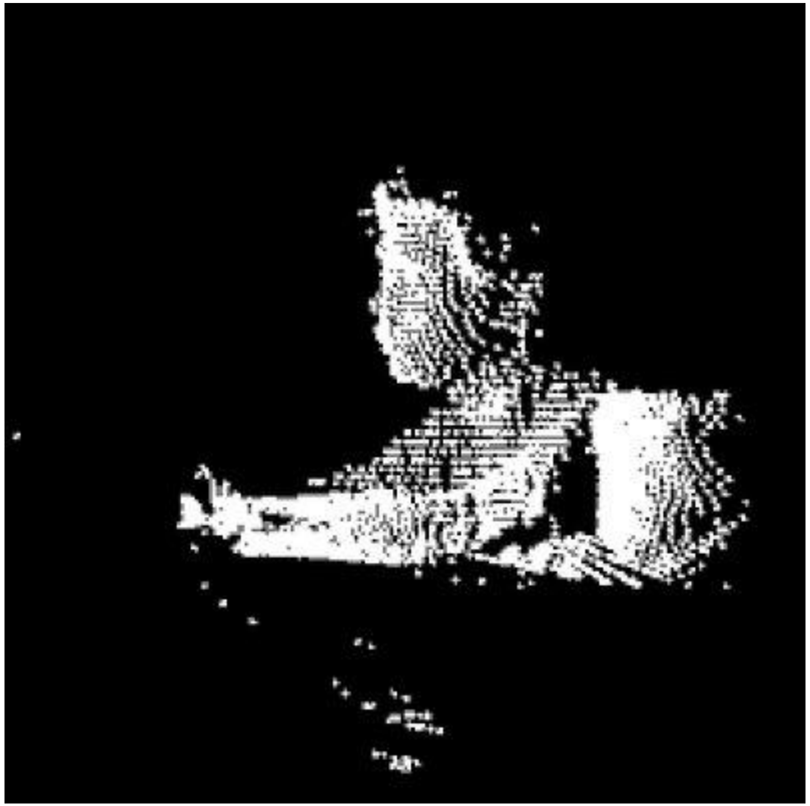
Hono + SvelteKit で楽々型安全！

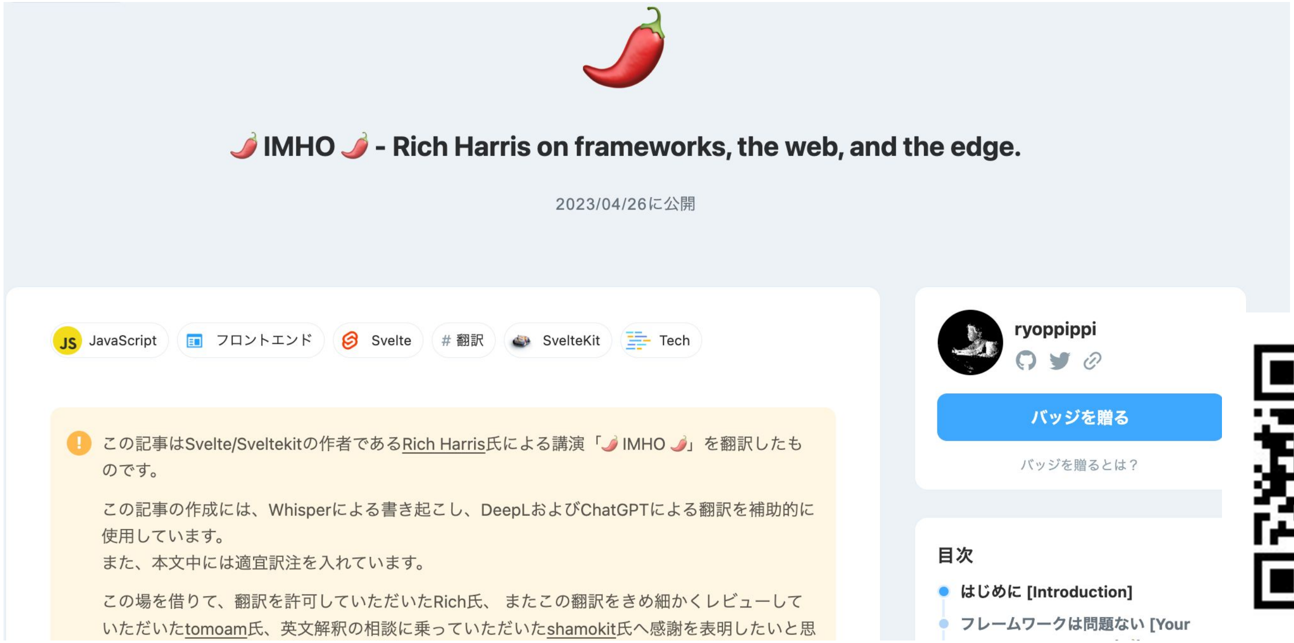
08/12/23 Svelte Japan Online Meetup
@ryoppippi

About me

- @ryoppippi
- フリーランス＆ 博士学生
- 2021年からSvelte大好きおじさん
- Webやったり、機械学習のPoC回してたりする
- この1年いくつかSvelteKit案件回しました
- Runeアツイ〜
- Neovim使いです！Neovimはいいぞ！



Zenn書いてます



🌶IMHO🌶 - Rich Harris on frameworks, the web, and the edge.

2023/04/26に公開

JavaScript　フロントエンド　Svelte　# 翻訳　SvelteKit　Tech

この記事はSvelte/Sveltekitの作者であるRich Harris氏による講演「🌶IMHO🌶」を翻訳したものです。

この記事の作成には、Whisperによる書き起こし、DeepLおよびChatGPTによる翻訳を補助的に使用しています。
また、本文中には適宜訳注を入れています。

この場を借りて、翻訳を許可していただいたRich氏、 またこの翻訳をきめ細かくレビューしていただいたtomoam氏、英文解釈の相談に乗っていただいたshamokit氏へ感謝を表明したいと思

ryoppippi

バッジを贈る

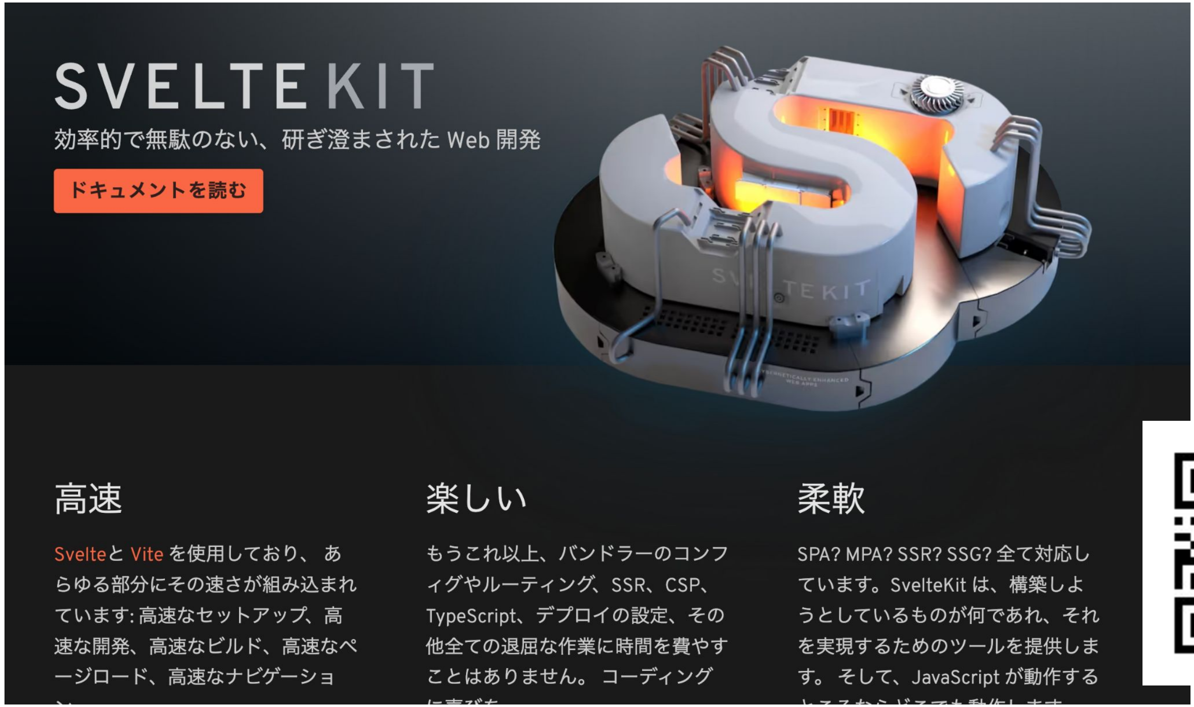
バッジを贈るとは？

目次

注意:この話はSvelteKit 2.0で状況が変わる可能性があります

注意:このスライドでは
Svelte5を多用しています

おさらい: SvelteKitって何？



SVELTE KIT

効率的で無駄のない、研ぎ澄まされた Web 開発

ドキュメントを読む

高速

Svelteと Vite を使用しており、あらゆる部分にその速さが組み込まれています: 高速なセットアップ、高速な開発、高速なビルド、高速なページロード、高速なナビゲーション

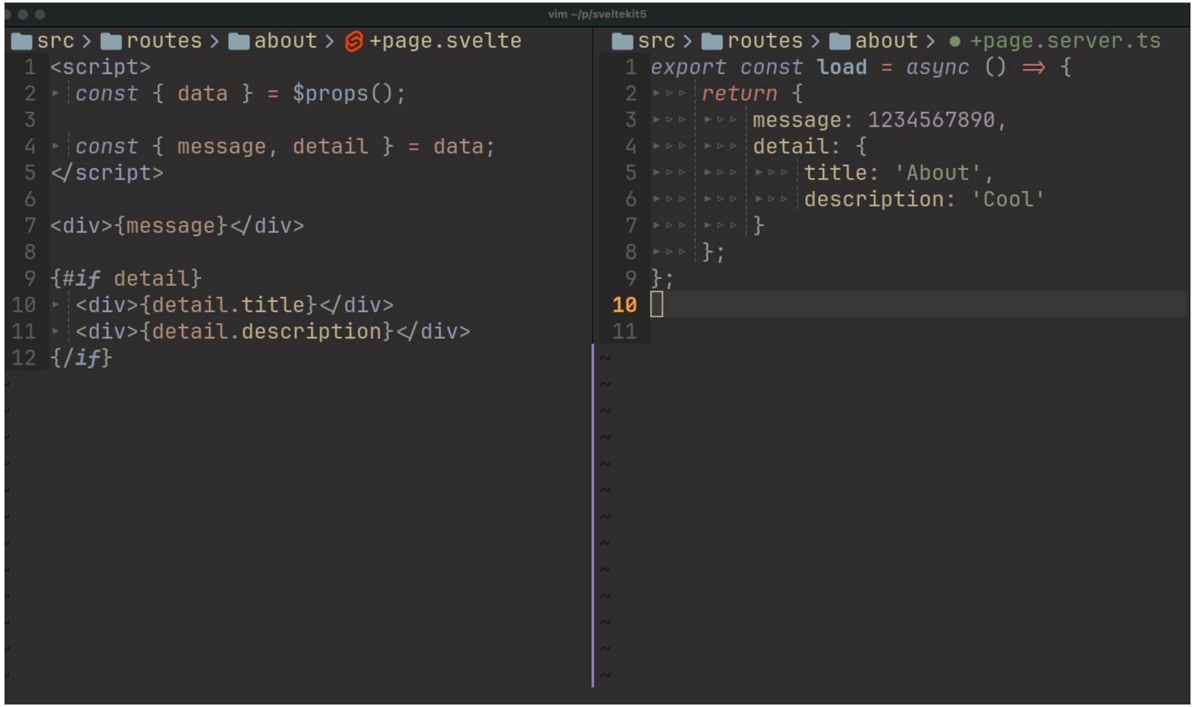
楽しい

もうこれ以上、バンドラーのコンフィグやルーティング、SSR、CSP、TypeScript、デプロイの設定、その他全ての退屈な作業に時間を費やすことはありません。コーディングに喜びを

柔軟

SPA? MPA? SSR? SSG? 全て対応しています。SvelteKit は、構築しようとしているものが何であれ、それを実現するためのツールを提供します。そして、JavaScript が動作するところならどこでも動作します

おさらい: SvelteKitのデータの流れ



src > routes > about > +page.svelte

```
<script>
  const { data } = $props();

  const { message, detail } = data;
</script>

<div>{message}</div>

{#if detail}
  <div>{detail.title}</div>
  <div>{detail.description}</div>
{/if}
```

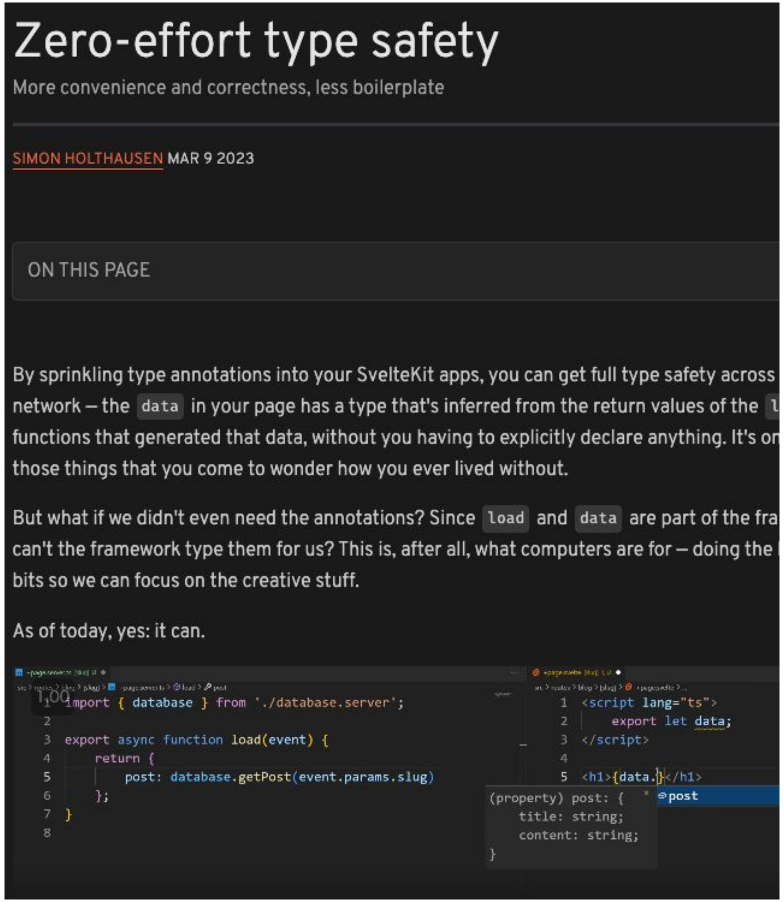
src > routes > about > +page.server.ts

```
export const load = async () => {
  return {
    message: 1234567890,
    detail: {
      title: 'About',
      description: 'Cool'
    }
  };
};
```

SvelteKitの型安全の話

SvelteKit + 型安全

- SvelteKitはfile base routing等規約で設計をガチガチに縛っている
- その制約の結果、ファイル間で型安全が**自動的に**保たれている
    - Devサーバが型生成を頑張っている
- SSRの型安全完璧/開発体験最強



Zero-effort type safety

More convenience and correctness, less boilerplate

SIMON HOLTHAUSEN MAR 9 2023

ON THIS PAGE

By sprinkling type annotations into your SvelteKit apps, you can get full type safety across
network – the `data` in your page has a type that's inferred from the return values of the `l`
functions that generated that data, without you having to explicitly declare anything. It's on
those things that you come to wonder how you ever lived without.

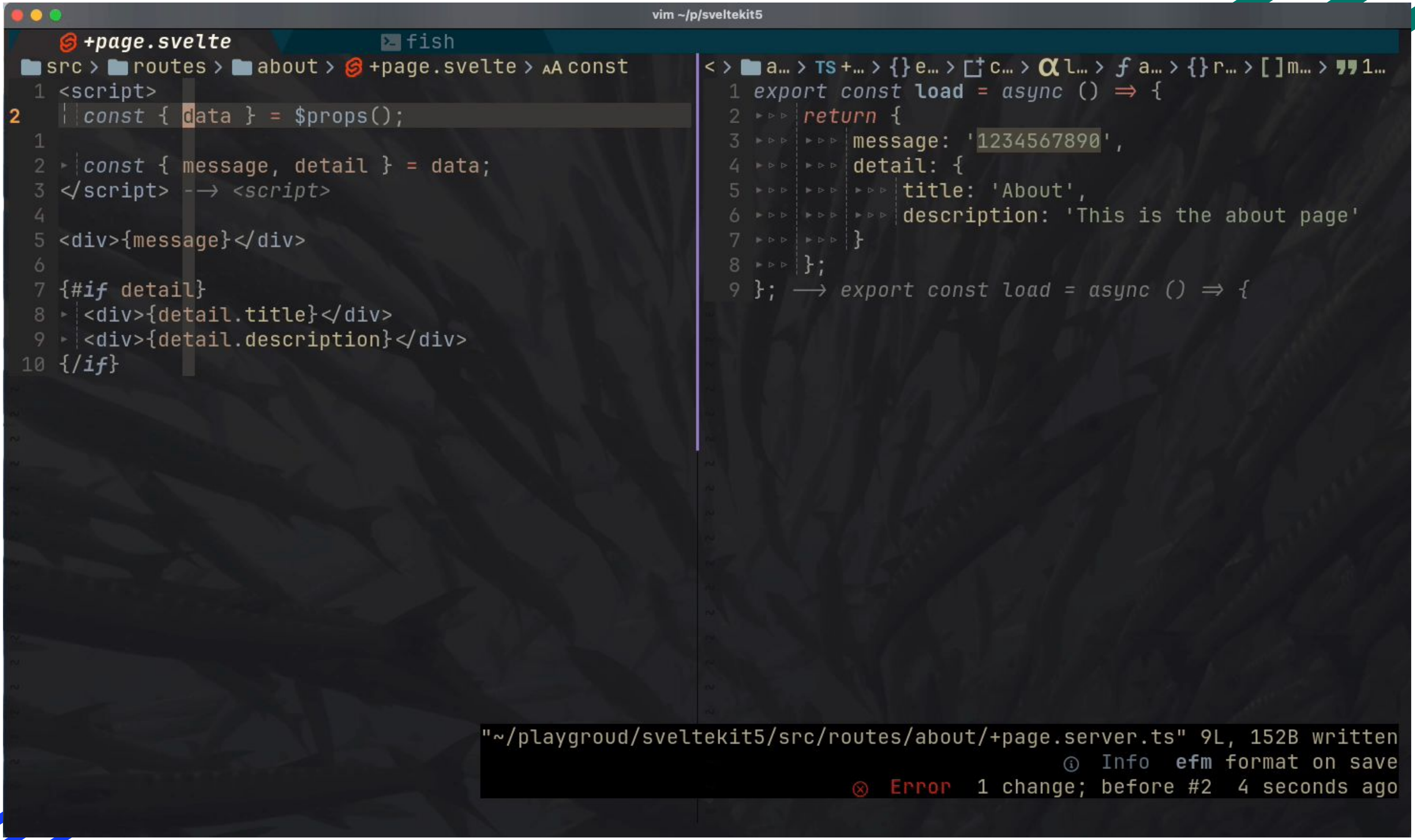
But what if we didn't even need the annotations? Since `load` and `data` are part of the fra
can't the framework type them for us? This is, after all, what computers are for – doing the
bits so we can focus on the creative stuff.

As of today, yes: it can.

```
import { database } from './database.server';

export async function load(event) {
    return {
        post: database.getPost(event.params.slug)
    };
}
```

```
<script lang="ts">
    export let data;
</script>

<h1>{data.}</h1>
```

```
(property) post: {
    title: string;
    content: string;
}
```



```
vim ~/p/sveltekit5
```

+page.svelte
fish
src > routes > about > +page.svelte > const
< > a… > TS +… > {} e… > c… > l… > a… > {} r… > [ ] m… > 1…

```
<script>
	const { data } = $props();

	const { message, detail } = data;
</script> --> <script>

<div>{message}</div>

{#if detail}
	<div>{detail.title}</div>
	<div>{detail.description}</div>
{/if}
```

```
export const load = async () => {
	return {
		message: '1234567890',
		detail: {
			title: 'About',
			description: 'This is the about page'
		}
	};
}; --> export const load = async () => {
```

```
"~/playgroud/sveltekit5/src/routes/about/+page.server.ts" 9L, 152B written
                                              Info  efm format on save
                               Error  1 change; before #2  4 seconds ago
```

だけど....

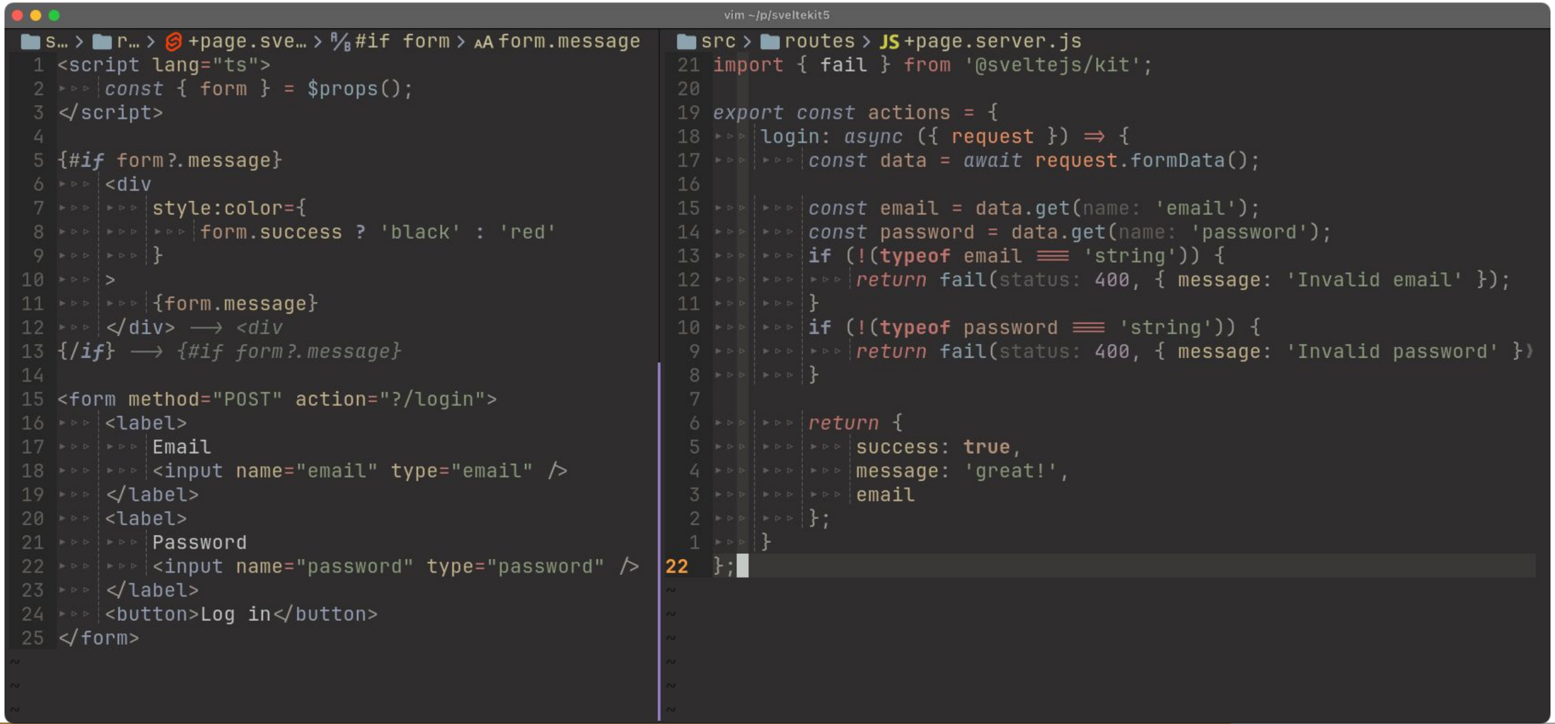
標準では型安全じゃない部分もあります

- Form Action
- API Endpoint

標準では型安全じゃない部分もあります

- Form Action
- API Endpoint

Form Action



vim ~/p/sveltekit5

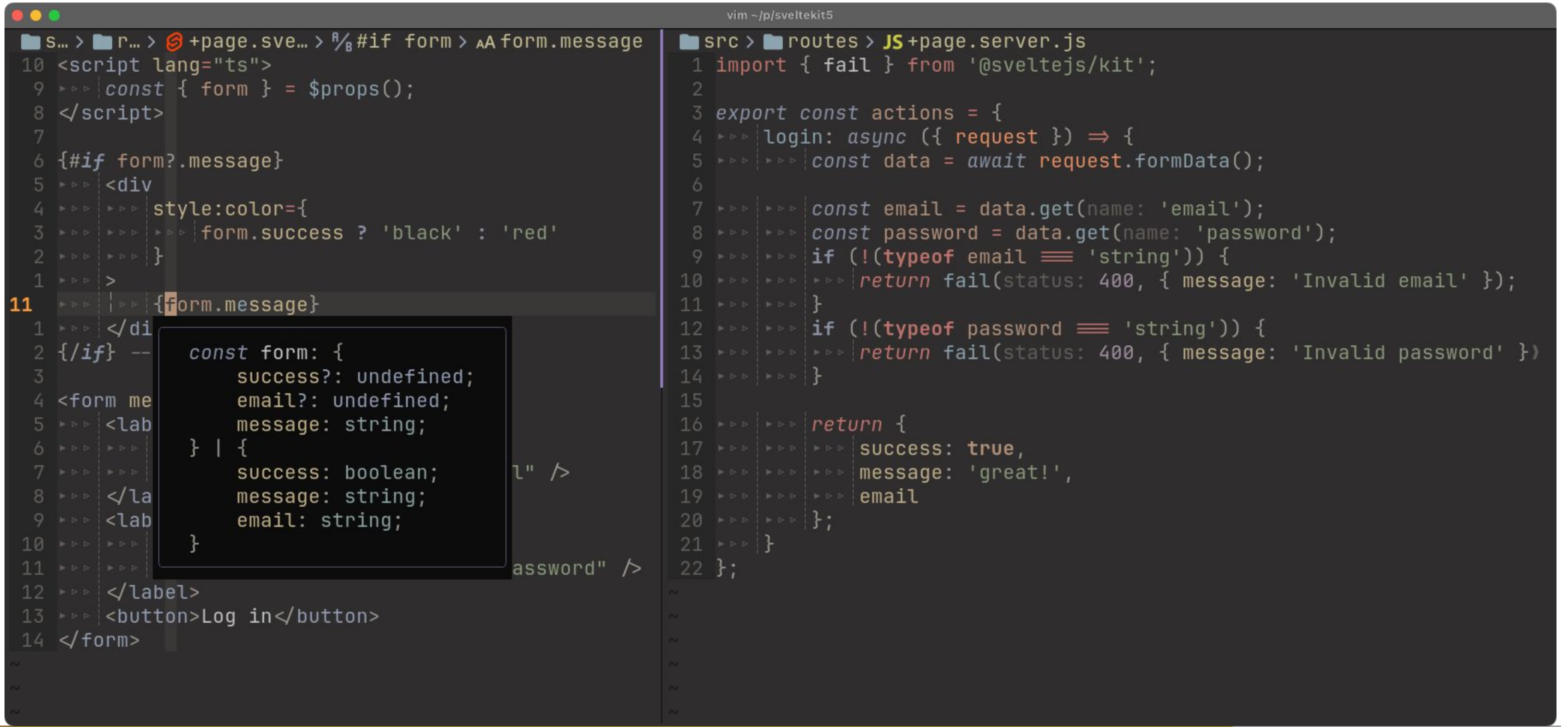
s… > r… > +page.sve… > #if form > form.message

```svelte
<script lang="ts">
    const { form } = $props();
</script>

{#if form?.message}
    <div
        style:color={
            form.success ? 'black' : 'red'
        }
    >
        {form.message}
    </div> ⟶ <div
{/if} ⟶ {#if form?.message}

<form method="POST" action="?/login">
    <label>
        Email
        <input name="email" type="email" />
    </label>
    <label>
        Password
        <input name="password" type="password" />
    </label>
    <button>Log in</button>
</form>
```

src > routes > +page.server.js

```js
import { fail } from '@sveltejs/kit';

export const actions = {
    login: async ({ request }) => {
        const data = await request.formData();

        const email = data.get(name: 'email');
        const password = data.get(name: 'password');
        if (!(typeof email === 'string')) {
            return fail(status: 400, { message: 'Invalid email' });
        }
        if (!(typeof password === 'string')) {
            return fail(status: 400, { message: 'Invalid password' })
        }

        return {
            success: true,
            message: 'great!',
            email
        };
    }
};
```

Form Action


vim ~/p/sveltekit5
s… > r… > +page.sve… > #if form > form.message
src > routes > +page.server.js

```
<script lang="ts">
	const { form } = $props();
</script>

{#if form?.message}
	<div
		style:color={
			form.success ? 'black' : 'red'
		}
	>
		{form.message}
	</di
{/if} --

<form me
	<lab
		                                l" />
	</la
	<lab

		                                assword" />
	</label>
	<button>Log in</button>
</form>
```

```
const form: {
    success?: undefined;
    email?: undefined;
    message: string;
} | {
    success: boolean;
    message: string;
    email: string;
}
```

```
import { fail } from '@sveltejs/kit';

export const actions = {
	login: async ({ request }) => {
		const data = await request.formData();

		const email = data.get(name: 'email');
		const password = data.get(name: 'password');
		if (!(typeof email === 'string')) {
			return fail(status: 400, { message: 'Invalid email' });
		}
		if (!(typeof password === 'string')) {
			return fail(status: 400, { message: 'Invalid password' })
		}

		return {
			success: true,
			message: 'great!',
			email
		};
	}
};
```

Form Action

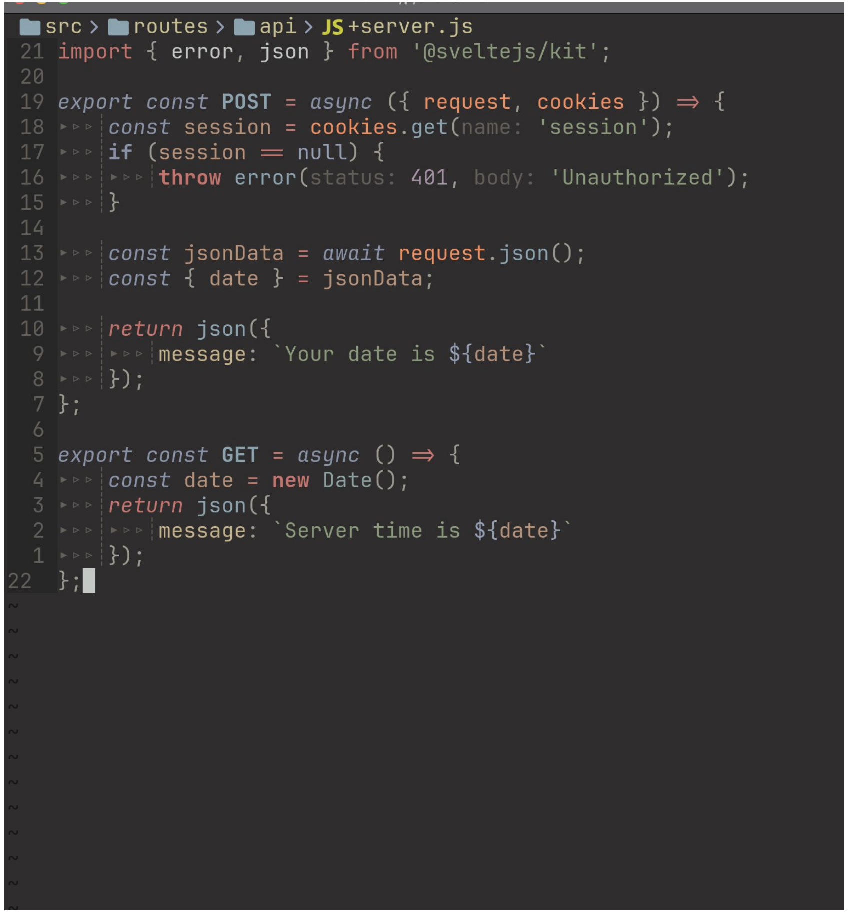
- 戻り値は型安全だが、引数が型安全じゃない
- Superforms等のライブラリに頼れば型安全になる
    - 詳しくは KenjiroKubotaさんの発表参照

標準では型安全じゃない部分もあります

- Form Action
- API Endpoint

Endpointって何？

- Rest APIが作れるやつ
- GET とか POST とかを作れる
- src/route/…/+server.js で定義できる


src > routes > api > JS +server.js

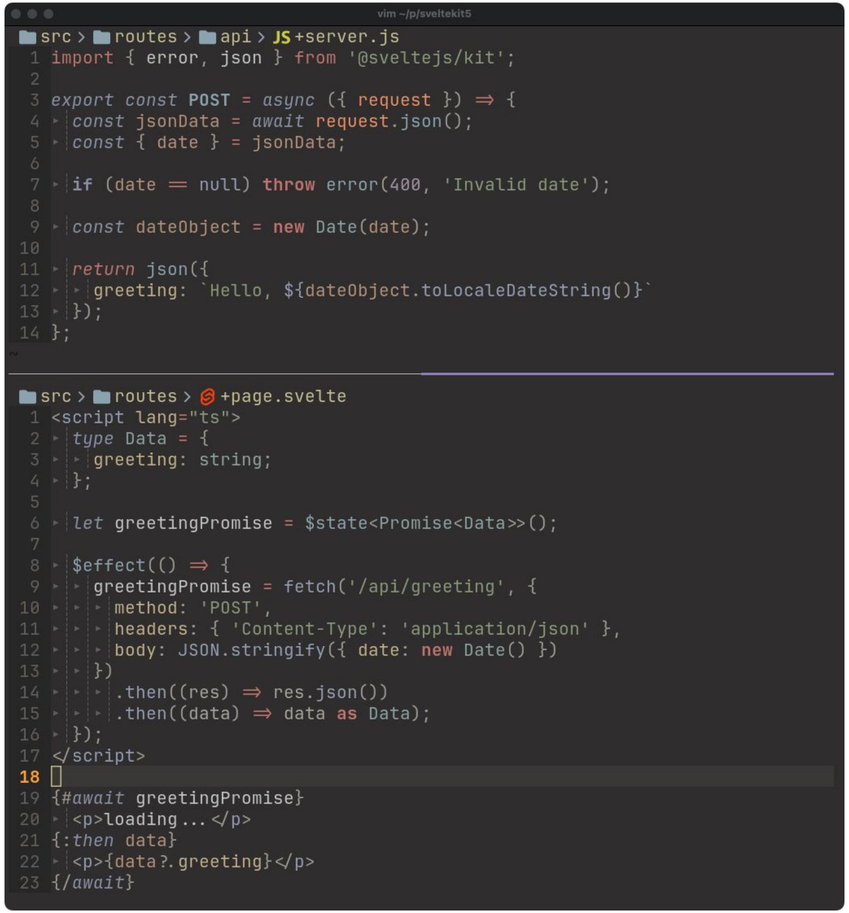
```
import { error, json } from '@sveltejs/kit';

export const POST = async ({ request, cookies }) => {
	const session = cookies.get(name: 'session');
	if (session == null) {
		throw error(status: 401, body: 'Unauthorized');
	}

	const jsonData = await request.json();
	const { date } = jsonData;

	return json({
		message: `Your date is ${date}`
	});
};

export const GET = async () => {
	const date = new Date();
	return json({
		message: `Server time is ${date}`
	});
};
```

そもそもなぜEndpointを使いたいのか

- 簡単なアプリケーションであれば page load関数とformを駆使すればいいと思う
- ある程度規模が大きくなると APIを整備した方が見通しが良くなる (BFFを立てたりしますよね？）

Endpointが型安全ではない(トテモツライ)

- 何一つ型安全でない
    - Path
    - Method
    - 引数
    - 戻り値
- 色々PRはあるけど後回しになってる
- SvelteKitに頼らずに作りたい



```js
import { error, json } from '@sveltejs/kit';

export const POST = async ({ request }) => {
	const jsonData = await request.json();
	const { date } = jsonData;

	if (date == null) throw error(400, 'Invalid date');

	const dateObject = new Date(date);

	return json({
		greeting: `Hello, ${dateObject.toLocaleDateString()}`
	});
};
```

```svelte
<script lang="ts">
	type Data = {
		greeting: string;
	};

	let greetingPromise = $state<Promise<Data>>();

	$effect(() => {
		greetingPromise = fetch('/api/greeting', {
			method: 'POST',
			headers: { 'Content-Type': 'application/json' },
			body: JSON.stringify({ date: new Date() })
		})
			.then((res) => res.json())
			.then((data) => data as Data);
	});
</script>

{#await greetingPromise}
	<p>loading...</p>
{:then data}
	<p>{data?.greeting}</p>
{/await}
```

標準では型安全じゃない部分もあります

- Form Action
- API Endpoint

小規模のアプリを作るなら目をつぶれるが、中規模〜だと型がないと厳しくなってくるので

型安全なEndpointを作るためのライブラリ候補

- tRPC
- Hono RPC
- GraphQL Schemaから自動生成?

などなど…

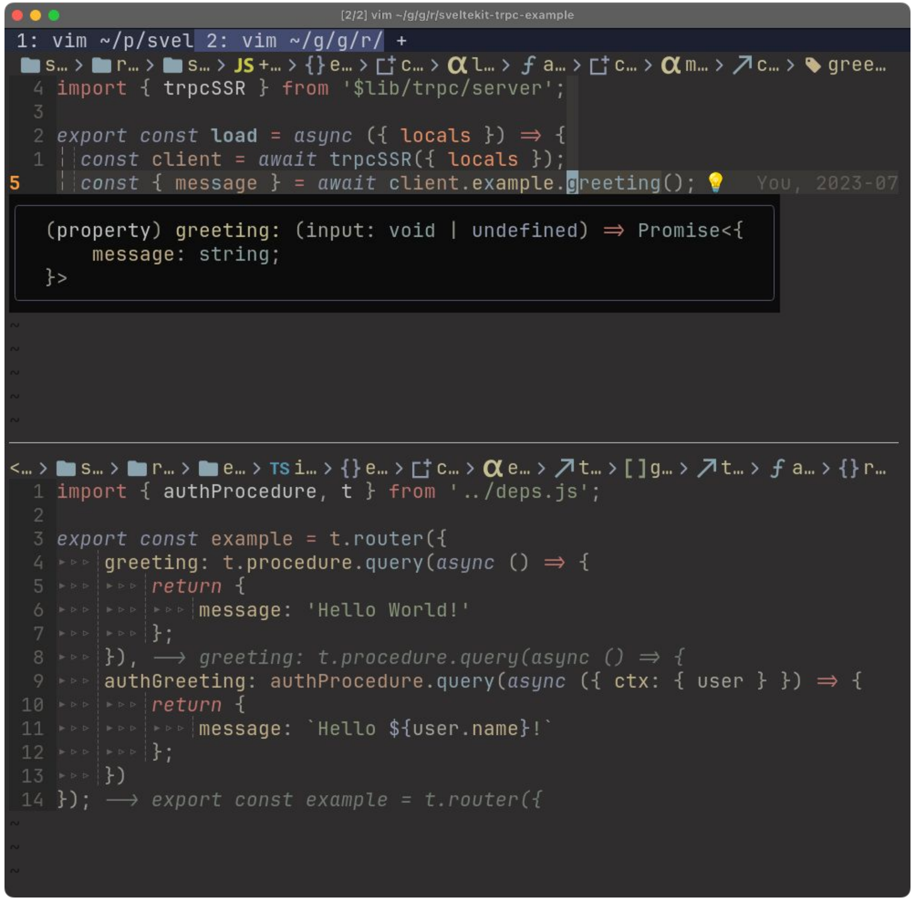
型安全なEndpointを作るためのライブラリ候補

- tRPC
- Hono RPC
- GraphQL Schemaから自動生成?

などなど…

tRPC 👍

- TypeScriptの型パズルを駆使してType-safeなEndpointを作れるライブラリ
- Pure Typescript Projectとの親和性良き
- Routing及び、引数&戻り値のValidationを担当する
    - API構築以外の部分は SvelteKitのエコシステムを流用

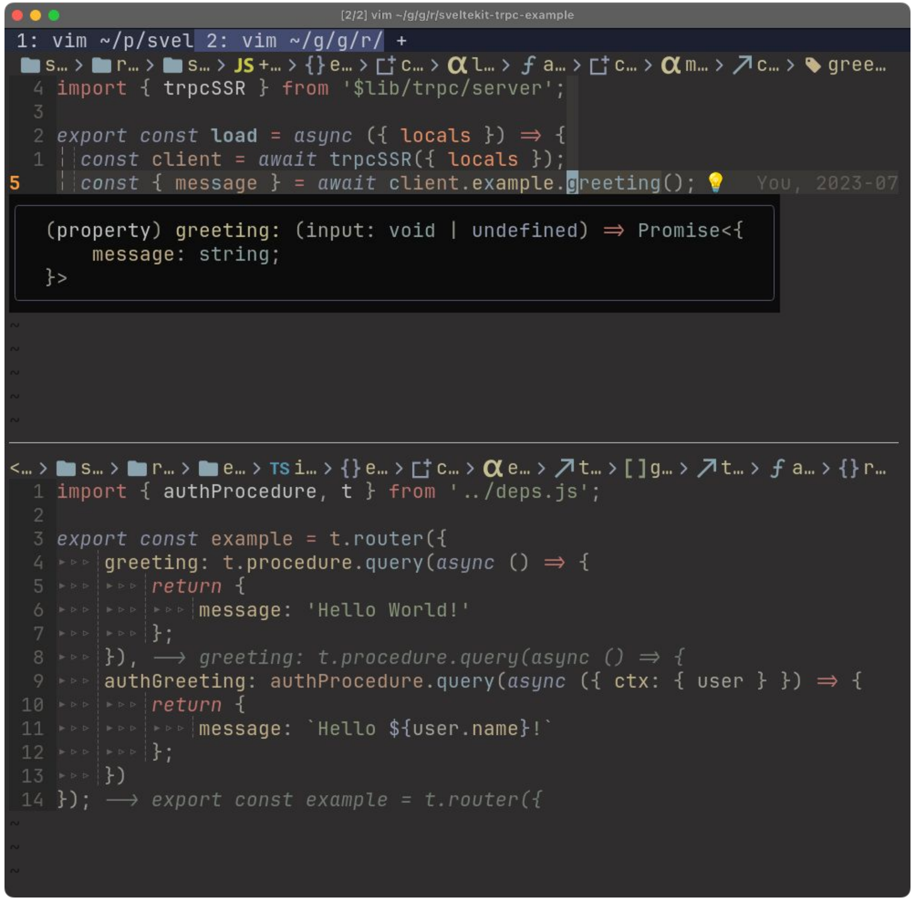

[2/2] vim ~/g/g/r/sveltekit-trpc-example
1: vim ~/p/svel 2: vim ~/g/g/r/ +

```
import { trpcSSR } from '$lib/trpc/server';

export const load = async ({ locals }) ⇒ {
  const client = await trpcSSR({ locals });
  const { message } = await client.example.greeting(); 💡  You, 2023-07

  (property) greeting: (input: void | undefined) ⇒ Promise<{
      message: string;
  }>
```

```
import { authProcedure, t } from '../deps.js';

export const example = t.router({
  greeting: t.procedure.query(async () ⇒ {
    return {
      message: 'Hello World!'
    };
  }), ⟶ greeting: t.procedure.query(async () ⇒ {
  authGreeting: authProcedure.query(async ({ ctx: { user } }) ⇒ {
    return {
      message: `Hello ${user.name}!`
    };
  })
}); ⟶ export const example = t.router({
```

tRPC 👎

- TypeScript以外のプロジェクトで使いずらい
  - 一応trpc-openapiなるプロジェクトはあるのだが、安定性に欠けるし、Edge Workerでは動かない
- 後々APIだけ分離したい時に少々めんどくさい


[2/2] vim ~/g/g/r/sveltekit-trpc-example
1: vim ~/p/svel 2: vim ~/g/g/r/ +

```
import { trpcSSR } from '$lib/trpc/server';

export const load = async ({ locals }) ⇒ {
  const client = await trpcSSR({ locals });
  const { message } = await client.example.greeting(); 💡 You, 2023-07
```

```
(property) greeting: (input: void | undefined) ⇒ Promise<{
    message: string;
}>
```

```
import { authProcedure, t } from '../deps.js';

export const example = t.router({
    greeting: t.procedure.query(async () ⇒ {
        return {
            message: 'Hello World!'
        };
    }), ⟶ greeting: t.procedure.query(async () ⇒ {
    authGreeting: authProcedure.query(async ({ ctx: { user } }) ⇒ {
        return {
            message: `Hello ${user.name}!`
        };
    })
}); ⟶ export const example = t.router({
```

型安全なEndpointを作るためのライブラリ候補

- tRPC
- Hono RPC
- GraphQL Schemaから自動生成?

などなど…

Honoとは

- Ultrafast Web Framework
- Routing baseでAPI Endpointを構築できる
- 軽い！(12KB)
- Web標準準拠でどこでも動く

Hono RPCとは

- HonoのAPIを型安全に呼び出せるClientを生成できる → tRPCライク
- tRPCを超えた強力なValidation、型安全
- いろんなFrameworkと自由自在に組み合わせられる

SvelteKitとHonoの組み合わせ方

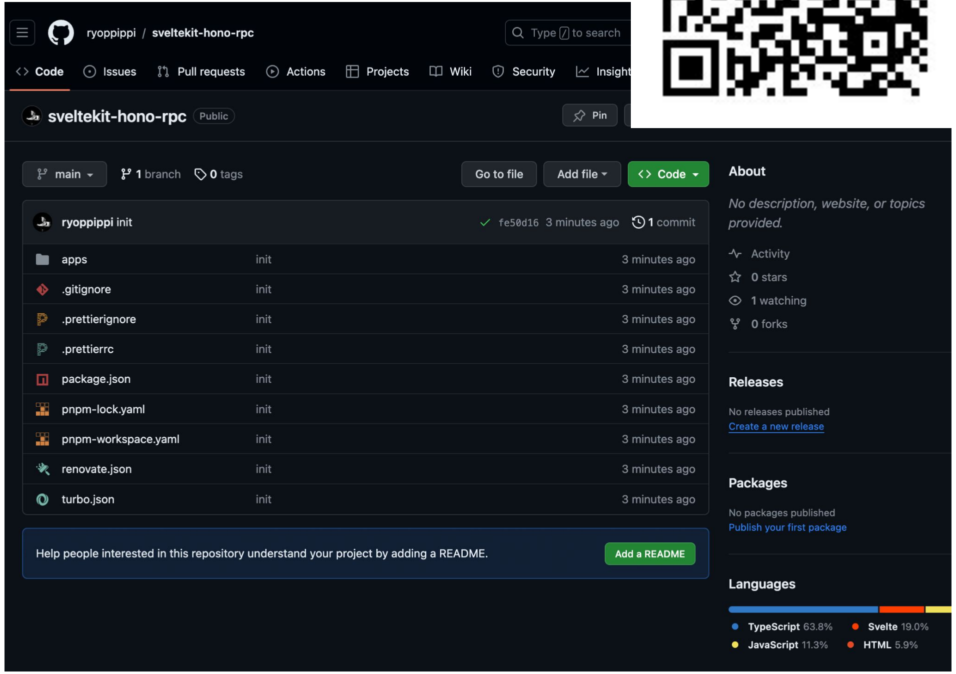
- SvelteKitのビルドに含めてしまう
- Hono AppとSvelteKit Appを別々にDeployする

SvelteKitとHonoの組み合わせ方

- SvelteKitのビルドに含めてしまう
- Hono AppとSvelteKit Appを別々にDeployする

サンプル作りました

- Hono RPC + SvelteKitのモノレポ
- 自分はこのスタイルがおすすめ


ryoppippi / sveltekit-hono-rpc

Code Issues Pull requests Actions Projects Wiki Security Insight

sveltekit-hono-rpc Public

main 1 branch 0 tags Go to file Add file Code

ryoppippi init fe50d16 3 minutes ago 1 commit

apps	init	3 minutes ago
.gitignore	init	3 minutes ago
.prettierignore	init	3 minutes ago
.prettierrc	init	3 minutes ago
package.json	init	3 minutes ago
pnpm-lock.yaml	init	3 minutes ago
pnpm-workspace.yaml	init	3 minutes ago
renovate.json	init	3 minutes ago
turbo.json	init	3 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

About

No description, website, or topics provided.

Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

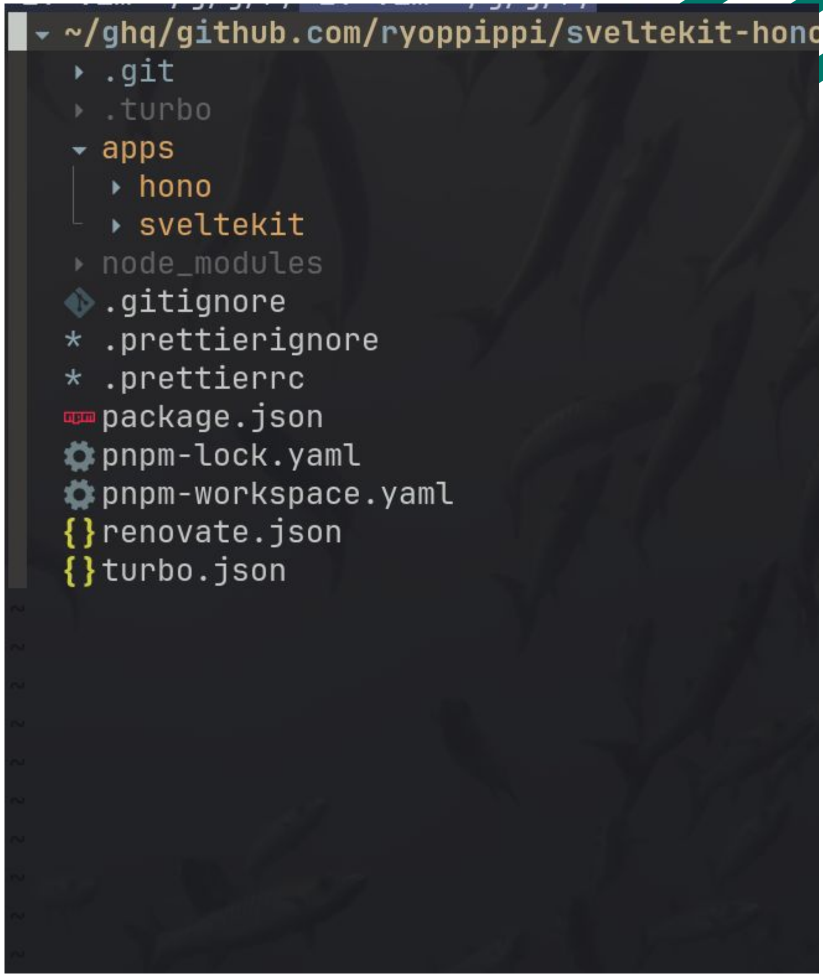
Packages

No packages published
Publish your first package

Languages

TypeScript 63.8% Svelte 19.0%
JavaScript 11.3% HTML 5.9%

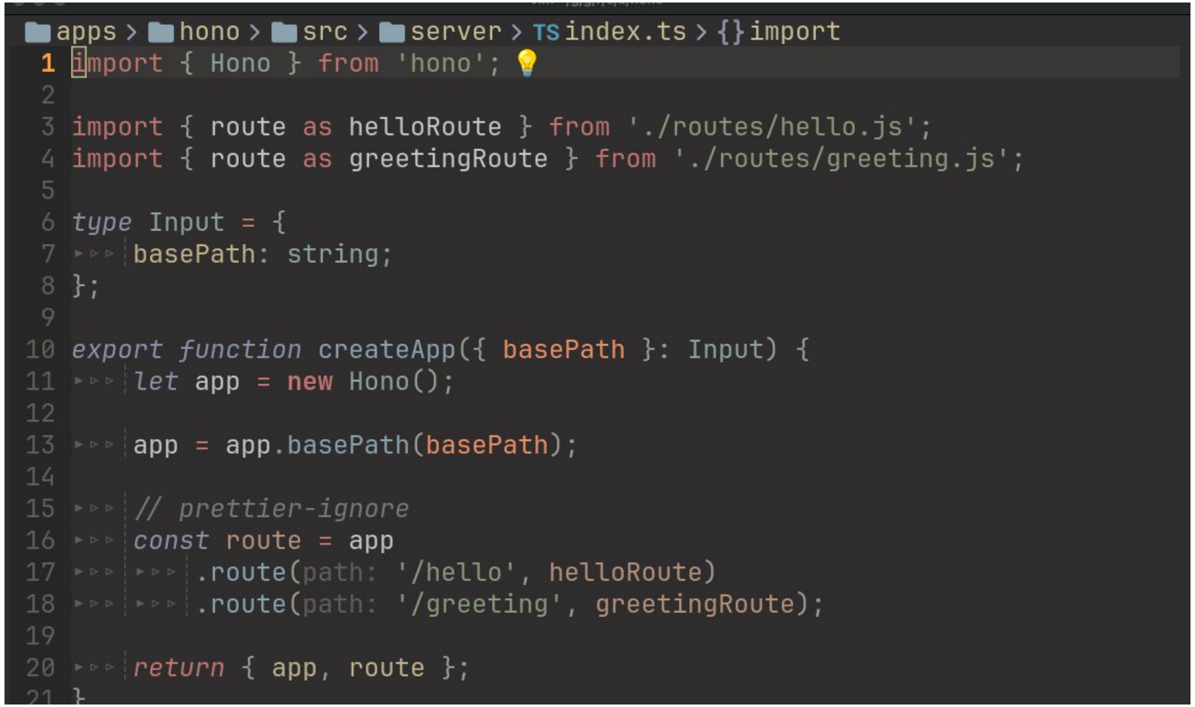
プロジェクト構成



```
▾ ~/ghq/github.com/ryoppippi/sveltekit-hono
  ▸ .git
  ▸ .turbo
  ▾ apps
    ▸ hono
    ▸ sveltekit
  ▸ node_modules
   .gitignore
  * .prettierignore
  * .prettierrc
   package.json
   pnpm-lock.yaml
   pnpm-workspace.yaml
  {}renovate.json
  {}turbo.json
```

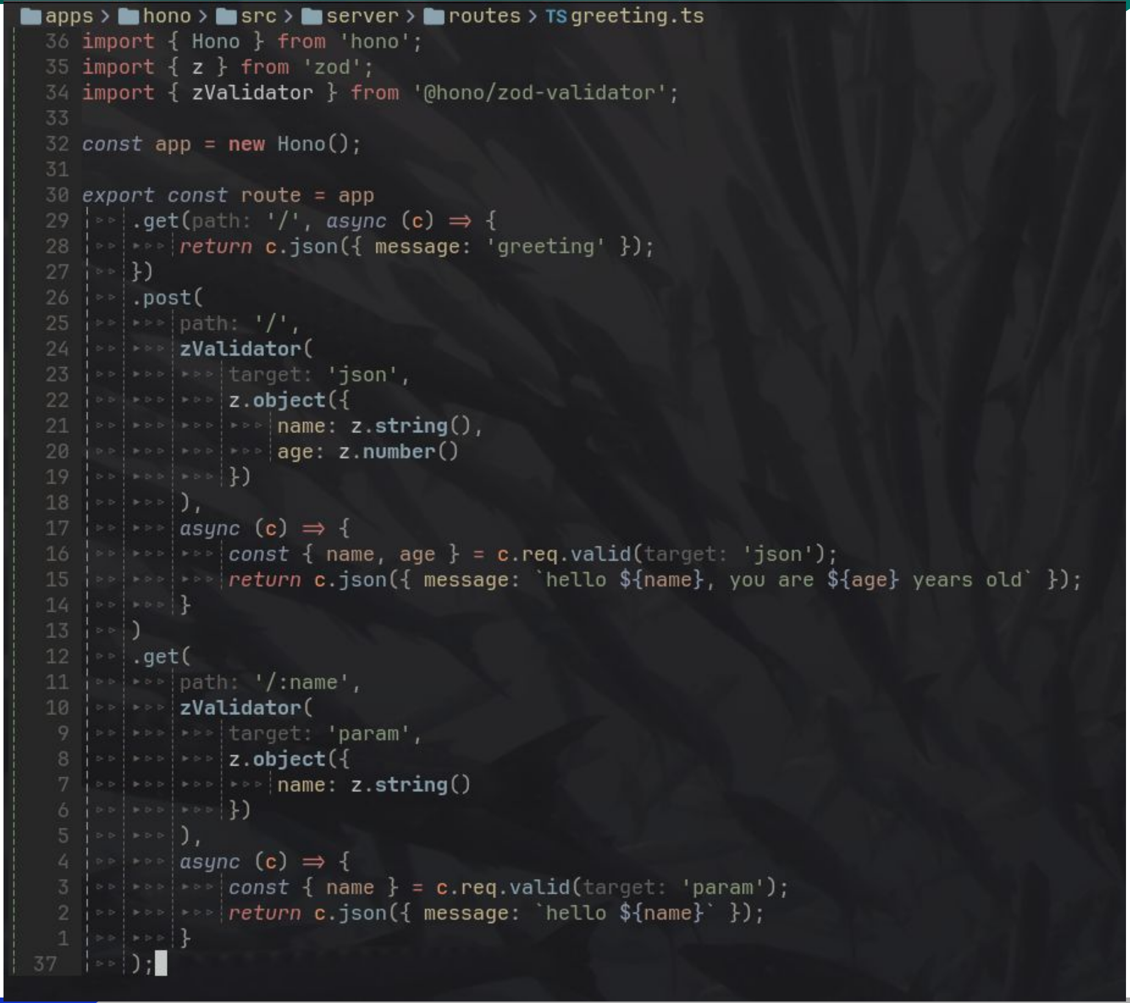
Hono側の実装

Appを定義


apps > hono > src > server > TS index.ts > {} import

```ts
import { Hono } from 'hono';

import { route as helloRoute } from './routes/hello.js';
import { route as greetingRoute } from './routes/greeting.js';

type Input = {
  basePath: string;
};

export function createApp({ basePath }: Input) {
  let app = new Hono();

  app = app.basePath(basePath);

  // prettier-ignore
  const route = app
    .route(path: '/hello', helloRoute)
    .route(path: '/greeting', greetingRoute);

  return { app, route };
}
```

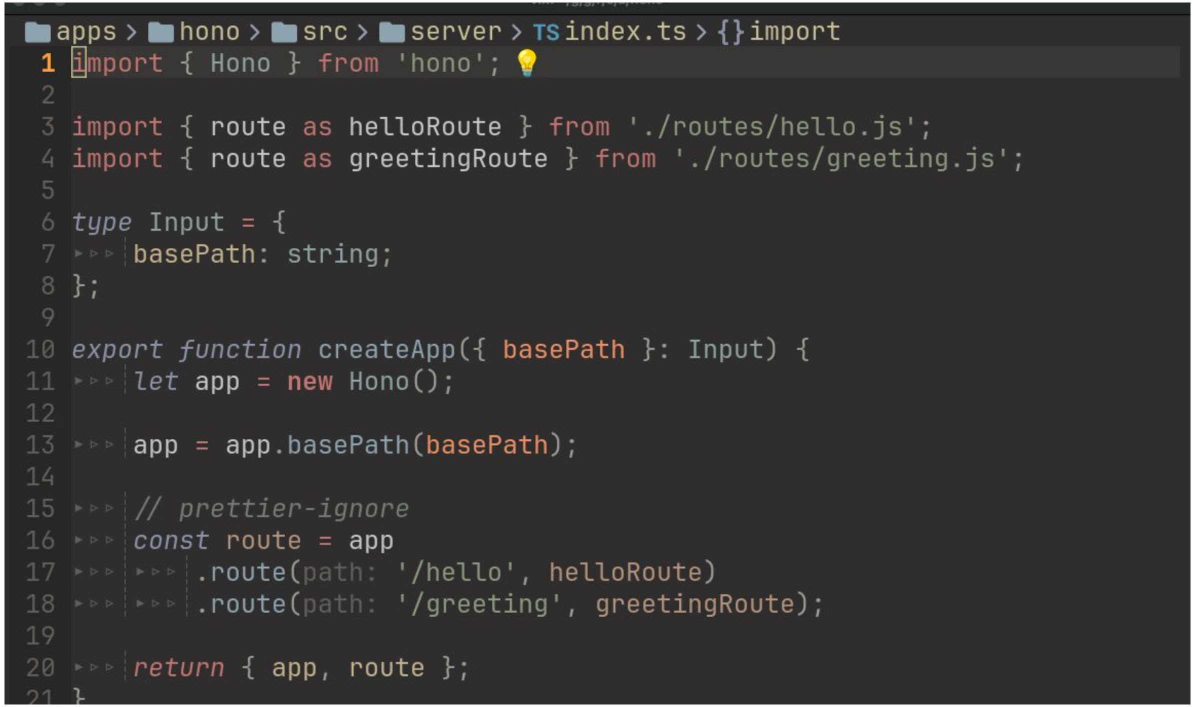
Routingを定義



apps > hono > src > server > routes > TS greeting.ts

```
import { Hono } from 'hono';
import { z } from 'zod';
import { zValidator } from '@hono/zod-validator';

const app = new Hono();

export const route = app
  .get(path: '/', async (c) => {
    return c.json({ message: 'greeting' });
  })
  .post(
    path: '/',
    zValidator(
      target: 'json',
      z.object({
        name: z.string(),
        age: z.number()
      })
    ),
    async (c) => {
      const { name, age } = c.req.valid(target: 'json');
      return c.json({ message: `hello ${name}, you are ${age} years old` });
    }
  )
  .get(
    path: '/:name',
    zValidator(
      target: 'param',
      z.object({
        name: z.string()
      })
    ),
    async (c) => {
      const { name } = c.req.valid(target: 'param');
      return c.json({ message: `hello ${name}` });
    }
  );
```

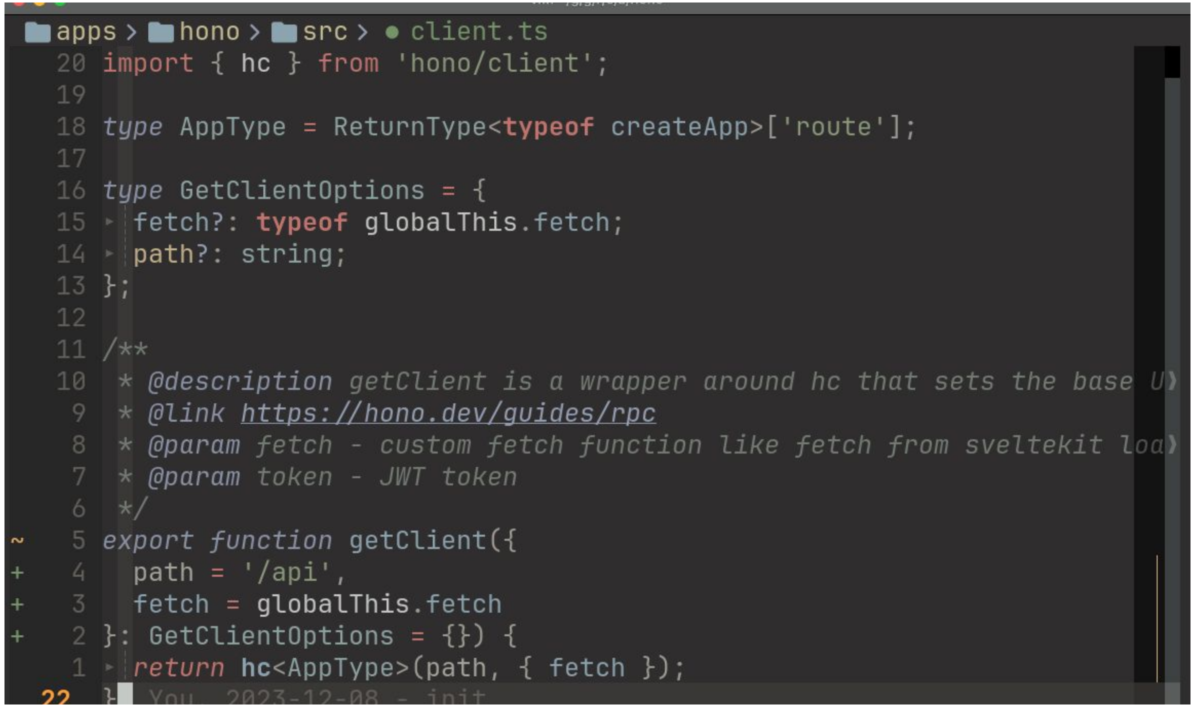
Appを定義



apps > hono > src > server > TS index.ts > {} import

```ts
import { Hono } from 'hono';

import { route as helloRoute } from './routes/hello.js';
import { route as greetingRoute } from './routes/greeting.js';

type Input = {
  basePath: string;
};

export function createApp({ basePath }: Input) {
  let app = new Hono();

  app = app.basePath(basePath);

  // prettier-ignore
  const route = app
    .route(path: '/hello', helloRoute)
    .route(path: '/greeting', greetingRoute);

  return { app, route };
}
```

RPC Clientを定義

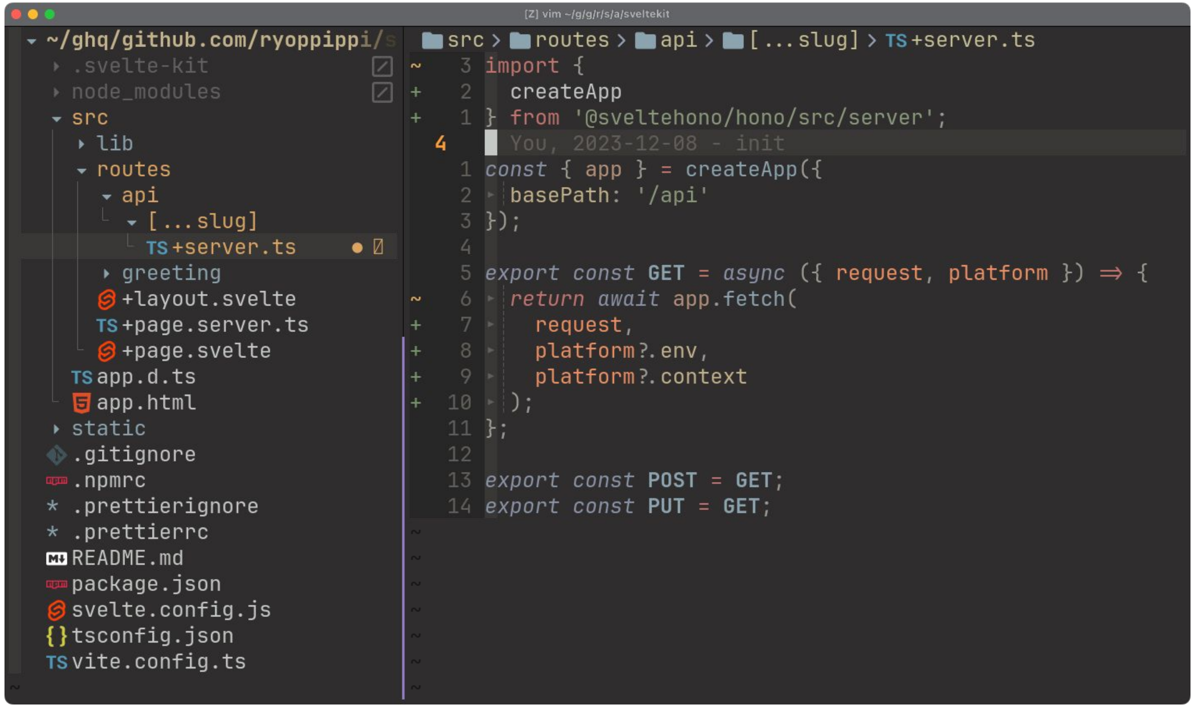


apps > hono > src > client.ts

```
import { hc } from 'hono/client';

type AppType = ReturnType<typeof createApp>['route'];

type GetClientOptions = {
  fetch?: typeof globalThis.fetch;
  path?: string;
};

/**
 * @description getClient is a wrapper around hc that sets the base U
 * @link https://hono.dev/guides/rpc
 * @param fetch - custom fetch function like fetch from sveltekit loa
 * @param token - JWT token
 */
export function getClient({
  path = '/api',
  fetch = globalThis.fetch
}: GetClientOptions = {}) {
  return hc<AppType>(path, { fetch });
}   You, 2023-12-08 - init
```

SvelteKit側の実装

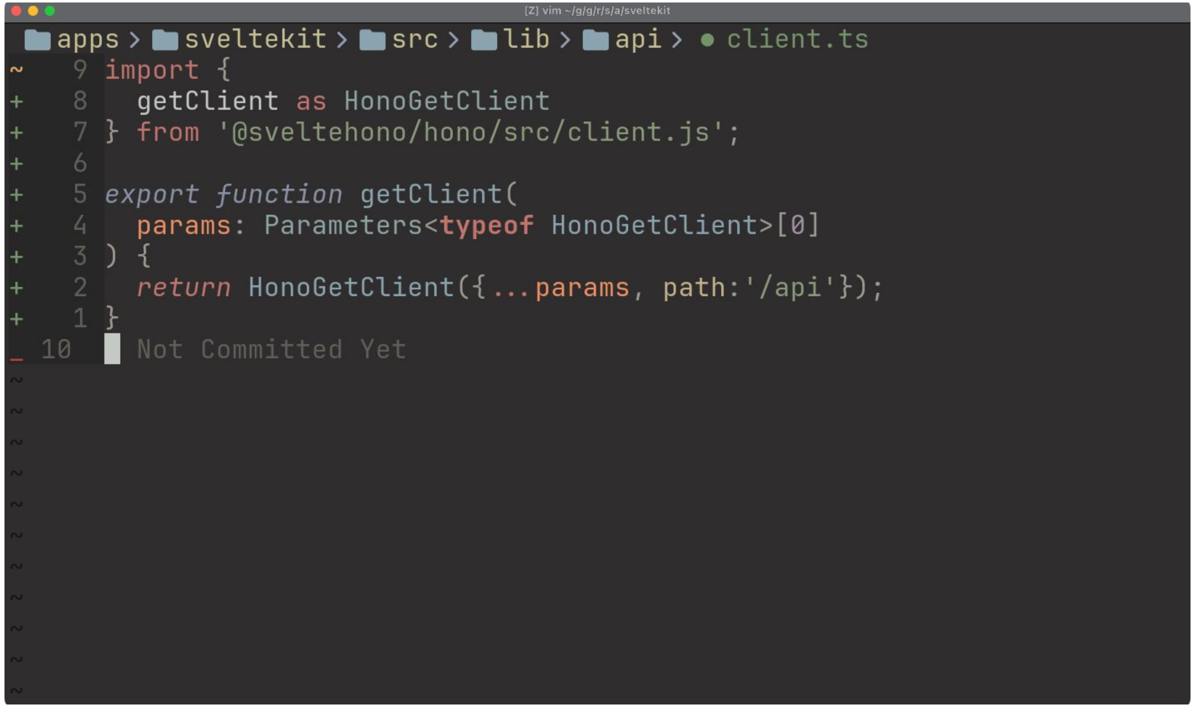
Hono を SvelteKitのルーターにMount


[Z] vim ~/g/g/r/s/a/sveltekit
src > routes > api > [...slug] > +server.ts

```
~/ghq/github.com/ryoppippi/s
  .svelte-kit
  node_modules
  src
    lib
    routes
      api
        [...slug]
          +server.ts
      greeting
      +layout.svelte
      +page.server.ts
      +page.svelte
    app.d.ts
    app.html
  static
  .gitignore
  .npmrc
  .prettierignore
  .prettierrc
  README.md
  package.json
  svelte.config.js
  tsconfig.json
  vite.config.ts
```

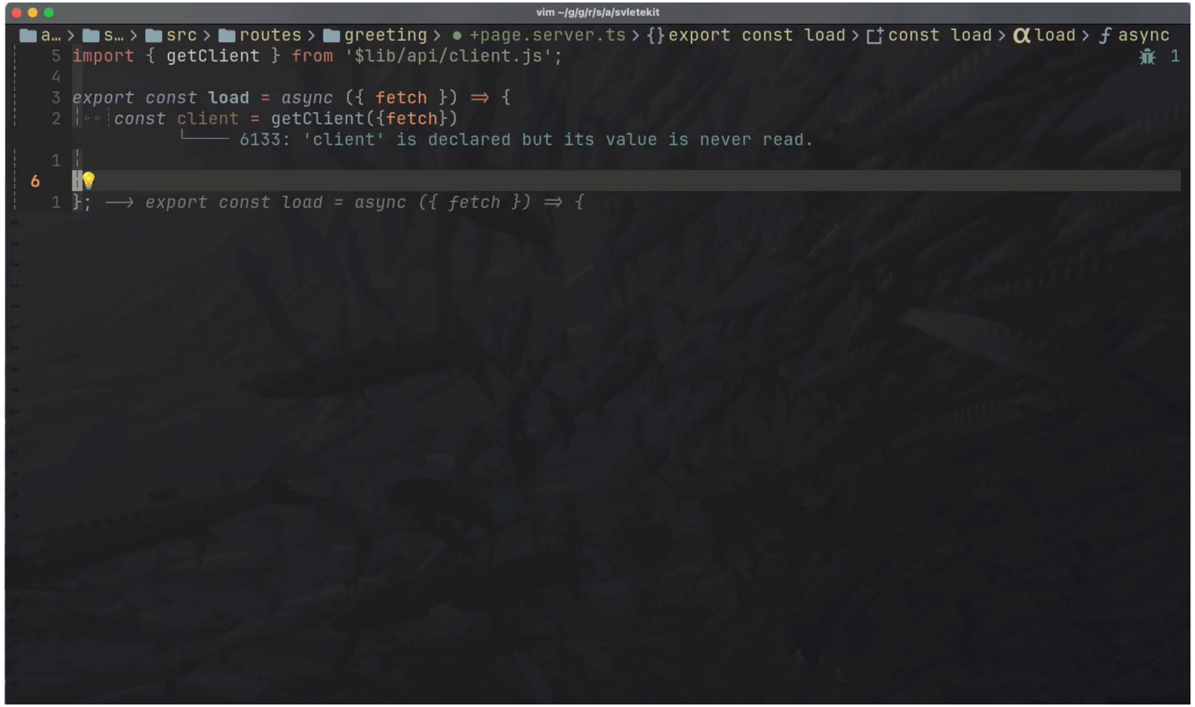
```ts
import {
  createApp
} from '@sveltehono/hono/src/server';
  You, 2023-12-08 - init
const { app } = createApp({
  basePath: '/api'
});

export const GET = async ({ request, platform }) => {
  return await app.fetch(
    request,
    platform?.env,
    platform?.context
  );
};

export const POST = GET;
export const PUT = GET;
```

getClientを定義


[Z] vim ~/g/g/r/s/a/sveltekit
apps > sveltekit > src > lib > api > ● client.ts

```
import {
  getClient as HonoGetClient
} from '@sveltehono/hono/src/client.js';

export function getClient(
  params: Parameters<typeof HonoGetClient>[0]
) {
  return HonoGetClient({...params, path:'/api'});
}
   Not Committed Yet
```

getClientを使ってゴリゴリ書いていく

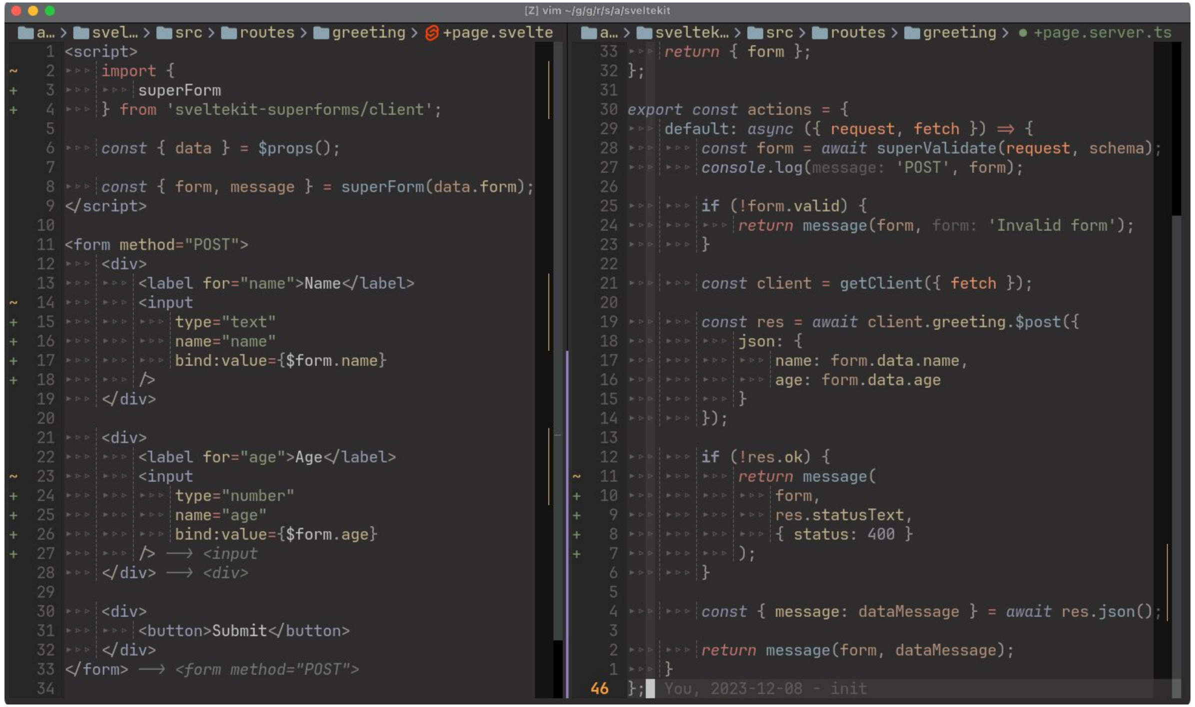


vim ~/g/g/r/s/a/svletekit

a… > s… > src > routes > greeting > ● +page.server.ts > {} export const load > const load > load > async

```
import { getClient } from '$lib/api/client.js';

export const load = async ({ fetch }) => {
    const client = getClient({fetch})
          └──── 6133: 'client' is declared but its value is never read.


}; ──→ export const load = async ({ fetch }) => {
```

Form ActionもSuperformsと組み合わせて



[Z] vim ~/g/g/r/s/a/sveltekit

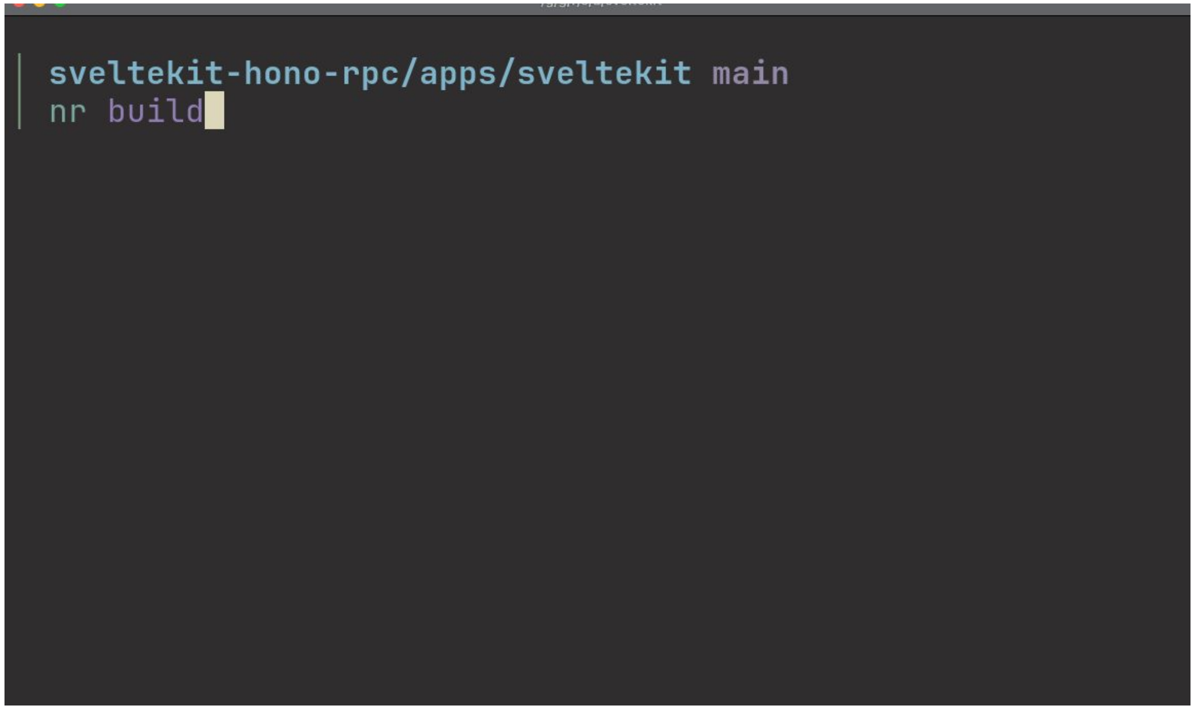
a… > svel… > src > routes > greeting > +page.svelte

```svelte
<script>
    import {
        superForm
    } from 'sveltekit-superforms/client';

    const { data } = $props();

    const { form, message } = superForm(data.form);
</script>

<form method="POST">
    <div>
        <label for="name">Name</label>
        <input
            type="text"
            name="name"
            bind:value={$form.name}
        />
    </div>

    <div>
        <label for="age">Age</label>
        <input
            type="number"
            name="age"
            bind:value={$form.age}
        /> ⟶ <input
    </div> ⟶ <div>

    <div>
        <button>Submit</button>
    </div>
</form> ⟶ <form method="POST">
```

a… > sveltek… > src > routes > greeting > +page.server.ts

```ts
    return { form };
};

export const actions = {
    default: async ({ request, fetch }) => {
        const form = await superValidate(request, schema);
        console.log(message: 'POST', form);

        if (!form.valid) {
            return message(form, form: 'Invalid form');
        }

        const client = getClient({ fetch });

        const res = await client.greeting.$post({
            json: {
                name: form.data.name,
                age: form.data.age
            }
        });

        if (!res.ok) {
            return message(
                form,
                res.statusText,
                { status: 400 }
            );
        }

        const { message: dataMessage } = await res.json();

        return message(form, dataMessage);
    }
}; You, 2023-12-08 - init
```

SvelteKitを build & deploy



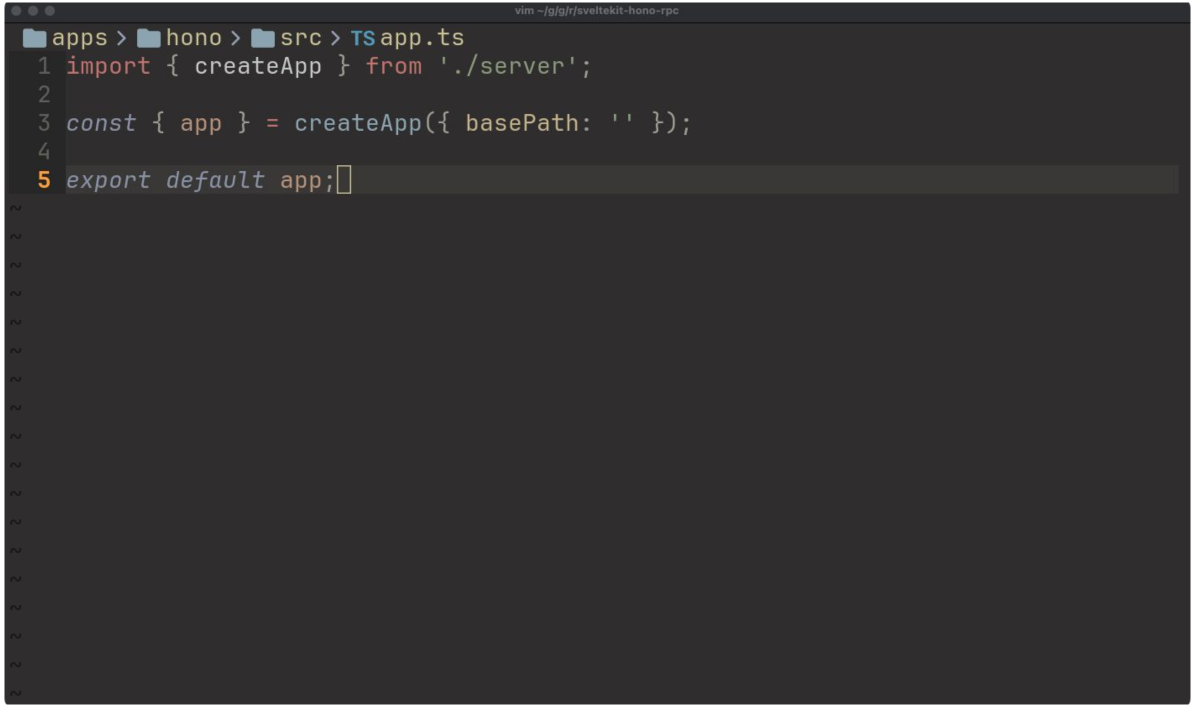
```
sveltekit-hono-rpc/apps/sveltekit main
nr build
```

SvelteKitとHonoの組み合わせ方

- SvelteKitのビルドに含めてしまう
- Hono AppとSvelteKit Appを別々にDeployする

Hono側の実装

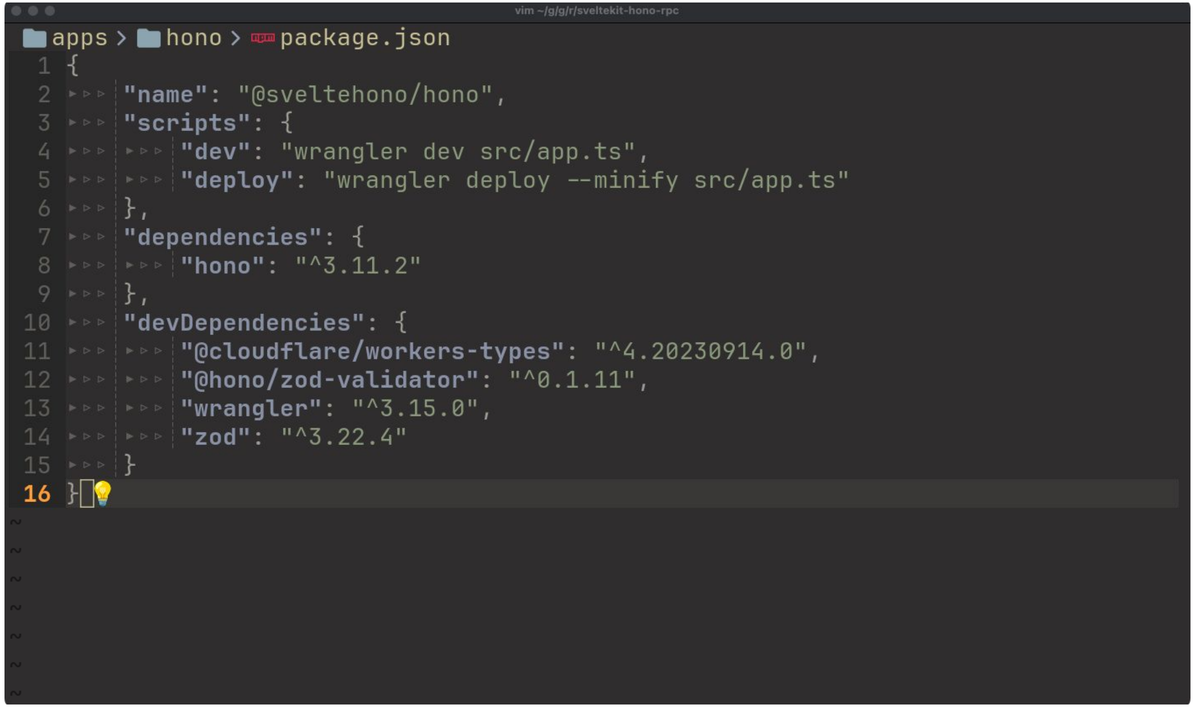
Appをdefault exportするファイルを定義


vim ~/g/g/r/sveltekit-hono-rpc

apps > hono > src > TS app.ts

```
import { createApp } from './server';

const { app } = createApp({ basePath: '' });

export default app;
```

wranglerを使ってdeploy

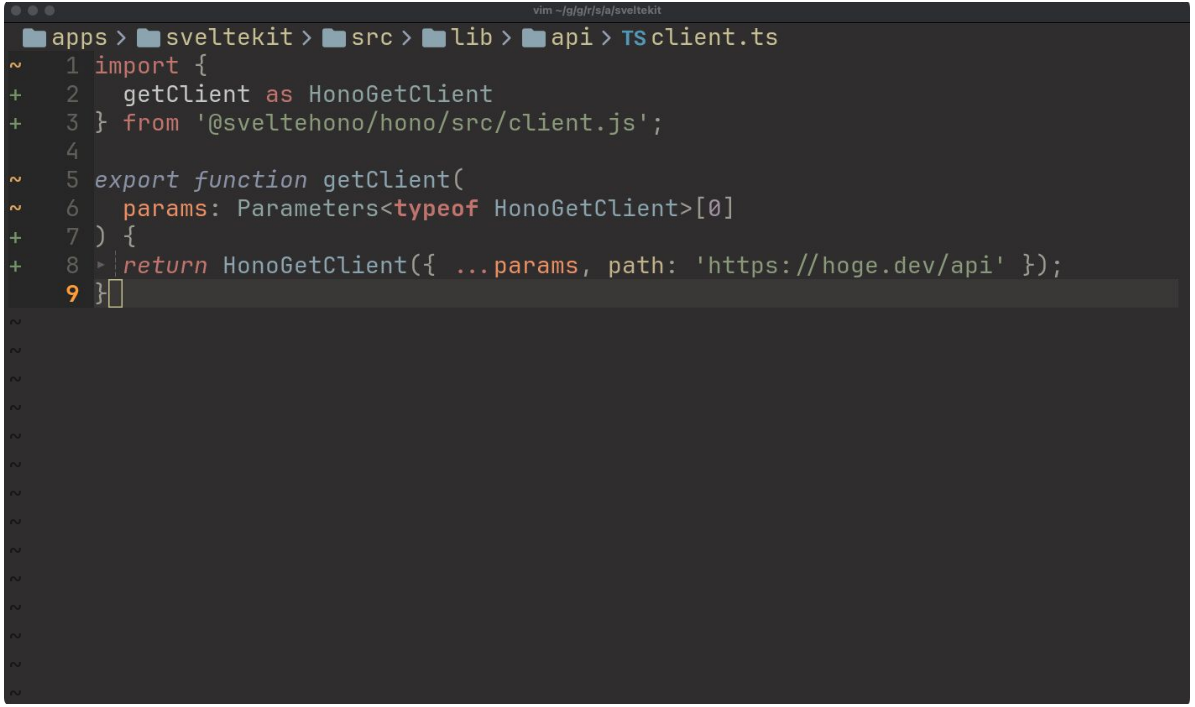

vim ~/g/g/r/sveltekit-hono-rpc

apps > hono > package.json

```
{
	"name": "@sveltehono/hono",
	"scripts": {
		"dev": "wrangler dev src/app.ts",
		"deploy": "wrangler deploy --minify src/app.ts"
	},
	"dependencies": {
		"hono": "^3.11.2"
	},
	"devDependencies": {
		"@cloudflare/workers-types": "^4.20230914.0",
		"@hono/zod-validator": "^0.1.11",
		"wrangler": "^3.15.0",
		"zod": "^3.22.4"
	}
}
```

SvelteKit側の実装

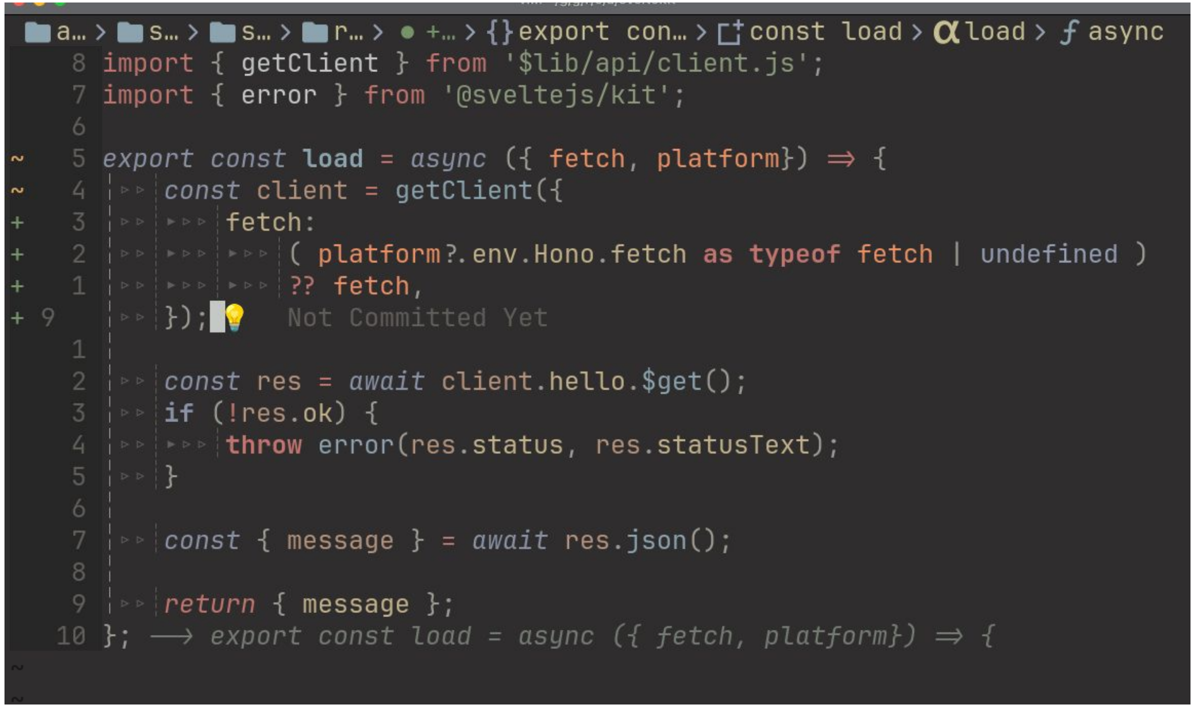
getClientのPathを修正


vim ~/g/g/r/s/a/sveltekit

apps > sveltekit > src > lib > api > TS client.ts

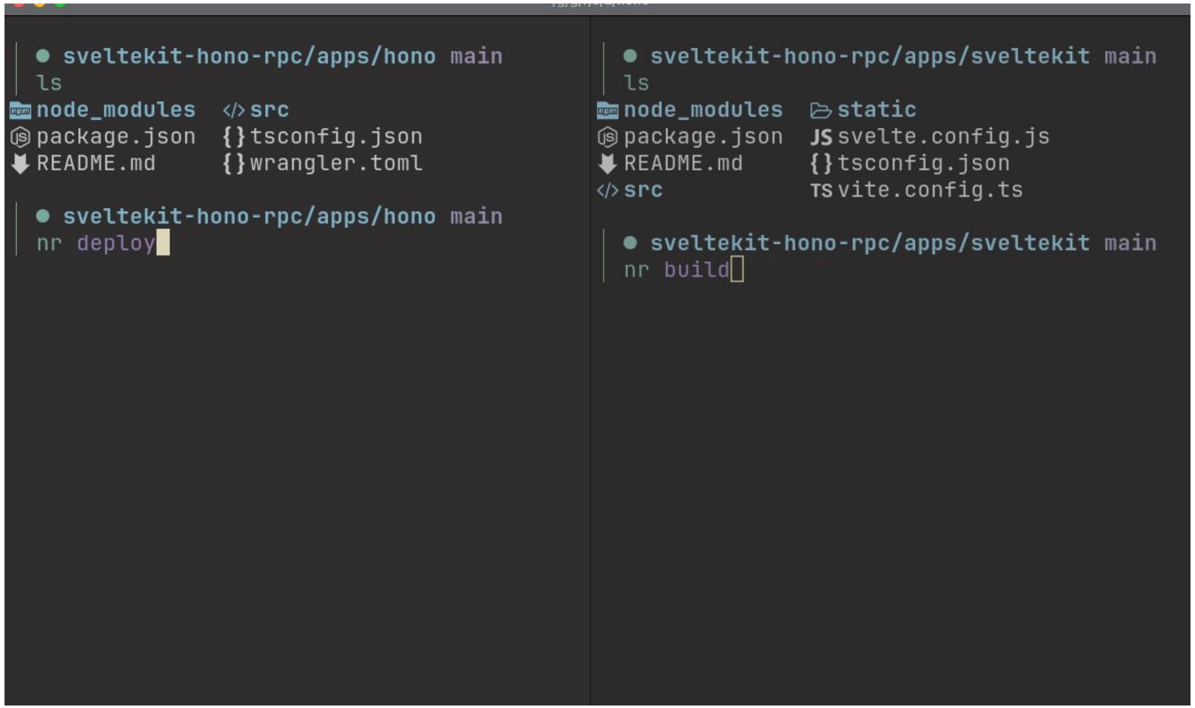
```
import {
  getClient as HonoGetClient
} from '@sveltehono/hono/src/client.js';

export function getClient(
  params: Parameters<typeof HonoGetClient>[0]
) {
  return HonoGetClient({ ...params, path: 'https://hoge.dev/api' });
}
```

(Cloudflareなどの）Service Bindingを使う


a… > s… > s… > r… > ● +… > {}export con… > const load > load > async

```
import { getClient } from '$lib/api/client.js';
import { error } from '@sveltejs/kit';

export const load = async ({ fetch, platform}) => {
    const client = getClient({
        fetch:
            ( platform?.env.Hono.fetch as typeof fetch | undefined )
            ?? fetch,
    });    Not Committed Yet

    const res = await client.hello.$get();
    if (!res.ok) {
        throw error(res.status, res.statusText);
    }

    const { message } = await res.json();

    return { message };
}; ⟶ export const load = async ({ fetch, platform}) => {
```

各々をbuild & deploy



```
● sveltekit-hono-rpc/apps/hono main
ls
node_modules   src
package.json   tsconfig.json
README.md      wrangler.toml

● sveltekit-hono-rpc/apps/hono main
nr deploy
```

```
● sveltekit-hono-rpc/apps/sveltekit main
ls
node_modules   static
package.json   svelte.config.js
README.md      tsconfig.json
src            vite.config.ts

● sveltekit-hono-rpc/apps/sveltekit main
nr build
```

Hono RPC 👍

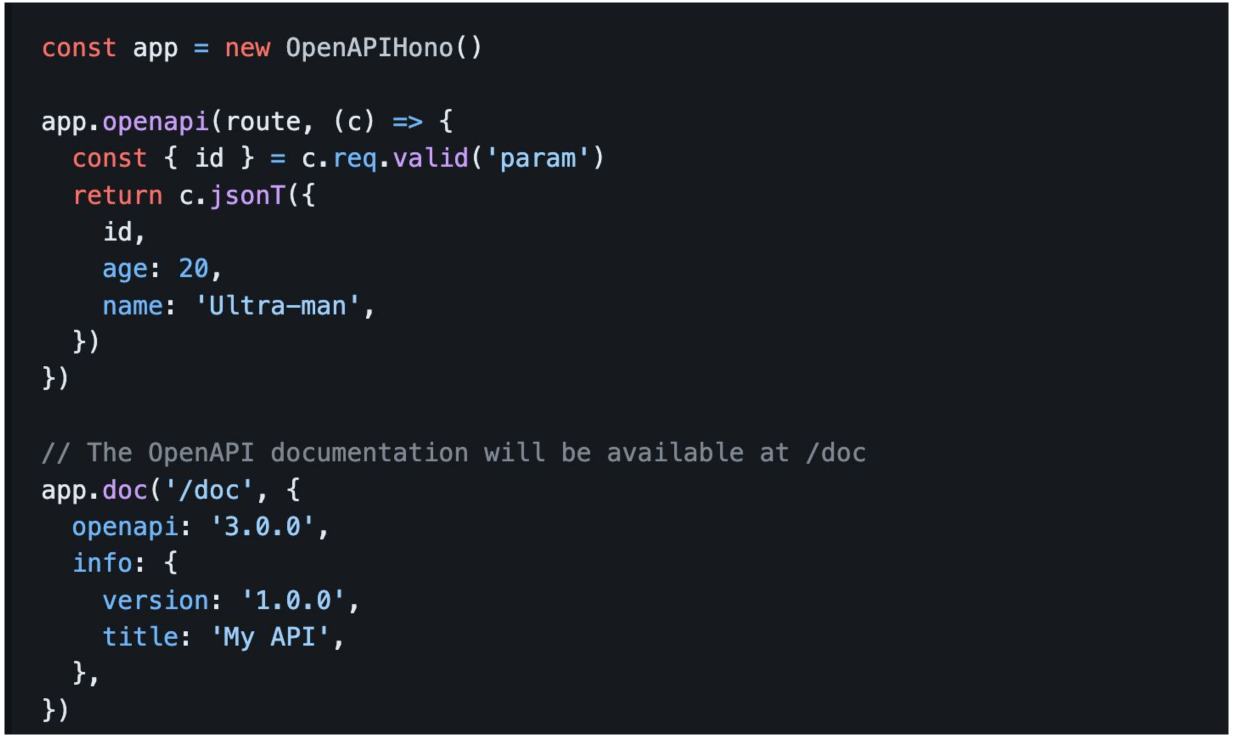
- HonoのAPIを型安全に呼び出せるClientを生成できる → tRPCライク
- Validationが強力
- API側をSvelteKitに組み込むも、分離するも自由
- API単体のテストがやりやすい

Hono RPC 👎

- HonoとSvelteKitでCookieやParameterのアクセス方法が違う
  - SveteKit - `cookie.get(‘hoge’)`
  - Hono - getCookie(c, ‘hoge’)

HonoとSvelteKitで
快適な開発をお楽しみください

おまけ: Hono + OpenAPI



```
const app = new OpenAPIHono()

app.openapi(route, (c) => {
  const { id } = c.req.valid('param')
  return c.jsonT({
    id,
    age: 20,
    name: 'Ultra-man',
  })
})

// The OpenAPI documentation will be available at /doc
app.doc('/doc', {
  openapi: '3.0.0',
  info: {
    version: '1.0.0',
    title: 'My API',
  },
})
```