**Section 30 – Communication Between Components**

Chapter 432 – Chapter 442

1. Printing Name in the different component

User.vue

<template>

    <div class="component">

        <h1>The User Component</h1>

        <p>I'm an awesome User!</p>

        <button @click="changeName">Change My Name</button>

        <hr>

        <div class="row">

            <div class="col-xs-12 col-sm-6">

                <app-user-detail name="Daiva"></app-user-detail>

            </div>

            <div class="col-xs-12 col-sm-6">

                <app-user-edit></app-user-edit>

            </div>

        </div>

    </div>

</template>

<script>

    import UserDetail from './UserDetail.vue';

    import UserEdit from './UserEdit.vue';

    export default {

        data: function() {

            return {

                name: 'Daiva'

            };

        },

        methods: {

            changeName() {

                this.name = 'Misael';

            }

        },

        components: {

            appUserDetail: UserDetail,

            appUserEdit: UserEdit

        }

    }

</script>

UserDetail.vue

<template>

    <div class="component">

        <h3>You may view the User Details here</h3>

        <p>Many Details</p>

        <p>User Name: {{ name }}</p>

    </div>

</template>

<script>

export default {

    props: ['name']

}

</script>

<style scoped>

    div {

        background-color: lightcoral;

    }

</style>

If we want to print dynamically, we can use v-bind in User.vue.

User.vue

<template>

    <div class="component">

        <h1>The User Component</h1>

        <p>I'm an awesome User!</p>

        <button @click="changeName">Change My Name</button>

        <hr>

        <div class="row">

            <div class="col-xs-12 col-sm-6">

                <app-user-detail :name="name"></app-user-detail>

            </div>

            <div class="col-xs-12 col-sm-6">

                <app-user-edit></app-user-edit>

            </div>

        </div>

    </div>

</template>

1. Props can be access like we access the data.

<template>

    <div class="component">

        <h3>You may view the User Details here</h3>

        <p>Many Details</p>

        <!-- <p>User Name: {{ myName }}</p> -->

        <p>User Name: {{ switchName() }}</p>

    </div>

</template>

<script>

export default {

    props: ['myName'],

    methods: {

        switchName() {

            return this.myName.split("").reverse().join("");

        }

    }

}

</script>

1. Making validation when send parameter from 1 component into another component like below. Validate props like below.

<script>

export default {

    // props: ['myName'],

    props: {

        myName: {

            type: String,

            default: 'Daiva',

            required: true

        }

    },

    methods: {

        switchName() {

            return this.myName.split("").reverse().join("");

        }

    }

}

</script>

1. If we want to change value of variable such as myName in child (UserDetail.vue), but the value of variable will not change such as name in parent (User.vue). Therefore, we need to do like this code below.

UserDetail.vue

<script>

export default {

    // props: ['myName'],

    props: {

        myName: {

            type: String,

            default: 'Daiva',

            required: true

        }

    },

    methods: {

        switchName() {

            return this.myName.split("").reverse().join("");

        },

        resetName() {

            this.myName = 'Daiva';

            this.$emit('nameWasReset', this.myName);

        }

    }

}

</script>

User.vue

<template>

    <div class="component">

        <h1>The User Component</h1>

        <p>I'm an awesome User!</p>

        <button @click="changeName">Change My Name</button>

        <p>Name is {{ name }}</p>

        <hr>

        <div class="row">

            <div class="col-xs-12 col-sm-6">

                <app-user-detail :myName="name" @nameWasReset="name = $event"> </app-user-detail>

            </div>

            <div class="col-xs-12 col-sm-6">

                <app-user-edit></app-user-edit>

            </div>

        </div>

    </div>

</template>

If **we don’t emit** and do @ on the parent, when we print the name first time ‘Daiva’, when we change name from parent, it become ‘Misael’. When we reset the name from child, it become ‘Daiva’, but the name in parent still ‘Misael’. Then, if I click change name in parent, the name still ‘Daiva’ in child and the name still ‘Misael’ in parent.

1. Parent and Child

Our connection is from parent to child, can not from child to child. Such as, from parent to child 1, child 1 back to parent, then parent to child 3.



1. We can pass parameter (data) or function from parent to child or back from child to parent using 2 methods.
2. Props and passing a callback as a prop like above (number 4).
3. Props and custom event like below.

User.vue

<template>

    <div class="component">

        <h1>The User Component</h1>

        <p>I'm an awesome User!</p>

        <button @click="changeName">Change My Name</button>

        <p>Name is {{ name }}</p>

        <hr>

        <div class="row">

            <div class="col-xs-12 col-sm-6">

                <app-user-detail

                    :myName="name"

                    @nameWasReset="name = $event"

                    :resetFn="resetName"></app-user-detail>

            </div>

            <div class="col-xs-12 col-sm-6">

                <app-user-edit></app-user-edit>

            </div>

        </div>

    </div>

</template>

<script>

    import UserDetail from './UserDetail.vue';

    import UserEdit from './UserEdit.vue';

    export default {

        data: function() {

            return {

                name: 'Daiva'

            };

        },

        methods: {

            changeName() {

                this.name = 'Misael';

            },

            resetName() {

                this.name = 'Daiva';

            }

        },

        components: {

            appUserDetail: UserDetail,

            appUserEdit: UserEdit

        }

    }

</script>

UserDetail.vue

<template>

    <div class="component">

        <h3>You may view the User Details here</h3>

        <p>Many Details</p>

        <!-- <p>User Name: {{ myName }}</p> -->

        <p>User Name: {{ switchName() }}</p>

        <button @click="resetName">Reset Name</button>

        <button @click="resetFn()">Reset Name</button>

    </div>

</template>

<script>

export default {

    // props: ['myName'],

    props: {

        myName: {

            type: String,

            default: 'Daiva',

            required: true

        },

        resetFn: Function

    },

    methods: {

        switchName() {

            return this.myName.split("").reverse().join("");

        },

        resetName() {

            this.myName = 'Daiva';

            this.$emit('nameWasReset', this.myName);

        }

    }

}

</script>

1. Move data from one child to another child.
2. Method 1

Parent age 27, children 2 age 30. If we run the program, user age will be 27, but after we edit age from the child 2, the age will be 30, but the age in child 1 still 27. Therefore, we can pass the age by using $emit. The age in child 1 and child 2 will be 27 after changing.

User.vue

<template>

    <div class="component">

        <h1>The User Component</h1>

        <p>I'm an awesome User!</p>

        <button @click="changeName">Change My Name</button>

        <p>Name is {{ name }}</p>

        <hr>

        <div class="row">

            <div class="col-xs-12 col-sm-6">

                <app-user-detail

                    :myName="name"

                    @nameWasReset="name = $event"

                    :resetFn="resetName"

                    :userAge="age"></app-user-detail>

            </div>

            <div class="col-xs-12 col-sm-6">

                <app-user-edit

                    :userAge="age"

                    @ageWasEdited="age = $event"

                    ></app-user-edit>

            </div>

        </div>

    </div>

</template>

<script>

    import UserDetail from './UserDetail.vue';

    import UserEdit from './UserEdit.vue';

    export default {

        data: function() {

            return {

                name: 'Daiva',

                age: 27

            };

        },

        methods: {

            changeName() {

                this.name = 'Misael';

            },

            resetName() {

                this.name = 'Daiva';

            }

        },

        components: {

            appUserDetail: UserDetail,

            appUserEdit: UserEdit

        }

    }

</script>

UserDetail.vue

<template>

    <div class="component">

        <h3>You may view the User Details here</h3>

        <p>Many Details</p>

        <!-- <p>User Name: {{ myName }}</p> -->

        <p>User Name: {{ switchName() }}</p>

        <p>User Age: {{ userAge }}</p>

        <button @click="resetName">Reset Name</button>

        <button @click="resetFn()">Reset Name</button>

    </div>

</template>

<script>

export default {

    // props: ['myName'],

    props: {

        myName: {

            type: String,

            default: 'Daiva',

            required: true

        },

        resetFn: Function,

        userAge: Number

    },

    methods: {

        switchName() {

            return this.myName.split("").reverse().join("");

        },

        resetName() {

            this.myName = 'Daiva';

            this.$emit('nameWasReset', this.myName);

        }

    }

}

</script>

UserEdit.vue

<template>

    <div class="component">

        <h3>You may edit the User here</h3>

        <p>Edit me!</p>

        <p>User Age: {{ userAge }}</p>

        <button @click="editAge">Edit Age</button>

    </div>

</template>

<script>

export default {

    props: ['userAge'],

    methods: {

        editAge() {

            this.userAge = 30;

            this.$emit('ageWasEdited', this.userAge);

        }

    }

}

</script>

1. Method 2 : Using a central class or object to pass the data. This implement below can send a parameter from one child to another child not using or not through the parent.

UserDetail.vue

<script>

import { eventBus } from '../main.js';

export default {

    // props: ['myName'],

    props: {

        myName: {

            type: String,

            default: 'Daiva',

            required: true

        },

        resetFn: Function,

        userAge: Number

    },

    methods: {

        switchName() {

            return this.myName.split("").reverse().join("");

        },

        resetName() {

            this.myName = 'Daiva';

            this.$emit('nameWasReset', this.myName);

        }

    },

    created() {

        eventBus.$on('ageWasEdited', (age) => {

            this.userAge = age;

        });

    }

}

</script>

UserEdit.vue

<script>

import { eventBus } from '../main.js';

export default {

    props: ['userAge'],

    methods: {

        editAge() {

            this.userAge = 30;

            // this.$emit('ageWasEdited', this.userAge);

            eventBus.$emit('ageWasEdited', this.userAge);

        }

    }

}

</script>

1. Store data and method centrally.

Main.js

import Vue from 'vue'

import App from './App.vue'

export const eventBus = new Vue({

  methods: {

    changeAge(age) {

      this.$emit('ageWasEdited', age);

    }

  }

});

new Vue({

  el: '#app',

  render: h => h(App)

})

UserEdit.vue

<script>

import { eventBus } from '../main.js';

export default {

    props: ['userAge'],

    methods: {

        editAge() {

            this.userAge = 30;

            // this.$emit('ageWasEdited', this.userAge);

            // eventBus.$emit('ageWasEdited', this.userAge);

            eventBus.changeAge(this.userAge);

        }

    }

}

</script>