Emit in Vue

The emit function in Vue 3 is used to send custom events from a child component to its parent component. It allows the child component to communicate or notify the parent of actions or changes.

In Vue 3, when using the Composition API with the `<script setup>` syntax, you use defineEmits() to define the custom events that your component will emit.

Child Component - Emitting Events

defineEmits(): This function is used to define the events that this component will emit. In this case, no need to define any specific parameters, but we're declaring the event button-clicked using defineEmits().

handleClick() method: This method gets triggered when the user clicks the button. Inside this method, we emit the event 'button-clicked' using emit().

Event Emission: When the button is clicked, it triggers the custom event button-clicked. This event will be captured by the parent component.

```
<script setup>
// Step 1: Define the 'emit' function
const emit = defineEmits()

// Step 2: Create a method that will trigger the event
const handleClick = () => {
  // Step 3: Emit a custom event 'button-clicked'
  emit('button-clicked')
}
</script>

<template>
  <!-- Step 4: Trigger 'handleClick' method when the button is clicked -->
  <button @click="handleClick">Click Me</button>
</template>
```

Parent Component - Listening for Events

Now, the parent component needs to listen for the emitted event and handle it. Here's how the parent component handles the emitted event.

@button-clicked: In the parent component's template, we use the @event-name syntax to listen for the event button-clicked. When the child emits this event, the parent component will call the method handleButtonClick.

handleButtonClick() method: This method will execute when the event button-clicked is emitted from the child component. In this case, it simply logs a message to the console.

```
<template>
  <!-- Step 1: Listen for the 'button-clicked' event from the child -->
  <ChildComponent @button-clicked="handleButtonClick" />
</template>

<script setup>
// Step 2: Define a method to handle the event
const handleButtonClick = () => {
  console.log('Button was clicked in the child component!')
}
</script>
```

Scenario: Parent Component: Displays the list of tasks and listens for an event from the child component when a new task is added.

Child Component: Allows the user to input a new task and emits an event to the parent with the new task details.

Addtask.Vue

```
<script setup>
import { ref } from 'vue'

// Step 1: Define 'emit' function to send events to parent
const emit = defineEmits()

// Step 2: Create a reactive variable to store the task name and priority
const taskName = ref('')
const taskPriority = ref('low')

// Step 3: Method to emit task data when the user adds a task
const addTask = () => {
  // Step 4: Emit the task data to parent
  if (taskName.value) {
    emit('task-added', {
      name: taskName.value,
      priority: taskPriority.value,
    })

    // Clear input fields after emitting
    taskName.value = ''
    taskPriority.value = 'low'
  }
}
</script>

<template>
  <div>
    <input
      v-model="taskName"
      type="text"
```

```
      placeholder="Enter task name"
    />
    <select v-model="taskPriority">
      <option value="low">Low</option>
      <option value="medium">Medium</option>
      <option value="high">High</option>
    </select>
    <button @click="addTask">Add Task</button>
  </div>
</template>
```

taskList.vue

```
<script setup>
import { ref } from 'vue'
import AddTask from './AddTask.vue'

// Step 1: Create a reactive list of tasks
const tasks = ref([])

// Step 2: Method to handle the 'task-added' event and update the task list
const handleTaskAdded = (task) => {
  tasks.value.push(task)
}
</script>

<template>
  <div>
    <!-- Step 3: Include the AddTask child component and listen for the 'task-
added' event -->
    <AddTask @task-added="handleTaskAdded" />

    <h2>Task List</h2>
    <ul>
      <!-- Step 4: Display the list of tasks -->
      <li v-for="(task, index) in tasks" :key="index">
        {{ task.name }} - {{ task.priority }}
      </li>
    </ul>
  </div>
</template>
```