# REACT ASSIGNMENT Inception

1. **What is Emmet?**

* Emmet is a widely used web development tool and shorthand syntax for writing HTML and CSS code
* Emmet is often integrated into text editors and integrated development environments (IDEs) to enhance productivity.

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

* **Library**

A library is a collection of pre-written functions or modules that developers can use to perform specific tasks or operations.

Example: React JS

In a library, the control flow of the program remains with the developer. The developer decides how and when to call functions from the library.

* **Framework**

A framework is a pre-defined architecture or structure that outlines how an application should be designed and built. It often includes a set of rules, conventions, and guidelines that developers must follow to create an application using the framework.

Example: Angular JS

With a framework, the control flow of the application is inverted. Instead of developers calling framework code, the framework calls developer code. This is known as the "Inversion of Control" principle.

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

CDN stands for Content Delivery Network, and it is a distributed network of servers strategically located around the world. The primary purpose of a CDN is to deliver web content, such as text, images, videos, JavaScript files, and other resources, to users in a faster and more efficient manner. CDNs are designed to improve the performance, reliability, and scalability of websites and web applications.

Here are some key aspects of CDNs:

* **Caching**: CDNs use caching to store frequently requested content closer to end-users. This means that static assets like images, stylesheets, and scripts can be cached at various CDN edge servers, reducing the need to fetch them from the origin server repeatedly.
* **Load** **Balancing**: CDNs distribute incoming traffic across multiple servers to balance the load and prevent server overload. This helps ensure consistent performance and availability even during traffic spikes or DDoS attacks.
* **Security**: CDNs often offer security features such as DDoS protection, web application firewalls, and SSL/TLS encryption. These features help protect websites and web applications from various online threats.
* **Improved Page Load Times:** By delivering content from servers closer to the user, CDNs significantly reduce the time it takes for web pages to load.
* **Global Reach:** CDNs have a global network of servers, making it possible for websites to serve content to users around the world with minimal latency.
* **Analytics and Reporting:** Many CDNs offer analytics tools and reporting features that provide insights into website performance, traffic patterns, and user behavior.

1. **Why is React known as React?**

React, the popular JavaScript library for building user interfaces is known as "React" because its core concept revolves around reacting to changes in data and automatically updating the user interface to reflect those changes. The name "React" reflects the fundamental principle of the library: creating reactive and responsive user interfaces.

1. **What is crossorigin in script tag?**

The `**crossorigin**` attribute in the `<script>` tag is used to specify how a browser should handle cross-origin requests when loading external JavaScript files.

Cross-origin requests are HTTP requests for resources (like scripts, stylesheets, or images) that originate from a different domain or origin than the one serving the web page.

These requests are subject to the same-origin policy, which is a security feature implemented by web browsers to prevent potentially harmful interactions between web pages from different origins.

The `crossorigin` attribute can take the following values:

**crossorigin="anonymous":** This is the default setting. It indicates that the script can be loaded from a different domain without sending user credentials (like cookies) along with the request. This is suitable for public resources.

**crossorigin="use-credentials":** This setting tells the browser to include user credentials (such as cookies) when loading the script from a different domain. It's used when the script requires access to user-specific data on that domain.

1. **What is difference between React and ReactDOM**

React and ReactDOM are two essential packages in the React ecosystem, and they serve distinct but interrelated roles when building web applications with React.

**React:**

* Core Library: React is the core library for building user interfaces in JavaScript. It provides the fundamental concepts and APIs for creating and managing components, handling component state and props, and rendering UI elements based on the component hierarchy.
* Platform-Agnostic: React is designed to be platform-agnostic, which means it can be used not only for web development but also for other platforms like mobile (React Native) or desktop (React Native for Windows and macOS).
* Doesn't Interact with the DOM: React does not interact directly with the Document Object Model (DOM). Instead, it creates and manages a virtual representation of the DOM called the Virtual DOM. React components describe how the UI should look at any given time based on their state and props.

**ReactDOM:**

* React for the Web: ReactDOM is a package specifically designed for rendering React components in web applications. It serves as the bridge between React and the actual DOM.
* ReactDOM.render(): The primary method provided by ReactDOM is `ReactDOM.render()`. This method is used to take a React component (or a tree of components) and render it into the actual DOM. It is typically called once at the entry point of a web application to mount the root component.
* Interacts with the DOM: Unlike React, ReactDOM directly interacts with the DOM. It is responsible for updating the DOM efficiently when changes occur in the Virtual DOM. It reconciles the virtual representation of the UI with the real DOM, making updates as needed without re-rendering the entire page.

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

React provides two main versions of its library files: `react.development.js` and `react.production.js`. These files serve different purposes and are intended for different stages of the development process:

**react.development.js:**

* + Development Build: This file is used during the development phase of a React application.
  + Larger File Size: The development version is larger in file size because it contains additional error-checking and debugging information. This includes helpful warnings, error messages, and development-related features, which aid developers in identifying and diagnosing issues in their code.
  + Better Developer Experience: It provides a better developer experience by providing clear error messages, warnings, and helpful stack traces. This makes it easier to identify and fix issues during development.
  + Performance Overheads: Due to the additional debugging code and checks, the development version is slower in terms of performance compared to the production version.

**react.production.js:**

* + Production Build: This file is intended for use in a production environment when you're ready to deploy your React application.
  + Smaller File Size: The production version is smaller in file size because it omits the debugging information and checks present in the development version. This results in a more compact and efficient library.
  + Optimized for Performance: It is optimized for performance, as the absence of debugging features reduces overhead, resulting in faster execution.
  + No Warnings: The production version does not produce warnings or provide detailed error messages in the console. It is less verbose than the development version to improve runtime efficiency.

1. **What is async and defer?**

`async` and `defer` are two attributes that can be used in HTML script tags to control how JavaScript files are loaded and executed on a web page. They are used to optimize the loading and execution of scripts, especially when dealing with external scripts

**`async` Attribute:**

* When you include the `async` attribute in a script tag, it tells the browser to load the script asynchronously while continuing to parse and render the rest of the HTML content. This means that the script is fetched in the background without blocking the rendering of the page.
* The script can execute as soon as it is downloaded, which might be before the entire HTML document is parsed.
* The order in which multiple `async` scripts are executed can vary based on which script finishes downloading first. There's no guarantee of execution order.
* Use `async` when the script does not depend on the order of execution with other scripts or the DOM being fully parsed before running. For example, non-blocking analytics scripts are often loaded with `async`.

**`defer` Attribute:**

* When you include the `defer` attribute in a script tag, it tells the browser to load the script asynchronously as well but with one key difference: the script execution is deferred until after the HTML document has been fully parsed and the DOM is ready.
* Multiple `defer` scripts will be executed in the order they appear in the HTML document, just before the `DOMContentLoaded` event is fired.
* Use `defer` when the script relies on the DOM being fully parsed or when you want to ensure scripts are executed in the order they appear in the HTML document.