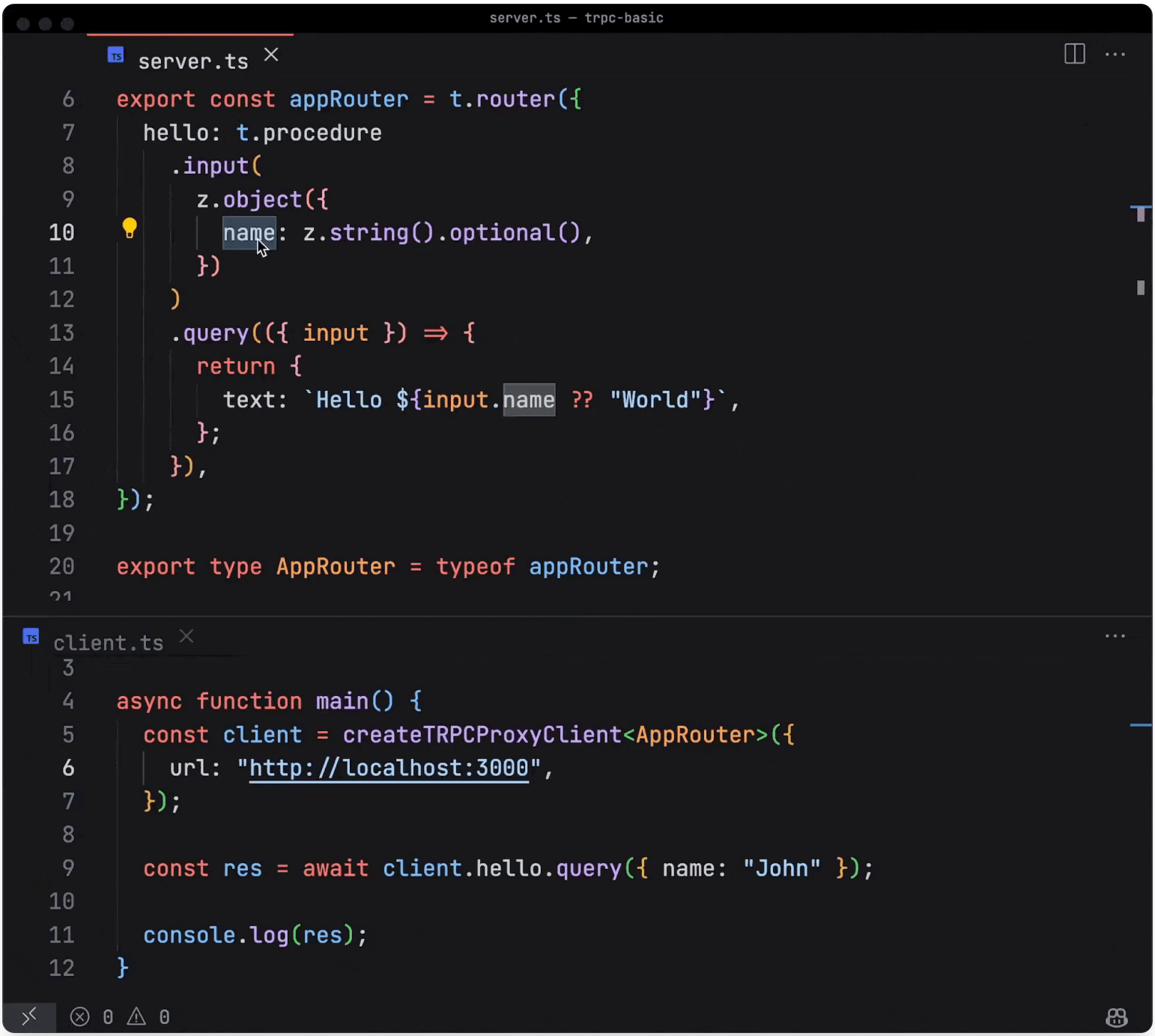
Move Fast and Break Nothing.
End-to-end typesafe APIs made easy.

Experience the full power of TypeScript inference to boost productivity for your full-stack application.

Star 19,040 Quickstart



SUPPORTED BY



Many thanks to all of our amazing sponsors!

Automatic typesafety

Made a server side change? TypeScript will warn you of errors on your client before you even save the file!

Snappy DX

tRPC has no build or compile steps, meaning no code generation, runtime bloat or build step.

Framework agnostic

Compatible with all JavaScript frameworks and runtimes. It's easy to add to your existing projects.

Autocompletion

Using tRPC is like using an SDK for your API's server code, giving you confidence in your endpoints.

Light bundle size

tRPC has zero dependencies and a tiny client-side footprint making it lightweight.

Batteries included

We provide adapters for React, Next.js, Express, Fastify, AWS Lambda, Solid, Svelte, and more.

Simple to use with unmatched developer experience

It's quick and easy to get started with tRPC to build a typesafe API.

1

Define your procedures

The first step to creating a tRPC API is to define your procedures.

Procedures are the functions we will use to build your backend. They're *composable* and can be queries, mutations, or subscriptions. Routers contain multiple procedures.

In this procedure, we use a Zod validator to ensure the input from the client has exactly the shape that our procedure expects. We will also return a simple text string from the query.

At the end of the file, we export the type of the router so we can use it in our frontend code in just a few moments.

```
const t = initTRPC.create();

const router = t.router;
const publicProcedure = t.procedure;

const appRouter = router({
  greeting: publicProcedure
    .input(z.object({ name: z.string() }))
    .query((req) => {
      const { input } = req;
                ^
                const input: {
                    name: string;
                }


      return {
```

```
        text: `Hello ${input.name}` as const,
      };
  }),
});

export type AppRouter = typeof appRouter;
```

2

Create your HTTP server

Next, we create our HTTP server using our `appRouter`. We now have a tRPC server running!

tRPC has many adapters so it can meet you where you are. Next.js, Express, the Fetch API (Astro, Remix, SvelteKit, Cloudflare Workers, etc.), Fastify, AWS Lambda, or a vanilla Node HTTP server.

```
const { listen } = createHTTPServer({
  router: appRouter,
});

// The API will now be listening on port 3000!
listen(3000);
```

3

Connect your client and start querying!

Now that we have the server running, we can create a client and start querying data.

We pass the `AppRouter` type when creating the client to give us TypeScript autocompletion and Intellisense that matches the backend API without requiring any code generation!

```
const trpc = createTRPCProxyClient<AppRouter>({
  links: [
    httpBatchLink({
      url: 'http://localhost:3000/trpc',
    }),
  ],
});

const res = await trpc.greeting.query({ name: 'John' });
```

```
const res: SerializeObject<UndefinedToOptional<{
    text: `Hello ${string}`;
}>>
```

You may not need a traditional API

> *I built tRPC to allow people to **move faster** by removing the need of a traditional API-layer, while still having confidence that our apps won't break as we rapidly iterate.*
>
> *Try it out for yourself and let us know what you think!*

Alex/KATT
Creator of tRPC

Don't take our word for it!

Many developers are loving tRPC and what it brings to them.

Theo - ping.gg
@t3dotgg

Sep 19

The amount that tRPC has improved the quality of our code, the speed of our delivery, and the happiness of our devs is hard to comprehend.

I know I shill it a lot but seriously, please try @trpcio

R. Alex Anderson 🚀
@ralex1993

Sep 23

🤯 tRPC 10 enables VS Code's "Change All Occurrences" feature to work _across the client/server boundary_!

In this video, I rename a procedure input using "Change All Occurrences", and that

change propagates to anywhere the input is used across the entire app. 🤩

cc @trpcio

Kent C. Dodds 🌌
@kentcdodds

Sep 20

If I didn't already get end-to-end type safety from @remix_run, I would 100% be investigating @trpcio very seriously. If you're not on Remix, I suggest you give it a look 👀

Sock, the dev 🧦
@sockthedev

Sep 13

If you are all in on TypeScript you MUST use tRPC for your API. No ifs, no buts.

tRPC destroys the boundary between frontend and backend. You get to focus on building features for your app.

Best tool for time to market hyper mode.

Marry me @alexdotjs 💍

Lee Byron
@leeb

Dec 15

Hearing @t3dotgg and @mxstbr #tRPC and @GraphQL and find they agree that both are awesome and there’s a time and a place for each 💖

Jökull Solberg
@jokull

Aug 23

tRPC is insane. I’m building a Stripe integration – I return Stripe API payloads from the server I get the response data typed for my React components without even saving the files, as if I’m using the Stripe library on the frontend not backend. /cc @alexdotjs

Christian Legge
@christian_legge

Sep 11

Spent all of yesterday learning and implementing @trpcio and wow, what a great investment. I can't believe how much time I spent (read: wasted) validating and parsing queries and responses!

Dominik 🇺🇦
@TkDodo

Sep 23

That being said, we _are_ starting a production project right now, and we're using @nextjs with @trpcio . It's so good I don't even know where to start 🔥. Probably with the e2e type-safety 😍
Haven't thought about client state much but the former probably applies.

Cory House
@housecor

Sep 11

@trpcio Love it.

Simple, strong types.

Feels like a more elegant choice than plain REST or GraphQL when using TS in a monorepo.

Anders Bech Mellson
@andersmellson

Sep 12

Spent today playing with @trpcio v10 and I'm officially in love 😍 ps. Don't tell my wife 🙊

Mike | grabbou.eth 🚀
@grabbou

Sep 19

@t3dotgg @trpcio Totally. I am literally smiling every time I write a procedure, because it reminds of how hard it used to be in the past. Built-in errors, typed middleware (that can alter context), input validation. It's just massive!

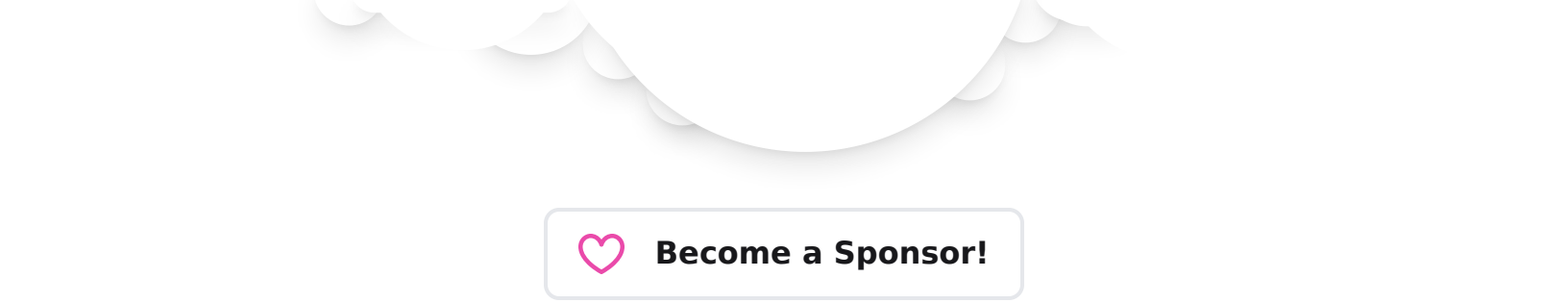
Martin
@wikitable

Aug 23

💖 I'm sponsoring @alexdotjs because tRPC has helped to build apps faster.
github.com/sponsors/KATT?…

All Sponsors

We really love all of our amazing sponsors who help make sure tRPC is here to stay.



Become a Sponsor!