shopware®

Quick Start – Composable Frontends

Topics

- What to expect, What not to expect?
- Templates & How to setup
- Packages & Composables
- Demo Store explained
- How to organize your projects?
- Missing the Eco-System?
- Roadmap, Future and Links

What to expect?

Custom Template

NUXT 3

VUE 3

TypeScript

Great Developer Experience

UnoCss

Tailwind

store **API** first

Nitro

Vite

Vitest

What NOT to expect?

Themes

Update stress

Compatibility problems

Out-of-the box-solution

Marketplace

Complete feature set

API support everywhere

What are good indicators for choosing Composable Frontends?

- Developer-Team has JS/TypeScript, Nuxt/Vue or React experience
- Developer-Team wants to own the frontend
- Custom corporate design and user experience
- Starting with a sub-store (sub-set of products)
- Payment Provider supports headless approach
- Needed Plugins/Apps rely on the store API

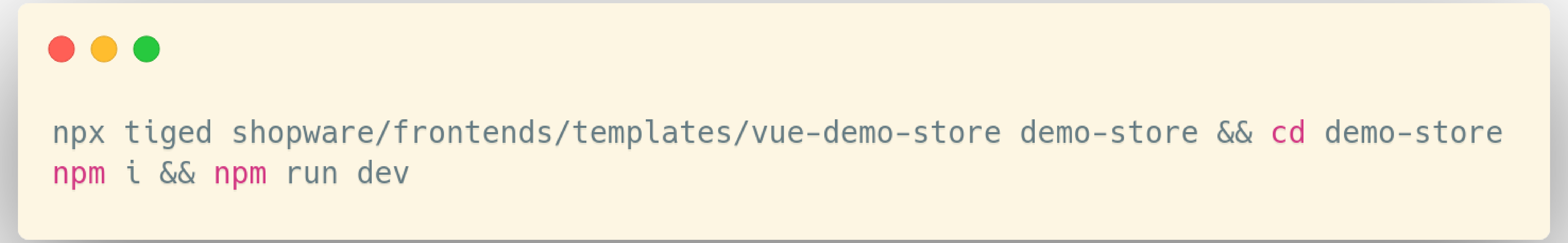
It's not for everyone, that's fine.

Templates

- astro (Astro + Vue)
- vue-blank (Vue + NUXT+ TypeScript)
- vue-vite-blank (Vue + Vite + TypeScript)
- vue-demo-store
  (Vue + NUXT + TypeScript + UnoCSS + Tailwind)
- shopware-vercel-commerce
  (React + Next.js + Tailwind, no Checkout)

How to setup

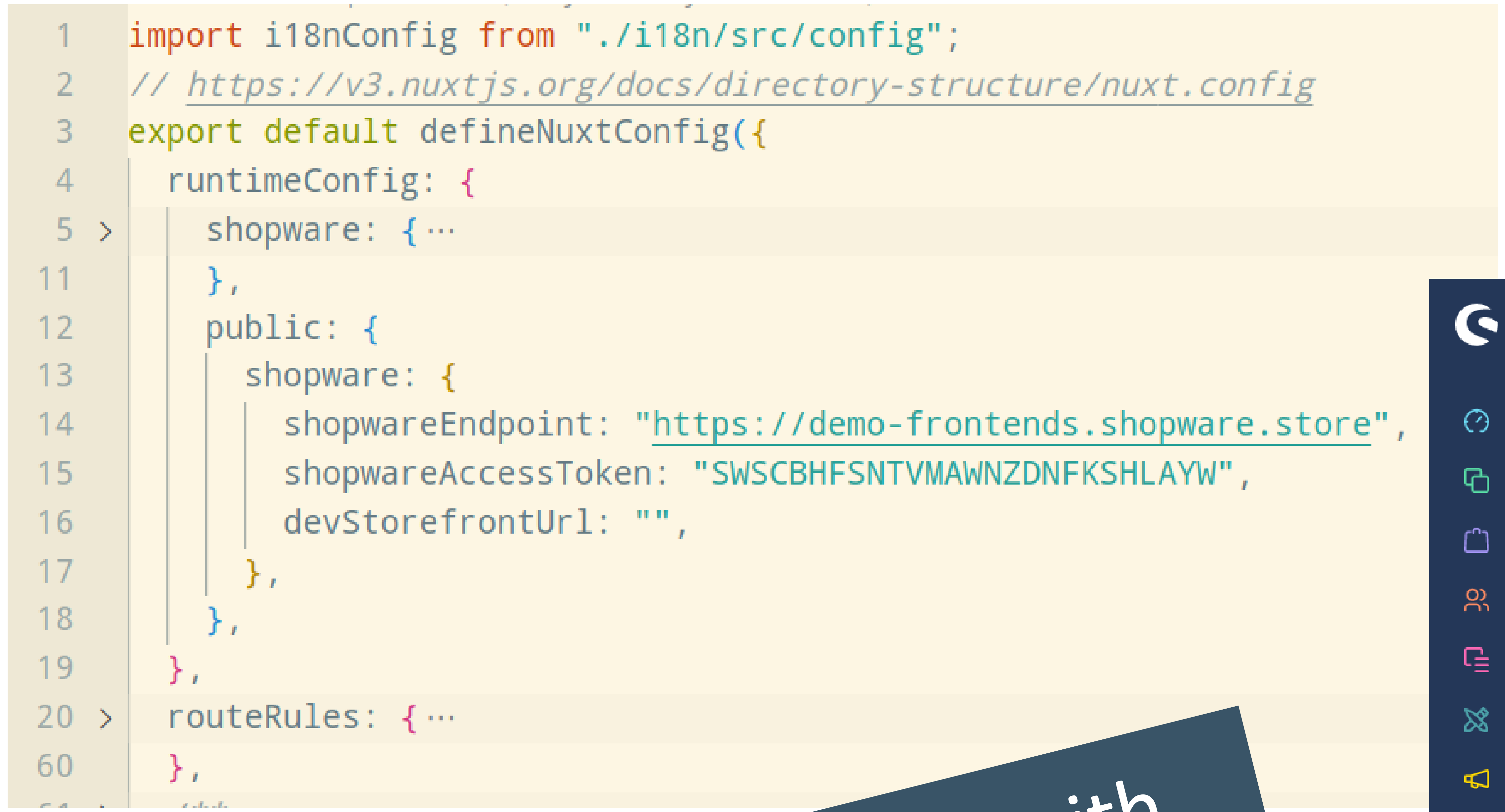
If you just want to play with the **vue-demo-store**:



```
npx tiged shopware/frontends/templates/vue-demo-store demo-store && cd demo-store
npm i && npm run dev
```

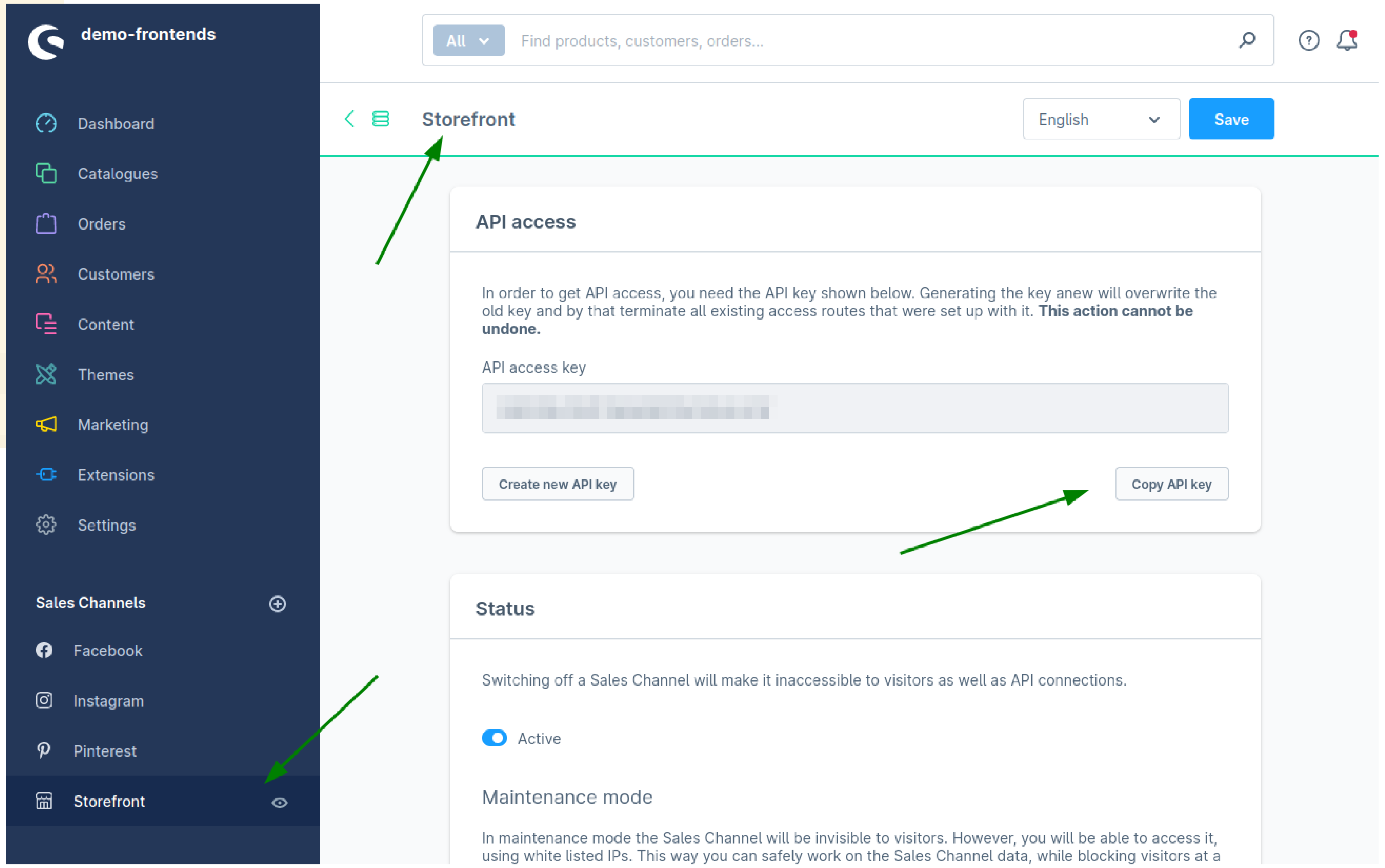
You do not need to change any config file.
It uses a Shopware cloud instance for the store API calls.
You can also test it on ⚡StackBlitz.

How to setup local



```
import i18nConfig from "./i18n/src/config";
// https://v3.nuxtjs.org/docs/directory-structure/nuxt.config
export default defineNuxtConfig({
  runtimeConfig: {
    shopware: {⋯
    },
    public: {
      shopware: {
        shopwareEndpoint: "https://demo-frontends.shopware.store",
        shopwareAccessToken: "SWSCBHFSNTVMAWNZDNFKSHLAYW",
        devStorefrontUrl: "",
      },
    },
  },
  routeRules: {⋯
  },
```

Works great with DEVENV, DDEV



Storefront

API access

In order to get API access, you need the API key shown below. Generating the key anew will overwrite the old key and by that terminate all existing access routes that were set up with it. **This action cannot be undone.**

API access key: [redacted]

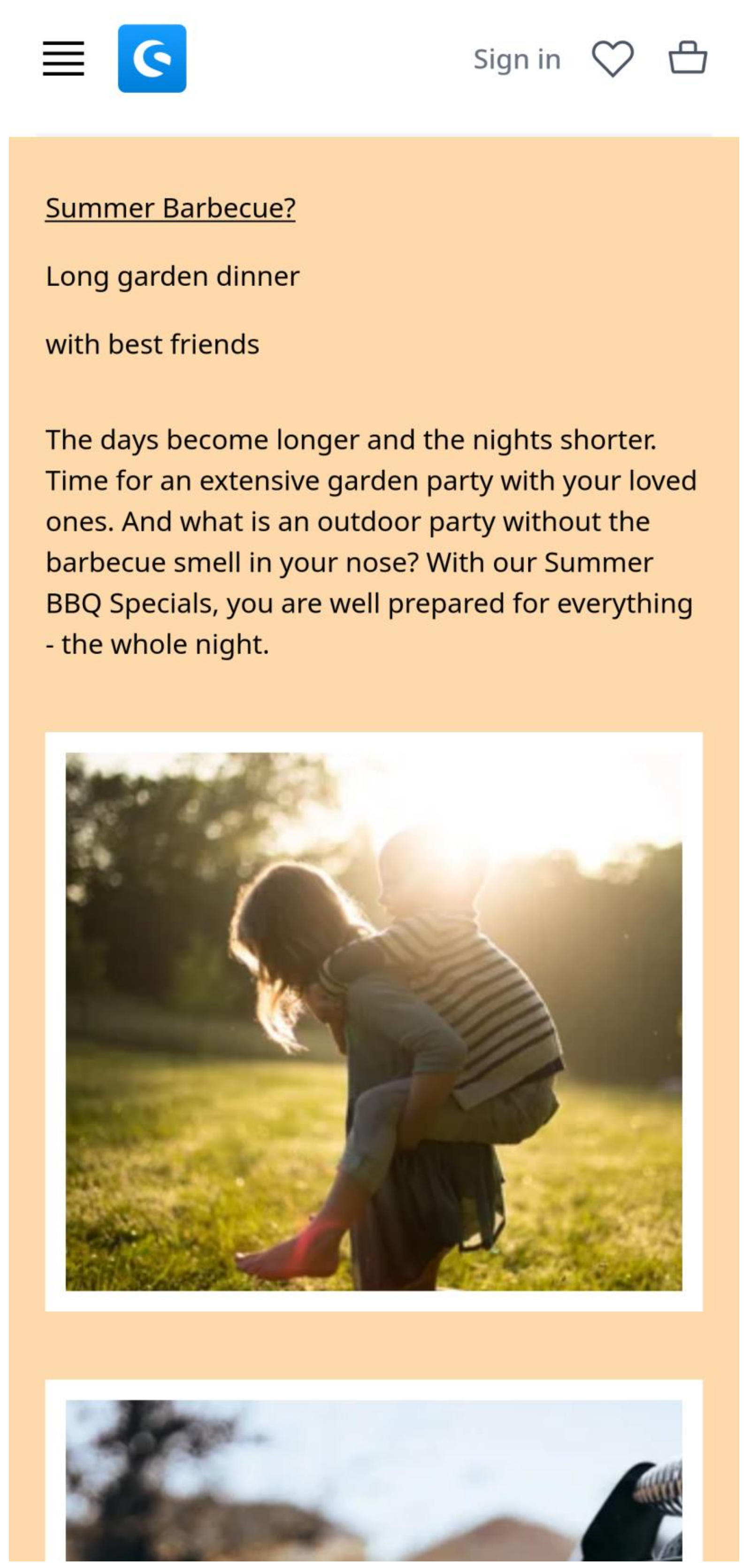
Status

Switching off a Sales Channel will make it inaccessible to visitors as well as API connections.

Active

Maintenance mode

In maintenance mode the Sales Channel will be invisible to visitors. However, you will be able to access it, using white listed IPs. This way you can safely work on the Sales Channel data, while blocking visitors at a

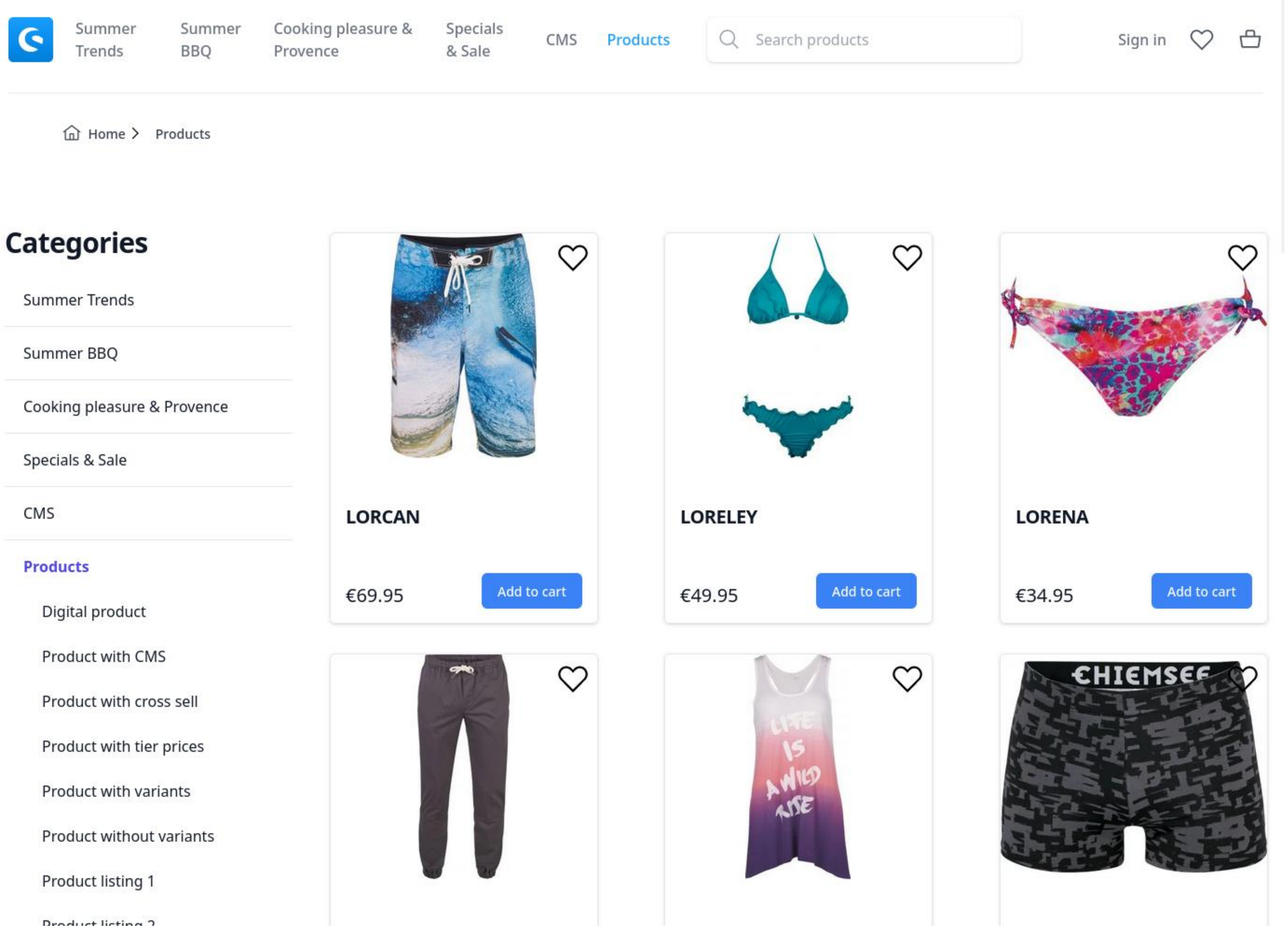

Sign in

Summer Barbecue?

Long garden dinner

with best friends

The days become longer and the nights shorter. Time for an extensive garden party with your loved ones. And what is an outdoor party without the barbecue smell in your nose? With our Summer BBQ Specials, you are well prepared for everything - the whole night.



Categories

- Summer Trends
- Summer BBQ
- Cooking pleasure & Provence
- Specials & Sale
- CMS
- Products
  - Digital product
  - Product with CMS
  - Product with cross sell
  - Product with tier prices
  - Product with variants
  - Product without variants
  - Product listing 1

LORCAN	€69.95	Add to cart
LORELEY	€49.95	Add to cart
LORENA	€34.95	Add to cart

Demo: https://frontends-demo.vercel.app

Packages

- api-client (handcrafted, TypeScript)
- api-client-next (generated, TypeScript)
- types (TypeScript) [useless with api-client-next]
- helpers (TypeScript)
- composables (TypeScript + Vue) [Composition API]
- nuxt3-module (TypeScript + Vue + Nuxt)
- cms-base (TypeScript + Vue + Nuxt + Tailwind)

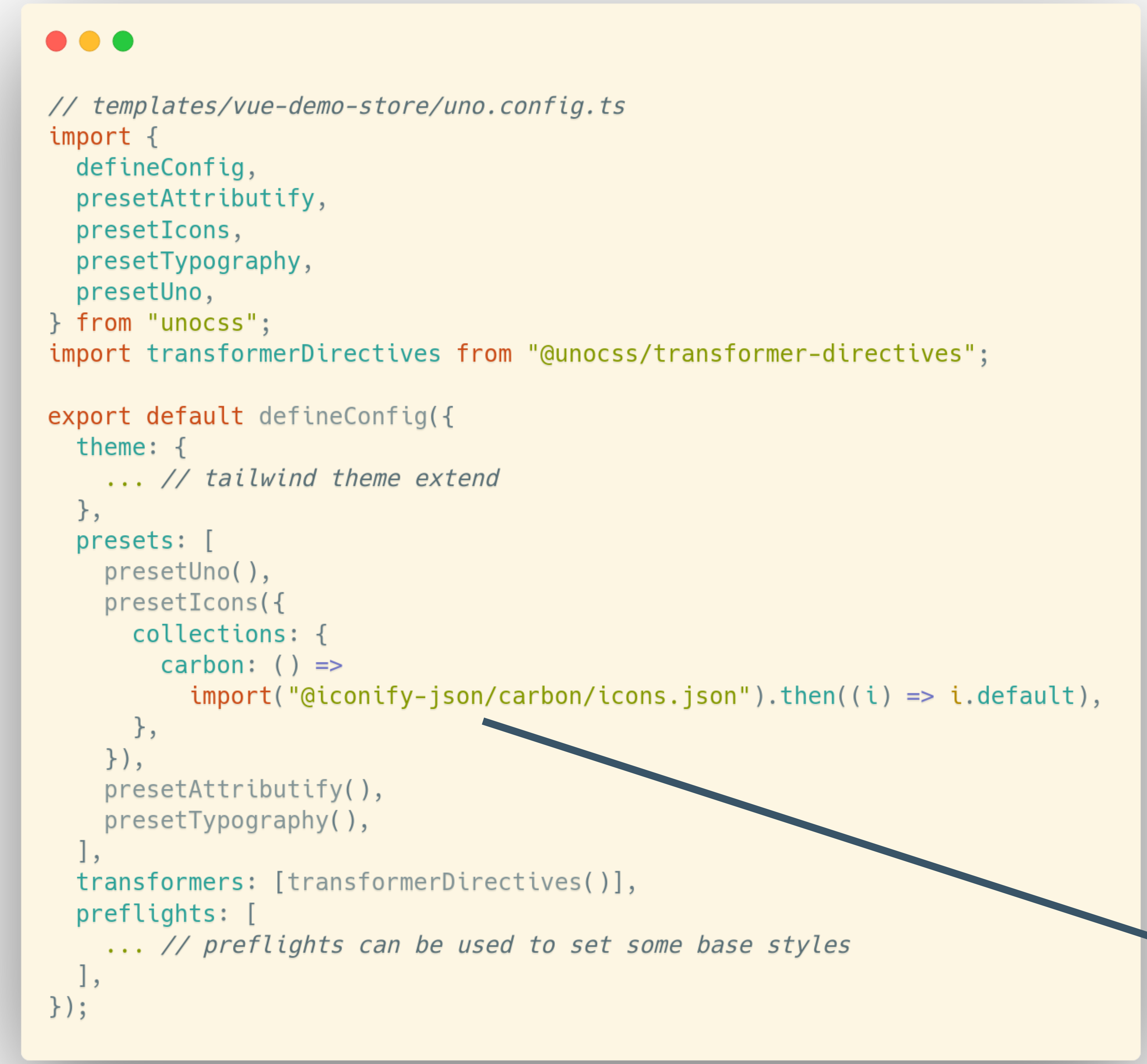
Demo-Store explained

Folder Structure:

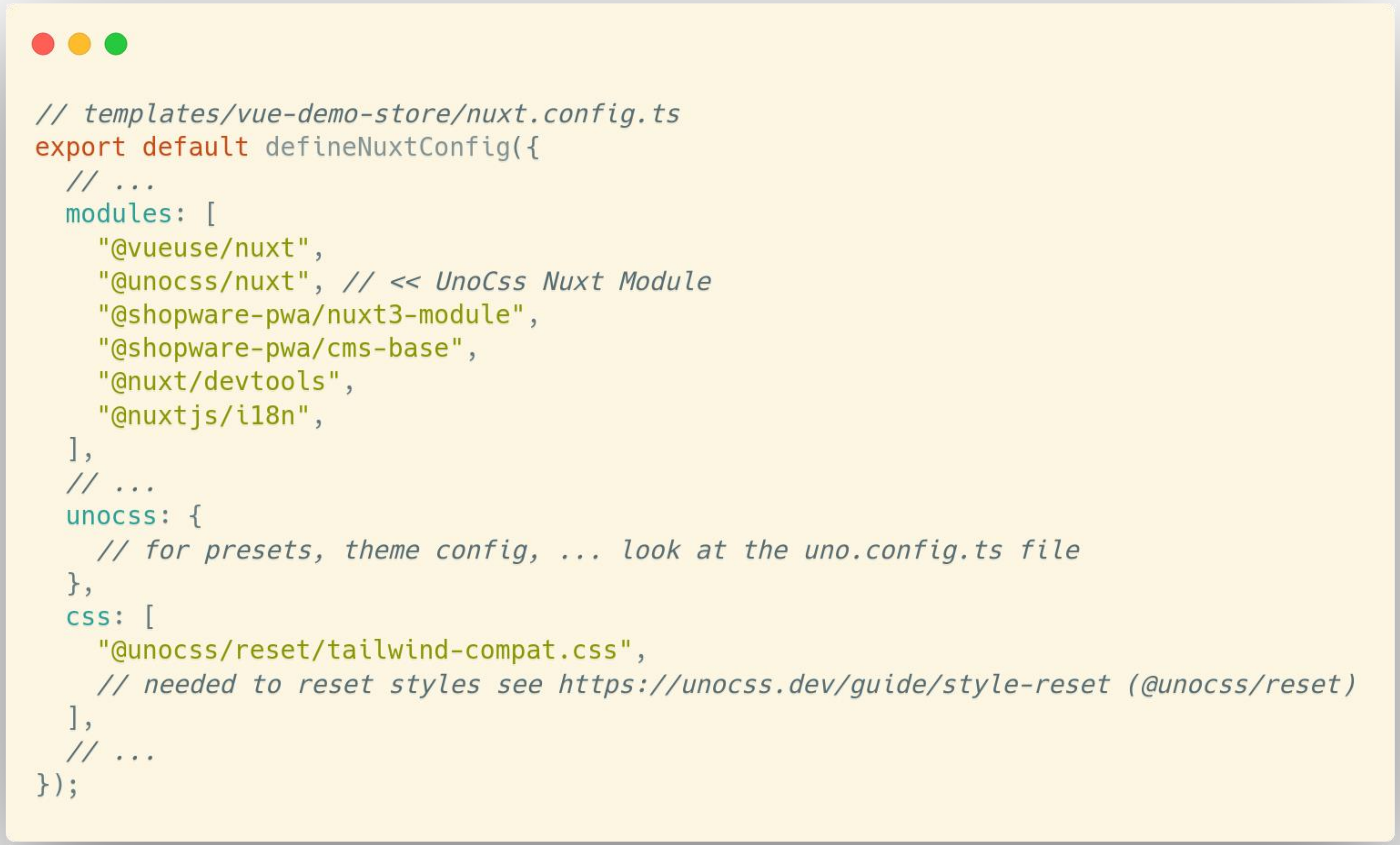
Nuxt Directory Docu

- 🗀 assets
- 🗀 components
- 🗀 composables
- 🗀 i18n
- 🗀 layouts
- 🗀 pages
- 🗀 public
- 🗀 server
- app.vue
- nuxt.config.ts
- package.json
- tsconfig.json
- uno.config.ts

UnoCss Configuration (see Docu)



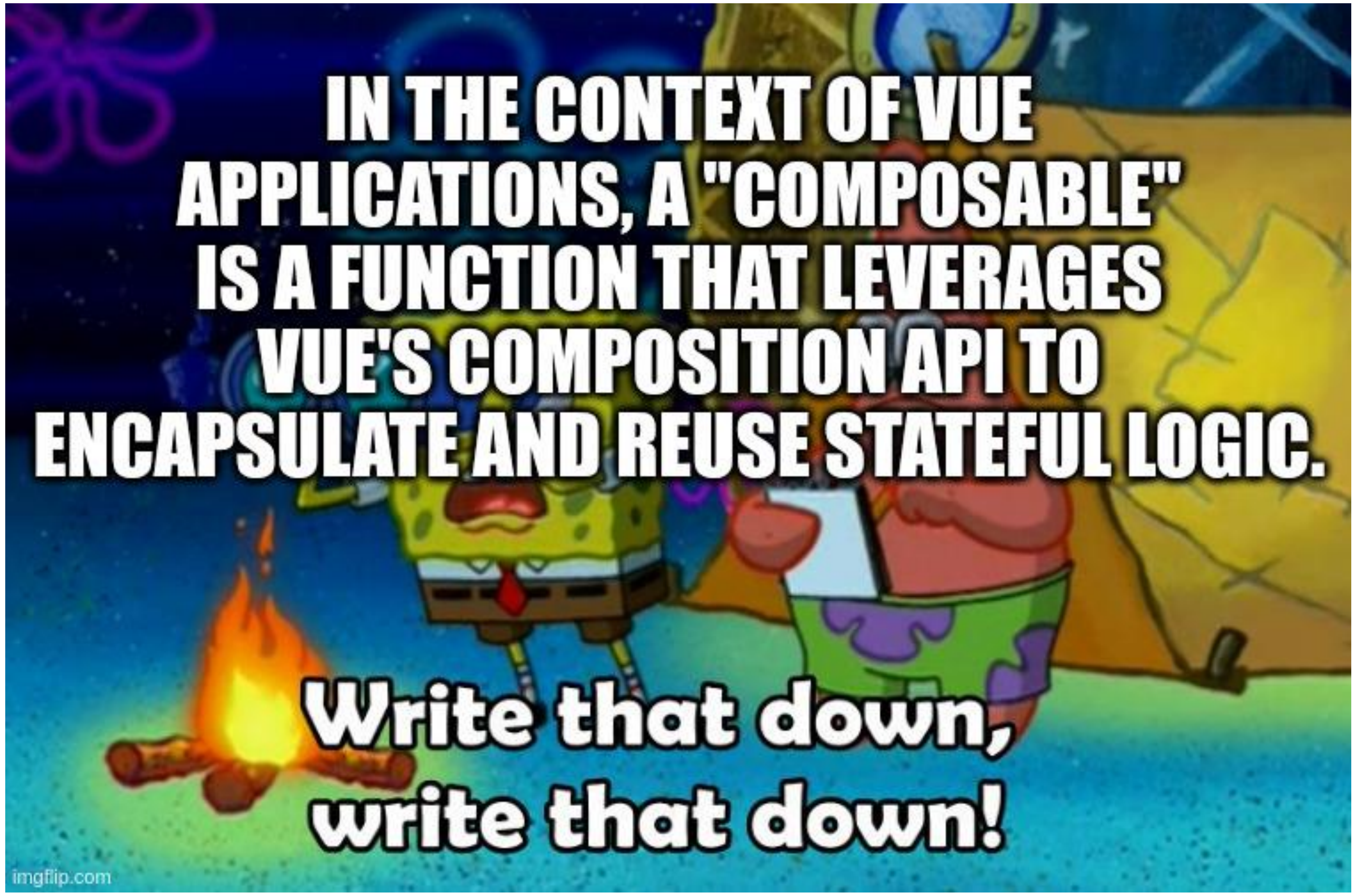
```ts
// templates/vue-demo-store/uno.config.ts
import {
  defineConfig,
  presetAttributify,
  presetIcons,
  presetTypography,
  presetUno,
} from "unocss";
import transformerDirectives from "@unocss/transformer-directives";

export default defineConfig({
  theme: {
    ... // tailwind theme extend
  },
  presets: [
    presetUno(),
    presetIcons({
      collections: {
        carbon: () =>
          import("@iconify-json/carbon/icons.json").then((i) => i.default),
      },
    }),
    presetAttributify(),
    presetTypography(),
  ],
  transformers: [transformerDirectives()],
  preflights: [
    ... // preflights can be used to set some base styles
  ],
});
```



```ts
// templates/vue-demo-store/nuxt.config.ts
export default defineNuxtConfig({
  // ...
  modules: [
    "@vueuse/nuxt",
    "@unocss/nuxt", // << UnoCss Nuxt Module
    "@shopware-pwa/nuxt3-module",
    "@shopware-pwa/cms-base",
    "@nuxt/devtools",
    "@nuxtjs/i18n",
  ],
  // ...
  unocss: {
    // for presets, theme config, ... look at the uno.config.ts file
  },
  css: [
    "@unocss/reset/tailwind-compat.css",
    // needed to reset styles see https://unocss.dev/guide/style-reset (@unocss/reset)
  ],
  // ...
});
```

Presets for Tailwind CSS, Windi CSS, Bootstrap, Tachyons and more.

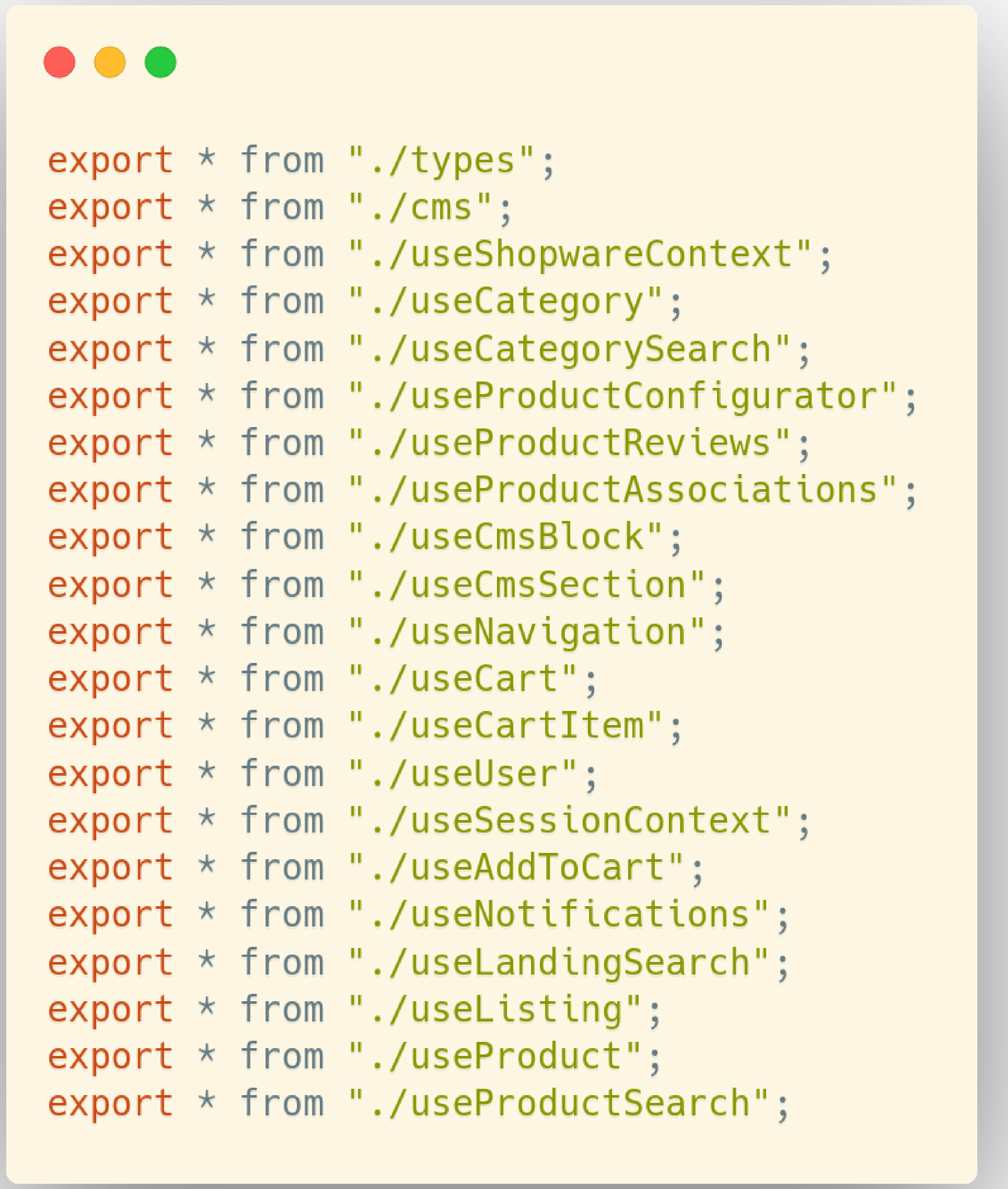
What is a "Composable"?


IN THE CONTEXT OF VUE APPLICATIONS, A "COMPOSABLE" IS A FUNCTION THAT LEVERAGES VUE'S COMPOSITION API TO ENCAPSULATE AND REUSE STATEFUL LOGIC.
Write that down, write that down!
imgflip.com

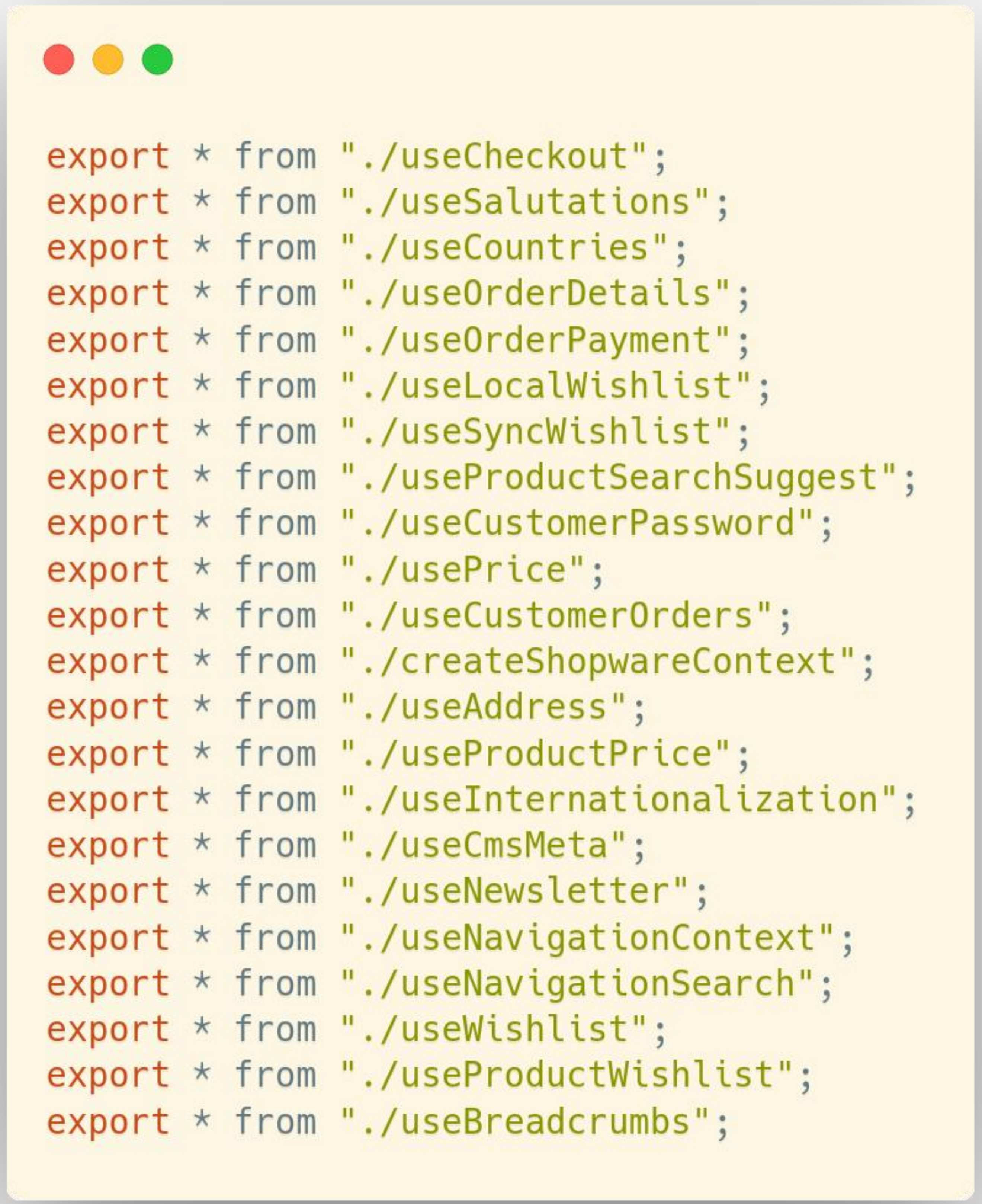
Stateful vs Stateless

Composition API (FAQ)

Composables provide by Frontends



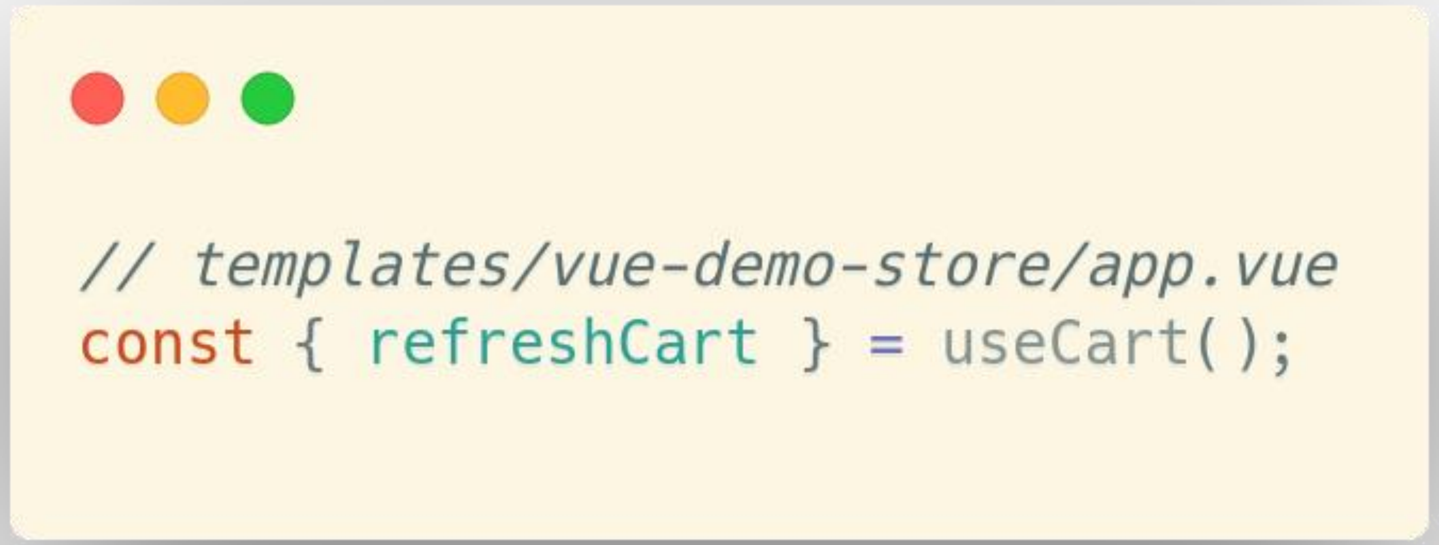
```
export * from "./types";
export * from "./cms";
export * from "./useShopwareContext";
export * from "./useCategory";
export * from "./useCategorySearch";
export * from "./useProductConfigurator";
export * from "./useProductReviews";
export * from "./useProductAssociations";
export * from "./useCmsBlock";
export * from "./useCmsSection";
export * from "./useNavigation";
export * from "./useCart";
export * from "./useCartItem";
export * from "./useUser";
export * from "./useSessionContext";
export * from "./useAddToCart";
export * from "./useNotifications";
export * from "./useLandingSearch";
export * from "./useListing";
export * from "./useProduct";
export * from "./useProductSearch";
```



```
export * from "./useCheckout";
export * from "./useSalutations";
export * from "./useCountries";
export * from "./useOrderDetails";
export * from "./useOrderPayment";
export * from "./useLocalWishlist";
export * from "./useSyncWishlist";
export * from "./useProductSearchSuggest";
export * from "./useCustomerPassword";
export * from "./usePrice";
export * from "./useCustomerOrders";
export * from "./createShopwareContext";
export * from "./useAddress";
export * from "./useProductPrice";
export * from "./useInternationalization";
export * from "./useCmsMeta";
export * from "./useNewsletter";
export * from "./useNavigationContext";
export * from "./useNavigationSearch";
export * from "./useWishlist";
export * from "./useProductWishlist";
export * from "./useBreadcrumbs";
```

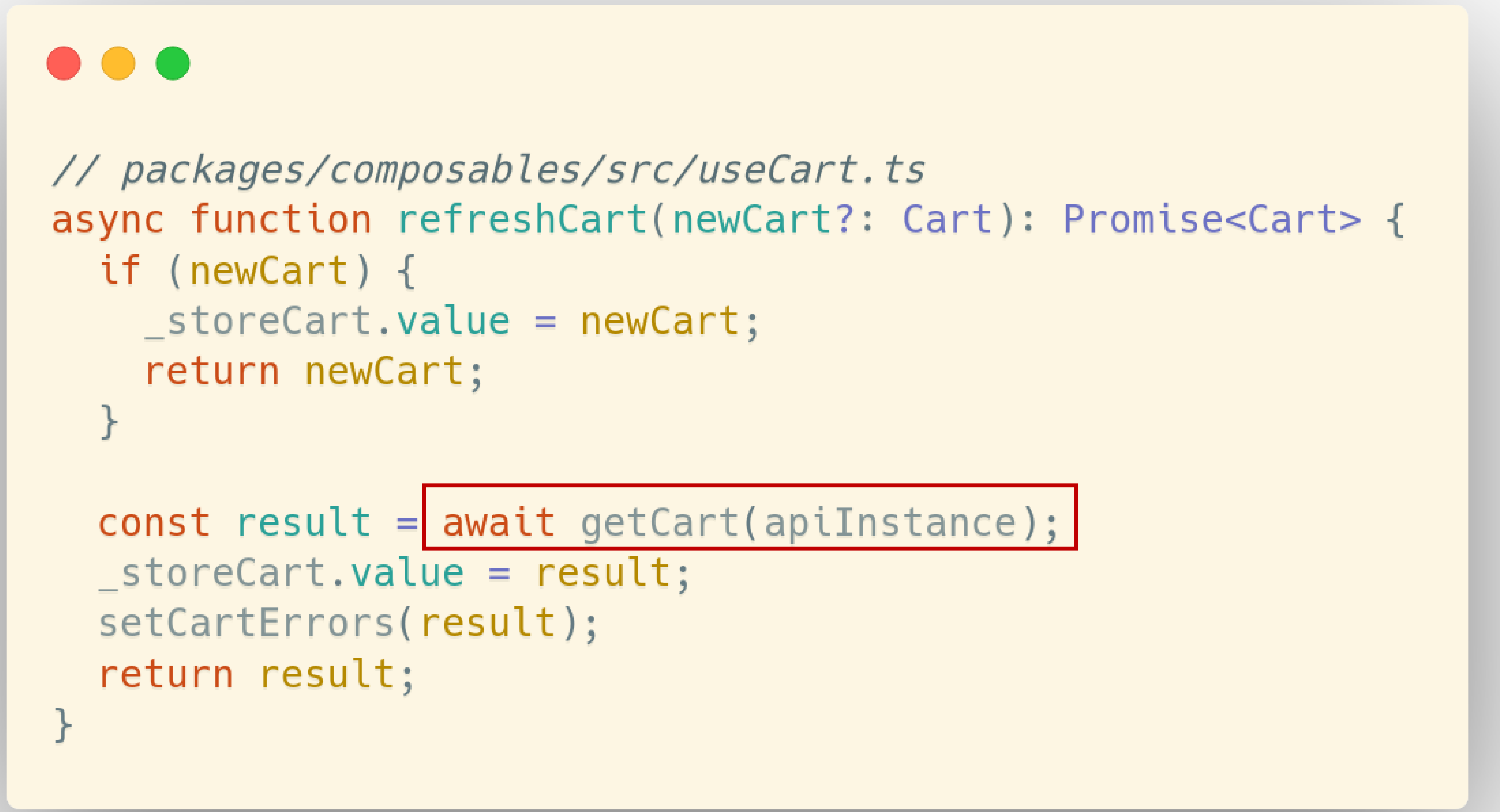
Composables, Data-Flow, old api-client

1



```
// templates/vue-demo-store/app.vue
const { refreshCart } = useCart();
```

2

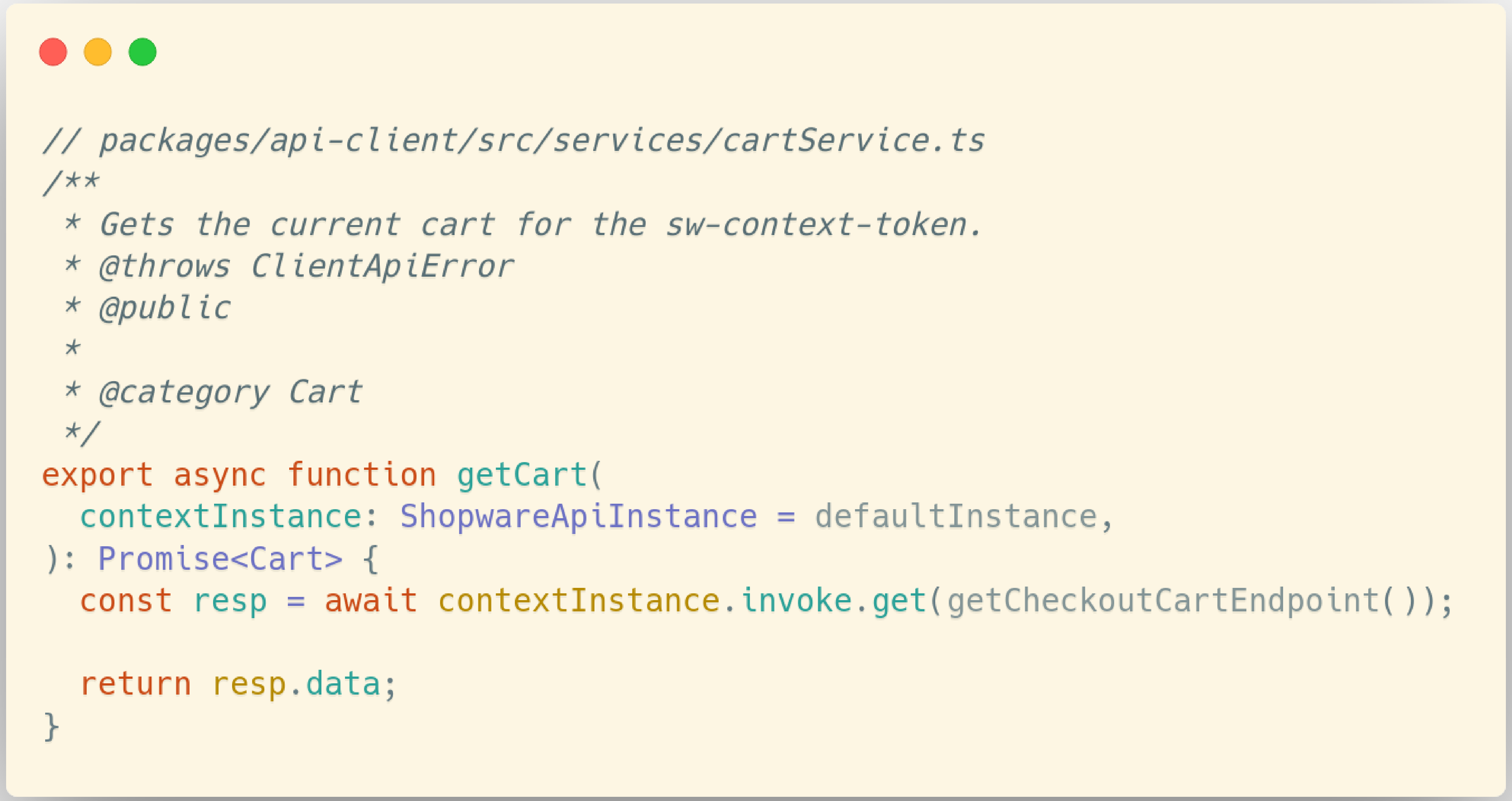


```
// packages/composables/src/useCart.ts
async function refreshCart(newCart?: Cart): Promise<Cart> {
  if (newCart) {
    _storeCart.value = newCart;
    return newCart;
  }

  const result = await getCart(apiInstance);
  _storeCart.value = result;
  setCartErrors(result);
  return result;
}
```

Auto Imports (Nuxt)

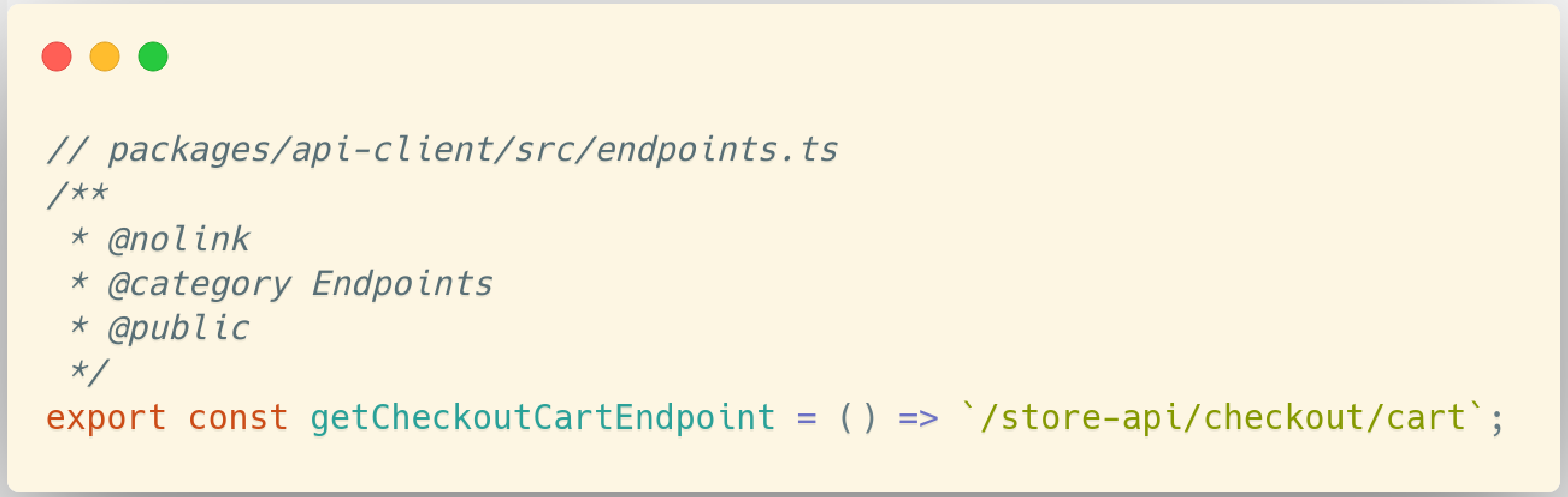
Composables, Data-Flow, old api-client

3



```ts
// packages/api-client/src/services/cartService.ts
/**
 * Gets the current cart for the sw-context-token.
 * @throws ClientApiError
 * @public
 *
 * @category Cart
 */
export async function getCart(
  contextInstance: ShopwareApiInstance = defaultInstance,
): Promise<Cart> {
  const resp = await contextInstance.invoke.get(getCheckoutCartEndpoint());

  return resp.data;
}
```

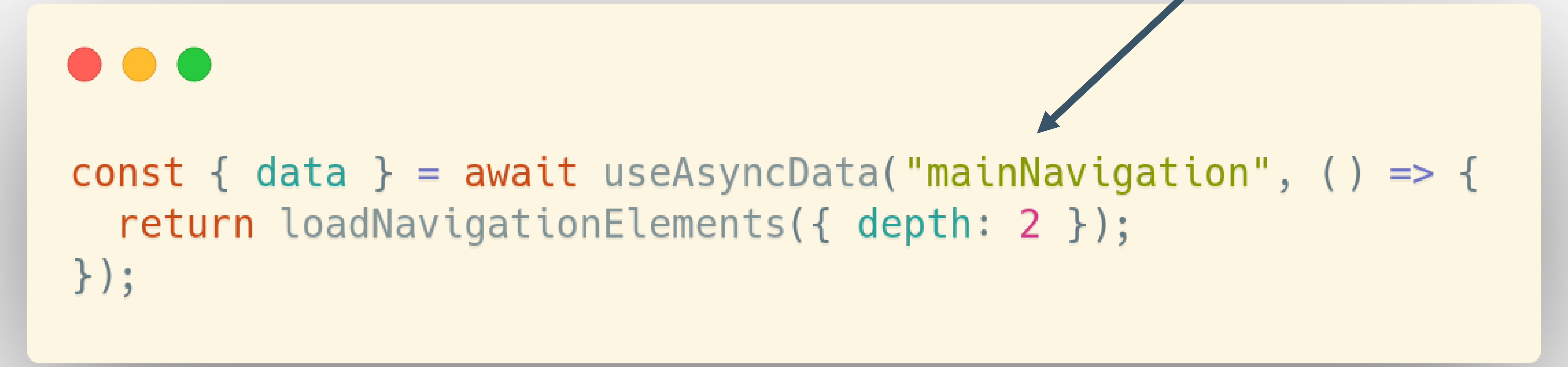
4



```ts
// packages/api-client/src/endpoints.ts
/**
 * @nolink
 * @category Endpoints
 * @public
 */
export const getCheckoutCartEndpoint = () => `/store-api/checkout/cart`;
```

useAsyncData (see Docu)

Unique Key for cache (optional)



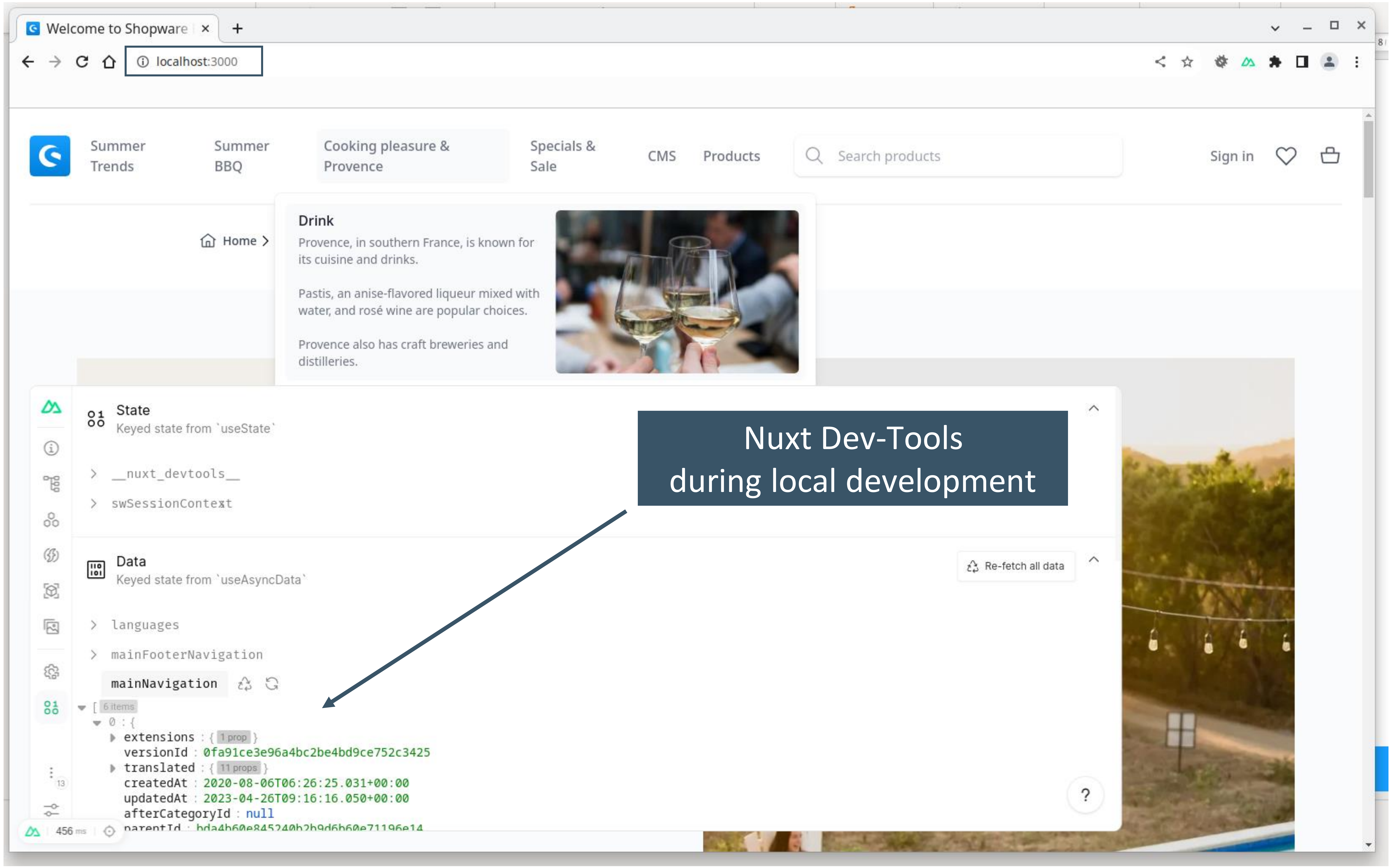
```
const { data } = await useAsyncData("mainNavigation", () => {
  return loadNavigationElements({ depth: 2 });
});
```

useAsyncData is a composable meant to be called directly in a **setup function, plugin, or route middleware**. It returns reactive composables and handles adding responses to the Nuxt payload so they can be passed from server to client **without re-fetching** the data on client side when the page hydrates.

useAsyncData Options

- Lazy (only client-side navigation, handle the loading state)
- Client-only fetching (server: false, with lazy: non-SEO sensitive data)
- Minimize payload size (pick only what you need)
- Caching and refetching (watch, manual refetch)

useAsyncData is Developer sugar and uses **fetch** (ofetch) under the hood. So, you can also use all fetch features.



Drink

Provence, in southern France, is known for its cuisine and drinks.

Pastis, an anise-flavored liqueur mixed with water, and rosé wine are popular choices.

Provence also has craft breweries and distilleries.

State
Keyed state from `useState`

```
> __nuxt_devtools__
> swSessionContext
```

Data
Keyed state from `useAsyncData`

```
> languages
> mainFooterNavigation
  mainNavigation
▾ [ 6 items
  ▾ 0 : {
    ▸ extensions : { 1 prop }
      versionId : 0fa91ce3e96a4bc2be4bd9ce752c3425
    ▸ translated : { 11 props }
      createdAt : 2020-08-06T06:26:25.031+00:00
      updatedAt : 2023-04-26T09:16:16.050+00:00
      afterCategoryId : null
      parentId : bda4b60e845240b2b9d6b60e71196e14
```

456 ms

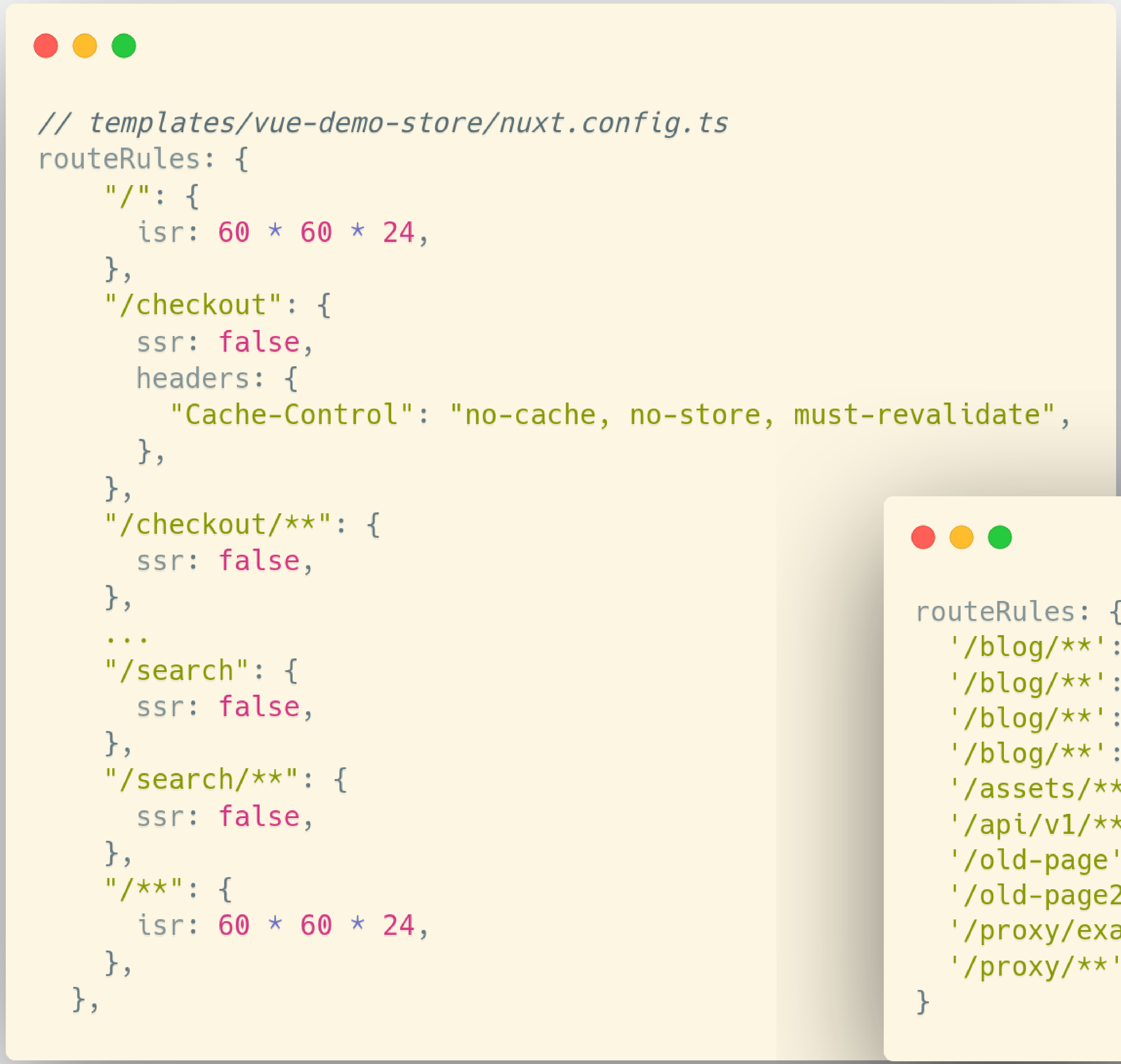
Nuxt Dev-Tools
during local development

Rendering Modes (see Docu)

- Universal Rendering (Full HTML, Download JS, Hydrate)
- Client-Side Rendering (Empty HTML, Download JS, Interactive)
- Hybrid Rendering (Route Rules + Nitro Caching Layer)
- Edge-Side Rendering (Hybrid + Edge Platforms)

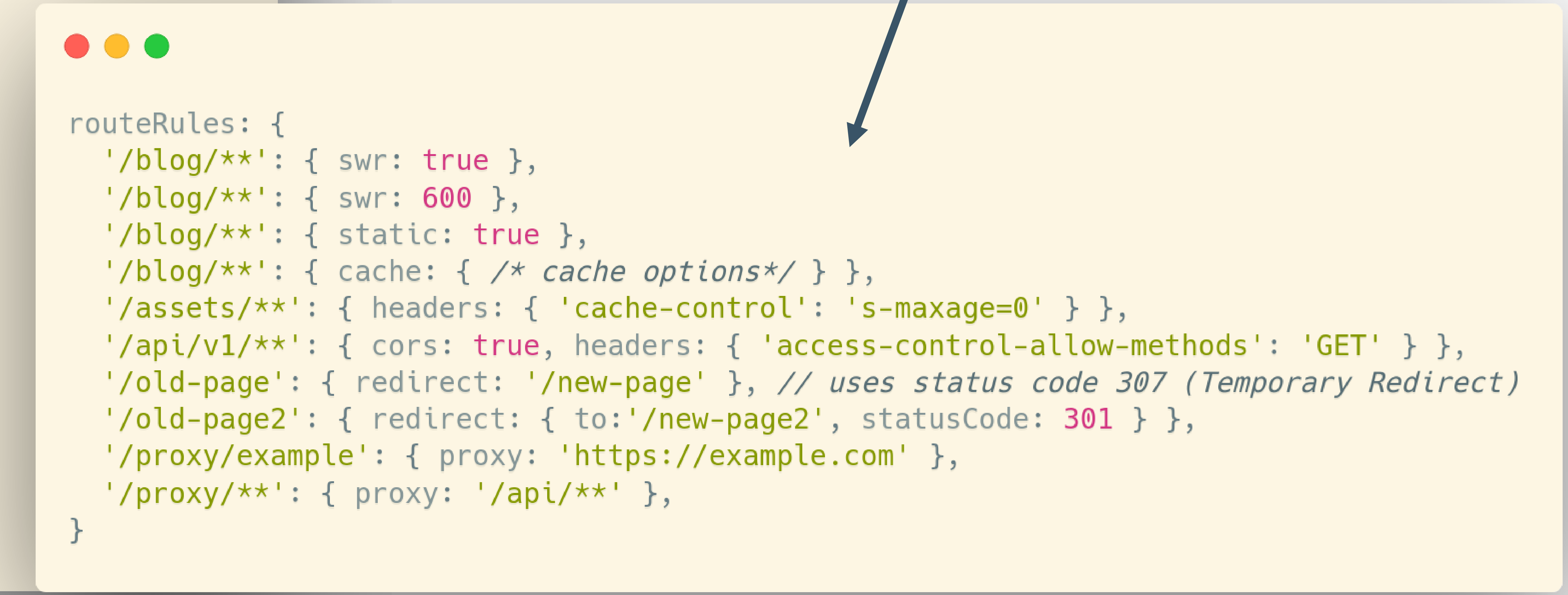
Hint: Think about pre-rendering with the **nuxt generate** command. You can also use selective pre-rendering for specific routes. So, you can use the speed of Static site generation (SSG) ***AND*** the dynamic from Universal Rendering. See Nuxt Deployment Docu.

routeRules config (Hybrid Rendering)



```
// templates/vue-demo-store/nuxt.config.ts
routeRules: {
    "/": {
      isr: 60 * 60 * 24,
    },
    "/checkout": {
      ssr: false,
      headers: {
        "Cache-Control": "no-cache, no-store, must-revalidate",
      },
    },
    "/checkout/**": {
      ssr: false,
    },
    ...
    "/search": {
      ssr: false,
    },
    "/search/**": {
      ssr: false,
    },
    "/**": {
      isr: 60 * 60 * 24,
    },
  },
```

Nitro routeRules Docu



```
routeRules: {
  '/blog/**': { swr: true },
  '/blog/**': { swr: 600 },
  '/blog/**': { static: true },
  '/blog/**': { cache: { /* cache options*/ } },
  '/assets/**': { headers: { 'cache-control': 's-maxage=0' } },
  '/api/v1/**': { cors: true, headers: { 'access-control-allow-methods': 'GET' } },
  '/old-page': { redirect: '/new-page' }, // uses status code 307 (Temporary Redirect)
  '/old-page2': { redirect: { to:'/new-page2', statusCode: 301 } },
  '/proxy/example': { proxy: 'https://example.com' },
  '/proxy/**': { proxy: '/api/**' },
}
```

How do you organize your projects?

With Nuxt Layers 😇

- Share **reusable configuration** presets across projects using **nuxt.config** and **app.config**
- Create a component library using **components/directory**
- Create utility and composable library using **composables/directory** and **utils/directory**
- Create **Nuxt themes** and **Nuxt module presets**
- Share **standard setup** across projects

Intro Video

Going Further
Nuxt Layer Docu

Missing the Eco-System?

Going Further
Nuxt Module Docu

There is no Marketplace 😱 There are Nuxt Modules 🚀
Examples: Storyblok, Prismic, Sanity, Stripe, Cookies 👀

Every App/Plugin that expose **store API** endpoints can be used ✅

Every App/Plugin that changes data that is exposed via **store API** endpoints can be used (e.g., Tax-Providers, Newsletter) ✅

Every Provider that supports **JS/TS SDK's** or **npm packages** can be integrated.
Examples: Contentful, Mollie, Payone 👀

You still can provide new store API endpoints via App/Plugin and then build your custom component/composable. Share it with the community. 🖤

Roadmap, Future and Links

Will there be an official Roadmap?

⊖ Not yet.

→ Follow our progress on GitHub in the [Project Board]().

♡ Participate and create feature requests, issues and discussions.

When will Version 1.0 be released?

Check [this Discussion]() on GitHub. 😉

Roadmap, Future and Links

What happened so far?
(95 Issues solved since we moved to GitHub, April/May 2023)

- 6.5 Compatibility
- Sitemap XML (PR)
- Language Switcher (i18n, PR)
- Digital Product
- New API Client
- Performance
  - Less seoURL Calls (PR)
  - Hybrid Rendering / Edge Caching (PR)
- Bugfixes, other Features and more 🤷

Roadmap, Future and Links

What are the next topics to work on?

- Integration of the new API Client into Composables Package
  - Improving the Test-Coverage for Composables
- RFC: Overwriting Composables (see)
- More Integrations (Examples) in general (Payments, CMS and so on)
- More Feature support for Core/B2B depending on Plan (Rise, Evolve and Beyond)
- Re-Work the Demo-Store Design (Mobile improvements, Figma Lib)
  - Image-Regression Testing for Demo-Store (after Re-Work)
- Make the Demo-Store accessibility friendly
- Documentation: New API client with Examples (currently only Readme)
- Documentation: Multi-Store Setup with Language Switcher
- Documentation: How to run the Demo-Store on The Edge
- Experiments: Progressive Enhancement, Bun runtime, Nitro Streaming

Roadmap, Future and Links

Topics not good? You want us to work on something else?

Community Survey

Shopware Community Slack
#shopware-frontends

THX @ TEAM FRONTENDS

Support us with a ⭐ on GitHub

shopware