Episode 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 cross-origin in the script tag?
- What is the difference between React and ReactDOM
- What is difference between react.development.js and react.production.js files via CDN?
- What are async and defer? - see my Youtube video ;)

Question 1: What is Emmet?
Answer: Emmet is an extension available for IDEs like VSCode, Brackets and others which helps us to write HTML, CSS faster by providing a plethora of keyboard shortcuts to generate generic syntactical code.

Question 2: Difference between Library and Framework?
Answer:
<u>In layman terms:</u>
Library:
We use printf(), scanf() in our programs. Where are these standard functions defined? In some files that came while installing the C language IDE. These files are collectively known as Library. There are some standard libraries that always come with IDE. There are some libraries which we import from outside when we need it. These libraries are written by someone else. They can be free or paid. For example, a library to use bluetooth in our app, a library to use text to speech and vice versa in our code, a library to generate graphs and many things like that. Basically we are getting some predefined functions, we just need to call them by passing our arguments and enjoy their functionality. We decide the flow of the program. These borrowed functions are collectively called as library.

Framework:
It is a SOFTWARE/APPLICATION that helps/assists/gives facility to us to make a particular kind of application while library is NOT A SOFTWARE. Framework provides guidelines, architecture, design patterns, rules to write our different pieces of code at a particular place following all the rules. For eg. Business logic must be written at this place, look and feel related code write at this place, URL related code write at this place. Here we write the code and the framework calls our code. Code we have written, but framework decides when to call what? It's not like we write our code anywhere and the framework calls it on on his own wish and screws everything. If we build a project from scratch, most probably we will fail to follow all standards and build a scalable application. So frameworks are the way! Additionally they manage security, scalability, and portability of applications.

Standard explaination:
A library is a collection of pre-written code that provides a set of reusable functions, classes, and modules that can be used to add specific features and functionality to an application. Libraries are usually designed to be flexible and versatile, and they can be used in a wide range of applications and contexts. With a library, we have more control over our code and can use it to create your own custom solution.
A **framework**, on the other hand, is a set of pre-written code that provides a complete structure for building an application, including the overall architecture, design patterns, and programming conventions. Frameworks are typically more opinionated and prescriptive than libraries, and they often **require developers to follow specific conventions, patterns and guidelines when building applications.** With a framework, you have less control over your code and need to work within the framework's guidelines to build your application.

Question 3: What is CDN? Why do we host it?
Answer:
CDN is an acronym which stands for Content Delivery Network.
A CDN (Content Delivery Network) is a distributed network of servers located in different parts of the world that work together to deliver content to end-users with high performance, availability, and scalability.

CDNs are used to improve the speed and reliability of delivering digital content (such as images, videos, audio files, web pages, software downloads, etc.) over the internet. By caching content on servers located closer to end-users, CDNs can reduce latency and improve the load times of websites and other digital applications. Additionally, CDNs can distribute the load of delivering content across multiple servers, which can help prevent server crashes and improve the overall availability of content.

Some other benefits of using a CDN include:
Better user experience: Faster load times and reduced latency can improve user engagement, satisfaction, and retention.
Cost savings: CDNs can reduce the bandwidth costs of serving content, especially for high-traffic websites and applications.
Global reach: CDNs can help content providers deliver their content to users in different regions around the world, which can improve the user experience and increase the reach of their content.

Question 4: Why is React known as React?
Answer:
The name "React" was chosen because the library's core concept is "reacting" to changes in data by updating the user interface in response.
In traditional web development, when data changes, the entire page would have to be reloaded. However, React takes a different approach by using a virtual DOM (Document Object Model) that can update only the necessary parts of the page when data changes, resulting in faster and more efficient updates.

Question 5: What is crossorigin and type=”module” in script tag?
Answer:
<script crossorigin src= “example.com/fun.js”></script>
The crossorigin attribute in the <script> tag is used to specify whether a script should be allowed to access resources from a different domain than the one that served the script.

By default, scripts are subject to the same-origin policy, which means that they can only access resources from the same domain that served them. However, in some cases, scripts may need to access resources from a different domain, for example, when loading scripts from a content delivery network (CDN).

The crossorigin attribute can have one of three values:

- **anonymous:** Indicates that the script can be fetched cross-origin and it does not require credentials such as cookies, client-side SSL certificates or HTTP authentication to be requested by the server.
- **use-credentials:** Indicates that the script requires credentials to be fetched cross-origin. This is used in scenarios where a server requires a user to be authenticated to access a resource.
- **null or omitted:** The default value if the attribute is not provided. The script is treated as if it was fetched from the same domain and follows the same-origin policy.
- When the **crossorigin** attribute is included in a <script> tag without a value, It is equivalent to setting crossorigin="anonymous".

When the crossorigin attribute is set, the server that serves the script must include the appropriate CORS (Cross-Origin Resource Sharing) headers to indicate which domains are allowed to access the resource.

<script type= ”module” src=”./App.js”></script>
The type="module" attribute in the <script> tag is used to indicate that the script file should be treated as a JavaScript module, and it allows the use of modern ES6 module syntax.

When the type="module" attribute is set, the script file is loaded and executed as an ECMAScript module, which means that it has its own scope and does not pollute the global namespace. This is different from the traditional JavaScript file, which runs in the global scope and can overwrite or conflict with other scripts or libraries.
type="module" attribute allows web developers to use modern module syntax in their scripts, which can make code easier to write, maintain, and debug. It also provides better encapsulation and avoids global namespace pollution, which can reduce conflicts between different scripts or libraries.

Question 6: What is the difference between React and ReactDOM and why are they hosted separately?
Answer:
React and ReactDOM are two distinct libraries in the React ecosystem, with different roles and responsibilities and they are hosted separately because they serve different purposes and have different release schedules.
React is a JavaScript library for building user interfaces. It provides a declarative and component-based approach to building UIs, making it easy to create reusable and modular UI components. React is responsible for managing the state and behaviour of UI components and rendering them efficiently to the browser.

On the other hand, ReactDOM is a separate library that provides the integration between React and the browser's DOM (Document Object Model). It provides methods for rendering React components to the DOM, updating the DOM based on changes in component state, and handling events in the browser.
React can be used without ReactDOM in environments other than the web, where a different rendering target is used.

For example, in React Native, the rendering target is native mobile components instead of the web DOM. In this case, React is used without ReactDOM, and the rendering logic is handled by the React Native framework. React provides the component model and other APIs for building mobile apps, while React Native provides the renderer that translates the React components to native mobile components.

Similarly, in server-side rendering, React can be used without ReactDOM to render components on the server and generate HTML markup that can be sent to the client. Libraries like Next.js and Gatsby provide server-side rendering capabilities for React applications, where React is used to render components on the server, and the resulting HTML is sent to the client for display.

In both cases, React is used without ReactDOM, but it still provides the component model and other APIs for building user interfaces. The difference is that the rendering target is different, and the rendering logic is handled by a different renderer.

Question 7: What is difference between react.development.js and react.production.js via CDN.
Answer:
The difference between react.development.js and react.production.js via a CDN is that the former is an uncompressed version of the library designed for development and debugging purposes, while the latter is a minified and optimised version of the library intended for production use.

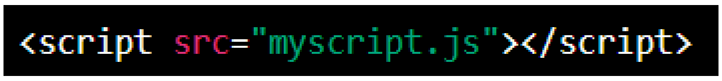
When you include react.development.js in your application via a CDN, it will be larger in size because it includes additional information such as error messages and warnings that can help developers debug their applications. This version of the library is intended to be used during development because it provides more information and feedback that can help developers identify and fix issues in their code.

On the other hand, when you include react.production.js in your application via a CDN, it will be smaller in size because it has been minified and optimised for production use. This version of the library is intended to be used in production because it is smaller in size and faster to load, which can help improve the performance of your application.

Question 8: What is the difference among normal script, async and differ?
Answer:

1. **Normal script:**

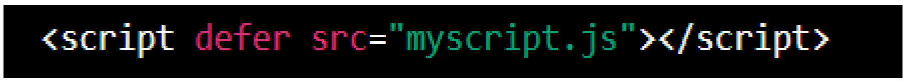
A normal script is executed as soon as it is encountered in the HTML document. This means that the script is loaded and executed before the page has finished loading. Normal scripts block the rendering of the page until they are fully loaded and executed.



```html
<script src="myscript.js"></script>
```

2. **Defer:**

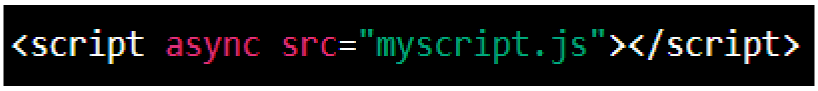
A deferred script is executed after the page has finished parsing. This means that the script is loaded in parallel with the page and executed only after the page has finished loading. Deferred scripts do not block the rendering of the page.



```html
<script defer src="myscript.js"></script>
```

3. **async:**

An async script is also executed after the page has finished parsing. This means that the script is loaded in parallel with the page and executed as soon as it is available. Async scripts do not block the rendering of the page, but they may execute before other scripts have finished loading.



```html
<script async src="myscript.js"></script>
```