Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Институт информационных технологий

Кафедра «Микропроцессорные системы и сети»

**КУРСОВАЯ РАБОТА**

**по дисциплине**

**«Анимационная графика»**

**НА ТЕМУ:**

**«Анимированный интерфейс web-страницы «Bayan»**

Слушатель гр 10423 О. М. Евстратова

(дата, подпись)

Руководитель d П. С. Львов,И. В,Кашникова

(дата, подпись)

МИНСК

2023

**Пример оформления листа задания**

Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Институт информационных технологий

Кафедра «Микропроцессорные системы и сети»

|  |  |  |
| --- | --- | --- |
|  | УТВЕРЖДАЮ | |
|  | Заведующий кафедрой МПСС | |
|  | *И.В.Кашникова* | |
|  |  | |
|  | (подпись) | |
|  |  | 2022 г. |

ЗАДАНИЕ

по курсовой работе

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Слушателю | | *Евстратовой Ольги Михайловны* | | | | | |
| Группа | | *10423* | | | | | |
| 1. Тема работы | | *Анимированный интерфейс web-страницы «Bayan»* | | | | | |
| 2. Срок сдачи студентом законченной работы | | | | | | *12.04.2023* | |
| 3. Исходные данные к работе | | | |  | | | |
| Web-страница(сайт) должен иметь простой понятный сюжет с использованием 2d объектов простых геометрических форм, цветовая гамма – яркая, но не резкая. | | | | | | | |
| 4. Перечень вопросов, которые подлежат разработке | | | | | | | |
|  | *Определение целей и задач* | | | | | | |
|  | *Анализ существующих разработок по выбранной теме* | | | | | | |
|  | *Идея и сюжет разрабатываемого приложения* | | | | | | |
|  | *Разработка макета* | | | | | | |
|  | *Графическое оформление и анимация* | | | | | | |
|  | *Программное обеспечение* | | | | | | |
|  | *Технологии анимации* | | | | | | |
| 6. Дата выдачи задания | | | | | *01.04.2023* | | |
| 7. Календарный график работы на весь период работы (с обозначением сроков выполнения и трудоемкости отдельных этапов): | | | | | | | |
| *01.04.2023* | | |  | | | | *50%* |
| *12.04.2023* | | |  | | | | *80%* |
| Руководитель | | |  | | | | *П. С.Львов,*  *И. В. Кашникова* |
|  | | | (подпись) | | | |  |
| Задание принял к исполнению | | | *01.04.2023* | | | |  |
|  | | | (дата) | | | | (подпись слушателя) |

**СОДЕРЖАНИЕ**

Введение 4

* Техническая концепция 5
* Определение целей и задач 5
* Анализ существующих разработок по выбранной теме 5
* Идея и сюжет разрабатываемого приложения 5
* Проектирование 6
* Разработка макета 6
* Графическое оформление и анимация 7

3. Разработка 10

3.1. Программное обеспечение 10

3.2. Тестирование 19

Заключение 21

Список использованных источников 22

Приложение 1 23

**ВВЕДЕНИЕ**

Целью данной курсовой работы является разработка анимационной web-страницы «Bayan».

Цель работы заключалась в том, чтобы создать анимированные мини истории на странице, который рассказывают кратко и в юмористическом стиле о проблемах с которыми сталкиваются молодые ИТ специалисты.

Страница должна быть простой и удобной, чтобы люди с разным уровнем подготовки могли ей воспользоваться, изучить и получить приятные эмоции. Удобные и интуитивно очевидные кнопки.

Для создания данного проекта использованы следующие технологии: HTML, CSS, JavaScript. HTML использован для разметки страницы, CSS — для создания стилей HTML-страницы, JavaScript — для создания динамических элементов.

* **техническая концепция**

**1.1. Цели создания web-страницы с анимированными историями**

Цель работы заключалась в том, чтобы создать анимированные мини истории на странице, который рассказывают кратко и в юмористическом стиле о проблемах с которыми сталкиваются молодые ИТ специалисты.

Страница должна быть простой и удобной, чтобы люди с разным уровнем подготовки могли ей воспользоваться и изучить. Удобные и интуитивно очевидные кнопки. Приятные и налаживающие на положительный лад картинки.

**1.2. Анализ существующих разработок по выбранной тематике**

Анализ схожих сайтов(страниц) может быть полезным. Проанализировав качественные сайты(страницы), можно получить новые идеи, узнать о маленьких хитростях, специфических способах продвижениях и фишках, которые повысят интерес к проекту. Можно выявить ошибки и недочеты, которых лучше избегать при создании собственного web-сайта(web-страницы).

Некоторые критерии целесообразно скомпоновать в один блок. Так, например, в блок «Дизайн» входят следующие критерии: красочная GIF-анимации; оптимизация сайта для различных браузеров, оптимизация сайта для мобильных устройств, наличие целостной структуры.

В блок «Удобство использования» входят следующие критерии: возможность запускать автоматическое пролистывание слайдов, возможность пролистывать слайды вперёд-назад, запускать анимации. Интуитивно удобные кнопки.

**1.3. Идея и сюжет разрабатываемого приложения**

При создании web-страницы основное внимание должно быть уделено таким его составляющим как содержание (контент), структура (навигация) и дизайн (оформление).

Целью данной работы являлась разработка анимационной web-страницы «Bayan».

Цель работы заключалась в том, чтобы создать анимированные мини истории на странице, который рассказывают кратко и в юмористическом стиле о проблемах с которыми сталкиваются молодые ИТ специалисты.

В курсовом проекте используется анимация на базе аккордеона с несколькими открытыми вкладками

**2. Проектирование**

**2.1. Разработка макета**

Макет построен на базеаккордеона с несколькими открытыми вкладками:



Рисунок 1.1 —структура анимационной web-страницы «Bayan».

Внутри аккордеона размешены слайды с кнопками запускающими анимацию:



Рисунок 1.2 —структура вкладки аккордеона web-страницы «Bayan».

На странице очень много «воздуха» — свободного пространства. Оно помогает ориентироваться на странице.

Концепцию web-сайта подчеркивает и его цветовая гамма. Страница создана яркой, чтобы настроить человека на положительный лад.

**2.2. Графическое оформление и анимация**

Для анимации использовались кнопки (яркого цвета), слайдеры, галерея с фотографиями для каждой истории своя.

Гибкие, приятные и удобный в использовании слайдеры. Они интуитивно понятно себя ведут, адаптивны, замечательно себя чувствует на любых платформах и разрешениях.

Для работы со слайдерами подключались:

Стили (css):

<link rel="stylesheet" href="./css/style.css" />

JavaScript(js):

<script src="./script/script.js"></script>

Содержимым слайдера может выступать что угодно. Картинки, текст, списки, видео, но оно обязательно должно быть обернуто в контейнер (Например <div>

<div class="acor-container">

        <input type="checkbox" name="chacor" id="chacor1" checked="checked" />

        <label for="chacor1">Первые собеседования</label>

        <div class="acor-body" id="interview">

          <!-- <img class="my-photo" src="assets/img/n1.png" alt="photo1">  -->

          <div id="gallery">

            <div class="photos">

                <!-- id для поиска картинки чтобы показать и скрыть её -->

                <!-- data-container для того чтобы найти нужную галерею картинок через дата атрибуты -->

                <!-- data-item чтобы узнать номер картинки -->

                <!-- тк картинок и галерей много и чтобы не писать фунцию для каждой картинки и галереи -->

                <!-- будем использовать дата атрибуты -->

                <!-- data-active чтобы  вычислять активную катртинку, ту которая сейчас показывается -->

                <!-- data-animate чтобы узнать есть ли анимация в данный момент -->

              <img

                id="img\_11"

                data-container="1"

                data-item="1"

                data-active="true"

                data-animate="false"

                class="my-photo"

                src="assets/img/nlo1.png"

              />

              <img

                id="img\_12"

                data-container="1"

                data-item="2"

                data-active="false"

                data-animate="false"

                class="my-photo"

                src="assets/img/nlo2.png"

              />

              <img

                id="img\_13"

                data-container="1"

                data-item="3"

                data-active="false"

                data-animate="false"

                class="my-photo"

                src="assets/img/nlo3.png"

              />

              <img

                id="img\_14"

                data-container="1"

                data-item="4"

                data-active="false"

                data-animate="false"

                class="my-photo"

                src="assets/img/nlo4.png"

              />

            </div>

            <div class="buttons">

              <div class="buttons\_move">

                <!-- data-button чтобы узнать номер галереи с которой в данный момент работает пользователь -->

                <button id="prev\_1" data-button="1" class="prev">Назад</button>

                <button id="center\_1" data-button="1" class="center">

                  Анимация

                </button>

                <button id="next\_1" data-button="1" class="next">Вперед</button>

              </div>

              <div class="button\_auto">

                <!-- data-slide чтобы включать и выключать авто проигрывание -->

                <button

                  id="auto\_1"

                  data-button="1"

                  data-slide="false"

                  class="auto"

                >

                  авто пролистывание

                </button>

              </div>

            </div>

          </div>

»

Одним из анимационных элементов на сайте является кнопки. При наведении на которые был добавлен эффект изменения цвета. Это сразу привлекает внимание и дает статическому объекту движение.



Рисунок 2.1 — Кнопки в состоянии покоя



Рисунок 2.2 — Состояние кнопки при наведении( в центре кнопка «Анимация»)

Следующим элементом анимации является при нажатии на кнопку «Анимация» у каждой истории запускается своя анимация.

1 - увеличение-уменьшение картинки, 2 - вращение картинки, 3 - «дрожь» картинки, 4 – «пульсирование» картинки.



Рисунок 2.3 —Пример. Изменение картинки(анимация) при нажатии кнопка «Анимация».

Так же при нажатии кнопки «авто пролистывание» фотографии меняются автоматически.

На сайте располагается несколько галерей с изображениями, которые создана с помощью функционала JS. При нажатии кнопки «назад» слайды перелистывается на один шаг назад, а при нажатии кнопки «вперёд» слайды перелистывается на один шаг вперёд,

**3. Разработка**

**3.1. Программное обеспечение**

Для написания HTML кода и стилей CSS, а также подключения Java Script использовалась программа **Visual Studio Code**. **Visual Studio Code** (VS Code) - [текстовый редактор](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0), разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для [кроссплатформенной](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) разработки [веб](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5)- и [облачных](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) приложений. Включает в себя [отладчик](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA), инструменты для работы с [Git](https://ru.wikipedia.org/wiki/Git), [подсветку синтакс HYPERLINK "https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%81%D0%B2%D0%B5%D1%82%D0%BA%D0%B0\_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81%D0%B0"и HYPERLINK "https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%81%D0%B2%D0%B5%D1%82%D0%BA%D0%B0\_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81%D0%B0"са](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%81%D0%B2%D0%B5%D1%82%D0%BA%D0%B0_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81%D0%B0), IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, [сочетания клавиш](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%87%D0%B5%D1%82%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BB%D0%B0%D0%B2%D0%B8%D1%88) и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), но готовые сборки распространяются под проприетарной лицензией.

Visual Studio Code основан на Electronи реализуется через веб-редактор Monaco, разработанный для Visual Studio Online.

Visual Studio Code был анонсирован 29 апреля 2015 года компанией Microsoft на [конференции HYPERLINK "https://ru.wikipedia.org/wiki/Build\_(%D0%BA%D0%BE%D0%BD%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D1%8F\_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%D0%B8%D0%BA%D0%BE%D0%B2)"Build](https://ru.wikipedia.org/wiki/Build_(%D0%BA%D0%BE%D0%BD%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%D0%B8%D0%BA%D0%BE%D0%B2)), и вскоре была выпущена [бета-версия](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%82%D0%B0-%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F).

18 ноября 2015 года Visual Studio Code был выпущен под [лицензией MIT](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_MIT), а исходный код был опубликован на GitHub. Анонсирована поддержка расширений.

14 апреля 2016 года Visual Studio Code вышел из стадии [бета-тестирования](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%82%D0%B0-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Visual Studio Code — это [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0). Он имеет многоязычный интерфейс пользователя и поддерживает ряд [языков программиров HYPERLINK "https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA\_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F"а HYPERLINK "https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA\_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F"ния](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), [подсветку синтаксиса](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%81%D0%B2%D0%B5%D1%82%D0%BA%D0%B0_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81%D0%B0), IntelliSense, рефакторинг, [отладку](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B), навигацию по коду, поддержку Git и другие возможности. Многие возможности Visual Studio Code недоступны через [графический интерфейс](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), зачастую они используются через палитру команд или [JSON](https://ru.wikipedia.org/wiki/JSON)-файлы (например, пользовательские настройки). Палитра команд представляет собой подобие [командной строки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8), которая вызывается [сочетанием клавиш](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%87%D0%B5%D1%82%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BB%D0%B0%D0%B2%D0%B8%D1%88).

VS Code также позволяет заменять [кодовую страницу](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) при сохранении документа, [символы перевода стр HYPERLINK "https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4\_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8"о HYPERLINK "https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4\_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8"ки](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8) и язык программирования текущего документа.

С 2018 года появилось расширение Python для Visual Studio Code с открытым исходным кодом. Оно предоставляет разработчикам широкие возможности для редактирования, отладки и тестирования кода.

Также VS Code поддерживает редактирование и выполнение файлов типа «Блокнот Jupyter» (Jupyter Notebook) напрямую «из коробки» без установки внешнего модуля в режиме визуального редактирования и в режиме редактирования исходного кода.

На март 2019 года посредством встроенного в продукт пользовательского интерфейса можно загрузить и установить несколько тысяч расширений только в категории «programming languages» (языки программирования).

Также расширения позволяют получить более удобный доступ к программам, таким как Docker, Git и другие. В расширениях можно найти линтеры кода, темы для редактора и поддержку синтаксиса отдельных языков.

Разработка страниц(сайта) начинается с создания макета. Макет может отличаться от конечного результата: он дает дизайнеру понять, как располагать элементы, какого стиля придерживаться и т. д

По макету начинается создание новой страницы (рисунок 3.1). Для этого в программе VS Code создаем новый документ, сразу сохраняем в формате .html, чтобы программа графически выделяла теги и атрибуты, предупреждала об ошибках, закрывала теги и т.д.



Рисунок 3.1 — Проект, открытый в программе VS Code.

Пишем основу страницы: теги <html>, <head>, <body>, указываем кодировку и подключаем файлы css и javascript.

Код подключения CSS

<html lang="en">

  <head>

    <title>Bayan</title>

    <link rel="icon" href="assets/img/logo.png" />

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <link rel="preconnect" href="https://fonts.gstatic.com" />

    <link

      href="https://fonts.googleapis.com/css2?family=Roboto:wght@100;300;500;700&display=swap"

      rel="stylesheet"

    />

    <link rel="stylesheet" href="./css/style.css" />

  </head>

Пример. Сектора аккордеона.

 <input type="checkbox" name="chacor" id="chacor3" />

        <label for="chacor3">Первые успехи</label>

        <div class="acor-body" id="triumph">

          <div id="gallery">

            <div class="photos">

              <img

                id="img\_31"

                data-container="3"

                data-item="1"

                data-active="true"

                data-animate="false"

                class="my-photo"

                src="assets/img/winner1.png"

              />

              <img

                id="img\_32"

                data-container="3"

                data-item="2"

                data-active="false"

                data-animate="false"

                class="my-photo"

                src="assets/img/winner2.png"

              />

              <img

                id="img\_33"

                data-container="3"

                data-item="3"

                data-active="false"

                data-animate="false"

                class="my-photo"

                src="assets/img/winner3.png"

              />

              <img

                id="img\_34"

                data-container="3"

                data-item="4"

                data-active="false"

                data-animate="false"

                class="my-photo"

                src="assets/img/winner4.png"

              />

            </div>

            <div class="buttons">

              <div class="buttons\_move">

                <button id="prev\_3" data-button="3" class="prev">Назад</button>

                <button id="center\_3" data-button="3" class="center">

                  Анимация

                </button>

                <button id="next\_3" data-button="3" class="next">Вперед</button>

              </div>

              <div class="button\_auto">

                <button

                  id="auto\_3"

                  data-button="3"

                  data-slide="false"

                  class="auto"

                >

                  авто пролистывание

                </button>

              </div>

            </div>

          </div>

Код подключения JavaScript

<script src="./script/script.js"></script>

Функция для кнопки назад

function move\_prev(event) {

  // узнаём, на каком конкретно элементе произошло событие

  //самый глубокий элемент, который вызывает событие, называется целевым элементом, и он доступен через event.target

  const this\_button = event.target;

  // находим номер галереи через дата атрибут button кнопки

  const number\_gallery = this\_button.dataset.button;

  // получаем все изображения галереи

  const images = document.querySelectorAll(

    `img[data-container="${number\_gallery}"]`

  );

  let current\_image = null;

  // находим текущую картинку через дата атрибут active

  for (let image of images) {

    if (image.dataset.active == "true") {

      current\_image = image.dataset.item;

    }

  }

  // вычисляем id картинки, которую надо скрыть

  // (id состоит из слова img, номера галереи и номера картинки)

  const id\_hide\_image = "img\_" + String(number\_gallery) + String(current\_image);

  // находим картинку

  const hide\_image = document.getElementById(id\_hide\_image);

  // прячем текущую картину

  hide\_image.style.display = "none";

  // выставляем дата атрибут активной картинки в false, потому что это уже не активная картинка

  hide\_image.dataset.active = "false";

  // уменьшаем счётчик активной картинки на единицу, чтобы показать предыдущую картинку

  current\_image--;

  if (current\_image < 1) {

    // переменная если счётчик меньше единицы, то делаем его равным длине массива картинок

    // тк отсчёт картинок начинается с 1

    current\_image = images.length;

  }

  // вычисляем id картики, которую надо показать

  // (id состоит из слова img, номера галереи и номера картинки)

  const id\_show\_image = "img\_" + String(number\_gallery) + String(current\_image);

  // находим нужную картинку по id

  const show\_image = document.getElementById(id\_show\_image);

  // делаем её видимой

  show\_image.style.display = "block";

  // выставляем дата атрибут активности в true

  show\_image.dataset.active = "true";

  // если картинки в данный момент перелистываются сами

  // то сбраываем необходимый счётчик, чтобы быстро не перелистывались

  if (this\_button.dataset.slide == "true") {

    switch (number\_gallery) {

      case "1":

        clearInterval(timer\_1);

        break;

      case "2":

        clearInterval(timer\_2);

        break;

      case "3":

        clearInterval(timer\_3);

        break;

      case "4":

        clearInterval(timer\_4);

        break;

    }

  }

}

Функция для кнопки вперёд.

// функция для кнопки вперёд

function move\_next(event) {

  // узнаём, на каком конкретно элементе произошло событие

  //самый глубокий элемент, который вызывает событие, называется целевым элементом, и он доступен через event.target

  const this\_button = event.target;

  const number\_gallery = this\_button.dataset.button;

  // находим все картинки нужной галереи через дата атрибут кнопки

  const images = document.querySelectorAll(

    `img[data-container="${number\_gallery}"]`

  );

  let current\_image = null;

  // вычисляем активную картинку

  for (let image of images) {

    if (image.dataset.active == "true") {

      current\_image = image.dataset.item;

    }

  }

  // вычисляем id активной на данный момент картинки чтобы её скрыть

  // (id состоит из слова img, номера галереи и номера картинки)

  const id\_hide\_image = "img\_" + String(number\_gallery) + String(current\_image);

  // находим картинку которую надо скрыть

  const hide\_image = document.getElementById(id\_hide\_image);

  // прячем картику

  hide\_image.style.display = "none";

  // выставляем дата атрибут активности картинки в false

  hide\_image.dataset.active = "false";

  // увеличиваем счётчик активной картинки, чтобы показать следующую картинку

  current\_image++;

  // если номер картинки больше чем длина массива с картинками

  // то текущий нормер ставим на единицу

  if (current\_image > images.length) {

    current\_image = 1;

  }

  // вычисляем id картинки которую надо показать

  // (id состоит из слова img, номера галереи и номера картинки)

  const id\_show\_image = "img\_" + String(number\_gallery) + String(current\_image);

  // находим картинку, которую надо показать

  const show\_image = document.getElementById(id\_show\_image);

  // выставляем её значение display в block чтобы показать картинку

  show\_image.style.display = "block";

  // ствим дата атрибут активности картинки в true, потому что теперь эта картинка активна

  show\_image.dataset.active = "true";

  // если в данный момент происходит движение слайдера, то надо сбросить таймер

  // чтобы быстро не перелистнулось на следующую картинку

  if (this\_button.dataset.slide == "true") {

    switch (number\_gallery) {

      case "1":

        clearInterval(timer\_1);

        break;

      case "2":

        clearInterval(timer\_2);

        break;

      case "3":

        clearInterval(timer\_3);

        break;

      case "4":

        clearInterval(timer\_4);

        break;

    }

  }

}

Функции анимации картинок

// функция анимации картинок

// включает анимацию или выключает

function animation(event) {

  // узнаём, на каком конкретно элементе произошло событие

  //самый глубокий элемент, который вызывает событие, называется целевым элементом, и он доступен через event.target

  const this\_button = event.target;

  const number\_gallery = this\_button.dataset.button;

  // получаем все картинки из текущей галереи

  const images = document.querySelectorAll(

    `img[data-container="${number\_gallery}"]`

  );

  // перебираем картинки и смотрим есть ли на них сейчас анимация

  // если нет анимации, то к картинкам добавляется класс с анимацией

  // если есть анимация, то класс удаляется

  // класс добавляется или удаляется только к тем картинкам, кнопка какой галереи была задействована

  for (let image of images) {

    // проверяем есть ли анимация

    // если есть

    if (image.dataset.animate == "true") {

      // выставляем дата атрибут анимации в false

      image.dataset.animate = "false";

      // в зависимости от номера галереи удаляем соответствующий класс с анимацией

      switch (number\_gallery) {

        case "1":

          image.classList.remove("puls");

          break;

        case "2":

          image.classList.remove("rotation");

          break;

        case "3":

          image.classList.remove("shaking");

          break;

        case "4":

          image.classList.remove("jumping");

          break;

      }

    } else {

      // если анимации не было

      // выставляем дата атрибут показывающий есть ли анимация в true

      image.dataset.animate = "true";

      // добавляем класс анимации к соответствующим картинкам

      switch (number\_gallery) {

        case "1":

          image.classList.add("puls");

          break;

        case "2":

          image.classList.add("rotation");

          break;

        case "3":

          image.classList.add("shaking");

          break;

        case "4":

          image.classList.add("jumping");

          break;

      }

    }

  }

}

Функции для запуска-остановки слайд-шоу

// функция для запуска и остановки слайд шоу

function make\_timer(event) {

  // узнаём, на каком конкретно элементе произошло событие

  //самый глубокий элемент, который вызывает событие, называется целевым элементом, и он доступен через event.target

  const this\_button = event.target;

  const number\_gallery = this\_button.dataset.button;

  // проверяем идёт ли сейчас слайд шоу соответсвующей галереи через атрибут slide кнопки

  // автоматического проигрывания нет

  if (this\_button.dataset.slide == "false") {

    // устанавливаем атрибут slide кнопки в true потому что начинаем проигрывание

    this\_button.dataset.slide = "true";

    // выбираем нужную галерею, чтобы остальные не были задействованы

    // setInterval позволяет вызывать функцию регулярно, повторяя вызов через определённый интервал времени

    switch (number\_gallery) {

      case "1":

        timer\_1 = setInterval(function () {

            // вызываем функцию которая пролистывает картинки

          show\_slides(number\_gallery);

        }, set\_timer);

        break;

      case "2":

        timer\_2 = setInterval(function () {

            // вызываем функцию которая пролистывает картинки

          show\_slides(number\_gallery);

        }, set\_timer);

        break;

      case "3":

        timer\_3 = setInterval(function () {

            // вызываем функцию которая пролистывает картинки

          show\_slides(number\_gallery);

        }, set\_timer);

        break;

      case "4":

        timer\_4 = setInterval(function () {

            // вызываем функцию которая пролистывает картинки

          show\_slides(number\_gallery);

        }, set\_timer);

        break;

    }

  } else {

    // если проигрывание сейчас происходит, то останавливаем его

    // нужный атрибут у нужной кнопки ставим в false

    this\_button.dataset.slide = "false";

    // выбираем нужную галерею через номер чтобы остальные не были задействованы

    // остановить дальнейшее выполнение функции setInterval, которая отвечает за автоматическое проигрывание,

    // необходимо вызвать clearInterval(timerId)

    switch (number\_gallery) {

      case "1":

        clearInterval(timer\_1);

        break;

      case "2":

        clearInterval(timer\_2);

        break;

      case "3":

        clearInterval(timer\_3);

        break;

      case "4":

        clearInterval(timer\_4);

        break;

    }

  }

}

И прочие многие функции.

**3.2 Тестирование**

Кроссбраузерность — это способность веб-ресурса отображаться одинаково и работать во всех популярных браузерах, без перебоев в функционировании и ошибок в верстке, а также с одинаково корректной читабельностью контента. Из-за стремительного развития технологий, сайт рекомендуется делать кроссбраузерным только по отношению к новейшим версиям браузеров [5].

Тестируя веб-ресурс на кроссбраузерность, необходимо применять самые новые версии популярнейших браузеров:

* Google Chrome
* Opera
* Internet Explorer

Было проверено в данных браузерах. Работа удовлетворяет требованиям.

Юзабилити-тестирование — исследование, выполняемое с целью определения, удобен ли пользовательский интерфейс или устройство для его предполагаемого применения.

Методы проведения:

Наблюдение — самый простой метод: респондент работает в обычном режиме, модератор смотрит и анализирует его действия. В завершении респондент заполняет опросник и делится впечатлениями о сайте. Метод хорош тем, что пользователь взаимодействует с сайтом естественно, на него не давят окружающие обстоятельства.

Мысли вслух. Этот популярный метод был предложен Якобом Нильсеном. Суть его заключается в том, что пользователь все свои действия проговаривает вслух. Однако при подобном поведении респонденты начинают более вдумчиво подходить к выполнению заданий и часть естественности теряется.

Теневой метод. Одновременно работают три участника: респондент, модератор и эксперт. Респондент свободно и независимо выполняет задания, модератор фиксирует, эксперт комментируют действия респондента.

Ретроспектива. Это метод, который сочетает в себе наблюдение и «мысли вслух». Сначала респондент выполняет задания, затем — смотрит видеозапись своих действия и комментирует их. Основной недостаток — значительное увеличение времени тестирования.

Валидация — это проверка чего-то на соответствие стандартам (требованиям). В контексте вебмастеров валидация сайта означает проверка его кода на корректность и ошибки. Другими словами, не должны встречаться непонятные теги, конструкций и естественно не должно быть ошибок html.

1. Валидация сайта влияет прежде всего на отображение сайта в браузерах. У каждого браузера есть свои механизмы обработки html кода. Если есть ошибки в верстке, то браузер сам их устраняет, и пользователь видит отображение сайта в корректном виде.

2. Валидация также можно повлиять и на траст сайта для поисковых систем, а значит и на позиции сайта тоже. Для поисковых систем html код это все, что видит поисковый робот. Если там есть ошибки, то он просто неправильно интерпретирует некоторые элементы, которые иногда могут быть критичными. Поисковая система не будет исправлять ошибки, как это делают браузеры. В итоге позиции сайта могут сильно пострадать из-за таких ошибок.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы была разработана анимация для раз web-страницы «Bayan».

Она обладает простым и привлекательным дизайном и удобной системой навигации. Содержит иллюстрации, кнопки, навигации.

Работа велась в программе **Visual Studio Code** (VS Code) . Были использованы различные динамические эффекты (кнопки, галерея, слайдер изображений). Анимация и динамически эффекты выполнялись средствами языка разметки CSS и языка программирования JavaScript.

В ходе разработки сайта были изучены основные теги html, стили css, язык JavaScript. В настоящее время в сети Интернет можно найти огромное количество плагинов, подключаемых модулей и библиотек, с помощью которых можно решить многие задачи, однако и их подключение требует знания основных принципов веб-дизайна.

Разработанный web-страница(сайт) представлен на флешке с пояснительной запиской.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* Оформление изображений на CSS3 [Электронный ресурс]. — Режим доступа: <https://habr.com/ru/articles/129313>. — Дата доступа: 12.04.2023.
* Автоматическое переключение слайдов JS [Электронный ресурс]. — Режим доступа: [https://ru.stackoverflow.com/questions/1248994/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5-%D0%BF%D0%B5%D1%80%D0%B5%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B5 HYPERLINK "https://ru.stackoverflow.com/questions/1248994/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5-%D0%BF%D0%B5%D1%80%D0%B5%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D1%81%D0%BB%D0%B0%D0%B9%D0%B4%D0%BE%D0%B2-js"-%D1%81%D0%BB%D0%B0%D0%B9%D0%B4%D0%BE%D0%B2-js](https://ru.stackoverflow.com/questions/1248994/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5-%D0%BF%D0%B5%D1%80%D0%B5%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D1%81%D0%BB%D0%B0%D0%B9%D0%B4%D0%BE%D0%B2-js) — Дата доступа: 12.04.2023.
* Планирование: setTimeout и setInterval [Электронный ресурс]. — Режим доступа: <https://learn.javascript.ru/settimeout-setinterval#setinterval> — Дата доступа: 12.04.2023.
* Введение в браузерные события [Электронный ресурс]. — Режим доступа: <https://learn.javascript.ru/introduction-browser-events> — Дата доступа: 12.04.2023.
* Всплытие и погружение [Электронный ресурс]. — Режим доступа: <https://learn.javascript.ru/bubbling-and-capturing> — Дата доступа: 12.04.2023.
* Автопрефиксер CSS онлайн [Электронный ресурс]. — Режим доступа: <http://autoprefixer.github.io/ru/> — Дата доступа: 12.04.2023.
* Web-сайт [Электронный ресурс]. — Режим доступа: <https://dic.academic.ru/dic.nsf/ruwiki/23679>. — Дата доступа: 12.04.2023.

ПРИЛОЖЕНИЕ 1

**Листинг кода**

// функция анимации картинок

// включает анимацию или выключает

function animation(event) {

// узнаём, на каком конкретно элементе произошло событие

//самый глубокий элемент, который вызывает событие, называется целевым элементом, и он доступен через event.target

const this\_button = event.target;

const number\_gallery = this\_button.dataset.button;

// получаем все картинки из текущей галереи

const images = document.querySelectorAll(

`img[data-container="${number\_gallery}"]`

);

// перебираем картинки и смотрим есть ли на них сейчас анимация

// если нет анимации, то к картинкам добавляется класс с анимацией

// если есть анимация, то класс удаляется

// класс добавляется или удаляется только к тем картинкам, кнопка какой галереи была задействована

for (let image of images) {

// проверяем есть ли анимация

// если есть

if (image.dataset.animate == "true") {

// выставляем дата атрибут анимации в false

image.dataset.animate = "false";

// в зависимости от номера галереи удаляем соответствующий класс с анимацией

switch (number\_gallery) {

case "1":

image.classList.remove("puls");

break;

case "2":

image.classList.remove("rotation");

break;

case "3":

image.classList.remove("shaking");

break;

case "4":

image.classList.remove("jumping");

break;

}

} else {

// если анимации не было

// выставляем дата атрибут показывающий есть ли анимация в true

image.dataset.animate = "true";

// добавляем класс анимации к соответствующим картинкам

switch (number\_gallery) {

case "1":

image.classList.add("puls");

break;

case "2":

image.classList.add("rotation");

break;

case "3":

image.classList.add("shaking");

break;

case "4":

image.classList.add("jumping");

break;

}

}

}

}

// таймеры для автоматического слайд шоу

let timer\_1 = 0;

let timer\_2 = 0;

let timer\_3 = 0;

let timer\_4 = 0;

// устанавливаем время задержки в милисекундах

const set\_timer = 3000;

// функция для запуска и остановки слайд шоу

function make\_timer(event) {

// узнаём, на каком конкретно элементе произошло событие

//самый глубокий элемент, который вызывает событие, называется целевым элементом, и он доступен через event.target

const this\_button = event.target;

const number\_gallery = this\_button.dataset.button;

// проверяем идёт ли сейчас слайд шоу соответсвующей галереи через атрибут slide кнопки

// автоматического проигрывания нет

if (this\_button.dataset.slide == "false") {

// устанавливаем атрибут slide кнопки в true потому что начинаем проигрывание

this\_button.dataset.slide = "true";

// выбираем нужную галерею, чтобы остальные не были задействованы

// setInterval позволяет вызывать функцию регулярно, повторяя вызов через определённый интервал времени

switch (number\_gallery) {

case "1":

timer\_1 = setInterval(function () {

// вызываем функцию которая пролистывает картинки

show\_slides(number\_gallery);

}, set\_timer);

break;

case "2":

timer\_2 = setInterval(function () {

// вызываем функцию которая пролистывает картинки

show\_slides(number\_gallery);

}, set\_timer);

break;

case "3":

timer\_3 = setInterval(function () {

// вызываем функцию которая пролистывает картинки

show\_slides(number\_gallery);

}, set\_timer);

break;

case "4":

timer\_4 = setInterval(function () {

// вызываем функцию которая пролистывает картинки

show\_slides(number\_gallery);

}, set\_timer);

break;

}

} else {

// если проигрывание сейчас происходит, то останавливаем его

// нужный атрибут у нужной кнопки ставим в false

this\_button.dataset.slide = "false";

// выбираем нужную галерею через номер чтобы остальные не были задействованы

// остановить дальнейшее выполнение функции setInterval, которая отвечает за автоматическое проигрывание,

// необходимо вызвать clearInterval(timerId)

switch (number\_gallery) {

case "1":

clearInterval(timer\_1);

break;

case "2":

clearInterval(timer\_2);

break;

case "3":

clearInterval(timer\_3);

break;

case "4":

clearInterval(timer\_4);

break;

}

}

}

// фунция, которая пролистывает картинки и которую вызывает фунция запуска и остановки слайд шоу make\_timer

// передаем в неё номер необходимой галереи

function show\_slides(number\_gallery) {

// находим картинки

const images = document.querySelectorAll(

`img[data-container="${number\_gallery}"]`

);

let current\_image = null;

// вычисляем картинку которая сейчас активна

for (let image of images) {

if (image.dataset.active == "true") {

current\_image = image.dataset.item;

}

}

// вычисляем id активной на данный момент картинки

// (id состоит из слова img, номера галереи и номера картинки)

const id\_hide\_image = "img\_" + String(number\_gallery) + String(current\_image);

// находим нужную куртинку

const hide\_image = document.getElementById(id\_hide\_image);

// прячем текущую картинку

hide\_image.style.display = "none";

// выставляем дата атрибут активности картинки active в false потому что сейчас она не активна

hide\_image.dataset.active = "false";

// увеличиваем счётчик активной картинки, чтобы показать следующую

current\_image++;

if (current\_image > images.length) {

// если счётчик больше чем элементов в массиве картинок

// то выставляем счётчик в 1 потому что номер картинок начинается с 1

current\_image = 1; // переменная i равна 0

}

// вычисляем id активной на данный момент картинки

// (id состоит из слова img, номера галереи и номера картинки)

const id\_show\_image = "img\_" + String(number\_gallery) + String(current\_image);

// находим нужную куртинку

const show\_image = document.getElementById(id\_show\_image);

// показываем нужную картинку выставляя display в block

show\_image.style.display = "block";

// выставляем дата атрибут активности в true

show\_image.dataset.active = "true";

}

// подключаем функции к кнопкам

// Метод addEventListener позволяет добавлять несколько обработчиков на одно событие одного элемента

// так же при использовании этого метода, можно удалять обработчик события используя removeEventListener

prev\_button\_1.addEventListener("click", move\_prev);

center\_button\_1.addEventListener("click", animation);

next\_button\_1.addEventListener("click", move\_next);

auto\_button\_1.addEventListener("click", make\_timer);

prev\_button\_2.addEventListener("click", move\_prev);

center\_button\_2.addEventListener("click", animation);

next\_button\_2.addEventListener("click", move\_next);

auto\_button\_2.addEventListener("click", make\_timer);

prev\_button\_3.addEventListener("click", move\_prev);

center\_button\_3.addEventListener("click", animation);

next\_button\_3.addEventListener("click", move\_next);

auto\_button\_3.addEventListener("click", make\_timer);

prev\_button\_4.addEventListener("click", move\_prev);

center\_button\_4.addEventListener("click", animation);

next\_button\_4.addEventListener("click", move\_next);

auto\_button\_4.addEventListener("click", make\_timer);