**Section 9 – Diving Deeper Into Components**

Chapter 107 – Global VS Local Components

1. Global components will be like below. It means Vue need to load all the components which written on main.js

app.component('the-header', TheHeader);

app.component('base-badge', BaseBadge);

app.component('badge-list', BadgeList);

app.component('user-info', UserInfo);

1. Therefore, to reduce the complexity, we can remove some components and input it inside another components, such as TheHeader can be removed and import it from App.vue

import TheHeader from './components/TheHeader.vue';

Add components in App.vue

export default {

  components: {

    'the-header': TheHeader

    // TheHeader: TheHeader

    // TheHeader

  },

  data() {

    return {

      activeUser: {

        name: 'Maximilian Schwarzmüller',

        description: 'Site owner and admin',

        role: 'admin',

      },

    };

  },

};

Chapter 108 – Scoped Styles

1. The problem in styling, maybe our style in parent part can be destroy styling in children part. Therefore, we can modify some styling in parent part. We also can move BadgeList and UserInfo into App.vue by adding scoped in style tag line.

<style scoped>

  header {

    width: 100%;

    height: 5rem;

    display: flex;

    justify-content: center;

  }

</style>

Chapter 109 – Introducing Slots

1. Slot allows HTML content which also may be using Vue features from outside of the component. Props are meant to be used for data, which components needs, and slots are meant to be used for HTML code.

BaseCard.vue

<template>

    <div>

        <slot></slot>

    </div>

</template>

UserInfo.vue

<template>

  <section>

    <base-card>

      <header>

        <h3>{{ fullName }}</h3>

        <base-badge :type="role" :caption="role.toUpperCase()"></base-badge>

      </header>

      <p>{{ infoText }}</p>

    </base-card>

  </section>

</template>

Chapter 110 – Named Slots

1. How if we need 2 slots, one for header and one for main content ?

UserInfo.vue

<template>

  <section>

    <base-card>

      <template v-slot:header>

        <h3>{{ fullName }}</h3>

        <base-badge :type="role" :caption="role.toUpperCase()"></base-badge>

      </template>

      <p>{{ infoText }}</p>

    </base-card>

  </section>

</template>

BaseCard.vue

<template>

    <div>

        <header>

            <slot name="header"></slot>

        </header>

        <slot></slot>

    </div>

</template>

Chapter 112 – More On Slots

1. Give slot default if there is no value or data on the slot, it will print the default data. In this example below, The Default will be printed if we delete Available Badgets.

BaseCard.vue

<header>

<slot name="header">

<h2>The Default</h2>

</slot>

</header>

BadgeList.vue

<template v-slot:header>

<h2>Available Badges</h2>

</template>

1. To check how many slots in the Vue file

export default {

    mounted() {

console.logs(this.$slots);

        console.logs(this.$slots.header);

console.logs(this.$slots.default);

    }

}

When print $slots.header it print 2 times, because on BaseCard.vue there is a default header, therefore we can add if on header tag line like below (in BaseCard.vue)

<header v-if="$slots.header">

1. Change v-slot=”” into #... like below

<template v-slot:header>

<template #header>

Chapter 113 – Scoped Slots

1. Scoped slots letting you pass data from inside the component where you defined a slot to the component where you pass the markup for the slot.
2. Sending data in for from child to parent using slot like below.

CourseGoal.vue

<template>

    <ul>

        <li v-for="goal in goals" :key="goal">

            <slot :item="goal" another-prop="..."></slot>

        </li>

    </ul>

</template>

App.vue

<course-goals>

      <template #default="slotProps">

        <h2>{{ slotProps.item }}</h2>

        <p>{{ slotProps['another-prop'] }}</p>

      </template>

</course-goals>

Or if you only have 1 slot in your child, you may write like below.

<course-goals #default="slotProps">

        <h2>{{ slotProps.item }}</h2>

        <p>{{ slotProps['another-prop'] }}</p>

</course-goals>

Chapter 114 – Dynamic Components

1. Can we write this v-if only once instead of write it twice ?

<active-goals v-if="selectedComponent === 'active-goals'"></active-goals>

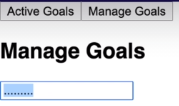
<manage-goals v-if="selectedComponent === 'manage-goals'"></manage-goals>

Change it like below.

<component :is="selectedComponent"></component>

Chapter 115 – Keeping Dynamic Components Alive

1. The problem is if we have two components, such as active goals and manage goals, and in manage goals we have 1 input type text, we input some data, and change into component active goals, and when back to manage goals, the inputted data will be lost. Can we save this data event we change into another components ?



<keep-alive> allows you to tell Vue that components should not be removed and deleted entirely

<keep-alive>

      <component :is="selectedComponent"></component>

</keep-alive>

Chapter 116 – Applying What We Know And A Problem

1. Alert in default has bad style, therefore we can make a new component for alert and pop it up when we need it. Open means the visibility is true.

ErrorAlert.vue

<template>

    <dialog open>

        <slot></slot>

    </dialog>

</template>

ManageGoals.vue

import ErrorAlert from './ErrorAlert.vue';

And the template will be like below.

<template>

    <div>

        <h2>Manage Goals</h2>

        <input type="text" ref="goal"/>

        <button @click="setGoal">Set Goal</button>

        <error-alert v-if="inputIsInvalid">

            <h2>Input is invalid !</h2>

            <p> Please enter at least a few characters... </p>

        </error-alert>

   </div>

</template>

Chapter 117 – Teleporting Elements

1. Teleport wants one prop, one attribute, CSS, selecting an HTML element onto our entire page.

<teleport to="body">

<error-alert v-if="inputIsInvalid">

<h2>Input is invalid !</h2>

<p> Please enter at least a few characters... </p>

<button @click="confirmError">Okay</button>

</error-alert>

</teleport>

Chapter 118 – Working With Fragments

1. Fragments in Vue3 permit you to remove the most outer element inside template, such as <div></div>. In Vue2 before we need at least one tag html which is outside another tag html.

Chapter 119 – The Vue Style Guide

1. There are many style that we can write for our Vue file, but Vue give some recommendation how you decide the name of the Vue file on their website, such as single instance component names can be use The… .vue and Base component names can be use … Base.vue.

Chapter 120 – Moving To A Different Folder Structure

1. In bigger project, you need to reorganize all your file into some sub folder, such as UI, Layout, etc

Chapter 121 – Module Summary

