**Angular**

**What is UI and UX? What is client side and server side rendering**

**UI (User Interface)** and **UX (User Experience)** are two crucial aspects of web and application design.

1. **UI**: User Interface is the series of screens, pages, buttons, icons, and all visual elements that enable a person to interact with a product or service. It is all about the look and feel of the application, how it is presented to the user.
2. **UX**: User Experience is the overall experience that a person has while interacting with a product or service. It encompasses all aspects of the end-user's interaction with the company, its services, and its products. UX is not only about ease of use, but also about emotions, perceptions, preferences, ease of access, usability, and performance of the application.

Client-side rendering and server-side rendering are two strategies in web development that determine where and how your webpage is assembled into a format that a user's browser can interpret and display.

1. **Client-side Rendering (CSR)**: In client-side rendering, your server sends a response with a relatively bare-bones HTML document and links to the JavaScript, CSS, and other assets required to fully render the page. Most of the work happens in the client's browser. The browser downloads the JavaScript and executes it to fill in the rest of the HTML document with actual content. This is the typical approach used by single-page applications (SPAs) built with frameworks and libraries such as React, Vue, and Angular.

Pros of CSR:

* + Once the initial page is loaded, navigating to different parts of the application feels faster and smoother without a full page reload.
  + The server can be simplified and does less work.
  + Can provide a more interactive and app-like user experience.

Cons of CSR:

* + The initial page load might be slower because the entire application (or a large part of it) needs to be downloaded.
  + Search Engine Optimization (SEO) can be more challenging because some search engines' web crawlers might not execute JavaScript or do it less efficiently, leading to poorly indexed sites.

1. **Server-side Rendering (SSR)**: In server-side rendering, the server's response to the browser includes a fully rendered HTML document populated with the data required, meaning the browser can start displaying the page to the user very quickly. This is the traditional way of serving web pages, and is common in multi-page applications.

Pros of SSR:

* + Initial page loads are usually faster because the browser can display the page as soon as it receives the HTML from the server.
  + It's generally better for SEO because web crawlers can easily crawl the site for indexing.

Cons of SSR:

* + Tends to require more server resources and puts more load on the server because the server must render a new HTML document for each request.
  + Page updates or navigation can cause a full page reload, which might not feel as smooth as with CSR.

There's also a method that combines both CSR and SSR, where the server sends a ready-assembled webpage first (for quick initial loading and better SEO), and then the browser takes over the job of assembling new pages (for smoother navigation). This is often used by websites built with modern tools like React, Angular, and Vue to try and get the best of both worlds.

Now, **client-side rendering** is a strategy in web development where the browser downloads the JavaScript, HTML, and CSS from the server and renders the page directly on the user's device. It is primarily used in Single Page Applications (SPAs) built with frameworks and libraries like Angular, React, Vue.js, etc.

Benefits of client-side rendering include:

1. **Interactivity**: Once the initial page load is complete, changes to the page (due to user actions or other events) can be rendered immediately without the need for a full page reload. This leads to a smoother and more interactive user experience.
2. **Reduced server load**: Since the client (browser) is doing most of the rendering work, the server can focus more on processing API requests and delivering static files. This can potentially reduce the load on the server and allow it to handle more concurrent users.
3. **Rich UI/UX**: Frameworks that use client-side rendering often provide powerful tools and abstractions for building rich, complex user interfaces.
4. **JavaScript** is a programming language that adds interactivity to websites. It has evolved to be used in many other areas like server-side programming, mobile, and desktop applications.
5. JavaScript is still a primary technology in web development. However, managing complexity and performance issues in large applications with pure JavaScript can be challenging.
6. Due to these challenges, developers often use **JavaScript frameworks and libraries**, which provide abstractions and patterns that make it easier to build complex applications. These include:
   * **React**: A library for building user interfaces, developed by Facebook. It uses a component-based architecture and a virtual DOM for efficient rendering.
   * **Angular**: A comprehensive framework developed by Google. Angular uses TypeScript (which adds static typing to JavaScript) and has many built-in features.
   * **Vue.js**: A lightweight and flexible framework. Like React and Angular, it uses a component-based architecture.
7. **TypeScript** is a statically-typed superset of JavaScript. It adds static typing to JavaScript, improving developer productivity and code quality.
8. **Svelte** is a newer JavaScript framework that compiles code to efficient, imperative code that updates the DOM directly.
9. **Node.js** allows JavaScript to be used on the server side. It's commonly used for back-end web development.

In conclusion, while pure JavaScript might not be used as often in large, complex applications, JavaScript, along with its various frameworks, libraries, and related languages, remains a core technology of the web.

Top of Form

Bottom of Form

The main technologies for client-side rendering are JavaScript frameworks and libraries, including:

1. **AngularJS / Angular**: A complete front-end framework developed by Google. Angular (2 and above) uses TypeScript and offers a lot of built-in functionality like HTTP services, form handling, and more.
2. **React**: A JavaScript library developed by Facebook, used for building user interfaces, especially those with complex stateful components.
3. **Vue.js**: A lightweight and flexible JavaScript framework that's seen a rise in popularity due to its simplicity and performance.
4. **Ember.js**: A robust framework for ambitious web applications. It provides a complete solution that includes data management and an application build pipeline.
5. **Backbone.js**: A lightweight MVC (Model-View-Controller) library that adds structure to JavaScript applications.
6. **Svelte**: A newer JavaScript framework that compiles your code to small, efficient imperative code that directly updates the DOM, resulting in fast initial load times and smooth updates.

**Component-based design** is a method of software development that involves building applications by combining reusable and independent components. Each component is a self-contained entity that encapsulates specific functionality or a group of functionalities.

Components can be thought of as building blocks of a system. Just like Lego blocks, these components can be individually created, tested, and then assembled together to build a complete application.

Advantages of Component-based design:

1. **Reusability**: Components are designed to be reused, which can lead to significant time savings as the same component can be used across different parts of an application or even across different projects.
2. **Separation of Concerns**: Each component is independent and encapsulates specific functionality, leading to a system design where each component only cares about its own functionality. This makes the system easier to understand, maintain, and update.
3. **Modularity**: Because an application is built up from individual components, it's easy to modify or replace one component without affecting the rest of the system. This makes the system more flexible and adaptable to changes.
4. **Ease of Testing**: Components can be tested independently, which simplifies the testing process and helps to ensure the reliability of the system.
5. **Efficiency in Development**: Since components can be developed in parallel, this can speed up the overall development process.
6. **Consistency**: Using components can lead to a more consistent user interface and user experience, as the same components are used throughout an application.

Component-based design has become a fundamental principle in many modern software development frameworks and libraries, particularly those used for web development, like React, Angular, and Vue.js. These frameworks provide tools and abstractions that make it easier to build, reuse, and manage components.

Software used for Angular

1. Node.js
2. Vs code (visual studio code editor) or any editor of your choice like atom, sublime etc
3. Any RDBMS or no sql databases like mysql or mongo db