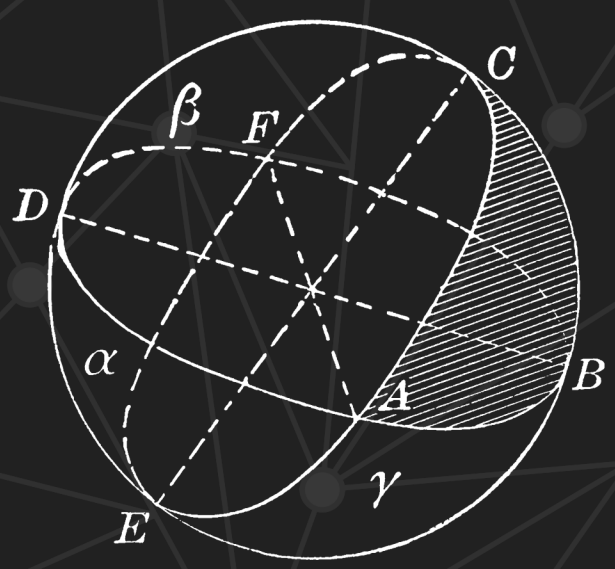
Faster loading in React



Step 1

Code Optimization

RAHUL PATIL

Minify and bundle JavaScript

Tool: Webpack

Minify and bundle JavaScript files to reduce their size and improve loading speed.

Enable code splitting to load only the necessary modules on each page.

RAHUL PATIL

Optimize components

Use React.lazy() and Suspense to lazy-load components when needed.

Split large components into smaller ones and load them dynamically based on user interactions.

RAHUL PATIL

Use memoization

Memoize expensive computations or use React.memo() to prevent unnecessary re-rendering.

Utilize PureComponent for class components to optimize rendering and avoid unnecessary updates.

RAHUL PATIL

Step 2

Asset Optimization

RAHUL PATIL

Compress and optimize images

Tools: Image compression libraries (e.g., imagemin, sharp)

Compress and optimize images to reduce their file size without compromising quality.

Consider using responsive images and lazy-loading techniques to load images as needed.

RAHUL PATIL

Minify and compress CSS

Tools: CSS minifiers (e.g., cssnano, csso)

Minify and compress CSS files to reduce their size.

Remove unused CSS rules and consider using CSS-in-JS solutions for more granular control.

Use font subsets and icon fonts

Tools: Font subsetting tools (e.g., Fontello, Glyphs)

Subset fonts to include only the characters required on your website.

Utilize icon fonts instead of individual SVGs to reduce the number of HTTP requests.

RAHUL PATIL

Step 3

Network Optimization

RAHUL PATIL

Enable caching

Tools: Service workers, Cache-Control headers.

Implement caching strategies using service workers or appropriate HTTP headers.

Cache static assets, API responses, and other frequently accessed resources.

Code splitting and lazy-loading routes

Tools: React Router, Dynamic imports

Split your application into multiple routes and load them dynamically as needed.

Use React Router and dynamic imports to achieve code splitting and optimize route loading.

RAHUL PATIL

Prioritize and optimize critical resources

Tools: Resource prioritization tools (e.g., preload, prefetch)

Identify critical resources required for the initial page load.

Prioritize the loading of critical resources using techniques like resource hints (preload, prefetch).

RAHUL PATIL

Hi I'm Rahul,

I post super practical concepts to get better at React. Follow