Directives HTTP and Routing

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://about.softuni.bg

Table of Contents

Have a Question?

Software University

sli.do

#vue-js

Creating Custom Directives

Custom directives

```
<template>
      <input v-focus />
</template>

<script>
      const focus = {
        mounted: (el) => el.focus()
      }

      export default {
        directives: {
          focus
        }
      }
</script>
```

Registering custom directives

- Locally via the **directives** option

```
<script>
export default {
    directives: {
        // make v-focus usable in this
    component
        focus:{ /* ... */ }
    }
}
</script>
```

- Globally in your app

```
const app = createApp({})

// make v-focus usable in all components
app.directive('focus', {
  /* ... */
})
```

Directive definition

- Definition **object**
- Provides several hook functions
- All hooks are **optional**

```
const myCustomDirective = {
    created(el, binding, vnode, prevVnode) { … },
    beforeMount(el, binding, vnode, prevVnode) ) { … },
    mounted(el, binding, vnode, prevVnode) ) { … },
    beforeUpdate(el, binding, vnode, prevVnode) ) { … },
    updated(el, binding, vnode, prevVnode) ) { … },
    beforeUnmount(el, binding, vnode, prevVnode) ) { … },
    unmounted(el, binding, vnode, prevVnode) ) { … },
}
```

Directive hooks

- **created()**
  - Called before bound element's attributes or event listeners are applied.
- **beforeMount()**
  - Called right before the element is inserted into the DOM.
- **mounted()**
  - Called when the bound element's parent component and all its children are mounted.
- **beforeUpdate()**
  - Called before the parent component is updated.
- **updated()**
  - Called after the parent component and all of its children have updated.
- **beforeUnmount()**
  - Called before the parent component is unmounted.
- **unmounted()**
  - Called when the parent component is unmounted.

Directive arguments

- **el**: the element the directive is bound to.
- **binding**: an object containing the following properties.
  - **value**: The value passed to the directive.
  - **oldValue**: The previous value, only available in beforeUpdate and updated.
  - **arg**: The argument passed to the directive, if any.
  - **modifiers**: An object containing modifiers, if any.
  - **instance**: The instance of the component where the directive is used.
  - **dir**: the directive definition object.
- **vnode**: the underlying VNode representing the bound element.
- **prevNode**: the VNode representing the bound element from the previous render. Only available in beforeUpdate() and updated().

Directive binding argument

- **binding**
  - **value**: The value passed to the directive.

```
v-my-directive="1 + 1"
// value: 2
```

  - **arg**: The argument passed to the directive, if any.

```
`v-my-directive:foo`
// arg: foo
```

  - **modifiers**: An object containing modifiers, if any.

```
`v-my-directive.foo.bar`
// modifiers: { foo: true, bar: true }
```

Problem - Directives

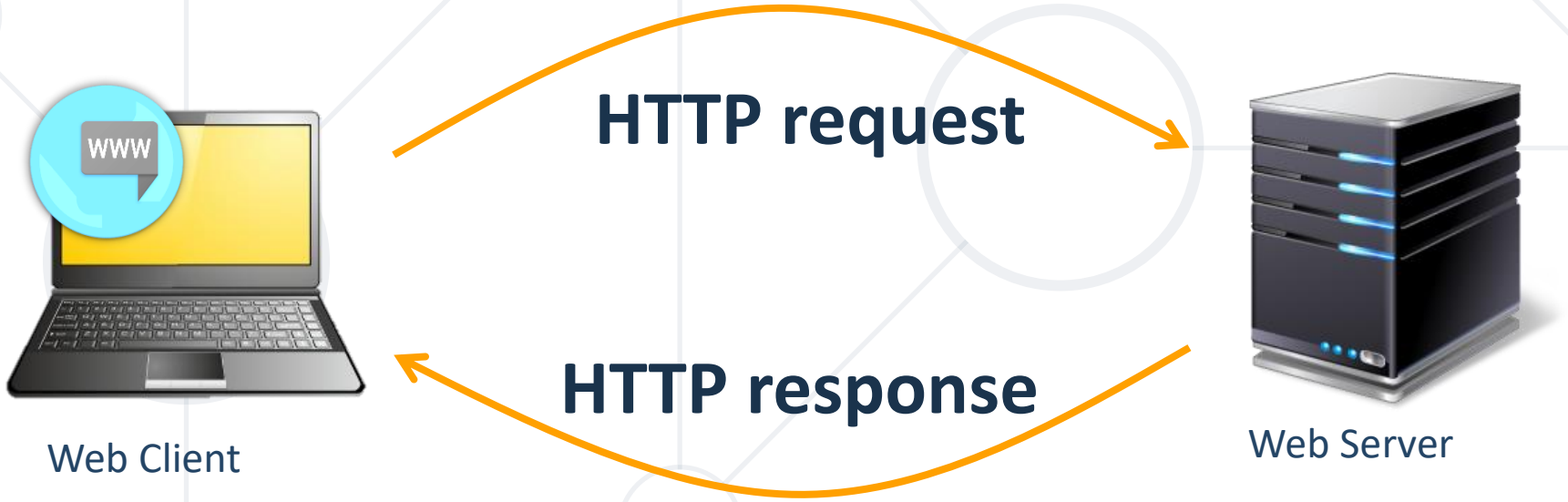
- Write a custom directive called **v-color**
- The directive should accept a string and change the **color** of the element
- Bonus
  - Rework the directive to accept a property as an argument – **color** or **background**
  - Depending on the argument change if you set the color or background-color of the element
  - If no argument is provided, use **color** as a default option

HTTP

Reaching Out to a Server, Axios

HTTP Basics

- HTTP (**H**yper **T**ext **T**ransfer **P**rotocol)
  - Text-based client-server protocol for the Internet
  - For transferring Web resources (HTML files, images, styles, etc.)
  - Request-response based



HTTP Request Methods

HTTP defines **methods** to indicate the desired action to be performed on the identified resource

Method	Description
GET 	Retrieve / load a resource
POST 	Create / store a resource
PUT 	Update a resource
DELETE 	Delete (remove) a resource
PATCH 	Update resource partially
HEAD 	Retrieve the resource's headers
OPTIONS	Returns the HTTP methods that the server supports for the specified URL

Axios

- **Axios** is promise based HTTP client for the browser and nodejs
  - Make **XMLHttpRequests** from the browser
  - Supports the **Promise** API
  - Intercept request and response
  - Transform request and response data
  - Automatic transforms for JSON data
  - Client-side support for protecting against **XSRF**

Axios - Usage

- Installing

```
npm install axios
```

- Docs - https://axios-http.com
- CDN

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

Axios GET request

Software University

```
<script>
import axios from 'axios';
export default {
  async created() {
    const apiUrl = 'https://api.example.com/your-endpoint';
    try {
      const response = await axios.get(apiUrl);
      this.data = response.data;
    } catch (error) {
      console.error('Error fetching data:', error);
    }
  },
};
</script>
```

Axios POST request

```
methods: {
    async postData() {
      const apiUrl = 'https://api.example.com/your-post-endpoint';
      try {
        const response = await axios.post(apiUrl, {
          firstName: 'John',
          lastName: 'Doe',
        });
        console.log('Post Response:', response.data);
      } catch (error) {
        console.error('Error posting data:', error);
      }
    },
  },
```

Performing API requests in Vue

- **created()** lifecycle:
  - Use the **created()** lifecycle hook to send API requests when the component is first initialized.
  - This is useful for fetching initial data required for rendering the component.
- **mounted()** lifecycle :
  - Use the mounted lifecycle hook to send API requests after the component is inserted into the DOM.
  - This is often used when you need to fetch data that interacts with the DOM or requires access to the component's root element.

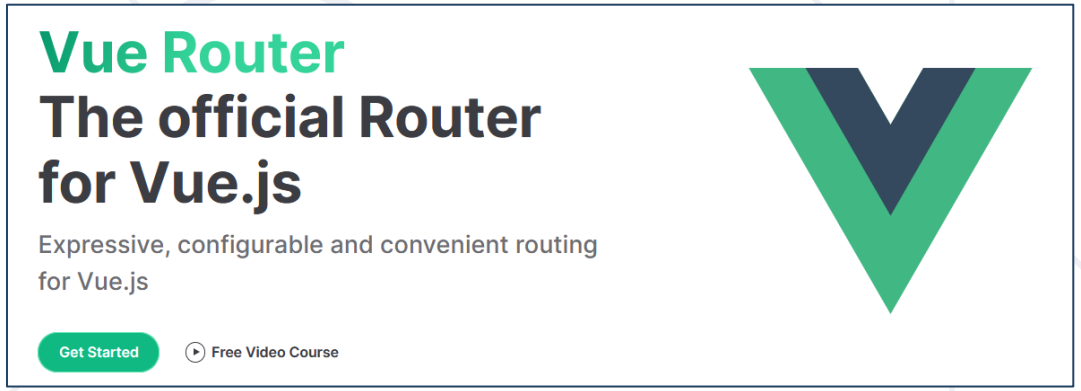
Creating Pages

Vue Router, Routes, Links and more

Vue router

- **Documentation:** Vue Router Documentation
- **Features:**
  - Nested route/view mapping
  - Modular, component-based router configuration
  - Route params, query, wildcards
  - Fine-grained navigation control
- **Installation**

```
npm install vue-router@4
```



Router setup

- Create a **routes/index.js**
- Create routes array
  - Array of Objects
  - **Path** to show in the url and **component** to render
- Create router instance with **createRouter()**
- Export the router instance

```
// routes/index.js
import { createRouter, createWebHistory } from
"vue-router";
import Home from "../components/Home.vue"
import About from "../components/About.vue"

const routes = [
  { path: '/', component: Home },
  { path: '/about', component: About },
]

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

export default router;
```

Router setup - register the plugin in Vue

- Define a place where the components from the router will be loaded
- Use the **<router-view/>** tag

```
// main.js
import router from "./routes/index";

const app = createApp(App);
app.use(router);
app.mount("#app");
```

```
// App.vue

<template>
  <router-view></router-view>
</template>
```

Navigating with Router Links

- **<router-link>**
  - Declare an "internal link" element to work with our router
  - Use the **to** prop to pass a Route Location the link should navigate to when clicked on.

```
<router-link to="/">Go to Home</router-link>
<router-link to="/about">Go to About</router-link>
```

```
// You can use the `custom` prop to not render an `a` tag
<router-link to="/about" custom>
      <li>About page</li>
</router-link>
```

Dynamic Router Links

- The **to** prop can of course be dynamic if we bind it to a data property

```
<router-link :to="link.url">
      {{ link.label }}
</router-link>
```

```
data(){
       return {
              link: {
                     url:'user/edit',
                     label: 'Edit your profile'
              }
       }
}
```

Programmatic Navigation

- Aside from using **<router-link>** to create anchor tags for declarative navigation, we can do this programmatically using the router's instance methods.

```
// Literal string path
this.$router.push("user/edit");

// Object with path
this.$router.push({ path: "user/edit" });
```

Route params / Dynamic Routes

- A param is denoted by a colon **:**
- When a route is matched, the value of its params will be exposed as **this.$route.params** in every component

```
const routes = [
  // dynamic segments start with a colon
  { path: '/users/:id', component: User },
]
```

```
data() {
  return {
    id: this.$route.params.id, // ID's
  }
}
```

Named Routes

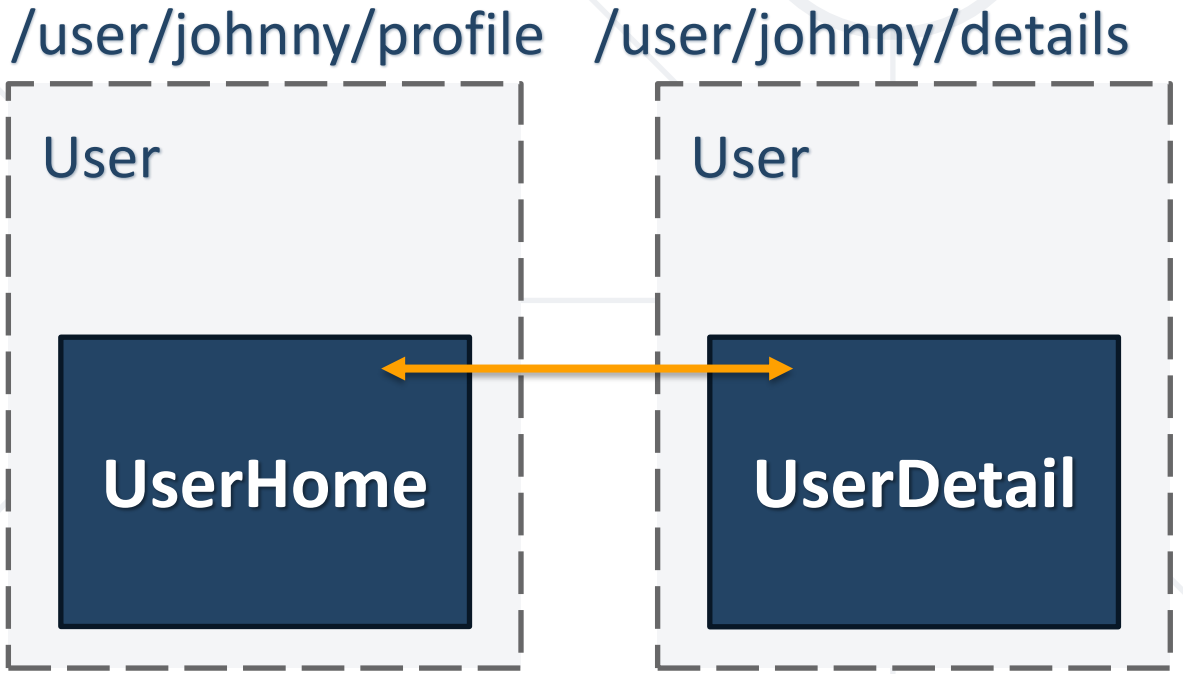
- Alongside the **path**, you can provide a **name** to any route
  - No hardcoded URLs
  - Automatic encoding/decoding of **params**
  - Prevents you from having a typo in the url

```
{ path: '/user/:id', name: 'userEdit', component: UserEdit }

<router-link :to="{ name: 'userEdit', params: { id: 123 }}">
     User Edit
</router-link>

this.$router.push({ name: 'userEdit', params: { id: 123 }})
```

Nested routes

- Setup nested/child routes with the children property
- Add a new **<router-view />** to the "parent" component (UserStart)
- Nested paths that start with / will be treated as root paths

```
const routes = [{
  path: '/user', component: User, children: [
    { path: '', component: UserHome },
    { path: ':id', component: UserDetail }
    { path: ':id/edit', component: UserEdit }
  ]
}]
```



Redirect & Wildcards

- Redirecting is also done in the `routes` configuration

```
const routes = [{ path: '/home', redirect: '/' }]

const routes = [{ path: '/home', redirect: { name: 'homepage' } }]
```

- Wildcards

```
const routes = [
  // will match everything and put it under `$route.params.pathMatch`
  { path: '/:pathMatch(.*)*', name: 'NotFound', component: NotFound },


  // will match anything starting with `/user-` and put it under
`$route.params.afterUser`
  { path: '/user-:afterUser(.*)', component: UserGeneric },
]
```

Navigation Guards

Navigation Guards

- Used to guard navigations either by redirecting it or canceling it
- There are several ways to hook into the route navigation process
  - Globally
  - Per-route
  - In-component

Global Before Guards

- **to** - the target route location being navigated to.
- **from** - the current route location being navigated away from.
- Can return any of these:
  - **Nothing, undefined, or true** - the navigation is valid and proceeds to the next step.
  - **False** - cancel the current navigation
  - **A route location** - redirect to a different location

```
// routes/index.js
const router = createRouter({ ...
})

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

Global After Hooks

- You can also register global after hooks, however unlike guards, these hooks cannot affect the navigation

```
router.afterEach((to, from) => {
  sendToAnalytics(to.fullPath)
})
```

Beware of Infinite redirect

```
router.beforeEach((to, from) => {
  if (
    // make sure the user is authenticated
    !isAuthenticated &&
    //  ❗ Avoid an infinite redirect
    to.name !== 'Login'
  ) {
    // redirect the user to the login page
    return { name: 'Login' }
  }
})
```

Per-Route Guard

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

In-Component Guards

Software University

You can add the following options to route components:

- **beforeRouteEnter(to, from)**:
  - Called before the route that renders this component is confirmed
  - Does NOT have access to **this** (component instance)

In-Component Guards

- **beforeRouteUpdate(to, from)**:
  - When the route that renders this component has changed a param
  - When we navigate between /users/1 and /users/2
  - Has access to this component instance
- **beforeRouteLeave(to, from)**:
  - When the route that renders this component is about to be navigated away from
  - Does NOT have access to **this** (component instance)

Data Fetching

With Vue Router

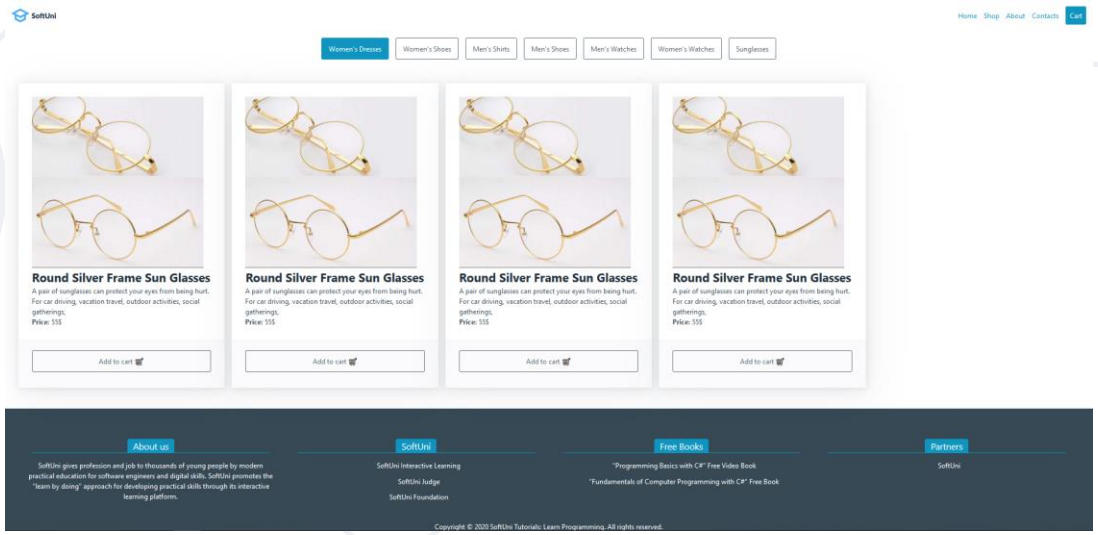
Fetching Before Navigation

- We fetch the data before actually navigating to the new route
- Perform the data fetching in the **beforeRouteEnter()** with a 3rd argument – **next()**
- Only call **next** when the fetch is complete
- The callback passed to **next** will be called **after the component is mounted**

```
export default {
  beforeRouteEnter(to, from, next) {
    getPost(to.params.id, (err, post) => {
      next(vm => vm.setData(post)) // `setData` is a method defined below
    })
  },
  ….
}
```

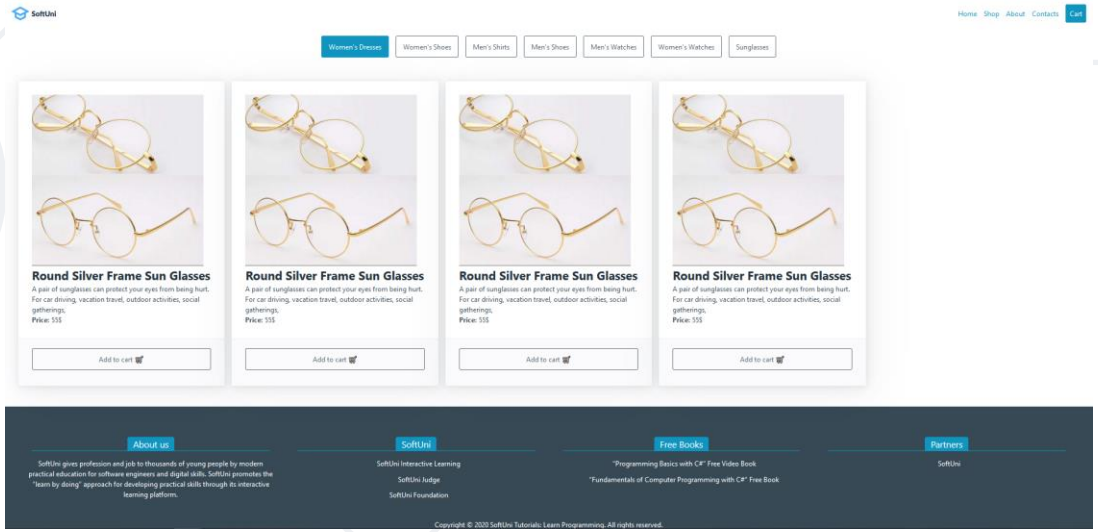
Exercise – Routing and Loading

- **Use your project from Workshop – Components**
- **Routing**
  - Install and connect **Vue Router** to your application
  - Create **routes for each page** from the navigation
  - Should render the correct page with Vue Router and update the URL in the address



Exercise – Routing and Loading

- **Loading**
  - Install **axios** and analyze the free API https://dummyjson.com/
  - Load and render all categories in "Store" from **dummyjson**, not from local JSON file
  - Load all products from **dummyjson**, not from local JSON file
  - Think about how you can filter the products with the API calls



Practice

Live Exercise in Class (Lab)

Summary

- We can create custom directives to customize the DOM elements
- We can easily perform data fetching in Vue components
- Vue Router is flexible and powerful way to add routes and navigation to our Vue app

Questions?

SoftUni

Software University | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids | Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter™ International

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank
Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg