CIKLUM | SPEAKER'S CORNER

June, 27 | 16:30 (CET) | English

Effect: The Missing TypeScript Standard Library

Tomáš Horáček

Principal Technology Lead for Web & Mobile at Ciklum

Experiences of tomorrow. Engineered together.

CIKLUM

We transform how people experience the business. All through next generation technology.

What we do:

- Product Engineering
- Intelligent Automation
- Data & Analytics

2002 founded

4000+ professionals

20+ offices

300+ clients

Leading companies choose us:

JUST EAT Takeaway.com · METRO MARKETS · eToro · ZURICH · pay · GreensKANTARRETAIL · dacadoo

CIKLUM

Meet the speaker

- Works with JavaScript since 2005
- Experienced with many web/mobile-related languages since then: Python, Ruby, Objective-C, Swift, Flow, Elm, and TypeScript
- Keeps his primary focus on TypeScript, React, React Native and functional programming since 2015

Tomáš Horáček
Principal Technology
Lead for Web & Mobile

Table of Contents

CIKLUM

CIKLUM

1️⃣ What is Effect 🤔

What is Effect 🤔

CIKLUM

- TypeScript library
- useful primitives:
  - 🧰 functional programming data primitives: `Option`, `Either`, `Chunk`, `Ref`, `Array`, `Record`, `Order`, `SortedMap`, `SortedSet`, `Queue`, ...
  - 🛟 type safe error handling
  - 👷 concurrency, fibers, observability, scheduling, interruptibility
  - 🥳 "dependency injection"
- helps build apps that are:
  - ✅ reliable
  - ✅ reusable
  - ✅ testable
  - ✅ maintainable
  - ✅ scalable


JS PRIMITIVES, CALLBACKS
LODASH, PROMISSES
RAMDA, RXJS
EFFECT
imgflip.com

CIKLUM

2️⃣ Effect Basics 🎒

🎒 Basics: Effect Type

CIKLUM

`Effect<ASuccess, Error, Requirements>`

... **immutable** representation of **lazy** program

- `ASuccess`: "returned" value
- `Error`: expected error(s)
- `Requirements`: contextual requirement(s)

🎒 Basics: Creating Effect

CIKLUM

`Effect<ASuccess, Error, Requirements>`

```ts
import { Effect } from 'effect';

const success: Effect.Effect<number>
  = Effect.succeed(42);

const failure = Effect.fail('Error');

const parse = (data: string) => Effect.try(
  () => JSON.parse(data)
);
```

😝 ~~ `Promise<ASuccess>`

```ts
const success = Promise.resolve(42);

const failure = Promise.reject('Error');

const parse = (data: string) => new Promise<any>(
  (resolve) => resolve(JSON.parse(data))
);
```

🎒 Basics: 🏃 Running ➡️ Sync Effect

CIKLUM

`Effect`

```
import { Effect } from "effect";

const parse = (data: string) => Effect.try(
  () => JSON.parse(data)
);

const program = parse('{"hello": "world"}');

console.log(Effect.runSync(program));
```

😝 ~~ `React`

```
import { createRoot } from 'react-dom/client';

const App = ({ data }: { data: string}) => (
  <div>Hello, {data}!</div>
);

const program = <App data="World" />;

const root = createRoot(
  document.getElementById('root')!
);
root.render(program);
```

🎒 Basics: 🏃 Running 🕰️ Async Effect

CIKLUM

```ts
import { Effect } from "effect";

const delay = (millis: number) =>
  Effect.tryPromise(
    () =>
      new Promise<string>((resolve) => {
        setTimeout(() => resolve("Done"), millis);
      }),
  );

const program = delay(200);

Effect.runSync(program) // 👎 throws an error
console.log(await Effect.runPromise(program)); // 👍
```

🎒 Basics: 🪈 `pipe`

CIKLUM

```ts
import { pipe } from 'effect';

const result = pipe(
  1,
  a => a + 2, // 1 + 2 = 3
  b => b * 3, // 3 * 3 = 9
  c => { console.log(c); return c; },
  d => `result: ${d}`, // result: 9
);
```

```ts
import { Effect, pipe } from 'effect';

const program = pipe(
  Effect.succeed(1),
  Effect.map(a => a + 2),
  Effect.flatMap(b => Effect.succeed(b * 3)),
  Effect.tap(c => Effect.log(c)),
  Effect.map(d => `result: ${d}`),
);

const result = await Effect.runPromise(program);
```

CIKLUM

🎒 Basics: Mini Demo 01

github.com/heracek/github-stars-effect

```
pnpm basics01
```

🎒 Basics: 🧬 `Effect.gen`

CIKLUM

😝 ~~ `Promise`

```ts
import sleep from 'sleep-promise';


const asyncRandom = async (delay: number) => {
  const data = await fetchRandomInt(
    0,
    100
  );

  await sleep(delay)

  return data
};
```

`Effect`

```ts
import { Effect, Random } from 'effect';

const asyncRandom = (delay: number) =>
  Effect.gen(function* () {
    const data = yield* Random.nextIntBetween(
      0,
      100
    );

    yield* Effect.sleep(delay);

    return data;
  });
```

🎒 Basics: 🏎️ `Effect.all`

CIKLUM

😝 ~~ `Promise.all`

```ts
const asyncRandom = async (delay: number) => {
  // ...
};

const data = Promise.all([
  asyncRandom(2000),
  asyncRandom(2000),
  asyncRandom(2000),
  asyncRandom(2000),
  asyncRandom(2000),
]);
```

👎 `Promise`: eager execution

`Effect.all`

```ts
const asyncRandom = (delay: number) =>
  Effect.gen(function* () {
    // ...
  });

const data = Effect.all([
  asyncRandom(2000),
  asyncRandom(2000),
  asyncRandom(2000),
  asyncRandom(2000),
  asyncRandom(2000),
], { concurrency: 'unbounded' });
```

👍 `Effect`: lazy execution

```ts
Effect.all([...], { concurrency: 3 });
```

CIKLUM

🎒 Basics: Mini Demo 02

github.com/heracek/github-stars-effect

```
pnpm basics02
```

📚 Effect Docs

CIKLUM

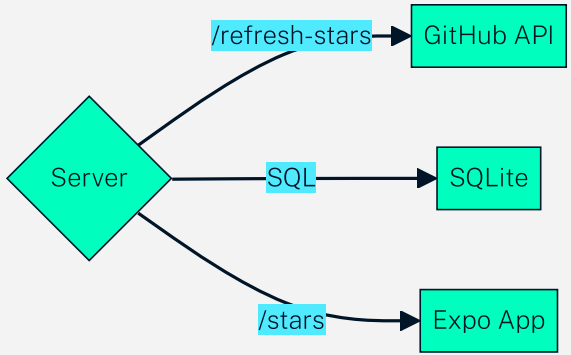
- 🤩 effect.website/docs
- 👨‍🏭 github.com/Effect-TS/effect/tree/main/packages
  - github.com/Effect-TS/effect/tree/main/packages/schema#readme
  - github.com/Effect-TS/effect/tree/main/packages/platform#readme
- 📺 youtube.com/@effect-ts
  - Effect Days 2024 conference videos
  - Effect Days 2024: Beginner / Intermediate Workshop: youtu.be/Lz2J1NBnHK4

CIKLUM

3️⃣ Live Coding 👨‍💻

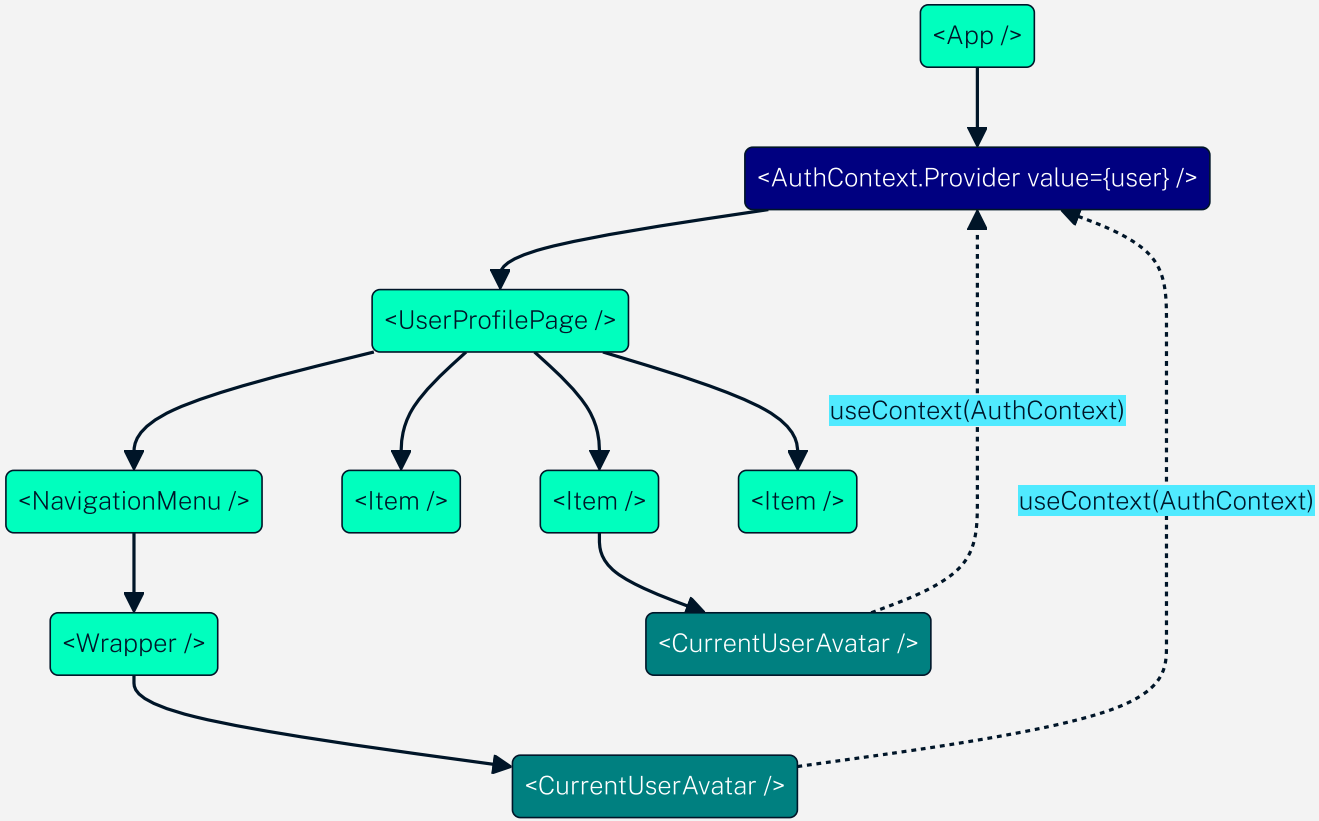
CIKLUM

👨‍💻 Live Coding: GitHub Stars App



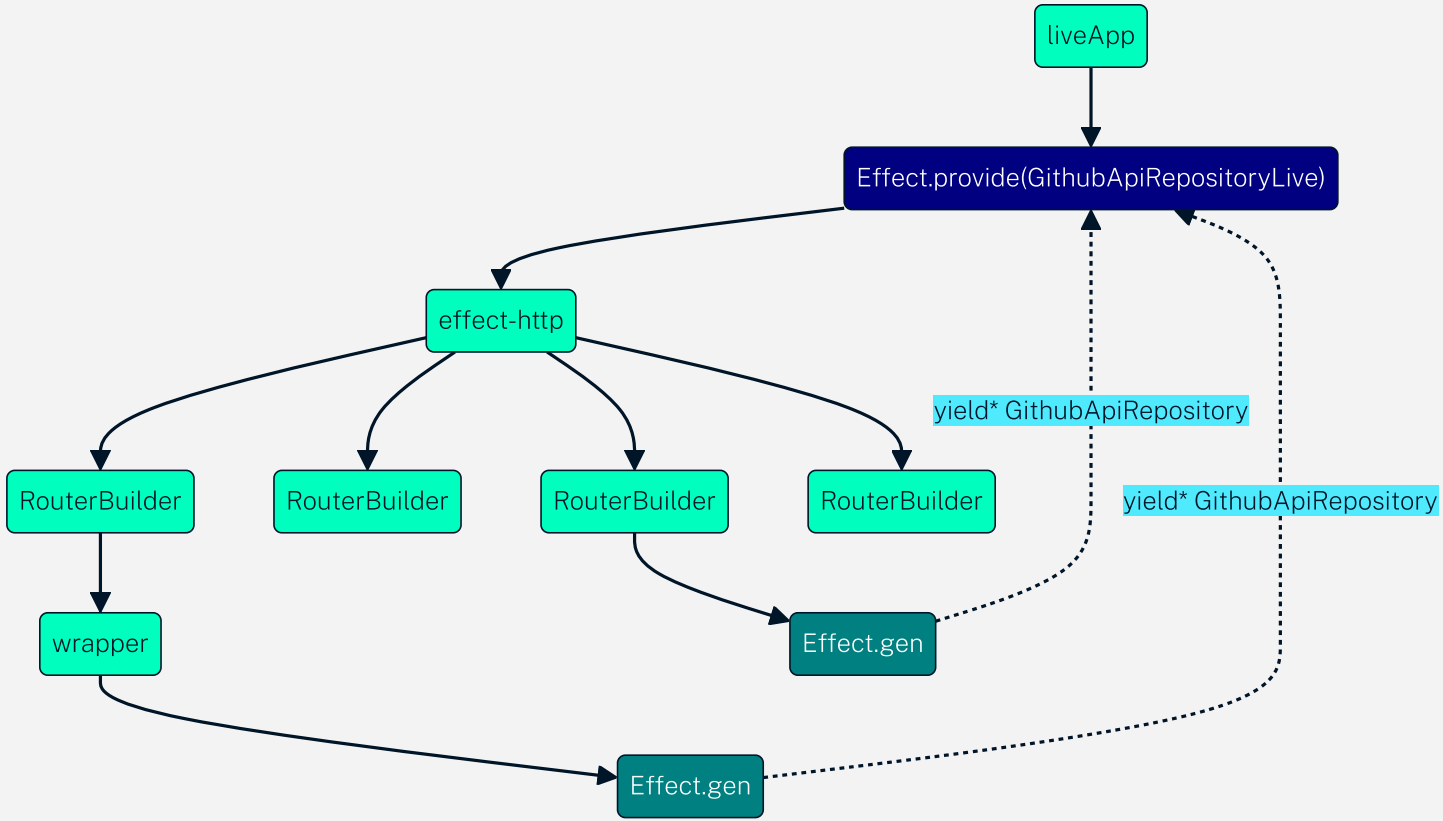
React Context

CIKLUM



🤩 Effect Context

CIKLUM



🤩 Effect Context

CIKLUM

```
import { Context, Effect, pipe } from 'effect';

class GithubApiRepository extends Context.Tag("GithubApiRepository")<
  GithubApiRepository,
  {
    getStarred: (options: { page: number }) => Effect.Effect<ResponseStarred>;
  }
>() {}

const program = Effect.gen(function* () {
  const repo = yield* GithubApiRepository;
  return yield* repo.getStarred({ page: 1 });
});

Effect.runPromise(pipe(
  program,
  Effect.provide(GithubApiRepositoryLive),
))
```

🤩 Effect Context

CIKLUM

```ts
import { Effect, Layer } from 'effect';

export const GithubApiRepositoryLive = Layer.effect(
  GithubApiRepository,
  Effect.gen(function* () {
    const getStarred = ({ page }: { page: number }) =>
        // TODO: HTTP fetch
        Effect.succeed<ResponseStarred>({})

    return GithubApiRepository.of({
      getStarred,
    });
  }),
);
```

CIKLUM

4️⃣ Closing Thoughts 💡

Closing Thoughts 💡

CIKLUM

- I ❤️ Effect
  - ✅ `effect` is production ready and API stable
  - 🏋️ looks hard, but ~1-5 days and your are productive (incremental adoption)
  - ❤️ Effect `Layer`
- `@effect/schema`, `@effect/platform`, ...
  - 🆗 less stable, but production ready
- Use it?
  - ✅ hobby projects
  - ✅ new projects (open-minded team)
  - 🤷 existing projects (it depends)
    - ✅ open-minded team & customer
    - ✅ proof of concept first
    - ❓ bundle size is an issue

CIKLUM

Questions?

Thank you!

github.com/heracek/github-stars-effect

toh@ciklum.com

CIKLUM

Share your feedback!

CIKLUM

Join our team