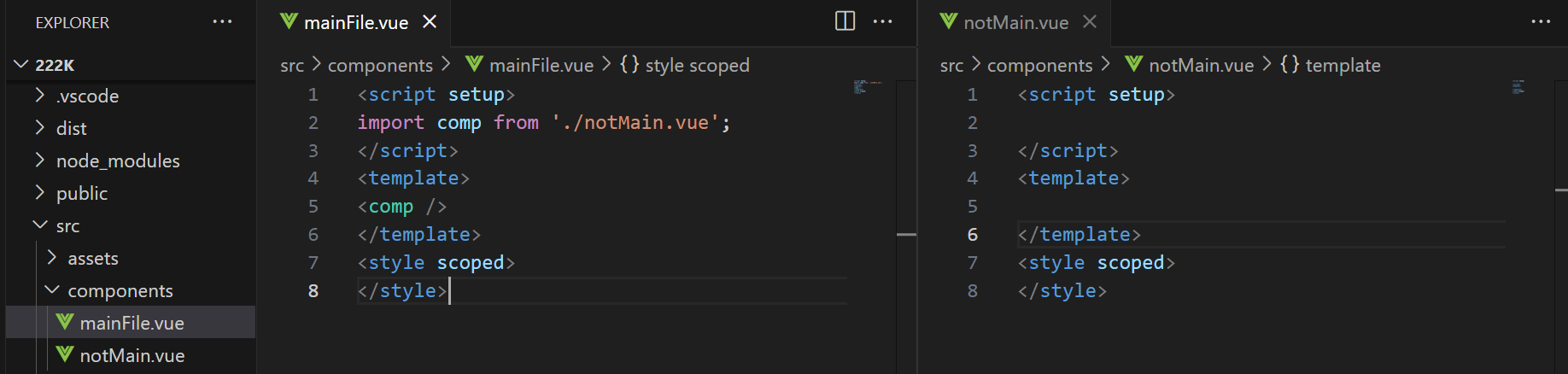
Довольно быстро в работе с компонентами появится необходимость передавать данные не только в компонент, но и из него. В дальнейшей работе будут использоваться обозначения дочерний и родительский(или материнский) компоненты. Здесь всё просто, два связанных компонента всегда делятся на тот, который вызывает и тот, который вызывается. То есть в материнском компоненте мы импортируем и применяем дочерний.

Разобравшись с терминологией, перейдём к практике. Нам необходимы как раз два компонента, один из которых будет главным, а второй – подчиненным. В главном компоненте будет вызван подчиненный.



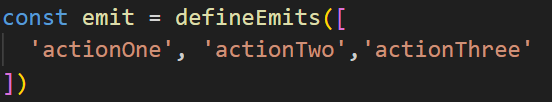
Как передавать данные мы уже знаем, данный процесс проходит в два этапа, отправка и приём. С обратным процессом всё немного сложнее. В первую очередь, это не чистые данные, а события, которые мы активируем на уровне дочернего компонента, а обрабатываем на уровне материнского. Вся работа с кодом в данном случае происходит в три этапа.

Этап 1 – объявление

Наподобие того, как мы регистрировали пропы при отправке данных, мы регистрируем все события, которые планируем использовать в компоненте. Регистрировать необходимо только те события, которые влияют на внешние компоненты! Предположим, я хочу с передать данные с дочернего на родительский компонент. Для этого сначала я создам переменную на материнском компоненте, которую буду выводить и потом буду изменять



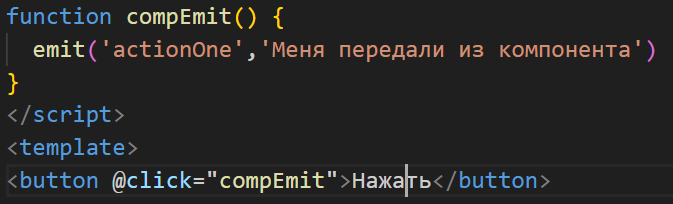
Теперь объявим для этого событие на дочернем компоненте. Объявление идет через создание переменной emit через функцию defineEmits, в которую передаётся список (он же массив) из строковых имён событий. Если их несколько, то можно писать множество названий, через запятую



Этап 2 – применение

Объявив нужные эмиты мы просто говорим программе, что такие сигналы есть. Теперь их необходимо применить, то есть записать либо в функцию, либо на кнопку напрямую, чтобы они выполнялись. В случае прямого применения по событию, в начало ставится знак доллара, в остальном всё идентично – пишется имя переменной (т.е. emit), далее в круглых скобках указываются параметры. Первым ВСЕГДА идёт имя применяемого эмита. Остальные – просто переменные, которые мы хотим передавать с этим событием.

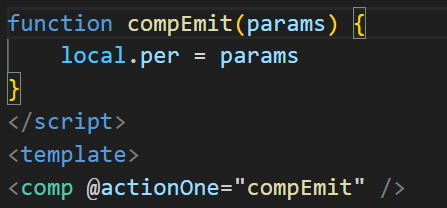




Два способа выше полностью идентичные, однако в случае работы с функцией, перед отправкой данных можно сделать что-то ещё, что зачастую необходимо.

Этап 3 – обработка

Не зря проскальзывало слово “события”, поскольку обработка эмитов аналогична обработке событий. То есть в предел тега компонента ставится @, пишется имя эмита и после равно функция, к которой он привязывается. Причем “ловля” переменных происходит на уровне функции, а не при обработке событий:



index

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Запись на прием к доктору</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<div class="form-container">

<h1>Форма записи на прием к доктору</h1>

<form id="registrationForm">

<div class="form-group">

<label for="fullName" class="required">Ваше ФИО</label>

<input type="text" id="fullName" name="fullName" placeholder="Иванов Иван Иванович">

<div id="fullNameError" class="error"></div>

</div>

<div class="form-group">

<label for="birthDate" class="required">Ваша дата рождения</label>

<input type="date" id="birthDate" name="birthDate">

<div id="birthDateError" class="error"></div>

</div>

<div class="form-group">

<label for="phoneNumber" class="required">Ваш номер телефона</label>

<input type="tel" id="phoneNumber" name="phoneNumber" placeholder="79123456789">

<div id="phoneNumberError" class="error"></div>

</div>

<div class="form-group">

<label for="appointmentDate">Дата приема</label>

<input type="date" id="appointmentDate" name="appointmentDate">

<div id="appointmentDateError" class="error"></div>

</div>

<div class="form-group">

<label for="appointmentTime">Время приема</label>

<input type="time" id="appointmentTime" name="appointmentTime">

<div id="appointmentTimeError" class="error"></div>

</div>

<button type="button" id="submitButton">Отправить заявку</button>

</form>

</div>

<footer>

<p>Защищено ФСБ.</p>

<p>Телефон: +7 (999) 444-33-22</p>

</footer>

<script src="script.js"></script>

</body>

</html>

script

document.addEventListener('DOMContentLoaded', function() {

// Установка минимальной даты приема (завтра)

const tomorrow = new Date();

tomorrow.setDate(tomorrow.getDate() + 1);

const minDate = tomorrow.toISOString().split('T')[0];

document.getElementById('appointmentDate').min = minDate;

// Находим кнопку отправки

const submitButton = document.getElementById('submitButton');

// Вешаем обработчик на кнопку

submitButton.addEventListener('click', function() {

let isValid = true;

// Сброс предыдущих ошибок

document.querySelectorAll('.error').forEach(el => {

el.textContent = '';

});

document.querySelectorAll('input').forEach(el => {

el.classList.remove('invalid');

});

// Проверка ФИО

const fullName = document.getElementById('fullName');

if (!fullName.value !/^[А-Яа-яЁё\s]+$/.test(fullName.value)) {

document.getElementById('fullNameError').textContent = 'Введите ФИО нормально';

fullName.classList.add('invalid');

isValid = false;

}

// Проверка даты рождения

const birthDate = document.getElementById('birthDate');

if (!birthDate.value) {

document.getElementById('birthDateError').textContent = 'Введите дату рождения';

birthDate.classList.add('invalid');

isValid = false;

} else {

const birthYear = new Date(birthDate.value).getFullYear();

if (birthYear > 2006) {

document.getElementById('birthDateError').textContent = 'Не позже 2006 года';

birthDate.classList.add('invalid');

isValid = false;

}

}

// Проверка телефона

const phoneNumber = document.getElementById('phoneNumber');

if (!phoneNumber.value !/^[0-9]{11}$/.test(phoneNumber.value)) {

document.getElementById('phoneNumberError').textContent = 'Тут должно быть 11 цифр!';

phoneNumber.classList.add('invalid');

isValid = false;

}

// Проверка даты приема (если указана)

const appointmentDate = document.getElementById('appointmentDate');

if (appointmentDate.value && new Date(appointmentDate.value) < tomorrow) {

document.getElementById('appointmentDateError').textContent = 'Не раньше завтрашнего дня';

appointmentDate.classList.add('invalid');

isValid = false;

}

// Проверка времени приема (если указано)

const appointmentTime = document.getElementById('appointmentTime');

if (appointmentTime.value) {

const [hours, minutes] = appointmentTime.value.split(':').map(Number);

if (hours < 8 hours > 22 (hours === 22 && minutes > 0)) {

document.getElementById('appointmentTimeError').textContent = 'с 8:00 до 22:00!';

appointmentTime.classList.add('invalid');

isValid = false;

}

}

if (isValid) {

alert('Ваша заявка принята!');

// Здесь можно отправить форму

// document.getElementById('registrationForm').submit();

} else {

// Прокрутка к первой ошибке

const firstError = document.querySelector('.invalid');

if (firstError) {

firstError.scrollIntoView({ behavior: 'smooth', block: 'center' });

}

}

});

// Очистка ошибок при вводе

document.querySelectorAll('input').forEach(input => {

input.addEventListener('input', function() {

if (this.classList.contains('invalid')) {

this.classList.remove('invalid');

const errorElement = document.getElementById(this.id + 'Error');

if (errorElement) errorElement.textContent = '';

}

});

});

});

style

body {

font-family: 'Arial', sans-serif;

margin: 0;

padding: 20px;

background-color: #f0f8ff; /\* Светло-голубой фон \*/

color: #333;

line-height: 1.6;

}

.form-container {

max-width: 600px;

margin: 20px auto;

background: white;

padding: 25px;

border: 1px solid #b3d4ff; /\* Светло-синяя граница \*/

}

h1 {

margin-top: 0;

font-size: 24px;

text-align: center;

padding-bottom: 15px;

border-bottom: 1px solid #e6f0ff;

color: #0066cc; /\* Синий цвет заголовка \*/

}

.form-group {

margin-bottom: 18px;

}

label {

display: block;

margin-bottom: 8px;

font-weight: bold;

color: #004080; /\* Темно-синий цвет меток \*/

}

.required:after {

content: " \*";

color: #cc0000;

}

input[type="text"],

input[type="date"],

input[type="time"],

input[type="tel"] {

width: 100%;

padding: 10px;

border: 1px solid #99c2ff; /\* Светло-синяя граница \*/

font-size: 16px;

}

input[type="text"]:focus,

input[type="date"]:focus,

input[type="time"]:focus,

input[type="tel"]:focus {

border-color: #0066cc; /\* Синий цвет при фокусе \*/

outline: none;

}

input.invalid {

border-color: #ff3333;

}

button {

width: 100%;

padding: 12px;

background-color: #0066cc; /\* Синий цвет кнопки \*/

color: white;

border: none;

font-size: 16px;

cursor: pointer;

margin-top: 15px;

}

button:hover {

background-color: #004d99; /\* Темно-синий при наведении \*/

}

.error {

color: #cc0000;

font-size: 13px;

margin-top: 5px;

display: block;

min-height: 18px;

}

footer {

margin-top: 30px;

text-align: center;

padding: 15px;

color: #666;

font-size: 14px;

}

footer p {

margin: 5px 0;

}