**Лабораторная работа 2**

**Компоненты Vue.js**

const app = Vue.createApp({})

app.component(название компонента, {

настройки

});

app.mount('#название компонента')

Рассмотрим компонент с названием **my-component**:

Давайте теперь добавим настройки нашему компоненту. Первая и самая главная настройка - это **template**. С ее помощью мы должны задать HTML код, который будет результатом работы нашего компонента.

Давайте добавим эту настройку:

**JS**

let app = Vue.createApp({

});

app.component('my-component', {

template: '<div>Пользовательский компонент!</div>',

});

app.mount('#app')

**HTML**

<div id="app">

<my-component></my-component>

</div>

Передача данных в компонент

**JS**

let app = Vue.createApp({

});

app.component('my-component', {

props: ['name'],

template: '<div>Привет, {{ name }}!</div>',

});

app.mount('#app')

**HTML**

<div id="app">

<my-component name="Вася"></my-component>

</div>

Результатом выполнения этого кода будет следующий HTML:

<div id="app">

<div class="task">Привет, Вася!</div>

</div>

Компоненты можно повторно использовать столько раз, сколько потребуется:

<div id="app">

<my-component name="Вася"></my-component>

<my-component name="Коля"></my-component>

<my-component name="Настя"></my-component>

</div>

Передача свойств из data

Пусть у нас теперь 'Вася' хранится в свойстве **value** объекта **data**:

let app = Vue.createApp({

data(){

return {

value: 'Вася',

}

}

});

Чтобы передать данные из свойства **value** нам нужно будет указать не **name="value"**, а **v-bind:name="value"**:

<div id="app">

<my-component v-bind:name="value"></my-component>

</div>

Если написать просто **name="value"**, то в компонент передастся строка 'value', а не содержимое свойства **value**.

Давайте совместим все вышеописанное и запустим:

let app Vue.createApp({

data(){

return {

value: 'Вася',

}

}

});

app.component('my-component', {

props: ['name'],

template: '<div>Привет, {{ name }}!</div>',

});

app.mount('#app')

<div id="app">

<my-component v-bind:name="value"></my-component>

</div>

Результатом выполнения этого кода будет следующий HTML:

<div id="app">

<div class="task">Привет, Вася!</div>

</div>

Итак, данные из **data** родительского компонента можно передавать в дочерний через директиву **v-bind**.

Компоненты и циклы

Пусть теперь в свойстве **values** хранится массив имен:

let app = Vue.createApp({

data(){

return{

values: ['Коля', 'Вася', 'Петя'], // массив имен

}

}

});

Давайте применим к свойству **values** директиву **v-for** и с ее помощью повторим наш компонент **my-component** в цикле:

<div id="app">

<my-component v-for="value in values"></my-component>

</div>

Так можно делать и при это переменная **value** становится доступна в атрибутах нашего компонента. Давайте передадим ее в компонент с помощью директивы **v-bind**:

<div id="app">

<my-component

v-for="(value, index) in values"

v-bind:name="value"

>

</my-component>

</div>

Есть еще один нюанс. Чтобы Vue проще было работать с компонентами, нам необходимо указать ключ через **v-bind:key** (подобное мы уже проходили при работе с обычными циклами):

<div id="app">

<my-component

v-for="(value, index) in values"

v-bind:name="value"

v-bind:key="index"

>

</my-component>

</div>

Итак, давайте напишем полный код и запустим его:

let app = Vue.createApp({

data(){

return{

values: ['Коля', 'Вася', 'Петя'], // массив имен

}

}

});

app.component('my-component', {

props: ['name'],

template: '<div>Привет, {{ name }}!</div>',

});

div id="app">

<my-component

v-for="(value, index) in values"

v-bind:name="value"

v-bind:key="index" >

</my-component>

</div>

Результатом выполнения этого кода будет следующий HTML:

<div id="app">

<div class="task">Привет, Коля!</div>

<div>Привет, Вася!</div>

<div>Привет, Петя!</div>

</div>

Итак, компоненты можно и нужно перебирать в цикле - это достаточно удобно.

**Обработка событий**

// Создаём приложение Vue

const app = Vue.createApp({})

// Определяем новый глобальный компонент с именем button-counter

app.component('button-counter', {

data() {

return {

count: 0

}

},

template: `

<button v-on:click="count++">

Счётчик кликов — {{ count }}

</button>`

})

app.mount('#components-demo')

<div id="components-demo">

<button-counter></button-counter>

</div>

Вместо операции инкремента v-on:click="count++" можно использовать метод, описанный в объекте “app”.

**Задания.** Во всех заданиях активно! использовать стилевое оформление компонентов.

**Задание 2.**

Разработать мини-сайт на vue.js с меню (меню может располагаться слева либо сверху). Сайт содержит не менее 5-ти страниц. Каждая страница сайта сформирована в отдельном компоненте. При выборе одного из пунктов меню страницы должны динамически переключаться (заменяется отображаемый объект). При этом активный пункт меню подсвевичивается.

**Задание 3.**

Дан массив с вопросами и вариантами ответов следующего типа:

[

{

question: 'Вопрос 1',

answers: [

'Ответ 1',

'Ответ 2',

'Ответ 3',

'Ответ 4',

'Ответ 5',

],

right: 3, // номер правильного ответа

},

{

question: 'Вопрос 2',

answers: [

'Ответ 1',

'Ответ 2',

'Ответ 3',

'Ответ 4',

'Ответ 5',

],

right: 1, // номер правильного ответа

},

{

question: 'Вопрос 3',

answers: [

'Ответ 1',

'Ответ 2',

'Ответ 3',

'Ответ 4',

'Ответ 5',

],

right: 5, // номер правильного ответа

},

]

Реализуйте тест с вопросами и вариантами ответов(не менее 5-ти вопросов). Каждый вопрос должен быть в своем абзаце, а под ним - 5 радиокнопочек, с помощью которых можно выбрать один из ответов. Если ответ правильный - текст вопроса должен быть выделен рамкой (border) зеленого цвета, а если неправильный - красного. **Необходимо сформировать компонент, который будет использоваться для вывода вопроса на страницу и управления его состоянием.**