МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБО6РОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЁТ   
ЗАЩИЩЁН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ассистент |  |  |  | Ю.В. Ветрова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ПРАКТИЧЕСКОМ ЗАДАНИИ №1 |
|  |
| по курсу: JavaScript, его библиотеки и фреймворки в Frontend-разработке |
|  |
|  |

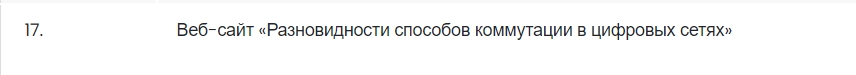
РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4136 |  |  |  | Бобрович Н.С. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

1. **Введение:**

Вариант:



Веб-сайт посвящён различным способам коммутации в цифровых сетях. Эта информация может помочь как новичка, так и специалистам в области цифровых сетей.

1. **Постановка задачи:**
2. Требования к проекту:
3. Проект должен представлять собой одностраничный веб-сайт.
4. Обязательными элементами интерфейса должны быть несколько вкладок (табов), оформленных как навигационное меню, в соответствии с которыми контент сайта должен быть разделен на несколько частей.
5. Навигационное меню должно быть фиксированным и не прокручиваться вместе со страницей, а нажатие на ссылки должны вести на соответствующий компонент без перезагрузки страницы (вкладки должны открываться без перезагрузки страницы).
6. В работе должны быть использованы современные технологии разработки web-сайтов, которые были изучены в процессе теоретического обучения по данному ИТ-модулю.
7. Обязательным является использование HTML5, технологии CSS3, а также программирования на JavaScript, Vue,js.
8. На странице обязательно должен присутствовать компонент «слайдер изображений в галерее». Для смены картинок необходимо наличие кнопок (вправо/влево или </> и т.п.). Кнопки должны иметь стилевое оформление.
9. При обеспечении адаптивности необходимо предусмотреть 3 контрольные точки (например, 1200px, 800px, 550px).
10. Состав команды:

Бобрович Н.С.

1. Задачи:
2. Следует обратить внимание на продуманность дизайна веб-страницы, ее архитектуры, на удобство навигации, привлекательность подачи информации.
3. **Проектирование сайта:**
4. Выбор программных средств и технологий для решения поставленных задач:

В работе должны быть использованы современные технологии разработки web-сайтов, которые были изучены в процессе теоретического обучения по данному ИТ-модулю. Обязательным является использование HTML5, технологии CSS3, а также программирования на JavaScript, Vue,js.

1. Макет сайта:
2. Реактивность сайта с использованием Vue.js:

Фреймворк Vue.js используется в первую очередь для разработки пользовательских интерфейсов и одностраничных веб-приложений (мой случай). Основные цели и преимущества использования Vue.js:

1. Построение интерактивных UI-компонентов:

- Vue.js позволяет легко создавать многократно используемые UI-компоненты с возможностью декларативного связывания данных.

- Это упрощает разработку и поддержку сложных веб-интерфейсов.

2. Эффективное управление состоянием приложения:

- Vue.js предоставляет встроенную систему управления состоянием, которая помогает эффективно отслеживать и обновлять данные приложения.

- Это особенно важно для SPA-приложений, где требуется поддержка сложной логики.

3. Гибкость и масштабируемость:

- Vue.js отличается простотой и гибкостью, что позволяет использовать его как для небольших проектов, так и для крупномасштабных приложений.

- Он легко интегрируется с другими библиотеками и фреймворками, обеспечивая высокую масштабируемость.

4. Быстрая производительность:

- Vue.js использует эффективную систему отслеживания изменений и виртуального DOM, что обеспечивает высокую производительность приложений.

- Это позволяет создавать плавные и отзывчивые интерфейсы.

В целом, Vue.js является мощным инструментом для создания современных веб-приложений с акцентом на производительность, масштабируемость и простоту разработки. Он широко используется как в небольших, так и в крупных проектах.

1. Описание взаимодействия с элементами на странице с использованием функционала Vue:
2. **Vue.js-компонент под названием ContactComponent**, который представляет форму для связи с пользователем.

Рассмотрим основные элементы этого компонента:

1. Шаблон (template):

- Содержит разметку HTML для отображения формы и модального окна.

- Форма включает поля для ввода имени, email и сообщения, а также кнопку "Отправить".

- Модальное окно отображается после успешной отправки формы.

2. Скрипт (script):

- Импортирует необходимые стили и скрипты для Bootstrap.

import 'bootstrap/dist/css/bootstrap.css'

import 'bootstrap/dist/js/bootstrap.js'

- Экспортирует объект, который является определением компонента Vue.js.

export default {

// Название компонента

name: 'ContactComponent',

// Данные компонента

data() {

return {

name: '',

email: '',

message: ''

}

}

- Свойство name устанавливает название компонента как 'ContactComponent'.

- Определяет данные компонента (data) - name, email и message.

- Содержит методы:

- submitForm(): Проверяет валидность полей формы, отображает модальное окно об успешной отправке и очищает форму.

- showSuccessModal(): Отображает модальное окно об успешной отправке формы.

- clearForm(): Очищает поля формы.

// Методы компонента

methods: {

// Обработчик отправки формы

submitForm() {

// Проверка валидности полей

if (!this.name || !this.email || !this.message) {

alert('Пожалуйста, заполните все поля формы.');

return;

}

// Отправка формы (можно добавить логику отправки на сервер)

this.showSuccessModal();

this.clearForm();

},

// Показ модального окна об успешной отправке

showSuccessModal() {

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

const modal = new bootstrap.Modal(successModal);

modal.show();

},

// Очистка полей формы

clearForm() {

this.name = '';

this.email = '';

this.message = '';

}

}

Данный компонент реализует следующую функциональность:

1. Форма для связи с пользователем: Компонент содержит форму с полями для ввода имени, email и сообщения. Пользователь может заполнить эти поля и отправить форму.

2. Валидация полей формы: Перед отправкой формы компонент проверяет, что все поля заполнены. Если какое-либо поле не заполнено, отображается сообщение об ошибке.

3. Отправка формы и очистка полей: При успешной отправке формы компонент отображает модальное окно с сообщением об успешной отправке и очищает поля формы.

4. Использование Bootstrap: Компонент использует стили и скрипты Bootstrap для оформления и функциональности модального окна.

2. **Vue.js-компонент под названием ImageSliderComponent**, который представляет собой простой слайдер изображений.

Рассмотрим основные элементы этого компонента:

1. Шаблон (template):

- Содержит разметку HTML для отображения слайдера.

- Включает обертку для слайдера, в которой отображается текущее изображение.

- Также содержит кнопки для перехода к предыдущему и следующему изображению.

2. Скрипт (script):

- Экспортирует объект, который является определением компонента Vue.js.

- Свойство name устанавливает название компонента как 'ImageSliderComponent'.

- Определяет данные компонента (data) - массив изображений images и текущий индекс currentIndex.

export default {

// Название компонента

name: 'ImageSliderComponent',

// Данные компонента

data() {

return {

// Массив изображений

images: [

'https://via.placeholder.com/800x400',

'https://via.placeholder.com/800x400/0000FF',

'https://via.placeholder.com/800x400/FF00FF'

],

// Текущий индекс изображения

currentIndex: 0

}

}

- Содержит вычисляемое свойство (computed) currentImage, которое возвращает текущее изображение из массива images.

// Вычисляемые свойства

computed: {

// Возвращает текущее изображение из массива

currentImage() {

return this.images[this.currentIndex];

}

}

- Определяет методы:

- prevImage(): Переключает на предыдущее изображение в слайдере.

- nextImage(): Переключает на следующее изображение в слайдере.

// Методы компонента

methods: {

// Переход к предыдущему изображению

prevImage() {

this.currentIndex = (this.currentIndex - 1 + this.images.length) % this.images.length;

},

// Переход к следующему изображению

nextImage() {

this.currentIndex = (this.currentIndex + 1) % this.images.length;

}

}

3. Стили (style):

- Определяет CSS-стили для визуального оформления слайдера.

- Включает стили для обертки слайдера, изображения и кнопок управления.

Данный компонент реализует следующую функциональность:

1. Слайдер изображений: Компонент отображает несколько изображений в виде слайдера. Пользователь может переключаться между изображениями, нажимая на кнопки "Предыдущее" и "Следующее".

2. Управление состоянием слайдера: Компонент использует данные (data) для хранения массива изображений и текущего индекса отображаемого изображения. Вычисляемое свойство (computed) currentImage возвращает текущее изображение для отображения.

3. Методы для навигации по слайдеру: Компонент определяет два метода - prevImage() и nextImage(), которые позволяют переключаться на предыдущее и следующее изображение соответственно.

4. Стилизация компонента: Компонент использует CSS-стили для визуального оформления слайдера, включая размещение и отображение изображений, а также стили для кнопок управления.

Этот компонент можно использовать в других частях вашего Vue.js-приложения, чтобы отображать слайдер изображений.

1. **Заключение:**

Проект выполнен в соответствии с поставленными задачами. В процессе его выполнения были получены новые знания в области фронтенд разработки.

1. **Приложения:**
2. **Приложение 1: main.js**

import Vue from 'vue'

import VueRouter from 'vue-router'

import 'bootstrap/dist/css/bootstrap.css'

import 'bootstrap-icons/font/bootstrap-icons.css'

Vue.use(VueRouter)

import HomeComponent from './components/HomeComponent.vue'

import CommutationComponent from './components/CommutationComponent.vue'

import PacketComponent from './components/PacketComponent.vue'

import ContactComponent from './components/ContactComponent.vue'

const routes = [

{ path: '/', component: HomeComponent },

{ path: '/commutationComponent', component: CommutationComponent },

{ path: '/packetComponent', component: PacketComponent },

{ path: '/contactComponent', component: ContactComponent }

]

const router = new VueRouter({

routes,

mode: 'history'

})

new Vue({

router

}).$mount('#app')

1. **Приложение 2: App.vue**

<template>

<div id="app">

<NavbarComponent></NavbarComponent>

<div class="container mt-5">

<router-view></router-view>

</div>

</div>

</template>

<script>

import NavbarComponent from './components/NavbarComponent.vue'

export default {

name: 'App',

components: {

NavbarComponent

}

}

</script>

<style>

</style>

1. **Приложение 3: CommutationComponent.vue**

<template>

<div>

<h2>Способы коммутации</h2>

<p>В цифровых сетях существует несколько основных способов коммутации:</p>

<ul>

<li>Коммутация каналов</li>

<li>Коммутация сообщений</li>

<li>Коммутация пакетов</li>

</ul>

<p>Каждый из этих способов имеет свои особенности и область применения.</p>

</div>

</template>

<script>

export default {

name: 'CommutationComponent'

}

</script>

1. **Приложение 4: ContactComponent.vue**

<template>

<div>

<h2>Свяжитесь с нами</h2>

<form @submit.prevent="submitForm">

<div class="mb-3">

<label for="name" class="form-label">Имя</label>

<input type="text" class="form-control" id="name" v-model="name" required>

</div>

<div class="mb-3">

<label for="email" class="form-label">Email</label>

<input type="email" class="form-control" id="email" v-model="email" required>

</div>

<div class="mb-3">

<label for="message" class="form-label">Сообщение</label>

<textarea class="form-control" id="message" rows="3" v-model="message" required></textarea>

</div>

<button type="submit" class="btn btn-primary">Отправить</button>

</form>

<div class="modal fade" id="successModal" tabindex="-1" aria-labelledby="successModalLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="successModalLabel">Спасибо!</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

Ваше сообщение успешно отправлено.

</div>

</div>

</div>

</div>

</div>

</template>

<script>

/\* eslint-disable no-undef \*/

import 'bootstrap/dist/css/bootstrap.css'

import 'bootstrap/dist/js/bootstrap.js'

export default {

name: 'ContactComponent',

data() {

return {

name: '',

email: '',

message: ''

}

},

methods: {

submitForm() {

// Проверка валидности полей

if (!this.name || !this.email || !this.message) {

alert('Пожалуйста, заполните все поля формы.');

return;

}

// Отправка формы (можно добавить логику отправки на сервер)

this.showSuccessModal();

this.clearForm();

},

showSuccessModal() {

// Показ модального окна об успешной отправке

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

const modal = new bootstrap.Modal(successModal);

modal.show();

},

clearForm() {

// Очистка полей формы

this.name = '';

this.email = '';

this.message = '';

}

}

}

</script>

1. **Приложение 5: HomeComponent.vue**

<template>

<div>

<h1>Добро пожаловать на сайт "Разновидности способов коммутации в цифровых сетях"</h1>

<p>Здесь вы можете найти информацию о различных способах коммутации в цифровых сетях.</p>

</div>

</template>

<script>

export default {

name: 'HomeComponent'

}

</script>

1. **Приложение 6: ImageSliderComponent.vue**

<template>

<div class="image-slider">

<div class="slider-wrapper">

<img :src="currentImage" alt="Slide Image">

</div>

<div class="slider-controls">

<button class="btn btn-primary" @click="prevImage">

<i class="bi bi-chevron-left"></i>

</button>

<button class="btn btn-primary" @click="nextImage">

<i class="bi bi-chevron-right"></i>

</button>

</div>

</div>

</template>

<script>

export default {

name: 'ImageSliderComponent',

data() {

return {

images: [

'https://via.placeholder.com/800x400',

'https://via.placeholder.com/800x400/0000FF',

'https://via.placeholder.com/800x400/FF00FF'

],

currentIndex: 0

}

},

computed: {

currentImage() {

return this.images[this.currentIndex];

}

},

methods: {

prevImage() {

this.currentIndex = (this.currentIndex - 1 + this.images.length) % this.images.length;

},

nextImage() {

this.currentIndex = (this.currentIndex + 1) % this.images.length;

}

}

}

</script>

<style>

.image-slider {

position: relative;

max-width: 800px;

margin: 0 auto;

}

.slider-wrapper {

width: 100%;

height: 400px;

overflow: hidden;

}

.slider-wrapper img {

width: 100%;

height: 100%;

object-fit: cover;

}

.slider-controls {

position: absolute;

top: 50%;

transform: translateY(-50%);

width: 100%;

display: flex;

justify-content: space-between;

padding: 0 20px;

}

.slider-controls button {

font-size: 24px;

}

</style>

1. **Приложение 7: NavbarComponent.vue**

<template>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">

<div class="container">

<a class="navbar-brand" href="#">Способы коммутации</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

<li class="nav-item">

<router-link to="/" class="nav-link">Главная</router-link>

</li>

<li class="nav-item">

<router-link to="/commutation" class="nav-link">Коммутация</router-link>

</li>

<li class="nav-item">

<router-link to="/packet" class="nav-link">Коммутация пакетов</router-link>

</li>

<li class="nav-item">

<router-link to="/contact" class="nav-link">Контакты</router-link>

</li>

</ul>

</div>

</div>

</nav>

</template>

<script>

export default {

name: 'NavbarComponent'

}

</script>

1. **Приложение 8: PacketComponent.vue**

<template>

<div>

<h2>Коммутация пакетов</h2>

<p>Коммутация пакетов является одним из наиболее распространенных способов коммутации в современных цифровых сетях. Она основана на передаче данных в виде небольших фрагментов (пакетов), которые могут быть маршрутизированы по разным путям в сети.</p>

<p>Преимущества коммутации пакетов:</p>

<ul>

<li>Эффективное использование пропускной способности сети</li>

<li>Возможность динамической маршрутизации</li>

<li>Масштабируемость и отказоустойчивость</li>

</ul>

</div>

</template>

<script>

export default {

name: 'PacketComponent'

}

</script>