tRPC

Move Fast & Break Nothing

I’m going to talk about a new way of building APIs that…

…doesn't let your app do anything new

…doesn't improve performance

…is a nonstarter for many projects

But it DOES let you…

…build stuff fast

…feel confident about making changes

…not lose time to dumb mistakes

…get an MVP out in record time!

I’m Chris (@ccccjjjjeeee) 👋

- Designer turned teacher turned developer
- Currently doing full stack & DX stuff at Moody’s Analytics
- **tRPC** (23k 🌟): Contributor, docs person, and educator
- **Create T3 App** (15k 🌟): Core maintainer
- Hobby is …*checks notes*… devtools and DX
- This is my first meetup talk 🎉

Raise your hand if you have

🙋 used tRPC

🙋 used GraphQL

🙋 used REST

🙋 Built a frontend that talks to an API

🙋 Used React Query

How should we build APIs?

- Everyone starts with REST
- It’s pretty good
- Tanstack Query makes it even better

but…

Have you ever lost an hour to something like this?

```
example.com/api/foo/bar?thing=this&otherthing=that&sonenumber=123&someconfig=true
                                                     ^
```

Have you ever struggled to find the api logic that some component calls?

```
// frontend/SomeComponent.tsx
const someQuery = useQuery({ queryKey: ["some", "key"], fetchFn: someFetchFn })

// frontend/fetchFunctions.ts
export function someFetchFn() {
  fetch("/api/something/otherthing", { body: { id: "abc123" }}).then(parseRes);
}

// api/index.ts
app.use("/api", require("./something/router"));

// api/something/router.ts
router.post("/otherthing", otherThing);

// api/something/otherThing/handler.ts
async function otherThing(req, res) {
  const id = req.query.id as number;
  const item = await prisma.item.findUnique({ where: { id }});
  return res.status(200).json({ item });
}
```

Do you remember if `` `id` `` on the previous slide was a string or a number?

and what happens if you send the wrong type?

Does your backend logic need to check for every way the request could be *incorrect*, or can you just tell it what a *correct* request looks like?

Did you ever break something by changing backend behavior but not every frontend consumer?

We already have solutions for this

GraphQL is pretty great!!

- 🔨 Schema as a source of truth / contract
- 🪖 Once you go typesafe you can’t go back
- 🤯 Your frontend KNOWS what your backend looks like!!!

(also OpenAPI)

But GraphQL is not ideal for every project

- Writing a schema is a lot of work
- Codegen is an extra build step
- Ecosystem is complex
- My data isn’t very graph-like, I’m just building a more complicated REST API

Can we do better?

(by using something that doesn’t need to solve EVERY problem)

- REST but with all the best parts of GraphQL
- Frontend knows about the backend
- Input validation in FE and BE (with Zod)
- No codegen / extra steps
- Full-stack Next.js + React Query ❤️

I’ll build the library for this 🚀

(famous last words)

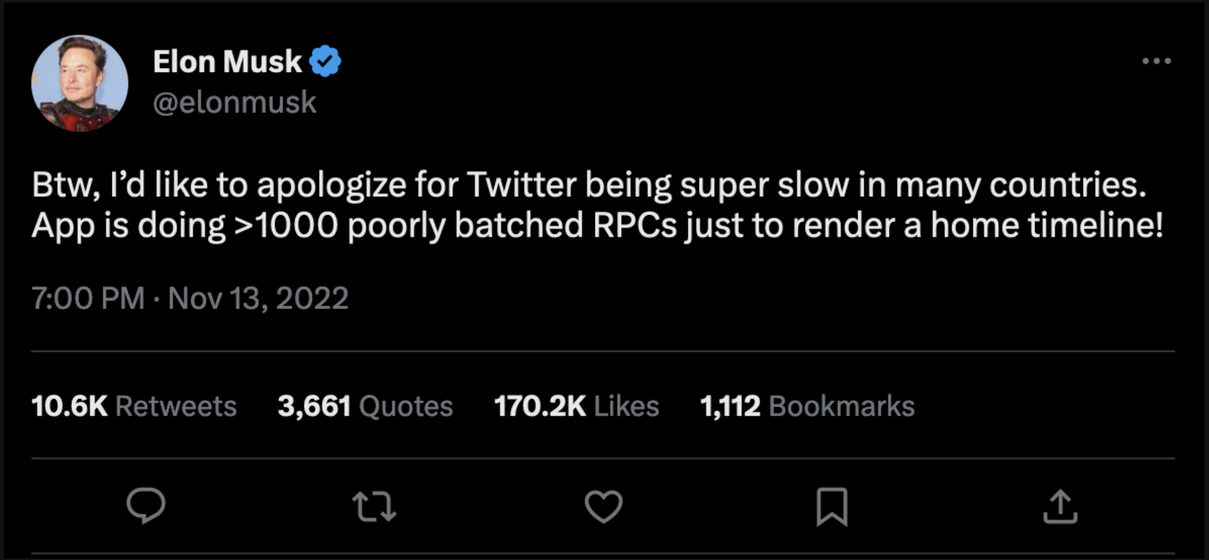
…it kind of worked, but not very well - I’m not a 🧙

(sharing types and validation is easy, letting your frontend know what your backend looks like is HARD)

Then I found tRPC

Someone else had already solved my exact problem

What is RPC?



Elon Musk
@elonmusk

Btw, I'd like to apologize for Twitter being super slow in many countries. App is doing >1000 poorly batched RPCs just to render a home timeline!

7:00 PM · Nov 13, 2022

10.6K Retweets **3,661** Quotes **170.2K** Likes **1,112** Bookmarks

Calling a function on one computer from another computer

...isn't that what a REST API is?

…and what is `` `t` ``?

Typesafety

How does it work?

- Define the backend as a big object
- Export it as a type, import in frontend
- ???
- React Query knows your backend

Vocabulary

- **Procedure**: API endpoint - can be a **query** or **mutation**
- **Query**: get some data
- **Mutation**: change some data
- **Router**: a collection of **procedures** (and/or other routers)
- **Context**: stuff that every **procedure** has access to (db, session)
- **Middleware**: do stuff before and after a **procedure**, can modify **context**
- **Validation**: "Does this input data contain the right stuff?"

Let’s look at some code 🤓

...there is so much more!

- Form input validation (Zod + hook-form ❤️)
- Easy optimistic UI updates
- Everything else React Query has
- Great error handling
- Easy SSR and SSG
- Procedures are easy to test (using context as "DI")
- tRPC Panel (like GraphQL Playground)
- tRPC-OpenAPI (if you need external consumers after all)
- msw-trpc (easily mock endpoints from your frontend)

Works with

Server

- Express
- Fastify
- Next.js
- 3rd party adapters for some others
- Vanilla node
- All major edge solutions

Notes

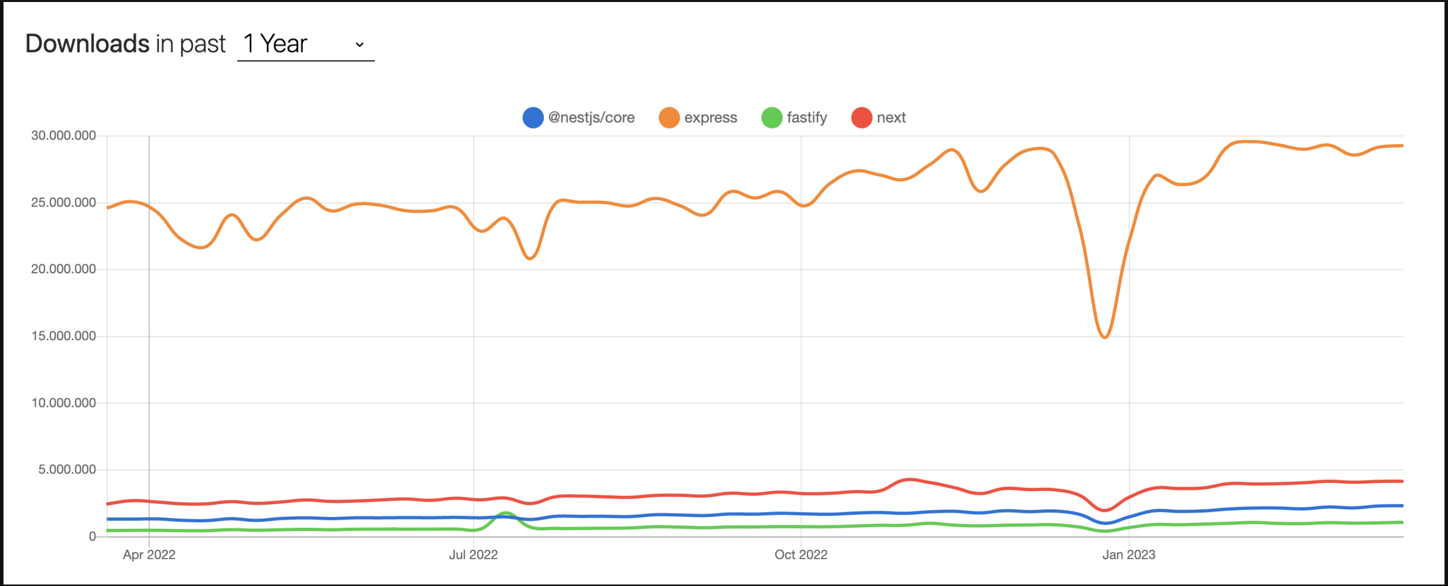
- You should use a monorepo (or a Next.js app)
- One backend can have multiple consumers
- Different backends can call each other’s procedures

Client

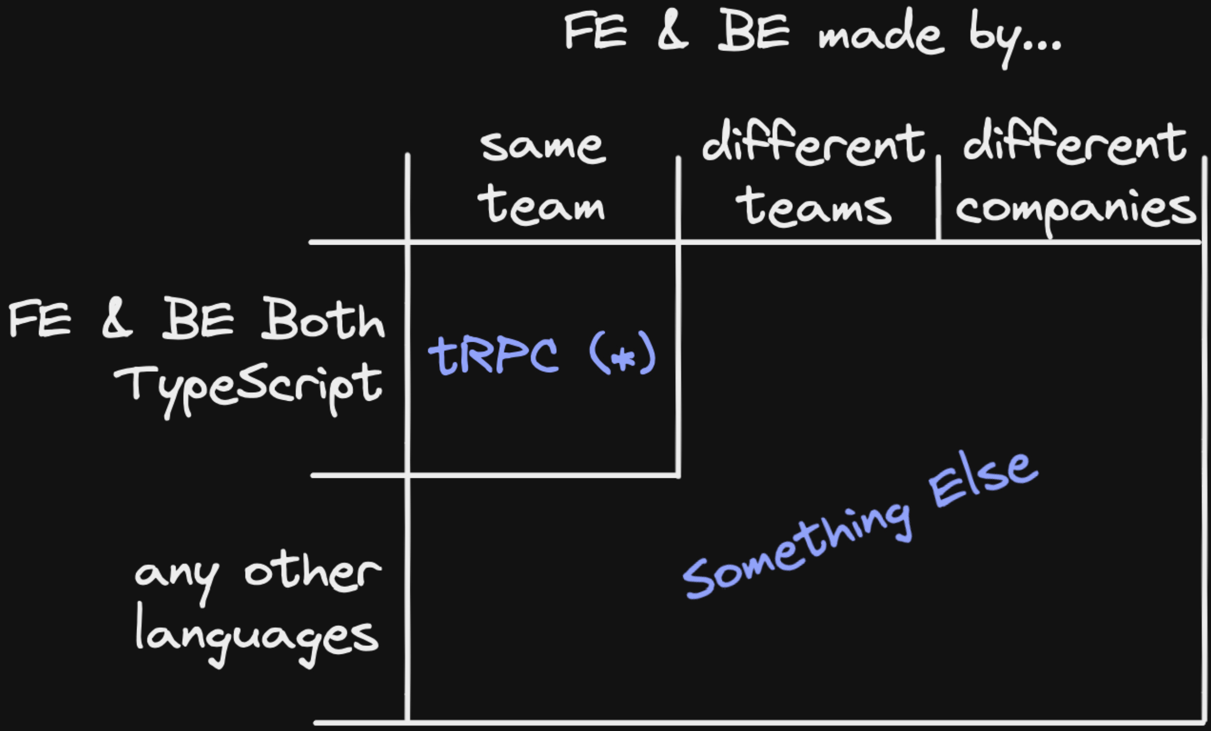
- React / Next.js
- React Native
- Vanilla client runs anywhere JavaScript can run
- 3rd party adapters
  - Vue / Nuxt
  - Solid Start
  - SvelteKit
  - Qwik

Performance

- tRPC with Next.js is:
  - Faster than Express
  - Slower than Fastify
  - (but seems like Express is fast enough for most people)



Who is it for, and who is it not for?



	FE & BE made by...		
	same team	different teams	different companies
FE & BE Both TypeScript	tRPC (*)	Something Else	
any other languages	Something Else		

(*) maybe still GraphQL if you have a good reason

Similar projects ("Typesafe REST/RPC")

- **ts-rest, Zodios, Remult**: Explicitly declare the API schema
    - Difference to tRPC: Do you want to declare a contract programmatically, or do you want your backend logic to *be* the contract?
- **Blitz.js**: Rails-like framework
    - Difference to tRPC: Blitz is more ambitious (full framework, compiler, etc), but you need to commit to it more

Companies using tRPC

- Netflix (for internal tools)
- Stately.ai (XState etc)
- Cal.com (OSS codebase)
- Skill Recordings (Total TypeScript, Kent C Dodds, Dan Abramov, etc courses - OSS codebase)
- Ping.gg
- …many more

What about React Server Components?

- We're working on it! (and waiting for the mutations RFC)
- Are you builing an app or a website? Or something inbetween?
- Most things are inbetween!
- tRPC as an answer to the "Two Applications Problem"
- Will be wild west of competing patterns for a while
- Talk to me afterwards if you want to nerd out about RSCs

Questions?

(scan for resources, videos, etc.)