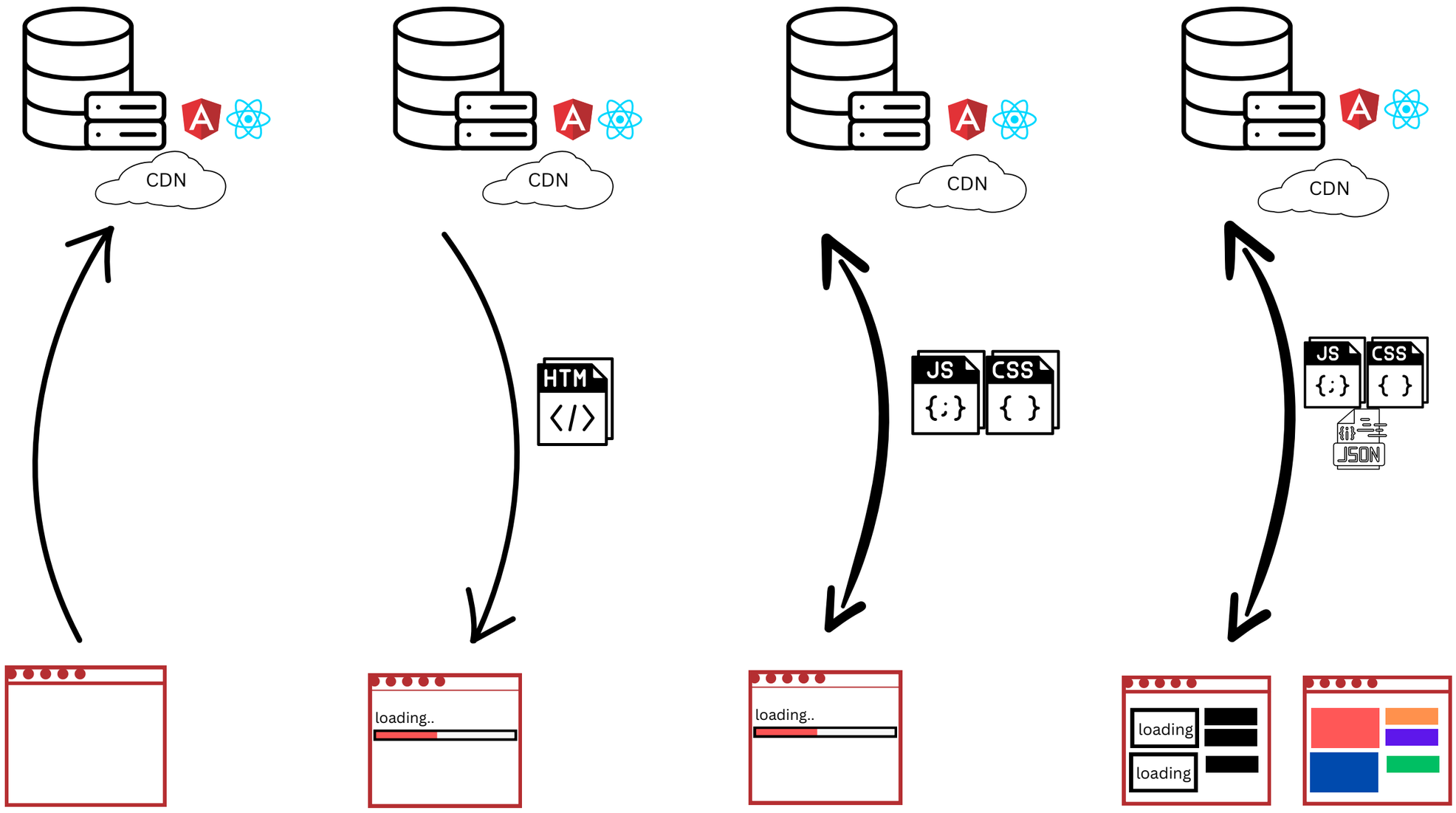
CSR - Client Side Rendering



User Request

The server sends Initial HTML response to the browser.

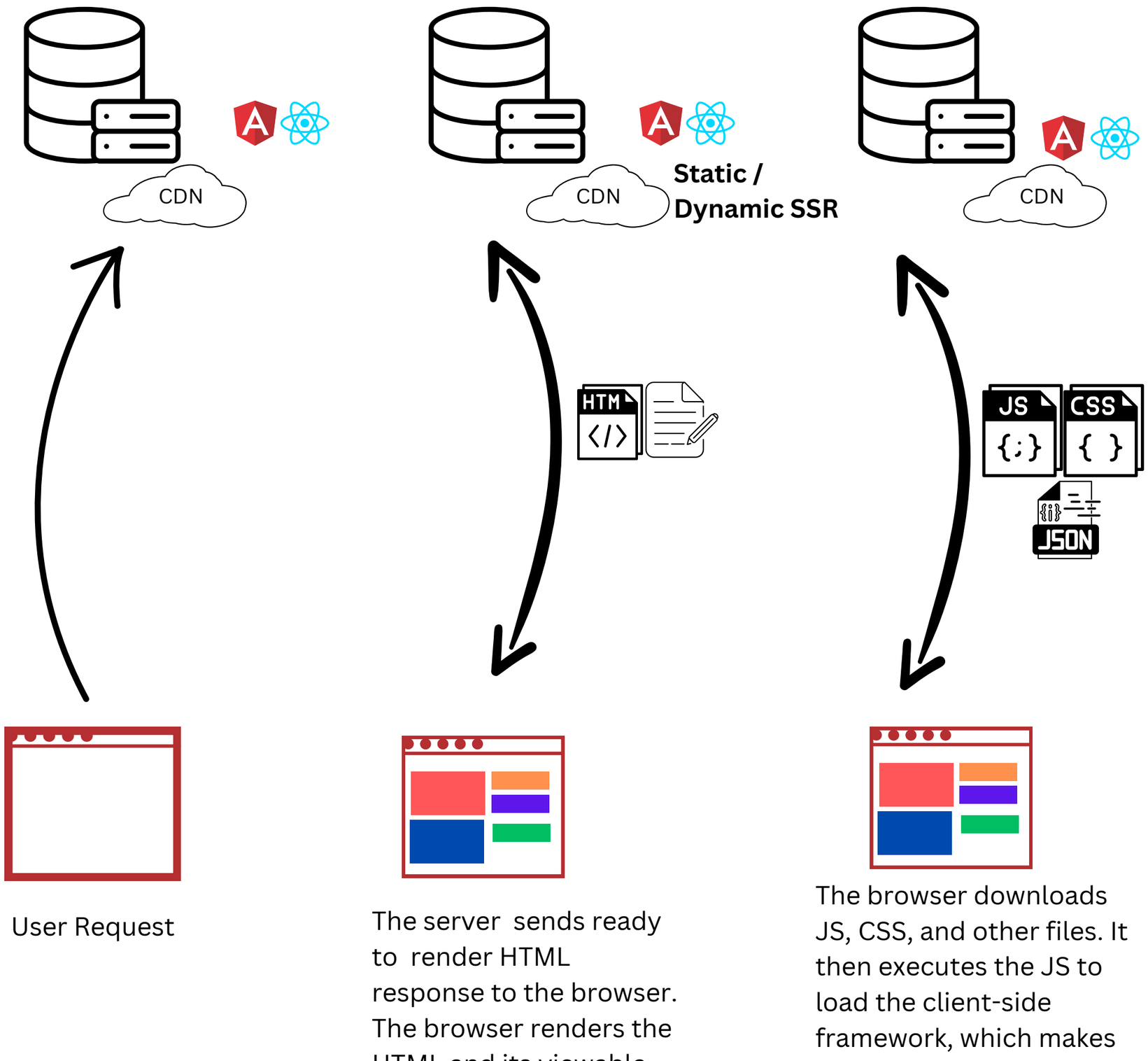
The browser downloads JS, CSS, and other files. It then executes the JS to load the client-side framework

The browser runs the JavaScript and then load the content by making API calls to the server. At this point, the page becomes viewable and interactive

How Client Side Rendering (CSR) works?

- You visit a website.
- The server sends basic files (HTML, CSS, JS) to your browser.
- Your browser downloads these files. You might see a blank page or a loading symbol at this time.
- Once the HTML is downloaded, the static parts of the webpage start to appear.
- The browser downloads the JavaScript, which then asks the server for additional data.
- This data is used to fill in the dynamic parts of the webpage.
- The JavaScript updates the webpage with this data, without needing to reload the page.

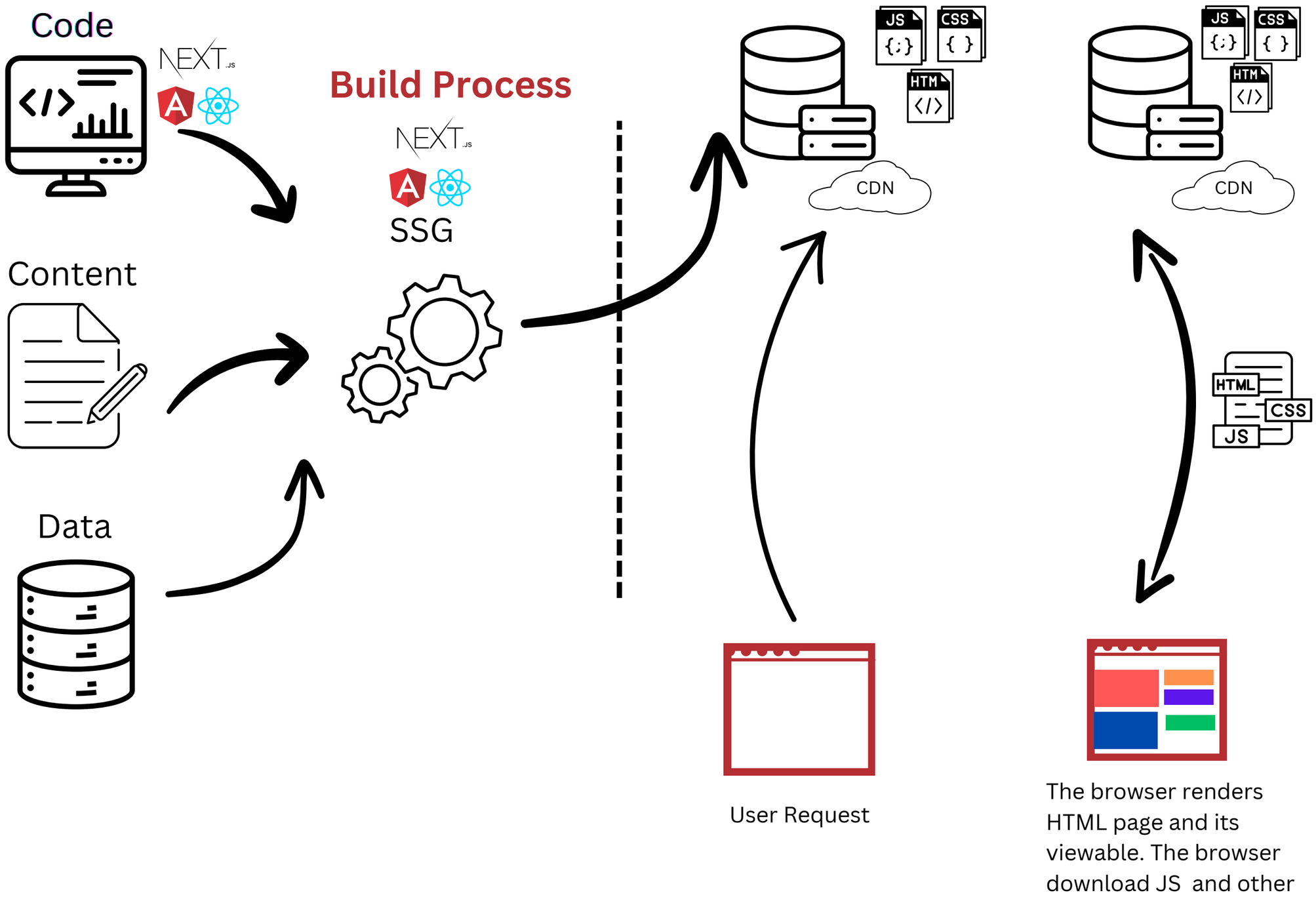
SSR - Server Side Rendering



How Server Side Rendering (SSR) works?

- You visit a website.
- The server prepares the full HTML for the page by running server-side scripts.
- This prepared HTML is sent to your browser.
- Your browser downloads and displays the HTML, making the site visible to you.
- Your browser then downloads and runs the JavaScript, which makes the page interactive.
- In Server-Side Rendering (SSR), the server does all the heavy lifting. It fetches dynamic content, turns it into HTML, and sends it to your browser.
- This can use up a lot of the server's memory and processing power. As a result, pages might load slower compared to static sites that don't have any dynamic content to render.

SSG - Static Site Generation



How Static Site Generation (SSG) works?

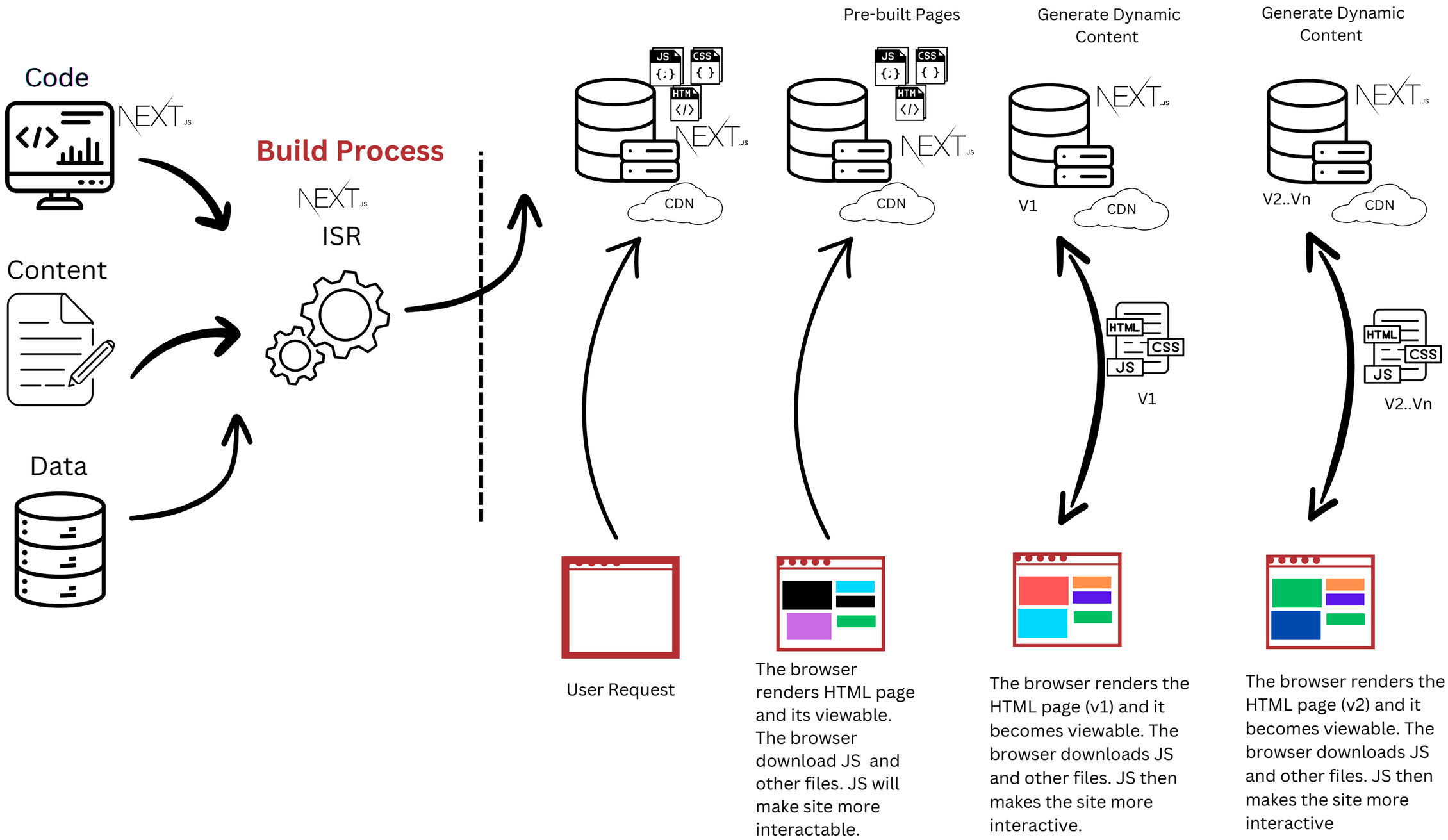
- **Build Process:** In the build process, your static site generator uses your content and templates to create HTML pages.

- **Deployment:** The created HTML, CSS, and JavaScript files are then sent to a web server or Content Delivery Network (CDN). Because these files are static, they can be served to users very quickly.

- **Browser Rendering:** When a user visits your site, their browser downloads the static files from the server or CDN. The HTML is displayed, and the user sees the webpage.

- **Interactivity:** Any interaction on the pages (like form submissions or dynamic content loading) is managed by JavaScript running in the user's browser. This JavaScript can make requests to APIs to fetch or submit data as needed.

ISR - Incremental Static Regeneration



How Incremental Static Regeneration (ISR) works?

- **Build Process:** During the build process, ISR generates a subset of pages. The remaining pages are not built immediately but are prepared to be generated on-demand.

- **Deployment**: The generated HTML, CSS, and JavaScript files are deployed to a web server or Content Delivery Network (CDN). As these files are static, they can be delivered to users rapidly, enhancing the site's performance. Dynamic content is served via web servers, similar to SSR.

- **On Initial Request**: When a user first requests a page that wasn't pre-generated at build time, the page is generated on-the-fly, similar to SSR. This freshly rendered page (let's call it version-1) is then cached and served to subsequent users, mimicking the behavior of SSG. This ensures that future visitors can access the page instantly.

- **On Subsequent Requests**: For any following requests, the server delivers the cached (version-1) page to the user. Concurrently, the JavaScript framework checks if the page is due for regeneration. If it is, the framework regenerates the page on the server-side. This newly generated page (let's call it version-2) then replaces the old page in the cache.