Frontend

Software Engineering, 2nd part - Lab

Marco Robol - marco.robol@unitn.it

Academic year 2021/2022 - Second semester

Plan for the second part of the course

- April 19-21: Design thinking, project arch, API
- April 26-28: Foundations JS, Node.js, git
- May 2-5: Agile Methodology, MongoDB, API
- ***May 9 - May 22: Sprint #1***
  - More on agile methodology, testing, git branching
- May 23 - June 7: Sprint #2
  - More on testing, devops/CI

Contents of today class

- Basic frontend with HTML, fetch() API, DOM manipulation
- Vue.js
- EasyLib

Repository:

> EasyLib - https://github.com/unitn-software-engineering/EasyLib
> EasyLibVue - https://github.com/unitn-software-engineering/EasyLibVue

Try:

> Basic Frontend - https://easy-lib.herokuapp.com/
> Vue Frontend - https://unitn-software-engineering.github.io/EasyLibApp/

Basic frontend technologies

> Fetch API - scotch.io/tutorials/how-to-use-the-javascript-fetch-api-to-get-data, developers.google.com/web/updates/2015/03/introduction-to-fetch
> Fetch vs XMLHttpRequest - www.sitepoint.com/xmlhttprequest-vs-the-fetch-api-whats-best-for-ajax-in-2019

> DOM manipulation - www.w3schools.com/js/js_htmldom.asp, developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents

> Responsive UI with Bootstrap - getbootstrap.com

EasyLib basic frontend

> https://easy-lib.herokuapp.com/

Nel progetto EasyLib trovate una semplice web app implementata in html + javascript. Le chiamate asincrone al backend sono effettuate manualmente tramite il metodo fetch(), mentre la pagina html è aggionata lavorando direttamente sul DOM della pagina html.

EasyLib basic frontend is composed by `EasyLib\static\index.html` and `EasyLib\static\script.js`. Let's run the server and check out the frontend

Una pagina html + javascript è la UI minima richiesta per il progetto!

Basic frontend - HTML

`EasyLib\static\index.html`

```
<h1>EasyLib</h1>

<form action="api/v1/students" method="post" name="loginform" id="loginform">
  <span>Logged User:</span> <span id="loggedUser"></span>
  <input name="email" value="mario.rossi@unitn.com" id="loginEmail"/>
  <input name="email" value="123" id="loginPassword"/>
  <button type="button" onclick="login()">LogIn</button>
</form>

<h2>Books:</h2> <ul id="books"></ul>

<h2>Insert new book:</h2>
<form action="api/v1/books" method="post" name="bookform" id="bookform">
  <input name="title" value="title" id="bookTitle"/>
  <button type="button" onclick="insertBook()">Insert new book</button>
</form>

<script src="script.js"></script>
```

Basic frontend - Javascript 1/2

`EasyLib\static\script.js`

```
var loggedUser = {} // This variable stores the logged in user

//This function is called when login button is pressed.
function login() { ... }
// This function refresh the list of books
function loadBooks() { ... }
loadBooks();
// This function is called by the Take button beside each book.
function takeBook(bookUrl) { ... }
// This function refresh the list of bookLendings.
function loadLendings() { ... }
// This function is called by clicking on the "insert book" button.
function insertBook() { ...}
```

Basic frontend - Javascript 2/2

`EasyLib\static\script.js` `function login()`

```
function login() {
  var email = document.getElementById("loginEmail").value;
  var password = document.getElementById("loginPassword").value;

  fetch('../api/v1/authentications', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify( { email: email, password: password } ),
  }).then((resp) => resp.json()) // Transform the data into json

  .then(function(data) { // Here you get the data to modify as you please
    loggedUser.token = data.token;
    loggedUser.email = data.email;
    loggedUser.id = data.id;
    loggedUser.self = data.self;
    document.getElementById("loggedUser").innerHTML = loggedUser.email;
    loadLendings();
    return;
  }).catch( error => console.error(error) );
};
```

Vue.js

> Vue (pronounced /vjuː/, like view) is a JavaScript framework for building user interfaces. It builds on top of standard HTML, CSS and JavaScript, and provides a declarative and component-based programming model that helps you efficiently develop user interfaces, be it simple or complex.
> https://vuejs.org/

```
import { createApp } from 'vue'
createApp({
  data() { return { count: 0 } }
}).mount('#app')
```

```
<div id="app">
  <button @click="count++">
    Count is: {{ count }}
  </button>
</div>
```

- **Declarative Rendering**: Vue extends standard HTML with a template syntax that allows us to declaratively describe HTML output based on JavaScript state.
- **Reactivity**: Vue automatically tracks JavaScript state changes and efficiently updates the DOM when changes happen.

Quick Start vuejs.org/guide/quick-start

With Build Tools - A build setup allows us to use Vue Single-File Components (SFCs). The official Vue build setup is based on Vite, a frontend build tool.

- **Online** - You can try Vue with SFCs online on StackBlitz (vite.new/vue). StackBlitz runs the Vite-based build setup directly in the browser, so it is almost identical to the local setup but doesn't require installing anything on your machine.
- **Local** - `npm init vue@latest` This command will install and execute create-vue, the official Vue project scaffolding tool. `npm run dev`

Without Build Tools - To get started with Vue without a build step, simply copy the following code into an HTML file - vuejs.org/guide/quick-start.html#without-build-tools

Vue 3 API Styles

> Vue components can be authored in two different API styles: Options API and Composition API - vuejs.org/guide/introduction.html#api-styles

With **Options API**, we define a component's logic using an object of options.

```
<script>
export default {
  data() { return { count: 0 } },
  methods: {} ...
```

With **Composition API**, we define a component's logic using imported API functions.

```
<script setup>
import { ref, onMounted } from 'vue'
const count = ref(0) // reactive state
function increment() { count.value++ } ...
```

vuejs.org/guide/quick-start#next-steps

- Continue the Guide - The guide walks you through every aspect of the framework.
- Try the Tutorial - For those who prefer learning things hands-on.
- Check out the Examples - Explore examples of core features and common UI tasks.

Let's check out the tutorial

https://vuejs.org/tutorial/

EasyLibVue

> Repository - https://github.com/unitn-software-engineering/EasyLibVue

TRY: https://unitn-software-engineering.github.io/EasyLibApp/

Creation with build tools - vuejs.org/guide/quick-start.html#local

Clone EasyLibVue, or start from scratch, `npm init vue@latest` to execute create-vue.

https://github.com/unitn-software-engineering/EasyLibVue/commit/c8c43e8d684817e91fd81a01074c27914899bf62 - Here is what create-vue provides us.

To run: `npm install` and `npm run dev`. Application will be server on port 3030.

Because our frontend origin resolves to a different domain with respect to where or APIs are exposed (port 8080), we won't be able to make asynchronous requests from the browser to our APIs server, because of the Cross-Origin policy of browser!

CORS and Preflight request

> Cross-Origin Resource Sharing (CORS) is an HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources.
> https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

> CORS also relies on a mechanism by which browsers make a "preflight" request to the server hosting the cross-origin resource, in order to check that the server will permit the actual request. In that preflight, the browser sends headers that indicate the HTTP method and headers that will be used in the actual request.
> https://developer.mozilla.org/en-US/docs/Glossary/Preflight_request

> Tutorial - https://web.dev/cross-origin-resource-sharing/

Supporting CORS in Node.js `EasyLib\app\app.js`

```
app.use(function (req, res, next) { // Add headers before the routes are defined
    // Website you wish to allow to connect
    res.setHeader('Access-Control-Allow-Origin', 'http://localhost:3000');
    // Request methods you wish to allow
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');
    // Request headers you wish to allow
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');
    // Set to true if you need the website to include cookies in the requests sent
    // to the API (e.g. in case you use sessions)
    res.setHeader('Access-Control-Allow-Credentials', true);
    // Pass to next layer of middleware
    next();
});
```

Using a module https://expressjs.com/en/resources/middleware/cors.html

```
const cors = require('cors')
app.use(cors())
```

Let's go back on our EasyLib Vue frontend

`EasyLibVue\src\App.vue`

```
import Login from '@/components/Login.vue'
...
<nav>
  ...
  <RouterLink to="/books">Books</RouterLink>
  <RouterLink to="/booklendings">Booklendings</RouterLink>
</nav>
<Login />
...
```

EasyLibVue\src\components\Login.vue

```
<script setup>
import { ref } from 'vue'
import { loggedUser, setLoggedUser, clearLoggedUser } from '../states/loggedUser.js'
const HOST = `http://localhost:8080`;
const email = ref('mario.rossi@unitn.com');
const password = ref('123')
function login() {
    fetch(HOST+'/api/v1'+'/authentications', {
        method: 'POST', headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify( { email: email.value, password: password.value } ),
    })
    .then((resp) => resp.json()) // Transform the data into json
    .then(function(data) { setLoggedUser(data) })
};
function logout() { clearLoggedUser() }
</script>
<template>
  <form>
    <span v-if="loggedUser.token">
      Welcome <a :href="HOST+'/'+loggedUser.self">{{loggedUser.email}}</a>
      <button type="button" @click="logout">LogOut</button>
    </span>
    <span v-if="!loggedUser.token">
      <input name="email" v-model="email" />
      <input name="password" v-model="password" />
      <button type="button" @click="login">LogIn</button>
    </span>
  </form>
</template>
```

EasyLibVue\src\states\loggedUser.js

> https://vuejs.org/guide/scaling-up/state-management.html#simple-state-management-with-reactivity-api

```
import { reactive } from 'vue'
const loggedUser = reactive({
    token: undefined, email: undefined,
    id: undefined, self: undefined
})

function setLoggedUser (data) {
    loggedUser.token = data.token; loggedUser.email = data.email;
    loggedUser.id = data.id; loggedUser.self = data.self;
}
function clearLoggedUser () {
    loggedUser.token = undefined; loggedUser.email = undefined;
    loggedUser.id = undefined; loggedUser.self = undefined;
}

export { loggedUser, setLoggedUser, clearLoggedUser }
```

EasyLibVue\src\router\index.js

```
import HomeView from '../views/HomeView.vue'
routes: [
  {
    path: '/',
    name: 'home',
    component: HomeView
  },
  {
    path: '/books',
    name: 'books',
    // route level code-splitting
    // this generates a separate chunk (About.[hash].js) for this route
    // which is lazy-loaded when the route is visited.
    component: () => import('../views/BooksView.vue')
  },
  {
    path: '/booklendings',
    name: 'booklendings',
    component: () => import('../views/BooklendingsView.vue')
  }
]
```

EasyLibVue\src\views\BooklendingsView.vue

```
<script setup>
import BooklendingsTable from '@/components/BooklendingsTable.vue'
</script>

<template>
  <div>
    <h1>Booklendings:</h1>
      <BooklendingsTable />
  </div>
</template>

<style>
</style>
```

EasyLibVue\src\components\BooklendingsTable.vue

```
<script setup>
import { loggedUser } from '../states/loggedUser.js'
import { books, fetchBooks, createBook, deleteBook } from '../states/books.js'

const HOST = `http://localhost:8080`; const API_URL = HOST+`/api/v1`;
const booklendings = ref([])
onMounted( () => { fetchBooks(); fetchData(); })
watch(loggedUser, (_loggedUser, _prevLoggedUser) => { fetchBooks(); fetchData(); })

async function fetchData() {
  if (!loggedUser.token) { booklendings.value = []; return; }
  const url = API_URL+'/booklendings?studentId=' + loggedUser.id + '&token=' + loggedUser.token
  booklendings.value = await (await fetch(url)).json()
}
async function deleteLending(lending) {...};
</script>
<template>
  <span v-if="loggedUser.token"> Here are you booklendings, {{loggedUser.email}}: </span>
  <span v-if="!loggedUser.token" style="color: red"> 'Please login to visualize booklendings!' </span>
  <ul>
    <li v-for="lending in booklendings" :key="lending.self">
      <a :href="HOST+lending.book">{{ ( books.value.find(b=>b.self==lending.book) || {title: 'unknown'} ).title}}</a> -
      <button @click="deleteLending(lending)">RETURN {{lending.self}}</button>
    </li>
  </ul>
</template>
```

Questions?

marco.robol@unitn.it

Back up slides

Vue.js additional pointers

Qui un tutorial su come consumare servizi REST da un applicazione Vue.js: https://bezkoder.com/vue-js-crud-app/

In particolare un tutorial sullo stack Vue.js + Node.js + Express + MongoDB example è presente qui: https://bezkoder.com/vue-node-express-mongodb-mevn-crud/

Autenticazione con JWT: https://bezkoder.com/jwt-vue-vuex-authentication/

Build and serve Vue app from our backend

When ready to ship app to production, run the following: `npm run build`. This generates minified html+javascript frontend in `.\dist` folder. We can then serve the frontend on a dedicated server or on our API server.

```
// Serving frontend files from process.env.FRONTEND
app.use('/', express.static(process.env.FRONTEND || 'static'));
// If request not handled, try in ./static
app.use('/', express.static('static'));
// If request not handled, try with next middlewares ...
```

`EasyLib\app\app.js`

```
# Path to external frontend - If not provided, basic frontend in static/index.html is used
FRONTEND='../EasyLibVue/dist'
```

`EasyLib\.env`

- Serving over HTTP using ES modules syntax

```
<script type="importmap">
  {
    "imports": {
      "vue": "https://unpkg.com/vue@3/dist/vue.esm-browser.js"
    }
  }
</script>

<div id="app">{{ message }}</div>

<script type="module">
  import { createApp } from 'vue'

  createApp({
    data() {
      return {
        message: 'Hello Vue!'
      }
    }
  }).mount('#app')
</script>
```

Vuetify - Material Design Framework

> Vue UI Library with beautifully handcrafted Material Components. No design skills required - everything you need to create amazing applications is at your fingertips.
> https://vuetifyjs.com/en/

Only for Vue 2.0