Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

ОТЧЕТ
По лабораторной работе №2

«*JavaScript* и *DOM*-манипуляции»

Выполнил ст. гр. 220601
А. А. Чубса

Проверил А. Л. Гончаревич

Минск 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В рамках лабораторной работы исследуются язык *JavaScript* и *DOM* манипуляции. *JavaScript* (*JS*) – высокоуровневый язык программирования, который используется для создания динамических и интерактивных веб-страниц. Он разработан Бренданом Эйхом в 1995 году и стал неотъемлемой частью веб-разработки. *JavaScript* позволяет изменять содержимое *HTML*-документа, управлять стилями *CSS* в реальном времени, обрабатывать пользовательские события и взаимодействовать с сервером без перезагрузки страницы.

Одной из ключевых особенностей *JavaScript* является его возможность работать с объектной моделью документа (*DOM, Document Object Model*). *DOM* являет собой программный интерфейс, который представляет структуру *HTML*-документа в виде дерева объектов, где каждый *HTML*-элемент является узлом. С помощью *JavaScript* разработчики могут изменять, добавлять или удалять элементы, изменять их атрибуты и стили, реагировать на пользовательские события, такие как клики, наведение курсора или ввод данных.

Современные версии *JavaScript* включают поддержку *ES*6 и новее, которые содержат улучшенные возможности, а именно стрелочные функции, модули, асинхронное программирование и классы. *JavaScript* активно используется для создания веб-приложений с применением фреймворков и библиотек *React, Angular* и *Vue.js*.

Взаимодействие *HTML, CSS* и *JavaScript* делает веб-страницы не только структурированными, но и стилизованными, динамичными и интерактивными, что является основой современного веб-развития.

1 ПОСТАНОВКА ЗАДАЧИ

Необходимо в созданный ранее сайт добавить модификации согласно варианту:

– функциональность «*Lazy loading*», которая загружает изображения только при прокрутке до них с помощью *JavaScript*;

– функциональность переключения между различными темами оформления на веб-странице с использованием *JavaScript*.

Основным инструментарием исходя из названия лабораторной работы являются *DOM* (*Document Object Model*) и функции *JavaScript*.

После завершения процесса разработки сайта его необходимо тщательно протестировать, чтобы убедиться в корректности работы всех функциональных элементов. Важно проверить правильность отображения контента, удобство нажатий кнопок, адаптивность модификаций сайта на экранах различных размеров, включая мобильные устройства и планшеты. После этого следует провести оптимизацию кода и проверить его на валидность, устранив возможные ошибки и недочёты.

В ходе разработки необходимо продолжить использовать систему управления версиями *Git*, что позволит вести контроль над процессом разработки, отслеживать изменения и эффективно работать в команде. Репозиторий проекта размещен на платформе *GitHub* для хранения кода и удобного взаимодействия с другими разработчиками. После завершения всех этапов сайт необходимо опубликовать в сети, используя сервис *GitHub Pages*, чтобы обеспечить к нему доступ пользователей и продемонстрировать результат проделанной работы.

2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Выбор метода зависит от требований к приложению, частоты обновления данных и архитектуры. *AJAX* и *WebSockets* - наиболее распространенные решения для динамического обновления данных на веб-странице, однако для решения данной проблемы есть и другие способы, вот некоторые из них:

– *Ajax* — позволяет периодически запрашивать данные;

– *WebSockets* — обеспечивают двустороннюю связь между клиентом и сервером, позволяя серверу отправлять данные в реальном времени;

– *Server-Sent Events (SSE)* — позволяет серверу отправлять обновления на клиент через *HTTP* (это полезно для приложений, где сервер должен регулярно отправлять данные);

– *setInterval* — позволяет обновлять данные с фиксированным интервалом;

– *React* — позволяет использовать хуки *useEffect* и *useState*.

DOM (*Document Object Model*) представляет собой структуру, которая позволяет взаимодействовать с элементами веб-страницы через *JavaScript*. Каждый элемент *HTML*-документа становится объектом, который можно изменять или манипулировать с ним с помощью методов *DOM*.

Аналогично для автоматического обновления данных на странице можно использовать события, такие как:

– *click* — срабатывает при клике на элемент;

– *change* — срабатывает при изменении значения в поле ввода;

– *keydown* — срабатывает при нажатии клавиши, что может использоваться для добавления элемента по клавише *Enter*.

Этот подход помогает эффективно реализовать автоматическое обновление данных, улучшая взаимодействие с пользователем.

3 ХОД РАБОТЫ

Согласно заданию лабораторной работы, необходимо добавить функциональность «*Lazy loading*», реализация которой показана в приложении А, загружает изображения только при прокрутке до них с помощью *JavaScript*. Функция, приведённая в данном коде, использует *IntersectionObserver* для ленивой загрузки изображений. При загрузке страницы она находит все изображения с классом *Lazy* и отслеживает их появление в области видимости. Когда изображение становится полностью загруженным в окне браузера с помощью *IntersectionObserver*, его атрибут *src* обновляется с использованием значения атрибута *data-src*, и класс *Lazy* удаляется. После этого наблюдатель перестаёт отслеживать это изображение. Частично прогруженные изображения представлены на рисунке 3.1.

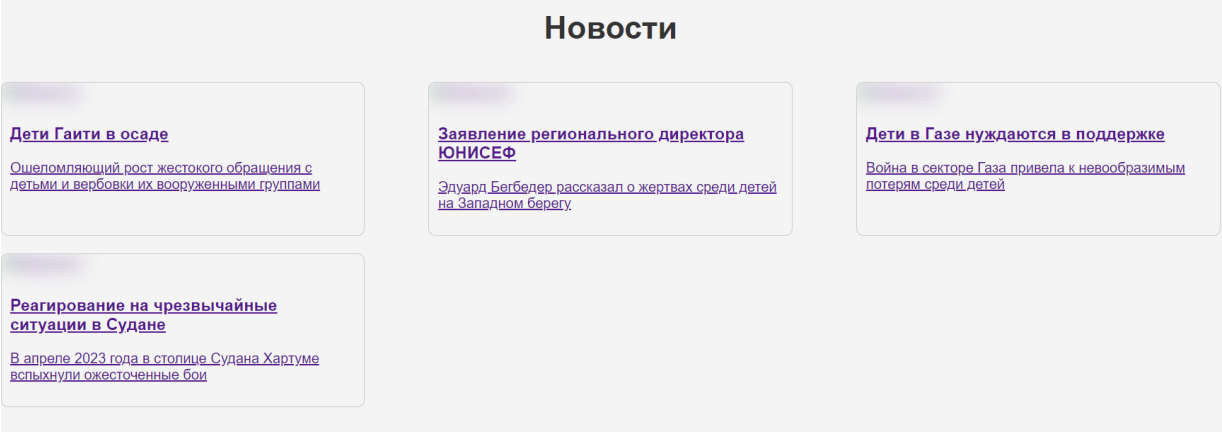

Рисунок 3.1 – Частично загруженные изображения

Изображение загружается через полсекунды после попадания его в зону видимости пользователя.

Для реализации функциональности переключения между различными темами оформления на веб-странице с использованием *JavaScript* необходимо организовать процесс изменения стилей страницы при нажатии кнопки, которая будет менять фон сайта. Для этого использовалось событие, такое как *click*, и методы работы с *DOM* для динамического изменения классов или стилей на странице.

На главной странице с кнопкой «Настройки», при нажатии на которую происходит смена фона, следует добавить обработчик события. Когда пользователь нажимает на кнопку, *JavaScript* изменяет класс и напрямую обновляет стили элементов страницы, таких как фон или другие визуальные характеристики, чтобы переключить тему оформления. Начальный фон главной страницы с кнопкой «Настройки» проиллюстрирован на рисунке 3.2.

Рисунок 3.2 – Главная страница сайта со стандартным фоном

После нажатия кнопки «Настройки» пользователь увидит изменение логотипа и фона как показано на рисунке 3.3.

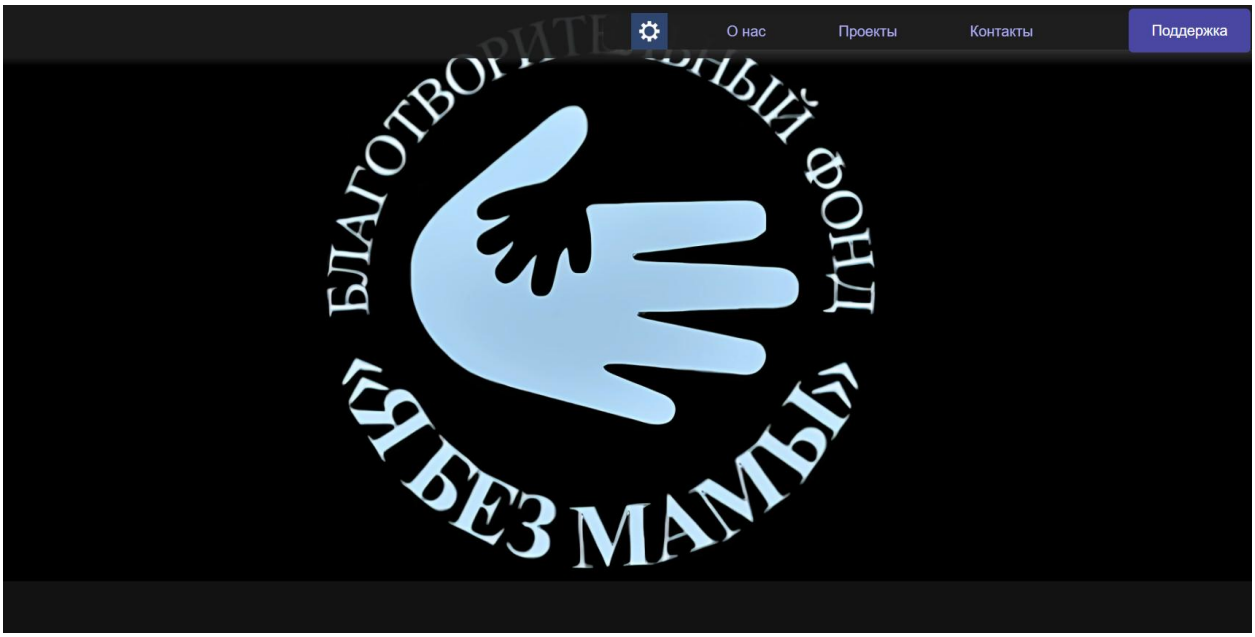

Рисунок 3.3 – Изменение фона главной страницы сайта

На других страницах произведены изменения в структуре и визуальном оформлении основных блоков и фона. Обновлена реализация бургер-меню для улучшения взаимодействия с пользователем и удобства навигации на мобильных устройствах. В результате этих изменений, внешний вид страниц стал более современным и адаптированным под различные размеры экранов. Результаты изменения фона и его влияния на общий стиль и восприятие сайта

продемонстрированы на рисунке 3.4. Эти изменения позволили улучшить визуальную составляющую сайта и создать более приятный пользовательский интерфейс.

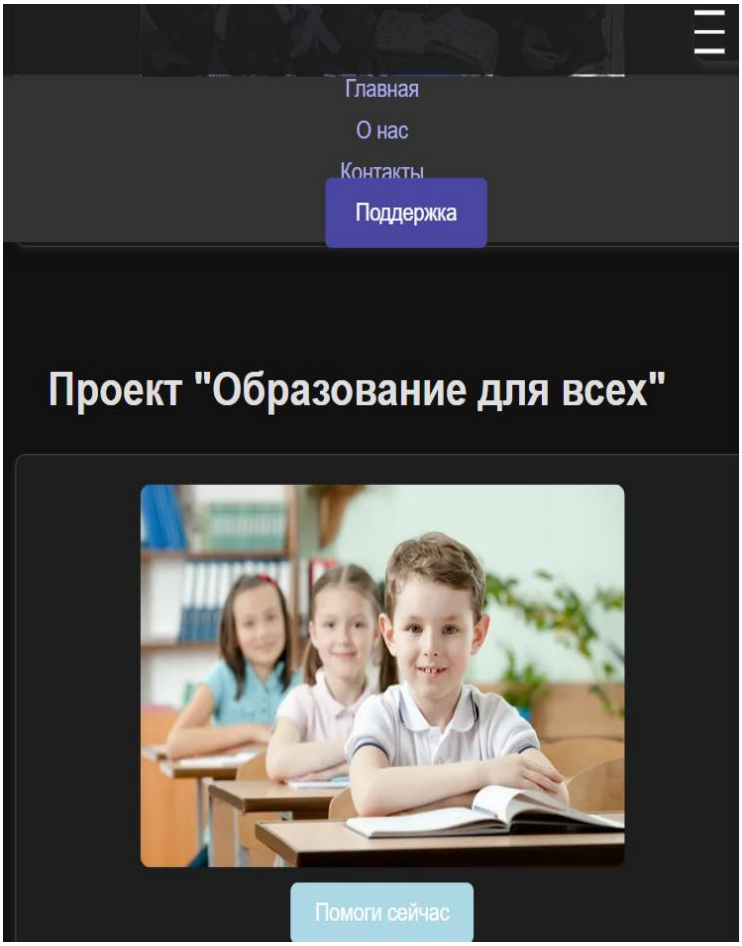

Рисунок 3.4 – Изменения фона
на странице «Проекты»

Функция, приведённая в приложении Б, после загрузки документа находит кнопку для переключения темы, контейнера с логотипом, оформления для темной темы и автоматически обновляет фон и логотип в зависимости от текущей темы (светлой или тёмной). Когда пользователь нажимает кнопку переключения, тема меняется на противоположную, и логотип с фоном обновляются соответственно. Логотипы для светлой темы и для тёмной темы хранятся в переменных с путями к изображениям. При каждом изменении темы она сохраняется в *localStorage*, чтобы при следующем посещении сайта тема сохраняла свое последнее положение. При загрузке страницы проверяется сохранённая тема в *localStorage* и применяется соответствующая тема. Функция *updateLogo* обновляет логотип в зависимости от текущей темы.

При клике на кнопку переключения темы логотип и тема обновляются, а новая тема сохраняется в *localStorage*.

Этот код позволяет динамически переключать темы и обеспечивать сохранение настроек, даже при перезагрузке страницы.

Исходный код веб-сайта расположен в репозитории на *GitHub*. Репозиторий располагается по ссылке *https://github.com/Kerik1234/Lab2-itivp*. Сайт развёрнут на *GitHub Pages* и располагается по адресу *https://kerik1234.github.io/Lab2-itivp*. Главная страница репозитория изображена на рисунке 3.5.

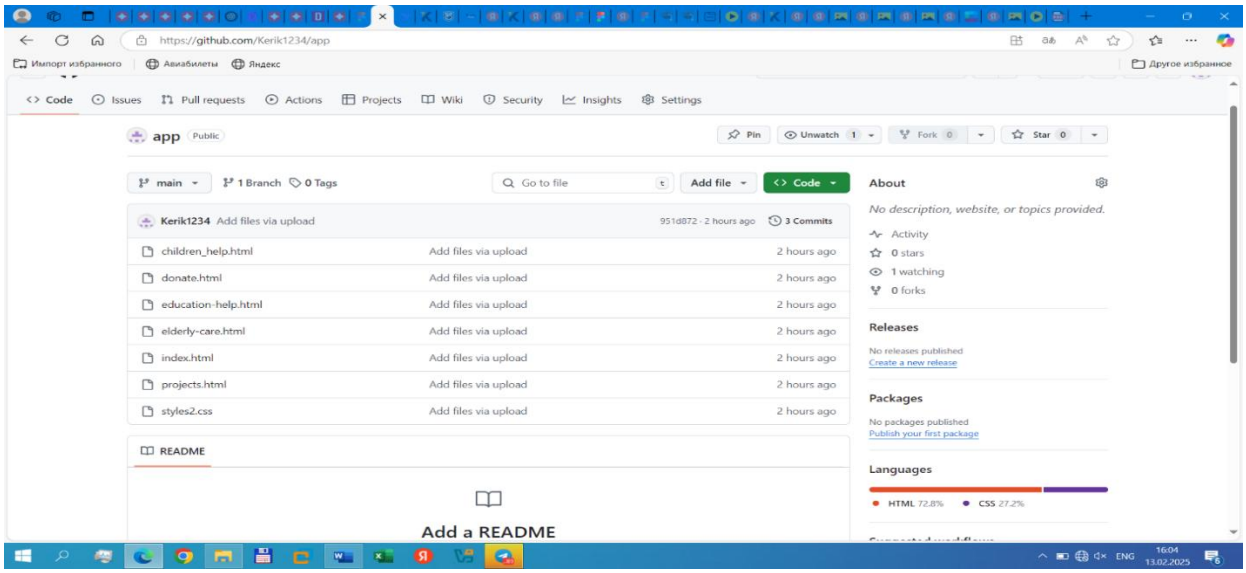

Рисунок 3.5 – Изображение репозитория на *GitHub*

На рисунке 3.6 изображена история действий репозитория.

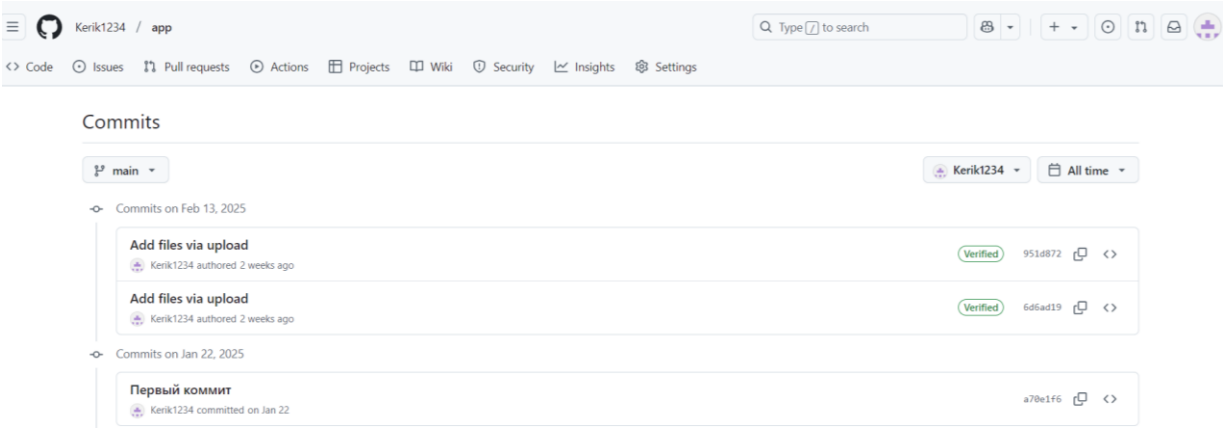

Рисунок 3.6 – История действий репозитория

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы исследовано использование *JavaScript* для динамического изменения веб-страниц и взаимодействия с объектной моделью документа *DOM*. Изучены основные методы манипуляции элементами *HTML*, изменения их стилей и обработки событий пользователя.

Практическое применение *JavaScript* позволило реализовать такие интерактивные элементы, как появление изображений только при наведении на них и продемонстрировало, как можно изменять фон страницы. Важное внимание уделено современным возможностям языка, включая обработку событий, работу с функциями и взаимодействие со структурой *DOM*.

В результате выполнения лабораторной работы изучен *JavaScript* для создания современных веб-приложений. Он помогает внедрять интерактивные элементы, улучшать взаимодействие с пользователем и динамически изменять контент страниц.

ПРИЛОЖЕНИЕ А
(обязательное)
Реализация «*LAZY LOADING*»

```
document.addEventListener("DOMContentLoaded",function (){
let LazyImages=document.querySelectorAll("img.Lazy");
let observer=new IntersectionObserver((entries,observer)=>{
entries.forEach(entry => {
if(entry.isIntersecting){
let img=entry.target;
img.src=img.getAttribute("data-src");
img.classList.remove("Lazy");
observer.unobserve(img);
}
});
});
LazyImages.forEach(img => observer.observe(img));
});
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Реализации изменения фона

```
document.addEventListener("DOMContentLoaded", function () {
const toggleButton = document.getElementById("settings-button");
const body = document.body;
const logo = document.getElementById("logo");
const lightLogo = "assets/dawn.jpg";
const darkLogo = "assets/Pica-enhance-20250301234303.jpg";
function updateLogo() {
if (body.classList.contains("dark-theme")) {
logo.src = darkLogo;
} else {
logo.src = lightLogo;
}
}
if (localStorage.getItem("theme") === "dark") {
body.classList.add("dark-theme");
}
updateLogo();
if (toggleButton) {
toggleButton.addEventListener("click", function () {
body.classList.toggle("dark-theme");
updateLogo();
if (body.classList.contains("dark-theme")) {
localStorage.setItem("theme", "dark");
```

```
} else {
localStorage.setItem("theme", "light");
}
});
}
});
```