REACTJS NOTES

Server-side rendering and Client-side rendering

ASHRAYA KK

Server-side rendering (SSR) & Client-side rendering (CSR)

In modern web development, rendering techniques play a crucial role in delivering webpages to users.

2 popular approaches for rendering web applications are :—

1. Server-side rendering (SSR)
2. Client-side rendering (CSR)

SERVER-SIDE RENDERING

With old SSR solutions, you built a webpage (eg:- with PHP) - the server compiled everything, included the data, and delivered a fully populated HTML page to the client. It was fast and effective.

But... everytime you navigate to another route, you have to make server to do all these work all over again.

i.e, Get PHP file, compile it, deliver the HTML with all CSS and JS delaying the pages to load to a few hundred ms.

React popularized this problem with an easy-to-use solution: renderToString() method.

* renderToString

```
const html = renderToString(reactNode)
```

→ On the server, call 'renderToString' to render your app to HTML.

→ This method is used to convert a React component hierarchy into its corresponding HTML markup as a string.

→ When using SSR with React, this method is called on server-side to generate the initial HTML of React components.

This process involves following steps :

1. Server-Side Setup :

On the server, a node.js environment is typically set up, and a server framework like express is used to handle incoming requests.

2. Request Handling :

When a request is received, server identifies the react components that need to be rendered for the requested route or page.

3. Component Rendering :

The server-side code executes the 'renderToString()' method on the ~~react~~ root React component, passing relevant props & data.

4. Component Rendering Process :

'renderToString()' method traverses the React component hierarchy & generate HTML markup for each component.

5. Data Fetching and Integration :

If data needs to be fetched from external sources (API or DB), it is done before or during rendering process. The fetched data is passed as props to the React components to ensure that they have the necessary data for rendering.

6. Generating HTML :

As the 'renderToString()' method traverses the component hierarchy, it builds the HTML markup as a string.

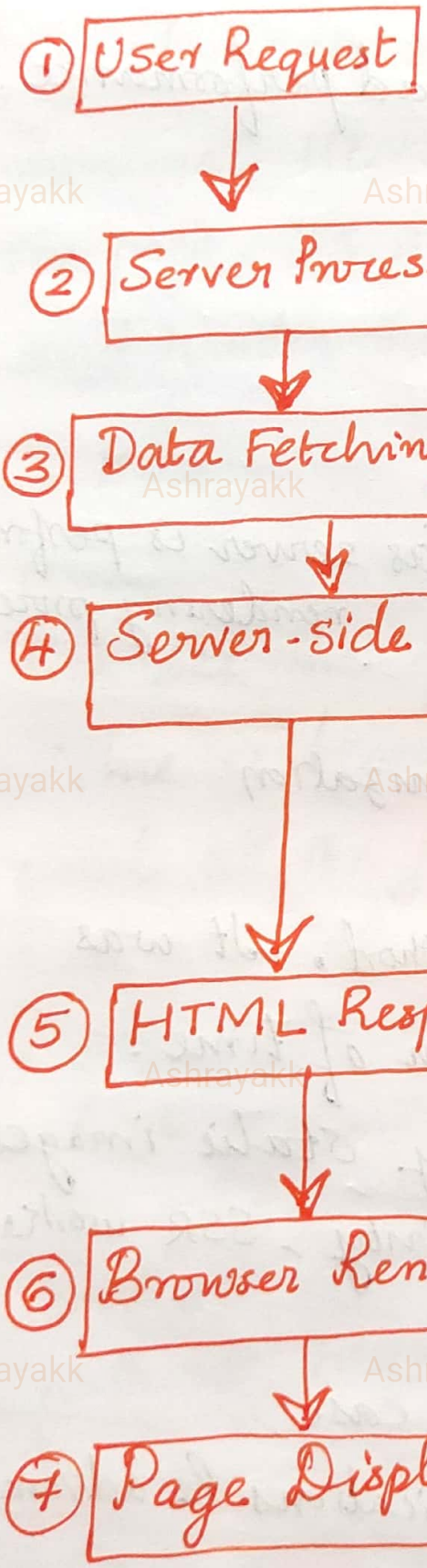
7. HTML Response :

Once the 'renderToString()' method completes, the server sends the fully rendered HTML string as the response to client's browser.

8. Client Hydration :

On the client-side, after the initial HTML is received, React takes over and rehydrates the HTML, setup component state, allow interactive elements to function.

SERVER-SIDE RENDERING PROCESS



1. User Request - User's browser sends a request to the server for a specific webpage.
2. Server Processing - Server receives the request and starts processing it.
3. Data Fetching - Server fetches data from API or DB.
4. Server-side rendering - server generates complete HTML for the requested page, including embedding the fetched data into HTML template.
5. HTML Response - Server sends fully rendered HTML response back to user's browser.
6. Browser Rendering - browser receives HTML and starts rendering page
7. Page Display - browser renders HTML, executes any embedded script & display fully rendered page to user.

Advantages - SSR

1) Improved Initial page load performance.
2) Enhanced SEO
3) progressive enhancement.

Disadvantages - SSR

1) Increased Server Load. (as server is performing rendering process)
2) Slower time to interactive
3) Limited client-side optimization

SSR is the conventional method. It was the only way since the dawn of time. Since most webpages display static images and text, with little interactivity, SSR worked great at old time.

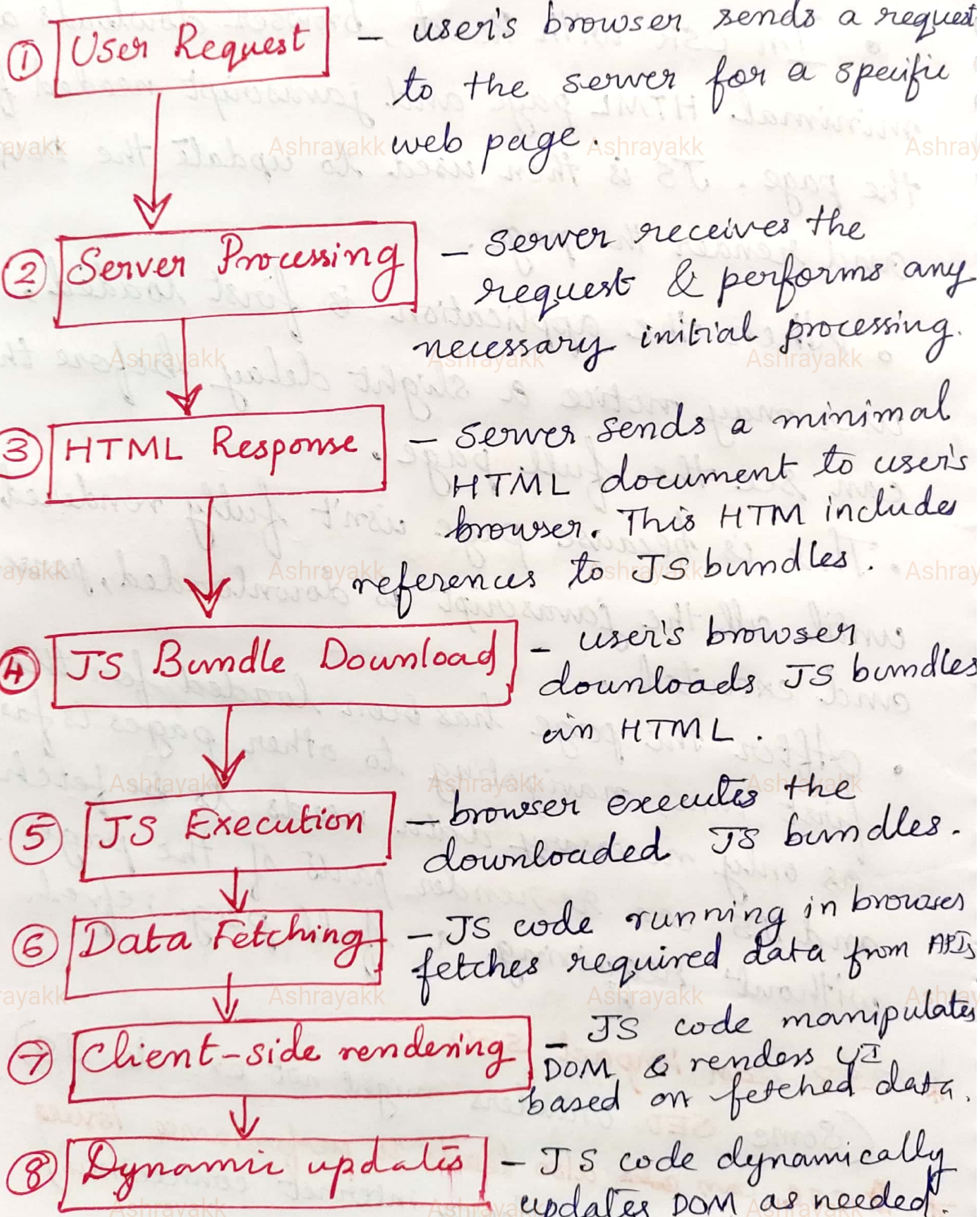
Today, it's no longer the case. Now websites are like applications & advanced. So, SSR slowly takes backseat

CLIENT-SIDE RENDERING

- In CSR with React, browser downloads a minimal HTML page and javascript needed for the page. JS is then used to update the DOM and render the page.
- When the application is first loaded, user may notice a slight delay before they can see the full page.
- This is because page isn't fully rendered until all the javascript is downloaded, parsed and executed.
- After the page has been loaded for the first time, navigating to other pages is faster. as only necessary data needs to be fetched, and JS can re-render parts of the page without requiring a full page refresh

* CSR can impact SEO
  (Some SEO crawlers might not execute JS)
* CSR can also leads to performance issues
  (for users with slower internet connections)

CLIENT-SIDE RENDERING PROCESS



Advantages – CSR

1. Improved interactivity
2. Enhanced Performance after initial load
3. Reduced Server load

CSR- Disadvantages

1. Slower initial page load.
2. Reduced SEO & social media sharing
3. Limited Accessibility without Javascript.