Q.1.) What is Emmet ?

Emmet is the essential toolkit for web-developers. It allows you to type shortcuts that are then expanded into full pieces of code for writing HTML & CSS, based on an abbreviation structure most developers already use that expands into full-fledged HTML markup and CSS rules.

Q.2.) Difference b/w a Library and Framework ?

A Library is a collection of packages that perform specific operations whereas a framework contains the basic flow and architecture of an application. The major difference b/w them is complexity. Libraries contains a number of methods that a developer can just call whenever they write code. Reactjs is library and Angular is framework. The framework provides the flow of a software application and tells the developer what it needs and calls the code provided by the developer as required. If a library is used, the application calls the code from the library.

Q. 3.) What is CDN ? Why do we use it ?

Content Delivery Network CDN is a network of servers located in diff. geographical locations that are used to deliver content to users based on their location in a secure and efficient way.

CDN's are used to improve the performance and availability of websites and applications by serving content from servers that are closer to the user, reducing the distance the data has to travel and minimizing latency.

CDN's are used for a variety of purposes, including :

→ Improving the speed of webpages.

→ Reducing bandwidth costs.

→ Enhancing security

→ Improves availability - provides multiple copies of content, if one server goes down.

→ CDN's are used by websites & apps, including e-commerce sites, entertainment platforms, and Software as a Service (Saas) providers.

Q. 4.) Why is React known as React ?

React is named 'React' because of its ability to react to changes in data. "react" to changes in the state of an application, by efficiently rendering and updating the UI in response to those changes.

React is called 'React' bcoz it was designed to be a declarative, efficient, and flexible javascript library for building user interfaces. The name React was chosen bcoz the library was designed to allow developers to "react" to changes in state and data within an application, and to update the user interface in a declarative and efficient manner.

React is a JavaScript-based UI development library. Facebook and an open-source developer community runs it.

Q. 5.) What is crossorigin in script tag ?

The crossorigin attribute is used in the script tag to indicate that the script should be loaded from a different origin (domain, protocol or port) than the current page. This is used to allow web pages to load resources from a diff. domain, while still allowing the browser to apply security measures such as the same-origin policy.

The crossorigin attribute can be set to either 'anonymous or use-credentials'. If set to anonymous, the browser will include an Origin header with the request, but will not send any cookies or other credentials. If set to use-credentials, the browser will include credentials with the request.

```
< script  src = "https... "  crossorigin = " anonymous | use-credentials ">
```

Basically, it is used to handle the CORS request that checks whether it is safe to allow for sharing the resources from other domains.

Q. 6.) What is diff. b/w React and ReactDOM ?

React and ReactDOM are two separate libraries that are often used together in the development of web applications with React. React is a library for building user interfaces, while ReactDOM is a library for interacting with the DOM and rendering React components to the webpage. While they are often used together, they serve diff. purposes and can be used independtly of each other.

The react package contains React.createElement(), React.Component, React.Children and other helpers related to elements and component classes.
The react-dom package contains ReactDOM.render(), and in react-dom/server we have server-side rendering support with ReactDOMServer.renderToString() and ReactDOMServer.renderToStaticMarkup().

Q. 7.) What is diff. b/w react.development.js and react.production.js files via CDN.

These files contains the same code, but are optimised for diff. environments and have diff. features enabled. Development is the stage of an application before it's made public while production is the term used for the same application when its made public. Development build is several times (maybe 3-5x) slower than the production build.

react.development.js file -> includes additional features & debugging tools that can be helpful when building & testing React applications. These features may include additional warning msgs, error checking & tools to identify problems or issues in code.

react.production.js file -> used in production environment & has been optimised for performance and size. It doesn't include debugging tools or additional features disable to reduce size and improve performance.

Q. 8.) What is async and defer ?

Async - The async attribute is a boolean attribute. The script downloaded in parallel (in the background) to parsing the page, and executed as soon as it is available (do not block HTML DOM construction during downloading process) and don't wait for anything.

< script src = " demo_async.js" async > < /script >

The defer attribute tells the browser to download the script in the background while the page is still loading, but to execute the script only after the page has finished loading. This can also improve the loading speed of the page, as the script ~~code~~ does not block the rendering of the page. However, unlike async, defer ensures that the script is executed in the order it appears in the HTML.

< script src = " demo_defer.js" defer > < /script >

Q.9.) What is NPM ?

It is a tool used for package management and the default package manager for Node projects. NPM is installed when Nodejs is installed on a machine. It comes with a command-line-interface (CLI) used to interact with the online database of NPM. This database is called the NPM registry, and it hosts public and private 'packages'. To add or update packages, we use the NPM CLI to interact with this database.

initialize npm ≡ npm init → creates package.json file

Q.10.) What is .parcel-cache ?

.parcel-cache is used by parcel (bundler) to reduce the building time. It stores info. about your project when parcel builds it, so that when it rebuilds, it doesn't have to re-analyze everything from scratch. It's a key reason why parcel can be so fast in development mode.

Q.11.) What is npx ?

npx is a tool that is used to execute the package. It comes with npm. It is an npm package runner that can execute any package that you want from npm registry without even installing that package.

Q.12.) What is diff. b/w dependencies vs devDependencies ?

Dependencies should contain library & framework in which our app is built on, needs to function effectively. such as Vue, React, Angular, Express, JQuery etc.

DevDependencies should contain modules/packages a developer needs during development. such as parcel, webpack, vite, mocha. These packages are necessary only while you are developing your project, not neccessary on production.

To save a dependency as a Devdependency on installation we need to do;

`npm install --save-dev`

Q. 13.) What is Tree shaking ?

Tree shaking is a process of removing the unwanted code that we do not use while developing the application. In computing, tree shaking is a dead code elimination technique that is applied when optimizing code.

Q. 14.) What is Hot Module Replacement ?

Hot Module Replacement (HMR) exchanges, adds or removes modules while ~~developing~~ an application is running, without a full reload. This can significantly speed up development in a few ways: Retains application state which is lost during a full reload.

Q. 15.) superpowers of Parcel ?

* HMR - adds, removes modules while an app.n is running without a full reload.
* File Watcher Algo. - monitors directories on the file system and perform specific actions when desired file appears.
* Minification
* Image Optimization - images takes time to load in a website.
* Caching while development

Q. 16.) 'dist' folder keeps the files minified for us.

When we run command : `npx parcel index.html`
This will create a faster development version of our project & serves it on the server.
When we tell parcel to make a production build, `npx parcel build index.html` It creates a lot of things, minify your file. And " parcel will build all the production files to the dist folder.

The /dist folder contains the minimized version of the source code. The code present in the /dist folder is actually the code which is used on production web applications. Along with the minified code, the /dist folder also comprises of all the compiled modules that may or may not be used with other systems.

Q. 17.) What is .gitignore ? What should we add and not add into it ?

The .gitignore file is a text file that tells Git which files or folders to ignore in a project during commit to the repository.

The types of files you should consider adding to a .gitignore files are any files that do not get committed.

For example ; For the security , the security key files and API's keys should get added to the gitignore. package-lock.json should not add into your .gitignore file.

e.g: .gitignore file look like :

```
# Ignore node_modules folder
node_modules
# Ignore all text files
*.txt
```

Q. 18.) What is diff. b/w package.json and package-lock.json ?

package.json :

- The file is mandatory for every project.
- It contains the basic information about project.
- Application name / version / scripts

package-lock.json :

- This file is automatically generated for those operations where npm modifies either the node_modules tree or package.json.
- It is generated after an npm install.

- It allows to go back to the past version of the dependencies without actual committing the node_modules.
- It records the same version of the installed packages which allows to reinstall them. Future installs will be capable of building identical descriptions tree.

Q. 19.) Why should we not modify package-lock.json ?

package-lock.json file contains the info. about the dependencies and their versions used in the project. Deleting it would cause dependencies issues in the production environment. So don't modify it, it's being handled automatically by NPM.

Q. 20.) What is node_modules ? Is it good to push on git ?

node_modules folder is like a cache for the external modules that your project depends upon. When you npm install them, they are downloaded from the web and copied into the node_modules folder and Node.js is trained to look for them, when you import them.

Don't push node_modules in github because it contains a lot of files (↑ 100 MB) it will cost more space and it can be regenerated.

Q. 21.) What is JSX ?

JSX stands for javascript XML. JSX allows us to write HTML elements in javascript and place them in the DOM without any createElement() or appendChild() methods. JSX makes it easier to write and add HTML in React. JSX converts HTML tags into react elements.

JSX uses React.createElement behind the scenes.

JSX ⇒ React.createElement ⇒ Object ⇒ HTML (DOM)

```
function greet (user) {
    return <> {user} Hii </> }
```

Q. 22.) Role of type attribute in script tag ? What options can we use there ?

The type attribute specifies the type of the script. The type attribute identifies the content b/w the <script> and </script> tags. It has a default value which is "text/javascript".

```
<script type="text/javascript"> </script>
```

- text/javascript : writing javascript code in script tag.
- text/ecmascript : script is following EcmaScript standards.
- module : script is a module that can import or export other files or modules inside it.
- text/babel : script is a babel type and required babel to transpile it.
- text/typescript : script is written in Typescript.

Q. 23.) {TitleComponent} : javascript expression or a variable.

<TitleComponent /> : function returning JSX value without child components.

<TitleComponent> </TitleComponent> : function with child.

Pollyfill : to make older browsers understand our new code, the new code is converted into a older code which browsers can understand called pollyfill.

→ babel do this conversion automatically.
"browserlist" → our code is converted to older one.

Babel is a javascript package / library used to convert code written new versions of js. to old.

To run our app, command is ;

```
npx parcel index.html
```

We always don't have to write this command.
Generally, we build a script inside package.json which runs this command in an easy way.

package.json

```
"scripts" : {
        "start" : "parcel index.html",
        "test" : "jest"
}
```

So, to run the project : npm run start ≡ npm start

Build command npx parcel build index.html

```
"scripts" : {
        "build" : "parcel build index.html"
}
```

Note :- npx = npm run

So, npm run build will build a project.

* Console logs are not removed automatically by parcel. we have to configure our projects to remove it.
There is a package which helps to remove console logs :-

babel-plugin-transform-remove-console

Before installing this package create a folder .babelrc for configuration. And include ;

```
{ "plugins": [["transform-remove-console",
        { "exclude": ["error", "warn"] }
    ]]
}
```

Then build : npm run build
to see that all console.logs are removed.

* Our browser can't understand jsx 'Babel' understand this code.

Render - means updating something in the DOM.

Spread Operator : (...)

It allows an iterable (an object that can be looped over , such as an array or a string) to be expanded in places where multiple elements are expected.

// E.g. In our code for Body (), multiple elements are to be written for every < RestaurantCard /> So, we can write,

< RestaurantCard {...restaurantList[0].data} />

Q. 25.) What is Reconciliation in React ?

Reconciliation is the process through which React updates the Browser DOM and makes React work faster . React uses a diffing algo. so that component updates are predictable and faster . React would first calculate the difference b/w the real DOM and virtual DOM when there's an update of components. React stores a copy of Browser DOM which is called virtual DOM . When we make changes or add data, React creates a new Virtual DOM and compares it with previous one. Comparison is done by diffing algo.. React compares the Virtual DOM with Real DOM. It finds out the changed nodes and updates only the changed nodes in Real DOM leaving the rest nodes as it is. This process is called Reconciliation.

Q. 26.) What is React Fiber ?

React Fiber is a concept of ReactJS that is used to render a system faster , smoother and smarter , for React 16 and above. Because Fiber is asynchronous. This algo. helps to compare two DOM trees and diff. them . React fiber helps to do it better.

- Fiber focuses on animations and responsiveness.
- It has the ability to split work into chunks and prioritize tasks.
- We can pause work and, resume and restart rendering work on components as new updates come in.
- Reuse previously completed work or maybe about it if it is not needed.
- As opposed to the old React reconciler, it is asynchronous.

Q. 27.) Why do we need Keys in React ?

A Key is a special attribute you need to include when creating list of elements in React. Keys are used in React to identify which items in the list are changed, updated or deleted. Keys are unique identifier.

Q. 28.) Diff. b/w Virtual DOM & Real DOM ?

DOM is an interface that allows scripts to update the content, style and structure of the document.

Real DOM	Virtual DOM
DOM manipulation is expansive.	DOM manipulation is easy.
There is too much memory wastage.	No memory wastage.
It updates slow.	It updates fast.
It can directly update HTML.	It can't update HTML directly.
Creates a new DOM if element updates.	Update the JSX if the element update.
It allows us to directly target any specific node (HTML element)	It can produce about 200,000 virtual DOM nodes / second.
It represents the UI of your app."	It is only a virtual representation of the DOM.

Q. 29.) What is the diff. b/w Named export, Default export and * as export ?

Named export

```
export const MyComponent = () => { }
export const MyComponent2 = () => { }
```

In named exports, one can have multiple named exports per file.

```
import { MyComponent, MyComponent2 } from "./MyComponent"
import { MyComponent2 as MyNewComponent } from "./MyComponent"
```

In Default export, one can have only one default export per file.

```
const MyComponent = () => { }
export default MyComponent;
```

```
import MyComponent from './MyComponent'
```

In * as export, it is used to import the whole module as a component and access the components inside the module.

MyComponent.js

```
export const MyComponent = () => { }
export const MyComponent2 = () => { }
export const MyComponent3 = () => { }
```

```
import * as MainComponents from "./MyComponent";
// now use as
<MainComponents.MyComponent />
<MainComponents.MyComponent2 />
<MainComponents.MyComponent3 />
```

We can use Named & Default export together

```
export const MyComponent2 = () => { }
const MyComponent = () => { }
export default MyComponent;
```

```
import MyComponent, { MyComponent2 } from './MyComponent';
```

Q. 30.) What is the importance of config.js file ?

→ Create a config file in your project

→ We put all the hardcoded things in config file

Also, known as (constants.js)

→ Config file will be like :-

```
const IMG_URL = "https://someurl";
export const IMG_URL ; make this as named export
```

→ `import { IMG_URL } from "./config"`

Q. 31.) What are React Hooks ?

React Hooks are simple Javascript functions that we can use to isolate the reusable part from a functional component. Hooks can be stateful and can manage side-effects. Hooks allow you to reuse stateful logic without changing your component hierarchy. This makes it easy to share Hooks among many components.

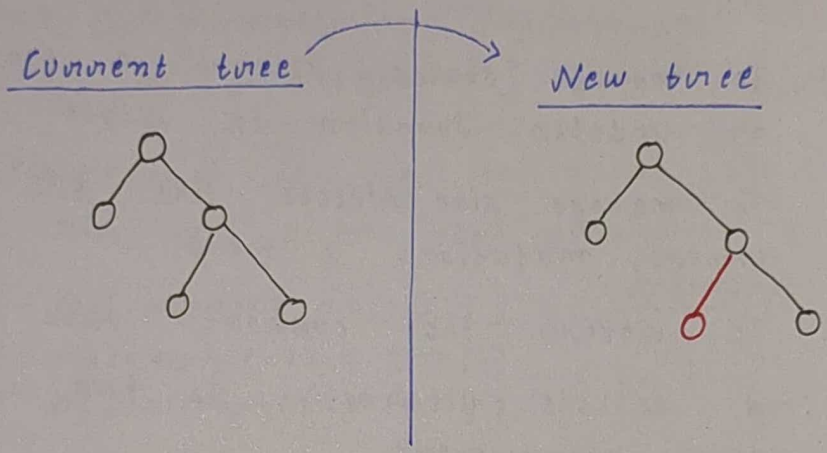
- useState : To manage states, Returns a stateful value and an updater function to update it.
- useEffect : To manage side effects like API calls, subscriptions, timers, mutations & more.
- useContext : To return the current value for a context.
- useReducer : A useState alternative to help with complex state management.
- useCallback : It returns a memorized version of a callback to help a child component not re-render unnecessarily.
- useMemo : It returns a memoized value that helps in performance optimization.
- useRef : It returns a ref object with a current property. The ref object is mutable. It is mainly used to access a child component imperatively.
- useLayoutEffect : It fires at the end of all DOM mutations. It's best to use useEffect as much as possible over this one as the useLayoutEffect fires synchronously.
- useDebugValue : Helps to display a label in ReactDevTools for custom hooks.

Q. 32.) Why is React fast ?

→ Because it has virtual DOM, Reconciliation, Diff algorithm.

→ In Diff algorithm, current tree is compared with the new tree and the difference is reflected on the dom.

→ React Fibre is the updated reconciliation algo.



→ React is fast because of it's fast DOM manipulation.

→ Diff Algo. detects what exactly got changed in the page & it will just change that while re-rendering the whole tree.

When a state variable is changed, React re-freshes or re-renders the whole component, and react do this super fastly.

component re-renders : State change / props change

Q. 33.) What is the difference b/w Monolith and Microservice architecture ?

In monolith architecture, there used to be a single big application. So, everything like API's, SMS, notification, UI, JSP pages etc used to be in same project. Suppose if we have to change one button, we used to deploy the whole project / application. It was such a mess. This architecture was known as monolith.

But in, Microservices Architecture we used to have small diff. projects. There are separate projects, there's separation of concern or single responsibility principle. The tools & languages used in a project depends on the usecase. All these projects are deployed in diff. ports but same domain name.

Q. 34.) Why do we need a useEffect Hook ?

useEffect Hook is javascript function provided by react. The useEffect hook allows you to eliminate side effects in your components. Some examples of side effects are; : fetching API data, directly updating the DOM, and setting up subscriptions or timers, etc can be lead to unwarranted side-effects.
useEffect accepts two arguments, a callback function and a dependency array. The second arg. is optional.

```
useEffect ( () => {}, [] )
```

If we don't pass empty dependency array then the useEffect runs everytime when the UI is rendered.

Q. 35.) What is Optional Chaining ?

Optional chaining (?.) operator accesses an object's property or calls a function. If the object accessed or function called is undefined or null, it returns undefined instead of throwing an error.
Optional chaining (?.) is a good way of accessing the object keys, it prevents the application from being crashed if the key that we are trying to access is not present. If the key is not present then instead of a throwing error, it returns undefined.

Q. 36.) What is the diff. b/w JS expression and JS statement ?

A JS expression returns a value [ ] ;

```
1 + 2
"abc". toUpperCase()
console. log (2)
isTrue ? true : false   // returns a value
```

A JS statement, does not return a value. [( )] ;

```
let x ;   // variable declaration
if () [ ]   // if condition
```

Q. 37.) What is Conditional Rendering ?

→ same as conditions (if operator or the conditional works in javascript. operator).

→ components needs to display diff. things depending on diff. conditions.

→ In React, you can conditionally render JSX using javascript syntax like :-

- if statement
- && operator
- ? : operator

avoid rendering components ?

Early Return

1.) if (restaurant) is empty ⇒ Load Shimmer UI

2.) if (restaurant) has data ⇒ Load actual UI

Q. 38.) What is CORS ?

Cross - Origin Resource Sharing (CORS) is an HTTP - header based mechanism that allows a server to indicate any origins (domain, scheme or port) other than its own from which a browser should permit loading resources. CORS define a way in which a browser and server can interact to determine whether it is safe to allow the cross - origin request.

Q. 39.) What is async and await ?

Async simply allows us to write promise-based code as if it was synchronous and it checks that we are not breaking the execution thread. It operates asynchronously via the event loop. Async functions will always return a promise. It makes sure that a promise is returned and if it is not returned then javascript automatically wraps it in a promise which is resolved with it's value.

Await could be used within the async block only. It makes the code wait until the promise returns a result. It only makes the async block wait.

Q. 40.) What is the use of const json = await data.json() ?

The data object, returned by the await fetch(), is a generic placeholder for multiple data formats. so we can extract the JSON object from a fetch response by using await data.json().
data.json() is a method on the data object that lets you extract a JSON object from the data or response. The method returns a promise because we have used await keyword.
so data.json() returns a promise resolved to a JSON object.

Q. 41.) What would happen if we do console.log(useState())?

If we do console.log(useState()), we get an array [undefined, function] where first item in an array is state is undefined and the second item in an array is setState function is bound dispatchSetState.

Q. 42.) What is SPA ?

Single Page Application (SPA) is a web application that dynamically updates the webpage with data from web server without reloading / refreshing the entire page. All the HTML, CSS, JS are retrieved in the initial load and other data/resources can be loaded dynamically whenever required. An SPA is sometimes referred to as a Single-Page-Interface (SPI).

Q. 43.) What is the diff. b/w Client Side Routing and Server side Routing ?

In Server-side routing or rendering (SSR), every change in URL, http request is made to server to fetch the webpage and replace the current webpage with the older one.

In Client-side routing or rendering (CSR), during the first load, the webapp is loaded from server to client, after which whenever there is a change in URL, the router library navigates the users to the new page without sending any request to backend. All Single-Page-App.ⁿ uses Client-Side-routing.

Q.44.) How do you create Nested Routes react-router-dom configuration ?

(1) first call createBrowserRouter for routing diff. pages.

```
const router = createBrowserRouter([
  {
    path : "/",          // show path for routing
    element : <Parent />, // show component for particular path
    errorElement : <Error />, // show error component for path is diff.
    children : [ // show children component for routing
      {
        path : "/path",
        element : <Child />
      }
    ],
  }
])
```

Now we can create a nested routing for "/path" using children :

```
const router = createBrowserRouter([
  {
    path : "/",
    element : <parent />,
    errorElement : <Error />,
    children : [
      {
        path : "/path",
        element : <Child />,
        children : [
          {
            path : "/child",
            element : <SubChild />,
          }
        ],
      },
    ],
  }
])
```

Q. 45.) (Research) Why can't we have the callback function of useEffect async ?

useEffect expects it's callback function to return nothing or return a function (cleanup function that is called when the component is unmounted).

If we make the callback function as async, it will return a promise and the promise will affect the clean-up function from being called.