**Section 31 – Advanced Component Usage**

Chapter 447 – Chapter 460

1. We can pass our content such as HTML code using slots from one file to another file.

App.vue

<template>

    <div class="container">

        <div class="row">

            <div class="col-xs-12">

               <app-quote>

                    <h2>The Quote</h2>

                    <p>A Wonderful Quote</p>

                </app-quote>

            </div>

        </div>

    </div>

</template>

<script>

import Queue from './components/Quote.vue';

export default {

    components: {

        appQuote: Quote

    }

}

</script>

Quote.vue

<template>

    <div>

        <slot></slot>

    </div>

</template>

<script>

export default {

}

</script>

<style scoped>

div {

    border: 1px solid #ccc;

    box-shadow: 1px 1px 2px;

    padding: 30px;

    margin: auto;

    text-align: center;

}

</style>

1. We also give some style to inputted slot (the tag html from another file, but the css is on the slot file). We need to put the style for slot in slot file (child file). If we put the style on the parent file, the slot (child file) will not get the style properties. But now, you can send the style for child from the parent component.

Quote.vue

<template>

    <div>

        <slot></slot>

    </div>

</template>

<script>

export default {

}

</script>

<style scoped>

h2 {

    color: red;

}

</style>

1. But if we put some data on the parent file, the data will send into the child file, like below.

<template>

    <div class="container">

        <div class="row">

            <div class="col-xs-12">

               <app-quote>

                    <h2>{{ quoteTitle }}</h2>

                    <p>A Wonderful Quote</p>

                </app-quote>

            </div>

        </div>

    </div>

</template>

<script>

import Queue from './components/Quote.vue';

export default {

    data: function() {

        return {

            quoteTitle: 'The Quote'

        }

    },

    components: {

        appQuote: Quote

    }

}

</script>

<style scoped>

h2 {

    color: red;

}

</style>

1. We also can divide the slot become two, such as title and component like below.

App.vue

<template>

    <div class="container">

        <div class="row">

            <div class="col-xs-12">

               <app-quote>

                    <h2 slot="title">{{ quoteTitle }}</h2>

                    <p slot="content">A Wonderful Quote</p>

                </app-quote>

            </div>

        </div>

    </div>

</template>

Quote.vue

<template>

    <div>

        <div class="title">

            <slot name="title"></slot>

        </div>

        <hr>

        <div>

            <slot name="content"></slot>

        </div>

    </div>

</template>

1. If some slot didn’t have name, it means it will render on the default slot like below.

App.vue

<template>

    <div class="container">

        <div class="row">

            <div class="col-xs-12">

               <app-quote>

                    <h2 slot="title">{{ quoteTitle }}</h2>

                    <p>A Wonderful Quote</p>

                </app-quote>

            </div>

        </div>

    </div>

</template>

Quote.vue

<template>

    <div>

        <div class="title">

            <slot name="title"></slot>

        </div>

        <hr>

        <div>

            <slot></slot>

        </div>

    </div>

</template>

1. We can add some slot for subtitle like below.

App.vue

<template>

    <div class="container">

        <div class="row">

            <div class="col-xs-12">

               <app-quote>

                    <h2 slot="title">{{ quoteTitle }}</h2>

                    <p>A Wonderful Quote</p>

                </app-quote>

            </div>

        </div>

    </div>

</template>

Quote.vue

<template>

    <div>

        <div class="title">

            <slot name="title"></slot>

            <span style="color: #ccc"><slot name="subtitle">The Subtitle</slot></span>

        </div>

        <hr>

        <div>

            <slot></slot>

        </div>

    </div>

</template>

1. We can make dynamic component like below. This way will create a component when some component is accessed and will destroy the previous component and recreate a new component if we change into another component.

App.vue

<template>

    <div class="container">

        <div class="row">

            <div class="col-xs-12">

                <button @click="selectedComponent = 'appQuote'">Quote</button>

                <button @click="selectedComponent = 'appAuthor'">Author</button>

                <button @click="selectedComponent = 'appNew'">New Quote</button>

                <hr>

                <p>{{ selectedComponent }}</p>

                <component :is="selectedComponent">

                    <!-- This code below will be the defult content,

                    if the component children dont have content -->

                    <p>Default Content</p>

                </component>

                <!-- <app-quote>

                    <h2 slot="title">{{ quoteTitle }}</h2>

                    <p>A Wonderful Quote</p>

                </app-quote> -->

                <hr>

                <app-author></app-author>

            </div>

        </div>

    </div>

</template>

<script>

import Queue from './components/Quote.vue';

import Author from './components/Quote.vue';

import NewQuote from './components/Quote.vue';

export default {

    data: function() {

        return {

            quoteTitle: 'The Quote',

            selectedComponent: 'appQuote'

        }

    },

    components: {

        appQuote: Quote,

        appAuthor: Author,

        appNew: NewQuote

    }

}

</script>

Author.vue

<template>

    <div>

        <h3>The Author</h3>

    </div>

</template>

New.vue

<template>

    <div>

        <h3>New Quote</h3>

    </div>

</template>

Quote.vue

<template>

    <div>

        <div class="title">

            <slot name="title"></slot>

            <span style="color: #ccc"><slot name="subtitle">The Subtitle</slot></span>

        </div>

        <hr>

        <div>

            <slot></slot>

        </div>

    </div>

</template>

To check the component has been destroyed or not.

<template>

    <div>

        <h3>New Quote</h3>

        <button @click="counter++">Increase !</button>

        <p>{{ counter }}</p>

    </div>

</template>

<script>

export default {

    data: function() {

        return {

            counter: 0

        };

    },

    destroyed() {

        console.log('Destroyed !');

    }

}

</script>

1. We can add tag html <keep-alive></keep-alive> so the component will not be destroyed.

App.vue

<template>

    <div class="container">

        <div class="row">

            <div class="col-xs-12">

                <button @click="selectedComponent = 'appQuote'">Quote</button>

                <button @click="selectedComponent = 'appAuthor'">Author</button>

                <button @click="selectedComponent = 'appNew'">New Quote</button>

                <hr>

                <p>{{ selectedComponent }}</p>

                <keep-alive>

                    <component :is="selectedComponent">

                    <!-- This code below will be the defult content,

                    if the component children dont have content -->

                    <p>Default Content</p>

                    </component>

                </keep-alive>

                <!-- <app-quote>

                    <h2 slot="title">{{ quoteTitle }}</h2>

                    <p>A Wonderful Quote</p>

                </app-quote> -->

                <hr>

                <app-author></app-author>

            </div>

        </div>

    </div>

</template>