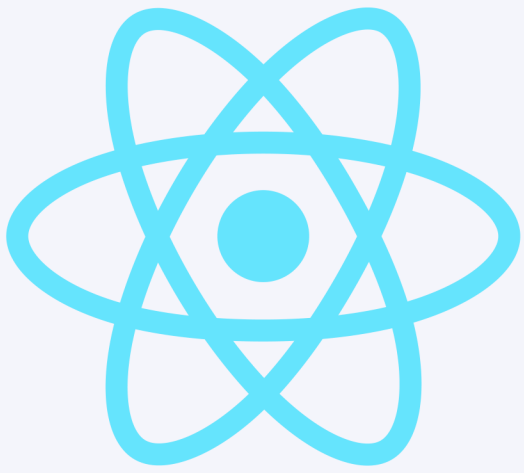
BEST PRACTICES IN REACTJS

RAHUL PATIL

Tip 1

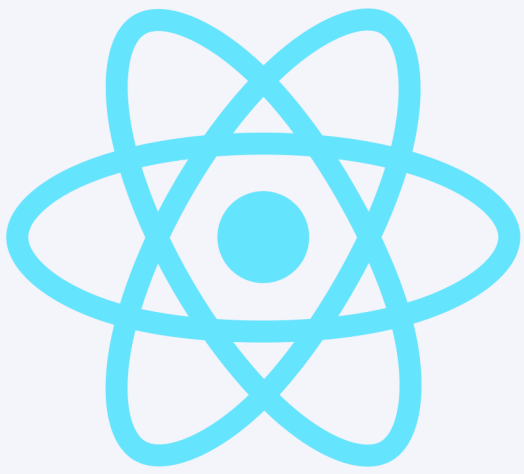
Use React Developer Tools to debug your components and inspect your application's state.

RAHUL PATIL



Tip 2

Break your application into reusable components to keep your code organized and maintainable.

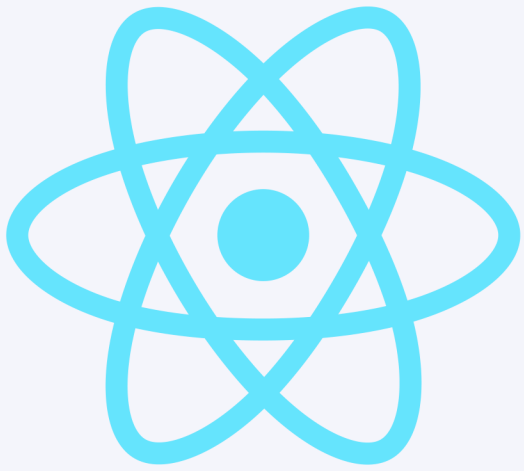


RAHUL PATIL

Tip 3

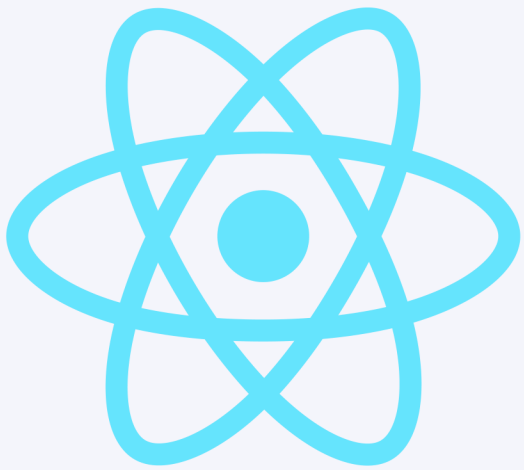
Use a linter like ESLint to enforce coding standards and catch errors before they become problems.

RAHUL PATIL



Tip 4

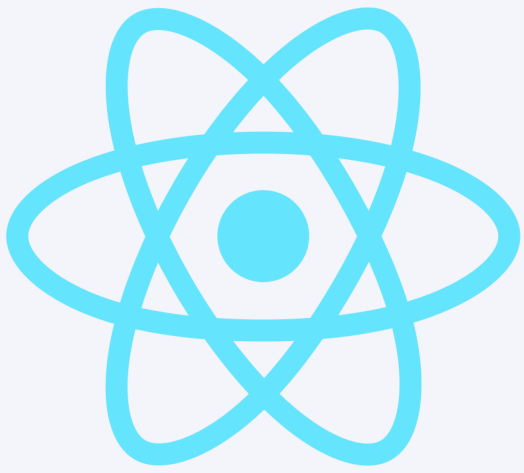
Use a bundler like Webpack to manage dependencies and optimize your application's performance.



RAHUL PATIL

Tip 5

Write unit tests with Jest to ensure that your components behave as expected and catch regressions early on.

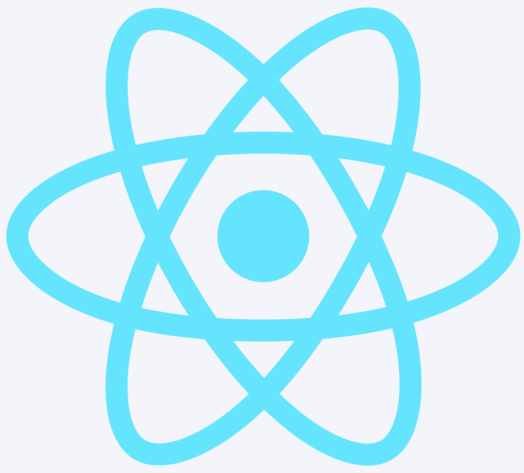


RAHUL PATIL

Tip 6

Use a CSS-in-JS library like Styled Components to make it easy to manage your application's styles.

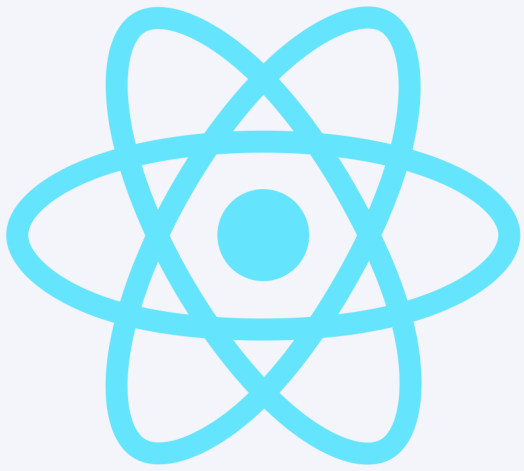
RAHUL PATIL



Tip 7

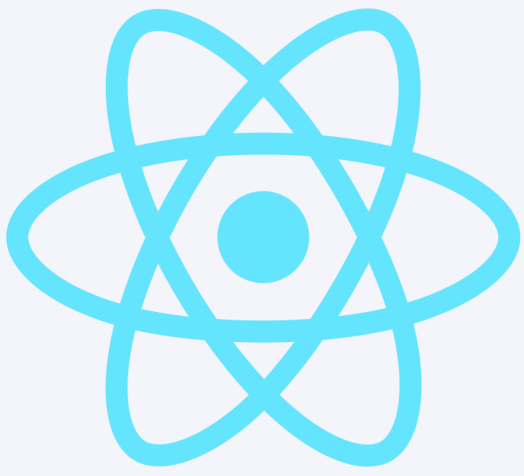
Containerize your application with Docker to make it easier to deploy and manage in production.

RAHUL PATIL



Tip 8

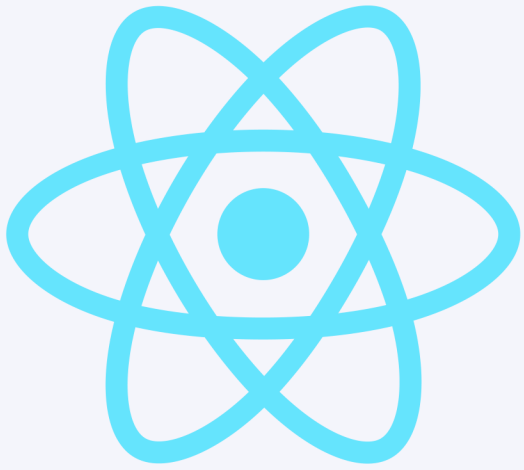
Monitor your application's performance with tools like React Profiler and Chrome DevTools to optimize rendering speed.



RAHUL PATIL

Tip 9

Keep your application's secrets secure with environment variables and tools like dotenv.

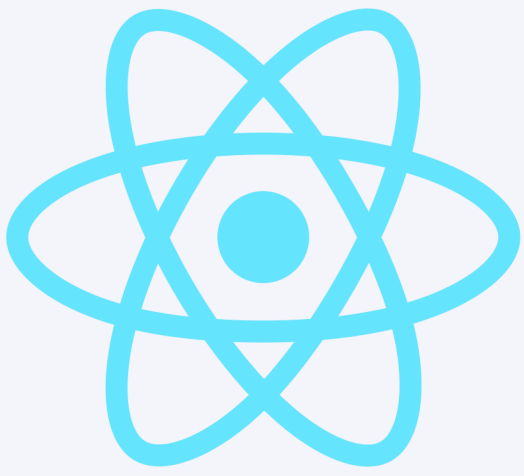


RAHUL PATIL

Tip 10

Use server-side rendering with Next.js to improve SEO and provide a better user experience.

RAHUL PATIL



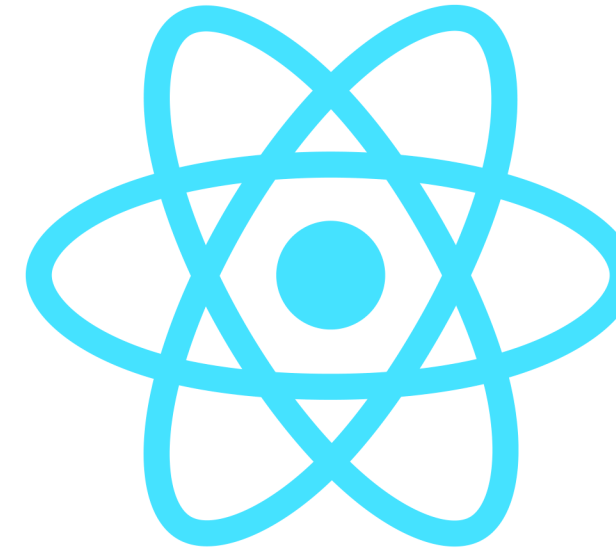
Let's recap:

- Debug with React DevTools
- Reusable components
- Use ESLint for standards
- Manage dependencies with Webpack
- Write tests with Jest
- Style with Styled Components
- Containerize with Docker
- Monitor with Profiler
- Secure with dotenv
- Use Next.js for SEO

RAHUL PATIL

Bonus tip

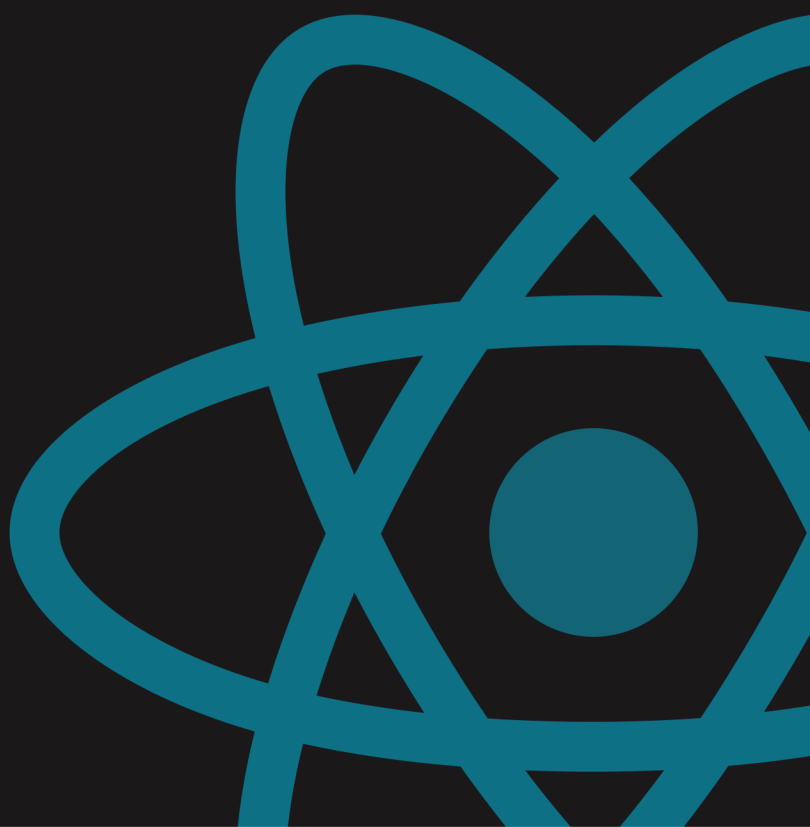
Use React.memo to memoize components and improve performance.



RAHUL PATIL

I make you 1% better at ReactJS everyday

```
Follow();
```



RAHUL PATIL