Nearform_

View online

Download PDF

Download PPTX

Don't trust your types
Type testing with TSTyche_

Luca Del Puppo Nearform_

N_

Hi 👋, I'm **Luca**!_



Nearform_ AI Native Engineer

GDE Web Technologies

delpuppo.net

puppo

puppo92

lucadelpuppo



We’re an independent team of engineers, designers and strategists who build digital solutions and enhanced capability at pace for ambitious enterprises seeking enduring business impact.

10+ Years experience

700+ Engagements

400+ Nearformers

380+ Customers

28 Countries



Nearform’s commitment to Open Source

15m+ Downloads	Over 15 million downloads every week
1192k+ Contributions	Thousands of contributions every year
166+ Packages	We maintain many popular packages for the community

We’re hiring! **nearform.com/careers**

Nearform_ N_

A green build can still ship a broken developer experience_

We will debate semicolons for hours.

Then we treat generic inference as an act of faith.

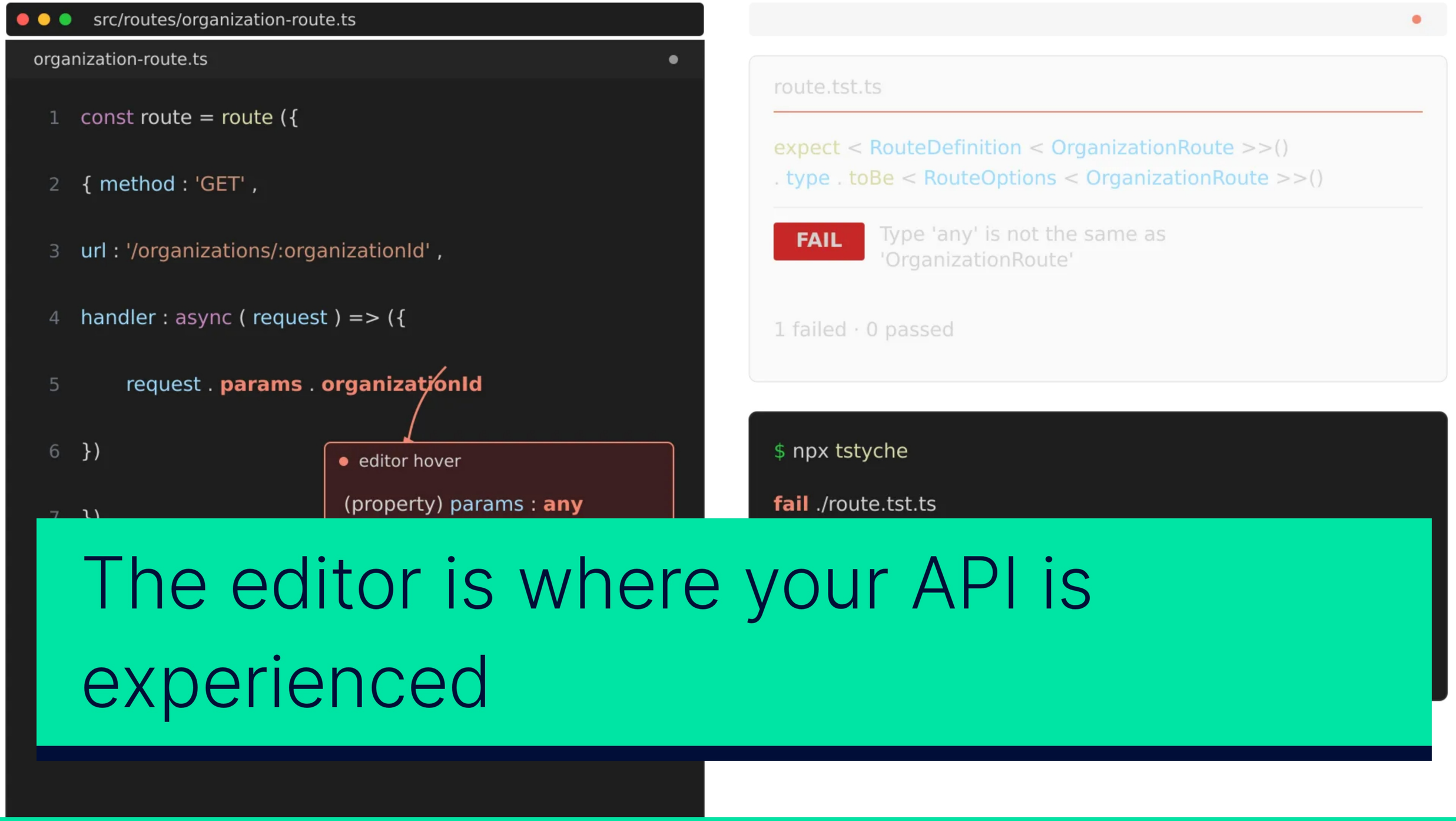
A refactor from T to any can erase autocomplete without breaking the build.

Runtime behavior ≠ type-time contract_

Runtime tests
Does the Fastify route return the right response?

Type tests
Can a consumer call the route with the right params, query, and reply?

The gap
JavaScript can behave perfectly while the public types quietly decay.



The editor is where your API is experienced

Make contracts executable_

Type testing with TSTyche



What TSTyche brings to the table_

1 **A test runner for types**
Familiar test files, assertions, focused failures.

2 **The TypeScript language service**
The same type-checking view behind editor hovers.

3 **Compatibility targets**
Run one contract suite against more than one TypeScript version.

First test in 60 seconds_

```
npm add -D tstyche

# route.tst.ts
npx tstyche
```

4

A tiny Fastify package with a public contract_

```ts
import type { FastifyPluginAsync, RouteGenericInterface, RouteOptions } from 'fastify'

export type RouteContract = RouteGenericInterface
export type RouteDefinition<T extends RouteContract> =
  RouteOptions<T>

// re-exported so consumers can write equality tests against the underlying type
export type { RouteOptions }

export type RouteFactory = <T extends RouteContract>(
  definition: RouteDefinition<T>,
) => FastifyPluginAsync

export const route: RouteFactory = definition => async fastify => {
  fastify.route(definition)
}
```

The regression is one character wide_

```
// Before: inference carries the consumer's route contract
export type RouteDefinition<T extends RouteContract> =
  RouteOptions<T>

// After a harmless-looking refactor ...
export type RouteDefinition<T extends RouteContract> =
  RouteOptions<any>

// The app still starts. The user's hover now says: any.
```

Equality catches the silent any_

```ts
import { expect, test } from 'tstyche'
import { route } from 'fastify-route-contracts'
import type {
  RouteDefinition,
  RouteOptions
} from 'fastify-route-contracts'

type OrganizationRoute = {
  Params: { organizationId: string }
  Querystring: { include?: 'members' | 'projects' }
  Reply: { id: string; name: string }
}

const organizationDefinition: RouteDefinition<OrganizationRoute> = {
  method: 'GET',
  url: '/organizations/:organizationId',
  handler: async request => ({
    id: request.params.organizationId,
    name: 'Acme'
  })
}

test('RouteDefinition preserves its generic', () => {
  // Fails the moment RouteDefinition widens to RouteOptions.
  expect<RouteDefinition<OrganizationRoute>>().type.toBe<
    RouteOptions<OrganizationRoute>
  >()
})
```

Can a consumer call it?_

```ts
import { expect, test } from 'tstyche'
import { route } from 'fastify-route-contracts'
import type {
  RouteDefinition,
  RouteOptions
} from 'fastify-route-contracts'

type OrganizationRoute = {
  Params: { organizationId: string }
  Querystring: { include?: 'members' | 'projects' }
  Reply: { id: string; name: string }
}

const organizationDefinition: RouteDefinition<OrganizationRoute> = {
  method: 'GET',
  url: '/organizations/:organizationId',
  handler: async request => ({
    id: request.params.organizationId,
    name: 'Acme'
  })
}

test('route factory accepts the intended definition', () => {
  expect(route).type.toBeCallableWith(organizationDefinition)
})
```

Reject bad input before runtime_

```ts
import { expect, test } from 'tstyche'
import { route } from 'fastify-route-contracts'
import type {
  RouteDefinition,
  RouteOptions
} from 'fastify-route-contracts'

type OrganizationRoute = {
  Params: { organizationId: string }
  Querystring: { include?: 'members' | 'projects' }
  Reply: { id: string; name: string }
}

test('a bad reply shape is rejected', () => {
  expect(route).type.not.toBeCallableWith({
    method: 'GET',
    url: '/organizations/:organizationId',
    handler: async () => ({ id: 42, name: 'Acme' }),
  })
})

test('a missing route param is rejected', () => {
  expect(route).type.not.toBeCallableWith({
    method: 'GET',
    url: '/organizations',
    handler: async () => ({ id: 'x', name: 'Acme' }),
  })
})
```

One suite, multiple TypeScript versions_

```
# The package supports both compiler generations
npx tstyche --target '5.4 || 5.9'

# Or let CI cover every supported minor release
npx tstyche --target '⩾5.4'
```

10

Make CI protect the package contract_

```
{
  "scripts": {
    "test:types": "tstyche",
    "test:types:compat": "tstyche --target '5.4 || 5.9'"
  },
  "devDependencies": {
    "tstyche": "latest"
  }
}
```

The feedback loop is deliberately small_

Edit
Change the generic or the declaration file.

Hover
See the consumer-facing type in VS Code.

Test
TSTyche checks the same language-service view.

Ship
CI prevents a type contract from regressing silently.

Types are active documentation

Start with the public type contract your users depend on.

Give your coding agent the TSTyche playbook_

```
npx skills add tstyche/tstyche-skills

# Choose the workflow you need
# tstyche-type-tests        assertions, inference, compatibility
# tstyche-project-setup     config, CLI, watch mode, CI
# tstyche-programmatic-api  runners, reporters, embedded runs
```

Live demo: make the type contract fail_

```
git clone https://github.com/Puppo/dont-trust-your-types
cd dont-trust-your-types

npm test              # runtime stays green
npm run test:types    # the contract catches the refactor
npm run test:types:compat
```

Keep exploring_

TSTyche
https://tstyche.org

Fastify TypeScript
https://fastify.dev/docs/latest/Reference/TypeScript/

Talk demo repo
https://github.com/Puppo/dont-trust-your-types

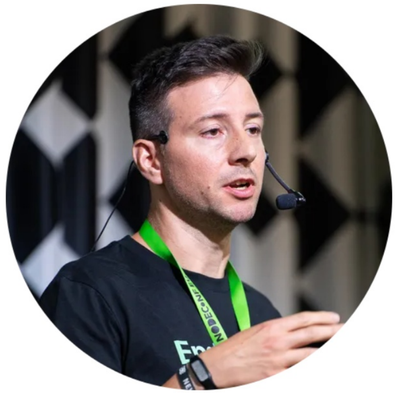


Thank you!

Luca Del Puppo @puppo92
Nearform_ luca@delpuppo.net

Nearform_

Bye Bye , I'm **Luca**!_



Nearform_ AI Native Engineer

GDE Web Technologies

delpuppo.net puppo puppo92 lucadelpuppo



Nearform_

View online

Download PDF

Download PPTX

Don't trust your types
Type testing with TSTyche

Luca Del Puppo Nearform_

N_