Namaste React: Assignment1

1. What is Emmet?

Ans: Emmet is a toolkit for web developers that enhances HTML and CSS coding efficiency. It allows you to write abbreviated syntax, which expands into full HTML or CSS code. This can significantly speed up the process of writing repetitive code.

Examples:

* In VS code if you write ‘ul>li\*3’ and press enter it will become:

<ul>

<li></li>

<li></li>

<li></li>

</ul>

* In VS code is you write h1\*3 and press enter it will become:

<h1></h1>

<h1></h1>

<h1></h1>

1. What is the difference between Library and framework?

Ans: Library is a collection of pre-written code that you can use in your program. It provides specific functionalities that you call when needed, allowing for more control over the flow of your application.

Example: React.

In React, Index.html page can be very large, it can be anything, we can use React anywhere in that html page. We can use React at any small part of that large Index.html page, (that’s the reason we call React as a library) depending upon how we render the Index.html page, content will be replaced by the React content or just append the React content to the Index.html page content.

So here we are having full control over the flow of our application.

Another way to call React as a library:

React is just a library, why it is a library means you can import it like a JavaScript file into your code and it gives you the superpowers. You can use React in your existing web application projects no matter if it is using jQuery or something else.

React is considered as a library because it primarily focuses on building user interfaces, React is more flexible, allowing developers to choose additional libraries and tools for features like routing.

Example 2: Using a math library to perform calculations. You have a toolbox (math) and you choose which tools (functions) to use when needed.

Import math: After importing its up to us what functions from math we can use.

Framework is a more comprehensive set of tools, guidelines and conventions that dictate the overall structure and flow of your application. The framework manages the flow of control.

Angular is considered as a framework because it offers a more comprehensive solution for building web applications. Angular includes features like a built-in router, dependency injection, forms handling and more making it more prescriptive in how you structure and build your application.

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

Ans: CDN stands for content delivery network or content distribution network. A content delivery network (CDN) is a network of interconnected servers that speeds up webpage loading for data-heavy applications.

When a user visits a website, data from that website's server has to travel across the internet to reach the user's computer. If the user is located far from that server, it will take a long time to load a large file, such as a video or website image. Instead, the website content is stored on CDN servers geographically closer to the users and reaches their computers much faster.

1. Why is React known as React?

Ans: In normal English React means to respond or behave in response to something. In the React programming whenever there is a change in the state then the React library will re-render the component by efficiently updating (updating only the required parts of the) the DOM, so that’s the reason React is known as React.

1. What is cross-origin in script tag?

Ans: When you make a request from one origin to another origin the browser will block the request due to security restrictions. To avoid security restrictions, we use crossorigin in the script tag.

1. What is the difference between React and ReactDOM?

Ans: React is a library for building components and managing the application state, while ReactDOM is the package responsible for rendering those React components in the browser’s DOM.

Example:

Creating a React element is the job of React:

const heading= React.createElement(“h1”, {id:”root”}, “This is h1 tag”)

Creating the root and rendering something inside it is the job ReactDOM:

const root = ReactDOM.createRoot(document.getElementById(“root”));

root.render(heading);

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

Ans: The difference between ‘react.developement.js’ and ‘react.production.js’ lies in the level of optimization and additional features for development and production environments.

* React.development.js: This version is intended for development. It includes extra warnings and debugging information, making it easier to identify and fix issues during the development. Its larger in file size due to the inclusion of these development tools (extra warnings and debugging information) but its valuable for a smoother debugging experience.

Examples:

1. Key warnings: React warns if it detects that elements in a list are missing unique keys. Keys are important for React to efficiently update the DOM .
2. React Strict Mode Warnings: If you are using React strict mode in development, additional warnings and information about potentially unsafe practices are logged to the console.
3. Error Stack Traces: When an error occurs, React provides detailed stack traces that point to the exact location in your code where the issue occurred. This aids developers in quickly identifying the source of errors.

* React.production.js: This version is optimized for production. It excludes the development-specific warnings and debugging information, resulting in a smaller size. The minimized file is meant to be used in production environments where the emphasis is on performance rather than developer convenience.

When including React via CDN in your web application, you typically use the production version (react.production.js) for better performance. During development its common to use the development version (react.development.js) to take advantage of development specific warnings and debugging information. But we should switch to production version when deploying our application in any environment other than development for optimal performance.

1. What are async and defer?

Ans: async and defer are the attributes of the script tag.

In script tag with src attribute we will load the external files, so with async and defer is related to how those files are loaded and the order of execution.

Example 1: Without async and defer

Index.html page:

<script src=”script1.js”> console.log(“Test”) <script>

<script src=”script2.js”> console.log(“Test”) <script>

Some HTML content.

Here the scripts are loaded synchronously, that means first script1 will be loaded and executed then script2 will be loaded and executed and then only the HTML content will be parsed.

Example 2: With sync:

Index.html page:

<script async src=”script1.js”>console.log(“Test”)</script>

<script async src=”script2.js”>console.log(“Test”)</script>

Some HTML content.

Here the scripts are loaded asynchronously, that means first script1 will start loading and without any waiting it goes to next line and start loading the script2 and parse the HTML content, once if any script is loaded successfully that script will be executed by pausing the HTML parsing. Here the order of script executions does not be guaranteed. Whichever the script is downloaded first, that script will be executed first. No time waste will be there in this approach.

Example 3: With Defer:

<script defer src=”Script1.js”> console.log(“Test”) </script>

<script defer src=”Script2.js”> console.log(“Test”) </script>

Some HTML Content

Here scripts will also be downloaded asynchronously, but they will only be executed once the entire HTML page is parsed. Here also no time waste will be there. Here order of the scripts will be guaranteed. Script1 will be executed first then only script2 will be executed.

When to use what?

* When the scripts are dependent on each other then it is better to use defer, because it follows the order of execution, and our code will not be broken. Example: script 2 is dependent on scrip1, then we can write the code like this:

<script defer src=”Script1.js”> console.log(“Test”) </script>

<script defer src=”Script2.js”> console.log(“Test”) </script>

So, our code will not be broken.

In this case if we use the async then our code may break, because consider script2 is downloaded before script1, since script2 is downloaded first it will start executing the script2 and it will break the code because it is dependent on script1.

* If the scripts are not dependent on each other then we can use the async.

Note:

In an arrow function body if there is only a single statement, then no need of using the return statement, that single statement will be automatically returned.

Example:

const Test = (a,b) => a+b

If you want to use the return then you should use the {}, both {} and return must be used together.

Example:

const Test1 = (a,b) => {return a+b}

For multiple statements you must use the {} and return statement

Example:

const Test2 = (a,b) =>{

const Result1 = a+b;

const Result2 = a\*b;

return Result1+Result2;

}

If we don’t use the return statement then “undefined” will be returned.

const Test3 = (a,b) =>{

const Result1= a+b;

const Result2 = a\*b;

Result1+Result2;

}

Here Result1+Result2 will be evaluated but here we are not using the “return” keyword so undefined will be returned.