Nuxt Fundamentals

Reusable components

Peter Kassenaar –
info@kassenaar.com

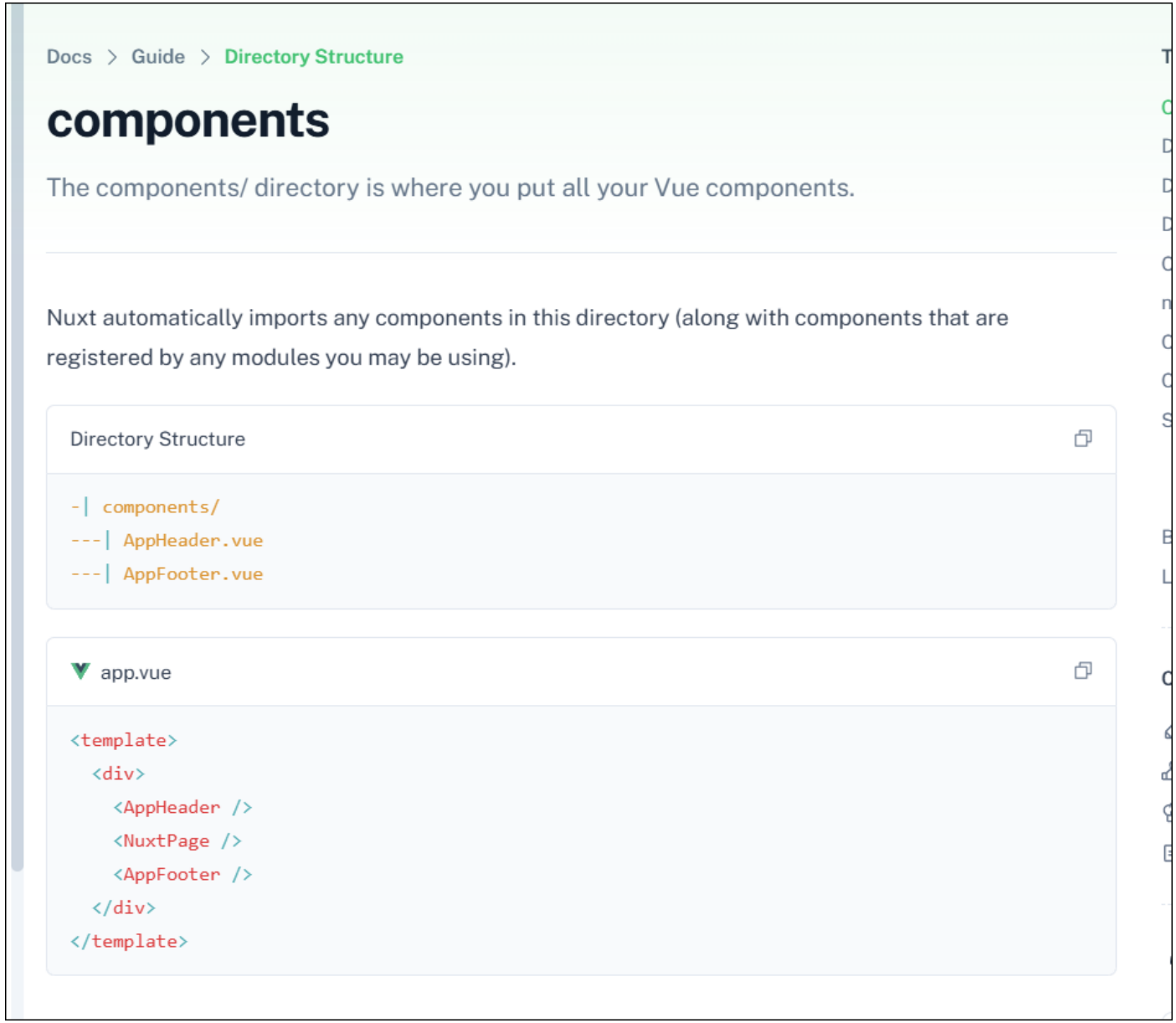
Reusable components

Using components multiple times in (possibly) different pages

Short recap - Components

- If you add Vue components to `./pages`, they become routes in your app
  - They look like a web ‘page’ to the outside world
- But often times a page is composed of multiple components.
  - As is already the case for `<MainNavigation/>` in our example.
- They should live in the `./components` folder
- Components you place here, are automatically imported by Nuxt, so you don’t have to write an `import … from` statement
- The path decides the component name!
  - So `./components/layout/myHeader.vue` becomes
  - `<LayoutMyHeader />` if used in your template

What are Components?



Docs > Guide > Directory Structure

components

The components/ directory is where you put all your Vue components.

Nuxt automatically imports any components in this directory (along with components that are registered by any modules you may be using).

Directory Structure

```
-| components/
---| AppHeader.vue
---| AppFooter.vue
```

app.vue

```
<template>
  <div>
    <AppHeader />
    <NuxtPage />
    <AppFooter />
  </div>
</template>
```

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

Creating a PostCard component

- As an example, instead of creating a `<ul>` list of Posts, we show each post as a `<PostCard />` component
- Again, we use Tailwind CSS classes we added in the previous workshop
- For the postcard, remember these issues:
  - Use `defineProps(['post'])` to receive a `post` property
  - Create a template, using and styling `<nuxt-link>` as desired.

<PostCard> component

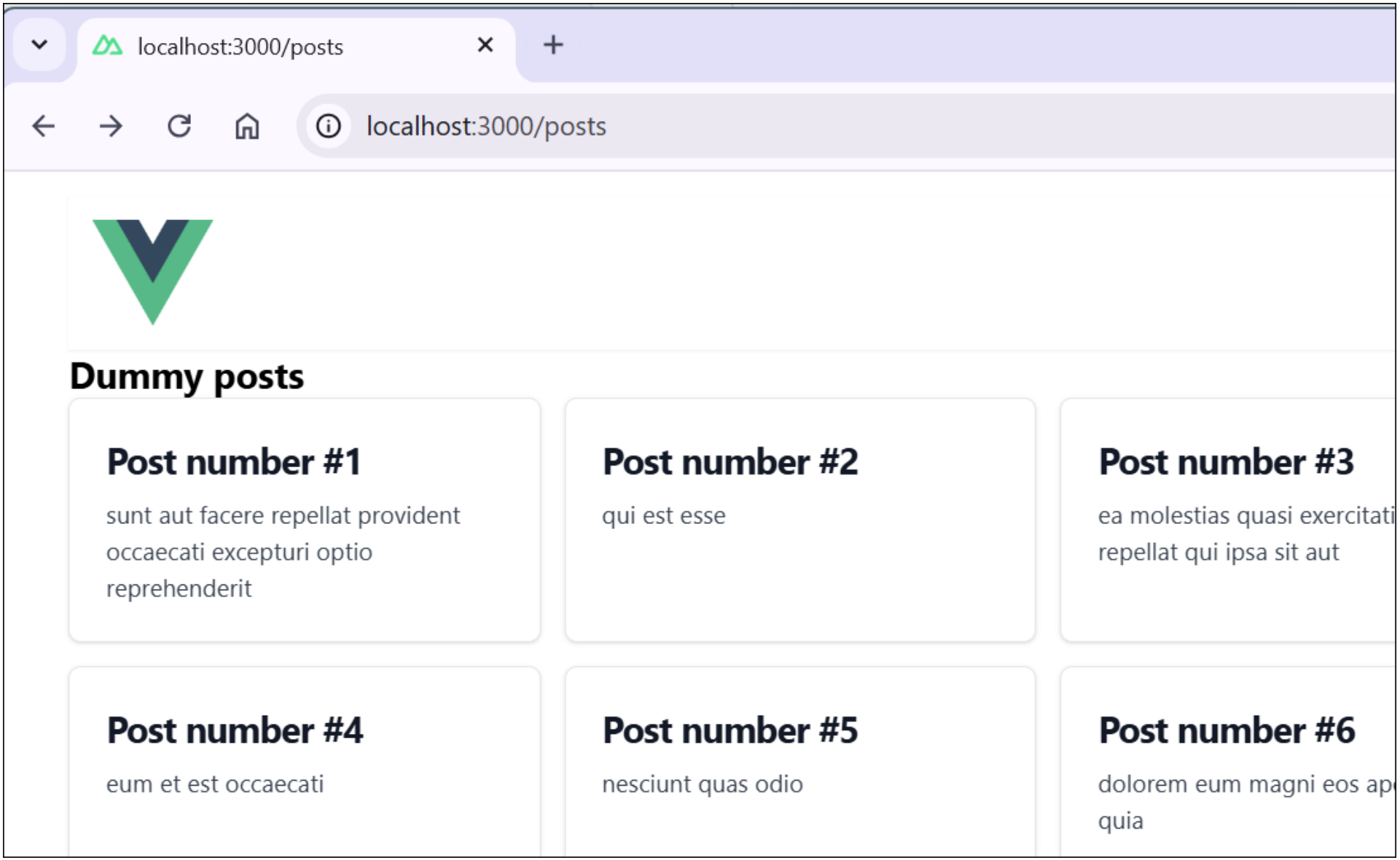
```
<script setup lang="ts">
const {post} = defineProps(['post'])
</script>

<template>
  <nuxt-link :to="`/posts/${post.id}`"
             class="block max-w-sm p-6 bg-white …">
    <h5 class="mb-2 text-2xl font-bold …">
      Post number #{{ post.id }}</h5>
    <p class="font-normal text-gray-700 dark:text-gray-400">
      {{ post.title }}
    </p>
  </nuxt-link>
</template>
```

Showing the cards

- We can now loop over the collection of Posts and assign each post to a `<PostCard>` component
  - Use the `v-for` directly on `<PostCard>` - no surrounding `<div>` needed
  - Give every post card a `:key`
  - Pass in each post to the `:post` property
  - Use desired Tailwind CSS classes

```
<!-- Posts list -->
<div v-else class="flex flex-wrap gap-4">
  <PostCard
      v-for="post in posts"
      :key="post.id"
      :post="post"
      class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 p-2"
  />
</div>
```

Browser like:



<PostDetail /> component

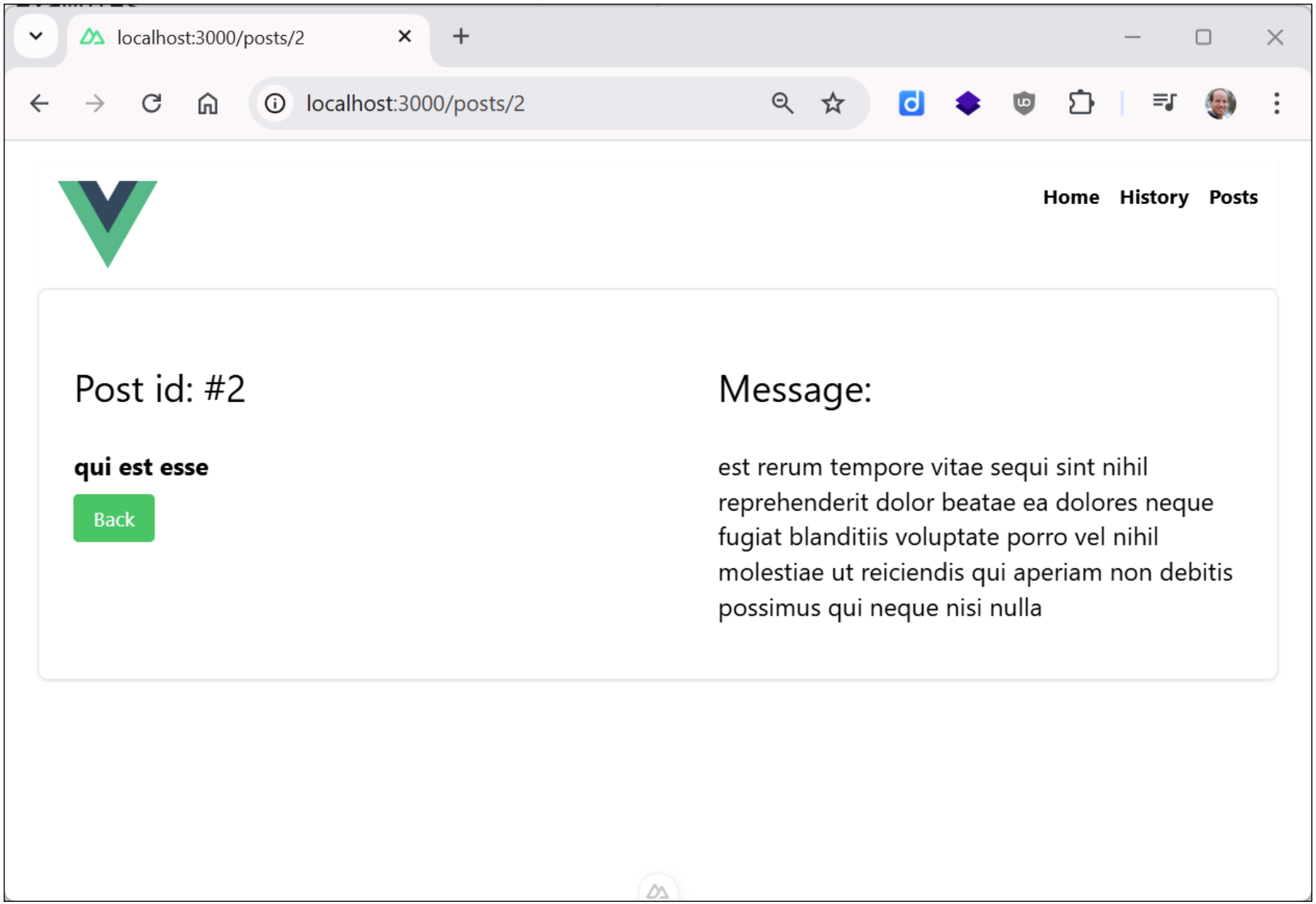
- Showing individual Posts in a detail page
- Repeat process
  - Create `./components/PostDetail.vue`
  - Create desired template (see next slide)
  - Pass in detail `prop(s)`
  - Replace template in `[id].vue` with `<PostDetail>` and forward props:

```
<template>
  <PostDetail v-if="!isLoading && post" :post="post"/>
</template>
```

<PostDetail> Component

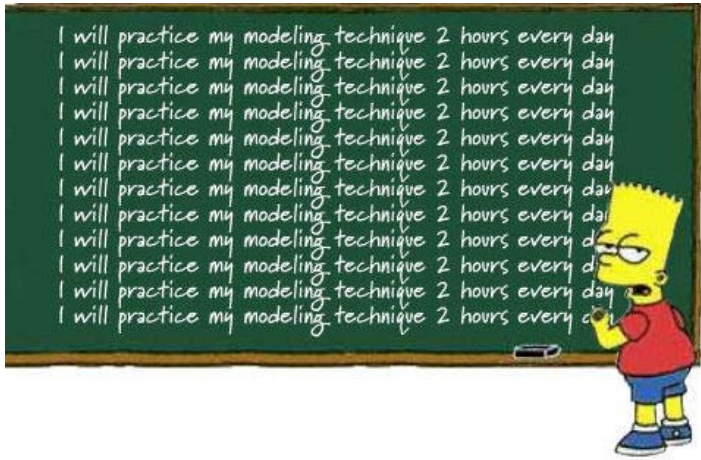
```
<script setup lang="ts">
  const {post} = defineProps(['post'])
</script>

<template>
  <div class="grid grid-cols-2 gap-10 border …">
    <div class="p-7">
      <h3 class="text-3xl my-8">Post id: #{{ post.id }}</h3>
      <p class="text-xl my-4"><strong>{{ post.title }}</strong></p>
      <nuxt-link to="/posts" class="btn btn-green">Back</nuxt-link>
    </div>
    <div class="p-7">
      <h3 class="text-3xl my-8">Message:</h3>
      <p class="text-xl my-4">{{ post.body }}</p>
    </div>
  </div>
</template>
```

Result


localhost:3000/posts/2
localhost:3000/posts/2
Home
History
Posts
Post id: #2
qui est esse
Back
Message:
est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla

Workshop

- Create one or more components in your app
- Transfer template and possibly script from Page → Component and update.
- (Re)use component in your page.
- Example: `../examples/150-reusable-components`



Checkpoint

- You know what reusable components are used for and how they are created
- You know the importance of the `./components` folder
- You are aware that Vue components in this folder are imported automatically
- You are able to create applications that use reusable components