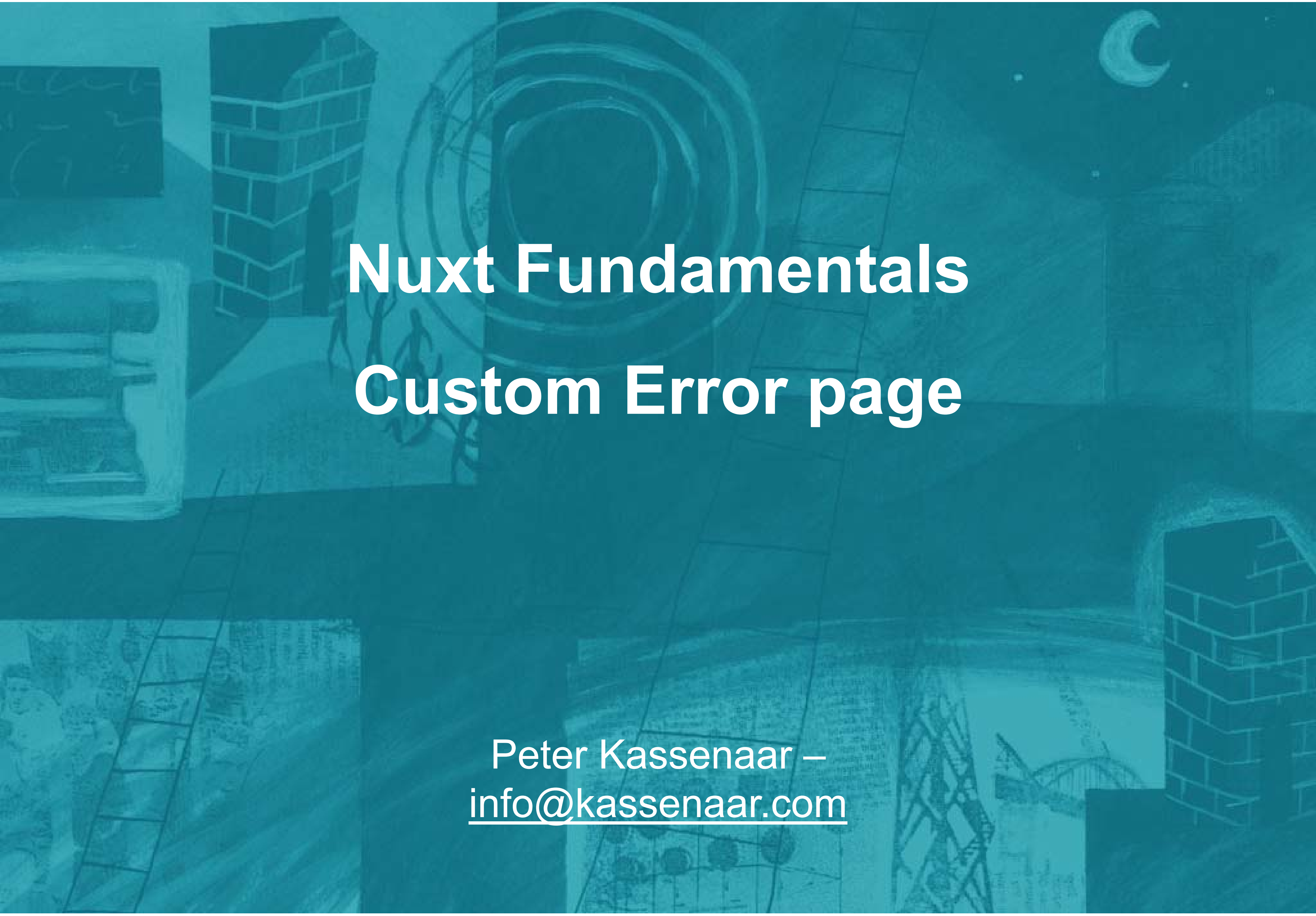


Nuxt Fundamentals

Custom Error page

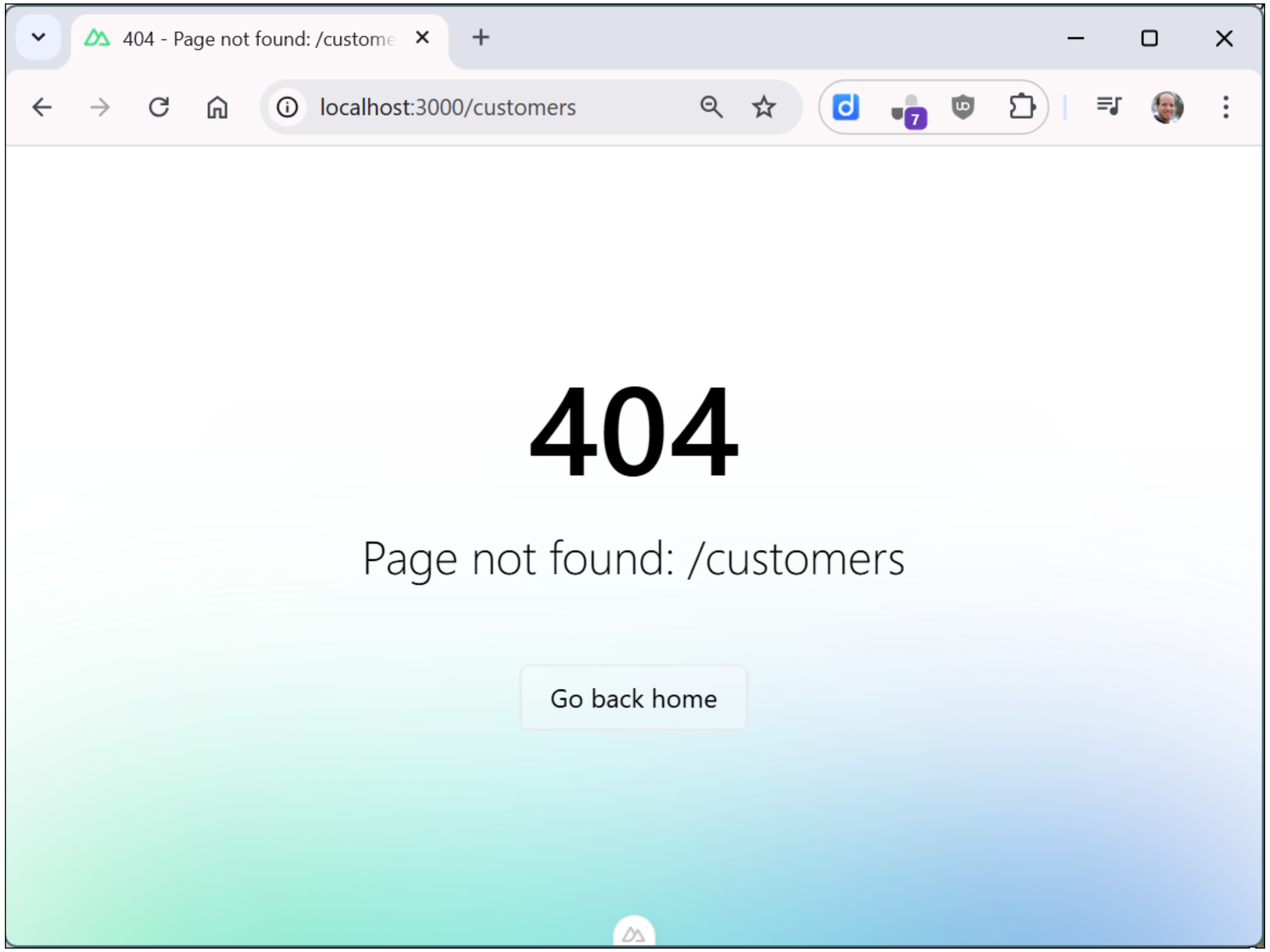
Peter Kassenaar –
info@kassenaar.com

Custom Error Pages

Creating an error / 404-page if something goes wrong

Default error page

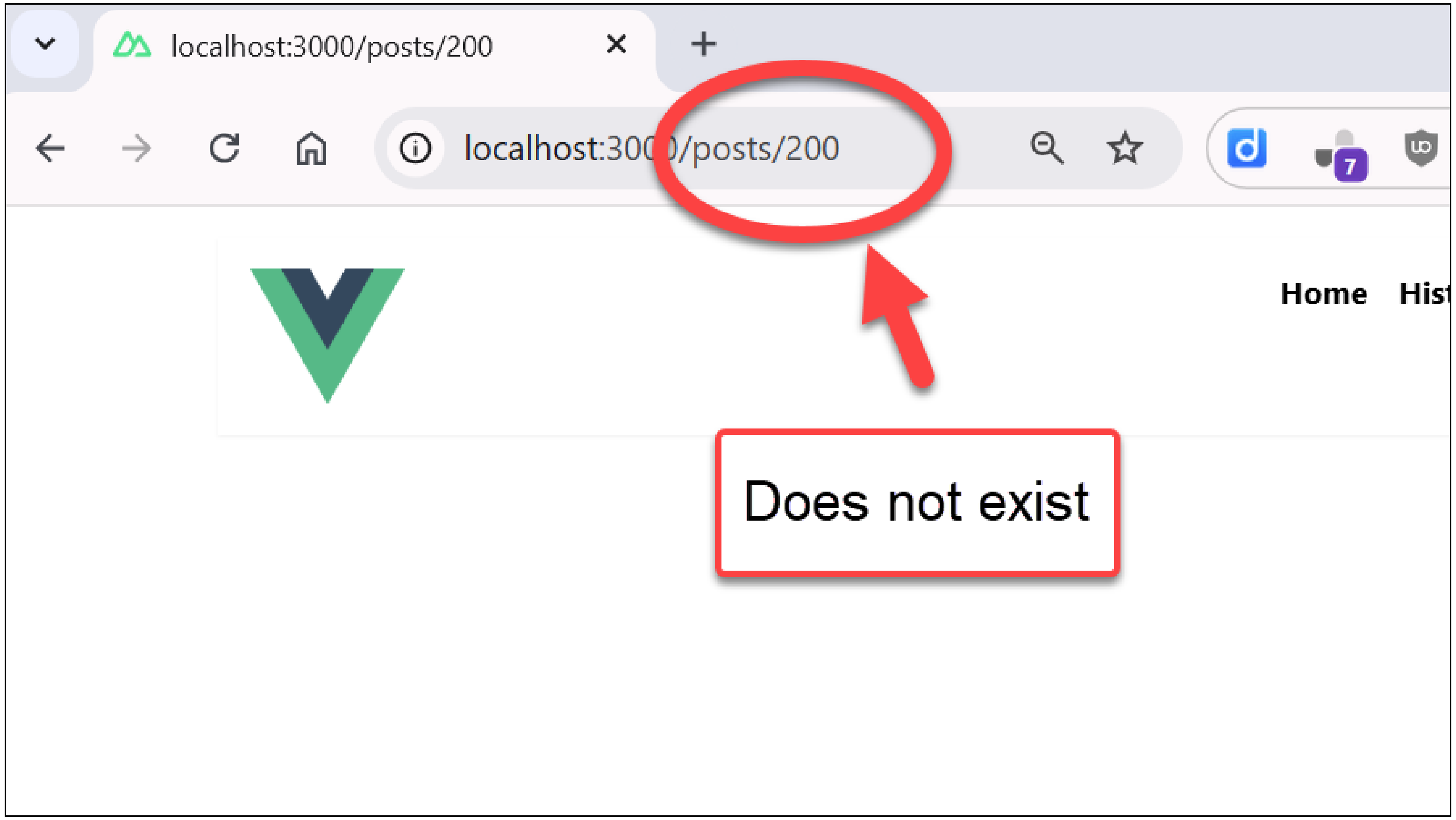
- If a non-existing route or invalid id is requested, the default Nuxt error page is shown:



404

Page not found: /customers

Non-existing id



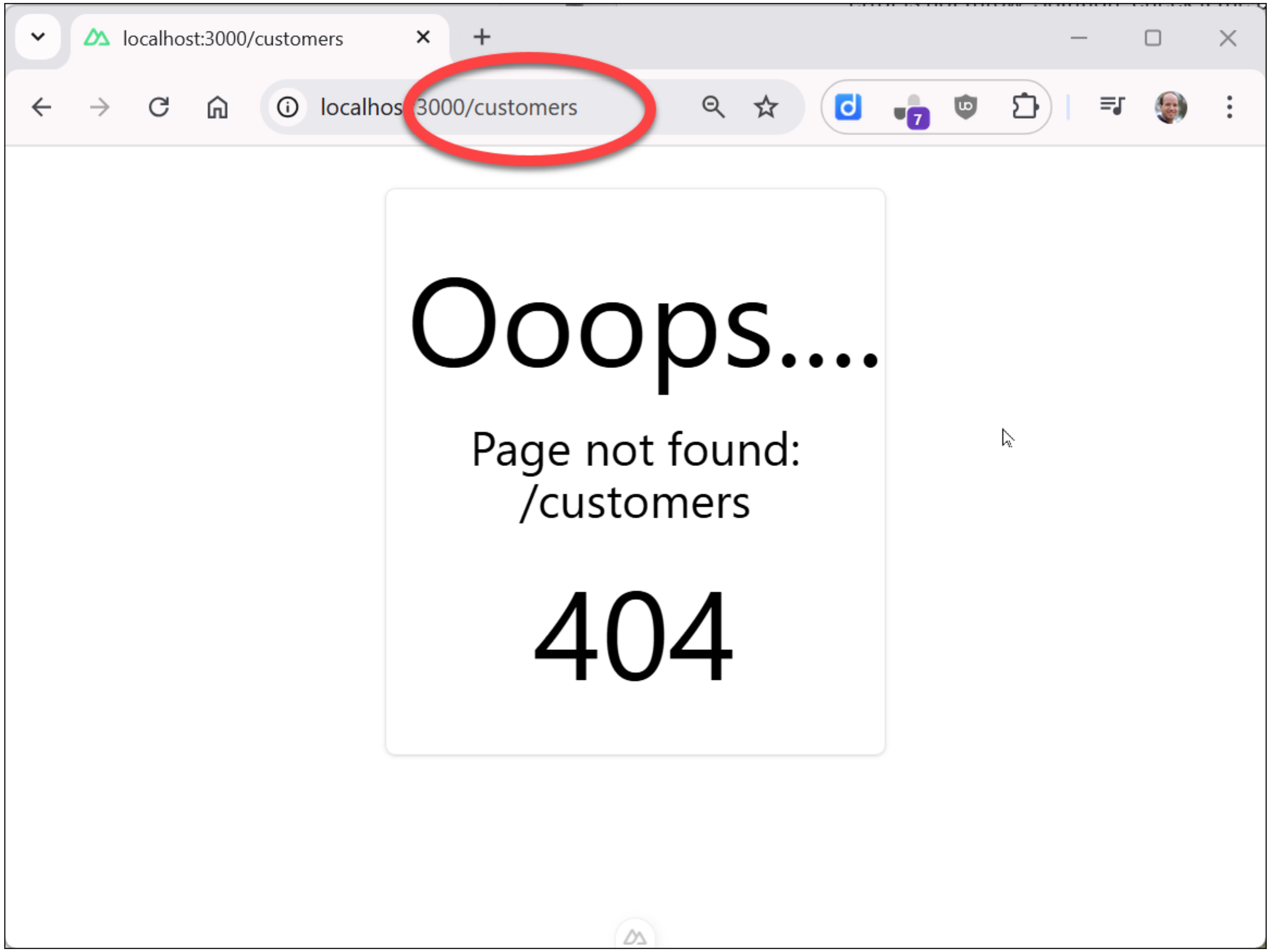
https://nuxt.com/docs/guide/directory-structure/components

Create error.vue

- Create an error.vue component *in the root of your project*
- Create an error variable using useError()
- Create template with your error message + classes
- No need to further config your app!

```
<!--error.vue-->
<script setup lang="ts">
  const error = useError();
</script>

<template>
  <div class="mt-8 p-4 max-w-sm mx-auto text-center …">
    <h2 class="mt-8 text-8xl">Ooops....</h2>
    <p class="mt-8 text-4xl">
      {{ error?.message }}
    </p>
    <p class="mt-8 mb-8 text-8xl">
      {{ error?.statusCode }}
    </p>
  </div>
</template>
```

Result


localhost:3000/customers

localhost:3000/customers

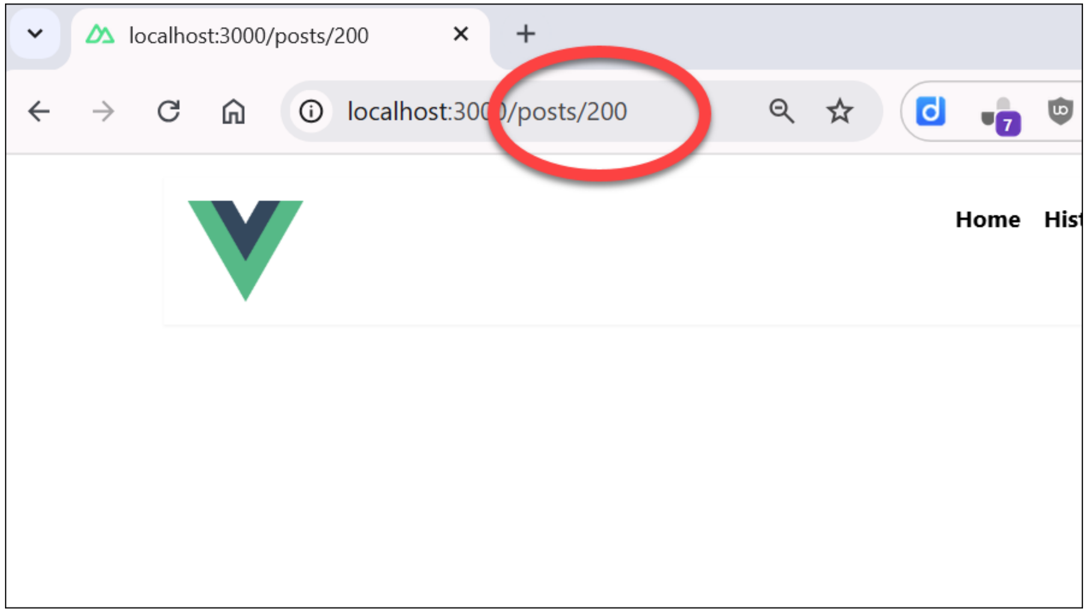
Ooops....

Page not found: /customers

404

Retrieving non-existing id's

- *Technically* all is OK. Because the page exists (!)
- So, the page is loaded
  - However, the `fetch()` fails because the `id` does not exist
  - We accounted for that by not showing empty data if `!post`.
  - We need to intercept the response and throw a custom error if something goes wrong



Update component logic

- We simplified component logic
- When `fetch()` fails, we throw a `createError()` with custom configuration.
- This is built-in in Nuxt and intercepted by the framework.
  - The `error.vue` page is shown instead.
- The error page is now used for not-found routes AND failing `fetch()` requests
- Lesson: use `createError()` for throwing custom errors

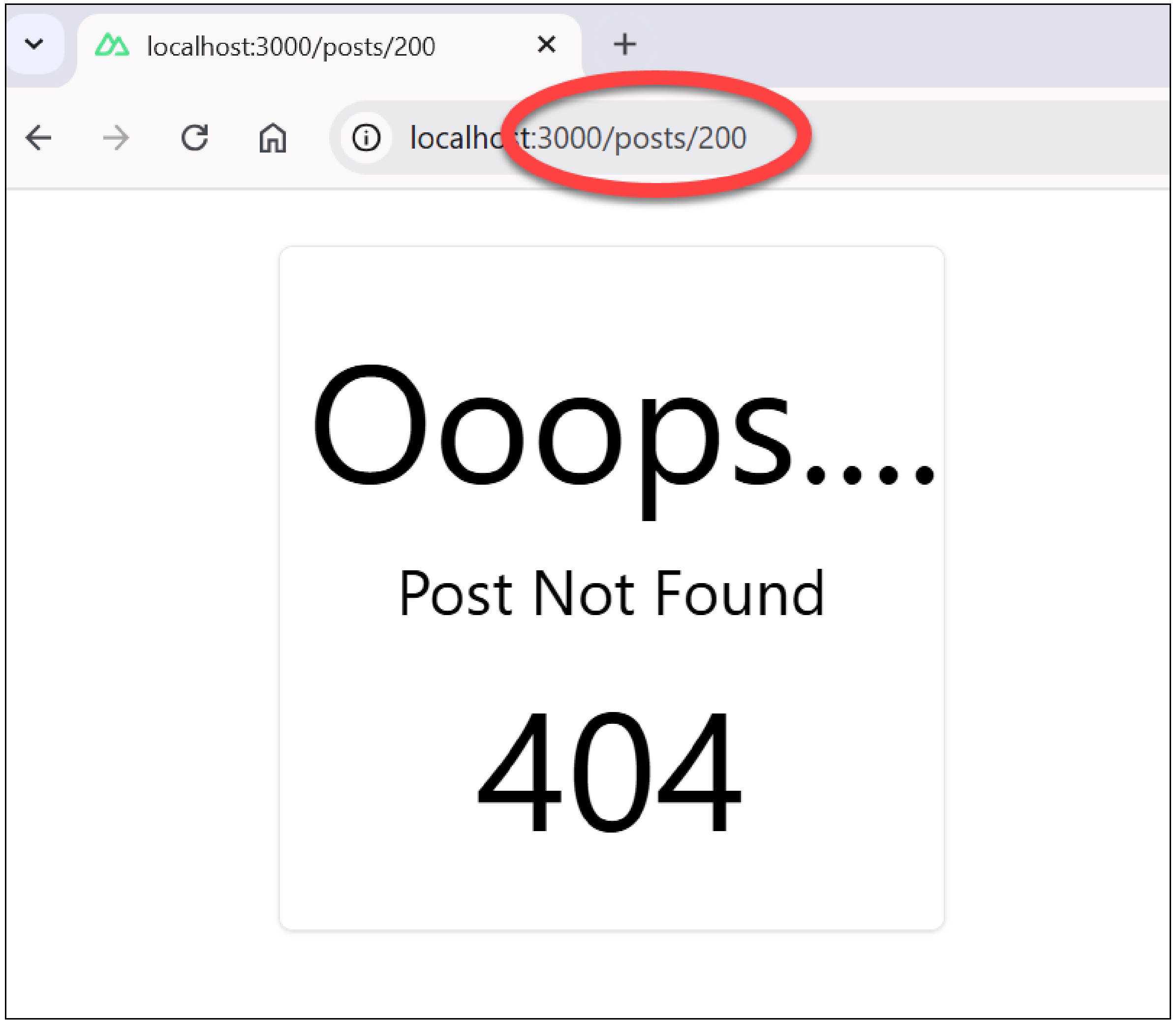
Simplified script...

```
<script setup lang="ts">
import type { Post } from "@/types/Post";

// Route params
const route = useRoute();
const id = route.params.id;

// URL to fetch data from
const detail = `https://jsonplaceholder.typicode.com/posts/${id}`;

// variable - doesn't have to be reactive let post:Post|null= null;

// Fetch post function
const fetchPost = async () => {
  const response = await fetch(detail);
  if (!response.ok) {
    throw createError({ statusCode: 404, statusMessage: "Post Not Found" });
  }
  post = await response.json();
};

// Call fetchPost in `async setup`
await fetchPost(); // This ensures Nuxt handles `createError()` correctly
</script>
```

Simplified template + result

```
<!--[id].vue-->
<template>
  <div v-if="post">
    <PostDetail :post="post" />
  </div>
</template>
```

You can also add a `loading` indicator, of course



Ooops....

Post Not Found

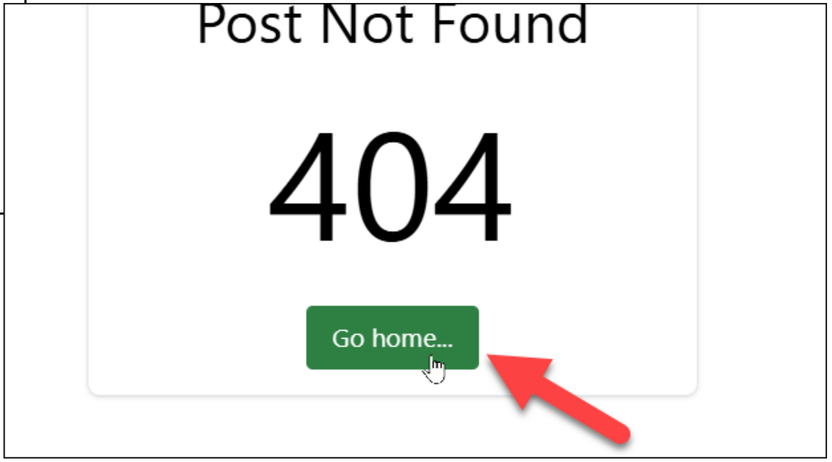
404

Clearing the error stack and return

- Update the `error.vue` component with a ‘Back’ button
- Inside the click handler for the button, call the built-in `clearError()` function and redirect to the home page

```
<button class="btn btn-green" @click="handleClear()">Go home...</button>
```

```
const handleClear = () => {
  clearError({
    redirect: '/'
  })
}
```



Tip: using `fatal`

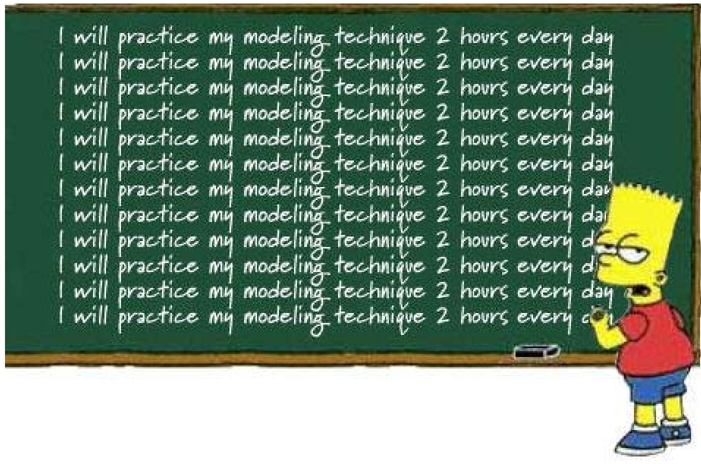
- Add `fatal` property to `createError()` to handle *all* errors, even if client-sided links are inadvertently incorrect.
- For instance:

```
throw createError({
  statusCode: 404,
  statusMessage: "Post Not Found",
  fatal: true
});
```

Workshop

- Create a custom Error page for your application
- Show the default status code and status message
- Define a custom message to be shown when an error happens
- Make sure the error page is shown when a non-existing route is requested and when fetching data fails

- Example: `../examples/160-custom-error-page`



Checkpoint

- You know how to create a custom error page
- You can show this error page on non-existing routes and when fetching data fails
- You are aware of the `createError()` and `clearError()` functions
- You know the behavior of `fatal`.