**Лабораторная работа № 1**

**Работа с фреймворком Vue js. Директивы Vue.**

**Подключение Vue js**

Для подключения Vue.js можно воспользоваться, приведенным ниже скриптом. Достаточно скопировать его в раздел <head> </head> Вашего html-документа.

<script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

Для запуска Vue нужно создать новый объект класса Vue. Этот класс доступен после подключения Vue.

Однако, просто создать объект мало - необходимо передать начальные настройки. Эти настройки передаются в виде объекта **{}**:

Самой главной настройкой является настройка **el** - в нее мы должны передать селектор того элемента, в котором будет работать наш **Vue**:

let app = new Vue({

el: '#app',

});

Следующая настройка - **data** - хранит в себе данные, которые мы хотим выводить на нашей HTML странице. Данные хранятся в виде объекта *ключ-значение*.

**Пример**

Пусть у нас в **data** в ключе **message** хранится сообщение с текстом 'hello'

let app = new Vue({

el: '#app',

data: {

message: 'hello',

},

});

Это сообщение мы можем вывести в любом месте нашего HTML следующим образом: **{{ message }}**

<div id="app">

{{ message }}

</div>

**Работа с циклами**

Пусть теперь в **data** у нас есть элемент **items**, который содержит внутри себя массив с некоторыми данными:

let app = new Vue({

el: '#app',

data: {

items: ['a', 'b', 'c', 'd', 'e'],

},

});

Давайте сделаем список **ul**, внутри которого разместим каждый элемент нашего массива **items** в отдельную **li**.

Очевидно, что нам нужен какой-то цикл, который сделает это за нас. Во Vue такой цикл делается следующим образом: **<li v-for="item in items">**.

Давайте разберем, что тут происходит. Атрибут **v-for** запускает цикл, строка **item in items** указывает на то, что мы будем перебирать массив **items**, а каждый элемент этого массива будет ложиться в переменную **item**.

В результате эта строка: **<li v-for="item in items">** - сделает столько тегов **li**, сколько раз прокрутится цикл.

Внутри **li** будет доступна переменная **item**, в которой каждой раз будет содержаться новый элемент массива. Мы можем вывести ее содержимое с помощью фигурных скобок, вот так: **{{ item }}**.

Иногда кроме элементов массива, мы хотели бы получить доступ еще и к ключам. Для этого вместо **item in items** следует написать **(item, index) in items** - тогда в переменную **item** будут попадать элементы массива, а в **index** - их ключи. Вместо **index** вы можете использовать любое другое название, как вам удобно.

**Пример 1**

let app = new Vue({

el: '#app',

data: {

items: ['a', 'b', 'c', 'd', 'e'],

},

});

<div id="app">

<ul>

<li v-for="item in items">{{ item }}</li>

</ul>

</div>

Если необходимо вывести и значения и порядковые номера.

**Пример 2**

let app = new Vue({

el: '#app',

data: {

items: ['a', 'b', 'c', 'd', 'e'],

},

});

<div id="app">

<ul>

<li v-for="(item, index) in items">{{ item }} - {{ index }}</li>

</ul>

</div>

**Обработка событий**

Для того, чтобы навесить событие на какой-нибудь DOM элемент, во Vue есть следующая директива:

**v-on:названиеСобытия**.

В примере 1 приведен код, который изменяет значение message по нажатию на кнопку с помощью функции changeMessage.

**Пример 1**

let app = new Vue({ el: '#app',

data: {

message: 'привет',

},

methods: {

changeMessage: function() {

this.message = 'пока';

}

}

});

<div id="app">

{{ message }}

<button v-on:click="changeMessage">Поменять сообщение</button>

</div>

**Передача параметров в метод**

**Пример 2.**

<div id="a6">

    <ol start=0>

        <li v-for="(item,index) in items" v-on:click="showindex(index,item)"> -- {{item}}</li>

    </ol>

</div>

let a6=new Vue({

    el:'#a6',

    data:{

        items:[1,2,3,5,7,6,4]

        },

    methods:{

        show:function(ind,value){

            alert('index:'+ind+' value '+value);

        }

    }

})

**Работа с классами во Vue**

Работа с атрибутом **class** происходит аналогичным образом - через **v-bind**..

Пусть в свойстве **str** хранится какой-то класс. Добавим его нашему абзацу:

let app = new Vue({

el: '#app',

data: {

str: 'www',

},

});

<div id="app">

<p v-bind:class="str">Абзац</p>

</div>

// здесь «str» - это имя сss класса .str{…}

!Через пробел в кавычках можно перечислить несколько имен классов. Например,

<p v-bind:class="str content header">Абзац</p>

## Добавление класса по условию

## Для этого в v-bind:class напишем объект, в котором ключом будет желаемый класс (в нашем случае active), а значением - свойство, в зависимости от которого этот класс будет добавлен нашему абзацу или не будет (в нашем случае это свойство isActive):

let app = new Vue({

el: '#app',

data: {

isActive: true,

},

});

<div id="app">

<p v-bind:class="{active: isActive}">Абзац</p>

</div>

**Задания**

**Задание 1.**

В объекте data задана следующая информация.

    data:{

        header:'список приглашенных',

        persons:[

                {id:0,name:'Иван',surname:'Иванов',donate:300},

                {id:1,name:'Петр',surname:'Петров',donate:200},

                {id:2,name:'Александр',surname:'Александров',donate:400},

        ]

    },

Вывести на html -страницу информацию из переменной header в виде заголовка и каждый элемент массива persons в отдельную строку (без использования директив). После выведенных элементов массива вычислить общую сумму взносов (donate).

**Задание 2.**

Даны три элемета input (text). С их помощью можно ввести такие данные как, наименование товара, количество и цену. По нажатию на кнопку «добавить» товар должен сохраняться и выводиться в таблицу ниже инпутов. Кроме того должен выводиться столбец сумма=количество\*цена и столбец со ссылкой «Удалить». Также необходимо перед таблицей необходимо добавить кнопку «Сортировать» после чего должна происходить сортировка по имени товара.

**Задание 3.**

Реализуйте чеклист. Должен быть инпут с добавлением задачи, а над ним - список с задачами. Возле каждой задачи чекбосc. Если нажать на чекбокс - задача становится выполненной, перечеркивается и становится серого цвета, чекбокс при этом пропадает. Справа от любой задачи должен быть крестик для удаления этой задачи.

Пусть новая задача добавляется по нажатию клавиши Enter.

Задачу также можно подредактировать, если сделать двойной клик по ней - в этом случае вместо пункта списка появляется инпут с текстом задачи, можно подредактировать текст задачи, нажать Enter и задача изменится. Инпут при этом исчезнет.

**Задание 4.**

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.