Vue预习课

核心知识07——vue必会API盘点

数据相关API

Vue.set

向响应式对象中添加一个属性，并确保这个新属性同样是响应式的，且触发视图更新。

使用方法：`Vue.set(target, propertyName/index, value)`

范例：批量设置商品价格

```
<template>
    <!--添加批量价格更新-->
    <p>
      <input v-model.number="price">
      <button @click="batchUpdate">批量更新价格</button>
    </p>

    <div class="course-list" v-else>
        <div v-for="c in courses" :key="c.name">
            <!--添加批量价格更新-->
            {{ c.name }} - ￥{{c.price}}
        </div>
    </div>
</template>
<script>
function getCourses() {
    return new Promise(resolve => {
        setTimeout(() => {
            // 修改返回数据结构为对象
            resolve([{ name: 'web全栈' }, { name: 'web高级' }])
        }, 2000);
    })
}
```

```
const app = new Vue({
    data() {
        return {
            price: 0 // 增加价格数据
        }
    },
    methods: {
        // 添加批量价格更新方法
        batchUpdate() {
            this.courses.forEach(c => {
                // c.price = this.price; // no ok
                Vue.set(c, 'price', this.price); // ok
            })
        }
    },
}
</script>
```

Vue.delete

删除对象的属性。如果对象是响应式的，确保删除能触发更新视图。

使用方法：`Vue.delete(target, propertyName/index)`

事件相关API

vm.$on

监听当前实例上的自定义事件。事件可以由`vm.$emit`触发。回调函数会接收所有传入事件触发函数的额外参数。

```
vm.$on('test', function (msg) {
  console.log(msg)
})
```

vm.$emit

触发当前实例上的事件。附加参数都会传给监听器回调。

```
vm.$emit('test', 'hi')
```

典型应用：事件总线

通过在Vue原型上添加一个Vue实例作为事件总线，实现组件间相互通信，而且不受组件间关系的影响

```
Vue.prototype.$bus = new Vue();
```

> 这样做可以在任意组件中使用`this.$bus`访问到该Vue实例

范例：批量清除多个消息窗口

```
/*重构样式：提取出.success，并添加.warning*/
.message-box {
    padding: 10px 20px;
}
.success {
    background: #4fc08d;
    border: 1px solid #42b983;
}
.warning {
    background: #f66;
    border: 1px solid #d63200;
}
```

```
<!--给之前新增成功消息添加.success-->
<message :show.sync="show" class="success">...</message>
<!--新增警告提示窗-->
<message :show.sync="showWarn" class="warning">
    <template v-slot:title>
        <strong>警告</strong>
    </template>
    <template v-slot:default>
        请输入课程名称！
    </template>
</message>
```

```
const app = new Vue({
    data() {
        return {
            // 控制警告信息显示状态
            showWarn: false,
        }
    },
    methods: {
        addCourse() {
            // 增加输入校验
            if (this.course) {
                // ...
            } else {
                // 提示警告信息
                this.showWarn = true
            }
        }
    },
})
```

下面实现功能：

```
// 弹窗组件
Vue.component('message', {
    // ...
    // 监听关闭事件
    mounted () {
        this.$bus.$on('message-close', () => {
            this.$emit('update:show', false)
        });
    },
})
```

```
<!-- 派发关闭事件 -->
<div class="toolbar">
    <button @click="$bus.$emit('message-close')">清空提示框</button>
</div>
```

vm.$once

监听一个自定义事件，但是只触发一次。一旦触发之后，监听器就会被移除。

```
vm.$on('test', function (msg) {
  console.log(msg)
})
```

vm.$off

移除自定义事件监听器。

- 如果没有提供参数，则移除所有的事件监听器；
- 如果只提供了事件，则移除该事件所有的监听器；
- 如果同时提供了事件与回调，则只移除这个回调的监听器。

```
vm.$off() // 移除所有的事件监听器
vm.$off('test') // 移除该事件所有的监听器
vm.$off('test', callback) // 只移除这个回调的监听器
```

组件或元素引用

ref和vm.$refs

`ref` 被用来给元素或子组件注册引用信息。引用信息将会注册在父组件的 `$refs` 对象上。如果**在普通的 DOM 元素上使用，引用指向的就是 DOM 元素**；如果**用在子组件上，引用就指向组件**

范例：设置输入框焦点

```
<input type="text" ... ref="inp">

mounted(){
    // mounted之后才能访问到inp
    this.$refs.inp.focus()
}
```

范例：改造message组件用打开、关闭方法控制显示

```
<!--自定义组件引用-->
<message ref="msg">新增课程成功! </message>

<script>
Vue.component('message', {
    // 组件显示状态
    data() {
        return {
            show: false
        }
    },
    template: `
        <div class="message-box" v-if="show">
            <slot></slot>
            <!--toggle内部修改显示状态-->
            <span class="message-box-close" @click="toggle">X</span>
        </div>
    `,
    // 增加toggle方法维护toggle状态
    methods: {
        toggle() {
            this.show = !this.show;
        }
    },
    // 修改message-close回调为toggle
    mounted () {
        this.$bus.$on('message-close', this.toggle);
    },
})
const app = new Vue({
    methods: {
        addCourse() {
            // 使用$refs.msg访问自定义组件
            this.$refs.msg.toggle()
        }
    }
})
</script>
```

> 注意：
>
> - ref 是作为渲染结果被创建的，在初始渲染时不能访问它们
> - `$refs` 不是响应式的，不要试图用它在模板中做数据绑定
> - 当 `v-for` 用于元素或组件时，引用信息将是包含 DOM 节点或组件实例的数组。