Программирование на стороне клиента

Модуль 2 / 5

Технологии

Что такое VueJS и зачем?

Vue (произносится /vjuː/, примерно как **view**) — это **прогрессивный фреймворк** для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов, Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками.

А если своими словами, использование фреймворка помогает создавать легко более сложные вещи, при этом писать меньше кода и абстрагироваться от технических деталей вроде ручной перерисовки DOM.

Подключение

В наших пробных проектах будем использовать подключение через cdn для быстрого старта:

```
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
```

Для более удобной работы с компонентами, модулями, роутингом и т.д., проект билдится через Vue CLI. Это потребует установки на компьютер NodeJS и npm.

Подробнее о способах установки:
https://ru.vuejs.org/v2/guide/installation.html

Структура объекта

```
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
<script>
    var app = new Vue({
        el: '#app',
        data: {
            name: 'Безымянный веб-разработчик',
            messageText: 'А кто говорил, что будет легко? :)',
            skills: ['HTML', 'CSS', 'JS', 'PHP', 'MySQL']
        },
        methods: {
            addSkill: function(val) {
                this.languages.push(val)
            }
        },
        mounted() {
            console.log('Я родился')
        }
    })
</script>
```

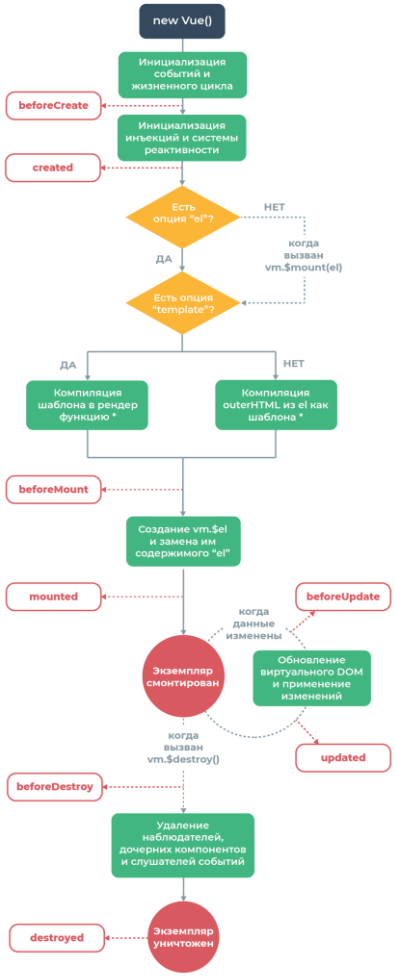
el – селектор элемента, в который рендерится приложение

data – наши данные. Обладают свойством реактивности, то есть при изменении сразу же отображаются

methods – методы (действия)

mounted – один из хуков жизненного цикла

Жизненный цикл объекта



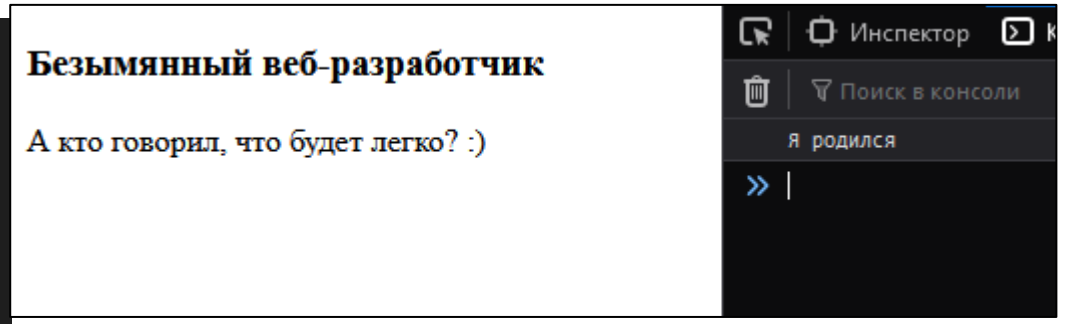
Тут не очень хорошо видно, так что лучше посмотреть по ссылке:

https://ru.vuejs.org/v2/guide/instance.html#%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0-%D0%B6%D0%B8%D0%B7%D0%BD%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE-%D1%86%D0%B8%D0%BA%D0%BB%D0%B0

☺

Отображение данных в шаблоне

```
<div id="app">
    <h3>{{ name }}</h3>
    <p>{{ messageText }}</p>
</div>
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
<script>
    var app = new Vue({
        el: '#app',
        data: {
            name: 'Безымянный веб-разработчик',
            messageText: 'А кто говорил, что будет легко? :)',
            skills: ['HTML', 'CSS', 'JS', 'PHP', 'MySQL']
        },
        methods: {
            addSkill: function(val) {
                this.languages.push(val)
            }
        },
        mounted() {
            console.log('Я родился')
        }
    })
</script>
```



Для рендеринга текста внутри тега используется {{ интерполяция }}

Возможности интерполяции

```
<div id="app">
    <h3>{{ name }}</h3>
    <p>{{ messageText }}</p>
    <p>Дважды два плюс пять: {{ Math.pow(2, 2) + 5 }}</p>
    <p>2022 уже наступил? {{ (new Date()).getFullYear() >= 2022 ? 'Ага' : 'Не-а' }}</p>
    <p>{{ skills.join(', ') }}</p>
    <div v-html="'а сейчас будет <b>жирный текст</b>'"></div>
</div>
```

Безымянный веб-разработчик

А кто говорил, что будет легко? :)

Дважды два плюс пять: 9

2022 уже наступил? Ага

HTML, CSS, JS, PHP, MySQL

а сейчас будет **жирный текст**

В шаблон можно передавать не только переменные, но и JS-выражения.

Стоит обратить внимание, что если нужно передать не просто текст, а именно форматированную разметку (сырой HTML), то следует использовать рендеринг через v-html, чтобы теги были распознаны.

Условная отрисовка

```
<div v-if="age < 18">
    А ты еще маленький
</div>
<div v-else>
    Вам пора на работу
</div>
```

С помощью директивы v-if, v-else-if, v-else компоненты отрисовываются по заданному условию.

Также есть такая вещь, как v-show. Оно аналогична v-if, но не поддерживает v-else. Разница в том, что v-if в зависимости от условия отрисовывает элементы либо удаляет их из DOM, а v-show просто переключает свойство display.

Циклическая отрисовка

```
<ul>
    <li v-for="(skill, index) in skills" :key="index">{{ skill }}</li>
</ul>
```

- HTML
- CSS
- JS
- PHP
- MySQL

Здесь работает директива v-for. В цикле, который она перебирает, второй опциональный параметр (index) – индекс текущей итерации. Советую перестраховываться и указывать его, также добавлять атрибут :key, чтобы избежать непредсказуемого поведения.

Кстати, в v-for можно передавать массивы, а делать итерацию по диапазону чисел, например v-for=“i in 10”, что даст нам цикл от 1 до 10.

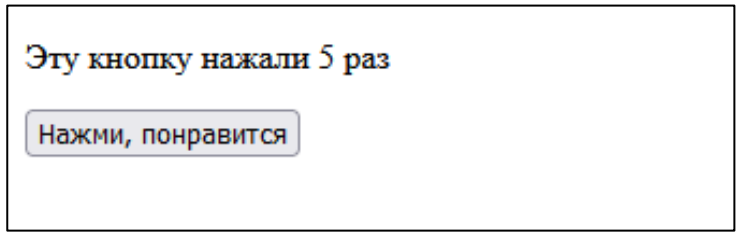
Динамическая передача атрибутов

```
<div id="app">
    <a :href="link.url">{{ link.title }}</a>
</div>
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
<script>
    var app = new Vue({
        el: '#app',
        data: {
            link: {
                title: 'ToU website',
                url: 'https://tou.edu.kz'
            }
        },
```

Атрибуты передаются через такой синтаксис:
:[имяатрибута]=“значение”

Навешивание обработчиков событий

```
<div id="app">
    <p>Эту кнопку нажали {{ clicksCount }} раз</p>
    <button @click="incrementClicks">Нажми, понравится</button>
</div>
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
<script>
    var app = new Vue({
        el: '#app',
        data: {
            clicksCount: 0
        },
        methods: {
            incrementClicks: function() {
                this.clicksCount++
            }
        },
```



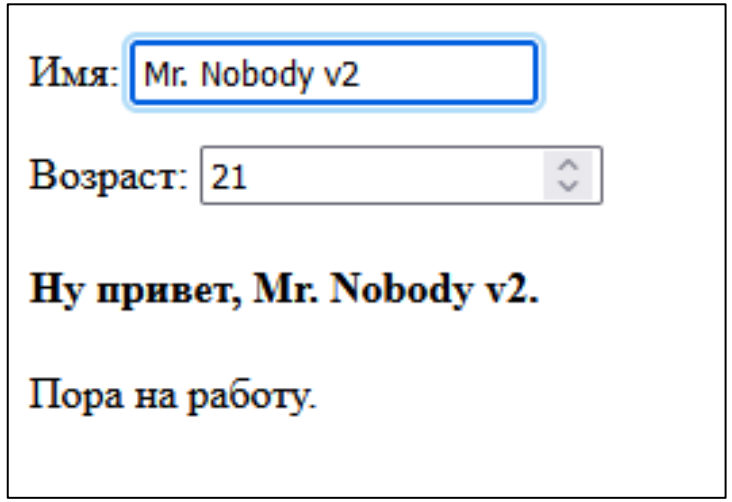
@[имясобытия]=“имяметода”

Стоит знать также модификатор .prevent, чтобы отменять стандартное поведение, как в случаях с формой и ссылкой, например:

@click.prevent=“…”
@submit.prevent=“…”

Управление данными через v-model

```
<div id="app">
    <p>
        Имя: <input type="text" v-model="name">
    </p>
    <p>
        Возраст: <input type="number" v-model="age">
    </p>
    <h4>Ну привет, {{ name }}.</h4>
    <p v-if="age < 18">
        Есть ли жизнь до 18?
    </p>
    <p v-else>
        Пора на работу.
    </p>
</div>
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
<script>
    var app = new Vue({
        el: '#app',
        data: {
            name: 'Mr. Nobody',
            age: 21
        },
```



Так как данные реактивны, при изменении значений в полях ввода, они динамически меняются.

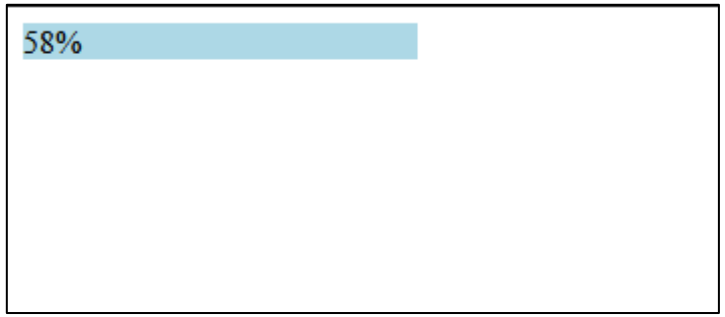
Динамические классы

```
<div :class="{'text-danger': hasError}"></div>
<div :class="{'current': item.id == activeItem.id}"></div>
<div :class="{'text-success': skills.length > 5}"></div>
```

В данном случае в качестве класса передается объект, где ключами будут имена классов, а значения приводятся к логическим (true или false), что указывает на то, будет ли добавлен данный класс к элементу.

Динамические стили

```
<body>
    <div id="app">
        <div :style="{width: progress + '%', backgroundColor: successState ? 'lightgreen' : 'lightblue'}">
            {{ progress }}%
        </div>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
    <script>
        var app = new Vue({
            el: '#app',
            data: {
                progress: 0,
                successState: false
            },
            mounted() {
                let interval = setInterval(() => {
                    this.progress++
                    if(this.progress >= 100) {
                        this.successState = true
                        clearInterval(interval)
                    }
                }, 50)
            }
        })
    </script>
</body>
```



Стили биндятся аналогично классам, через объект, ключами которого являются имена свойств CSS, а значениями – значения.

Вычисляемые свойства

```
var app = new Vue({
    el: '#app',
    data: {
        firstName: 'Айдана',
        lastName: 'Маратова'
    },
    computed: {
        fullName: function() {
            return this.firstName + ' ' + this.lastName
        }
    }
})
```

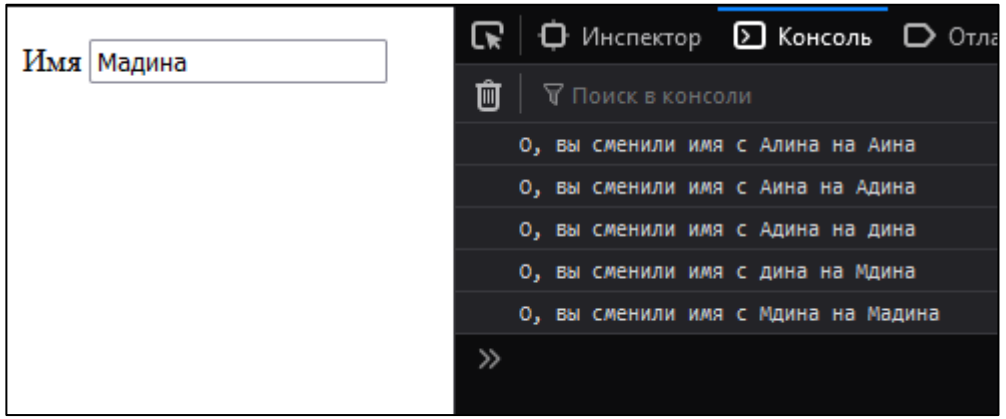
С помощью данного вычисляемого свойства мы можем использовать в шаблоне свойство fullName, и оно будет равно “Айдана Маратова”.

При этом, если изменится значение firstName или lastName, то вычисляемое свойство тоже отреагирует и изменится автоматически.

Computed часто используются для форматирования строк или фильтрации списков, например, если у нас функционал по сортировке и поиску.

Слежение за изменением

```
<div id="app">
    <p>
        <label>
            Имя: <input type="text" v-model="name">
        </label>
    </p>
</div>
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
<script>
    var app = new Vue({
        el: '#app',
        data: {
            name: 'Алина',
        },
        watch: {
            name: function (newValue, oldValue) {
                console.log('О, вы сменили имя с ' + oldValue + ' на ' + newValue)
            }
        },
```



С помощью watch мы можем выполнять какие-либо действия в момент изменения значений. Например, автосохранение.

Слежение за сложными данными

```html
<div id="app">
    <p>
        <label>
            Имя
            <input type="text" v-model="info.name">
        </label>
    </p>
    <p>
        <label>Позиция
            <input type="text" v-model="info.position">
        </label>
    </p>
    <h4>Навыки:</h4>
    <ul>
        <li v-for="(skill, i) in info.skills" :key="i">
            {{ skill }}
        </li>
    </ul>
    <form action="" @submit.prevent="addSkill">
        <input type="text" placeholder="Добавить навык" v-model="skillText">
        <button>+</button>
    </form>
</div>
```

```js
var app = new Vue({
    el: '#app',
    data: {
        skillText: '',
        info: {
            name: 'Алина',
            position: 'Веб-разработчик',
            skills: [],
        }
    },
    watch: {
        info: {
            deep: true,
            handler: function (newValue, oldValue) {
                localStorage.setItem('info', JSON.stringify(newValue))
            }
        }
    },
    methods: {
        addSkill: function() {
            this.info.skills.push(this.skillText)
        }
    }
})
```

Чтобы прослушиватель срабатывал на изменение свойств объекта, надо указывать опцию deep: true и обработчик помещать в handler.

Немного о localStorage

localStorage – это локальное хранилище браузера. Подробнее на https://developer.mozilla.org/ru/docs/Web/API/Window/localStorage

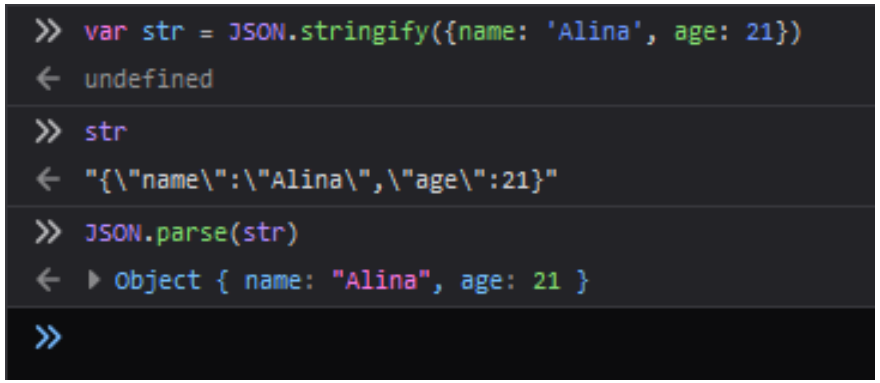
Ключи и значения - всегда строки. Установка, извлечение и удаление выполняется следующим образом:

```
localStorage.setItem('name', 'Student')
localStorage.getItem('name')
localStorage.removeItem('name')
```

JSON сериализация

Так как localStorage может работать только со строками, сложные данные типа объектов или массивов надо сериализовать в JSON. Делается это так:

```
localStorage.setItem('name', JSON.stringify({name: 'Alina', age: 21}))
let infoObject = JSON.parse(localStorage.getItem('name'))
```



- Пример сериализации и десереализации

Полезные ссылки

https://ru.vuejs.org/v2/guide/installation.html
https://developer.mozilla.org/ru/docs/Web/API/Window/localStorage
https://learn.javascript.ru/json