

# Vike

Next-gen framework architecture



# Привет!

## Оловянников Илья

- Создаю сообщество
- Инвестирую в Open Source 🚀
- Люблю хороший DX



Вы когда-нибудь мечтали...



Стать лучшей версией себя?

# Rom(uald) Brillout

- Vike  <https://vike.dev/>
- Photon  <https://photonjs.dev/>
- Telefunc RPC  <https://telefunc.com/>



# Вайк? Вик? Вике?



Vikings + Like = Vike

Pronunciation of 'Vike' #1230

Answered by brillout spdiswal asked this question in Help & Questions

spdiswal on Nov 1, 2023

What is the correct way to say Vike? I haven't found any official mentioning of this in text or videos.

Does it lean towards the pronunciation of Vite, i.e. 'veek'?  
Or does it rhyme with bike/like, i.e. 'vai-k?' Or perhaps even 'vai-key' like Nike? 😊

1 comment · 1 reply

brillout on Nov 1, 2023 · Maintainer

Like Viking, i.e. 'vai-k!' — contribution welcome to add an audio recording to <https://vike.dev/> 😊

Marked as answer

spdiswal on Nov 1, 2023 · Author

Cool, thanks for clarifying!

Write a reply

Answer selected by spdiswal

# О чём поговорим

# О чём поговорим

- Что такое Meta Framework;

# О чём поговорим

- Что такое Meta Framework;
- Что такое Vike;

# О чём поговорим

- Что такое Meta Framework;
- Что такое Vike;
- Из чего состоит приложение Vike;

# О чём поговорим

- Что такое Meta Framework;
- Что такое Vike;
- Из чего состоит приложение Vike;
- Когда применять.

# Что такое Meta Framework?

# Что такое Meta Framework?

Высокоуровневый набор функций для разработки приложений

# Что такое Meta Framework?

**Высокоуровневый набор функций для разработки приложений**

Надстройка над UI библиотекой, которая объединяет и дополняет инструменты для полноценной разработки приложения:

# Что такое Meta Framework?

**Высокоуровневый набор функций для разработки приложений**

Надстройка над UI библиотекой, которая объединяет и дополняет инструменты для полноценной разработки приложения:

- Встроенный SSR;

# Что такое Meta Framework?

**Высокоуровневый набор функций для разработки приложений**

Надстройка над UI библиотекой, которая объединяет и дополняет инструменты для полноценной разработки приложения:

- Встроенный SSR;
- Встроенный маршрутизатор;

# Что такое Meta Framework?

**Высокоуровневый набор функций для разработки приложений**

Надстройка над UI библиотекой, которая объединяет и дополняет инструменты для полноценной разработки приложения:

- Встроенный SSR;
- Встроенный маршрутизатор;
- Готовая сборка;

# Что такое Meta Framework?

**Высокоуровневый набор функций для разработки приложений**

Надстройка над UI библиотекой, которая объединяет и дополняет инструменты для полноценной разработки приложения:

- Встроенный SSR;
- Встроенный маршрутизатор;
- Готовая сборка;
- Упрощенная работа с SEO;

# Что такое Meta Framework?

**Высокоуровневый набор функций для разработки приложений**

Надстройка над UI библиотекой, которая объединяет и дополняет инструменты для полноценной разработки приложения:

- Встроенный SSR;
- Встроенный маршрутизатор;
- Готовая сборка;
- Упрощенная работа с SEO;
- Понятная работа с данными;

# Что такое Meta Framework?

**Высокоуровневый набор функций для разработки приложений**

Надстройка над UI библиотекой, которая объединяет и дополняет инструменты для полноценной разработки приложения:

- Встроенный SSR;
- Встроенный маршрутизатор;
- Готовая сборка;
- Упрощенная работа с SEO;
- Понятная работа с данными;
- Примеры:
  - RedwoodJS, Remix, Gatsby, Nuxt, Next.js, Vike.

# На какие вопросы отвечает Meta Framework?

# На какие вопросы отвечает Meta Framework?

- Какой стек в проекте?

# На какие вопросы отвечает Meta Framework?

- Какой стек в проекте?
- Какая файловая структура?

# На какие вопросы отвечает Meta Framework?

- Какой стек в проекте?
- Какая файловая структура?
- Какие правила и ограничения на проекте?

# Что же такое Vike?

# Что же такое Vike?

- Модульный Мета-фреймворк;

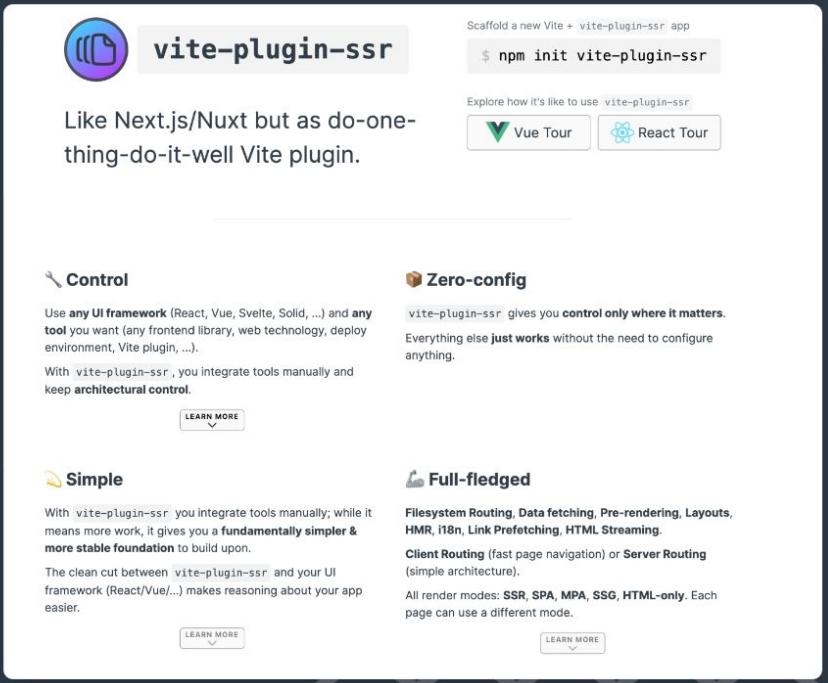
# Что же такое Vike?

- Модульный Мета-фреймворк;
- Архитектурная свобода;

# Что же такое Vike?

- Модульный Мета-фреймворк;
- Архитектурная свобода;
- Возможность очень просто построить свой фреймворк.

# Vite Plugin SSR



The screenshot shows the official landing page for the `vite-plugin-ssr` plugin. At the top, there's a logo featuring a blue circle with a white icon, followed by the text "vite-plugin-ssr". Below the logo, a tagline reads "Like Next.js/Nuxt but as do-one-thing-do-it-well Vite plugin." To the right, there are links to scaffold a new app with `npm init vite-plugin-ssr` and explore it with `Vue Tour` or `React Tour`.

**Control**

Use **any UI framework** (React, Vue, Svelte, Solid, ...) and **any tool** you want (any frontend library, web technology, deploy environment, Vite plugin, ...).

With `vite-plugin-ssr`, you integrate tools manually and keep architectural control.

[LEARN MORE](#)

**Simple**

With `vite-plugin-ssr` you integrate tools manually; while it means more work, it gives you a **fundamentally simpler & more stable foundation** to build upon.

The clean cut between `vite-plugin-ssr` and your UI framework (React/Vue/...) makes reasoning about your app easier.

[LEARN MORE](#)

**Zero-config**

`vite-plugin-ssr` gives you **control only where it matters**. Everything else **just works** without the need to configure anything.

**Full-fledged**

Filesystem Routing, Data fetching, Pre-rendering, Layouts, HMR, i18n, Link Prefetching, HTML Streaming.

Client Routing (fast page navigation) or Server Routing (simple architecture).

All render modes: **SSR, SPA, MPA, SSG, HTML-only**. Each page can use a different mode.

[LEARN MORE](#)

vite-plugin-ssr

# Vite Plugin SSR

- В 2021 году как небольшой плагин для Vite;

The screenshot shows the official website for `vite-plugin-ssr`. At the top, there's a header with the project name and a purple circular icon. Below the header, a main heading says "Like Next.js/Nuxt but as do-one-thing-do-it-well Vite plugin." To the right, there are links for "Scaffold a new Vite + vite-plugin-ssr app" and the command "\$ npm init vite-plugin-ssr". A "Vue Tour" button is also present.

**Control**: Describes the plugin as giving control only where it matters. It mentions integrating tools manually and keeping architectural control. A "LEARN MORE" button is available.

**Zero-config**: States that `vite-plugin-ssr` gives you control only where it matters. Everything else just works without needing to configure anything.

**Simple**: Explains that the plugin integrates tools manually, which means more work but provides a fundamentally simpler & more stable foundation. It highlights the clean cut between the plugin and the UI framework. A "LEARN MORE" button is available.

**Full-fledged**: Lists various features: Filesystem Routing, Data fetching, Pre-rendering, Layouts, HMR, i18n, Link Prefetching, and HTML Streaming. It describes Client Routing (fast page navigation) or Server Routing (simple architecture). It notes that all render modes (SSR, SPA, MPA, SSG, HTML-only) are supported. A "LEARN MORE" button is available.

vite-plugin-ssr

# Vite Plugin SSR

- В 2021 году как небольшой плагин для Vite;
- Инструмент архитектурного контроля;

Scaffold a new Vite + vite-plugin-ssr app  
\$ npm init vite-plugin-ssr

Explore how it's like to use vite-plugin-ssr

[Vue Tour](#) [React Tour](#)

**vite-plugin-ssr**

Like Next.js/Nuxt but as do-one-thing-do-it-well Vite plugin.

**Control**

Use **any UI framework** (React, Vue, Svelte, Solid, ...) and **any tool** you want (any frontend library, web technology, deploy environment, Vite plugin, ...).

With `vite-plugin-ssr`, you integrate tools manually and keep architectural control.

[LEARN MORE](#)

**Zero-config**

`vite-plugin-ssr` gives you **control only where it matters**. Everything else **just works** without the need to configure anything.

**Simple**

With `vite-plugin-ssr` you integrate tools manually; while it means more work, it gives you a **fundamentally simpler & more stable foundation** to build upon.

The clean cut between `vite-plugin-ssr` and your UI framework (React/Vue/...) makes reasoning about your app easier.

[LEARN MORE](#)

**Full-fledged**

**Filesystem Routing, Data fetching, Pre-rendering, Layouts, HMR, i18n, Link Prefetching, HTML Streaming.**

**Client Routing** (fast page navigation) or **Server Routing** (simple architecture).

All render modes: **SSR, SPA, MPA, SSG, HTML-only**. Each page can use a different mode.

[LEARN MORE](#)

vite-plugin-ssr

# Vite Plugin SSR

- В 2021 году как небольшой плагин для Vite;
- Инструмент архитектурного контроля;
- Оказался в разы быстрее по сравнению с Next.js 14;

```
Index stats:  
- 4 entries  
- 4,826 search terms  
- 63,071 bytes per entry  
- 52 bytes per search term  
[vite-plugin-ssr@0.4.85][Warning] `export { onBeforePrerender }` of /src/page
```

```
Days      : 0  
Hours     : 0  
Minutes   : 0  
Seconds   : 7  
Milliseconds : 969  
Ticks     : 79690898  
TotalDays : 9,22348356481481E-05  
TotalHours: 0,0022136360555556  
TotalMinutes: 0,1328181633333333  
TotalSeconds: 7,9690898  
TotalMilliseconds : 7969,0898
```

vite-plugin-ssr

```
▶ homepage ▶ master= ⌂ +16 ~16 -91 ✓ |
```

```
Success: Index built successfully, wrote 252,286 bytes.
```

```
Index stats:
```

```
- 4 entries  
- 4,826 search terms  
- 63,071 bytes per entry  
- 52 bytes per search term
```

```
Days      : 0  
Hours     : 0  
Minutes   : 0  
Seconds   : 37  
Milliseconds : 921  
Ticks     : 379214838  
TotalDays : 0,0004389060625  
TotalHours: 0,0105337455  
TotalMinutes: 0,63202473  
TotalSeconds: 37,9214838  
TotalMilliseconds : 37921,4838
```

nextjs

```
▶ homepage-nextjs ▶ release/v0.1= ⌂ ✓ |
```

# Let me introduce to you – The Future

В 2023 году `vite-plugin-ssr` мигрирует в репозиторий `vike`

# Let me introduce to you – The Future

В 2023 году `vite-plugin-ssr` мигрирует в репозиторий `vike`

-  **Все преимущества плагина — SSG, SSR, HTML-Only, SPA, MPA итп;**

# Let me introduce to you – The Future

В 2023 году `vite-plugin-ssr` мигрирует в репозиторий `vike`

- 📎 Все преимущества плагина — SSG, SSR, HTML-Only, SPA, MPA итп;
- 🎨 Появились экстешены — `vike-react/vike-vue/vike-solid` итп;

# Let me introduce to you – The Future

В 2023 году `vite-plugin-ssr` мигрирует в репозиторий `vike`

- 📎 **Все преимущества плагина** — SSG, SSR, HTML-Only, SPA, MPA итп;
- 🎨 **Появились экстешены** — `vike-react/vike-vue/vike-solid` итп;
- 🤖 **Еще больше контроля** — появились новые хуки ЖЦ и конфигурация на уровне файла `config.ts` ;

# Let me introduce to you – The Future

В 2023 году `vite-plugin-ssr` мигрирует в репозиторий `vike`

- 📈 **Все преимущества плагина** — SSG, SSR, HTML-Only, SPA, MPA итп;
- 🎨 **Появились экстешены** — `vike-react/vike-vue/vike-solid` итп;
- 🧑‍💻 **Еще больше контроля** — появились новые хуки ЖЦ и конфигурация на уровне файла `config.ts` ;
- 🎩 **Пользовательские хуки** — возможность создавать собственные серверные/клиентские хуки;

# Let me introduce to you – The Future

В 2023 году `vite-plugin-ssr` мигрирует в репозиторий `vike`

- 📄 **Все преимущества плагина** — SSG, SSR, HTML-Only, SPA, MPA итп;
- 🎨 **Появились экстешены** — `vike-react/vike-vue/vike-solid` итп;
- 🧑‍💻 **Еще больше контроля** — появились новые хуки ЖЦ и конфигурация на уровне файла `config.ts` ;
- 🎖️ **Пользовательские хуки** — возможность создавать собственные серверные/клиентские хуки;
- 🎵 **Streaming** — поддержка Streaming и RSC (на уровне беты);

# Let me introduce to you – The Future

В 2023 году `vite-plugin-ssr` мигрирует в репозиторий `vike`

- 📄 **Все преимущества плагина** — SSG, SSR, HTML-Only, SPA, MPA итп;
- 🎨 **Появились экстешены** — `vike-react/vike-vue/vike-solid` итп;
- 🧑‍💻 **Еще больше контроля** — появились новые хуки ЖЦ и конфигурация на уровне файла `config.ts` ;
- 🎖️ **Пользовательские хуки** — возможность создавать собственные серверные/клиентские хуки;
- 🎵 **Streaming** — поддержка Streaming и RSC (на уровне беты);
- 🏗️ **Scaffold** — возможность создать приложение с помощью CLI `Bâti` ;

# Let me introduce to you – The Future

В 2023 году `vite-plugin-ssr` мигрирует в репозиторий `vike`

- 📄 **Все преимущества плагина** — SSG, SSR, HTML-Only, SPA, MPA итп;
- 🎨 **Появились экстешены** — `vike-react/vike-vue/vike-solid` итп;
- 🧑‍💻 **Еще больше контроля** — появились новые хуки ЖЦ и конфигурация на уровне файла `config.ts` ;
- 🎖️ **Пользовательские хуки** — возможность создавать собственные серверные/клиентские хуки;
- 🎵 **Streaming** — поддержка Streaming и RSC (на уровне беты);
- 📜 **Scaffold** — возможность создать приложение с помощью CLI `Bâti` ;
- 🔨 **Open Source** — полностью открытый независимый исходный код;

# Bâti

## (фр. "основа / каркас")

The screenshot shows the Bâti web application generator interface. At the top right is the Bâti logo, which consists of a red lightning bolt icon above the word "Bâti". Below the logo is a brief description: "Scaffolds a web app using Vike, React, Mantine, Telefunc, Express, Google Analytics, Vercel, ESLint, Prettier, and Sentry." A command-line input field contains the command: "npm create vike@latest --react --mantine --telefunc --express --google-analytics --vercel --eslint --prettier --sentry". To the right of the input field is a "Try me in Stackblitz" button with a lightning bolt icon.

The interface is organized into several sections:

- Presets:** Buttons for Frontend, Full-stack, Next.js, Nuxt, and CMS.
- Frontend:**
  - Frontend Framework (required):** Buttons for Vike (selected), React, Vue, and SolidJS.
  - UI Framework (required):** Buttons for TailwindCSS (selected), Compiled, daisyUI, shadcnui, and Mantine.
- Data:**
  - Auth:** Buttons for Auth.js (selected) and Auth0.
  - Data fetching:** Buttons for Telefunc (selected), GraphQL, and ts-rest.
- Server:**
  - Server:** Buttons for Hono (selected), h3, Express, and Fastify.
  - Database:** Buttons for Drizzle, SQLite, and Prisma.
- Deployment:**
  - Hosting:** Buttons for Cloudflare (selected), Vercel, Now, and AWS.
- Utilities:**
  - Linter:** Buttons for ESLint (selected), Prettier, and Biome.
  - Analytics:** Buttons for Plausible.io (selected), Google Analytics, and Segment.
  - Error tracking:** Buttons for Sentry (selected) and LogRocket.

<https://batijs.dev/>

# Bâti (фр. "основа / каркас")

- CLI для скаффолдинга вашего приложения;

The screenshot shows the Bâti CLI interface, which generates web applications using Vite, React, Mantine, Telefunc, Express, Google Analytics, Vercel, ESLint, Prettier, and Sentry. The interface includes a command-line input field with the command: `>_ npm create vite@latest --react --mantine --telefunc --express --google-analytics --vercel --eslint --prettier --sentry`. Below this is a navigation bar with tabs: Presets, Frontend, Full-stack, Next.js, Nuxt, and CMS. The Frontend section contains fields for Frontend Framework (Vite), UI Framework (React, Vue, SolidJS), CSS (TailwindCSS, Compiled), and UI Component Libraries (daisyUI, shadcnui, Mantine). The Data section includes Auth (Auth.js, Auth0) and Data fetching (Telefunc, GraphQL, ts-rest). The Server section lists Hono, h3, Express, and Fastify. The Database section includes Drizzle, SQLite, and Prisma. The Utilities section includes Linter (ESLint, Prettier, Biome), Analytics (Plausible.io, Google Analytics, Segment), and Error tracking (Sentry, LogRocket).

<https://batijs.dev/>

# Bâti (фр. "основа / каркас")

- CLI для скраффолдинга вашего приложения;
- Есть пресеты -

Frontend / Full-Stack / Next.js / Nuxt ;

The screenshot shows the Bâti web application interface. At the top, there's a logo featuring a red lightning bolt icon next to the word "Bâti". Below the logo, a sub-header reads "Scaffolds a web app using Vike, React, Mantine, Telefunc, Express, Google Analytics, Vercel, ESLint, Prettier, and Sentry." A command-line input field contains the command: "npm create vike@latest --react --mantine --telefunc --express --google-analytics --vercel --eslint --prettier --sentry". To the right of the input field is a "Try me in Stackblitz" button.

The main interface is divided into several sections:

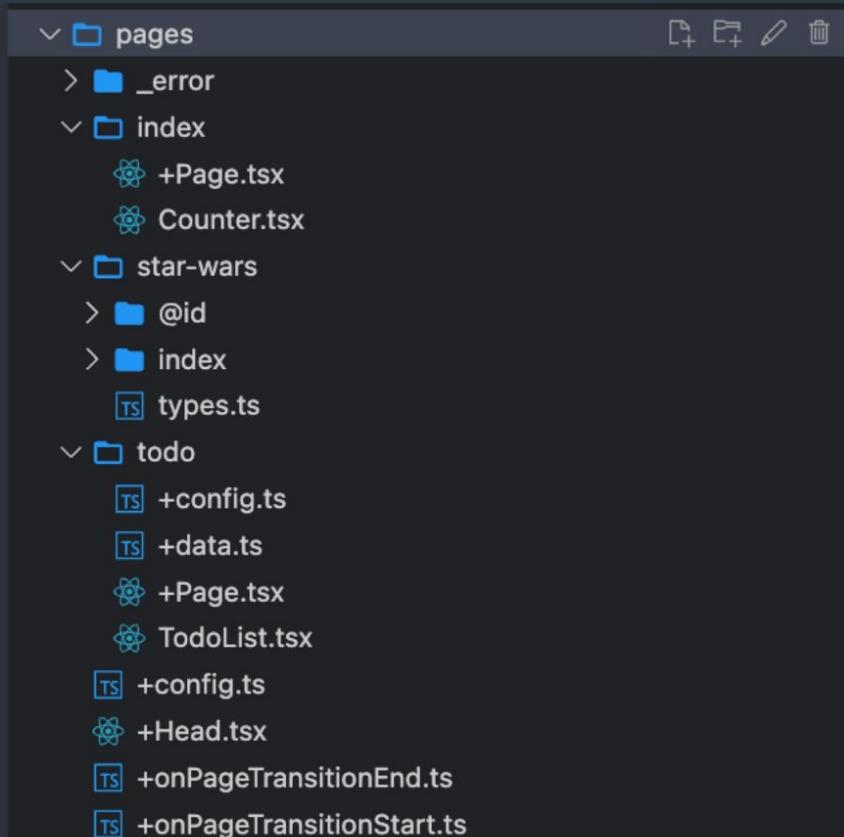
- Presets:** A tab bar with options: Presets, Frontend, Full-stack, Next.js, Nuxt, CMS.
- Frontend:** A section for configuring the Frontend Framework (Vike), UI Framework (React, Vue, SolidJS), CSS (TailwindCSS, Compiled), and UI Component Libraries (daisyUI, shadcnui, Mantine).
- Data:** A section for Data fetching (Telefunc, GraphQL, ts-rest).
- Auth:** A section for Auth (Auth.js, Auth0).
- Server:** A section for Server (Hono, h3, Express, Fastify).
- Database:** A section for Database (Drizzle, SQLite, Prisma).
- Deployment:** A section for Hosting (Cloudflare, Vercel, Now, AWS).
- Utilities:** A section for Utilities (ESLint, Prettier, Biome).
- Linter:** A section for Linter (ESLint, Prettier, Biome).
- Analytics:** A section for Analytics (Plausible.io, Google Analytics, Segment).
- Error tracking:** A section for Error tracking (Sentry, LogRocket).

<https://batijs.dev/>

# Begin

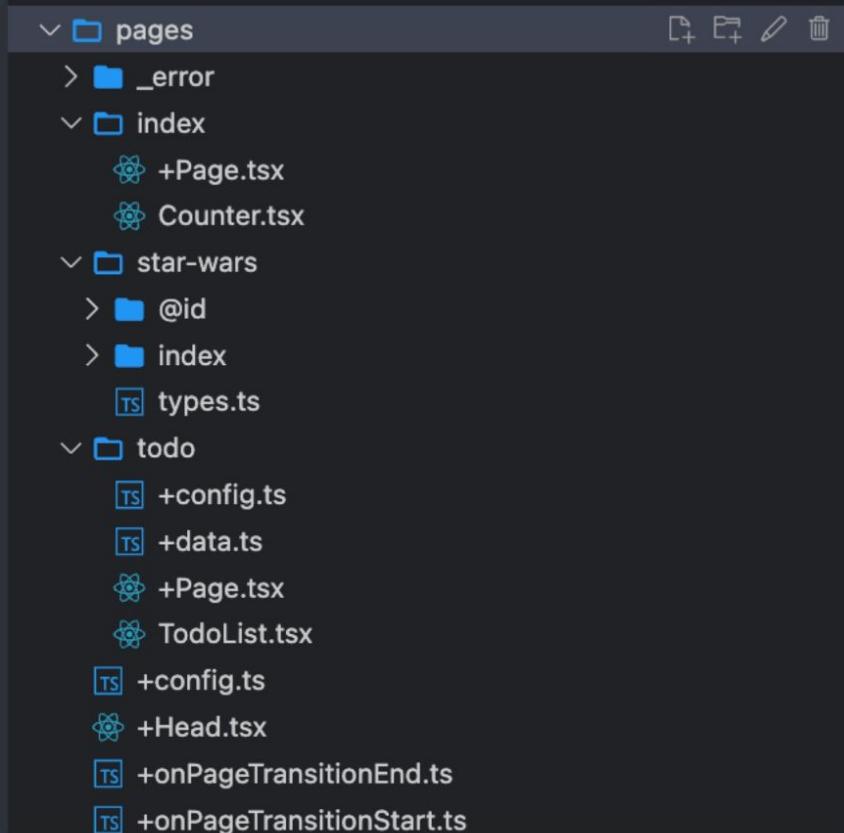
```
pnpm create vite@latest --react --mantine --express --eslint
```

# Из чего состоит приложение



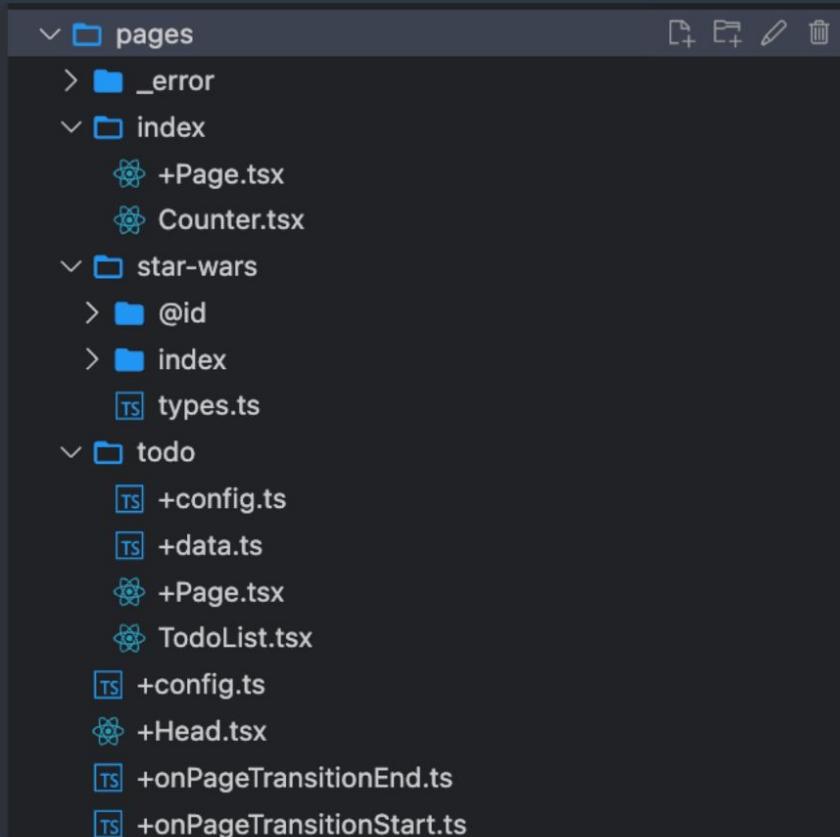
# Из чего состоит приложение

- Optional zero-config приложение;



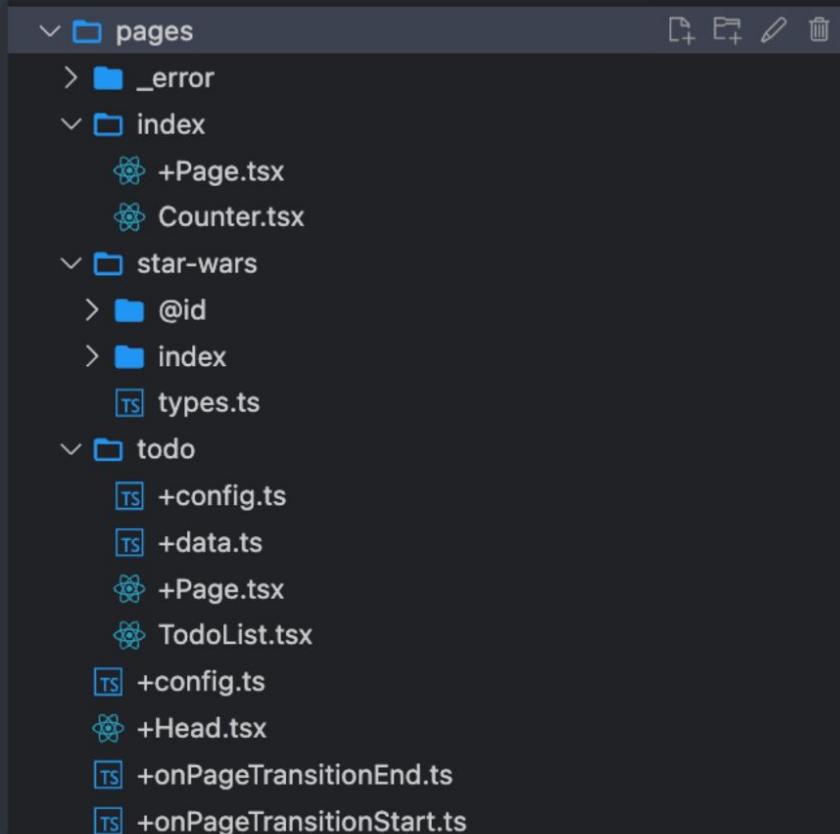
# Из чего состоит приложение

- Optional zero-config приложение;
- Набор правил по именованию сущностей;



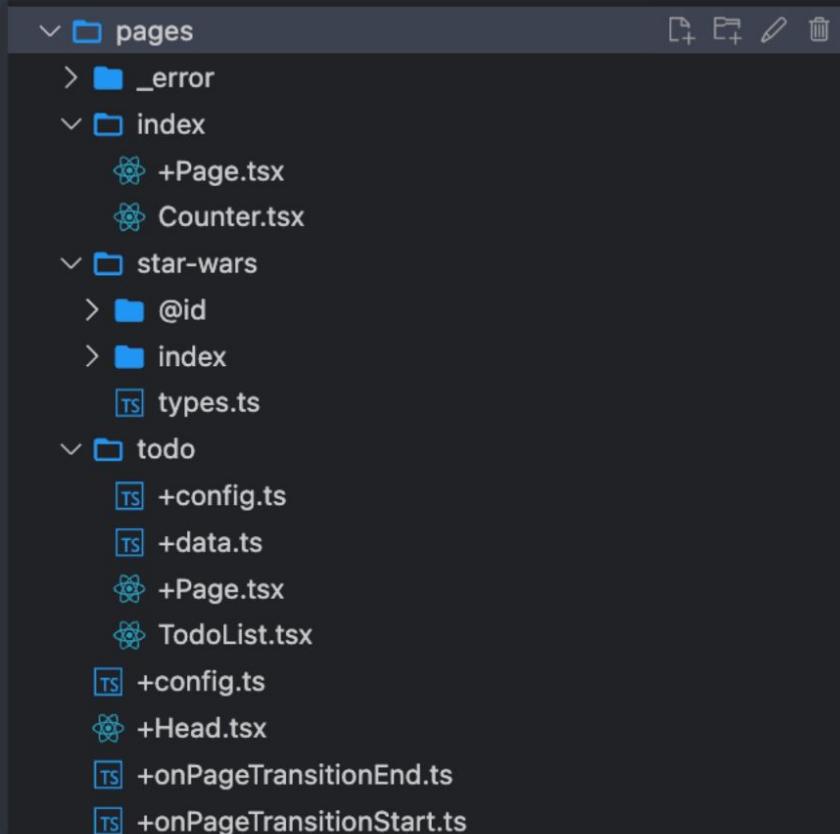
# Из чего состоит приложение

- Optional zero-config приложение;
- Набор правил по именованию сущностей;
- Конфиг `+config.ts` ;



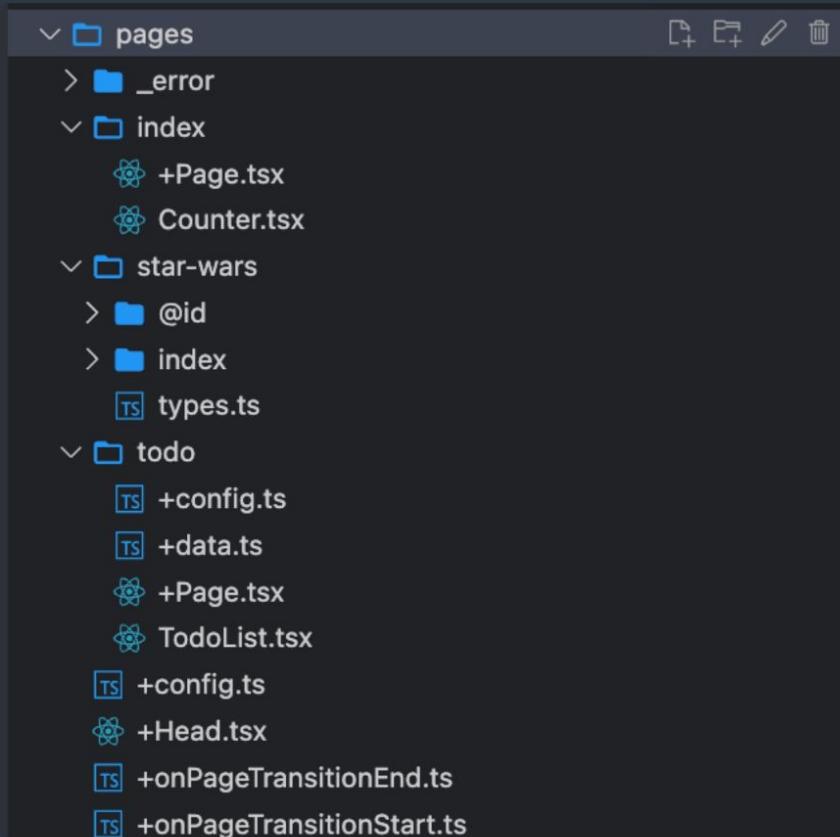
# Из чего состоит приложение

- Optional zero-config приложение;
- Набор правил по именованию сущностей;
- Конфиг `+config.ts` ;
- File-based роутинг в каталоге `pages` ;



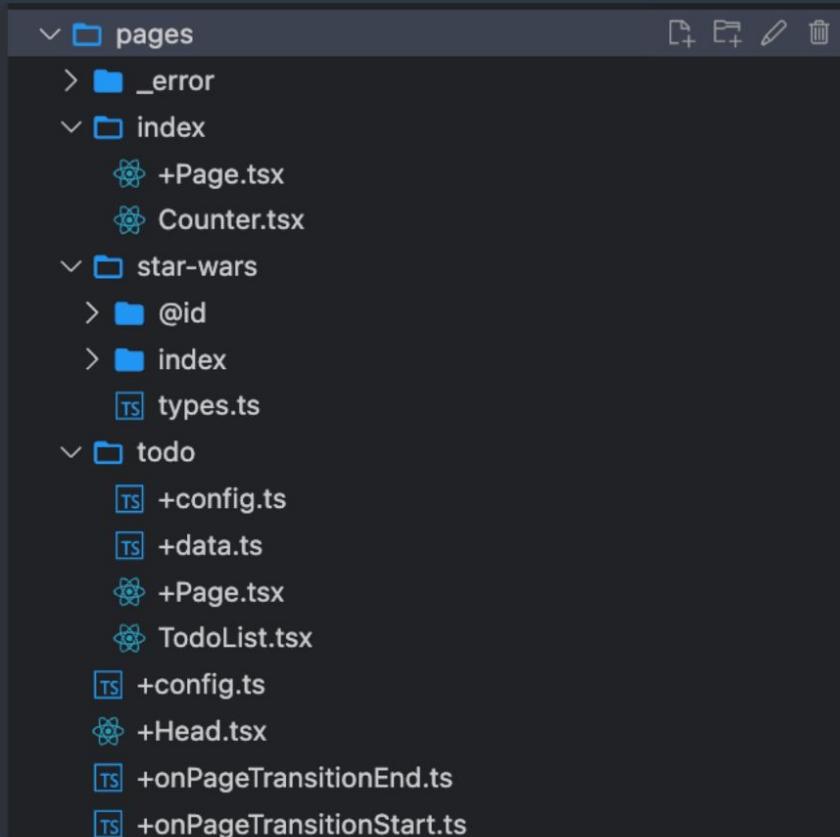
# Из чего состоит приложение

- Optional zero-config приложение;
- Набор правил по именованию сущностей;
- Конфиг `+config.ts` ;
- File-based роутинг в каталоге `pages` ;
- Страницы внутри каталогов `+Page.tsx` ;



# Из чего состоит приложение

- Optional zero-config приложение;
- Набор правил по именованию сущностей;
- Конфиг `+config.ts` ;
- File-based роутинг в каталоге `pages` ;
- Страницы внутри каталогов `+Page.tsx` ;
- Примеры vike-хуков.



# Изначально Zero-Config

```
pnpm run dev
```

# App Starter

```
"dev": "tsx ./express-entry.ts",  
"build": "vike build",  
"start": "pnpm run build && cross-env NODE_ENV=production tsx ./express-entry.ts"
```

# Let's begin

```
8 // express-entry.ts
9 async function startServer() {
10   const app = express();
11
12   if (isProduction) {
13     app.use(express.static(`.${root}/dist/client`));
14   } else {
15     const { devMiddleware } = await createDevMiddleware({ root });
16     app.use(devMiddleware);
17   }
18
19   /* Перехватываем запросы */
20   app.all('*', createHandler(vikeHandler()));
21
22   // Если есть RPC
23   app.post('/api/todo/create', createHandler(createTodoHandler()));
24
25   return app;
26 }
27 export default (await startServer());
```

# Let's begin

```
8 // express-entry.ts
9 async function startServer() {
10     const app = express();
11
12     if (isProduction) {
13         app.use(express.static(`.${root}/dist/client`));
14     } else {
15         const { devMiddleware } = await createDevMiddleware({ root });
16         app.use(devMiddleware);
17     }
18
19     /* Перехватываем запросы */
20     app.all('*', createHandler(vikeHandler()));
21
22     // Если есть RPC
23     app.post('/api/todo/create', createHandler(createTodoHandler()));
24
25     return app;
26 }
27 export default (await startServer());
```

# Let's begin

```
8 // express-entry.ts
9 async function startServer() {
10     const app = express();
11
12     if (isProduction) {
13         app.use(express.static(`.${root}/dist/client`));
14     } else {
15         const { devMiddleware } = await createDevMiddleware({ root });
16         app.use(devMiddleware);
17     }
18
19     /* Перехватываем запросы */
20     app.all('*', createHandler(vikeHandler()));
21
22     // Если есть RPC
23     app.post('/api/todo/create', createHandler(createTodoHandler()));
24
25     return app;
26 }
27 export default (await startServer());
```

# Let's begin

```
8 // express-entry.ts
9 async function startServer() {
10   const app = express();
11
12   if (isProduction) {
13     app.use(express.static(`.${root}/dist/client`));
14   } else {
15     const { devMiddleware } = await createDevMiddleware({ root });
16     app.use(devMiddleware);
17   }
18
19   /* Перехватываем запросы */
20   app.all('*', createHandler(vikeHandler()));
21
22   // Если есть RPC
23   app.post('/api/todo/create', createHandler(createTodoHandler()));
24
25   return app;
26 }
27 export default (await startServer());
```

# Let's begin

```
8 // express-entry.ts
9 async function startServer() {
10   const app = express();
11
12   if (isProduction) {
13     app.use(express.static(`.${root}/dist/client`));
14   } else {
15     const { devMiddleware } = await createDevMiddleware({ root });
16     app.use(devMiddleware);
17   }
18
19   /* Перехватываем запросы */
20   app.all('*', createHandler(vikeHandler()));
21
22   // Если есть RPC
23   app.post('/api/todo/create', createHandler(createTodoHandler()));
24
25   return app;
26 }
27 export default (await startServer());
```

# Let's begin

```
8 // express-entry.ts
9 async function startServer() {
10   const app = express();
11
12   if (isProduction) {
13     app.use(express.static(`.${root}/dist/client`));
14   } else {
15     const { devMiddleware } = await createDevMiddleware({ root });
16     app.use(devMiddleware);
17   }
18
19   /* Перехватываем запросы */
20   app.all('*', createHandler(vikeHandler()));
21
22   // Если есть RPC
23   app.post('/api/todo/create', createHandler(createTodoHandler()));
24
25   return app;
26 }
27 export default (await startServer());
```

# Let's begin

```
8 // express-entry.ts
9 async function startServer() {
10   const app = express();
11
12   if (isProduction) {
13     app.use(express.static(`.${root}/dist/client`));
14   } else {
15     const { devMiddleware } = await createDevMiddleware({ root });
16     app.use(devMiddleware);
17   }
18
19   /* Перехватываем запросы */
20   app.all('*', createHandler(vikeHandler()));
21
22   // Если есть RPC
23   app.post('/api/todo/create', createHandler(createTodoHandler()));
24
25   return app;
26 }
27 export default (await startServer());
```

# Let's begin

```
8 // vike-handler.ts
9 export const vikeHandler: Get<[], UniversalHandler> = () => async (request, context) => {
10   const UA = request.headers.get('user-agent');
11   const device = determineLayoutTypeFromUserAgent(UA ?? '');
12   const isMobile =
13     Boolean(UA?.match(/Android|BlackBerry|iPhone|iPad|iPod|Opera Mini|IEMobile|WPDesktop/i));
14
15   const pageContextInit = {
16     ...context, ...request, isMobile, device
17   };
18   const pageContext = await renderPage(pageContextInit);
19   const response = pageContext.httpResponse;
20
21   const { readable, writable } = new TransformStream();
22
23   response?.pipe(writable);
24
25   return new Response(readable, {
26     status: response?.statusCode,
27     headers: response?.headers,
28   });
29};
```

# Let's begin

```
8 // vike-handler.ts
9 export const vikeHandler: Get<[], UniversalHandler> = () => async (request, context) => {
10   const UA = request.headers.get('user-agent');
11   const device = determineLayoutTypeFromUserAgent(UA ?? '');
12   const isMobile =
13     Boolean(UA?.match(/Android|BlackBerry|iPhone|iPad|iPod|Opera Mini|IEMobile|WPDesktop/i));
14
15   const pageContextInit = {
16     ...context, ...request, isMobile, device
17   };
18   const pageContext = await renderPage(pageContextInit);
19   const response = pageContext.httpResponse;
20
21   const { readable, writable } = new TransformStream();
22
23   response?.pipe(writable);
24
25   return new Response(readable, {
26     status: response?.statusCode,
27     headers: response?.headers,
28   });
29};
```

# Let's begin

```
8 // vike-handler.ts
9 export const vikeHandler: Get<[], UniversalHandler> = () => async (request, context) => {
10   const UA = request.headers.get('user-agent');
11   const device = determineLayoutTypeFromUserAgent(UA ?? '');
12   const isMobile =
13     Boolean(UA?.match(/Android|BlackBerry|iPhone|iPad|iPod|Opera Mini|IEMobile|WPDesktop/i));
14
15   const pageContextInit = {
16     ...context, ...request, isMobile, device
17   };
18   const pageContext = await renderPage(pageContextInit);
19   const response = pageContext.httpResponse;
20
21   const { readable, writable } = new TransformStream();
22
23   response?.pipe(writable);
24
25   return new Response(readable, {
26     status: response?.statusCode,
27     headers: response?.headers,
28   });
29};
```

# Let's begin

```
8 // vike-handler.ts
9 export const vikeHandler: Get<[], UniversalHandler> = () => async (request, context) => {
10   const UA = request.headers.get('user-agent');
11   const device = determineLayoutTypeFromUserAgent(UA ?? '');
12   const isMobile =
13     Boolean(UA?.match(/Android|BlackBerry|iPhone|iPad|iPod|Opera Mini|IEMobile|WPDesktop/i));
14
15   const pageContextInit = {
16     ...context, ...request, isMobile, device
17   };
18   const pageContext = await renderPage(pageContextInit);
19   const response = pageContext.httpResponse;
20
21   const { readable, writable } = new TransformStream();
22
23   response?.pipe(writable);
24
25   return new Response(readable, {
26     status: response?.statusCode,
27     headers: response?.headers,
28   });
29};
```

# pageContext

- Содержит информацию о текущей странице:

```
const pageContext = {  
  routeParams,  
  urlOriginal,  
  urlPathname,  
  urlParsed: { pathname, search, hash, ... }  
  headers,  
  config,  
  isClientSide,  
  isHydration,  
  ...  
}
```

# Start App

```
"dev": "tsx ./express-entry.ts",
"build": "vike build",
"start": "pnpm run build && cross-env NODE_ENV=production tsx ./express-entry.ts"
```

# Start App

```
"dev": "tsx ./express-entry.ts",
"build": "vike build",
"start": "pnpm run build && cross-env NODE_ENV=production tsx ./express-entry.ts"
```

- Запускаем express-entry.ts ;

# Start App

```
"dev": "tsx ./express-entry.ts",
"build": "vike build",
"start": "pnpm run build && cross-env NODE_ENV=production tsx ./express-entry.ts"
```

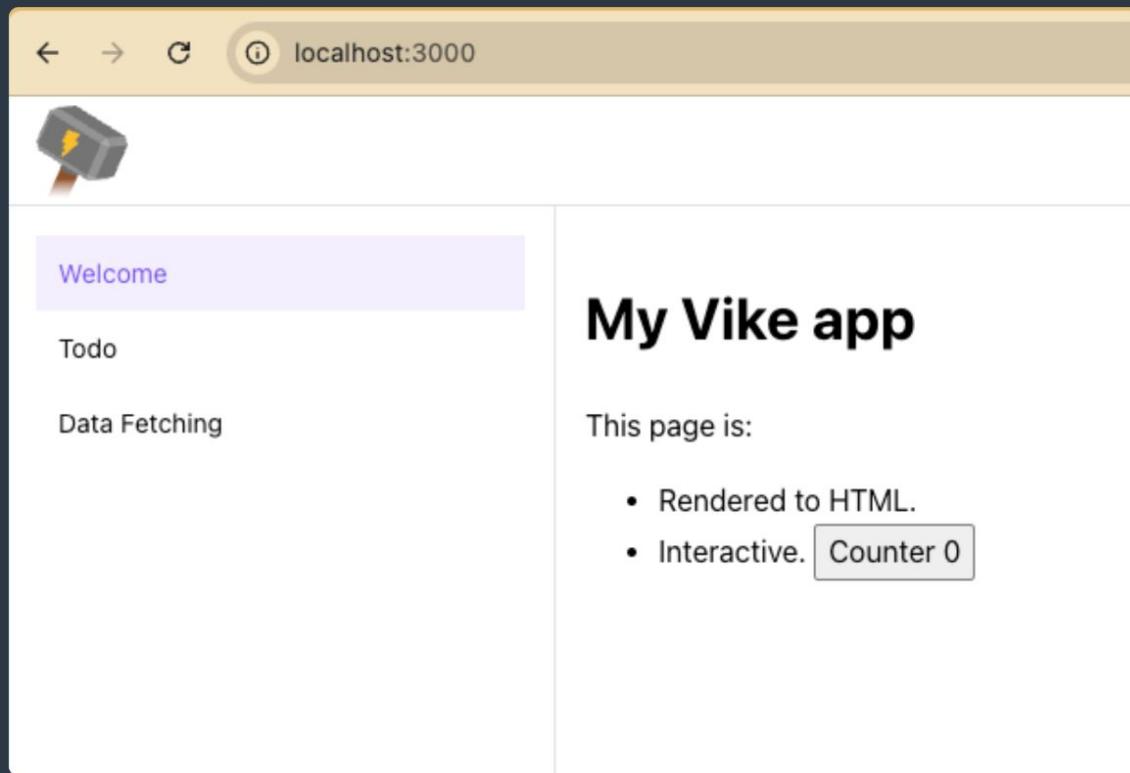
- Запускаем express-entry.ts ;
- Сервер вызывает vikeHandler ;

# Start App

```
"dev": "tsx ./express-entry.ts",
"build": "vike build",
"start": "pnpm run build && cross-env NODE_ENV=production tsx ./express-entry.ts"
```

- Запускаем `express-entry.ts` ;
- Сервер вызывает `vikeHandler` ;
- Запускается наше приложение.

# Welcome



A screenshot of a web browser window titled "localhost:3000". The browser has a light blue header bar with back, forward, and refresh buttons. The main content area shows a sidebar on the left with a dark grey background and white text. The sidebar contains three items: "Welcome" (highlighted with a purple background), "Todo", and "Data Fetching". To the right of the sidebar, the main content area has a white background. It features a large, bold, black heading "My Vike app". Below the heading, the text "This page is:" is followed by a bulleted list: "Rendered to HTML." and "Interactive. Counter 0". A small, rounded rectangular button with a thin black border and white text is positioned next to the "Interactive" text.

← → ⌛ ⓘ localhost:3000

Welcome

Todo

Data Fetching

# My Vike app

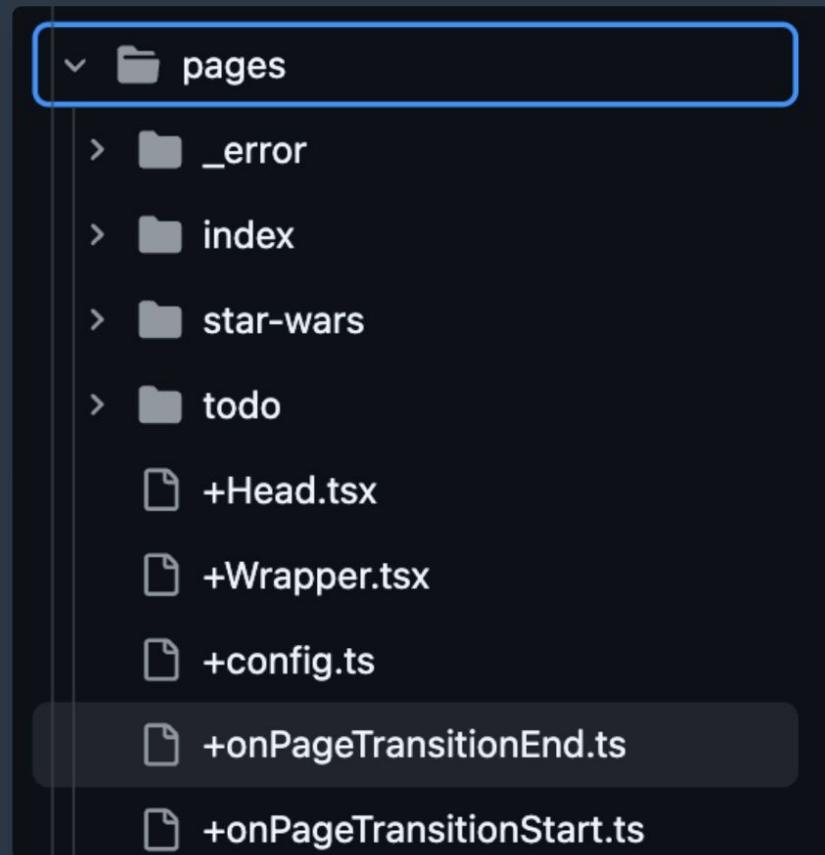
This page is:

- Rendered to HTML.
- Interactive. Counter 0

# Из чего состоит приложение?

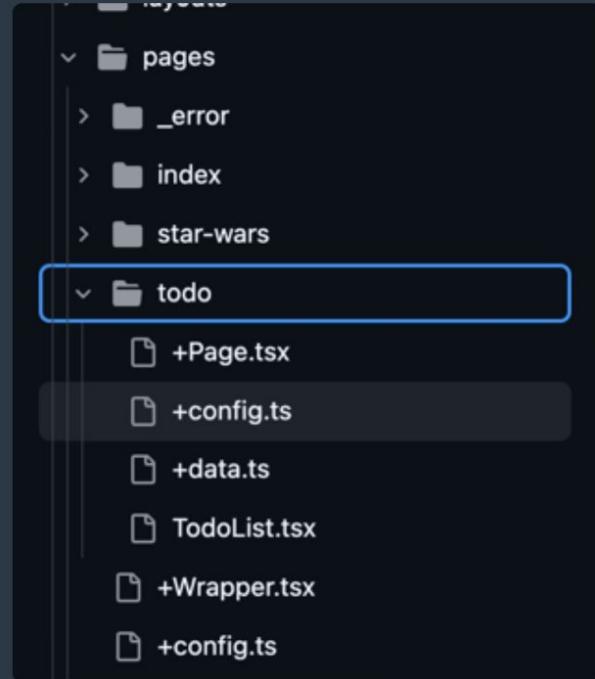
Вообще vike - *zero-config*, но мы же хотим гибкости:

- Корневая папка `pages` с настройками;
- Набор built-in хуков для управления ЖЦ;
- Набор правил и нейминг для хуков, роутинга и страниц с ошибками.
- Набор настроек (`+config`) для тонкой настройки поведения приложения;



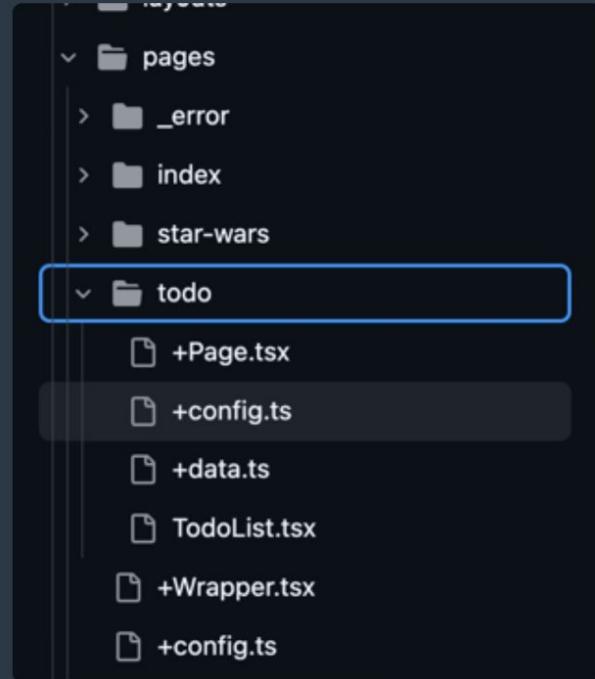
# +config.ts

```
4 // Default config (can be overridden by pages)
5 export default {
6     // base options
7     extends: [vikeReact], // vikeVue, vikeReactQuery etc ...
8     reactStrictMode: false,
9     cacheControl: "public, max-age=86400",
10    prefetchStaticAssets: "viewport",
11    ssr: true, // true by default
12
13    // Wrappers
14    Layout: RootLayout, // Layout всего приложения
15    Head: HeadDefault, // Дефолтный <head />
16
17    // meta tags
18    lang: "en",
19    title: "My Vike App",
20    description: "Demo showcasing Vike",
21
22    // meta info
23    passToClient: ['scopeValues', 'device', 'isMobile'],
24
25 } satisfies Config;
```



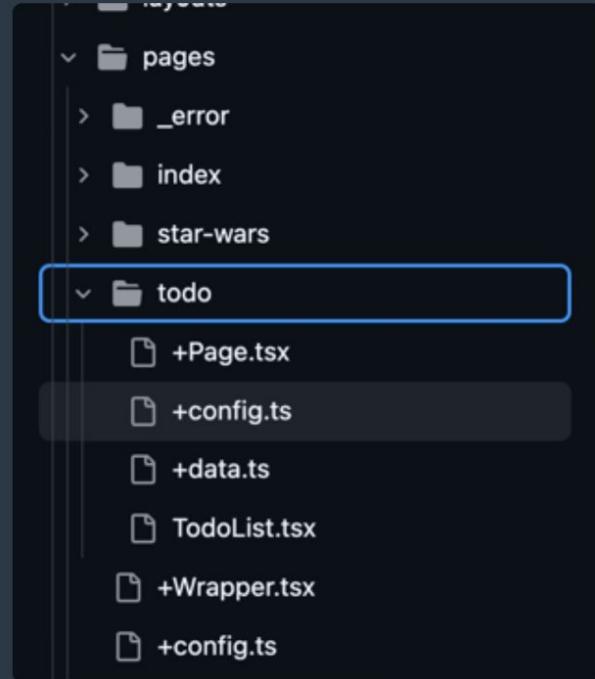
# +config.ts

```
4 // Default config (can be overridden by pages)
5 export default {
6     // base options
7     extends: [vikeReact], // vikeVue, vikeReactQuery etc ...
8     reactStrictMode: false,
9     cacheControl: "public, max-age=86400",
10    prefetchStaticAssets: "viewport",
11    ssr: true, // true by default
12
13    // Wrappers
14    Layout: RootLayout, // Layout всего приложения
15    Head: HeadDefault, // Дефолтный <head />
16
17    // meta tags
18    lang: "en",
19    title: "My Vike App",
20    description: "Demo showcasing Vike",
21
22    // meta info
23    passToClient: ['scopeValues', 'device', 'isMobile'],
24
25 } satisfies Config;
```



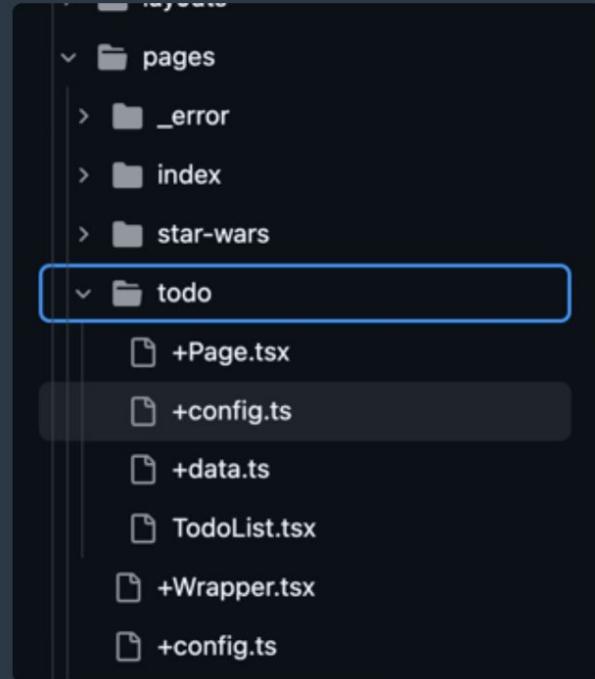
# +config.ts

```
4 // Default config (can be overridden by pages)
5 export default {
6     // base options
7     extends: [vikeReact], // vikeVue, vikeReactQuery etc ...
8     reactStrictMode: false,
9     cacheControl: "public, max-age=86400",
10    prefetchStaticAssets: "viewport",
11    ssr: true, // true by default
12
13    // Wrappers
14    Layout: RootLayout, // Layout всего приложения
15    Head: HeadDefault, // Дефолтный <head />
16
17    // meta tags
18    lang: "en",
19    title: "My Vike App",
20    description: "Demo showcasing Vike",
21
22    // meta info
23    passToClient: ['scopeValues', 'device', 'isMobile'],
24
25 } satisfies Config;
```



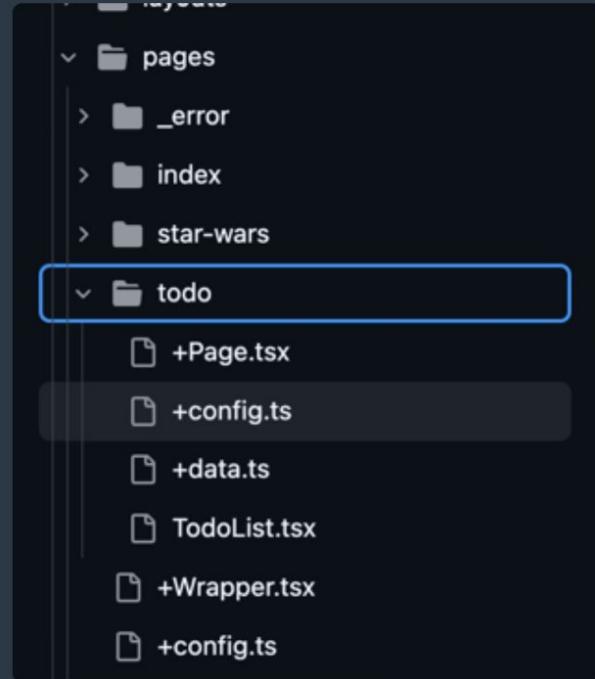
# +config.ts

```
4 // Default config (can be overridden by pages)
5 export default {
6     // base options
7     extends: [vikeReact], // vikeVue, vikeReactQuery etc ...
8     reactStrictMode: false,
9     cacheControl: "public, max-age=86400",
10    prefetchStaticAssets: "viewport",
11    ssr: true, // true by default
12
13    // Wrappers
14    Layout: RootLayout, // Layout всего приложения
15    Head: HeadDefault, // Дefолтный <head />
16
17    // meta tags
18    lang: "en",
19    title: "My Vike App",
20    description: "Demo showcasing Vike",
21
22    // meta info
23    passToClient: ['scopeValues', 'device', 'isMobile'],
24
25 } satisfies Config;
```



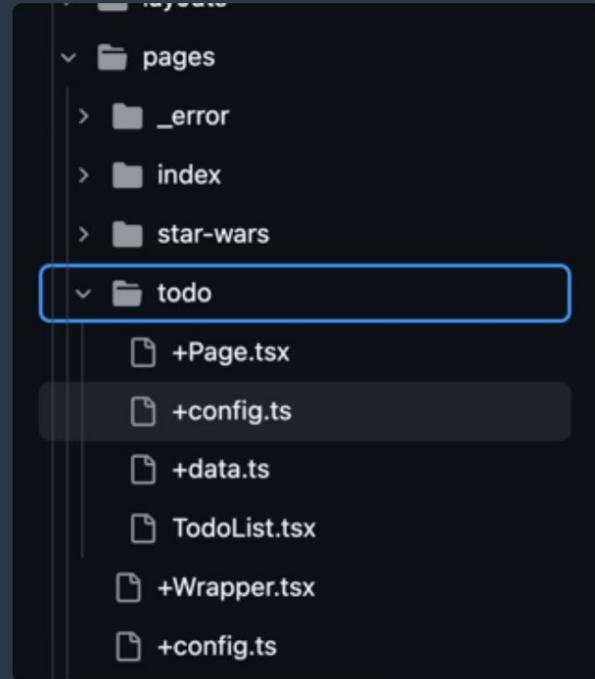
# +config.ts

```
4 // Default config (can be overridden by pages)
5 export default {
6     // base options
7     extends: [vikeReact], // vikeVue, vikeReactQuery etc ...
8     reactStrictMode: false,
9     cacheControl: "public, max-age=86400",
10    prefetchStaticAssets: "viewport",
11    ssr: true, // true by default
12
13    // Wrappers
14    Layout: RootLayout, // Layout всего приложения
15    Head: HeadDefault, // Дефолтный <head />
16
17    // meta tags
18    lang: "en",
19    title: "My Vike App",
20    description: "Demo showcasing Vike",
21
22    // meta info
23    passToClient: ['scopeValues', 'device', 'isMobile'],
24
25 } satisfies Config;
```



# +config.ts

```
4 // Default config (can be overridden by pages)
5 export default {
6     // base options
7     extends: [vikeReact], // vikeVue, vikeReactQuery etc ...
8     reactStrictMode: false,
9     cacheControl: "public, max-age=86400",
10    prefetchStaticAssets: "viewport",
11    ssr: true, // true by default
12
13    // Wrappers
14    Layout: RootLayout, // Layout всего приложения
15    Head: HeadDefault, // Дефолтный <head />
16
17    // meta tags
18    lang: "en",
19    title: "My Vike App",
20    description: "Demo showcasing Vike",
21
22    // meta info
23    passToClient: ['scopeValues', 'device', 'isMobile'],
24
25 } satisfies Config;
```



# passToClient

```
4 // vike-handler.ts
5 export const vikeHandler: Get<[], UniversalHandler> = () => async (request, context) => {
6   const UA = request.headers.get('user-agent');
7   const device = determineLayoutTypeFromUserAgent(UA ?? '');
8   const isMobile =
9     Boolean(UA?.match(/Android|BlackBerry|iPhone|iPad|iPod|Opera Mini|IEMobile|WPDesktop/i));
10
11  const pageContextInit = {
12    ...context, ...request, isMobile, device
13  };
14  const pageContext = await renderPage(pageContextInit);
15  const response = pageContext.httpResponse;
16
17  const { readable, writable } = new TransformStream();
18
19  response?.pipe(writable);
20
21  return new Response(readable, {
22    status: response?.statusCode,
23    headers: response?.headers,
24  });
25};
```

# passToClient

```
4 // vike-handler.ts
5 export const vikeHandler: Get<[], UniversalHandler> = () => async (request, context) => {
6   const UA = request.headers.get('user-agent');
7   const device = determineLayoutTypeFromUserAgent(UA ?? '');
8   const isMobile =
9     Boolean(UA?.match(/Android|BlackBerry|iPhone|iPad|iPod|Opera Mini|IEMobile|WPDesktop/i));
10
11  const pageContextInit = {
12    ...context, ...request, isMobile, device
13  };
14  const pageContext = await renderPage(pageContextInit);
15  const response = pageContext.httpResponse;
16
17  const { readable, writable } = new TransformStream();
18
19  response?.pipe(writable);
20
21  return new Response(readable, {
22    status: response?.statusCode,
23    headers: response?.headers,
24  });
25};
```

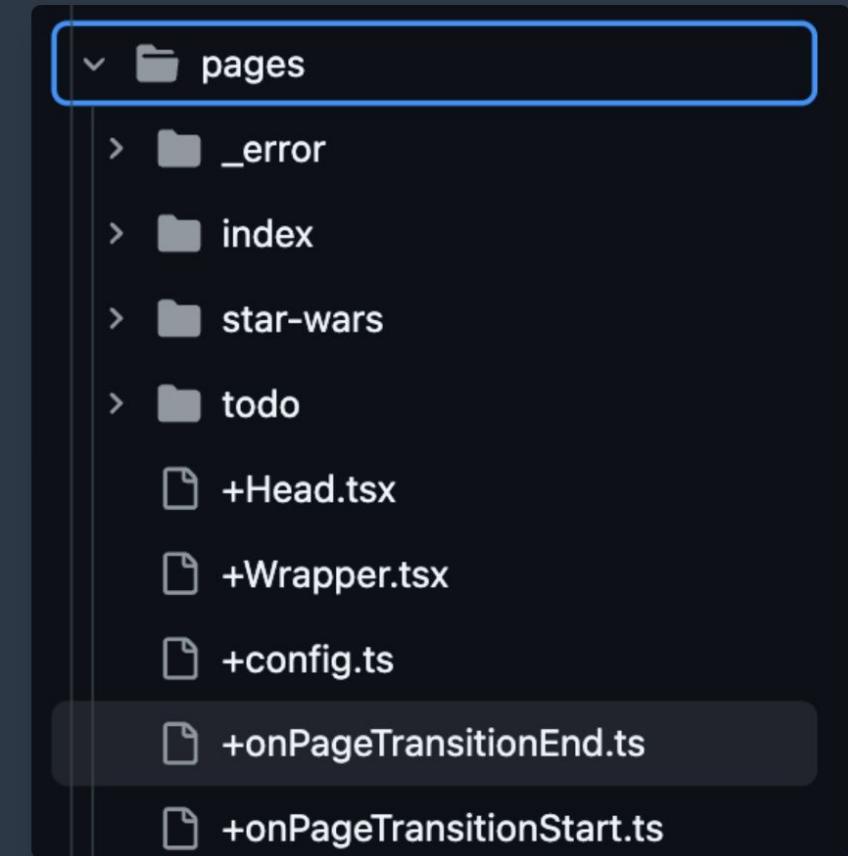
# passToClient

```
4 // pages/index/+Page.tsx
5 import { usePageContext } from "vike-react/usePageContext";
6
7 export default function IndexPage() {
8     const { isMobile } = usePageContext();
9     console.log(isMobile); // false
10
11     return (/**)
12 }
```

# passToClient

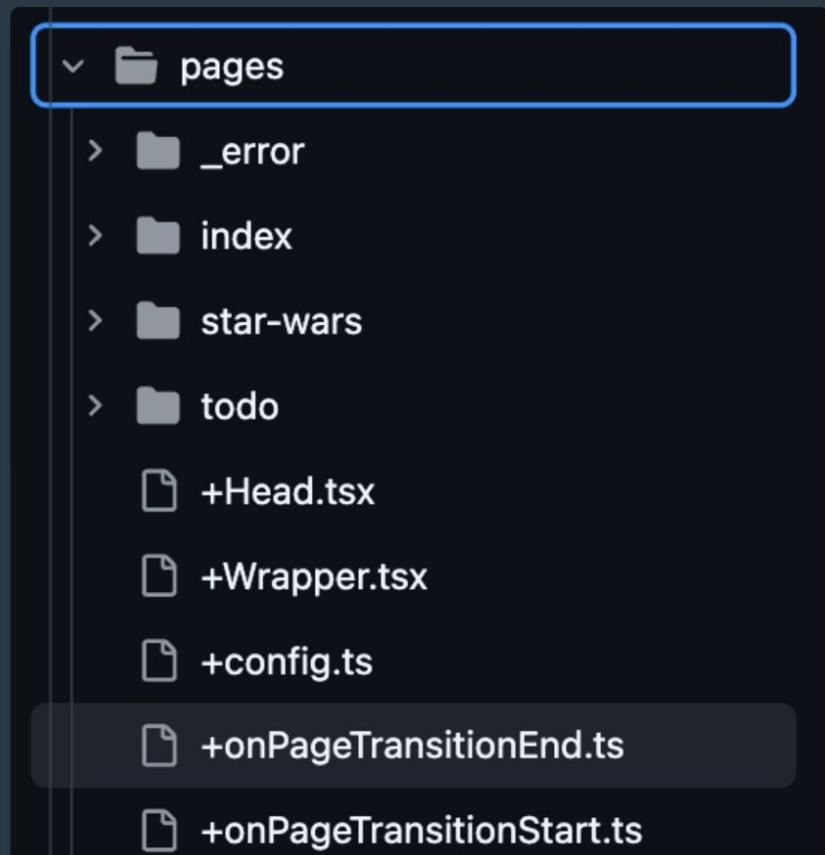
```
4 // pages/index/+Page.tsx
5 import { usePageContext } from "vike-react/usePageContext";
6
7 export default function IndexPage() {
8     const { isMobile } = usePageContext();
9     console.log(isMobile); // false
10
11     return (/**/)
12 }
```

# Что еще?



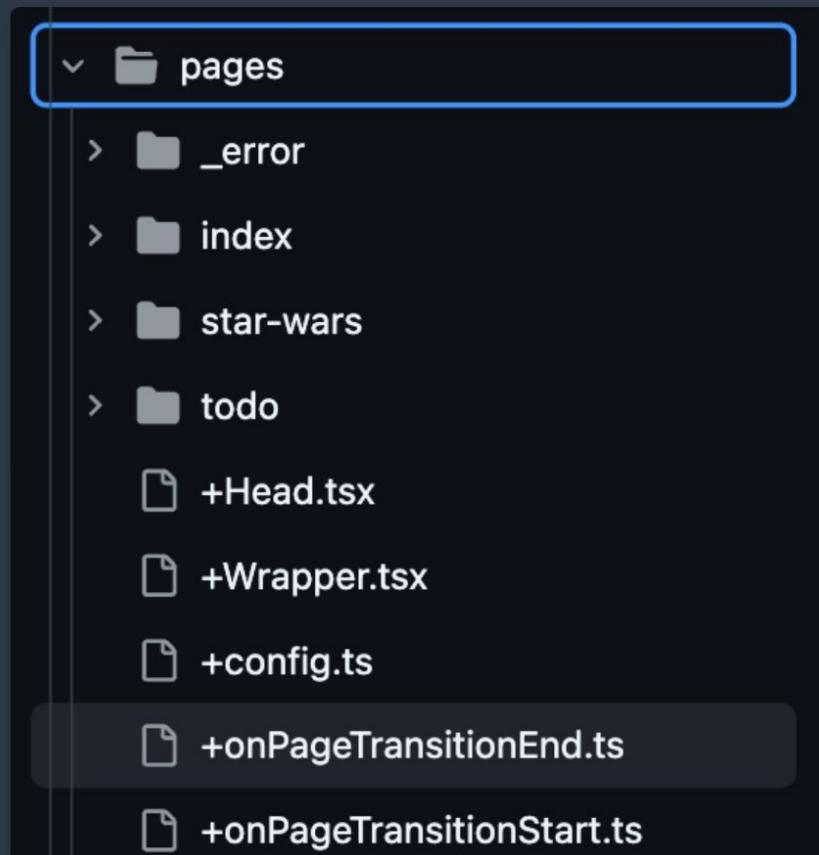
# Что еще?

- Хуки `+onPageTransitionStart` и `+onPageTransitionEnd` отвечают за трекинг перехода по страницам;



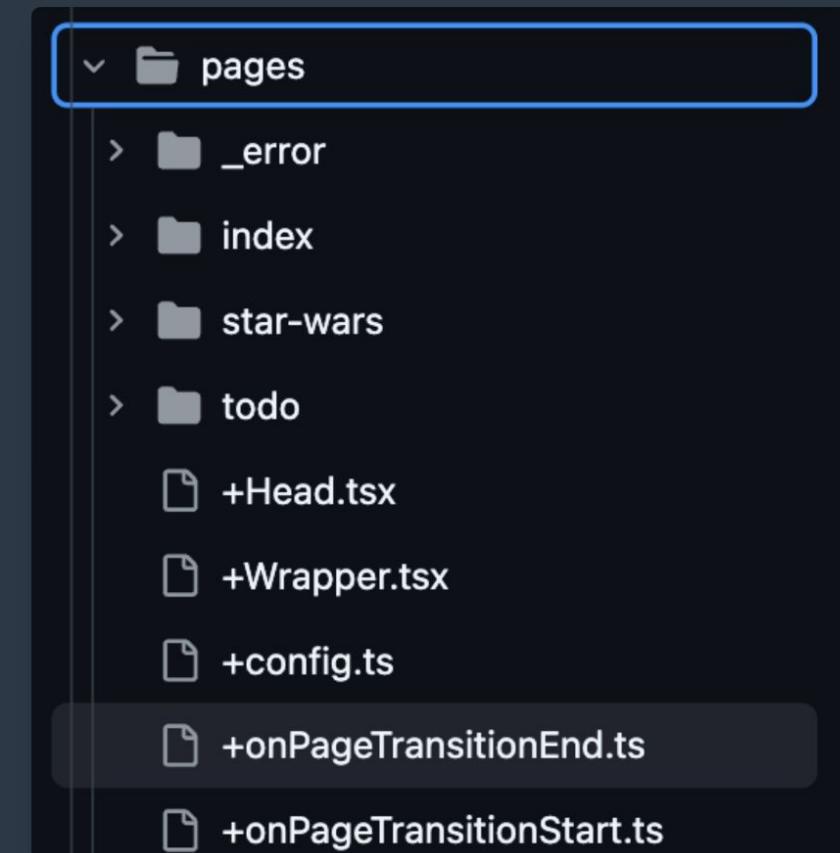
# Что еще?

- Хуки `+onPageTransitionStart` и `+onPageTransitionEnd` отвечают за трекинг перехода по страницам;
- `+onBeforeRender` и `+onBeforeRenderClient` помогут сделать что-то перед рендером на сервере и перед рендером на клиенте - например синхронизировать Store;



# Что еще?

- Хуки `+onPageTransitionStart` и `+onPageTransitionEnd` отвечают за трекинг перехода по страницам;
- `+onBeforeRender` и `+onBeforeRenderClient` помогут сделать что-то перед рендером на сервере и перед рендером на клиенте - например синхронизировать Store;
- `+Wrapper` - оборачивает всё приложение и `Layout` - обычно там определяют подключение глобальных провайдеров;



# Но самое интересное

# Кастомные хуки

```
4 // pages/+config.ts
5
6 export default {
7     ... config,
8     meta: {
9         // Event - fires on server side when the page gets initiated
10        pageInitiatedOnServer: {
11            env: { client: false, server: true },
12        },
13        // Event - fires on client side when the page started
14        pageStartedOnClient: {
15            env: { client: true, server: false },
16        },
17    },
18}
```

# Кастомные хуки

```
4 // pages/+config.ts
5
6 export default {
7     ... config,
8     meta: {
9         // Event - fires on server side when the page gets initiated
10        pageInitiatedOnServer: {
11            env: { client: false, server: true },
12        },
13        // Event - fires on client side when the page started
14        pageStartedOnClient: {
15            env: { client: true, server: false },
16        },
17    },
18}
```

# Кастомные хуки

```
4 // pages/+config.ts
5
6 export default {
7     ... config,
8     meta: {
9         // Event - fires on server side when the page gets initiated
10        pageInitiatedOnServer: {
11            env: { client: false, server: true },
12        },
13        // Event - fires on client side when the page started
14        pageStartedOnClient: {
15            env: { client: true, server: false },
16        },
17    },
18}
```

# Кастомные хуки

```
4 // pages/+config.ts
5
6 export default {
7     ... config,
8     meta: {
9         // Event - fires on server side when the page gets initiated
10        pageInitiatedOnServer: {
11            env: { client: false, server: true },
12        },
13        // Event - fires on client side when the page started
14        pageStartedOnClient: {
15            env: { client: true, server: false },
16        },
17    },
18}
```

# Кастомные хуки

```
4 // pages/+config.ts
5
6 export default {
7     ... config,
8     meta: {
9         // Event - fires on server side when the page gets initiated
10        pageInitiatedOnServer: {
11            env: { client: false, server: true },
12        },
13        // Event - fires on client side when the page started
14        pageStartedOnClient: {
15            env: { client: true, server: false },
16        },
17    },
18}
```

# Кастомные хуки

```
4 // vike.d.ts
5
6 declare global {
7     namespace Vike {
8         interface PageContext {
9             isMobile: boolean;
10            config: {
11                /** Page init event - server side */
12                pageInitiatedOnServer?: <T>(ctx: PageContextServer<T>) => void;
13                /** Page start event - client side */
14                pageInitiatedOnClient?: <T>(ctx: PageContextClient<T>) => void;
15            };
16        }
17    }
18 }
```

# Кастомные хуки

```
4 // vike.d.ts
5
6 declare global {
7     namespace Vike {
8         interface PageContext {
9             isMobile: boolean;
10            config: {
11                /** Page init event - server side */
12                pageInitiatedOnServer?: <T>(ctx: PageContextServer<T>) => void;
13                /** Page start event - client side */
14                pageInitiatedOnClient?: <T>(ctx: PageContextClient<T>) => void;
15            };
16        }
17    }
18 }
```

# Кастомные хуки

```
4 // vike.d.ts
5
6 declare global {
7     namespace Vike {
8         interface PageContext {
9             isMobile: boolean;
10            config: {
11                /** Page init event - server side */
12                pageInitiatedOnServer?: <T>(ctx: PageContextServer<T>) => void;
13                /** Page start event - client side */
14                pageInitiatedOnClient?: <T>(ctx: PageContextClient<T>) => void;
15            };
16        }
17    }
18 }
```

# Кастомные хуки

- +Wrapper.tsx
- +config.ts
- +onBeforeRender.ts
- +onBeforeRenderClient.ts
- +onPageTransitionEnd.ts
- +onPageTransitionStart.ts
- +pageInitiated.ts

# Кастомные хуки

```
export async function onBeforeRender(pageContext: PageContext) {  
  const { pageInitiatedOnServer } = pageContext.config;  
  
  pageInitiatedOnServer?.(pageContext);  
}
```

-  +Wrapper.tsx
-  +config.ts
-  +onBeforeRender.ts
-  +onBeforeRenderClient.ts
-  +onPageTransitionEnd.ts
-  +onPageTransitionStart.ts
-  +pageInitiated.ts

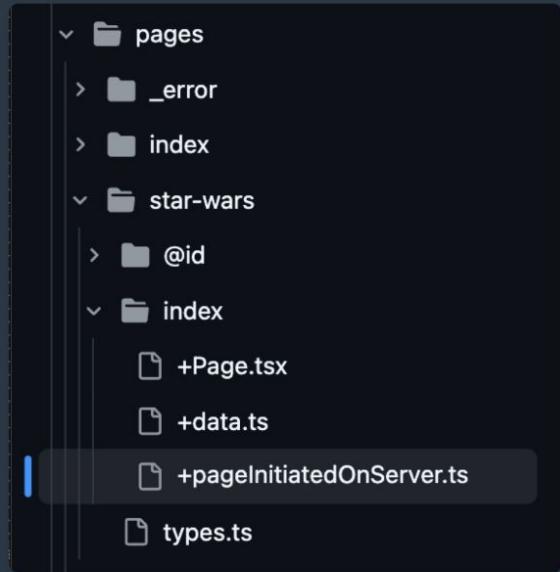
# Кастомные хуки

```
export async function onBeforeRender(pageContext: PageContext) {
  const { pageInitiatedOnServer } = pageContext.config;

  pageInitiatedOnServer?.(pageContext);
}
```

```
1 import { PageContextServer } from "vike/types";
2
3 export const pageInitiatedOnServer =
4   (ctx?: PageContextServer) => {
5     console.log(
6       "Page initiated on server",
7       "isMobile:", ctx?.isMobile
8     );
9   };

```



# Кастомные хуки

```
export async function onBeforeRender(pageContext: PageContext) {
  const { pageInitiatedOnServer } = pageContext.config;

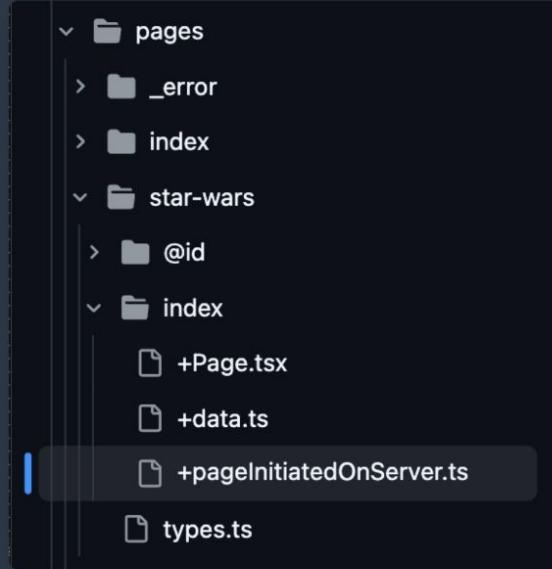
  pageInitiatedOnServer?.(pageContext);
}
```

```
1 import { PageContextServer } from "vike/types";
2
3 export const pageInitiatedOnServer =
4   (ctx?: PageContextServer) => {
5     console.log(
6       "Page initiated on server",
7       "isMobile:", ctx?.isMobile
8     );
9   };

```

```
// server logs:
```

```
7:08:20 PM [vike][request(2)] HTTP request: /star-wars
Page initiated on server, isMobile: false
```



# Запрос данных

- Хук `+data` ;
- Кастомный хук, например `+pageInitiatedOnServer` ;
- Готовые экстеншены типа `vike-react-query` .

# Запрос данных

```
12 // /pages/movies/+data.ts
13
14 // По-умолчанию работает на сервере, но можно сделать изоморфным на уровне +config.ts
15 import fetch from 'node-fetch'
16
17 export async function data(pageContext) {
18   const response = await fetch('https://star-wars.brillout.com/api/films.json')
19   let { movies } = await response.json()
20   /* Or with an ORM:
21   let movies = await Movie.findAll() */
22   /* Or with SQL:
23   let movies = await sql.run('SELECT * FROM movies; */

25   movies = movies.map(({ title, release_date }) => ({ title, release_date }))
26
27   return {
28     movies
29   }
30 }
```

# Запрос данных

```
12 // /pages/movies/+data.ts
13
14 // По-умолчанию работает на сервере, но можно сделать изоморфным на уровне +config.ts
15 import fetch from 'node-fetch'
16
17 export async function data(pageContext) {
18   const response = await fetch('https://star-wars.brillout.com/api/films.json')
19   let { movies } = await response.json()
20   /* Or with an ORM:
21   let movies = await Movie.findAll() */
22   /* Or with SQL:
23   let movies = await sql.run('SELECT * FROM movies; */

25   movies = movies.map(({ title, release_date }) => ({ title, release_date }))
26
27   return {
28     movies
29   }
30 }
```

# Запрос данных

```
12 // /pages/movies/+data.ts
13
14 // По-умолчанию работает на сервере, но можно сделать изоморфным на уровне +config.ts
15 import fetch from 'node-fetch'
16
17 export async function data(pageContext) {
18   const response = await fetch('https://star-wars.brillout.com/api/films.json')
19   let { movies } = await response.json()
20   /* Or with an ORM:
21   let movies = await Movie.findAll() */
22   /* Or with SQL:
23   let movies = await sql.run('SELECT * FROM movies; */

25   movies = movies.map(({ title, release_date }) => ({ title, release_date }))
26
27   return {
28     movies
29   }
30 }
```

# Запрос данных

```
12 // /pages/movies/+data.ts
13
14 // По-умолчанию работает на сервере, но можно сделать изоморфным на уровне +config.ts
15 import fetch from 'node-fetch'
16
17 export async function data(pageContext) {
18   const response = await fetch('https://star-wars.brillout.com/api/films.json')
19   let { movies } = await response.json()
20   /* Or with an ORM:
21   let movies = await Movie.findAll() */
22   /* Or with SQL:
23   let movies = await sql.run('SELECT * FROM movies; */

25   movies = movies.map(({ title, release_date }) => ({ title, release_date }))
26
27   return {
28     movies
29   }
30 }
```

# Запрос данных

```
12 // /pages/movies/+data.ts
13
14 // По-умолчанию работает на сервере, но можно сделать изоморфным на уровне +config.ts
15 import fetch from 'node-fetch'
16
17 export async function data(pageContext) {
18   const response = await fetch('https://star-wars.brillout.com/api/films.json')
19   let { movies } = await response.json()
20   /* Or with an ORM:
21   let movies = await Movie.findAll() */
22   /* Or with SQL:
23   let movies = await sql.run('SELECT * FROM movies; */

25   movies = movies.map(({ title, release_date }) => ({ title, release_date }))
26
27   return {
28     movies
29   }
30 }
```

# Запрос данных

```
12 // /pages/movies/+Page.tsx
13
14 import { useData } from 'vike-react/useData' // or vike-vue
15
16 export default function Page() {
17     const data = useData<T>();
18     const { title, release_date } = data;
19
20     return <h1>{ title }</h1>
21 }
```

```
12 // /pages/movies/+Page.vue
13
14 <script lang="ts" setup>
15     import { useData } from 'vike-vue/useData';
16     const movie = useData<T>();
17 </script>
18
19 <template>
20     <h1>{{ movie.title }}</h1>
21 </template>
```

# Запрос данных

```
12 // /pages/movies/+Page.tsx
13
14 import { useData } from 'vike-react/useData' // or vike-vue
15
16 export default function Page() {
17     const data = useData<T>();
18     const { title, release_date } = data;
19
20     return <h1>{ title }</h1>
21 }
```

```
12 // /pages/movies/+Page.vue
13
14 <script lang="ts" setup>
15     import { useData } from 'vike-vue/useData';
16     const movie = useData<T>();
17 </script>
18
19 <template>
20     <h1>{{ movie.title }}</h1>
21 </template>
```

# Запрос данных

```
12 // /pages/movies/+Page.tsx
13
14 import { useData } from 'vike-react/useData' // or vike-vue
15
16 export default function Page() {
17     const data = useData<T>();
18     const { title, release_date } = data;
19
20     return <h1>{ title }</h1>
21 }
```

```
12 // /pages/movies/+Page.vue
13
14 <script lang="ts" setup>
15     import { useData } from 'vike-vue/useData';
16     const movie = useData<T>();
17 </script>
18
19 <template>
20     <h1>{{ movie.title }}</h1>
21 </template>
```

# Запрос данных

```
12 // /pages/movies/+Page.tsx
13
14 import { useData } from 'vike-react/useData' // or vike-vue
15
16 export default function Page() {
17     const data = useData<T>();
18     const { title, release_date } = data;
19
20     return <h1>{ title }</h1>
21 }
```

```
12 // /pages/movies/+Page.vue
13
14 <script lang="ts" setup>
15     import { useData } from 'vike-vue/useData';
16     const movie = useData<T>();
17 </script>
18
19 <template>
20     <h1>{{ movie.title }}</h1>
21 </template>
```

# Запрос данных

```
12 // /pages/movies/+Page.tsx
13
14 import { useData } from 'vike-react/useData' // or vike-vue
15
16 export default function Page() {
17     const data = useData<T>();
18     const { title, release_date } = data;
19
20     return <h1>{ title }</h1>
21 }
```

```
12 // /pages/movies/+Page.vue
13
14 <script lang="ts" setup>
15     import { useData } from 'vike-vue/useData';
16     const movie = useData<T>();
17 </script>
18
19 <template>
20     <h1>{{ movie.title }}</h1>
21 </template>
```

# Запрос данных

```
12 // /pages/movies/+Page.tsx
13
14 import { useData } from 'vike-react/useData' // or vike-vue
15
16 export default function Page() {
17     const data = useData<T>();
18     const { title, release_date } = data;
19
20     return <h1>{ title }</h1>
21 }
```

```
12 // /pages/movies/+Page.vue
13
14 <script lang="ts" setup>
15     import { useData } from 'vike-vue/useData';
16     const movie = useData<T>();
17 </script>
18
19 <template>
20     <h1>{{ movie.title }}</h1>
21 </template>
```

# Запрос данных

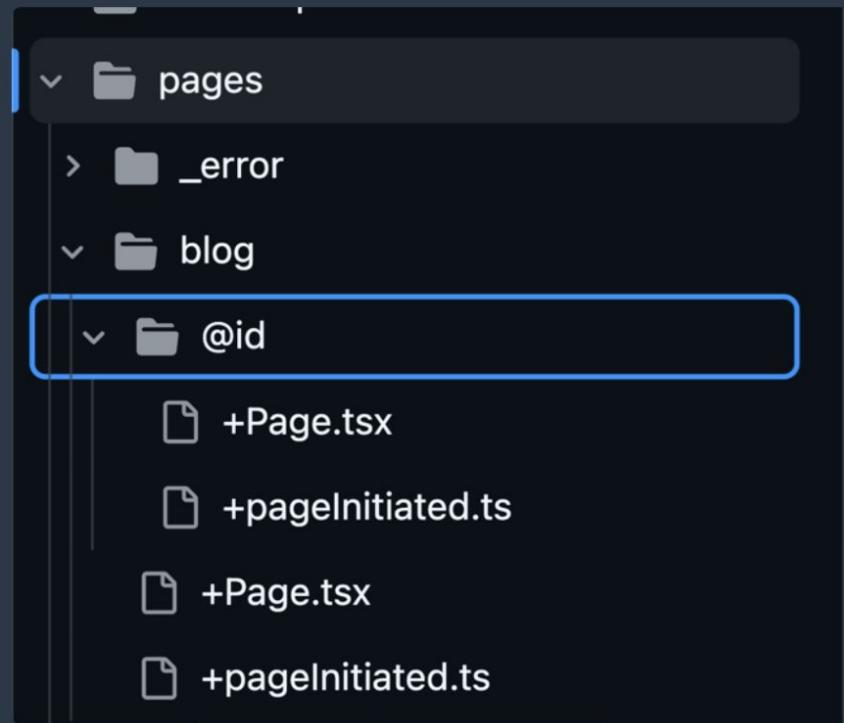
```
12 // /pages/movies/+Page.tsx
13
14 import { useData } from 'vike-react/useData' // or vike-vue
15
16 export default function Page() {
17     const data = useData<T>();
18     const { title, release_date } = data;
19
20     return <h1>{ title }</h1>
21 }
```

```
12 // /pages/movies/+Page.vue
13
14 <script lang="ts" setup>
15     import { useData } from 'vike-vue/useData';
16     const movie = useData<T>();
17 </script>
18
19 <template>
20     <h1>{{ movie.title }}</h1>
21 </template>
```

# Навигация

# Навигация

- По-умолчанию File-Based Routing ;



# Навигация

- По-умолчанию File-Based Routing ;
- Можно переопределять route string  
или route matching function ;

The screenshot shows a code editor with a sidebar on the left displaying a file tree:

- pages
- \_error
- index
- product/edit
  - +Page.tsx
  - +route.ts
- star-wars
- todo
  - +Wrapper.tsx
  - +config.ts
  - +onBeforeRender.ts

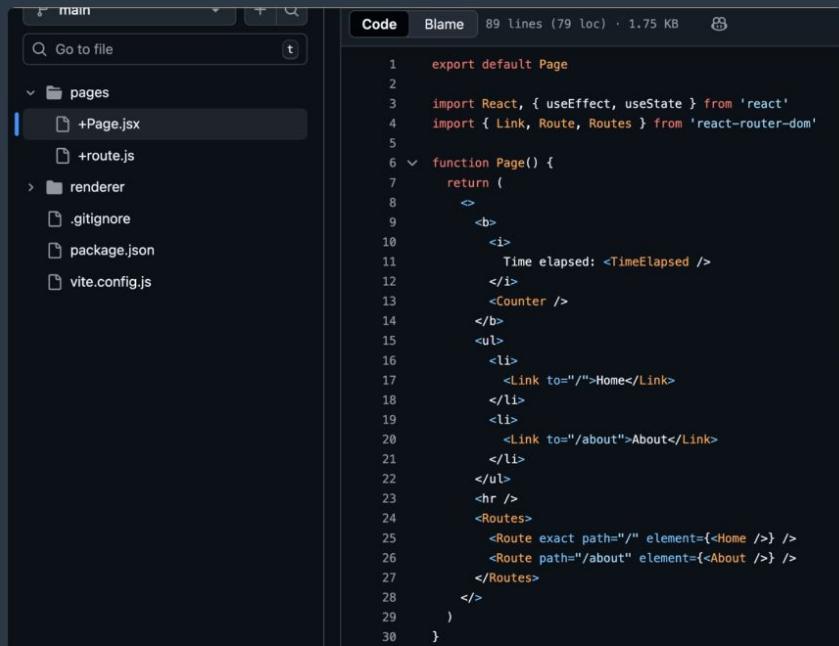
The file `+route.ts` is currently selected in the sidebar.

The main pane displays the following TypeScript code:

```
1 import { PageContext } from "vike/types";
2
3 const routeRegex = /product\/(0-9+)\/edit/;
4
5 export default function route(pageContext: PageContext) {
6     if (!pageContext?.user?.isAdmin) return false;
7
8     const match = pageContext.url.pathname.match(routeRegex);
9
10    if (!match) return false;
11
12    const [, id] = match;
13
14    return {
15        routeParams: { id },
16    };
17}
```

# Навигация

- По-умолчанию `File-Based Routing` ;
- Можно переопределять `route string` или `route matching function` ;
- Можно сделать полностью `client-side routing` с интеграцией `React Router/Vue Router` итд;



The screenshot shows a code editor interface with a dark theme. On the left is a file tree with the following structure:

- main
- pages
- +Page.jsx
- +route.js
- renderer
- .gitignore
- package.json
- vite.config.js

The file `+Page.jsx` is selected. On the right is the code editor pane, which displays the following component definition:

```
1  export default Page
2
3  import React, { useEffect, useState } from 'react'
4  import { Link, Route, Routes } from 'react-router-dom'
5
6  function Page() {
7    return (
8      <>
9        <b>
10          Time elapsed: <TimeElapsed />
11        </b>
12        <i>
13          <Counter />
14        </i>
15        <ul>
16          <li>
17            <Link to="/">Home</Link>
18          </li>
19          <li>
20            <Link to="/about">About</Link>
21          </li>
22        </ul>
23        <hr />
24        <Routes>
25          <Route exact path="/" element={<Home />} />
26          <Route path="/about" element={<About />} />
27        </Routes>
28      </>
29    )
30  }
```

# Навигация

- По-умолчанию `File-Based Routing` ;
- Можно переопределять `route string`  
или `route matching function` ;
- Можно сделать полностью `client-side routing`  
с интеграцией `React Router/Vue Router` итд;

# Навигация

- По-умолчанию `File-Based Routing` ;
- Можно переопределять `route string`  
или `route matching function` ;
- Можно сделать полностью `client-side routing`  
с интеграцией `React Router/Vue Router` итд;

# Vike vs Next.js

Vike

Next.js

Bundler

Vite

Web(Turbo)Pack

# Vike vs Next.js

|         | Vike | Next.js        |
|---------|------|----------------|
| Bundler | Vite | Web(Turbo)Pack |
| UI      | ANY  | React          |

# Vike vs Next.js

|              | Vike | Next.js        |
|--------------|------|----------------|
| Bundler      | Vite | Web(Turbo)Pack |
| UI           | ANY  | React          |
| Optional SSR | Yes  | Limited        |

# Vike vs Next.js

|              | Vike | Next.js        |
|--------------|------|----------------|
| Bundler      | Vite | Web(Turbo)Pack |
| UI           | ANY  | React          |
| Optional SSR | Yes  | Limited        |
| HTTP Server  | No*  | Baked-in*      |

# Vike vs Next.js

|                          | Vike | Next.js        |
|--------------------------|------|----------------|
| Bundler                  | Vite | Web(Turbo)Pack |
| UI                       | ANY  | React          |
| Optional SSR             | Yes  | Limited        |
| HTTP Server              | No*  | Baked-in*      |
| Build your own framework | YES  | NO             |

# Relations

# Relations

- Нет спонсоров (Vercel, Meta etc);

# Relations

- Нет спонсоров (Vercel, Meta etc);
- Нет телеметрии и не будет;

# Relations

- Нет спонсоров (Vercel, Meta etc);
- Нет телеметрии и не будет;
- Пока полностью свободно;

# Relations

- Нет спонсоров (Vercel, Meta etc);
- Нет телеметрии и не будет;
- Пока полностью свободно;
- Есть планы на Open Source Pricing для компаний.

# Когда использовать?

# Когда использовать?

- Надоел Next.js ;

# Когда использовать?

- Надоел Next.js ;
- Нужен действительно полный контроль;

# Когда использовать?

- Надоел Next.js ;
- Нужен действительно полный контроль;
- Нужен единый механизм SSR;

# Когда использовать?

- Надоел Next.js ;
- Нужен действительно полный контроль;
- Нужен единый механизм SSR;
- Хочется попробовать сделать мир лучше.



## Contacts

- [@olovyannikov\\_frontend](mailto:@olovyannikov_frontend)
- [@moscowjschat](mailto:@moscowjschat)
- [@Olovyannikov](https://github.com/Olovyannikov)