



# Fastify & TypeScript

By Ethan Arrowood

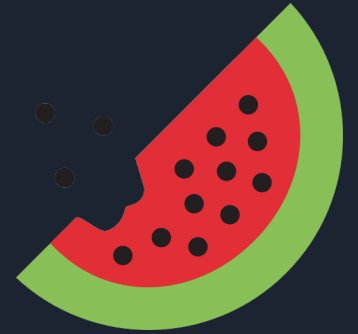
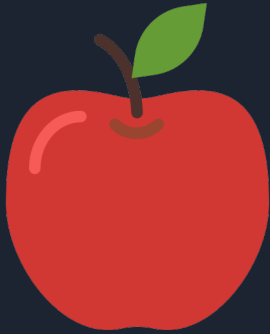
 @ArrowoodTech

 @Ethan-Arrowood

# Discriminant Unions & Function Overloads

- *Discriminant*: a characteristic that enables things to be distinguished from one another
- TypeScript is smart enough to use these discriminants to do type narrowing automatically

*type fruits* =



- all are fruits
- apple is **red** and has **inner-seeds**
- orange is **orange** and has **inner-seeds**
- strawberry is **red** and has **outer-seeds**
- watermelon is **green** and has **inner-seeds**



# Discriminant unions continued

[TypeScript Playground](#)



# Named Generic Parameters

```
function f ({ foo, bar }) {  
  console.log(foo, bar)  
}  
f({ foo: true, bar: 100 })
```

[TypeScript Playground](#)

# Declaration Merging

- Plugin systems are hard in TypeScript, and in Fastify, everything is a plugin 😅

```
const fastify = require('fastify')()
fastify.register(require('fastify-multipart'))

fastify.post('/', async (req, res) => {
  • const data = await req.file()
  • await pump(data.file, fs.createWriteStream(data.filename))
  • return res
})
```

## fastify.ts

```
export interface FastifyRequest {}
export interface FastifyResponse {}
type RouteHandler = (req: FastifyRequest, res: FastifyResponse) => Promise<unknown>
type Route = (path: string, handler: RouteHandler) => void

export interface FastifyInstance {
  register(plugin: any): void;
  get: Route;
  post: Route;
  put: Route;
}

const fastify = () => ({} as FastifyInstance)
export default fastify
```

## fastify-multipart.ts

```
declare module 'fastify' {
  export type Data = {
    file: ReadStream,
    filename: string
  }
  interface FastifyRequest {
    file(): Promise<Data>
  }
}
```

## app.ts

```
import fastify from 'fastify'
import fastifyMultipart from 'fastify-multipart'

const server = fastify()
server.register(fastifyMultipart)

server.post('/', async (req, res) => {
  const data = await req.file()
  await pump(data.file, fs.createWriteStream(data.filename))
  return res
})
```

# TypeScript Playground

(the types displayed here are heavily simplified)

# TypeScript 4.1 Template Literal Types



Dan Vanderkam  
@danvdk

Another use of [@TypeScript](#) 4.1's template literal types: extracting the URL parameters from an express route. Pretty amazing you can do this in the type system!  
[typescriptlang.org/play?ts=4.1.0-...](https://typescriptlang.org/play?ts=4.1.0-...)

```
type C = {  
  postId: string;  
  commentId: string;  
}
```

```
type C = ExtractRouteParams<'/posts/:postId/:commentId'>;
```



Jamie Kyle  
@buildsgghost

I am so sorry for this...

I wrote a JSON parser using [@typescript](#)'s type system

[typescriptlang.org/play?ts=4.1.0-...](https://typescriptlang.org/play?ts=4.1.0-...)

unfortunately the JSON parser was too big to fit in one screenshot, but you can check it out at this playground link:  
<https://tsplay.dev/jwga9m>



# Fastify validation and serialization

- Powered by JSON Schema and AJV
- Makes routes faster and more secure

```
const fastify = require('fastify')()
const routeSchema = require('./routeSchema.json')
fastify.post('/', { schema: routeSchema }, async req => {
  const { querystring, body, headers, params } = req
})
```

- What about TypeScript?

# Existing Solutions

## json-schema-to-typescript

9.1.1 • Public • Published 4 months ago

- JSON Schema is typed as usual
- Types are then generated based on the JSON
- Adds a potential extra-step in a project tool chain

```
import fastify from 'fastify'
import routeSchema from './routeSchema.json'
// generated by json-schema-to-typescript
import RouteSchema from './RouteSchema'

const server = fastify()

server.post<RouteSchema>('/', { schema: routeSchema }, async req => {
  const { querystring, body, headers, params } = req
  // ...
})
```

## @sinclair/typebox

0.11.0 • Public • Published 5 days ago

- JSON Schema is typed using a fluent-like API
- Static types can be immediately derived
- Can produce very clean code

```
import fastify from 'fastify'
import { Type, Static } from '@sinclair/typebox'

const routeSchema = Type.Object({
  body: Type.Object({
    foo: Type.String()
  })
})

const server = fastify()

server.post<Static<typeof routeSchema>>(
  '/',
  { schema: routeSchema },
  async req => {
    const { querystring, body, headers, params } = req
    // ...
  }
)
```


# What if?

TypeScript 4.1 + Fastify + JSON Schema = ✨🧪🚀




```
import fastify from 'fastify'
import routeSchema from './routeSchema.json'


const server = fastify()

// the Route method type infers the types directly from the schema
server.post('/', { schema: routeSchema }, async req => {
  • const { querystring, body, headers, params } = req
  • // ...
})
```



PullsIssuesMarketplaceExplore





# Fastify

Fast and low overhead web framework, for Node.js

<http://fastify.io> Verified

Repositories89


Packages

People30

Teams12


Projects

Pinned repositories

[fastify](#)


Fast and low overhead web framework, for Node.js

JavaScript ☆ 16.1k 🍴 1.2k

[fast-json-stringify](#)


2x faster than JSON.stringify()

JavaScript ☆ 1.9k 🍴 114

[fastify-cli](#)


Run a Fastify application with one command!

JavaScript ☆ 211 🍴 59

[help](#)


Need help with Fastify? File an Issue here.

☆ 26 🍴 2

[fastify-helmet](#)

Important security headers for Fastify

JavaScript ☆ 122 🍴 18

[point-of-view](#)

Template rendering plugin for Fastify

JavaScript ☆ 125 🍴 52

github.com/fastify

# Thank You!



## Ethan Arrowood

Software Engineer II, Microsoft

 @ArrowoodTech