**Assignment-01 : inception**

**Theory –**

* What is Emmet?
* Difference between a Library and Framework?
* What is CDN? Why do we use it?
* Why is React known as React?
* What is crossorigin in script tag?
* What is diference between React and ReactDOM
* What is difference between react.development.js and react.production.js files via CDN?
* What is async and defer? - see my Youtube video ;)

**Ques : What is Emmet?**

**Ans** :

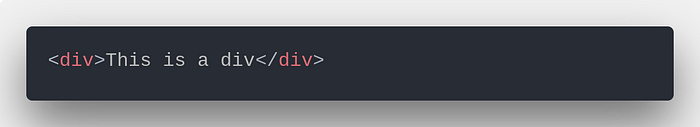
Emmet is a third party extension we have to install it. ( by default available in vs code). It allows you to type shortcuts that are then expanded into full pieces of code.

For example :

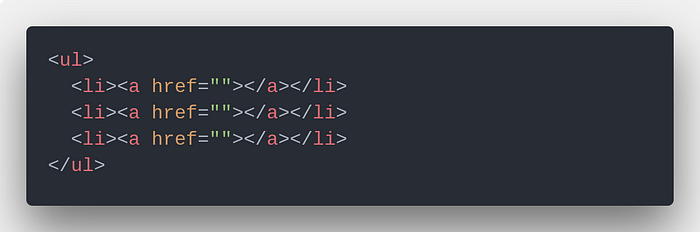
1. **Generating HTML Skeleton :** Just type**‘!`** in the editor and Hit enter button.



1. **Elements with text content inside them : just type “ div{This is a div} “**



1. **Nested elements : type “ ul>li\*3>a “**



**Ques : Difference between a Library and Framework?**

**Ans :**

|  |  |
| --- | --- |
| **Framework** | **Library** |
| It comprises of lot of APIs , compilers , support programs , libraries etc. | It is a collection of helper modules , classes , objects , functions , pre-written code , etc. |
| A framework development requires a lot of code that decrease performance and increase the load time. | Building a library requires less code , so there is better performance and fast load time. |
| Including framework smoothly into an existing project is impossible. | Libraries can be integrated easily into existing projects to add some specific functionality. |
| Example : AngularJS, Spring , NodeJS, etc. | Example : JQuery , React JS , etc. |

**Qeus : What is CDN? Why do we use it?**

## Ans :

A CDN (content delivery network), also called a *content distribution network*, is a group of geographically distributed and interconnected servers. They provide [cached](https://www.techtarget.com/searchstorage/definition/cache) internet content from a network location closest to a user to speed up its delivery.

The primary goal of a CDN is to improve web performance by reducing the time needed to send content and rich media to users.

For example :

Imagine it like a highway system for the internet, with servers spread out in different cities and towns instead of cars. When you request a website or a file from a CDN, it finds the server closest to you and delivers the content from there, instead of making you wait for it to travel from the original source, which could be much farther away.

Here's why we use CDNs:

* **Speed**: CDNs significantly improve website loading times and file download speeds. By delivering content from a server closer to you, you avoid the lag of waiting for data to travel long distances. This can be especially noticeable for large files like videos or images.
* **Reliability**: CDNs are designed to be highly available and fault-tolerant. If one server goes down, the others can pick up the slack and continue delivering content. This makes websites and online services less prone(कम होने का खतरा) to outages and disruptions.
* **Security**: Some CDNs offer additional security features, such as DDoS (distributed denial-of-service) attack protection and malware scanning. This can help to protect websites and online services from malicious attacks.
* **Global** **reach**: CDNs can help to deliver content to users all over the world, regardless of their location. This is important for businesses that want to reach a global audience.

**Ques : Why is React known as React?**

**Ans :**

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 originally developed by Facebook in 2011 for use in their own applications, and it was open-sourced in 2013. React's main purpose is to simplify the creation of complex user interfaces by breaking them down into smaller, reusable components.

React is aptly named because:

* It "reacts" quickly to changes without reloading the whole page.
* It uses the virtual DOM to efficiently update parts of a webpage.
* It is a component based approach that can reusable. This modular approach makes it easier to manage complex interfaces.

**Ques : What is crossorigin in script tag?**

**Ans :**

It stands for cross-origin resource sharing. It is a mechanism by which one webpage requests to another domain for fetching out the resource like audio, video, script, etc. from the third party server without leaking their credentials information.

**Values:**This attribute contains two values which are given below –

* **anonymous:** It has a default value. It defines a CORS request which will be sent without passing the credential information.
* **use-credentials:** A cross-origin request will be sent with credentials, cookies, and certificate.

**Example:**Below code demonstrates the use of crossorigin attribute in a <script> element.

Index.js

<!DOCTYPE html>

<html>

<head>

    <title>

        What is the purpose of crossorigin

        Attribute in HTML?

    </title>

</head>

<body style="text-align: center">

    <h1>GeeksForGeeks</h1>

    <h2>

        What is the purpose of

        crossorigin Attribute in HTML?

    </h2>

    <script id="myGeeks" type="text/javascript"

        src="my\_script.js" crossorigin="anonymous">

    </script>

    <br />

    <button>

        Submit

    </button>

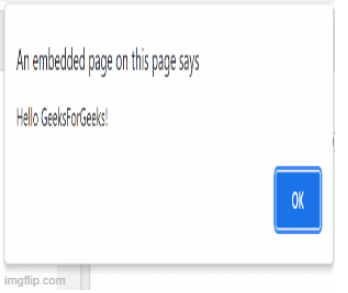
</body>

</html>

my\_script.js

alert("Hello GeeksForGeeks")a

Output :

**Ques : What is diference between React and ReactDOM**

**Ans :**

|  |  |
| --- | --- |
| **React** | **ReactDOM** |
| React provides the tools and concepts to define component-based user interfaces  **Or**  React library is responsible for creating views | ReactDOM handles the task of rendering those interfaces in a web environment.  **Or**  ReactDOM library is responsible to actually render UI in the browser |
| Whenever we use **component, classes, elements**, etc. We’re using **React.** | whenever we use methods like render() or findDOMNode() we’re using **React-DOM** |

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

**Ans :**

The React library provides two versions of its JavaScript files: react.development.js and react.production.js. These files serve different purposes and are typically used in different stages of the development and deployment process. When serving these files via a Content Delivery Network (CDN), there are a few key differences between them:

**File Size:** The react.development.js file is typically larger in size compared to react.production.js. The development version contains additional code, comments, and debugging information that aids in development and troubleshooting. On the other hand, the production version is optimized for performance and has undergone various optimizations, including minification and removal of unnecessary code, resulting in a smaller file size.

**Performance**: The production version (react.production.js) is optimized for performance and is typically used in production environments. It includes various optimizations, such as dead code elimination, to make the React library run more efficiently. These optimizations help reduce the overall bundle size, improve runtime performance, and enhance the user experience.

**Error Messages:** The development version (react.development.js) provides more detailed error messages and warnings compared to the production version. This helps developers identify and diagnose issues during the development process. The development version includes additional checks and warnings to provide useful information when something goes wrong. However, these additional checks and messages are not included in the production version for performance reasons.

When using a CDN, it's common to load the development version (react.development.js) during development and debugging stages. This allows developers to take advantage of the detailed error messages and warnings provided by the development version for easier debugging. However, in production environments, it is recommended to use the production version (react.production.js) for better performance and reduced file size.

It's worth noting that the specific file names (react.development.js and react.production.js) may vary depending on the version and release of React you are using.

**Ques : What is async and defer Attribute?**

**Ans :**

### **Async**

The async attribute tells the browser to continue parsing the HTML content while the script is being downloaded in the background. Once the script is downloaded, it’s executed immediately, potentially interrupting the HTML parsing. This is particularly useful for scripts that do not rely on any DOM elements or other scripts.

### **Defer**

The defer attribute also allows the browser to continue parsing HTML while the script is downloaded in the background, but it defers the execution of the script until after the HTML parsing is complete. This ensures that if your script needs to interact with the DOM or if it relies on other scripts that are also deferred, it will work as expected.

### 

### Comparison and Use Cases

* **Order of Execution**:
  + **async** scripts execute as soon as they are downloaded, which is not necessarily in the order in which they appear in the document.
  + **defer**, on the other hand, preserves the order and ensures that scripts execute after the document has been parsed.
* **DOM Readiness**:
  + Since **async** scripts can load at any time, they may attempt to interact with DOM elements that are not yet loaded, leading to errors.
  + **defer** scripts wait until the DOM is ready, making them a safer bet for scripts that need to manipulate the DOM.
* **Dependencies**:
  + For scripts that are dependent on other scripts or on a fully-parsed DOM, defer is the way to go. For independent scripts, such as tracking scripts or ads, async can be used without worry.



### Why Not Place <script> Inside the <head>?

Placing a <script> tag inside the <head> without async or defer will block the HTML parser while the script is being downloaded and executed. This can significantly increase the time it takes for the user to see any content on the page, especially if the script is large or the user has a slow internet connection.

Even with async and defer, best practices suggest minimizing the number of scripts loaded in the <head> to only those that are necessary. Other scripts should be loaded just before the closing </body> tag to ensure that the critical rendering path is not blocked, allowing the browser to render the page's content as quickly as possible for the user.