



شروع شدن

اول از همه از اینکه این قالب را خریداری کردید و مشتری گرامی ما هستید بسیار سپاسگزاریم. شما عالی هستید! شما حق دریافت آپدیت های مادام العمر رایگان برای این محصول و پشتیبانی مستقیم از تیم نینجا CSS را دارید. **Vuero** محصولی است که توسط **Css Ninja** و **Digisquad** ساخته شده است.

این مستندات برای کمک به شما در مورد هر مرحله از راه اندازی و سفارشی سازی نوشته شده است. لطفاً اسناد را با دقت مرور کنید تا متوجه شوید این الگو چگونه ساخته شده است و چگونه آن را به درستی ویرایش کنید. برای سفارشی کردن این الگو، دانش چارچوب HTML CSS و Vue JS مورد نیاز است.

شما در حال خواندن مستندات **Vuero v2.9.0** هستید. محصول استفاده می کند:

- **Vue.js** (نسخه 3.3+) Composition API
- **Typescript** (v5+) خارج از جعبه، برای برنامه های جاوا اسکریپت در مقیاس بزرگ برای هر مرورگری
- ابزار ساخت و توسعه سریع (v5+) **Vite**.
- محیط توسعه (v16.15+) **Node.js**.
- آخرین (v0.9.3+) **bulma** ادغام با **sass**
- پاسخ های تمسخر آمیز HTTP Rest API برای کمک به شما برای ایجاد باطن خود
- تولید تصاویر آماده **docker** بر اساس **بیتنامی**
- **stylelint**، **eslint** و **prettier** از پیش پیکربندی شده است

داخل ریلیز چیست؟

نسخه Vuero شامل دو پروژه است:

- الگو: حاوی کد منبع کامل قالب است که نسخه آزمایشی <https://vuero.cssninja.io> را اجرا می کند

- **Quickstarter** : حاوی کد منبع یک پروژه کوچکتر است که می توانید از آن برای شروع پروژه خود استفاده کنید، فقط شامل فایل های لازم برای شروع کار با Vuerو است.

ضربه شدید

```
release-vuero-v2.9.0.zip
├─ template-vuero-v2.9.0.zip
└─ quickstarter-vuero-v2.9.0.zip
```

نکته

همچنین می توانید قفل دسترسی github خود را باز کنید تا آخرین نسخه قالب را مستقیماً از github دریافت کنید.

<https://cssninja.io/faq/github-access>

پیش نیازها

1. یک ویرایشگر کد خوب تنظیمات [VSCode](#) / [از قبل پیکربندی شده است](#)
2. یک مرورگر وب پشتیبانی شده (Chrome، Edge، Firefox، ...)
3. [Node.js LTS](#) (> 16.15.x) نصب شده است
4. آشنایی با خط فرمان
5. دانش با [TypeScript](#) (> 4.x) (نباید به صورت جهانی نصب شود)

Node.js را نصب کنید

ابتدا بررسی کنید که Node.js و pnpm را نصب کرده اید یا خیر. برای بررسی اینکه آیا Node.js را نصب کرده اید، این دستور را در ترمینال خود اجرا کنید:

ضربه شدید

```
node -v
```

اگر Node.js روی دستگاه شما نصب نیست، می توانید به وب سایت رسمی nodejs.org بروید و بسته به سیستم عامل خود نسخه را انتخاب کنید:

[Node.js را روی Windows، Linux یا Mac OSX نصب کنید](#)

نکته

همچنین می توانید از [Node Version Manager](#) یا [Volta.sh](#) برای مدیریت چندین نسخه Node.js در دستگاه خود استفاده کنید.

pnpm را با corepack فعال کنید

ما [pnpm](#) را توصیه می کنیم که می توان آن را با موارد زیر فعال کرد:

ضربه شدید

```
corepack enable
corepack prepare pnpm@latest --activate
```

Corepack یک اسکریپت است که به عنوان پلی بین پروژه های Node.js و مدیران بسته هایی که قرار است در طول توسعه با آنها استفاده شود، عمل می کند. از نظر عملی، Corepack به شما امکان می دهد از Yarn و pnpm بدون نیاز به نصب آنها استفاده کنید - درست مانند آنچه در حال حاضر با npm اتفاق می افتد، که به طور پیش فرض در Node.js ارسال می شود.

نکته

Corepack با Node.js از نسخه x.16.9 نصب شده است .

اگر نسخه شما در زیر است، آن را نصب کنید: `npm install -g corepack`

حمایت کردن

اگر هنگام ویرایش این الگو مشکلی دارید یا اگر می خواهید در مورد چیزی سؤال بپرسید، می توانید درخواست خود را در قسمت پشتیبانی ما در cssninja.io/faq/support ارسال کنید.



دریافت پشتیبانی

ما به شما کمک می کنیم تا مشکلات خود را حل کنید!



قفل دسترسی GitHub خود را باز کنید

دسترسی به مخزن git خصوصی ما!



در Discord به ما بپیوندید
با ما ارتباط برقرار کنید و چت کنید!



از وبسایت ما دیدن کنید
منتظر جدیدترین محصولات ما باشید!

شما می توانید [تغییرات را در اینجا](#) و داخل پوشه منبع Vuero پیدا کنید یا می توانید تغییرات را در صفحه فروش موضوع بررسی کنید.

یک بار دیگر از شما برای خرید این تم بسیار سپاسگزارم. همانطور که در ابتدا گفتم، خوشحال می شویم اگر سوالی در رابطه با این موضوع دارید به شما کمک کنیم. هیچ تضمینی وجود ندارد، اما ما تمام تلاش خود را برای کمک و حمایت از شما انجام خواهیم داد. اگر سؤال کلی تری در رابطه با Vuero دارید، می توانید یک بلیط باز کنید و سؤال خود را در [پورتال پشتیبانی CSS Ninja](#) بپرسید .

آخرین به روز رسانی:

[این صفحه را در GitHub ویرایش کنید](#)

صفحه بعد

[پروژه خود را راه اندازی کنید](#)



سفارشی سازی Vuero

فلسفه

Vuero به گونه ای ساخته شده است که محصولی بسیار ماژولار و انعطاف پذیر باشد. طرح بندی ها تخریب شده اند، بنابراین می توانید بدون بارگیری مجدد صفحه، کل طرح بندی را تغییر دهید. بنابراین، محتوای داخلی صفحه به عنوان مولفه ای در نظر گرفته می شود که در طرح بندی فعال فعلی تزریق می شود.

Vuero با پروژه QuickStarter عرضه می شود. Quickstarter یک پروژه کوچکتر است که فقط شامل مواردی است که برای شروع با یک پایگاه کد سبک تر و کم حجم تر نیاز دارید. روش پیشنهادی کار این است که هر دو پروژه را باز کنید، و شروع به کپی و چسباندن آنچه از پروژه اصلی Vuero به پروژه Quickstarter نیاز دارید، کنید.

my-vuero-quickstarter-project/

```
├─ src/
│   └─ scss
│       ├── abstracts
│       ├── bulma-generated
│       ├── components
│       ├── css-variables
│       ├── elements
│       ├── extensions
│       ├── layout
│       ├── pages
│       └─ vendors
```

```
| | └─ main.scss
| └─ main.ts
└─ index.html
```

از زمان نسخه 2.0.0 ، Vuerio با یک تغییر پارادایم بزرگ از نظر اجرای Sass همراه است. در حالی که همه با متغیرهای سنتی Sass مانند آشنا هستند ، ما تصمیم گرفتیم این قالب را حذف کنیم تا از قدرت بومی `color$` استفاده کنیم . برای حمایت از این موضوع، مجبور شدیم چندین تغییر اساسی در نحوه مدیریت کامپایل پالت‌های رنگی مختلف ایجاد کنیم. `CSS-vars`

چارچوب اصلی Bulma با متغیرهای سنتی Sass ساخته شده است و از CSS-vars پشتیبانی نمی کند. ما باید برای این موضوع راه حلی پیدا می کردیم. بنابراین، ما تصمیم گرفتیم بسته Bulma موجود خود را با این افزونه bulma ارتقا دهیم: <https://github.com/wtho/bulma-css-vars> . این افزونه به طور کامل از CSS-vars پشتیبانی می کند و پایه کد اولیه Bulma را اصلاح می کند و این پیاده سازی را ممکن می کند.

2 زیر پوشه SCSS جدید در Vuerio وجود دارد `2.0.0` : `/scss/bulma-generated` و `scss/css-` `variables` . اولین مورد به طور خودکار توسط یک ابزار NodeJS رندر می شود و وظیفه تولید همه متغیرهای Bulma را بر اساس پیکربندی شما بر عهده دارد.

تغییر رنگ اصلی

اگر نیاز به تغییر رنگ اصلی Vuerio دارید، باید یک مرحله کامپایل کوتاه را طی کنید. رنگ اصلی از فرمت HSL تولید می شود. این بدان معنی است که برای کار کردن باید رنگ اصلی vuerio خود را در قالب HSL تعریف کنید. در اینجا مراحل مختلفی وجود دارد که باید طی کنید:

- یک رنگ اصلی برای پروژه خود انتخاب کنید. می‌تونه فرمت hex یا rgb باشه فرقی نمیکنه. بیایید برای مثال با رنگ بنفش مانند `#6621cf` .
- در هر انتخابگر رنگی که انتخاب می کنید، رنگ خود را به رنگ HSL با مقداری برای هر ویژگی، Hue، Saturation و Luminance تبدیل کنید. در مورد ما این منجر به `264° , 73% , 47%` .
- باز کن `vuero/bulma-css-vars.config.js` . در آن فایل مقادیر بلوک را `primary: hsl(153, 48,` (49) با مقداری که یک مرحله قبل دریافت کردید جایگزین کنید. همچنین می‌توانید مقادیر پیش‌فرض برخی از متغیرهای اصلی Bulma مانند `link` ، `dark` و `info` غیره را تغییر دهید.
- پس از اتمام کار، `pnpm build:update-bulma-colors` اسکریپت را اجرا کنید.
- برای تکمیل تنظیمات رنگ، مقدار `primary` متغیر داخل را تغییر دهید. `vuero/src/scss/css-variables/_colors.scss`

- تادا! اکنون کار شما تمام شده است و تمام رنگ های جدید شما برای شما تولید شده است.

نکته

از انتخابگر رنگ در <https://vuerocssninja.io/elements/colors> برای پیش نمایش و کپی تکه های رنگ استفاده کنید!

نحو CSS vars

متغیرهای CSS از نحو متفاوتی نسبت به متغیرهای Sass استفاده می کنند. اعلان یک متغیر CSS جدید به این صورت است:

اعلام

SCSS

```
// :root is an alias for html element but with higher priority
:root {
  --myVariable: #fff;
}
```

SCSS

```
// we can override the variable value inside a class scope
.my-red-variable {
  --myVariable: red;
}
```

vue

```
<!-- we can also override the variable value inside a style scope -->
<span style="--myVariable: blue">
  <!-- ... -->
</span>
```

استفاده

SCSS

```
.my-variable-color {
  color: var(--myVariable);
}
```

```
}
```

نادیده گرفتن متغیرهای CSS

متغیرهای CSS بسیار انعطاف‌پذیر هستند و تقریباً از همه جا می‌توان آنها را بدون تأثیرگذاری بر سایر مؤلفه‌ها لغو کرد. به عنوان مثال، فرض کنید که شما یک مؤلفه به نام دارید `<SuperButton></SuperButton>` و آن مؤلفه دارای ویژگی سبک (`<style lang="scss" scoped></style>`) scoped است، به این معنی که سبک‌ها به طور دقیق به همان مؤلفه محدود می‌شوند. شما می‌توانید `primary--` متغیر پایه را در داخل کامپوننت بدون تأثیرگذاری بر متغیر دیگری خارج از محدوده این کامپوننت لغو کنید. به عنوان مثال، می‌توانید انجام دهید:

vue

```
<!-- SuperButton.vue -->
<template>
  <button class="super-button">
    <!-- ... -->
  </button>
</template>

<style lang="scss" scoped>
  .super-button {
    --primary: blue;
  }
</style>
```

سایر فایل‌های Sass

Vuero به ویژگی‌های قدرتمند Sass و ساختار مدولار متکی است که به شما امکان می‌دهد سبک‌های پیچیده را در یک نسیم مدیریت کنید. شما باید تمام جزئی‌های SCSS را به فایل اصلی خود وارد کنید. به این ترتیب فایل‌های scss سازماندهی می‌شوند. نام فایل‌های SCSS جزئی همیشه با زیرخط مانند این شروع می‌شود: `button.scss_`. آنها به عنوان تکه‌های کد عمل می‌کنند که فقط در صورت نیاز می‌توانید آنها را وارد کنید. البته برخی از آنها برای اجرای پروژه اجباری هستند. Vuero یک کیت UI است که در آن هر فایل SCSS هدف متفاوتی را انجام می‌دهد:

- **Abstracts** : تمام فایل های مربوط به میکسین ها، سبک های پیش فرض جهانی و سایر تنظیمات تایپوگرافی را در خود نگه می دارد.
- **Components** : تمامی فایل های مربوط به کامپوننت های Vuerو را نگه می دارد. هر نوع جزء فایل مخصوص به خود را دارد.
- **Elements** : تمام فایل های مربوط به Vuerو Elements را نگه می دارد. هر نوع عنصر فایل مخصوص به خود را دارد.
- **layout** : تمام فایل های چیدمان اولیه و اجباری را در خود جای می دهد. اگر یکی از آن فایل ها را حذف کنید، پروژه خراب می شود.
- **pages** : تمام سبک های خاص را برای هر نسخه نمایشی یا گروهی از دموها در خود نگه می دارد.
- **vendors** : همه سبک های CSS فروشنده را نگه می دارد.

وارد کردن سبک ها با Vite

برای بارگیری شیوه نامه ها در برنامه ما (مثلاً اگر نیاز به اضافه کردن سبک های اضافی از یک `node_modules` افزونه دارید)، فقط باید فایل ها یا `css` فایل ها را وارد کنیم . این فایل در بسته نرم افزاری شما گنجانده شده است زیرا در داخل فایل ریشه ارجاع داده شده است `index.html` `src/styles.ts` `scss` `sass`

تایپ

```
// file: ./src/styles.ts

// ...
import "notyf/notyf.min.css";

import "../scss/main.scss";
// ...
```

همه فایل های وارد شده در اینجا باید به تبدیل شوند `css` و `index.html` در زمان ساخت و اجرا به طور خودکار به آن تزریق شوند. سبک های تزریق شده در سطح جهانی در دسترس هستند.

ادغام Bulma

کلاسیک Bulma قبلا به طور کامل با Vuerio ادغام شده بود. این بدان معناست که وقتی متغیر رنگ Vuerio را تغییر دادید `primary$`، بر هر متغیر مرتبط با Bulma اولویت داشت. در `2.0.0` Vuerio، همه چیز به شدت تغییر کرده است، همانطور که در بالا بحث کردیم. Vuerio اکنون به طور کامل از متغیرهای بومی CSS پشتیبانی می کند و پشتیبانی از متغیرهای Sass را کنار گذاشته است.

حالت تاریک بومی

Vuerio دارای حالت Dark بومی است. این بدان معناست که همه اجزا برای حالت تاریک از پیش طراحی شده اند. لازم نیست نگران آن باشید، وقتی آن را روشن می کنید، رنگ ها یکپارچه تغییر می کنند. استایل حالت تاریک از طریق یک `is-dark` کلاس جهانی که به `<html>` عنصر ریشه صفحه اضافه شده است ساخته می شود. حالت تاریک روی بدنه با جاوا اسکریپت تغییر می کند. در نوع دیگری از پیاده سازی، بدنه باید توسط سرور با کلاس مناسب قبل از ارائه به مشتری، بر اساس انتخاب کاربر، ارائه شود.

نکته

`is-dark` کلاس محدود به عنصر نیست `<html>`، شما می توانید این کلاس را به هر عنصری اضافه کنید، بنابراین همه کودکان در حالت تاریک خواهند بود!

حالت تاریک و متغیرهای CSS

در نسخه های قبلی Vuerio، پیاده سازی حالت تاریک نیاز به داشتن استایل های تودرتوی اضافی در یک `is-.` `dark` اصلاح کننده کلاس داشت. با معرفی انواع CSS، حالت تاریک اکنون در سطح رنگ مدیریت می شود. شما می توانید نحوه رفتار یک متغیر CSS در زمان اجرا را بر اساس کلاس های والد کنترل کنید. برای مثال فرض کنید یک متغیر CSS مانند این داریم: `color: red--`. می توانیم با ویرایش متغیر، مقدار این رنگ را در حالت تاریک تغییر دهیم و از نوشتن کد CSS اضافی در حالت تاریک جلوگیری کنیم.

SCSS

```
// Normal mode
:root {
  --color: red;
}

// Dark mode
.is-dark {
```

```
--color: blue;
}
```

Lazyloading Scss

ضربه شدید

```
my-vuero-quickstarter-project/
├─ src/
│   └─ components/
│       └─ pages/
```

پوشه pages صفحات الگو را به عنوان تکه‌هایی از UI قابل استفاده مجدد نگه می‌دارد که می‌توانند در همه انواع طرح‌بندی موجود درج شوند. هر صفحه یک جزء Vue 3 با `<style>` عنصری است که SCSS مورد نیاز آن صفحه را در خود نگه می‌دارد. به این ترتیب، CSS غیر ضروری را هنگام مرور بارگیری نمی‌کنید.

آنها به طور پیش‌فرض به اضافه نمی‌شوند `main.scss`، در عوض ما آنها را در طرح‌بندی یا مستقیماً در صفحات بارگذاری می‌کنیم:

vue

```
<!-- file ./src/pages/status.vue -->
<script setup lang="ts">
// ...
</script>

<template>
  <!-- ... -->
</template>

<style lang="scss">
/* files imported in components will be loaded only once they are needed */
@import "../scss/abstracts/_mixins.scss";
@import "../scss/pages/generic/_utility.scss";

.status-page-wrapper {
  /* custom scss for this page */
  &:hover {
    color: var(--primary);
  }
}
```

```
}  
}  
</style>
```

محدوده SCSS در اجزای Vue

Vue 3 بسیار قدرتمند است، ما Vuerو را ساخته‌ایم تا دیگر نگران انتقال اجزا از پوشه‌ای به پوشه دیگر یا از پروژه‌ای به پوشه دیگر نباشید. این کار به سادگی کپی و چسباندن اجزای خود در پوشه هدف است. از آنجایی که این پروژه از `unplugin-vite-components`، تمام اجزای شما تجزیه شده و بدون یک عبارت `import` در صفحات و سایر اجزای شما موجود است. ما کامپوننت‌های CSS را از `vue.` فایل‌ها دور نگه داشته‌ایم، بنابراین کاوش در سبک‌های قالب و تطبیق آنها با نیازهای خود برای شما آسان‌تر است.

با این حال، توصیه می‌کنیم وقتی پروژه خود را با Vuerو می‌سازید، از `vue.` پتانسیل فایل‌ها با محدود کردن سبک‌های خود در مؤلفه‌تان استفاده کنید، مانند کاری که ما با مؤلفه‌های صفحه انجام می‌دهیم. به این ترتیب، سبک‌های مؤلفه تنها زمانی بارگذاری می‌شوند که مؤلفه نمایش داده شود.

vue

```
<script setup lang="ts">  
export type MyComponentColors = 'red' | 'blue' | 'green'  
  
export interface MyComponentsProps = {  
  color?: MyComponentColors  
  label: string  
}  
  
const props = defineProps<MyComponentsProps>();  
</script>  
  
<template>  
  <button class="button" :class="[props.color && `is-${props.color}`]">  
    {{ props.label }}  
  </button>  
</template>  
  
<style lang="scss" scoped>  
.button {  
  &.is-red {
```

```
color: var(--red);
}

&.is-blue {
  color: var(--blue);
}

&.is-green {
  color: var(--green);
}

&.is-purple {
  color: var(--purple);
}
}
</style>
```

پشتیبانی RTL

ما پشتیبانی RTL را در Vuerro 2.7.0 معرفی کردیم. این بدان معنی است که اکنون می توانید از Vuerro در زبان های RTL مانند عربی، عبری، فارسی، اردو و غیره استفاده کنید. پشتیبانی RTL به طور پیش فرض فعال است و به `i18n`. شما می توانید یک مثال عینی را در [صفحه ثبت نام Vuerro](#) مشاهده کنید. این مثال از منوی کشویی انتخاب زبان استفاده می کند و هنگامی که یک زبان RTL (مانند عربی) انتخاب می شود، به طور خودکار به RTL تغییر می کند.

به غیر از آن، اگر به یک برنامه فقط RTL نیاز دارید، می توانید ویژگی های مورد نیاز را در موارد زیر تغییر دهید

`index.html`:

```
<html dir="rtl" lang="ar">
</html>
```

html

این به طور خودکار طرح را به حالت RTL تغییر می دهد.



Vue 3 و Vuero

استفاده مجدد از منطق stateful با composable

Composable ها قطعات کد قابل استفاده مجدد هستند که با استفاده از توابع Reactivity API و Composition API منطق حالتی را مدیریت می کنند. آنها فقط باید به صورت همزمان در یک `script>` زمینه فراخوانی شوند. ممکن است بخواهید در وبسایت اسناد *Vue.js* بیشتر بخوانید: <https://vuejs.org/guide/extras/composition-api-faq.html>

در اینجا چند نمونه از composable ها از `vueuse/core` کتابخانه مورد استفاده در Vuero آورده شده است:

- `useCssVar` : <https://vueuse.org/core/useCssVar>
- `usePointer` : <https://vueuse.org/core/usepointer>
- `useDraggable` : <https://vueuse.org/core/usedraggable>
- `useLocalStorage` : <https://vueuse.org/core/useLocalStorage>

همچنین می توانید ترکیب پذیرهای داخلی `src/composable/` مانند `useFetch` ، `useNotyf` و را پیدا کنید `useDropdown`

چرا Vue 3 Composition API؟

نکته

دوره کامل موجود در Vue Mastery را مشاهده کنید :

<https://www.vuemastery.com/courses/vue-3-essentials/why-the-composition-api>

داده های واکنشی با Pinia ذخیره می شود

هنگامی که نیاز به اشتراک گذاری داده ها در کل برنامه دارید، می توانید از کتابخانه [Pinia](#) استفاده کنید که جایگزین [Vuex](#) در اکوسیستم Vue 3 می شود.

زمان استفاده از فروشگاه های پینیا:

- زمانی که نیاز دارید داده ها را در همه جای برنامه خود با یک منبع حقیقت به اشتراک بگذارید.
- زمانی که باید داده ها را در تمام چرخه عمر برنامه نگه دارید (مثلاً: بین انتقال صفحات).

چه زمانی از فروشگاه ها استفاده نکنید:

- هنگامی که نیاز دارید داده ها را فقط در یک درخت جزء (مثلاً یک صفحه) به اشتراک بگذارید، به جای آن از [ارائه/تزریق](#) استفاده کنید .

بیا ببینیم چگونه می توانیم یک فروشگاه ساده برای مدیریت داده های جلسه کاربر ایجاد کنیم:

```
ts
// src/stores/userSession.ts
import { defineStore } from "pinia";
import { ref, computed } from "vue";
import { useStorage } from "@vueuse/core";

export type UserData = Record<string, any> | null;

export const useUserSession = defineStore("userSession", () => {
  // token will be synced with local storage
  // @see https://vueuse.org/core/useSessionStorage/
  const token = useSessionStorage("token", "");

  // we use ref and computed to handle reactive data
  const user = ref<Partial<UserData>>>();
  const loading = ref(true);
  const isLoggedIn = computed(
    () => token.value !== undefined && token.value !== ""
  );

  // but we also declare functions to alter the state
  function setUser(newUser: Partial<UserData>) {
    user.value = newUser;
  }
});
```



```

function setToken(newToken: string) {
  token.value = newToken;
}
function setLoading(newLoading: boolean) {
  loading.value = newLoading;
}
async function logoutUser() {
  token.value = undefined;
  user.value = undefined;
}

// we return the entire store
return {
  // this is our state
  user,
  token,
  isLoggedIn,
  // and our mutations
  loading,
  logoutUser,
  setUser,
  setToken,
  setLoading,
} as const;
});

```

اکنون می‌توانیم از فروشگاه در همه جای برنامه خود استفاده کنیم:

```

<script setup lang="ts">
import { watch } from "vue";
import { useUserSession } from "@/src/stores/userSession";

const userSession = useUserSession();

watch(userSession.isLoggedIn, () => {
  console.log("login status changed", userSession.isLoggedIn);
});
</script>

<template>

```

vue

```
<div v-if="userSession.isLoggedIn">
  Welcome {{ userSession.user.name }} -
  <VButton @click="userSession.logoutUser">Logout</VButton>
</div>
<div v-else>You are not logged in</div>
</template>
```

نکته

اسناد پینیا را در وب سایت رسمی بخوانید: [/https://pinia.vuejs.org](https://pinia.vuejs.org)

پلاگین های Vuerو

شما می توانید به سادگی با ایجاد `ts.*` فایل ها در `src/plugins` فهرست، پلاگین هایی برای Vuerو ایجاد کنید.

همه افزونه ها قبل از اجرای برنامه به طور خودکار ثبت می شوند.

در اینجا یک پلاگین دیگ بخار وجود دارد:

```
import { definePlugin } from '@src/app'

export default definePlugin(async ({ app, api, router, head, pinia }) => {
  // run your plugin code here

  // If you need to perform conditional logic based on SSR vs. Client Only,
  you can use
  if (import.meta.env.SSR) {
    // ... server only logic
  }

  // You can do the same for Client Only logic
  if (!import.meta.env.SSR) {
    // ... client only logic

    // you can lazyload libraries that won't work on server side
    asynchronously
```

```
import('client-only-library').then((module) => {  
  // ... do stuff with the module  
})  
}  
})
```

نکته

این مکان خوبی برای واکنشی اطلاعات کاربر / تنظیمات جهانی قبل از شروع برنامه است

مدیریت پیشرفته روتر

تگ های متا سر

در Vuerو، می توانید به راحتی تگ های متا صفحه را با استفاده از composables های ارائه شده توسط

`useHead`، like `unhead` یا `useSeoMeta`.

هشدار

شما باید از SSG یا SSR استفاده کنید تا متا تگ ها برای هر صفحه ارائه شوند. در صورتی که نمی توانید از SSG یا SSR استفاده کنید، باید متا تگ های head خود را به صورت دستی در `index.html` فایل تنظیم کنید، اما امکان داشتن متا تگ های مختلف برای هر صفحه را از دست خواهید داد.

vue

```
<script setup lang="ts">  
useHead({  
  templateParams: {  
    site: {  
      name: 'My Site',  
      url: 'https://example.com',  
    },  
    separator: '-',  
  },  
  title: 'My page',  
  titleTemplate: '%s %separator %site.name',  
  meta: [  

```

```

    {
      property: 'og:site_name',
      content: '%site.name',
    },
    {
      name: 'description',
      content: 'My page description',
    },
  ],
  link: [
    {
      rel: 'stylesheet',
      href: 'https://fonts.googleapis.com/css2?family=Roboto+Condensed&display=swap',
    },
  ],
})
</script>

```

بارگذارهای داده روتر Vue

تجربی

به جای بارگیری داده ها در چرخه حیات کامپوننت، می توانید از `defineLoader` تابع کمکی جدیدی که توسط نویسنده vue-router در این Vue RFC رسمی معرفی شده است استفاده کنید:

<https://github.com/vuejs/rfcs/discussions/460>

به خاطر داشته باشید که این یک ویژگی آزمایشی از vue-router است که ممکن است در آینده تغییر کند. اگر بازخوردی دارید، در بحث های RFC مشارکت کنید!

این یک مثال در مورد نحوه استفاده `defineLoader` در یک جزء صفحه است (مثلا:

(`src/pages/article/[slug].vue/` .

```

<script lang="ts">
// note that we are not in a <script setup> context here
import { defineLoader } from 'vue-router/auto'
import { useFetch } from '@src/composable/useFetch'

interface Article {

```

vue

```

    id: string
    title: string
    slug: string
    content: string
    comments: string[]
  }

export const useArticle = defineLoader(async (route) => {
  const slug = (route.params?.slug as string) ?? ''
  const $fetch = useFetch()

  const data = await $fetch<Article[]>(`/articles`, {
    query: {
      slug,
    }
  })
  const article = data?.[0]

  return article
})
</script>

<script setup lang="ts">
const { data: article, pending } = useArticle()

const router = useRouter()

if (!article.value) {
  // If the article does not exist, we replace the route to the 404 page
  // we also pass the original url to the 404 page as a query parameter
  // http://localhost:3000/article-not-found?original=/blog/a-fake-slug
  router.replace({
    name: '/[...all]', // this will match the ./src/pages/[...all].vue route
    params: {
      all: 'article-not-found',
    },
    query: {
      original: router.currentRoute.value.fullPath,
    },
  })
}

```

```
// Setup our page meta with our article data
useHead({
  title: computed(() => article.value?.title ?? 'Loading article...'),
})
</script>

<template>
  <LandingLayout theme="light">
    <div v-if="pending">Loading article...</div>
    <div v-else class="blog-detail-wrapper">
      <!--
        Page content goes here

        You can see more complete pages content samples from
        files in /src/components/pages directory
      -->

      <h1>{{ article?.title }}</h1>
      <div>{{ article?.content }}</div>
    </div>
  </LandingLayout>
</template>

<style lang="scss" scoped>
.blog-detail-wrapper {
  // Here we can add custom styles for the blog page
  // They will be only applied to this component
}
</style>
```

[Return to top](#)

Menu ☰

در vuejs، می‌توانید با ایجاد یک گروه، محافظ‌های نوبتی را به راحتی تنظیم کنید.

```
import { definePlugin } from '@/src/app'
import { useUserSession } from '@/src/stores/userSession'

export default definePlugin(({ router, api, pinia }) => {
```

ts

```

const userSession = useUserSession(pinia)

// 1. Check token validity before the app start
if (userSession.isLoggedIn) {
  try {
    // Do api request call to retrieve user profile.
    // Note that the api requests the json-server
    const { data: user } = await api.get('/api/users/me')
    userSession.setUser(user)
  } catch (err) {
    // delete stored token if it fails
    userSession.logoutUser()
  }
}

router.beforeEach((to) => {
  // 2. If the page requires auth, check if user is logged in
  // if not, redirect to login page.
  if (to.meta.requiresAuth && !userSession.isLoggedIn) {
    return {
      name: '/auth/login',
      // save the location we were at to come back later
      query: { redirect: to.fullPath },
    }
  }
})
}

```

افزودن متادیتا به مسیرها

`meta` شاید متوجه شده باشید که از ویژگی مسیر استفاده کردیم `to`. این یک ویژگی عالی برای افزودن ابرداده به مسیرها، مانند `layout`، `requiresAuth`، یا هر چیز دیگری است که اطلاعات دلخواه در مورد مسیرها است.

با تشکر از `unplugin-vue-router`، ما می‌توانیم به سادگی یک عنصر ریشه جدید `<route>` به فایل های خود `vue.*` که در `src/pages/.` دایرکتوری قرار دارند اضافه کنیم

بیا ببیند مثال وبلاگ را از بخش [Setup Your Project - Creating new pages](#) در نظر بگیریم.

مورد اول: تنظیم ابرداده در یک مسیر

ما می خواهیم صفحه جدیدی را اضافه کنیم که در آن `blog/new/` برای ایجاد مقاله قابل دسترسی است، که فقط برای کاربرانی که وارد شده اند قابل دسترسی خواهد بود، در حالی که بقیه وبلاگ برای همه کاربران قابل دسترسی خواهد بود:

ضربه شدید

```
my-vuero-quickstarter-project/
├─ src/
│   └─ pages/
│       └─ blog/           // blog nested routes
+   └─ └─ └─ new.vue       // the article creation page accessible at
  `-/blog/new`
│   └─ └─ └─ index.vue    // the articles listing page accessible at
  `-/blog/`
│   └─ └─ └─ [slug].vue   // the article detail page accessible at
  `-/blog/:slug`
│   └─ └─ └─ blog.vue     // blog nested routes wrapper, should contain a
  `<<RouterView />`
```

در `src/pages/blog/new.vue` فایل، ما `requiresAuth` متادیتا را به مسیر اضافه می کنیم:

vue

```
<route lang="yaml">
meta:
  requiresAuth: true
</route>

<script setup lang="ts">
// the article creation script
</script>

<template>
  <!-- the article creation template -->
</template>
```

مورد دوم: تنظیم ابرداده بر روی مسیرهای تودرتو

ما می خواهیم کل `*/blog/` صفحات خود را به کاربران وارد شده محدود کنیم:


```

my-vuero-quickstarter-project/
├─ src/
│   └─ pages/
│       └─ blog/                // blog nested routes
│           └─ index.vue        // the articles listing page accessible at
`/blog/`
│       └─ [slug].vue           // the article detail page accessible at
`/blog/:slug`
+ └─ blog.vue                  // blog nested routes wrapper, should contain a
  `<RouterView />`

```

در `src/pages/blog.vue` فایل، ما `requiresAuth` متادیتا را به مسیر wrapper اضافه می کنیم:

vue

```

<route lang="yaml">
meta:
  requiresAuth: true
</route>

<script setup lang="ts">
// we do not need to do anything special here, but we can!
</script>

<template>
  <LandingLayout theme="light">
    <RouterView />
  </LandingLayout>
</template>

```

سفارشی سازی vue پیشرفته

با استفاده از VueUse

Vueuse مجموعه ای از ابزارهای ضروری Vue Composition API برای ساخت برنامه بعدی Vue شما است.

ما به شما توصیه می کنیم از این کتابخانه برای جلوگیری از اختراع مجدد چرخ استفاده کنید، کتابخانه به خوبی مستند شده است و ویژگی های مفید زیادی دارد که به چند مورد اشاره می کنیم:

- `useShare` : Reactive [Web Share API](#)
- `useWebNotification` : رابط [Web Notification API Notifications](#)
- `onClickOutside` : به کلیک های خارج از یک عنصر گوش دهید
- `useIntersectionObserver` : نمایان بودن عنصر هدف را تشخیص می دهد.

نکته

اسناد vueuse را در وب سایت رسمی بخوانید:

[/https://vueuse.org](https://vueuse.org)

ثبت اجزای جهانی

در Vuerو، می توانید به راحتی کامپوننت های جهانی را راه اندازی کنید یا با ایجاد یک افزونه، دامنه جهانی vue را گسترش دهید. ما به شما توصیه می کنیم که یک پلاگین واحد را با ایمپورت ایجاد کنید، به عنوان مثال یکی برای vue-tippy، دیگری برای v-calendar و غیره...

```
import { plugin as VueTippy } from "vue-tippy";

import { definePlugin } from '@src/app'

export default definePlugin(({ app }) => {
  // register the vue-tippy plugin globally
  app.use(VueTippy, {
    component: "Tippy",
    defaultProps: {
      theme: "light",
    },
  });

  // register global components
  // here we use defineAsyncComponent, so the component will be lazyloaded
  app.component(
    "VCalendar",
    defineAsyncComponent({
      loader: () => import("v-calendar").then((mod) => mod.Calendar),
    })
  );
});
```

```
    delay: 0,
    suspensible: false,
  })
};

// register global directives
app.directive("focus", {
  mounted: (el) => el.focus(),
});
}
```

Vue و Typescript ؟

نکته

مشاهده دوره کامل موجود در Vue Mastery :

- <https://www.vuemastery.com/courses/vue3-typescript/why-vue-&-typescript>
- <https://www.vuemastery.com/courses/typescript-friendly-vue3/introduction-to-the-script-setup-syntax>

Vue and Typescript | When to use it



تایپ اسکریپت فقط گسترش جاوا اسکریپت است، اگر در آن تازه کار هستید، نترسید زیرا تمام کدهای معتبر جاوا اسکریپت نیز کد TypeScript هستند. ([مستندات TypeScript](#)) مزایای اصلی استفاده از Typescript عبارتند از:

- کد شما را زودتر از موعد تأیید می کند
- تکمیل خودکار را ارائه می دهد
- از چک کردن نوع پیچیده پشتیبانی می کند

Vuero کاملاً با Typescript سازگار است. بنابراین، شما از مزایای بررسی تایپ و تکمیل کد بهبود یافته بهره مند خواهید شد! اگر از نماد تایپ اسکریپت در همه اجزای خود استفاده می کنید، می توانید `vue-tsc` کامپایلر را اجرا کنید تا برنامه خود را از نظر خطا بررسی کند.

بیایید یک ساده `MyCustomInput` با ویژگی های Typescript ایجاد کنیم:

```
vue
<script setup lang="ts">
export interface MyCustomInputProps {
  modelValue: string;
  label?: string;
  id?: string;
  disabled?: boolean;
}
export interface MyCustomInputEmits {
  (e: "update:modelValue", value: string): void;
  (e: "disable"): void;
}

// withDefaults and defineProps are global helpers from @vue/sfc-compiler
const props = withDefaults(defineProps<MyCustomInputProps>(), {
  label: undefined,
  id: undefined,
});

// defineEmits is global helpers from @vue/sfc-compiler
const emit = defineEmits<MyCustomInputEmits>();

// this is our method that emits the event
function onInputChanged(event: Event) {
  emit("update:modelValue", (event.target as HTMLInputElement).value);
}
```

```

</script>

<template>
  <label v-if="props.label" :for="props.id">
    {{ props.label }}
  </label>

  <input
    :value="props.modelValue"
    :disabled="props.disabled"
    @change="onInputChanged"
  />

  <VButton v-if="!props.disabled" @click="emit('disable')">disable me!
</VButton>
</template>

```

ما می توانیم تأیید کنیم که مؤلفه ما مطابق انتظار کار می کند، به سادگی در مؤلفه دیگری بارگذاری می شود

```

<template>
  <MyCustomInput :model-value="42" />
</template>

```

vue

برای تأیید اینکه `vue-tsc` کامپایلر کار می کند، دستور زیر را اجرا کنید:

```
pnpm test:tsc
```

ضربه شدید

توجه داشته باشید که `test:tsc` دستور ناموفق خواهد بود، زیرا ما یک عدد را در prop تنظیم کرده ایم

`. model-value:`

باید خطای زیر را خروجی دهد:

```

src/pages/index.vue:13:23 - error TS2322: Type 'number' is not assignable to
type 'string'.

```

ضربه شدید

هشدار

در برنامه های بزرگ، `test:tsc` ممکن است حافظه آن تمام شود. با اجرای دستور زیر می توانید محدودیت حافظه را افزایش دهید:

```
NODE_OPTIONS=--max_old_space_size=4096 pnpm test:tsc
```

ضربه شدید

آن را با لنگر تمیز نگه دارید

از آنجایی که ما عاشق کد پاک هستیم، 4 لیتر را پیکربندی کرده ایم که هدف خاص خود را دارند:

- `eslint`: جلوگیری از نگرانی های مربوط به کیفیت کد (بدون متغیرهای استفاده نشده، ...)
- `stylelint`: جلوگیری از نگرانی های کیفیت CSS (بدون رنگ نامعتبر، ...)
- `prettier`: قوانین قالب بندی را کنترل می کند (حداکثر طول خط، ...)
- `commitlint`: قالب بندی پیام های `commit` را کنترل می کند

می توانید کیفیت کد خود را با اجرا بررسی کنید

```
pnpm test:lint
```

ضربه شدید

لینترها به تنهایی می توانند بسیاری از مشکلات را برطرف کنند. برای انجام این کار، دویدن را امتحان کنید

```
pnpm lint
```

ضربه شدید

کاربران VSCode

پسوندهای توصیه شده را نصب کنید، پس از آن هر بار که فایل ها ذخیره می شوند، پرز ایجاد می شود!

آخرین به روز رسانی:

[این صفحه را در GitHub ویرایش کنید](#)

صفحه بعد
[سفارشی سازی Vuerو](#)

صفحه قبلی
[Vite و Vuerو](#)



مرکز اسناد



Vuero را یاد بگیرید 

پروژه خود را راه اندازی کنید

با شروع سریع شروع کنید

با ایجاد یک پوشه جدید (به عنوان مثال `my-vuero-quickstarter-project`) شروع کنید و `quickstarter-vuero-v2.9.0.zip` محتوای آرشیو را در آن استخراج کنید.

ضربه شدید

```
# Create a new folder
mkdir my-vuero-quickstarter-project

# Extract the quickstarter archive
unzip quickstarter-vuero-vv2.9.0.zip -d my-vuero-quickstarter-project

# Go to the newly created folder
cd my-vuero-quickstarter-project
```

▼ یک مخزن git را راه اندازی کنید

توصیه می کنیم یک مخزن git جدید برای پروژه خود راه اندازی کنید و اولین commit خود را در این مرحله ایجاد کنید.

ضربه شدید

```
# Create a new folder
git init

# Add all files to git
git add .
```



```
# Create your first commit
git commit -m "Initial commit"
```

هشدار

به خاطر داشته باشید که اگر یک مخزن git جدید ایجاد می‌کنید، مخزن خود را خصوصی کنید، زیرا این الگو یک محصول پولی است.

نمای کلی پروژه

ضربه شدید

```
my-vuero-quickstarter-project/
├─ public/                # static files (robots.txt, favicon.ico, etc.)
├─ src/
│   ├─ assets/            # static files, will be processed by vite (e.g.
optipng, svgo, etc.)
│   ├─ components/        # global components
│   ├─ composable/        # reusable composition functions
│   ├─ directives/        # global vuejs directives
│   ├─ layouts/           # layout components
│   ├─ locales/           # global i18n locales
│   └─ pages/             # pages components, each file will be accessible
as a route
│   ├─ scss/              # scss files
│   ├─ stores/            # pinia stores
│   ├─ utils/             # utility functions
│   ├─ plugins/           # router guards, vue plugins installations, etc.
│   └─ app.ts             # vuero initialization (head, i18n, router,
pinia, etc.)
│   ├─ entry-client.ts    # client entry point
│   ├─ entry-server.ts    # SSR entry point
│   ├─ router.ts          # base vue-router configuration
│   ├─ styles.ts          # stylesheet configuration (scss, vendor, etc.)
│   └─ VueroApp.vue       # vuero root component
├─ .env                   # environment variables available to the project
├─ index.html             # main entry point (loads src/entry-client.ts)
├─ package.json           # project dependencies
└─ tsconfig.json          # typescript configuration
```

```
└─ server.ts # Node.js SSR server
└─ vite.config.ts # vite plugins and configuration
```

این یک نمای کلی از مهمترین فایل ها و پوشه های پروژه شما است. سایر فایل ها برای لینتر، تست، داکر و غیره هستند.

نصب وابستگی ها

با اجرای یکی از دستورات زیر، وابستگی های پروژه را نصب کنید:

```
pnpm install
```

حالت توسعه را شروع کنید

برای راه اندازی سرور توسعه، یکی از دستورات زیر را اجرا کنید:

```
pnpm dev
```

با این کار هم اسکریپت ها و هم `dev:json-server` از فایل اجرا می شود `package.json`.

متوجه خواهید شد که دو سرور راه اندازی شده اند: یکی برای (vite) frontend و دیگری برای (json-backend server).

- اسکریپت `dev:vite` سرور (vite) frontend را شروع می کند. Vite ابزار ساختی است که ما از آن برای کامپایل کد فرانت اند استفاده می کنیم. این جایگزین وب پک و `vue-cli` است که در اکوسیستم `vue 2` استفاده می شود.

بیشتر در مورد آن در vitejs.dev بخوانید

- اسکریپت `dev:json-server` سرور (vite) frontend را شروع می کند. `Json-server` یک سرور REST-API جعلی است که ما از آن برای شبیه سازی backend استفاده می کنیم. پیکربندی و پایگاه داده در `json-/server` دایرکتوری هستند. نحوه استفاده ما از آن را می توانید در منبع کامل قالب در فایل ها

[اطلاعات بیشتر در مورد](#) `src/composable/useChatApi.ts/` و `src/pages/messaging-v1.vue/`

[آن را در github.com/typicode/json-server](https://github.com/typicode/json-server)

بخوانید

نکته

- در مرورگر خود به قسمت واسط Vuero دسترسی پیدا کنید `/http://localhost:3000`
- دسترسی به باطن سرور json در مرورگر خود در `/http://localhost:8080`

هشدار

اگر هنگام نصب با مشکلی مواجه شدید، [مشکلات رایج را بررسی کنید](#) یا در [پورتال پشتیبانی ما](#) یا [ما تماس بگیرید](#)

API جعلی با JSON-Server

[ممکن است نام سرویس jsonplaceholder.typicode.com](#) را شنیده باشید که یک REST-API جعلی برای اهداف آزمایشی ارائه می دهد. Vuero همچنین از بسته [json-server](#) برای ارائه یک REST-API سفارشی استفاده می کند. این ما را قادر می سازد تا پیاده سازی هایی را در دنیای واقعی ارائه دهیم. به طور پیش فرض، سرور json را در حالت فقط خواندنی راه اندازی می کنیم، اما می توانید با ویرایش `package.json` فایل، آن را تغییر دهید.

هنگامی که اسکریپت را اجرا می کنید `dev:json-server`، پیامی را در کنسول خواهید دید که به شما در مورد نقاط پایانی موجود اطلاع می دهد:

```
\{^_^}/ hi!
```

```
Loading ./json-server/db.json
```

```
Loading ./json-server/routes.json
```

```
Done
```

```
Resources
```

```
http://localhost:8080/conversations
```

```
http://localhost:8080/messages
```

```
http://localhost:8080/companies
```

```
http://localhost:8080/users
```

```
Other routes
```

```
/api/* -> /$1
```

```
/users/me -> /users/3
```

```
/conversations/:cid/messages -> /messages?conversationId=:cid  
/conversations/:cid/messages/:mid -> /messages?  
conversationId=:cid&messageId=:mid  
  
Home  
http://localhost:8080
```

مصرف api

همانطور که می بینید، json-server روی پورت 8080 اجرا می شود. می توانید در مرورگر خود به آن دسترسی داشته باشید `http://localhost:8080`. این همچنین به این معنی است که شما می توانید از هر مشتری http برای مصرف api استفاده کنید. برای مثال، می توانید از **axios** یا مستقیماً از **Fetch API** استفاده کنید.

خوشبختانه ما یک **composable** `useFetch` ارائه می کنیم، که یک بسته بندی در اطراف واکنش بومی با رهگیرهای سفارشی برای احراز هویت، از طریق توکن های JWT Bearer، و `baseURL` تنظیم روی `VITE_API_BASE_URL` متغیر محیطی است. اطلاعات بیشتر در مورد **Vuero** و **Vue 3** - استفاده مجدد از منطق **stateful** یا بخش **composable**.


یک مثال خوب از نحوه بارگیری داده های همگام سازی از یک api، صفحه **پیام رسانی v1** است. (منابع را می توانید در قالب کامل در `src/pages/messaging-v1.vue/` ببابید. در این بخش نحوه تقسیم داده ها را به دو بخش خواهید یافت: وضعیت برنامه (`useChat` فروشگاه پینیا) و api مربوطه (`useChatApi` قابل ترکیب).

برای سرعت بخشیدن به توسعه، می توانید api را با داده های خود گسترش دهید، اما از تکیه بر آن در تولید خودداری کنید. اطلاعات بیشتر در مورد آن را در صفحه github بخوانید:

<https://github.com/typicode/json-server>

با api واقعی جایگزین کنید

برای زنده کردن برنامه خود باید یک Backend برای احراز هویت کاربر، داده ها و غیره ایجاد کنید.

می توانید به پروژه هایی مانند **supabase** ، **nitro** یا **strapi**، که باطن منبع باز هستند، نگاهی بیندازید که می توان به خوبی با **Vuero** از آنها استفاده کرد (**storyblok** نیز می تواند انتخاب خوبی باشد)!

هنگامی که آماده جایگزینی api جعلی با یک Api واقعی هستید، می توانید مراحل زیر را دنبال کنید:

1. `VITE_API_BASE_URL` فایل موجود در فایل را ویرایش کنید `env.` تا به api خود اشاره کنید.

توجه داشته باشید که می توانید این متغیر محیطی را در `env.local.` فایل لغو کنید.

2. json-server را از `package.json` فایل حذف کنید.

json

```
{
  "scripts": {
+    "dev": "vite --open",
-    "dev": "run-p dev:vite dev:json-server",
-    "dev:vite": "vite --open",
-    "dev:json-server": "json-server --read-only --routes ./json-
server/routes.json --port 8080 --delay 200 --watch ./json-server/db.json",
+    "preview": "serve dist -s -p 5000",
-    "preview": "run-p preview:vite preview:json-server",
-    "preview:vite": "serve dist -s -p 5000",
-    "preview:json-server": "json-server --read-only --routes ./json-
server/routes.json --host 0.0.0.0 --port 8080 ./json-server/db.json",
  },
  "devDependencies": {
-    "json-server": "0.17.0",
  },
}
```

نکته

آیا حذف آن و استفاده از GraphQL API کاملاً خوب است، در این صورت می توانید به <https://v4.apollo.vuejs.org> نگاهی بیندازید.

تمدید راه اندازی سریع

آناتومی صفحات Quick Starter

راه اندازی سریع دارای چند صفحه است که می توانید برای شروع پروژه خود از آنها استفاده کنید.

- یک صفحه فرود که می توانید از آن برای معرفی پروژه خود استفاده کنید، همه می توانند به آن دسترسی داشته باشند.

- بخش احراز هویت با فرم های ورود/ثبت نام.
- یک صفحه خصوصی که برای دسترسی به آن باید وارد شوید.
- و یک صفحه 404 زمانی که هیچ جزء منطقی برای مسیر درخواستی یافت نشد.

این هم نمای کلی صفحات:

ضربه شدید

```
my-vuero-quickstarter-project/
├─ src/
│   └─ pages/
│       └─ app/                # app nested routes
│           └─ index.vue      # the app page accessible at `/app/`
│       └─ auth/              # auth nested routes
│           └─ index.vue      # the auth page accessible at `/auth/`
│           └─ login.vue      # the auth-login page accessible at
`/auth/login`
│           └─ signup.vue     # the auth-signup page accessible at
`/auth/signup`
│       └─ [...all].vue       # the catch all page (404)
│       └─ index.vue          # the index page accessible at `/`
│       └─ app.vue            # optional app nested routes wrapper, should
contain a `

```

مسیر به طور خودکار از `/pages` پوشه تولید می شود، این کار با `unplugin-vue-router` افزونه انجام می شود. می توانید اطلاعات بیشتری در مورد این افزونه در صفحه *github* آن در اینجا بخوانید:

<https://github.com/posva/unplugin-vue-router>

همانطور که می بینید، برخی از صفحات به طور مستقیم قابل دسترسی نیستند، اما برای مسیرهای تودرتو بسته بندی هستند. این به ما این امکان را می دهد که طرح بندی را برای کل بخش برنامه خود تنظیم کنیم. می توانید اطلاعات بیشتری در مورد نحوه کار آنها بر روی اسناد رسمی *vue-router* در اینجا بخوانید:

<https://router.vuejs.org/guide/essentials/nested-routes.html>

ایجاد صفحات جدید

بیایید تصور کنیم که می خواهیم یک بخش وبلاگ جدید در برنامه خود ایجاد کنیم. ما می خواهیم وبلاگ ما در دسترس باشد `/blog/` و می خواهیم صفحه وبلاگی داشته باشیم که در دسترس باشد `blog/my-pretty-/`

ما باید حداقل دو فایل ایجاد کنیم:

ضربه شدید

```
my-vuero-quickstarter-project/
├─ src/
│   └─ pages/
+   │   │   └─ blog/           // blog nested routes
+   │   │   │   └─ index.vue   // the articles listing page accessible at
+   │   │   │   │   └─ `./blog/`
+   │   │   │   └─ [slug].vue  // the article detail page accessible at
+   │   │   │   │   └─ `./blog/:slug`
```

1. فایل را ایجاد کنید `src/pages/blog/index.vue`

vue

```
<script setup lang="ts">
// we import our useFetch helper
import { useFetch } from '@src/composable/useFetch'

// We may want to retrieve the posts from an API
// as we are using typescript, it is a good practice to always define our
types
interface Article {
  id: string
  title: string
  slug: string
}

const $fetch = useFetch()

// articles and fetchArticles variables can be provided by a composable
function
const articles = ref<Article[]>([]) // we know that the articles will be an
array of Article

async function fetchArticles() {
  try {
    articles.value = await $fetch<Article[]>('/articles') // we know that
our api respond with an array of Article
```

```

    } catch (error) {
      // here we can handle the error
      console.error(error)
    }
  }

// We trigger the fetchArticles function when the component is mounted
useEffect(fetchArticles)

// don't forget to setup our page meta
useHead({
  title: 'My blog',
})
</script>

<template>
  <LandingLayout theme="light">
    <div class="blog-list-wrapper">
      <!-- This is a simple page example -->
      <h1>My blog posts:</h1>
      <ul>
        <li v-for="article in articles" :key="article.id">
          <!-- Here we are linking to the article detail page with a dynamic
"slug" parameter -->
          <RouterLink
            :to="{
              name: '/blog/[slug]',
              params: {
                slug: article.slug,
              },
            }"
          >
            {{ article.title }}
          </RouterLink>
        </li>
      </ul>
    </div>
  </LandingLayout>
</template>

<style lang="scss" scoped>

```



```

.blog-list-wrapper {
  // Here we can add custom styles for the blog page
  // They will be only applied to this component
}
</style>

```

2. فایل را ایجاد کنید `src/pages/blog/[slug].vue`

```

<script setup lang="ts">
// we import our useFetch helper
import { useFetch } from '@src/composable/useFetch'

interface Article {
  id: string
  title: string
  slug: string
  content: string
  comments: string[]
}

const article = ref<Article>()
const loading = ref(false)
const $fetch = useFetch()
const router = useRouter()
const route = useRoute()

// Trigger the function when the slug changes
watchEffect(async () => {
  const slug = (route.params?.slug as string) ?? ''

  loading.value = true

  try {
    const data = await $fetch<Article[]>(`/articles`, {
      query: {
        slug,
      }
    })

    if (!data?.length) {

```

```

        throw new Error('Article not found')
    }

    article.value = data[0]
  } catch (error) {
    // If the article does not exist, we replace the route to the 404 page
    // we also pass the original url to the 404 page as a query parameter
    // http://localhost:3000/article-not-found?original=/blog/a-fake-slug
    router.replace({
      name: '/[...all]', // this will match the ./src/pages/[...all].vue
route
      params: {
        all: 'article-not-found',
      },
      query: {
        original: router.currentRoute.value.fullPath,
      },
    })
  } finally {
    loading.value = false
  }
})

// Setup our page meta with our article data
useHead({
  title: computed(() => article.value?.title ?? 'Loading article...'),
})
</script>

<template>
  <LandingLayout theme="light">
    <div v-if="loading">Loading article...</div>
    <div v-else class="blog-detail-wrapper">
      <!--
        Page content goes here

        You can see more complete pages content samples from
        files in /src/components/pages directory
      -->

      <h1>{{ article?.title }}</h1>

```

```

    <div>{{ article?.content }}</div>
  </div>
</LandingLayout>
</template>

<style lang="scss" scoped>
.blog-detail-wrapper {
  // Here we can add custom styles for the blog page
  // They will be only applied to this component
}
</style>

```

نکته

برای شروع سفارشی‌سازی صفحات خود، می‌توانید محتوای موجود در پوشه یا فایل‌ها `/src/components/pages` را بررسی کنید `src/pages/wizard-v1.vue` ، `src/pages/inbox.vue` ، `src/pages/auth/login-1.vue` تا `src/pages/marketing-1.vue` ببینید چگونه می‌توانید عناصر را برای ایجاد برنامه‌های عالی مرتب کنید!

افزودن داده های جعلی

در این مرحله می‌توانید بخش وبلاگ را مرور کنید `/blog/` و صفحه وبلاگ را با مقالات لیست شده مشاهده خواهید کرد. در واقع هیچ مقاله ای نشان داده نخواهد شد تا زمانی که تعدادی را به Fake API اضافه کنید.

برای انجام این کار، به سادگی این را در `json-server/db.json/` فایل اضافه کنید:

json

```

{
+   "articles": [
+     {
+       "id": 1,
+       "slug": "my-first-post",
+       "title": "My First Post"

```

بازگشت به بالا

منو

```

+     {
+       "id": 2,
+       "slug": "my-pretty-blog",
+       "title": "My pretty blog",
+       "content": "Vue.js is awesome!"

```

```
+   },
+ ],
  "conversations": [
  ]
}
```

شما می توانید با جستجوی خود در مورد `blog/my-first-post/`، صفحه جزئیات مقاله را مشاهده خواهید کرد در حالی که با رفتن به `blog/random-slug/` صفحه 404 را نشان می دهد.

استفاده از مسیر تودرتو برای استفاده مجدد از طرح بندی ها

ممکن است متوجه شده باشید که هر دو صفحه ما دارای یک مولفه طرح بندی هستند `LandingLayout` `<"theme="light`. اگر هر دو صفحه طرح بندی متفاوتی داشته باشند، می تواند خوب باشد، اما در این مورد، طرح بندی در هر تغییر صفحه از حالت نصب خارج می شود و دوباره نصب می شود (به عنوان مثال، هنگام رفتن از `blog/` به `blog/my-first-post/`، یا از `blog/` اولین پست من به `blog/my-pretty-blog/`). این می تواند باعث سوسو زدن و مشکلات عملکرد شود.

برای جلوگیری از این امر، می توانیم با افزودن یک پوشش در اطراف صفحات، از قدرت مسیرهای تودرتو استفاده کنیم `*/blog/`. برای انجام این کار، ما فقط باید فایلی ایجاد کنیم که نام آن برگرفته از دایرکتوری باشد، در این مورد `src/pages/blog.vue`.

ضربه شدید

```
my-vuero-quickstarter-project/
├─ src/
│   └─ pages/
│       └─ blog/           // blog nested routes
│           └─ index.vue  // the articles listing page accessible at
`-/blog/`
│   └─ [slug].vue         // the article detail page accessible at
`-/blog/:slug`
+ └─ blog.vue             // blog nested routes wrapper, should contain a
  <RouterView />
```

1. حذف `<"LandingLayout theme="light` از

`src/pages/blog/[slug].vue` و `src/pages/blog/index.vue`

2. فایل `src/pages/blog.vue` را با محتوای زیر ایجاد کنید :

```
<script setup lang="ts">
// we do not need to so anything special here, but we can!
</script>

<template>
  <LandingLayout theme="light">
    <RouterView />
  </LandingLayout>
</template>
```

هشدار

این مؤلفه باید دارای یک `</ RouterView>` باشد ، که صفحه تودرتوی صحیح را بر اساس مسیر فعلی ارائه می دهد.

نکته

همچنین می‌توانید انتقال صفحه را در اینجا اعلام کنید (`src/scss/abstracts/_transitions.scss` برای اطلاعات بیشتر در مورد انتقال‌های موجود به فایل مراجعه کنید) :

```
<template>
  <RouterView v-slot="{ Component }">
    <Transition name="fade-fast" mode="out-in">
      <component :is="Component" />
    </Transition>
  </RouterView>
</template>
```

و ویلا! شما اولین صفحات خود را با مسیرهای پویا و بارگذاری ناهمزمان داده ایجاد کرده اید.

استفاده از اجزای قالب کامل

هشدار

بازیابی اجزا از قالب کامل باید با دقت انجام شود، زیرا می تواند حاوی `<RouterLink>` صفحه ای باشد که در پروژه شما وجود ندارد.

با استفاده از یک طرح از قالب کامل

Layout ها فقط اجزایی هستند که دارای یک اسلات پیش فرض هستند. آنها بیشتر برای پیچیدن مسیرهای تو در تو استفاده می شوند.

شما باید یک طرح را از یکی از موجود انتخاب کنید

Sidebar Layout

`src/layouts/SidebarLayout.vue`

نام	موضوع	نسخه ی نمایشی
نوار کناری معمولی	default	نسخه ی نمایشی
نوار کناری منحنی	curved	نسخه ی نمایشی
نوار کناری رنگی	color	نسخه ی نمایشی
نوار کناری رنگی منحنی	color-curved	نسخه ی نمایشی
نوار کناری برچسب ها	labels	نسخه ی نمایشی
برچسب ها نوار کناری شناور	labels-hover	نسخه ی نمایشی
نوار کناری شناور	float	نسخه ی نمایشی

NavbarLayout

`src/layouts/NavbarLayout.vue`

نام	موضوع	نسخه ی نمایشی
نوار ناوبری معمولی	default	نسخه ی نمایشی
محو شدن نوار ناوبری	fade	نسخه ی نمایشی
نوار ناوبری رنگی	colored	نسخه ی نمایشی

NavbarDropdownLayout

`src/layouts/NavbarDropdownLayout.vue`

نام	موضوع	نسخه ی نمایشی
نوار ناوبری کشویی	default	نسخه ی نمایشی
نوار ناوبری کشویی رنگی	colored	نسخه ی نمایشی

NavbarSearchLayout

`src/layouts/NavbarSearchLayout.vue`

نام	موضوع	نسخه ی نمایشی
نوار ناوبری را پاک کنید	default	نسخه ی نمایشی
Clean Center Navbar	center	نسخه ی نمایشی
نوار ناوبر محو را پاک کنید	fade	نسخه ی نمایشی

SideblockLayout

`src/layouts/SideblockLayout.vue`

نام	موضوع	نسخه ی نمایشی
بلوک جانبی معمولی	default	نسخه ی نمایشی
بلوک جانبی منحنی	curved	نسخه ی نمایشی
بلوک کناری رنگی	color	نسخه ی نمایشی
بلوک کناری رنگی منحنی	color-curved	نسخه ی نمایشی

طرح را در `src/layouts` فهرست پروژه خود کپی کنید.

توجه داشته باشید که نیز وجود دارد `MinimalLayout` ، `AuthLayout` و `MinimalLayoutLayout` در دسترس است. آنها را می توان برای صفحات فرود، auth یا هر چیز دیگری استفاده کرد.

اکنون باید تمام اجزای مرتبط آن را نیز به صورت دستی کپی کنید (ناوبری، پانل ها، و غیره...). می توانید آنها را در هر جایی از `src/components` فهرست خود کپی کنید.

نکته

ممکن است تصمیم بگیرید که برخی از مؤلفه‌ها را حذف کنید، مثلاً `</ LanguagesPanel>` اگر به آنها نیاز ندارید.

هشدار

شما باید تمام `<RouterLink>` پارامترها را جایگزین کنید. یک راه خوب برای انجام این کار این است که همه نام‌ها را جایگزین کنید /

IDE خود را راه اندازی کنید

VSCode و Volar

راه اندازی IDE توصیه شده `VSCode` با پسوند `Volar` است. `Volar` برجسته سازی نحو و `IntelliSense` پیشرفته را برای عبارات قالب، اجزای سازنده و حتی اعتبارسنجی اسلات ارائه می دهد. اگر می‌خواهید بهترین تجربه ممکن را با `Vue SFC` داشته باشید، قویاً این راه‌اندازی را توصیه می‌کنیم.

نکته

هنگامی که افزونه `Volar` را فعال کردید:

1. پوشه پروژه را در `VSCode` باز کنید و این راهنما را دنبال کنید تا حالت `Take Over Mode` را فعال کنید:

<https://github.com/johnsoncodehk/volar/discussions/471#discussioncomment-1361669>

2. دستور `VSCode` را اجرا کنید `...Volar: Select TypeScript Version` و سپس انتخاب کنید `Use Workspace`

Version

این مراحل باید در هر پروژه جدیدی که `Vue 3` ایجاد می کنید انجام شود.

ابزارهای توسعه `Vue.js`

اگر از یک مرورگر مبتنی بر `Chromium` استفاده می‌کنید، توصیه می‌کنیم افزونه `Vue.js devtools` را از هر فروشگاه اینترنتی نصب کنید: <https://devtools.vuejs.org/guide/installation.html>

آخرین به روز رسانی:

[این صفحه را در GitHub ویرایش کنید](#)

صفحه بعد
[Vite و Vuer](#)

صفحه قبلی
[شروع شدن](#)