- 师资培训课件 （Vue3 题目）
  - 高频考点讲解：
  - 1. 模板语法
  - 2. 基础示例
  - 3. Class 与 Style 绑定
  - 4. 条件渲染
  - 5. 列表渲染
  - 6. Props
  - 7. 组件事件
  - 总结：vue3 题目属于今年新改变考点，题目难度不会特别难，vue 题目 2 -4 道，占比 30 - 40 分，对于今年备战的学生来说，建议把 vue3 的基础部分学好。
  - 备赛建议：vue3 composition 写法一定是 vue3 考试的重点。一定通过 html 直接引入 vue3 方式进行考核。

师资培训课件 （Vue3 题目）

> 难度 : ⭐⭐ 到 ⭐⭐⭐⭐

1. 表单验证器

```
//  TODO: 目标 1 当输入框的值变化时，触发 input 事件更新父组件的 v-model 值
watch(inputValue, (newValue) => {
  emit("update:modelValue", newValue);
});

//  TODO: end

// 目标 2
const is_email = (val) => {
  // TODO: 待补充代码
  return /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/.test(val);
};

// 目标 3
// TODO: 编写通用的表单验证规则 15 分
for (const field in props.rules) {
  const fieldRules = props.rules[field];
  const value = props.formData[field];

  for (const rule of fieldRules) {
    if (typeof rule === "object" && rule.validator) {
      // 处理包含 validator 函数的验证规则
      rule.validator(rule, value, (error) => {
        if (error) {
```

```
            // 如果存在错误，将错误信息添加到 errors 对象中
            errors.value[field] = error.message;
          }
        });
      } else if (rule.required && !value) {
        // 处理预定义验证规则：必填项
        errors.value[field] = rule.message;
      } else if (rule.type) {
        // 根据类型进行验证
        const validationError = validateByType(rule.type, value);
        if (!validationError) {
          if (!errors.value[field]) {
            errors.value[field] = rule.message;
          }
        }
      }
      // 验证字段长度是否在指定范围内
      if (value.length < rule.min || value.length > rule.max) {
        if (!errors.value[field]) {
          errors.value[field] = rule.message;
        }
      }
    }
  }
  // TODO: END
```

API 地址

2. 不翼而飞的余额

- 考点 vue-router、pinia

```
// 目标 1
const router = createRouter({
  history: createWebHistory(),
  routes: [
    { path: "/", component: WalletPage },
    { path: "/deposit", component: DepositPage },
  ],
});
```

```
// store.js
// 存款事件
const depositMoney = (money) => {
  balance.value = Number(balance.value) + Number(money.value);
};
return {
  // ...
  depositMoney,
};
```

```
// DepositPage.js
<span id="deposit-balance">{{ store.balance }}</span>

function deposit() {
      store.depositMoney(depositAmount)
       depositAmount.value = '';
    }

return {
  // ...
  store,
  depositAmount
};
```

高频考点讲解：

以下是使用 Vue 3 Composition API 语法的具体代码示例，以 HTML 形式呈现：

1. 模板语法

```
<div id="app">
  <p>{{ message }}</p>
</div>

<script>
  const { ref } = Vue;

  const app = Vue.createApp({
    setup() {
      const message = ref('Hello Vue!');
      return { message };
    }
  });

  app.mount('#app');
</script>
```

2. 基础示例

```
<div id="app">
  <button @click="increase">{{ count }}</button>
</div>
```

```
<script>
  const { ref } = Vue;

  const app = Vue.createApp({
    setup() {
      const count = ref(0);
      const increase = () => {
        count.value++;
      };
      return { count, increase };
    }
  });

  app.mount('#app');
</script>
```

3. Class 与 Style 绑定

```
<div id="app">
  <div :class="{ active: isActive }"></div>
  <div :style="{ color: textColor, fontSize: textSize + 'px' }"></div>
</div>

<script>
  const { ref } = Vue;

  const app = Vue.createApp({
    setup() {
      const isActive = ref(true);
      const textColor = ref('red');
      const textSize = ref(20);
      return { isActive, textColor, textSize };
    }
  });

  app.mount('#app');
</script>
```

4. 条件渲染

```
<div id="app">
  <p v-if="isDisplayed">This paragraph is displayed.</p>
  <p v-else>This paragraph is not displayed.</p>
</div>

<script>
  const { ref } = Vue;
```

```
  const app = Vue.createApp({
    setup() {
      const isDisplayed = ref(true);
      return { isDisplayed };
    }
  });

  app.mount('#app');
</script>
```

5. 列表渲染

```
<div id="app">
  <ul>
    <li v-for="(item, index) in items" :key="index">{{ item }}</li>
  </ul>
</div>

<script>
  const { ref } = Vue;

  const app = Vue.createApp({
    setup() {
      const items = ref(['Apple', 'Banana', 'Orange']);
      return { items };
    }
  });

  app.mount('#app');
</script>
```

6. Props

```
<div id="app">
  <child-component message="Hello from parent"></child-component>
</div>

<script>
  const { defineComponent } = Vue;

  const ChildComponent = defineComponent({
    props: ['message'],
    template: `<p>{{ message }}</p>`
  });

  const app = Vue.createApp({
```

```
    components: {
      'child-component': ChildComponent
    }
  });

  app.mount('#app');
</script>
```

```
<div id="app">
  <child-component :message="parentMessage"></child-component>
</div>

<script>
  const { defineComponent, ref, PropTypes } = Vue;

  const ChildComponent = defineComponent({
    props: {
      message: {
        type: String,
        required: true,
        validator: (value) => {
          return value.length <= 10; // 自定义验证函数，确保消息长度不超过10个字符
        }
      }
    },
    template: `<p>{{ message }}</p>`
  });

  const app = Vue.createApp({
    setup() {
      const parentMessage = ref('Hello from parent');
      return { parentMessage };
    },
    components: {
      'child-component': ChildComponent
    }
  });

  app.mount('#app');
</script>
```

7. 组件事件

```
<div id="app">
  <child-component @custom-event="handleCustomEvent"></child-component>
  <p>{{ eventData }}</p>
</div>

<script>
  const { defineComponent, ref } = Vue;
```

```
  const ChildComponent = defineComponent({
    template: `<button @click="emitEvent">Click me</button>`,
    setup(_, { emit }) {
      const emitEvent = () => {
        emit('custom-event', 'Data from child component');
      };
      return { emitEvent };
    }
  });

  const app = Vue.createApp({
    setup() {
      const eventData = ref('');
      const handleCustomEvent = (data) => {
        eventData.value = data;
      };
      return { eventData, handleCustomEvent };
    },
    components: {
      'child-component': ChildComponent
    }
  });

  app.mount('#app');
</script>
```

总结：vue3 题目属于今年新改变考点，题目难度不会特别难，vue 题目 2 -4 道，占比 30 - 40 分，对于今年备战的学生来说，建议把 vue3 的基础部分学好。

备赛建议：vue3 composition 写法一定是 vue3 考试的重点。一定通过 html 直接引入 vue3 方式进行考核。
