Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Санкт-Петербургский государственный университет промышленных технологий и дизайна»
ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАТЕХНОЛОГИЙ

Институт: Полиграфических технологий и оборудования
Кафедра: Информационных и управляющих систем
Направление подготовки: 09.03.02 Информационные системы и технологии
Профиль подготовки: Информационные технологии в медиаиндустрии

КУРСОВАЯ РАБОТА

Дисциплина: Web-программирование

Тема: Разработка адаптивного сайта художественной галереи

Выполнил:
студент группы 2-ТИД-3 Домнина К. Д.
(подпись)
Руководитель:
ст. преп. Пасечник П. А.
(уч. степень, звание) *(подпись)*

Дата защиты работы _____

Оценка _____

Санкт-Петербург
2023

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ И ДИЗАЙНА
ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАТЕХНОЛОГИЙ
Институт полиграфических технологий и оборудования

Кафедра ИиУС
УТВЕРЖДАЮ: _____
Зав. кафедрой Е. В. Горина

ЗАДАНИЕ
на курсовую работу

Дисциплина Web-программирование

Студент Домнина Ксения Денисовна Группа 2-ТИД-3

1. Тема проекта Разработка адаптивного сайта художественной галереи

2. Срок сдачи студентом законченного проекта 22.05.2023

3. Исходные данные к проекту список актуальных технологий в сфере web-программирования; примеры исходного кода

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов) : описание структуры разрабатываемого сайта; теория веб-разработки; общая структура и содержание файлов верстки; разработка основного содержания сайта; разработка адаптивности сайта.

5. Перечень графического материала захват экрана; фрагменты исходного кода проекта

6. Литература и прочие материалы, рекомендуемые для изучения: Беликова, С. А. Основы HTML и CSS: проектирование и дизайн веб-сайтов : учебное пособие по курсу «Web-разработка»; Богун, В. В. Web-программирование. Интерактивность статических Интернет-сайтов с применением форм : учебное пособие для СПО.

Дата выдачи задания «13» марта 2023 г.

Студент _____ (Домнина К.Д.)
Подпись Ф.И.О.
Руководитель _____ (Пасечник П.А.)
Подпись Ф.И.О.

Заключение руководителя о качестве курсовой работы

Оценка _____ Руководитель _____ (Пасечник П.А.)
подпись Ф.И.О.

РЕФЕРАТ

Отчет 44 с., 44 рис., 7 табл., 7 источн.

ВЕРСТКА, HTML, CSS, JAVASCRIPT, GITHUB

Объектом исследования является создание сайта.

Цель курсовой работы: разработка адаптивного сайта художественной галереи.

Исходные данные: список актуальных технологий в сфере web-программирования, примеры исходного кода.

Область применения:

а) Веб-сайт может использоваться с целью получения справочной информации с целью ознакомления с коллекциями произведений искусства;

б) Веб-сайт может использоваться в качестве учебного пособия при изучении произведений искусства.

Задачи:

а) Изучение предметной области;

б) Создание структуры разрабатываемого сайта;

в) Рассмотрение веб-технологий языков HTML и JavaScript, а также таблиц каскадных стилей CSS;

г) Разработка сайта.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время ни одна компания не может полноценно работать без собственного веб-сайта. Сайты позволяют облегчить процесс общения с клиентом путем размещения всей необходимой информации. С развитием веб-технологий появляются новые средства, позволяющие разработчикам упростить и улучшить процесс создания веб-страниц, а также добавить различные интерактивные элементы.

Художественные галереи являются важной частью жизни и культуры как для отдельного человека, так и для каждой страны в целом. Искусство объединяет различные эпохи, отражает исторические события и переживания людей.

Актуальность выбранной темы обусловлена важностью поддерживания и развития культурного образования среди населения посредством создания сайта художественной галереи для увеличения количества посетителей.

Целью данной курсовой работы является разработка адаптивного сайта художественной галереи.

Задачи, которые необходимо выполнить для достижения цели:

а) изучение предметной области;

б) создание структуры разрабатываемого сайта;

в) рассмотрение веб-технологий языков HTML и JavaScript, а также таблиц каскадных стилей CSS;

г) разработка сайта.

Объектом исследования является создание сайта.

Предметом исследования является разработка адаптивного сайта художественной галереи.

Теоретическая значимость состоит в возможности использования данной работы в качестве пособия для самостоятельного обучения.

Практическая значимость заключается в возможности использования материалов данной работы для верстки адаптивного сайта художественной галереи.

1 Теоретическая часть

1.1 Описание сферы деятельности

Художественная галерея – место, где происходит демонстрация изобразительного искусства, а также выставление его на продажу. Направления художественной деятельности галереи определяются по видам экспонатов или течениям в живописи.

Художественные галереи предоставляют образовательную сферу деятельности, а также являются коммерческой организацией, занимающейся продажей и продвижением сферы искусства.

1.2 Описание структуры разрабатываемого сайта

Необходимо создать многостраничный адаптивный сайт, содержащий основную информацию по картинам, художникам и товарам, а также содержать главную и информационную страницы, дающие представление о деятельности компании.

На каждой странице должны быть реализованы следующие блоки:

а) Шапка, зафиксированная сверху, содержащая логотип, информацию о странице, а также навигацию;

б) Подвал сайта с контактной информацией и логотипом;

в) Основные блоки, содержащие информацию, связанную с разделом сайта.

Также должны присутствовать следующие элементы и интерактивность:

а) Для страниц о картинах, художниках и товарах общая информация должна быть представлена в виде таблиц;

б) Информация о художниках должна быть реализована с помощью раскрывающегося текста по нажатию контекстной кнопки;

в) На главной странице для всех разделов, кроме информационной, должны быть реализованы слайдеры;

г) Единичные элементы такие как изображения и кнопки при наведении должны масштабироваться;

д) Логотип в шапке должен быть ссылкой на главную страницу сайта.

Общие требования к дизайну:

а) Логотип должен соответствовать деятельности компании;

б) Цветовая гамма разрабатываемого сайта – белый и RGB(168, 50, 82);

в) Используемый шрифт – «Old Standart TT» в начертании «serif»;

г) Текст должен быть удобочитаемым для размеров экранов до 350 пикселей.

Верстка должна быть реализована с помощью HTML, CSS, функционал может быть описан через JavaScript.

1.3 Теория веб-разработки

1.3.1 Язык гипертекстовой разметки HTML

Язык HTML (HyperText Markup Language) – язык гипертекстовой разметки для просмотра веб-страниц в браузере.

Основной структурой являются теги, обозначающие начало и окончание элемента на странице. Они имеют разделение на парные и одиночные по способу написания. Одиночные указываются в виде открывающей части, а информация содержится внутри атрибутов. У парных помимо открывающей части есть обязательная закрывающая, содержимое указывается между ними. Присутствует возможность вложения элементов друг в друга.

Каждый тег может иметь глобальные атрибуты. Основные из них – class и id. При этом атрибут id является уникальным идентификатором для всего документа. Помимо глобальных, у тегов могут быть собственные атрибуты.

По особенностям отображения теги разделяются на три вида: строчные, блочные и служебные.

Служебными тегами являются необходимыми для структурирования HTML-документа, а также для управления параметрами браузера.

Элемент <!DOCTYPE> предназначен для указания типа текущего документа, располагается в начале. Таким образом браузер всегда будет корректно интерпретировать веб-страницу. Для версии языка HTML5, указывается <!DOCTYPE html>.

В таблице 1 указаны служебные теги, а также их описание.

Таблица 1 — Служебные теги

Название тега	Описание
<html>…</html>	Является контейнером, содержащим в себе все содержимое страницы.
<head>…</head>	Содержит элементы для работы с данными, которые не отображаются на странице. Находится внутри тега <html>.
<body>…</body>	Внутри данного тега указывается содержимое веб-страницы, которое отображается в окне браузера.

Атрибут «lang» тега html устанавливает язык контента веб-страницы. Для русского языка используется значение ru.

Служебные теги, находящиеся внутри head перечислены в таблице 2.

Таблица 2 — Служебные теги внутри head

Название тега	Описание
<title>…</title>	Определяет название веб-страницы, указанное внутри тега.
<meta>	Мета-тег, содержащий дополнительные параметры для хранения информации, предназначенной для браузеров и поисковых систем.
<link>	Содержит указание на файлы, связанные со страницей, кроме кода JavaScript. Может задавать подключение CSS-кода и иконки веб-страницы.

Для тега meta существует большое количество различных атрибутов, позволяющих производить гибкую настройку параметров.

Например, для определения кодировки используется атрибут «charset», значение UTF-8 определяет Unicode.

С помощью атрибута «name» определяется предназначение тега. Таким образом значение viewport заключается в обрабатывании размеров страницы и ее масштаба. Атрибут «name» определяется в атрибуте «content». Так, значение width задает ширину области просмотра, а значение initial-scale

задает масштабирование. Для адаптивных сайтов параметр width имеет значение device-width, что означает ширину устройства или окна браузера.

Тег link имеет основные атрибуты «href» и «rel». В атрибут «href» передается абсолютный или относительный путь URL. Таким образом можно подключить CSS-код или шрифты. Атрибут «rel» устанавливает тип ссылки. Так, значение stylesheet определяет ресурс в качестве таблицы стилей.

Внутри тега body содержатся блочные и строчные теги.

Блочными тегами являются элементы, размер которых определяется содержимым, занимают всю доступную ширину. В таблице 3 перечислены блочные теги. В свою очередь они разделяются на обычные, семантические теги и теги составных конструкций.

Таблица 3 — Блочные теги

Название тега	Описание
…	Обычный тег. Предназначен для создания абзацев текста. По умолчанию отображает текст с вертикальными отступами на новой строке.
…	Обычный тег. Служит контейнером для других тегов. Позволяет изменять содержимое выделенного фрагмента документа.
<header>…</header>	Семантический тег. Является «шапкой» страницы, содержащей навигацию по сайту, логотип. Также применяется как вводная часть любого раздела.
<footer>…</footer>	Семантический тег. Задает «подвал» сайта, где обычно размещается дополнительная, контактная информация, а также для заключительной части раздела.
<main>…</main>	Семантический тег. Определяет основное содержимое страницы, не повторяющееся на других. В отличие от других семантических тегов может быть объявлен только один раз.

Продолжение таблицы 3

Название тега	Описание
<article>…</article>	Семантический тег. Является автономной частью страницы, например, статьей.
…	Тег составной конструкции. Задает маркированный список. Внутри обязательно содержатся теги .
…	Тег составной конструкции. Определяет пункт списка.
<table>…</table>	Тег составной конструкции. Является корневым тегом для создания таблиц. Внутри содержит теги <tr>.
<tr>…</tr>	Тег составной конструкции. Задает строку таблицы.
<th>…</th>	Тег составной конструкции. Определяет ячейку-заголовок. По умолчанию текст внутри имеет жирное начертание и расположен по центру.
<td>…</td>	Тег составной конструкции. Определяет ячейку таблицы.

Перечисленные выше семантические теги определяют структурную разметку HTML5, при которой упрощается процесс создания структуры страниц путем уменьшения использования тегов div с аналогичным функционалом для размещения содержимого. Из перечисленных только тег main используется на странице один раз, остальные теги могут являться структурной частью отдельных элементов. Теги составных конструкций применяются вместе с их внутренними тегами. Так, маркированные списки задаются с помощью ul и li, а таблицы с указанием table внутри содержат элементы tr, th, td.

Строчные теги характеризуются тем, что они являются частью текста, при этом чаще всего используются для изменения внешнего вида текста или его логического выделения. Например, в таблице 4 перечислены некоторые строчные теги.

Таблица 4 — Строчные теги

Название тега	Описание
<a>…	Является ссылкой, позволяющая связать текст или изображение с гипертекстовыми документами.
<img>	Предназначен для отображения изображений.
<label>…</label>	Устанавливает связь между определенной меткой и элементом формы, например, <input>. При изменении состояния позволяет изменять значения элементов.
<input>	Задает различные поля для управления.

Для задания тегу «a» гиперссылки используется атрибут «href», в который передается имя документа или адрес веб-страницу. Атрибут «target» указывает способ просмотра, по умолчанию имеющий значение _self, который задает текущее окно браузера.

Атрибутами тега img являются «src», «alt». Атрибут «src» определяет путь или ссылку на изображение, а в атрибут «alt» передается подстановочный текст, если браузер не сможет его загрузить картинку. Оба параметра являются обязательными.

Тег label имеет атрибут «for», в который передается id связанного элемента.

Главным атрибутом тега input является «type», определяющий поведение элемента. По умолчанию используется тип text. Тип checkbox преобразует элемент в переключатель, что дает пользователю возможность совершить выбор отдельных значений из предложенных вариантов.

С помощью тега <script>…</script> в документ можно добавить код JavaScript. Данный тег является служебным, но его можно использовать в конце тега body, чтобы браузер сначала интерпретировал все содержимое страницы, после чего корректно выполнял код для необходимых элементов.

1.3.2 Таблицы каскадных стилей CSS

Язык CSS (Cascading Style Sheets) – технология описания внешнего вида документа, написанного с использованием языка разметки.

Стиль определяется набором свойств, являющихся частью CSS-правила. Каждое свойство определяет изменяемую характеристику, параметр или параметры которой указываются через двоеточие.

Для обращения к элементам веб-страницы используются селекторы, часть CSS-правила. Селекторы тега используются для обращения ко всем элементам какого-либо тега с указанием его обозначения. Селекторы класса выбирают элементы по значению их атрибута class через точку. Селекторы по идентификатору обращаются к элементам по значению атрибута id через знак решетки. Универсальным селектором для всех элементов страницы является знак ‘*’.

Применение CSS к документам HTML основано на принципах наследования и каскадирования.

Принцип наследования: свойства элементов-предков практически всегда наследуются элементами-потомками.

Принцип каскадирования вводит правила приоритета: низким приоритетом обладает стиль браузера, затем – пользовательский стиль настройки браузера, высоким приоритетом обладает стиль, заданный автором страницы.

Исходя из вышеуказанных принципов выделяются сложные селекторы, состоящие из нескольких простых и комбинатора.

В таблице 5 представлены некоторые виды комбинаторов и их описание.

Таблица 5 — Виды комбинаторов

Комбинатор	Описание
Запятая	Используется при применении одного стиля для нескольких селекторов, называется группировкой.
Пробел	Выбирает селектор, имеющий определенную вложенность.
Тильда	Определяет второй селектор в том случае, если ему предшествует первый. При этом оба должны иметь общего элемента-предка.

Псевдоклассы отвечают за изменение внешнего вида в результате определенных действий пользователем или за положение элемента в теле документа и задаются через двоеточие:

- «:hover» – состояние элемента при наведении на него мышью,
- «:visited» – состояние посещенной гиперссылки,
- «:first-child» – первый дочерний элемент другого элемента,
- «:checked» – положение значения переключателей и флажков,
- «:not()» – отрицательный псевдокласс, определяет элементы, которые не соответствуют передаваемому селектору.

Псевдоэлементы применяют стиль элемента в зависимости от контекста. Например, «:after» отвечает за объявление стиля для данных, находящихся после указанного элемента.

Каждый элемент, определенный в HTML-документе, имеет позиционирование, задающееся свойством position. В таблице 6 приведены его значения и их описание.

Таблица 6 — Значения свойства position

Значение	Описание
static	Статическое позиционирование, значение по умолчанию. Положение элемента определяет порядк HTML-документа.
relative	Относительное позиционирование. Оно позволяет изменять расположение элемента относительно его исходной позиции.
absolute	Абсолютное позиционирование. Элементу задается конкретное положение на странице. При этом остальные элементы располагаются без учета данного элемента.
fixed	Фиксированное позиционирование. При прокрутке страницы элемент всегда находится в заданном месте на экране.

Смещение элемента может задаваться с помощью свойств left, right, top, bottom, определяющие расстояние от краев родительского элемента.

Совмещение значений relative родительского элемента и absolute у дочернего позволяет располагать последний относительно границ родительского.

Если при позиционировании происходит наложение элементов, используется свойство z-index, определяющее положение по оси z. Значением является целое число, обозначающее уровень нахождения элемента.

Для позиционирования элементов также используется технология Flexbox, которая позволяет создать гибкие макеты. Элементы могут выстраиваться в любом направлении и занимать любое пространство.

Свойство display определяет тип отображения блока. При значении flex элементы отображаются как контейнер Flex, располагаясь в направлении слева направо на одной линии. Для разрешения переноса используется свойство flex-wrap, имеющее значения:

- wrap – выстраивание элементов в несколько строк,
- nowrap – выстраивание в одну линию.

Свойство justify-content выравнивает Flex-элементы контейнера, когда они не используют все пространство на горизонтальной оси, и принимает значения:

- flex-start – элементы располагаются в начале контейнера,
- flex-end – в конце контейнера,
- center – в центре контейнера,
- space-between – в промежутке между линиями,
- space-around – с пробелами до, между и после линий.

В CSS все элементы HTML-документа представляются в виде блоков, разделенных на отдельные области: margin, border, padding и содержимое.

- margin задает поле, отступ снаружи от границы элемента,
- padding задает отступ внутрь от границы,
- border является границей блока.

Свойство box-sizing определяет алгоритм расчета ширины и высоты элемента. Значение border-box в высоту и ширину включает значения полей и границ, но не отступов.

Свойства margin и padding задают значения для всех направлений через пробел. При указании одной величины отступы будут заданы одинаково во всех направлениях, при двух – попарно сверху-снизу и слева-справа, при трех – сверху, снизу и слева-справа, при четырех – по очереди сверху, справа, снизу, слева. Добавляя «-left», «-right», «-top», «-bottom» к необходимому свойству, аналогично можно задавать отступы по направлениям.

Свойство border является совмещением таких стилей как border-width, border-style, border-color, отвечающие за ширину, тип и цвет границы элемента. Ширина может задаваться в пикселях или процентах, цвет задается с помощью ключевых слов. Тип границы имеет несколько значений. Например, значение solid определяет сплошную линию.

Содержимое всегда имеет размер, который может задаваться напрямую или рассчитываться автоматически браузером. Свойства width и height позволяют задать высоту и ширину элементу в пикселях или в процентах. Максимальные ширину и высоту определяют свойства max-width и max-height.

Внутреннее содержимое подстраивается под заданные размеры, для указания способа поведения элементов, не вмещающиеся в параметры, используется свойство overflow. Например, при значении hidden видна лишь часть, которая помещается в указанные ширину и высоту, а остальное скрывается.

В таблице 7 приведены свойства и их описание для оформления текста.

Таблица 7 — Свойства для оформления текста

Свойство	Описание
text-align	Определяет выравнивание текста в зависимости от значения: left – по левому краю, right – по правому краю, center – по центру.
font-family	Указывает тип шрифта, а также его начертание.

Продолжение таблицы 7

font-size	Задает размер текста, который можно указать в пунктах, пикселях или в процентах.
text-decoration	Описывает такие эффекты как подчеркивание, перечеркивание, линию сверху, а также отмене любых эффектов.
text-transform	Управляет регистром текста.
color	Указывает цвет текста. Цвет можно выбрать из предопределенных по ключевому слову, в формате HEX-кода, RGB, HSL.

Также в CSS присутствуют свойства, позволяющие задавать внешнее оформление фона элементов.

Свойство box-shadow добавляет тень к элементу. Задается с указанием сдвига по оси x и y, радиуса размытия и растяжения, а также цвета. Фон элемента задает свойство background-color. Принимает значения аналогично свойству color. Свойство background универсальное, объединяющее в себе одновременно все характеристики фона. Например, градиент считается фоновым изображением, поэтому он задается через свойство background как linear_gradient(to top/bottom/left/right, *начальный цвет*, *конечный цвет*). Первый параметр задает направление градиента.

Для таблиц и списков используются специальные свойства. Таким образом за изменение маркеров списков используется свойство list-style-type. Например, для маркированного списка ul можно убрать маркеры, используя свойство none.

Границы таблицы задаются с помощью свойства border-collapse. По умолчанию принимает значение collapse, при котором границы соседних ячеек сжимаются так, что остается видна одна полоса границы без промежутков.

Ширина таблицы определяется с помощью алгоритмов расчета, выбор которых указывается в свойстве table-layout. Значение fixed определяет

фиксированный алгоритм вычислений, при котором ширина зависит от свойства width столбцов и ячеек. Сначала просматривается ширина столбцов с конкретно заданными значениями, потом – ширина ячейки первой строки. Итоговой шириной таблицы становится сумма ширины столбцов или значение width элемента table, при этом выбирается максимальная величина.

При определении стиля с помощью динамических псевдоклассов можно задать анимацию для элемента. Для плавных переходов используется transition, объединяющее в себе несколько свойств: transition-property – указывает, к каким стилям применяется свойство, transition-duration – время анимации, указанное в секундах, transition-timing-function – определяет ускорение свойства во время перехода, а также transition-delay – задержка перед началом анимации в секундах.

Свойство transform позволяет масштабировать, вращать, сдвигать, наклонять элемент, за что отвечают функции трансформации scale, rotate, translate, skew соответственно. Возможно задание изменения трансформации по конкретной оси «X» или «Y».

Правила «@» позволяют настраивать параметры файла стилей. Таким образом, @media позволяет задавать CSS-правила для различных устройств в пределах одной таблицы стилей. При передаче значения @media(max-width:*величина*) можно задать отображение для ширины экранов, не больше указанной, что позволяет сделать сайт адаптивным.

1.3.3 Язык программирования JavaScript

JavaScript является основным в web-разработке, так как все современные браузеры включают в себя его интерпретатор.

Переменные объявляются с помощью ключевых слов var, let или const. Ключевое слово let является современным способом объявления переменных.

Язык JavaScript является динамическим, поэтому тип переменных определяется автоматически в момент выполнения программы. В связи с этим есть два метода сравнения переменных: с помощью «==» происходит

сравнение на равенство с учетом приведения типов, а с помощью «===» - сравнение на идентичность.

Для выполнения арифметических операций используются операторы «+», «-», «*», «/», «%», «**». Последние два являются нахождением модуля от числа и возведением в степень. При этом оператор сложения работает по-разному в зависимости от типов переменных. Если один из слагаемых является строковой переменной, то производится конкатенация строк.

С помощью ключевого слова if создается условный оператор: if(*условие*){*выполняемый код при верности условия*}.

Функции определяются с помощью ключевого слова «function» с указанием имени и «()», где в скобках указываются аргументы функции. Далее в «{}» помещается выполняемый код.

Для работы с отложенными вычислениями в языке JavaScript присутствует функция setInterval(), повторяющаяся с указанной периодичностью. Первым аргументом данной функции является либо созданная другая функция, либо код, который передается с помощью «function(){}». Вторым аргументом является временной интервал, который задается в миллисекундах.

JavaScript позволяет управлять сайтом. После прочтения HTML-документа браузером создается дерево представления, с помощью которого определяется способ получения доступа к каждому элементу на странице.

Одним из основных элементов в документе является «document», который обрабатывает все HTML документы. Таким образом метод querySelector() возвращает первый элемент с определенным селектором CSS. Если элемент с указанным селектором не найден, функция возвращает ноль.

Помимо методов есть свойства, которые позволяют выделять отдельную информацию. Так, свойство offsetWidth возвращает значение ширины элемента.

Ниже приведен пример использования метода querySelector() и свойства offsetWidth для получения ширины первого элемента с селектором «.header» с помещением его в переменную header:

let header = document.querySelector(‘.header’).offsetWidth

Для изменения стиля элемента используется свойство style с указанием названия CSS-свойства. Если название последнего состоит из нескольких слов, то оно преобразуется в camelCase.

Ниже приведен пример изменения свойства text-decoration элемента header в значение none:

header.style.textDecoration = ‘none’

1.4 Система контроля версий GitHub

Система контроля версий GitHub используется для эффективного управления разработкой и поддержкой кода, а также предоставляет разработчикам мощные инструменты для отслеживания изменений, совместной работы и управления версиями исходного кода. При создании проекта на GitHub инициализируется репозиторий для хранения всей истории изменений.

После завершения отдельных этапов разработки выполняются коммиты, в которых фиксируются изменения в коде, а также содержатся их описание. Это позволяет легко отследить, какие изменения были внесены и почему.

Для того чтобы обеспечить доступность проекта в Интернете, используется GitHub Pages. С помощью него можно разместить статический веб-сайт, основанный на ветке репозитория, в которой хранятся файлы исходного кода и ресурсы веб-сайта.

Таким образом использование системы контроля версий GitHub позволяет эффективно управлять разработкой проекта и поддерживать стабильность решения. Использование GitHub Pages дает возможность публиковать проект в Интернете и делиться им с другими пользователями.

2 Практическая часть

2.1 Планирование разработки

Перед началом разработки веб-сайта для структурированного и последовательного выполнения этапов формируется план действий:

а) составление общей структуры всех страниц сайта, состоящей из «шапки», основного содержания и «подвала»;

б) насыщение основного содержания основными интерактивными элементами и элементами, отвечающими за визуальное отображение;

в) создание стилей используемых элементов, определяющих их визуальную составляющую и распределение на экране;

г) добавление функционала интерактивным элементам;

д) адаптация сайта под различные размеры окна отображения.

2.2 Общая структура и содержание файлов верстки

Для сохранения единообразия внешнего вида создаваемого сайта необходимо создать элементы, которые будут повторяться на каждой странице сайта.

В служебном теге head с помощью тегов meta устанавливается кодировка страниц Unicode, а также определяется область просмотра для разных размеров экранов. В теге title указывается наименование текущей страницы, через тег link подключается шрифт и CSS-код.

В зависимости от раздела сайта меняется только содержимое тега title. Код, реализующий вышеописанное, представлен на рисунке 1.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Картины</title>
    <link href="https://fonts.googleapis.com/css2?family=Old+Standard+TT&family=Poiret+One&display=swap"
        rel="stylesheet">
    <link rel="stylesheet" href="style.css">
</head>
```

Рисунок 1 — HTML-код тега head для раздела «Картины»

Для применения выбранного шрифта в CSS-коде перечисляются все селекторы тегов, содержащие текст, через комбинатор запятую, которым применяется свойство font-family, что представлено на рисунке 2.

```
p,
a,
li,
th,
td,
label {
    font-family: 'Old Standard TT', serif;
}
```

Рисунок 2 — CSS-правило для применения шрифта

Также каждая страница имеет разделение по семантическим тегам <header class="header">, <footer class="footer">, <main class="main">, обозначающие «шапку», «подвал» и основной блок соответственно.

Элемент <header class="header"> содержит элементы навигации по сайту, логотип компании, а также поясняющий текст содержания страницы.

Для создания навигации используется маркированный список <ul class="navigation">. Его элементами <li class="link_decor navigation_item"> являются ссылки на другие разделы сайта, поэтому внутри данного тега расположен тег <a class="link">. Навигация одинакова для всех страниц.

Позиционирование элементов списка осуществляется с помощью технологии Flexbox. Содержимое располагается с промежутками по ширине без возможности переноса элементов на следующие строки. Навигация зафиксирована наверху экрана, что достигается указанием свойству position значения fixed, а чтобы меню всегда находилось выше других элементов, устанавливается свойство z-index с большим значением. На рисунке 3 представлен код, описывающий стиль «navigation».

```
.navigation {
    position: fixed;
    width: 100%;
    display: flex;
    justify-content: space-around;
    flex-wrap: nowrap;
    z-index: 99;
    padding: 10px 0;
    background-color: rgb(168, 50, 82);
}
```

Рисунок 3 — CSS-правило для класса navigation

Элементы списка объединяют в себе несколько стилей. Тегу li установлены классы «link_decor» и «navigation_item».

Первый служит для задания рамки, внутренних отступов и размера текста для всех ссылок. Тег a имеет класс «link», для которого внешнее оформление текста ссылок убирается с помощью свойства text-decoration, а также устанавливается белый цвет.

У класса «navigation_item» убирается маркировка. С использованием псевдокласса «:hover» изменяется цвет и масштаб. Плавность анимации задается с помощью свойства transition. При наведении мышью на какой-либо элемент происходит инверсия цветов, а также создается эффект свечения с помощью box-shadow. Ниже на рисунке 4 представлены стили, описывающие изменение элементов.

```css
.link {
    text-decoration: none;
    color: white;
}
.navigation {
    position: fixed;
    width: 100%;
    display: flex;
    justify-content: space-around;
    flex-wrap: nowrap;
    z-index: 99;
    padding: 10px 0;
    background-color: rgb(168, 50, 82);
}

.navigation_item {
    list-style-type: none;
    transition: transform 0.5s ease 0s;
}

.navigation_item:hover {
    transform: scale(1.1);
    padding: 5px 5px;
    box-shadow: 0 0 15px white;
    background-color: white;
}
```

Рисунок 4 — CSS-правила для изменения элементов навигации

На рисунке 5 представлено изменение внешнего вида при наведении мышью на пункт «Художники».

Рисунок 5 — Навигация сайта с выделением элемента «Художники»

Логотип и заголовок создаются с помощью тегов <img class=“header_logo”> и <p class=“header_text”> соответственно. Логотип является ссылкой на главную страницу сайта, поэтому тег img обернут в тег a. Все перечисленные элементы объединены в блок , для позиционирования содержимого используется технология Flexbox с указанием положения по центру.

На рисунке 6 представлен фрагмент кода header страницы «Главная».

```
<header class="header">
    <ul class="navigation">
        <li class="link_decor navigation_item"><a class="link" href="index.html">Главная</a></li>
        <li class="link_decor navigation_item"><a class="link" href="art.html">Картины</a></li>
        <li class="link_decor navigation_item"><a class="link" href="artist.html">Художники</a></li>
        <li class="link_decor navigation_item"><a class="link" href="shop.html">Товары</a></li>
        <li class="link_decor navigation_item"><a class="link" href="about.html">О нас</a></li>
    </ul>
    <div class="header_content">
        <a href="index.html"><img class="header_logo" src="common/Logo.png" alt=""></a>
        <p class="header_text">Добро пожаловать на сайт художественной галереи!</p>
    </div>
</header>
```

Рисунок 6 — HTML-код шапки для страницы «Главная»

Результат шапки для данной страницы представлен на рисунке 7.

Рисунок 7 — Внешний вид шапки для страницы «Главная»

Тег <footer class=“footer”> содержит внутри себя логотип и контактные данные в тегах <img class=“footer_logo”> и <p class=“footer_text_info”> соответственно. Текст объединен в . «footer_logo» и «footer_text» расположены с указанием display в позиции flex в классе «footer». Подвал сайта идентичен для каждого раздела. На рисунке 8 представлен HTML-код для задания «подвала» страниц сайта.

```
<footer class="footer">
    <img class="footer_logo" src="common/Logo.png">
    <div class="footer_text">
        <p class="footer_text_info">Телефон: +7(000)000-00-00</p>
        <p class="footer_text_info">Почта: art_gallery@mail.ru</p>
        <p class="footer_text_info">Адрес: ул. Атласова, д. 123</p>
    </div>
</footer>
```

Рисунок 8 — HTML-код шапки для страницы «Главная»

Внешний вид «подвала» представлен на рисунке 9.

Рисунок 9 — Внешний вид шапки для страницы «Главная»

Основная информация содержится в теге <main class=“main”>, который в свою очередь содержит теги <article class=“article”> для размещения всех необходимых элементов. Внутри тега <article class=“article”> расположены такие структурные составляющие как <p class=“article_header”> и . В разделе «О нас» есть только последний элемент. «article_header» – название статьи, в «article_content» помещается все содержимое, расположенное с помощью технологии Flexbox с указанием невозможности переноса элементов. Между статьями вставлена полоса-разделитель .

2.3 Разработка основного содержания сайта

2.3.1 Изображения и текст

В блоке располагаются элементы для реализации статьи. Общими в данном блоке являются теги <img class=“article_image”> и <p class=“article_text”> для изображения и текста.

Ширину картинок определяют стили «image_main», «image_art», «image_artist», «image_shop», «image_about» для каждого раздела. Стиль класса «article_image» указывает внешние отступы и высоту изображения 100%, чтобы избежать искажения. Задается масштабирование с помощью псевдокласса «:hover» и свойств transform и transition. Для каждой страницы в теге img указывается класс «article_image» и соответствующий разделу сайта класс изображения. На рисунке 10 представлен код стилей, описанных выше.

```css
.article_image {
    padding: 0 0 20px 25px;
    height: 100%;
}

.article_image:hover {
    transform: scale(1.1);
}

.image_main,
.image_art,
.image_artist,
.image_shop,
.image_about {
    transition: all 0.5s ease 0s;
}

.image_main,
.image_art,
.image_shop {
    width: 400px;
}

.image_artist {
    width: 250px;
}

.image_about {
    width: 300px;
}
```

Рисунок 10 — CSS-правила для оформления изображений

Для текста в классе «article_text» указываются внешние и внутренние отступы, а также внешние параметры для шрифта.

На рисунке 11 представлен пример масштабирования изображения.

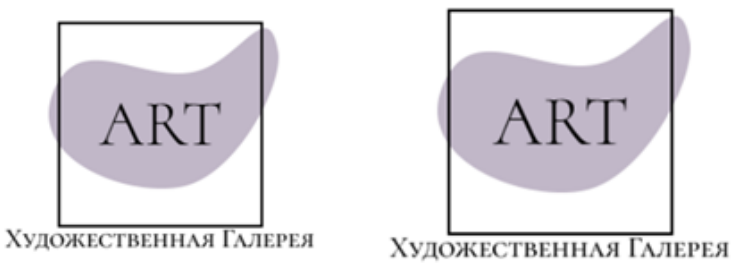

Рисунок 11 — Состояние изображения до и после наведения мышью

Для страницы «О нас» теги <p class=“article_text”> объединены в блок , где с помощью технологии Flexbox элементам задается перенос по строкам и расположение по центру (рисунок 12).

```css
.article_text_about {
    width: 60%;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
}
```

Рисунок 12 — CSS-правило для расположения текста страницы «О нас»

На данном этапе создается страница «О нас», результат представлен на рисунке 13.

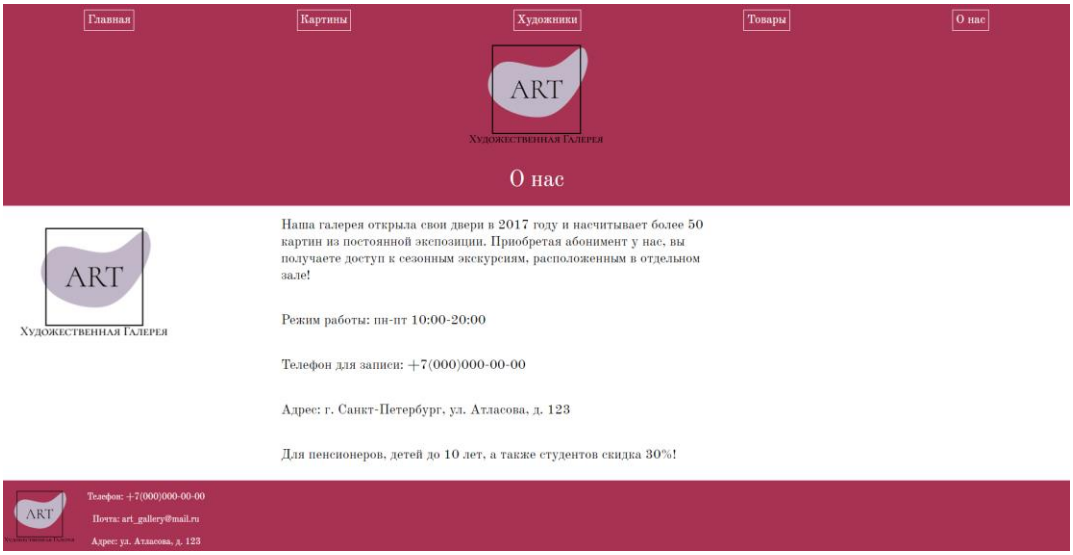

Рисунок 13 — Внешний вид страницы «О нас»

2.3.2 Таблицы

Для размещения обобщающей информации статьи создаются таблицы для страниц «Картины», «Художники», «Товары». В объединены таблица и <p class=“article_text”>, где с помощью технологии Flexbox указывается положение элементов в столбец. Стиль класса «article_table» задает ширину всего блока, при этом для страницы «Товары» создан свой стиль класса «table_shop» для корректного отображения.

Стили таблицы создаются с помощью селекторов тегов table, th, td для сохранения оформления при дальнейшем добавлении новых таблиц. Для тега table ширина таблицы и столбца определяется с помощью table-layout в позиции fixed. Граница элементов устанавливается свойством border-collapse со значением collapse. Ширина таблицы имеет значение 50%. Для тегов td и th устанавливается оформление текста, а также граница border.

На рисунке 14 представлен код для стилей table, th, td.

```css
table {
    table-layout: fixed;
    width: 50%;
    border-collapse: collapse;
    border: 3px solid rgb(168, 50, 82);
}

th,
td {
    font-size: 15pt;
    border: 2pt solid rgb(168, 50, 82);
    text-align: center;
}
```

Рисунок 14 — CSS-правила для оформления элементов таблиц

На рисунке 15 представлен пример вида таблицы в статье на странице «Картины».

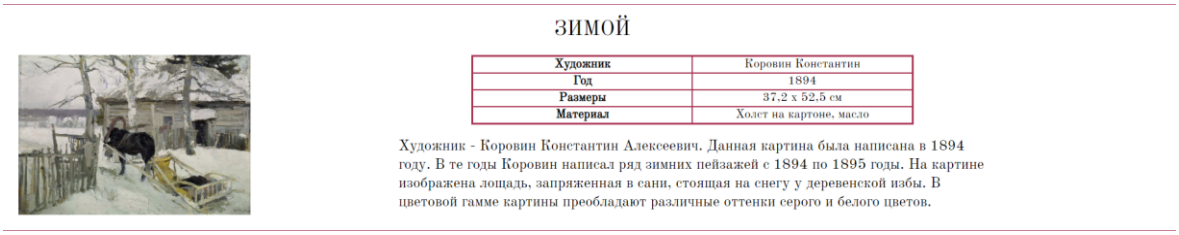

Рисунок 15 — Пример таблицы на странице «Картины»

На данном этапе создается страница «Картины», результат представлен на рисунке 16.

Рисунок 16 — Внешний вид страницы «Картины»

2.3.3 Слайдеры

На странице «Главная» расположены три слайдера, функционал которых описывается с помощью JavaScript.

Внутри находится <p class=“article_text”> и . Последний отвечает за отображение слайдера, который содержит четыре изображения <img class=“slider_image”>. На рисунке 17 представлен HTML-код для слайдера о картинах.

```
<div class="slider">
    <div class="slider_elements" id="sliderJs1">
        <img class="slider_image" src="main/Slider_Art_1.png" alt="">
        <img class="slider_image" src="main/Slider_Art_2.png" alt="">
        <img class="slider_image" src="main/Slider_Art_3.png" alt="">
        <img class="slider_image" src="main/Slider_Art_4.png" alt="">
    </div>
</div>
```

Рисунок 17 — HTML-код для слайдеров

Для класса «slider» задается необходимая ширина слайдера, все, что не вмещается, скрывается с помощью свойства overflow со значением hidden. В классе «slider_elements» с помощью технологии Flexbox изображения располагаются друг за другом по правому краю с указанием смещения left со значением 0. В классе «slider_image» указывается только ширина изображений.

Ниже представлен код для оформления слайдеров на рисунке 18.

```
.slider {
    overflow: hidden;
    width: 300px;
    margin-left: 10px;
    border: 1px solid rgba(168, 50, 82, 0);
}

.slider_elements {
    width: 100%;
    display: flex;
    position: relative;
    left: 0px;
    transition: left 1s ease 0s;
}

.slider_image {
    width: 300px;
}
```

Рисунок 18 — CSS-правила для слайдеров

Также для блока указывается id каждого слайдера, имеющий вид id=“sliderJs*порядковый номер слайдера*”.

Изображения сменяют друг друга изменением свойства left. Данное изменение реализуется с помощью JavaScript.

Определяются переменные slider1, slider2, slider3, инициализируемые методом document.querySelector() с аргументом в виде id каждого слайдера (рисунок 19).

```
let slider1 = document.querySelector('#sliderJs1')
let slider2 = document.querySelector('#sliderJs2')
let slider3 = document.querySelector('#sliderJs3')
```

Рисунок 19 — Переменные для получения элементов слайдеров

Таким образом появляется возможность изменять стили для конкретных слайдеров с помощью кода JavaScript.

Далее определяются три переменные: elemWidth1, elemWidth2 для вычисления ширины «slider_image» и переменная offset для задания смещения (рисунок 20).

```
let elemWidth1 = 0
let elemWidth2 = 0
let offset = 0
```

Рисунок 20 — Переменные для реализации смещения

Функция setInterval(function(){}, 2000) используется для указания интервала обновления элемента, где 2000 – количество миллисекунд перед каждым смещением. function() определяет выполняемый код на каждом интервале.

Переменная elemWidth1 принимает значение ширины изображения с помощью метода document.querySelector().offsetWidth. Происходит сравнение величин elemWidth2 и elemWidth1 через условный оператор if. Если значения не равны, и elemWidth2 имеет значение, не равное изначальному, то offset устанавливается в 0, и свойству left слайдеров присваивается данная величина. Таким образом смещение подстраивается под размеры изображений. Код, описывающий данное поведение, представлен на рисунке 21.

```
elemWidth1 = document.querySelector('.slider_image').offsetWidth
if ((elemWidth1 != elemWidth2) && (elemWidth2 != 0)) {
    offset = 0
    slider1.style.left = offset + 'px'
    slider2.style.left = offset + 'px'
    slider3.style.left = offset + 'px'
}
```

Рисунок 21 — Проверка на изменение ширины изображений

Далее значение смещения offset уменьшается на значение ширины изображения elemWidth1. Для смены последнего изображения на первое, происходит проверка на максимально возможное значение offset с помощью условного оператора if. Максимальный размер равен количеству элементов, умноженному на ширину изображений elemWidth1 и на минус один, потому что величина смещения имеет отрицательное значение. Вышеописанное представлено на рисунке 22.

```
offset -= elemWidth1
if (offset < (-1) * elemWidth1 * 3) {
    offset = 0
}
```

Рисунок 22 — Задание смещения и его проверка

При изменении размеров изображений в процессе выполнения интервала, в переменную elemWidth2 передается значение ширины, и происходит сравнение значений elemWidth1 и elemWidth2 аналогично первой проверке, только без условия нулевого значения elemWidth2. Данная проверка представлена на рисунке 23.

```
elemWidth2 = document.querySelector('.slider_image').offsetWidth
if (elemWidth1 != elemWidth2) {
    offset = 0
    slider1.style.left = offset + 'px'
    slider2.style.left = offset + 'px'
    slider3.style.left = offset + 'px'
}
```

Рисунок 23 — Проверка на изменение размеров изображения в ходе выполнения цикла

Если в процессе выполнения интервала размер изображений не был изменен, то полученная величина offset присваивается свойству left у каждого

слайдера. В конце выполнения опять считывается величина ширины изображений elemWidth2 для сравнения в начале интервала. Данная часть представлена на рисунке 24.

```
slider1.style.left = offset + 'px'
slider2.style.left = offset + 'px'
slider3.style.left = offset + 'px'
elemWidth2 = document.querySelector('.slider_image').offsetWidth
```

Рисунок 24 — Задание смещения параметру left и считывание ширины

На рисунке 25 представлен пример, демонстрирующий последовательную смену двух изображений.

Рисунок 25 — Пример работы слайдера на странице «Главная»

На данном этапе создается страница «Картины», результат представлен на рисунке 26.

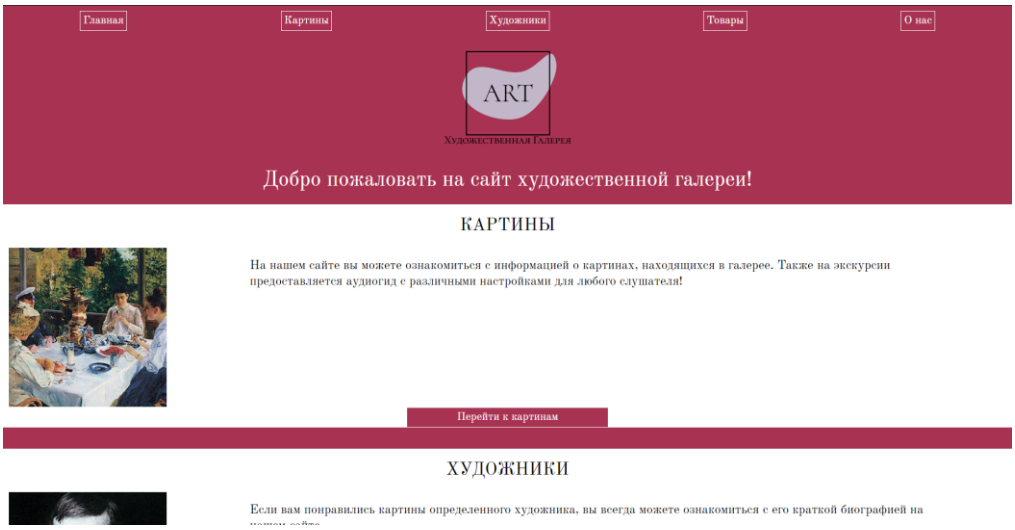

Рисунок 26 — Внешний вид страницы «Главная»

2.3.4 Раскрывающийся текст

На странице «Художники» присутствует возможность раскрытия текста.

Внутри находятся <img class=“article_image image_artist”> для изображения и , содержащий в себе таблицу и .

Данный блок «checker» состоит из тегов <input type=“checkbox”> class=“read_more” id=“read_more_*порядковый номер*”, , <label for=“read_more_*порядковый номер*” class=“button_read_more”>. На рисунке 27 представлен данный блок в HTML-документе.

```
<div class="checker">
    <input type="checkbox" class="read_more" id="read_more_1" />
    <div class="article_information">
        <p class="article_text">Родился 5 декабря 1861 года в Москве. Происходил из состоятельной
            купеческой семьи. В отротчестве переехал в Большие Мытищи, где начался его путь
            художника. В 13 лет поступил на архитектурное отделение Московского училища живописи,
            ваяния и зодчества, через год перешел на отделение живописи. Учился у Алексея Саврасова,
            Василия Поленова и Василия Перова. Много путешествовал, где самое большое впечатление на
            художника произвел Париж. Умер 11 сентября 1939 года.</p>
        <div class="hidden_part"></div>
    </div>
    <label for="read_more_1" class="button_read_more"></label>
</div>
```

Рисунок 27 — Пример HTML-кода для раскрывающегося текста

Значение элемента input меняется при нажатии на кнопку, и текст, находящийся в «article_information», раскрывается или скрывается.

Элементы в классе «checker» распределены с помощью технологии Flexbox с расположением элементов по центру. Для класса «read_more» устанавливается свойство opacity со значением 0, чтобы скрыть элемент. В «button_read_more» указывается внешний вид кнопки и текста, в том числе свойство cursor в значении pointer. Псевдокласс «:hover» обеспечивает масштабирование элемента. Описанное представлено на рисунке 28.

```css
.checker {
    width: 90%;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
}

.read_more {
    opacity: 0;
    position: absolute;
}

.button_read_more {
    cursor: pointer;
    display: inline-block;
    padding: 10px;
    margin-left: 20px;
    margin-bottom: 20px;
    color: white;
    background-color: rgb(168, 50, 82);
    transition: 0.3s;
}

.button_read_more:hover {
    transform: scale(1.1);
    color: rgb(168, 50, 82);
    background-color: white;
    box-shadow: 0 0 15px rgb(168, 50, 82);
}
```

Рисунок 28 — CSS-правила для элементов класса checker

Блок состоит из элементов <p class=“article_content”>, являющийся текстом статьи, и . С помощью последнего осуществляется скрытие текста.

Стиль класса «article_information» определяет максимальную величину внутреннего наполнения, когда текст скрыт. Ниже на рисунке 29 представлен код для «article_information».

```css
.article_information {
    max-height: 100px;
    overflow: hidden;
    position: relative;
}
```

Рисунок 29 — CSS-правило для article_information

Для класса «hidden_part» указываются такие свойства, как градиентная заливка от прозрачности до белого цвета с помощью background-color: linear-gradient(). Значение opacity для данного элемента равно 1. Ниже представлен код для «hidden_part» на рисунке 30.

```css
.hidden_part {
    position: absolute;
    bottom: 0;
    background: linear-gradient(to bottom, rgba(255, 255, 255, 0), rgba(255, 255, 255, 1) 80%);
    width: 100%;
    margin-left: 20px;
    height: 60px;
    opacity: 1;
    transition: 0.3s;
}
```

Рисунок 30 — CSS-правило для hidden_part

Изначально текст скрыт, поэтому для отображения слова «Показать» используется свойство content у «button_read_more». При дальнейшем скрытии текста слово также меняется. С помощью комбинатора тильда и псевдоэлемента «:after» селектор класса «button_read_more» связывается с «read_more» (рисунок 31).

```css
.read_more~.button_read_more::after {
    content: 'Показать';
}
```

Рисунок 31 — CSS-правило для отображения слова «Показать»

Описание поведения при нажатии на кнопку реализуется с помощью псевдокласса «:checked» для класса «read_more». Изменяются такие классы как «article_information», «hidden_part», «button_read_more». Для записи используется комбинатор тильда.

У «article_information» убирается значение максимальной высоты, у «hidden_part» значение прозрачности устанавливается в ноль, у «button_read_more» с помощью свойства content отображается слово «Скрыть». Ниже на рисунке 32 представлен код для вышеописанных действий.

```css
.read_more:checked~.article_information {
    max-height: none;
}

.read_more:checked .hidden_part {
    opacity: 0;
    transition: 0.3s;
}

.read_more:checked~.button_read_more::after {
    content: 'Скрыть';
}
```

Рисунок 32 — CSS-правило для раскрытого текста

На рисунке 33 представлен пример статьи с разворачивающимся текстом на странице «Художники».

Рисунок 33 — Пример разворачивания текста на странице «Художники»

На данном этапе создается страница «Художники», результат представлен на рисунке 34.

Рисунок 34 — Внешний вид страницы «Художники»

2.4 Разработка адаптивности сайта

Для того, чтобы сайт корректно отображался при различных размерах окон или устройств, используются медиа-запросы по максимальному значению ширины экрана в пикселях. Сайт адаптируется для различных размеров до 350 пикселей.

Выделяются несколько максимальных значений ширины: 1014, 770, 630 и 430 пикселей. Примеры ниже описаны для раздела «Художники».

При максимальном значении 1014 пикселей изменяется позиционирование элементов таким образом, что элементы находятся по центру. Убираются отступы слева у слайдеров «slider» и изображений «article_image». Также увеличивается ширина таблиц «article_table» и текста «article_text».

Ниже на рисунке 35 представлен медиа-запрос при максимальной ширине в 1014 пикселей.

```
@media (max-width: 1014px) {
    .article_content {
        justify-content: center;
        flex-wrap: wrap;
    }

    .article_image {
        padding-left: 0;
    }

    .article_text {
        width: 80%;
    }

    .article_table {
        width: 100%;
    }

    .slider {
        margin-left: 0;
    }

    .checker {
        width: 100%;
    }
}
```

Рисунок 35 — Медиа-запрос для максимальной ширины 1014 пикселей

На рисунке 36 представлен внешний вид при ширине 1013 пикселей.

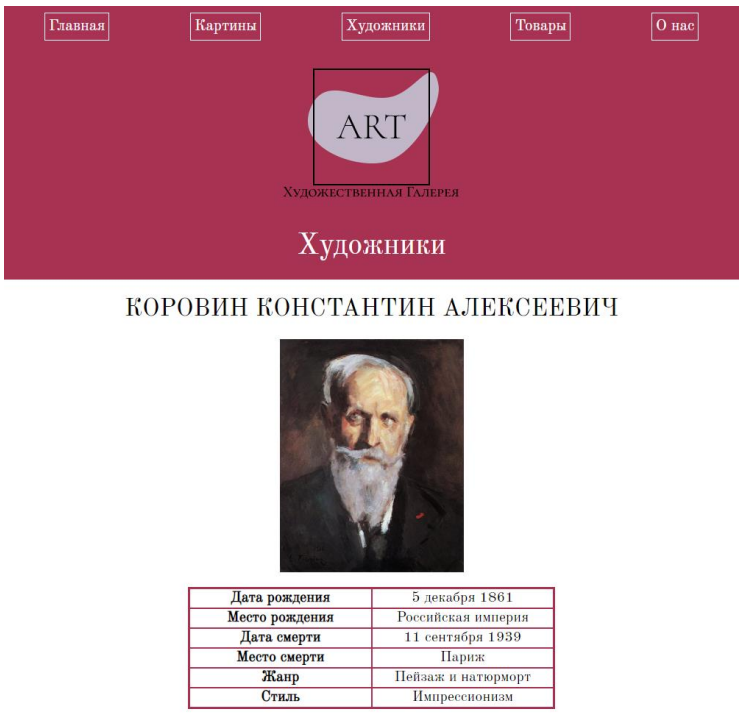

Рисунок 36 — Страница «Художники» при ширине 1013 пикселей

При максимальном значении 770 пикселей уменьшается размер шрифта для «link_decor», «header_text», «article_header», «article_text», th, td, а также размер изображений, в том числе и для слайдеров, и таблиц. Также уменьшается высота для разделителя «space».

Ниже на рисунках 37 и 38 представлен медиа-запрос при максимальной ширине в 770 пикселей.

```
@media (max-width: 770px) {
    .link_decor {
        font-size: 14pt;
    }

    .header_logo {
        width: 200px;
    }

    .header_text {
        font-size: 23pt;
    }

    .article_header {
        font-size: 20pt;
    }

    .space {
        height: 30px;
    }

    .image_main,
    .image_art,
    .image_shop {
        width: 300px;
    }
```

Рисунок 37 — Медиа-запрос для максимальной ширины 770 пикселей

```
    .article_text {
        font-size: 15pt;
    }

    .article_text_about {
        width: 80%;
    }

    table {
        width: 55%;
    }

    th,
    td {
        font-size: 11pt;
    }

    .slider {
        width: 200px;
    }

    .slider_image {
        width: 200px;
    }
}
```

Рисунок 38 — Медиа-запрос для максимальной ширины 770 пикселей

На рисунке 39 представлен внешний вид при ширине 769 пикселей.

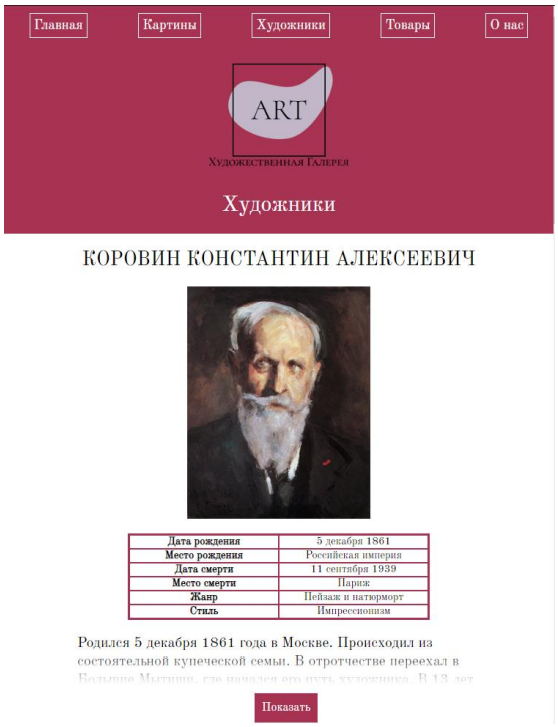

Рисунок 39 — Страница «Художники» при ширине 1013 пикселей

При максимальном значении 630 пикселей аналогично вышеописанному уменьшаются параметры ширины изображений, таблиц и размер текста, разделителя основного раздела, «шапки» и «подвала». С

помощью псевдокласса «:not(:first-child)» сокращается верхний отступ для «article_text» для страниц, где есть несколько текстов подряд.

Ниже на рисунках 40 и 41 представлен медиа-запрос при максимальной ширине в 770 пикселей.

```
@media (max-width: 630px) {
    .link_decor {
        font-size: 12pt;
    }

    .header_logo {
        width: 150px;
    }

    .header_text {
        font-size: 20pt;
    }

    .footer_logo {
        width: 100px;
        height: 100%;
    }

    .footer_text_info {
        font-size: 9pt;
        padding-top: 10px;
    }

    .article_header {
        font-size: 15pt;
    }

    .space {
        height: 25px;
    }

    .article_image {
        padding-left: 0;
    }
```

Рисунок 40 — Медиа-запрос для максимальной ширины 630 пикселей

```
    .image_main,
    .image_art,
    .image_shop {
        width: 250px;
    }

    .image_artist,
    .image_about {
        width: 200px;
    }

    .article_text {
        font-size: 12pt;
    }

    .article_text:not(:first-child) {
        padding-top: 10px;
    }

    th,
    td {
        font-size: 9pt;
    }
}
```

Рисунок 41 — Медиа-запрос для максимальной ширины 630 пикселей

На рисунке 42 представлен пример внешнего вида для раздела «Художники» при ширине 629 пикселей.

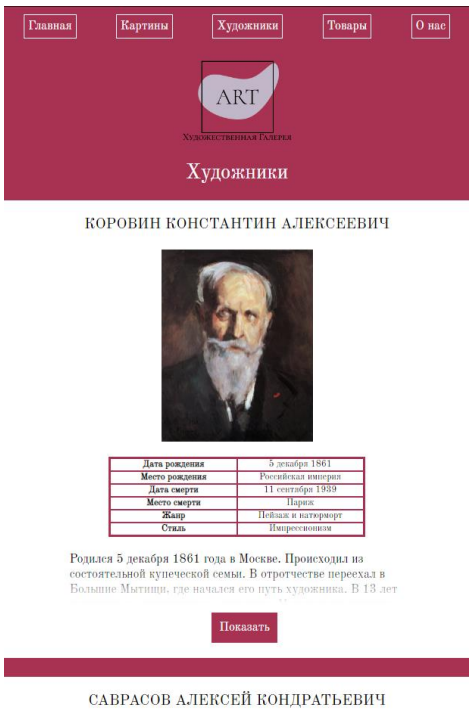

Рисунок 42 — Страница «Художники» при ширине 629 пикселей

При максимальном значении 430 пикселей уменьшаются размеры текста, изображений для страниц «Художники» и «Магазин» «image_artist» и «image_shop» соответственно, а также ширина таблиц.

Ниже на рисунке 43 представлен медиа-запрос при максимальной ширине в 770 пикселей.

```
@media (max-width: 430px) {
    .link_decor {
        font-size: 9pt;
    }

    .header_text {
        font-size: 18pt;
    }

    .image_artist,
    .image_shop {
        width: 150px;
    }

    .article_text {
        font-size: 11pt;
    }

    .article_text_about {
        width: 95%;
    }

    table {
        width: 60%;
    }
}
```

Рисунок 43 — Медиа-запрос для максимальной ширины 430 пикселей

На рисунке 44 представлен пример внешнего вида для раздела «Художники» при ширине 429 пикселей.

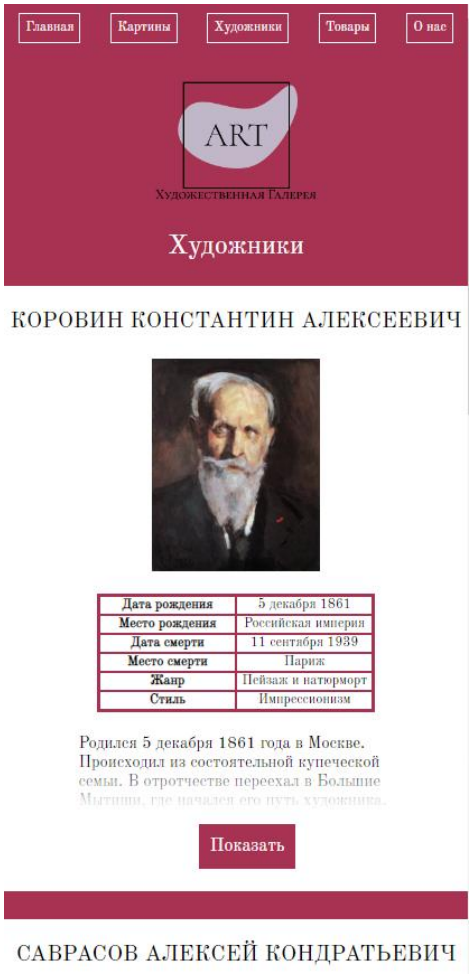

Рисунок 44 — Страница «Художники» при ширине 429 пикселей

ЗАКЛЮЧЕНИЕ

В результате курсовой работы был разработан адаптивный сайт художественной галереи с учетом разработанной структуры и применения веб-технологий.

Для достижения поставленной цели были использованы знания, полученные в процессе обучения по дисциплине «Web-технологии», а также информация из дополнительной учебной литературы.

В результате выполнения работы была достигнута цель и выполнены все поставленные задачи:

а) Была изучена предметная область и получены знания о теме работы, включая изучение основных понятий и принципов. На основе полученной информации был разработан итоговый результат;

б) Была создана структура разрабатываемого сайта, определены основные разделы, которые будут присутствовать на сайте. Была организована система навигации между страницами и определены основные функциональные элементы;

в) Были рассмотрены веб-технологии языков HTML и JavaScript, а также таблиц каскадных стилей CSS. HTML использовался для создания структуры веб-страницы. CSS применен для стилизации и визуального оформления сайта. JavaScript использовался для добавления функционала отдельных элементов на сайт;

г) На основе полученных знаний предметной области, созданной структуры сайта и использованных веб-технологий был разработан сайт. Был проведен процесс написания кода, создания веб-страниц, добавления стилей и функциональности. Отображение веб-сайта было адаптировано для различных размеров окон.

Исходный код сайта, реализованного в процессе выполнения работы, опубликован в репозитории на GitHub [6]. Для размещения веб-сайта [7] в интернете с целью дальнейшего ознакомления пользователями был использован хостинг GitHub Pages.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Беликова, С. А. Основы HTML и CSS: проектирование и дизайн веб-сайтов : учебное пособие по курсу «Web-разработка» / С. А. Беликова, А. Н. Беликов. — Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2020. — 174 с. — ISBN 978-5-9275-3435-7. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/100186.html (дата обращения: 09.04.2023). — Режим доступа: для авторизир. Пользователей

2. Богун, В. В. Web-программирование. Интерактивность статических Интернет-сайтов с применением форм : учебное пособие для СПО / В. В. Богун. — Саратов : Профобразование, Ай Пи Ар Медиа, 2020. — 65 с. — ISBN 978-5-4488-0815-9, 978-5-4497-0481-8. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/92633.html (дата обращения: 21.05.2023). — Режим доступа: для авторизир. пользователей. - DOI: https://doi.org/10.23682/92633

3. Маркин, А. В. Web-программирование : учебник / А. В. Маркин. — Москва : Ай Пи Ар Медиа, 2021. — 286 с. — ISBN 978-5-4497-1002-4. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/104883.html (дата обращения: 09.04.2023). — Режим доступа: для авторизир. Пользователей

4. Рындин, Н. А. Технологии разработки клиентских WEB-приложений на языке JavaScript : учебное пособие / Н. А. Рындин. — Воронеж : Воронежский государственный технический университет, ЭБС АСВ, 2020. — 54 с. — ISBN 978-5-7731-0888-7. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/108188.html (дата обращения: 09.04.2023). — Режим доступа: для авторизир. Пользователей

5. Торопова, О. А. Основы web-программирования. Технологии HTML, DHTML : учебное пособие / О. А. Торопова, И. Ф. Сытник. — Саратов : Саратовский государственный технический университет имени Ю.А.

Гагарина, ЭБС АСВ, 2012. — 106 с. — ISBN 978-5-7433-2606-8. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/76493.html (дата обращения: 09.04.2023). — Режим доступа: для авторизир. пользователей. - DOI: https://doi.org/10.23682/76493

6. Web-CW. [Электронный ресурс]. URL: https://github.com/KseniaDo/Web-CW (дата обращения: 21.05.2023).

7. Художественная галерея «Art». [Электронный ресурс]. URL: https://kseniado.github.io/Web-CW/ (дата обращения: 21.05.2023).