A quest for optimal HTTP client DX

WARSAW IT DAYS

A lecture selected by a Program Council consisting of recognized leaders in the IT and Data Science field.

Warsaw,
04.04.2025 - 05.04.2025

OFFICIAL LECTURE OF THE WARSAW IT DAYS

Igor Savin

Principal Software Engineer at Lokalise

Event organized by: Academic Partners Foundation

whoami

Igor Savin aka **kibertoad**

Principal Software
Engineer
Lokalise

- Member of **fastify** organization
- Lead maintainer of **knex.js**
- Presented “Advanced caching in Node.js”
- Author of:
  - **message-queue-toolkit**
  - **node-service-template**
  - **layered-loader**
  - **opinionated-machine**
  - **toad-scheduler**
  - **@lokalise/api-contracts**
  - **@lokalise/backend-http-client**

Reinventing bicycles 101

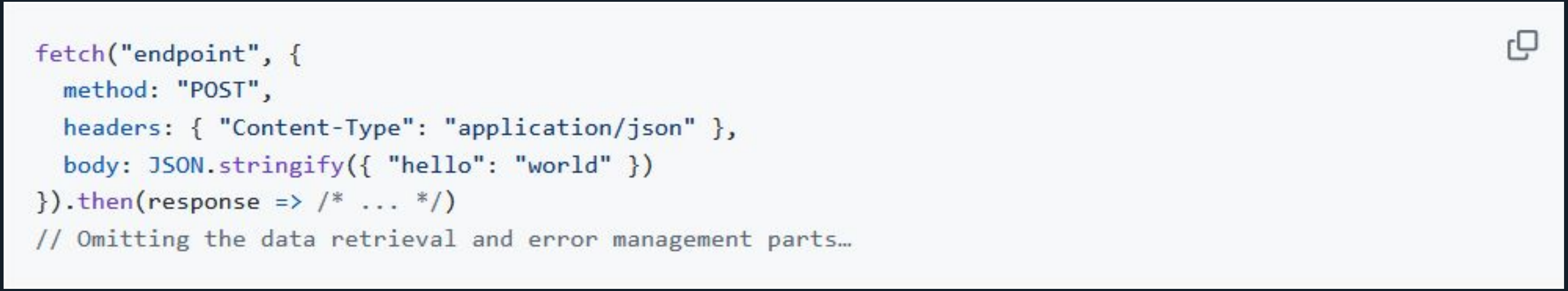
- "I'll just use fetch" - number 1 source of solving already solved problems in every frontend application since 2015.
- Boilerplate-heavy JSON handling;
- Boilerplate-heavy error handling;
- No middleware support.

Leveling up your HTTP client game

Level “Basic”

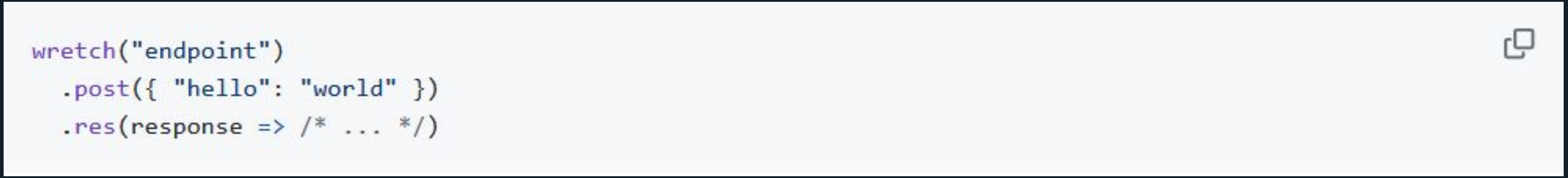
Eliminating boilerplate: JSON requests

fetch



```
fetch("endpoint", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ "hello": "world" })
}).then(response => /* ... */)
// Omitting the data retrieval and error management parts…
```

wretch



```
wretch("endpoint")
  .post({ "hello": "world" })
  .res(response => /* ... */)
```

Eliminating boilerplate: JSON responses

fetch

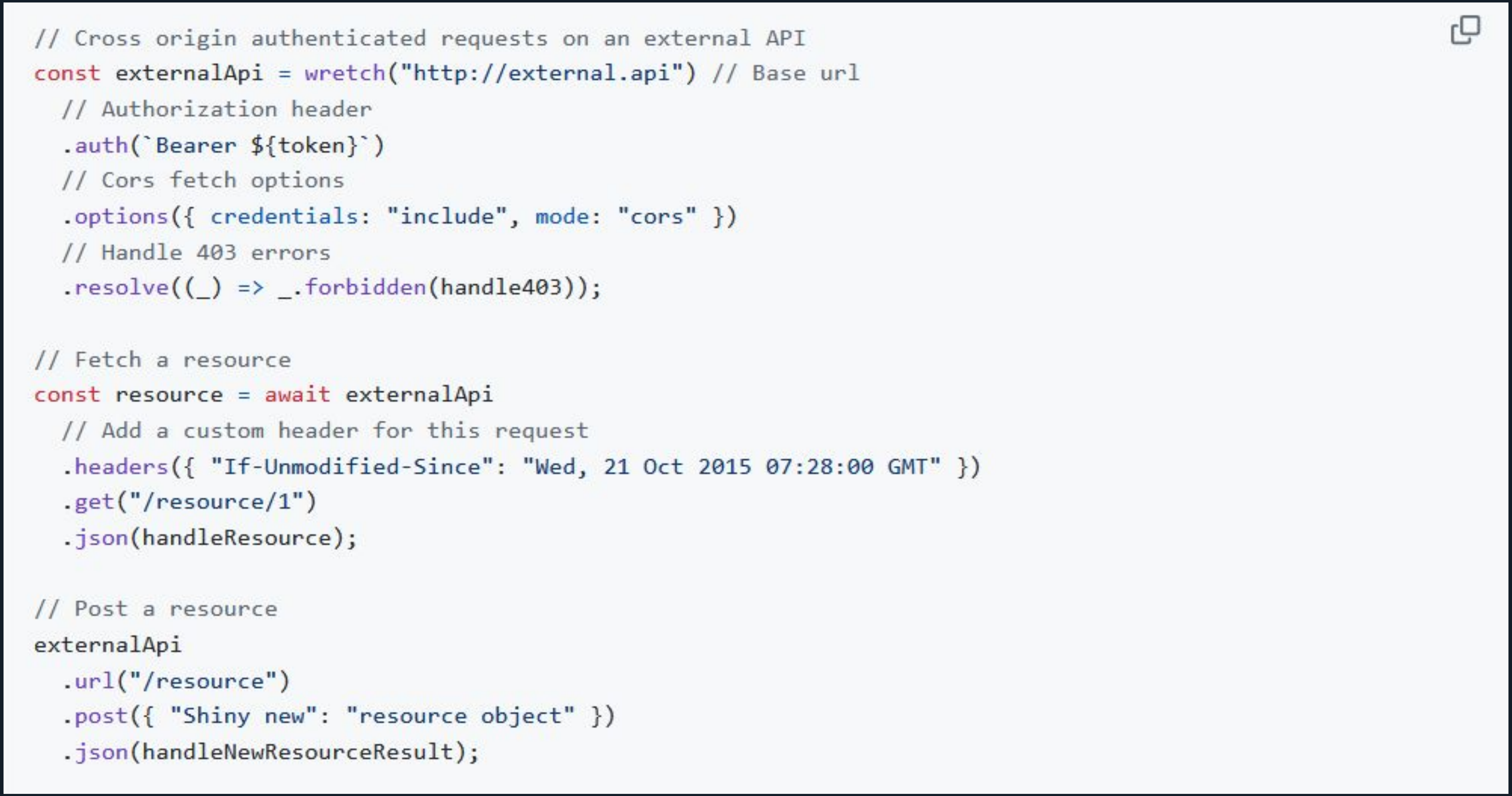
```
fetch("examples/example.json")
  .then(response => response.json())
  .then(json => {
    //Do stuff with the parsed json
  });
```

wretch

```
// Use .res for the raw response, .text for raw text, .json for json, .blob for a blob ...
wretch("examples/example.json")
  .get()
  .json(json => {
    // Do stuff with the parsed json
  });
```

Eliminating boilerplate: reuse

wretch

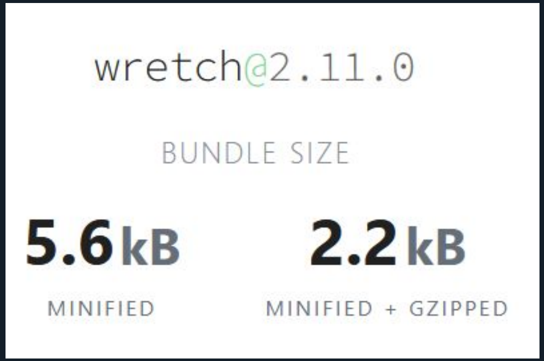


```
// Cross origin authenticated requests on an external API
const externalApi = wretch("http://external.api") // Base url
  // Authorization header
  .auth(`Bearer ${token}`)
  // Cors fetch options
  .options({ credentials: "include", mode: "cors" })
  // Handle 403 errors
  .resolve((_) => _.forbidden(handle403));

// Fetch a resource
const resource = await externalApi
  // Add a custom header for this request
  .headers({ "If-Unmodified-Since": "Wed, 21 Oct 2015 07:28:00 GMT" })
  .get("/resource/1")
  .json(handleResource);

// Post a resource
externalApi
  .url("/resource")
  .post({ "Shiny new": "resource object" })
  .json(handleNewResourceResult);
```

But my bundle size!..

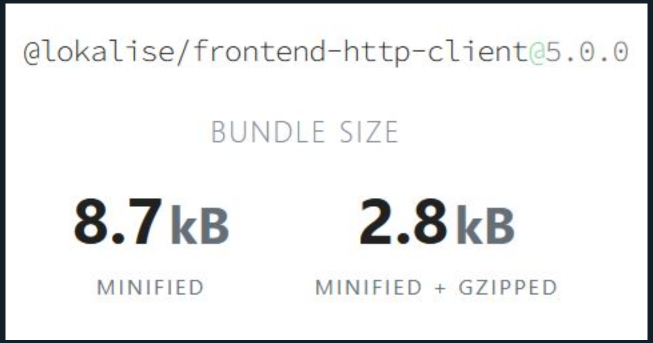


wretch@2.11.0

BUNDLE SIZE

5.6kB MINIFIED

2.2kB MINIFIED + GZIPPED

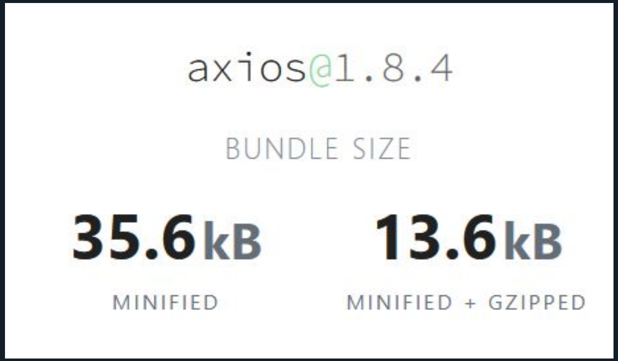


@lokalise/frontend-http-client@5.0.0

BUNDLE SIZE

8.7kB MINIFIED

2.8kB MINIFIED + GZIPPED



axios@1.8.4

BUNDLE SIZE

35.6kB MINIFIED

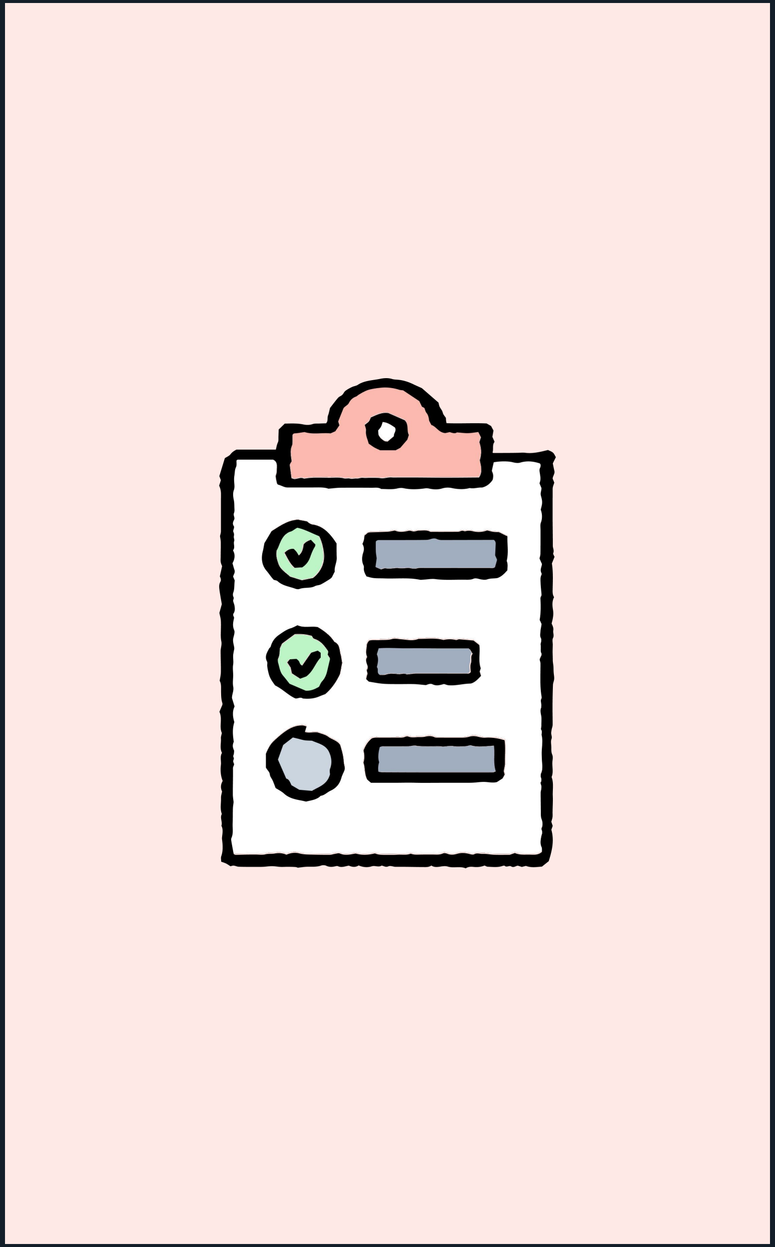
13.6kB MINIFIED + GZIPPED

Leveling up your HTTP client game

Level “Advanced”

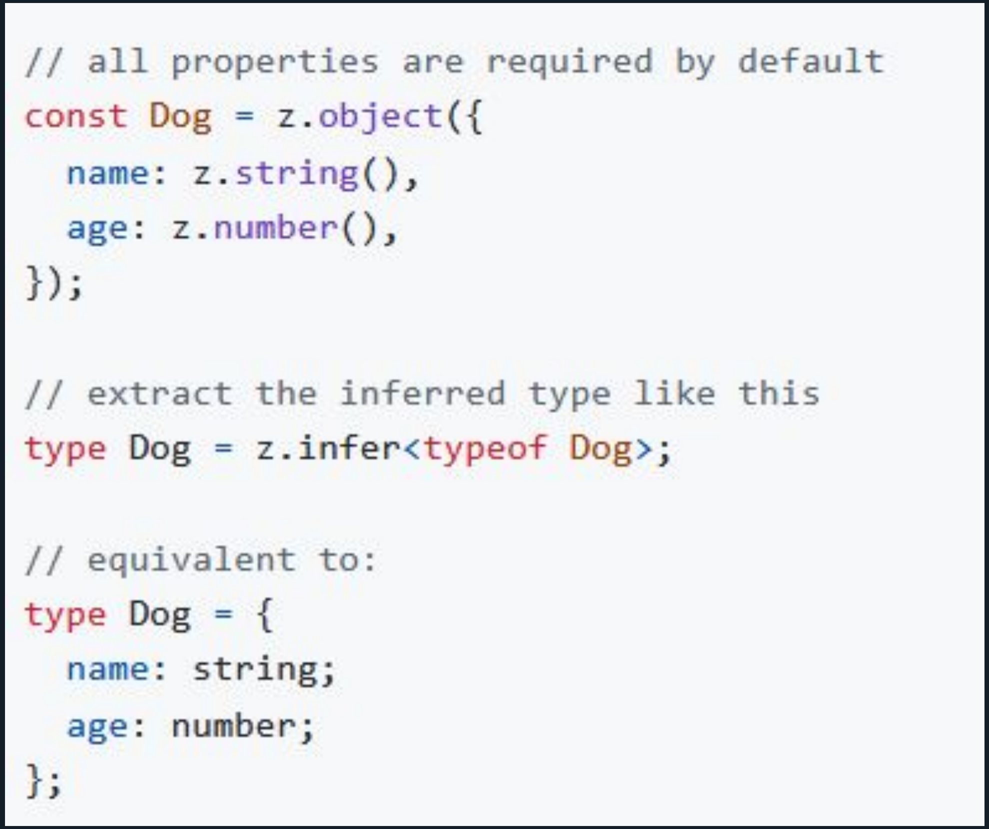
Validation, schemas and type-safety

- **Validation schemas:**
  - Runtime validation
  - Compilation time TypeScript types
- **Type-safety:**
  - Type-checks at compilation time catch problems before they happen
  - Runtime validation catches deviations from expected state during boundary transitions



Validation schemas

zod

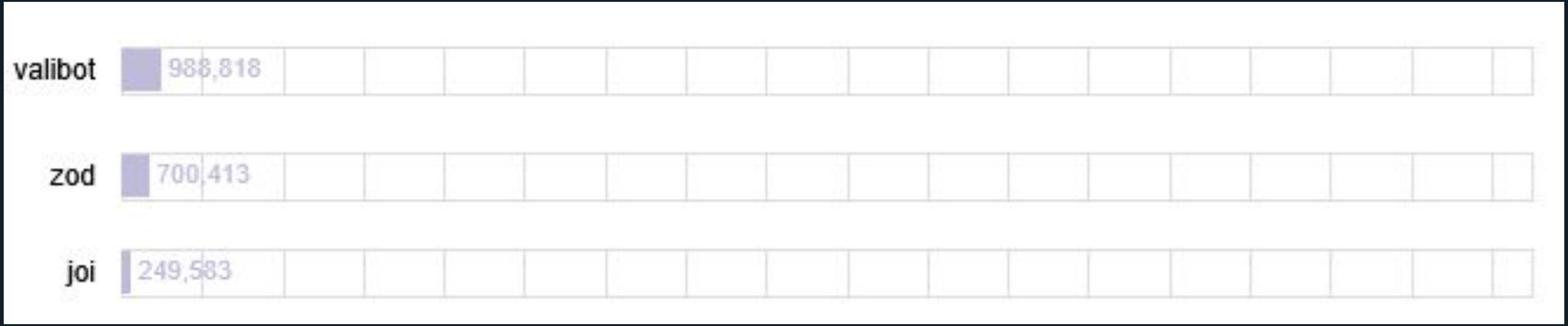


```
// all properties are required by default
const Dog = z.object({
  name: z.string(),
  age: z.number(),
});

// extract the inferred type like this
type Dog = z.infer<typeof Dog>;

// equivalent to:
type Dog = {
  name: string;
  age: number;
};
```

Benchmarks for popular schema-based parsing libraries



https://github.com/moltar/typescript-runtime-type-benchmarks

Leveling up your HTTP client game
Level “Expert”

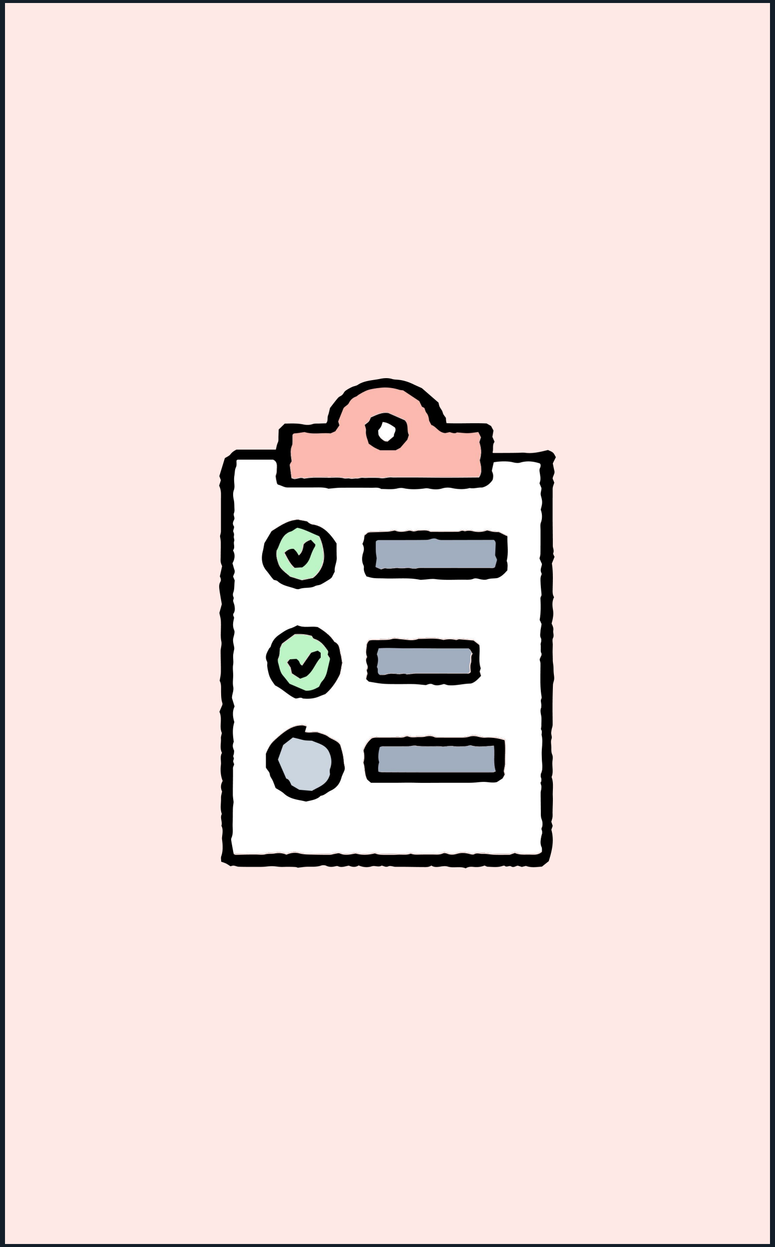
End-to-end contracts

Define complete contracts for your endpoints:

- HTTP request method
- Request, response, query, path params

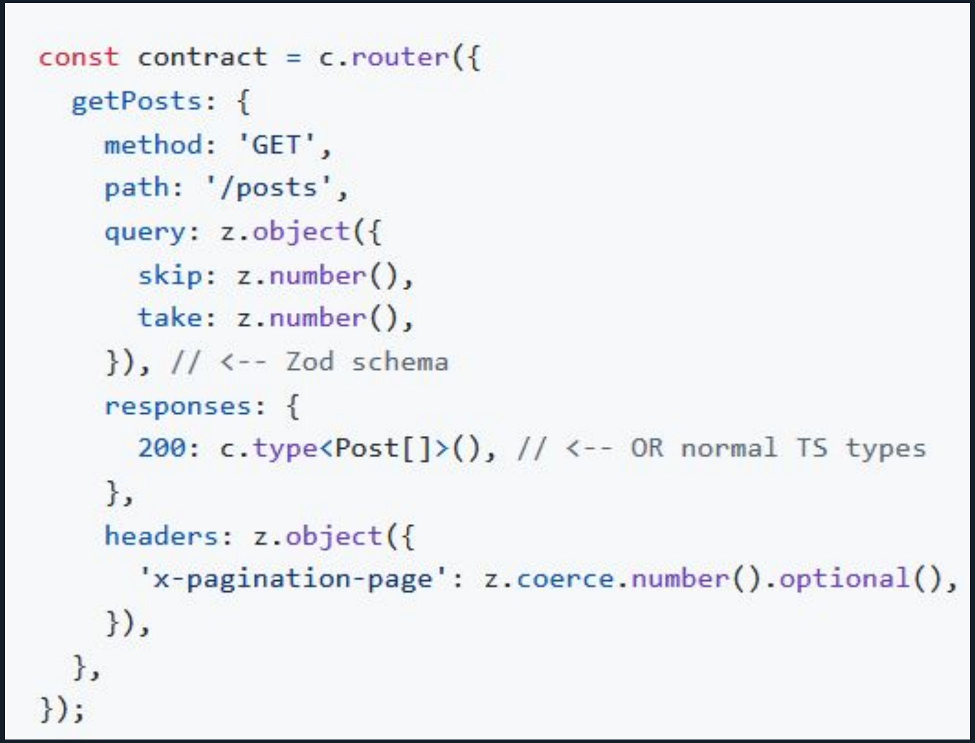
Use for:

- Controller definition on the backend
- Request validation and type inference on backend & frontend
- Type-safe mocking



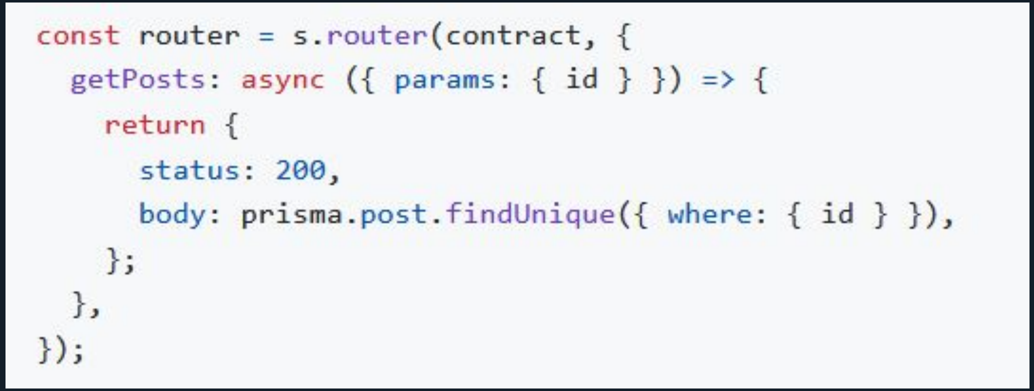
TS-REST

Contract definition



```
const contract = c.router({
  getPosts: {
    method: 'GET',
    path: '/posts',
    query: z.object({
      skip: z.number(),
      take: z.number(),
    }), // <-- Zod schema
    responses: {
      200: c.type<Post[]>(), // <-- OR normal TS types
    },
    headers: z.object({
      'x-pagination-page': z.coerce.number().optional(),
    }),
  },
});
```

Controller definition



```
const router = s.router(contract, {
  getPosts: async ({ params: { id } }) => {
    return {
      status: 200,
      body: prisma.post.findUnique({ where: { id } }),
    };
  },
});
```

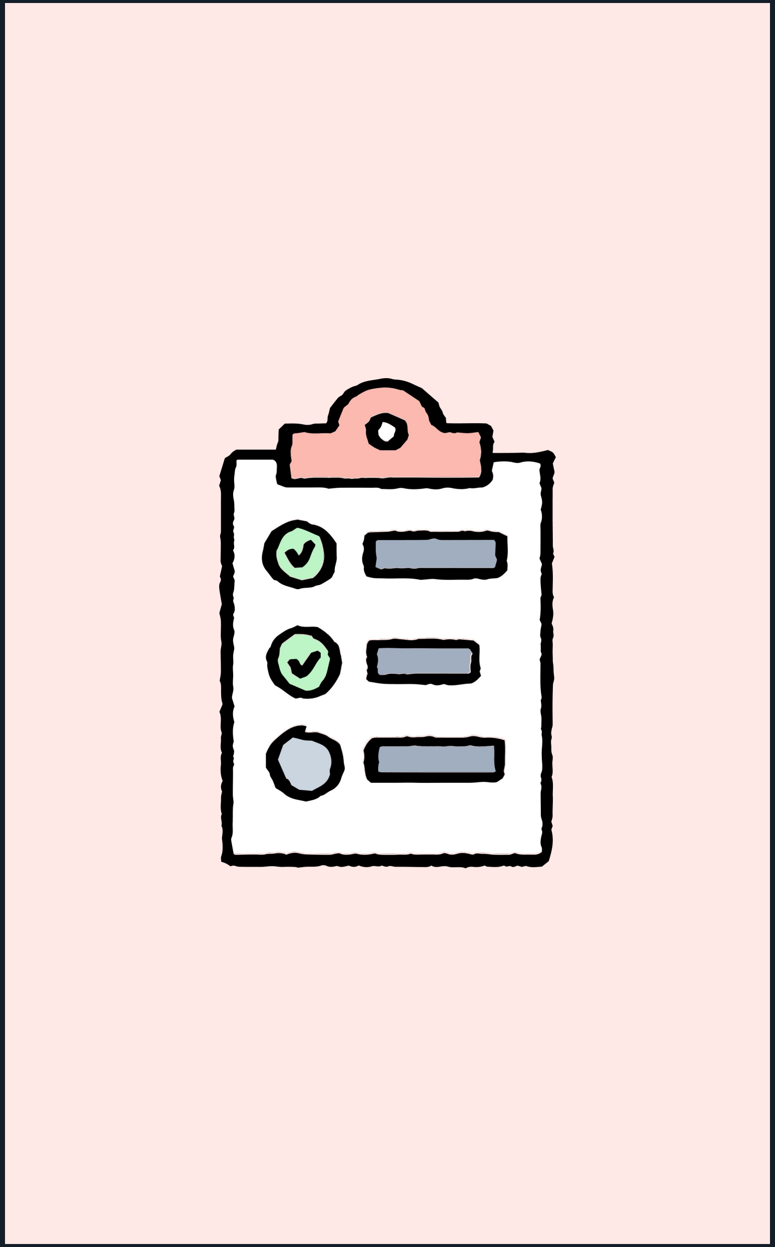
Leveling up your HTTP client game

Level “Galaxy Brain”

Best-practices-as-a-code

Enforce best practices via wrapper:

- Mandatory schemas?
- Mandatory contracts?
- Mandatory extended metadata for debugging purposes?
- Specific logging and error handling?



@lokalise/api-contracts

Demo time!

https://github.com/kibertoad/a-quest-for-http-client-dx

Thank you for watching!

Remember to leave your questions and rate the presentation in the section below.

WARSAW IT DAYS

A lecture selected by a Program Council consisting of recognized leaders in the IT and Data Science field.

Warsaw,
04.04.2025 - 05.04.2025

OFFICIAL LECTURE OF THE WARSAW IT DAYS

ACADEMIC PARTNERS

Feedback

Zeskanuj kod i zostaw swoją opinię

WARSZAWSKIE DNI INFORMATYKI

A Quest for Optimal HTTP Client DX

Igor Savin

https://warszawskiedniinformatyki.pl/user.html#!/lecture/WDI25-c6a1/rate