Namaste React

Assignment - 01

Q1. What is Emmet?

Emmet is a plugin for many popular text editors which greatly improves HTML and CSS workflow by using short expressions.

General Abbreviation:- tagName[series of expressions]
Few examples:

1. Generating HTML Skeleton: Type '!' to get the HTML skeleton.
2. Elements with text content inside them: div{This is the content that goes inside the div}
3. Nested elements:
    a. *ul>li* (using the '>' operator we can create nested elements)
    b. *ul>li*3>a* (Repetition can be done using '*' operator)
    c. *ul>li{$}*3* (Using '$' operator for printing numbers inside each list item)
    d. *ul>li{$@-}*5* (You can also use $ multiple times so the number is padded with 0. You can set base number with '@N' and direction with '@-')
    e. *ul>li{$@10}*5* (it prints number from 10 to 14 in increasing order)
    f. *ul>li{$@-10}*5* (using point 'd' and 'e' together)

4. What about classes and Id:
    a. *div#main.container.responsive* –
    <div id="main" class="container responsive"></div>
    We can only have a single ID, however multiple classes are possible. If there are multiple id's then the 2nd one will override the 1st one.

5. Custom attributes:
    If we want a div with custom data-property then do the following:
    div[data-name=logo] –
    <div data-name="logo"></div>

6. Generating Siblings:
    To generate siblings use "+" operator – *header+div+footer*
    <header></header>
    <div></div>
    <footer></footer>

7. Grouping:
    You can use '()' operator to group complex abbreviations.
    *div>(a>p>span+span)*3*

8. Lorem Ipsum
    Just typing 'lorem' will generate the text. If we want to repeat it multiple times then we can use 'lorem*5'
9. CSS

Margin of 10 on all sides :
m10-10-10-10 – This expands as *margin: 10px 10px 10px 10px*
Same thing for padding:
p10-10-10-10 – This expands as *padding: 10px 10px 10px 10px*

https://medium.com/@kartik2406/web-development-with-vs-code-part-1-emmet-6af80f0f630c
https://code.visualstudio.com/docs/editor/emmet

Q2. What is the difference between a Library and Framework?

When an application code uses a library, the developer writing the code is in charge of the application flow. This means the developer decides when to call the library. However, when we use a framework, the framework decides when to call the library. This shift in control of calling the library from the application code to the framework is an inversion of control.

A framework enables a coding environment that contains low-level libraries to address conventional coding issues. The objective of a framework is to deliver faster development of an application. This includes everything we need to build large-scale applications, such as templates based on best practices. Let's consider the inner workings of a framework. If we're using a graphical user interface (GUI) framework, it calls the code through event handlers. If we're using a web framework, it calls the code through a request-response model. For a web application, JavaScript frameworks provide a skeleton with the tools for fast development. Internally, such a framework contains a large collection of libraries that provides the developer with inbuilt functionalities, which helps to develop an application without extensive coding knowledge.
ADVANTAGES of a FRAMEWORK:

- A change in one part of the application doesn't affect the entire application.
- Provides caching and optimized processes for the network traffic
- Enables faster methods for web development with less code
- Supports cross-platform application development
- Provides a superior user experience by creating rich and dynamic content
- Some JavaScript frameworks, such as Angular, are based on MVC, hence the use of data binding.

LIMITATIONS of a FRAMEWORK:

- The ready-to-use features of a framework prevent programmers from gaining an in-depth understanding of the programming language.
- Options to tweak functionalities are limited.
- Sometimes application development is complex using a framework.
- We need to choose the right framework for the scale of the application, or else performance and user experience may be adversely impacted.
- Clear separation of business logic from the presentation layer in MVC is sometimes difficult.
- We have to be up-to-date with new/deprecated features in every version.

issues associated with web development frameworks are:

- Browser version dependencies
- JavaScript frameworks can only run with a JavaScript-enabled browsing environment.
- Chance of a security breach if the prescribed framework guideline isn't followed during development

Its main USES include:

- Web Development – This uses different languages, such as PHP (CodeIgniter, Laravel), Python (Django), and JavaScript (Angular).
- Artificial Intelligence – Popular frameworks include Apache Spark, PyTorch, and Tensorflow.
- Mobile App Development – Some of the popular frameworks are Native Script, React Native, and Flutter.

A library is a collection of reusable, compiled, and tested code that can facilitate the automation or augmentation of application functionalities. It's designed to support both the code developer and code compiler during the build process and the running of the application. A library implements many functions, variables, and parameters.

ADVANTAGES of a LIBRARY:

- Improves performance of a program by the selective inclusion of a library by the compiler during run time
- Provides reusable functions that can be referred to within the code without defining them explicitly
- Eliminates the need for writing code for complex functions
- Prevents us from having to write code to solve the same problem over and over again
- Focuses only on the feature that the library implements, without worrying about global state management, like HTTP and routing
- Provides us with the opportunity to pick libraries that we want
- Reduces application development cost
- Encourages programmers to focus on configurable and reusable library development
- Provides pre-tested code for multiple environments and use cases

LIMITATIONS of a LIBRARY:

- Using a library means that our code is tied to that library.
- To change libraries, our code might have to undergo changes to use the new library.
- Using a library in an unsupported environment needs a wrapper, resulting in a performance impact on the application.
- Lack of support by the developer of a library may result in incompatibility issues with the new version of the application.
- Using several libraries might impact the performance of an application adversely because of dependency conflicts.
- Sometimes libraries are vulnerable to malicious attacks.

USE of a LIBRARY:

We tell libraries what to do, so they have a vast area of use. Since software development started involving complex functionalities, the need for reusable libraries increased. JQuery is one such example because it makes event handling, animation, and Ajax simpler, and works across a large number of browsers.

https://www.baeldung.com/cs/framework-vs-library
http://www.differencebetween.net/technology/difference-between-library-and-framework/

Q3. What is CDN (Content Delivery Network)? Why do we use it?

A CDN, or content delivery network, is a network or collection of servers in locations all over the world. Also known as a content distribution network, a CDN can refer to many types of content delivery services, such as load balancing and video streaming.

A CDN's network of servers allows companies to deliver content from their websites and mobile applications to people more quickly and efficiently, based on their geographic location. In short, a CDN moves data and applications closer to the end user — increasing speed, enhancing security, and improving the user experience.

What are the benefits of a CDN?

There are many benefits of a content delivery network, from improved user experience to increased security to lower costs.

Reduced page load time CDNs eliminate the need for data to travel over long distances because they deliver content from servers that are close to the end user. As a result, CDNs dramatically decrease the amount of time it takes to load a webpage, including those with high-bandwidth, media-rich content. For many businesses, this improvement in the user experience translates into better brand reputation and more efficient sales.

Improved availability No more frustrating error messages; CDNs allow websites and apps to be "always on" for the end user. If an origin server goes down, the CDN can continue serving whatever content was last in cache to users from POPs, points of presence, that are geographically and strategically distributed for peak performance.

Increased scalability CDNs allow businesses to scale on demand. While web traffic is typically consistent for most of the year, events like Black Friday sales or breaking news events can cause traffic to surge, sometimes unexpectedly. In an effort to preserve the user experience, companies traditionally have had to buy or rent enough servers to account for the peak times. This means, for the rest of the year, they may be paying for wasted storage.

In contrast, CDNs allow companies to normalize their server spend and buy a more reasonable amount of space. When a traffic surge does occur, companies can send it to be served from a distributed POP instead of from the origin. This scalability happens on demand; as soon as a company needs more capacity, they get it. In addition, load balancing allows servers to distribute the requests over the network for more optimal routing decisions and greater resilience.

Increased security Content delivery networks provide extra layers of security. First, CDNs are more resistant to certain types of cybersecurity threats because traffic is routed through POPs. For example, CDNs protect your website against Distributed Denial-of-Service (DDoS) attacks, which overload origin servers with fake traffic to slow down or even crash websites. A CDN's large, high-bandwidth, globally distributed network is able to absorb that traffic and prevent it from hitting the customer's origin server.

Second, CDNs assist with data encryption. Because data moving across the internet is vulnerable, it must be encrypted using protocols such as Transport Layer Security (TLS) and Secure Socket Layer

(SSL) so that only the intended recipient can decode and read the information. CDNs can help protect a site by providing TLS and/or SSL certificates that ensure a high standard of authentication and encryption.

In addition, CDNs can help protect your websites and apps through a Web Application Firewall (WAF). WAFs offered by CDNs will analyze and channel traffic to and from a website, blocking application layer (Layer 7) threats. They examine every HTTP request and act as a shield to block suspicious traffic, only allowing secure traffic through. This is seamless to the user, and prevents threats such cross-site scripting (XSS) and SQL injection attacks. CDNs can also help protect against bots, which are software programs that perform automated tasks and can be used for malicious attacks. CDNs can use detection technology to quickly identify and neutralize bot threats.

Reduced bandwidth costs Lastly, CDNs can help save companies money on bandwidth. By rerouting traffic from the origin server to the CDN's servers, CDNs reduce spend on origin infrastructure and egress costs. If content is held in cache by a CDN, there are fewer reasons to travel back to the origin.

Who needs a CDN?

Businesses with an online presence stand to benefit from a content delivery network, including ecommerce platforms, digital publishers, social media sites, and entertainment websites:

- A CDN can benefit large ecommerce platforms by handling heavy traffic, regulating seasonal and unexpected traffic spikes, and increasing security around transactions. These platforms span a range of industries, including retail and hospitality.
- Similarly, digital publishers with large global readerships can use CDNs to handle traffic, page downloads, transactions, and other demands.
- A CDN is also ideal for entertainment websites (like streaming sites that deliver real-time, high-definition content) where users have come to expect a predictable, high-quality experience.
- Financial services providers enhance their users' experiences by using CDNs to cache APIs so they can serve highly dynamic content, like stock prices. -
- Social media sites, which experience high traffic volumes and feature rich multimedia content, can use a CDN to enhance and regulate the user experience.
- High tech companies use CDNs to deliver insights and analysis they can act on to constantly improve the user experience — a CDN that provides real-time analytics and streaming logs provides actionable insights they can use to differentiate themselves.

These are just a few examples of businesses that benefit from CDNs. Your business might need a CDN if your site experiences a lot of traffic or uses a lot of bandwidth — or both.

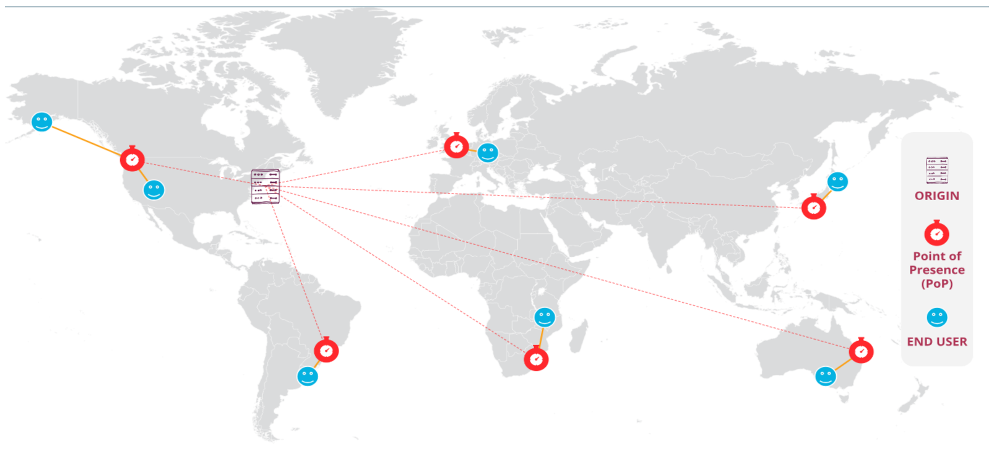
Example use case: A CDN in action

Let's look at a use-case example to understand how a CDN provides a quicker and more efficient way of serving content to users.

Take an end user visiting a favorite news website. Once they type in the URL to initiate communication between the browser and the news site server where the webpage is hosted (the origin), the back-and-forth communication between the user's browser and the origin goes through a number of steps: first relaying the DNS lookup, then routing, then a TCP and TLS handshake, and finally the HTML transmission, along with various files like CSS, JavaScript, and videos or images on the news site.

Now, let's say the user is in San Francisco, and the news site server is in New York. Remember that all the communication between the user's browser and the news site server happens over real wires and cables. So, the physical distance between San Francisco and New York adds time, or "latency," to the browsing experience. Imagine if our user was in Australia and the communication had to travel back and forth between Australia and New York! As you might predict, that distance would dramatically increase latency, degrading the user experience.

In contrast, let's picture this scenario again – this time, using a CDN.



We know that the closer the user is to the news site server, the faster the experience will be. CDNs put servers physically closer to end users, thereby speeding up load times.

With this knowledge, the news site has opted to cache—meaning, temporarily store — its content on CDNs. Now, instead of going back and forth between New York and San Francisco, the user's browser can communicate with a server much closer to home. And for the user over in Australia, their browser can connect to a POP in that part of the world, for example, Sydney.

And because cached content is only stored temporarily, CDNs also purge, or remove and update, content constantly. Companies don't have to worry about serving their users outdated content; the most up-to-date content is delivered automatically. As a result, not only will our user's article load significantly faster, but they'll also see the latest headlines as the day unfolds. A CDN has dramatically enhanced the user experience — and, we might presume, the loyalty to visit that news site again and again.

Get started with a CDN

This is just one of countless examples of how a content delivery network can add value — optimizing web performance and enhancing a user's online experience, resulting in enhanced brand reputation and loyalty. Indeed, by increasing performance, availability, scalability, and security, all while reducing costs, CDNs are a win-win for many types of businesses.

https://www.fastly.com/learning/what-is-a-cdn
https://codedamn.com/news/reactjs/cdn-in-react-js-framework

Q4. Why is React known as React?

React is a JavaScript library that helps developers to build user interfaces – the things you interact with on websites. It has become popular because of its simplicity and flexibility and is used by companies like Facebook, Instagram and Airbnb.

First of all, since React is a JavaScript library, we need to have some basic knowledge of what JavaScript (and a library) is. If HTML structures your website and CSS styles it, then JavaScript makes it dynamic and is probably what makes things move, refresh or change on the site you're visiting.
To avoid writing time consuming repetitive functions in JavaScript you can use different libraries, i.e. pre-written JavaScript code. React is one of many libraries and it's completely frontend and specializes in things that the user interacts with when they're using a website. This could be buttons, search bars and menus.
"And it's called React because it reacts. It was developed by Facebook (a site that CONSTANTLY updates their data) to improve the user interface development and more effectively change (REACT to) what the user sees when they're doing things like mouse clicking, submitting and typing."

Why you should use React

So what makes React so popular?

- Well, React is declarative which means that you tell it WHAT you want (written in React language) and then it solves the HOW and builds the user interfaces in the web browser.
- In order to simplify and not focus on the whole website at the same time, you break it down into smaller components that you can reuse wherever you want to.
- Instead of having a complete website re-render every time something changes, React can update only the things that are different than they were before an event happened. This means that if you, for example, change your profile picture, the image is the only thing that is re-rendered – nothing else on the site is updated and replaced with data that is basically the same as it was before. Because that would be kind of unnecessary, right?

How does React work?

React Components

When you're using React you start by dividing your site into smaller components. For a news site, this could be a header, a search bar and an article with title, image and text. You build your different components separately and then assemble them in a main file.

JSX

As previously mentioned, HTML is what structures your website (the skeleton), and it's written in its own file. With React, you can use something called JSX which allows you to bring HTML into your JavaScript files and build components that consist of both.
JSX accepts something called "props" (properties) which make it possible to insert custom data into the components. In our newspaper example, this means we can use the article component as a kind of template but then add unique props like title, image and text to create different articles for our news.

Virtual DOM

There is something called the DOM, which is the programming interface for HTML documents. It represents the data on a web page and makes it possible to change what we see on it. When you're using JSX, React creates something called a Virtual DOM which is a copy of the actual DOM. It compares the data on both places and only replaces the thing in the DOM that is different, like a comment being added, a button being pressed etc. It may not sound like a big deal, but when you have a lot of things constantly changing on a site, this selective updating results in way better loading time and computing power.

https://www.technigo.io/explained/what-is-react
https://medium.com/@dhawalpandya/why-is-react-called-react-92f83b10aeac

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

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 (e.g. 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.
Tip: 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).
Tip: Also look at the integrity attribute.

The integrity attribute allows a browser to check the fetched script to ensure that the code is never loaded if the source has been manipulated.
Subresource Integrity (SRI) is a W3C specification that allows web developers to ensure that resources hosted on third-party servers have not been altered. Use of SRI is recommended!
When using SRI, the webpage holds the hash and the server holds the file (the .js file in this case). The browser downloads the file, then checks it, to make sure that it is a match with the hash in the integrity attribute. If it matches, the file is used, and if not, the file is blocked.

https://www.w3schools.com/tags/att_script_crossorigin.asp#:~:text=The%20crossorigin%20attribute%20sets%20the,or%20scripts)%20from%20another%20domain.
https://www.w3schools.com/tags/att_script_integrity.asp

Q6. What is the difference between React and ReactDOM?

https://www.geeksforgeeks.org/how-react-and-reactdom-works/
https://stackoverflow.com/questions/34114350/react-vs-reactdom

The *react* package holds the react source for components, state, props and all the code that is react.
The *react-dom* package as the name implies is the glue between React and the DOM. Often, you will only use it for one single thing: mounting your application to the index.html file with ReactDOM.render().

The react package contains React.createElement(), React.Component, React.Children, and other helpers related to elements and component classes. You can think of these as the isomorphic or universal helpers that you need to build components. The react-dom package contains ReactDOM.render(), and in react-dom/server we have server-side rendering support with ReactDOMServer.renderToString() and ReactDOMServer.renderToStaticMarkup().

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

The production version will be optimized in a few ways--smaller file size by minimizing variable names and removing white space, etc--whereas the development version will remain readable, might include source maps, etc., making it better for debugging and development usage. Functionally they're the same.
In development mode, we can enable and utilize React developer tools, the devtools profiler, and the debugging environment attached to the source code. We can utilize various functionalities, such as Hot Module replacement and diagnostics, so that the development environment will help debug code.
In production mode, compression and minification of Javascript and other resources reduce the size of the code, which is not the case when it comes to development mode. Performance will be much faster in production mode when compared to development mode.

Q8. What are async and defer?

https://codedamn.com/news/javascript/async-and-defer-in-script-tag
https://javascript.info/script-async-defer

Async allows your script to run as soon as it's loaded, without blocking other elements on the page. Defer means your script will only execute after the page has finished loading.

Async in script tag in JavaScript is a way to load scripts asynchronously. That means, if a script is async, it will be loaded independently of other scripts on the page, and will not block the page from loading.
If you have a page with several external scripts, loading them all asynchronously can speed up the page load time, because the browser can download and execute them in parallel.
To use async, simply add the async attribute to your script tag:
<script async src="script.js"></script>

By using the defer attribute in HTML, the browser will load the script only after parsing (loading) the page. This can be helpful if you have a script that is dependent on other scripts, or if you want to improve the loading time of your page by loading scripts after the initial page load.
To use defer, simply add the defer attribute to your script tag:
<script defer src="script.js"></script>

The async and defer attributes both allow the browser to continue parsing the HTML document while JavaScript files are being downloaded, but they differ in when those files are executed.
Async downloads and executes JavaScript as soon as it’s available, while defer attribute waits until the HTML document has been parsed before downloading and executing any external scripts.

In most cases, it doesn’t matter which attribute you use – both will improve performance by allowing the browser to continue parsing while waiting for JavaScript to download. However, there are some situations where one attribute may be preferable to the other.