffect hook operates independently and is triggered based on its own specified dependencies. They do not interact directly with each other.

4. ## What is the purpose of the cleanup function returned by the useEffect hook? How can you utilize it effectively?

   The cleanup function returned by the useEffect hook is important for cleaning up any resources or subscriptions created by the effect. To perform cleanup, you simply return a cleanup function from the effect. The cleanup function is useful for removing event listeners, cancelling subscriptions, or performing other necessary cleanup tasks.

```
     useEffect(() => {
  const timerId = setInterval(() => {
    console.log('Running effect...');
  }, 1000);

  return () => {
   // Cleanup function: Clear interval when component unmounts
    clearInterval(timerId);
  };
}, []);
```

5. **What are the potential pitfalls or common mistakes when using the useEffect hook? How can you avoid them?**

   Some common mistakes with useEffect are: forgetting dependencies, causing stale data or infinite loops; not cleaning up properly, leading to memory leaks; and modifying state or props without proper dependency management, resulting in unexpected behavior. To avoid these, include all dependencies, update state or props conditionally, perform cleanup operations, and handle errors appropriately.

Day 90

1. **What is server-side rendering (SSR) in React.js?**

Server-side rendering (SSR) in React.js is the process of rendering React components on the server and sending the pre-rendered HTML to the client.

2. **Why would you choose to use server-side rendering instead of client-side rendering in React.js?**

   Server-side rendering is chosen over client-side rendering in React.js for benefits like improved performance, SEO friendliness, and better initial page load experience.

3. **How does server-side rendering differ from client-side rendering in React.js?**

   Server-side rendering differs from client-side rendering in that the rendering process occurs on the server before sending the HTML to the client, whereas client-side rendering renders components in the browser.

4. **Explain the benefits and drawbacks of server-side rendering in React.js.**

   The benefits of server-side rendering in React.js include improved SEO, faster initial page load, and better performance for low-end devices. However, it can introduce more complexity and may not be suitable for all applications.

5. **What are the performance implications of server-side rendering in React.js?**

   Server-side rendering can improve performance by reducing the time required for the initial render, but it can also increase the server load and network traffic for subsequent or additional requests.

Day 91

1. **How can you implement server-side rendering with React.js without using frameworks like Next.js?**

   To perform server-side rendering with React.js without frameworks like Next.js, you need to set up a Node.js server, use a build system like Webpack or Babel, and implement server-side rendering logic using libraries like react-dom/server.

2. **Describe the steps involved in setting up server-side rendering with React.js from scratch.**

The steps for setting up server-side rendering with React.js from scratch involve creating a Node.js server, configuring a build system, creating a server-side entry point, implementing rendering logic, and setting up routing.

3. **What libraries or tools can be used to perform server-side rendering with React.js?**

   Libraries like react-dom/server, ReactDOMServer, and express can be used to perform server-side rendering with React.js.

4. **How can you handle data fetching and asynchronous operations during server-side rendering in React.js?**

   Data fetching and asynchronous operations during server-side rendering can be handled by making use of lifecycle methods like componentDidMount or using libraries like react-async.

5. **What considerations should you keep in mind when implementing server-side rendering for a large-scale React.js application?**

   Considerations for server-side rendering in large-scale React.js applications include optimizing performance, managing data fetching efficiently, and dealing with complex application states.

Day 92

1. **Can you explain the concept of code splitting and how it relates to server-side rendering in React.js?**

   Code splitting is the process of splitting the JavaScript bundle into smaller chunks to improve performance. It can be utilized with server-side rendering to load only the necessary JavaScript code for each route or component.

2. **How can you optimize server-side rendered React.js applications for search engine optimization (SEO)?**

   To optimize server-side rendered React.js applications for SEO, ensure that important content is present in the initial HTML, use proper meta tags, and provide server-side rendering for dynamic content.

3. **What are some common challenges or pitfalls associated with server-side rendering in React.js, and how can you address them?**

Common challenges with server-side rendering include handling client-side interactions, managing state across the server and client, and dealing with third-party libraries that are not SSR-friendly. These challenges can be addressed by using techniques like rehydration and carefully handling asynchronous operations.

4. **How does server-side rendering impact the development and debugging process compared to client-side rendering?**

   Server-side rendering impacts development and debugging differently from client-side rendering, as errors and issues can occur on both the server and the client. Proper error handling and debugging techniques are necessary.

5. **Can you explain the concept of hydration in the context of server-side rendering in React.js?**

   Hydration in server-side rendering refers to the process of attaching event listeners and reattaching React components on the client-side after the initial server-rendered HTML has been received. It enables interactivity and seamless transition to client-side rendering.

Day 93

1. **What are the best practices for organizing and structuring a React application?**

   Best practices for organizing and structuring a React application include using a component-based architecture, separating concerns with folders and files, and utilizing state management libraries like Redux or React Context.

2. **How do you handle authentication and authorization in React applications?**

   There are two main ways to handle authentication and authorization in React applications. One way is to use a third-party library, such as Auth0 or Firebase. These libraries provide a number of features that make it easy to implement authentication and authorization in your application, such as user management, password hashing, and session management.

3. **What are the pros and cons of using React in a project compared to other JavaScript frameworks?**

The pros of using React in a project include its component-based architecture, virtual DOM for efficient rendering, a large and active community, and the ability to create reusable UI components. The cons can include a steep learning curve, complex configuration for larger projects, and potential performance issues with improper optimization.

4. **Why do we need Nextjs?**

   Next.js is a React framework that provides server-side rendering (SSR), static site generation (SSG), and other features like routing and API handling. It is useful for building optimized and performant React applications, improving SEO, and enabling server-side functionality.

5. **What are some of the challenges you have faced working with React?**

   Write your own answer

Typescript

Day 94

1. **What is TypeScript, and how does it differ from JavaScript?**

   TypeScript is a statically typed superset of JavaScript that adds optional static typing and other features to JavaScript. It differs from JavaScript by introducing type checking at compile-time, which helps catch errors early and improves code maintainability.

2. **What are the benefits of using TypeScript?**

   The benefits of using TypeScript include enhanced code quality through static typing, improved developer productivity with better tooling and autocompletion, increased maintainability through code organization and documentation, and easier collaboration within teams.

3. **How do you define variables with specific types in TypeScript?**

   Variables with specific types in TypeScript can be defined using type annotations. For example, you can declare a variable with a specific type like this: `let myVariable: string = "Hello";` . Here, `myVariable` is defined as a string type.

4. **What are the basic data types in TypeScript?**

   The basic data types in TypeScript are number, string, boolean, object, array, tuple, and enum.

5. **What is tuple datatype in TypeScript?**

   In TypeScript, a tuple is a type that allows you to define an array with a fixed number of elements of different types. Tuples are similar to arrays, but the types of elements in a tuple are fixed and their order matters. Tuples are useful when you want to work with a specific set of values, each with its own type, and maintain their order throughout the program.

Day 95

1. **What is the purpose of type annotations in TypeScript, and how are they used?**

   The purpose of type annotations in TypeScript is to specify the types of variables, function parameters, and return values. They help catch errors during development and provide better tooling support. Annotations are used by adding a colon after the variable or function parameter name, followed by the type.

2. **How can you define custom types or interfaces in TypeScript?**

   In TypeScript, custom types can be defined using interfaces or type aliases. Interfaces define the structure of an object and can be implemented by classes, while type aliases create a new name for a specific type or combination of types. They are defined using the `interface` and `type` keywords, respectively.

3. **Explain the concept of type inference in TypeScript and how it works.**

   Type inference in TypeScript is the ability of the compiler to automatically determine the type of a variable based on its initialization value. TypeScript analyzes the assigned value and infers the most appropriate type for the variable, reducing the need for explicit type annotations.

4. **What is the difference between interfaces and type aliases in TypeScript? When would you use one over the other?**

   Interfaces in TypeScript define how an object should look or what a class should implement. They can be extended or implemented by other interfaces and classes. Type aliases, on the other hand, give new names to existing types or combine multiple types together. Use interfaces when you want to describe object structure or class requirements. Use type aliases when you want to create shorter names for types or combine them in a meaningful way.

5. **How does TypeScript support class-based object-oriented programming?**

TypeScript supports class-based object-oriented programming. It allows defining classes, inheritance, access modifiers (public, private, protected), abstract classes, and interfaces for contracts. It enforces type checking and helps organize complex class hierarchies and interactions.

Day 96

1. **What is the use of the tsconfig.json file?**

   The tsconfig.json file is used in TypeScript to configure the compiler options for a project. It allows developers to specify settings such as target version, module system, output directory, and more.

2. **Explain the concept of type assertion in TypeScript.**

   Type assertion in TypeScript is a way to tell the compiler the specific type of a value when it cannot be inferred automatically. It allows you to override the compiler’s default assumptions and treat a value as a particular type. Type assertions are useful when you have more knowledge about the type of a value than what the compiler can determine on its own.

3. **What is the “any” type in TypeScript, and when should it be used?**

   The “any” type in TypeScript is a type that represents a value of any type. It essentially disables type checking for that particular value, allowing it to be assigned or used in any context.

4. **How can you enforce strict null checks in TypeScript?**

   To enforce strict null checks in TypeScript, you can enable the “strictNullChecks” compiler option in the tsconfig.json file. When enabled, TypeScript will check for null and undefined values more strictly, helping to prevent common errors related to null and undefined.

5. **What is the “readonly” modifier in TypeScript, and how does it affect properties and arrays?**

   The “readonly” modifier in TypeScript is used to make properties or array elements read-only, meaning they cannot be modified once initialized. It provides a way to enforce immutability and prevent accidental modifications to certain values. When applied to properties, it prevents reassignment, and when applied to arrays, it disallows adding or removing elements after initialization.

Day 97

1. **Can TypeScript be used for the backend?**

   Yes, TypeScript can be used for backend development. TypeScript is a superset of JavaScript that adds static typing and other features to JavaScript, making it more suitable for larger, complex applications. It can be used with popular backend frameworks and platforms such as Node.js to build server-side applications.

2. **what is optional properties in an interface in TypeScript?**

   In TypeScript, you can define optional properties in an interface by appending a question mark (?) to the property name. This indicates that the property is optional and may or may not be present in the object that implements the interface.

```
//age & email are marked as optional so it will not cause any error if they're not assig
interface Person {
  name: string;
  age?: number;
  email?: string;
}
```

3. **Explain generics in Typescript.**

   Generics in TypeScript enable you to create reusable code that can work with different data types. They act as placeholders for types and provide flexibility without sacrificing type safety. By using angle brackets (<>), you can create functions, interfaces, classes, or type aliases that can handle multiple types.

```
function identity<T>(arg: T): T {
  return arg;
}
//this is how generics increase flexibility

let result1 = identity<number>(42); // result1 is of type number
let result2 = identity<string>("Hello"); // result2 is of type string
```

4. **Explain the concept of type guards in TypeScript.**

Type guards in TypeScript are checks that help determine the type of a variable. They help TypeScript understand the specific type of the variable, making your code safer and more accurate. Type guards are used within conditional blocks to refine the type based on conditions, improving type inference and reducing the risk of errors.

5. **What is the `keyof` operator in TypeScript?**

   The `keyof` operator in TypeScript is used to get a union type of all the keys (property names) of an object type. It allows you to access and use the keys of an object type as string literals in type operations and transformations.

```
interface Person {
    name: string;
    age: number;
  }

  type PersonKeys = keyof Person;
  const key1: PersonKeys = 'name'; //Valid key
  const key3: PersonKeys = 'address'; //Error
```

Day 98

1. **Explain the difference between “interface” and “class” in TypeScript.**

   An interface in TypeScript is used to describe the structure of an object, without providing an implementation. It defines the properties and methods that an object should have. On the other hand, a class is a blueprint for creating objects that encapsulates data and behavior. It allows you to create instances of objects, implement interfaces, and leverage object-oriented programming concepts like inheritance and encapsulation.

2. **what is enum in Typescript?**

   Enums in TypeScript is a way to define a collection of related constants. Enums assign automatic numeric values to each enumerator by default, but you can also customize them. Enums are useful when you have a fixed set of values that you want to refer to using meaningful names instead of explicit values throughout your code.

3. **Can you erase the values of array elements in TypeScript?**

   In TypeScript, we cannot directly erase or remove values from individual array elements. Once a value is assigned to an element in an array, it remains in that position until we replace it with a new value. However, we can assign `null`, `undefined`, or any other special value to indicate an empty or erased state for an array element.

4. **What is Intersection types in Typescript?**

   Intersection types allow you to combine multiple types into a single type that has all the properties and methods of each constituent type. It is denoted by the “&” symbol.

```
interface A {
  propA: number;
}

interface B {
  propB: string;
}

type IntersectionType = A & B;

const obj: IntersectionType = {
  propA: 123,
  propB: "hello",
};
```

5. **What is Union types in Typescript?**

   Union types allow you to define a type that can hold values of multiple types. It is denoted by the “|” symbol.

```
type UnionType = string | number;

let val: UnionType;
val = "hello"; // Valid
val = 123;     // Valid
val = true;    // Error, as boolean is not part
```

Day 99

6. **Which TypeScript types are immutable?**

   In TypeScript, all primitive data types, such as strings, numbers, booleans, etc., are immutable, meaning that the value cannot be changed more than once. Additionally, it means that passing them to functions has no negative impacts.

7. ### What is namespace in Typescript?

   In TypeScript, a namespace is a way to organize and group related code elements such as classes, interfaces, functions, and variables under a single name. Namespaces provide a mechanism to avoid naming conflicts and create a logical hierarchy within your codebase.

8. ### What is modules in Typescript?

   Modules in TypeScript allow you to organize and encapsulate code into separate files. By using the `export` keyword, you can make specific code elements accessible to other modules, while the `import` keyword allows you to use those exported elements in your module. Modules make it easier to manage dependencies, reuse code, and maintain a modular structure in your TypeScript projects.

9. ### What is mixins in Typescript?

   In TypeScript, mixins allow you to extend a class by combining the properties and methods from other classes. They provide a way to reuse code across multiple classes without using inheritance. By applying a mixin to a target class, you can add additional functionality to the class in a modular and flexible manner.

10. ### What is declaration merging in Typescript?

    Declaration merging in TypeScript lets you combine multiple declarations with the same name into a single declaration. It allows you to add or extend properties, methods, or functionality to existing types, making it easier to build and extend code across multiple sources without duplicating definitions.

```
interface Box {
  height: number;
  width: number;
}
interface Box {
  scale: number;
}
let box: Box = { height: 5, width: 6, scale: 10 };
```

Day 100

1. ### What are the Disadvantages of TypeScript?

TypeScript has a few drawbacks. It requires extra time and effort to annotate types, which can slow down development. It also has a learning curve, especially for developers new to static typing. The compilation step can add complexity to the workflow, and integrating TypeScript with existing JavaScript codebases may require extra work and introduce compatibility challenges.

2. **What are decorators in TypeScript?**

   Decorators in TypeScript allows us to add metadata or behavior to classes, methods, properties, or parameters at design time. They are declared using the @ symbol followed by the decorator name, placed just before the class, method, property, or parameter declaration.

3. **Does TypeScript Support All Object Oriented Principles?**

   Yes, Typescript support all object oriented principles. There are 4 main principles to Object Oriented Programming: Encapsulation, Inheritance, Abstraction, and Polymorphism. TypeScript can implement all four of them with its smaller and cleaner syntax

4. **Does TypeScript supports function overloading?**

   Yes, TypeScript supports function overloading, which means you can define multiple versions of a function with different parameter types or arity. This allows you to provide different implementations of the function based on the arguments passed to it.

5. **What are the best practices when using TypeScript?**

   To use TypeScript effectively, follow these best practices: use static typing for error prevention and code clarity, enable strict mode and null checks, minimize the use of any type, follow consistent naming conventions, keep TypeScript and compiler options up to date, etc.