**Q1. What is Emmet?**

**A1.** Emmet is a free add-on for your text editor. It allows you to type shortcuts that are then expanded into full pieces of code. By using Emmet, developers type less, they save both on keystrokes and time.

Emmet is a web development tool that allows developers to write HTML and CSS code quickly and efficiently. It is a plugin for text editors and integrated development environments (IDEs) that provides a set of abbreviations and shortcuts, which can be expanded into complete HTML and CSS code. For example, typing "div.container>ul.list>li.item" in an Emmet-enabled editor and expanding the abbreviation would generate the following HTML code:

<div class="container">  
 <ul class="list">  
 <li class="item"></li>  
 </ul>  
</div>

Emmet also supports a wide range of other features, including CSS-style selectors, mathematical operations, and custom snippets. By using Emmet, web developers can significantly speed up their workflow and reduce the amount of manual coding required.

**Q2. Difference between a Library and Framework?**

**A2.** In software development, a library and a framework are both tools that can be used to build applications, but they differ in their approach and level of control they provide to the developer.

A library is a collection of reusable code that can be called upon by the developer to perform specific tasks or provide certain functionality. Libraries typically focus on a specific area of functionality, such as data processing or user interface design. The developer has control over how and when to use the library, but the library itself does not dictate the overall architecture of the application.

A framework, on the other hand, is a more comprehensive tool that provides a structure and a set of rules for building applications. A framework includes libraries, but also defines the overall architecture and design patterns that should be followed. The developer has less control over the details of the application, but can take advantage of the framework's built-in functionality to streamline the development process.

In summary, a library is a collection of code that can be used to perform specific tasks, while a framework is a more comprehensive tool that provides a structure and a set of rules for building applications.

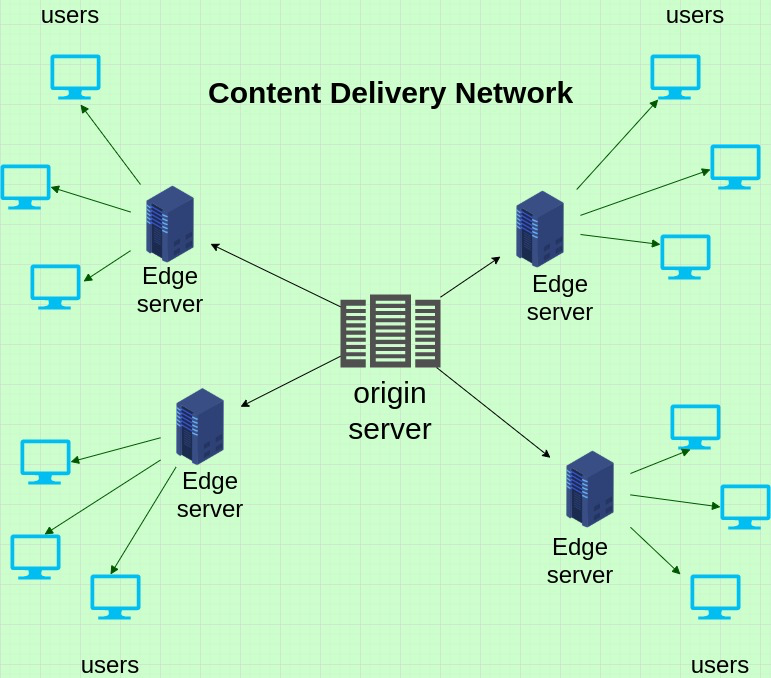
**Q3. What is CDN? Why do we use it?**

**A3.** CDN stands for Content Delivery Network or Content Distribution Network. It is a network of servers distributed across multiple geographic locations that work together to provide fast and reliable delivery of content, such as images, videos, and web pages, to end-users.

When a user requests content, the CDN serves it from the server that is closest to the user, reducing the distance that the data needs to travel and therefore reducing the load time.

CDNs also help to distribute the traffic across multiple servers, reducing the load on any one server and improving the overall performance and availability of the content.

CDNs are commonly used by websites, mobile apps, and other digital content providers to improve the user experience and increase the scalability and reliability of their services.



**Content Delivery Network (CDN) consists of two components:-**

First, **Origin Server**- where the content to be distributed over the internet is stored originally.

And, **Cache Server**- where the content is duplicated.

In simpler words, CDN is a network of servers that delivers content to users. Both React and ReactDOM are available over a CDN, links are mentioned below:

<script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>

<script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>

**Q4. Why is React Known as React?**

**A4.** React is called React because it is a library that reacts to changes in the data and renders the user interface accordingly. The name "React" comes from the concept of reactive programming, which is a programming paradigm that emphasizes the declarative description of the behavior of a system in response to changes in its inputs.

React was developed for applications (Facebook) that have constantly changing data. Since React is a front-end framework or the “View” in MVC, this means that as the user clicks around and changes the app’s data, the view should “react” or change with those user events. User events being mouse clicks, typing, submitting a form.

React's primary goal is to simplify the creation of complex, interactive user interfaces by breaking them down into smaller, reusable components. These components are built using a declarative syntax that allows developers to describe what the interface should look like at any given moment, and React takes care of efficiently updating the DOM (Document Object Model) as needed when the data changes.

In short, React's "reactive" approach to rendering UI components allows developers to build interfaces that are more responsive and easier to maintain.

The name "React" also reflects the fact that the library is built around the concept of a unidirectional data flow, where changes in data flow down through the component hierarchy, triggering updates and re-renders as necessary. This approach helps to avoid common issues with two-way data binding, such as performance problems and hard-to-debug code.

**Q5. What is cross-origin in the script tag?**

**A5.** The crossorigin attribute sets the mode of the request to an HTTP CORS Request.

Web pages often make requests to load resources on other servers. Here is where CORS comes in.

A cross-origin request is a request for a resource (example, style sheets, iframes, images, fonts, or scripts) from another domain.

CORS is used to manage cross-origin requests.

CORS stands for Cross-Origin Resource Sharing, and is a mechanism that allows resources on a web page to be requested from another domain outside their own domain. It defines a way of how a browser and server can interact to determine whether it is safe to allow the cross-origin request. CORS allows servers to specify who can access the assets on the server, among many other things.

The opposite of cross-origin requests is same-origin requests. This means that a web page can only interact with other documents that are also on the same server. This policy enforces that documents that interact with each other must have the same origin (domain).

In simpler words, 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.

**Q6. What is the difference between React & ReactDOM?**

**A6.** React and ReactDOM are both libraries in the React ecosystem, but they serve different purposes.

React is a JavaScript library for building user interfaces. It provides a set of tools for creating reusable UI components and managing the state of those components. With React, you can build complex, interactive user interfaces for web applications.

ReactDOM, on the other hand, is a library that provides a way to render React components to the DOM (Document Object Model) of a web page. It is responsible for updating the DOM based on changes in the React components. ReactDOM is used to mount React components to a web page and to update those components as needed.

In other words, React provides the logic and structure for building user interfaces, while ReactDOM handles the actual rendering of those interfaces to the web page. React and ReactDOM work together to create dynamic and interactive web applications.

In short, React is a JavaScript library for building User Interfaces and ReactDOM is the JavaScript library that allows React to interact with the DOM.

**Q7. What is the difference between react.development.js and react.production.js files via CDN?**

**A7.** React is a popular JavaScript library for building user interfaces, and it comes with two versions of its code: **react.development.js** and **react.production.js**.

These two versions have some key differences, especially in terms of their file size and performance, which can affect the user experience.

**react.development.js** is a development version of the React library. It contains extra warnings and debugging information that can be helpful during development and testing, but these features also make the file size larger. This file is not optimized for performance and is therefore slower compared to the production version.

On the other hand, **react.production.js** is an optimized version of the React library, designed for use in production environments. It has been stripped of unnecessary code and debugging information, resulting in a smaller file size and better performance. This file does not include any debugging information or warnings, making it faster but less helpful during development.

When using React via a CDN, you should use *react.production.js* for your production website, as it will provide a better user experience due to its smaller size and improved performance. During development, you can use *react.development.js* to take advantage of its debugging features and warnings.

**Q8. What are async and defer?**

**A8.** In web development, "async" and "defer" are attributes that can be added to the <script> tag to control how the browser loads and executes JavaScript files.

"**async**" stands for asynchronous, and when added to a script tag, it tells the browser to continue loading the rest of the page while the script is being fetched from the server. This means that the script is downloaded in the background, and as soon as it's available, it will be executed. This can improve the page loading speed because the browser doesn't have to wait for the script to be downloaded before continuing to load other resources.

"**defer**" also tells the browser to continue loading the page while the script is being fetched, but it ensures that the script is only executed after the page has finished parsing. This can be useful if the script depends on other resources on the page to be loaded, such as images or stylesheets, because it ensures that the script won't be executed until those resources are available.

In summary, "async" loads and executes the script asynchronously, while "defer" loads the script asynchronously but defers its execution until after the page has finished parsing.

