IP Viva Question

1) Explain how a web browser works when rendering a web page.

Ans: A web browser fetches HTML, CSS, and JavaScript from a server, builds a DOM, applies styles, runs scripts, and renders the web page visually on the screen.

2) What is the purpose of the HTTP protocol?

Ans: The HTTP (Hypertext Transfer Protocol) is designed to facilitate communication between clients (like web browsers) and servers. It defines how requests for resources (like web pages, images, or data) are made by clients and how servers respond by sending the requested data.

3) What are promises in JavaScript, and how do they handle asynchronous operations?

Ans:

Promises in JavaScript are a way to handle tasks that take time, like loading data from the internet. A promise represents a value that may not be available yet but will be in the future.

A promise can be in one of three states:

1. **Pending**: The promise is still waiting for the operation to complete.
2. **Fulfilled**: The operation finished successfully, and the promise has a value.
3. **Rejected**: The operation failed, and the promise has an error.

You can use `.then()` to run some code when the promise is fulfilled and `.catch()` to handle any Error.

4) How does HTTPS differ from HTTP?

ans:

HTTPS (Hypertext Transfer Protocol Secure) differs from HTTP by adding a layer of security through encryption. While HTTP transmits data in plain text, making it vulnerable to eavesdropping, HTTPS encrypts the data exchanged between the client and server using SSL/TLS protocols, ensuring privacy

5) What is DNS, and why is it important for the web?

Ans:

DNS (Domain Name System) is a system that translates human-readable domain names (like www.example.com) into IP addresses (like 192.0.2.1) that computers use to identify each other on the internet. It is important for the web because it enables users to access websites easily without needing to remember complex numerical addresses, facilitating smooth navigation and

6) Explain the concept of a REST API. How does it interact with the client and server?

Ans:

A **REST API** allows clients to interact with servers using standard HTTP methods to manage resources identified by URLs. It operates statelessly, meaning each request from the client contains all necessary information for the server to process it and respond accordingly.

7) What is the Document Object Model (DOM), and how is it used in web programming?

Ans:

The **Document Object Model (DOM)** is an interface that represents a web page's structure as a tree of objects. It enables developers to manipulate HTML and XML documents dynamically, making it crucial for interactive web applications.

8) What are the differences between URI and URL?

Ans:

A **URI** (Uniform Resource Identifier) is a general term for any string that identifies a resource on the internet, like a name or a link. A **URL** (Uniform Resource Locator) is a specific type of URI that tells you how to find that resource, including the address and the method to access it (like using HTTP).

9) What are the key differences between ES5 and ES6 in JavaScript?

Ans:

ES6 (ECMAScript 2015) introduced features like `let` / `const`, arrow functions, classes, modules, and template literals, whereas ES5 had more basic syntax and functionality. ES6 focuses on cleaner, more modern code structures, enhancing readability and performance.

10) Can you explain how arrow functions differ from traditional functions in JavaScript?

Ans:

Arrow functions have a shorter syntax and do not have their own `this` context, inheriting it from their surrounding scope, unlike traditional functions which bind `this` dynamically. They also do not support `arguments` or `new` keywords.

11) How does JavaScript interact with the DOM for manipulation of web content?

Ans:

JavaScript interacts with the DOM by accessing and modifying elements, attributes, and content, enabling dynamic changes to a web page.

12)What are iterators and generators, and how do they differ?

Ans:

Iterators in JavaScript provide a way to sequentially access elements in a collection, while generators are functions that can pause and resume execution, producing values lazily using `yield`.

13) Can you explain client-server communication using Fetch API?

Ans:

The Fetch API allows client-server communication by sending asynchronous HTTP requests and receiving responses, handling data through Promises.

14) What is asynchronous JavaScript, and how does it differ from synchronous operations?

Ans:

Asynchronous JavaScript executes tasks without blocking the main thread, allowing other code to run while waiting for operations to complete, unlike synchronous operations which execute sequentially, blocking further execution.

15) What is JSON, and how is it used in JavaScript?

Ans:

JSON (JavaScript Object Notation) is a lightweight data format used in JavaScript to store and exchange data, typically between a client and a server.

16) What is React, and why is it used for building web applications?

Ans:

React is a tool for building interactive web pages by breaking them into reusable parts (components), making it easy to create and update content quickly.

17) Can you explain the component lifecycle in React?

Ans:

The React component lifecycle consists of phases like mounting, updating, and unmounting, where components are created, updated, and removed from the DOM with methods like `componentDidMount` and `componentDidUpdate`.

18) What are props and state in React, and how are they used in components?

Ans:

In React, **props** are like ingredients you give to a component to customize it, while **state** is the component's own data that can change, helping it keep track of things like user input or settings.

19) How does React Router work, and what is its purpose in single-page applications?

Ans:

React Router helps manage navigation in single-page applications by letting you switch between different views or components without reloading the whole page, making the app feel faster and smoother.

20) ? Can you explain how events and forms are handled in React applications?

Ans:

In React, events and forms are handled by using special functions called event handlers to capture user actions, like clicks or typing, and updating the component's state to reflect changes.

21) What is the difference between React and Angular?

Ans:

React is a library focused on building user interfaces with reusable components, while Angular is a full framework that provides a complete solution for building web applications with features like routing and state management.

22) What are functional components, and how do they differ from class-based components in React?

Ans:

Functional components are simpler, stateless functions that return JSX to create UI, while class-based components are more complex, using classes and managing their own state and lifecycle methods.

23) Explain the role of hooks in React.

Ans:

Hooks in React are special functions that let you use state and other React features in functional components, making it easier to manage data and side effects without needing to use class-based components.

24) What is the Model-View-Controller (MVC) framework, and how does it apply to React?

Ans:

The MVC framework organizes code by separating data (Model), user interface (View), and logic (Controller), with React components primarily serving as the View that manages state and user interactions.

25) What is Flux architecture in React, and how does it manage data flow?

Ans:

Flux architecture in React is a way to manage data where information flows in one direction: actions change the data in a store, and then the views update to reflect those changes, making it easier to keep track of everything.

26) ⍰ What is Webpack, and why is it important for bundling React applications?

⍰ ans:

Webpack is a tool that combines all the files and assets of a React application into one package, making it faster to load and easier to manage by optimizing the code for the web.

27) How does React Native differ from standard React?

Ans:

React Native is used for building mobile applications using native components for iOS and Android, while standard React is focused on building web applications using HTML and web technologies.

28) What is Node.js, and how does it differ from traditional server-side technologies?

Ans:

Node.js lets you run JavaScript on the server, using a method that can handle many connections at once without waiting, while traditional server technologies often slow down by processing one request at a time.

29) Explain how asynchronous programming works in Node.js.

Ans:

Asynchronous programming in Node.js allows tasks to run in the background without stopping the main thread, enabling the server to handle other requests while waiting for operations like database queries or file reads to complete.

30) What are event emitters in Node.js, and how do they function?

Ans:

Event emitters in Node.js are objects that allow you to create and manage custom events, where you can trigger events and set up listeners to respond when those events occur.

31)What is the purpose of the REPL in Node.js?

Ans:

The **REPL (Read-Eval-Print Loop)** in Node.js is an interactive environment that lets you execute JavaScript code, evaluate expressions, and see results immediately, making it great for testing and debugging.

32) How does Node.js handle file system operations?

Ans:

Node.js handles **file system operations** using its built-in `fs` module, allowing you to read, write, and manipulate files asynchronously without blocking other tasks.

33) Can you explain the networking module in Node.js?

Ans:

The **networking module** in Node.js provides tools for creating and managing network connections, enabling you to build servers and handle requests and responses over the internet easily.

34) What is Express.js, and how is it used to build web applications?

Ans:

1. **Express.js** is a web framework for Node.js that simplifies building web applications by providing tools and features for routing, middleware, and handling requests and responses.

35) How does the Express router work, and what is its purpose?

Ans:

The **Express router** is a way to organize routes in your application, allowing you to define different endpoints and group them logically, making your code cleaner and easier to manage.

36) What is a REST API, and how does Express.js implement it?

Ans:

A **REST API** is a way to create web services that follow specific rules for how clients and servers communicate; Express.js helps build REST APIs by providing easy methods for handling HTTP requests like GET, POST, PUT, and DELETE.

37) How does Express handle authentication and session management?

Ans:

Express handles **authentication and session management** by using middleware like Passport or session libraries that manage user login states, keeping track of who is logged in and their session data.

38) What are the key benefits of integrating Express with React?

Ans:

The key benefits of integrating **Express with React** include easier management of API routes, better organization of server-side code, and the ability to serve both the front end and back end in a single application.

39) How does MongoDB work with Node.js and Express?

Ans:

MongoDB works with Node.js and Express by using a library like Mongoose to connect to the database, allowing you to perform operations like creating, reading, updating, and deleting data within your web application.