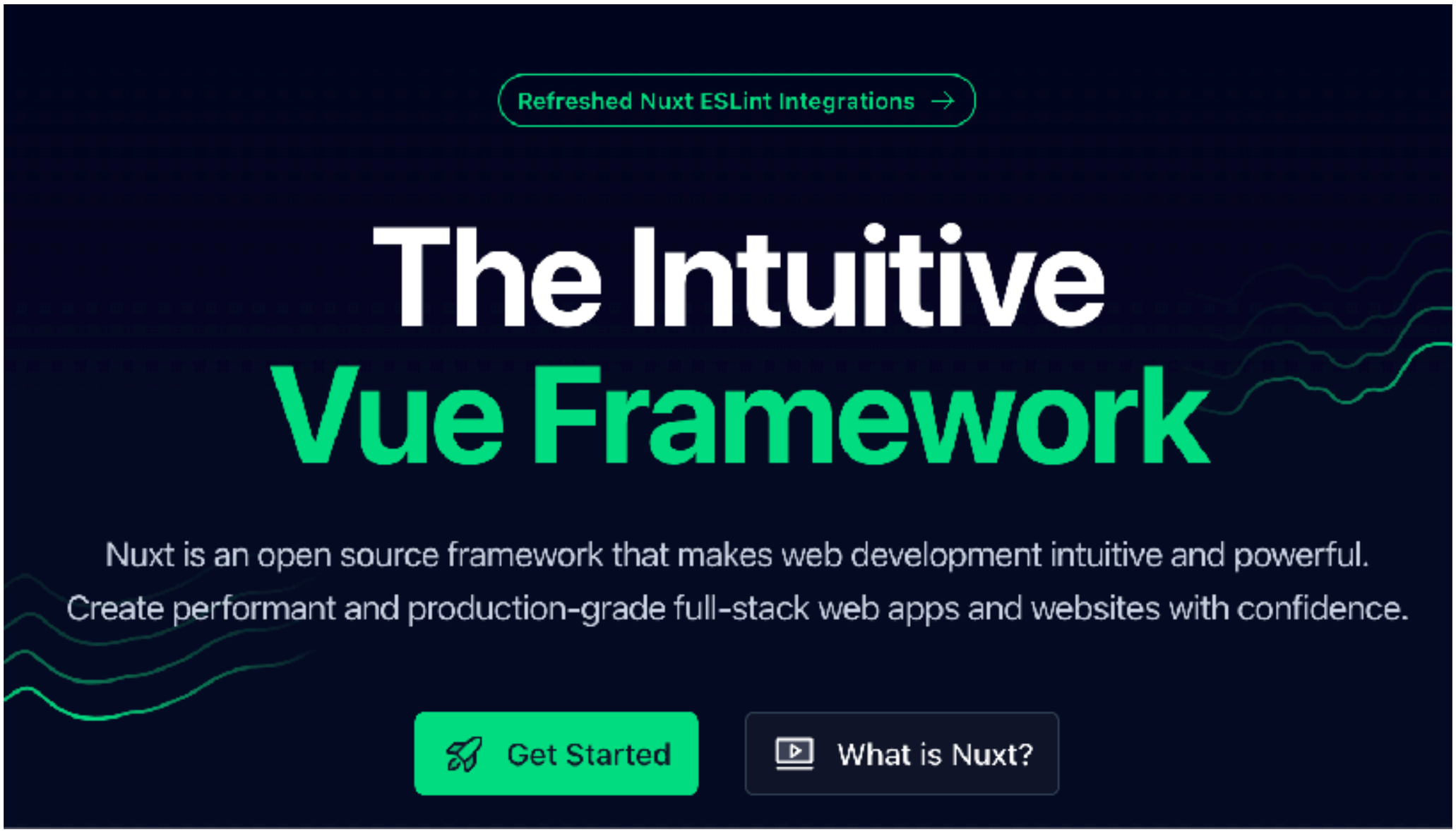
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Frontend Development

https://github.com/up1/course-full-stack-development

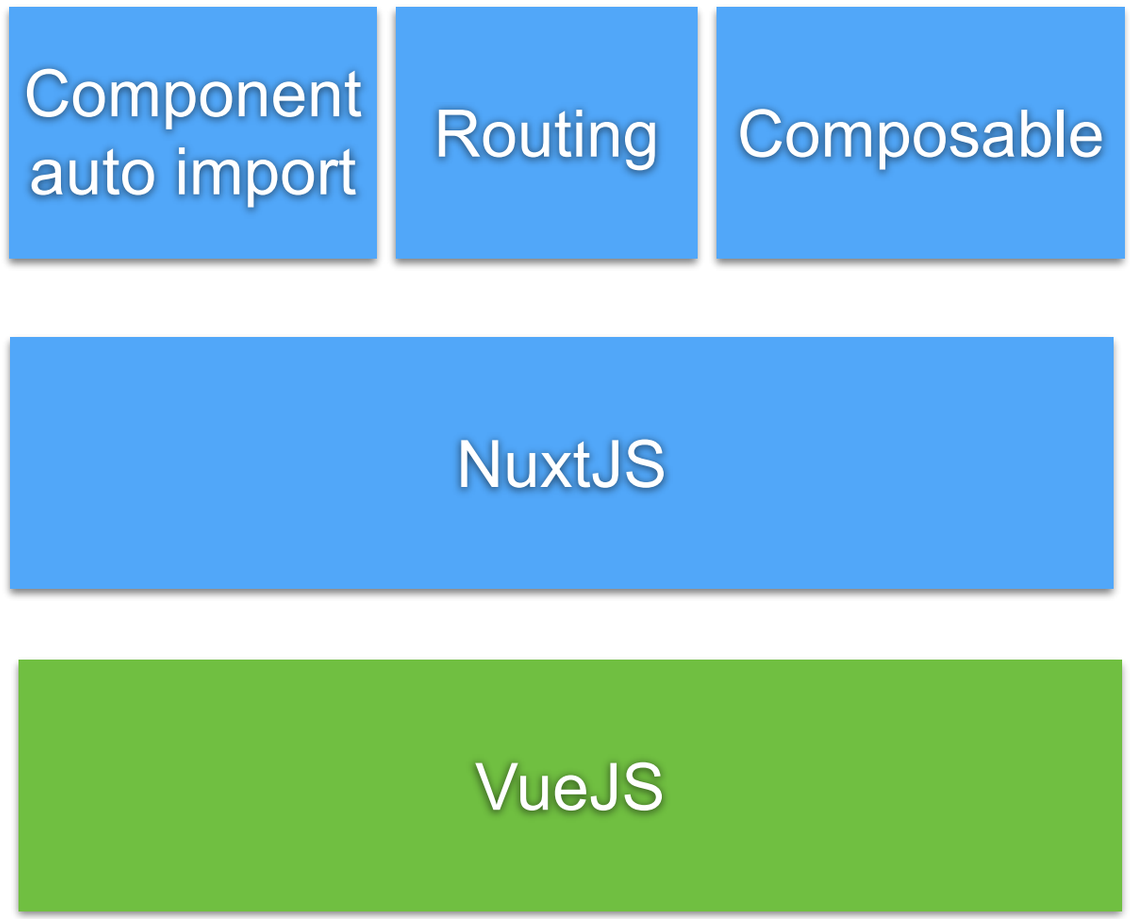
Frontend Development

NuxtJS



https://nuxt.com/

Learning NuxtJS



NuxtJS features

Code splitting (automatic)
Pre-rendering
Server-Side Rendering (SSR)
Static Side Generator (SSG)
SEO and Meta tags
File-based routing
Data fetching
Zero-config TypeScript
State management

https://nuxt.com/docs/getting-started/introduction

Create NuxtJS Project

npx nuxi@latest init <project-name>

npm run dev

```
{
  "dependencies": {
    "nuxt": "^3.11.2",
    "vue": "^3.4.21",
    "vue-router": "^4.3.0"
  }
}
```

https://nuxt.com/

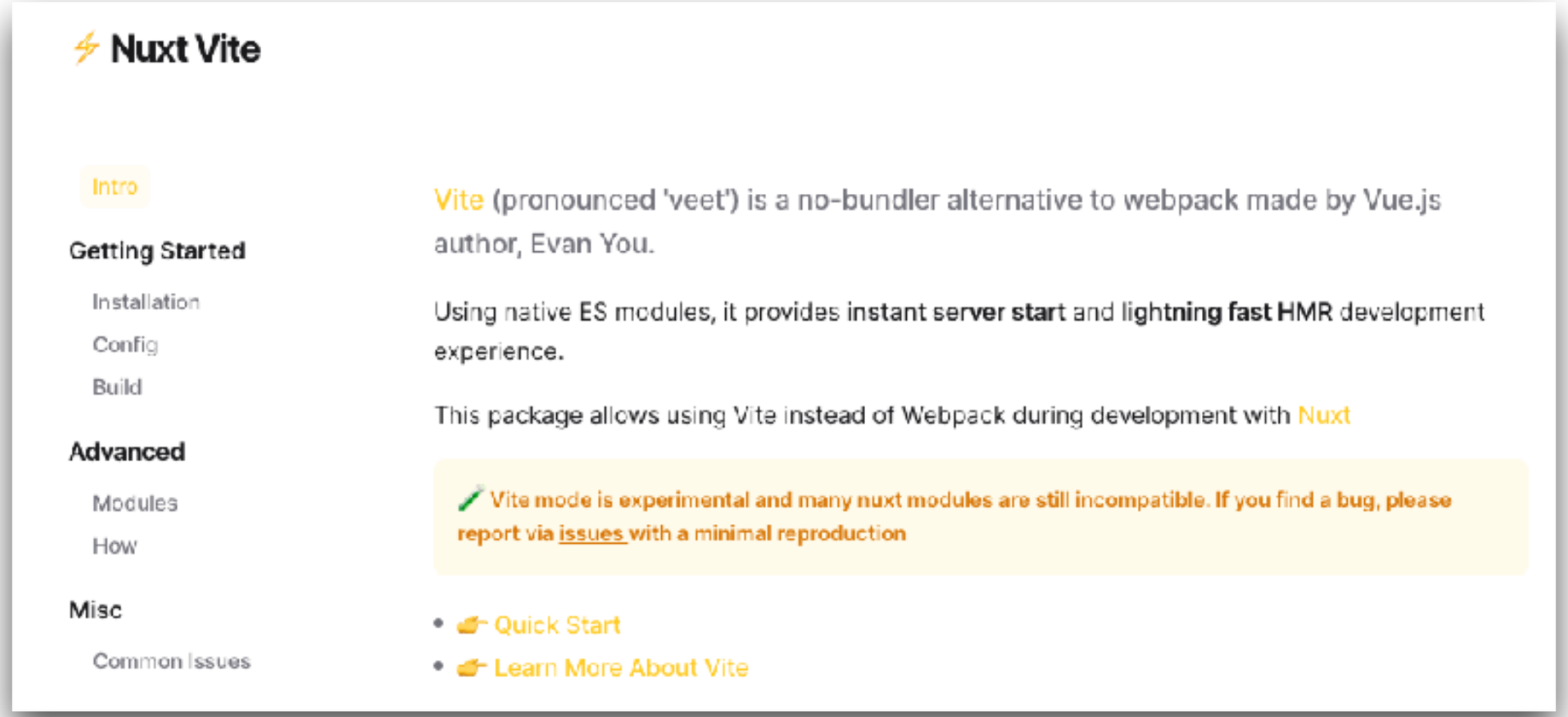
Run in dev mode

npm run dev

```
> dev
> nuxt dev

Nuxt 3.11.2 with Nitro 2.9.6

  ➜ Local:    http://localhost:3000/
  ➜ Network:  use --host to expose

  ➜ DevTools: press Shift + Option + D in the browser (v1.2.0)

ℹ Vite server warmed up in 537ms
✔ Nuxt Nitro server built in 507 ms
ℹ Vite client warmed up in 1036ms
```

Vite server

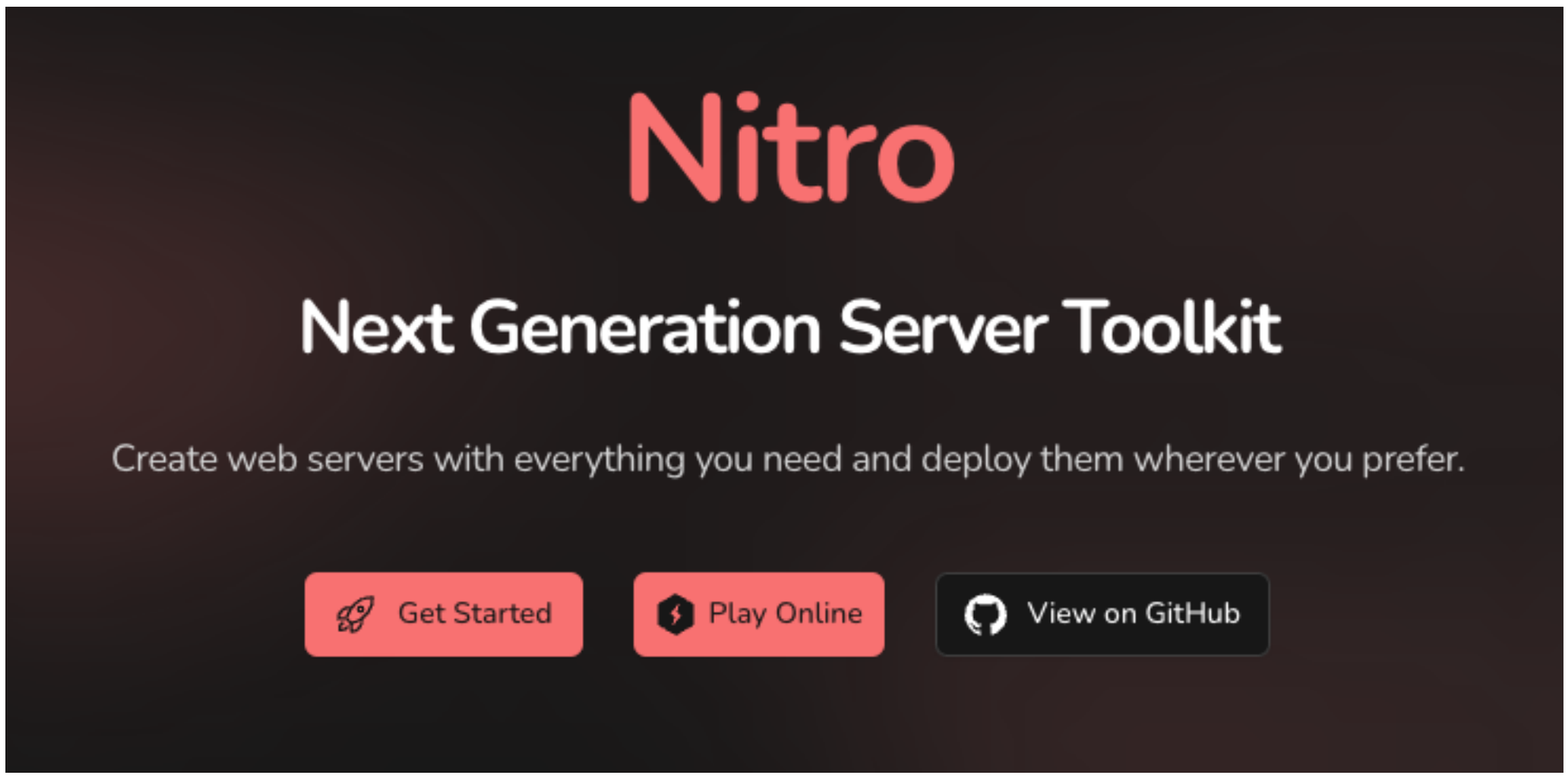
Dev server for HMR (Hot Module Replacement)



https://vite.nuxtjs.org/

Nuxt Nitro Server

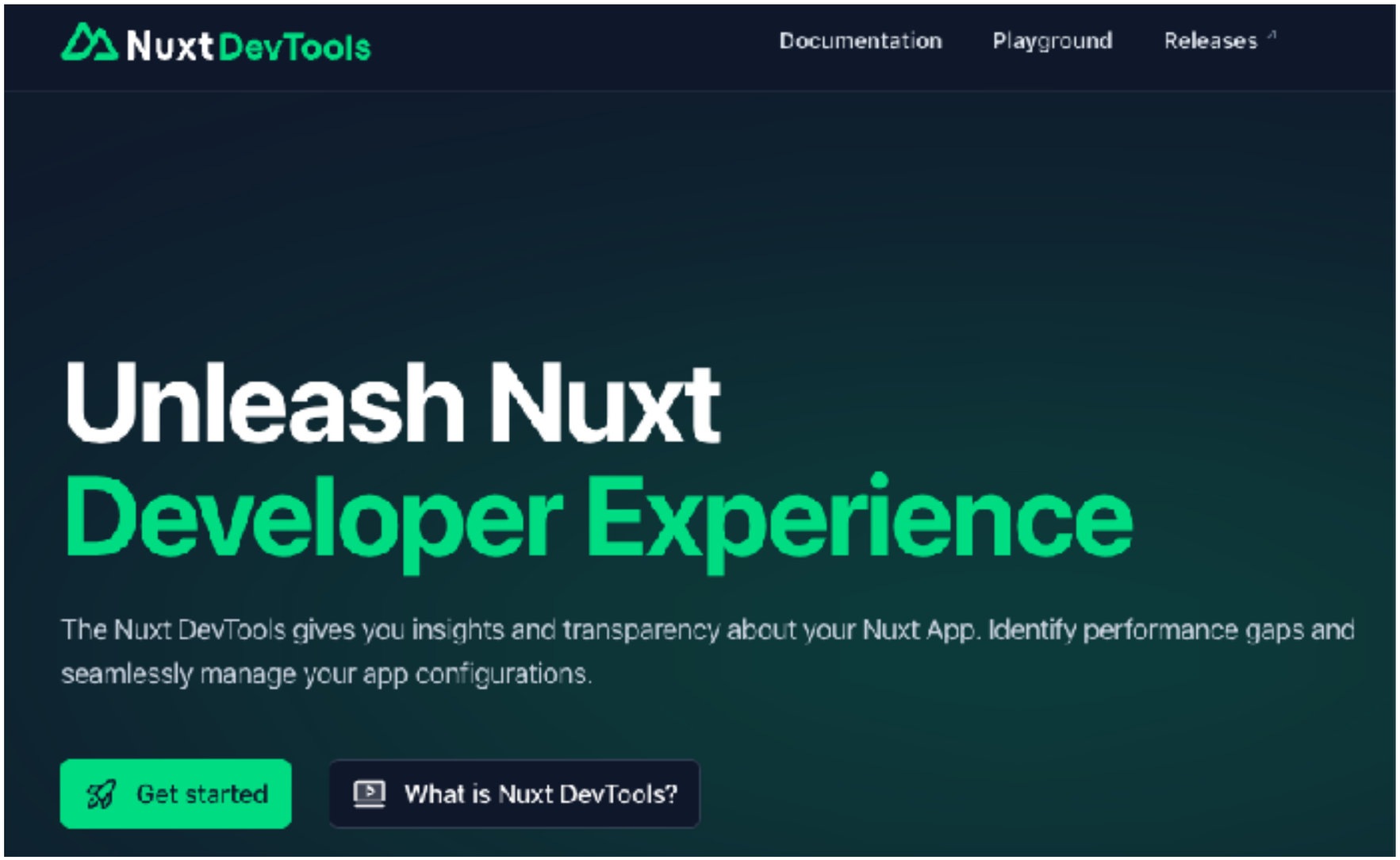
A server for the modern web



https://nitro.unjs.io/

Nuxt DevTools

Insights and transparency on your app



https://devtools.nuxt.com/

Nuxt DevTools

Insights and transparency on your app

next.config.ts

```ts
export default defineNuxtConfig({
  devtools: { enabled: true }
})
```

https://devtools.nuxt.com/

Show warning and errors !!

```
WARN  [Vue Router warn]: No match found for location with path "/user/login"

WARN  [Vue Router warn]: No match found for location with path "/user/register"

WARN  [Vue Router warn]: No match found for location with path "/user/login"

WARN  [Vue Router warn]: No match found for location with path "/user/register"
```

Nuxt DevTools

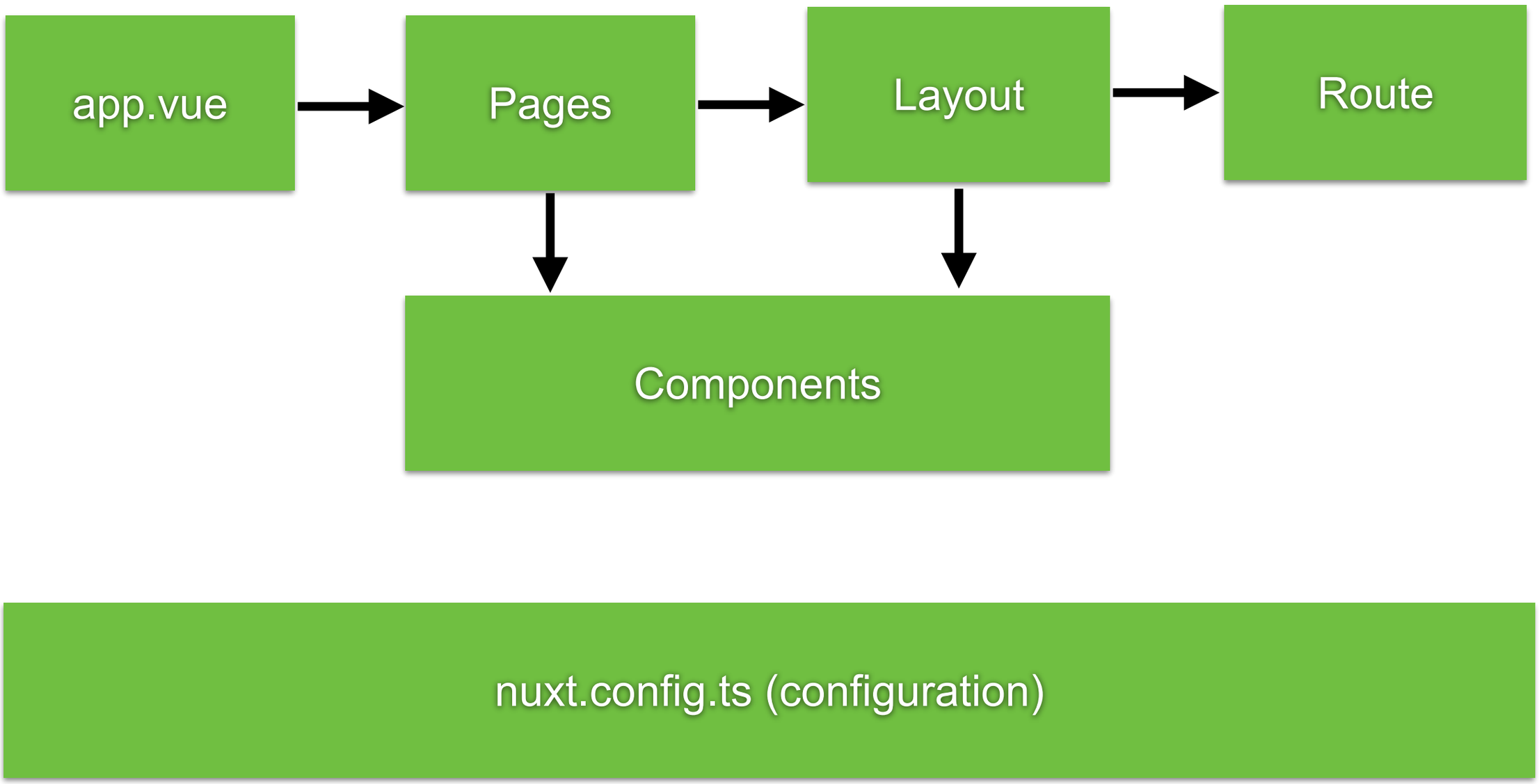
https://devtools.nuxt.com/

Nuxt DevTools

List of pages and routes
List of components
Imported libraries
Modules and plugins
Assets
Missing tags (SEO)

Workflow of application

NuxtJS Workflow



App.vue

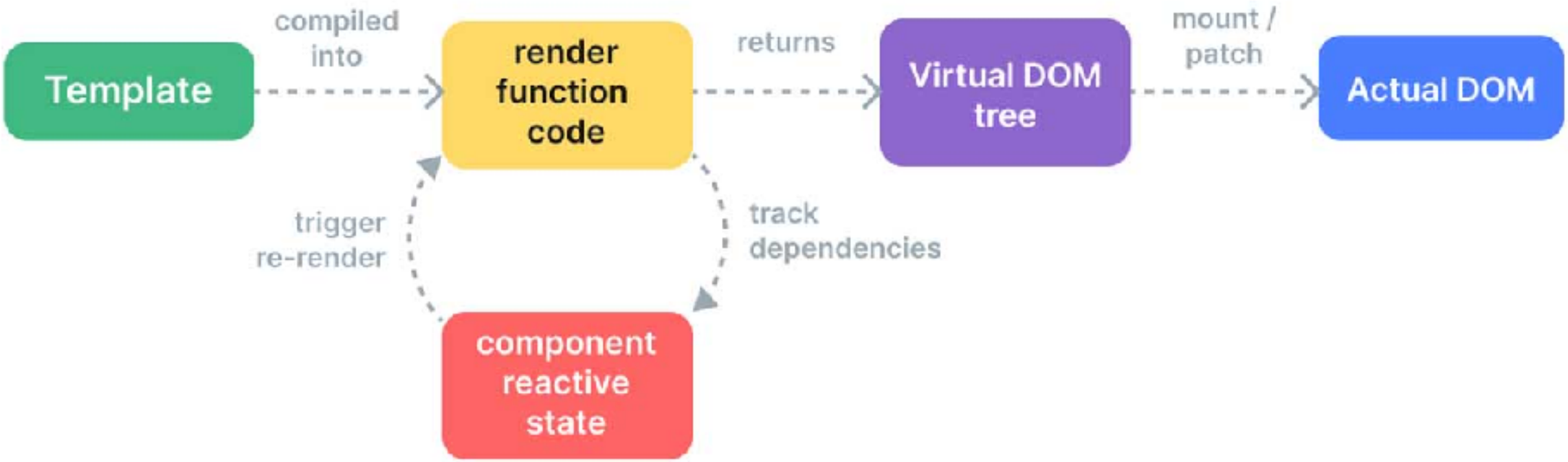
Entry point of application

```
<template>
  <div>
    <NuxtWelcome />
  </div>
</template>
```

Components in view

https://nuxt.com/docs/api/components/nuxt-welcome

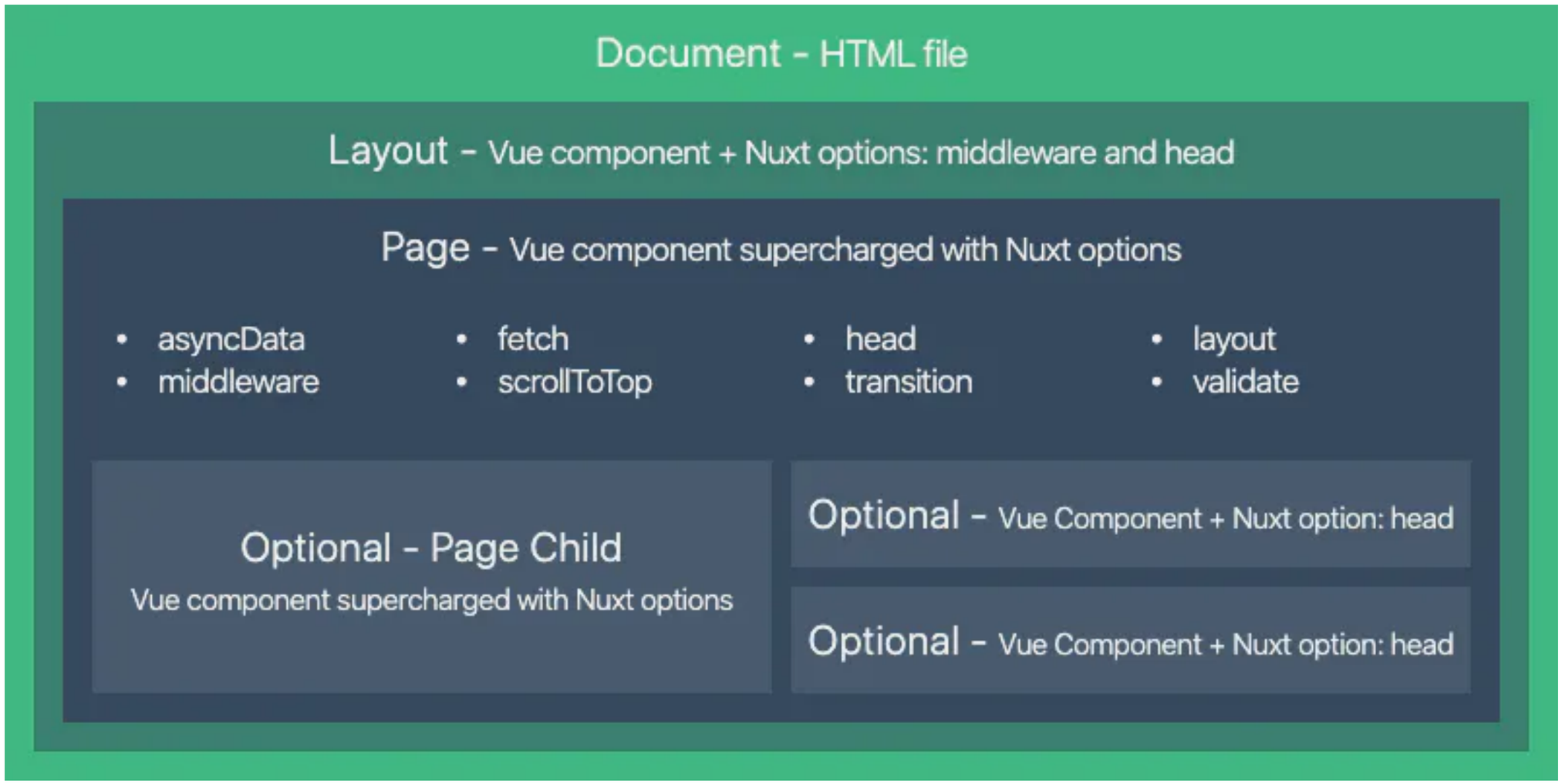
Render System



https://www.vuemastery.com/blog/the-future-of-vue-vapor-mode/

Views

Layout
Page/Template



https://nuxt.com/docs/getting-started/views

Learning path

Component	Routing and pages	VueX states	Layout
Data fetching	Plugins	Mutations with VueX	Testing

Components

Types of Component

Client component
Server component
Standalone Server component (Island)

https://nuxt.com/docs/guide/directory-structure/components

Components

Reusable pieces of the user interface
Create in folder **/components**
Auto-import components

app.vue

```
<template>
  <div>
    <h1>Welcome to the homepage</h1>
    <Hello>
      This is an auto-imported component.
    </Hello>
  </div>
</template>
```

components/Hello.vue

```
<template>
  <span>
    <slot />
  </span>
</template>
```

Single File Component (SFC)

Template/Logic/Style

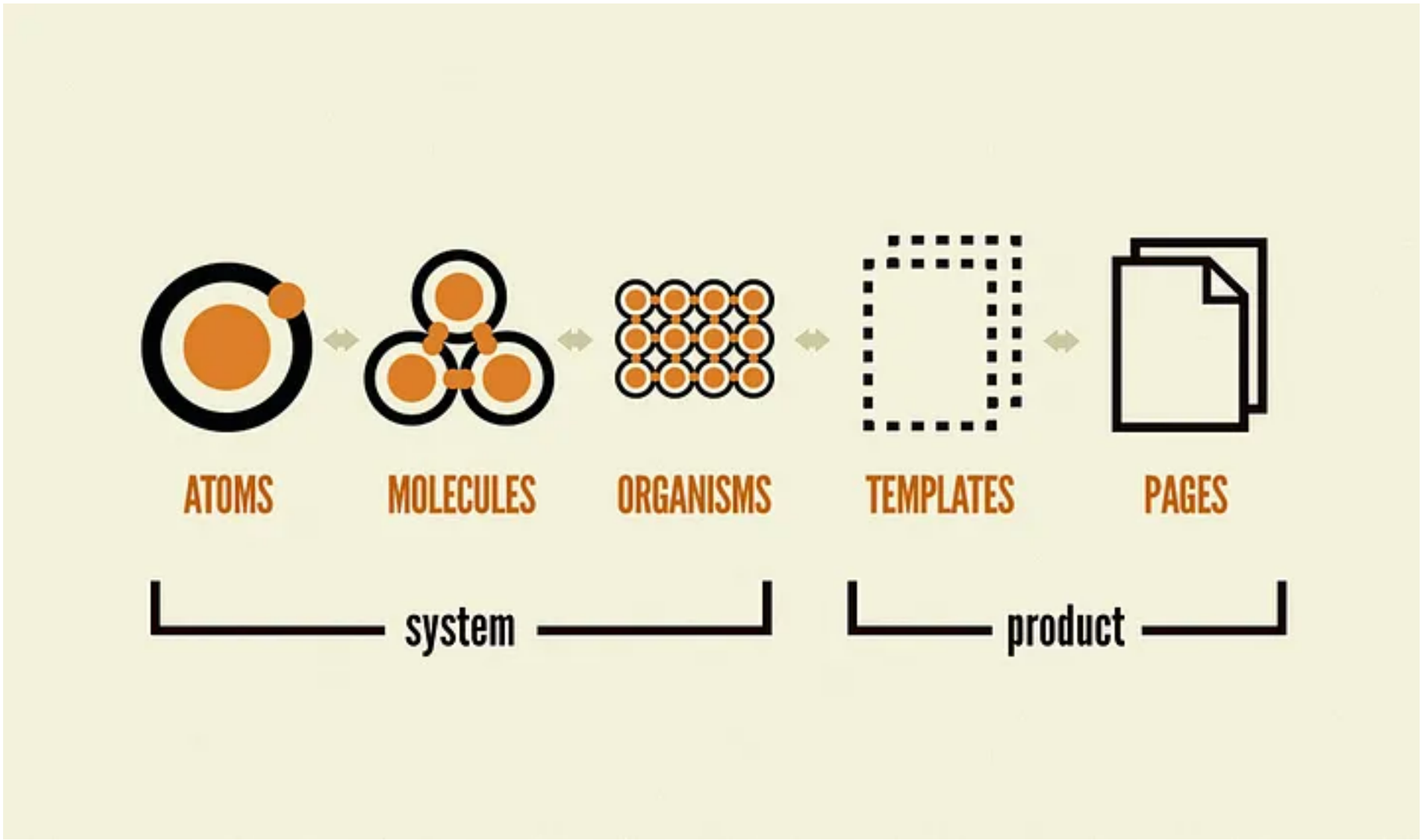
```vue
<script setup>
import { ref } from 'vue'
const greeting = ref('Hello World!')
</script>

<template>
  <p class="greeting">{{ greeting }}</p>
</template>

<style>
.greeting {
  color: red;
  font-weight: bold;
}
</style>
```

https://vuejs.org/guide/scaling-up/sfc.html

Component-Driven

https://www.componentdriven.org/

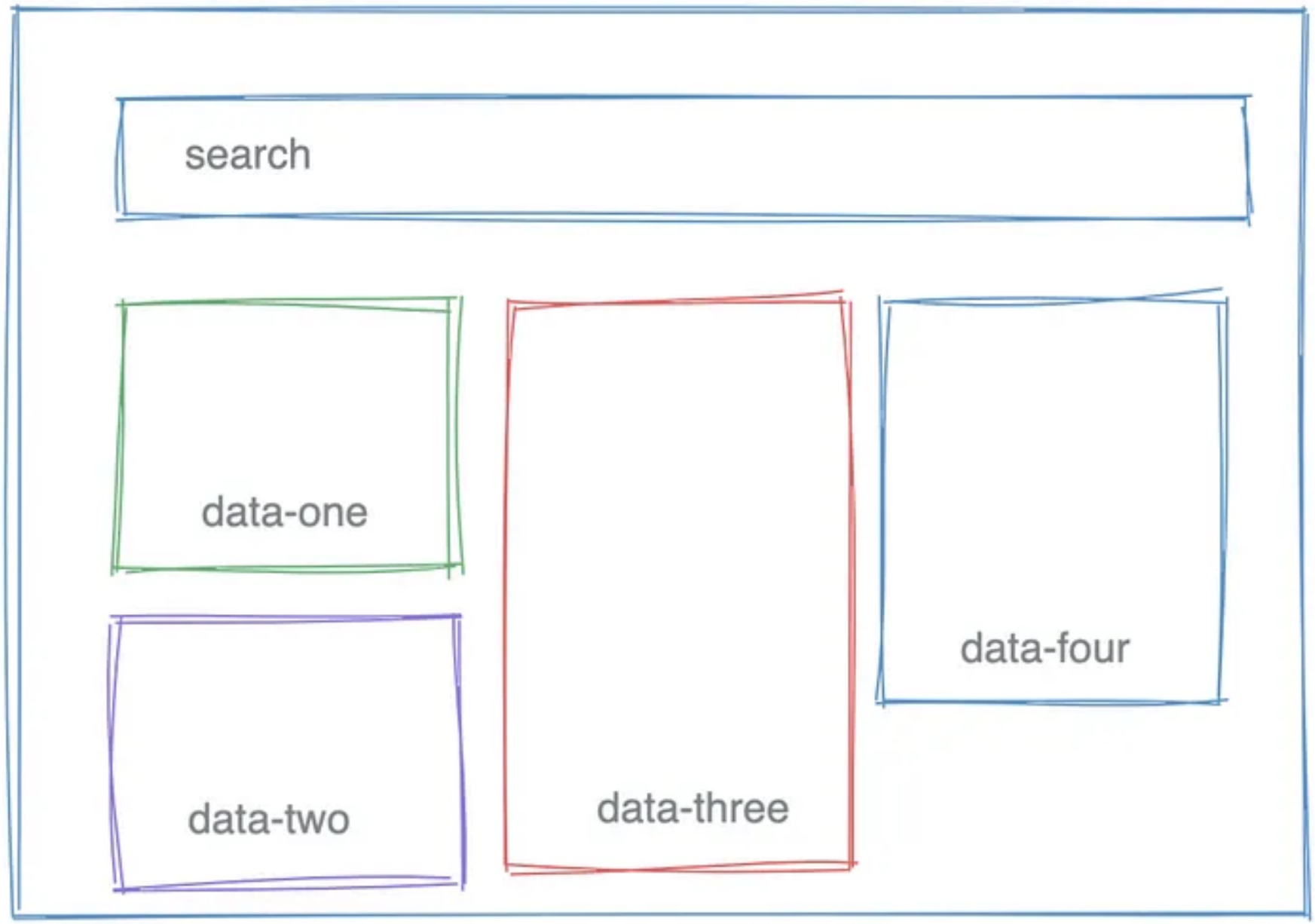
Atomic Design



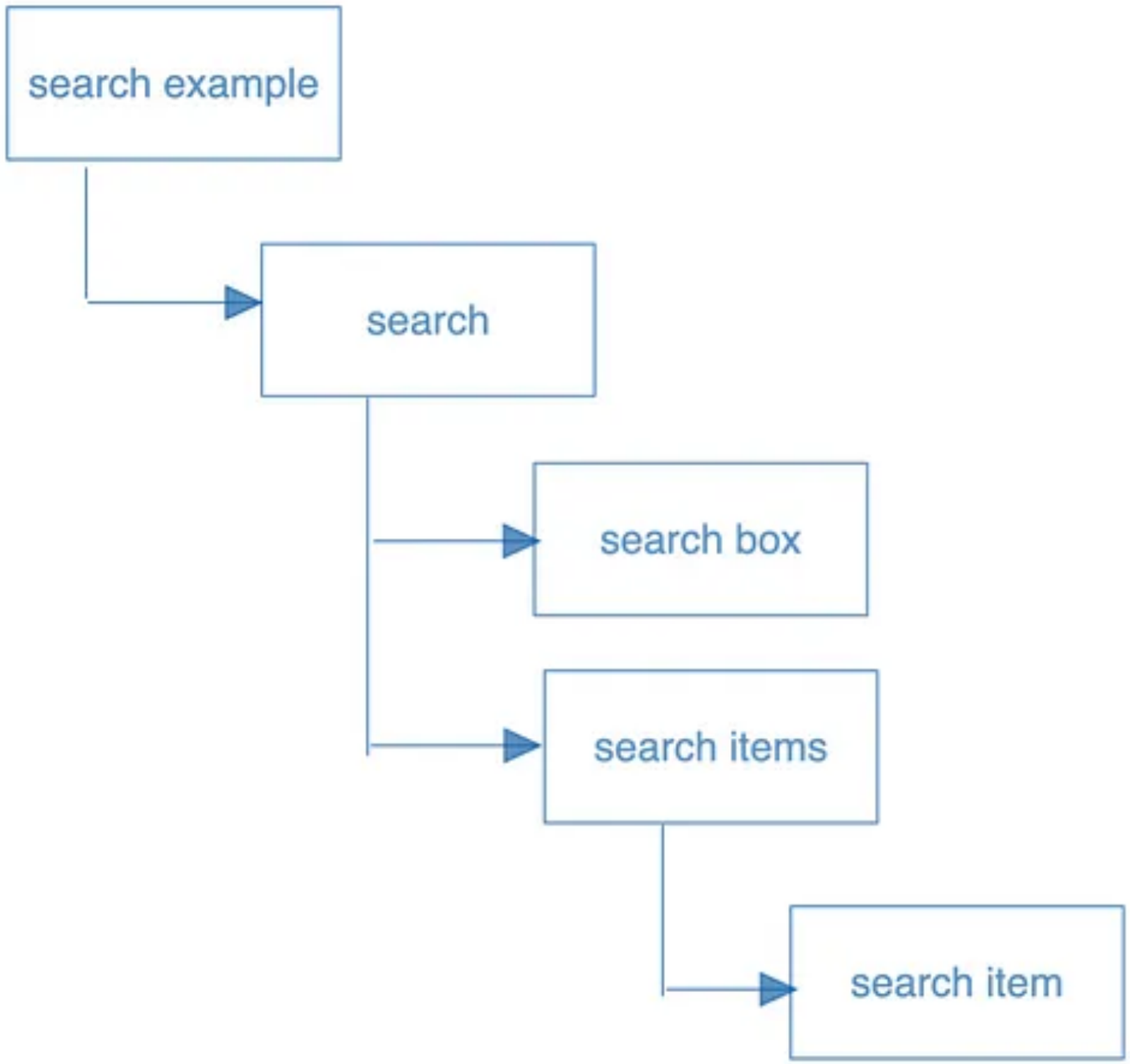
https://atomicdesign.bradfrost.com/

Example

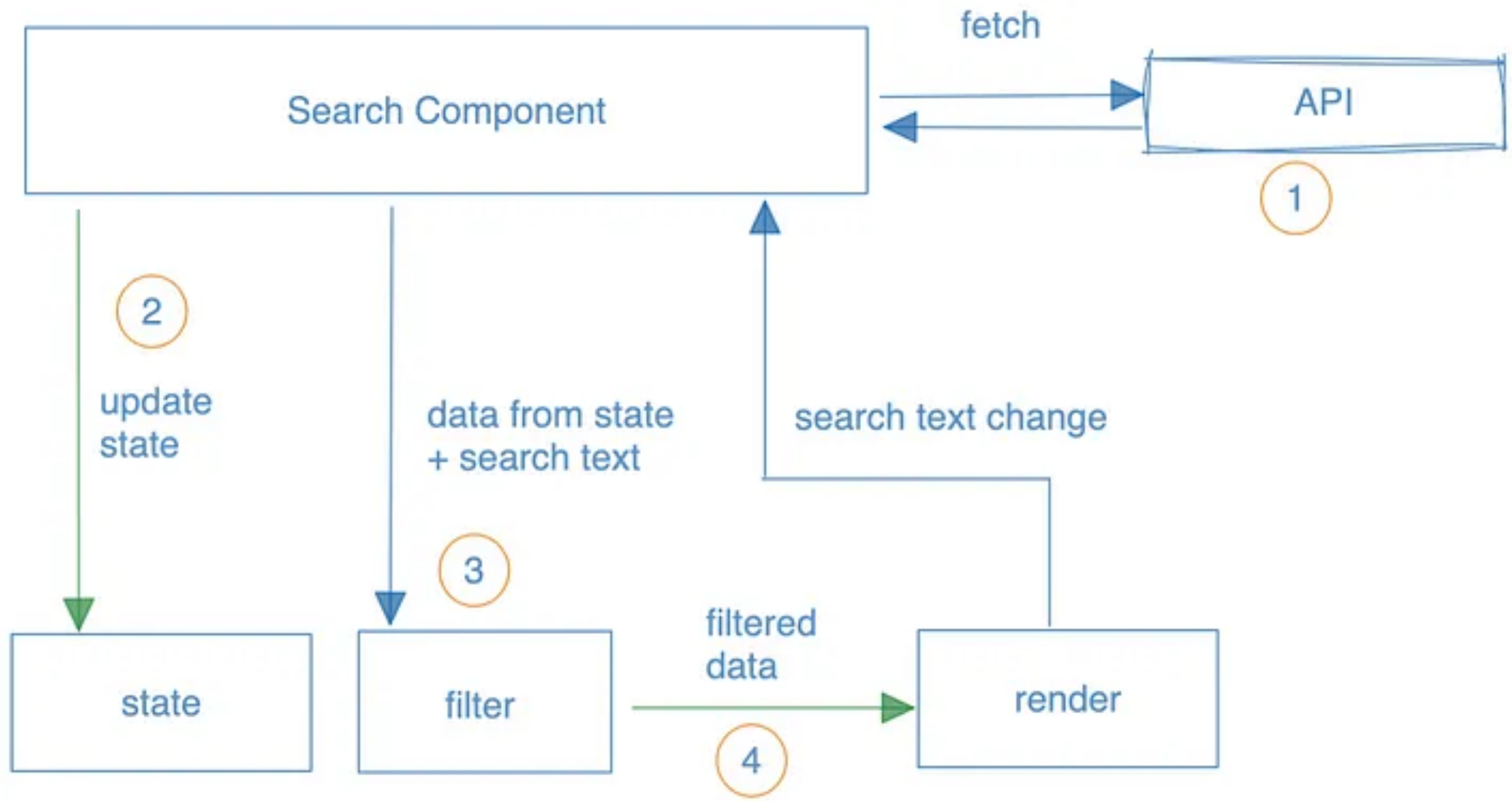
1. Component Design



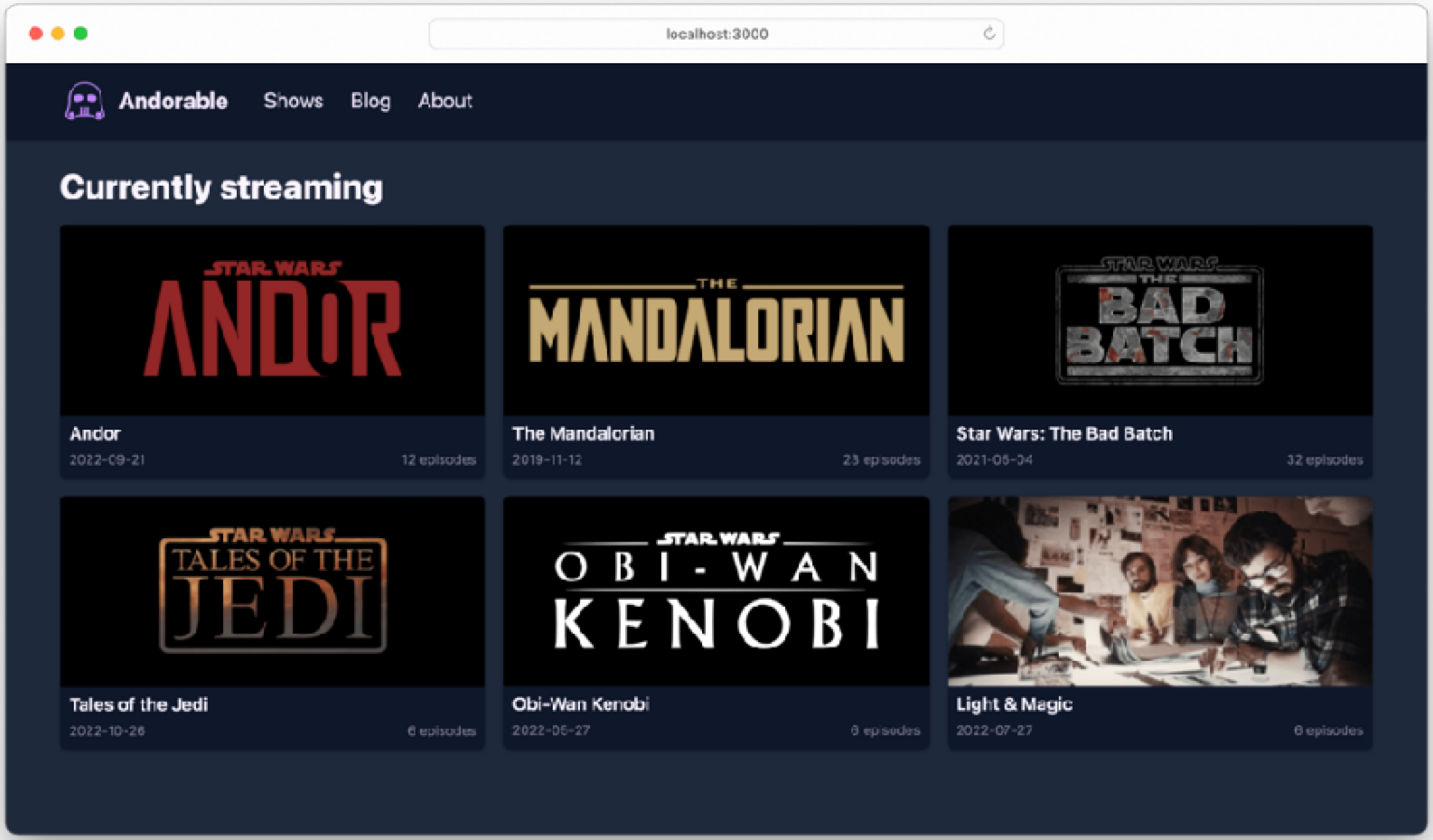
2. Component Diagram



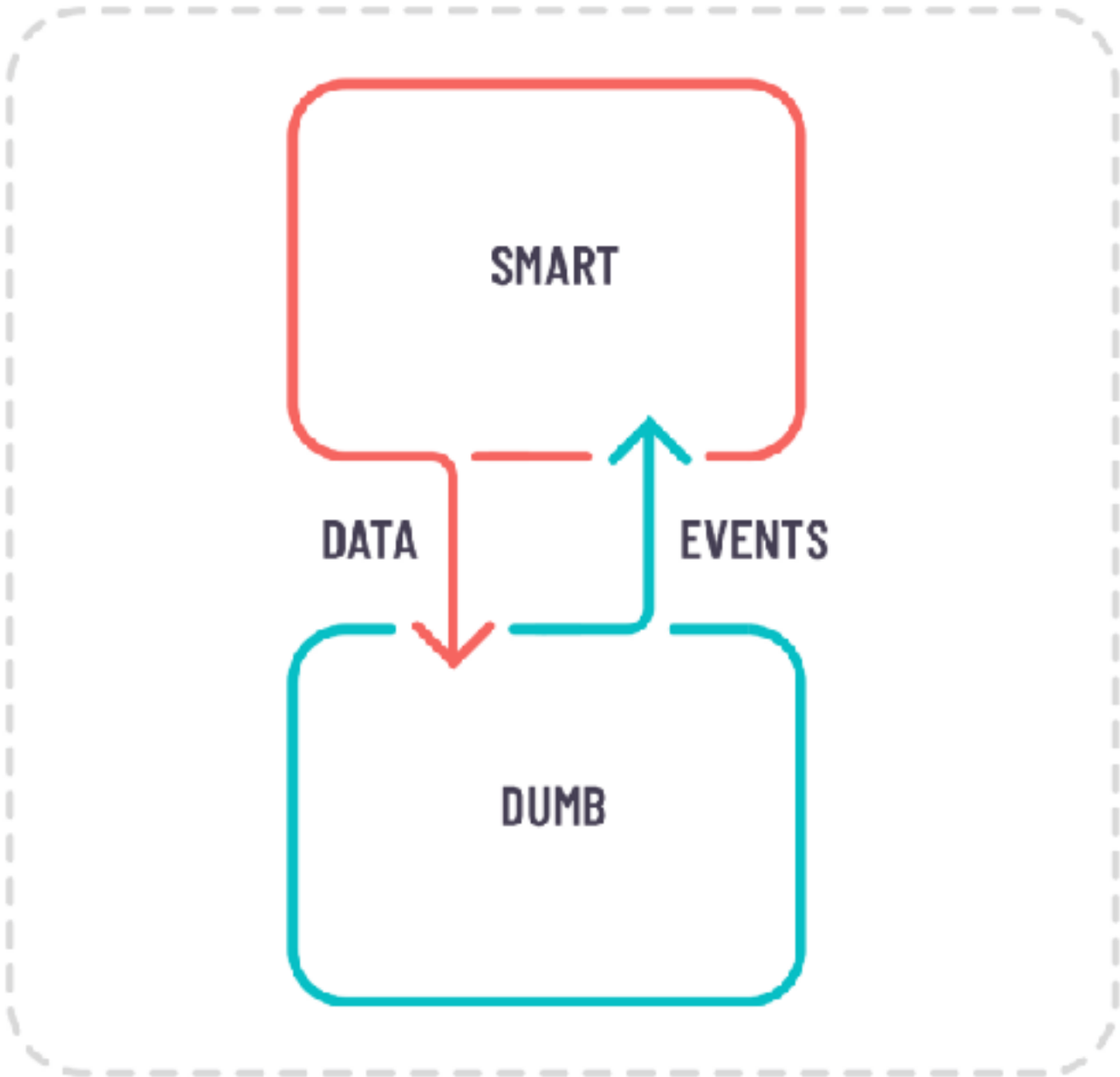
3. Code Design



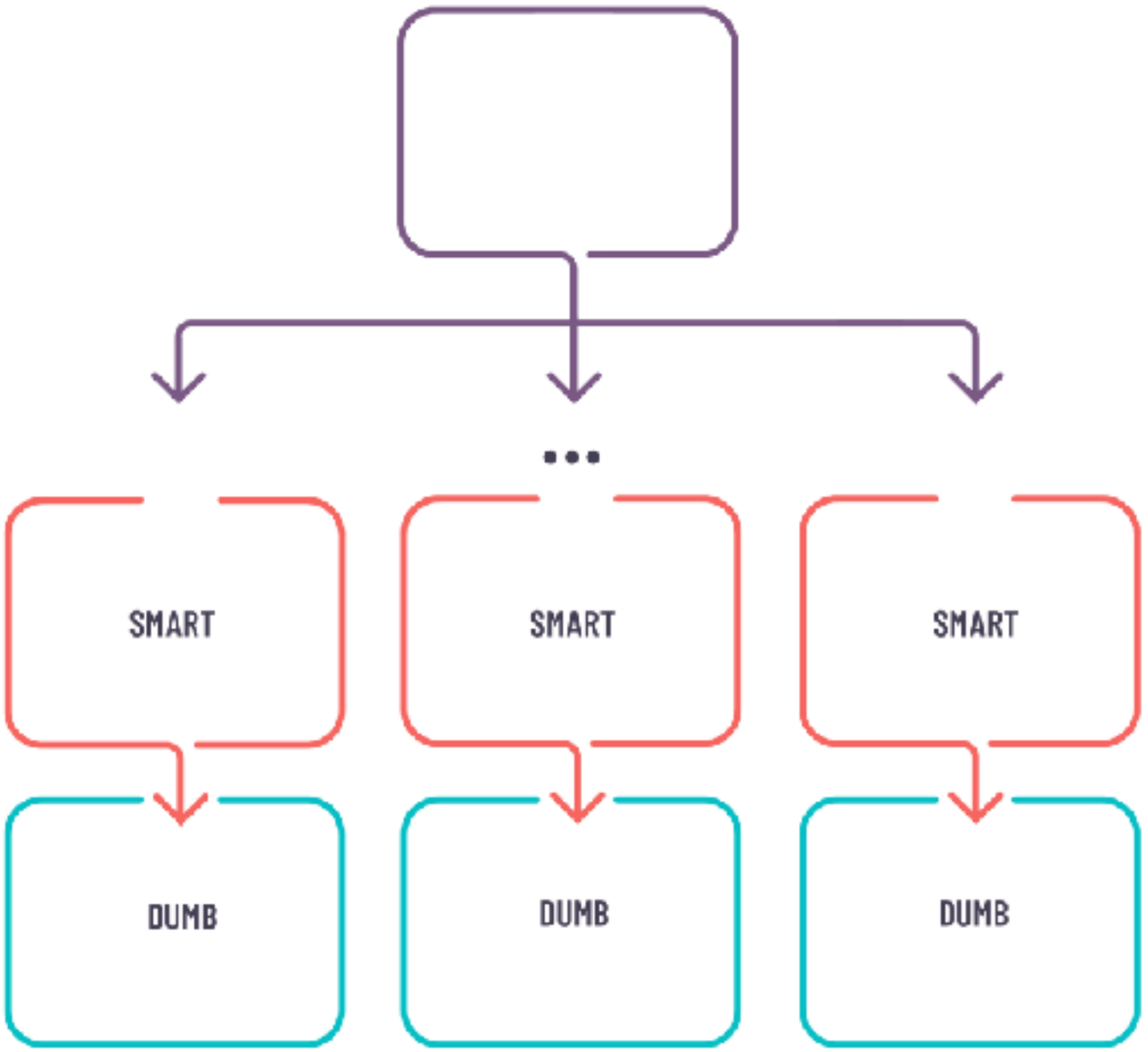
Design your component ?



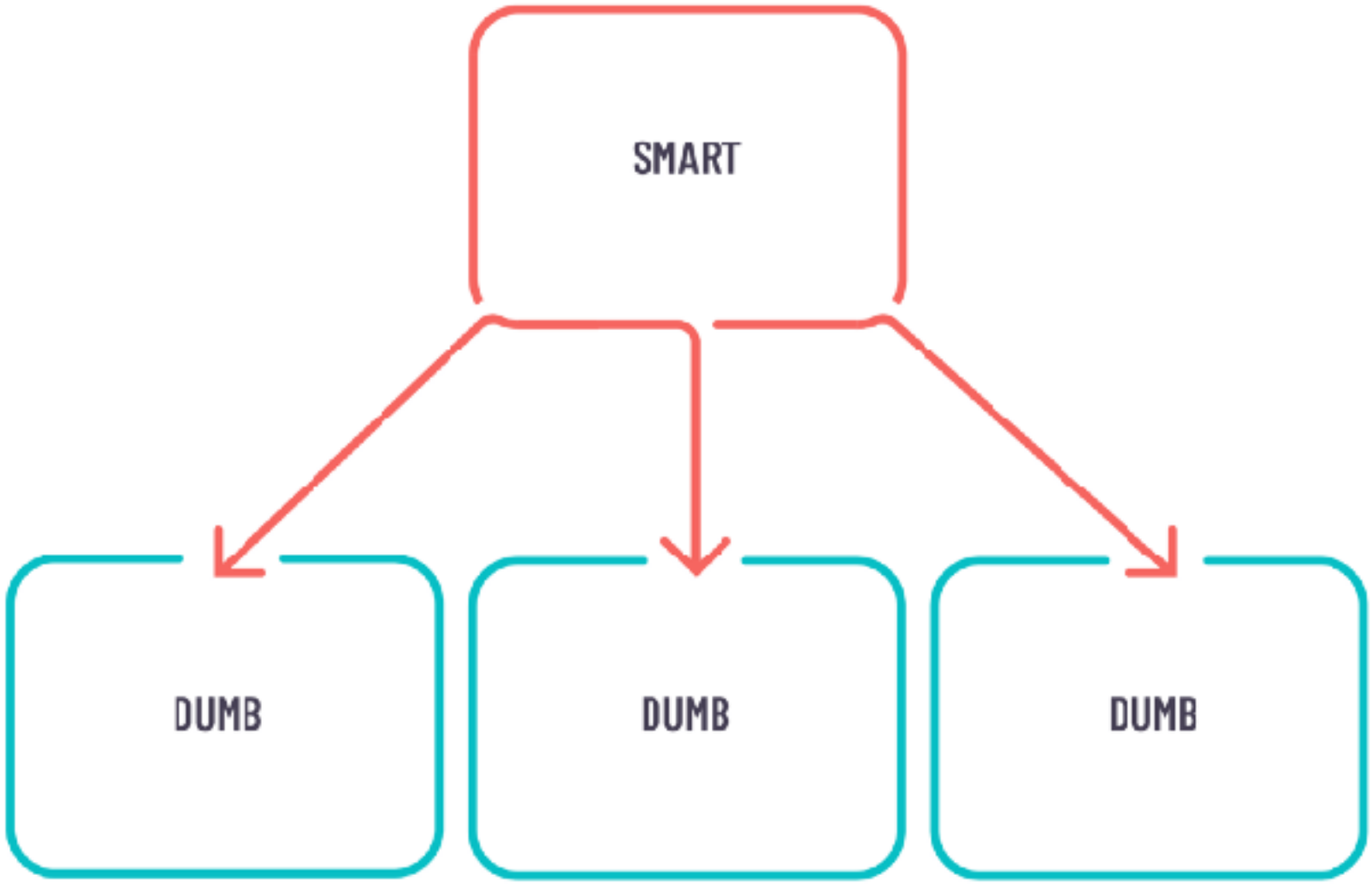
Types of component



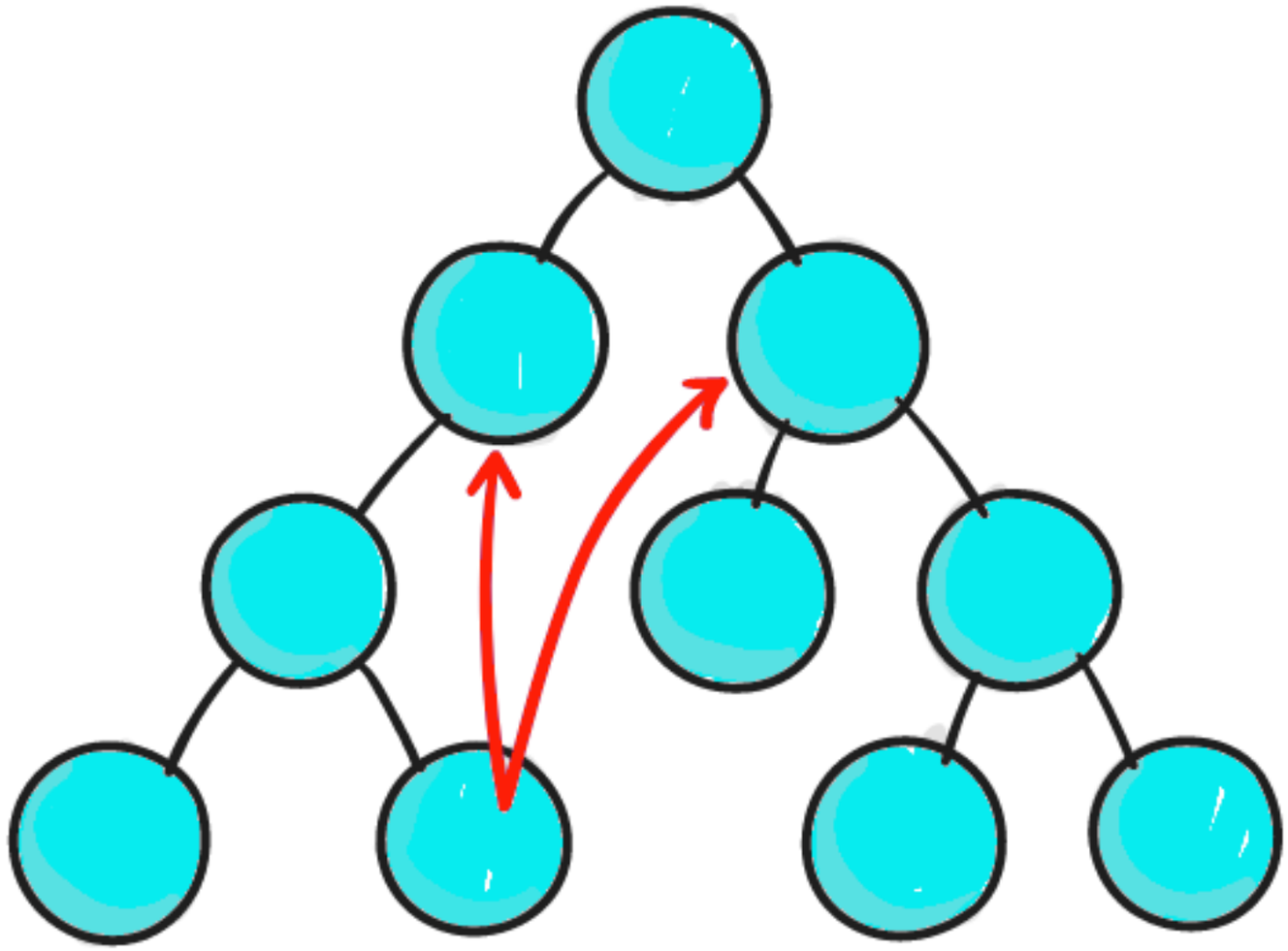
Types of component



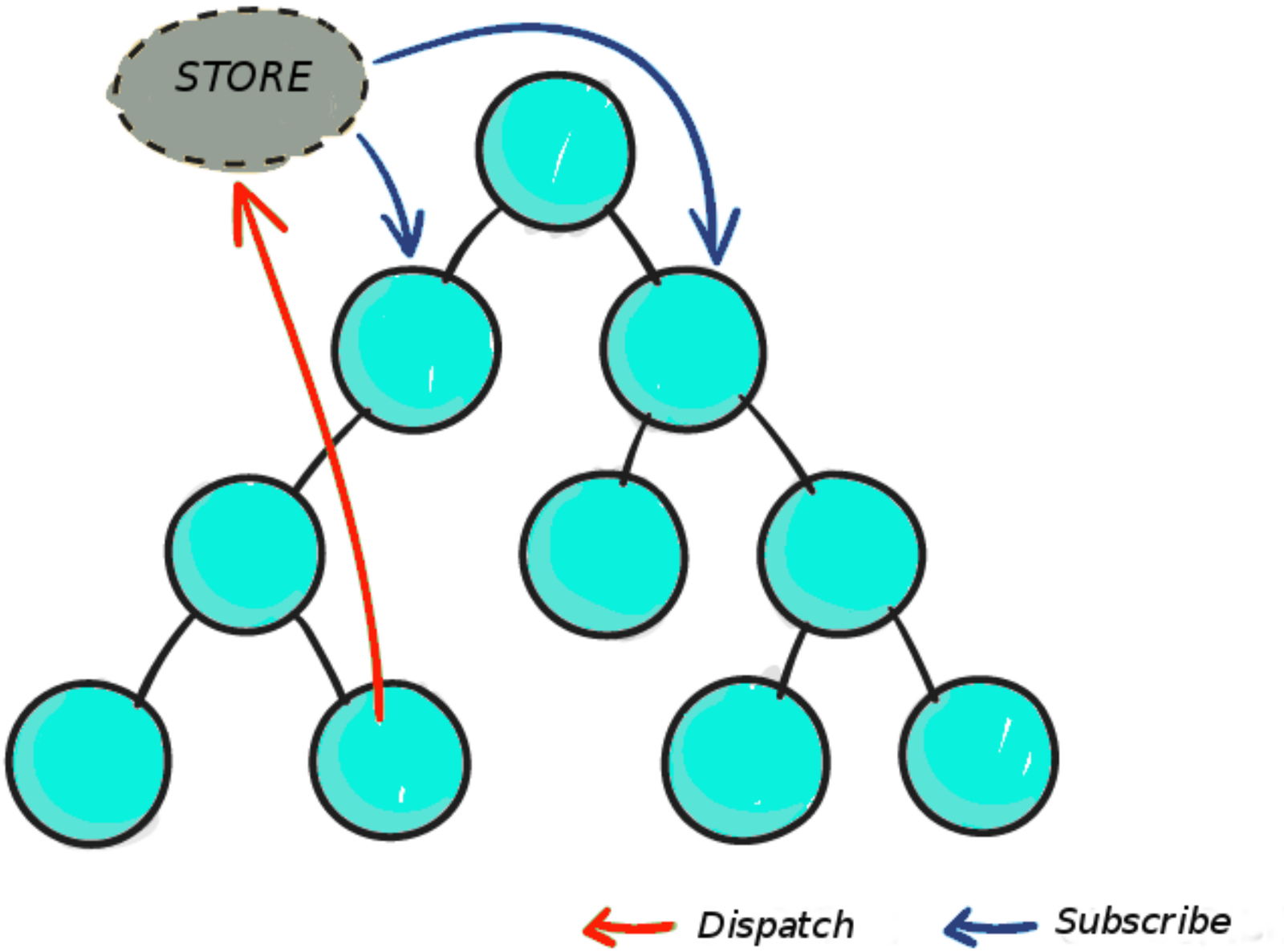
Types of component



Manage state of components ?



Manage state of components ?



Pages

Pages

Represent views for each specific route pattern
Create in folder **/pages**
Auto-import components

app.vue

```
<template>
  <NuxtLayout>
    <NuxtPage />
  </NuxtLayout>
</template>
```

localhost:3000

pages/index.vue

```
<template>
  <h1>Hello Nuxt.js</h1>
</template>
```

localhost:3000/demo

pages/demo.vue

```
<template>
  <h1>Demo page</h1>
</template>
```

Routing

Use **vue-router** package
Auto-generated router file

Page name	Route path
index.vue	/
demo.vue	/demo
/posts/[id].vue	/posts/:id
/hello/index.vue	/hello
/hello/view.vue	/hello/view

https://nuxt.com/docs/getting-started/routing

Posts/:id

posts/index.vue

```
<template>
  <h1>Post Index</h1>
</template>
```

posts/[id].vue

```
<template>
    <div>Post id =
{{ $route.params.id }}</div>
</template>
```

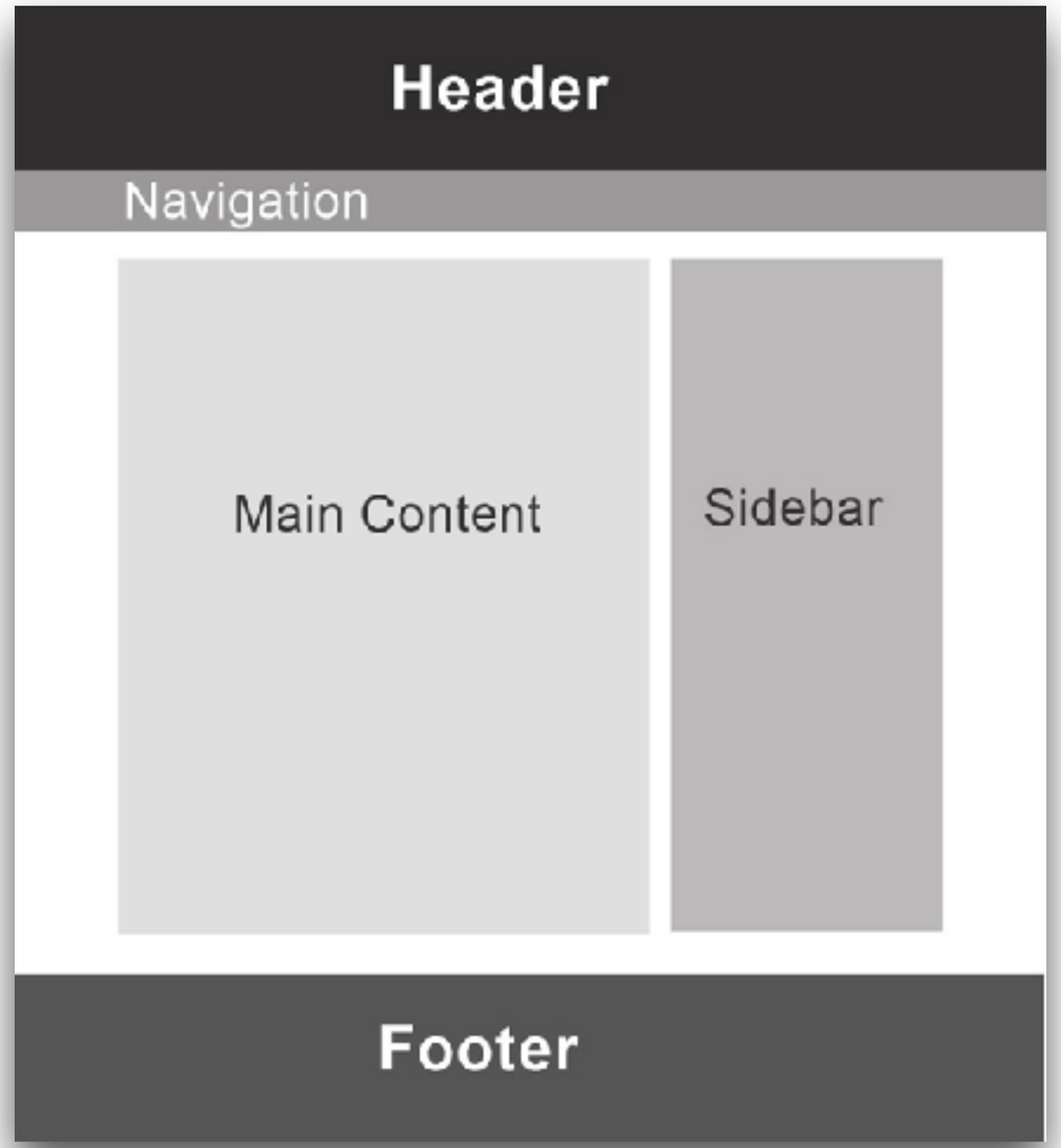
Layouts

Layouts

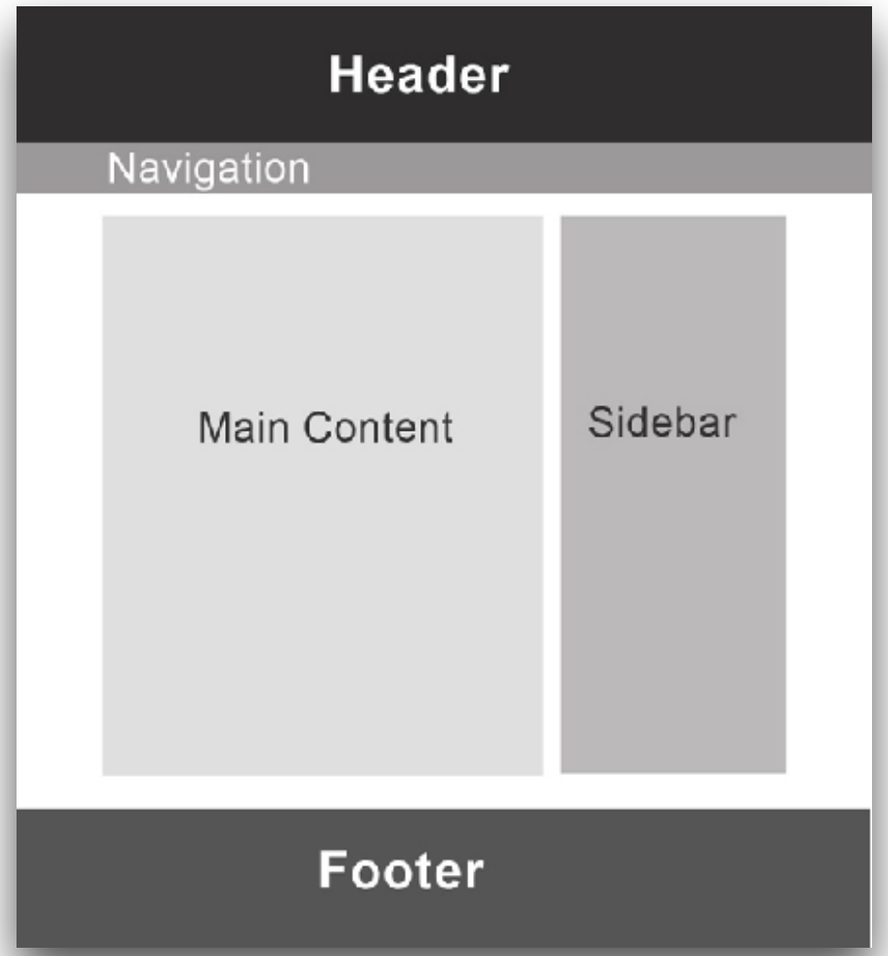
Wrapper around pages
eg. Header, Menu and Footer
Create in folder **/layouts**

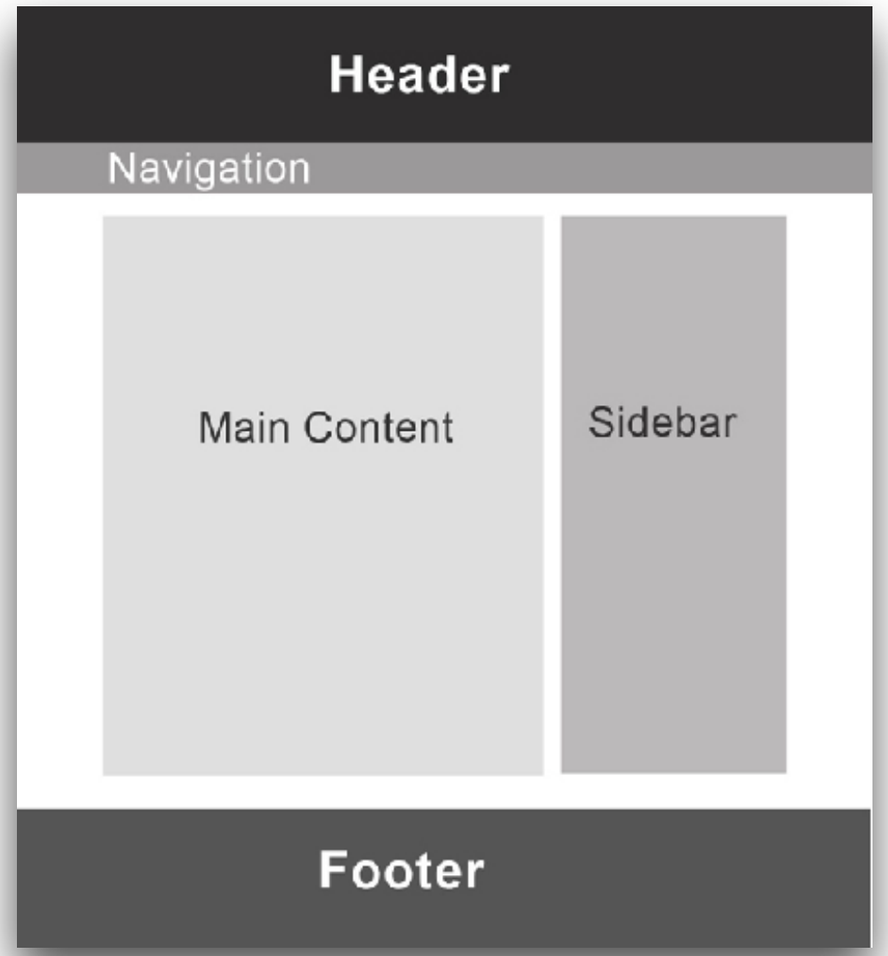
https://nuxt.com/docs/guide/directory-structure/layouts

Layouts



More layout !!





Layouts config

Layout Name	File name
default	/layouts/default.vue
dashboard	/layouts/dashboard.vue
custom-phone	/layouts/custom/phone.vue
custom	/layouts/custom/index.vue

Select layout

Layout per page
Layout for all page (app.vue)

pages/dashboard.vue

```vue
<script setup lang="ts">
definePageMeta({
  layout: 'dashboard'
})
</script>
```

app.vue

```vue
<script setup lang="ts">
  const layout =
"custom";
</script>

<template>

<NuxtLayout :name="layout
">
    <NuxtPage />
  </NuxtLayout>
</template>
```

Composables

Composables

Composables are functions that leverage Vue.js composition API to create **reusable stateful logic**

Preventing repetition in writing functions that apply in multiple components

Create in folder **/composables**

https://nuxt.com/docs/guide/directory-structure/composables

Layers

https://nuxt.com/docs/getting-started/layers

Layers

Extend a default Nuxt app to reuse

Components

Utils

Configuration

https://nuxt.com/docs/getting-started/layers

Layers

Extends from ?

Local layer

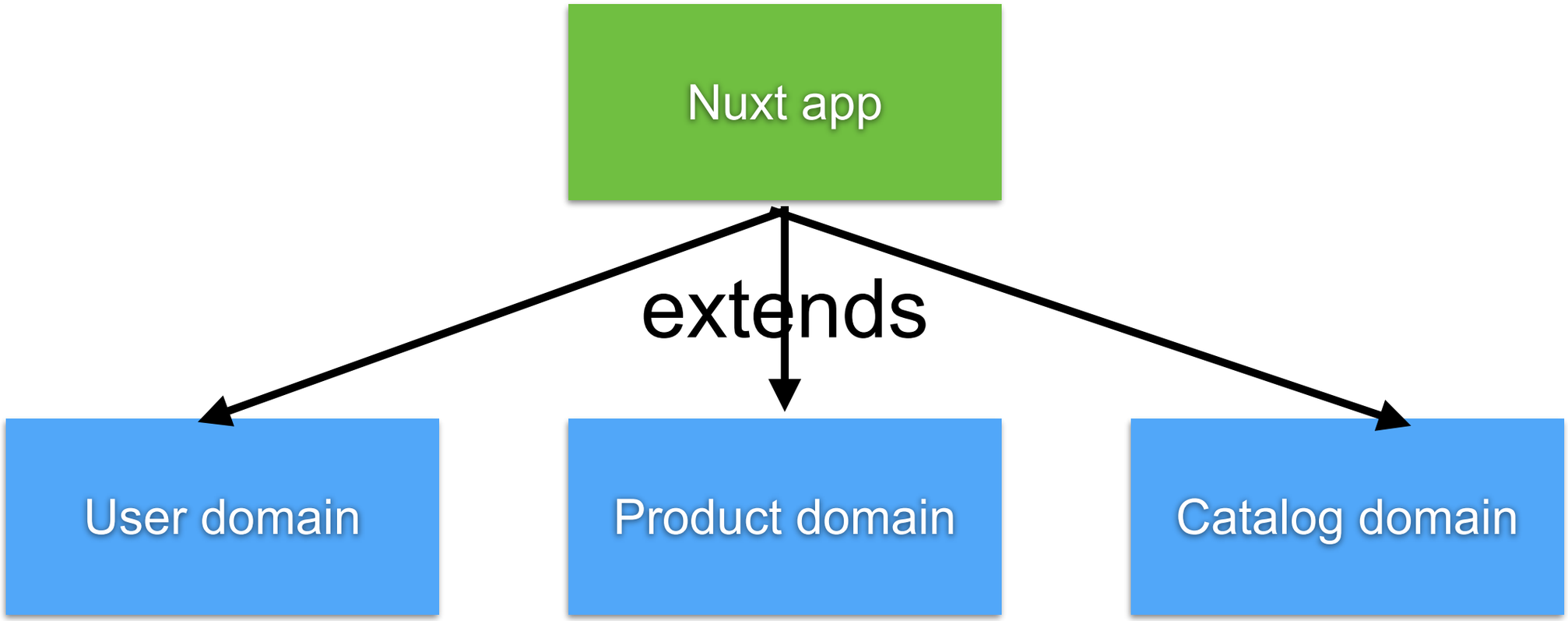
NPM module

Git repository

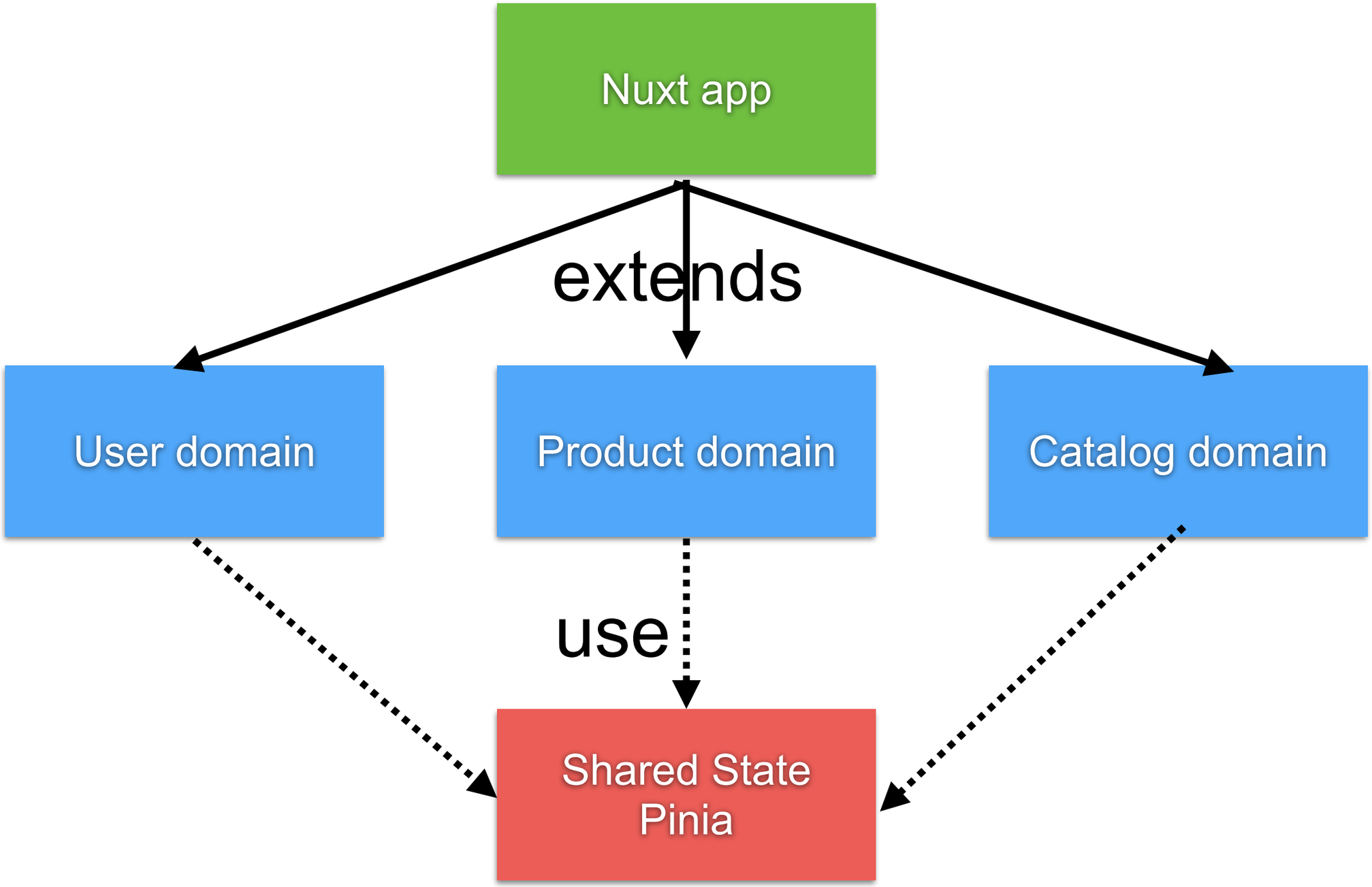
https://nuxt.com/docs/getting-started/layers

Project structure

DDD (Domain-Drive Design)



Shared state



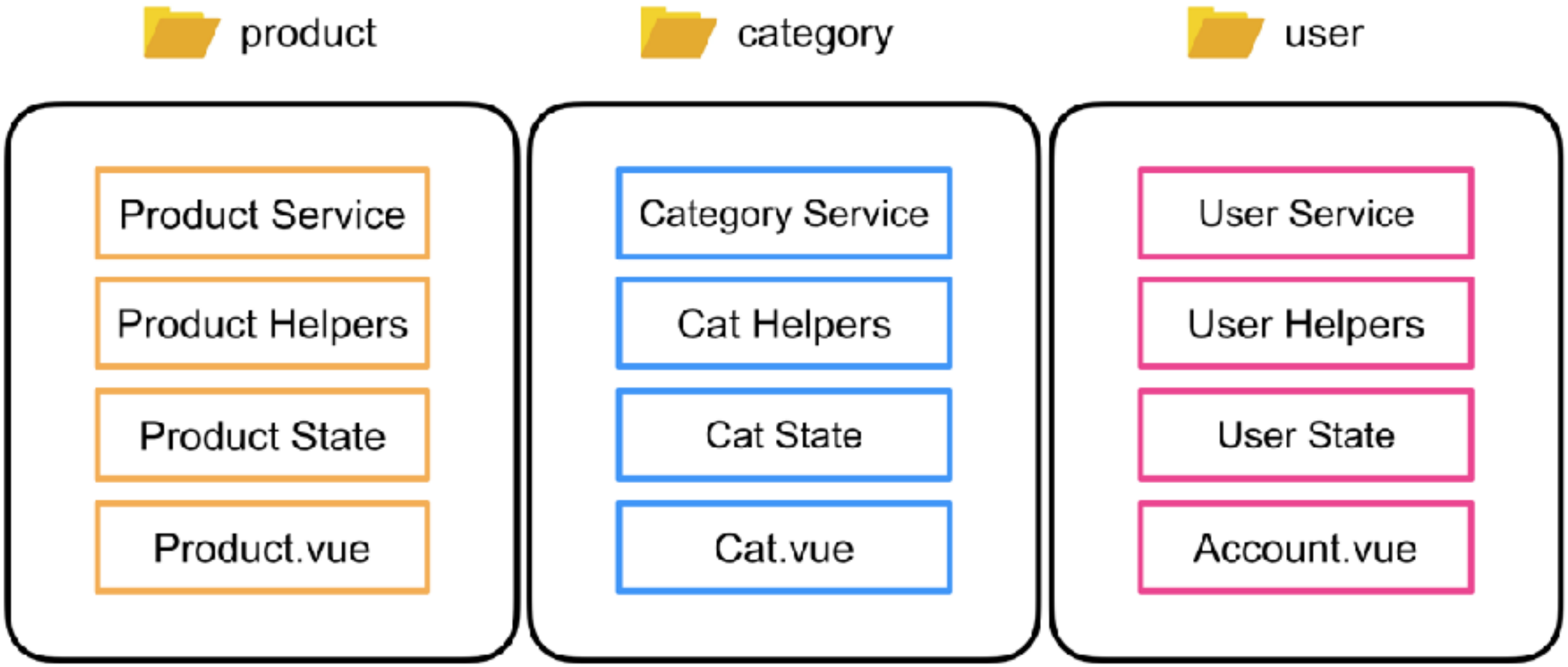
Modules

Local module

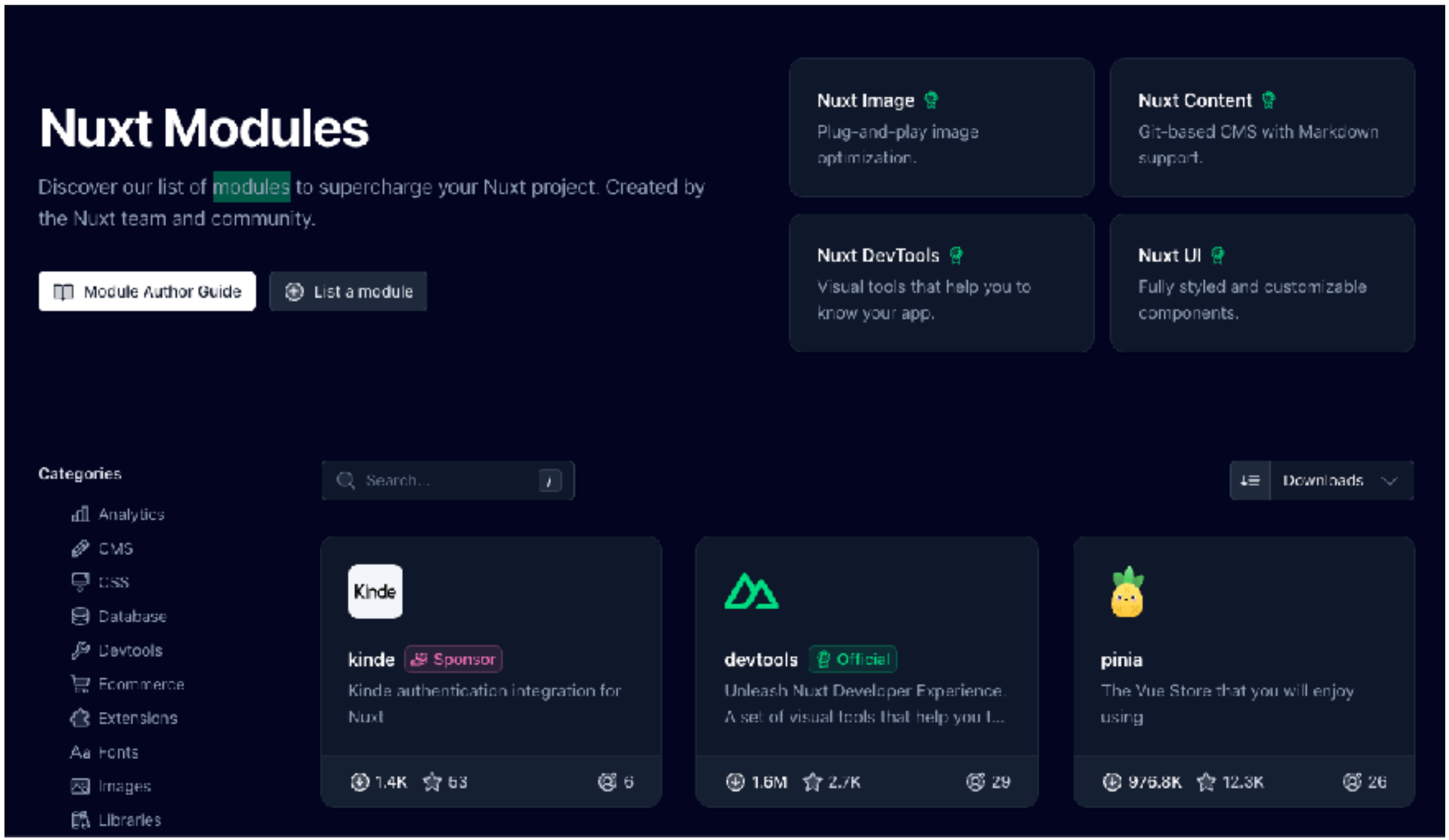
Create in folder **/modules**

https://nuxt.com/docs/guide/directory-structure/modules

Multi-modules (by domain)



Nuxt modules



https://nuxt.com/modules

Nuxt Configuration

Store

Static

Assets

Middlware

Nuxt Config

Components

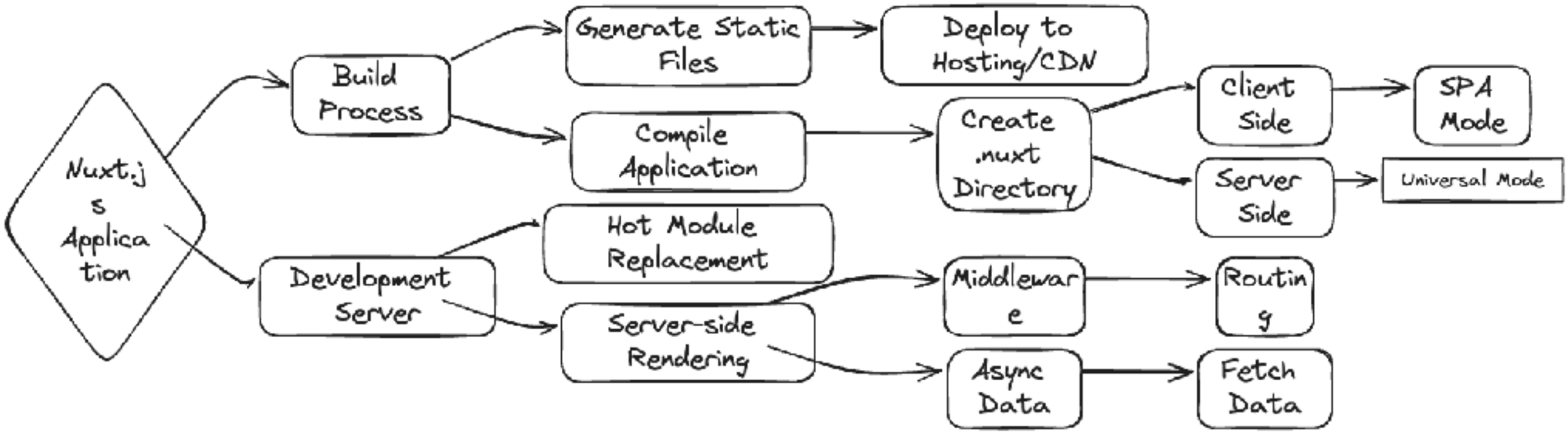
Layouts

Pages

Plugins

https://nuxt.com/docs/api/nuxt-config

NuxtJS Workflow



Rendering Mode

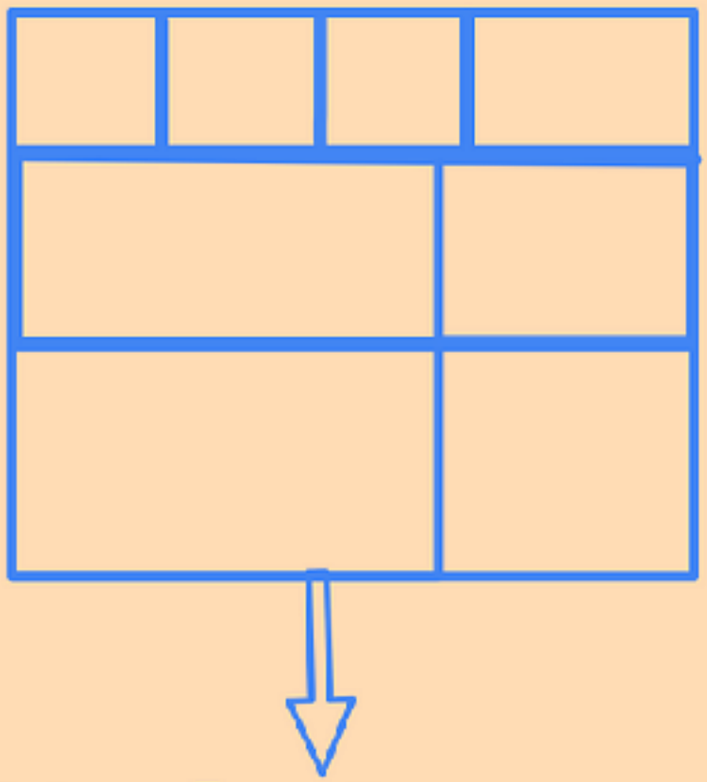
NuxtJS Rendering Modes

Config in file next.config.ts

```ts
export default defineNuxtConfig({
  devtools: { enabled: true },
  ssr: true,
  routeRules: {
    // revalidate after 10 seconds
    "/spa": { ssr: false },
    "/ssr": { ssr: true },
    "/ssg": { prerender: true },
    "/isr_ttl": { isr: 10 },
  },
})
```

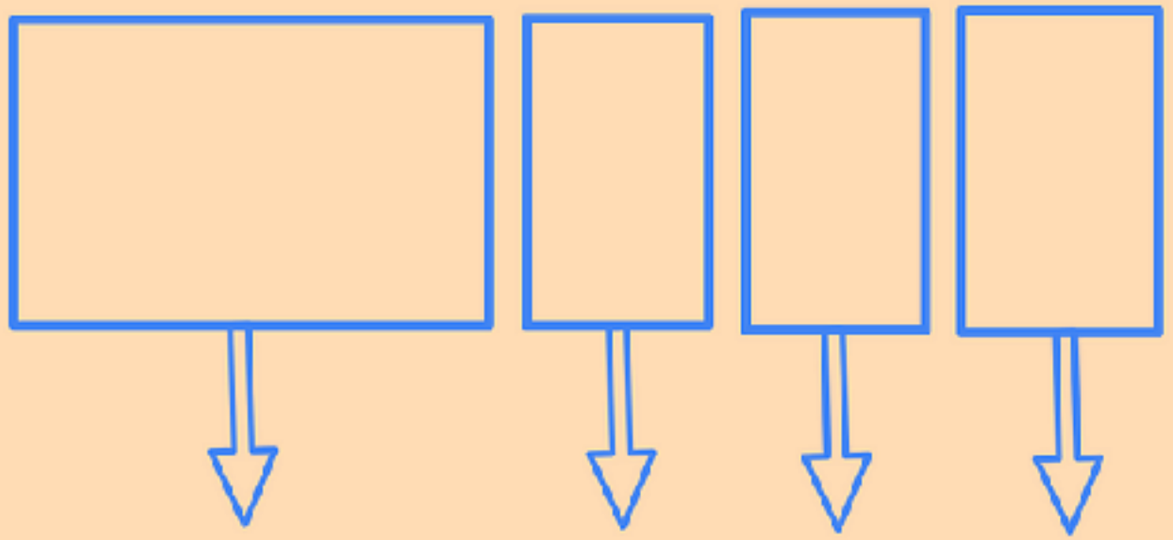
https://nuxt.com/docs/guide/concepts/rendering

Code Splitting



Single Request

* One Request, reduced latency
* Large Bundle, increased time to Interaction



Multiple Request

* Fetch only what you want
* Reduced time to Interaction
* Multiple requests, increased latency
* Increased complexity

https://nuxt.com/docs/getting-started/introduction

1. SPA or CSR

SPA (Single Page Application)

Client Side Rendering (CSR)

HTML elements are generated after the browser downloads and parses all the JavaScript code containing the instructions to create the current interface

Single-Page Application (SPA)

Step 1

The page loads and requests static assets from the web server.

Step 2

The frontend framework renders the layout using the HTML and CSS.

The JavaScript is executed and requests data.

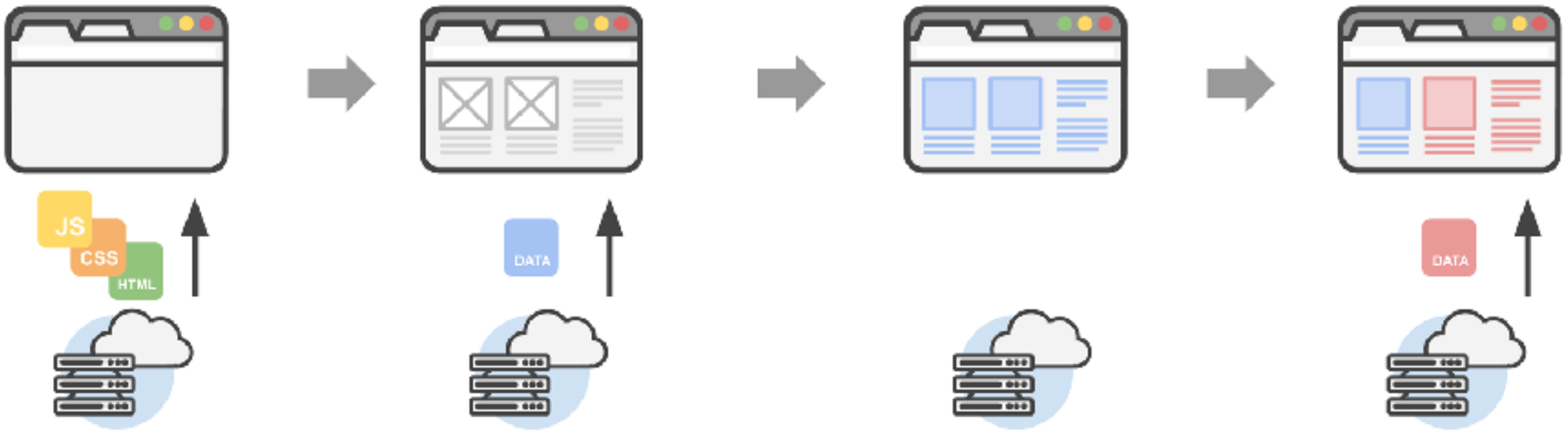
Step 3

The JavaScript loads the data in the page.

Step 4

The user interacts with the page, more data is requested and loaded.

The page is updated without having to reload.



SPA (Single Page Application)

Client Side Rendering (CSR)

```
export default defineNuxtConfig({
  routeRules: {
      "/spa": { ssr: false },
  }
})
```

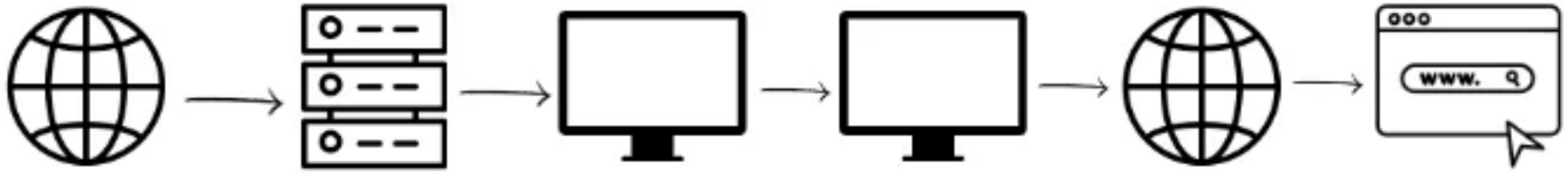
2. SSR

SSR (Server Side Rendering)

Universal Rendering

The Nuxt server generates HTML on demand and delivers a fully rendered HTML page to the browser

SSR (Server-side Rendering)



User request a site

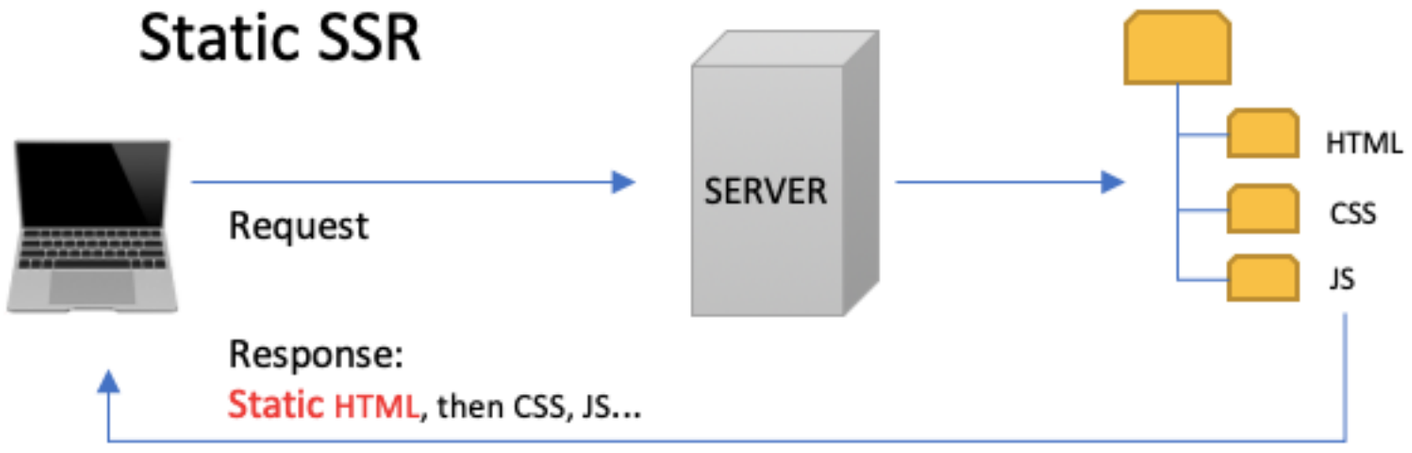
Server create ready HTML files

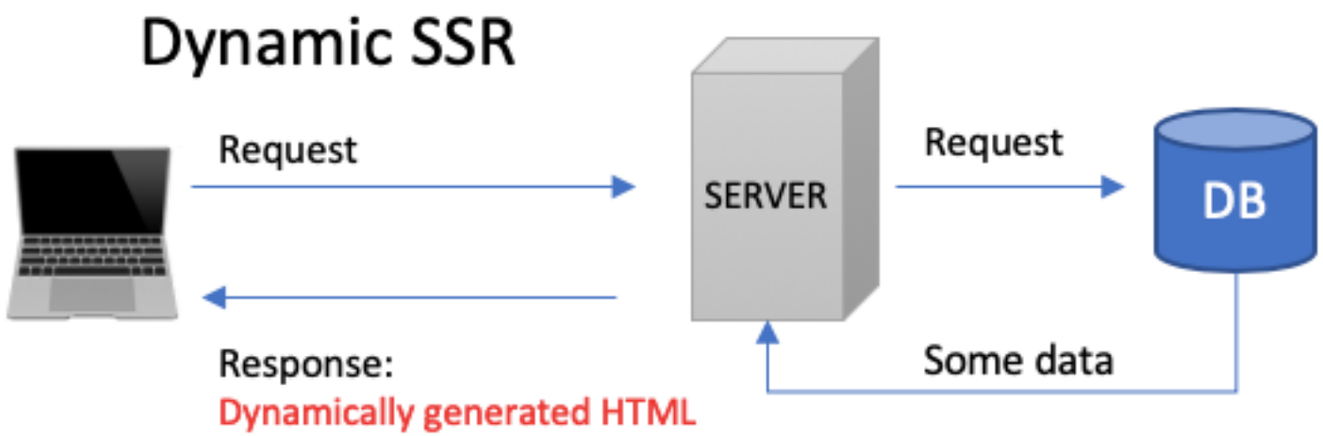
Browser render HTML but its not interactive

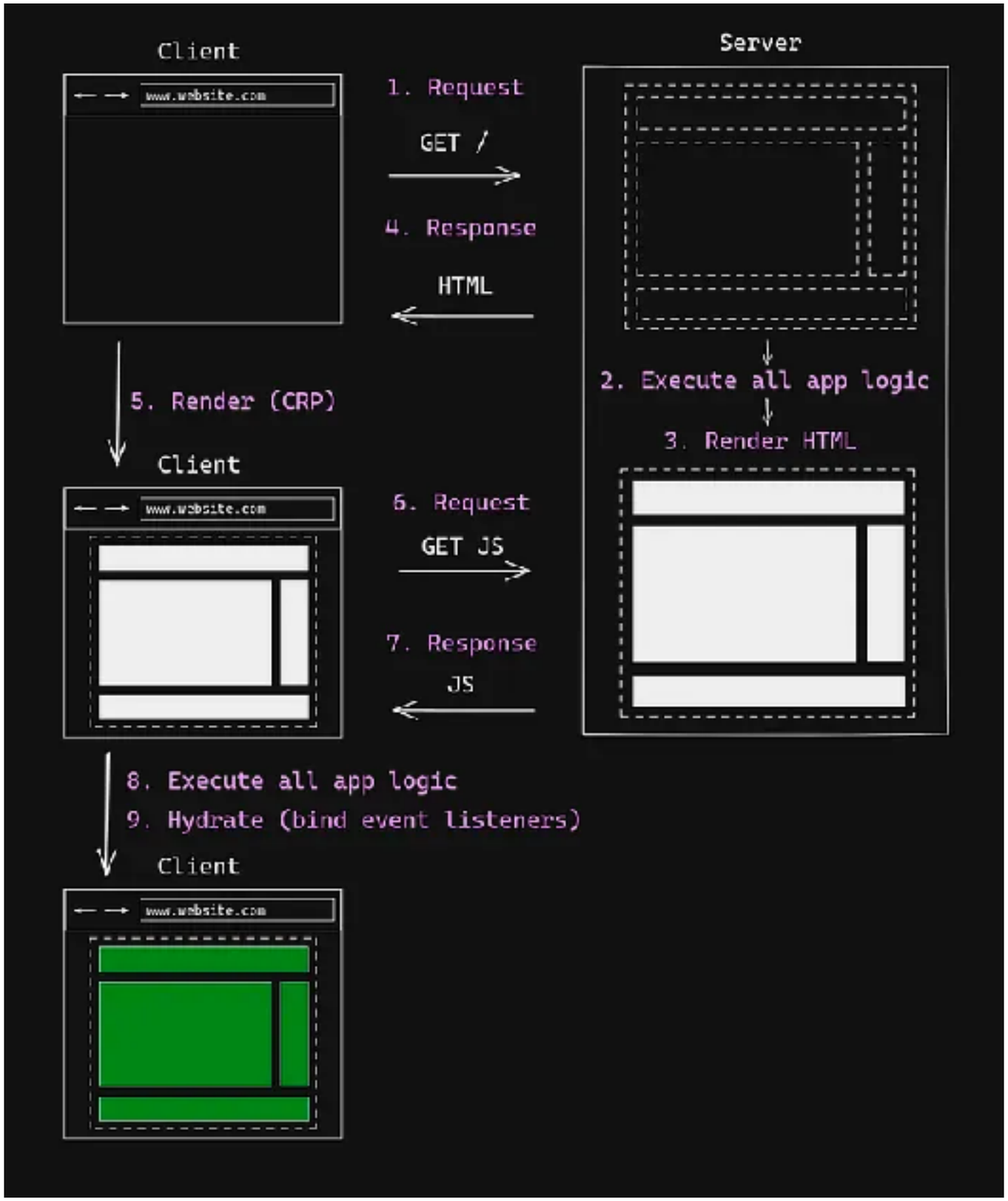
Browser downloads JS

Browser executes JS

Website is fully interactive





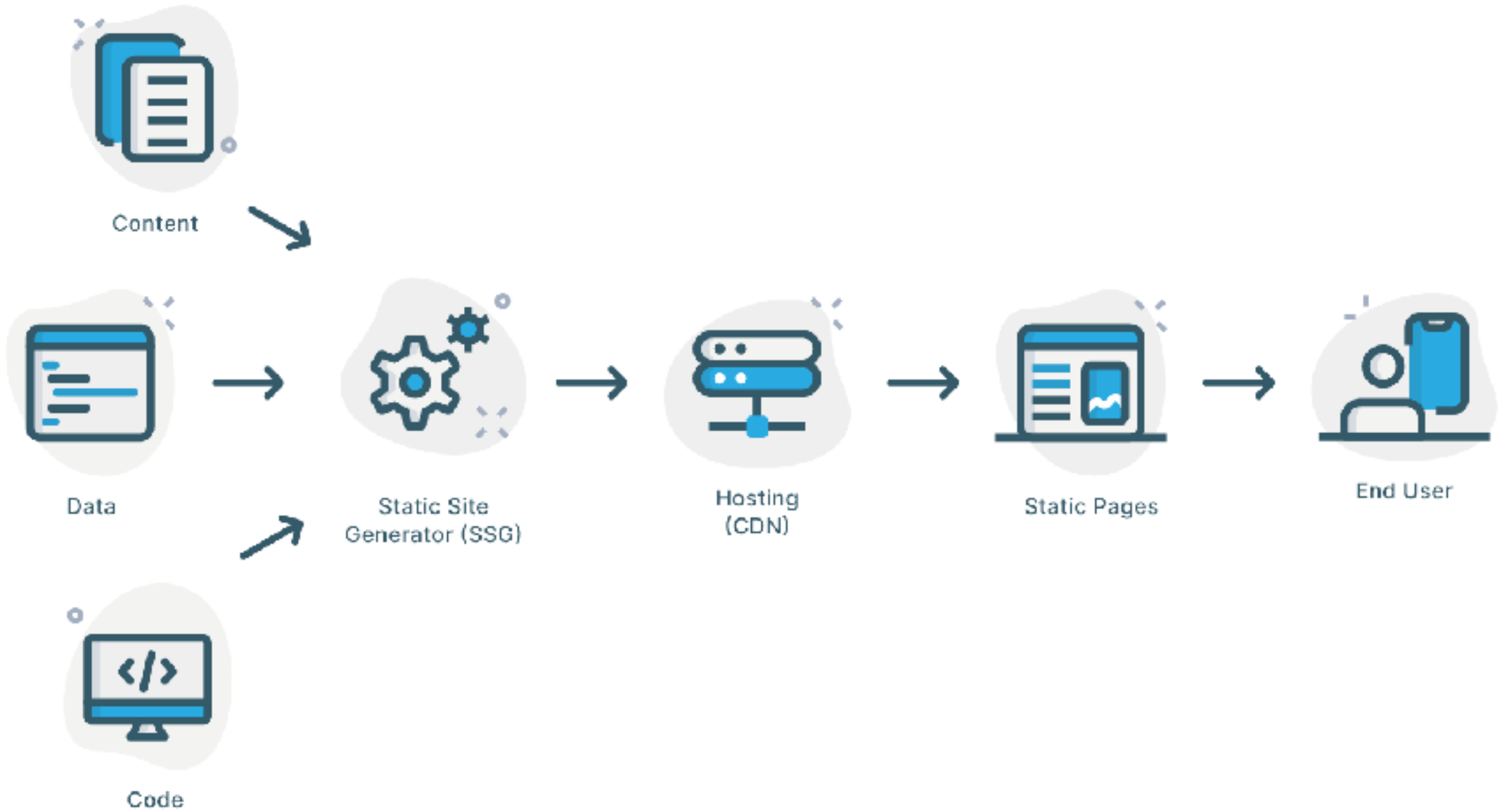


3. SSG

SSG (Static Site Generation)

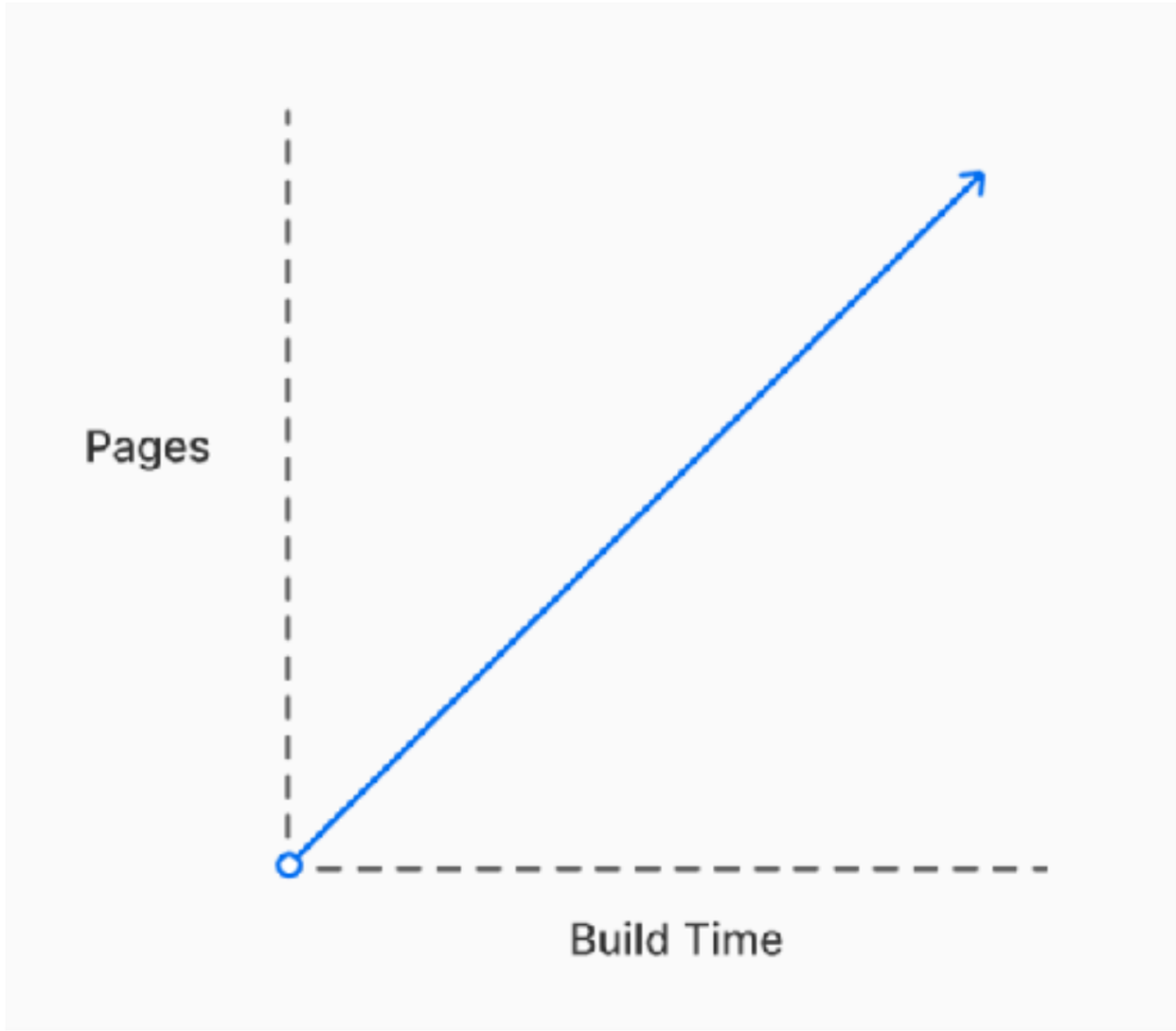
The page is generated at build time, served to the browser, and will not be regenerated again until the next build

SSG (Static Site Generation)



SSG (Static Site Generation)

Problem !!

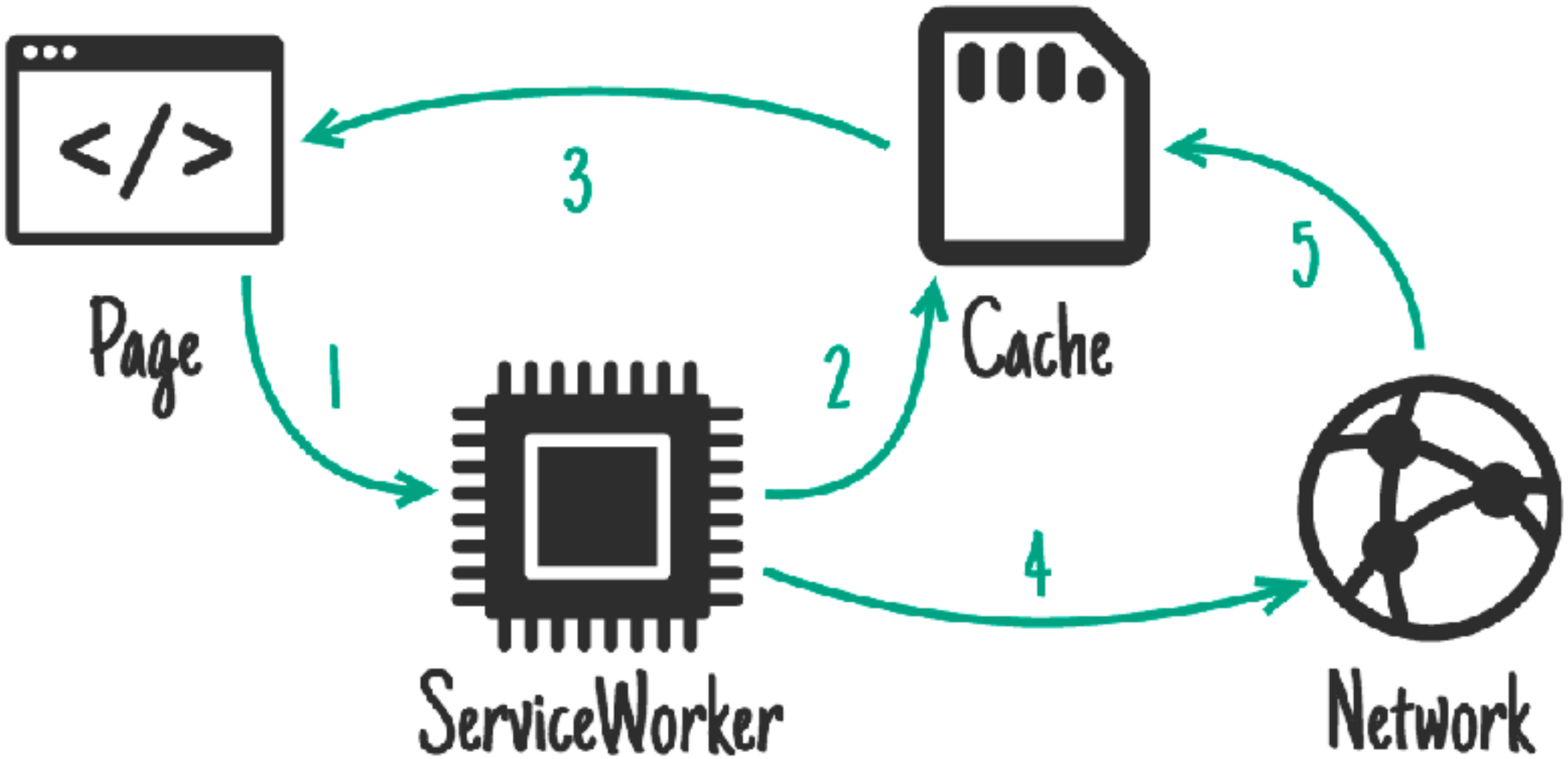


SWR (Stale While Revalidate)

Enable the server to provide stale data while simultaneously revalidating it in the background

Provide caching mechanism for data

SWR (Stale While Revalidate)



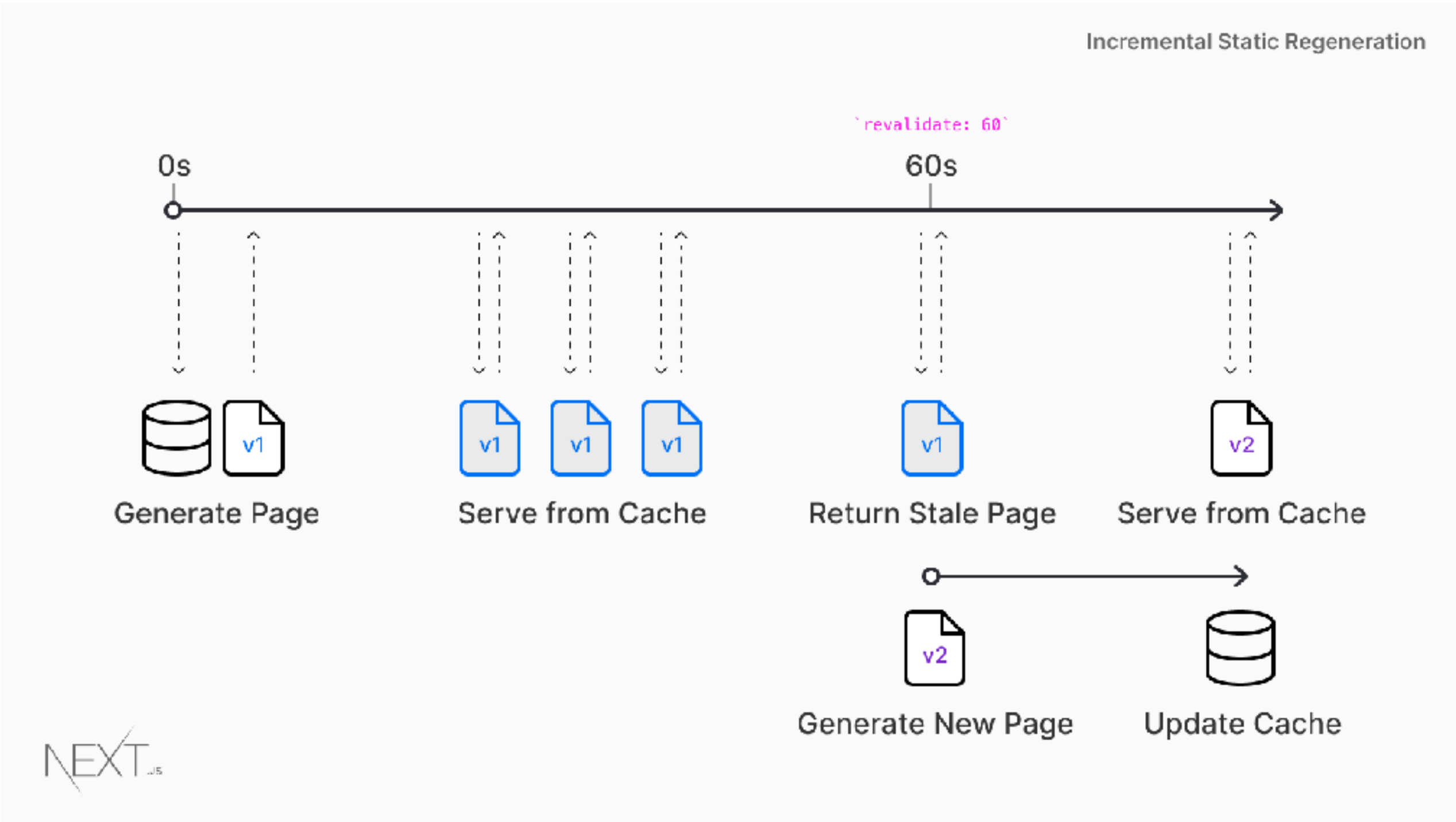
ISR

ISR (Incremental Static Regeneration)

Hybrid mode

This rendering mode operates similarly to SWR (Stale-While-Revalidate), with the primary distinction being that the response is cached on a CDN (Content Delivery Network)

ISR (Incremental Static Regeneration)



Workshop with NuxtJS

Learning by doing …

Steps to develop ?

Design components
Design layout
Design page
State management
Develop
Testing
Deploy with Docker

My Application

Logo | Home | Sign in | Sign up

Application name

Description and detail

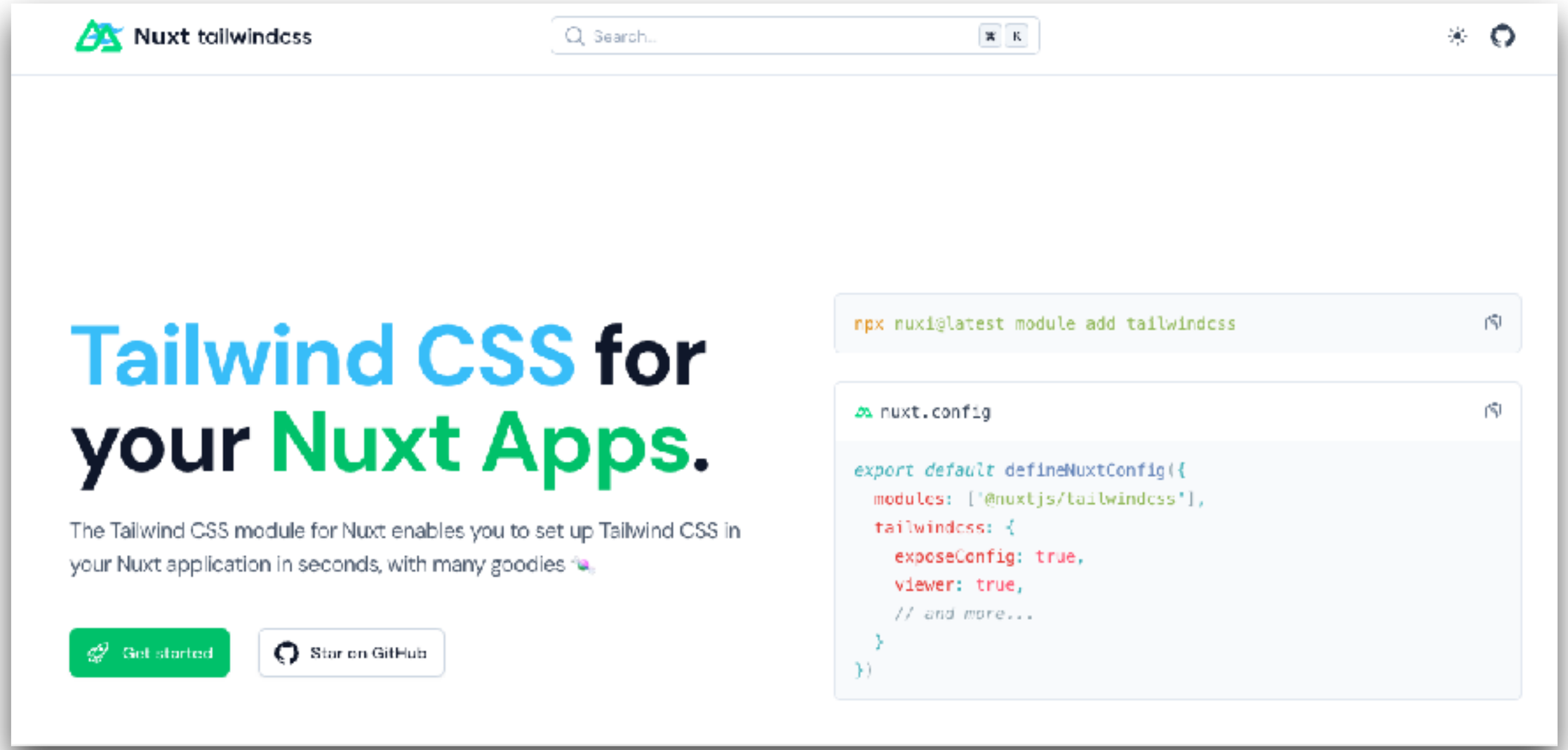
Topic name
Topic description
tag1, tag2, tag3

Popular tags

Topic name
Topic description
tag1, tag2, tag3

Paging 1, 2, 3, 4

Use Tailwind CSS



https://tailwindcss.nuxtjs.org/

Install tailwind

npm install -D @nuxtjs/tailwindcss

next.config.ts

```ts
export default defineNuxtConfig({
  devtools: { enabled: true },

  modules: ['@nuxtjs/tailwindcss'],
  css: [‘~/assets/css/global.css'],

})
```

Add tailwind module and Global CSS

Config tailwind

Create file tailwind.config.ts

```
module.exports = {
  theme: {
    extend: {
      colors: {
        'custom-green': '#5CB85C',
      },
      backgroundColor: (theme) => ({
        ...theme('colors'),
        'custom-green': '#5CB85C',
      }),
      borderColor: {
        'custom-green': '#5cb85c',
      },
      fontFamily: {
        titillium: ['Titillium Web', 'sans-serif'],
      },
    },
  },
};
```

https://tailwindcss.nuxtjs.org/tailwind/config

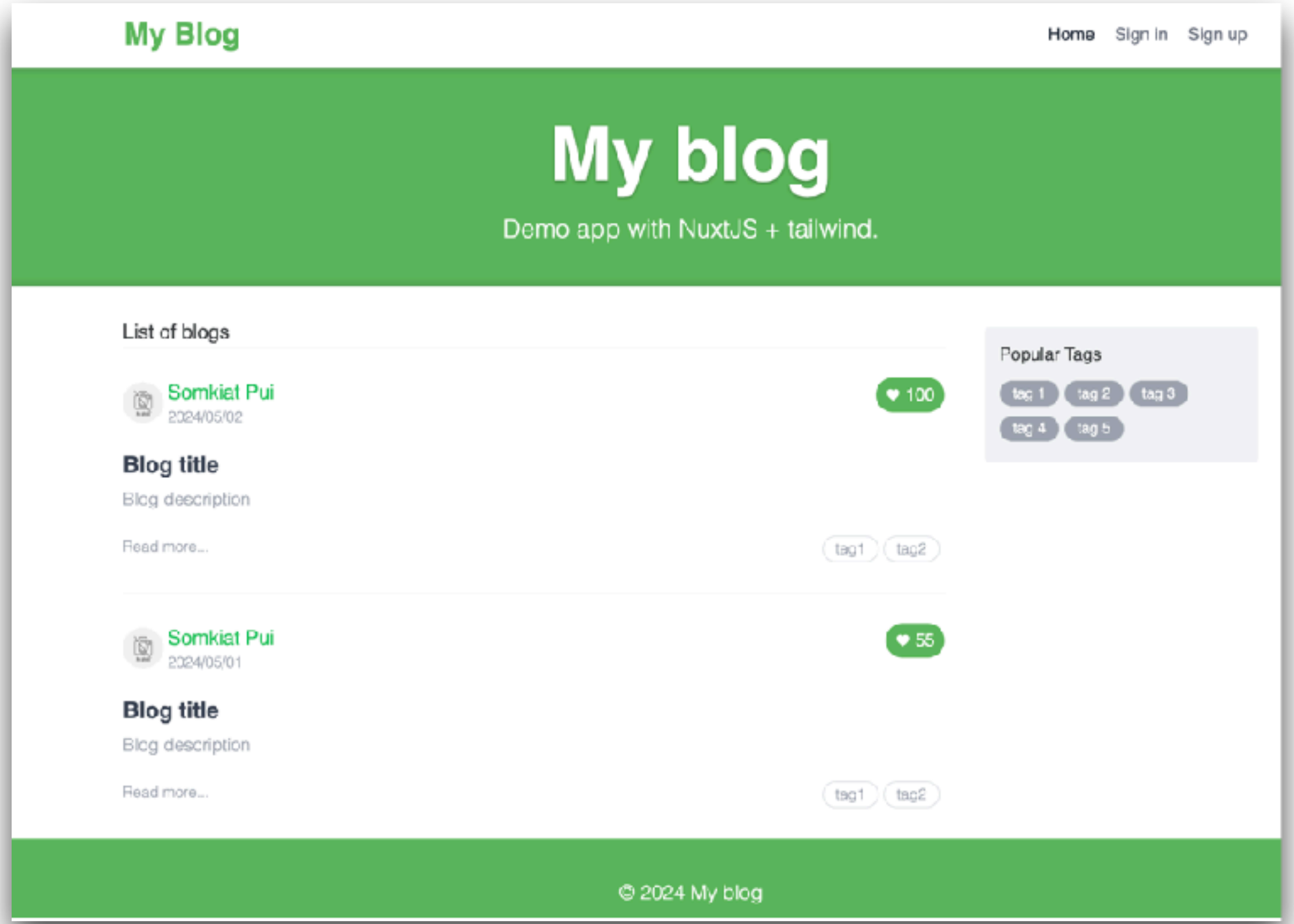
Run application again !!

npm run dev

```
Nuxt 3.11.2 with Nitro 2.9.6

  ➜ Local:    http://localhost:3000/
  ➜ Network:  use --host to expose

i Using default Tailwind CSS file                                  nuxt:t
  ➜ DevTools: press Shift + Option + D in the browser (v1.2.0)

i Tailwind Viewer: http://localhost:3000/_tailwind/                nuxt:t
i Vite client warmed up in 800ms
i Vite server warmed up in 879ms
✔ Nuxt Nitro server built in 431 ms
```

Let’s start

Home page

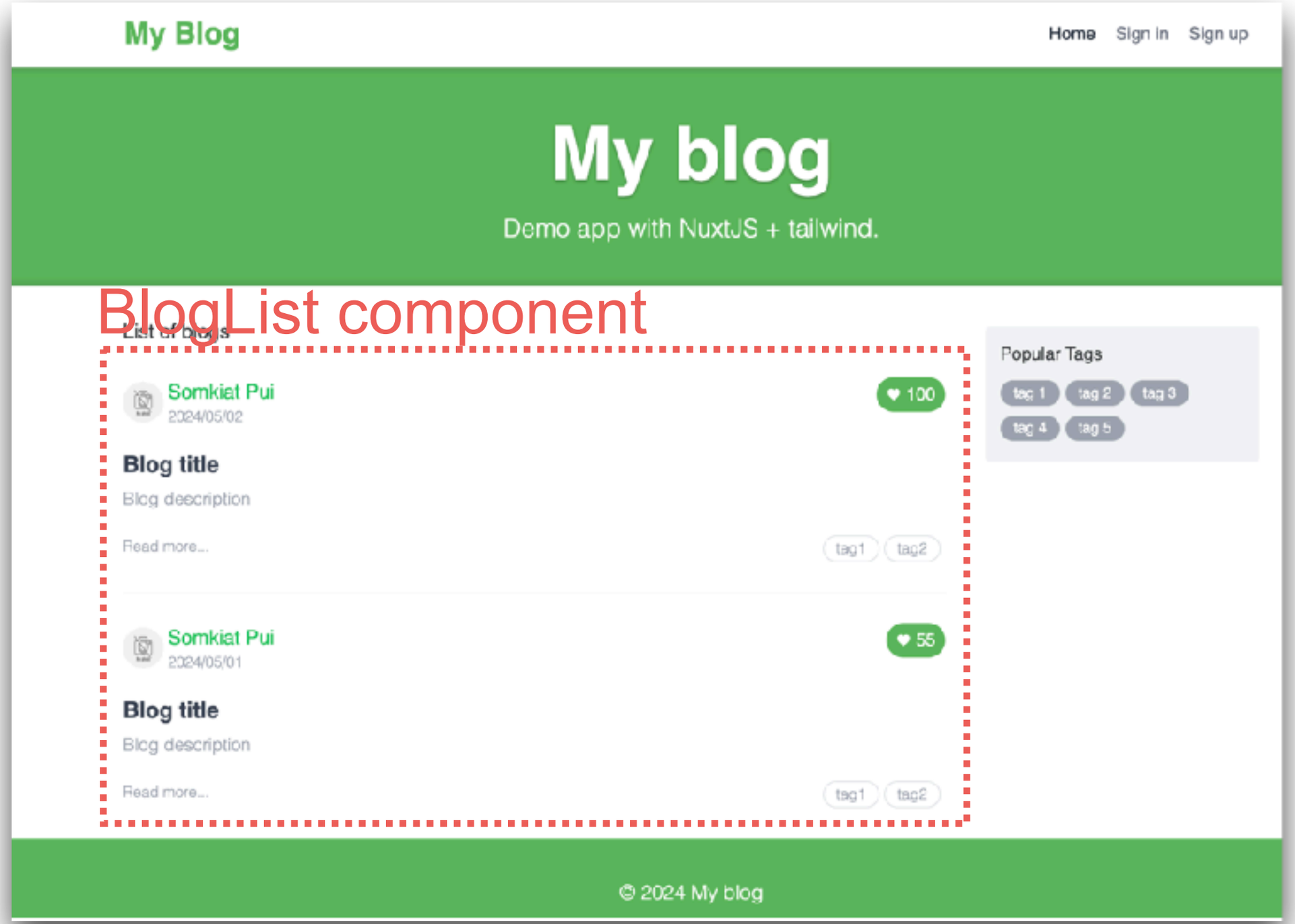


Design your components
Design your layout
Design your page

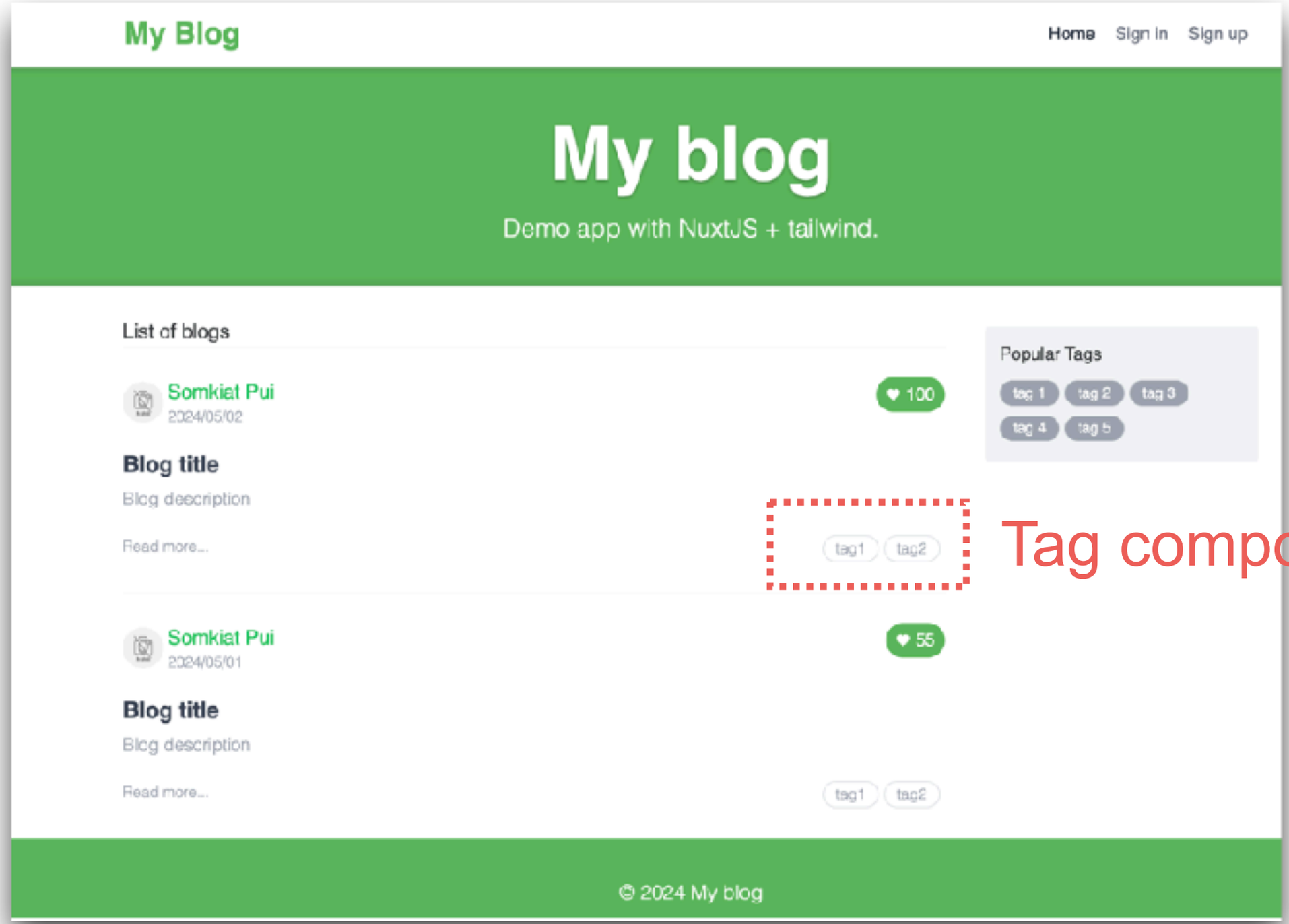
Layout of page

List of blogs

List of blogs

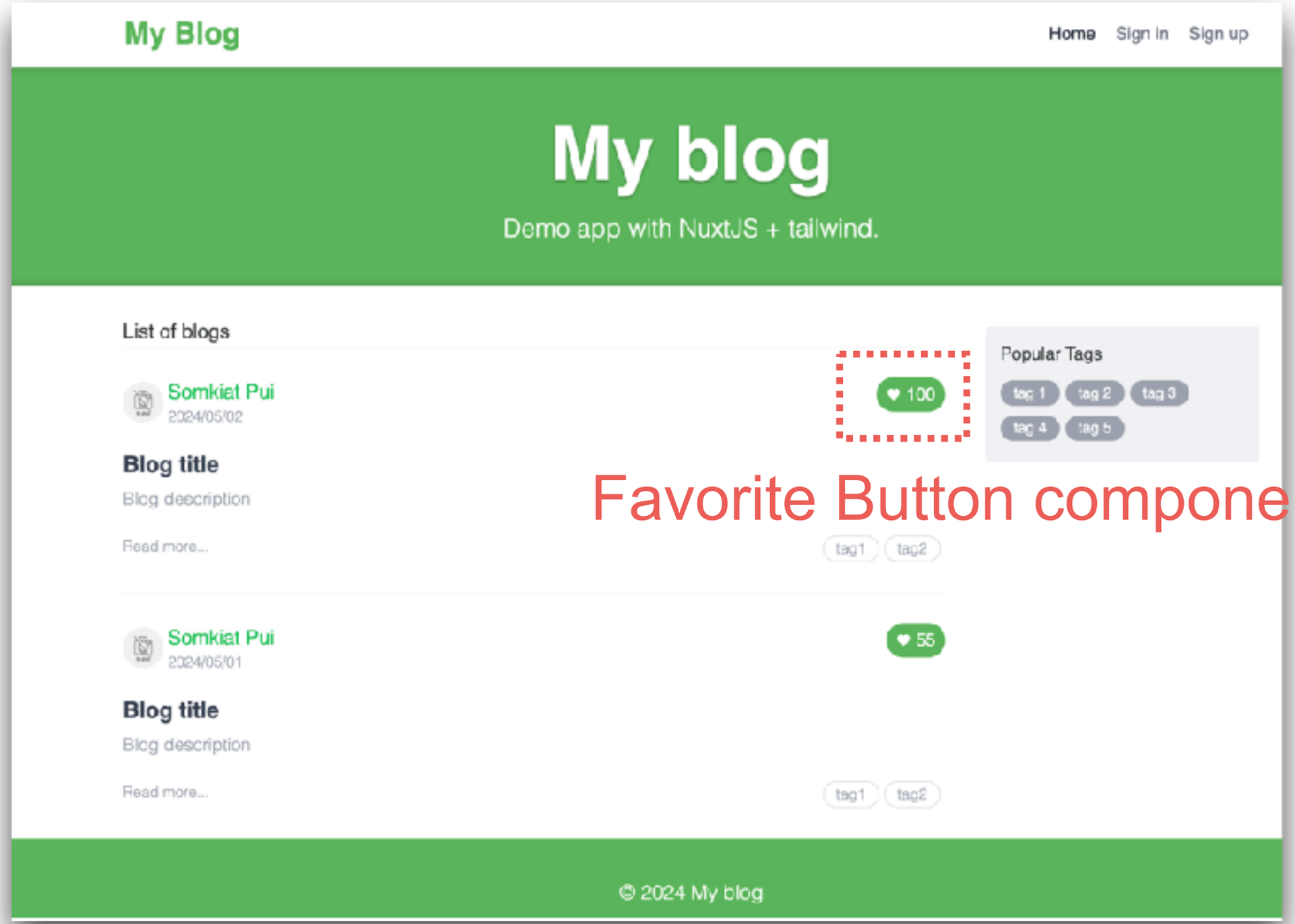


List of tags



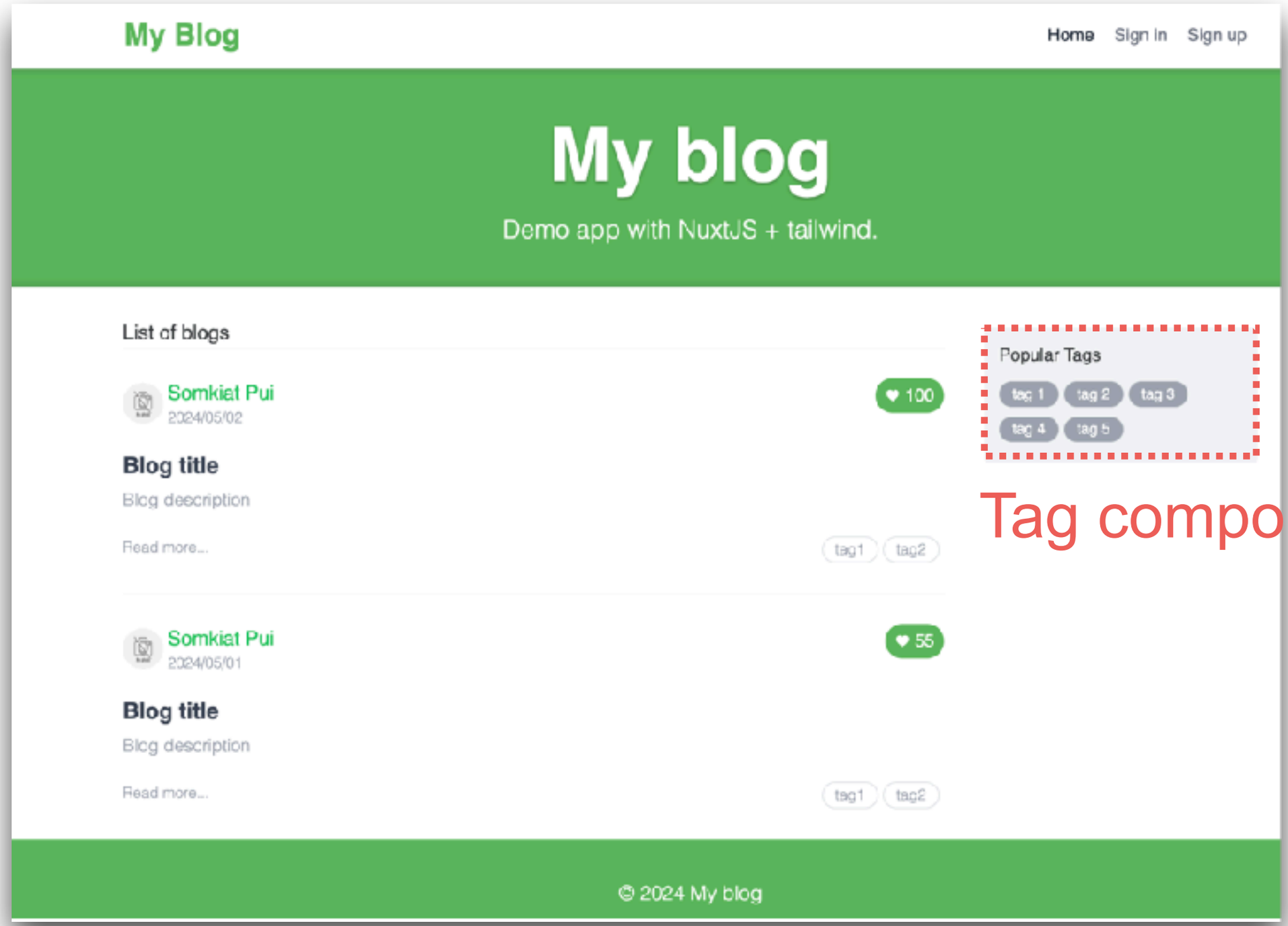
Tag component

Favorite button



Favorite Button component

Popular tags



Tips :: Assets

Images from assets or public folder

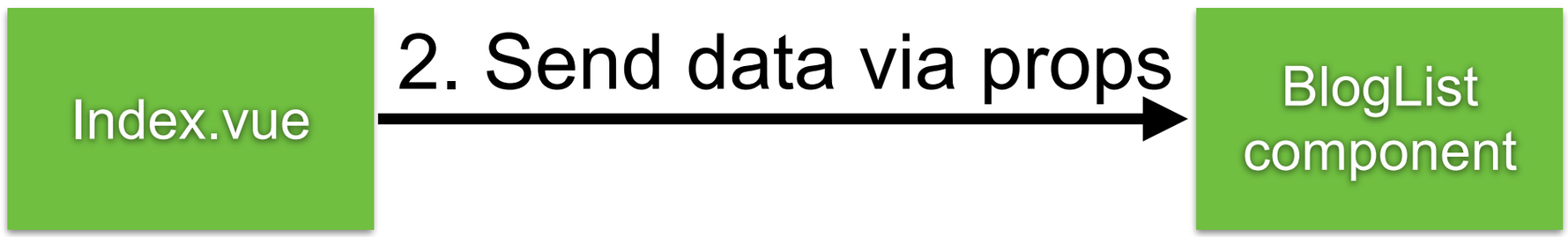
https://nuxt.com/docs/getting-started/assets

List of blogs (data flow)

List of blogs (data flow)

1. Get all blogs from API

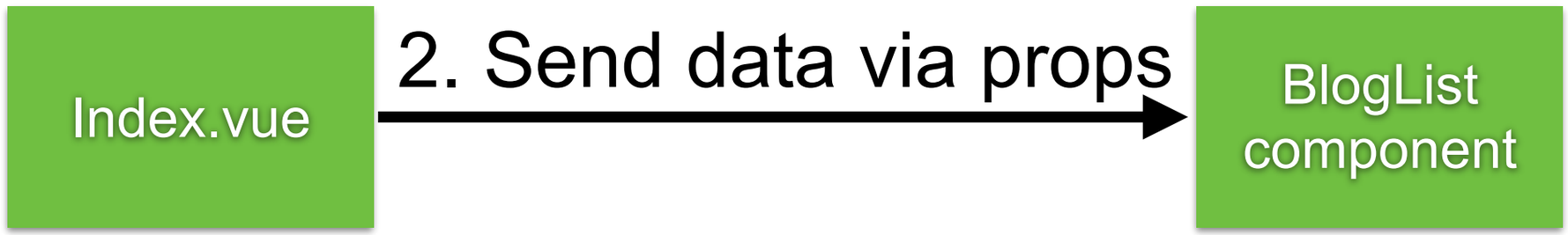
List of blogs (data flow)



1. Get all blogs from API

https://vuejs.org/guide/components/props.html

List of blogs (data flow)



1. Get all blogs from API

3. Receive and render

https://vuejs.org/guide/components/props.html

Example (1)

Page/index.vue

```vue
<script setup lang="ts">
import type { Blog } from '~/types';

 const blogList: Blog[] = [
    {
      id: 1,
      title: 'Blog 1',
      description: 'Desscription of blog 1',
      tags: ['tag 1', 'tag 2'],
      username: 'Somkiat Pui',
      createdDate: '2024/05/02',
      favoritesCount: 100,
    },
  ];
</script>
```

Types/index.ts

```ts
export interface Blog {
    id: number;
    title: string;
    description: string;
    tags: string[];
    username: string;
    createdDate: string;
    favoritesCount: number;
}
```

Send data from index page via props

```vue
<BlogList :blogs="blogList" />
```

Example (2)

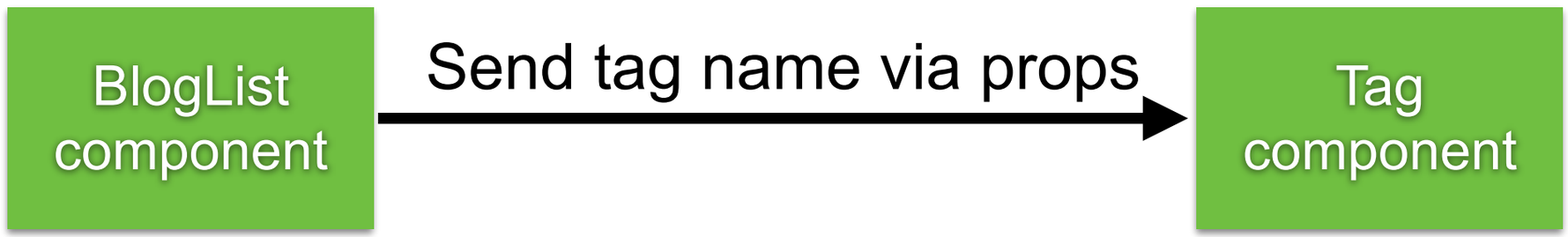
Declare props and use in BlogList component

```
<script setup lang="ts">

import { defineProps } from 'vue';
import type { Blog } from '~/types';
const props = defineProps<{
    blogs: Blog[];
}>();

</script>
```

```
<template>
    <div>
        <div v-for="blog in blogs"
            :key="blog.id">


{{ blog.username }}
{{ blog.createdDate}}
```

https://vuejs.org/api/built-in-directives.html#v-for

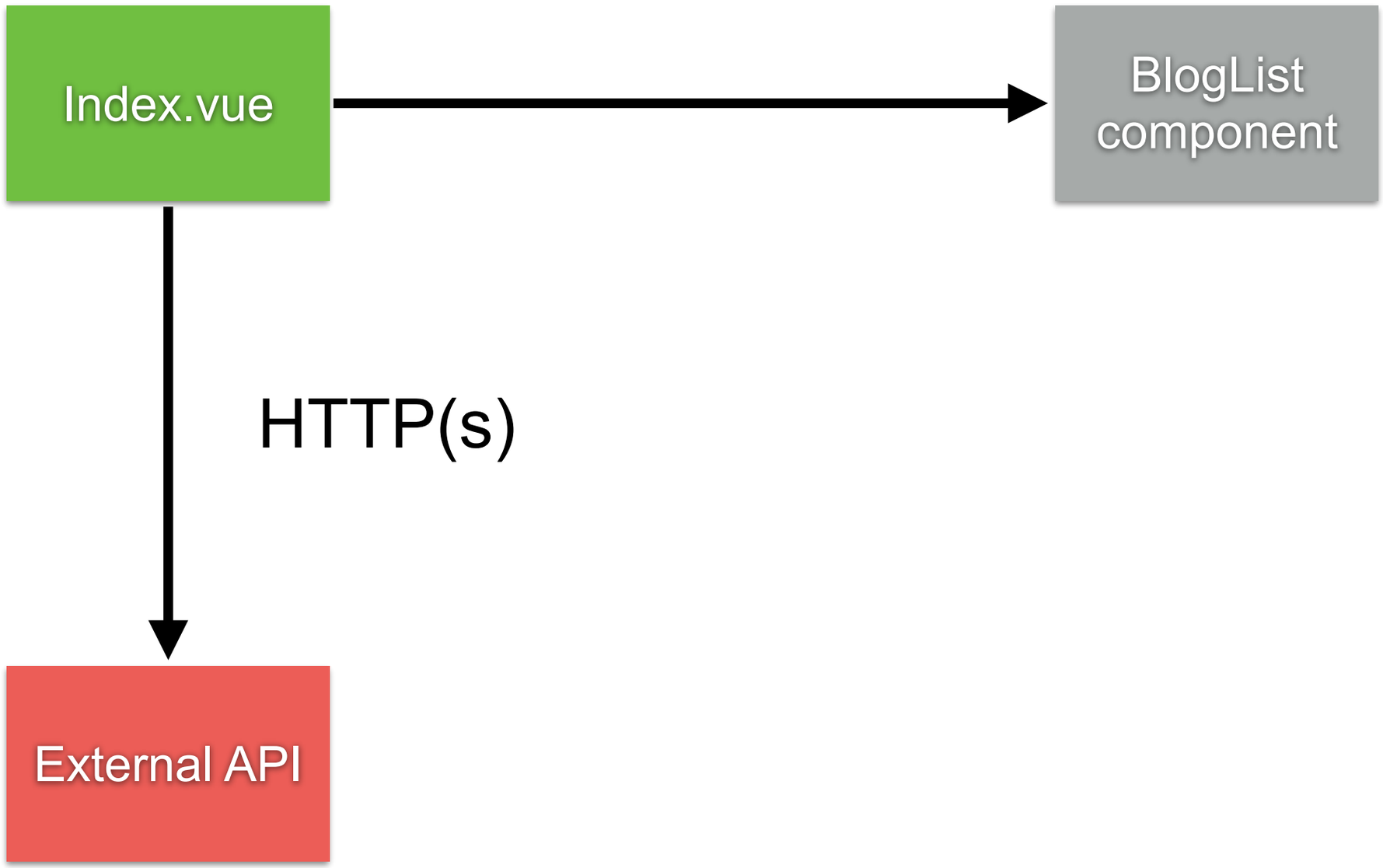
List of tags



Read data from APIs

Data fetching in NuxtJS

1. Get all blogs from API



https://nuxt.com/docs/getting-started/data-fetching

Setup API Server

Process !!

https://www.npmjs.com/package/openapi-typescript

Generate from OpenAPI

npx openapi-typescript ./postman/openapi.yml -o ./openapi.gen.ts

Types/index.ts

```
import type { components } from '~/openapi.gen';
export type Blog = components['schemas']['Blog'];
```

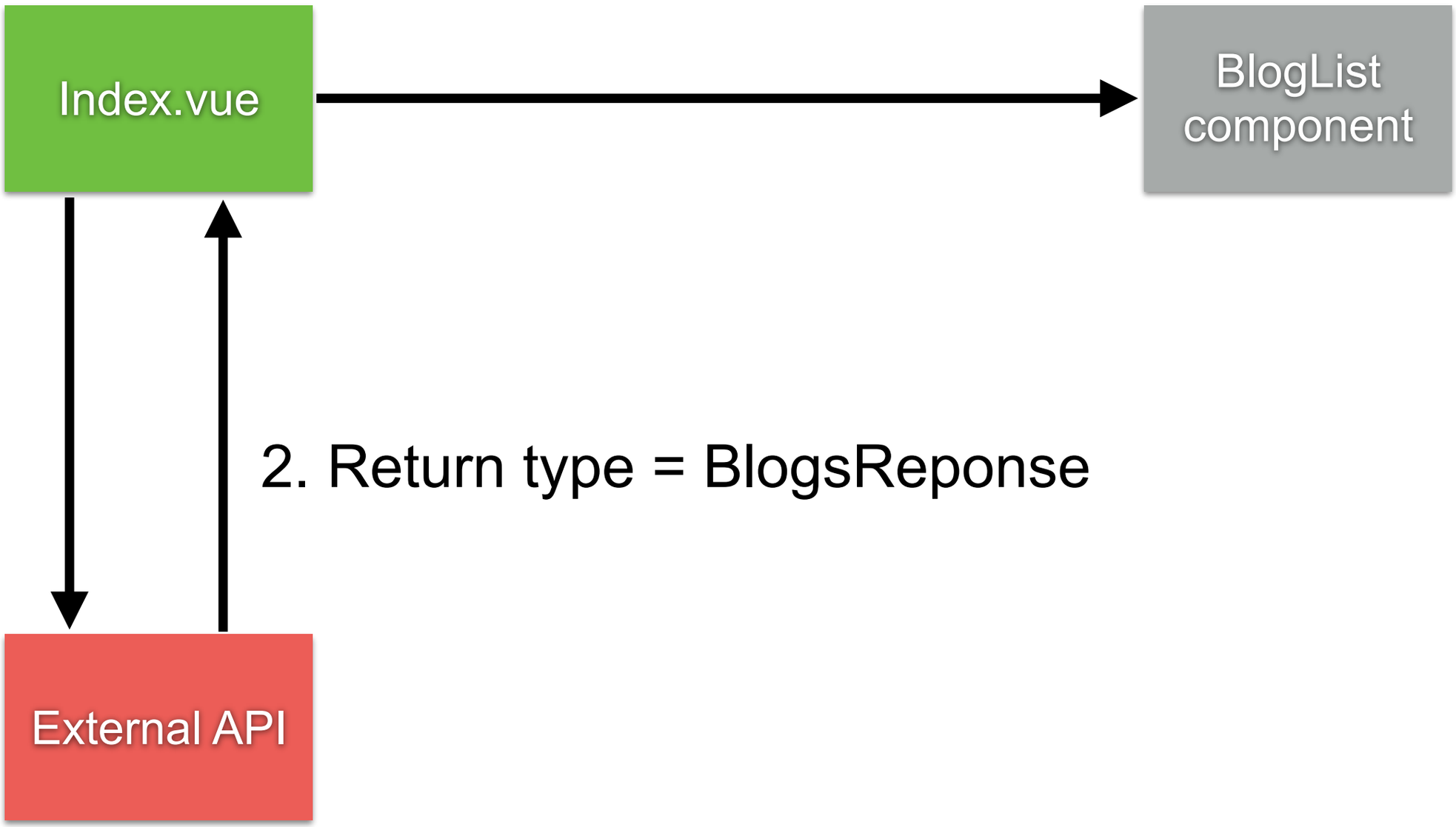
Lib/api/blog.ts

```
import type { components } from "~/openapi.gen";
export type BlogsResponse = components[“responses"]
["BlogsResponse"]["content"]["application/json"];
```

https://www.npmjs.com/package/openapi-typescript

index.vue

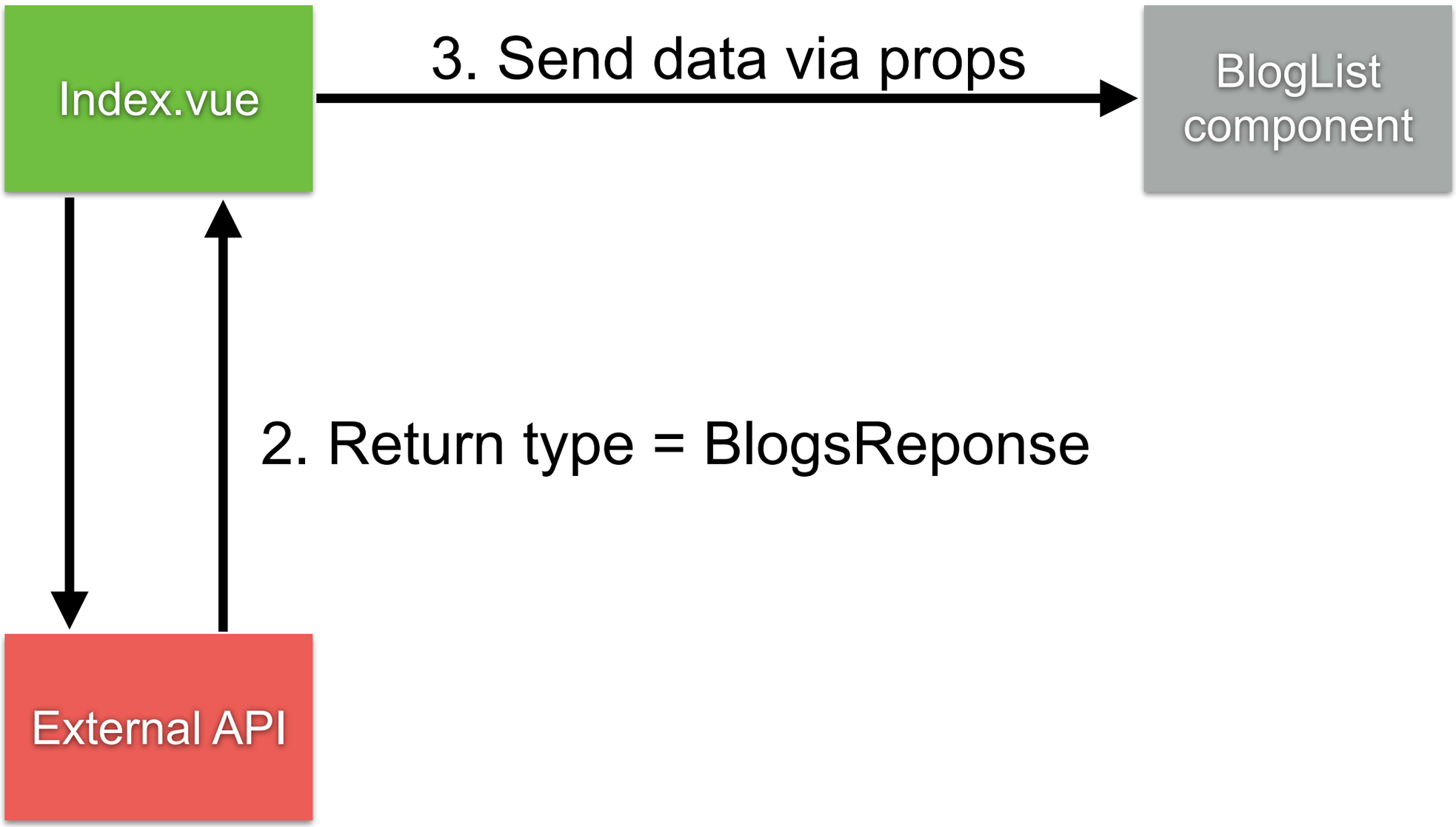
1. Get all blogs from API



2. Return type = BlogsReponse

Send data to BlogList component

1. Get all blogs from API



index.vue

```vue
<script setup lang="ts">
// List all blogs
import { API_BASE_URL } from "~/constants";
import type { BlogsResponse } from '~/lib/api/blog';
const { data: blogList2, pending: blogList2Pending,
                         error: blogList2Error } =
  useFetch<BlogsResponse>(`${API_BASE_URL}/api/blogs`, {
    method: 'GET',
  });
</script>
```

Pending

Success

Error

Pending or loading data

```
<div class="md:w-3/4 mr-4”>
  <p v-if="blogList2Pending">Loading blogs…</p>
  <div v-else>
    <p v-if="
      blogList2 &&
      blogList2.body &&
      blogList2.body.length === 0
    ">
      No blogs are here... yet.
    </p>

    <p v-if="
      blogList2Error
    ">
      Error to loading blog.
    </p>

    <div v-else-if="blogList2 && blogList2.body">
      <BlogList :blogs="blogList2.body" />
    </div>

  </div>
</div>
```

Success

```
<div class="md:w-3/4 mr-4”>

  <p v-if="blogList2Pending">Loading blogs…</p>

  <div v-else>
    <p v-if="
      blogList2 &&
      blogList2.body &&
      blogList2.body.length === 0
    ">
      No blogs are here... yet.
    </p>

    <p v-if="
      blogList2Error
    ">
      Error to loading blog.
    </p>

    <div v-else-if="blogList2 && blogList2.body">
      <BlogList :blogs="blogList2.body" />
    </div>

  </div>
</div>
```

Error

```
<div class="md:w-3/4 mr-4”>

  <p v-if="blogList2Pending">Loading blogs…</p>

  <div v-else>
    <p v-if="
      blogList2 &&
      blogList2.body &&
      blogList2.body.length === 0
    ">
      No blogs are here... yet.
    </p>

    <p v-if="
      blogList2Error
    ">
      Error to loading blog.
    </p>

    <div v-else-if="blogList2 && blogList2.body">
      <BlogList :blogs="blogList2.body" />
    </div>

  </div>
</div>
```

BlogsList component

/components/BlogsList.vue

```vue
<script setup lang="ts">

import type { Blog } from '~/types';
const props = defineProps<{
    blogs: Blog[];
}>();

</script>
```

```vue
<div v-for="blog in props.blogs"
     :key="blog.id" >

          {{ blog.username }}
<div/>
```

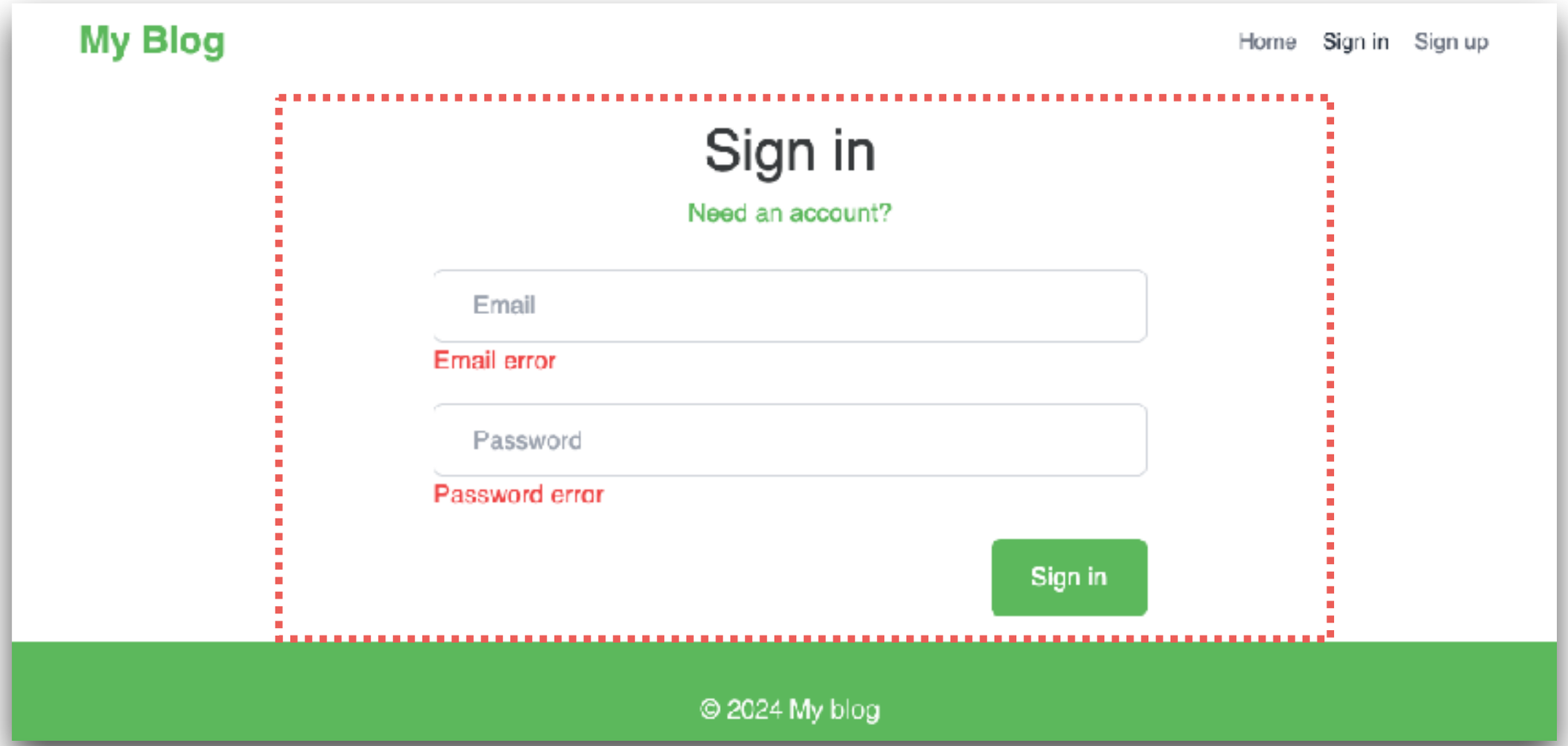
Create more pages

```
WARN  [Vue Router warn]: No match found for location with path "/user/login"

WARN  [Vue Router warn]: No match found for location with path "/user/register"
```

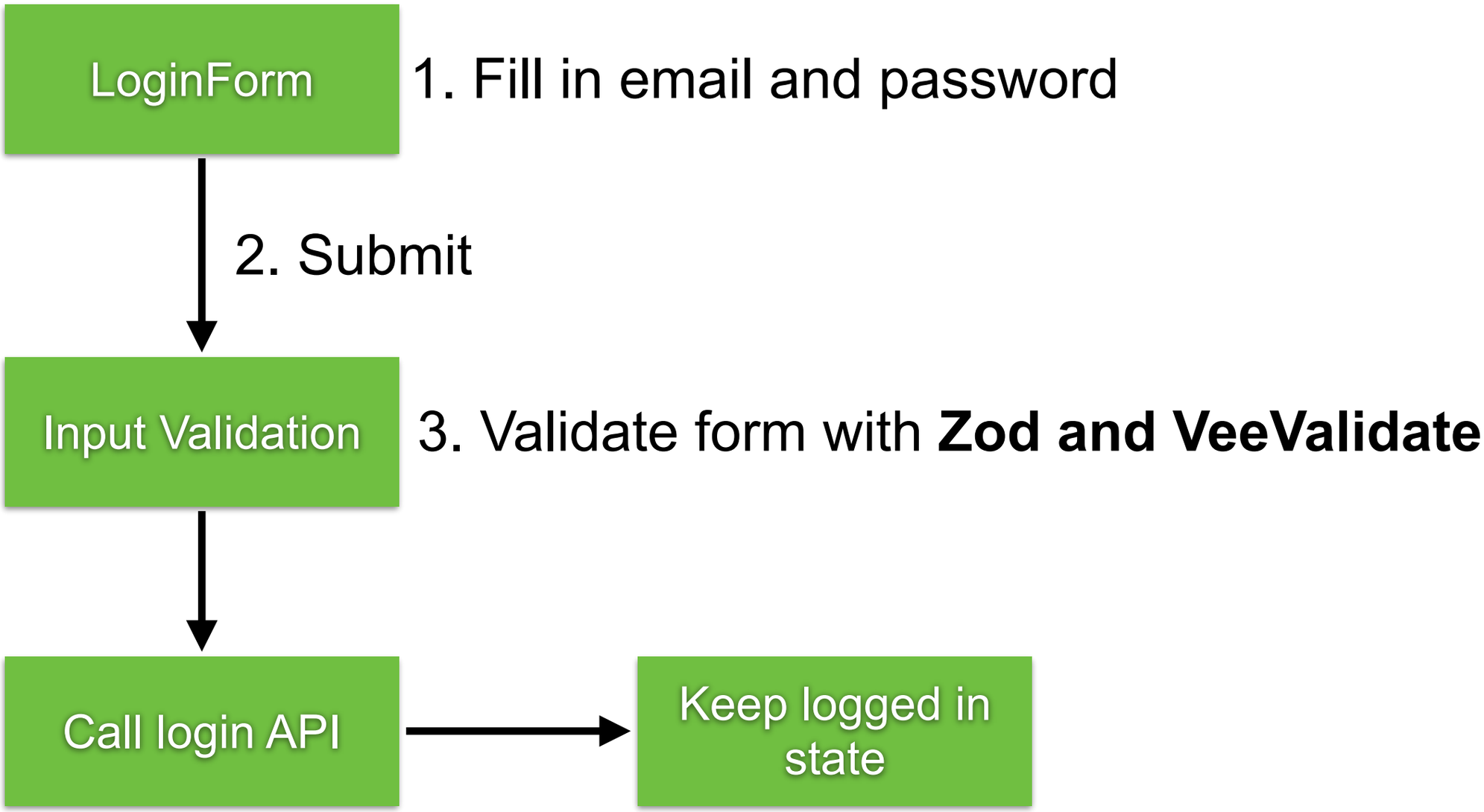
Login page

Page = /user/login.vue

Component = LoginForm.vue



Login flow



Use VueX or **Pinia**

Schema validation

- Zod
- Yup
- Joi
- Valibot

Zod :: Schema validation

Zod

✨ https://zod.dev ✨

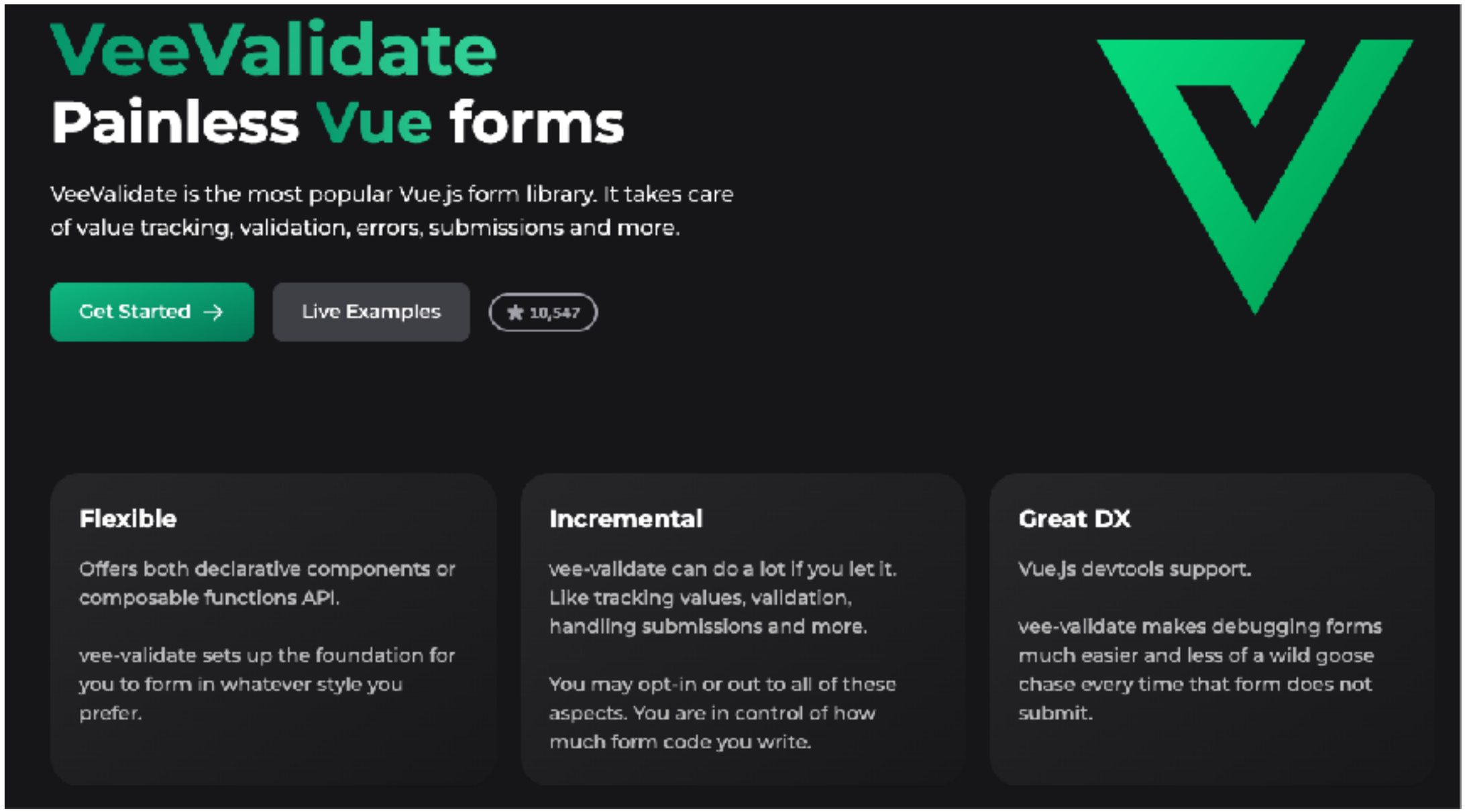
TypeScript-first schema validation with static type inference

test passing | created by @colinhacks | license MIT | downloads 8.9M/week | Stars 30k | Discord 290 online

Documentation • Discord • npm • deno • Issues • @colinhacks • tRPC

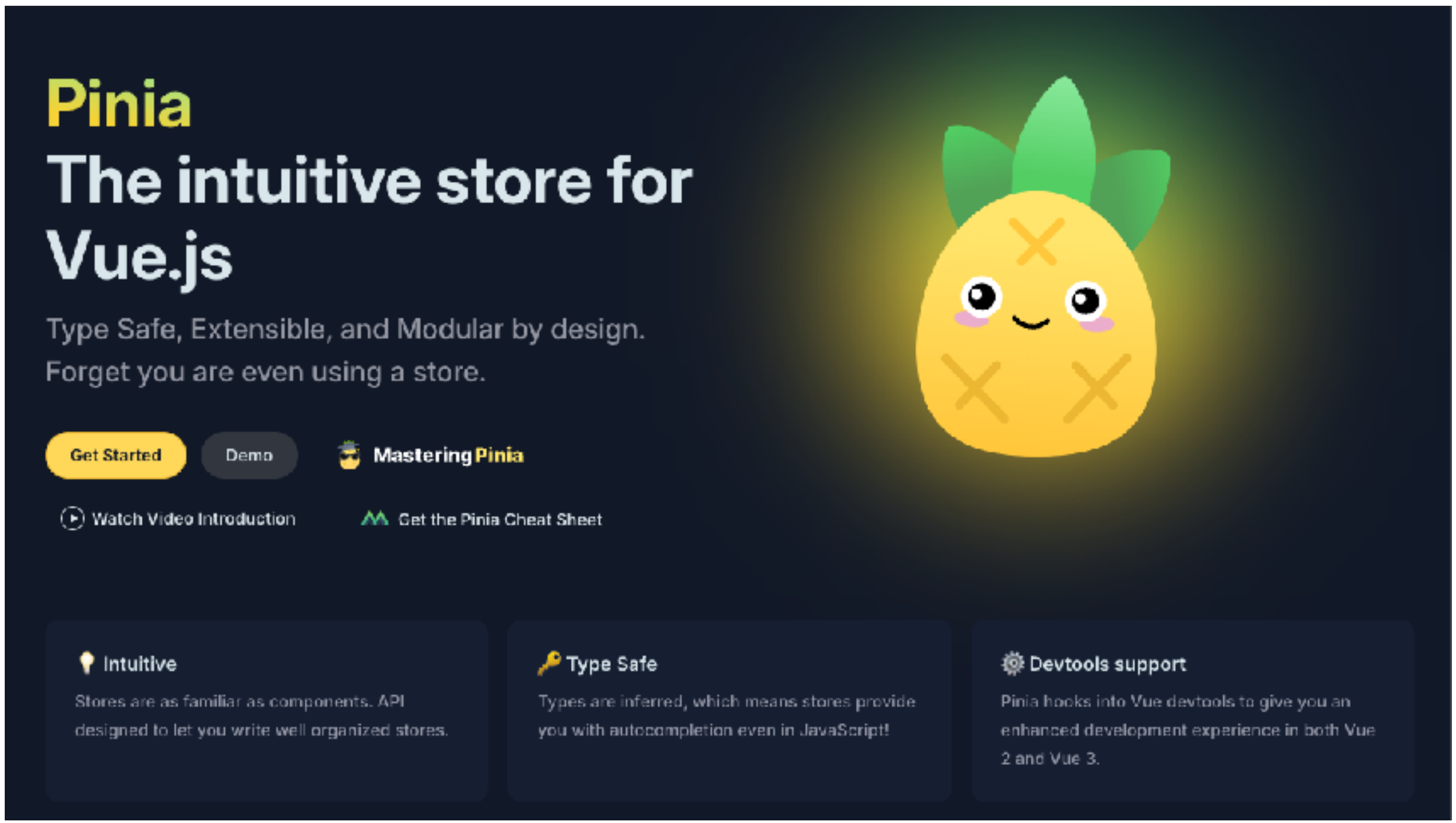
https://zod.dev/

VeeValidate



https://vee-validate.logaretm.com/v4/

State management



https://pinia.vuejs.org/

Login Page

https://github.com/up1/demo-nuxt-blog/wiki/login-page

1. Login Page

/pages/user/login.vue

```vue
<template>
  <div>
    <Head>
      <title>Login</title>
    </Head>

    <MyContainer>
      <div class="w-full md:w-1/2 md:mx-auto">
        <h1 class="text-center text-4xl font-medium mb-2">Sign in</h1>
        <p class="text-center mb-6">
          <NuxtLink
            href="/user/register"
            class="text-custom-green no-underline hover:text-green-600 hover:underline"
          >
            Need an account?
          </NuxtLink>
        </p>
        <LoginForm />
      </div>
      <div class="my-4"></div>
    </MyContainer>
  </div>
</template>
```

2. LoginForm

/components/LoginForm.vue

```
<template>
    <form>
        <fieldset class="space-y-4">
            <div class="form-group">
                <input name="email" type="email" placeholder="Email"
                    class="w-full px-6 py-3 text-base leading-5 text-gray-600 bg-white border
border-gray-300 rounded-md" />
                <span class="text-red-500">Email error</span>
            </div>
            <div class="form-group">
                <input name="password" type="password" placeholder="Password"
                    class="w-full px-6 py-3 text-base leading-5 text-gray-600 bg-white border
border-gray-300 rounded-md" />
                <span class="text-red-500">Password error</span>
            </div>
            <button
                class="float-right px-6 py-3 text-base text-white bg-custom-green border border-
custom-green rounded-md hover:bg-green-600">
                Sign in
            </button>
        </fieldset>
    </form>
</template>
```

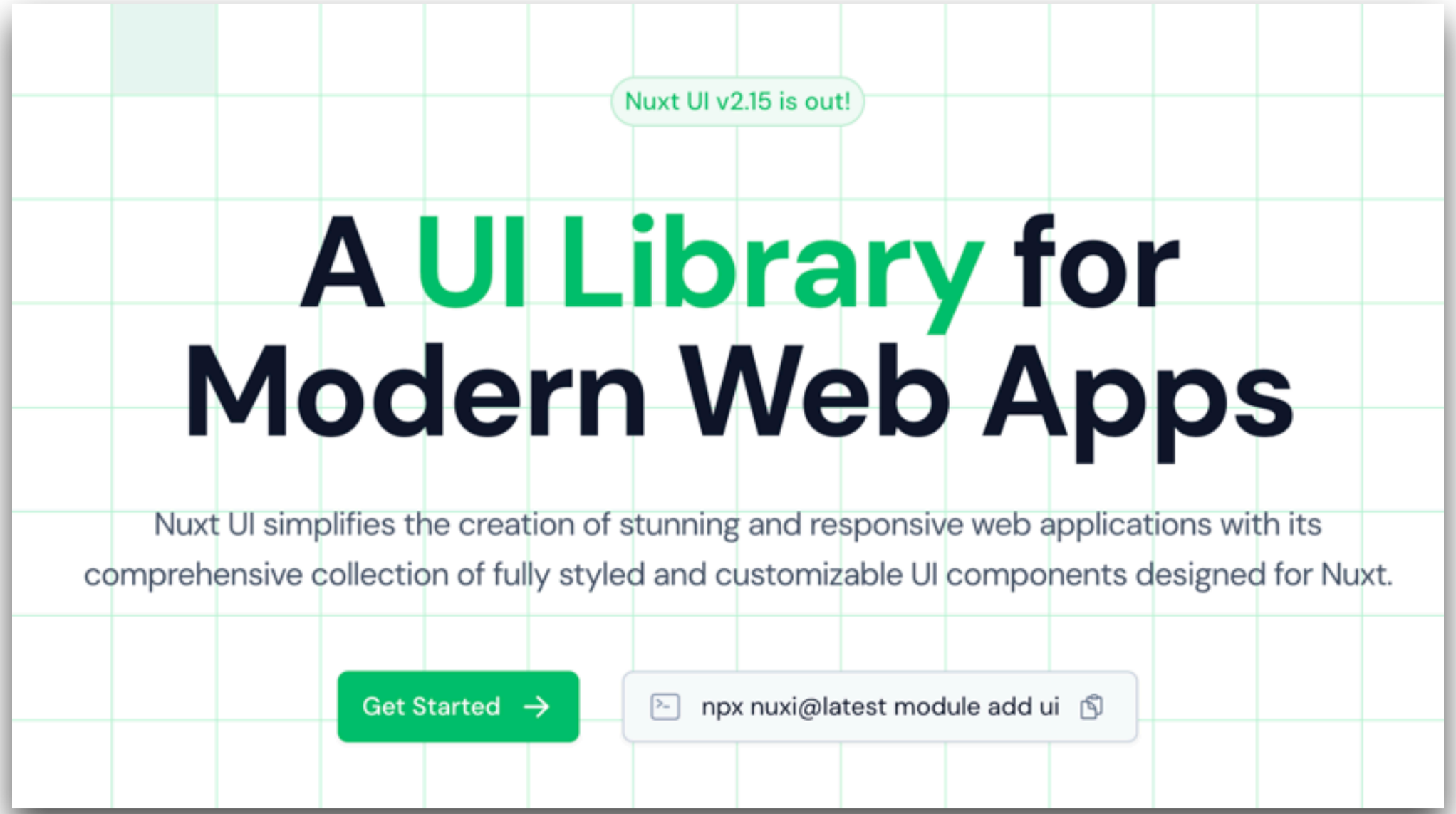
3. LoginForm

Validate login form after submitted
Install Zod and VeeValidate module

npm install vee-validate zod @vee-validate/zod

https://vee-validate.logaretm.com/v4/integrations/zod-schema-validation/

NuxtUI



https://ui.nuxt.com/components/form

4. Validate (1)

Use **Zod** to validate data

```
import * as zod from 'zod';
import { useField, useForm } from 'vee-validate';
import { toTypedSchema } from '@vee-validate/zod';

const isSubmitting = ref(false);

const validationSchema = toTypedSchema(
    zod.object({
        email: zod
            .string()
            .min(1, 'This is required')
            .email({ message: 'Must be a valid email' }),
        password: zod
            .string()
            .min(1, 'This is required')
            .min(6, { message: 'Too short' }),
    })
);

const { handleSubmit, errors } = useForm({
    validationSchema,
});
```

4. Validate (2)

Use **VeeValidate** to mapping with Input Form

```
const { value: email } = useField('email');
const { value: password } = useField(‘password');
```

```
<template>

<input v-model="email" name="email" type="email" placeholder="Email"
<input v-model="password" name="password" type="password" placeholder="Password"

<template>
```

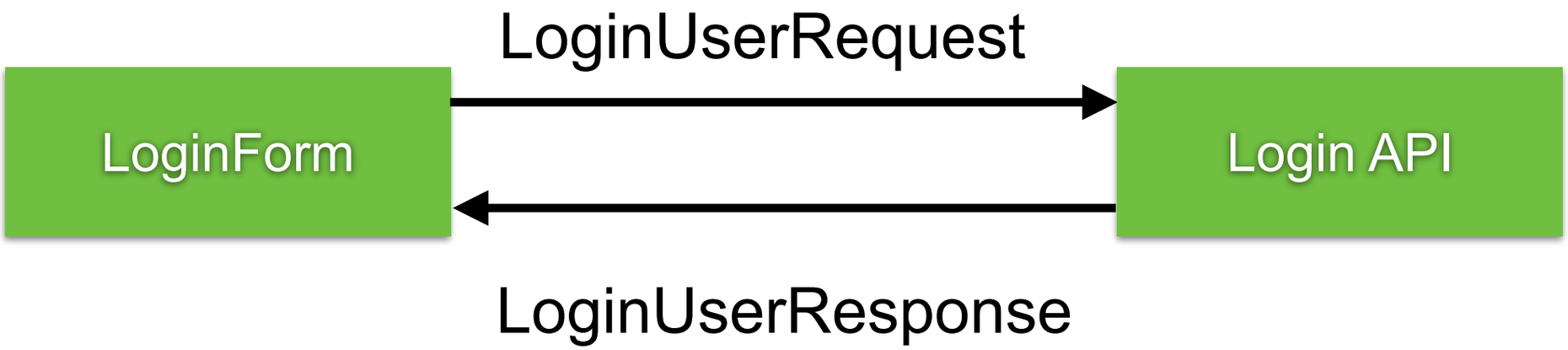
5. Submit Form

```
const onSubmit = handleSubmit(async (values) => {
    isSubmitting.value = true;

    try {
        // Call Login api
        // Keep logged in user in auth store
        // Redirect to home page
        await navigateTo('/');
    } catch (error) {
        alert(error);
    } finally {
        isSubmitting.value = false;
    }
});
```

```
<form @submit="onSubmit">

</form>
```

6. Call Login API

Create file /lib/api/auth.ts



6. Call Login API

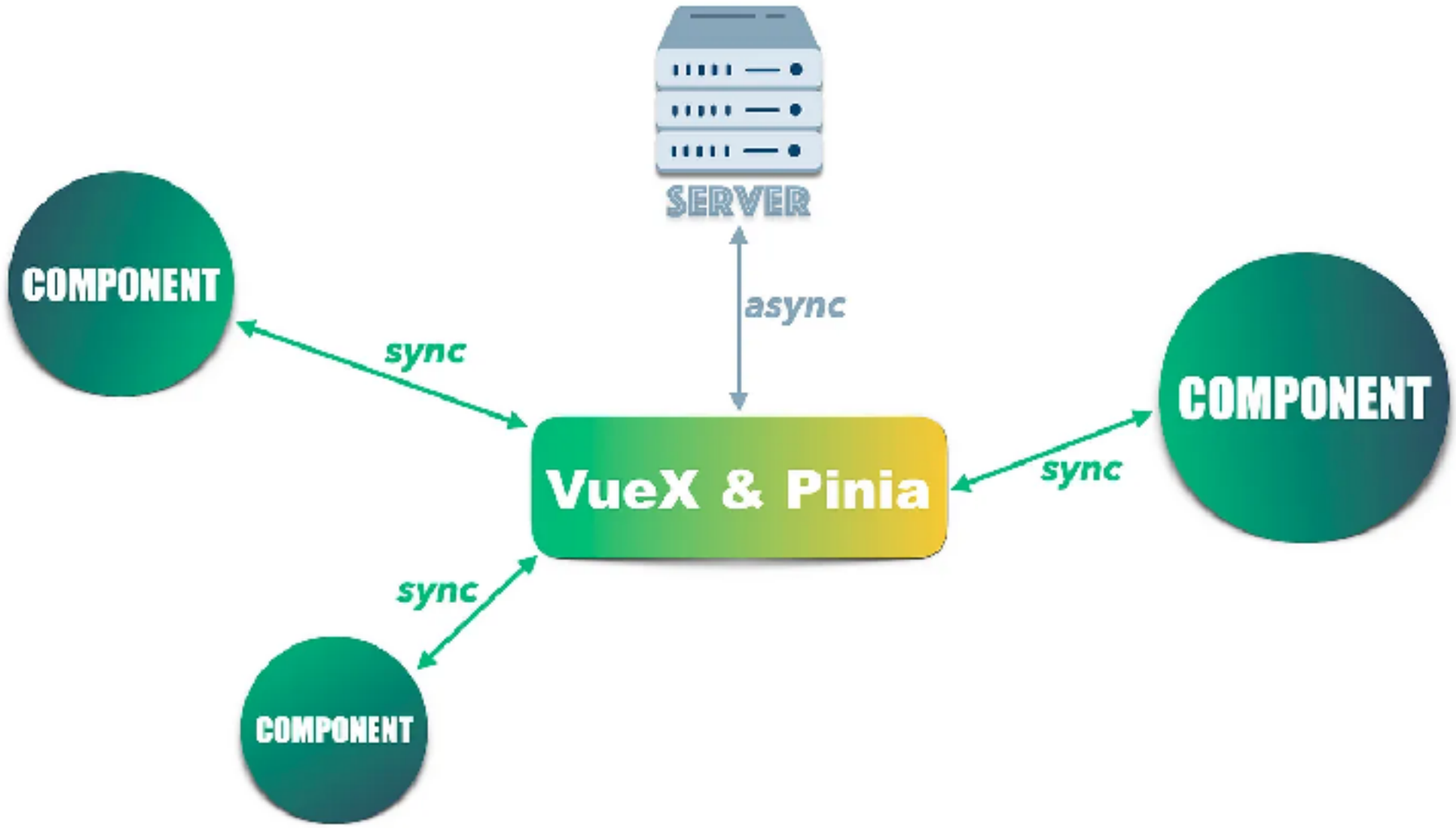
Create file /lib/api/auth.ts

```ts
import { API_BASE_URL } from '~/constants';
import type { paths } from '~/openapi.gen';

type LoginUserRequest =
  paths['/users/login']['post']['requestBody']['content']['application/json'];
type LoginUserResponse =
  paths['/users/login']['post']['responses']['200']['content']['application/json'];

export const login = (user: LoginUserRequest) => {
  return $fetch<LoginUserResponse>(`${API_BASE_URL}/api/users/login`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: user,
  });
};
```

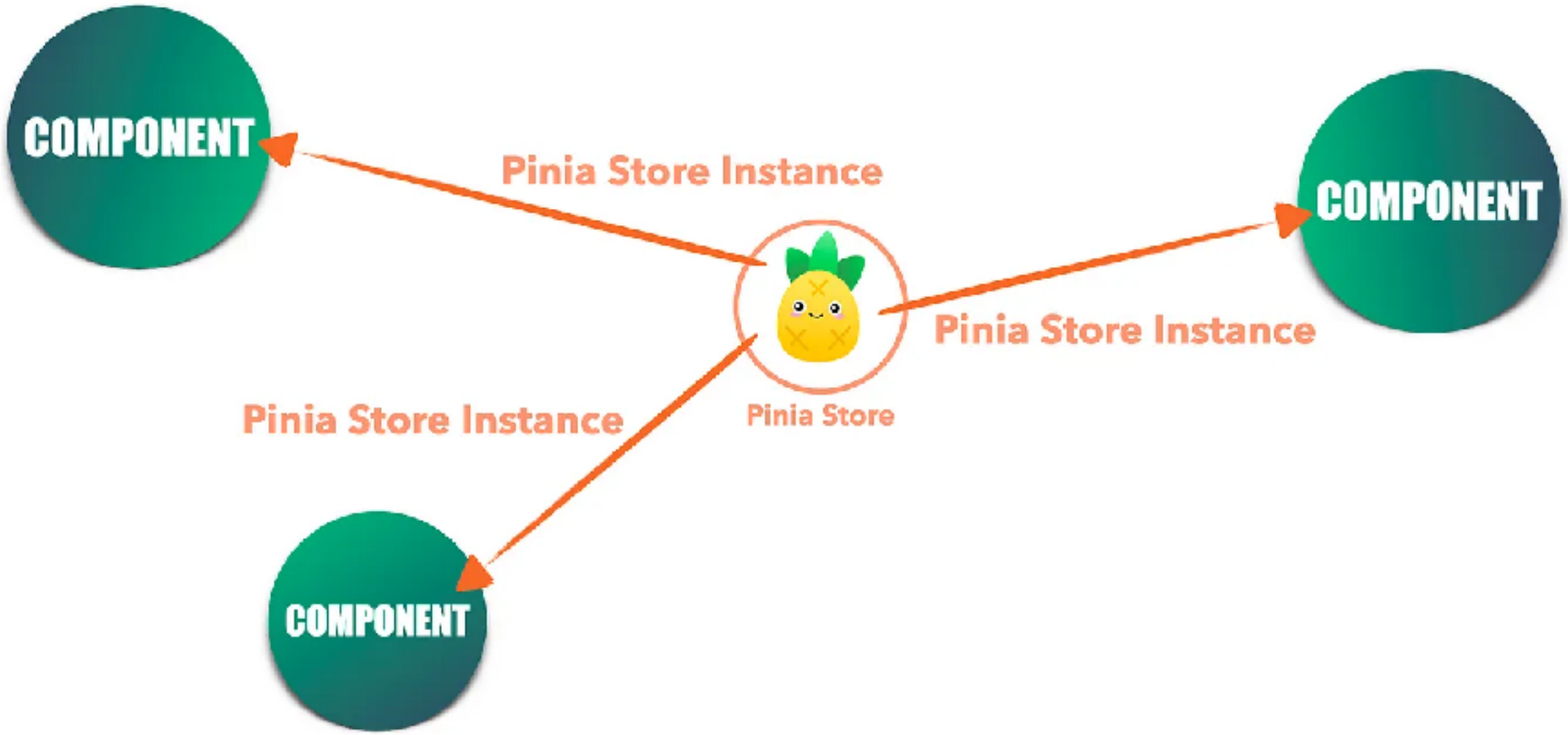
Call API from LoginForm

```
import { login } from '~/lib/api/auth';

const onSubmit = handleSubmit(async (values) => {
    isSubmitting.value = true;

    try {
        // Call Login api
        const response = await login({
            email: values.email,
            password: values.password,
        });
        console.log(response);
        // Keep logged in user in auth store
        // Redirect to home page
        await navigateTo('/');
    } catch (error) {
        alert(error);
    } finally {
        isSubmitting.value = false;
    }
});
```

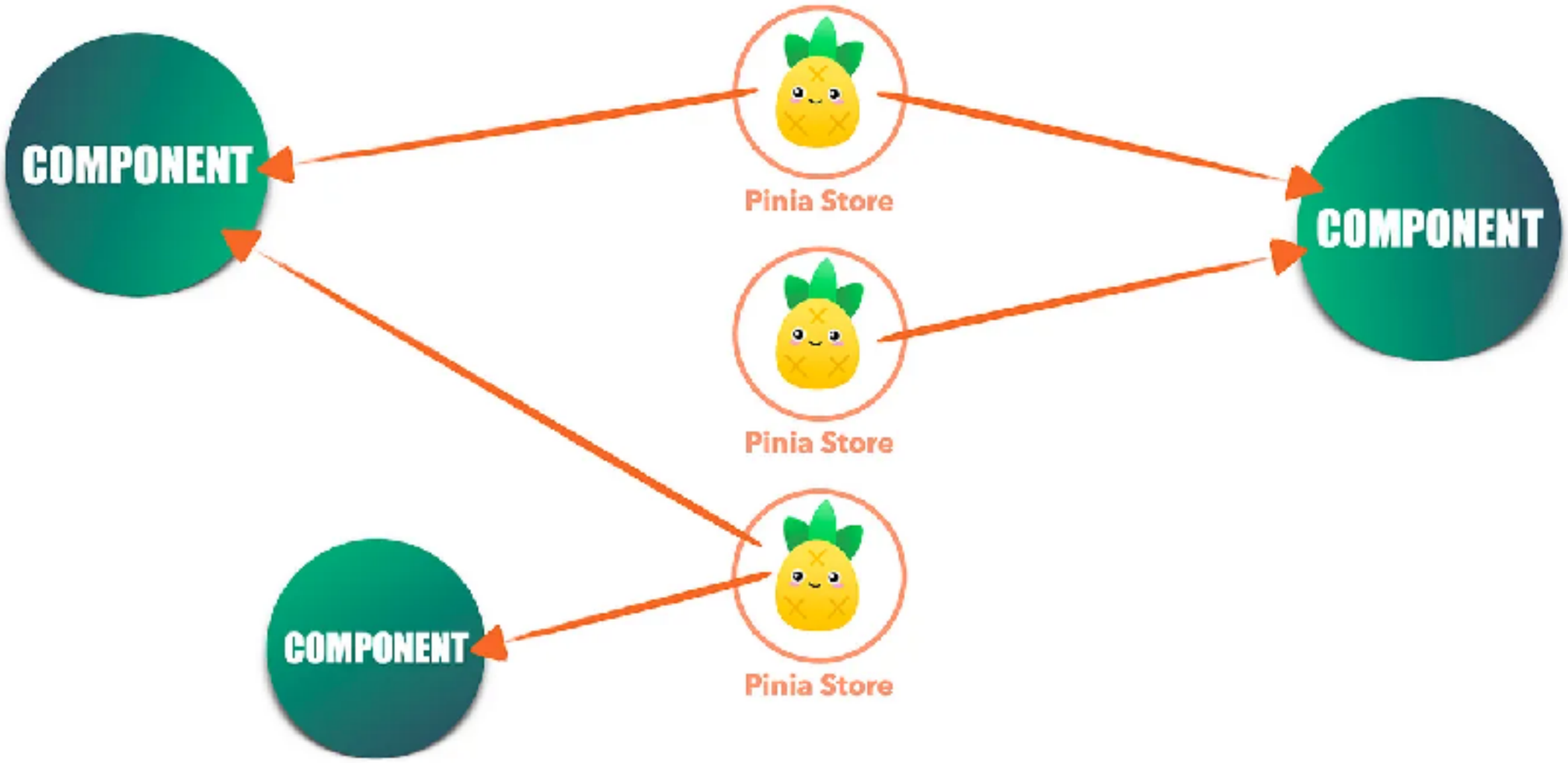
State management



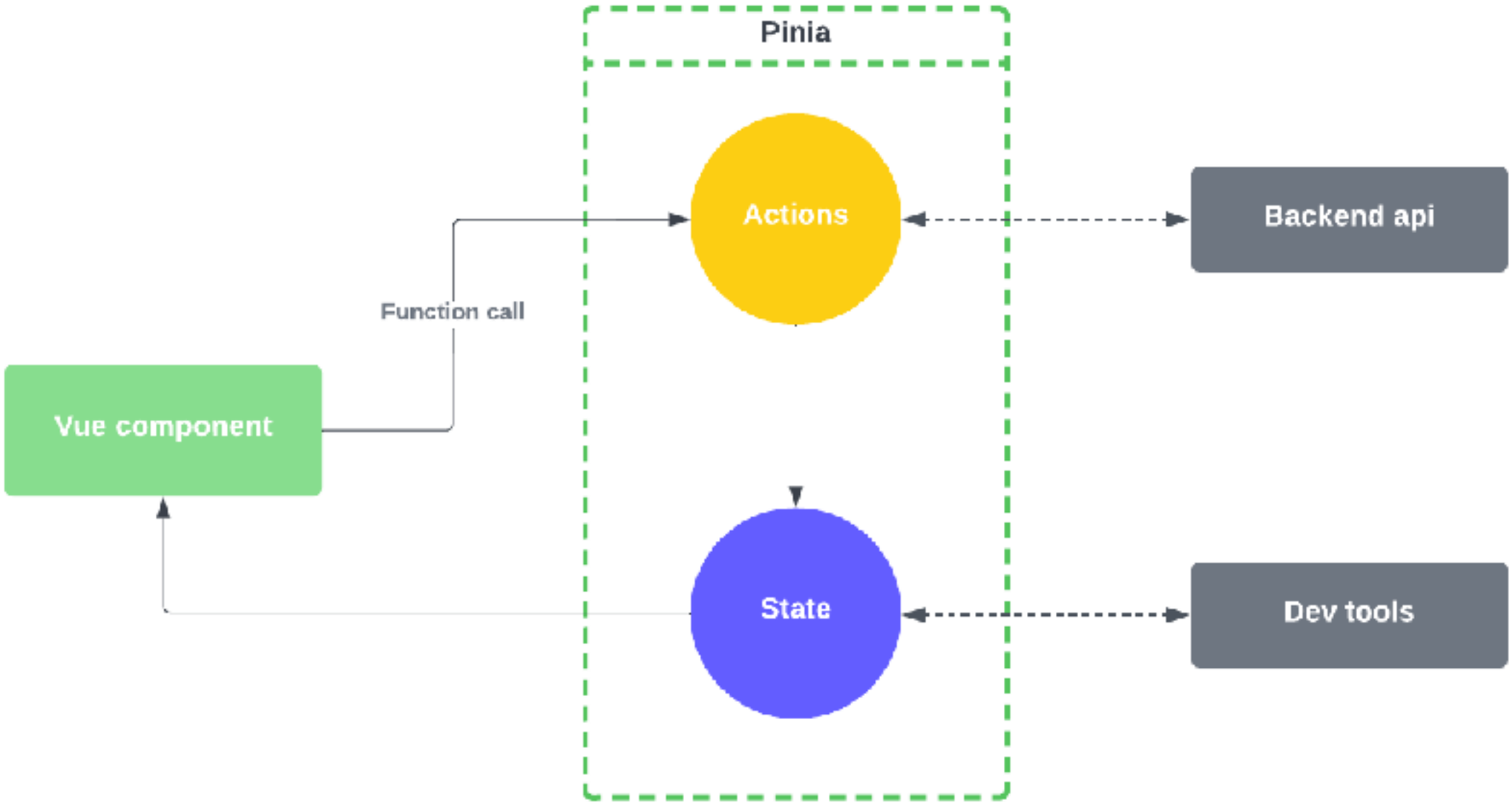
State management with Pinia



State management with Pinia

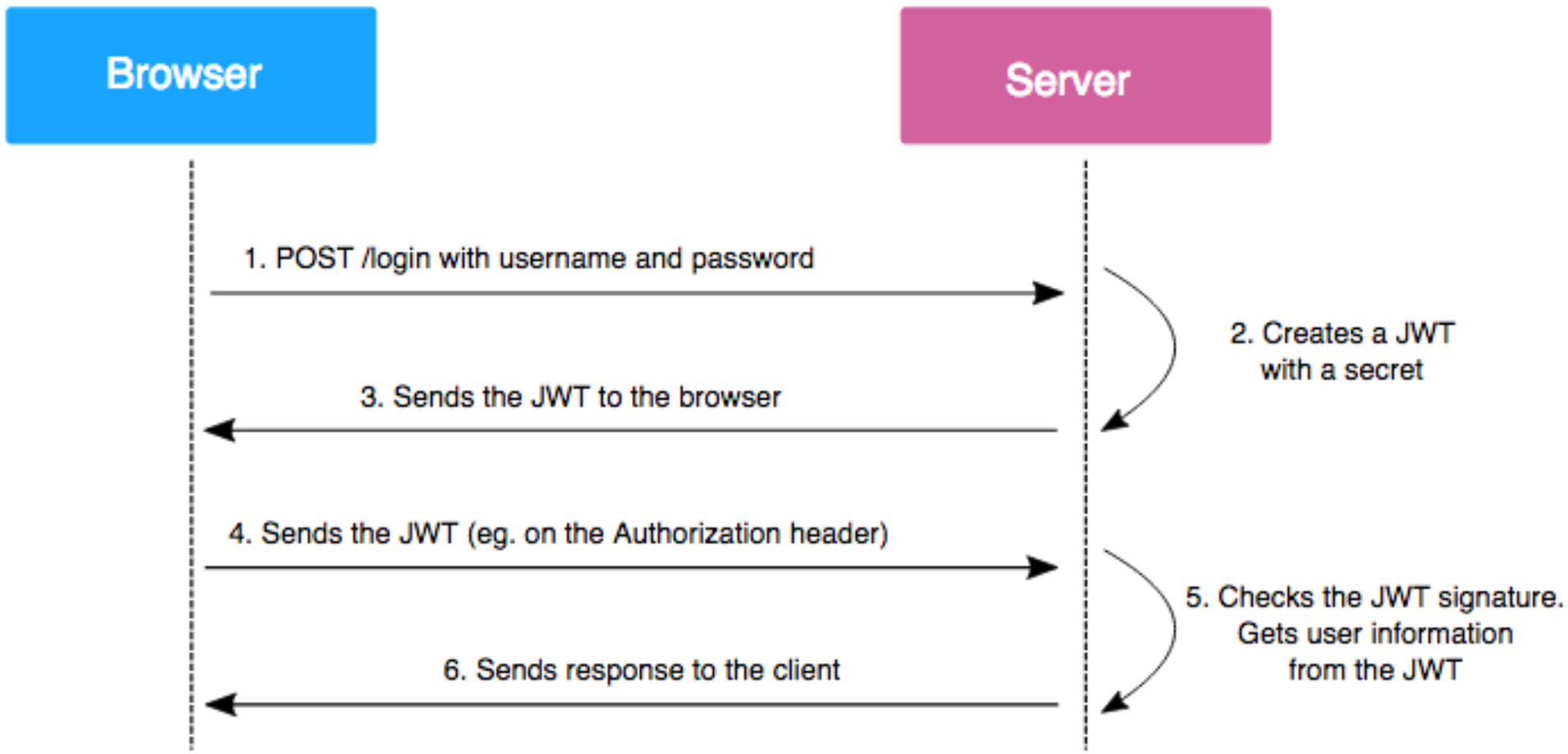


State management with Pinia

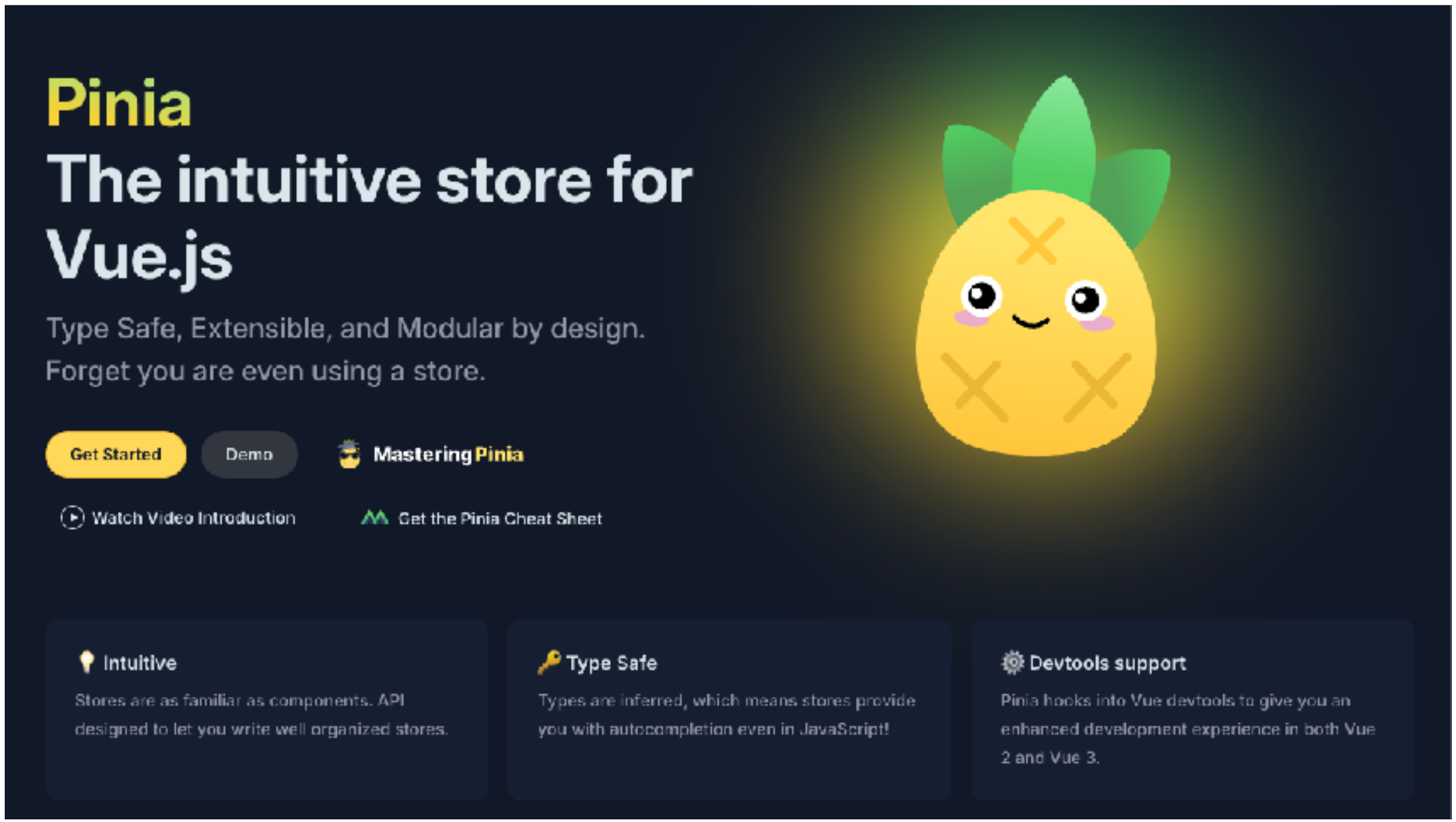


7. State management with Pinia

Keep JWT Token from API in web browser

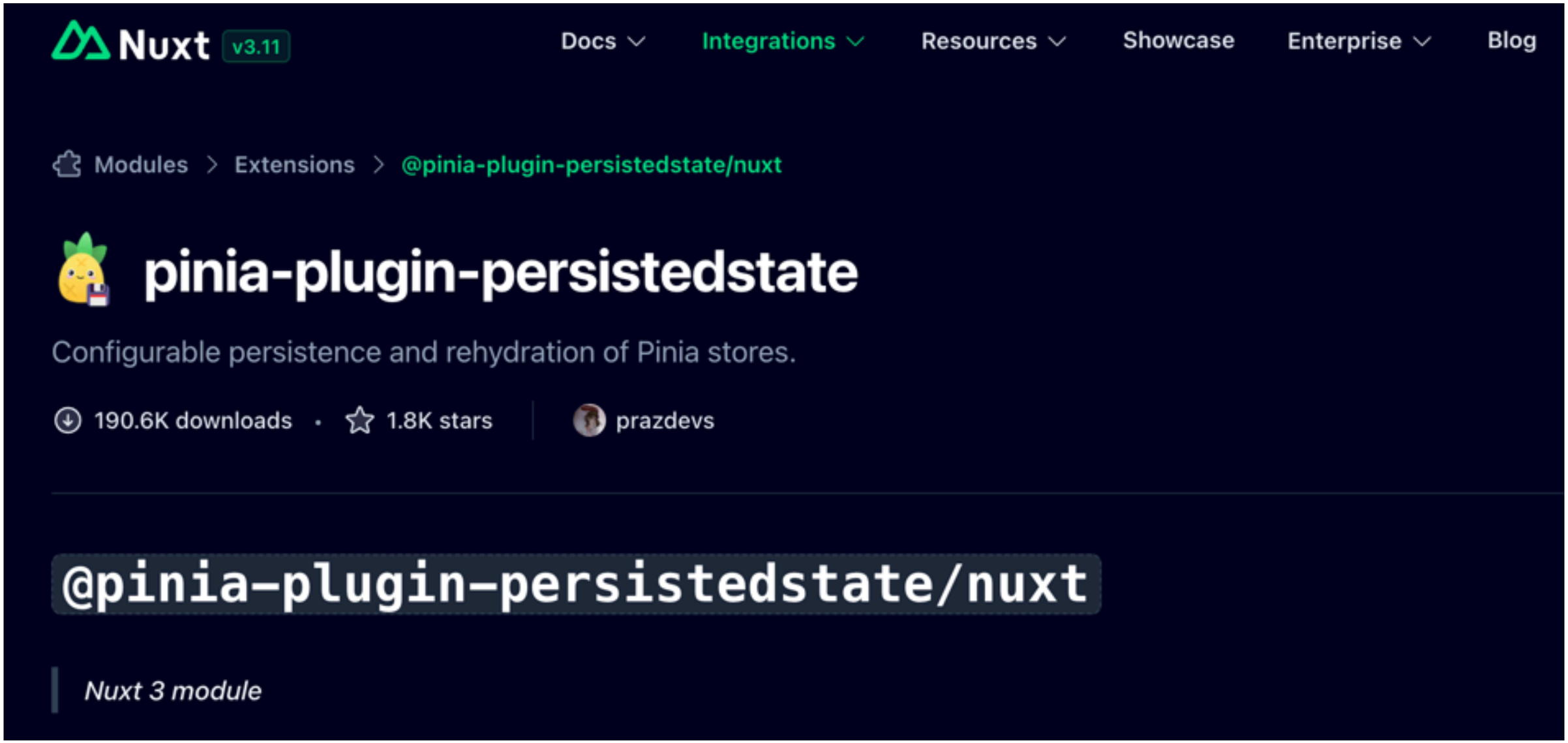


Pinia with Nuxt



https://pinia.vuejs.org/ssr/nuxt.html

Pinia Persist State



https://nuxt.com/modules/pinia-plugin-persistedstate

Pinia Persist State

Cookie
By default

LocalStorage

SessionStorage

Cookie
With Options

https://prazdevs.github.io/pinia-plugin-persistedstate/frameworks/nuxt-3.html

Install Pinia

npm install pinia @pinia/nuxt
npm i -D @pinia-plugin-persistedstate/nuxt

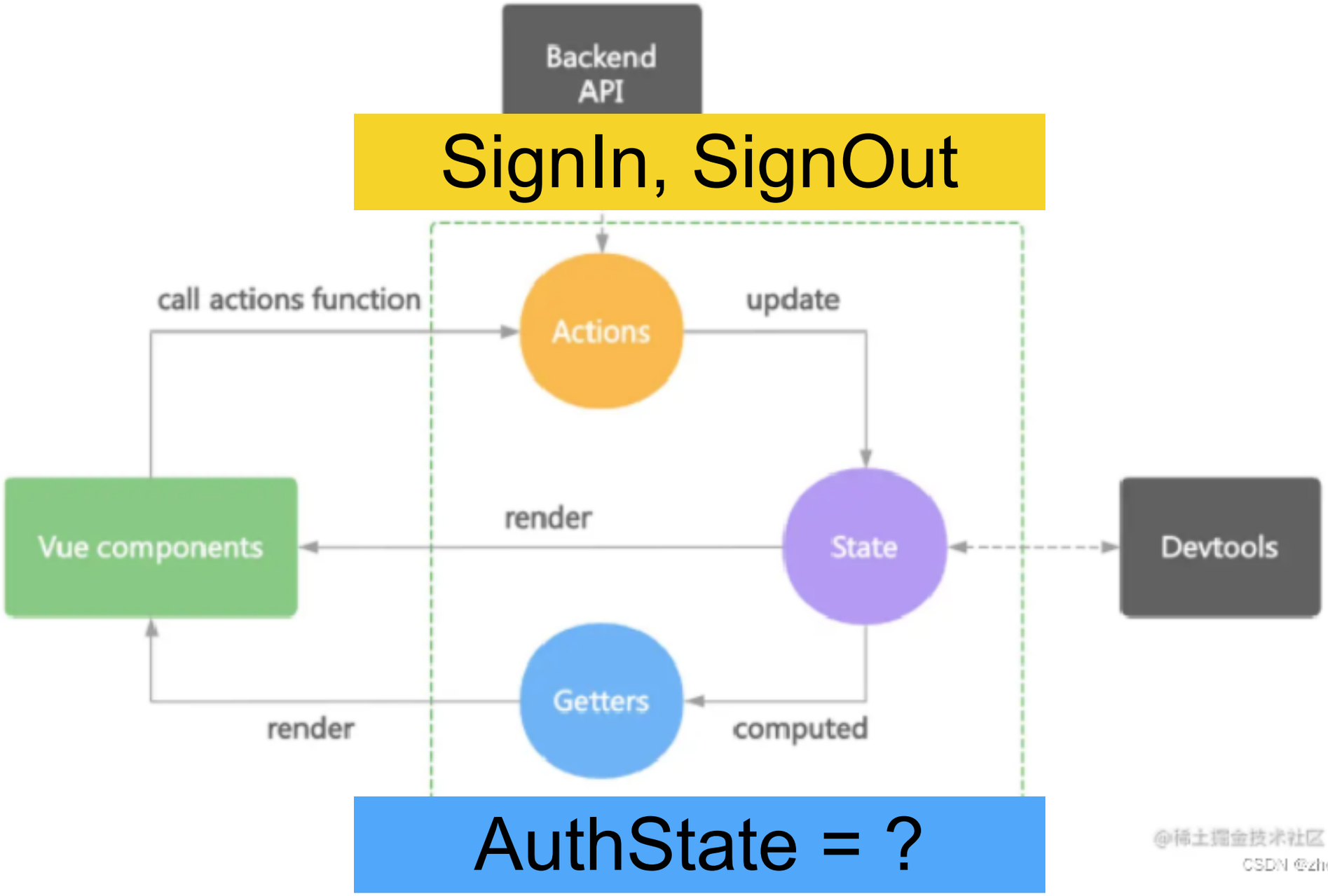
Import modules

Edit file next.config.ts

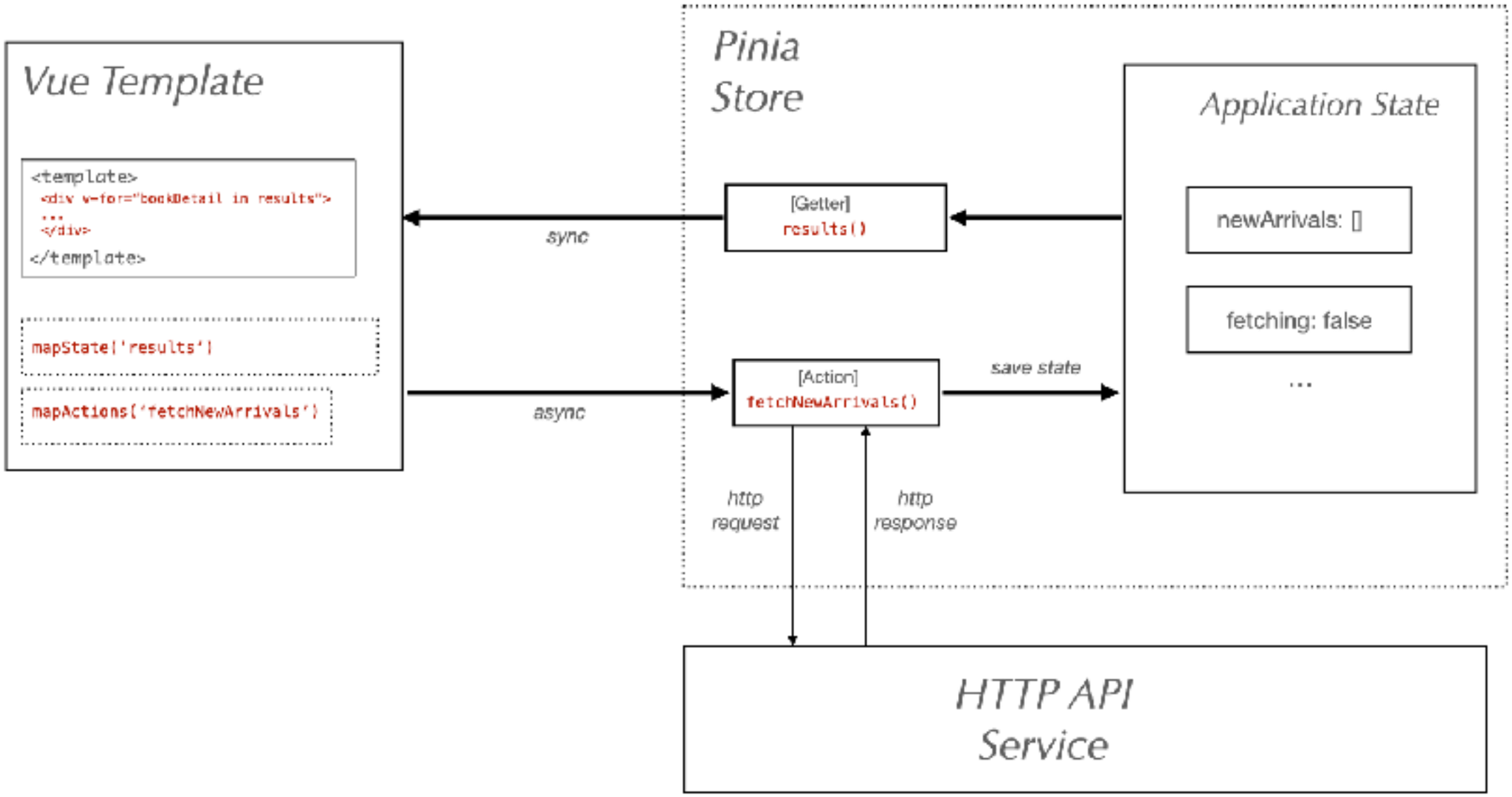
```ts
export default defineNuxtConfig({
  devtools: { enabled: true },
  typescript: { strict: true },
  modules: [
    '@nuxtjs/tailwindcss',
    '@pinia/nuxt',
    '@pinia-plugin-persistedstate/nuxt',
  ],
  css: ['~/assets/css/global.css'],
})
```

Auth Store to keep JWT Token

State management with Pinia



State management with Pinia



Auth Store to keep JWT Token

/stores/auth.ts

Declare states

```ts
import { defineStore } from 'pinia';
import type { User } from '@/types';

type authState = {
    currentUser: User | undefined;
};

const defaultState: authState = {
    currentUser: undefined,
};
```

Auth Store to keep JWT Token

Create Auth Store

```
export const authStore = defineStore({
    id: 'auth',
    state: () => defaultState,
    getters: {
        isAuthenticated: (state) => state.currentUser !== undefined,
        jwtToken: (state) => state.currentUser?.token,
    },
    actions: {
        signIn(user: User) {
            this.currentUser = user;
        },
        signOut() {
            this.currentUser = undefined;
        },
    },
    persist: true,
});
```

Auth Store to keep JWT Token

Create Getter function of store

```
export const authStore = defineStore({
    id: 'auth',
    state: () => defaultState,
    getters: {
        isAuthenticated: (state) => state.currentUser !== undefined,
        jwtToken: (state) => state.currentUser?.token,
    },
    actions: {
        signIn(user: User) {
            this.currentUser = user;
        },
        signOut() {
            this.currentUser = undefined;
        },
    },
    persist: true,
});
```

Auth Store to keep JWT Token

Create actions

```
export const authStore = defineStore({
    id: 'auth',
    state: () => defaultState,
    getters: {
        isAuthenticated: (state) => state.currentUser !== undefined,
        jwtToken: (state) => state.currentUser?.token,
    },
    actions: {
        signIn(user: User) {
            this.currentUser = user;
        },
        signOut() {
            this.currentUser = undefined;
        },
    },
    persist: true,
});
```

Auth Store to keep JWT Token

Persist state in Cookie

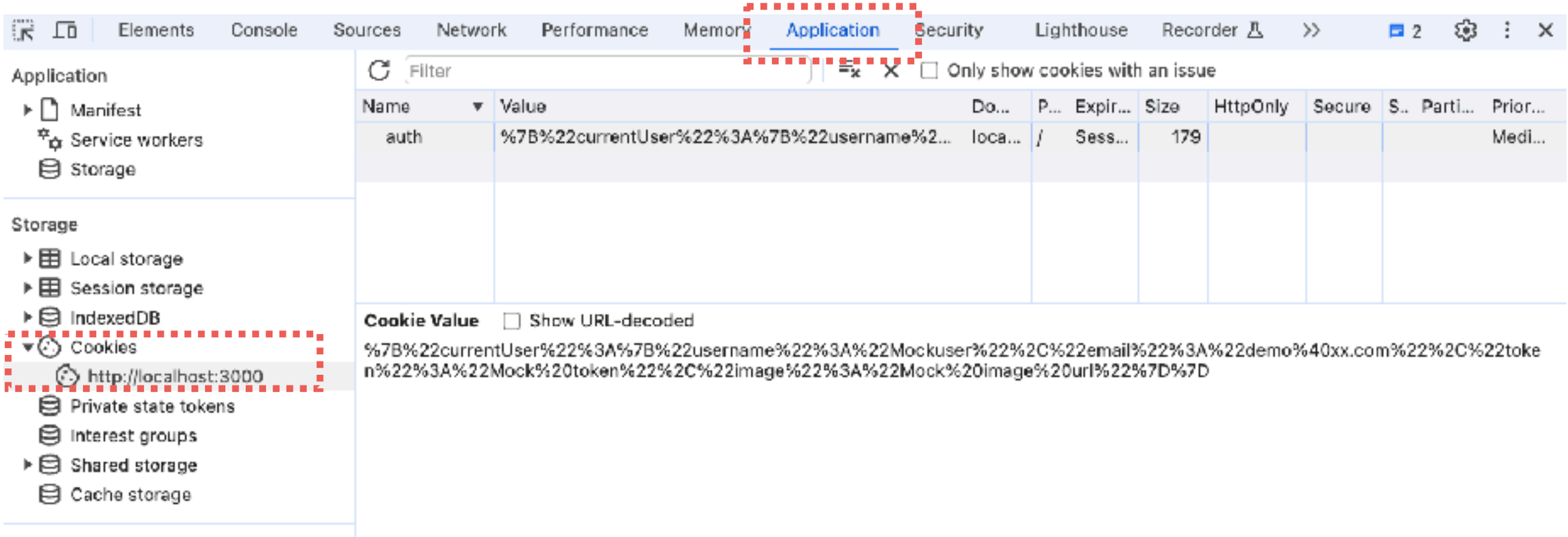
```
export const authStore = defineStore({
    id: 'auth',
    state: () => defaultState,
    getters: {
        isAuthenticated: (state) => state.currentUser !== undefined,
        jwtToken: (state) => state.currentUser?.token,
    },
    actions: {
        signIn(user: User) {
            this.currentUser = user;
        },
        signOut() {
            this.currentUser = undefined;
        },
    },
    persist: true,
});
```

8. Use state from LoginForm

/components/LoginForm.vue

```
const onSubmit = handleSubmit(async (values) => {
    isSubmitting.value = true;

    try {
        // Call Login api
        const response = await login({
            email: values.email,
            password: values.password,
        });

        // Keep logged in user in auth store
        const auth = authStore();
        auth.signIn(response.user);
...
    } catch (error) {
        alert(error);
    } finally {
        isSubmitting.value = false;
    }
});
```

Check data in web browser !!



9. Use auth state in menu

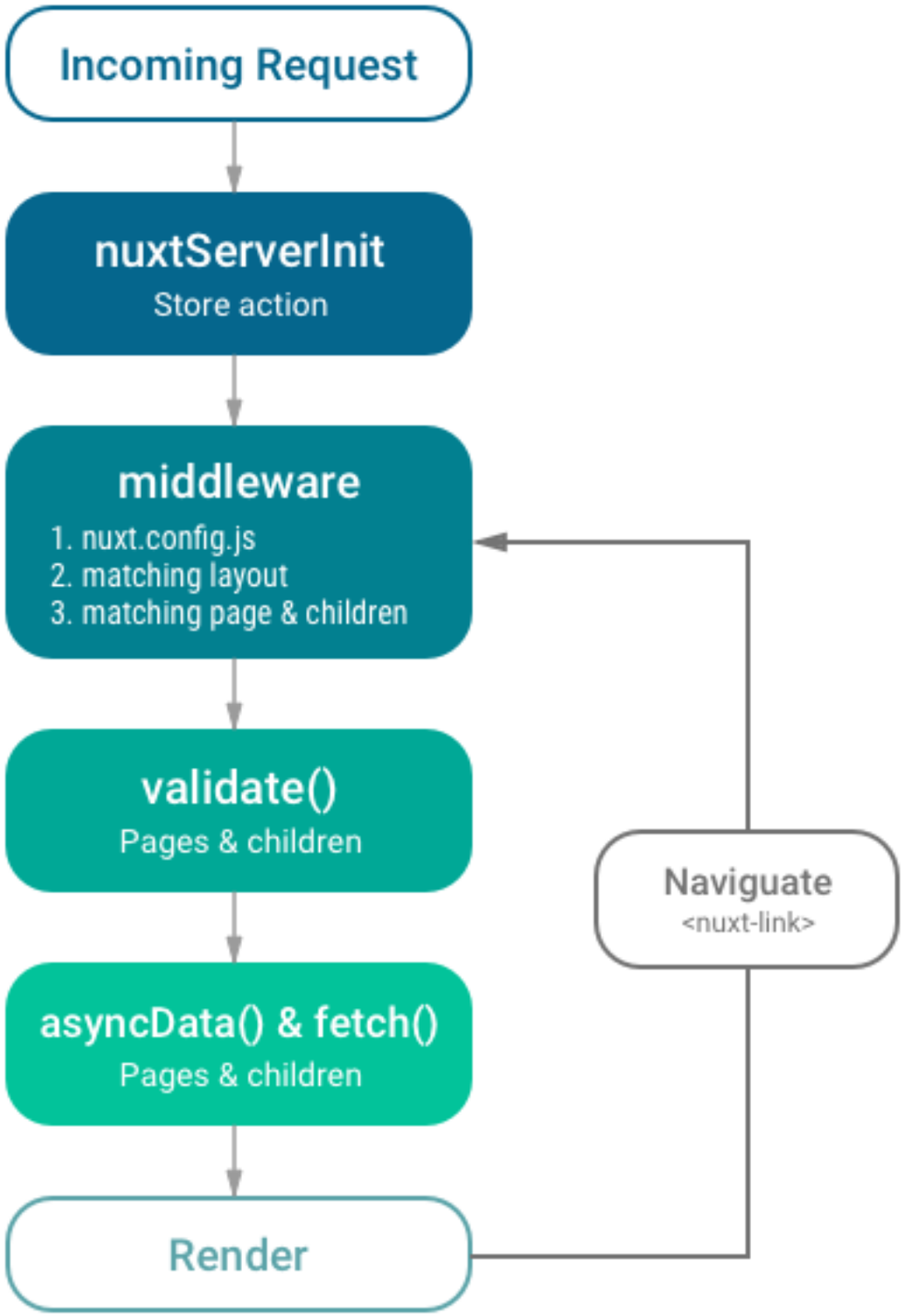
/components/MyHeader.vue

```vue
<script setup lang="ts">
import { authStore } from '~/stores/auth';
const auth = authStore();
</script>


<div class="menu">
  <MyHeaderMemberMenu v-if="auth.isAuthenticated" />
  <MyHeaderGuestMenu v-else />
</div>
```

Working with middleware

https://nuxt.com/docs/guide/directory-structure/middleware

Global middleware
Page/component



https://nuxt.com/docs/guide/directory-structure/middleware

Auth middleware

/middleware/auth.ts

```
import { authStore } from '~/stores/auth';

export default defineNuxtRouteMiddleware((to) => {
    const auth = authStore();
    console.log('Current user', auth.currentUser);
    if (!auth.isAuthenticated) {
        return navigateTo('/user/login');
    }
});
```

Use in page/component

/component/MyHeader.vue

```vue
<script setup lang="ts">
import { authStore } from '~/stores/auth';
const auth = authStore();
</script>

<template>
  <header class="top-0 bg-white z-10 p-0.5">
    <nav class="py-2 px-4">
      <div class="container mx-auto px-4 flex justify-between items-center">
        <div class="flex items-center">
          <MyHeaderAppLogo />
        </div>
        <div class="menu">
          <MyHeaderMemberMenu v-if="auth.isAuthenticated" />
          <MyHeaderGuestMenu v-else />
        </div>
      </div>
    </nav>
  </header>
</template>
```

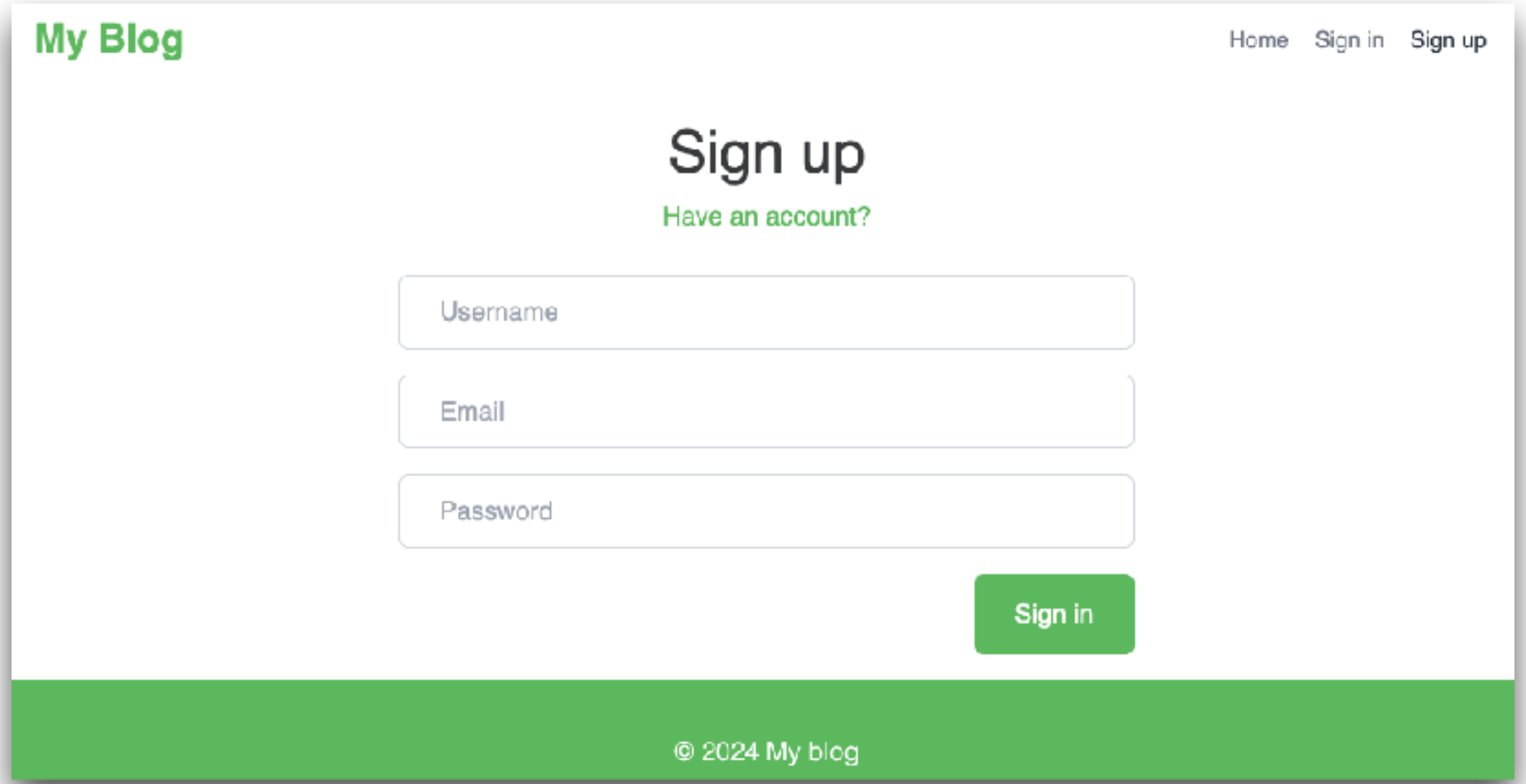
Nuxt-Auth

https://auth.nuxtjs.org/

Register Page

https://github.com/up1/demo-nuxt-blog/wiki/register-page

Register page



https://github.com/up1/demo-nuxt-blog/wiki/register-page

Testing

Testing

Unit testing

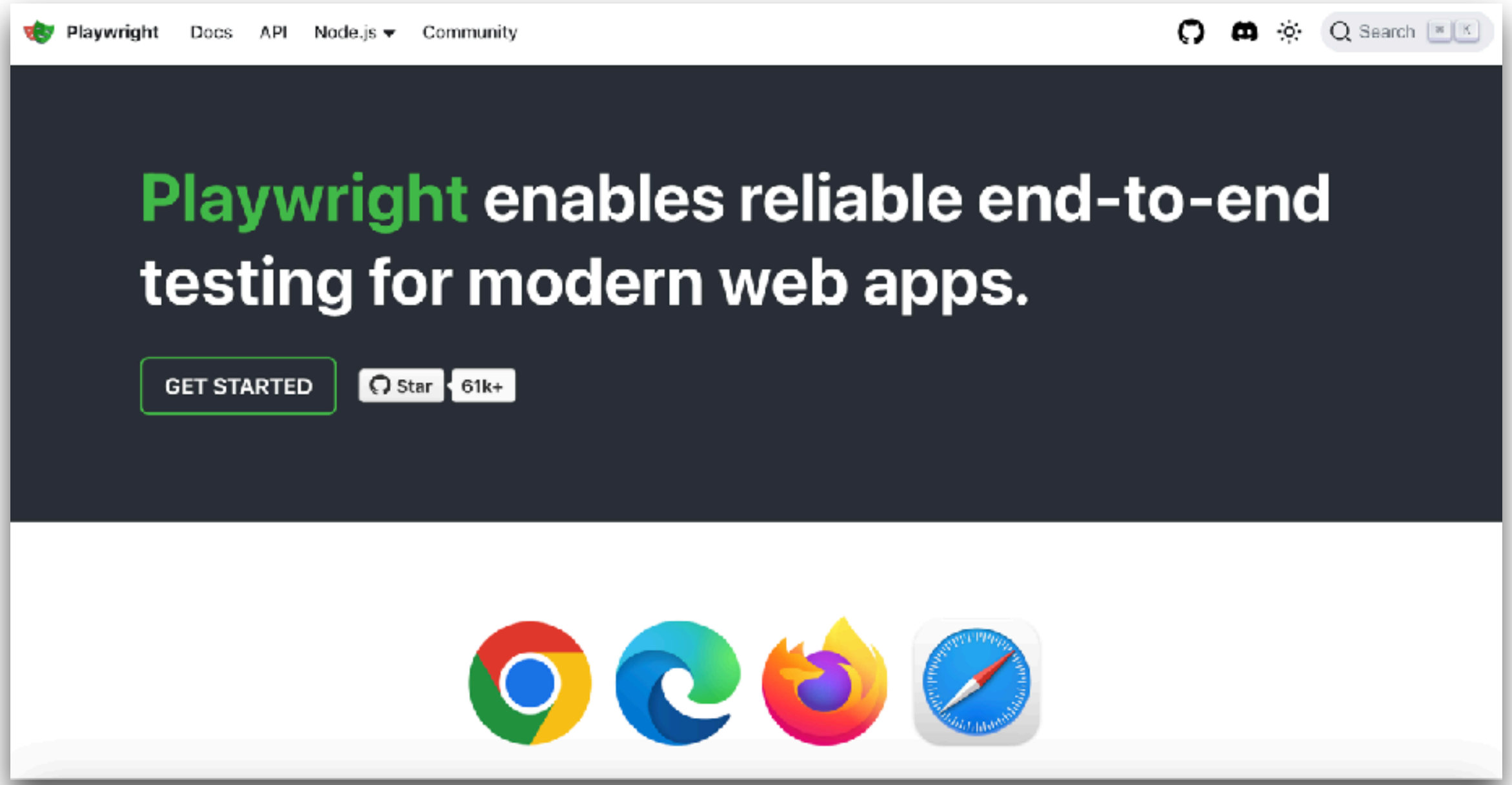
Component testing

UI testing

Screenshot testing

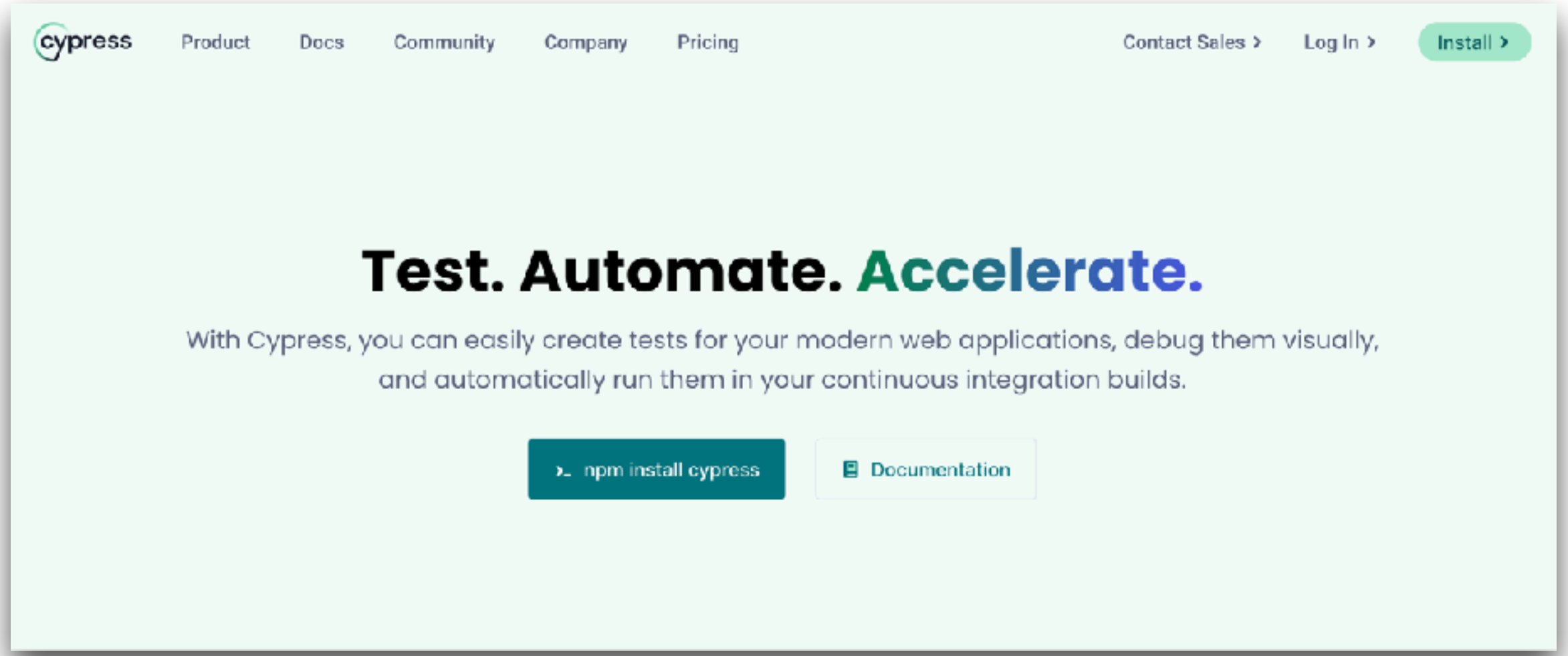
https://nuxt.com/docs/getting-started/testing

Playwright



https://playwright.dev/

Cypress



https://www.cypress.io/

Deployment

https://nuxt.com/docs/getting-started/deployment

Q/A