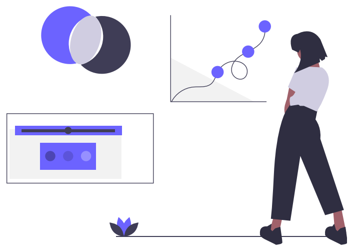
Safely Handling Dynamic Data with TypeScript

Ethan Arrowood - @ArrowoodTech

Handeling data



- API routes
- Forms
- Authentication
- Environment Variables

Payload Record - JSON Object

```json
{
  "id": "abc123",
  "name": "Sarah Jones",
  "employed": true,
  "company": "Microsoft",
  "age": 30,
  "projects": [
    "calculator",
    "todo",
    "blog"
  ]
}
```

API Route

```
fastify.post(
  '/add-user',
  async (request, response) => {
    const { body } = request
  }
)
```

What type is *body* ?

```
Record<T>

object

any
```

unknown

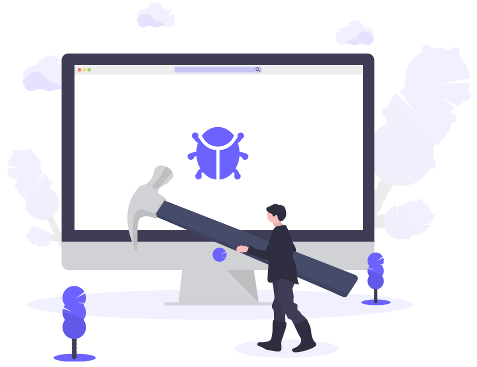
```
{
  "id": "abc123",
  "name": "Sarah Jones",
  // ...
}
```

```
fastify.post(
  '/add-user',
  async (request, response) => {
    const { body } = request
    // body is unknown
    body.name
  }
)
```

What is *body.name* ?

`string`

`// Object is of type 'unknown'. (2571)`

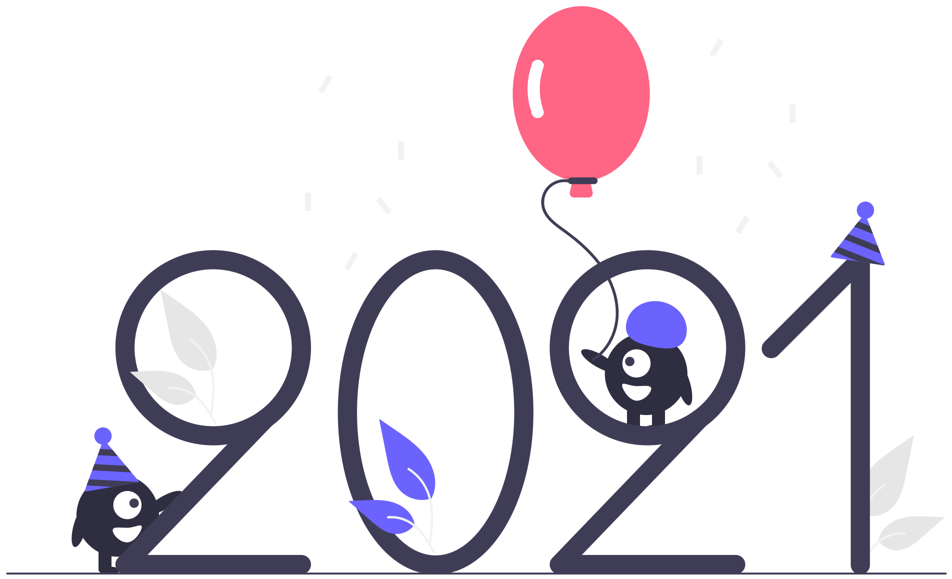


JSON Schema

JSON Schema

```
{
  "type": "object",
  "properties": {
    "id": { "type": "string" },
    "name": { "type": "string" },
    "employed": { "type": "boolean" },
    "company": { "type": "string" },
    "age": { "type": "number" },
    "projects": {
      "type": "array",
      "items": { "type": "string" }
    }
  },
  "required": [ "id", "name" ],
  "additionalProperties": false
}
```

Generally, JSON Schema is intended for validation



TypeBox

```
npm install @sinclair/typebox
```

```
import { Static, Type } from '@sinclair/typebox'
const T = Type.String() /* const T = { "type": "string" } */
type T = Static<typeof T> /* type T = string */
```

JSON Schema + TypeBox

```
const BodySchema = Type.Object({
  id: Type.String(),
  name: Type.String(),
  employed: Type.Optional(Type.Boolean()),
  company: Type.Optional(Type.String()),
  age: Type.Optional(Type.Number()),
  projects: Type.Optional(
    Type.Array(Type.String())
  )
})

type BodySchema = Static<typeof BodySchema>

fastify.post<{ Body: BodySchema }>(
  '/add-user',
  { schema: { body: BodySchema }},
  async (request, response) => {
    const { body } = request
    // type body = BodySchema
    // type body.name = string
    // type body.age = number | undefined
  }
)
```

JSON Schema + TypeBox + Fastify

It gets better!

Undici + AJV

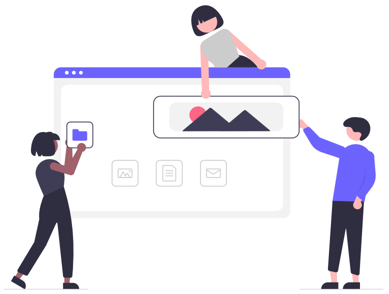
```typescript
const AddUserResponseSchema = Type.Object({
  message: Type.String()
})

type AddUserResponseSchema = Static<typeof AddUserResponseSchema>

function isAddUserResponse (responseData: unknown):
  responseData is AddUserResponseSchema {
  const ajv = new Ajv().addKeyword('kind')

  return ajv.validate(AddUserResponseSchema, responseData)
}

const user: UserSchema = {
  id: '123',
  name: 'Clippy'
}

const { body } = await undici.request(
  `http://localhost:${PORT}/add-user`,
  {
    method: 'POST',
    body: JSON.stringify(user),
    headers: { 'content-type': 'application/json' }
  }
)

const data = await readBody(body)

const addUserResponse = JSON.parse(data)

if (isAddUserResponse(addUserResponse)) {
  console.log(addUserResponse.message) // 🎉
}
```

Run it live!

Shoutout to unDraw

https://undraw.co/

Built with highlight.js & tmcw/big



https://highlightjs.org/
https://github.com/tmcw/big

Thank You!

Ethan Arrowood
Twitter: @ArrowoodTech
GitHub: @Ethan-Arrowood