06-v-model进阶+ref+nextTick

一、v-model进阶

1. 剖析原理

1.1 作用

实现双向数据绑定：

- 数据变了，视图跟着变
- 视图变了，数据跟着变

1.2 原理

v-model本质上是一个语法糖。

例如应用在输入框上，就是 value 属性 和 input 事件的合写

```
<script setup>
    import { ref } from 'vue'
    const msg = ref('')
</script>
<template>
    <p>{{ msg }}</p>
    <input type="text" v-model="msg">
    <input type="text" :value="msg" @input="msg = $event.target.value" >
</template>
```

1.3 注意

$event 出现在模板中, 用来获取事件的形参

1.4 总结

1. v-model的原理是什么?

   答：v-model只是个语法糖，作用在原生输入框上，本质是 :value+@input 的组合

```
<input v-model="msg"/>

<!-- 等同于 -->
<input type="text" :value="msg" @input="msg = $event.target.value" >
```

2. v-model用在组件上

2.1 分析

v-model除了用在表单元素上, 还可以用在组件上, 实现数据的双向绑定

2.2 需求

实现父组件和子组件数据的双向绑定

2.3 代码演示

`BitSelect.vue`

```vue
<script setup>
  // 接收 modelValue
  const props = defineProps({
    modelValue: {
      type: String,
      default: ''
    }
  })

  // 获取 emit
  const emit = defineEmits()
</script>

<template>
    <select
      :value="props.modelValue"
      @change="emit('update:modelValue', $event.target.value)">
      <option value="111">北京</option>
      <option value="222">上海</option>
      <option value="333">深圳</option>
      <option value="444">杭州</option>
      <option value="555">苏州</option>
    </select>
</template>
```

`App.vue`

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

```
  const activeId = ref('222')
</script>

<template>
  <BitSelect v-model="activeId"/>
</template>
```

2.4 简化代码

可以使用 `defineModel()` 进行简化上述代码

`BitSelect.vue`

```
<script setup>
  const model = defineModel()
</script>

<template>
  <select v-model="model">
    <option value="111">北京</option>
    <option value="222">上海</option>
    <option value="333">深圳</option>
    <option value="444">杭州</option>
    <option value="555">苏州</option>
  </select>
</template>
```

2.5 总结

1. 今后在组件上使用 v-model 的规则是?

   答：1、父组件绑定 v-model="数据"

   2、子组件 const model = defineModel() 接收数据，子组件可以直接操作model，即可实现数据双向绑定

二、ref

1. 作用

ref 可以用于 获取原生 DOM 元素 或 组件实例

2. 步骤

1. 声明并绑定 ref
2. 通过 ref.value 获取

3. 获取原生DOM

3.1 静态代码

`components/MyChart.vue`

```vue
<script setup>
    // npm i echarts
    import * as echarts from 'echarts'

    const option = {
      title: {
        text: 'ECharts 入门案例',
      },
      xAxis: {
        data: ['衬衫', '羊毛衫', '雪纺衫', '裤子', '高跟鞋', '袜子'],
      },
      yAxis: {},
      series: [
        {
          name: '销量',
          type: 'bar',
          data: [5, 20, 36, 10, 10, 20],
        }
      ]
    }
</script>

<template>
  <div class="chart-box"></div>
</template>

<style scoped>
.chart-box {
  width: 400px;
  height: 300px;
  margin: 100px auto;
  border: 3px solid #000;
  border-radius: 6px;
}
</style>
```


`App.vue`

```vue
<script setup>
```

```
    import MyChart from './components/MyChart.vue'
</script>

<template>
  <MyChart />
</template>

<style>
</style>
```

3.2 完整代码

`MyChart.vue`

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

    import * as echarts from 'echarts'

    const chartRef = ref(null)

    const option = {
      title: {
        text: 'ECharts 入门案例',
      },
      xAxis: {
        data: ['衬衫', '羊毛衫', '雪纺衫', '裤子', '高跟鞋', '袜子'],
      },
      yAxis: {},
      series: [
        {
          name: '销量',
          type: 'bar',
          data: [5, 20, 36, 10, 10, 20],
        }
      ]
    }

    onMounted(()=> {
      const chart = echarts.init(chartRef.value)
      chart.setOption(option)
    })
</script>

<template>
  <div class="chart-box" ref="chartRef"></div>
```

```
</template>

<style scoped>
.chart-box {
  width: 400px;
  height: 300px;
  margin: 100px auto;
  border: 3px solid #000;
  border-radius: 6px;
}
</style>
```

4. 调用组件方法

4.1 静态代码

`components/MyForm.vue`

```
<script setup></script>
<template>
  <div class="login-box">
    账户: <input type="text" /><br /><br />
    密码: <input type="password" /><br /><br />
  </div>
</template>

```

`App.vue`

```
<script setup>
  import MyForm from './components/MyForm.vue'
</script>

<template>
  <MyForm />

  <button>登 录</button>
</template>

<style>
  #app {
    width: 300px;
    margin: 100px auto;
```

```
  }
</style>
```

4.2 完整代码

`components/MyForm.vue`

```
<script setup>
  // 表单校验
  const validate = () => {
    return Math.random() > 0.5 ? true : false
  }

  // 暴漏给组件，目的是父组件可以通过 ref 可以拿到子组件的方法
  defineExpose({
    validate
  })
</script>
<template>
  <div class="login-box">
    账户: <input type="text" /><br /><br />
    密码: <input type="password" /><br /><br />
  </div>
</template>
```

`App.vue`

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

  const formRef = ref(null)

  // 登录
  const onLogin = () => {
    console.log(formRef.value)

    // 进行校验
    if (formRef.value.validate()) {
      console.log('ok')
    } else {
      console.log('error')
    }
  }
```

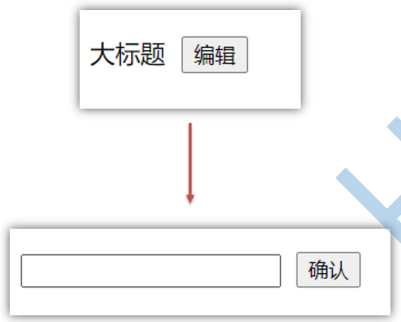
```
</script>

<template>
  <MyForm ref="formRef" />

  <button @click="onLogin">登 录</button>
</template>

<style>
  #app {
    width: 300px;
    margin: 100px auto;
  }
</style>
```

三、nextTick

1. 需求

编辑标题, 编辑框自动聚焦

1. 点击编辑，显示编辑框
2. 让编辑框，立刻获取焦点



2. 代码实现

```
// 显示输入框
isShowEdit.value = true
// 获取焦点
inputRef.value.focus()
```

3. 问题

显示之后，立刻获取焦点是不能成功的！

原因： `Vue 是异步更新DOM` (提升性能)

4. 解决方案

nextTick：**等DOM更新后**, 才会触发执行此方法里的函数体

语法: nextTick(函数体)

```
nextTick(() => {
  inputRef.value.focus()
})
```