Top characteristics of software are:

- **Functionality:** It refers to the software performance compared to the purpose it was created for.
- **Reliability:** It is a characteristics of software that refers to its ability to perform what it was designed to do accurately and consistently over time.
- **Usability (User-friendly):** It refers to the extent to which the software can be used with ease. The amount of effort or time required to learn how to use the software.
- **Efficiency:** It refers to the ability of the software to use system resources in the most effective and efficient manner.
- **Flexibility:** It refers to how simple it is to improve and modify the software.
- **Maintainability:** It refers to how easily a software system can be modified to add feature, improve speed, or repair faults.
- **Portability:** It refers to how well the software can work on different platforms or situations without making major modifications.
- **Integrity:** It refers to how well the software maintains the accuracy and consistency of data throughout its cycle.

For more details please refer to the following article Characteristics of Software.

2. What are the Various Categories of Software?

The software is used extensively in several domains including hospitals, banks, schools, defense, finance, stock markets, and so on. It can be categorized into different types:

1. Based on Application

1. System Software- This type of software helps manage the hardware of your computer, like an operating system that controls how the computer works.
2. Application Software- These are the programs we use every day, such as word processors or music players, to perform specific tasks.
3. Web Applications Software- These are programs you access via a web browser, like checking your email or managing your bank account online.
4. **Embedded Software-**This software is built into devices like cars, home

5. **Reservation Software**: Used to manage bookings and reservations, such as for hotels, flights, or restaurant tables.
6. **Business Software-** Designed to help businesses run smoothly, this software includes things like CRM (Customer Relationship Management) tools and ERP (Enterprise Resource Planning) systems.
7. **Artificial Intelligence Software-** These programs use machine learning to mimic human intelligence and perform tasks like recommendations or chatbots.
8. **Scientific Software-** Software used by researchers for tasks like simulations, analyzing data, or modeling experiments.

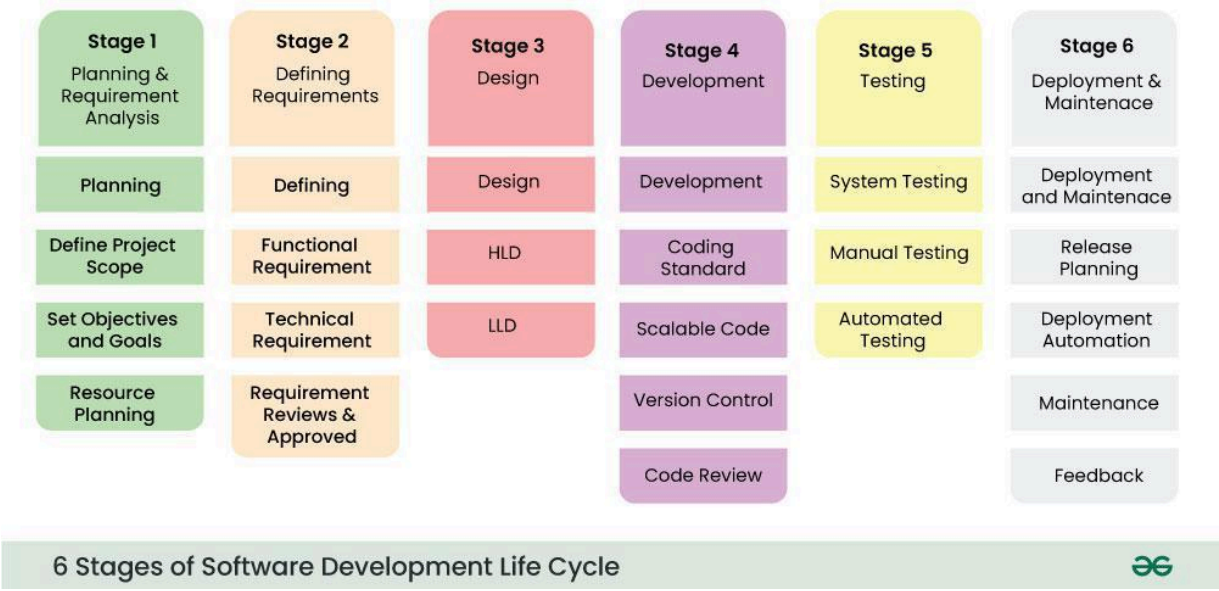
2. **Based on Copyright**

1. **Commercial Software**: This is software you pay for, usually with a license that limits how it can be used, shared, or changed.
2. **Shareware Software:** This software is available for free for a limited time or with limited features, with the hope that users will eventually pay for the full version.
3. **Freeware Software**: Software that's completely free to use, but the source code is usually not available for you to change or share.
4. **Public Domain Software**: This software is free for everyone to use, modify, and share without any copyright restrictions.

For more details please refer to the following article Classifications of Software.

3. Explain SDLC and its Phases?

SDLC stands for **Software Development Life Cycle**. It is a process followed for software building within a software organization**.** SDLC consists of a precise plan that describes how to develop, maintain, replace, and enhance specific software. The life cycle defines a method for improving the quality of software and the all-around development process.



6 Stages of Software Development Life Cycle

Phases of SDLC: Following are the phases of SDLC:

1. **Planning and Requirement Analysis**: This is where we figure out the project's goals, understand what users need, and set clear expectations for what the software should do.
2. **Defining Requirements**: Here, we get into the specifics—laying out exactly what the software needs to do (functional) and how well it should do it (non-functional).
3. **Designing Architecture**: At this stage, we build a blueprint for the software, deciding on the overall structure and the technical details to make sure it meets the requirements.
4. **Developing Product**: This is where the coding happens, turning the design into a working product by writing the actual software.
5. **Product Testing and Integration**: Now, we test the software for bugs, check that everything works together smoothly, and make sure all the parts are integrated properly.
6. **Deployment and Maintenance of Products**: Finally, the software is deployed for use, and we continue to monitor and maintain it to fix any issues and keep it running smoothly over time.

For more details, please refer to the following article Software Development Life Cycle.

4. What are different SDLC Models Available?

1. Waterfall Model- This is a straightforward, step-by-step process where each phase must be finished before moving on to the next. It's ideal for projects with clear and fixed requirements from the start.
2. V-Model- Also called the Verification and Validation model, this approach focuses on testing each part of the development process as you go. For every phase of development, there's a corresponding testing phase.
3. Incremental Model- Instead of developing everything at once, this model breaks the software into smaller, manageable pieces or "increments." These pieces are developed and delivered one at a time, allowing users to get a working version early on.
4. RAD Model - This model focuses on getting the software up and running quickly through prototypes and constant user feedback. It works well for projects with clear user requirements and tight deadlines.
5. Iterative Model - In this approach, development happens in cycles. After each cycle, a version of the software is tested, feedback is gathered, and improvements are made in the next cycle.
6. Spiral Model - This model combines design and prototyping while focusing heavily on risk assessment. It involves repeating phases of planning, design, development, and testing with each loop, addressing any risks along the way.
7. Prototype model- Here, a working version of the product (prototype) is created early on. This allows users to interact with it and give feedback, which helps shape the final product.
8. Agile Model- Agile is all about flexibility. Development happens in short bursts, or sprints, with constant feedback from the customer. This model allows for changes at any stage, making it adaptable and responsive to evolving needs.

For more details, please refer to the following article Top Software Development Models (SDLC) Models.

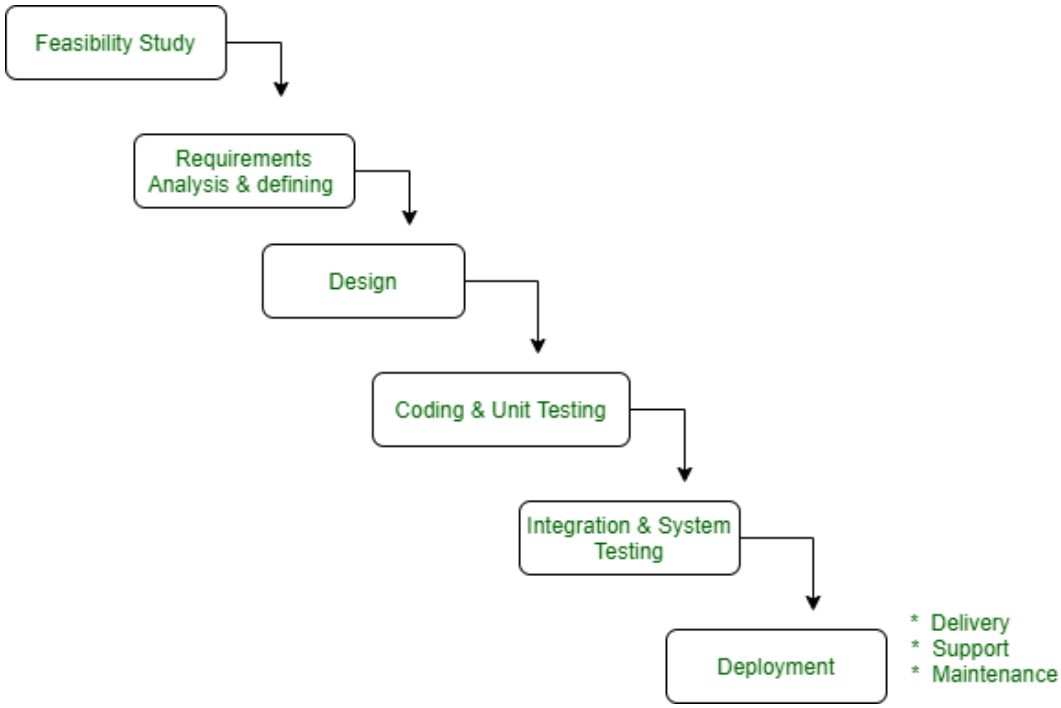
5. What is the Waterfall Method and What are its Use Cases?

The waterfall model is a software development model used in the context of large, complex projects, typically in the field of information technology. It is characterized by a structured, sequential approach to project management and software development.

1. **Requirements Gathering and Analysis**: This is the first step where you gather all the details about what the client needs and analyze them to understand exactly what the software should do.
2. **Design Phase**: Once you have a clear idea of the requirements, you move on to designing how the system will work. This is where you figure out the structure and how different parts of the software will fit together.
3. **Implementation and Unit Testing**: Now it's time to start coding. In this phase, developers write the actual software and test each small piece (or module) to make sure it works properly on its own.
4. **Integration and System Testing**: After all the individual modules are built and tested, they're put together to form the complete system. The whole system is tested to ensure everything works as expected when all the parts are connected.
5. **Deployment**: Once the software is tested and ready, it's launched and made available to users in the real world.
6. **Maintenance**: After the software is up and running, it enters the maintenance phase. This involves fixing any issues users find and making updates as needed to keep everything running smoothly.

Use Case of Waterfall Model

- Requirements are clear and fixed that may not change.
- There are no ambiguous requirements (no confusion).
- It is good to use this model when the technology is well understood.
- The project is short and cast is low.
- Risk is zero or minimum.



Waterfall Model

For more details, please refer to the following article Waterfall Model.

6. What is Black Box Testing?

The black box test (also known as the conducted test/ closed box test/ opaque box test) is software testing technique. In this technique, tester does not care about the internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements. The name "black box" refers to the idea that the internal workings are hidden from the tester's view.

For more details, please refer to the following article Software Engineering - Black Box Testing.

7. What is White Box Testing?

White Box Testing is a method of analyzing the internal structure, data structures used, internal design, code structure, and behavior of software, as well as functions such as black-box testing. Also called glass-box test or clear box test or structural test.

For more details, please refer to the following article Software Engineering - White Box Testing.

Following are the differences between Alpha and Beta Testing:

Alpha Testing	Beta Testing
Alpha testing involves both white box and black box testing.	Beta testing commonly uses black-box testing.
Alpha testing is performed by testers who are usually internal employees of the organization.	Beta testing is performed by clients who are not part of the organization.
Alpha testing is performed at the developer's site.	Beta testing is performed at the end-user, the of the product.
Reliability and security testing are not checked in alpha testing.	Reliability, security, and robustness are checked during beta testing.
Alpha testing ensures the quality of the product before forwarding it to beta testing.	Beta testing also concentrates on the quality of the product but collects the user's time-long input on the product and ensures that the product is ready for real-time users.
Alpha testing requires a testing environment or a lab.	Beta testing doesn't require a testing environment or lab.
Alpha testing may require a real-time long execution cycle.	Beta testing requires only a few weeks of execution.
Developers can immediately address the critical issues or fixes in alpha testing.	Most of the issues or feedback collected from the beta testing will be implemented in future versions of the product

For more details, please refer to the following article Alpha Testing and Beta Testing.

Debugging is the process of identifying and resolving errors, or bugs, in a software system. It is an important aspect of software engineering because bugs can cause a software system to malfunction, and can lead to poor performance or incorrect results. Debugging can be a time-consuming and complex task, but it is essential for ensuring that a software system is functioning correctly.

> *For more details, please refer to the following article What is Debugging?*

10. What is a Feasibility Study?

The Feasibility Study in Software Engineering is a study that analyze whether a proposed software project is practical or not. It early detects the potential issues, analyzes technological possibilities, and determines the project's financial and operational viability. This decreases the chance of project failure that also save time and money.

> *For more details, please refer to the following article Types of Feasibility Study in Software Project Development article.*

11. What is a Use Case Diagram?

A use case diagram is a behavior diagram and visualizes the observable interactions between actors and the system under development. The diagram consists of the system, the related use cases, and actors and relates these to each other:

- **System**: What is being described?
- **Actor**: Who is using the system?
- **Use Case**: What are the actors doing?

> *For more details, please refer to the following article use case diagram.*

12. What is the difference between Verification and Validation?

Here are the difference between Verification and Validation

Verification	Validation
Verification is a static practice of verifying documents, design, code, black-box, and programs human-based.	Validation is a dynamic mechanism of validation and testing the actual product.
It does not involve executing the code.	It always involves executing the code.
It is human-based checking of documents and files.	It is computer-based execution of the program.
Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking, etc.	Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing, etc.
Verification is to check whether the software conforms to specifications.	Validation is to check whether the software meets the customer's expectations and requirements.
It can catch errors that validation cannot catch.	It can catch errors that verification cannot catch.
Target is requirements specification, application and software architecture, high level, complete design, and database design, etc.	Target is an actual product-a unit, a module, a bent of integrated modules, and an effective final product.
Verification is done by QA team to ensure that the software is as per the specifications in the SRS document.	Validation is carried out with the involvement of the testing team
It generally comes first done before validation.	It generally follows after verification.
It is low-level exercise.	It is a High-Level Exercise.

13. What is a Baseline?

A baseline is a measurement that defines the completeness of a phase. After all activities associated with a particular phase are accomplished, the phase is complete and acts as a baseline for next phase.

For more details, please refer to the following article baseline.

Software Engineering Interview Questions for Intermediate

Here are the Top Software Engineering Interview Questions for **Intermediate:**

14. What is Cohesion and Coupling?

Cohesion indicates the relative functional capacity of the module. Aggregation modules need to interact less with other sections of other parts of the program to perform a single task. It can be said that only one coagulation module (ideally) needs to be run. Cohesion is a measurement of the functional strength of a module. A module with high cohesion and low coupling is functionally independent of other modules. Here, functional independence means that a cohesive module performs a single operation or function. The coupling means the overall association between the modules.

Coupling relies on the information delivered through the interface with the complexity of the interface between the modules in which the reference to the section or module was created. High coupling support Low coupling modules assume that there are virtually no other modules. It is exceptionally relevant when both modules exchange a lot of information. The level of coupling between two modules depends on the complexity of the interface.

For more details, please refer to the following article Coupling and cohesion.

15. What is the Agile software development model?

The
agile SDLC model is a combination of iterative and incremental process models with a focus on process adaptability and customer

involves cross-functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.

Advantages:

- Customer satisfaction by rapid, continuous delivery of useful software.
- Customers, developers, and testers constantly interact with each other.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed.

For more details, please refer to the following article Software Engineering - Agile Development Models.

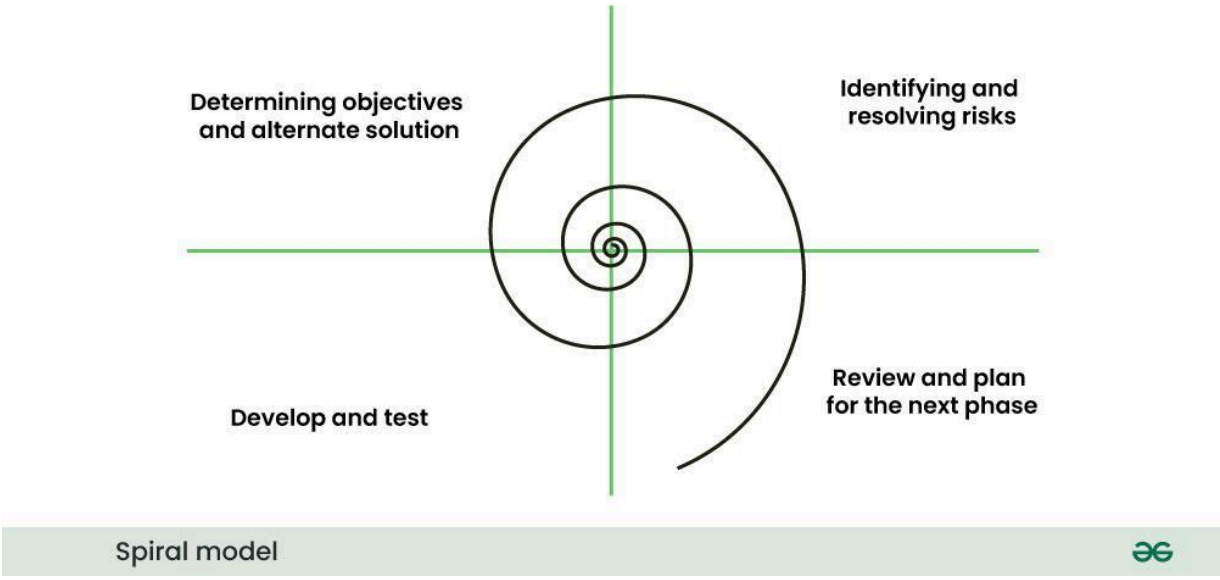
16. What is the Difference Between Quality Assurance and Quality Control?

Quality Assurance (QA)	Quality Control (QC)
It focuses on providing assurance that the quality requested will be achieved.	It focuses on fulfilling the quality requested.
It is the technique of managing quality.	It is the technique to verify quality.
It does not include the execution of the program.	It always includes the execution of the program.
It is a managerial tool.	It is a corrective tool.
It is process-oriented.	It is product-oriented.
The aim of quality assurance is to prevent defects.	The aim of quality control is to identify and improve the defects.
It is a preventive technique.	It is a corrective technique.

Quality Assurance (QA)	Quality Control (QC)
It is responsible for the full software development life cycle.	It is responsible for the software testing life cycle.
Example: Verification	Example: Validation

17. What is the Spiral Model, and its Disadvantages?

The Spiral Model is a **Software Development Life Cycle (SDLC)** model that provides a systematic and iterative approach to software development. In its diagrammatic representation, looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a **phase** of the software development process.



Spiral Model

Following are the disadvantages of spiral model:

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- The project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects

For more details, please refer to the following article Software Engineering - Spiral Model.

IBM first proposed the Rapid Application Development or RAD Model in the 1980s. The RAD model is a type of incremental process model in which there is a concise development cycle. The RAD model is used when the requirements are fully understood and the component-based construction approach is adopted.

Following are the limitations of RAD Model:

- For large but scalable projects RAD requires sufficient human resources.
- Projects fail if developers and customers are not committed in a much-shortened time frame.
- Problematic if a system cannot be modularized

For more details, please refer to the following article Software Engineering - Rapid Application Development Model (RAD).

19. What is Regression Testing?

Regression testing is defined as a type of software testing that is used to confirm that recent changes to the program or code have not adversely affected existing functionality. Regression testing is just a selection of all or part of the test cases that have been run. These test cases are rerun to ensure that the existing functions work correctly. This test is performed to ensure that new code changes do not have side effects on existing functions. Ensures that after the last code changes are completed, the above code is still valid.

*For more details, please refer to the following article **regression testing**.*

20. What are CASE Tools?

CASE stands for Computer-Aided Software Engineering. CASE tools are a set of automated software application programs, which are used to support, accelerate and smoothen the SDLC activities. It is a software package that helps with the design and deployment of information systems. It can record a database design and be quite useful in ensuring design consistency.

21. What is Physical and Logical DFD (Data Flow Diagram) ?

Physical DFD and Logical DFD both are the types of DFD (Data Flow

Physical DFD focuses on how the system is implemented. The next diagram to draw after creating a logical DFD is physical DFD. It explains the best method to implement the business activities of the system. Moreover, it involves the physical implementation of devices and files required for the business processes. In other words, physical DFD contains the implantation-related details such as hardware, people, and other external components required to run the business processes.

Logical data flow diagram mainly focuses on the system process. It illustrates how data flows in the system. In the Logical Data Flow Diagram (DFD), we focus on the high-level processes and data flow without delving into the specific implementation details. Logical DFD is used in various organizations for the smooth running of system. Like in a Banking software system, it is used to describe how data is moved from one entity to another.

22. What is Software Re-engineering?

Software re-engineering is the process of scanning, modifying, and reconfiguring a system in a new way. The principle of reengineering applied to the software development process is called software reengineering. It has a positive impact on software cost, quality, customer service, and shipping speed. Software reengineering improves software to create it more efficiently and effectively.

For more details please refer to What Is Software Re-Engineering?

23. What is Reverse Engineering?

Software Reverse Engineering is a process of recovering the design, requirement specifications, and functions of a product from an analysis of its code. It builds a program database and generates information from this. The purpose of reverse engineering is to facilitate maintenance work by improving the understandability of a system and producing the necessary documents for a legacy system.

Reverse Engineering Goals:

- Cope with Complexity.
- Recover lost information.

- Facilitate Reuse.

For more details, please refer to the following article Software Engineering - Reverse Engineering.

24. What are Software Project Estimation Techniques Available?

There are some software project estimation techniques available:

- PERT
- WBS
- Delphi method
- User case point

*For more details, please refer to the following article **Software Project Estimation Techniques**.*

25. How to Measure the Complexity of Software?

To measure the complexity of software there are some methods in software engineering:

- Line of codes
- Cyclomatic complexity
- Class coupling
- Depth of inheritance

For more details, please refer to the following article complexity of software.

26. Mentions Some Software Analysis and Design Tools?

Following are some software analysis and design tools:

- Data Flow Diagrams
- Structured Charts
- Structured English
- Data Dictionary
- Hierarchical Input Process Output diagrams

27. What is the Name of Various CASE Tools?

- Requirement Analysis Tool
- Structure Analysis Tool
- Software Design Tool
- Code Generation Tool
- Test Case Generation Tool
- Document Production Tool
- Reverse Engineering Tool

For more details, please refer to the following article Computer-Aided Software Engineering(CASE).

28. What is SRS?

Software Requirement Specification (SRS) Format is a complete specification and description of requirements of the software that needs to be fulfilled for successful development of software system. These requirements can be functional as well as non-requirements depending upon the type of requirement. The interaction between different customers and contractors is done because it is necessary to fully understand the needs of customers.

For more details please refer software requirement specification format article.

29. What is level-0 DFD?

The highest abstraction level is called Level 0 of DFD. It is also called context-level DFD. It portrays the entire information system as one diagram.

For more details, please refer to the following article DFD.

30. What is a Function Point?

Function point metrics provide a standardized method for measuring the various functions of a software application. Function point metrics, measure functionality from the user's point of view, that is, on the basis of what the user requests and receives in return.

31. What is the formula to Calculate the Cyclomatic Complexity?

The formula to calculate the cyclomatic complexity of a program is:

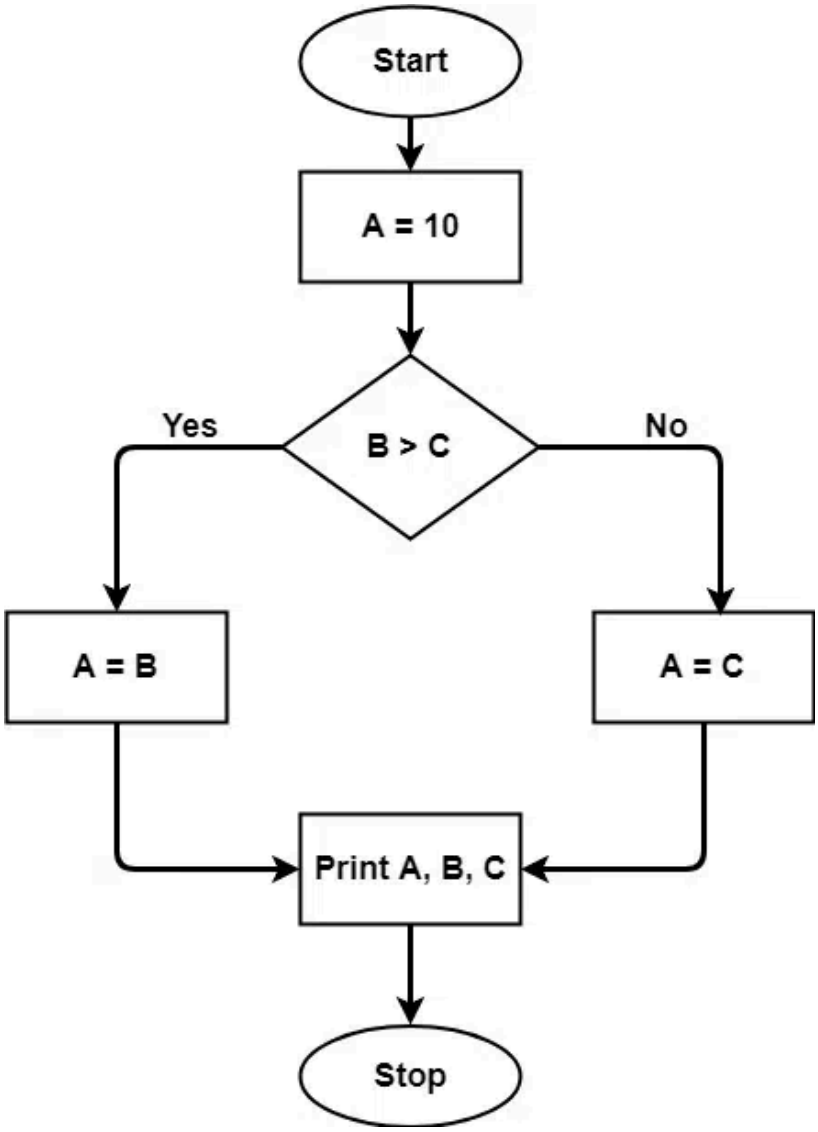
$$c = e - n + 2p \qquad \textit{where,}$$

1. *e = number of edges*
2. *n = number of vertices*
3. *p = predicates*

Example:

```
A = 10
   IF B > C THEN
      A = B
   ELSE
      A = C
   ENDIF
Print A
Print B
Print C
```

Control Flow Graph of the above code:



The cyclomatic complexity calculated for the above code will be from the control flow graph. The graph shows seven shapes(nodes), and seven lines(edges), hence cyclomatic complexity is 7-7+2 = 2.

32. What is the Cyclomatic Complexity of a Module that has 17 Edges and 13 Nodes?

The Cyclomatic complexity of a module that has seventeen edges and thirteen nodes = E – N + 2

E = Number of edges, N = Number of nodes
Cyclomatic complexity = 17 – 13 + 2 = 6

33. What is COCOMO Model?

A COCOMO model stands for Constructive Cost Model. As with all estimation models, it requires sizing information and accepts it in three forms:

- Function points
- Lines of source code

For more details, please refer to the following article Software Engineering - COCOMO Model.

34. Define an Estimation of Software Development Effort for Organic Software in the basic COCOMO Model?

Estimation of software development effort for organic software in the basic COCOMO model is defined as

```
Organic: Effort = 2.4(KLOC) 1.05 PM
```

Software Engineering Interview Questions for Advance

Here are the Top Software Engineering Interview Questions for **Advance:**

35. What Activities Come Under the Umbrella Activities?

The activities of the software engineering process framework are complemented by a variety of higher-level activities. Umbrella activities typically apply to the entire software project and help the software team manage and control progress, quality, changes, and risks. Common top activities include Software Project Tracking and Control Risk Management, Software Quality Assurance Technical Review Measurement Software Configuration Management Reusability Management Work Product Preparation and Production, etc.

For more details, please refer to the following article Umbrella activities in Software Engineering.

36. Which SDLC Model is the Best?

The selection of the best SDLC model is a strategic decision that requires a thorough understanding of the project's requirements, constraints, and goals. While each model has its strengths and weaknesses, the key is to align the chosen model with the specific characteristics of the project. Being flexible, adaptable, and communicating well are crucial in dealing with the

the end, the best way to develop software is the one that suits the project's needs and situation the most.

> *For more details, please refer to the following article Which SDLC Model is Best and Why ?*

37. What is the Black Hole Concept in DFD?

A block hole concept in the data flow diagram can be defined as "A processing step may have input flows but no output flows". In a black hole, data can only store inbound flows.

38. Which of the Testing is Used for Fault Simulation?

With increased expectations for software component quality and the complexity of components, software developers are expected to perform effective testing. In today's scenario, mutation testing has been used as a fault injection technique to measure test adequacy. Mutation Testing adopts "fault simulation mode".

39. Which Model is Used to Check Software Reliability?

A Rayleigh model is used to check software reliability. The Rayleigh model is a parametric model in the sense that it is based on a specific statistical distribution. When the parameters of the statistical distribution are estimated based on the data from a software project, projections about the defect rate of the project can be made based on the model.

40. What is the Difference between Risk and Uncertainty?

- Risk is able to be measured while uncertainty is not able to be measured.
- Risk can be calculated while uncertainty can never be counted.
- You are capable of make earlier plans in order to avoid risk. It is impossible to make prior plans for the uncertainty.
- Certain sorts of empirical observations can help to understand the risk but on the other hand, the uncertainty can never be based on empirical observations.
- After making efforts, the risk is able to be converted into certainty. On the

- After making an estimate of the risk factor, a decision can be made but as the calculation of the uncertainty is not possible, hence no decision can be made.

41. What is CMM?

To determine an organization's current state of process maturity, the SEI uses an assessment that results in a five-point grading scheme. The grading scheme determines compliance with a capability maturity model (CMM) that defines key activities required at different levels of process maturity. The SEI approach provides a measure of the global effectiveness of a company's software engineering practices and establishes five process maturity levels that are defined in the following manner:

- Level 1: Initial
- Level 2: Repeatable
- Level 3: Defined
- Level 4: Managed
- Level 5: Optimizing

For more details, please refer to the following article CMM.

43. A software does not wear out in the traditional sense of the term, but the software does tend to deteriorate as it evolves, why?

The software does not wear out in the traditional sense of the term, but the software does tend to deteriorate as it evolves because Multiple change requests introduce errors in component interactions. Unlike hardware, software doesn't physically wear out. However, it tends to deteriorate as it evolves because every time new features or updates are added, there's a chance of introducing new bugs or compatibility issues. Over time, multiple changes can cause different parts of the software to interact in unexpected ways, making it harder to maintain. If these issues aren't managed properly, the software becomes more complex and prone to failure, which affects its performance and reliability.

44. What are the elements to be considered in the System Model

The type and size of the software, the experience of use for reference to predecessors, difficulty level to obtain users' needs, development techniques and tools, the situation of the development team, development risks, the software development methods should be kept in mind. It is an important prerequisite to ensure the success of software development that designing a reasonable and suitable software development plan.

45. Define Adaptive Maintenance?

Adaptive maintenance defines as modifications and updating when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware and software.

46. Define the term WBS?

The full form of WBS is Work Breakdown Structure. Its **Work Breakdown Structure** includes dividing a large and complex project into simpler, manageable, and independent tasks. For constructing a work breakdown structure, each node is recursively decomposed into smaller sub-activities, until at the leaf level, the activities become undividable and independent. A WBS works on a top-down approach.

For more detail please refer Work breakdown structure article.

47. How to Find the Size of a Software Product?

Estimation of the size of the software is an essential part of Software Project Management. It helps the project manager to further predict the effort and time which will be needed to build the project. Various measures are used in project size estimation. Some of these are:

- Lines of Code
- Number of entities in ER diagram
- Total number of processes in detailed data flow diagram
- Function points

48. What is Concurrency, and How to Achieve it ?

Concurrency refers to a system's ability to execute numerous tasks or processes at the same time, ostensibly concurrently. It is a wider concept that includes the idea of completing many jobs in concurrent intervals. In a concurrent system, tasks can begin, run, and finish in overlapping time frames, increasing overall system efficiency and responsiveness. There are many programming language available that support concurrency using unthreading example Java, C++ etc.

49. Why is Modularization important in Software Engineering?

Modular programming makes your code easier to read by dividing it into functions that only deal with one part of the overall functionality. When compared to monolithic code, it can make your files significantly smaller and easier to read.

50. Which Process Model Removes Defects before Software get into trouble?

Clean room software engineering is a software development approach to producing quality software. In clean room software engineering, an efficient and good quality software product is delivered to the client as QA (Quality Assurance) is performed each and every phase of software development.

51. Difference between an EXE and DLL?

DLL refers to Dynamic Link Library, and EXE is an executable. An EXE assembly can execute in its own address space, whereas a DLL cannot. It does not have its own address space, therefore it must run within a host and requires a consumer to invoke it.

Conclusion

Preparing for a **Software Engineering Interview** means having a solid foundation of both the basics and more advanced topics, like **SDLC models**, **testing methods**, and **design principles**. By mastering these areas, practicing problem-solving, and staying up-to-date with new technologies, You will be ready to clear any interview confidently with the well knowledge.

React Interview Questions and Answers

Last Updated : 05 Aug, 2025

React is an efficient, flexible, and open-source **JavaScript library** that allows developers to create simple, fast, and scalable web applications. Jordan Walke, a software engineer who was working for Facebook, created React. Developers with a JavaScript background can easily develop web applications with React.

In this Top React Interview Questions article, we've covered the **70+ Interview Questions of React** that cover everything from basic to advanced React concepts such as **Virtual DOM, Components, State and Props, JSX, Hooks, Routing**, and more. Whether you are a **fresher** or an **experienced professional with 2 - 10 years of experience**, these React Interview Questions give you all the confidence you need to ace your next technical interview.

React Interview Questions For Freshers

Let's discuss some common questions that you should prepare for the interviews. These questions will help clear the interviews, especially for the front-end development role.

1. What is ReactJS?

ReactJS is a JavaScript library used to build reusable components for the view layer in the MVC architecture. It is used to build the Single Page Application (SPA) due to its component-based architecture, efficient re-rendering with the Virtual DOM, and ability to manage dynamic content without needing full page reloads. It is written in JSX.

- **Virtual DOM:** React uses a virtual DOM to efficiently update and render components, ensuring fast performance by minimizing direct DOM manipulations.
- **Component-Based Architecture:** React builds UI using reusable, isolated components, making code more modular, maintainable, and scalable.
- **Hooks**: React Hooks allow functional components to manage state and side effects, making them powerful and more flexible.
- **Server-Side Rendering (SSR)**: React can be used for server-side rendering, where HTML content is generated on the server and sent to the client. This improves the app's performance, especially for SEO.
- **React Router:** React Router enables navigation in a React application. It allows you to define different routes for different views in a single-page application (SPA).

2. What is the latest version of the React?

The latest stable version of React is **v19.1.0**, released on **March 28, 2025** . This version builds upon the major updates introduced in **React v19.0.0**, which was officially released on **December 5, 2024**.

3. Explain the MVC architecture.

The Model-View-Controller (MVC) framework is an architectural/design pattern that separates an application into three main logical components: Model, View, and Controller. Each architectural component is built to handle specific development aspects of an application. It isolates the business, logic, and presentation layers from each other

4. Explain the building blocks of React.

- **JSX:** It stands for JavaScript and XML and allows you to write HTML in React.
- **Props and State:** props are like function parameters and State is similar to variables.
- **Context:** This allows data to be passed through components as props in a hierarchy.
- **Virtual DOM:** It is a lightweight copy of the actual DOM which makes DOM manipulation easier.

5. Explain props and state in React with differences

Props are used to pass data from one component to another. The state is local data storage that is local to the component only and cannot be passed to other components.

Here is the difference table of props and state In react

PROPS	STATE
The Data is passed from one component to another.	The Data is passed within the component only.
It is Immutable (cannot be modified).	It is Mutable (can be modified).
Props can be used with state and functional components.	The state can be used only with the state components/class component (Before 16.0).
Props are read-only.	The state is both read and write.

6. What is virtual DOM in React?

interface by comparing the current and previous virtual DOM states using a process called **diffing**.

How Virtual DOM Works

- **Efficient Rendering**: The Virtual DOM is an in-memory representation of the actual DOM that React uses to optimize the process of updating and rendering UI changes.
- **Diffing Algorithm**: React compares the current and previous versions of the Virtual DOM using a diffing algorithm, identifying the minimal set of changes required to update the real DOM.
- **Batch Updates**: Instead of updating the real DOM immediately, React batches multiple changes to reduce unnecessary re-renders, improving performance.
- **Faster Updates**: Since updating the real DOM is slow, React minimizes direct DOM manipulations by only making updates where necessary after comparing the Virtual DOM.
- **Declarative UI**: With the Virtual DOM, React allows developers to write code in a declarative style, letting React handle when and how to efficiently update the UI.

7. Differentiate between Real DOM and virtual DOM?

Real DOM	Virtual DOM
The actual DOM, a tree-like structure representing the UI elements.	A lightweight copy of the Real DOM used to optimize updates.
Slower as it requires direct updates to the actual DOM.	Faster because it minimizes direct manipulation of the Real DOM.
Directly manipulates the Real DOM, causing re-	Updates are made to the Virtual DOM first, then changes are batched and only

Real DOM	Virtual DOM
Entire UI might need to be re-rendered when changes occur.	Only the necessary components are re-rendered, reducing unnecessary re-renders.
Less efficient due to repeated direct updates to the Real DOM.	More efficient by minimizing direct DOM manipulation and batch updates.

8. What is JSX?

JSX is basically a syntax extension of regular JavaScript and is used to create React elements. These elements are then rendered to the React DOM. All the React components are written in JSX. To embed any JavaScript expression in a piece of code written in JSX we will have to wrap that expression in curly braces {}.

Example of JSX: The name written in curly braces { } signifies JSX

```
const name = "Learner";
const element = (
    <h1>
        Hello,
        {name}.Welcome to GeeksforGeeks.
    </h1>
);
```

9. What are components and their type in React?

A Component is one of the core building blocks of React. In other words, we can say that every application you will develop in React will be made up of pieces called components. Components make the task of building UIs much easier.

- **Functional Components:** Functional components are simply JavaScript functions. Initially, they were limited in terms of features like state and lifecycle methods. However, with the introduction of Hooks, functional components can now use state, manage side effects, and access other features that were once exclusive to class components.
- **Class Components:** Class components are more complex than functional components. They are able to manage state, handle lifecycle methods, and can also interact with other components. Class components can pass data between each other via props, similar to functional components.

10. How do browsers read JSX?

In general, browsers are not capable of reading JSX and only can read pure JavaScript. The web browsers read JSX with the help of a transpiler. Transpilers are used to convert JSX into JavaScript. The transpiler used is called Babel.

11. Explain the steps to create a react application and print Hello World?

To install React, first, make sure Node is installed on your computer. After installing Node. Open the terminal and type the following command.

```
npx create-react-app <<Application_Name>>
```

Note: The above method is now deprecated, so use the below given method for creating the React application.

```
npm create vite@latest
```

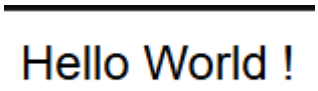
This is the first code of ReactJS Hello World!

```
import React from "react";
import "./App.css";
function App() {
    return <div className="App">Hello World !</div>;
}
export default App;
```

Output:

Type the following command to run the application

```
npm start
```



React app

12. How to create an event in React?

To create an event in React, attach an event handler like onClick, onChange, etc., to a JSX element. Define the handler function to specify the action when the event is triggered, such as updating state or executing logic.

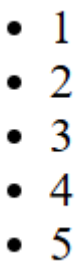
```
function Component() {
    doSomething(e);
    {
        e.preventDefault();
        // Some more response to the event
    }
    return <button onEvent={doSomething}></button>;
}
```

13. Explain the creation of a List in React?

Lists are very useful when it comes to developing the UI of any website. Lists are mainly used for displaying menus on a website. To create a list in React use the map method of array as follows.

```
const numbers = [1, 2, 3, 4, 5];

const updatedNums = numbers.map((number) => {
    return <li key={number}>{number}</li>;  // Add a unique "key" prop
});

const root = ReactDOM.createRoot(document.getElementById("root"));  //
Create the root
root.render(<ul>{updatedNums}</ul>);  // Render the list into the root
element
```

Output:



List

14. What is a key in React?

A key is a special string attribute you need to include when creating lists of elements in React. Keys are used in React to identify which items in the list are changed, updated, or deleted. In other words, we can say that keys are used to give an identity to the elements in the lists.

15. How to write a comment in React?

There are two ways to write comments in React.

- **Multi-line comment:** We can write multi-line comments in React using the asterisk format /* */.

```
/*
  This is a multi-line comment.
  It can span multiple lines.
*/
```

```
// This is a single-line comment
```

16. Explain the difference between React and Angular?

React	Angular
React is a JavaScript library. As it indicates react js updates only the virtual DOM is present and the data flow is always in a single direction.	Angular is a framework. Angular updates the Real DOM and the data flow is ensured in the architecture in both directions.
React is more simplified as it follows MVC ie., Model View Control.	The architecture is complex as it follows MVVM models ie., Model View-ViewModel.
It is highly scalable.	It is less scalable than React JS.
It supports Uni-directional data binding which is one-way data binding.	It supports Bi-directional data binding which is two way data binding.
It has a virtual DOM.	It has regular DOM.

17. Explain the use of render method in React?

React renders HTML to the web page by using a function called render(). The purpose of the function is to display the specified HTML code inside the specified HTML element. In the render() method, we can read props and state and return our JSX code to the root component of our app.

18. What is state in React?

behaviour of the component. In other words, the State of a component is an object that holds some information that may change over the lifetime of the component.

19. Explain props in React?

React allows us to pass information to a Component using something called props (which stands for properties). Props are objects which can be used inside a component

We can access any props inside from the component's class to which the props is passed. The props can be accessed as shown below:

```
this.props.propName;
```

20. What is higher-order component in React?

Higher-order components or HOC is the advanced method of reusing the component functionality logic. It simply takes the original component and returns the enhanced component. HOC are beneficial as they are easy to code and read. Also, helps to get rid of copying the same logic in every component.

21. Explain the difference between functional and class component in React?

Functional Components	Class Components
A functional component is just a plain JavaScript pure function that accepts props as an argument	A class component requires you to extend from React. Component and create a render function

Functional Components	Class Components
Also known as Stateless components	Also known as Stateful components
React lifecycle methods (for example, componentDidMount) cannot be used in functional components.	React lifecycle methods can be used inside class components (for example, componentDidMount).
Constructors are not used.	Constructor is used as it needs to store state.
Uses hooks like useState for managing state.	Uses this.state and this.setState for state management.

22. Explain one way data binding in React?

ReactJS uses one-way data binding which can be Component to View or View to Component. It is also known as one-way data flow, which means the data has one, and only one way to be transferred to other parts of the application. In essence, this means child components are not able to update the data that is coming from the parent component. It is easy to debug and less prone to errors.

23. What is Context API in React?

The Context API is a way to share data (such as theme, language preference, etc.) across components without having to pass props down manually at every level. It provides a Provider component to set the value and a Consumer component or useContext() hook to access it.

shouldComponentUpdate() is a lifecycle method that allows you to control whether a component should re-render when it receives new props or state. If it returns false, React will skip the re-render process for that component.

25. What is the use of dangerouslySetInnerHTML in React?

dangerouslySetInnerHTML is an attribute used to set raw HTML inside a component. It is generally discouraged because it can expose the application to XSS (cross-site scripting) attacks, but it is sometimes used for rendering third-party HTML content.

26. What are Pure Components in React?

A Pure Component is a type of React component that only re-renders if the props or state it receives change. React provides React.PureComponent, which is a base class that automatically performs a shallow comparison of props and state to determine if a re-render is necessary.

27. What is the significance of setState() in React?

setState() is a method used to update the state of a component. When the state is updated, React re-renders the component and its child components to reflect the changes.

React Intermediate Interview Questions

Here, we cover all intermediate level react interview questions with answers, that recommeded for freshers as well as for experienced professionals having 1 - 2 years of experience.

28. What is conditional rendering in React?

allowing for dynamic and responsive user interfaces in React applications.

Let us look at this sample code to understand conditional rendering.

```
{isLoggedIn == false ? <DisplayLoggedOut /> : <DisplayLoggedIn
/>}
```

Here if the boolean isLoggedIn is false then the DisplayLoggedOut component will be rendered otherwise DisplayLoggedIn component will be rendered.

29. What is React Router?

React Router is a standard library for routing in React. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.

To install react router type the following command.

```
npm i react-router-dom
```

30. Explain the components of a react-router.

The main components of a react-router are:

1. **Router(usually imported as BrowserRouter):** It is the parent component that is used to store all of the other components. Everything within this will be part of the routing functionality
2. **Switch:** The switch component is used to render only the first route that matches the location rather than rendering all matching routes.
3. **Route:** This component checks the current URL and displays the component associated with that exact path. All routes are placed within the switch components.

React Routing	Conventional Routing
Used in Single Page Applications (SPA).	Typically used in Multi-Page Applications (MPA).
No full page reloads. React updates only the necessary parts of the UI.	Full page reload is triggered for every navigation request.
React uses a **client-side router** (e.g., React Router) to handle navigation and manage different views within the same page.	Uses **server-side routing** where the server responds with new HTML for each navigation.
Routes are managed by JavaScript, and the browser's address bar reflects the application's state.	Routes correspond to different server-side routes, and the server responds with new HTML files.

32. Explain the lifecycle methods of components

A React Component can go through four stages of its life as follows.

- **Initialization:** This is the stage where the component is constructed with the given Props and default state. This is done in the constructor of a Component Class.
- **Mounting:** Mounting is the stage of rendering the JSX returned by the render method itself.
- **Updating:** Updating is the stage when the state of a component is updated and the application is repainted.
- **Unmounting:** As the name suggests Unmounting is the final step of the component lifecycle where the component is removed from the page.

33. Explain the methods used in mounting phase of

Mounting is the phase of the component lifecycle when the initialization of the component is completed and the component is mounted on the DOM and rendered for the first time on the webpage. he mounting phase consists of two such predefined functions as described below

- componentWillMount() Function: This function is invoked right before the component is mounted on the DOM.
- componentDidMount() Function: This function is invoked right after the component is mounted on the DOM.

34. What is this.setState function in React?

We use the setState() method to change the state object. It ensures that the component has been updated and calls for re-rendering of the component. The state object of a component may contain multiple attributes and React allows using setState() function to update only a subset of those attributes as well as using multiple setState() methods to update each attribute value independently.

35. What is the use of ref in React?

Refs are a function provided by React to access the DOM element and the React element that you might have created on your own. They are used in cases where we want to change the value of a child component, without making use of props and all. They have wide functionality as we can use callbacks with them.

The syntax to use ref is :

```
const node = this.myCallRef.current;
```

36. What are hooks in React?

any existing React concepts. Instead, Hooks provide a direct API to react concepts such as props, state, context, refs and life-cycle

37. Explain the useState hook in React?

The most used hook in React is the useState() hook. Using this hook we can declare a state variable inside a function but only one state variable can be declared using a single useState() hook. Whenever the useState() hook is used, the value of the state variable is changed and the new variable is stored in a new cell in the stack.

Syntax:

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

- **state**: The current state value.
- **setState:** A function used to update the state value.
- **initialState**: The initial value of the state.

38. Explain the useEffect hook in React?

The useEffect hook in React eliminates the side effect of using class based components. It is used as an alternative to componentDidUpdate() method. The useEffect hook accepts two arguments where second argument is optional.

```
useEffect(function, dependency)
```

The dependency decides when the component will be updated again after rendering.

39. What is React Fragments?

when we are trying to render more than one root element we have to put the entire content inside the 'div' tag which is not loved by many

```
<React.Fragment>
    <h2>Child-1</h2>
    <p> Child-2</p>
</React.Fragment>
```

40. What is a react developer tool?

React Developer Tools is a Chrome DevTools extension for the React JavaScript library. A very useful tool, if you are working on React applications. This extension adds React debugging tools to the Chrome Developer Tools. It helps you to inspect and edit the React component tree that builds the page, and for each component, one can check the props, the state, hooks, etc.

41. How to use styles in ReactJS?

CSS modules are a way to locally scope the content of your CSS file. We can create a CSS module file by naming our CSS file as App.modules.css and then it can be imported inside App.js file using the special syntax mentioned below.

Syntax:

```
import styles from './App.module.css';
```

42. Explain styled components in React?

Styled-component Module allows us to write CSS within JavaScript in a very modular and reusable way in React. Instead of having one global CSS file for a React project, we can use styled-component for enhancing the developer experience. It also removes the mapping between components and styles – using components as a low-level styling construct

Using the below code we can custom style a button in React

```
import styled from 'styled-components'

const Button = styled.div`
    width : 100px ;
    cursor: pointer ;
    text-decoration : none;
`
export default Button;
```

43. What is prop drilling and its disadvantages?

Prop drilling is basically a situation when the same data is being sent at almost every level due to requirements in the final level. The problem with Prop Drilling is that whenever data from the Parent component will be needed, it would have to come from each level, Regardless of the fact that it is not needed there and simply needed in last.

44. What is conditional rendering in React?

Conditional rendering in React is used when you want to render different UI elements based on certain conditions. For example, rendering a login button if a user is not logged in or rendering a logout button when the user is logged in.

```
const isLoggedIn = true;
return (
  <div>
    {isLoggedIn ?<button>Logout</button> : <button>Login</button>}
  </div>
);
```

45. What are controlled components in React?

Controlled components are React components where the form data is handled by the state within the component. The form elements value

*For further reading, check out our dedicated article on **Intermediate ReactJS Intermediate Interview Questions**. Inside, you'll discover over 20 questions with detailed answers.*

React Interview Questions for Experienced

Here, we cover **advanced react interview questions with answers** for experienced professionals, who have over 5+ years of experience.

46. What is custom hooks in React?

Custom hooks are normal JavaScript functions whose names start with "use" and they may call other hooks. We use custom hooks to maintain the DRY concept that is Don't Repeat Yourself. It helps us to write a logic once and use it anywhere in the code.

47. How to optimize a React code?

We can improve our react code by following these practices:

- Using binding functions in constructors
- Eliminating the use of inline attributes as they slow the process of loading
- Avoiding extra tags by using React fragments
- Lazy loading

48. What is the difference between useref and createRef in React ?

Here is the difference table of useref and createRef in React

useRef	createRef
It is a hook.	It is a function.

useRef	createRef
It saves its value between re-renders in a functional component.	It creates a new ref for every re-render.
It returns a mutable ref object (i.e., can be changed).	It returns a read-only ref object (cannot be modified directly).
Used for accessing DOM elements, persisting values, or managing timers in functional components.	Used for class components, especially when referencing DOM elements.
const myRef = useRef();	const myRef = React.createRef();

49. What is react-redux?

React-redux is a state management tool which makes it easier to pass these states from one component to another irrespective of their position in the component tree and hence prevents the complexity of the application. As the number of components in our application increases it becomes difficult to pass state as props to multiple components. To overcome this situation we use react-redux

50. What are benefits of using react-redux?

They are several benfits of using react-redux such as:

- It provides centralized state management i.e. a single store for whole application
- It optimizes performance as it prevents re-rendering of component
- Makes the process of debugging easier
- Since it offers persistent state management therefore storing data for long times become easier

There are four fundamental concepts of redux in react which decide how the data will flow through components

- Redux Store: It is an object that holds the application state
- Action Creators: These are functions that return actions (objects).
- Actions: Actions are simple objects which conventionally have two properties- type and payload
- Reducers: Reducers are pure functions that update the state of the application in response to actions

52. How can we combine multiple reducers in React?

When working with Redux we sometimes require multiple reducers. In many cases, multiple actions are needed, resulting in the requirement of multiple reducers. However, this can become problematic when creating the Redux store. To manage the multiple reducers we have function called combineReducers in the redux. This basically helps to combine multiple reducers into a single unit and use them.

Syntax

```
import { combineReducers } from "redux";
const rootReducer = combineReducers({
    books: BooksReducer,
    activeBook: ActiveBook
});
```

53. Explain CORS in React?

In ReactJS, Cross-Origin Resource Sharing (CORS) refers to the method that allows you to make requests to the server deployed at a different domain. As a reference, if the frontend and backend are at two different domains, we need CORS there.

We can setup CORS evironment in frontend using two methods:

54. What is axios and how to use it in React?

Axios, which is a popular library is mainly used to send asynchronous HTTP requests to REST endpoints. This library is very useful to perform CRUD operations.

- This popular library is used to communicate with the backend. Axios supports the Promise API, native to JS ES6.
- Using Axios we make API requests in our application. Once the request is made we get the data in Return, and then we use this data in our project.

To install axios package in react use the following command.

```
npm i axios
```

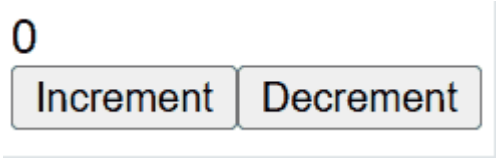
55. Write a program to create a counter with increment and decrement?

We can create the counter app with increment and decrement by writing the below code in the terminal:

```
import React, { useState } from "react";

const App = () => {
    const [counter, setCounter] = useState(0)
    const handleClick1 = () => {
        setCounter(counter + 1)
    }

    const handleClick2 = () => {
        setCounter(counter - 1)
    }

    return (
        <div>
            <div>
                {counter}
            </div>
            <div className="buttons">
                <button onClick={handleClick1}>
```

```
            </div>
        </div>
    )
}

export default App
```

Output:



Counter App using React

56. Explain why and how to update state of components using callback?

It is advised to use a callback-based approach to update the state using setState because it solves lots of bugs upfront that may occur in the future.We can use the following syntax to update state using callback

```
this.setState(st => {
    return(
        st.stateName1 = state1UpdatedValue,
        st.stateName2 = state2UpdatedValue
    )
})
```

57. What is React-Material UI?

React Material UI is an open-source React component library, offering prebuilt components for creating React applications. Developed by Google in 2014, it's compatible with JavaScript frameworks like Angular.js and Vue.js. Renowned for its quality designs and easy customization, it's favored by developers for rapid development.

Flux architecture in Redux is a design pattern used for managing application state in a unidirectional data flow. In this architecture, actions are dispatched to modify the store, which holds the entire application state. The store sends the updated state to the view (UI), and the cycle repeats when new actions are triggered. Redux follows this structure to ensure a predictable and maintainable state management system for large applications.

59. What are custom hooks in React?

Custom hooks are user-defined functions that use built-in hooks like useState, useEffect, etc., to reuse stateful logic across components. They allow you to extract and share common logic.

60. How can you optimize React performance?

Optimizing React performance can be achieved using:

- React.memo() for preventing unnecessary re-renders.
- Lazy loading components with React.lazy() and Suspense.
- Using useMemo() and useCallback() hooks to memoize values and functions.
- Avoiding unnecessary state updates.

61. What is the Strict Mode in React?

Strict Mode in React is a tool that helps developers find and fix problems in their app while developing. It only works in development and doesn't affect the app when it's running in production. When enabled, it checks for potential issues, such as old features that should no longer be used, and gives warnings to help avoid bugs.

We can enable the strict mode by by wrapping your application or specific components inside the **<React.StrictMode>**.

```
  return (
    <div>
      <h1>Hello, World!</h1>
    </div>
  );
};

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
);
```

62. What is Redux and how does it work with React?

Redux is a state management library that helps manage the application state globally. It uses actions, reducers, and a central store to control state. React-Redux is used to connect Redux state to React components.

63. How does React handle concurrency?

React's Concurrent Mode is a set of features that help React apps stay responsive and gracefully adjust to the user's device capabilities and network speed. It allows React to interrupt rendering and prioritize high-priority tasks.

64. How does React handle server-side rendering (SSR)?

Server-side rendering (SSR) is the process of rendering a React application on the server and sending the fully rendered HTML to the client. This improves the initial page load performance and SEO.

65. What are forms in React?

In React, **forms** are a way to capture user input, such as text fields, checkboxes, radio buttons, and buttons. React provides controlled and

66. How to create forms in React?

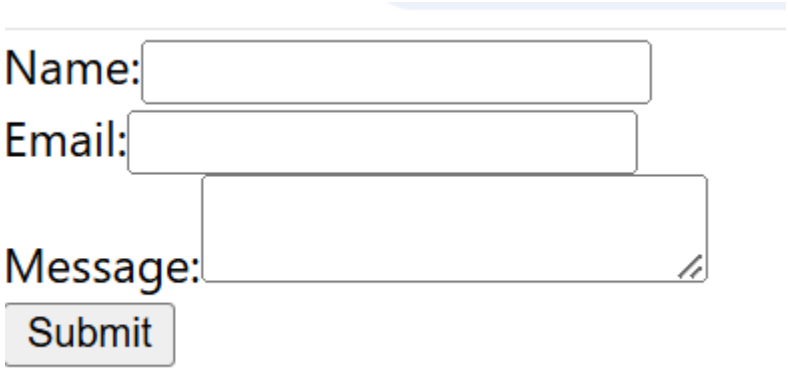
Creating forms in React involves handling user input and managing the form's state. In React, form elements like <input>, <textarea>, and <select> are controlled components, meaning their values are controlled by the React state.

- **Create the State for the Form:** Use the useState hook to manage form input values.
- **Set the Value of Form Elements:** Bind the form elements' value attribute to the corresponding state variable to make the inputs controlled.
- **Handle User Input**: Use an onChange event handler to update the state whenever the user types in the form fields.
- **Submit the Form:** Handle the form submission with an onSubmit event, and prevent the default behavior to stop the page from refreshing.

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

function MyForm() {
    // State to store input values
    const [name, setName] = useState('');
    const [email, setEmail] = useState('');
    const [message, setMessage] = useState('');

    // Handle input change for each field
    const handleNameChange = (e) => setName(e.target.value);
    const handleEmailChange = (e) => setEmail(e.target.value);
    const handleMessageChange = (e) => setMessage(e.target.value);

    // Handle form submission
    const handleSubmit = (e) => {
        e.preventDefault();  // Prevent default form submission behavior
        console.log('Form submitted:', { name, email, message });
    };

    return (
        <form onSubmit={handleSubmit}>
            <div>
                <label>Name:</label>
                <input type="text" value={name} onChange=
```

```
            </div>
            <div>
                <label>Message:</label>
                <textarea value={message} onChange={handleMessageChange}
/>
            </div>
            <button type="submit">Submit</button>
        </form>
    );
}

export default MyForm;
```

Output:



React Form

67. What is Lazy Loading in React?

Lazy Loading in React is a technique used to load components only when they are needed, instead of loading everything at once when the app starts. This helps improve the performance of the app by reducing the initial loading time.

In React, React.lazy() is used to implement lazy loading for components, which allows you to split your code into smaller bundles and load them only when required.

68. How is React different from React Native?

React	React Native

React	React Native
Focuses on developing web applications.	Focuses on developing native mobile applications for iOS and Android.
Renders HTML using the **Virtual DOM**.	Renders native mobile components using **native APIs** (e.g., `View`, `Text`, `Image`).
Uses **web technologies** like HTML, CSS, and JavaScript.	Uses **native mobile components** and styles for building mobile UIs.
Runs in the browser on a web server.	Runs on mobile devices and communicates with native code.
Can be deployed on browsers like Chrome, Firefox, etc.	Can be deployed as native apps on iOS and Android platforms.

69. What is Memoization in React?

Memoization in React is an optimization technique used to improve the performance of a component by preventing unnecessary re-renders. It involves caching the results of expensive function calls and reusing the cached result when the inputs to the function haven't changed. This is particularly useful when a component's rendering process is slow, and you want to avoid recalculating or re-rendering unnecessarily.

How Memoization Works in React:

React provides two key APIs for memoization:

1. **`React.memo()`**: This is a higher-order component (HOC) used to prevent re-renders of functional components if their props have not

70. What is Reoncillation in React?

Reconciliation in React is the process of updating the **DOM** when a component's state or props change. React uses the **Virtual DOM** to efficiently determine what parts of the actual DOM need to be updated. Here's a simplified overview:

- **Triggering Reconciliation**: React triggers reconciliation when state or props change (e.g., through `setState()` or `useState()`).
- **Virtual DOM Diffing**: React compares the old Virtual DOM with the new one to identify changes.
- **Efficient Updates**: React calculates the minimal set of changes and applies them to the real DOM.
- **Fiber Algorithm**: React's Fiber algorithm allows the reconciliation process to be interrupted and prioritized, improving performance for complex tasks.

Top 10 Commonly Asked ReactJS Interview Questions

1. **What is React?**
2. **What are the key features of React?**
3. **What is the Virtual DOM, and how does it work?**
4. **What are React components?**
5. **What is the difference between a functional component and a class component in React?**
6. **What are React Hooks?**
7. **What is `useState` in React?**
8. **What are Custom Hooks?**
9. **What is JSX?**
10. **What is the difference between `props` and `state` in React?**

JavaScript Interview Questions and Answers

Last Updated : 14 Aug, 2025

JavaScript is the most used programming language for developing websites, web servers, mobile applications, and many other platforms.

In Both Front-end and Back-end Interviews, JavaScript was asked, and its difficulty depends upon the on your profile and company. Here, we compiled 70+ JS Interview questions on every difficulty level

JavaScript Interview Questions for Freshers

Let's discuss some common questions that you should prepare for the interviews. These questions will help clear the interviews, especially for the frontend development role.

1. How to concatenate two strings in JavaScript?

we can concatenate two strings using the "+" operator as code below:

```
let str1 = "Hello";
let str2 = "World";
let result = str1 + str2;
console.log(result);
```

Output

```
Hello World
```

2. What would be the result of 3+2+"7"?

57

- `3 + 2` is evaluated first, and since both are numbers, it results in `5`.
- Then, `5 + "7"` is calculated. Since one of the operands is a string (`"7"`), JavaScript converts the number `5` to a string and concatenates it with `"7"`, resulting in `"57"`.

3. Are JavaScript and Java related?

No, there names sounds similar but they are not related in any terms, below are some key differences:

Java	JavaScript
Java is a strongly typed language and variables must be declared first to use in the program. In Java, the type of a variable is checked at compile-time.	JavaScript is a loosely typed language and has a more relaxed syntax and rules.
Java is an object-oriented programming language primarily used for developing complex enterprise applications.	JavaScript is a scripting language used for creating interactive and dynamic web pages.
Java applications can run in any virtual machine(JVM) or browser.	JavaScript code used to run only in the browser, but now it can run on the server via Node.js.
Objects of Java are class-based even we can't make any program in java without creating a class.	JavaScript Objects are prototype-based.

JavaScript is a Dynamically Typed language which means the developers do not have to provide data types of variables. Dynamically typed languages have advantages like easy to learn, flexible and faster development. However, there are disadvantages also like slowness and runtime errors. TypeScript is a superset of JavaScript that primarily uses static typing and has support for dynamic typing. TypeScript is developed and maintained by Microsoft, it compiles down to plain JavaScript, making it compatible with all JavaScript environments, including web browsers and Node.js.

5. What is a Variable Scope in JavaScript?

In JavaScript, we have each variable are accessed and modified through either one of the given scope:

- **Global Scope**: Outermost level (accessible everywhere).
- **Local Scope**: Inner functions can access variables from their parent functions due to lexical scoping.
- **Function Scope**: Variables are confined to the function they are declared in.
- **Block Scope**: Variables declared with `let` or `const` are confined to the nearest block (loops, conditionals, etc.).

6. What the difference between Lexical and Dynamic Scoping?

Lexical Scoping (Static Scoping)

- **Definition**: The scope of a variable is determined **by its position in the source code** at the time of writing.
- **JavaScript uses lexical scoping**.
- The **inner function looks up variables in the outer function** where it was **defined**, not where it was **called**.

Dynamic Scoping (Not in JS)

- **Languages like older versions of Lisp or Bash** use dynamic scoping.
- The function **uses variables from the function that called it**, even if it was defined else.

7. What is the use of the isNaN function?

The number isNan function determines whether the passed value is NaN (Not a number) and is of the type "Number". In JavaScript, the value NaN is considered a type of number. It returns true if the argument is not a number, else it returns false.

8. What does this code log?

```
const arr = [1, 2, 3];
arr[10] = 99;
console.log(arr.length);
```

Output:

```
11
```

Explanation: When you assign to `arr[10]` you create empty slots from index 3 to 9, making the new `length` one more than the highest index: `10 + 1 = 11`.

9. What is negative infinity?

The negative infinity is a constant value represents the lowest available value. It means that no other number is lesser than this value. It can be generate using a self-made function or by an arithmetic operation. JavaScript shows the NEGATIVE_INFINITY value as -Infinity.

Yes, it is possible to break the JavaScript code into several lines in a string statement. It can be broken by using the **'\n'** (backslash n).

Example:

```
console.log("A Online Computer Science Portal\n for Geeks")
```

The code-breaking line is avoid by JavaScript which is not preferable.

```
let gfg= 10, GFG = 5,
Geeks =
gfg + GFG;
console.log(Geeks)
```

11. What are "truthy" and "falsy" values in JavaScript

- **Falsy**: false, 0, "" (empty string), null, undefined, NaN.
- **Truthy**: Everything else (e.g., any non-empty string, any non-zero number, objects, arrays).

12. What are undeclared and undefined variables?

- Undefined**:** It occurs when a variable is declare but not assign any value. Undefined is not a keyword.
- Undeclared**:** It occurs when we try to access any variable which is not initialize or declare earlier using the var or const keyword. If we use 'typeof' operator to get the value of an undeclare variable, we will face the runtime error with the return value as "undefined". The scope of the undeclare variables is always global.

13. What will be the result of this expression?

```
console.log(null ?? 'default');
console.log(undefined ?? 'default');
console.log(false ?? 'default');
```

```
default
default
false
```

Explanation:

The nullish coalescing operator `??` returns the right-hand side only if the left is `null` or `undefined`. So:

- `null ?? 'default'` → `'default'`
- `undefined ?? 'default'` → `'default'`
- `false ?? 'default'` → `false` (because `false` is neither `null` nor `undefined`)

14. Write a JavaScript code for adding new elements dynamically.

```
<html>
<head>
</head>
<body>
    <button onclick="create()">
        Click Here!
    </button>

    <script>
        function create() {
            let geeks = document.createElement('geeks');
            geeks.textContent = "Geeksforgeeks";
            geeks.setAttribute('class', 'note');
            document.body.appendChild(geeks);
        }
    </script>
</body>
</html>
```

15. What are global variables? How are these variables

In contrast, global variables are the variables that define outside of functions. These variables have a global scope, so they can be used by any function without passing them to the function as parameters.

Example:

```
let petName = "Rocky"; // Global Variable
myFunction();

function myFunction() {
    console.log("Inside myFunction - Type of petName:", typeof
petName);
    console.log("Inside myFunction - petName:", petName);
}

console.log("Outside myFunction - Type of petName:", typeof petNa
console.log("Outside myFunction - petName:", petName);
```

Output

```
Inside myFunction - Type of petName: string
Inside myFunction - petName: Rocky
Outside myFunction - Type of petName: string
Outside myFunction - petName: Rocky
```

It is difficult to debug and test the code that relies on global variables.

16. What do you mean by Null in JavaScript?

The null value represents that no value or no object. It is known as empty value/object.

17. How to delete property-specific values?

The **delete keyword** deletes the whole property and all the values at once like

18. What will be the output of this code?

```
let x = 0;
console.log(x++);
console.log(++x);
```

Output:

```
0
2
```

Explanation:

- x++ returns the current value (0), then increments → x becomes 1.
- ++x increments first (1 → 2), then returns the new value (2).

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

undefined:

- A **primitive value** automatically assigned to:
  - **Uninitialized variables**
  - **Missing function arguments**
  - **Missing object properties**
- It means: **"value not assigned yet"**

```
let x;
console.log(x); // undefined

function foo(a) {
  console.log(a); // undefined if no argument is passed
}
foo();
```

null:

- It means: **"value is deliberately empty"**

```
let user = null; // explicit assignment
console.log(user); // null
```

20. What is the output of this snippet?

```
const a = [1, 2, 3];
const b = [1, 2, 3];
console.log(a == b, a === b);
```

Output:

```
false
false
```

Explanation: Arrays are objects and compared by reference. `a` and `b` are distinct objects, so both loose (`==`) and strict (`===`) comparisons yield `false`.

21. What is a prompt box?

The prompt box is a dialog box with an optional message prompting the user to input some text. It is often used if the user wants to input a value before entering a page. It returns a string containing the text entered by the user, or null.

22. What is the 'this' keyword in JavaScript?

Functions in JavaScript are essential objects. Like objects, it can be assign to variables, pass to other functions, and return from functions. And much like objects, they have their own properties. 'this' stores the current execution context of the JavaScript program. Thus, when it use inside a function, the value of 'this' will change depending on how the

23. Explain the working of timers in JavaScript. Also explain the drawbacks of using the timer, if any.

The timer executes some specific code at a specific time or any small amount of code in repetition to do that you need to use the functions **setTimout, setInterval,** and **clearInterval**. If the JavaScript code sets the timer to 2 minutes and when the times are up then the page displays an alert message "times up". The **setTimeout()** method calls a function or evaluates an expression after a specified number of milliseconds.

24. What will be logged by this code?

```
for (let i = 0; i < 3; i++) {
  setTimeout(() => console.log(i), i * 100);
}
```

Output:

```
0 1 2
```

Explanation: Using `let i` in the loop gives each callback its own `i` binding. The timeouts fire after 0ms, 100ms, and 200ms, logging `0`, then `1`, then `2`.

25. What is the difference between ViewState and SessionState?

- **ViewState:** It is specific to a single page in a session.
- **SessionState:** It is user specific that can access all the data on the web pages.

26. How to submit a form using JavaScript?

27. Does JavaScript support automatic type conversion?

Yes, JavaScript supports automatic type conversion.

28. What is a template literal in JavaScript?

A **template literal** in JavaScript is a way to define strings that allow embedded expressions and multi-line formatting. It uses backticks (`) instead of quotes and supports `${}` for embedding variables or expressions inside the string.

29. What is a higher-order function in JavaScript?

A **higher-order function** in JavaScript is a function that either takes one or more functions as arguments, or returns a function as its result. These functions allow for more abstract and reusable code, enabling functional programming patterns

For example, map() and filter() are higher-order functions because they take callback functions as arguments.

JavaScript Intermediate Interview Questions

30. What are all the looping structures in JavaScript?

- **while loop:** A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.
- **for loop:** A for loop provides a concise way of writing the loop structure. Unlike a while loop, for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.
- **do while:** A do-while loop is similar to while loop with the only difference that it checks the condition after executing the

Lexical scope in JavaScript refers to the way variables are resolved based on their location in the source code. A variable's scope is determined by the position of the code where it is defined, and it is accessible to any nested functions or blocks. This means that functions have access to variables in their own scope and the outer (lexical) scopes, but not to variables in inner scopes.

```
let outer = "I am outside!";
function inner() {
    console.log(outer);
}
inner();
```

In this example, inner() can access the outer variable because of lexical scoping.

32. How does lexical scoping work with the `this` keyword in JavaScript?

In JavaScript, lexical scoping primarily applies to variable resolution, while the behavior of the this keyword is determined by how a function is called, not by its position in the code. The value of this is dynamically determined at runtime based on the function's context (e.g., whether it's called as a method, in a global context, or with call, apply, or bind).

```
const obj = {
    name: "JavaScript",
    greet: function () {
        console.log(this.name);
    }
};
obj.greet(); // "JavaScript"
```

Here, this refers to obj because the function is called as a method of the object. Lexical scoping affects variable lookups but doesn't alter

The readFile() functions is used for reading operation.

```
readFile( Path, Options, Callback)
```

The writeFile() functions is used for writing operation.

```
writeFile( Path, Data, Callback)
```

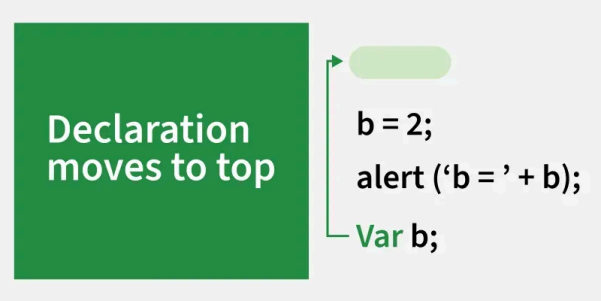
34. What is called Variable typing in JavaScript?

The **variable typing** is the type of variable used to store a number and using that same variable to assign a “string”.

```
Geeks = 42;
Geeks = "GeeksforGeeks";
```

35. What is hoisting in JavaScript?

Hoisting in JavaScript is the behavior where variable and function declarations are moved to the top of their containing scope during compilation, before the code is executed.



This means you can reference variables and functions before they are declared in the code. However, only declarations are hoisted, not initializations.

```
console.log(a); // undefined
var a = 5;
```

In this case, the declaration of a is hoisted, but its value (5) is not

36. How to convert the string of any base to integer in JavaScript?

In JavaScript, parseInt() function is used to convert the string to an integer. This function returns an integer of base which is specified in second argument of parseInt() function. The parseInt() function returns Nan (not a number) when the string doesn't contain number.

37. Explain how to detect the operating system on the client machine?

To detect the operating system on the client machine, one can simply use navigator.appVersion or navigator.userAgent property. The Navigator appVersion property is a read-only property and it returns the string that represents the version information of the browser.

38. What are the types of Pop up boxes available in JavaScript?

There are three types of pop boxes available in JavaScript.

- Alert
- Confirm
- Prompt

39. What is the use of void(0) ?

The void(0) is used to call another method without refreshing the page during the calling time parameter "zero" will be passed.

40. What are JavaScript modules, and how do you import/export them?

JavaScript modules allow you to split your code into smaller, reusable pieces. They enable the export of variables, functions, or objects from

```
// In file1.js
export const greet = () => "Hello";

// In file2.js
import { greet } from './file1';
console.log(greet());
```

Modules help organize code and avoid global namespace pollution. They are natively supported in modern JavaScript through import and export statements.

41. What are WeakMap and WeakSet, and how are they different from Map and Set?

A WeakMap is a collection of key-value pairs where keys are objects and the values can be any data type. The key-value pairs in a WeakMap are "weakly" held, meaning if no other references to a key exist, the entry can be garbage collected. A **WeakSet** is a collection of unique objects, and like WeakMap, the objects are weakly held.

The main difference from **Map** and **Set** is that in **Map** and **Set**, entries are strongly held, meaning they prevent garbage collection, while in **WeakMap** and WeakSet, entries can be garbage collected if no other references to the objects exist.

42. What is the role of the setImmediate function in Node.js, and how is it different from setTimeout?

In Node.js, the `setImmediate()` function schedules a callback to be executed in the next iteration of the event loop, specifically after the current event loop phase (which includes I/O events). It's commonly used for deferring tasks to be executed once the current operation completes.

The key difference between `setImmediate()` and `setTimeout()` is that `setImmediate()` runs after the current event loop iteration, following I/O

I/O or other tasks. While `setTimeout()` schedules tasks with a minimum delay, `setImmediate()` executes as soon as the event loop reaches a free point in the "check" phase, after I/O events have been processed.

JavaScript Interview Questions for Experienced

43. What is the 'Strict' mode in JavaScript and how can it be enabled?

Strict Mode is a new feature in ECMAScript 5 that allows you to place a program or a function in a "strict" operating context. This strict context prevents certain actions from being taken and throws more exceptions. The statement "use strict" instructs the browser to use the Strict mode, which is a reduced and safer feature set of JavaScript.

44. What are the advantages and disadvantages of using `async/await` over traditional callbacks or promises?

Advantages of async/await:

- **Improved Readability** : Async/await makes asynchronous code look like synchronous code, making it easier to read and maintain.
- **Simplified Error Handling** : With try/catch, error handling is more straightforward compared to .catch() with promises or callback-based error handling.
- **Avoids callback hell** : It eliminates deeply nested callbacks, reducing complexity in asynchronous logic.

Disadvantages of async/await:

- **Requires modern JavaScript** : It's supported in ES2017 and above, so older environments may need transpiling.
- **Limited concurrency control** : Unlike promises with .all() or .race(), async/await can be less flexible for handling multiple parallel tasks.

The closure is created when a child functions to keep the environment of the parent's scope even after the parent's function has already executed. The Closure is a locally declared variable related to a function. The closure will provide better control over the code when using them.

```
function foo() {
    let b = 1;
    function inner() {
        return b;
    }
    return inner;
}
let get_func_inner = foo();

console.log(get_func_inner());
console.log(get_func_inner());
console.log(get_func_inner());
```

Output

```
1
1
1
```

46. What is the difference between call() and apply() methods ?

Both methods are used in a different situation

- **call() Method:** It calls the method, taking the owner object as argument. The keyword this refers to the 'owner' of the function or the object it belongs to. We can call a method that can be used on different objects.
- **apply() Method:** The apply() method is used to write methods, which can be used on different objects. It is different from the

47. How to target a particular frame from a hyperlink in JavaScript ?

This can be done by using the **target** attribute in the hyperlink. Like

```
<a href="/geeksforgeeks.htm" target="newframe">New Page</a>
```

48. Write the errors shown in JavaScript?

There are three different types of errors in JavaScript.

- Syntax error **:** A syntax error is an error in the syntax of a sequence of characters or tokens that are intended to be written in a particular programming language.
- **Logical error:** It is the most difficult error to be traced as it is the error on the logical part of the coding or logical error is a bug in a program that causes to operate incorrectly and terminate abnormally.
- Runtime Error **:** A runtime error is an error that occurs during the running of the program, also known as an exception.

49. What is the difference between JavaScript and Jscript?

JavaScript

- It is a scripting language developed by Netscape.
- It is used to design client and server-side applications.
- It is completely independent of Java language.

JScript

- It is a scripting language developed by Microsoft.
- It is used to design active online content for the word wide Web.

50. How many ways an HTML element can be accessed in

There are four possible ways to access HTML elements in JavaScript which are:

- getElementById() Method: It is used to get the element by its id name.
- getElementsByClass() Method: It is used to get all the elements that have the given classname.
- getElementsByTagName() Method: It is used to get all the elements that have the given tag name.
- querySelector() Method: This function takes CSS style selector and returns the first selected element.

51. What is an event bubbling in JavaScript?

Consider a situation an element is present inside another element and both of them handle an event. When an event occurs in bubbling, the innermost element handles the event first, then the outer, and so on.

52. Explain the concept of memoization in JavaScript?

Memoization in JavaScript is an optimization technique that stores the results of expensive function calls and reuses them when the same inputs occur again. This reduces the number of computations by caching the results. Memoization is typically implemented using an object or a map to store function arguments and their corresponding results. When the function is called with the same arguments, the cached result is returned instead of recalculating it. This improves performance, especially for functions with repeated calls and expensive computations.

53. What is the difference between == and === in JavaScript?

In JavaScript, == is the **loose equality** operator, which compares two values for equality after performing type coercion if necessary. This

=== is the **strict equality** operator, which compares both the values and their types, without performing type conversion.

54. Explain the concept of promises and how they work.

A **Promise** in JavaScript is an object that represents the result of an asynchronous operation. It can be in one of three states: pending, fulfilled (resolved), or rejected. You create a promise using new Promise(), passing an executor function with resolve and reject callbacks. When the operation succeeds, resolve() is called; if it fails, reject() is used. Promises are handled with .then() for success and .catch() for failure. They can be chained to handle sequences of asynchronous tasks in a more readable way.

55. What is the difference between a shallow copy and a deep copy?

A **shallow copy** creates a new object but copies references to the original nested objects, meaning changes to the nested objects affect both the original and the copy. A **deep copy**, on the other hand, creates a new object and recursively copies all nested objects, ensuring that the original and the copy are completely independent. In a shallow copy, nested objects are shared, while in a deep copy, they are fully duplicated.

56. Explain the concept of the event loop and the call stack in JavaScript. How does JavaScript handle asynchronous code execution?

In JavaScript, the **event loop** manages the execution of code, handling both synchronous and asynchronous operations. The **call stack** stores function calls and executes them in a Last In, First Out (LIFO) order. When asynchronous code (e.g., setTimeout, promises) is encountered

from the callback queue to the stack when it's empty, allowing asynchronous code to run without blocking the main thread.

57. What are Web Workers, and how do you use them to run scripts in the background?

Web Workers are JavaScript threads that run in the background, separate from the main thread, allowing long-running scripts to be executed without blocking the user interface. You can create a Web Worker using the `Worker` constructor, passing a JavaScript file as an argument. Once created, the worker can perform tasks asynchronously, and you can communicate with it via postMessage and onmessage events, ensuring the main thread remains responsive.

58. Explain the concept of "debouncing" and "throttling" in JavaScript. How can these techniques optimize performance?

Debouncing and throttlingare techniques used to optimize performance by limiting the frequency of function executions in response to events like scrolling or resizing.

- **Debouncing** ensures that a function is only executed after a certain amount of idle time, i.e., it delays the execution until the event stops triggering for a specified time (e.g., for search input).
- **Throttling** limits the number of times a function can be executed in a given period, ensuring it runs at regular intervals (e.g., during scroll or window resizing).

JavaScript MCQ Coding Interview Questions

59. Which of the following is a JavaScript data type?

Options:

4. All of the above

Answer:

4

Explanation:

- JavaScript has several built-in data types, and number, string, and boolean are all valid types.
- A number represents numeric values, a string represents sequences of characters, and a boolean represents either true or false. All of these are fundamental data types in JavaScript.

60. What is the result of the following code?

```
let a = [1, 2, 3];
let b = a;
b[0] = 100;
console.log(a);
```

Options:

1. [100, 2, 3]
2. [1, 2, 3]
3. [100, 100, 100]
4. undefined

Answer :

1

Explanation :

- The code snippet represents two arrays [100, 2, 3] and [1, 2, 3] being compared using the equality operator (==).
- In JavaScript, arrays are reference types, meaning each array is a reference to an object in memory.

Options:

1. `null`
2. undefined
3. ''
4. []

Answer :

3

Explanation:

- The `+` operator concatenates two empty arrays, which results in an empty string `''` .

62. What will be the output of the following code?

```
(function() {
    var a = b = 5;
})();
console.log(typeof a);
console.log(typeof b);
```

Options:

1. `typeof a: "undefined"`
   `typeof b: "number"`
2. typeof a: "number"
   typeof b: "number"
3. typeof a: "undefined"
   typeof b: "undefined"
4. typeof a: "number"
   typeof b: "undefined"

Answer:

- Inside the IIFE, `b = 5` is treated as a global variable (since no `var`, `let`, or `const` keyword is used).
- However, `a` is declared with `var` and is local to the function, so it is `undefined` outside.

63. What will be logged in the console?

```
console.log(1 < 2 < 3);
console.log(3 > 2 > 1);
```

Options:

1. `true, true`
2. `true, false`
3. `false, true`
4. `false, false`

Answer:

2

Explanation:

- `1 < 2 < 3 is evaluated as (1 < 2) < 3, which becomes true < 3. In JavaScript, true is treated as 1, so 1 < 3 is true.`
- 3 > 2 > 1 becomes (3 > 2) > 1, which results in true > 1. Since true is 1, the comparison becomes 1 > 1, which is false.

64. What will be the output of the following code?

```
const obj1 = { a: 1 };
const obj2 = { a: 1 };
console.log(obj1 == obj2);
console.log(obj1 === obj2);
```

3. `false, true`
4. `false, false`

Answer:

4

Explanation:

- In JavaScript, objects are compared by reference, not by value. Since obj1 and obj2 point to different memory locations, both == and === comparisons return false.

65. What will be the result of the following code?

```
let x = 10;
let y = (x++, x + 1, x * 2);
console.log(y);
```

Options :

1. `22`
2. `12`
3. `21`
4. `20`

Answer:

22

Explanation:

- The comma operator (,) evaluates all expressions but returns the value of the last one.
- x++ increments x to 11, but the result of this expression is the original 10.

66. What will be the output of this asynchronous JavaScript code?

```
console.log('A');
setTimeout(() => console.log('B'), 0);
Promise.resolve().then(() => console.log('C'));
console.log('D');
```

Options:

1. A D B C
2. A B C D
3. A D C B
4. A C D B

Answer:

3

Explanation:

- The synchronous code runs first, logging 'A' and 'D'.
- Promise callbacks (microtasks) are executed before setTimeout (macrotasks). So 'C' is logged before 'B'.

67. What will be the output of this recursive function?

```
function foo(num) {
  if (num === 0) return 1;
  return num + foo(num - 1);
}
console.log(foo(3));
```

Options:

1. 3
2. 6

Answer:

3

Explanation:

- The function works recursively:
  - foo(3) → 3 + foo(2)
  - foo(2) → 2 + foo(1)
  - foo(1) → 1 + foo(0)
  - foo(0) → 1
- So, the total is 3 + 2 + 1 + 1 = 7.

68. What will be printed in the following code?

```
let a = [1, 2, 3];
let b = a;
b.push(4);
console.log(a);
console.log(b);
```

Options:

1. `[1, 2, 3]`
   `[1, 2, 3, 4]`
2. [1, 2, 3, 4]
   [1, 2, 3, 4]
3. [1, 2, 3]
   [1, 2, 3]
4. [1, 2, 3, 4]
   [1, 2, 3]

Answer:

2

- In JavaScript, arrays are reference types. Both a and b point to the same array in memory. Modifying b also affects a.

69. What will be logged by the following code?

```
function test() {
  console.log(this);
}
test.call(null);
```

Options:

1. `null`
2. undefined
3. Window or global object
4. TypeError

Answer:

3

Explanation:

- In non-strict mode, calling a function with this set to null defaults to the global object (Window in browsers or global in Node.js).

In Interviews sometimes interviewer ask some tricky output based JS questions to test your grasp on concepts. Let's see some of tricky questions that may ask in your upcoming interview.

Note: we recommend you should guess first then confirm your output by hit on run.

70.

```
let a = 5;
let b = '5';

if (a == b) {
  console.log('Equal');
} else {
  console.log('Not Equal');
}
```

Explanation: *The `==` operator performs type coercion, converting both operands to the same type before comparison. Here, `'5'` is coerced to a number, making the comparison `5 == 5`, which evaluates to `true`. To avoid such issues, it's recommended to use the strict equality operator `===`, which checks both value and type without coercion.*

71.

```
for (var i = 0; i < 3; i++) {
  setTimeout(function() {
    console.log(i);
  }, 100);
}
```

Explanation: *Due to JavaScript's function scoping with `var`, the variable `i` is shared across all iterations. By the time the `setTimeout` callbacks execute, the loop has completed, and `i` equals `3`. To capture the value of `i` at each iteration, you can*

72.

```
function arrayFromValue(item) {
  return
    [item];
}

console.log(arrayFromValue(10)); // ???
```

Explanation: *JavaScript's automatic semicolon insertion adds a semicolon after the `return` statement, causing the function to return `undefined` instead of the intended array. To fix this, ensure the return statement is on the same line as the array:*

73.

```
const car = {
  name: 'Toyota',
  getName: function() {
    return this.name;
  },
};

const getCarName = car.getName;
console.log(getCarName()); // ???
```

Explanation: *When `getCarName` is called, it's no longer in the context of the `car` object. Therefore, `this` refers to the global object (or `undefined` in strict mode), not the `car` object. To maintain the correct context, you can use `.bind(car)`:*

const getCarName = car.getName.bind(car);
console.log(getCarName()); // 'Toyota'

74.

```
console.log(a); // ???
```

```
let b = 2;
```

Explanation: *Variables declared with `var` are hoisted and initialized with `undefined`, so `console.log(a)` outputs `undefined`. Variables declared with `let` are hoisted but not initialized, leading to a `ReferenceError` when accessed before their declaration.*

75.

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

Explanation: *This is a well-known quirk in JavaScript. Despite `null` being a primitive value representing the intentional absence of any object value, the `typeof` operator returns `"object"`. This behavior is considered a bug in JavaScript that has been maintained for backward compatibility.*

76.

```
let arr = new Array(3).fill([]);
arr[0].push(10);
console.log(arr); // ???
```

Explanation: *The `fill` method fills all elements of the array with the same reference to the same array. Therefore, when you modify one element (e.g., `arr[0].push(10)`), all elements reflect this change because they all point to the same array in memory.*

77.

```
const obj = { x: 1 };
const { x, x: y } = obj;
console.log(y); // ???
```

Explanation: *In this destructuring assignment, `x` is first assigned the value `1` from the object. Then `x: y` creates a new*

78.

```
console.log(typeof undeclaredVar); // ???
console.log(typeof y); // ???
let y = 1;
```

Explanation: *The first `console.log` outputs `"undefined"` because `undeclaredVar` is not declared at all. The second `console.log` throws a `ReferenceError` because `y` is declared with `let` and is in a "temporal dead zone" from the start of the block until the declaration is encountered. During this period, accessing `y` results in a `ReferenceError`.*

Common JavaScript Interview Questions

1. What are the different data types in JavaScript?

JavaScript has two types of data: **primitive** and **non-primitive**.

- **Primitive data types**: `string`, `number`, `boolean`, `undefined`, `null`, `symbol`, `bigint`
- **Non-Primitive data types**: `objects`, arrays and functions.

2. What is the difference between `==` and `===` in JavaScript?

- **`==` (loose equality):** They compares only the values, allowing type coercion (i.e., converts types if necessary).
- **`===` (strict equality):** They compares both value and type, without type conversion.

3. What is the difference between `let`, `const`, and `var`?

- `var`: Function-scoped, can be re-assigned and redeclared within its scope.

- `const`: Block-scoped, cannot be reassigned or redeclared, and the value assigned to it remains constant.

4. Explain hoisting in JavaScript.

Hoisting is JavaScript's default behavior of moving all variable and function declarations to the top of their containing scope during the compile phase. However, only the declarations are hoisted, not the initialization.

5. What is the difference between `null` and `undefined` in JavaScript?

- `null`: Represents an intentional absence of any object value. It is explicitly assigned to indicate "no value."
- **`undefined`**: Indicates that a variable has been declared but has not yet been assigned a value.

```
let a = null;    // Explicitly assigned nulllet b;          //
Variable declared but not assigned, hence undefined
```

6. What are promises in JavaScript and how do they work?

A promise is an object representing the eventual completion or failure of an asynchronous operation. Promises are used to handle asynchronous operations like API calls, ensuring cleaner code compared to callbacks. It has three states:

- **`pending`:** The initial state.
- **`fulfilled`:** The operation completed successfully.
- **`rejected`:** The operation failed.

```
const myPromise = new Promise((resolve, reject) => {
```

```
    }
    else {
        reject("Operation failed");
});});
myPromise.then(result => console.log(result)).catch(error =>
console.log(error));
```

7. What is the event loop in JavaScript?

The **event loop** is a mechanism that allows JavaScript to handle asynchronous operations (like I/O, timers, etc.) without blocking the main thread. It continuously checks the call stack for any code to execute and moves tasks from the callback queue to the call stack when the stack is empty.

8. What are closures in JavaScript?

A **closure** is a function that retains access to its lexical scope (the scope in which it was created) even after that scope has finished execution. Closures allow functions to access variables from an outer function after the outer function has returned.

```
function outer() {
    let x = 10;
    return function inner() {
        console.log(x);
    }
}
const closureFunc = outer();
closureFunc(); // prints 10
```

Output

```
10
```

In JavaScript, the `this`keyword refers to the context in which a function is called. It is used to refer to the object that is executing the current piece of code.

The value of `this` can change depending on how the function is called. Here are the different scenarios where `this` behaves differently:

- **Global context**: In non-strict mode, `this` refers to the global object (`window` in browsers).
- **Object method**: `this` refers to the object the method belongs to.
- **Constructor function**: `this` refers to the instance of the object being created.
- **Arrow functions**: In arrow functions, `this` is lexically bound to the surrounding context.

10. What is a callback function in JavaScript?

A **callback function** in JavaScript is a function that is passed as an argument to another function and is executed after the completion of that function. Callback functions are primarily used for handling asynchronous operations, such as API requests or timeouts, ensuring that certain code runs only after a specific task is completed.

```
function greet(name, callback) {
    console.log("Hello " + name);
    callback();
}

function sayGoodbye() {
    console.log("Goodbye!");
}

greet("Anjali", sayGoodbye);
```

Output

Introduction to Git

A **version Control system** is a system that maintains different versions of your project when working in a team or as an individual. (System managing changes to files) As the project progresses, new features get added to it. So, a version control system maintains all the different versions of your project for you, and you can roll back to any version you want without causing any trouble to you for maintaining different versions by giving names to them, like MyProject, MyProjectWithFeature1, etc.

Distributed Version control system means every collaborator (any developer working on a team project)has a local repository of the project in his/her local machine unlike central where team members should have an internet connection to every time update their work to the main central repository.

So, by distributed we mean: the project is distributed. A repository is an area that keeps all your project files, images, etc. In terms of GitHub: different versions of projects correspond to commits.

> *For more details on the introduction to Github, you can refer: Introduction to Github*

Key Git Concepts

- **Repository**- A directory where git monitors your project files and records their revision history.
- **Clone**- It created a local copy of a remote repository on your machine.
- **Stage** - It Selects specific changes that you want Git to include in the next snapshot.
- **Commit-** It records the staged changes as a permanent version in the project history.
- **Branch**- Lets you develop new features or experiment without affecting the main project.
- **Merge**- Integrates changes from one branch into another.

- **Pull**- It fetches and applies updates from a remote repository to your local one.
- **Push-** It Uploads your local commits to a remote repository

Git Repository Structure

It consists of 4 parts:

1. **Working directory:** This is your local directory where you make the project (write code) and make changes to it.
2. **Staging Area (or index):** this is an area where you first need to put your project before committing. This is used for code review by other team members.
3. **Local Repository:** this is your local repository where you commit changes to the project before pushing them to the central repository on Github. This is what is provided by the distributed version control system. This corresponds to the .git folder in our directory.
4. **Central Repository:** This is the main project on the central server, a copy of which is with every team member as a local repository.

All the repository structure is internal to Git and is transparent to the developer.

Some commands which relate to repository structure:

```
// transfers your project from working directory
// to staging area.
git add .

// transfers your project from staging area to
// Local Repository.
git commit -m "your message here"

// transfers project from local to central repository.
// (requires internet)
git push
```

Note: For installation purposes on ubuntu, you can refer to this article: How to Install, Configure and Use GIT on Ubuntu?

Working with existing repos(git clone) and new repos(git init)

Git allows you to manage existing repository as well as new one.You can clone remote repository to work on a project that already exists, or initialize a new repository to begin tracking changes in a fresh project.

- Working with existing repos (git clone)- When you are collaborating on a project that already exists on the remote platform like Github, you don't need to start from scratch . Instead you can clone the repository to your local machine using:

```
git clone <repository-url>
```

This commands downloads the entire repository including its history, branches, and files.

```
git clone https://github.com/username/project
```

Now you have a full copy of the project and can start contributed locally.

- Started a new Repository (git init) - If you are beginning a brand new project you use:

```
git init
```

This Command will create a new git repository in your current directory. It sets up a . git folder that will track all your changes made to files inside the project.

```
mkdir my-project
cd my-project
git init
```

it will create a new folder , cd will redirect to current directory and git init will initializes a new empty repository.

How to Use `git merge` Effectively in Git

git merge is used to combine changes from one branch into another. It allows teams to work on different features or fixes in isolation and then bring work together . Merging Preserves the history of both branches and is a key part of collaborative workflows in git.

- Switch the branch you want to merge into your changes

```
git checkout main
```

- Merge Changes from another branch into the current branch

```
git merge feature-branch
```

- If there are merge conflicts resolve them manually, then stage the changes

```
git add .
```

- Commit the merge if conflicts were resolved manually

```
git commit
```

Creation of new Branch

Branches allows you to work on new features or fixes independently without affecting the main codebase. This isolates works and makes collaboration easier.

```
git branch branch-name
```

Creating a new Branch without switching to it.

- View All Branches

```
git branch
```

Deletion of Branches

Once a branch has been emerged or no longer needed, you can delete it to keep your repo clean and organized.

- Delete a local branch

```
git branch -d branch-name
```

Deletes the branch if it has already been merged.

- Force delete a local branch(unmerged changes)

```
git branch -D branch-name
```

Delete the branch regardless of its mege status

- Delete a remote branch

```
git push origin --delete branch-name
```

Removes the branch from the remote repository (e.g., GitHub).

How to Use `git fetch` Effectively in Git

git fetch is used to download latest changes from a remote repository without automatically merging them into your current branch. It updates your local view of the remote branches, so you can review or merge changes .

- Fetch all branch from the remote

```
git fetch
```

Download update from the default remote (usually origin)

- Fetch from a specific remote

```
git fetch origin
```

- Fetch from a specific branch

```
git fetch origin feature-branch
```

Fetches only the `feature-branch` from the remote.

- View changes after fetching

```
git log HEAD..origin/main
```

It shows changes to your main branch that aren't not in your local branch.

How to Use `git stash` Effectively in Git

git stash is a handy git command that lets you temporarily save your uncommitted changes(both staged and unstaged) so you can switch branches or perform other tasks without losing your work. Once you're ready, you can **reapply** those changes exactly as you left them.

- Stash your current changes

```
git stash
```

Saves your changes and reverts your working directory to a clean state.

- Stash with a custom message(recommended)

```
git stash save "WIP: added login form"
```

Adds a label so you remember what was stashed.

- View list all stashes

```
git stash list
```

Displays stashes like `stash@{0}`, `stash@{1}`, etc.

- Apply the most recent stash

```
git stash apply
```

Restores the changes but keeps the stash in the list.

- Apply a specific stash

```
git stash apply stash@{1}
```

- Apply and delete a stash in one step

```
git stash pop
```

- Clear all stash

```
git stash clear
```

It removes all saved stashes.

Github

Github basically is a for-profit company owned by Microsoft, which hosts Git repositories online. It helps users share their git repository online, with other users, or access it remotely. You can also host a public repository for free on Github.

User share their repository online for various reasons including but not limited to project deployment, project sharing, open source contribution, helping out the community and many such.

Accessing Github central repository via HTTPS or SSH

Here, transfer project means transfer changes as git is very lightweight and works on changes in a project. It internally does the transfer by using Lossless Compression Techniques and transferring compressed files. Https is the default way to access Github central repository.

- **By git remote add origin http_url:** remote means the remote central repository. Origin corresponds to your central repository which you need to define (hereby giving HTTPS URL) in order to push changes to Github.
- **Via SSH:** connect to Linux or other servers remotely.

If you access Github by ssh you don't need to type your username and password every time you push changes to GitHub.

Terminal commands:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
This does the ssh key generation using RSA cryptographic
algorithm.

eval "$(ssh-agent -s)" -> enable information about local login
session.

ssh-add ~/.ssh/id_rsa -> add to ssh key.
cat ~/.ssh/id_rsa (use .pub file if not able to connect)
add this ssh key to github.

Now, go to github settings -> new ssh key -> create key

ssh -T git@github.com -> activate ssh key (test connection)

Refresh your github Page.
```

Working with git - Important Git commands

Git user configuration (First Step)

```
git --version (to check git version)
git config --global user.name "your name here"
git config --global user.email "your email here"
```

These are the information attached to commits.

Initialize directory

```
git init
```

initializes your directory to work with git and makes a local repository. .git folder is made (OR)

```
git clone http_url
```

This is done if we have an existing git repository and we want to copy its content to a new place.

Connecting to the remoterepository

```
git remote add origin http_url/ssh_url
```

connect to the central repo to push/pull. pull means adopting the changes on the remote repository to your local repository. push merges the changes from your local repository to the remote repository.

```
git pull origin master
```

One should always first pull contents from the central repo before pushing so that you are updated with other team members' work. It helps prevent merge conflicts. Here, master means the master branch (in Git).

Stash Area in git

```
git stash
```

Whichever files are present in the staging area, it will move that files to stash before committing it.

```
git stash pop
```

Whenever we want files for commit from stash we should use this command.

```
git stash clear
```

By doing this, all files from stash area is been deleted.

Steps to add a file to a remote Repository:

First, your file is in your working directory, Move it to the staging area by typing:

```
git add -A (for all files and folders)
#To add all files only in the current directory
git add .
```

git status: here, untracked files mean files that you haven't added to the staging area. Changes are not staged for commit means you have staged the file earlier than you have made changes in that files in your working directory and the changes need to be staged once more. Changes ready to be committed: these are files that have been committed and are ready to be pushed to the central repository.

```
git status


git commit -a -m "message for commit"
-a: commit all files and for files that have been
     staged earlier need not to be git add once more
-a option does that automatically.


git push origin master -> pushes your files to
                          github master branch
git push origin anyOtherBranch -> pushes any
                       other branch to github.
git log ; to see all your commits


git checkout commitObject(first 8 bits) file.txt->
revert back to this previous commit for file file.txt
```

Previous commits might be seen through the git log command.

```
HEAD -> pointer to our latest commit.
```

Ignoring files while committing

In many cases, the project creates a lot of logs and other irrelevant files which are to be ignored. So to ignore those files, we have to put their names in**".gitignore"** file.

```
touch .gitignore
echo "filename.ext" >>.gitignore
```

```
#to ignore all files with .log extension
echo "*.log" > .gitignore
```

Now the filenames written in the .gitignore file would be ignored while pushing a new commit. To get the changes between commits, commit, and working tree.

```
git diff
```

The 'git diff' command compares the staging area with the working directory and tells us the changes made. It compares the earlier information as well as the current modified information.

Branching in Git

```
create branch ->
git branch myBranch
or
git checkout -b myBranch -> make and switch to the
                                    branch myBranch
```

Do the work in your branch. Then,

```
git checkout master ; to switch back to master branch
```

Now, merge contents with your myBranch By:

```
git merge myBranch (writing in master branch)
```

This merger makes a new commit.

Another way

```
git rebase myBranch
```

This merges the branch with the master in a serial fashion. Now,

```
git push origin master
```

To remove or delete a file

To remove. a file from the Git repository we use

```
git rm “file name”
```

To remove only from the staging area

```
git rm –cached “ file name”
```

Undoing change

To change all the files to as same as the previous commit then use

```
git checkout -f
```

Git vs Github

Git and GitHub are often mentioned together, but they serve different purposes. **Git** is a local tool used for tracking changes in your code, while **GitHub** is an online platform for hosting and sharing Git repositories. Together, they enable powerful version control and team collaboration in software development.

Feature	Git	Github
Type	Version control system (VCS)	Web based Hosting Service
Function	Tracks and manages code changes locally	Stores Git repositories in the cloud and enables collaboration
Installation	Must be installed on your system	Accessible through a web browser (no installation needed)

Feature	Git	Github
Usage	Used for local version control and branching	Used for sharing, reviewing, and managing Git projects online
Collaboration	Limited to local or manual sharing	Real-time collaboration with teams and contributors
Interface	Command-line tool (CLI)	Web-based UI and also supports Git CLI
Main Purpose	Track code history and manage changes locally	Centralized hub to host, view, and contribute to Git projects
Offline Support	Fully functional offline	Requires internet to access remote features

Contributing to Open Source

Open Source might be considered as a way where user across the globe may share their opinions, customizations or work together to solve an issue or to complete the desired project together. Many companies host there repositories online on Github to allow access to

necessarily all) rewards their contributors in different ways.

You can contribute to any open source project on Github by forking it, making desired changes to the forked repository, and then opening a pull request. The project owner will review your project and will ask to improve it or will merge it.

Conclusion

Apart from that, we use different CSS frameworks (like Bootstrap, Tailwind CSS) and JavaScript frameworks (like React, Angular, Vue.js) to enhance development efficiency and improve user experience.

Frontend Developer Interview Questions - Beginners

1. What is HTML?

HTML stands for Hyper Text Markup Language. It is a markup language that consists of different tags. It is used to define the structure of the web pages. HTML is the basic building block of the web page, which is used to display text, images, and other content.

2. What are Semantic elements in HTML?

The semantic elements in HTML are the elements that contain the meaning of the content and the structure of the HTML document. These elements contain content that is related to their names or reflects their names. These are some of the semantic HTML elements listed below:

- Header
- Main
- Section
- Article
- Aside
- Footer etc.

3. Differentiate between the Inline and the Block elements in HTML.

Inline Element	Block Element
Does not start on a new line	Always starts on a new line

Inline Element	Block Element
Takes up only as much width as necessary	Takes up the full width available
Cannot contain block elements	Can contain both inline and block elements
Height and width are usually not adjustable	Height and width can be set freely
Example: , <a>, , etc	Example: , <p>, <h1> etc

4. What is a list in HTML? Explain the different types of lists available in HTML.

In HTML, lists are used to represent a collection of different items. There are three types of lists available in HTML, as listed below:

1. **Unordered List:** It is defined using the <ul> and <li> tags. By default, it represents the items with a bulleted dot.

```
<ul>
    <li>List Item 1</li>
    <li>List Item 3</li>
    <li>List Item 3</li>
</ul>
```

2. Ordered List: It is defined using the <ol> and <li> tags. By default, it represents the list of items with numeric digits.

```
<ol>
    <li>List Item 1</li>
    <li>List Item 3</li>
    <li>List Item 3</li>
 </ol>
```

3. Definition List: It is a special kind of list that is used to list the terms with their definitions in a structured way. It can be defined using the <dl>(definition list), <dt>(definition term), and <dd>(definition description) tags.

```
<dl>
    <dt>First  term</dt>
    <dd>Definition 1</dd>

    <dt>Second term</dt>
    <dd>Definition 2</dd>

    <dt>Third term</dt>
    <dd>Definition 3</dd>
</dl>
```

5. What is the difference between and ?

The table below will show the differences between the div and span tags in HTML:

tag	tag
It is a block-level element.	It is an inline element.
It can be used to group and structure the content of the web page.	It is mainly used to interact with and style a particular part of the web page.
It represents a bigger section of the web page.	It is used to target small parts of the web page.
It starts from a new line and takes up the full width available.	It does not start from a new line and takes up only the required width as taken by the content.

6. What is the DOCTYPE declaration?

In HTML, the DOCTYPE declaration is also known as the document type declaration. It defines the document type and tells the browser in which version of HTML the document is written.

7. What is the purpose of the <iframe> tag?

The <iframe> tag is used to embed external documents or web pages inside the current document by specifying their link inside it. It is mainly used to embed external videos, maps, and other external content.

8. What is the difference between <b> and tags in HTML?

The <b> and tags are both used for making the text bold. The <b> is used when we want to highlight only the text. The is used when we want to indicate the importance of the text.

9. What are meta tags in HTML?

In HTML, meta tags are used to define the metadata about the HTML document, including character set, description, keywords, author, and viewport settings. Placed within the <head> element.

10. Explain tags in HTML.

The HTML tags are used to define the elements on the web page. They are the keywords that are enclosed inside the angle brackets(<>). The examples of HTML tags are , <p>, <a>, , <img> etc.

11. What is CSS?

CSS stands for Cascading Style Sheets. It helps us to design and style the web page to make it attractive for users. It is used to describe how

the elements should be displayed on the screen. CSS provides us with a lot of selectors to select the HTML elements and style them according to the requirements. Some of the CSS selectors are Element Selectors, Class Selectors, and ID Selectors.

12. Explain selectors in CSS.

In CSS, selectors are used to select elements and style them element by providing CSS properties to them. Below is the list of some common CSS selectors:

- **Element Selector:** Select directly by using the name of the element.
- **ID Selector:** Define the ID attribute and select using the # prefix followed by the value of the ID attribute.
- **Class Selector:** Define the Class attribute and select using the. A prefix followed by the value of the class attribute.
- **Universal Selector (*):** Select using the * sign.
- **Attribute Selector:** Select the elements based on the attribute values. Eg: input[type="text"]{}
- **Direct Child Selector:** Select an element using any of the above selectors and use > followed by a direct child selector. Eg: parent > child{}
- **Pseudo Selectors:** These are selectors like :hover, :nth-child(), ::after, ::before etc.

13. What is the difference between visibility: hidden and display: none properties in CSS?

The visibility: hidden property only hides the content of the element on which it is used. It does not remove the element from the document and keeps the space as it is, so that no other element can replace it on the UI.

The display: none property not only hides the element but removes it from the document, and the space acquired by the element is now free to be acquired by the other elements.

14. What is the difference between CSS Grid and Flexbox?

CSS Grid	Flexbox
Two-dimensional layout	One-dimensional layout
Controls both rows and columns	Controls either row or column, not both
Suitable for complex, structured layouts	Ideal for simple, linear layouts
Allows item placement anywhere in the grid	Items follow the document/source order
Can define both rows and columns together	Defines layout in a single direction

15. What is the use of the float property?

The float property allows you to set the child elements of a container either on the left side of it or on the right side of it. The possible values for this property are left, right, initial, inherit, and none.

16. What is JavaScript?

JavaScript is a high-level scripting language. It is a dynamically typed language. JavaScript is used to add dynamic elements and styles to the HTML document to make the web page more interactive and attractive. JavaScript is used in both the frontend as well as backend.

17. What is the difference between let, var, and const?

var	let	const
Function-scoped	Block-scoped	Block-scoped
Hoisted but initialized as undefined	Hoisted but not initialized	Hoisted but not initialized
No TDZ (accessible before declaration)	Has TDZ (not accessible before declaration)	Has TDZ (not accessible before declaration)
Reassigning and redeclaring within the same scope is allowed.	Reassigning is allowed, but redeclaration in the same scope is not allowed.	Neither reassigning nor redeclaring is allowed; it is a constant and cannot be reassigned or redeclared.

18. What is difference between == and === in JavaScript?

The == operator is known as the double-equal operator in JavaScript. The == operator checks only for the values of the operands and returns true if the values are the same.

The === operator is known as the triple-equal the operator. It not only checks for the values of the operands but also the types of the operands. It returns true only if the values and the type of the operands are the same.

```
let num = 5;
let str = '5';

console.log(num == str);
console.log(num === str);
```

Output

```
true
false
```

19. What is the DOM?

DOM stands for the Document Object Model. In JavaScript, the DOM is used to represent the structure of the web page. It allows us to manipulate the elements, attributes, and content of a web page using JavaScript. It is a tree-like representation of a web page's HTML structure.

20. Difference between null and undefined in JS.

Undefined Value:

- The undefined is the default value that is assigned to a variable that is declared but not initialized.
- It is also the default return value of a function.
- When you try to access some value or property that is not the part of an object, it returns undefined.

Null Value:

- It can be assigned to a variable to make it an empty variable.
- It represents that the accessing variable is not present in the code.

```
let a;      // variable is declared but not assigned
let b = null; // variable is explicitly assigned a null value

console.log(a);
console.log(b);

console.log(a === b);  // false (undefined is not equal to
null)
```

Output

```
undefined
null
false
```

21. What is React?

React is an open-source front-end JavaScript library created by Facebook that is used for building user interfaces(UI) based on components. There are various features that are provided by React.

Key Features of React

- Component-Based Architecture
- Virtual DOM (DOM)
- JSX (JavaScript XML)
- One-Way Data Binding
- Single Page Application (SPA)
- State Management

22. Explain the building blocks of React.

There are five building blocks of React

- **Components:** These are reusable blocks of code that return HTML.
- **JSX:** It stands for JavaScript and XML and allows you to write HTML in React.
- **Props and State:** props are like function parameters, and State is similar to variables.
- **Context:** This allows data to be passed through components as props in a hierarchy.
- **Virtual DOM:** It is a lightweight copy of the actual DOM, which makes DOM manipulation easier.

23. What is virtual DOM in React?

DOM refers to the Document Object Model. Virtual DOM is a virtually created DOM. It is exactly like DOM and it has all the properties that

DOM has. But the main difference is Whenever a code runs JavaScript Framework updates the whole DOM at once which gives a slow performance. Whereas virtual DOM updates only the modified part of the DOM.

24. What is JSX React?

JSX stands for the JavaScript XML. It is the syntax extension for JavaScript. JSX allows us to write the HTML code inside the JavaScript, then React compiles this code into pure JavaScript that can be rendered by the browser.

```
import React from 'react';
function App() {
    return <h1>Hello, React!</h1>;
}
export default App;
```

In this example, the JSX code inside the App function returns an <h1> element with the text "Hello, React!" which will be displayed in the browser when the component is rendered.

25. What are the components In React?

A Component is one of the core building blocks of React. In other words, we can say that every application we develop in React will be made up of pieces called components.

There are the **two types of the components in the React**

- Functional Component: They are the simple JavaScript functions that accept the props as the input and return the JSX as output. Functional components can manage the state and the lifecycle with the help of the React Hooks.
- Class Component: The class components are a little more complex than the functional components. The functional components are not aware of the other components in your program whereas the class

components can work with each other. We can pass data from one class component to another class component.

26. What is Angular?

Angular is an open-source framework that is used for building web applications using TypeScript. Angular is used for building single-page applications (SPAs). It was developed by Google.

Key Features of Angular

- **Two-way data binding:** Synchronizes data between the model and the view automatically.
- **Dependency injection:** Manages and injects dependencies efficiently to enhance modularity.
- **Modularization:** Break down the application into smaller, reusable modules.
- **Templating:** Uses templates to define the view, providing dynamic and efficient UI updates.
- **RESTful API handling:** Simplifies interaction with RESTful services and APIs.

27. What is AngularJS?

AngularJS, developed by Google, has been important in web development since its inception in 2009. AngularJS excels at building SPAs. AngularJS, developed by Google, has been important in web development since its inception in 2009. AngularJS excels at building SPAs.

Unlike Angular, which is TypeScript-based, AngularJS relies on JavaScript and has limited mobile optimization. As AngularJS is now deprecated, developers are encouraged to migrate to Angular for better performance and maintainability.

28. What are the Components in Angular?

Components are the building blocks of Angular. It consists of a TypeScript class, an HTML template, and optional CSS styles.

Example: Creating a reusable header component for your application.

```
import { Component, Input } from '@angular/core';

@Component({
    selector: 'app-header',
    templateUrl: './header.component.html',
    styleUrls: ['./header.component.css']
})
export class HeaderComponent {
    @Input() title: string;
    @Input() links: { name: string, url: string }[];

    constructor() { }
}
```

29. What is Two-Way Data Binding in Angular?

Angular supports two-way data binding, which allows the automatic synchronization of data between the view and the component. If any changes are made in the component will be reflected in the view, and if any changes are made in the view, they will be reflected in the component. This is known as the two-way data binding.

30. What is Lazy Loading in Angular?

In Angular, Lazy Loading is a technique that is used to load the modules on demand instead of loading all the modules at the start. Due to this, the performance of the application is improved, and the initial load time of the application is also reduced.

31. What are Angular Pipes?

Angular Pipes are used to transform the displayed data in the template. They apply the transformation on the input values and return the transformed value. Due to this, we can directly manipulate the data in the HTML templates.

Syntax

```
{{ value | pipeName }}
```

32. What is Vue.js?

Vue.js is the framework of JavaScript, that is used for building single-page applications(SPAs) and user interfaces (UI) for websites. It stands out for its simplicity, seamless integration with other libraries, and reactive data binding.

33. What are VueJS components?

A component in VueJS is a reusable piece of code that signifies a part of the user interface. Components make it simpler to handle complex applications by breaking them down into minor, controllable pieces.

34. What are Props in VueJS?

Props in VueJS are custom attributes that permit parent components to pass data to child components. They empower component reusability by making it possible to handle child components from their parent component. They are specified in the child component and received as arguments from the parent.

Syntax

```
Vue.component('child-component', {
    props: ['propName'],
        // define component
});
```

35. What is a VueJS Router?

Routing in VueJs is used to navigate the Vue application, permitting the formation of single-page applications with multiple views. Vue Router helps link between the browser's URL/History and Vue's components empowering certain paths to render whatever view is linked with it. A Vue router is utilized in making single-page applications.

36. What is Git?

Git is an open-source version control system that helps developers to store, track, and manage code. Git was created by Linus Torvalds in 2005 for Linux kernel development. The latest version of git is 2.48.1.

37. What are the advantages of using Git?

The advantages of using Git is mentioned below

- **Version Control:** Git helps in tracking and managing the changes in the code.
- **Secure:** It uses cryptographic hashing to secure the codebase.
- **Easy to use:** It is easy to use because we need to manage the less commands.
- **Collaboration:** Many developers can work together on the same project.

38. Explain the difference between Git and GitHub.

Git	GitHub
Git is a distributed version control system (VCS) that helps track changes in source code.	GitHub is a cloud-based platform for hosting Git repositories and facilitating collaboration.

Git	GitHub
Used to manage source code locally and track changes.	Provides a remote hosting service for Git repositories with collaboration features.
Does not provide cloud hosting for repositories.	Offers cloud hosting for repositories with features like private/public repositories.
Helps developers track, commit, and merge code changes.	Provides features like issue tracking, pull requests, and code review.
Other Git hosting services: GitLab, Bitbucket, etc.	GitHub is a Git hosting service with additional features for teamwork.

39. What is a repository in Git?

In Git, the repository is the location in which all the files and the versions of projects are stored, which allows the developers to track the changes made in the code. There are two types of repository.

- **Local Repository:** Local Repositories exist on a developer's local machine.
- **Remote Repository:** Remote Repositories are stored on the server (GitHub, GitLab).

Frontend Developer Interview Questions - Intermediate

40. What is localStorage?

localStorage is a client-side web storage mechanism that allows web applications to store key-value pairs persistently in a user's web browser. It provides a simple interface for storing data locally.

41. What is sessionStorage?

sessionStorage is a web storage API provided by web browsers to store data in a similar way as stored in the localStorage in the form of key-value pairs. The data stored in the sessionStorage will only be accessible for one session, such that if the user closes the window or tab, the stored data will be lost.

42. How to create a table In HTML?

In HTML, tables are created using the <table> element. Each table consists of rows <tr> and cells <td> for data, and <th> for headers.

```
<table border="1">
    <tr>
        <th>Name</th>
        <th>Age</th>
        <th>Country</th>
    </tr>
    <tr>
        <td>GFG</td>
        <td>12</td>
        <td>India</td>
    </tr>
    <tr>
        <td>Riya</td>
        <td>24</td>
        <td>India</td>
    </tr>
</table>
```

43. Difference between the GET and the POST methods in HTML forms.

GET Method	POST Method
It is an insecure way to send data to the server.	It is a secure way of sending the form data.
All the form data parameters are visible in the URL.	None of the parameters are visible anywhere.
It has a URL length limit that varies for different browsers.	It has a bigger URL length limit as compared to the limit of the GET method.
Results are cached by the browser by default.	Does not cache the responses in the browser by default.
Users can bookmark the form submission.	Responses can be bookmarked easily.

44. What are the attributes in HTML?

The HTML attributes are used to define the behavior of the HTML element. The attributes are used along with the HTML tags.

Example: div class="container", <p title="This is a GFG">. Here, class and title are the attributes that are used along with the HTML tags to define the HTML element.

45. What are void elements in HTML?

In HTML, void elements are also known as empty elements. Void elements are the elements that do not have a closing tag, they only have the opening tag. Example: <img>, <meta>, <link>,
.

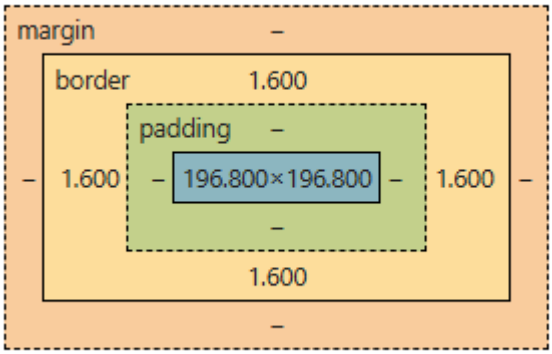
46. What is z-index in CSS?

The z-index property is used to control the stacking order of the elements that are positioned using the position property in CSS.

- The default value of the z-index is auto(0).
- z-index works only with the positioned elements (relative, absolute, fixed, or sticky).
- The elements having a positive z-index value will appear in front and the element having a negative z-index value will appear behind.

47. What is the Box Model in CSS?

The CSS Box Model defines how elements are sized, positioned, and rendered on a webpage. When a browser loads an HTML document, it creates a DOM tree and assigns a box to each element. This box calculates the element's dimensions and position relative to its parent or the root <html> element, ensuring accurate layout and spacing.



48. What is the flexbox?

In CSS, Flexbox is the one-dimensional layouts, that align the items either in a single row or column and distribute the space between them due to the elements can flex or grow and shrink to fit the available space within a container.

49. How can we include the CSS in the webpage?

We can include the CSS in the webpage in the following ways:

1. Inline CSS

Inline CSS is added directly to the HTML element using the style attribute.

```
<p style="color: blue; font-size: 20px;">This is a paragraph
with inline CSS.</p>
```

2. External CSS

External CSS is written in a separate .css file and linked to the HTML document using the <link> tag.

```
p {
    color: red;
    font-size: 16px;
}
```

3. Internal CSS

Internal CSS is written inside a <style> tag within the <head> section of the HTML file.

```
<style>
        p {
            color: green;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <p>This paragraph is styled using internal CSS.</p>
</body>
```

4. CSS @import Method

We can import an external CSS file inside another CSS file using @import.

HTML CSS

```
<link rel="stylesheet" href="main.css">
<body>
```

```
    <h1>This heading is styled using imported CSS.</h1>
</body>
```

50. What is the CSS preprocessor?

CSS Preprocessor is a scripting language that generates the Cascading Style Sheets (CSS) from its own syntax.

- The most used CSS preprocessors are SASS and LESS.
- SASS is based on the Ruby language, while LESS is based on JavaScript, Initially, it was also based on Ruby but now shifted to JavaScript. They provide some extra features like creating variables, mixins, code nesting, and some other features to enhance code maintainability.

51. What is hoisting in JavaScript?

In JavaScript, Hoisting is the behavior in which during the compilation phase, the variables and the functions declarations are moved to the top of their respective scopes.

- Hoisting is applied on the var.
- But with let and const it is technically hoisted which goes under the TDZ (Temporal Dead Zone) and shows the Reference Error, where we cannot access them before their declaration point.

Example with the var

```
a = 10
var a
console.log(a)
```

Output

```
10
```

Example with let

```
a = 10
```

```
let a
console.log(a);
```

```
It will show the Reference Error
```

52. Difference between Implicit and Explicit Conversion in JavaScript?

Implicit Conversion (Coercion)	Explicit Conversion
Automatically converts data types during operations.	Manually converts data types using built-in functions.
Controlled by JavaScript.	Controlled by the developer.
Converts string → number, boolean → number, etc., based on context.	Uses methods like Number(), String(), Boolean(), parseInt(), etc.
console.log("5" + 3); // "53" (number converted to string) console.log("5" - 3); // 2 (string converted to number)	console.log(Number("5") + 3); // 8 console.log(String(5) + " is a number"); // "5 is a number"

53. What is Implicit Type Coercion in JavaScript?

In JavaScript, the implicit type conversion or coercion conversion can be defined as the automatic conversion of the data type of the variables from one type to another type. Implicit type conversion mostly occurs when we are performing the arithmetic or the logical operations.

- String + Number (Concatenation)
- Boolean to Number

- Equality Comparison (==)

54. What are the closures?

A closure is a function that has access to its scope, the outer function's variables, and global variables, even after the outer function has finished executing. This enables functions to "remember" their environment.

```
function outer() {
    let outerVar = "I'm in the outer scope!";
    function inner() {
        console.log(outerVar);
    }
    return inner;
}
const closure = outer();
closure();
```

Output

```
I'm in the outer scope!
```

In this example

- outer() defines a local variable outerVar.
- Inside outer(), we define innerFunction, which logs the value of outerVar.
- outer() returns inner(), and closure() stores the returned value (which is inner()).
- Even though outer() has finished execution, inner() can still access outerVar because it "remembers" the environment where it was created. This is the closure at work!

55. What is the use of 'this' keyword in JS?

The this eyword in JavaScript is an identifier that based on the scope of the function or variable from which it is invoked. The scoping behavior of this keyword changes based on the scope where it is used.

```
const person = {
    name: "GFG",
    greet: function() {
        console.log("Hello, " + this.name);
    }
};

person.greet();
```

Output

```
Hello, GFG
```

In this example

- This keyword refers to the object that calls the function.
- Here, this.name refers to a person.name, which is "GFG".

56. How do browsers read JSX in React?

Browsers are not capable of reading JSX they can only read pure JavaScript. The web browsers read JSX with the help of a transpiler. Transpilers are used to convert JSX into JavaScript. The transpiler used is called Babel.

57. What is a react router?

React Router is a standard library for routing in React. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.

To install react router type the following command.

```
npm i react-router-dom
```

58. What are Hooks in React?

React hooks was introduced in React 16.8, with the help of the React hooks we can use the state and lifecycle features in the functional components without using the class components.

Some of the commonly used React Hooks are

- **useState:** Enables functional components to manage their own state.
- **useEffect:** Allows performing side effects in functional components, similar to lifecycle methods in class components.
- **useContext:** Provides a way to access context values within functional components.
- **useRef:** Creates a mutable reference that persists across renders.
- **useMemo:** Memoizes the result of a function, preventing unnecessary recalculations.
- **useCallback:** Memoizes a function itself, useful for optimizing performance when passing callbacks to child components.

59. What are Custom Hooks in React?

Custom Hooks are user-defined functions that encapsulate reusable logic. They enhance code reusability and readability by sharing behavior between components.

60. What are the lifecycle methods in the React?

Lifecycle methods in the React are the special methods in class component that allow us to run the code at different stages of a component lifecycle.

There are the three main phases of the Component Lifecycle

- **Mounting:** When the component is created and inserted into the DOM.
- **Updating:** When the component's state or props change.
- **Unmounting:** When the component is removed from the DOM.

61. What are the main features of Angular?

- **Two-way data binding**: Synchronizes data between the model and the view automatically.
- **Dependency injection**: Manages and injects dependencies efficiently to enhance modularity.
- **Modularization**: Break down the application into smaller, reusable modules.
- **Templating**: Uses templates to define the view, providing dynamic and efficient UI updates.
- **RESTful API handling**: Simplifies interaction with RESTful services and APIs.

62. What is Angular CLI?

Angular CLI is a command-line interface tool that helps automate the development workflow, including creating, building, testing, and deploying Angular applications.

63. What is a module in Angular?

A module is a container for a cohesive group of components, directives, pipes, and services. It is defined using the @NgModule decorator.

64. What is a directive in Angular?

A directive in Angular is a special instruction that extends HTML functionality by attaching custom behaviors to elements in the DOM. Directives help manipulate the structure, appearance, and behavior of elements dynamically.

Angular provides three types of directives

- **Component Directives:** These are directives with a template, and they form the building blocks of Angular applications (e.g., @Component).
- **Structural Directives:** These alter the layout by adding or removing elements from the DOM (e.g., *ngIf, *ngFor, *ngSwitch).

- **Attribute Directives:** These change the appearance or behavior of an element (e.g., ngClass, ngStyle, custom directives).

Apart from that, Angular allows us to create custom directives to add reusable functionalities and enhance HTML elements based on specific project needs.

65. What is scope and Data Binding in AngularJS?

- **Scope:** Scope in AngularJS is the binding part of HTML view and JavaScript controller. When you add properties into the scope object in the JavaScript controller, only then the HTML view gets access to those properties.
- **Data Binding:** Angular provides a function Data Binding which helps us to have an almost real-time reflection of the input given by the user i.e. it creates a connection between Model and View.

66. What is a Vue instance? How can we create a Vue instance?

VueJS instance is the root of all Vue application. It is JavaScript object formed by the 'Vue' constructor function and assists as the initiating point for building a Vue application.

Creating Vue Instance

```
var app = new Vue({
    // Options object
    el: '#app', // The element to mount the Vue instance to
    data:
    {
        message: 'Hello GeeksForGeeks!'
    }
)}
```

67. What is the watcher in VueJS?

A watcher in Vue.js is a mechanism designed to monitor changes in data properties within a component. It allows developers to execute

specific functions in response to those changes, making it useful for handling complex logic when reactive properties are updated.

68. Explain Virtual Dom in VueJs?

In VueJS, the Virtual DOM (VDOM) is an approach used to upgrade execution when updating web pages. It works like a blueprint of the real web page's structure, kept in memory.

- When a Vue component changes, Vue first update VDOM instead of directly editing the real DOM.
- Then, it matches the improved VDOM with the earlier one to figure out the smallest number of changes vital in the real DOM.

69. Explain Hooks in VueJS?

Hooks are a new feature introduced in VueJS 3. Hooks are functions that permit adding reusable logic to components. They permit managing state, lifecycle events, and side effects within functional components.

Some of the most typically used hooks include

- '**onMounted**' : The 'onMounted' hook enacts directly after the component is mounted onto the DOM. It is best for executing tasks like DOM manipulations or fetching data, assuring these operations happen at the right moment while the component's lifecycle.
- '**onUnmounted**' : The 'onUnmounted' hook initiates just before the component is eliminated from the DOM. It is impeccable for doing cleanup tasks, assuring that any vital operations are done before the component is no longer active in the application.

70. What is Vue CLI and how is it used?

Vue CLI (Command Line Interface) is a tool for fast support VueJS projects. This is a command-line utility that permits you to choose from a range of build tools. With Vue CLI, developers can rapidly set

up a new project with best practices and modern build tools like Webpack or Vite.

Frontend Developer Interview Questions - Advanced

71. What is an anchor tag in HTML?

The <a> tag (anchor tag) in HTML is used to create a hyperlink on the webpage. This hyperlink is used to link the webpage to other web pages. It's either used to provide an absolute reference or a relative reference as its "href" value. Click Here to know more in detail.

Syntax

```
<a href = "link"> Link Name </a>
```

72. How to create scrolling text or images on a webpage?

This task can be achieved through <marquee> tag in HTML that helps to create scrolling text or image on a webpage. It scrolls either from horizontally left to right or right to left, or vertically from top to bottom or bottom to top.

Syntax: The marquee element comes in pairs. It means that the tag has an opening and closing elements.

```
<marquee>
<--- contents --->
</marquee>
```

73. How can you apply JS in your HTML?

Scripts can be placed inside the body, the head section of an HTML page, inside both head and body, or can be added externally.

- **JavaScript in head:** A JavaScript function is placed inside the head section of an HTML page and the function is invoked when a button is clicked.

- **JavaScript in the body:** A JavaScript function is placed inside the body section of an HTML page and the function is invoked when a button is clicked.
- **External JavaScript:** JavaScript can also be used as external files. JavaScript files have file extension .js . To use an external script put the name of the script file in the src attribute of a script tag.

74. How you can merge the rows and columns of a HTML table?

We can use the colspan and the rowspan attributes with the <td> element and specify the number of rows and columns to be merged by passing a numerical value to the defined attributes. The colspan attribute can be used to merge columns while the rowspan to merge the rows.

75. What is the purpose of using <figure> and <figcaption> elements in HTML5?

The <figure> element is used to display the media content on the web page like audios, videos etc. While, the <figcaption> element is used to give a caption or legend to the content shown by the <figure> element.

76. What are pseudo classes and pseudo elements in CSS?

The pseudo classes and pseudo elements are different entities in CSS. They are combinely known as pseudo selectors in CSS. Below is the explanation for them:

- **pseudo classes:** These are the classes that selects the elements based on their state and the position. Some pseudo classes are :hover, :focus, :nth-child etc.
- **pseudo elements:** These are the virtual elements that are mainly defined to style a particular part of an element in the HTML document. Some pseudo elements are :before and :after.

77. What is Media Queries in CSS?

Media queries are the block of CSS code defined for a particular width or range of the width. These can be defined using the @media keyword with screen to specify styles for a particular width or range of width. They are used very commonly to create responsive designs.

78. How to create responsive designs?

There are some key concepts available in CSS that can help you in creating responsive designs as listed below

- Using Media queries
- Using the flexbox layout
- Using the grid layout
- Using responsive CSS properties like percentage and vh, vw.ow to create responsive designs

79. How you can optimize the loading of CSS files in browser?

There are multiple techniques available to optimize loading of CSS files as listed below

- By minimizing number of CSS files.
- CSS minification
- By leveraging the browser cache
- Load the un-necessary styles using asynchronous or deffered loading.

80. What is the meaning of 'Cascading' in Cascading Style Sheet?

Cascading represents the specificity order in applying styles. These styles can be defined by the user, author or they can be the default browser styles. The specificity order for the styles is user styles > author styles > default browser styles.

81. What is Redux in React?

Redux is the state management library for React applications. Redux simply helps to manage the state of your application or in other words, it is used to manage the data of the application. It is used with a library like React.

To install the redux follow the below command

```
npm install redux react-redux
```

82. What is the Context API?

Context API in React is used to share data between the components without passing the props(prop drilling) manually through every level. It allows to create global state of data providing global access to all the components.

The Context API consists of the three main parts

- **createContext():** Creates a Context object.
- **Provider:** Provides the state to components.
- **useContext():** Accesses the state inside any component.

83. Difference between the Redux and the Context API.

Context API	Redux
Built-in React feature for prop drilling prevention and state sharing across components.	A state management library for complex global state handling.
Works best for lightweight state sharing (e.g., theme, language, auth state).	Best for large-scale applications with complex data flow.

Context API	Redux
No extra installation needed (built into React).	Requires installing Redux (npm install redux react-redux).
Every consumer component re-renders on state updates.	Efficient updates using Redux's selective rendering (connect, useSelector).
Small to medium apps user authentication, language settings.	Large-scale apps, Complex state like API caching, notifications, user sessions.

84. What is prop drilling?

Prop drilling in React refers to the process of passing data (props) from a parent component down to deeply nested child components through multiple intermediate components, even if those intermediate components do not directly use the data. This can lead to unnecessary complexity and reduced maintainability.

We can avoid the prop drilling by two ways

- Using Context API
- Using Custom Hooks

85. What is Debouncing in JavaScript?

In JavaScript, debouncing is commonly used to enhance browser performance by ensuring that expensive operations (like complex calculations, API calls, or DOM updates) are executed only when necessary. JavaScript operates in a single-threaded environment, meaning it can only handle one operation at a time. When certain actions are triggered too frequently, such as during continuous

scrolling or typing, it can overload the browser and cause sluggish performance.

86. What is the differences between Java and JavaScript?

Java	JavaScript
Object-Oriented Programming (OOP) Language	Scripting Language for web development
Compiled into bytecode (.class files) that runs on JVM	Runs in the browser (Frontend) and also on servers (Node.js)
Class-based OOP (Objects are created from classes)	Prototype-based OOP (Objects inherit from other objects)
Faster for heavy computations (since it runs on JVM)	Slower for CPU-intensive tasks (single-threaded)
Automatic Garbage Collection (JVM handles it)	Automatic Garbage Collection (handled by the browser/Node.js)

87. What is a template literal in JavaScript?

Template Literal in ES6 provides new features to create a string that gives more control over dynamic strings. Traditionally, String is created using single quotes (‘) or double quotes (“) quotes. Template literal is created using the backtick (`) character.

```
let s=`some string`;
```

88. What is a higher-order function in React.js?

Higher-order components or HOC is the advanced method of reusing the component functionality logic. It simply takes the original component and returns the enhanced component.

```
const EnhancedComponent =
higherOrderComponent(OriginalComponent);
```

89. What is the Temporal Dead Zone (TDZ) in JavaScript?

The Temporal Dead Zone refers to the period between the entering of a scope and the actual declaration of a variable using let or const. During this period, the variable is in an "uninitialized" state and accessing it will result in a ReferenceError.

90. What is the difference between call() and apply() methods ?

call() Method	bind() Method
Calls a function immediately with a given this value and arguments.	Returns a new function with this value and arguments bound, but does not execute it immediately.
function.call(thisArg, arg1, arg2, ...)	function.bind(thisArg, arg1, arg2, ...)
Executes the function immediately.	Does not execute the function immediately, instead returns a new function.
Returns the result of the function execution.	Returns a new function with this permanently bound.
js function greet() { console.log(this.name); } let user	js function greet() { console.log(this.name); } let user = {

call() Method	bind() Method
= { name: "Sandeep" }; greet.call(user); // Output: "Sandeep"	name: "Sandeep" }; let newGreet = greet.bind(user); newGreet(); // Output: "Sandeep"

91. How many types of Directives are available in AngularJS?

There are four kinds of directives in AngularJS those are described below

- Element directives
- Attribute directives
- CSS class directives
- Comment directives

92. What is factory method in AngularJS?

AngularJS Factory Method makes the development process of AngularJS application more robust.

- A factory is a simple function that allows us to add some logic to a created object and return the created object.
- The factory is also used to create/return a function in the form of reusable code which can be used anywhere within the application.
- Whenever we create an object using a factory it always returns a new instance for that object.
- The object returned by the factory can be integrated(injectible) with different components of the Angularjs framework such as controller, service, filter or directive.

93. What is the digest cycle in AngularJS?

It is the most important part of the process of data binding in AngularJS. It basically compares the old and new versions of the scope

model. The digest cycle triggered automatically. If we want to trigger the digest cycle manually then we can use $apply().

94. What is an Angular router?

The Angular router is a library that helps to manage navigation and routing in Angular applications, enabling single-page application (SPA) behavior.

95. What are Angular lifecycle hooks?

Angular lifecycle hooks are methods that allow you to tap into key moments in a component's lifecycle. Here are the main lifecycle hooks:

- **ngOnInit():** Called once after the component's data-bound properties have been initialized.
- **ngOnChanges(changes: SimpleChanges):** Called whenever one or more data-bound input properties change.
- **ngDoCheck():** Called during every change detection run, allowing you to implement your own change detection.
- **ngAfterContentInit():** Called once after Angular projects external content into the component's view.
- **ngAfterContentChecked():** Called after every check of projected content.
- **ngAfterViewInit():** Called once after the component's view (and child views) has been initialized.
- **ngAfterViewChecked():** Called after every check of the component's view (and child views).
- **ngOnDestroy():** Called just before Angular destroys the component, allowing you to clean up resources.

96. What is Vue-loader?

Vue Loader is a webpack loader for VueJS that permits to write Vue components in '.vue' file format. It empowers VueJS applications to use Single File Components (SFCs), which encapsulate HTML, CSS, and

JavaScript into a single file. They compile these components into JavaScript modules that the browser can grasp, easing component-based development in VueJS projects.

97. Discuss the v-cloak directive in VueJS?

The 'v-cloak' directive in VueJS hides components until they are compiled, preventing the display of compiled Vue templates on page load. It assures an effortless user experience by excluding the wavering outcome generated by delayed rendering, utilizing CSS to hide elements as far as Vue processing completes.

98. What is vue plugin?

Developers can form and add global level features to Vue via Vue plugin. This can be utilized to add globally available methods to an application. VueFire, Vue plugin that adds Firebase specific methods and binding to the intact application, is an example of Vue plugin.

99. Name some websites which are using VueJS?

- Netflix
- Adobe
- Facebook
- Gitlab
- Xiaomi
- Alibaba

100. How can you install VueJS in your project?

To install VueJS in your project, you can follow these steps:

Using CDN: Include VueJS directly via a CDN link in your HTML file: