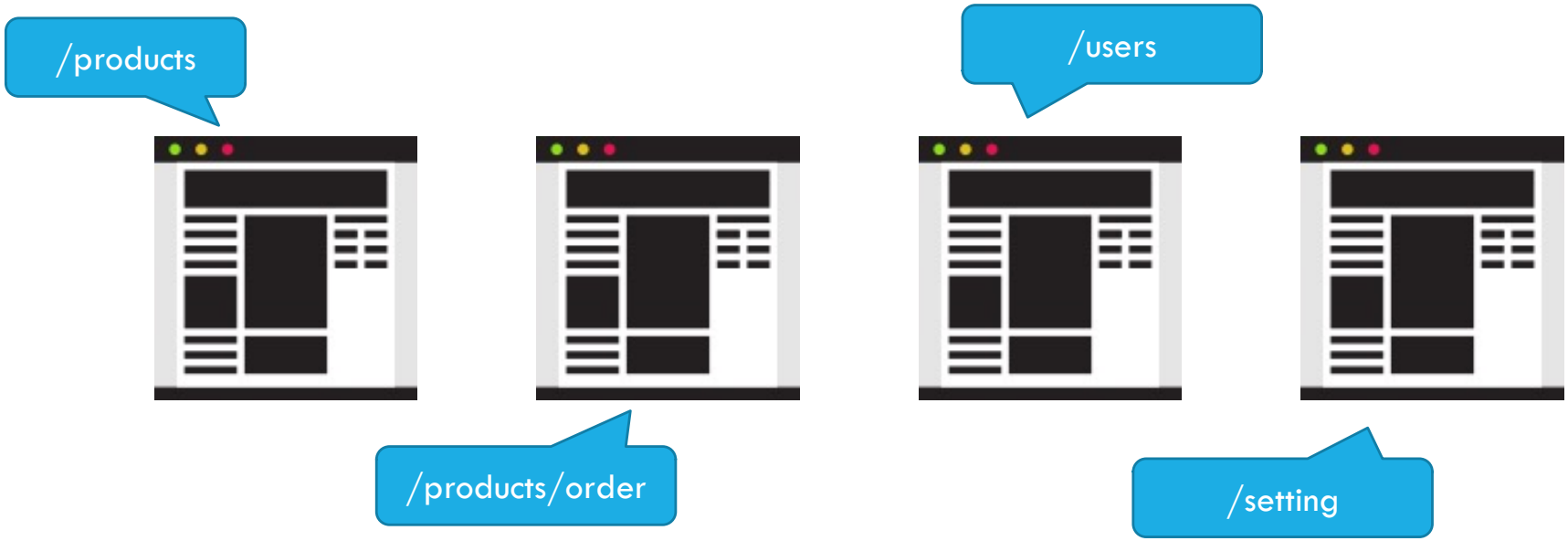
VUE ROUTER

The dynamic client routing

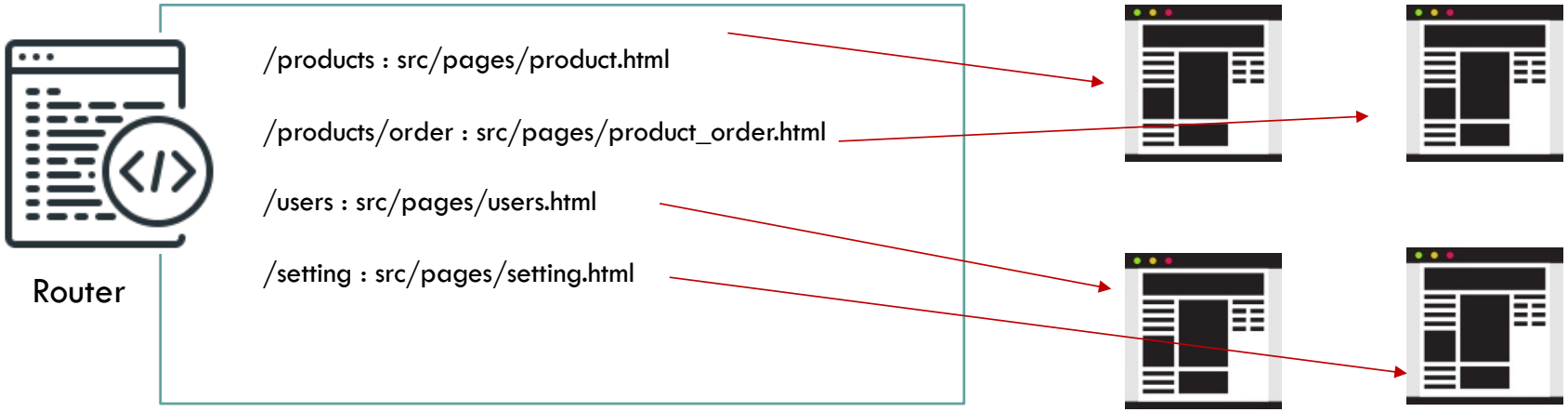
WHY VUE ROUTER?

When you are building a **SPA**, it is unavoidable that you gotta have multiple pages which are render on the client side.



WHY VUE ROUTER?

So how can you organize your application url and map them with your page document?



SETUP VUE ROUTER

1. **Install vue router module**

If you are using VueJs 3

```
$ npm i vue-router@next
```

Or **vue-router** module otherwise

SETUP VUE ROUTER

2 **Add routing directory and configuration file**

There are a few important items to take note:

- Path : url path where this route can be found
- Name: optional name to use when we link to this route
- Component: component to load when route is called

```
import { createWebHistory, createRouter } from "vue-router";
import Home from "@/views/Home.vue";
import About from "@/views/About.vue";

const routes = [
  {
    path: "/",
    name: "Home",
    component: Home,
  },
  {
    path: "/about",
    name: "About",
    component: About,
  },
];

const router = createRouter({
  history: createWebHistory(),
  routes,
});

export default router;
```

SETUP VUE ROUTER

3 **Import and use Vue Router**

Update main.js file to use our router in our application.

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router' // <---

createApp(App).use(router).mount('#app')
```

Apply the imported router to the vue instance with function **use()**

SETUP VUE ROUTER

4 **Use <router-view> and <router-link>**

There are 2 directives related to Vue router:

<router-view /> - When a route is navigated to in the browser, this is where the component is rendered. For example, in our code going to / will render the Home component where we list <router-view />.

<router-link> - This is the directive we use to create links between our different component pages, instead of using <a href>.

```vue
<template>
  <div id="nav">
      <router-link to="/">Home</router-link> |
      <router-link to="/about">About</router-link>
  </div>
  <router-view />
</template
```

/src/App.vue

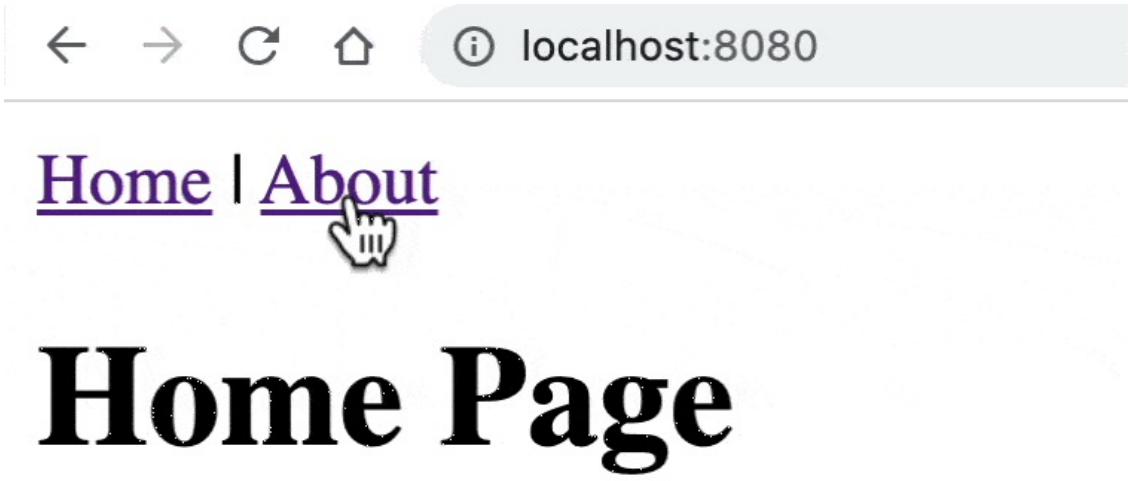
```vue
<template>
  <h1>About Page</h1>
</template>
```

/src/views/About.vue

```vue
<template>
  <h1>Home Page</h1>
</template>
```

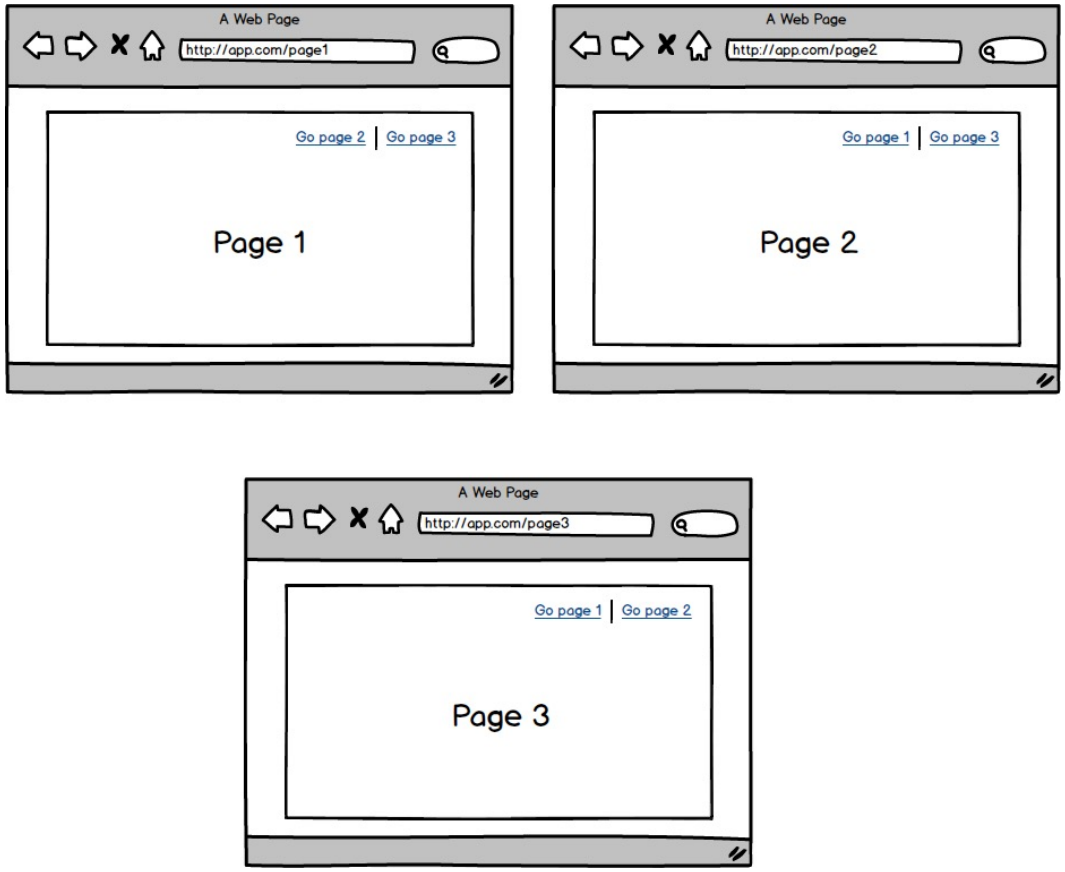
/src/views/Home.vue

You will get something like this:



EXERCISE 1

Create something like this:



Named route

```
<router-link :to="{ name: 'Home' }">Home</router-link> |
<router-link :to="{ name: 'About' }">About</router-link>
```

You can use route name instetad of path

Dynamic segments

You may pass param through the url

```
{
  path: "/user/:name",  // <-- notice the colon
  name: "User",
  component: User,
},
```

The parameter can be accessed this way:

```
<template>
  <h1>The user is {{ $route.params.name }}</h1>
</template>
```

Or this way:

```
<template>
  <h1>The user is {{ name }}</h1>
</template>
<script>
export default {
  props: ["name"],
};
</script>
```

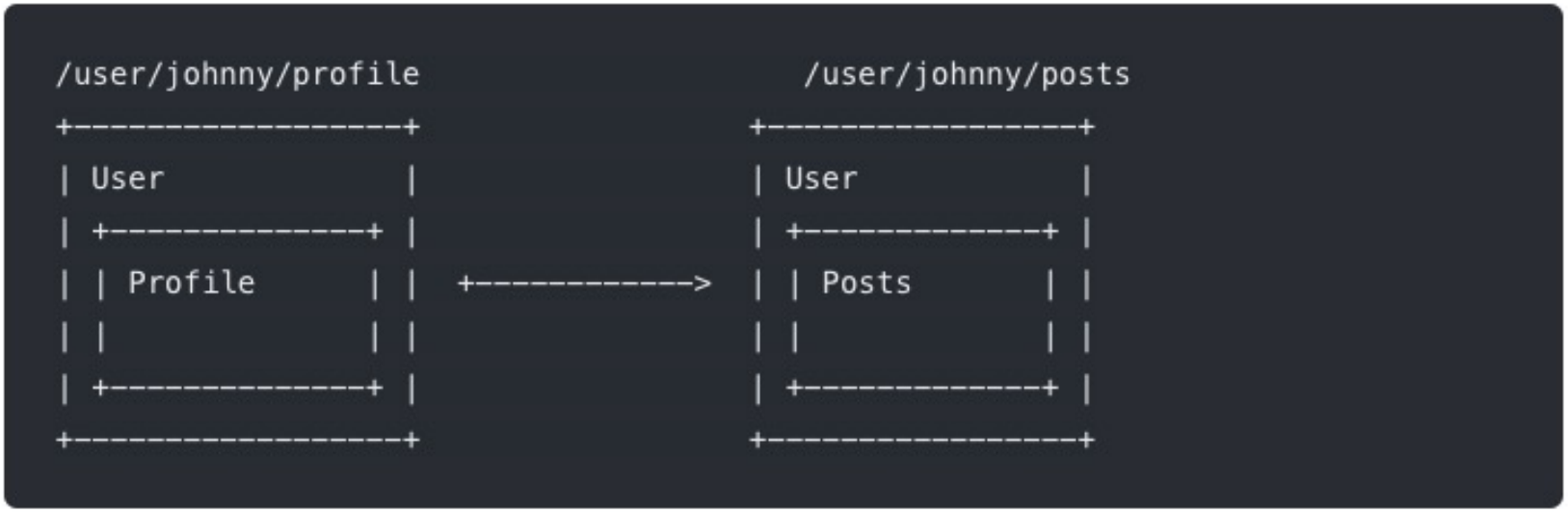
Handling 404 Not Found

```
{
  path: "/:catchAll(.*)",
  component: NotFound,
},
```

/src/views/NotFound.vue

```
<template>
  <h1>Oops, it looks like the page you're looking for
doesn't exist.</h1>
</template>
```

Nested routes



```js
const routes = [
  {
    path: '/user/:id',
    component: User,
    children: [
      {
        // UserProfile will be rendered inside User's <router-view>
        // when /user/:id/profile is matched
        path: 'profile',
        component: UserProfile,
      },
      {
        // UserPosts will be rendered inside User's <router-view>
        // when /user/:id/posts is matched
        path: 'posts',
        component: UserPosts,
      },
    ],
  },
]
```

EXERCISE 2

Create something like this: