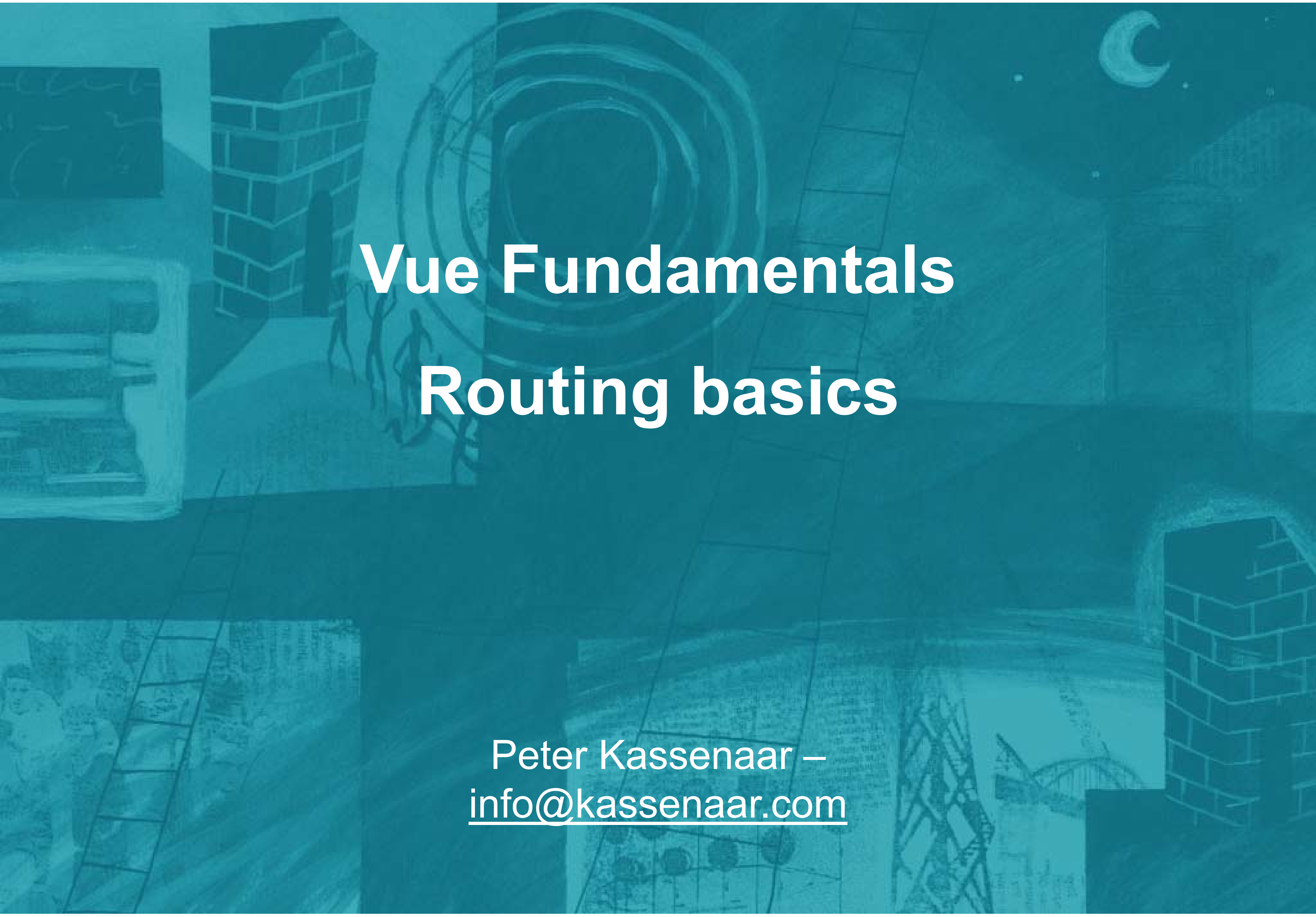
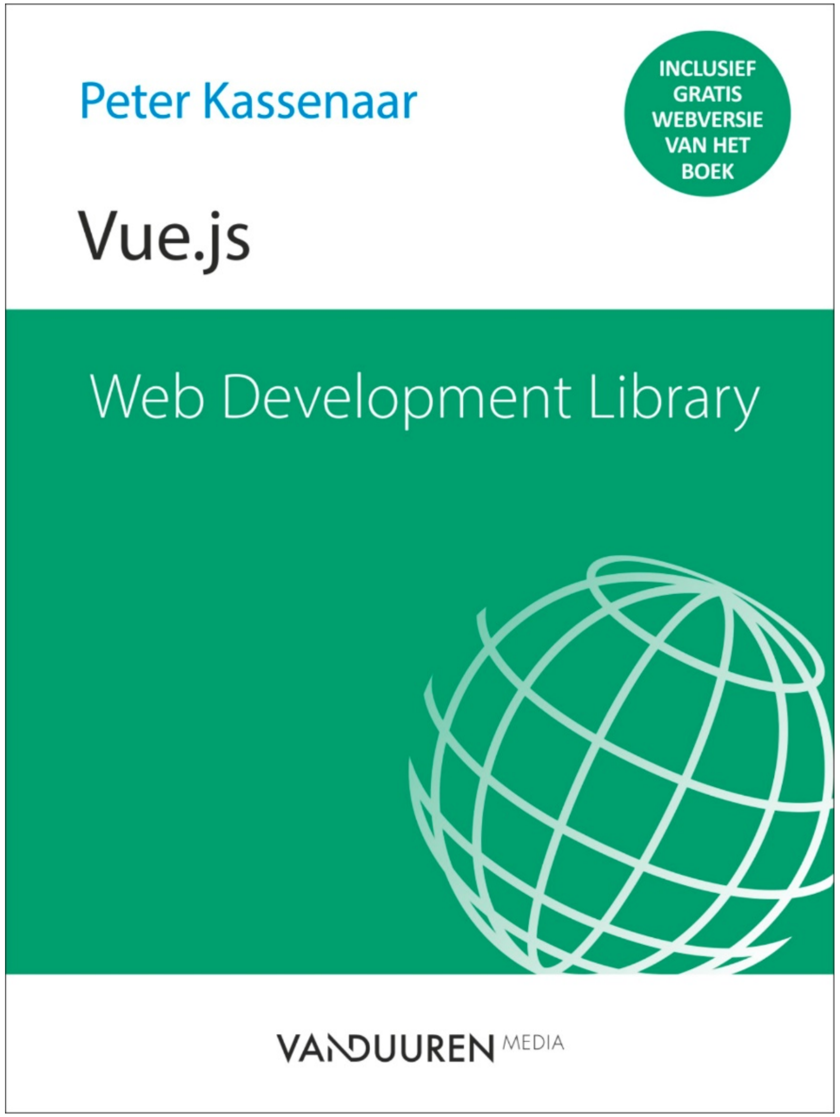


Vue Fundamentals

Routing basics

Peter Kassenaar –
info@kassenaar.com



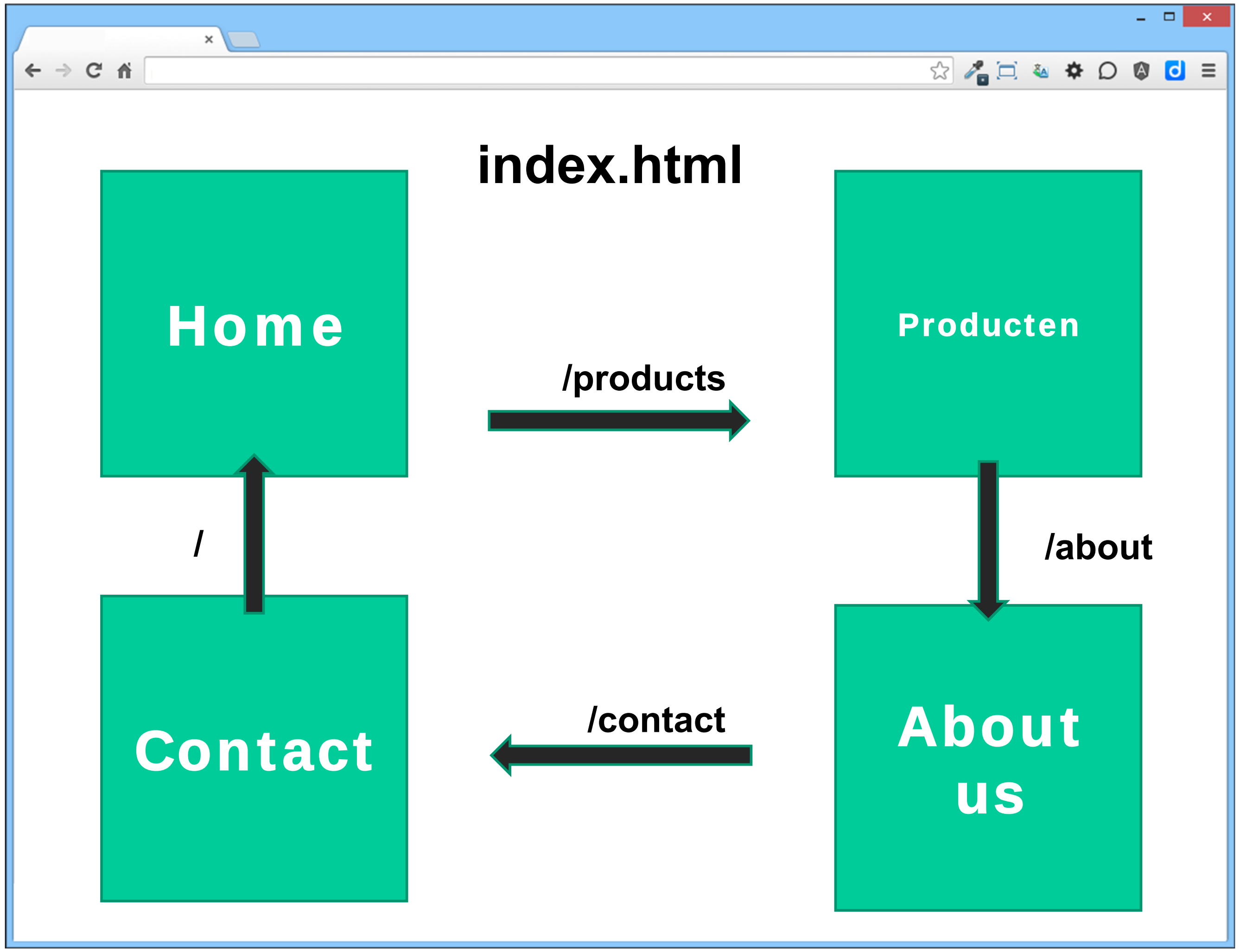
Chapter 6, P. 164

Client-side Routing

“For most Single Page Applications, it’s recommended to use the officially-supported vue-router library. For more details, see vue-router’s documentation.` ”

[*https://router.vuejs.org/*](https://router.vuejs.org/)

Routing architecture and goal



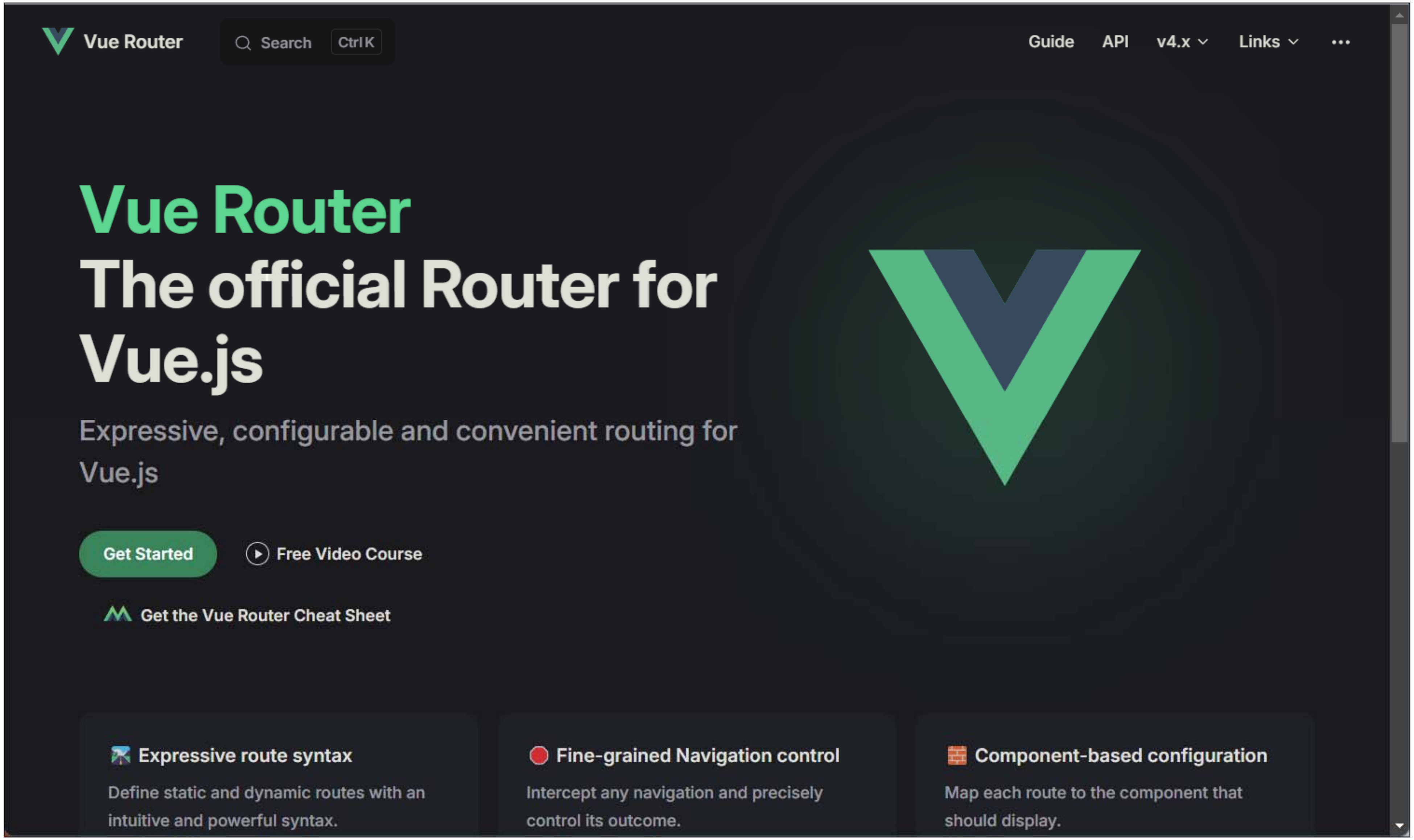
- Make use of SPA principle
- Making deep links possible

Router capabilities / characteristics

- Update URL when changing to routes/components
- Nested routes/view mapping
- Lazy Loading
- Modular, component based router configuration
- Route params, query, wildcards
- View transition effects
- Link with automatic active CSS classes to denote active route
- HTML5 history mode
- Prevent navigating to a route, or navigating away

https://router.vuejs.org/

https://router.vuejs.org/



Vue Router
The official Router for Vue.js

Expressive, configurable and convenient routing for Vue.js

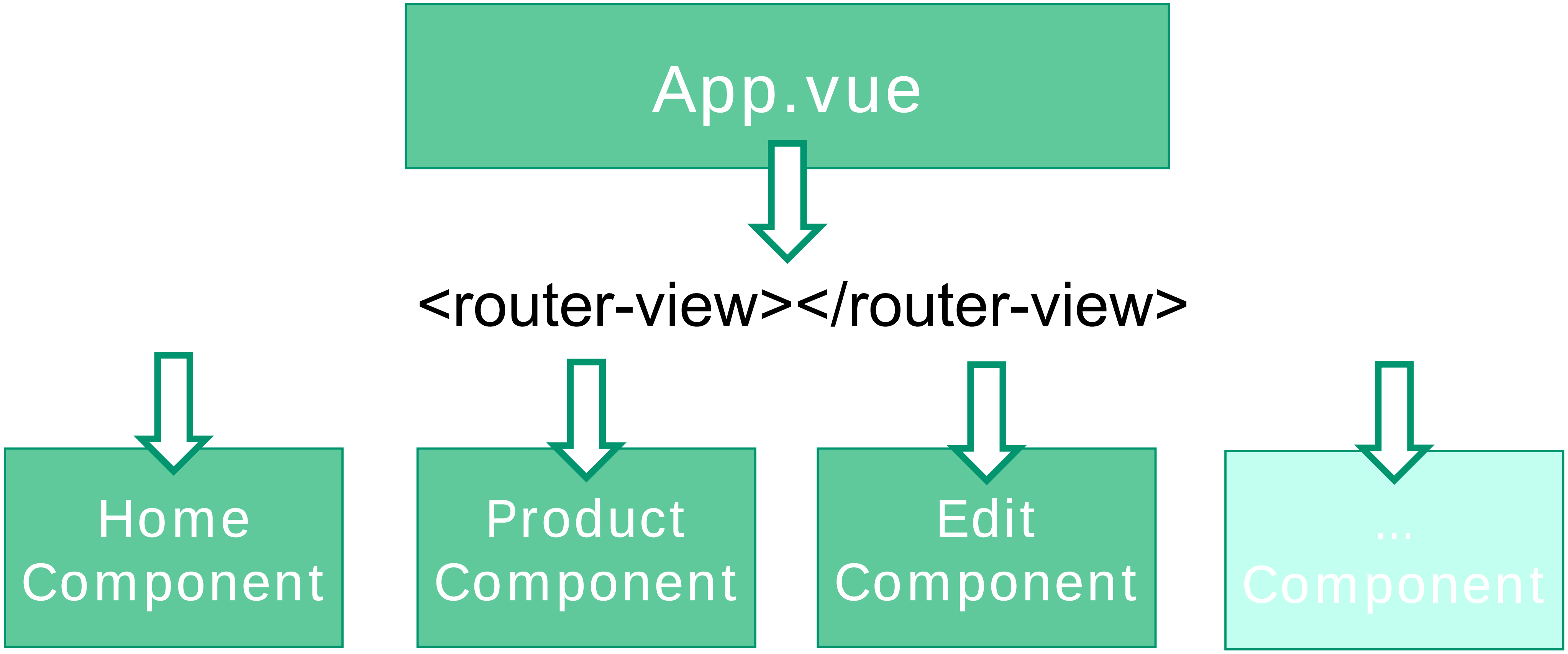
Expressive route syntax
Define static and dynamic routes with an intuitive and powerful syntax.

Fine-grained Navigation control
Intercept any navigation and precisely control its outcome.

Component-based configuration
Map each route to the component that should display.

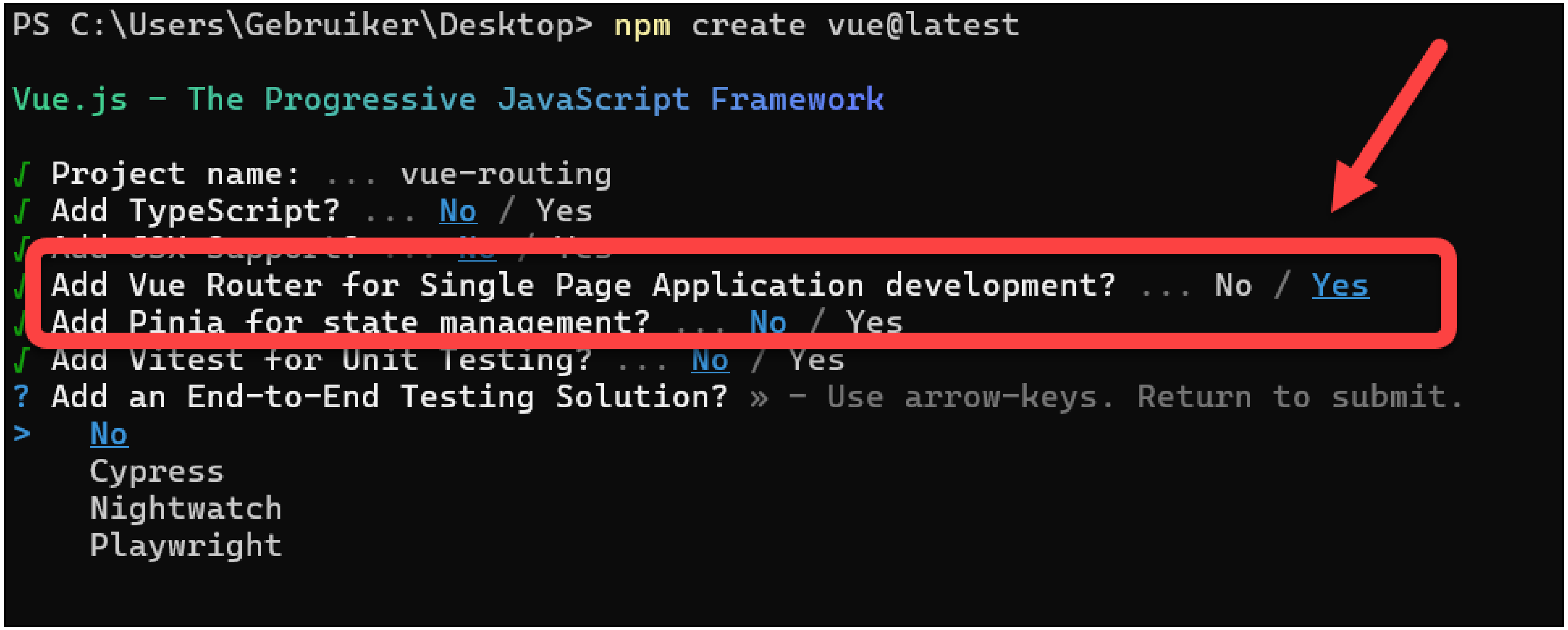
Routing – every route is a Component

- `App.vue` gets a main menu (typically in its own component :-)
- Components are injected in `<router-view></router-view>`



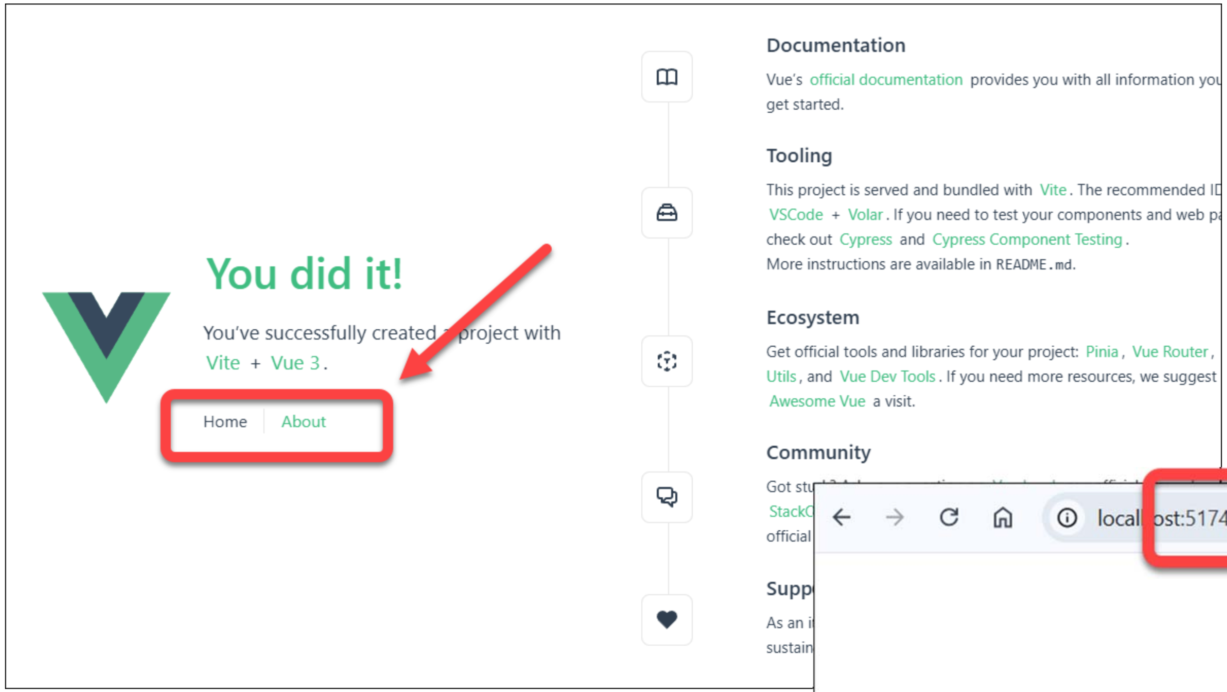
Sidestep - routing via Vue CLI

- For instance: `npm create vue@latest`
- Select Yes with `Vue Router` (you can have multiple selections!)
- You get a fully featured, simple, but working example



```
PS C:\Users\Gebruiker\Desktop> npm create vue@latest

Vue.js - The Progressive JavaScript Framework

√ Project name: ... vue-routing
√ Add TypeScript? ... No / Yes
√ Add JSX Support? ... No / Yes
√ Add Vue Router for Single Page Application development? ... No / Yes
√ Add Pinia for state management? ... No / Yes
√ Add Vitest for Unit Testing? ... No / Yes
? Add an End-to-End Testing Solution? » - Use arrow-keys. Return to submit.
>   No
    Cypress
    Nightwatch
    Playwright
```

Result – a default routing app



You did it!

You've successfully created a project with Vite + Vue 3.

Home | About

Documentation

Vue's official documentation provides you with all information you
get started.

Tooling

This project is served and bundled with Vite. The recommended ID
VSCode + Volar. If you need to test your components and web pa
check out Cypress and Cypress Component Testing.
More instructions are available in `README.md`.

Ecosystem

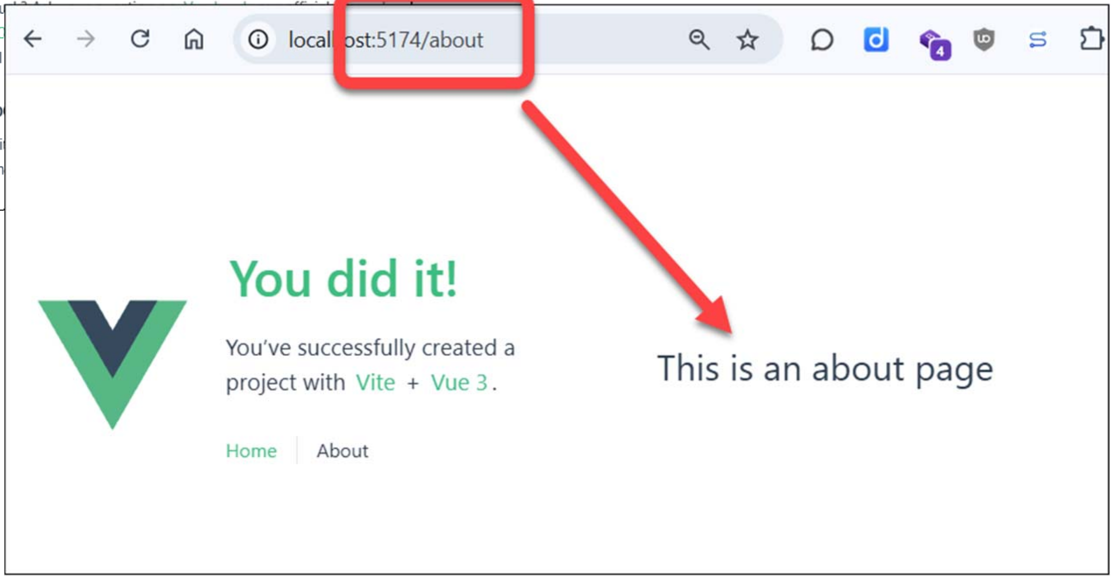
Get official tools and libraries for your project: Pinia, Vue Router,
Utils, and Vue Dev Tools. If you need more resources, we suggest
Awesome Vue a visit.

Community

Got stu
StackO
official

Supp

As an i
sustain



localhost:5174/about

You did it!

You've successfully created a project with Vite + Vue 3.

Home | About

This is an about page

This is what we are going to create manually

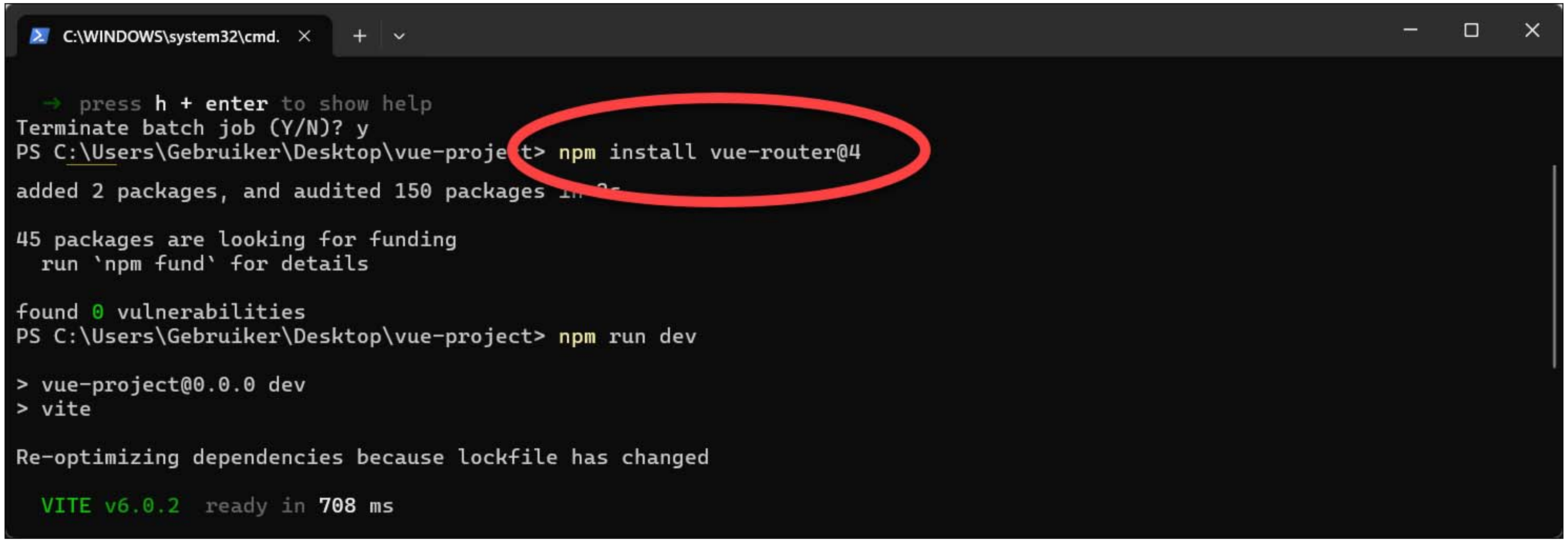
Agenda - In this module

- Adding routing to an existing app
    - See before – start a new app with routing from scratch, using the CLI
- Linking to routed pages
- Styling links, based on the active route
- Working with route params
- Nested router views
- More info on routing

https://router.vuejs.org/guide/

1. Installing the router to an existing app

- Install the router module: `npm install vue-router@4`
  - Yes, Vue 3 uses router V4
  - (Vue 2 used router V3 – we can't help the version numbering...)
- Check if the application still runs!
- Nothing is visible, but is should still run



C:\WINDOWS\system32\cmd.

```
  → press h + enter to show help
Terminate batch job (Y/N)? y
PS C:\Users\Gebruiker\Desktop\vue-proje t> npm install vue-router@4

added 2 packages, and audited 150 packages in

45 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Gebruiker\Desktop\vue-project> npm run dev

> vue-project@0.0.0 dev
> vite

Re-optimizing dependencies because lockfile has changed

  VITE v6.0.2  ready in 708 ms
```

2. Update App.vue

- We created a new navigation component, `<MainNavigation />`
  - If you want to, you can add navigation directly to `App.vue`
- Notice the usage of `<RouterLink />` and `<RouterView />`

```
<!-- MainNavigation.vue -->
<li class="nav-item">
  <RouterLink to="/" class="nav-link" >Home</RouterLink>
</li>
<li class="nav-item">
  <RouterLink to="add" class="nav-link">Add Country</RouterLink>
</li>
<li class="nav-item">
  <RouterLink to="update" class="nav-l
</li>
```

```
<!-- App.vue -->
<template>
  <div id="app" class="container">
    <MainNavigation/>
    <RouterView />
  </div>
</template>
```

3. Create routes folder

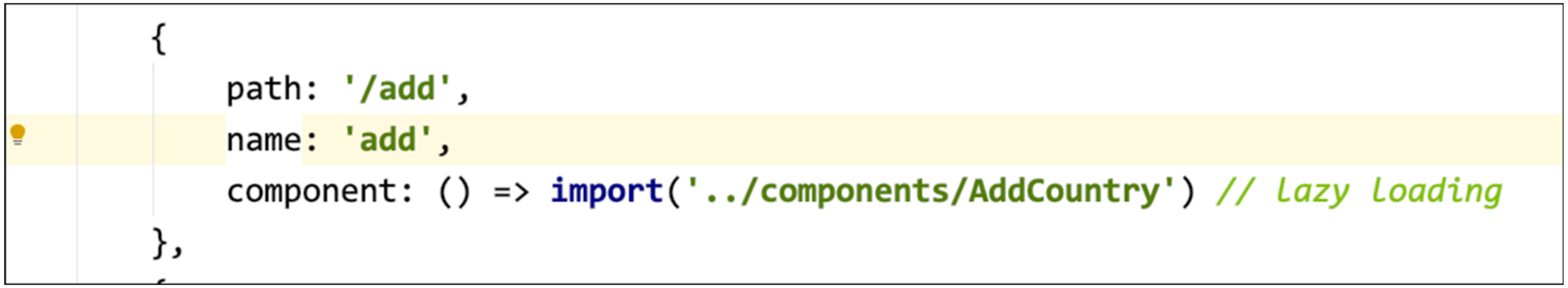
- Create a routes folder, holding all router configuration
  - Not required, just for good code organization
  - Otherwise, define routes directly in router instance (see next)
- Create an `index.js` inside `../router`
- Basic lay-out for routing table:

```
// router/index.js

// import the required components
import VacationPicker from "@/components/VacationPicker.vue";
…
export const routes =[
    // define all routes here....
    {
        path: '/',
        name: 'home',
        component: VacationPicker
    },
    …
]
```

Tell Vue which component to load when a specific route is requested

2a. Optional: lazy loading

- You can *lazy load* components, i.e. only load them once the user navigates to them
- Use the `import` statement for that
- Don't forget to remove the component from the list with `import` statements at the top of the file! (if applicable)
  - Otherwise it would still *always* be loaded, and not lazy loaded.



```
{
    path: '/add',
    name: 'add',
    component: () => import('../components/AddCountry') // Lazy Loading
},
```

Alternative w/ lazy loading

- Assign the lazy loaded component to a variable and use that variable in the routing table:

```
// alternative for lazy loading
const AddCountry = () => import('../components/AddCountry.vue')

export const routes = [
    // define all routes here....
    …,
    {
        path: '/add',
        name: 'add',
        component: AddCountry, // lazy loaded component
    },
```

3. Create the router instance in `main.js`

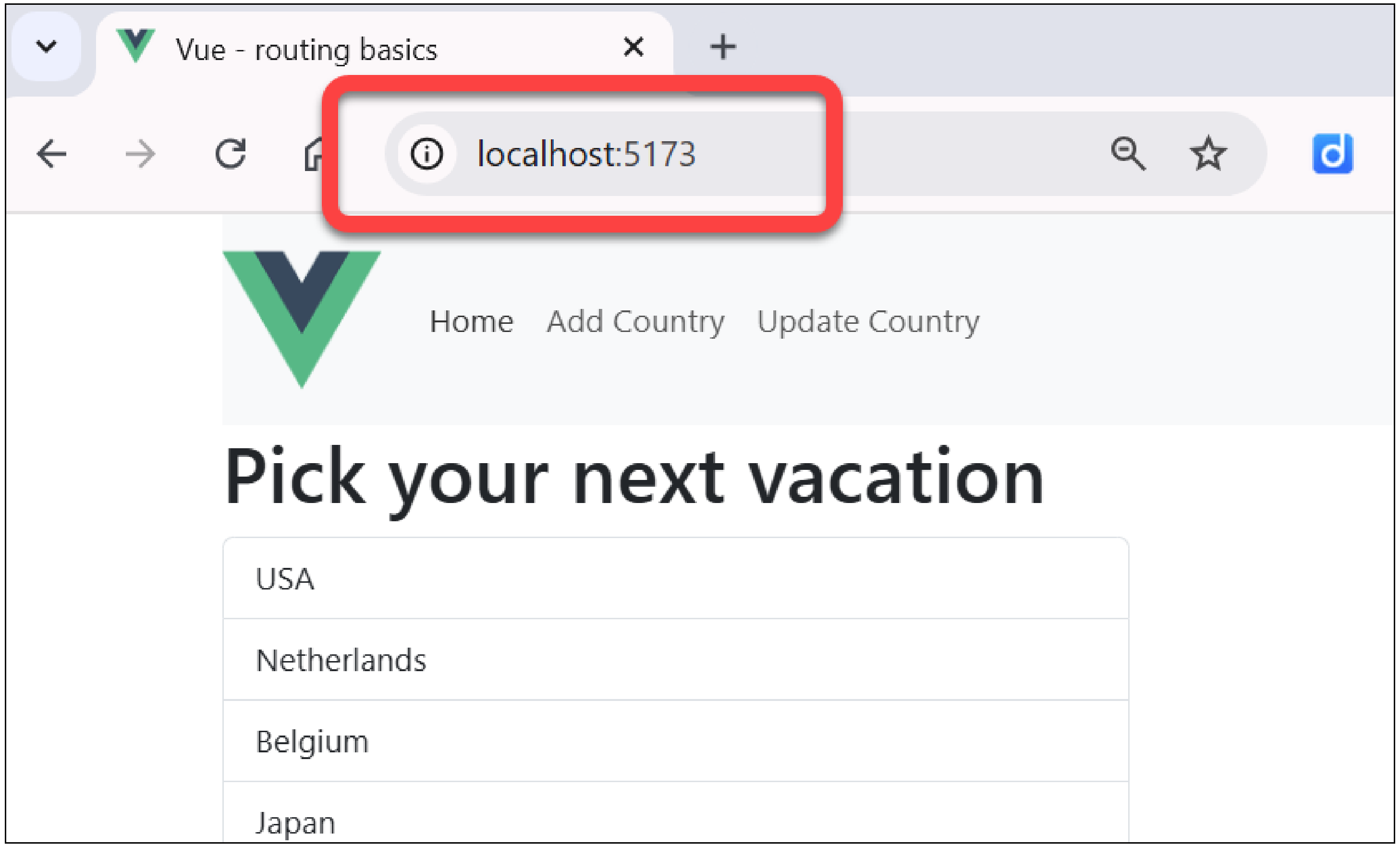
```
// main.js
// 1. importing the router stuff
import {createRouter, createWebHistory} from "vue-router";
import {routes} from "@/routes/index.js";

// 2. creating the router
const router = createRouter({
    history: createWebHistory(),
    routes
})

// 3. mounting the router to App
createApp(App)
    .use(router) // 'use' before 'mount'
    .mount('#app')
```

4. Tell Vue where to project routed content

- Use `<router-view />` to tell Vue where to project the components
- You don't need the import for `VacationPicker` in `App.vue` anymore
  - We *do* need `MainNavigation`, b/c we placed it inside its own component

```
<!--App.vue-->
<script setup>
import MainNavigation from "@/components/MainNavigation.vue";
</script>

<template>
  <div id="app" class="container">
    <MainNavigation/>
    <RouterView/>
  </div>
</template>
```

Result so far:



Pick your next vacation

USA

Netherlands

Belgium

Japan

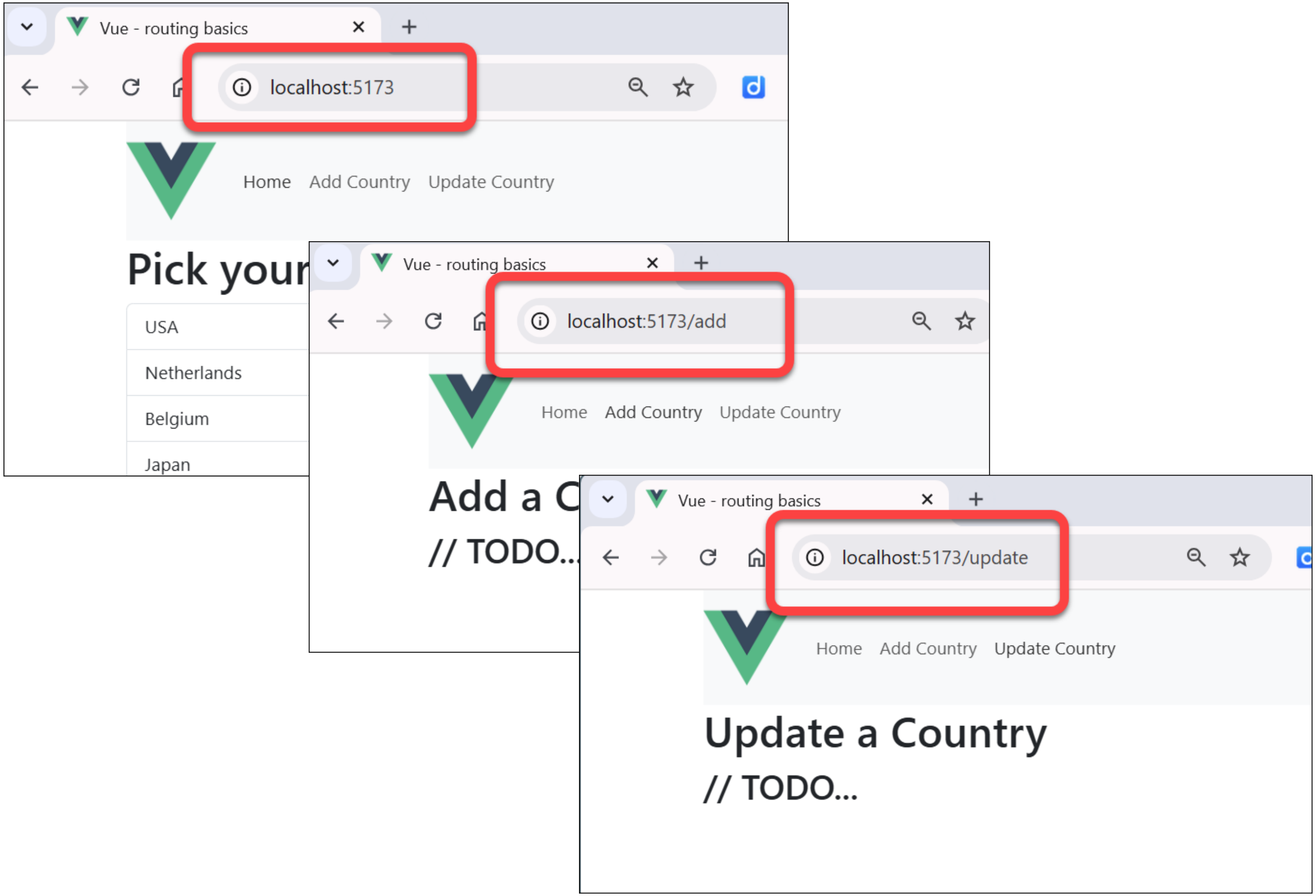
Everything should work now. Check this!

6. Recap: linking to routed pages

- We can route to pages by typing the route in the address bar.
  - But, nobody wants to do this!
  - We want to achieve this by clicking links!
- No more `<a href="…">` -tags!
- Use Vue-specific `<router-link to="name-of-route">` tags

```
…
<li class="nav-item">
    <router-link to="/" class="nav-link">Home</router-link>
</li>
<li class="nav-item">
    <router-link to="add" class="nav-link">Add Country</router-link>
</li>
…
```

Result



Pick your

USA

Netherlands

Belgium

Japan

Add a C

// TODO...

Update a Country

// TODO...

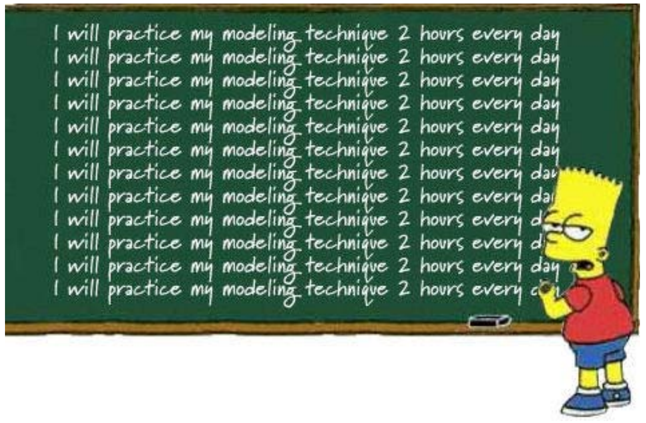
Alternative syntax – navigating by binding

- We also defined a name property for our routes
- Use the name to navigate by binding to `:to`
- By clicking the logo, we navigate to the home page (i.e. `VacationPicker` component)
- This is completely optional, but you see how you can also link to routes and bind via code.
- You can create dynamic routes this way, because `name` is a variable on your component

```
<router-link class="navbar-brand" :to="{name: 'home'}">
    <img src="../assets/logo.png" alt="Vue Logo">
</router-link>
```

Workshop

1. Add routing to your own application, using the steps in this module. OR:
2. Start a new project from scratch, using the CLI
   - Add routing via the prompt
   - Add a few components, route to them
   - Don’t forget to link to the new routes/components, OR:
3. Generic example: `../300-routing-basics`
   - Add a new component
   - Add a route to that component
   - Update the `<MainNavigation />`
   - Optional: use lazy loading
   - Make sure it works


I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day

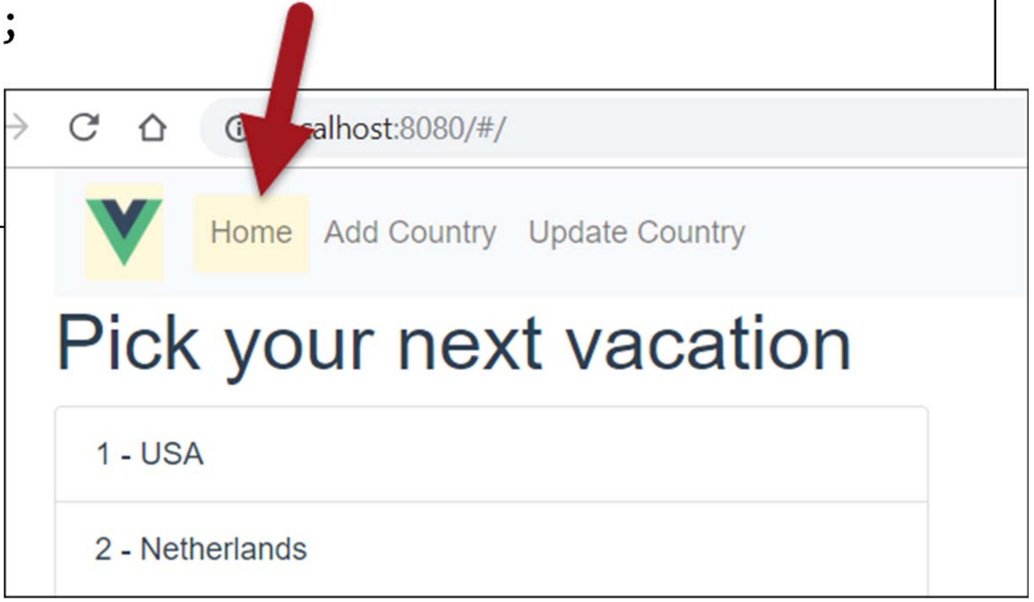
Styling the active link

Emphasizing the current route in the UI, based on a URL match

Special class names

- Vue assigns a `router-link-active` class to the current active route.
- You can write a CSS-class for that
- No need to update the HTML/UI. Vue assigns it automatically.
- Of course you can add any property you want/need

```
<style scoped>
    /*Automatic assignment of this class by Vue, based on router state*/
    .router-link-active{
        background-color: cornsilk;
        color: black;
    }
</style>
```



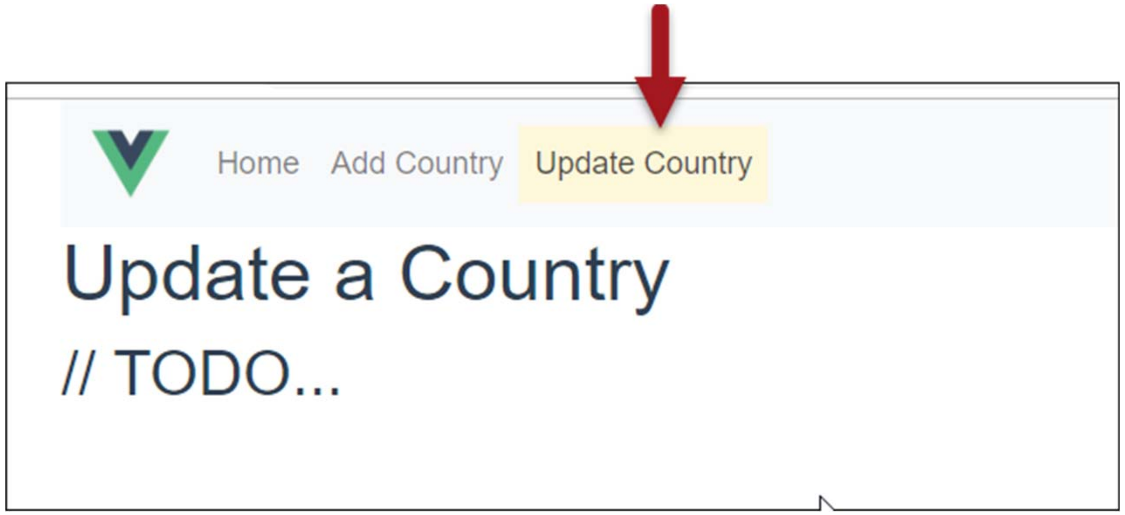
Pick your next vacation

1 - USA

2 - Netherlands

Also: use custom CSS with exact

```
<style scoped>
  .router-link-exact-active {
     /* …your styling … */
  }
</style>
```



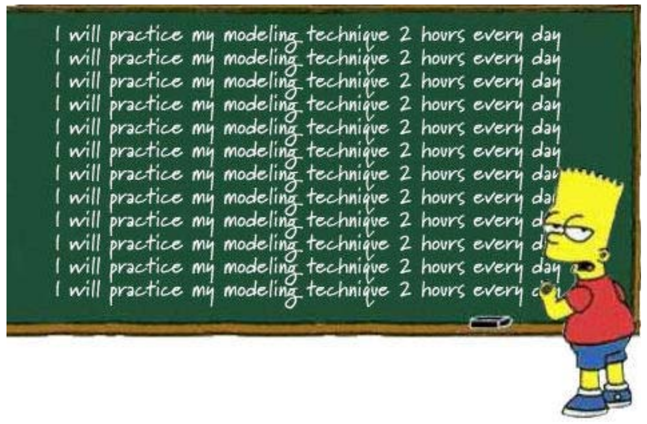
Update a Country

// TODO...

(Now only the logo still has a yellow background, b/c this links to the homepage, but IRL you will overcome this.)

Workshop

- Add a style to your active links, as described in this section.
  - Create a style for your active links
  - Add the style to the Navigation component
  - Use the `exact` keyword to highlight only the correct style
- Use:
  - 1. The `exact` keyword on your navigation links, OR
  - 2. The `router-link-exact-active` class in your CSS



Navigating from code

Using the router from JavaScript, instead of HTML

Navigating from code

- We're going to navigate to the `<CountryDetail />` component via code
- We'll use this page later on for route params
- First – create a route for it in `../routes/index.js`.

```
export default new Router({
   routes: [
      // define all routes here....
      …
      {
         path: '/detail',
         name : 'detail',
         component: CountryDetail
      }
   ]
})
```

Navigate from code

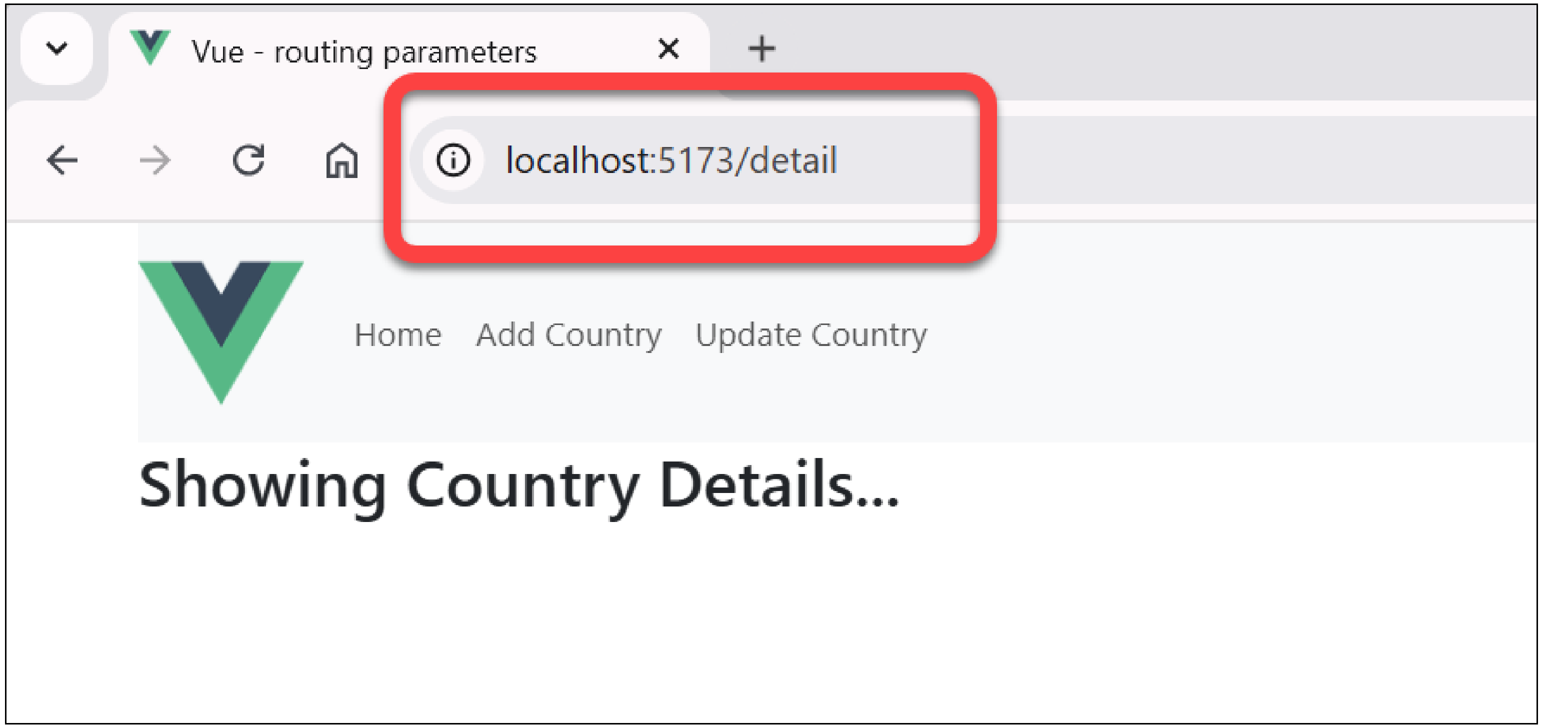
- Next, define a method that uses the router in the VacationPicker component
- With the Composition API, import useRouter instance and use its methods and properties

```
<li class="list-group-item"
    @click="showCountry(country)"
    v-for="country in data.countries" :key="country.id">
     …
</li>
```

```
// Routing from code
import {useRouter} from "vue-router";
const router = useRouter()

const showCountry = (country) => {
  console.log('navigate to: ', country.name);
  router.push({ path: '/detail' }); // you can use 'name' if desired
}
```

Result

- Hardcoded text for now – we're going to solve that by using *route parameters*.



Showing Country Details...

Using Route Parameters

Passing detail parameters to the next view

Peter Kassenaar

INCLUSIEF GRATIS WEBVERSIE VAN HET BOEK

Vue.js

Web Development Library

VANDUUREN MEDIA

P.189 e.v.

So far, we're just displaying hardcoded text on the Detail page.

The goal of course, is to show data from the passed in country.

For that we use *Route Parameters*

1. Update the route

`../router/index.js` → add `/:id` and possible other parameters to the detail route

```
{
   path: '/detail/:id/:name',
   name : 'detail',
   component: CountryDetail
}
```

2. Update the `<CountryDetail />` component, to grab the parameters from the route. We can do this with the `useRouter()` composable

```
// using the current URL with useRoute().
const route = useRoute();

// convert string from URL to number
const id = parseInt(route.params.id, 10);
const name = route.params.name;
```

3. Update routerlink, to send the parameters

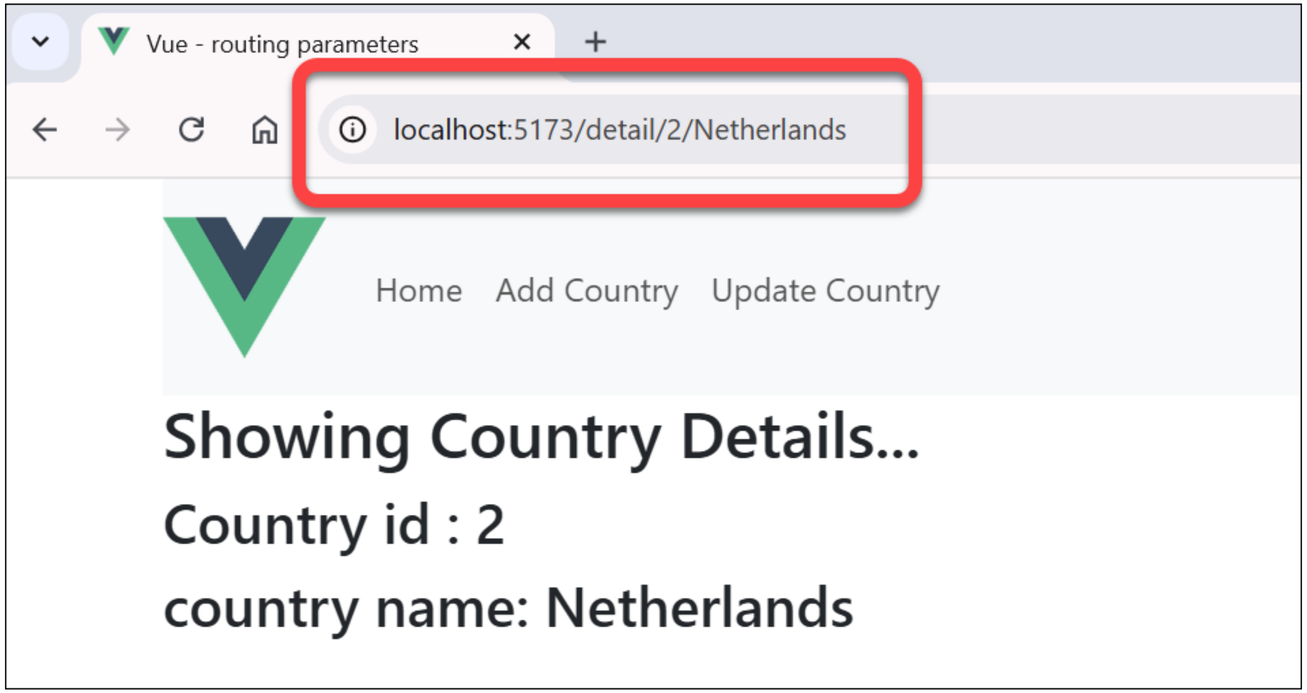
- Update the `<VacationPicker />` component to send the correct id and name in the URL
- Vue needs an object notation for that.
  - We can't just send a manually composed string like '`detail/1/Netherlands`'.

```
router.push({
  name: 'detail',
  params: {
    id: country.id,
    name: country.name,
  }
}); // When using 'params', you MUST use 'name'
```

4. Show the parameter in the UI

- Start simple – just show the passed paramater in the UI.

```
<h2>Showing Country Details...</h2>
<h3>Country id : {{ id }}</h3>
<h3>country name: {{ name }}</h3>
```



Showing Country Details...

Country id : 2

country name: Netherlands

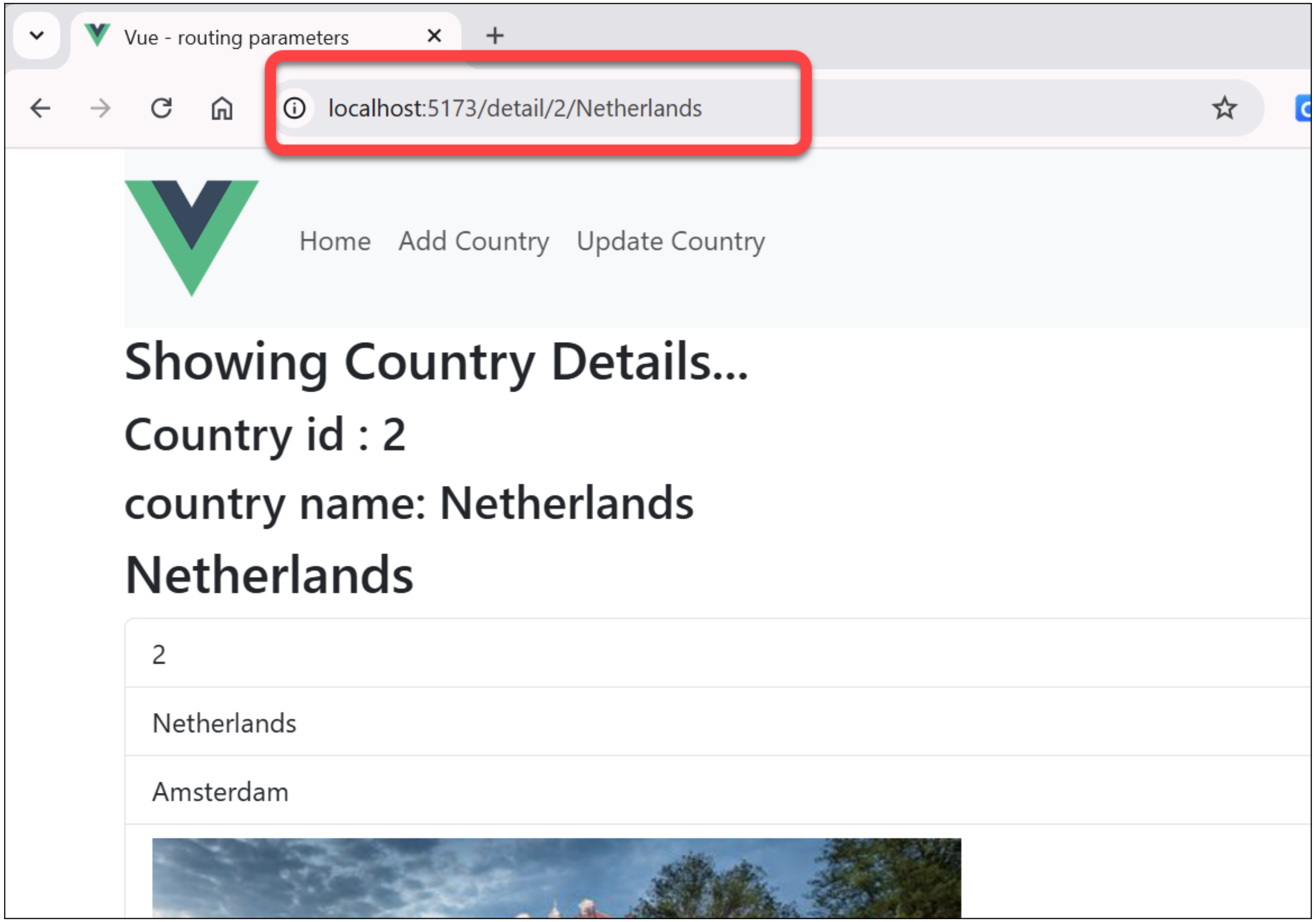
Show country data

- Grab the data (for now: local, to get the correct country)
  - We're going to fetch data from a real API later on!
- Look for the data based on the id

```
import countryData from "@/data/CountryData.js";
```

```
// get correct country, based on route parameter
const country = countryData.countries.find((country) => country.id === id);
```

Result



Showing Country Details...

Country id : 2

country name: Netherlands

Netherlands

2

Netherlands

Amsterdam

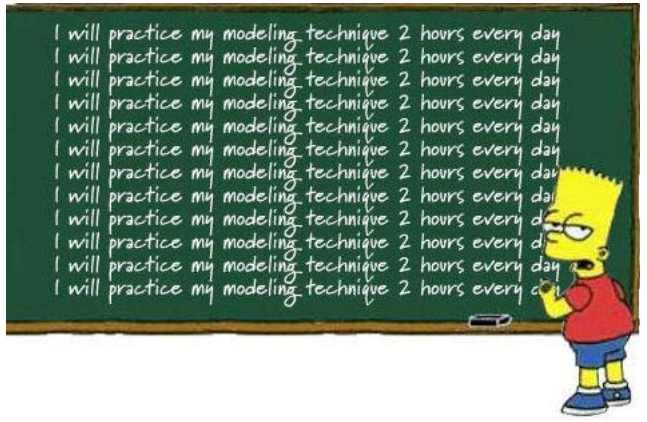
Remember:

*`useRouter()` gives access to the (application wide) **router instance***

*`useRoute()` gives access to the **currently selected route** inside a component!*

Workshop

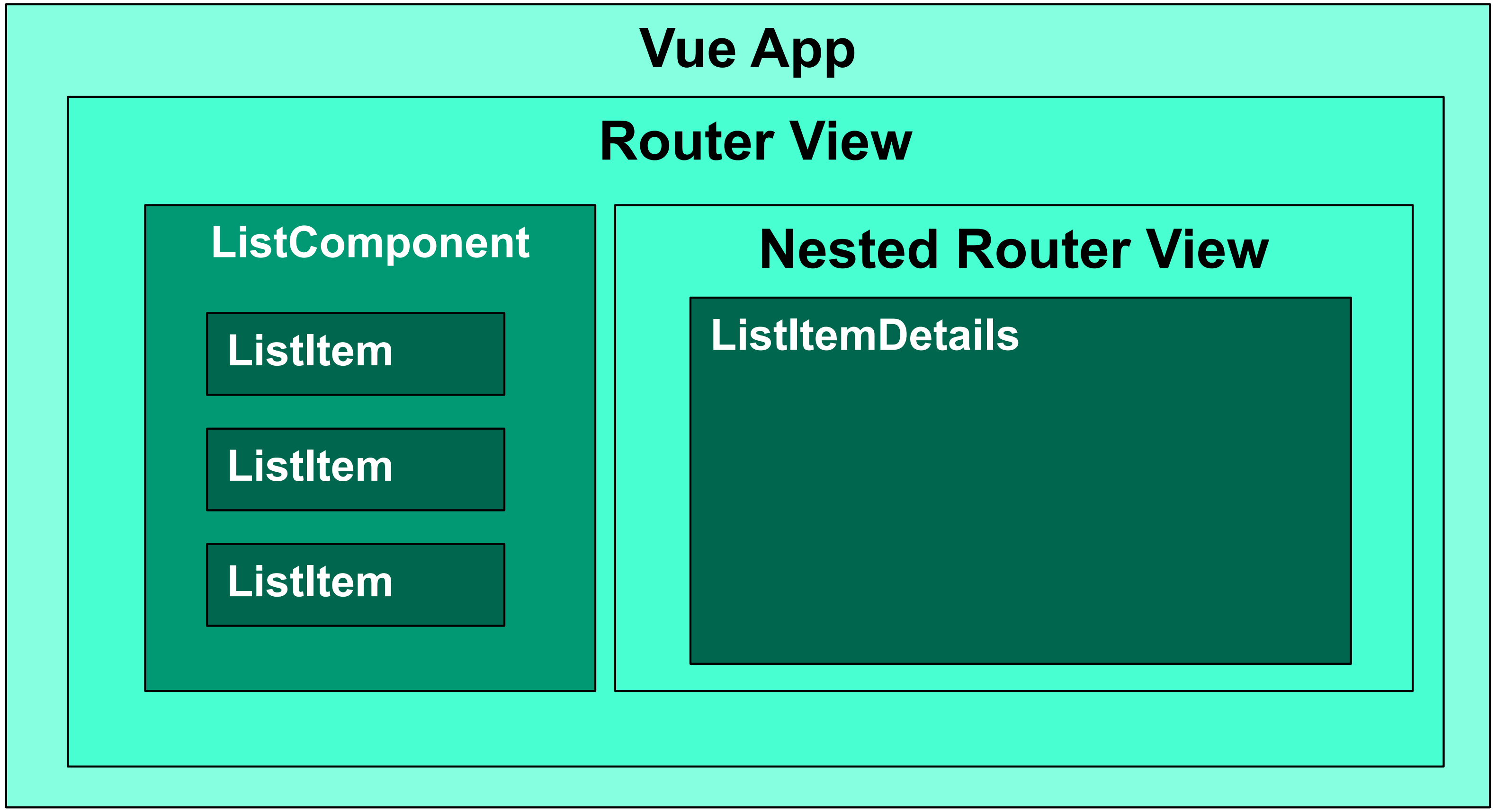
- Add routing parameters to your own application, using the steps in this module
  - Create a parametrized route, using `/:<name>` notation
  - Use `useRoute()` and `route.params.*` in the detail component to fetch the route parameters
  - Push a parametrized object to the `router` when element is clicked
- Generic example: `../310-routing-parameters`



Nested Router Views

Enabling Children: views inside views

What are nested router views?



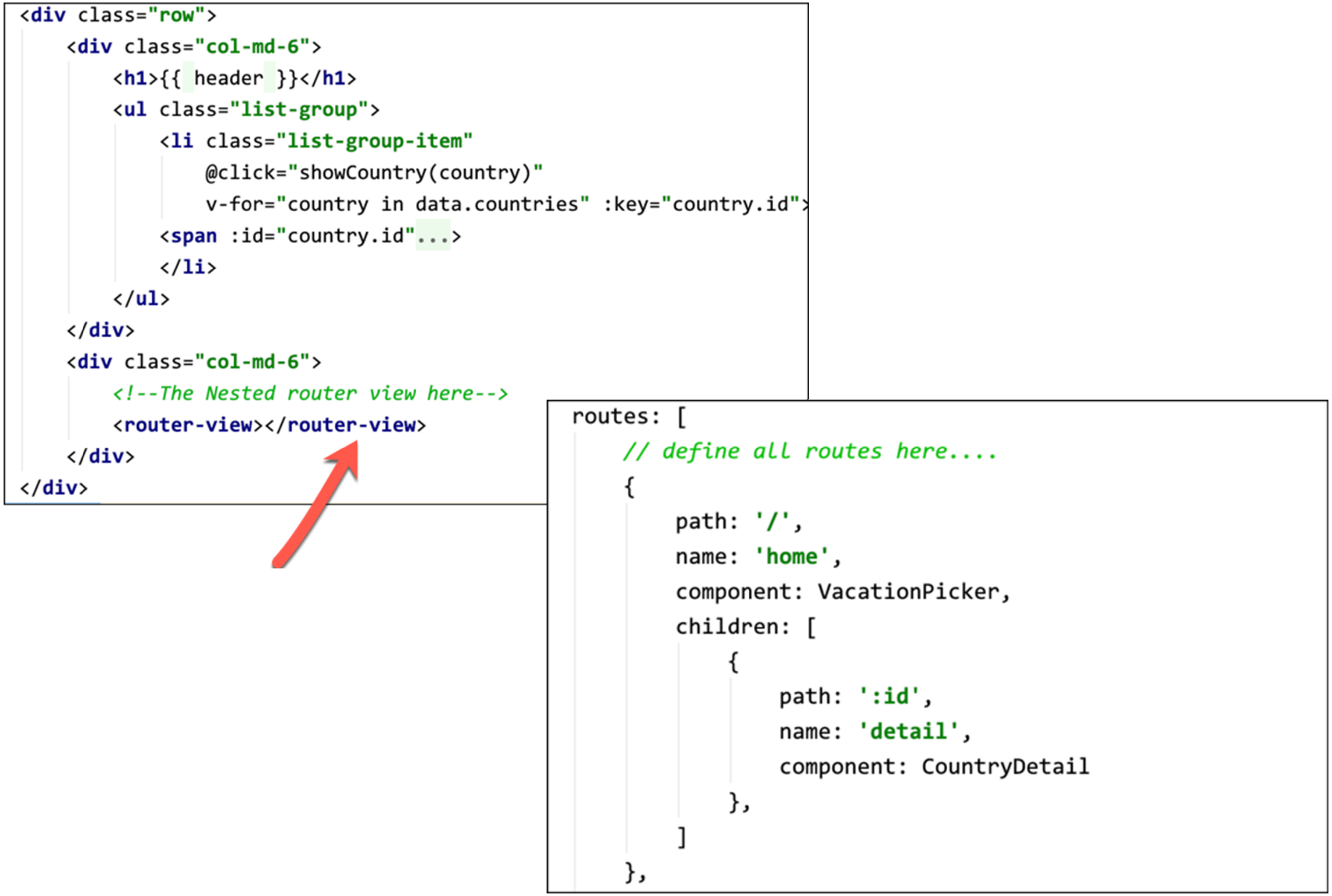
Show router links *inside* another router view, but still update the URL

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

Adding a nested router

- Add a new `<router-view>` to the component you want the nested routes to appear
  - In our case: `<VacationPicker />`
- Add an array of childroutes to the main routing table

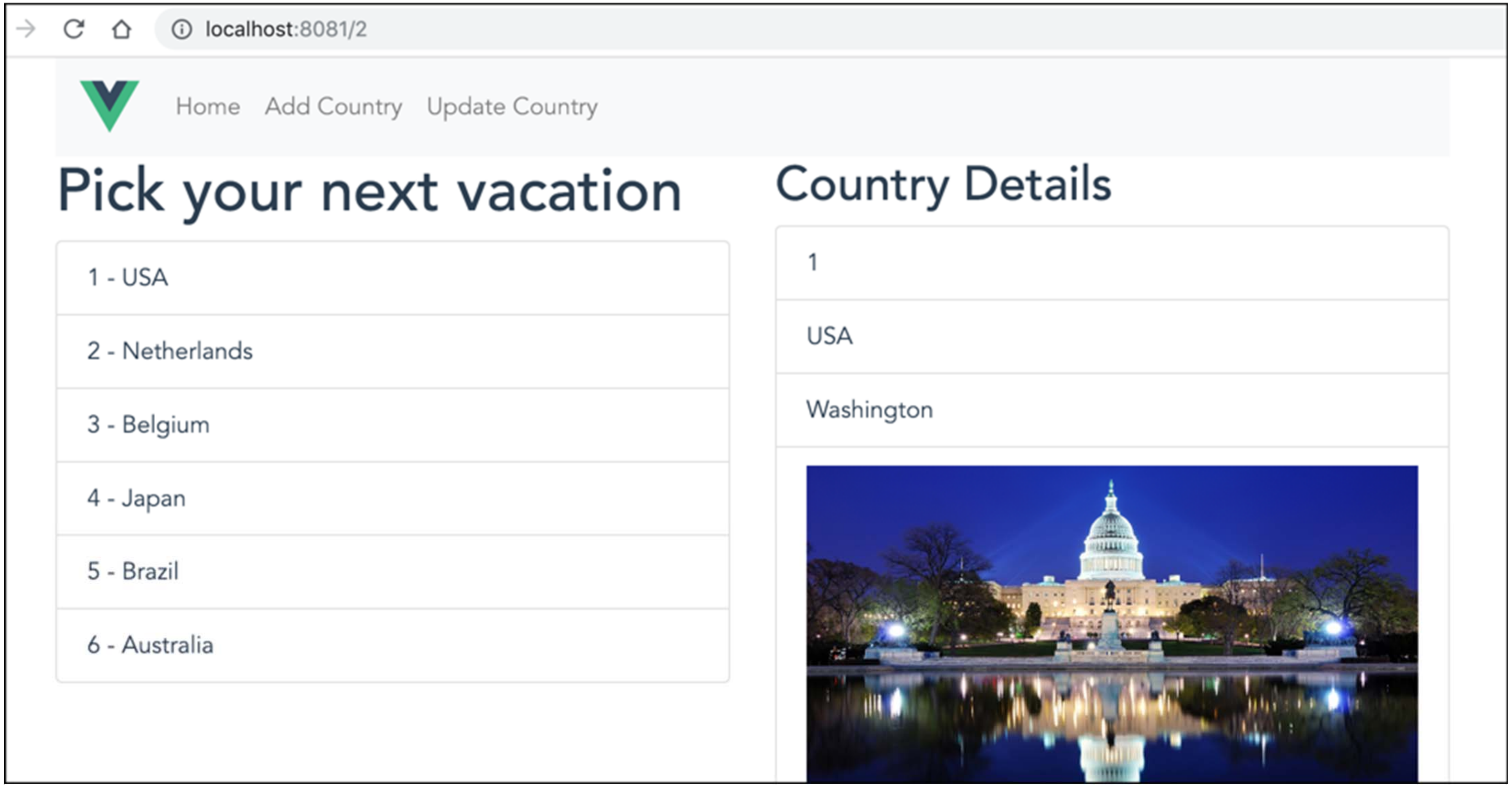
  -
    ```
    children: [
        { <child-router-path-1> }
        { <child-router-path-2> }
    ]
    ```
- Don’t forget to remove other dynamic routes (if you have some, of course)



```
<div class="row">
    <div class="col-md-6">
        <h1>{{ header }}</h1>
        <ul class="list-group">
            <li class="list-group-item"
                @click="showCountry(country)"
                v-for="country in data.countries" :key="country.id"
            <span :id="country.id"...>
            </li>
        </ul>
    </div>
    <div class="col-md-6">
        <!--The Nested router view here-->
        <router-view></router-view>
    </div>
</div>
```

```
routes: [
    // define all routes here....
    {
        path: '/',
        name: 'home',
        component: VacationPicker,
        children: [
            {
                path: ':id',
                name: 'detail',
                component: CountryDetail
            },
        ]
    },
```

Now it works…the first time



Pick your next vacation

1 - USA

2 - Netherlands

3 - Belgium

4 - Japan

5 - Brazil

6 - Australia

Country Details

1

USA

Washington

Updating the nested view

- A nested view is (by default) *not* recreated once the route updates
  - Better for performance
  - You need to tell the component to watch for changes
  - Use the `watch()` function on the Detail component
- BUT: you *have* to use reactive variables then
  - Because we now want the update the variable inside the component when the route changes.

Updating the nested view

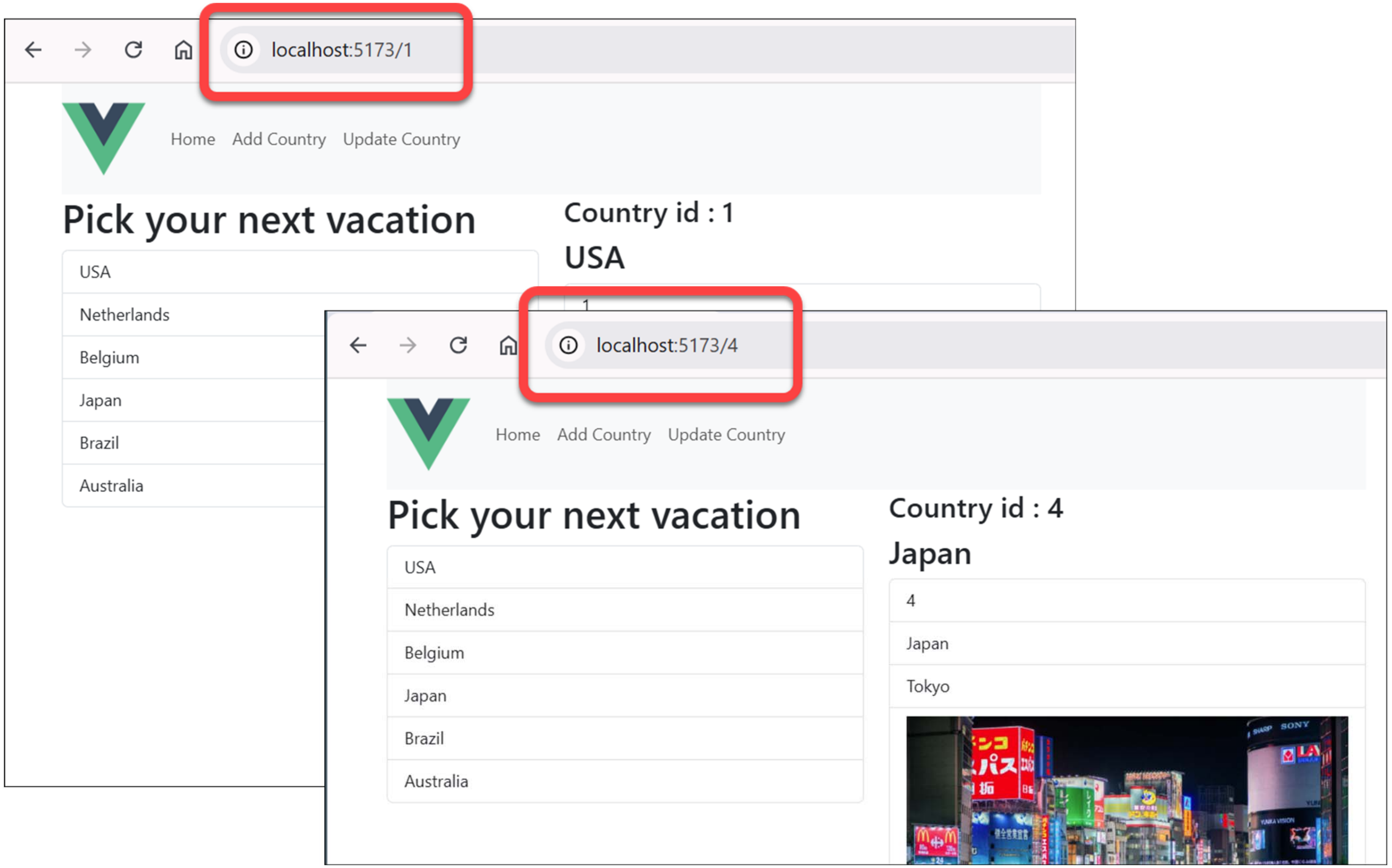
- Set a watcher on `route.params.id`.
  - The component is updated on route change

```
// ref()'s for local variables
const id = ref(parseInt(route.params.id, 10));
const country = ref(countryData.countries.find((country) => country.id === id));

// Update country once the url/route changes
watch(
    () => route.params.id,
    (newId) =>{
      console.log('load newId:: ', newId);
      id.value = +newId; // shortcut for parseInt(newId, 10);
      country.value = countryData.countries.find((country) => country.id === id.value);
    }
)
```

Update computed property `imgUrl`

- Because `country` is now a reactive value, wrapped in `ref()`, we need to update the `imgUrl`
  - Otherwise we can not load the correct image.

```
const imgUrl = computed(() => {
  // check if country exists and return the 'img' property from the 'ref()',
  // otherwise return an empty string
  return country.value
    ? new URL(`/src/assets/countries/${country.value.img}`, import.meta.url).href
    : '';

})
```



localhost:5173/1

Home Add Country Update Country

Pick your next vacation

USA
Netherlands
Belgium
Japan
Brazil
Australia

Country id : 1

USA

1

localhost:5173/4

Home Add Country Update Country

Pick your next vacation

USA
Netherlands
Belgium
Japan
Brazil
Australia

Country id : 4

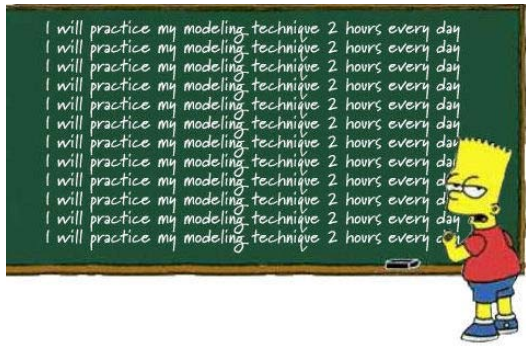
Japan

4
Japan
Tokyo

More info: https://router.vuejs.org/guide/essentials/nested-routes.html

Workshop

- Use your own application, or start from `../310-route-parameters` (the *previous* example)
- Add a nested `<router-view>`, as shown in the previous slides
  - If a country is selected, the view is updated in the same page.
  - Update the application, so that not only the `id` is shown in the URL, but also the country name.
  - Optional – update the master view, so that no `@click` event handler is used, but dynamic binding on the `<router-link :to="…">` parameter
- Generic example: `../320-nested-routing`



Guarding Routes

Globally using guards to prevent users from entering a specific route, or updating other stuff

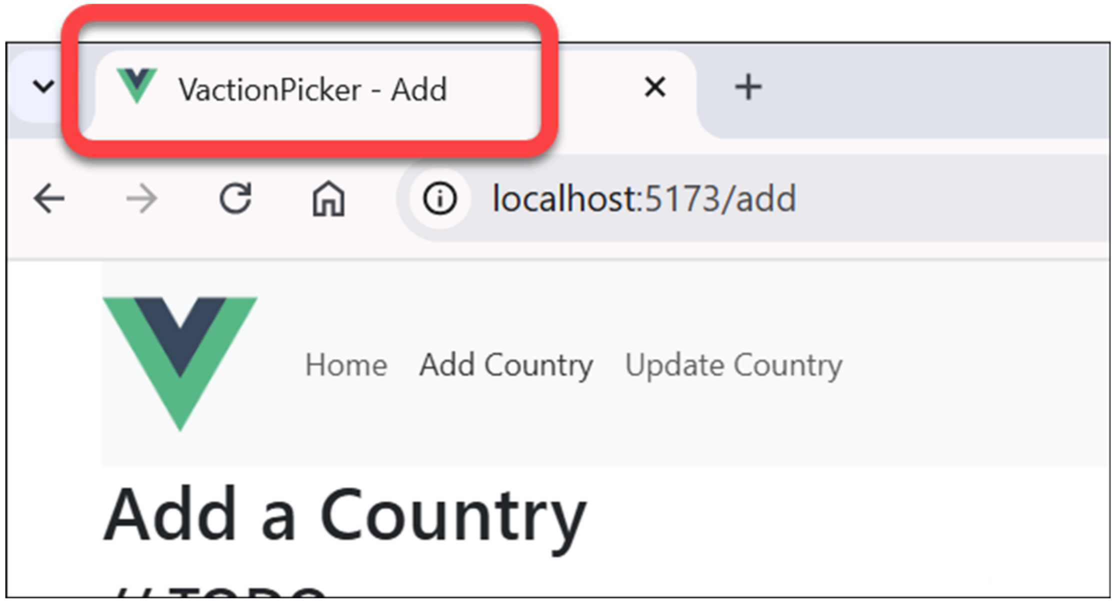
Before each route change

You can do different things before each route change.

For instance (this is global!):

```
router.beforeEach((to, from, next) => {
    // 1. Do something globally BEFORE entering a route, for instance setting the page title
    // Make sure to have a global .env file with a key and title.
    // Also see routes/index.js for the meta-information on the title per component
    const title = `${import.meta.env.VITE_APP_TITLE} - ${to.meta.title}`
    document.title = title || import.meta.env.VITE_APP_TITLE; // Fallback to app title;
    next(); // Don't forget to call next() to allow navigation

    // 2. Authenticate a user before entering a route
    // ...call authentication process....
    // return true | false, based on auth.
})
```

Bottomline: use `router.beforeEach(to, from, next)`



More info: Router Navigation Guards

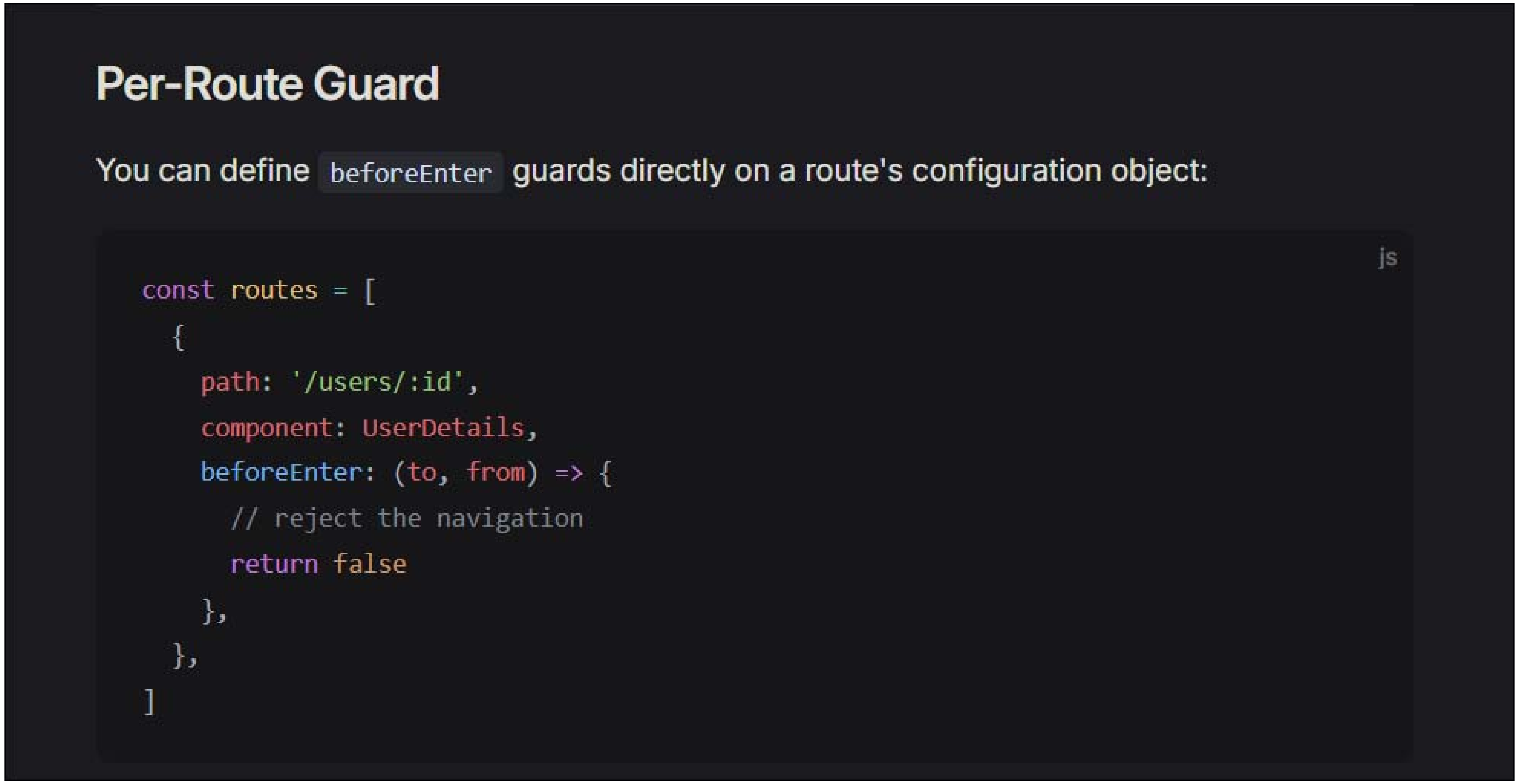
https://router.vuejs.org/guide/advanced/navigation-guards.html

Param / query changes:

“params or query changes won't trigger enter/leave navigation guards. You can either watch the `route` object to react to those changes, or use the `beforeRouteUpdate` in-component guard”

Local route guards

- Not guarding all global routes, but on a per-route base
- Use the `router` configuration object:



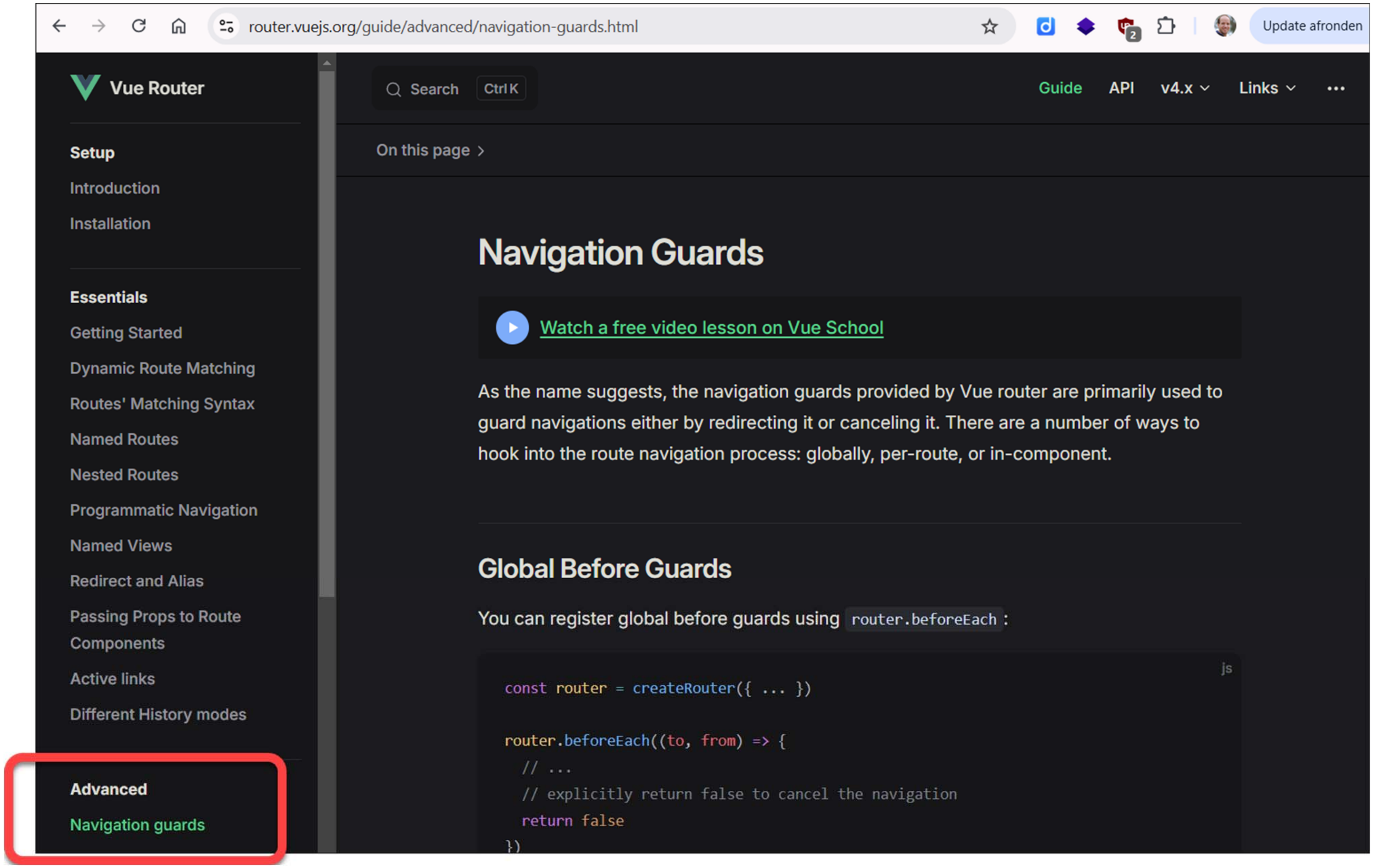
Per-Route Guard

You can define `beforeEnter` guards directly on a route's configuration object:

js

```js
const routes = [
  {
    path: '/users/:id',
    component: UserDetails,
    beforeEnter: (to, from) => {
      // reject the navigation
      return false
    },
  },
]
```

More on Navigation Guards



router.vuejs.org/guide/advanced/navigation-guards.html

Navigation Guards

Watch a free video lesson on Vue School

As the name suggests, the navigation guards provided by Vue router are primarily used to guard navigations either by redirecting it or canceling it. There are a number of ways to hook into the route navigation process: globally, per-route, or in-component.

Global Before Guards

You can register global before guards using `router.beforeEach`:

```js
const router = createRouter({ ... })

router.beforeEach((to, from) => {
  // ...
  // explicitly return false to cancel the navigation
  return false
})
```

https://router.vuejs.org/guide/advanced/navigation-guards.html

Checkpoint

- You know how to add routing to a project
- You can create a routing table in an `index.js`, or `router.js`
- You can use `<router-view>` and `<router-link>`
- You can add styles and conditional styles to routes
- You know how to navigate from code, by using `router.push(…)`
- You can use route parameters (`/:<some-name>`) to create dynamic routes
- You know how to enable HTML5 history mode on the router
- You are able to use nested router views