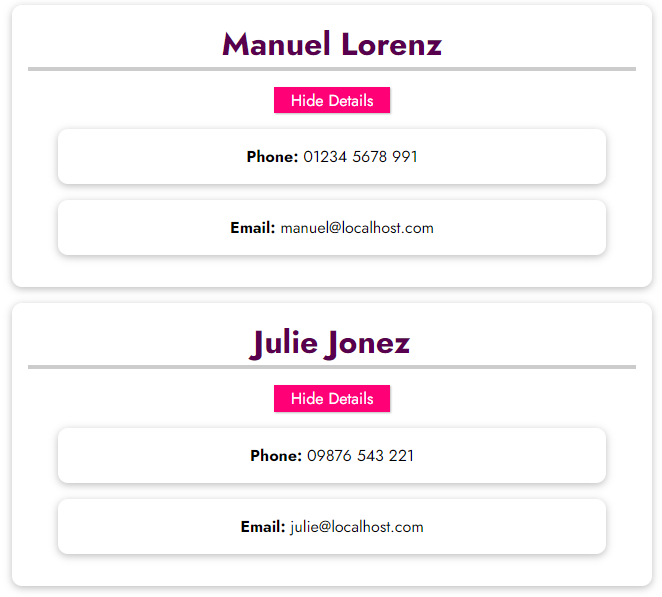
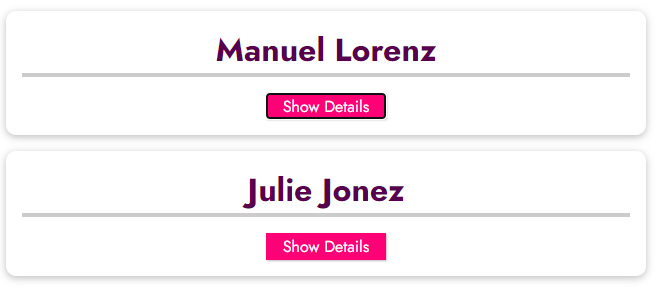
**Section 6 – Introducing Components**

Chapter 70 – Introducing Components

1. The problem if we make a list without components will show below. When we want to hide or show, all the data will be hidden or shown.

1. Building components, component only can be written in the tag of component, outside it, the result will not be printed.

app.component('friend-contact', {

    template: `

        <li>

          <h2>{{ friend.name }}</h2>

          <button @click="toggleDetails(friend.id)">

              {{ detailsAreVisible ? 'Hide' : 'Show' }} Details

          </button>

          <ul v-if="detailsAreVisible">

            <li><strong>Phone:</strong> {{ friend.phone }}</li>

            <li><strong>Email:</strong> {{ friend.email }}</li>

          </ul>

        </li>

    `,

    data() {

        return {

            detailsAreVisible: false,

            friend: {

                    id: 'manuel',

                    name: 'Manuel Lorenz',

                    phone: '01234 5678 991',

                    email: 'manuel@localhost.com'

            }

        };

    },

    methods: {

        toggleDetails() {

            this.detailsAreVisible = !this.detailsAreVisible;

        }

    }

});

app.mount('#app');

The HTML code will be like below.

<!-- Will not be printed -->

<friend-contact></friend-contact>

    <section id="app">

      <ul>

        <!-- Will be printed 2 times -->

        <friend-contact></friend-contact>

        <friend-contact></friend-contact>

      </ul>

</section>

Chapter 71 – The Why: Building Complex User Interfaces With Components

1. To encapsulate different templates and logic tied to that templates into reusable building blocks, you will see how that makes it easier to structure the code phase of complex apps and keep it manageable by splitting your code, your logic, and all of your templates and what’s on the screen into smaller reusable pieces.

Chapter 72 – Multiple Vue Apps VS Multiple Components

* You might recall lecture 3 ("Different Ways of Using Vue"): You can use Vue.js to control parts of (possibly multiple HTML) pages OR you use it to build so-called "Single Page Applications" (SPAs).
* If you control multiple, independent parts of HTML pages, you will often work with multiple Vue apps (i.e. you create multiple apps by calling createApp() more than once).
* On the other hand, if you are building a SPA, you typically work with just one "root app" (i.e. createApp() is only used once in your entire codebase) and you instead build up a user interface with multiple components.
* You absolutely are allowed to also use components in cases where you have multiple Vue apps, but you typically will not use multiple Vue apps if you build one big, connected user interface.
* Why?

Because Vue apps are independent from each other - they cannot really communicate with each other. You might find "hacks" to make it work but there is no great "official" way of sharing data between apps, updating something in app A in case something happens in app B etc.

* Components on the other hand - as you will learn soon - DO offer certain communication mechanisms that allow you to exchange data between them. Hence you can build one connected UI if you work with one root app that holds multiple components.
* You will see that in the lectures and throughout the entire course, especially in the course projects of course!