### **Заметки по Vue 3**

#### **1. Установка и запуск проекта**

* **Установка зависимостей**:

Установите все необходимые пакеты проекта с помощью команды:  
npm i

**Запуск проекта в режиме разработки**:

Запустите сервер разработки для Vue с помощью команды:  
  
npm run dev

#### **2. Composition API**

Composition API — это новый способ работы с состоянием и логикой компонентов в Vue 3. Он упрощает реорганизацию кода и делает его более масштабируемым.

**ref**:

ref используется для создания реактивного состояния. Например:  
  
import { ref } from 'vue';

const count = ref(0);

Теперь count — реактивная переменная, и любое изменение её значения будет автоматически обновлять привязанные данные в шаблоне.

#### **3. Компоненты**

Vue поощряет разбиение приложения на небольшие, повторно используемые компоненты. Компоненты помогают структурировать код и управлять разными частями приложения.

**Создание компонента**:

Компоненты в Vue 3 создаются в формате SFC (Single File Component) с использованием .vue файлов, где определяются шаблон, логика и стили.  
<template>

<div>Пример компонента</div>

</template>

<script>

export default {

name: 'ExampleComponent',

};

</script>

#### **4. v-model**

v-model — это директива для двусторонней привязки данных между компонентом и родительским элементом.

Пример использования v-model с полем ввода:  
<input v-model="inputValue" />

При изменении значения inputValue будет обновляться в компоненте.

#### **5. Условные выражения**

Vue поддерживает директивы для условного рендеринга.

**v-if и v-else**:

Используйте v-if для условного отображения элементов:  
  
<div v-if="isVisible">Элемент виден</div>

<div v-else>Элемент скрыт</div>

#### **6. События**

В Vue события обрабатываются с помощью директивы @.

Пример обработки клика:  
<button @click="handleClick">Нажми меня</button>

Все события, такие как @click, @submit, @input, могут обрабатываться аналогичным образом.

#### **7. Жизненный цикл: onMounted**

**onMounted**:

Хук жизненного цикла, который запускается, когда компонент полностью отрисован.  
  
import { onMounted } from 'vue';

onMounted(() => {

console.log('Компонент смонтирован');

});

#### **8. Vue Router**

Vue Router позволяет добавлять маршрутизацию (переходы между страницами) в приложение.

Пример настройки маршрута в router/index.js:  
  
import { createRouter, createWebHistory } from 'vue-router';

import Home from '../views/Home.vue';

const routes = [

{ path: '/', name: 'Home', component: Home },

];

const router = createRouter({

history: createWebHistory(process.env.BASE\_URL),

routes,

});

export default router;

#### **9. Функция для запросов к API**

Эта функция позволяет отправлять запросы на сервер с поддержкой авторизации и обработки ответа.

export default async function (method, route, body = null, isBlob = false) {

const options = {

method,

headers: {

Accept: 'application/json',

}

};

// Добавляем токен авторизации, если он сохранен в localStorage

if (localStorage.getItem('user-token')) {

options.headers['Authorization'] = `Bearer ${localStorage.getItem('user-token')}`;

}

// Проверяем наличие тела запроса

if (body) {

if (body instanceof FormData) {

options.body = body;

} else {

options.headers['Content-Type'] = 'application/json';

options.body = JSON.stringify(body);

}

}

// Выполняем запрос

const response = await fetch(`http://localhost/laravel-edu/public/api-kosmos${route}`, options);

let result = null;

// Проверка статуса 401 (неавторизован)

if (response.status === 401) {

localStorage.removeItem('user-token');

window.location.replace('/auth');

}

try {

result = isBlob ? await response.blob() : await response.json();

} catch (e) {

// Обработка ошибки в случае неудачного парсинга

}

return result;

}

* **Параметры**:
  + method — HTTP-метод (GET, POST, PUT, DELETE).
  + route — маршрут для запроса.
  + body — данные для отправки.
  + isBlob — если true, то ожидается бинарный файл (например, изображение).

Эта функция делает запрос на сервер, обрабатывает авторизацию и возвращает данные или переадресует на /auth при ошибке 401.