ДИСЦИПЛИНА	Фронтенд и бэкенд разработка
ИНСТИТУТ	ИПТИП
КАФЕДРА	Индустриального программирования
ВИД УЧЕБНОГО МАТЕРИАЛА	Методические указания к практическим занятиям
ПРЕПОДАВАТЕЛЬ	Астафьев Рустам Уралович
СЕМЕСТР	1 семестр, 2025/2026 уч. год

Ссылка на материал:
https://github.com/astafiev-rustam/frontend-and-backend-development/tree/practice-1-28

Практическое занятие 28: Компоненты, пропсы, v-model и Vue Router

Теоретическая часть

Как и всегда, предварительно необходимо подключить компоненты в `App.vue`. Теоретически, можно дополнить правильным образом импорты и массив компонентов в файле, либо заменить названия, например:

```
<template>
  <div id="app">
    <header class="app-header">
      <h1>🎯 Vue 3 Практика - Основы</h1>
      <p>Изучаем реактивность, директивы и компоненты</p>
    </header>

    <!-- Навигация между примерами -->
    <nav class="navigation">
      <button
        @click="currentDemo = 'reactive'"
        :class="{ active: currentDemo === 'reactive' }"
        class="nav-button"
      >
        Пример 1: Реактивность
      </button>
      <button
        @click="currentDemo = 'conditional'"
        :class="{ active: currentDemo === 'conditional' }"
        class="nav-button"
      >
        Пример 2: Списки и условия
      </button>
      <button
        @click="currentDemo = 'events'"
```

```
        :class="{ active: currentDemo === 'events' }"
        class="nav-button"
      >
        Пример 3: События
      </button>
      <button
        @click="currentDemo = 'usercards'"
        :class="{ active: currentDemo === 'usercards' }"
        class="nav-button"
      >
        Пример 4: Компоненты и пропсы
      </button>
      <button
        @click="currentDemo = 'searchexample'"
        :class="{ active: currentDemo === 'searchexample' }"
        class="nav-button"
      >
        Пример 5: Поиск и кастомные элементы
      </button>
      <button
        @click="currentDemo = 'example6'"
        :class="{ active: currentDemo === 'example6' }"
        class="nav-button"
      >
        Пример 6
      </button>
    </nav>

    <!-- Отображаем выбранный компонент -->
    <main class="main-content">
      <!-- Компонент ReactiveDemo -->
      <ReactiveDemo v-if="currentDemo === 'reactive'" />

      <!-- Компонент ConditionalListDemo -->
      <ConditionalListDemo v-else-if="currentDemo === 'conditional'" />

      <!-- Компонент EventComputedDemo -->
      <EventComputedDemo v-else-if="currentDemo === 'events'" />

      <!-- Компонент UserCards -->
      <UserCards v-else-if="currentDemo === 'usercards'" />

      <!-- Компонент SearchExample -->
      <SearchExample v-else-if="currentDemo === 'searchexample'" />

      <!-- Компонент EventComputedDemo -->
      <EventComputedDemo v-else-if="currentDemo === 'events'" />

      <!-- Сообщение если ничего не выбрано -->
      <div v-else class="welcome-message">
        <h2>Добро пожаловать!</h2>
        <p>Выберите пример для изучения из навигации выше.</p>
      </div>
    </main>
```

```
    <footer class="app-footer">
      <p>Vue 3 + Vite • Практика 27</p>
    </footer>
  </div>
</template>

<script>
// Импортируем наши компоненты, пока файлов нет - должно быть закомментировано
import ReactiveDemo from './components/ReactiveDemo.vue'
import ConditionalListDemo from './components/ConditionalListDemo.vue'
import EventComputedDemo from './components/EventComputedDemo.vue'
import UserCards from './components/UserCards.vue'
import SearchExample from './components/SearchExample.vue'

import { ref } from 'vue'

export default {
  name: 'App',

  // Регистрируем компоненты чтобы использовать их в шаблоне
  components: {
    ReactiveDemo,
    ConditionalListDemo,
    EventComputedDemo,
    UserCards,
    SearchExample
  },

  setup() {
    // Текущий активный демо-компонент
    const currentDemo = ref('reactive')

    return {
      currentDemo
    }
  }
}
</script>

<style>
/* Глобальные стили */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #f5f5f5;
  color: #333;
  line-height: 1.6;
}
```

```
#app {
  min-height: 100vh;
  display: flex;
  flex-direction: column;
}

/* Стили шапки */
.app-header {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  padding: 2rem;
  text-align: center;
}

.app-header h1 {
  margin-bottom: 0.5rem;
  font-size: 2.5rem;
}

.app-header p {
  opacity: 0.9;
  font-size: 1.1rem;
}

/* Навигация */
.navigation {
  display: flex;
  justify-content: center;
  gap: 1rem;
  padding: 1.5rem;
  background-color: white;
  box-shadow: 0 2px 10px rgba(0,0,0,0.1);
  flex-wrap: wrap;
}

.nav-button {
  padding: 0.75rem 1.5rem;
  border: 2px solid #667eea;
  background-color: transparent;
  color: #667eea;
  border-radius: 25px;
  cursor: pointer;
  font-size: 1rem;
  transition: all 0.3s ease;
}

.nav-button:hover {
  background-color: #667eea;
  color: white;
  transform: translateY(-2px);
}

.nav-button.active {
```

```
  background-color: #667eea;
  color: white;
}

/* Основное содержимое */
.main-content {
  flex: 1;
  padding: 2rem;
  max-width: 1200px;
  margin: 0 auto;
  width: 100%;
}

.welcome-message {
  text-align: center;
  padding: 4rem 2rem;
  color: #666;
}

.welcome-message h2 {
  margin-bottom: 1rem;
  color: #333;
}

/* Подвал */
.app-footer {
  background-color: #333;
  color: white;
  text-align: center;
  padding: 1rem;
  margin-top: auto;
}
</style>
```

Пример 1: Компоненты и пропсы

Файл: `src/components/UserCard.vue`

```
<template>
  <div class="user-card" :class="user.role">
    <h3>{{ user.name }}</h3>
    <p>Email: {{ user.email }}</p>
    <p>Роль: {{ user.role }}</p>
    <p>Статус: {{ isActive ? 'Активен' : 'Неактивен' }}</p>

    <!-- Слот для дополнительного контента -->
    <slot name="actions"></slot>

    <!-- Слот по умолчанию -->
    <slot>
      <p>Нет дополнительной информации</p>
```

```
    </slot>
  </div>
</template>

<script>
export default {
  name: 'UserCard',

  // Определяем пропсы, которые компонент принимает
  props: {
    user: {
      type: Object,
      required: true,
      // Валидация объекта
      validator: (value) => {
        return value.name && value.email
      }
    },
    isActive: {
      type: Boolean,
      default: false
    }
  },

  // Локальное состояние компонента
  data() {
    return {
      localClicks: 0
    }
  },

  methods: {
    handleClick() {
      this.localClicks++
      // Отправляем событие родителю
      this.$emit('user-clicked', this.user)
    }
  }
}
</script>

<style scoped>
.user-card {
  border: 1px solid #ddd;
  padding: 16px;
  margin: 10px;
  border-radius: 8px;
}

.user-card.admin {
  border-color: #ff6b6b;
  background-color: #fff5f5;
}
```

```
.user-card.user {
  border-color: #4ecdc4;
  background-color: #f0fff4;
}
</style>
```

Использование в родительском компоненте (например, `UserCards.vue`):

```
<template>
  <div>
    <h2>Список пользователей</h2>

    <!-- Передаем данные через пропсы -->
    <UserCard
      :user="adminUser"
      :is-active="true"
      @user-clicked="handleUserClick"
    >
      <!-- Именованный слот -->
      <template #actions>
        <button @click="editUser(adminUser)">Редактировать</button>
      </template>

      <!-- Слот по умолчанию -->
      <p>Администратор системы</p>
    </UserCard>

    <UserCard
      v-for="user in users"
      :key="user.id"
      :user="user"
      @user-clicked="handleUserClick"
    />
  </div>
</template>

<script>
import UserCard from './UserCard.vue'

export default {
  components: {
    UserCard
  },

  data() {
    return {
      adminUser: {
        id: 1,
        name: 'Анна Иванова',
        email: 'anna@example.com',
        role: 'admin'
      },
```

```
      users: [
        {
          id: 2,
          name: 'Петр Сидоров',
          email: 'petr@example.com',
          role: 'user'
        }
      ]
    }
  },

  methods: {
    handleUserClick(user) {
      console.log('Клик по пользователю:', user)
    },

    editUser(user) {
      console.log('Редактирование:', user)
    }
  }
}
</script>
```

Пример 2: Кастомный v-model

Файл: `src/components/SearchInput.vue`

```
<template>
  <div class="search-container">
    <label v-if="label">{{ label }}</label>
    <input
      :value="modelValue"
      @input="$emit('update:modelValue', $event.target.value)"
      :placeholder="placeholder"
      class="search-input"
      type="text"
    />
    <button
      v-if="modelValue"
      @click="$emit('update:modelValue', '')"
      class="clear-btn"
    >
      ×
    </button>

    <!-- Дополнительные события -->
    <div class="search-actions">
      <button @click="$emit('search')">Найти</button>
      <button @click="$emit('reset')">Сбросить</button>
    </div>
  </div>
```

```
</template>

<script>
export default {
  name: 'SearchInput',

  // Специальные пропсы для v-model
  props: {
    modelValue: {
      type: String,
      default: ''
    },
    label: {
      type: String,
      default: ''
    },
    placeholder: {
      type: String,
      default: 'Поиск...'
    }
  },

  // Можно определить эмитируемые события
  emits: ['update:modelValue', 'search', 'reset']
}
</script>

<style scoped>
.search-container {
  position: relative;
  margin: 20px 0;
}

.search-input {
  width: 100%;
  padding: 10px 40px 10px 15px;
  border: 1px solid #ddd;
  border-radius: 5px;
  font-size: 16px;
}

.clear-btn {
  position: absolute;
  right: 10px;
  top: 50%;
  transform: translateY(-50%);
  background: none;
  border: none;
  font-size: 20px;
  cursor: pointer;
}

.search-actions {
  margin-top: 10px;
```

```
  display: flex;
  gap: 10px;
}
</style>
```

Использование кастомного v-model, например, `SearchExample.vue`:

```
<template>
  <div>
    <h2>Поиск пользователей</h2>

    <!-- Используем кастомный v-model -->
    <SearchInput
      v-model="searchQuery"
      label="Поиск по имени:"
      placeholder="Введите имя пользователя..."
      @search="performSearch"
      @reset="resetSearch"
    />

    <p>Текущий запрос: "{{ searchQuery }}"</p>

    <div v-if="searchResults.length">
      <h3>Результаты поиска:</h3>
      <ul>
        <li v-for="user in searchResults" :key="user.id">
          {{ user.name }} - {{ user.email }}
        </li>
      </ul>
    </div>
  </div>
</template>

<script>
import SearchInput from './SearchInput.vue'
import { ref, computed } from 'vue'

export default {
  components: {
    SearchInput
  },

  setup() {
    const searchQuery = ref('')
    const users = ref([
      { id: 1, name: 'Анна', email: 'anna@test.com' },
      { id: 2, name: 'Борис', email: 'boris@test.com' },
      { id: 3, name: 'Виктор', email: 'victor@test.com' }
    ])

    const searchResults = computed(() => {
      if (!searchQuery.value) return []
```

```
      return users.value.filter(user =>
        user.name.toLowerCase().includes(searchQuery.value.toLowerCase())
      )
    })

    const performSearch = () => {
      console.log('Выполняем поиск:', searchQuery.value)
    }

    const resetSearch = () => {
      searchQuery.value = ''
    }

    return {
      searchQuery,
      searchResults,
      performSearch,
      resetSearch
    }
  }
}
</script>
```

Пример 3: Vue Router - маршрутизация

Для этого примера создадим новое приложение и реализуем функционал в нём.

Понял! Давай исправим третий пример с Vue Router. Проблема в том, что нужно правильно настроить структуру и добавить все необходимые компоненты.

Пример 3: Работа с Vue Router

Шаг 1: Устанавливаем Vue Router

```
npm install vue-router@4
```

Шаг 2: Создаем структуру папок

```
src/
├── views/
│   ├── HomePage.vue
│   ├── UserProfile.vue
│   ├── UserSettings.vue
│   └── NotFound.vue
├── router/
│   └── index.js
└── main.js
```

Шаг 3: Создаем компоненты страниц

`src/views/HomePage.vue`

```
<template>
  <div class="home-page">
    <h1>🏠 Главная страница</h1>
    <p>Добро пожаловать в наше приложение!</p>

    <div class="quick-actions">
      <button @click="goToProfile" class="btn">Перейти в профиль</button>
      <button @click="goToSettings" class="btn">Настройки</button>
    </div>

    <div class="user-list">
      <h3>Список пользователей:</h3>
      <ul>
        <li
          v-for="user in users"
          :key="user.id"
          @click="viewUser(user.id)"
          class="user-item"
        >
          {{ user.name }}
        </li>
      </ul>
    </div>
  </div>
</template>

<script>
import { useRouter } from 'vue-router'

export default {
  name: 'HomePage',

  setup() {
    const router = useRouter()

    const users = [
      { id: 1, name: 'Анна Иванова' },
      { id: 2, name: 'Петр Сидоров' },
      { id: 3, name: 'Мария Петрова' }
    ]

    const goToProfile = () => {
      router.push('/profile')
    }

    const goToSettings = () => {
      router.push('/settings')
    }
```

```
    const viewUser = (userId) => {
      router.push(`/profile/${userId}`)
    }

    return {
      users,
      goToProfile,
      goToSettings,
      viewUser
    }
  }
}
</script>

<style scoped>
.home-page {
  max-width: 800px;
  margin: 0 auto;
}

.quick-actions {
  margin: 20px 0;
  display: flex;
  gap: 10px;
}

.btn {
  padding: 10px 20px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

.user-list {
  margin-top: 30px;
}

.user-item {
  padding: 10px;
  border: 1px solid #ddd;
  margin: 5px 0;
  border-radius: 5px;
  cursor: pointer;
}

.user-item:hover {
  background-color: #f5f5f5;
}
</style>
```

src/views/UserProfile.vue

```
<template>
  <div class="user-profile">
    <h1>👤 Профиль пользователя</h1>

    <!-- Показываем ID пользователя из параметров маршрута -->
    <div v-if="$route.params.id" class="user-info">
      <h2>Профиль пользователя #{{ $route.params.id }}</h2>
      <p>Это страница конкретного пользователя</p>
    </div>

    <div v-else class="current-user">
      <h2>Ваш профиль</h2>
      <p>Email: user@example.com</p>
      <p>Дата регистрации: 2024-01-01</p>
    </div>

    <!-- Навигация между разделами профиля -->
    <nav class="profile-tabs">
      <router-link to="/profile/info" class="tab">Информация</router-link>
      <router-link to="/profile/posts" class="tab">Посты</router-link>
      <router-link to="/profile/friends" class="tab">Друзья</router-link>
    </nav>

    <!-- Отображаем вложенные маршруты -->
    <div class="tab-content">
      <router-view></router-view>
    </div>

    <!-- Кнопки навигации -->
    <div class="navigation-buttons">
      <button @click="goBack" class="btn">Назад</button>
      <button @click="goHome" class="btn">На главную</button>
      <button @click="goToSettings" class="btn">Настройки</button>
    </div>
  </div>
</template>

<script>
import { useRouter } from 'vue-router'

export default {
  name: 'UserProfile',

  setup() {
    const router = useRouter()

    const goBack = () => {
      router.back()
    }

    const goHome = () => {
      router.push('/')
```

```
    }

    const goToSettings = () => {
      router.push('/settings')
    }

    return {
      goBack,
      goHome,
      goToSettings
    }
  },

  // Хуки навигации
  beforeRouteEnter(to, from, next) {
    console.log('Заходим в профиль пользователя')
    next()
  },

  beforeRouteUpdate(to, from, next) {
    console.log('Обновляем параметры маршрута профиля')
    next()
  }
}
</script>

<style scoped>
.user-profile {
  max-width: 800px;
  margin: 0 auto;
}

.profile-tabs {
  display: flex;
  gap: 10px;
  margin: 20px 0;
  border-bottom: 2px solid #eee;
}

.tab {
  padding: 10px 20px;
  text-decoration: none;
  color: #333;
  border-bottom: 3px solid transparent;
}

.tab.router-link-active {
  border-bottom-color: #007bff;
  color: #007bff;
}

.tab-content {
  padding: 20px;
  border: 1px solid #eee;
```

```
  border-radius: 5px;
  min-height: 200px;
}

.navigation-buttons {
  margin-top: 20px;
  display: flex;
  gap: 10px;
}
</style>
```

src/views/UserSettings.vue

```
<template>
  <div class="user-settings">
    <h1>⚙ Настройки</h1>

    <div class="settings-tabs">
      <router-link to="/settings/general" class="tab">Основные</router-link>
      <router-link to="/settings/security" class="tab">Безопасность</router-link>
      <router-link to="/settings/notifications" class="tab">Уведомления</router-
link>
    </div>

    <div class="settings-content">
      <!-- Параметры маршрута передаются как пропсы -->
      <p>Активная вкладка: {{ currentTab }}</p>

      <router-view :current-tab="currentTab"></router-view>
    </div>
  </div>
</template>

<script>
export default {
  name: 'UserSettings',

  // Получаем параметр маршрута как пропс
  props: {
    tab: {
      type: String,
      default: 'general'
    }
  },

  computed: {
    currentTab() {
      return this.tab || 'general'
    }
  }
}
</script>
```

```
<style scoped>
.settings-tabs {
  display: flex;
  gap: 10px;
  margin: 20px 0;
}

.tab {
  padding: 10px 20px;
  text-decoration: none;
  color: #333;
  border: 1px solid #ddd;
  border-radius: 5px;
}

.tab.router-link-active {
  background-color: #007bff;
  color: white;
  border-color: #007bff;
}

.settings-content {
  padding: 20px;
  border: 1px solid #eee;
  border-radius: 5px;
}
</style>
```

src/views/NotFound.vue

```
<template>
  <div class="not-found">
    <h1>404 - Страница не найдена</h1>
    <p>Запрошенная страница не существует.</p>
    <button @click="goHome" class="btn">Вернуться на главную</button>

    <div class="debug-info">
      <p>Путь: {{ $route.path }}</p>
      <p>Параметры: {{ $route.params }}</p>
    </div>
  </div>
</template>

<script>
import { useRouter } from 'vue-router'

export default {
  name: 'NotFound',

  setup() {
    const router = useRouter()
```

```
    const goHome = () => {
      router.push('/')
    }

    return {
      goHome
    }
  }
}
</script>

<style scoped>
.not-found {
  text-align: center;
  padding: 50px 20px;
}

.debug-info {
  margin-top: 30px;
  padding: 15px;
  background-color: #f8f9fa;
  border-radius: 5px;
  font-family: monospace;
}
</style>
```

Шаг 4: Настраиваем роутер

`src/router/index.js`

```
import { createRouter, createWebHistory } from 'vue-router'
import HomePage from '../views/HomePage.vue'
import UserProfile from '../views/UserProfile.vue'
import UserSettings from '../views/UserSettings.vue'
import NotFound from '../views/NotFound.vue'

const routes = [
  {
    path: '/',
    name: 'Home',
    component: HomePage
  },
  {
    path: '/profile',
    name: 'Profile',
    component: UserProfile
  },
  {
    // Динамический маршрут с параметром
    path: '/profile/:id',
    name: 'UserDetail',
```

```
      component: UserProfile,
      props: true
    },
    {
      path: '/settings',
      name: 'Settings',
      component: UserSettings,
      // Перенаправляем на вложенный маршрут
      redirect: '/settings/general'
    },
    {
      path: '/settings/:tab',
      name: 'SettingsTab',
      component: UserSettings,
      props: true
    },
    {
      // Обработка 404
      path: '/:pathMatch(.*)*',
      name: 'NotFound',
      component: NotFound
    }
  ]

  const router = createRouter({
    history: createWebHistory(),
    routes
  })

  export default router
```

Шаг 5: Обновляем main.js

src/main.js

```
  import { createApp } from 'vue'
  import App from './App.vue'
  import router from './router'

  const app = createApp(App)
  app.use(router)
  app.mount('#app')
```

Шаг 6: Обновляем App.vue

src/App.vue

```
  <template>
    <div id="app">
      <nav class="main-nav">
```

```
      <router-link to="/" class="nav-link">Главная</router-link>
      <router-link to="/profile" class="nav-link">Профиль</router-link>
      <router-link to="/settings" class="nav-link">Настройки</router-link>
    </nav>

    <main class="main-content">
      <router-view></router-view>
    </main>
  </div>
</template>

<script>
export default {
  name: 'App'
}
</script>

<style>
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #f5f5f5;
}

#app {
  min-height: 100vh;
}

.main-nav {
  background-color: #2c3e50;
  padding: 1rem 2rem;
  display: flex;
  gap: 2rem;
}

.nav-link {
  color: white;
  text-decoration: none;
  padding: 0.5rem 1rem;
  border-radius: 4px;
  transition: background-color 0.3s;
}

.nav-link:hover {
  background-color: #34495e;
}

.nav-link.router-link-active {
  background-color: #42b883;
```

```
}

.main-content {
  padding: 2rem;
  max-width: 1200px;
  margin: 0 auto;
}
</style>
```

Данная практика является в последней по блоку контрольной работы №5. Завершаем и сдаем семестровые практики.