



# Frontend Development



**[https://github.com/up1/  
course-full-stack-development](https://github.com/up1/course-full-stack-development)**



# Frontend Development



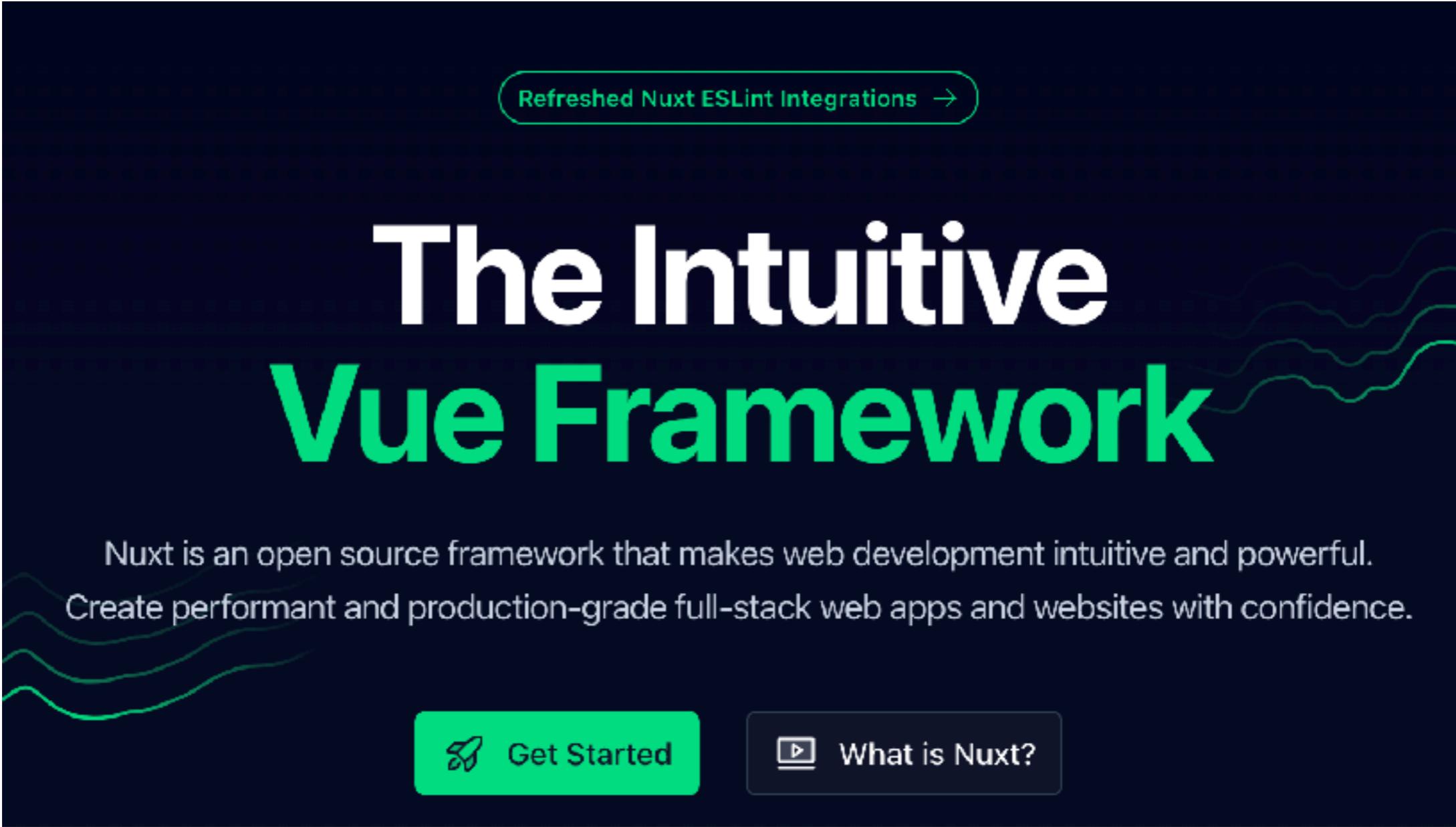
Vue.js



NUXTJS



# NuxtJS



The landing page for Nuxt.js features a dark background with green wavy lines on the left and right sides. At the top, there's a green button labeled "Refreshed Nuxt ESLint Integrations →". Below it, the main title "The Intuitive Vue Framework" is displayed in large white and green text. A subtitle below the title reads: "Nuxt is an open source framework that makes web development intuitive and powerful. Create performant and production-grade full-stack web apps and websites with confidence." At the bottom, there are two buttons: a green one with a rocket icon labeled "Get Started" and a grey one with a video camera icon labeled "What is Nuxt?".

Refreshed Nuxt ESLint Integrations →

# The Intuitive Vue Framework

Nuxt is an open source framework that makes web development intuitive and powerful.  
Create performant and production-grade full-stack web apps and websites with confidence.

Get Started

What is Nuxt?

<https://nuxt.com/>



# Learning NuxtJS

Component  
auto import

Routing

Composable

NuxtJS

VueJS



# 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

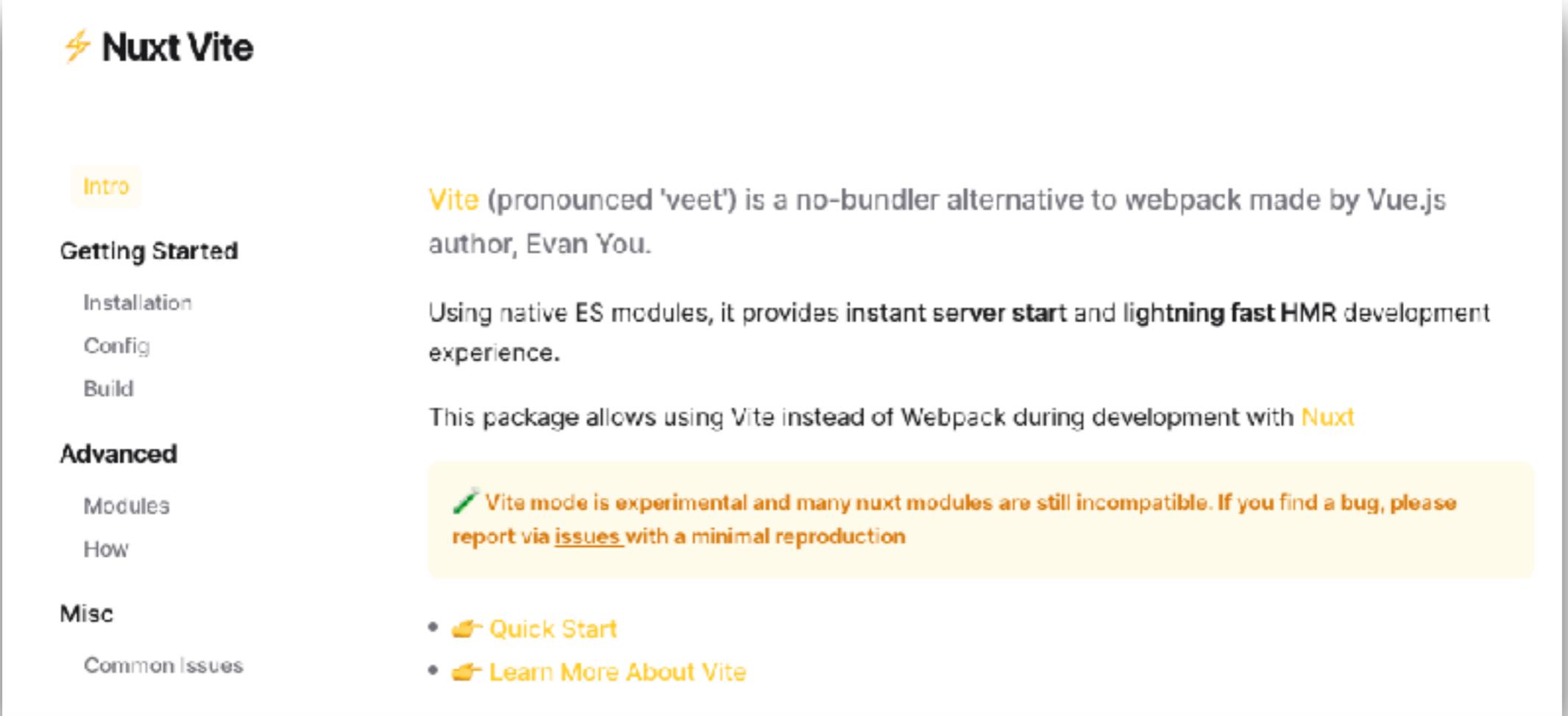
→ DevTools: press Shift + Option + D in the browser (v1.2.0)

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)



The screenshot shows the Nuxt Vite documentation page. At the top left is the Nuxt logo (a stylized orange 'N') followed by 'Nuxt Vite'. Below the header, there's a sidebar with sections for 'Getting Started' (Installation, Config, Build) and 'Advanced' (Modules, How). Under 'Misc', there's a link to 'Common Issues'. The main content area starts with an 'Intro' section that describes Vite as a no-bundler alternative to webpack made by Vue.js author, Evan You. It highlights the use of native ES modules and instant server start. A note in a yellow box states: 'Vite mode is experimental and many nuxt modules are still incompatible. If you find a bug, please report via [issues](#) with a minimal reproduction'. Below this note, there are two links: 'Quick Start' and 'Learn More About Vite'.

**Nuxt Vite**

[Intro](#)

**Getting Started**

- Installation
- Config
- Build

**Advanced**

- Modules
- How

**Misc**

- Common Issues

Vite (pronounced 'veet') is a no-bundler alternative to webpack made by Vue.js author, Evan You.

Using native ES modules, it provides instant **server start** and **lightning fast HMR** development experience.

This package allows using Vite instead of Webpack during development with [Nuxt](#).

 Vite mode is experimental and many nuxt modules are still incompatible. If you find a bug, please report via [issues](#) with a minimal reproduction

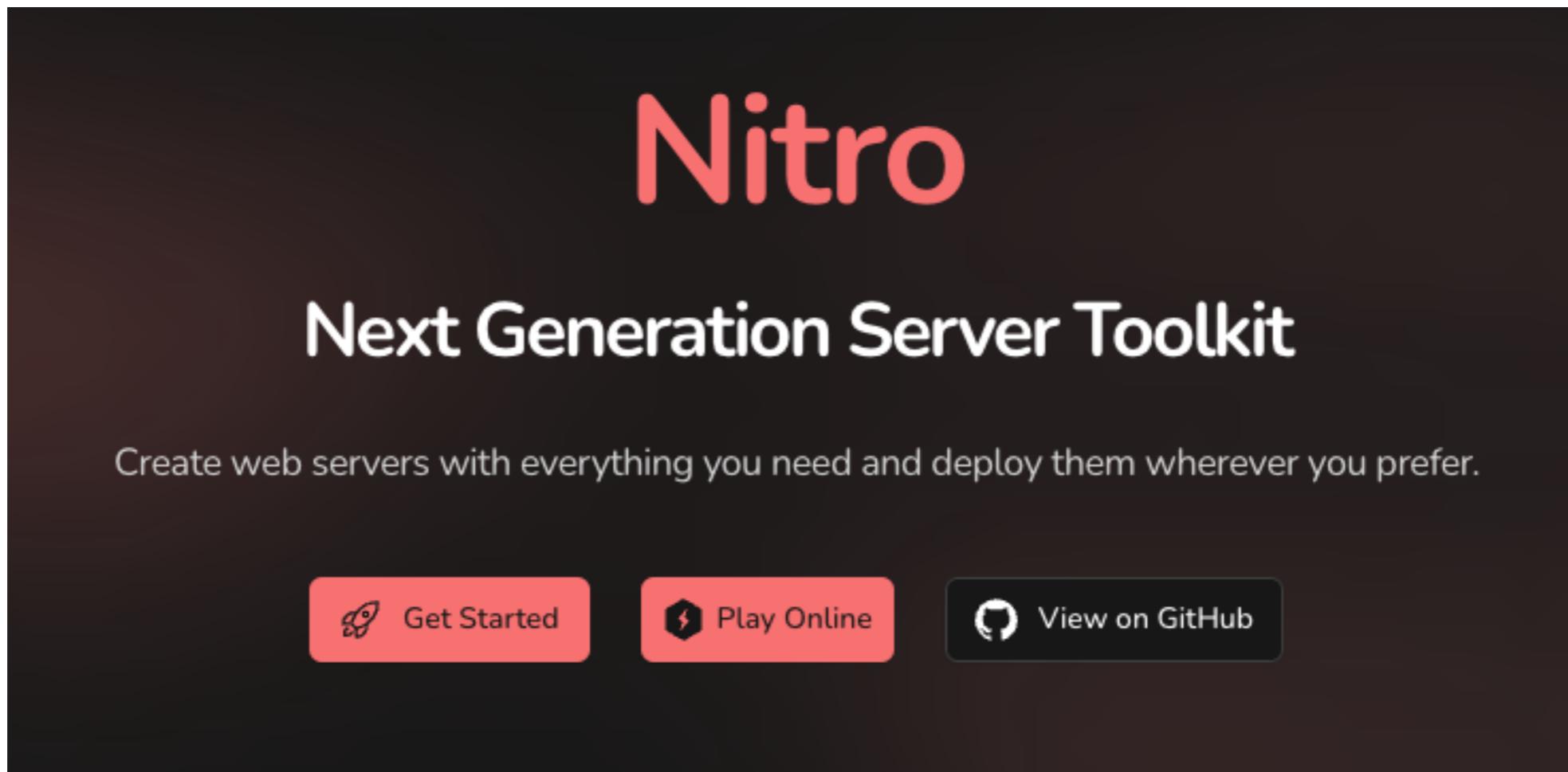
-  [Quick Start](#)
-  [Learn More About Vite](#)

<https://vite.nuxtjs.org/>



# Nuxt Nitro Server

A server for the modern web

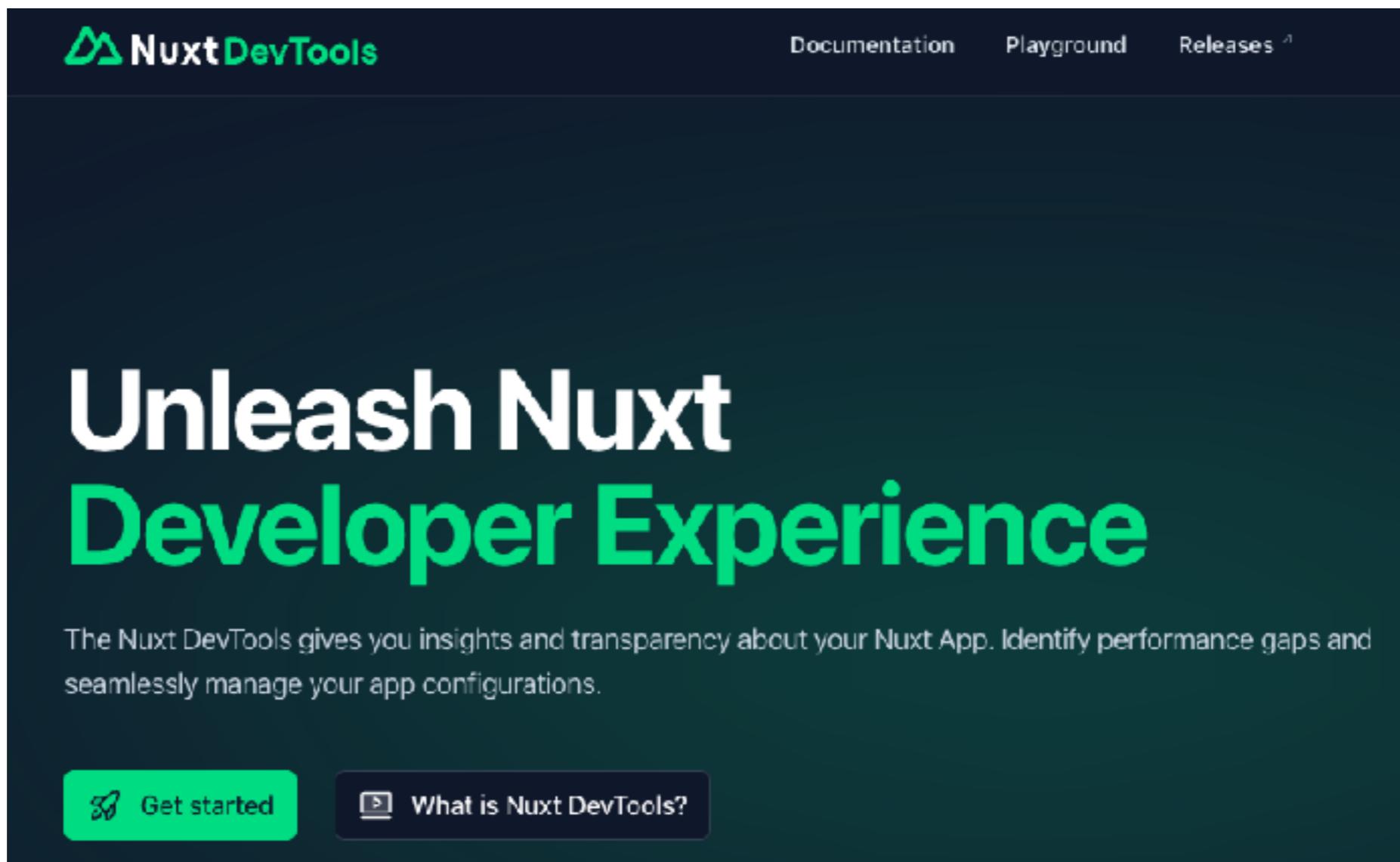


<https://nitro.unjs.io/>



# Nuxt DevTools

Insights and transparency on your app



The screenshot shows the homepage of the Nuxt DevTools website. At the top, there is a dark header with the Nuxt DevTools logo on the left and three navigation links: "Documentation", "Playground", and "Releases". Below the header, the main title "Unleash Nuxt Developer Experience" is displayed in large, bold, white and green text. A descriptive subtitle follows, stating: "The Nuxt DevTools gives you insights and transparency about your Nuxt App. Identify performance gaps and seamlessly manage your app configurations." At the bottom of the page are two buttons: a green "Get started" button and a dark blue "What is Nuxt DevTools?" button.

<https://devtools.nuxt.com/>



# Nuxt DevTools

Insights and transparency on your app

## next.config.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

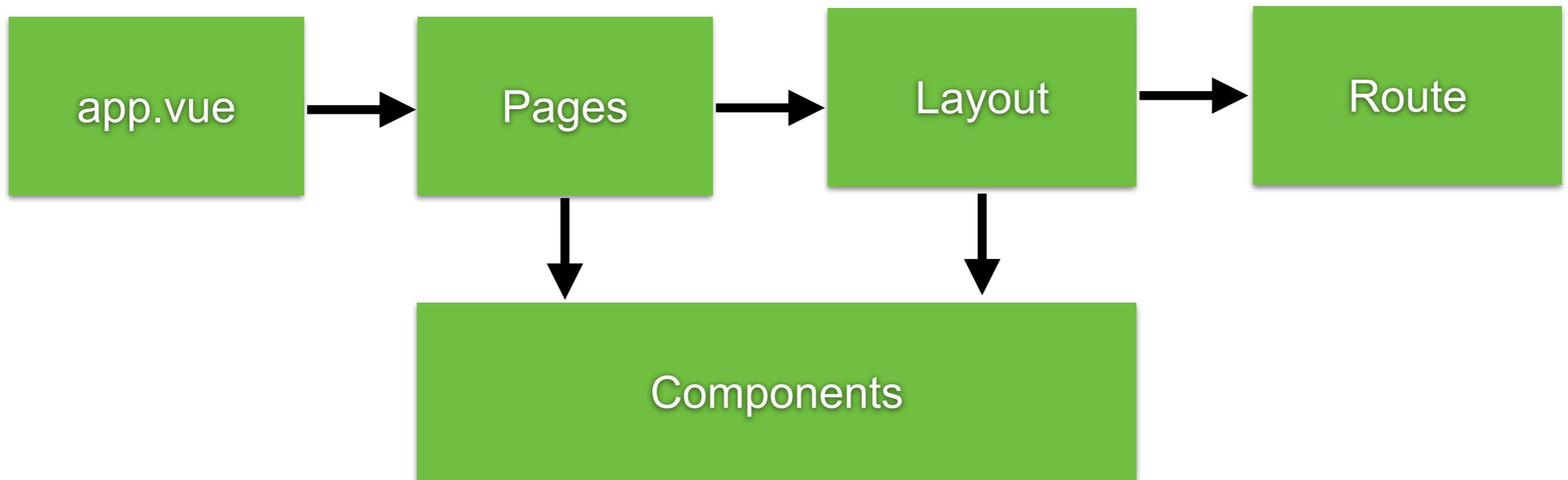
Missing tags (SEO)



# Workflow of application



# NuxtJS Workflow



`nuxt.config.ts` (configuration)



# App.vue

Entry point of application

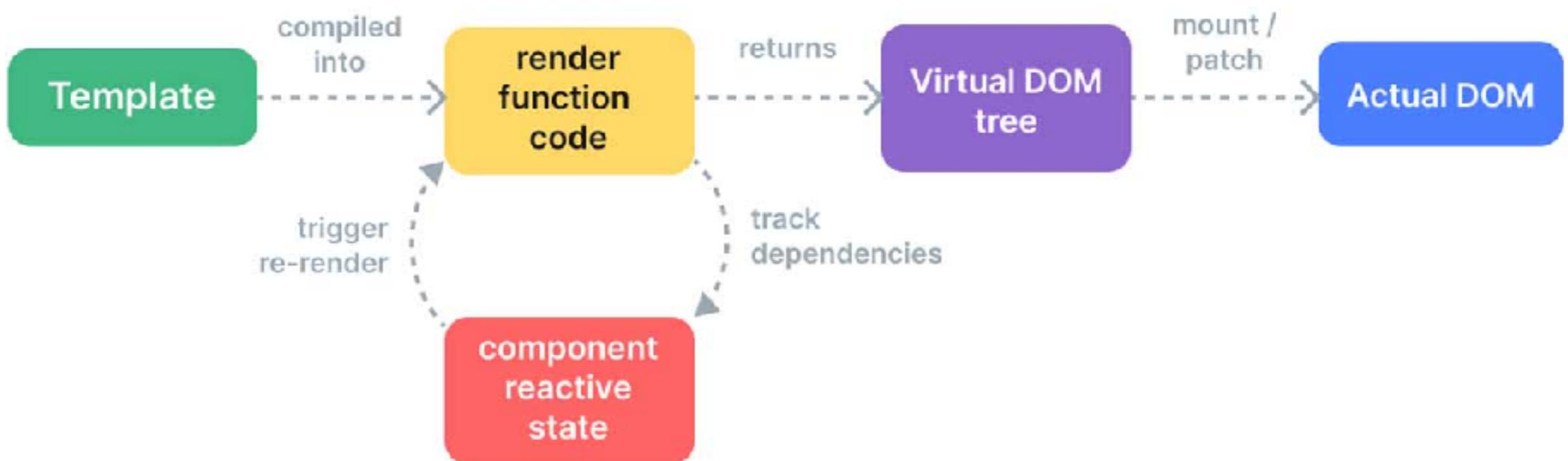
```
<template>
  <div>
    <NuxtWelcome />
  </div>
</template>
```

Components in view

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



# Render System

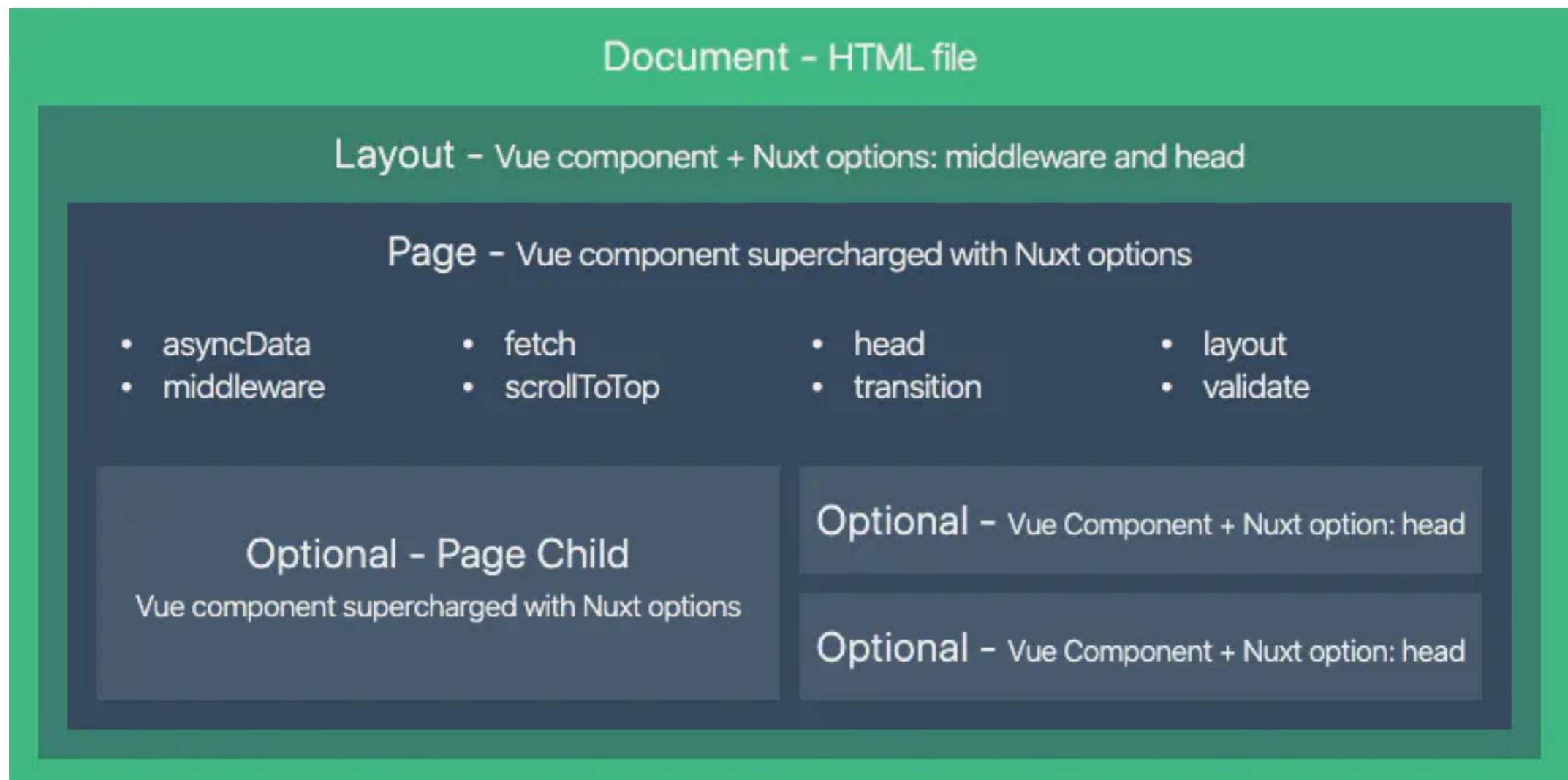


<https://www.vuemastery.com/blog/the-future-of-vue-vapor-mode/>



# 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

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

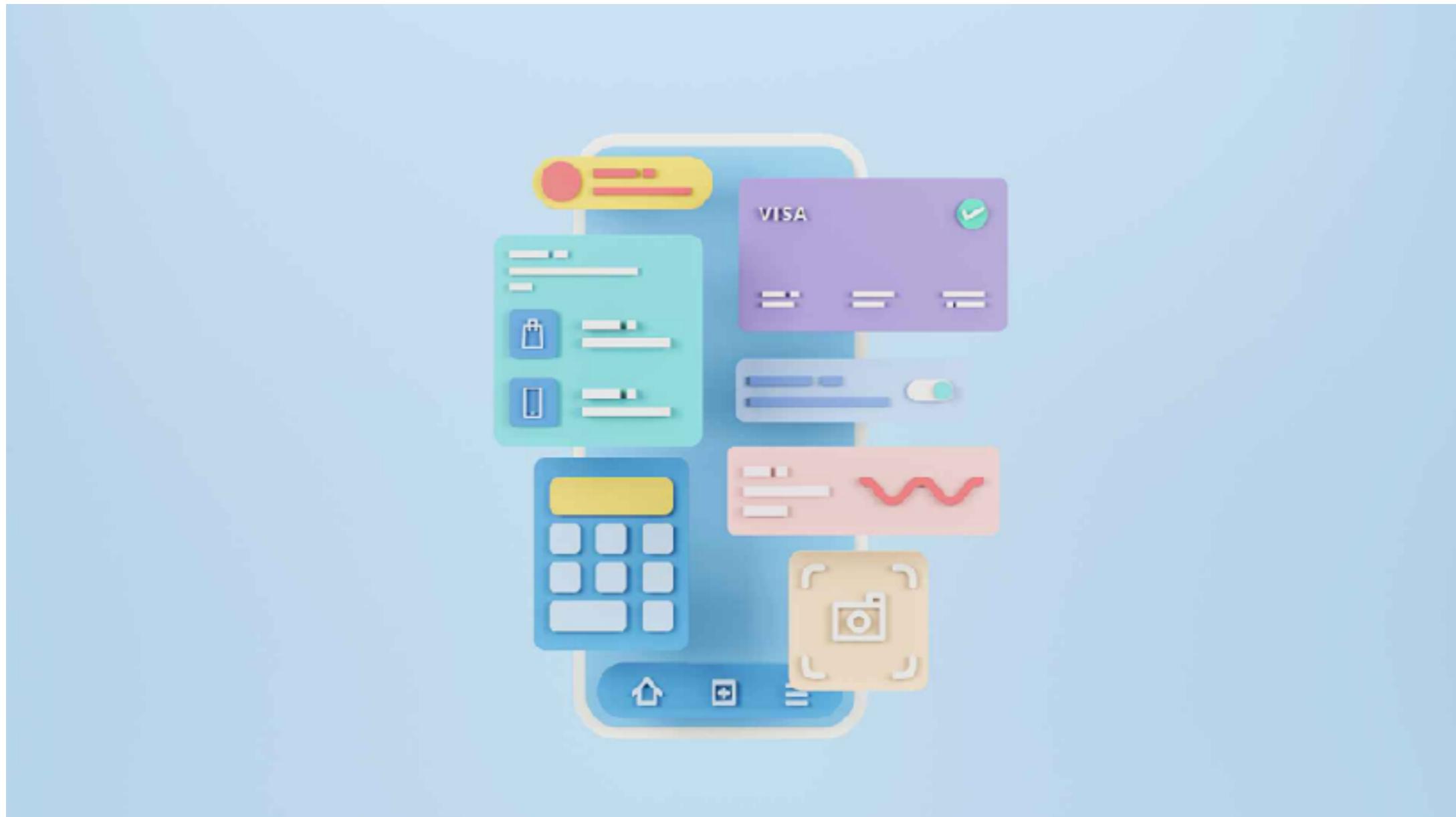
<template>
  <p class="greeting">{{ greeting }}</p>
</template>

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

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



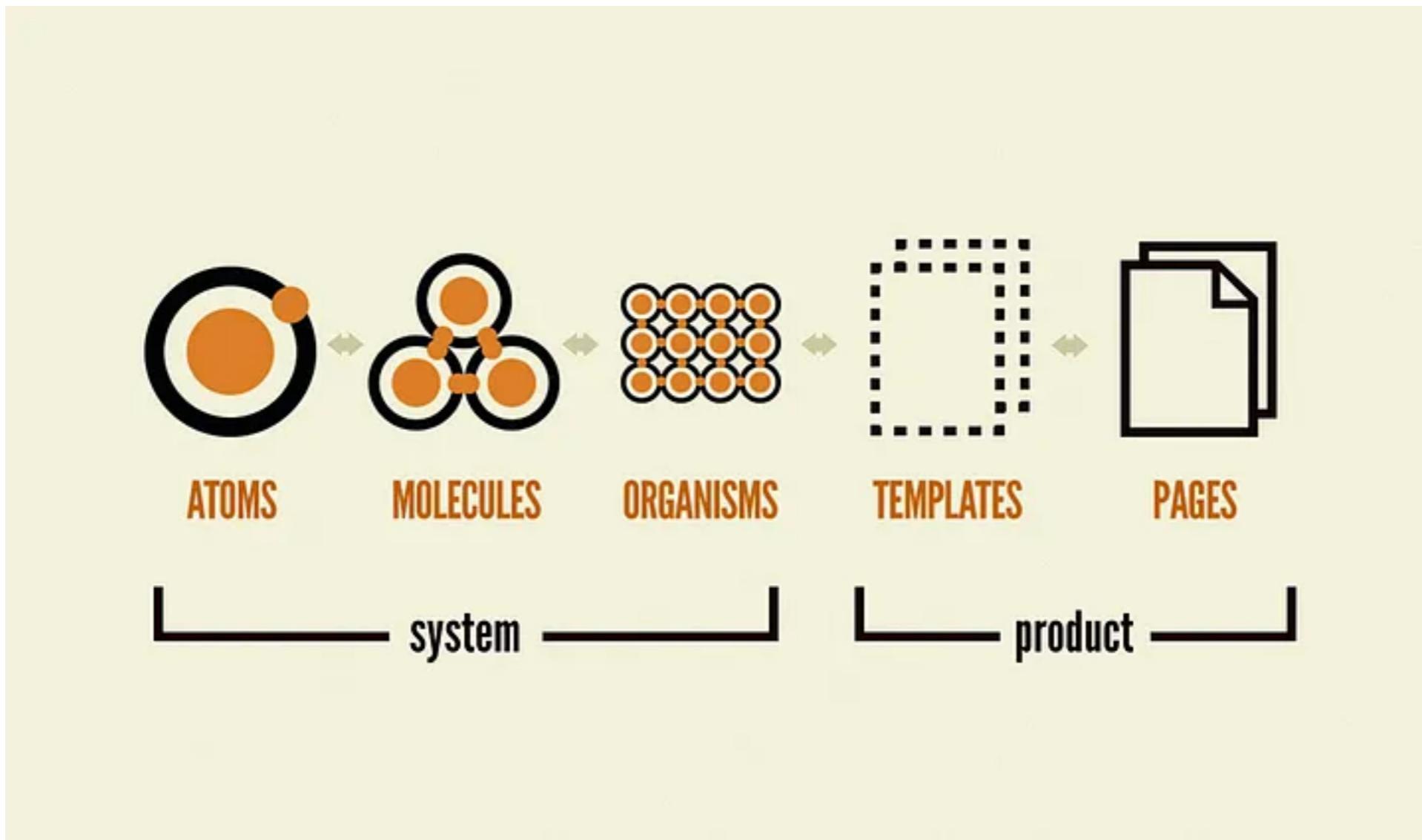
# Component-Driven



<https://www.componentdriven.org/>



# Atomic Design



<https://atomicdesign.bradfrost.com/>



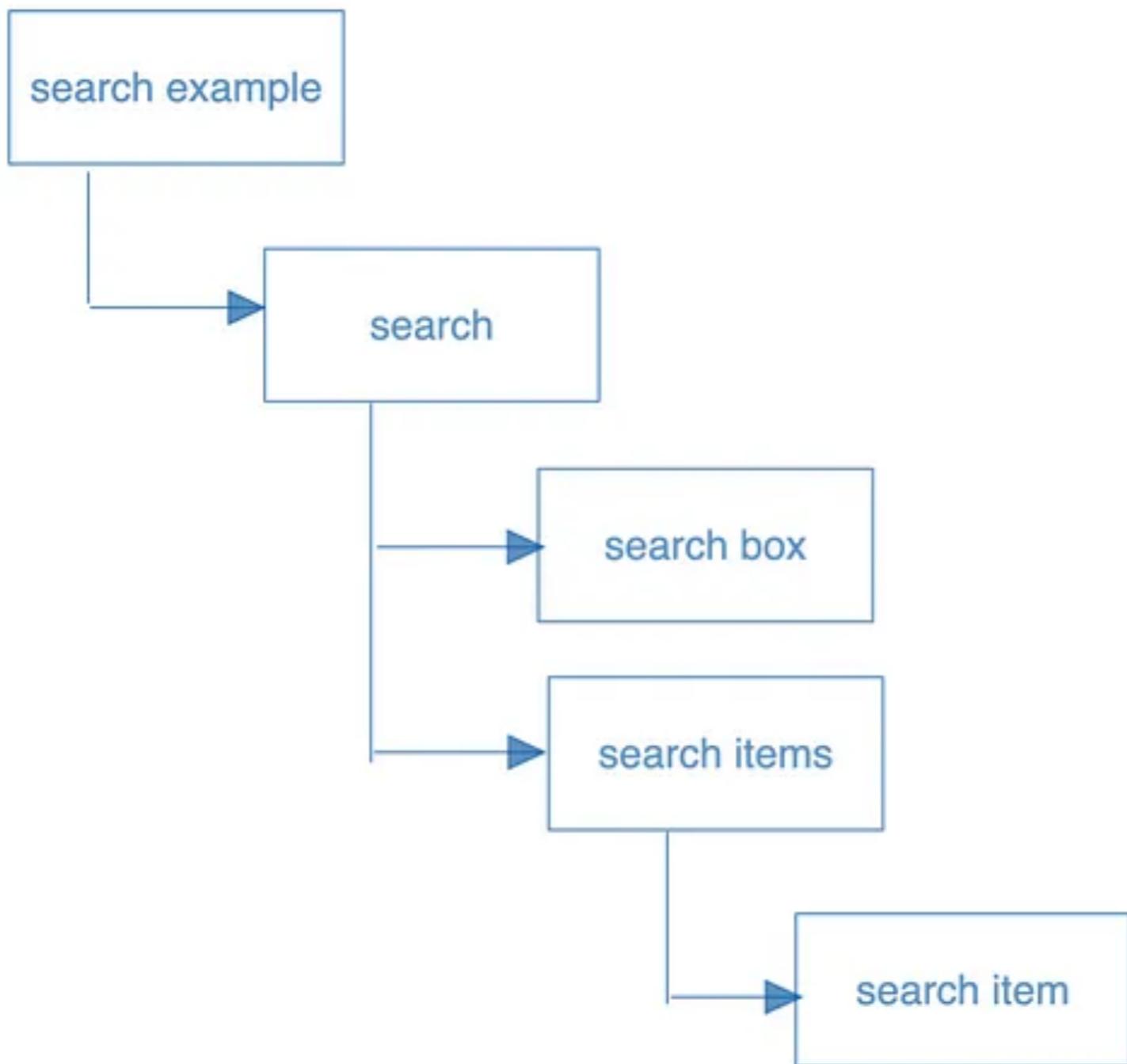
# Example



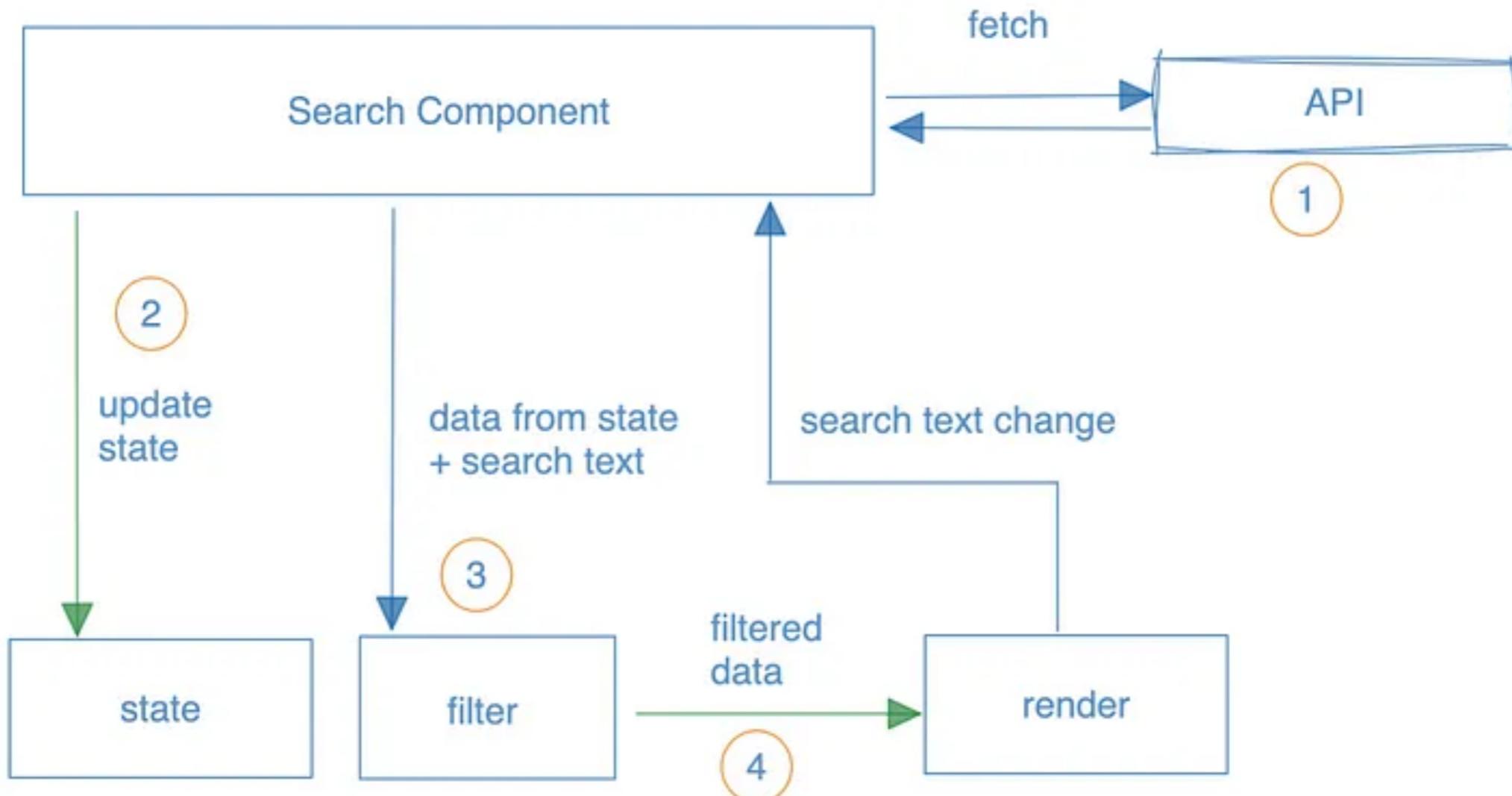
# 1. Component Design



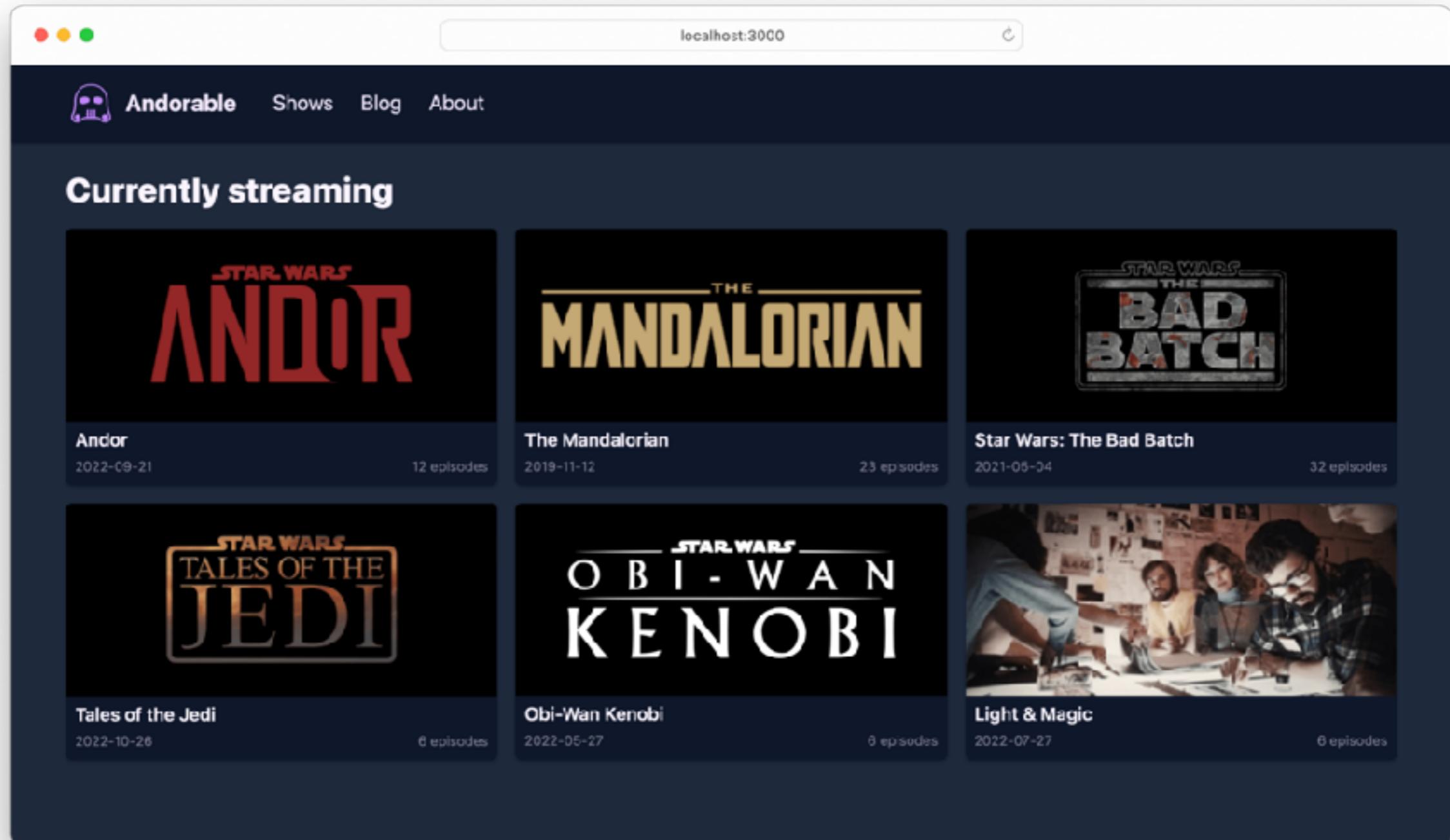
# 2. Component Diagram



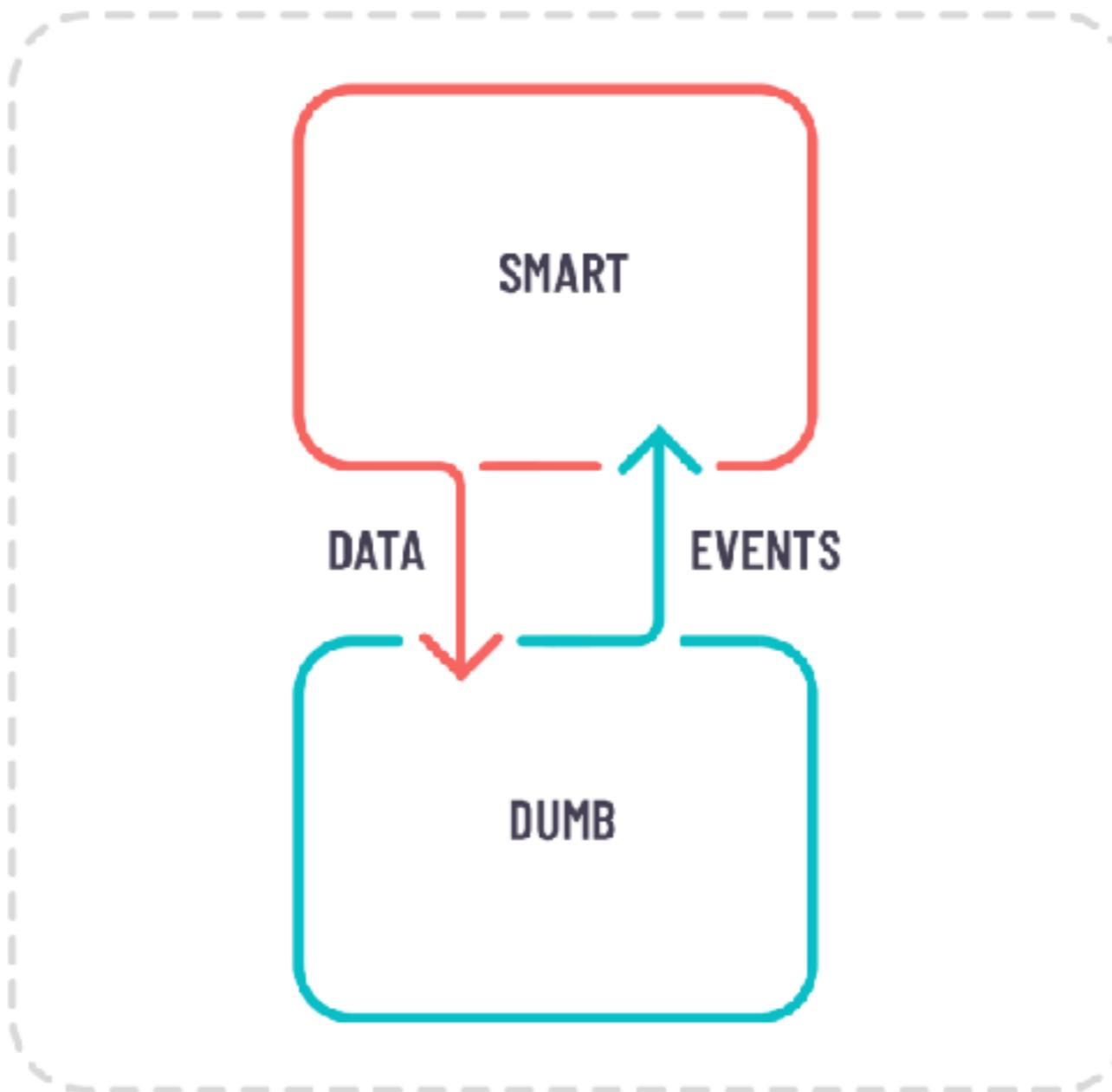
# 3. Code Design



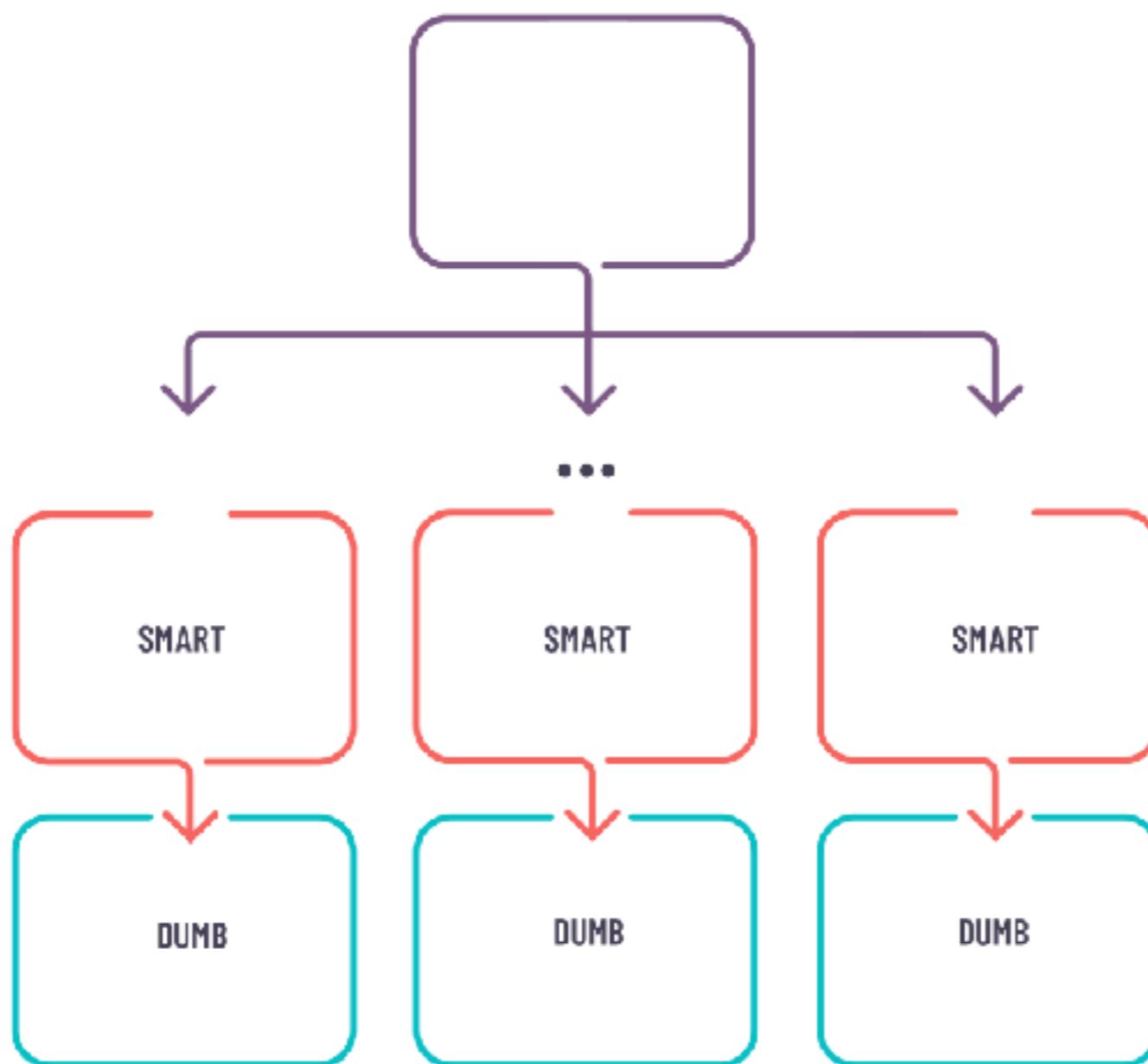
# Design your component ?



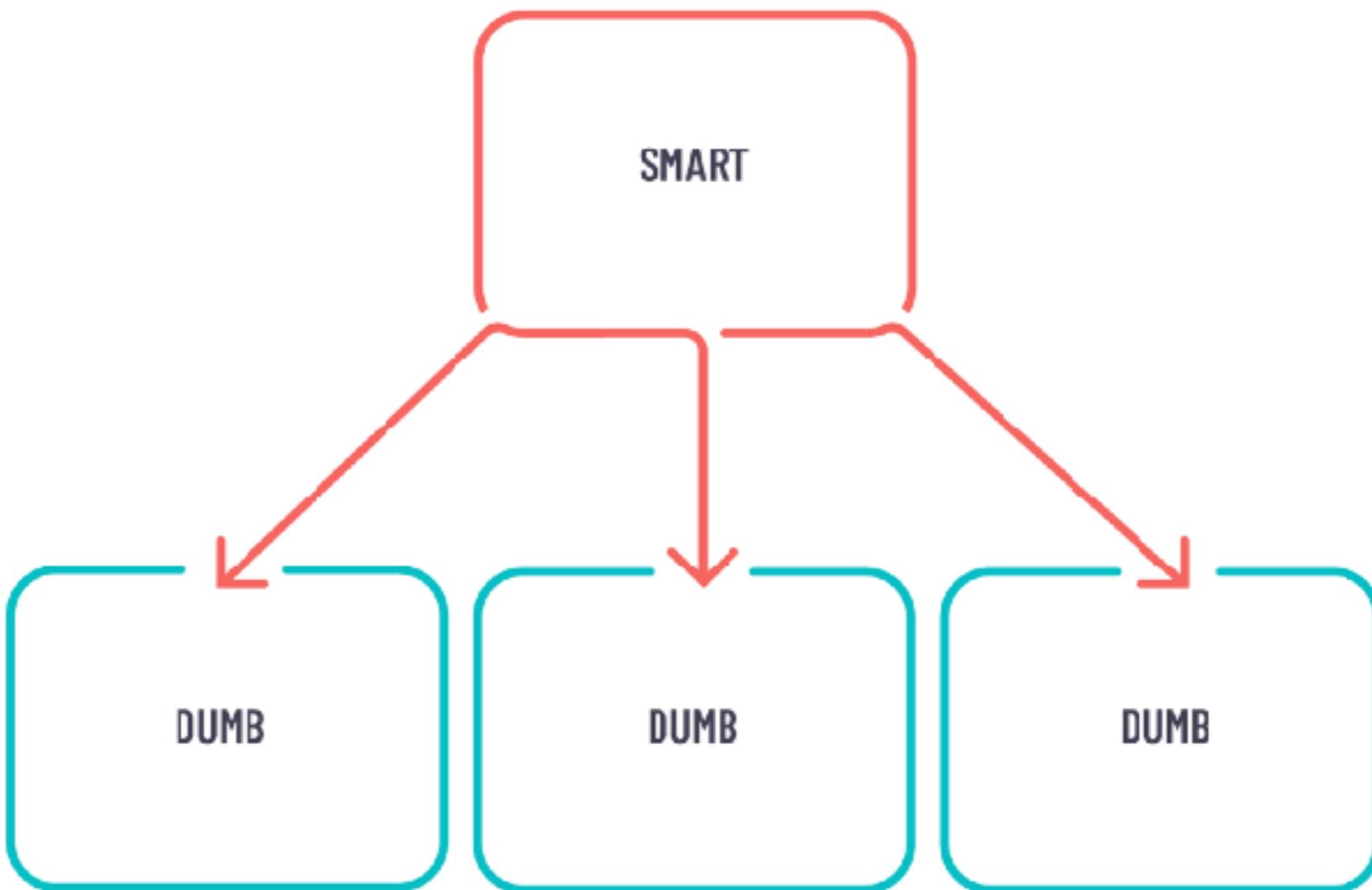
# Types of component



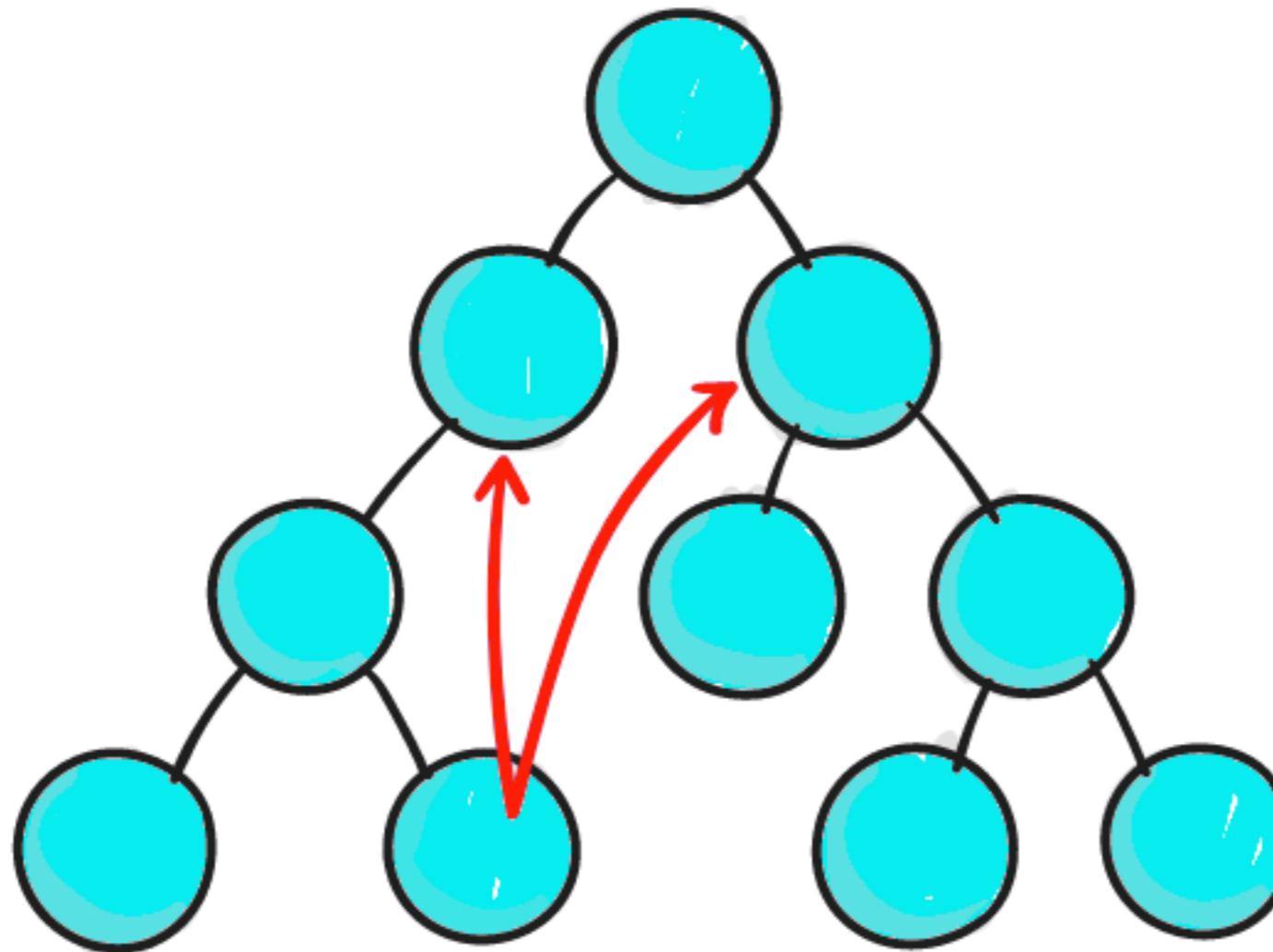
# Types of component



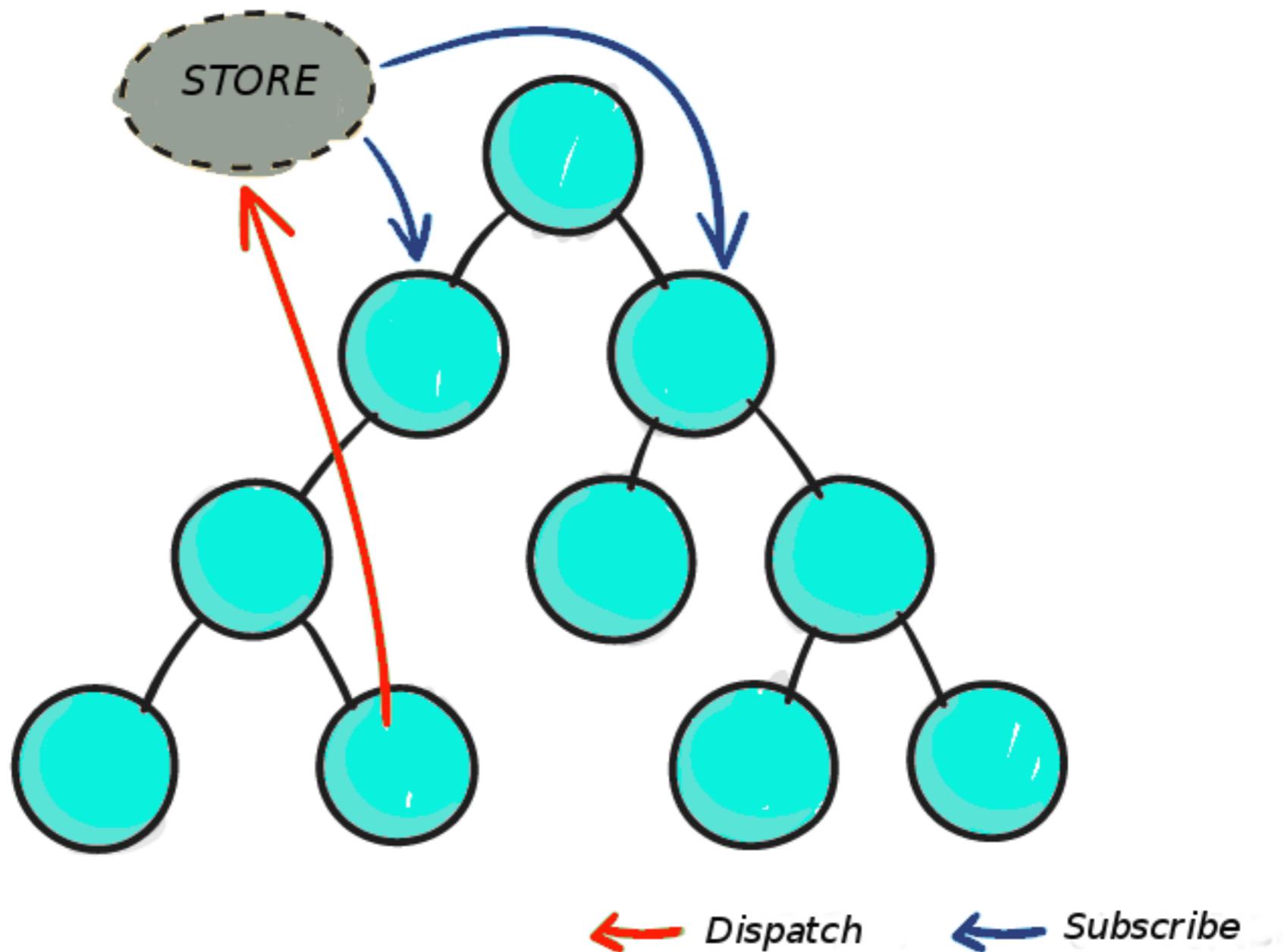
# Types of component



# Manage state of components ?



# Manage state of components ?

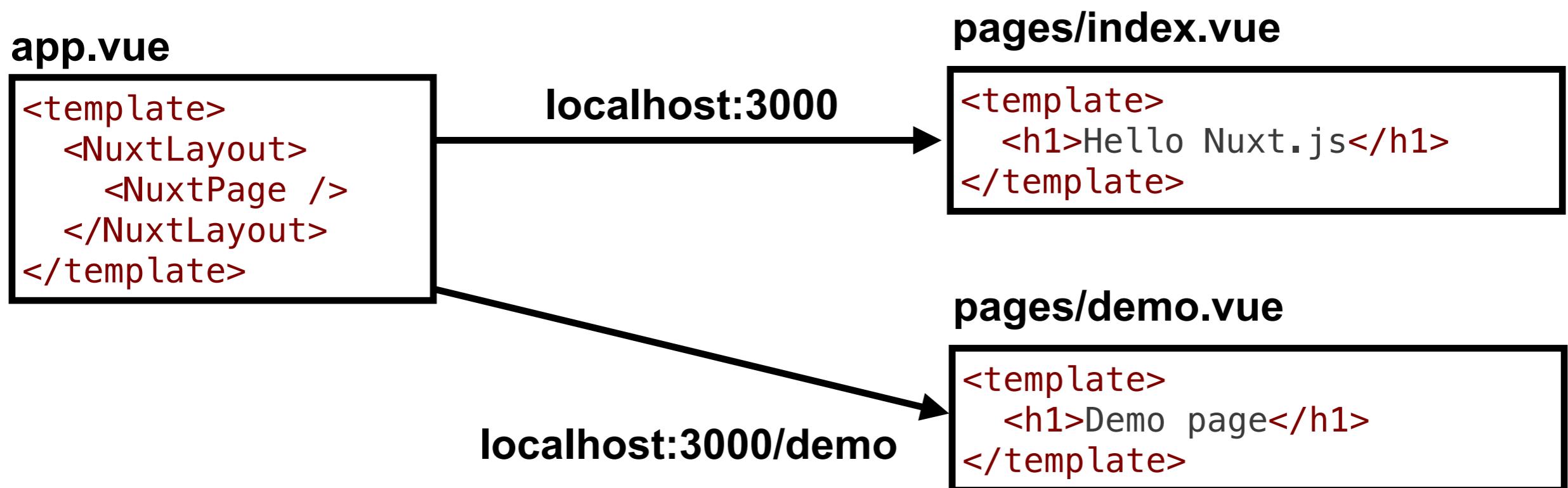


# Pages



# Pages

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



# 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

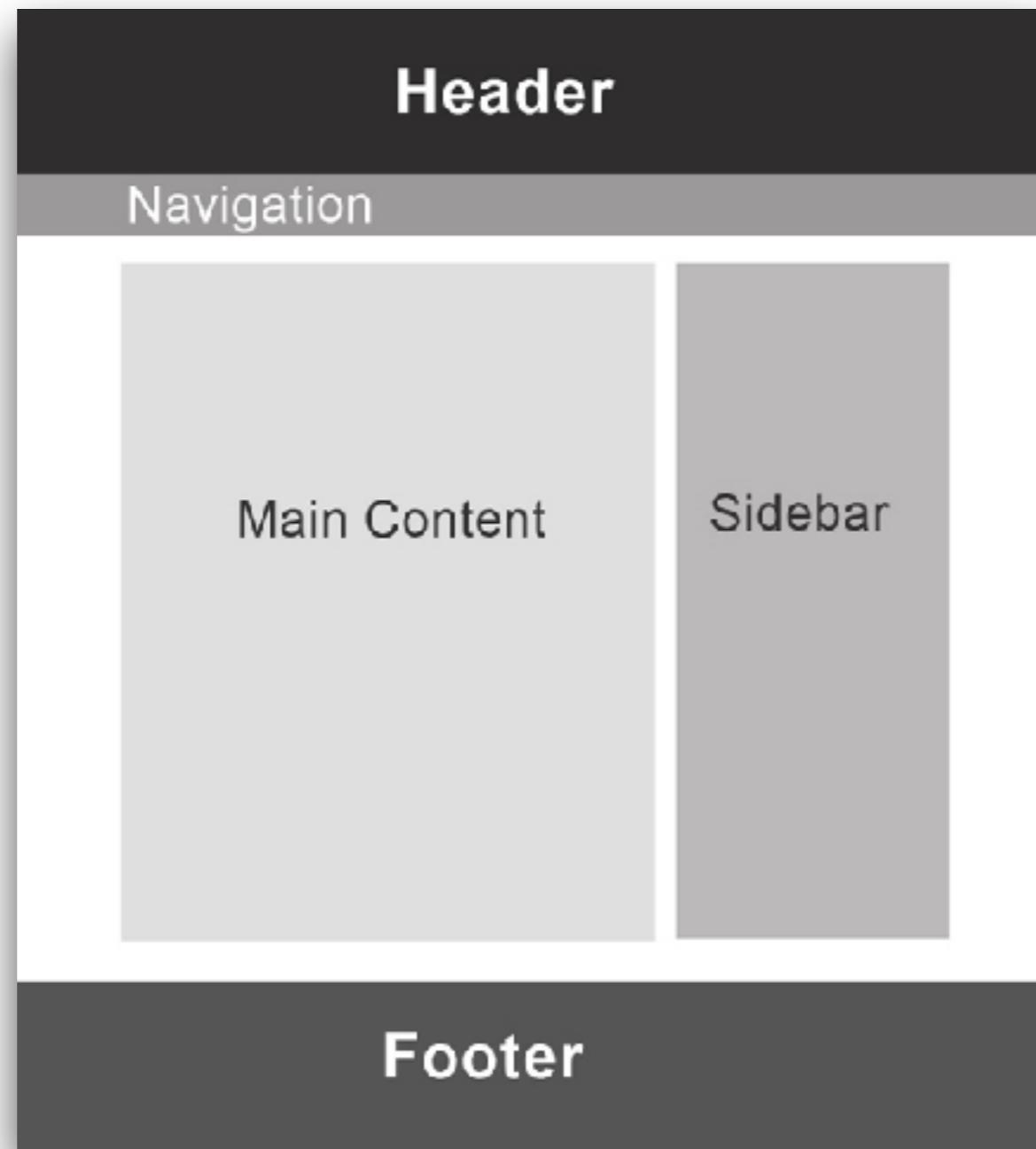
eg. Header, Menu and Footer

Create in folder **/layouts**

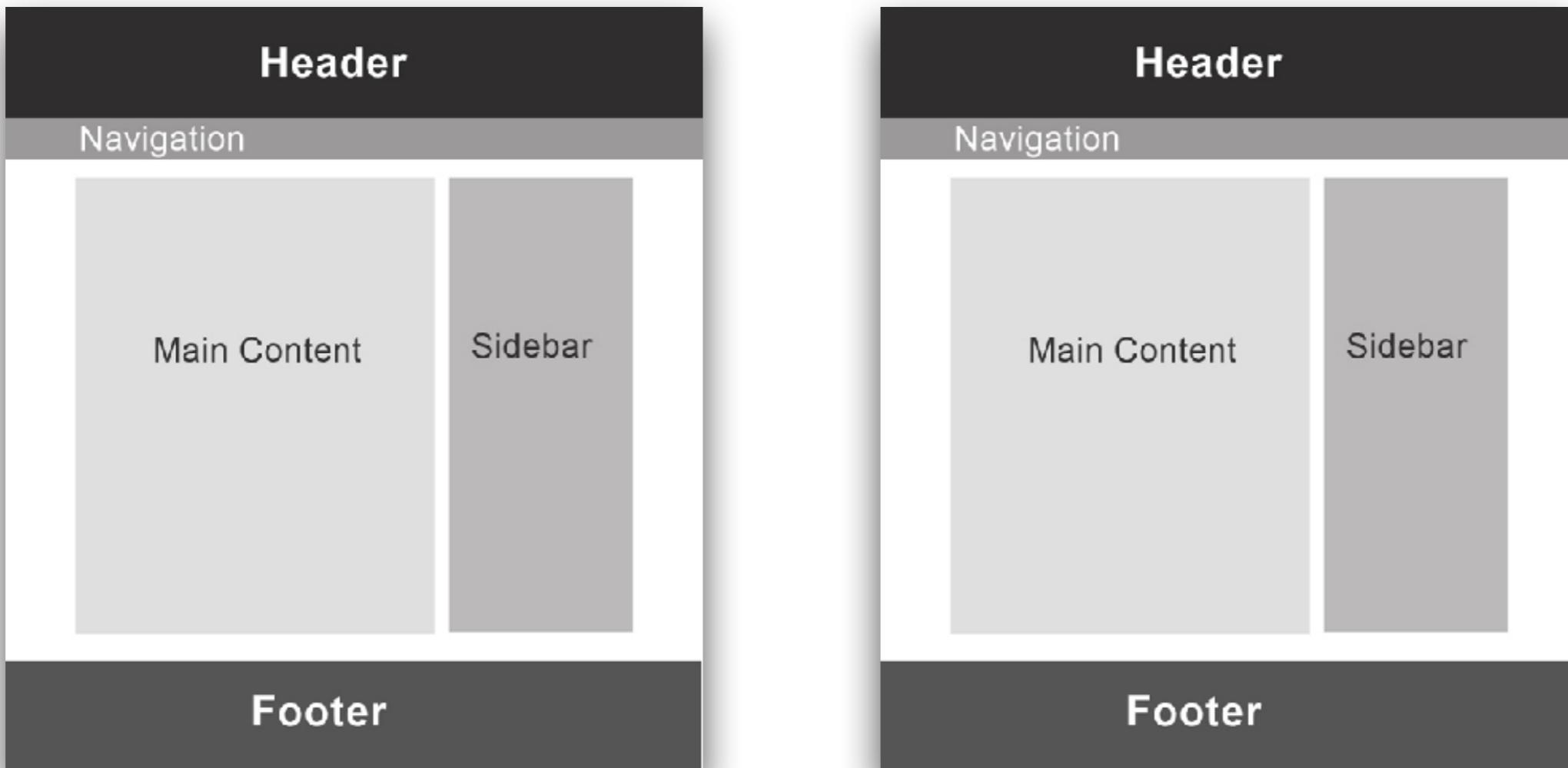
<https://nuxt.com/docs/guide/directory-structure/layouts>



# 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

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

## app.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

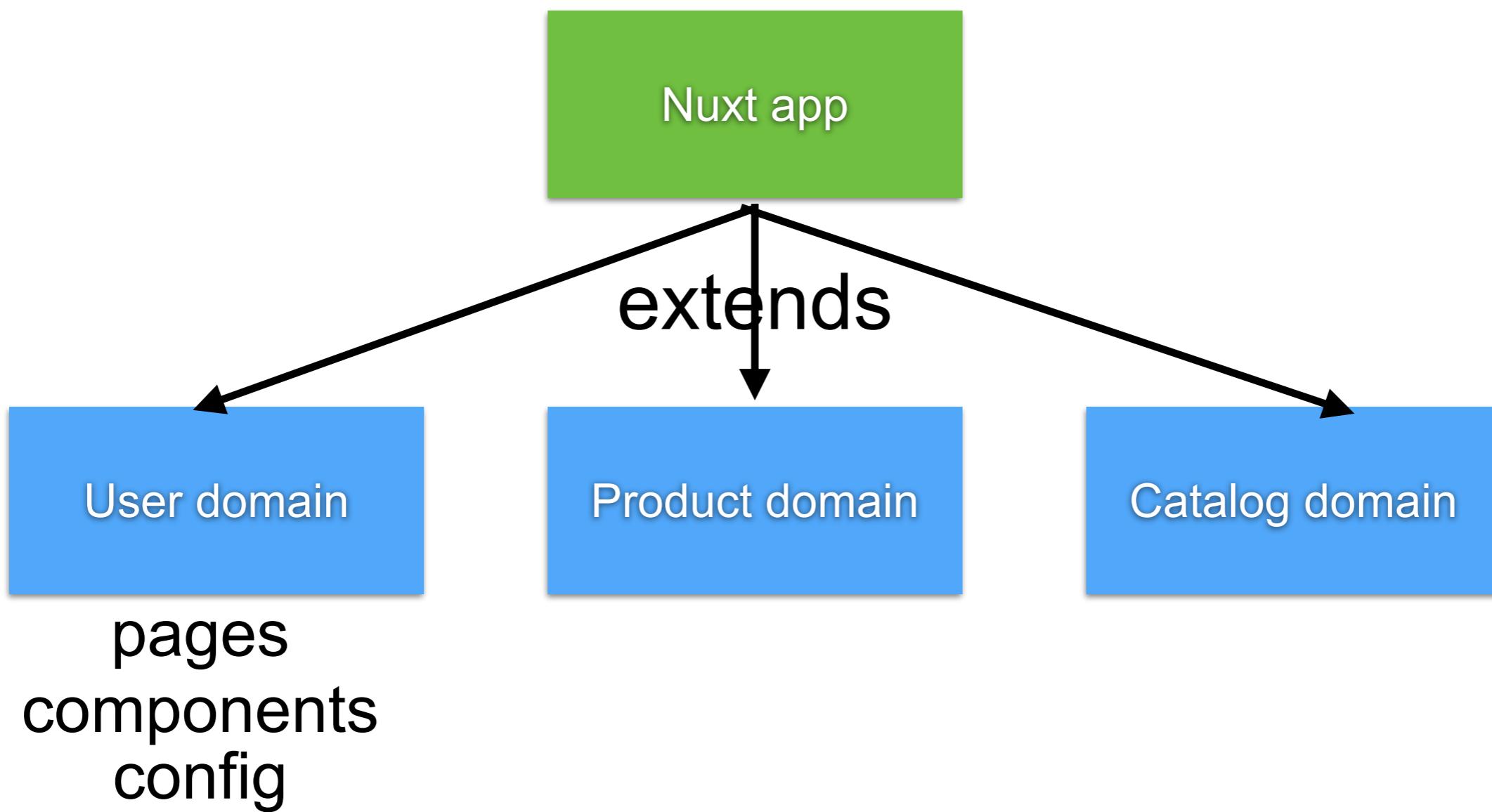
Git repository

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

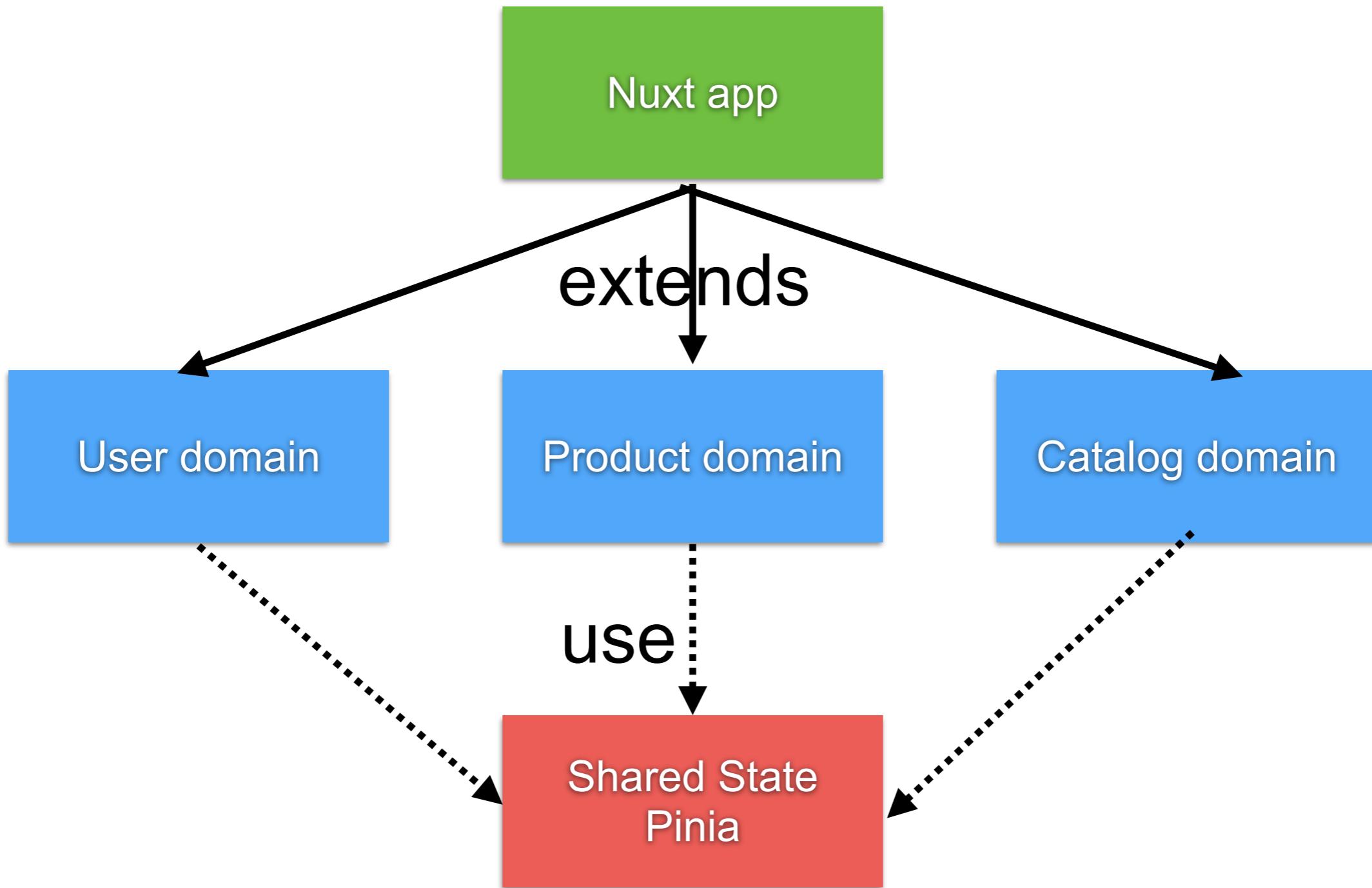


# Project structure

## DDD (Domain-Drive Design)



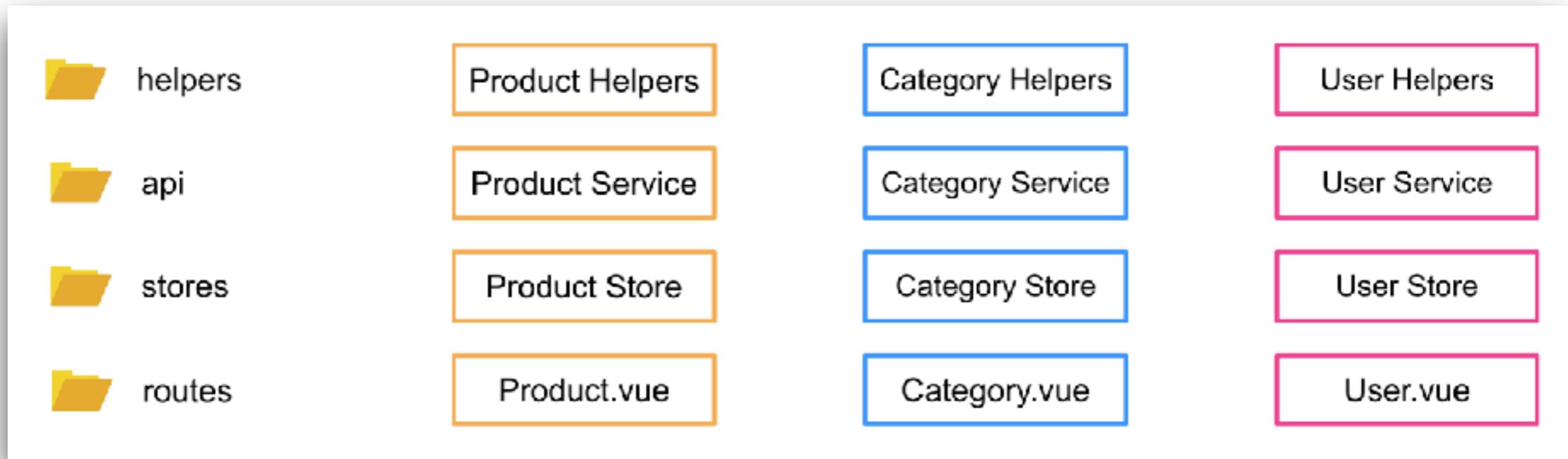
# Shared state



# Modules



# Local module

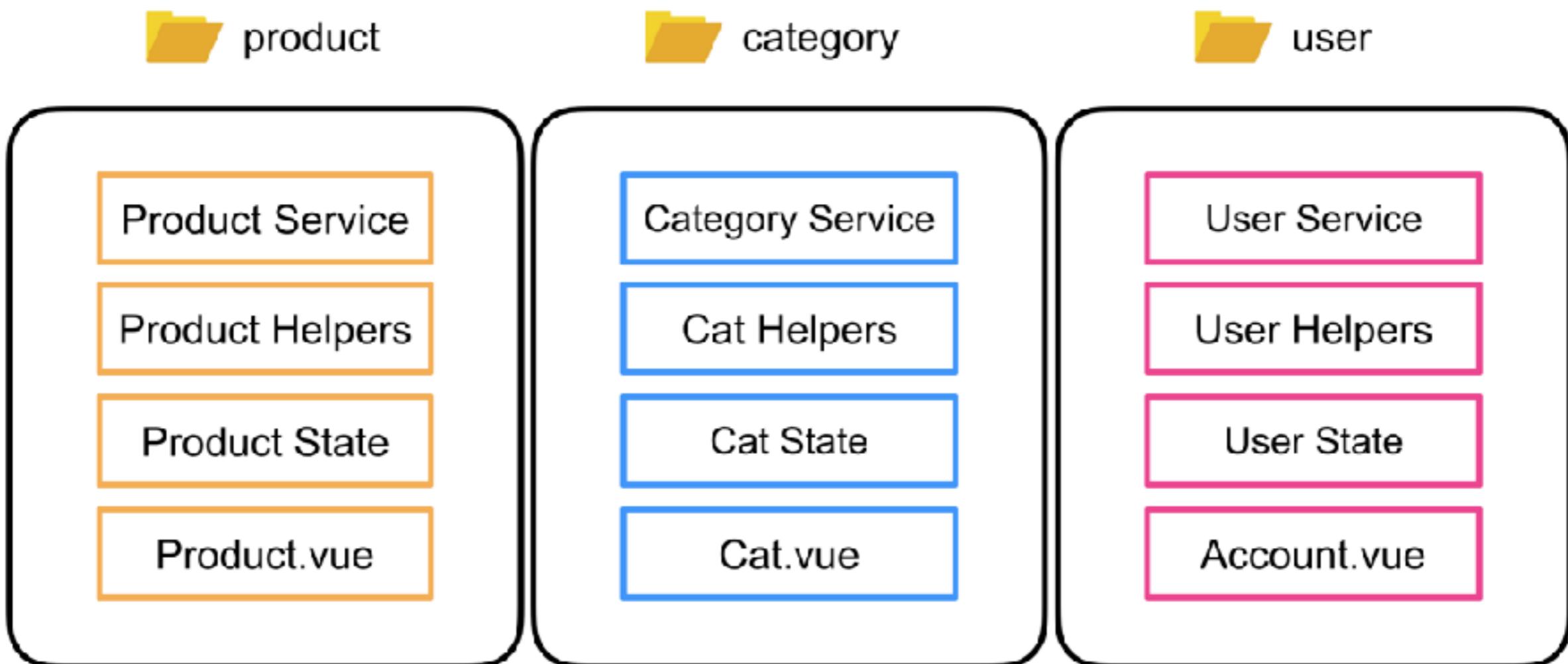


Create in folder **/modules**

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



# Multi-modules (by domain)



# Nuxt modules

The screenshot shows the Nuxt Modules website interface. At the top, there's a navigation bar with a logo, a search bar, and a "Categories" dropdown. Below the categories, there's a sidebar with a list of module categories: Analytics, CMS, CSS, Database, Devtools, Ecommerce, Extensions, Fonts, Images, and Libraries. The main content area displays four cards for specific modules: "Nuxt Image" (Plug-and-play image optimization), "Nuxt Content" (Git-based CMS with Markdown support), "Nuxt DevTools" (Visual tools that help you to know your app), and "Nuxt UI" (Fully styled and customizable components). Below these cards, there's a section for featured modules: "Kinde" (Kindle authentication integration for Nuxt), "devtools" (Unleash Nuxt Developer Experience), and "pinia" (The Vue Store that you will enjoy using). Each module card includes its name, icon, status (e.g., Sponsor, Official), description, and metrics (e.g., 1.4K stars, 1.6M downloads).

**Nuxt Modules**

Discover our list of **modules** to supercharge your Nuxt project. Created by the Nuxt team and community.

Module Author Guide | List a module

**Categories**

Analytics | CMS | CSS | Database | Devtools | Ecommerce | Extensions | Fonts | Images | Libraries

**Nuxt Image** 🌟  
Plug-and-play image optimization.

**Nuxt Content** 🌟  
Git-based CMS with Markdown support.

**Nuxt DevTools** 🌟  
Visual tools that help you to know your app.

**Nuxt UI** 🌟  
Fully styled and customizable components.

**Kinde** 🌟 Sponsor  
Kindle authentication integration for Nuxt

**devtools** 🌟 Official  
Unleash Nuxt Developer Experience. A set of visual tools that help you t...

**pinia**  
The Vue Store that you will enjoy using

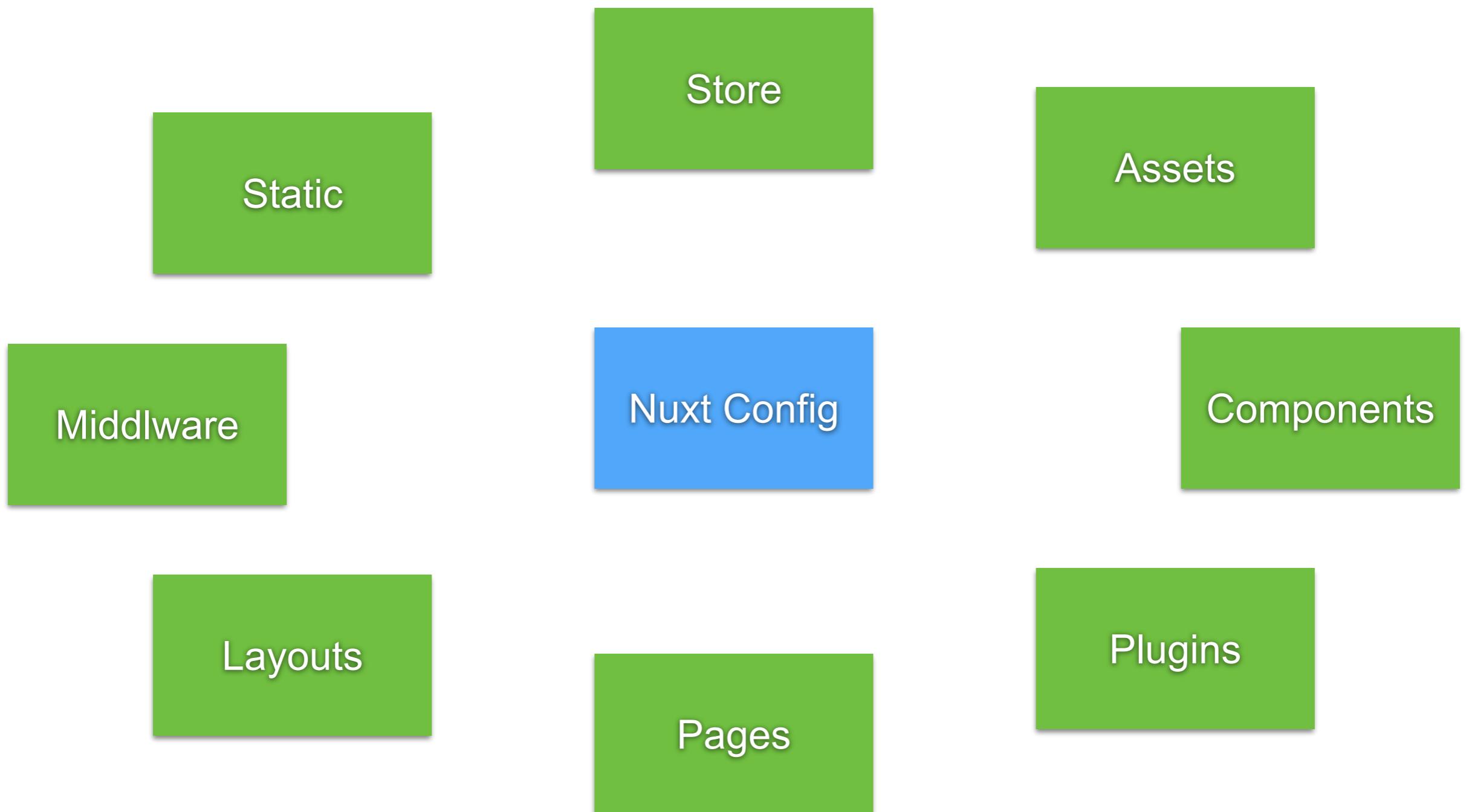
Search... | Downloads

<https://nuxt.com/modules>



# Nuxt Configuration

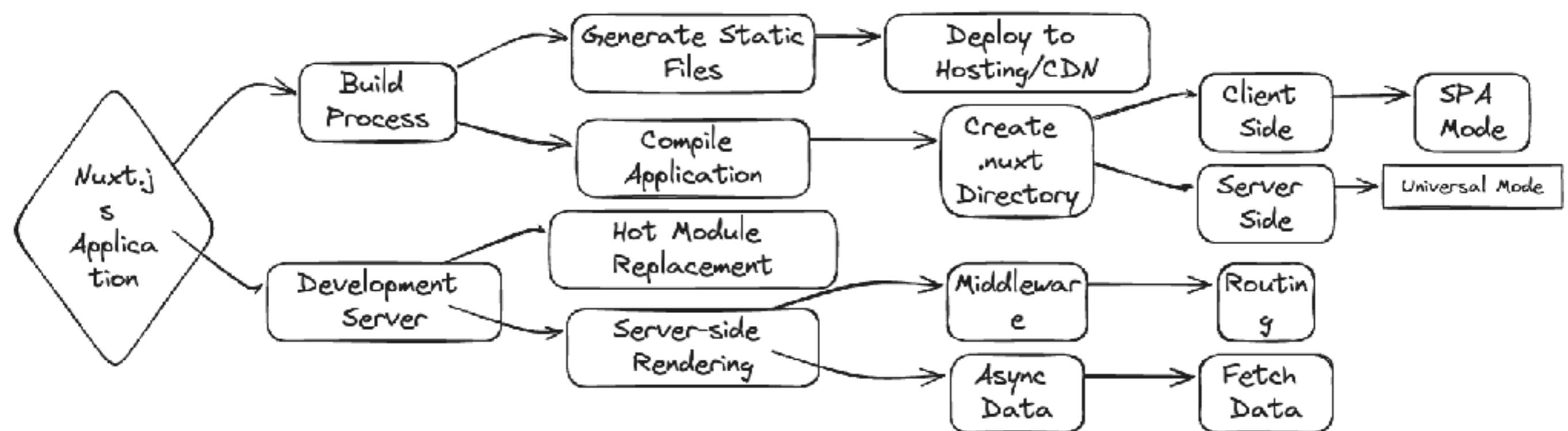




<https://nuxt.com/docs/api/nuxt-config>



# NuxtJS Workflow



# Rendering Mode



# NuxtJS Rendering Modes

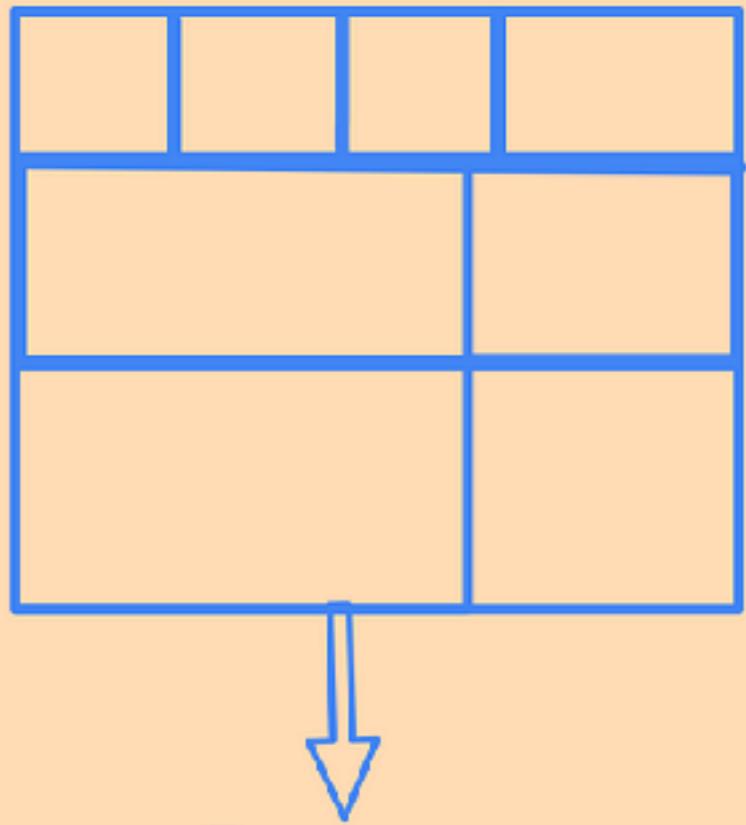
Config in file next.config.ts

```
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 },
  },
})
```

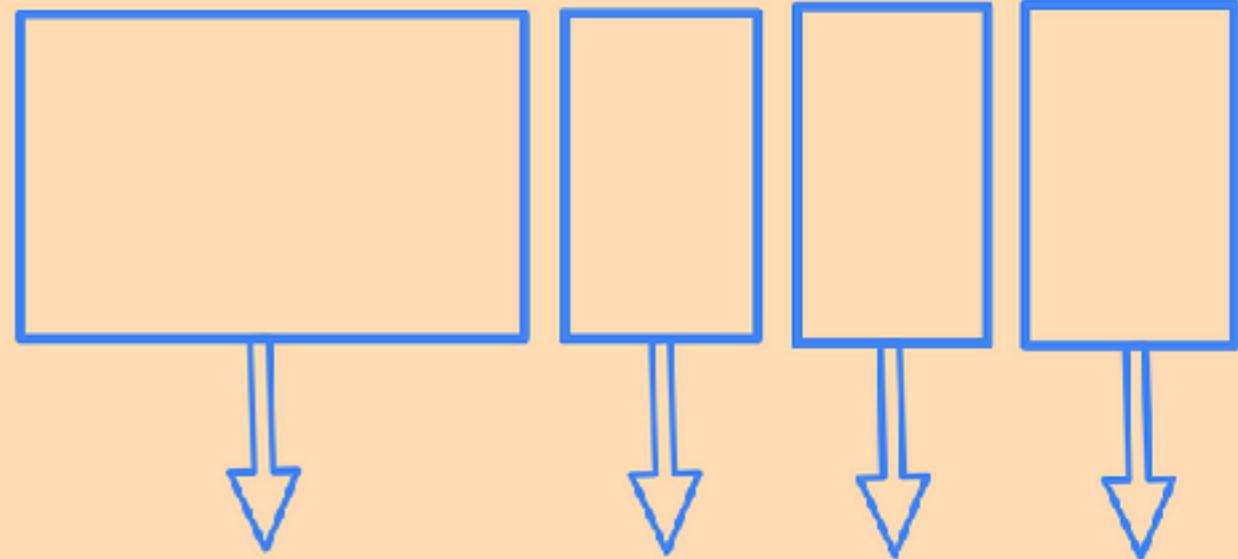
<https://nuxt.com/docs/guide/concepts/rendering>



# Code Splitting



Single Request



Multiple Requests

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

- \* 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

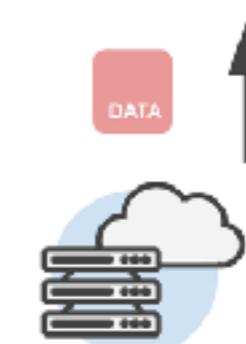
The JavaScript loads the data in the page.



## Step 4

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

The page is updated without having to reload.



codecapsule.com



# SPA (Single Page Application)

## Client Side Rendering (CSR)

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



# 2. SSR



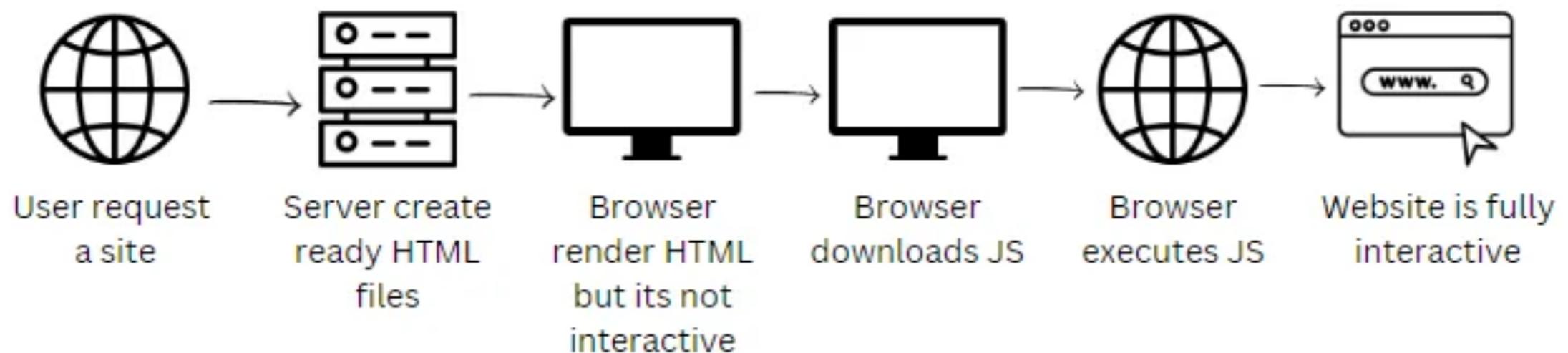
# SSR (Server Side Rendering)

## Universal Rendering

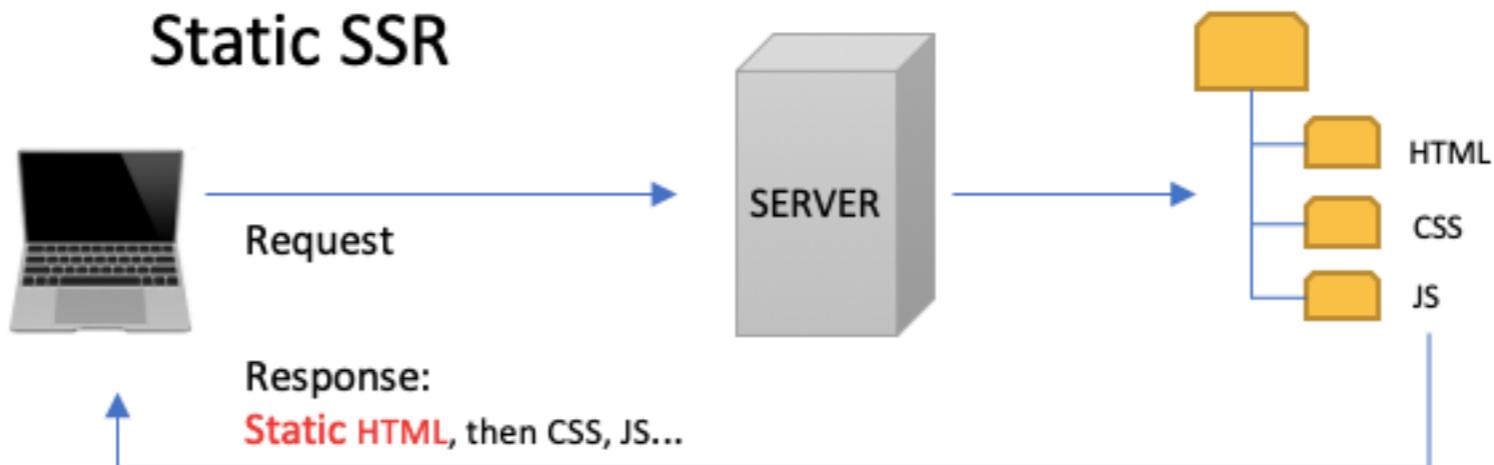
The Nuxt server generates HTML on demand  
and delivers a fully rendered HTML page to the  
browser



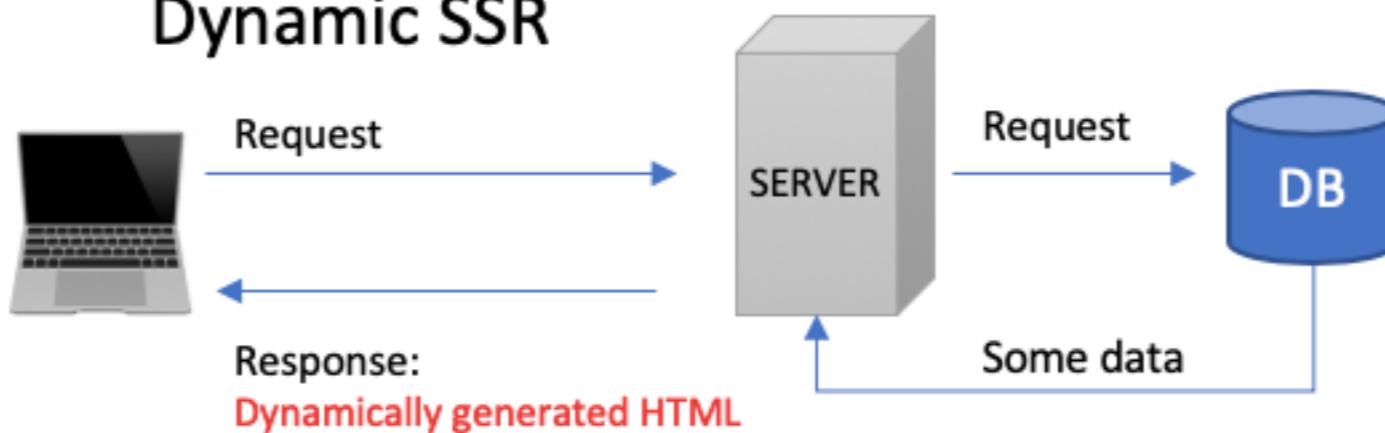
## SSR (Server-side Rendering)

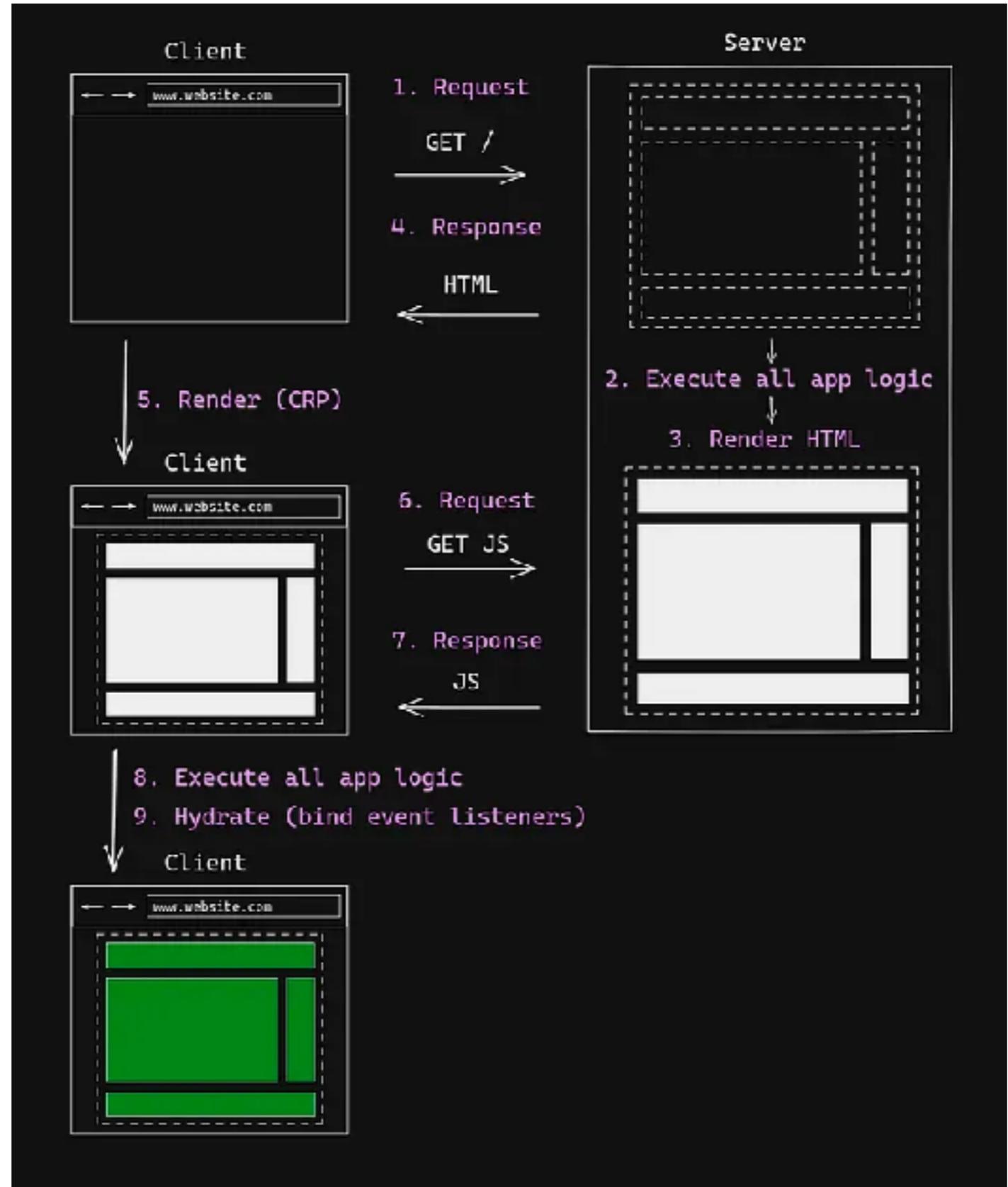


## Static SSR



## Dynamic SSR





# 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

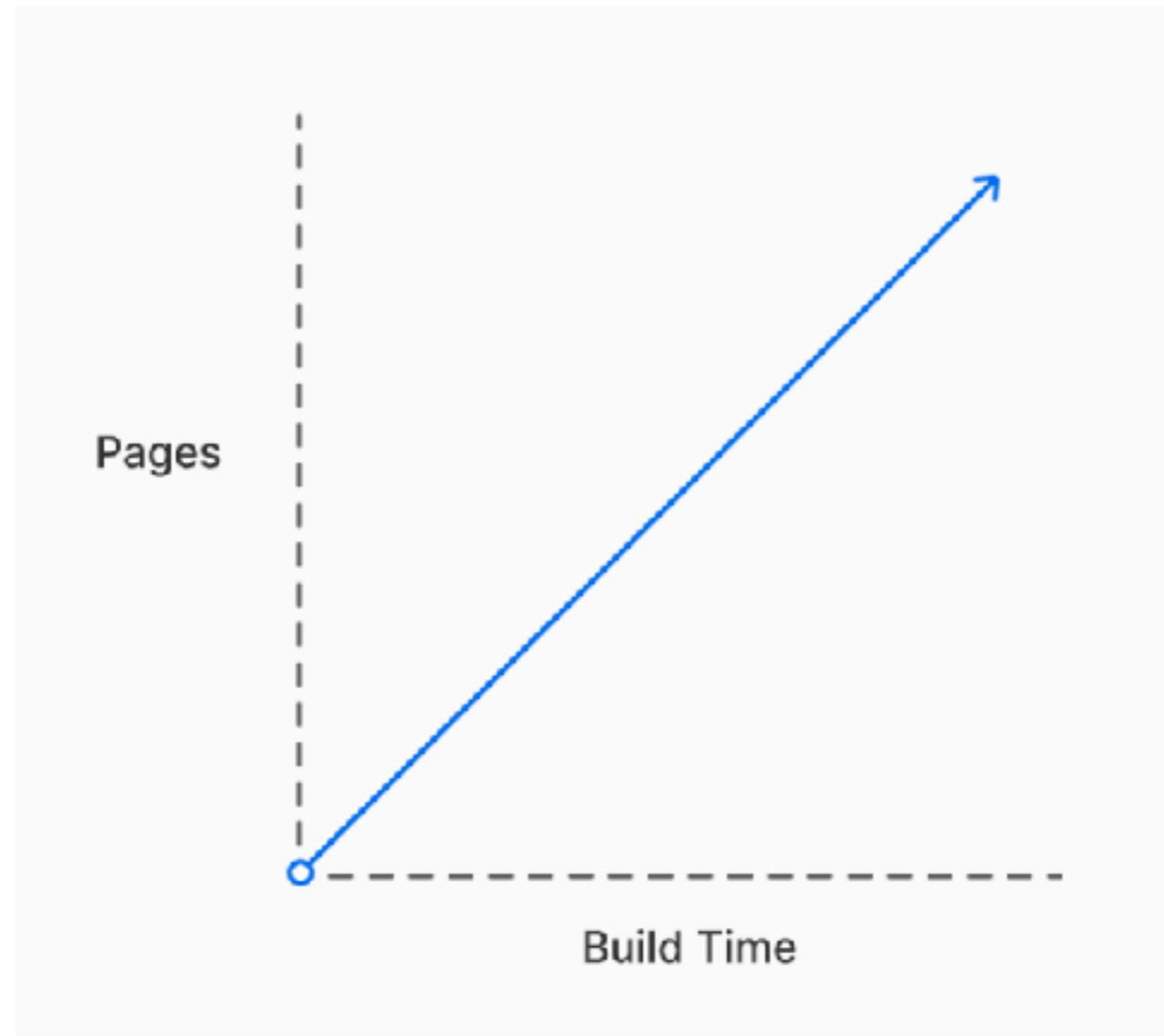


# SSG (Static Site Generation)



# SSG (Static Site Generation)

Problem !!



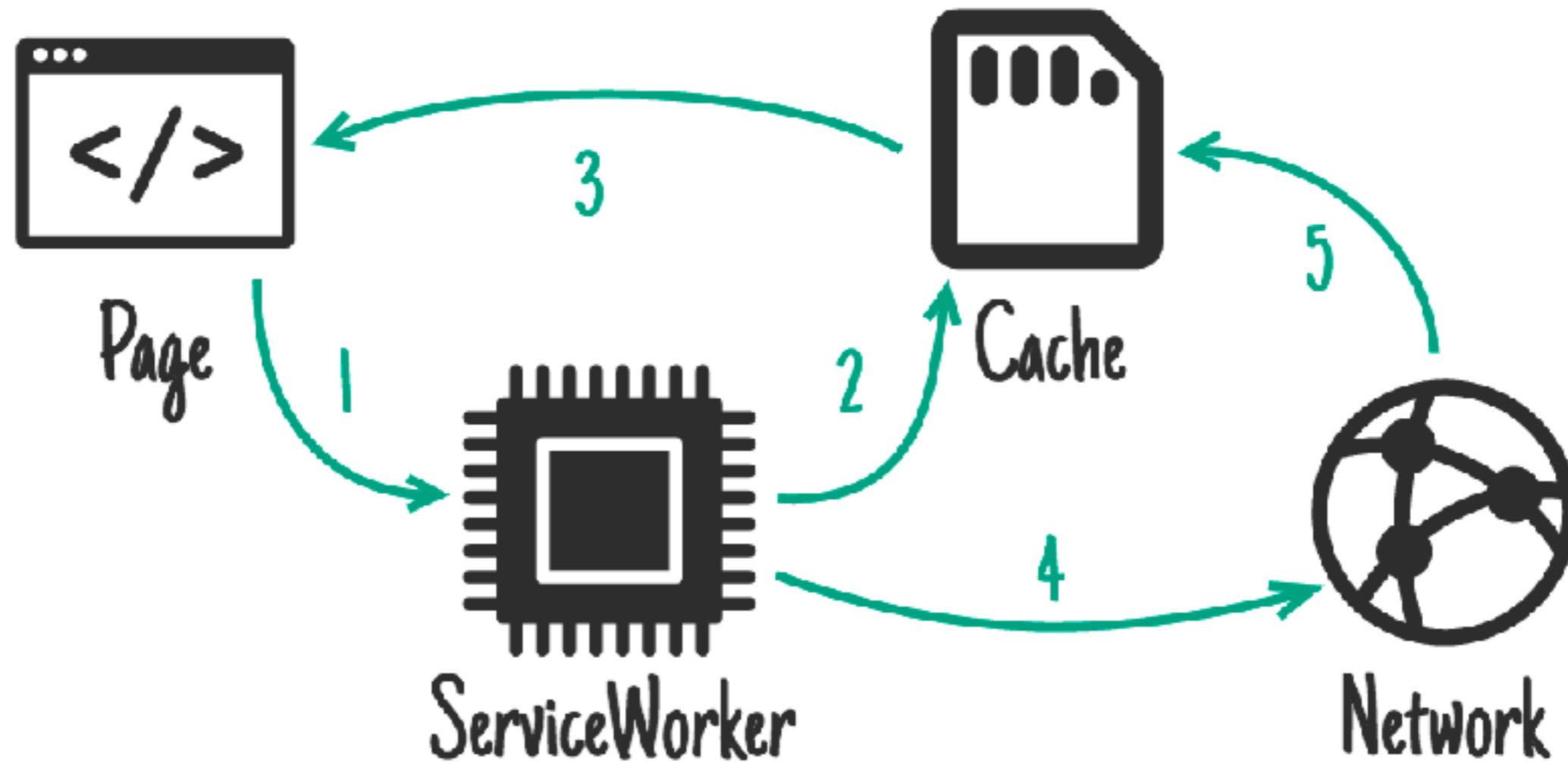
# SWR (Stale While Revalidate)

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

Provide caching mechanism for data



# SWR (Stale While Revalidate)



# ISR



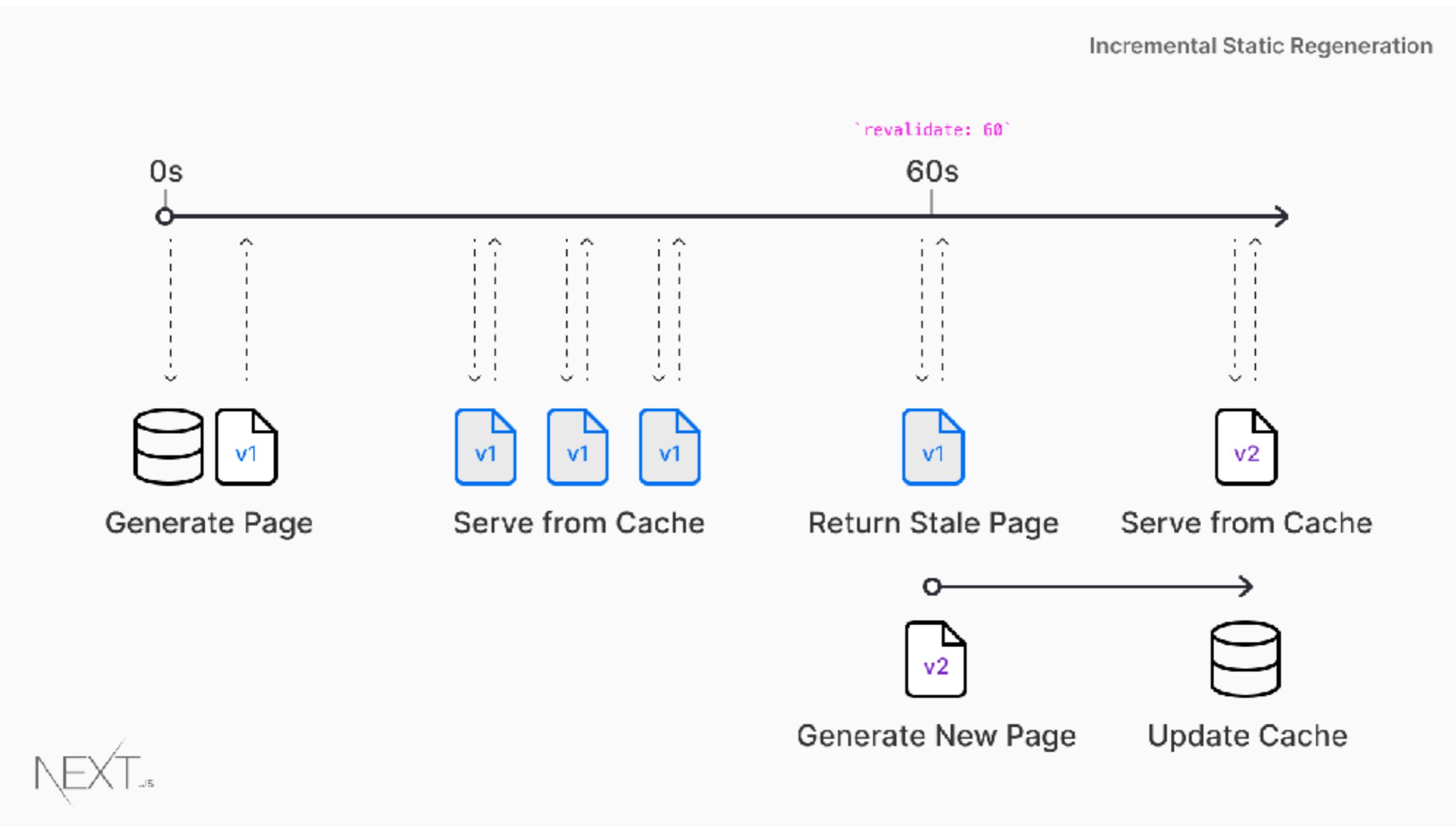
# ISR (Incremental Static Regeneration)

## 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

The screenshot shows the GitHub page for the `nuxt/tailwindcss` repository. The main heading is "Tailwind CSS for your Nuxt Apps." Below it, a paragraph explains that the Tailwind CSS module for Nuxt enables you to set up Tailwind CSS in your Nuxt application in seconds, with many goodies. Two buttons are present: "Get started" and "Star on GitHub". On the right side, there are two code snippets. The first snippet shows the command `npx nuxi@latest module add tailwindcss`. The second snippet shows the configuration in `nuxt.config`:

```
rpx nuxi@latest module add tailwindcss

nuxt.config

export default defineNuxtConfig({
  modules: ['@nuxtjs/tailwindcss'],
  tailwindcss: {
    exposeConfig: true,
    viewer: true,
    // and more...
  }
})
```

<https://tailwindcss.nuxtjs.org/>



# Install tailwind

npm install -D @nuxtjs/tailwindcss

## next.config.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/](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

My Blog

Home Sign In Sign up

# My blog

Demo app with NuxtJS + tailwind.

List of blogs

 Somkiat Pui  
2024/05/02

**Blog title**  
Blog description  
[Read more...](#)

 Somkiat Pui  
2024/05/01

**Blog title**  
Blog description  
[Read more...](#)

Popular Tags

tag 1 tag 2 tag 3  
tag 4 tag 5

© 2024 My blog



# **Design your components**

## **Design your layout**

### **Design your page**



# Layout of page

My Blog

## Header

Home Sign In Sign up

# My blog

Demo app with NuxtJS + tailwind.

List of blogs

 Somkiat Pui  
2024/05/02

**Blog title**  
Blog description  
[Read more...](#)

 Somkiat Pui  
2024/05/01

**Blog title**  
Blog description  
[Read more...](#)

Popular Tags

tag 1 tag 2 tag 3  
tag 4 tag 5

Footer

© 2024 My blog



# List of blogs



# List of blogs

The screenshot shows a web application interface for a blog. At the top, there is a navigation bar with the text "My Blog" on the left and "Home Sign In Sign up" on the right. Below the navigation bar is a green header section with the title "My blog" in large white font and a subtitle "Demo app with NuxtJS + tailwind." in smaller white font. The main content area features a heading "BlogList component" in red. Below this, there is a list of two blog posts, each enclosed in a red dashed box. Each post includes a user profile picture, the author's name "Somkiat Pui", the publication date "2024/05/02" or "2024/05/01", a "Blog title" placeholder, a "Blog description" placeholder, a "Read more..." link, and a green heart icon with a count of 100 or 55. To the right of the posts is a sidebar with a "Popular Tags" section containing five tags: tag1, tag2, tag3, tag4, and tag5. At the bottom of the page is a green footer bar with the copyright notice "© 2024 My blog".

My Blog

Home Sign In Sign up

# My blog

Demo app with NuxtJS + tailwind.

## BlogList component

Somkiat Pui 2024/05/02

Somkiat Pui 2024/05/01

Blog title

Blog description

Read more...

Blog title

Blog description

Read more...

Popular Tags

tag1 tag2 tag3

tag4 tag5

© 2024 My blog



# List of tags

The screenshot shows a blog application interface. At the top, there's a navigation bar with 'My Blog' on the left and 'Home' 'Sign in' 'Sign up' on the right. Below the navigation is a green header section with the title 'My blog' and a subtitle 'Demo app with NuxtJS + tailwind.' In the main content area, there's a list of blog posts under the heading 'List of blogs'. Each post includes a user icon, the author's name 'Somkiat Pui', the publication date, a 'Blog title' link, a 'Blog description' link, and a 'Read more...' link. To the right of each post is a green circular badge with a heart icon and a number (e.g., 100, 55). Below the posts is a sidebar titled 'Popular Tags' containing five tags: tag1, tag2, tag3, tag4, and tag5. A red dashed box highlights a section of the sidebar. To the right of the sidebar, the text 'Tag component' is displayed in red. At the bottom of the page is a green footer bar with the copyright notice '© 2024 My blog'.



# Favorite button

The screenshot shows a blog application interface. At the top, there's a navigation bar with 'My Blog' on the left and 'Home Sign In Sign up' on the right. The main title 'My blog' is displayed prominently in a large white font on a green header. Below it, a subtitle 'Demo app with NuxtJS + tailwind.' is shown. On the left, there's a sidebar with 'List of blogs' and two entries. Each entry includes a user icon, the name 'Somkiat Pui', the date '2024/05/02' or '2024/05/01', a 'Blog title' section, a 'Blog description' section, and a 'Read more...' link. To the right of the sidebar, there are two blog post cards. Each card features a user icon, the name 'Somkiat Pui', the date '2024/05/02' or '2024/05/01', a 'Blog title' section, a 'Blog description' section, and a 'Read more...' link. Next to each blog card is a 'Favorite' button component. The first component has a red dashed border and a green button with a heart icon and the number '100'. The second component has a solid green border and a green button with a heart icon and the number '55'. Both components have 'tag1' and 'tag2' labels below them. At the bottom of the page is a green footer bar with the text '© 2024 My blog'.

My Blog

Home Sign In Sign up

# My blog

Demo app with NuxtJS + tailwind.

List of blogs

Somkiat Pui  
2024/05/02

**Blog title**  
Blog description  
Read more...

Somkiat Pui  
2024/05/01

**Blog title**  
Blog description  
Read more...

Popular Tags

tag 1 tag 2 tag 3  
tag 4 tag 5

Favorite Button component

tag1 tag2

tag1 tag2

© 2024 My blog



# Popular tags

My Blog

Home Sign In Sign up

# My blog

Demo app with NuxtJS + tailwind.

List of blogs

 Somkiat Pui 2024/05/02 ♥ 100

**Blog title**  
Blog description  
[Read more...](#) tag1 tag2

 Somkiat Pui 2024/05/01 ♥ 55

**Blog title**  
Blog description  
[Read more...](#) tag1 tag2

© 2024 My blog

Popular Tags

tag 1 tag 2 tag 3  
tag 4 tag 5

Tag component



# Tips :: Assets

Images from assets or public folder

The screenshot shows a web application titled "My blog" with a green header. The header includes a "My Blog" logo, a navigation bar with "Home", "Sign In", and "Sign up" links, and a title "My blog" with a subtitle "Demo app with NuxtJS + tailwind". Below the header is a "List of blogs" section. It displays two blog entries, each with a thumbnail image (circles with "Sompiket Pui" and a date), a title ("Blog title"), a snippet ("Blog description"), a "Read more..." link, and a heart icon with a count (100 for the top post, 55 for the bottom post). To the right of the posts is a "Popular Tags" sidebar with five tags: tag1, tag2, tag3, tag4, and tag5. At the bottom is a green footer bar with the text "© 2024 My blog".

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



# List of blogs (data flow)



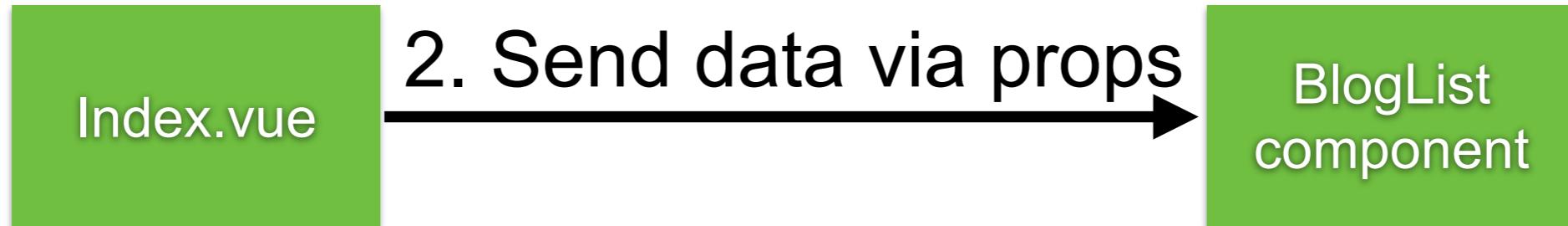
# List of blogs (data flow)



1. Get all blogs from API



# List of blogs (data flow)

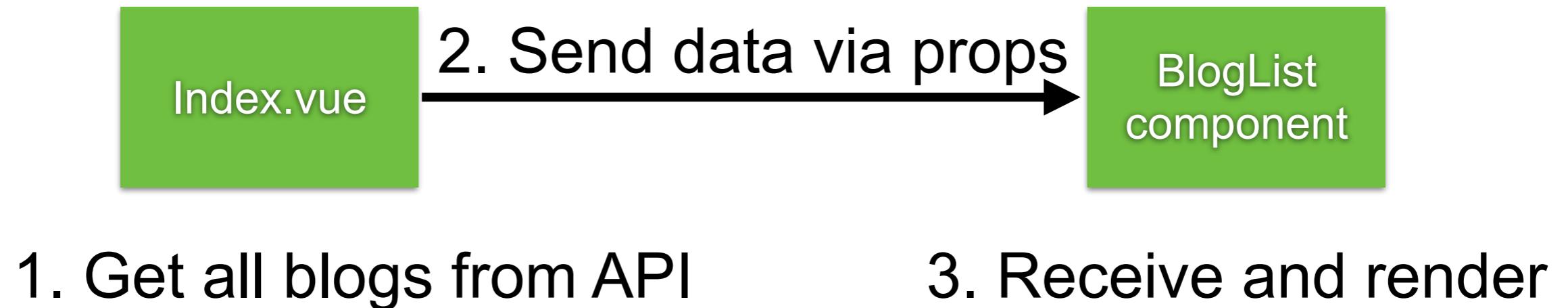


1. Get all blogs from API

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



# List of blogs (data flow)



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



# Example (1)

## Page/index.vue

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

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

## Types/index.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

```
<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>
```

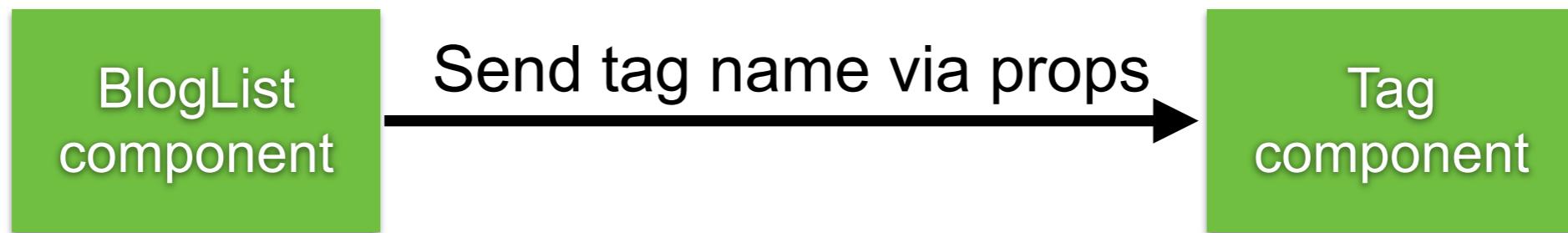
```
<template>
  <div>
    <div v-for="blog in blogs"
         :key="blog.id">

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

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



# List of tags

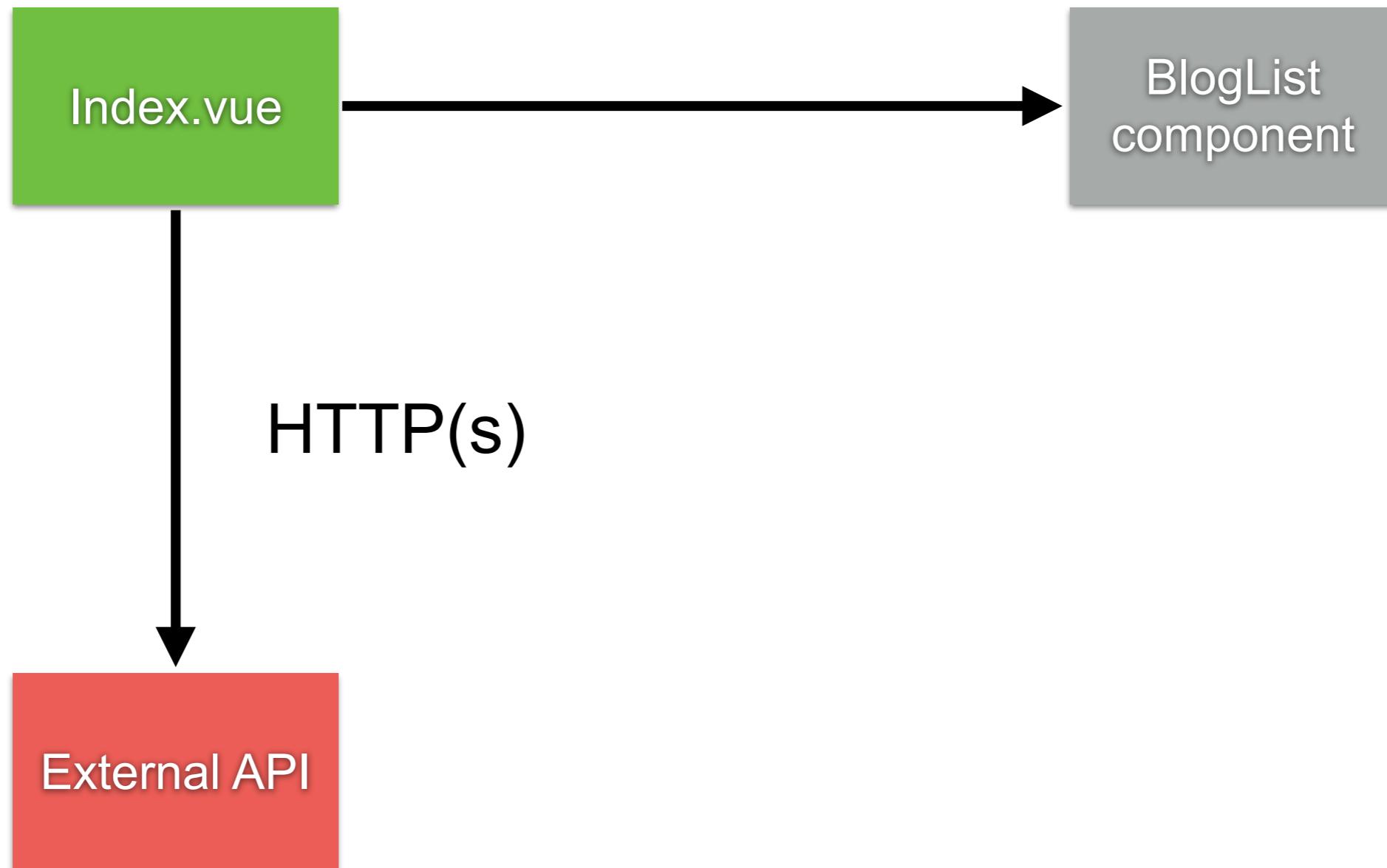


# Read data from APIs



# Data fetching in NuxtJS

## 1. Get all blogs from API



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



# Setup API Server



POSTMAN



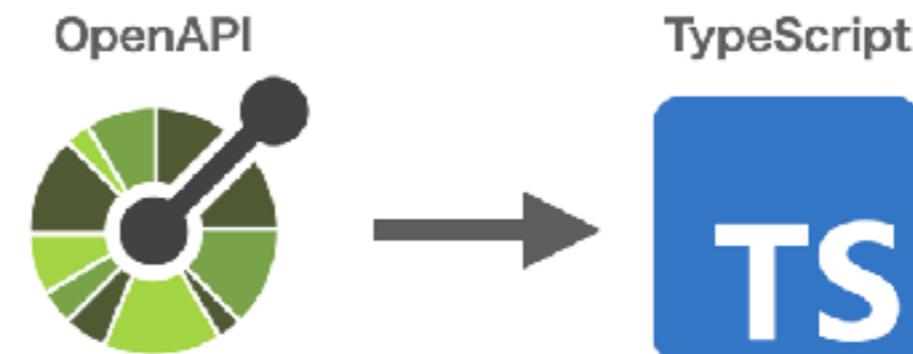
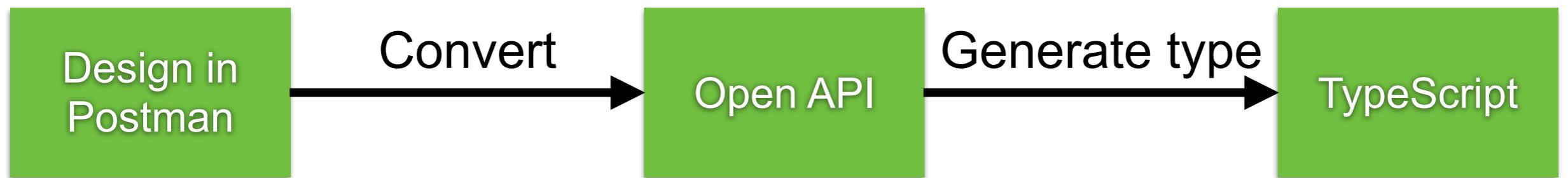
Open API  
Specification



Swagger



# 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

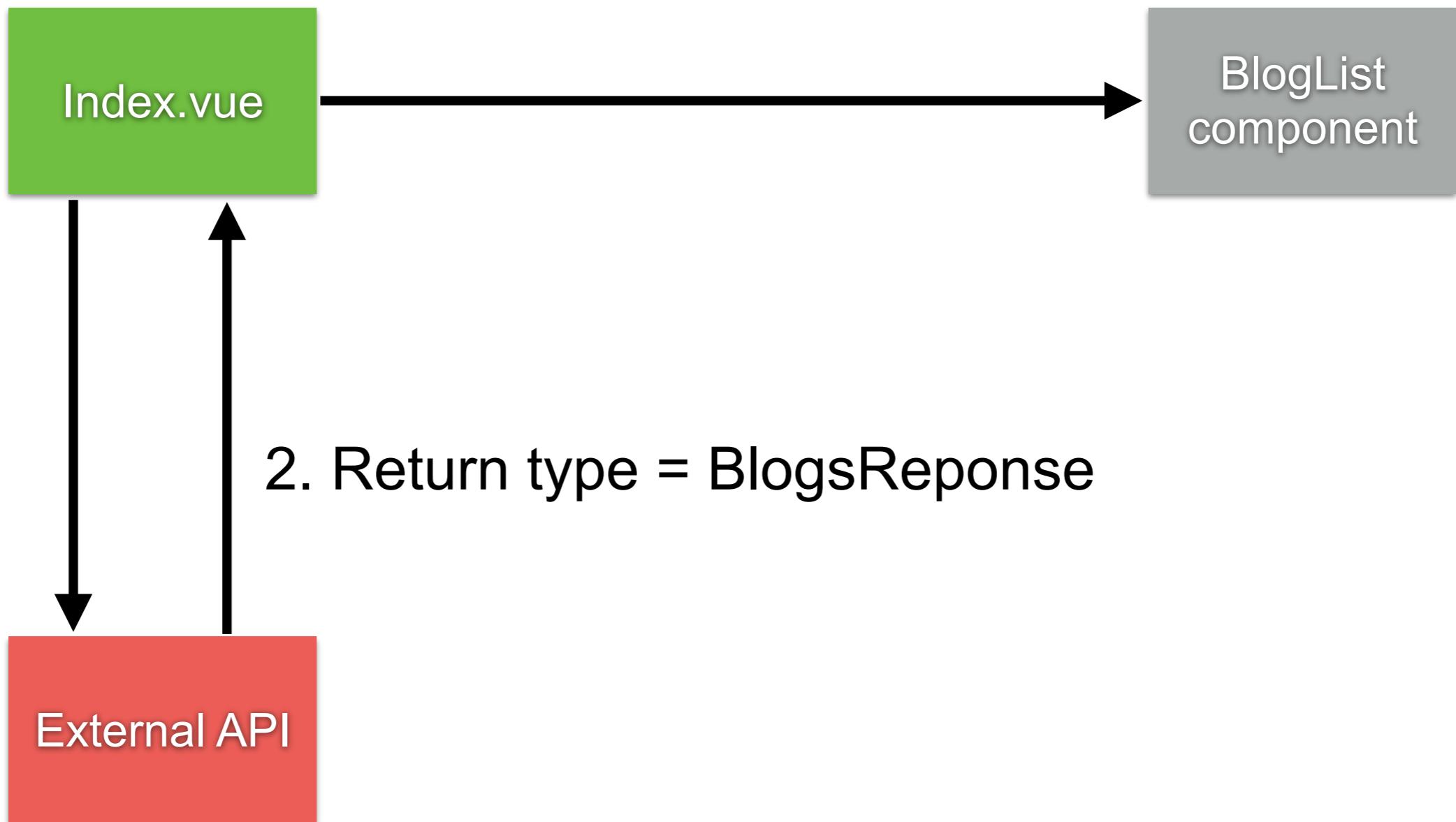
```
import type { components } from "~openapi.gen";
export type BlogsResponse = components["responses"]
  ["BlogsResponse"]["content"]["application/json"];
```

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



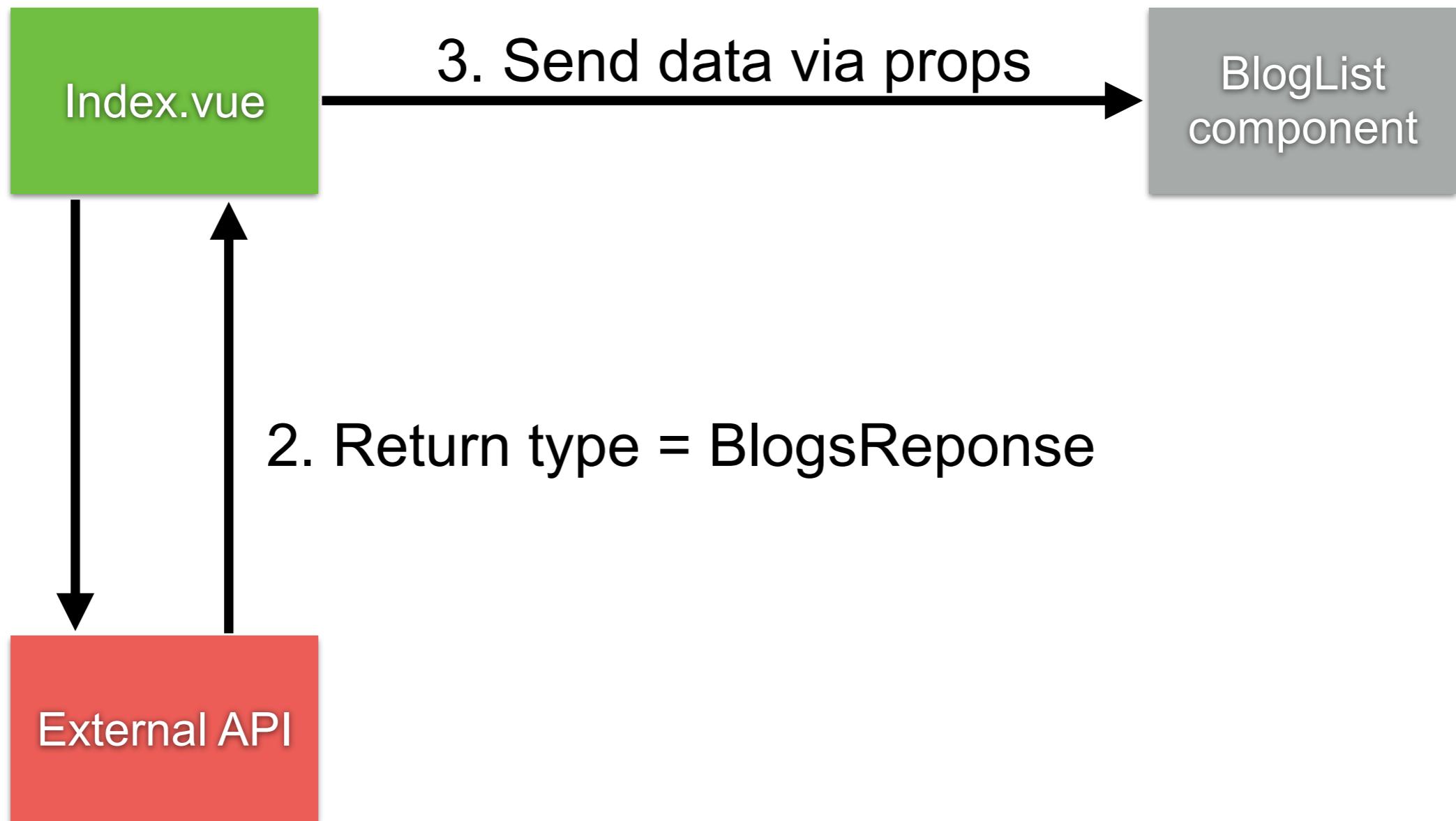
# index.vue

1. Get all blogs from API



# Send data to BlogList component

1. Get all blogs from API



# index.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>
  </div>

  <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

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

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

</script>
```



```
<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

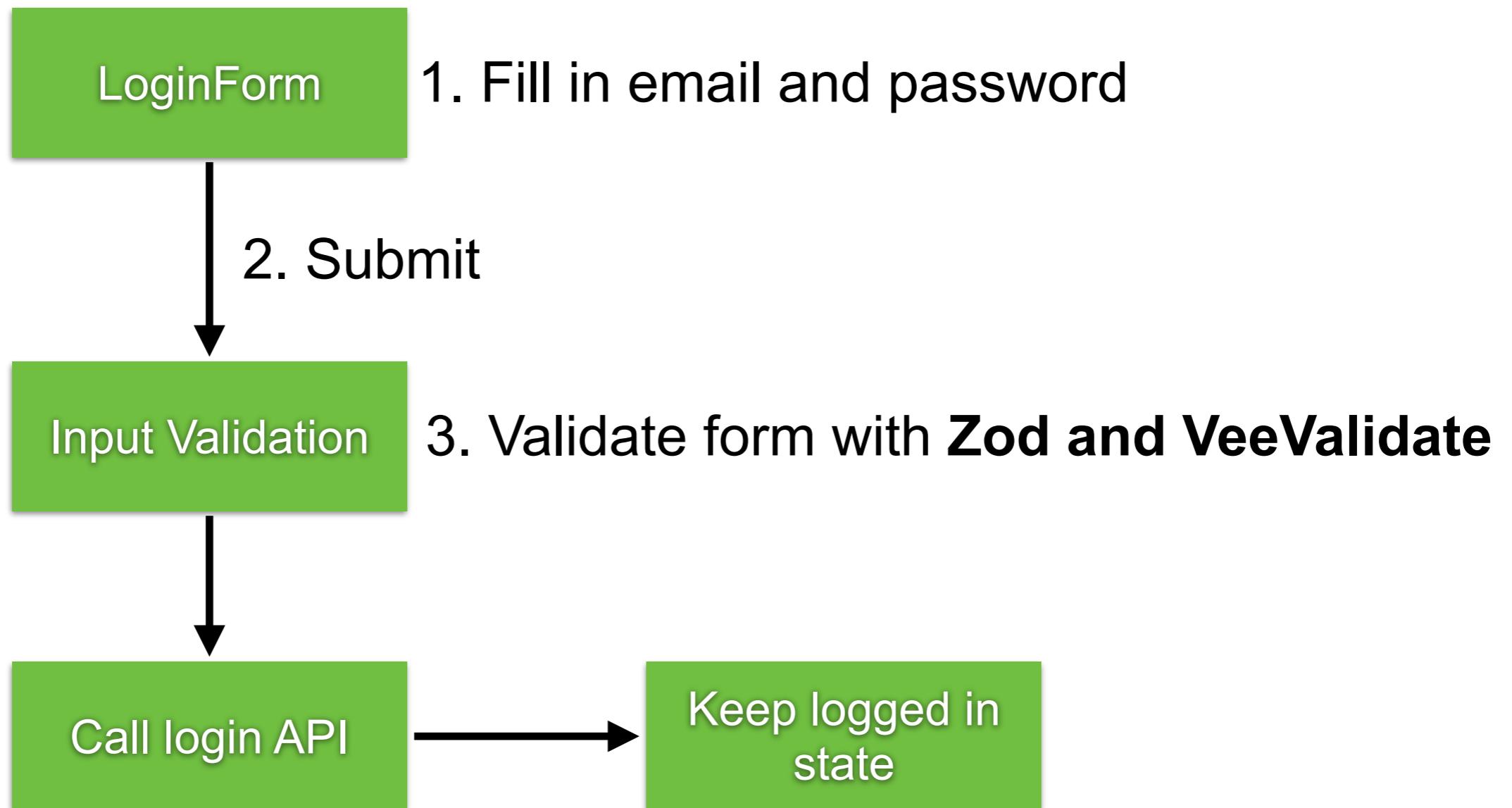
Page = /user/login.vue

Component = LoginForm.vue

The screenshot shows a login interface for a blog. At the top left is the site title "My Blog". On the right, there are links for "Home", "Sign in", and "Sign up". The main content area is a "Sign in" form. It features two input fields: "Email" and "Password", each with an associated error message below it ("Email error" and "Password error"). A green "Sign in" button is positioned at the bottom right of the form area. The entire form is enclosed within a red dotted rectangular border. At the very bottom of the page is a green footer bar containing the copyright notice "© 2024 My blog".



# 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

[Documentation](#) • [Discord](#) • [npm](#) • [deno](#) • [Issues](#) • [@colinhacks](#) • [tRPC](#)

<https://zod.dev/>



# VeeValidate

## VeeValidate Painless Vue forms

VeeValidate is the most popular Vue.js form library. It takes care of value tracking, validation, errors, submissions and more.

[Get Started →](#)[Live Examples](#)

★ 10,547



### Flexible

Offers both declarative components or composable functions API.

vee-validate sets up the foundation for you to form in whatever style you prefer.

### Incremental

vee-validate can do a lot if you let it. Like tracking values, validation, handling submissions and more.

You may opt-in or out to all of these aspects. You are in control of how much form code you write.

### Great DX

Vue.js devtools support.

vee-validate makes debugging forms much easier and less of a wild goose chase every time that form does not submit.

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



# State management

## Pinia

### The intuitive store for Vue.js

Type Safe, Extensible, and Modular by design.  
Forget you are even using a store.

[Get Started](#) [Demo](#)  [Mastering Pinia](#)

 [Watch Video Introduction](#)  [Get the Pinia Cheat Sheet](#)

 **Intuitive**  
Stores are as familiar as components. API designed to let you write well organized stores.

 **Type Safe**  
Types are inferred, which means stores provide you with autocompletion even in JavaScript!

 **Devtools support**  
Pinia hooks into Vue devtools to give you an enhanced development experience in both Vue 2 and Vue 3.



<https://pinia.vuejs.org/>



# Login Page

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



# 1. Login Page

/pages/user/login.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

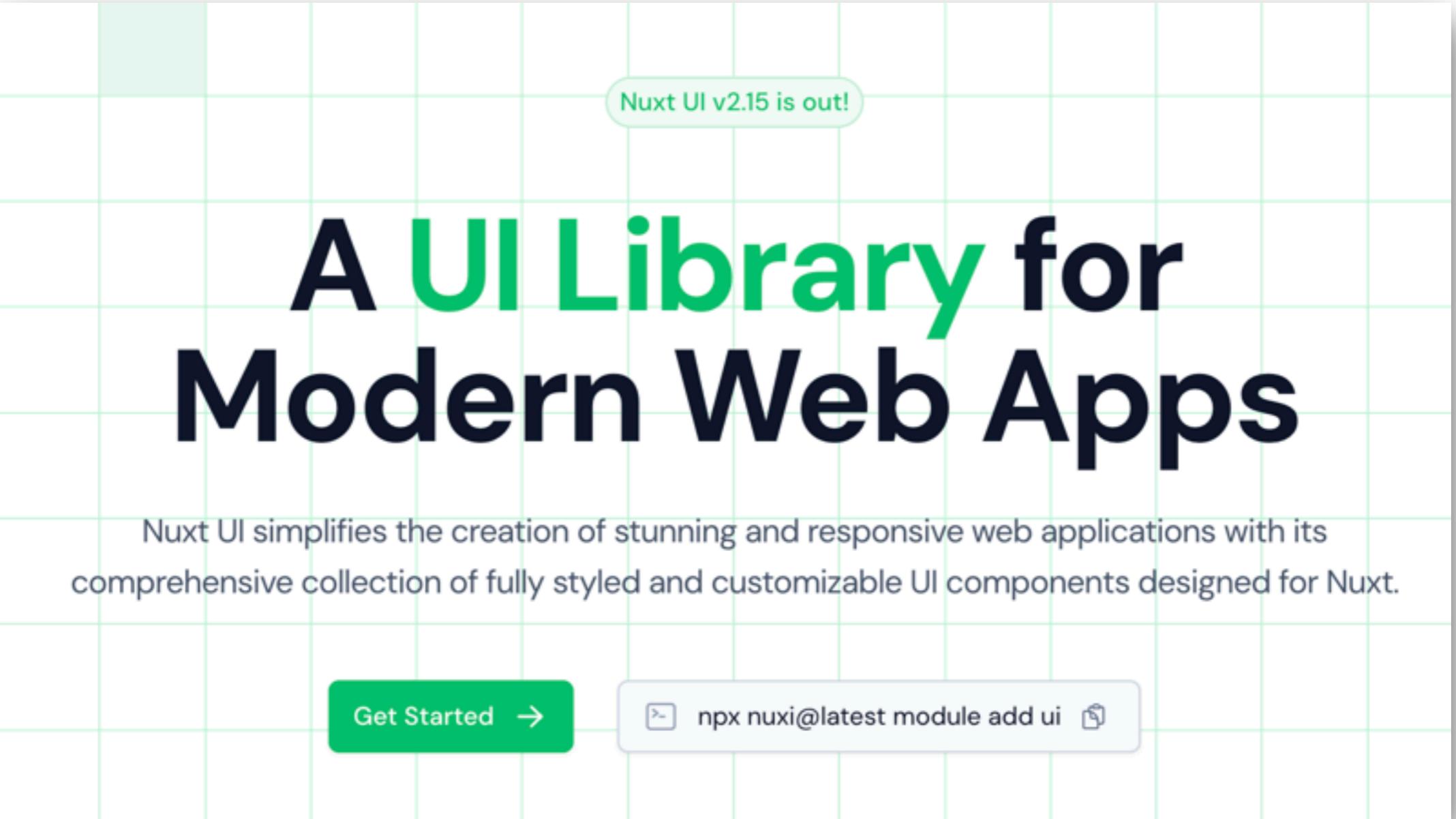
Validate login form after submitted  
Install Zod and VeeValidate module

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

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



# NuxtUI



A screenshot of the Nuxt UI landing page. The page features a light gray background with a faint green grid. At the top right, a green rounded rectangle contains the text "Nuxt UI v2.15 is out!". Below this, the main title "A UI Library for Modern Web Apps" is displayed in large, bold, dark blue and green letters. A descriptive paragraph follows, stating: "Nuxt UI simplifies the creation of stunning and responsive web applications with its comprehensive collection of fully styled and customizable UI components designed for Nuxt." At the bottom left is a green button labeled "Get Started →". To the right is a code snippet in a terminal window: "npx nuxi@latest module add ui".

<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;
  }
});
```



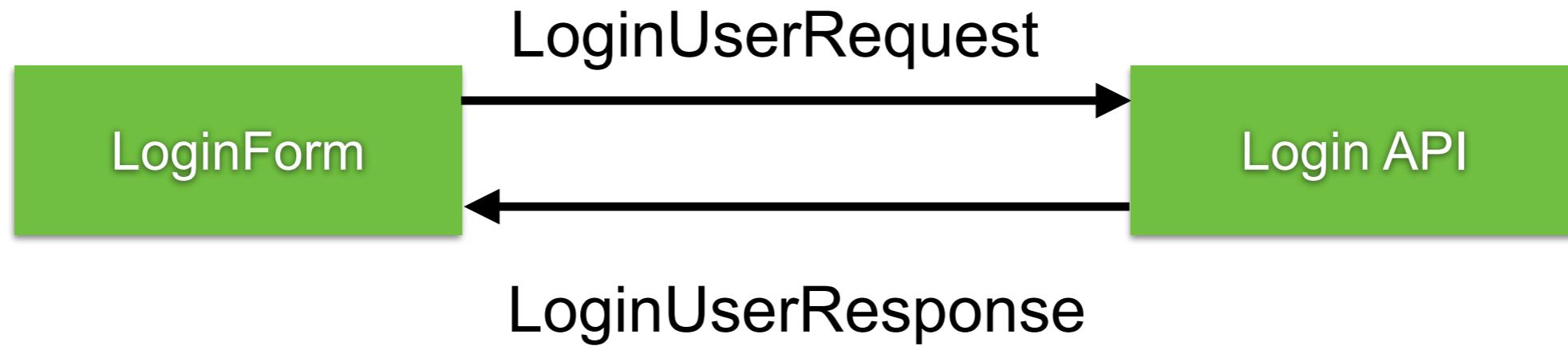
```
<form @submit="onSubmit">

</form>
```



# 6. Call Login API

Create file /lib/api/auth.ts



# 6. Call Login API

Create file /lib/api/auth.ts

```
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

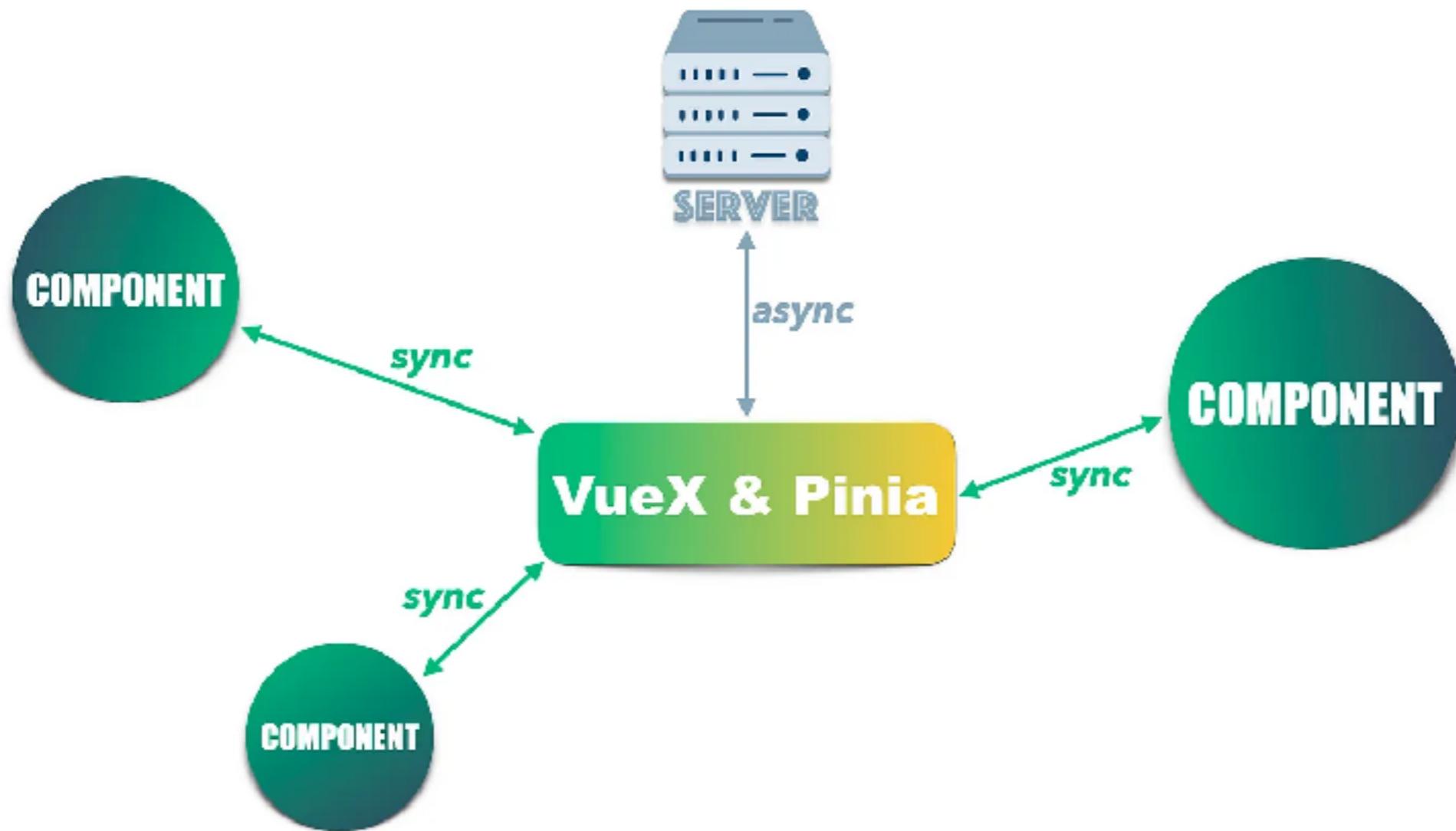
```
import { login } from '~/lib/api/auth';

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

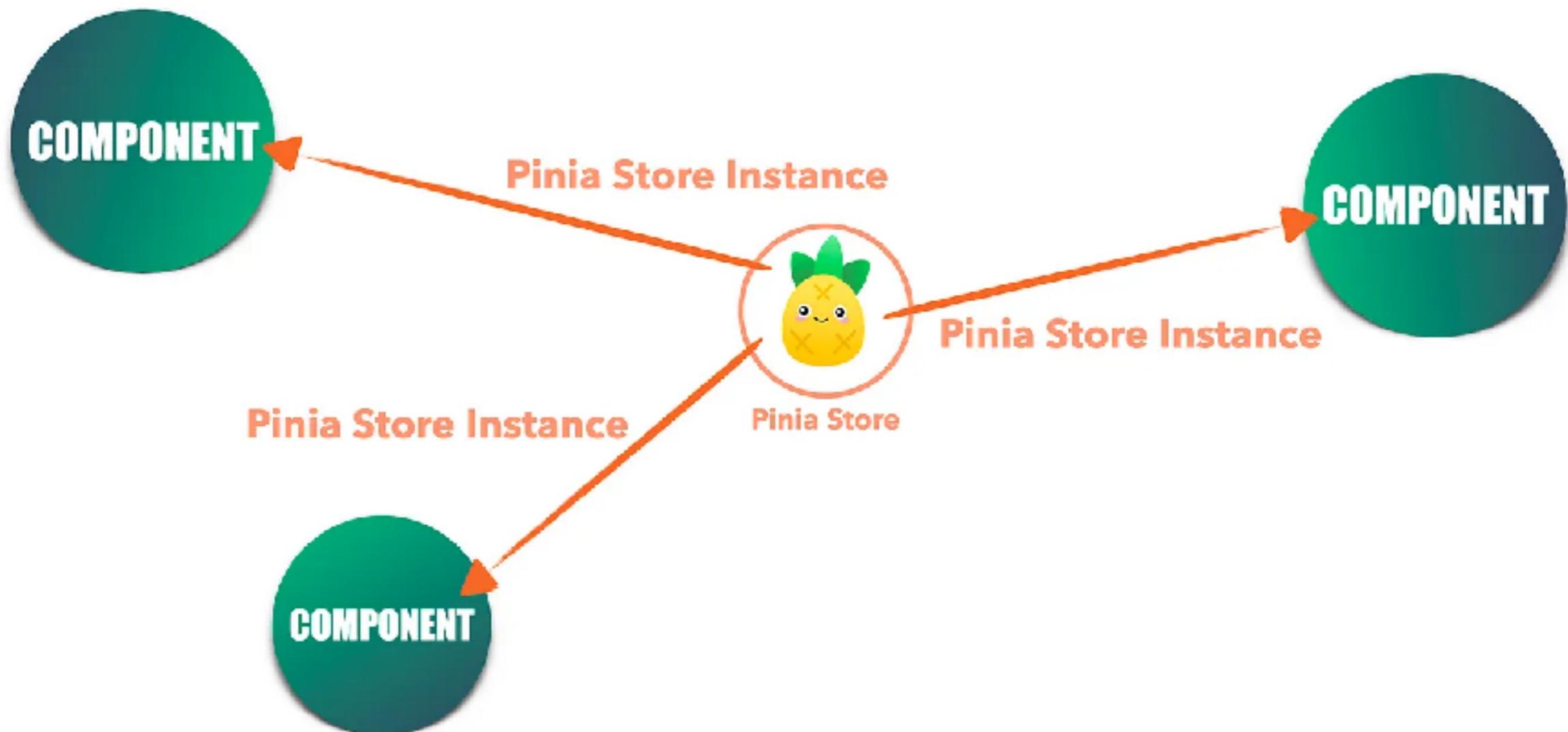
  try {
    // Call Login api
    const response = await login({
      email: values.email,
      password: values.password,
    });
    console.log(response);
    // Keep logged in user in auth store
    // Redirect to home page
    await navigateTo('/');
  } catch (error) {
    alert(error);
  } finally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



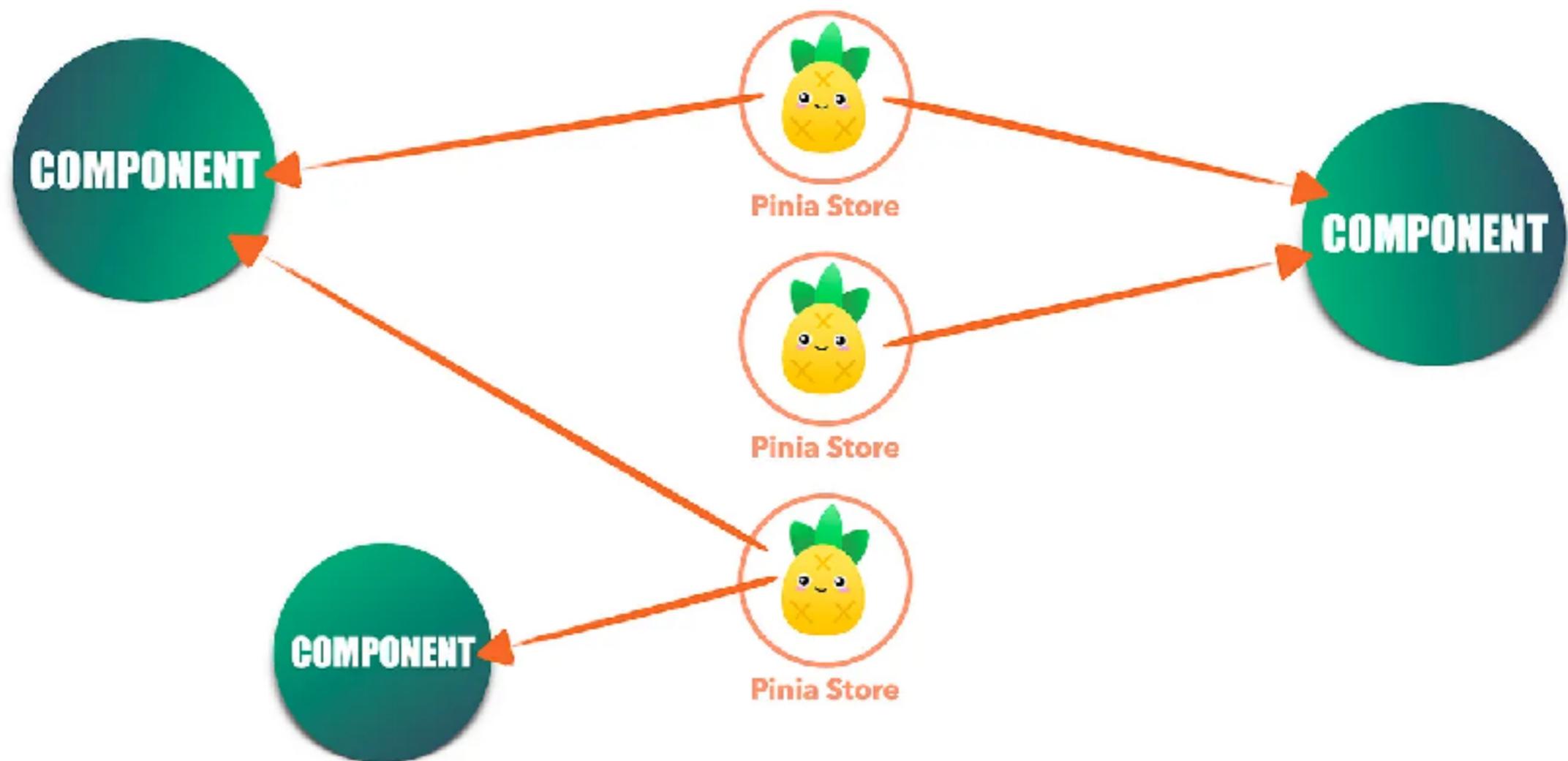
# State management



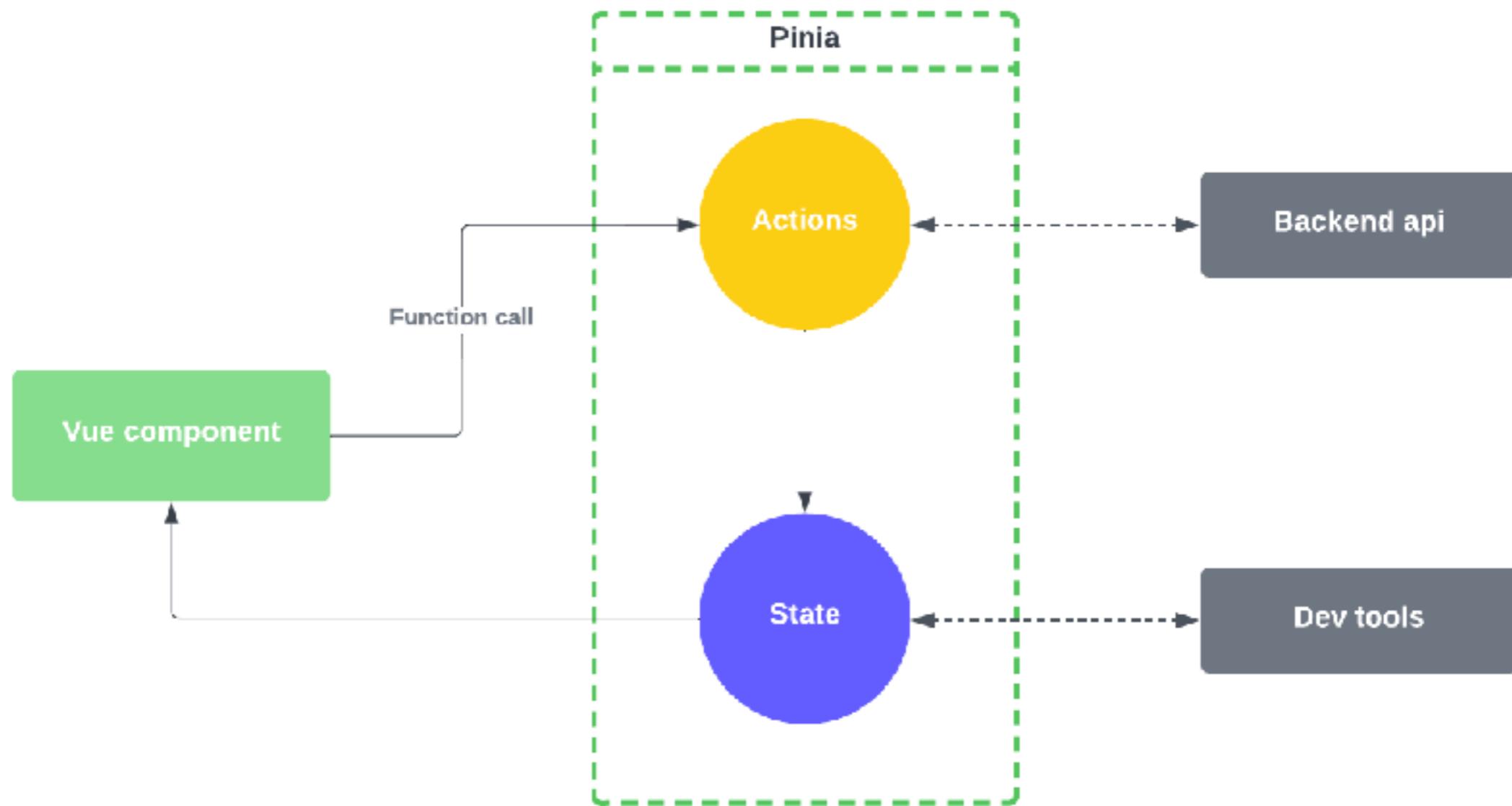
# State management with Pinia



# State management with Pinia

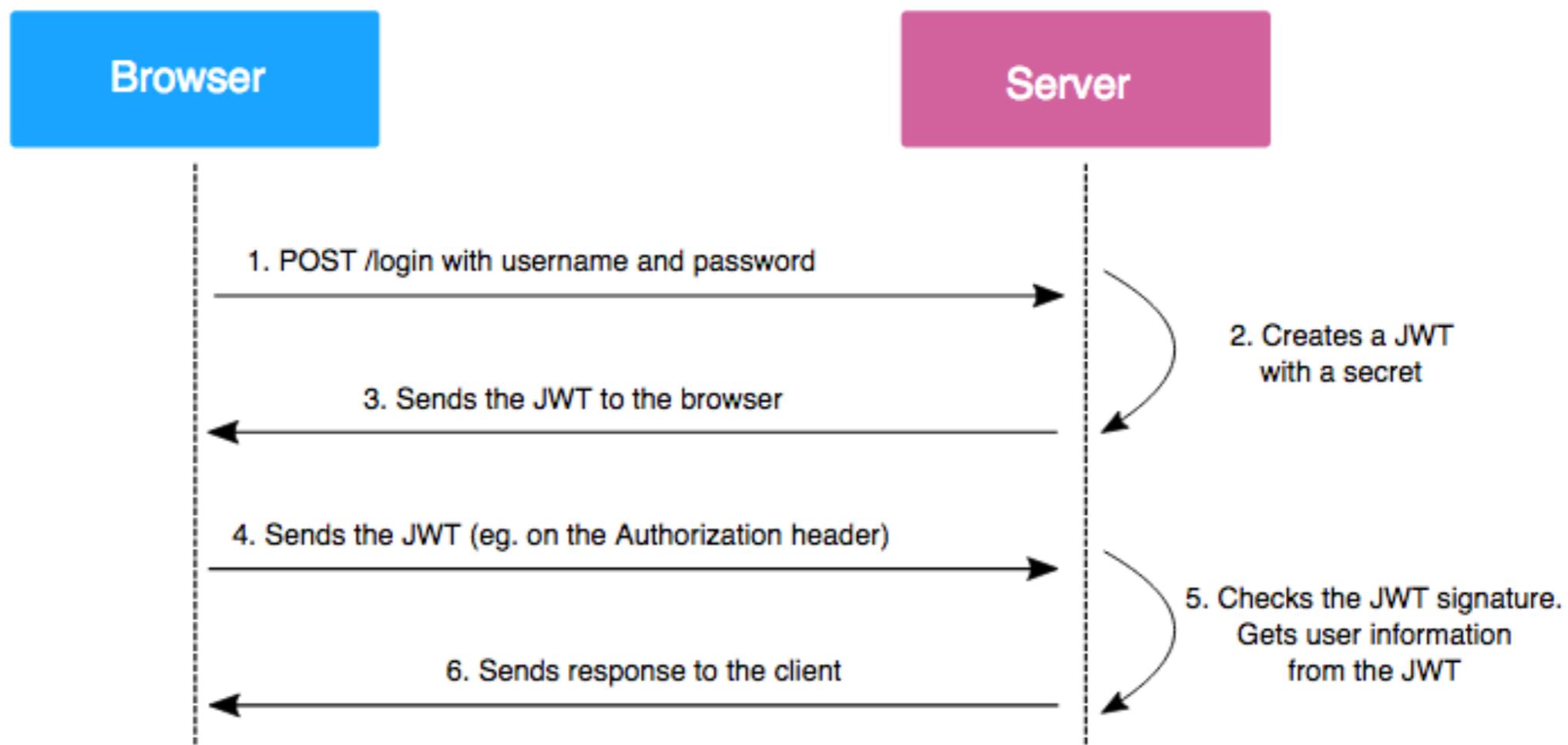


# State management with Pinia

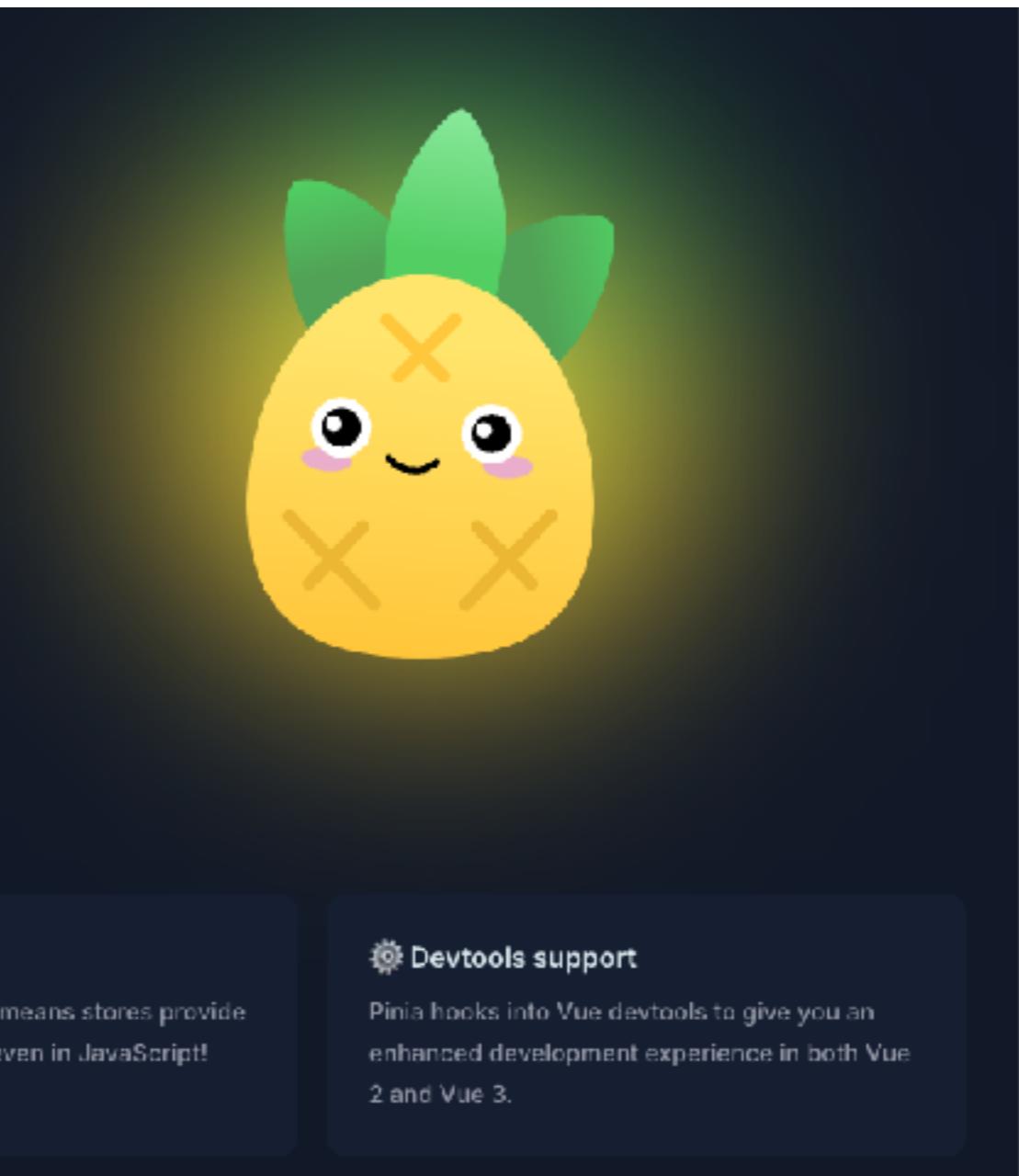


# 7. State management with Pinia

Keep JWT Token from API in web browser



# Pinia with Nuxt



**Pinia**  
**The intuitive store for**  
**Vue.js**

Type Safe, Extensible, and Modular by design.  
Forget you are even using a store.

[Get Started](#) [Demo](#) [!\[\]\(528491cbe031c36fc3d86d3ec410c5b5\_img.jpg\) Mastering Pinia](#)

[!\[\]\(1d852c3738934ef9fd5d3b62fbcc1456\_img.jpg\) Watch Video Introduction](#) [!\[\]\(311b2adcd64fbaea242017f25d1d1463\_img.jpg\) Get the Pinia Cheat Sheet](#)

 **Intuitive**  
Stores are as familiar as components. API designed to let you write well organized stores.

 **Type Safe**  
Types are inferred, which means stores provide you with autocompletion even in JavaScript!

 **Devtools support**  
Pinia hooks into Vue devtools to give you an enhanced development experience in both Vue 2 and Vue 3.

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



# Pinia Persist State

 Nuxt v3.11      Docs ▾      Integrations ▾      Resources ▾      Showcase      Enterprise ▾      Blog

Modules > Extensions > [@pinia-plugin-persistedstate/nuxt](#)

 **pinia-plugin-persistedstate**

Configurable persistence and rehydration of Pinia stores.

⬇️ 190.6K downloads • ⭐ 1.8K stars |  prazdevs

---

**@pinia-plugin-persistedstate/nuxt**

| *Nuxt 3 module*

<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

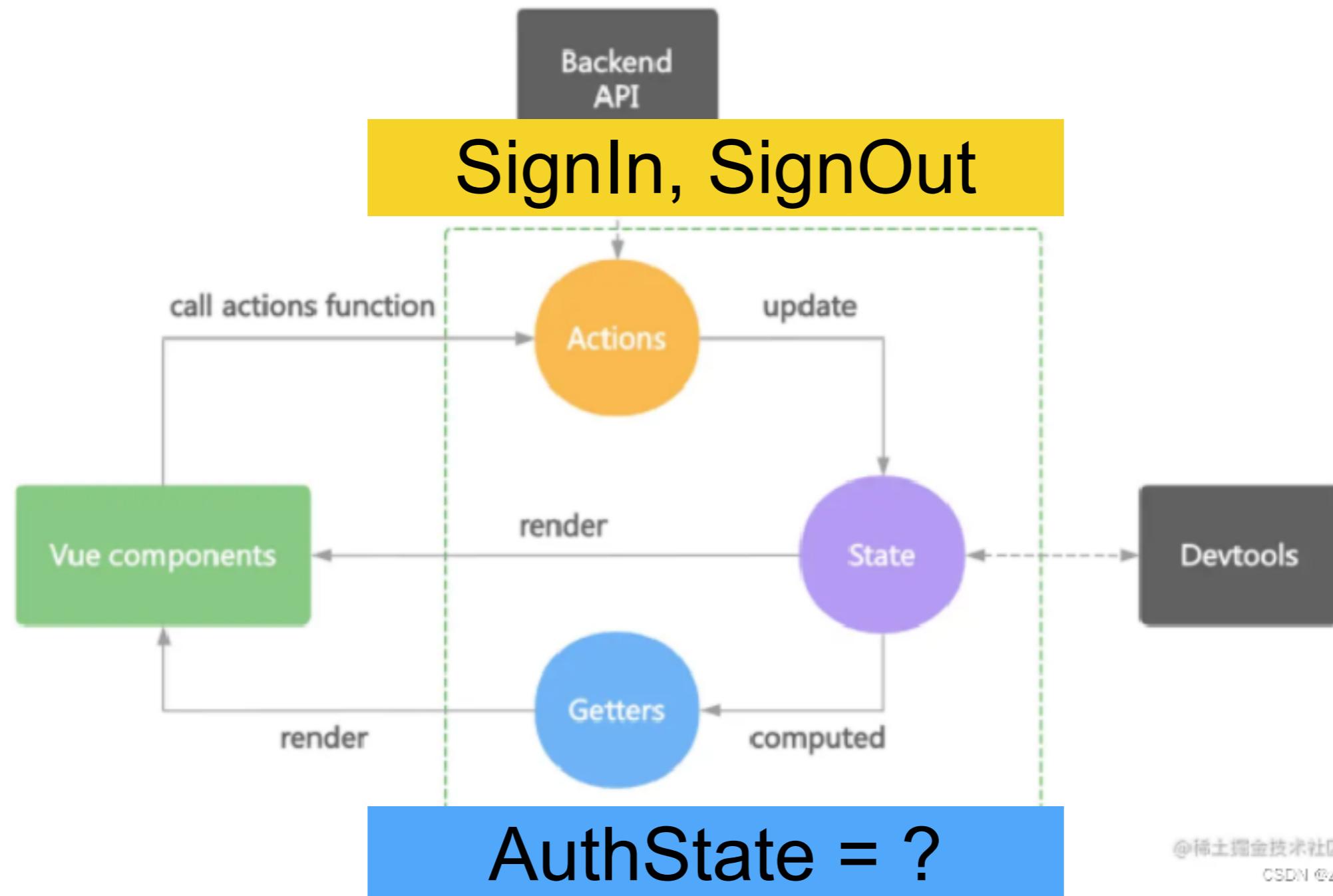
```
export default defineNuxtConfig({
  devtools: { enabled: true },
  typescript: { strict: true },
  modules: [
    '@nuxtjs/tailwindcss',
    '@pinia/nuxt',
    '@pinia-plugin-persistedstate/nuxt',
  ],
  css: ['~/assets/css/global.css'],
})
```



# **Auth Store to keep JWT Token**



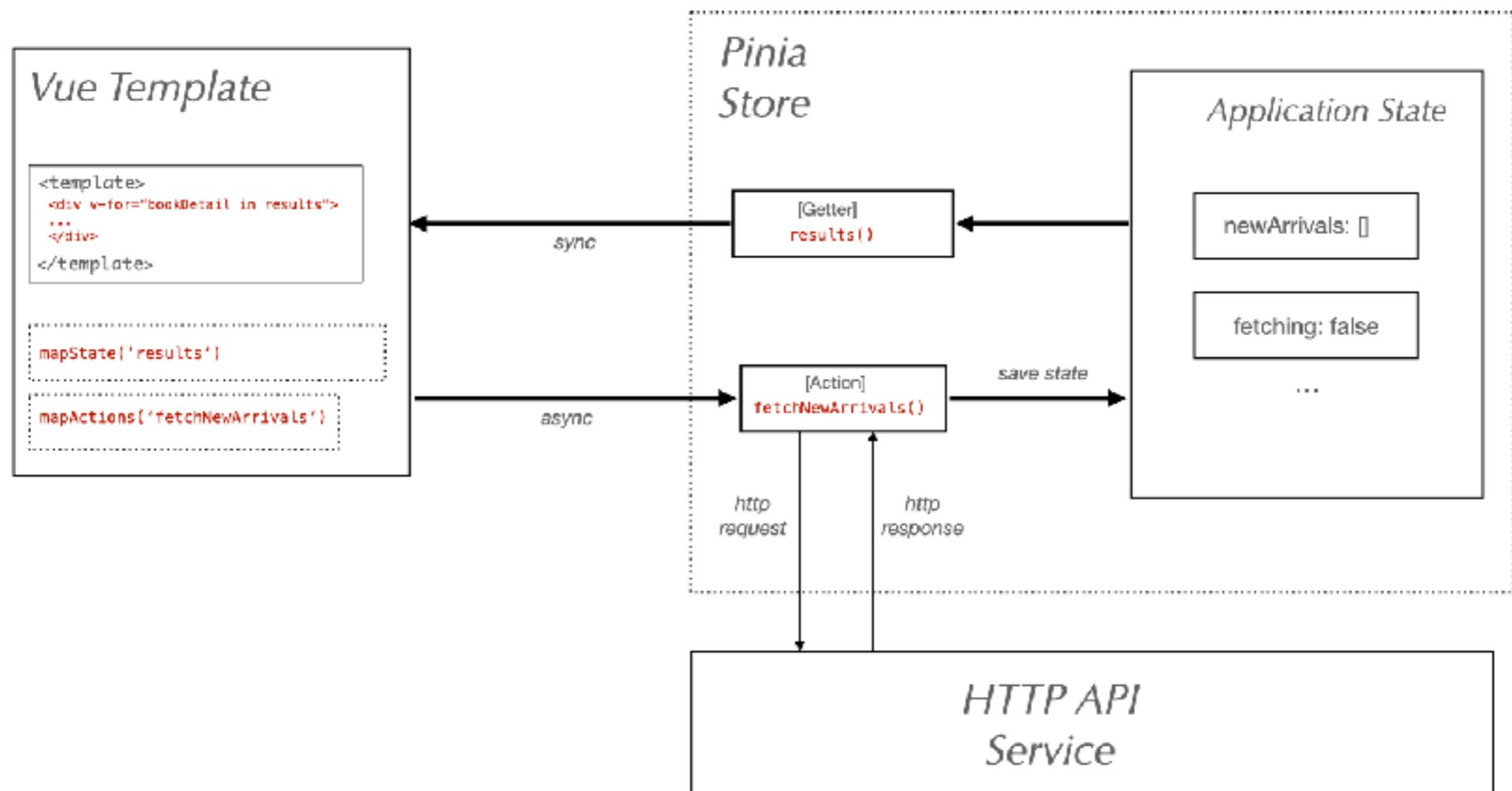
# State management with Pinia



@稀土掘金技术社区  
CSDN @zhooson



# State management with Pinia



# Auth Store to keep JWT Token

/stores/auth.ts

## Declare states

```
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;

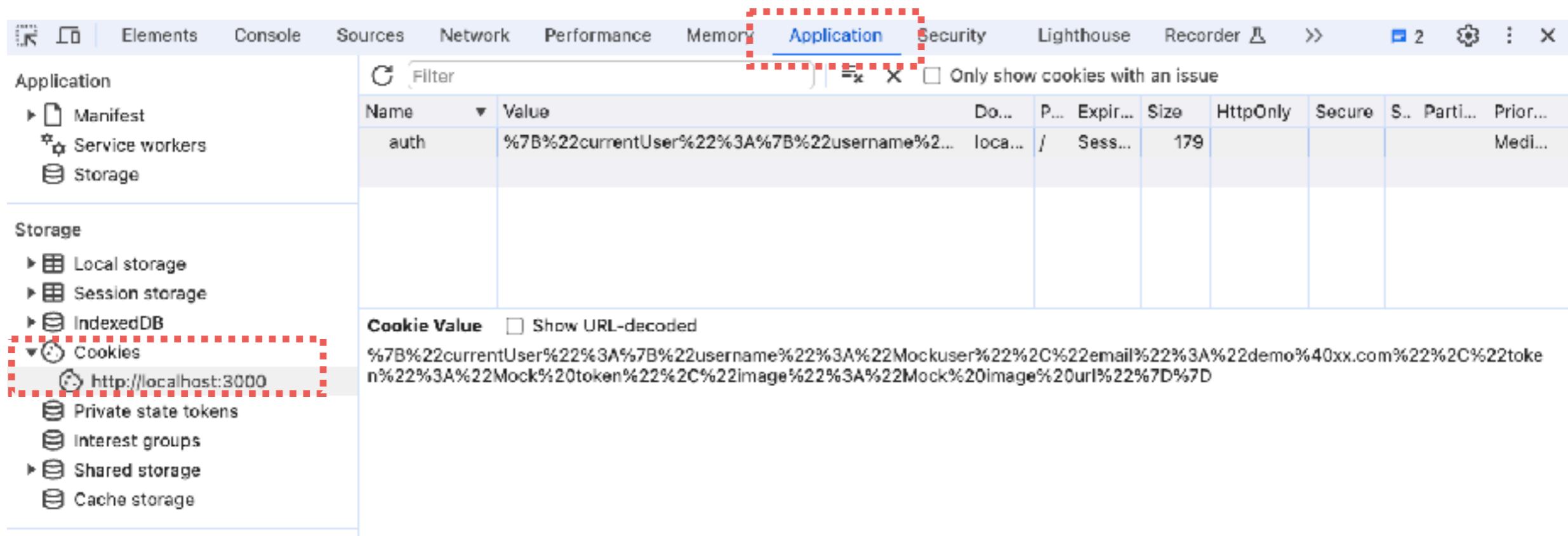
  try {
    // Call Login api
    const response = await login({
      email: values.email,
      password: values.password,
    });

    // 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 !!



The screenshot shows the Chrome DevTools interface with the Application tab selected. On the left, the Storage section is expanded, showing Local storage, Session storage, IndexedDB, Cookies, and other options like Private state tokens and Interest groups. The Cookies section is highlighted with a red dashed box. A cookie named 'auth' is listed in the main pane, with its value decoded as '%7B%22currentUser%22%3A%7B%22username%22%3A%22Mockuser%22%2C%22email%22%3A%22demo%40xx.com%22%2C%22token%22%3A%22Mock%20token%22%2C%22image%22%3A%22Mock%20image%20url%22%7D%7D'. There is also a checkbox for 'Show URL-decoded'.

Name	Value	Do...	P...	Expir...	Size	HttpOnly	Secure	S...	Parti...	Prior...
auth	%7B%22currentUser%22%3A%7B%22username%22%3A%22Mockuser%22%2C%22email%22%3A%22demo%40xx.com%22%2C%22token%22%3A%22Mock%20token%22%2C%22image%22%3A%22Mock%20image%20url%22%7D%7D	loca...	/	Sess...	179					Medi...



# 9. Use auth state in menu

/components/MyHeader.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>
```

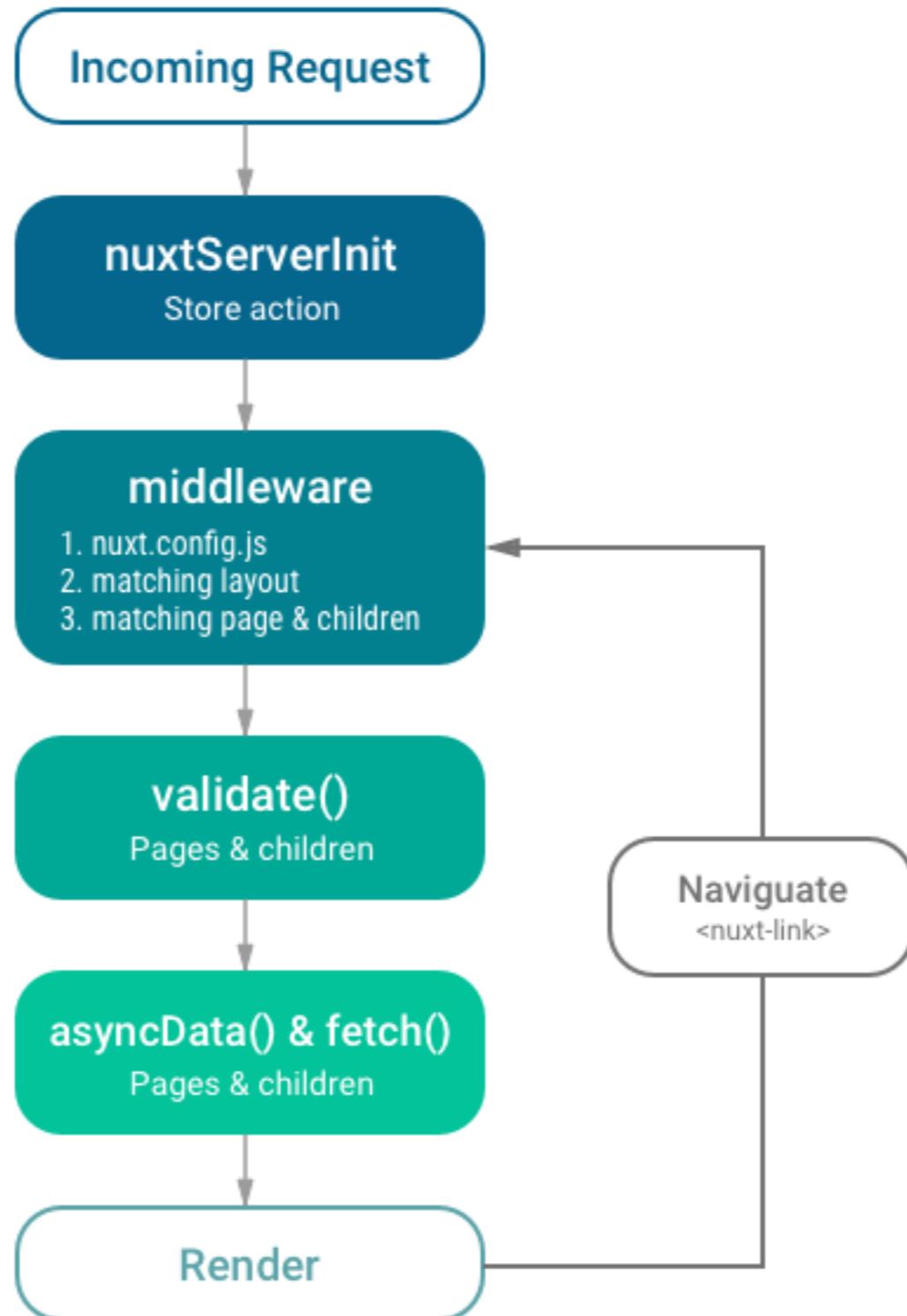


# Working with middleware

<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

```
<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



**nuxt/auth**

Zero-boilerplate authentication  
support for Nuxt.js!



<https://auth.nuxtjs.org/>



# Register Page

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



# Register page

My Blog

Home Sign in Sign up

## Sign up

Have an account?

Username

Email

Password

Sign in

© 2024 My blog

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



# Testing



# Testing

Unit testing

Component  
testing

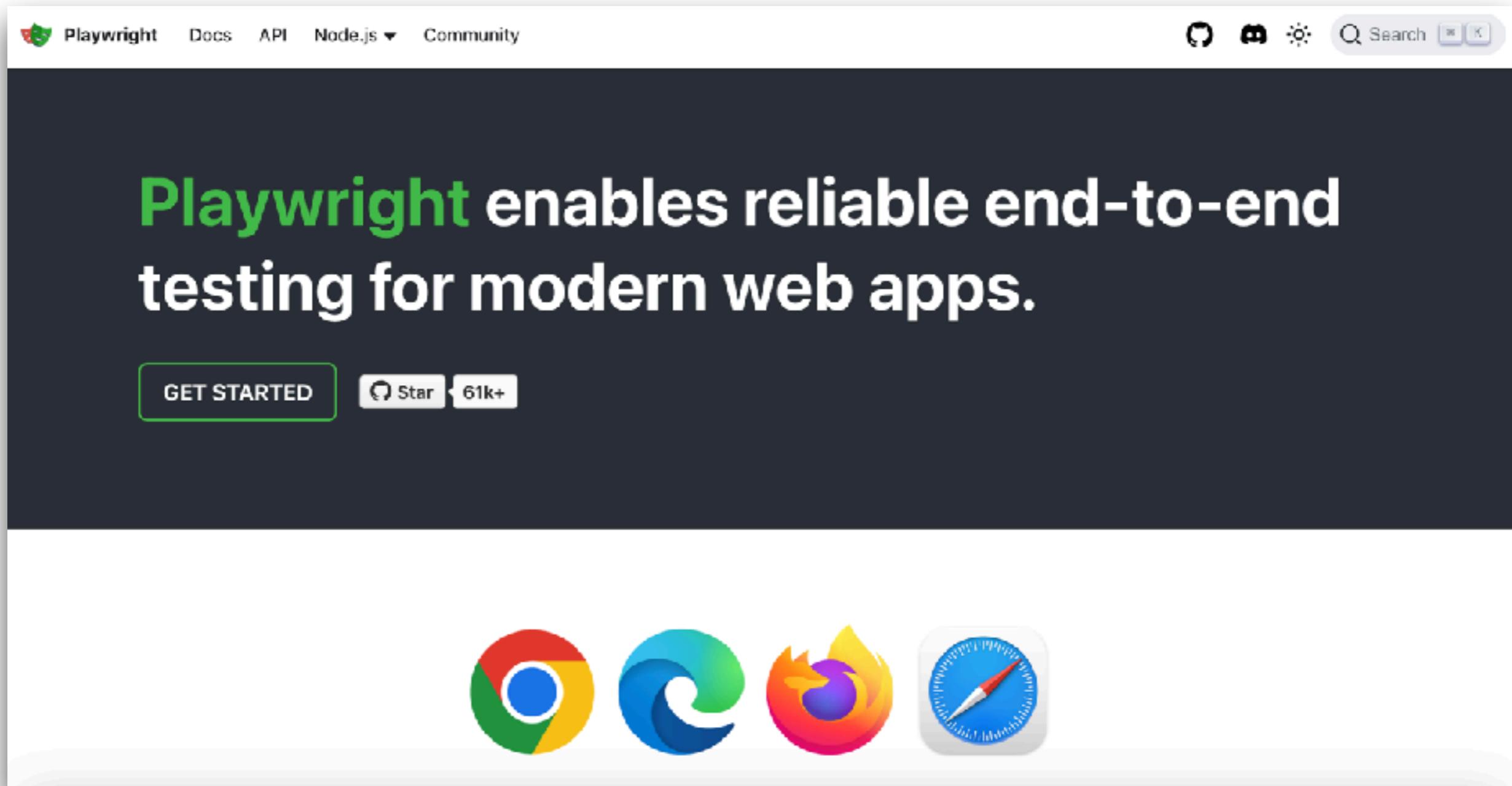
UI testing

Screenshot  
testing

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



# Playwright



The screenshot shows the official website for Playwright. At the top, there's a navigation bar with links for "Playwright", "Docs", "API", "Node.js ▾", and "Community". To the right of the navigation are icons for GitHub, npm, and a search bar. The main title "Playwright" is at the top in large black letters. Below it is a large green sub-headline: "Playwright enables reliable end-to-end testing for modern web apps." Underneath this, there's a green button labeled "GET STARTED", a "Star" button with "61k+", and four browser logos for Chrome, Edge, Firefox, and Safari. The background of the main section is dark.

<https://playwright.dev/>



# Cypress

[Product](#)[Docs](#)[Community](#)[Company](#)[Pricing](#)[Contact Sales >](#)[Log In >](#)[Install >](#)

## Test. Automate. Accelerate.

With Cypress, you can easily create tests for your modern web applications, debug them visually, and automatically run them in your continuous integration builds.

[`npm install cypress`](#) [Documentation](#)

<https://www.cypress.io/>



# Deployment

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



# Q/A

