TRPC: Type-Safe APIs in a Snap

TRPC in 100 Seconds

Introduction

* TRPC: A way to build type-safe APIs without schemas or code generation.

* Two main API approaches: REST and GraphQL

* TRPC offers autocompletion, automatic type safety, and request patching.

REST vs. TRPC

REST:

* Fetch data using `useEffect` on component mount.

* No strong contract between frontend and backend, leading to potential errors.

TRPC:

* Enforces type safety, providing early error detection in the frontend.

Getting Started with TRPC

* Create a new project with `npm create e3f trpc`.

* Set up routes (like folders) and procedures (like files) for your API.

Using TRPC in the Frontend

* Use the `API` object to navigate to a specific procedure (e.g., `exampleRouter.load`).

* TRPC uses `rc-query` to make HTTP requests.

* Changes to the backend code will automatically trigger TypeScript errors in the frontend.

Batching Requests

* TRPC batches multiple requests into a single HTTP request for efficiency.

When Not to Use TRPC

* When frontend and backend are not written in TypeScript.

* When separate backend and frontend teams work independently.

Conclusion

* TRPC is a powerful tool for building type-safe APIs.

* It offers significant benefits over REST, including autocompletion, type safety, and request patching.

* However, TRPC may not be suitable for all projects.