Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)**

УТВЕРЖДАЮ
Заведующий кафедрой ИСПИ
_____И.Е. Жигалов
«20» марта 2025 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ МЕЖДИСЦИПЛИНАРНОГО КУРСА

«ПРОЕКТИРОВАНИЕ, ДИЗАЙН И ВЕРСТКА ИНТЕРФЕЙСОВ»

В РАМКАХ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

«РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ НА СТОРОНЕ КЛИЕНТА»

09.02.09 Веб-разработка
Разработчик веб приложений

Владимир, 2025

Методические указания к лабораторным работам междисциплинарного курса «Проектирование, дизайн и верстка интерфейсов» разработал преподаватель КИТП Фисина И.С.

Методические указания к лабораторным работам рассмотрены и одобрены на заседании УМК специальности 09.02.09 Веб-разработка
протокол № 1 от «10» марта 2025 г.

Председатель УМК специальности _____ И.Е. Жигалов

Методические указания к лабораторным работам рассмотрены и одобрены на заседании кафедры ИСПИ
протокол № 7а от «12» марта 2025 г.

Рецензент от работодателя:
руководитель группы обеспечения
качества программного обеспечения
ООО «БСЦ МСК» _____ С.С. Смирнова

СОДЕРЖАНИЕ

Лабораторная работа №1

Создание простой веб-страницы с использованием основных элементов HTML.

Цель работы: изучение основных концепций HTML, необходимых для создания простейших файлов HTML, содержащий форматированный текст.

1. Краткое содержание теории

1.1. Структура HTML документа

В идеальном случае HTML - документ состоит из трех частей:

- информации о версии используемого HTML;

- заголовка документа;

- тела документа.

Пример простейшего HTML - документа, содержащий все структурные элементы.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
	<HEAD>
		<TITLE>HTML-документ</TITLE>
	</HEAD>
	<BODY>
		<H1>Заголовок</H1>
		<P>Первый абзац
		<P>Второй абзац
	</BODY>
</HTML>
```

Первый тэг, который должен находиться в любом HTML-документе, это <HTML> ... </HTML>. Этот тэг указывает на то, что данный документ действительно содержит в себе HTML-текст. Все, что вы напишете в своем документе, должно находиться внутри данного тэга:

```
<HTML>
	…
	Текст документа
	…
</HTML>
```

1.1.1. Заголовок документа

В заголовке (его еще называют «шапкой») HTML – документа содержаться сведения о документе: название (тема документа), ключевые слова (используемые поисковыми машинами), а также ряд других данных, которые не являются содержимым документа.

Определение заголовка должно содержаться внутри тэга <HEAD>…</HEAD>:

```
<HTML>
<HEAD>
	…
	описание заголовка
	…
```

```
</HEAD>
…
текст документа
…
</HTML>
```

В разделе описания заголовка можно указать заглавие документа, для этого используется тэг <TITLE> ... </TITLE>. Когда браузер встречает этот тэг, он отображает все, что находится внутри него как заглавие.

1.1.2. Определение тела документа

Весь остальной HTML-документ, включая весь текст, содержится внутри тэга <BODY> ...</BODY> (тело). Теперь наш документ выглядит примерно так:

```
<HTML>
      <HEAD>
            <TITLE>Это заглавие документа</TITLE>
      </HEAD>
      <BODY> …
            текст документа …
      </BODY>
</HTML>
```

Наиболее часто используемые атрибуты элемента BODY:

background – URL, указывающий расположение изображения для фона (обычно берется небольшое изображение, которое размножается для заполнения фона всего документа);

bgcolor – цвет фона HTML-документа;

text – цвет шрифта документа;

link – цвет непосещенных гиперссылок;

vlink - цвет посещенных гиперссылок;

alink – цвет гиперссылок при выборе их пользователем (при нажатии

«Enter» произойдет переход по такой ссылке);

contenteditable – позволяет разрешить или запретить пользователю редактирование содержимого HTML-документа при просмотре его браузером (значения true, false, inherit).

Все атрибуты, позволяющие задать цвет, имеют тип %Color. Значения таких атрибутов могут задаваться шестнадцатеричными числами с символом # в начале каждого числа, например:

```
bgcolor = "#FF0005"
```

Также атрибутам задания цвета можно присваивать предопределенные идентификаторы некоторых наиболее часто употребляемых цветов.

1.2. Особенности ввода текста

Для того чтобы поместить простой текст на страницу, его достаточно ввести в тело документа. При этом браузер отобразит текст со шрифтом по умолчанию и цветом, заданным для текста тела страницы. Чтобы чтение информации, содержащейся в HTML-документе, стало приятным занятием, применяется форматирование текста.

Существуют специальные команды, выполняющие перевод строки и задающие начало нового абзаца. Кроме того, имеется команда, которая запрещает программе браузера каким-либо образом изменять форматирование текста и позволяет точно воспроизвести на экране заданный фрагмент текстового файла.

Тэг перевода строки
 отделяет строку от последующего текста или графики. Тэг абзаца <P> тоже отделяет строку, но добавляет еще пустую строку, которая зрительно выделяет абзац. Оба тэга являются одноэлементными.

Задание начертания текста является, возможно, самым простым средством форматирования содержимого документа, которое доступно в HTML. Для изменения начертания текста в HTML-код вводятся элементы, обозначающие текст, написанный с соответствующим начертанием. В табл. 1 приведен список элементов, которые применяются при задании начертания текста.

Таблица 1 – Элементы задания начертания текста

Элемент	Описание
b	Полужирное начертание
i	Курсивное начертание
u	Подчеркнутый текст
strike, s	Перечеркнутый текст
big	Текст с увеличенным размером шрифта
small	Текст с уменьшенным размером шрифта
sup	Верхний индекс
sub	Нижний индекс
tt	Текст, записанный моноширинным шрифтом (все символы имеют одинаковую ширину)
blink	Мерцающий текст (редко поддерживается браузерами).

1.2.1. Задание шрифта текста

Если нужно отобразить некоторый текст с использованием определенного шрифта, а не применяемого браузером по умолчанию, то в HTML предусмотрен элемент FONT. Он вводится при помощи парных тегов <FONT> и </FONT>.

Параметры шрифта для элемента FONT устанавливаются заданием значений следующих его атрибутов:

- face — задает название шрифта, например Arial или System;

- size — задает размер шрифта (значение от 1 до 7, по умолчанию используется значение 3; для атрибута size могут использоваться только семь значений);

- color — задает цвет шрифта.

1.3. Запрет разрыва строки

Иногда бывает нужно разорвать в определенных местах строки в тексте документа, а наоборот, не допустить разделения некоторых слов в строках. Для этого HTML позволяет использовать неразрывные пробелы и элементы NOBR.

Если текст, слова которого разделены неразрывными пробелами, не помещается в окне браузера, то появится горизонтальная полоса прокрутки.

Когда необходимо выделить большой участок текста, для которого недопустим автоматический перенос слов, то целесообразнее использовать элемент NOBR.

Ему соответствуют парные теги <NOBR> и </NOBR>. Весь текст, находящийся между этими тегами, будет отображаться браузером в одной строке.

Остерегайтесь создания слишком длинных неразрывных строк, поскольку необходимость горизонтальной прокрутки для прочтения таких строк только ухудшает чтение HTML-документа. Для вставки разрывов строк в тексте, заключенном между <NOBR> и </NOBR>, можно использовать рассмотренный ранее элемент BR.

1.4. Заголовки

Важным этапом в структурировании HTML-документа является использование заголовков (в их обычном понимании) для обозначения начала больших фрагментов текста.

В HTML поддерживаются шесть видов заголовков. Им соответствуют элементы H1, H2, H3, H4, H5, H6. Элементы H1 - H6 задаются при помощи соответствующих парных тэгов.

1.5. Специальные символы

Для добавления в текст специальных символов используется числовой или именной код. В обоих случаях код символа начинается с символа амперсанда (&), за которым следует номер символа (числовой код) или сокращенное имя (именной код).

&#34(числовой код)&quot (именной код) - прямые двойные кавычки

&#60	&lt	- меньше
&#62	&gt	- больше
&#177	&plusmn	- плюс-минус.

2. Задания на лабораторную работу

2.1. Задание 1

— Создайте папку HTML, в которой вы будете сохранять сконструированные Web-страницы.

— Запустите программу VS Code.

— Наберите в окне редактора простейший текст файла HTML

— Сохраните файл под именем «Упражнение 1.html».

— Загрузите свой документ. Убедитесь, что название Web-страницы отразилось в верхней, статусной, строке браузера. Пример работы представлен на рисунке 1.

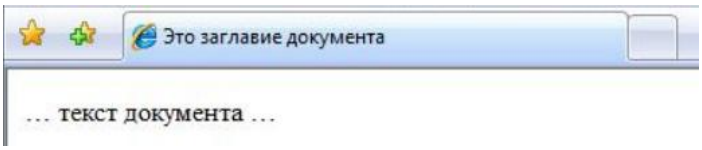

Рисунок 1 – Пример выполнения первого задания

2.2. Задание 2

— Запустите программу VS Code;

— Наберите в окне редактора простейший текст файла HTML;

— Сохраните файл под именем «Упражнение 2.html»;

— Загрузите свой документ;

— На экране вы увидите результат своей работы, пример представлен на рисунке 2.

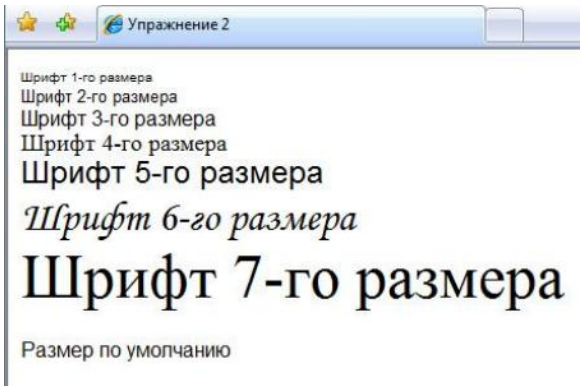

Рисунок 2 – Пример выполнения второго задания

2.3. Задание 3

— Запустите программу VS Code;

— Наберите в окне редактора текст файла HTML;

— Сохраните файл под именем «Упражнение 3.html»;

— Загрузите свой документ;

— На экране вы увидите результат своей работы, пример работы изображен на рисунке 3.

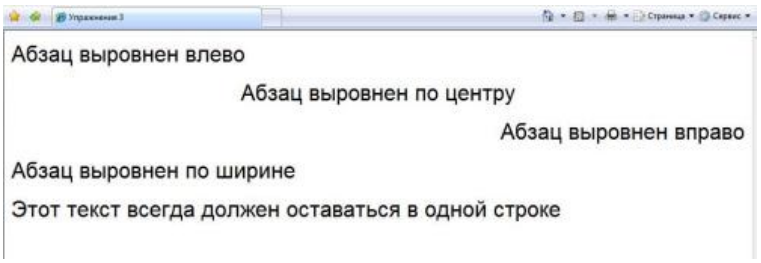

Рисунок 3 – Пример работы третьего задания

2.4. Задание 4

— Запустите программу VS Code;

— Наберите в окне редактора текст файла HTML;

— Сохраните файл под именем «Упражнение 4.html»;

— Загрузите свой документ. На экране вы увидите результат своей работы, пример представлен на рисунке 4.

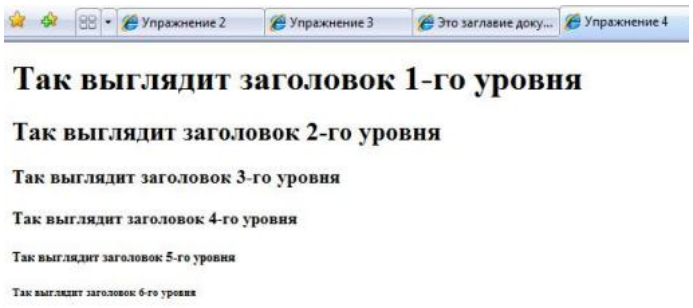

Рисунок 4 – Пример выполнения четвертого задания

2.5. Задание 5

Создайте HTML документ и внесите в его следующие правки, которые представлены на рисунке 5.

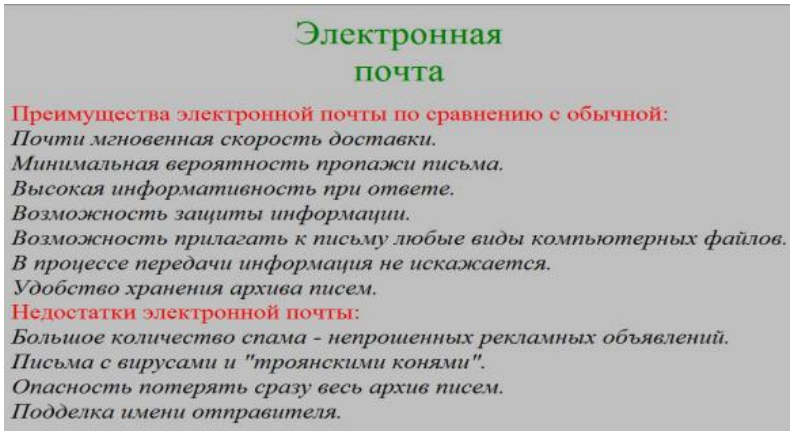

Рисунок 5 – Пример выполнения пятого задания

Содержание отчёта:

1. Титульный лист;
2. Цель работы;
3. Задание;
4. Ход работы:
    a. Описание задания, скриншот выполнения, листинг представить в приложении;
5. Вывод;
6. Приложения.

Критерии отчёта:

1. Оформление шрифтом Times New Roman (14 кегль);
2. Поля: верх 2 см, лево 3 см, право 1,5 см, нижнее 2 см;
3. Междустрочный интервал 1,5;
4. Отступ красной строки 1,25.

Контрольные вопросы:

1. Что такое HTML и для чего он используется при создании веб-страниц?
2. Какой элемент HTML используется для создания заголовков, и сколько уровней заголовков существует?
3. Каким тегом создается абзац на веб-странице?
4. Что такое атрибуты HTML и как они используются в тегах?
5. Какой тег используется для вставки изображения на веб-страницу? Какие атрибуты нужно указать для корректного отображения изображения?

6. Как создать гиперссылку на другой сайт или страницу внутри сайта с помощью HTML?
7. Чем отличается тег <ol> от тега <ul>, и в каких ситуациях их следует использовать?
8. Как вставить таблицу на веб-страницу с помощью HTML? Какие теги нужны для создания строк и ячеек таблицы?
9. Что такое семантические теги в HTML и для чего их использование считается хорошей практикой?
10. Как добавить комментарий в код HTML, и зачем это может понадобиться?
11. Какая структура базового HTML-документа? Назовите основные элементы структуры.
12. Что такое мета-теги, и какую информацию они предоставляют браузерам и поисковым системам?
13. Как правильно создавать вложенные списки в HTML?
14. Чем отличается блоковый элемент от строчного, и приведите примеры каждого типа.
15. Как можно добавить горизонтальную линию на веб-страницу с помощью HTML?

Лабораторная работа №2

Разработка многосекционной страницы с семантической разметкой

Цель работы: приобретение навыков разработки многосекционной веб-страницы с использованием семантической разметки HTML5. Студенты должны научиться структурировать веб-страницу, используя семантические теги для улучшения логики и доступности контента, а также для создания более удобного взаимодействия с браузерами и поисковыми системами.

1. Краткое содержание теории

Семантическая разметка HTML5 — это ключевая технология, используемая для структурирования веб-страниц таким образом, чтобы содержание имело четкое смысловое значение не только для пользователей, но и для поисковых систем, а также браузеров и вспомогательных технологий (например, экранных читалок для людей с нарушением зрения).

До появления HTML5 разработчики часто использовали элементы вроде и для создания структуры страницы. Однако, эти теги не передают смысл содержимого, а используются только для оформления. Семантические теги HTML5 вводят более осмысленную структуру, что улучшает:

— Удобочитаемость кода: Человеку, читающему код, становится легче понять структуру веб-страницы.

— SEO (поисковая оптимизация): Поисковые системы лучше интерпретируют содержимое страницы и могут предоставлять более релевантные результаты.

— Доступность: Семантические элементы улучшают взаимодействие с вспомогательными технологиями (экранными читалками), что важно для пользователей с ограниченными возможностями.

— Поддержка стандартов: Использование семантических тегов соответствует современным стандартам HTML5, что повышает совместимость веб-страниц с различными браузерами и устройствами.

1.1. Основные сематические тэги

Семантические теги предназначены для передачи определенного значения, контекста и роли конкретных блоков информации. Вот наиболее важные из них:

1.1.1. <header> — элемент заголовка страницы или раздела. Обычно включает логотип, навигационное меню и основную информацию о странице.

Пример:

```
<header>
    <h1>Мой сайт</h1>
    <nav>
        <ul>
            <li><a href="#home">Главная</a></li>
            <li><a href="#about">О нас</a></li>
        </ul>
    </nav>
</header>
```

1.1.2. <nav> — элемент для навигационных ссылок. Обычно включает меню с ссылками на другие страницы или разделы сайта.

Пример:

```
<nav>
    <ul>
        <li><a href="#section1">Секция 1</a></li>
        <li><a href="#section2">Секция 2</a></li>
    </ul>
</nav>
```

1.1.3. <main> — основной контент страницы. Используется для уникальной информации, присущей только этой странице (без повторяющихся элементов вроде навигации или подвала).

Пример:

```
<main>
    <section>
        <h2>О нас</h2>
        <p>Информация о компании.</p>
    </section>
</main>
```

1.1.4. <section> — раздел контента. Этот элемент используется для логической группировки содержимого, которое может быть тематически связано.

Пример:

```
<section>
    <h2>Наши услуги</h2>
    <p>Описание услуг.</p>
</section>
```

1.1.5. <article> — автономная часть контента, которая может быть независима от других разделов страницы. Подходит для статей, блогов, новостей.

Пример:

```
<article>
    <h3>Заголовок статьи</h3>
    <p>Текст статьи.</p>
</article>
```

1.1.6. <aside> — дополнительный контент, связанный с основным содержимым страницы, но не являющийся его центральной частью. Часто используется для боковых панелей с рекламой, дополнительными ссылками или информацией.

Пример:

```
<aside>
    <h2>Связанные статьи</h2>
    <ul>
        <li><a href="#article1">Статья 1</a></li>
        <li><a href="#article2">Статья 2</a></li>
    </ul>
</aside>
```

1.1.7. <footer> — элемент для подвала страницы. Обычно содержит авторские права, контактную информацию и дополнительные ссылки.

Пример:

```
<footer>
    <p>&copy; 2024 Моя компания. Все права защищены.</p>
</footer>
```

1.1.8. <figure> и <figcaption> — используются для представления изображений или иллюстраций с пояснениями. Тег <figure> группирует контент, а <figcaption> добавляет подпись.

Пример:

```
<figure>
    <img src="image.jpg" alt="Описание изображения">
    <figcaption>Подпись к изображению</figcaption>
</figure>
```

2. Задание на лабораторную работу

2.1. Взять пример сайта и разобрать на нем сематическую структуру. Выделить где находится шапка, основной контент, подвал, раздел контента и так далее (обратить внимание на п. 1 методических указаний). Пример разбора сайта представлен на рисунке 1-3.

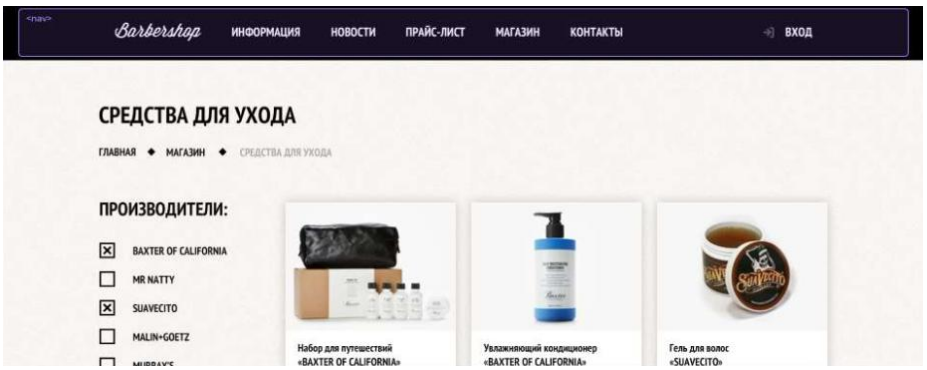

Рисунок 1 – Пример тэга <nav>

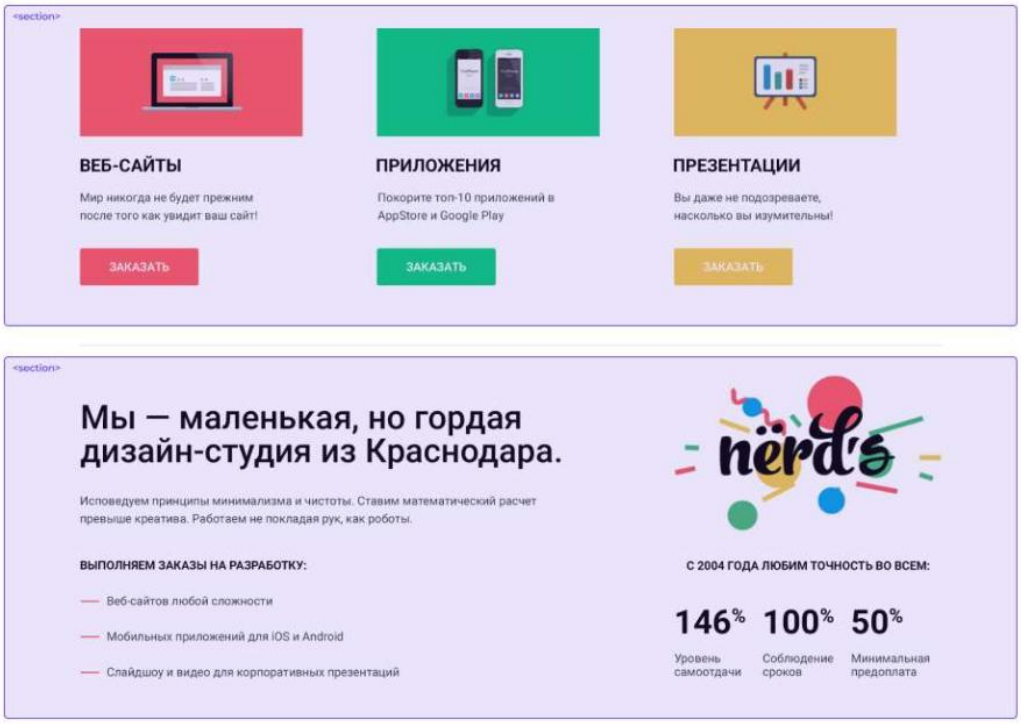

Рисунок 2 – Пример тэга <section>

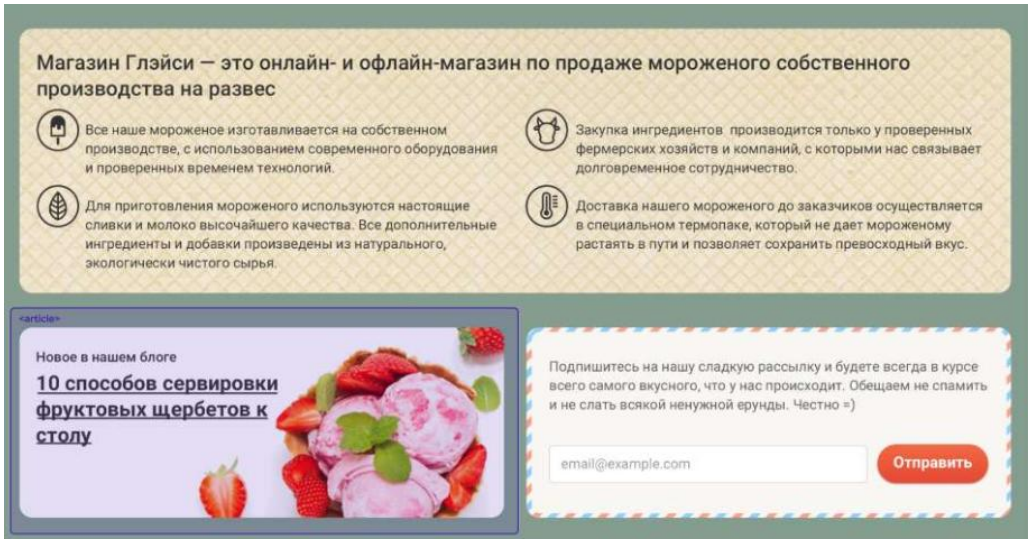

Рисунок 3 – Пример тэга <article>

2.2. Создать веб-страницу, на которой будет размещаться все основные сематические тэги. Работа по вариантам, номер в журнале = номер варианту на лабораторную работу.

Варианты заданий:

1. Создание простой многосекционной страницы

Задание: Создать веб-страницу с использованием семантических элементов HTML5, таких как <header>, <nav>, <main>, <section>, <footer>. Страница должна включать минимум три секции (например, о компании, услуги, контакты).

2. Разработка страницы с встроенной статьей

Задание: Создать страницу с отдельным разделом для статьи, используя тег <article>. Добавить заголовки и несколько параграфов. Дополнительно оформить секцию изображением с подписью, используя <figure> и <figcaption>.

3. Разработка страницы с навигационным меню

Задание: Разработать многосекционную страницу с меню навигации (использовать <nav>). Каждая ссылка в меню должна быть якорной и вести на соответствующий раздел внутри страницы.

4. Создание страницы с боковой панелью

Задание: Создать страницу с боковой панелью, использовав тег <aside>. Панель должна содержать дополнительную информацию, такую как ссылки на социальные сети или новости.

5. Разработка лендинга для мероприятия

Задание: Создать лендинг для мероприятия (например, музыкального фестиваля или конференции), разделив контент на несколько секций: «О мероприятии», «Программа», «Участники», «Контакты». Использовать теги <header>, <section>, <footer>.

6. Оптимизация страницы для SEO

Задание: Создать страницу для личного блога и добавить к ней семантические теги (например, <header>, <nav>, <article>, <footer>), включая правильное использование заголовков (от <h1> до <h6>). Написать мета-теги для улучшения SEO (title, description).

7. Создание страницы с отзывами

Задание: Разработать страницу для компании с разделом для отзывов клиентов, использовав тег <section> для основного контента и <article> для каждого отдельного отзыва.

8. Создание адаптивного веб-сайта с семантической разметкой

Задание: Создать адаптивную веб-страницу с использованием медиазапросов и семантических тегов. Структура сайта должна включать разделы, которые корректно отображаются на разных экранах (например, смартфон, планшет, компьютер).

9. Создание страницы с мультимедийным контентом

Задание: Добавить на страницу видео и аудио с описанием с помощью тегов <video>, <audio>, <figure> и <figcaption>. Разместить эти элементы в отдельном разделе.

10. Создание страницы с разделами "Вопросы и ответы"

Задание: Разработать страницу, содержащую блок вопросов и ответов (FAQ). Каждый вопрос оформить в <section>, а ответы — внутри <article>.

11. Создание одностраничного сайта для фрилансера

Задание: Разработать одностраничный сайт для фрилансера с разделами: «О себе», «Портфолио», «Отзывы», «Контакты». Каждый раздел должен быть оформлен с использованием семантических тегов.

12. Создание страницы с формой обратной связи

Задание: Разработать страницу с разделом для формы обратной связи. Использовать семантические теги для организации других разделов страницы и теги формы (<form>, <input>, <textarea> и др.).

13. Создание страницы с разделом "Новости"

Задание: Разработать страницу с разделом новостей компании, где каждая новость будет оформлена как отдельная статья с помощью <article>.

14. Создание страницы для проекта (портфолио)

Задание: Разработать сайт для проекта, включающий описание проекта, изображения, видео и отзывы. Разделы должны быть четко организованы с помощью семантических тегов.

15. Создание раздела "Наши партнеры"

Задание: Разработать страницу с секцией "Наши партнеры". Использовать теги <section> и <figure> для логичной организации списка партнеров с их логотипами и подписями.

16. Создание страницы с описанием процесса (например, как проходит заказ услуг)

Задание: Создать страницу с пошаговым описанием процесса заказа услуг. Каждый шаг оформить в отдельную секцию с использованием тегов <section> и <article>.

17. Создание страницы биографии

Задание: Разработать страницу с биографией знаменитого человека. Включить разделы для биографии, достижений, наград и интересных фактов.

18. Создание страницы с галереей работ

Задание: Разработать страницу с галереей изображений или видео. Использовать теги <figure> и <figcaption> для каждой работы.

19. Создание страницы с разделом "Часто задаваемые вопросы" (FAQ)

Задание: Создать FAQ-раздел, где каждый вопрос и ответ будет находиться в отдельной статье с использованием <article>.

20. Создание страницы с расписанием

Задание: Разработать страницу с расписанием мероприятий или занятий, используя семантические теги для логической группировки по датам и времени.

21. Создание страницы-отчета о событии

Задание: Создать страницу, посвященную прошедшему мероприятию (например, конференции), с разделами: «Описание», «Фотогалерея», «Отзывы участников».

22. Создание страницы для фестиваля

Задание: Разработать страницу для культурного фестиваля с разделами: «Программа», «Участники», «Местоположение».

23. Создание страницы с отзывами клиентов

Задание: Разработать сайт для сервиса с разделом отзывов клиентов. Каждый отзыв должен быть представлен с помощью <article>.

24. Создание сайта с блогом

Задание: Разработать страницу блога с несколькими статьями, каждая из которых оформлена в тег <article>. Добавить разделы навигации и контактов.

25. Создание одностраничного сайта для агентства

Задание: Создать сайт агентства (например, рекламного или IT) с разделами: «О компании», «Услуги», «Портфолио», «Контакты».

Содержание отчёта:

1. Титульный лист;
2. Цель работы;
3. Задание;
4. Ход работы:
    a. Описание задания, скриншот выполнения, листинг представить в приложении;
5. Вывод;
6. Приложения.

Критерии отчёта:

1. Оформление шрифтом Times New Roman (14 кегль);
2. Поля: верх 2 см, лево 3 см, право 1,5 см, нижнее 2 см;
3. Междустрочный интервал 1,5;
4. Отступ красной строки 1,25.

Контрольные вопросы:

1. Что такое семантическая разметка в HTML и зачем она используется?
2. Назовите основные семантические теги HTML5 и объясните, для чего они предназначены.
3. Какой семантический тег следует использовать для создания шапки (header) страницы, и какую информацию обычно включает этот блок?
4. Что представляет собой тег <main>, и почему важно использовать его на веб-странице?
5. Какой тег используется для создания разделов страницы, и в каких случаях его использование оправдано?

6. Какую роль играет тег <article> в структуре веб-страницы, и когда его следует использовать?
7. В чем отличие между тегами <section> и , и как правильно применять эти теги?
8. Для чего предназначен тег <footer>, и какие элементы обычно включаются в футер страницы?
9. Что такое тег <nav>, и когда следует его использовать на странице?
10. Какие преимущества даёт использование семантических тегов для поисковой оптимизации (SEO) веб-страницы?
11. Как использование семантических тегов влияет на доступность веб-страниц для людей с ограниченными возможностями (например, для пользователей экранных читалок)?
12. Как правильно структурировать многосекционную страницу с учетом семантики HTML5?
13. В чем разница между тегами <aside> и <section>, и в каких случаях уместно их использование?

Лабораторная работа №3

Разработка CSS-стилей для веб-страницы. Использование основных селекторов, свойств текста, фона, цветовой схемы

Цель работы: изучить основы использования CSS (Cascading Style Sheets) для стилизации веб-страниц. Научится применять CSS для управления внешним видом HTML-элементов, изучить основные селекторы и научится задавать стили текста, фона и цветовые схемы. Рассмотреть три способа подключения CSS к HTML: встроенные стили, подключение внешнего файла и использование внутреннего стиля в элементе <style>.

1. Краткое содержание теории

CSS (Cascading Style Sheets, или "каскадные таблицы стилей") используется для задания внешнего вида элементов на веб-странице, которая написана на языке разметки, например, HTML. CSS помогает управлять стилем элементов веб-страницы: изменять цвета, шрифты, размеры и другие визуальные параметры.

Главное удобство CSS в том, что один файл с CSS-правилами можно подключить к нескольким веб-страницам. Это означает, что если изменить, например, цвет текста или размер шрифта в CSS-файле, изменения сразу применятся ко всем страницам, которые используют этот файл.

CSS состоит из "стилей", которые представляют собой наборы свойств (атрибутов), описывающих, как должен выглядеть элемент на странице. Например, цвет текста, размер шрифта или отступы.

Основной синтаксис CSS:

```
Selector1 {
      Atribut1: znachenie;
      Atribut2: znachenie;
      …
}
Selector2 {
      Atribut1: znachenie;
      Atribut2: znachenie;
      …
}
```

1.1. Основные понятия CSS:

Селекторы — это имена элементов, к которым применяются стили. Это могут быть теги HTML, классы или уникальные идентификаторы. Атрибуты — это свойства, которые задают внешний вид элемента (например, цвет текста или размер шрифта).

К одному элементу можно применить несколько стилей, используя классы. Классы — это особые селекторы, которые позволяют задавать стили для группы элементов с одинаковыми характеристиками.

Основные селекторы:

— Теговые селекторы — применяются ко всем элементам определённого тега (например, p, h1);

— Классовые селекторы — применяются к элементам, у которых есть атрибут class (например, .classname);

— Идентификаторные селекторы — применяются к элементам с уникальным идентификатором id (например, #idname);

— Цветовые схемы: CSS поддерживает задание цвета с помощью цветовых кодов (например, #ff0000 для красного), предопределённых названий цветов (например, red), а также значений RGB, HSL и других моделей.

Свойства текста: CSS позволяет изменять шрифт, размер текста, межстрочное расстояние, выравнивание и другие характеристики текста. Примеры свойств: font-family, font-size, color, text-align.

Фон: CSS позволяет задавать фоновый цвет, изображение или градиент для любого элемента с помощью свойств background-color, background-image, background-size.

1.2. Подключение CSS к HTML:

— Встроенные стили: прописываются непосредственно внутри HTML-элемента через атрибут style.

— Внутренние стили: задаются в блоке <style> внутри <head> HTML-документа.

— Внешние стили: подключаются через файл с расширением .css с помощью тега <link>.

2. Задания на лабораторную работу

2.1. Создание веб-страницы с использованием встроенных стилей (inline styles)

— Создать простую веб-страницу с тремя элементами: заголовком, параграфом и изображением;

— Оформить заголовок с использованием встроенных стилей:

  — Цвет текста: любой по выбору;

  — Шрифт: Arial или любой другой на выбор;

  — Размер текста: 36px.

— Оформить параграф с применением следующих стилей:

  — Цвет текста: серый (#808080);

  — Выравнивание текста: по центру;

  — Межстрочное расстояние: 1.5.

— Изображение должно быть с рамкой (border) толщиной 2px, цветом black, с использованием встроенных стилей;

— Способ подключения CSS: Использование встроенных стилей через атрибут style в каждом HTML-элементе.

2.2. Создание веб-страницы с использованием внутренних стилей (internal styles)

— Создать страницу с блоками контента: заголовок, два параграфа, изображение;

— Внутри тега <head> добавить блок <style> для стилизации элементов:

  — Заголовок должен иметь размер шрифта 40px, цвет — синий;

  — Все параграфы должны быть зелёного цвета, шрифт — Tahoma, размер — 18px;

  — Изображение должно занимать 50% ширины страницы, быть центрировано (использовать display: block; margin: auto;).

— Фон веб-страницы должен быть светло-серого цвета;

— Способ подключения CSS: Использование внутренних стилей в блоке <style> внутри HTML-документа.

2.3. Создание веб-страницы с использованием внешнего CSS-файла

— Создать веб-страницу с тремя секциями: заголовок, описание и галерея изображений (3 изображения);

— Внешние стили должны быть подключены через файл styles.css, который будет содержать следующие стили:

  — Для заголовка: размер шрифта 50px, шрифт — Times New Roman, цвет текста — чёрный;

  — Для параграфов: цвет текста — тёмно-синий, отступы сверху и снизу — 20px;

  — Изображения в галерее должны иметь размер 300px на 300px, расположение в ряд с отступами между ними 10px;

  — Фон страницы — светло-голубой.

— В HTML-документе использовать тег <link> для подключения внешнего файла стилей.

— Способ подключения CSS: Внешний CSS-файл, подключённый через тег <link> в разделе <head> HTML.

Содержание отчёта:

1. Титульный лист;
2. Цель работы;
3. Задание;

4. Ход работы:
    a. Описание задания, скриншот выполнения, листинг представить в приложении;
5. Вывод;
6. Приложения.

Критерии отчёта:

1. Оформление шрифтом Times New Roman (14 кегль);
2. Поля: верх 2 см, лево 3 см, право 1,5 см, нижнее 2 см;
3. Междустрочный интервал 1,5;
4. Отступ красной строки 1,25.

Контрольные вопросы:

1. Что такое CSS и для чего он используется?
2. Какие существуют способы подключения CSS к HTML-документу?
3. Что такое селекторы в CSS и какие их основные типы?
4. Как задать цвет фона элемента в CSS?
5. Чем отличается теговый селектор от классового селектора в CSS?
6. Какие модели цвета поддерживаются в CSS и как они задаются?
7. Как с помощью CSS изменить размер шрифта и его семейство?
8. Что такое отступы (margin) и внутренние поля (padding) в CSS и как они задаются?
9. Как изменить выравнивание текста с помощью CSS?

Лабораторная работа №4

Верстка макета с использованием Flexbox для создания адаптивного лейаута

Цель работы: изучить механизм Flexbox (Flexible Box Layout) для создания адаптивных макетов веб-страниц. Освоить основные свойства и принципы Flexbox, которые позволяют легко управлять положением и выравниванием элементов, организовывать их адаптацию под различные размеры экранов и создавать гибкие и отзывчивые лейауты.

1. Краткая теория

Flexbox (Flexible Box Layout Module) — это современная технология CSS, которая предоставляет гибкую и эффективную систему для построения макетов страниц. Flexbox был разработан специально для решения проблем, связанных с выравниванием элементов и их изменением размера в условиях динамических и неизвестных размеров контейнера.

Flexbox значительно упрощает задачи, связанные с адаптивной верстkой, благодаря набору мощных инструментов для работы с расположением элементов в строках и колонках, управлением их размерами, выравниванием и распределением свободного пространства между ними.

1.1. Основные понятия и компоненты FlexBox

Flex-контейнер — это родительский элемент, к которому применяется свойство display: flex. Все дочерние элементы этого контейнера становятся flex-элементами. Flexbox автоматически выстраивает их в зависимости от указанных настроек и свойств. Flex-контейнер играет роль "контролёра" для дочерних элементов, управляя их поведением и взаиморасположением.

Пример объявления flex-контейнера:

```
.container {
      display: flex;
}
```

1.2. Главная ось и перекрестная ось

Flexbox работает с двумя основными осями:

Главная ось (Main Axis) — это ось, вдоль которой располагаются flex-элементы. По умолчанию это горизонтальная ось (в направлении слева направо). Это направление можно изменить с помощью свойства flex-direction.

Перекрёстная ось (Cross Axis) — это ось, перпендикулярная главной оси. Если flex-элементы располагаются по горизонтали, то перекрёстная ось — это вертикальная.

1.3. Основные свойства Flexbox для управления поведением контейнера

1. display: flex; — активирует режим Flexbox у элемента. Дочерние элементы становятся flex-элементами.

2. flex-direction — определяет направление расположения flex-элементов вдоль главной оси:
    a. row — элементы располагаются в строку (по умолчанию).

b. column — элементы располагаются в колонку.
c. row-reverse — элементы располагаются в строку в обратном порядке.
d. column-reverse — элементы располагаются в колонку в обратном порядке.

Пример:

```
.container {
  display: flex;
  flex-direction: row;
}
```

3. flex-wrap — задаёт поведение flex-элементов при переполнении контейнера:
   a. nowrap — элементы остаются в одной строке (по умолчанию).
   b. wrap — элементы переносятся на новую строку по мере заполнения пространства.
   c. wrap-reverse — элементы переносятся на новую строку, но в обратном порядке.

Пример:

```
.container {
  display: flex;
  flex-wrap: wrap;
}
```

4. justify-content — управляет распределением свободного пространства между flex-элементами вдоль главной оси:
   a. flex-start — элементы выравниваются по началу главной оси (по умолчанию).
   b. flex-end — элементы выравниваются по концу главной оси.
   c. center — элементы выравниваются по центру главной оси.
   d. space-between — элементы равномерно распределяются, но первый элемент у начала, а последний — у конца контейнера.
   e. space-around — элементы распределяются с одинаковыми отступами с обеих сторон.

Пример:

```
.container {
      display: flex;
      justify-content: space-between;
}
```

5. align-items — управляет выравниванием flex-элементов вдоль перекрёстной оси:
    a. stretch — элементы растягиваются по высоте контейнера (по умолчанию).
    b. flex-start — элементы выравниваются по началу перекрёстной оси.
    c. flex-end — элементы выравниваются по концу перекрёстной оси.
    d. center — элементы выравниваются по центру перекрёстной оси.
    e. baseline — элементы выравниваются по базовой линии текста.

Пример:

```
.container {
  display: flex;
  align-items: center;
}
```

6. align-content — управляет выравниванием ряда flex-элементов в случае, если доступного пространства больше, чем требуется:
    a. stretch — ряды заполняют всё доступное пространство.
    b. flex-start — ряды выравниваются по началу перекрёстной оси.
    c. flex-end — ряды выравниваются по концу перекрёстной оси.
    d. center — ряды выравниваются по центру перекрёстной оси.
    e. space-between — ряды равномерно распределяются с интервалами между ними.
    f. space-around — ряды распределяются с равными отступами по обе стороны каждого ряда.

1.4. Свойства для flex-элементов

flex-grow — указывает, насколько flex-элемент может увеличиваться относительно других элементов, чтобы заполнить свободное пространство. Если значение равно 0, элемент не увеличивается. Если больше 0, элемент растёт относительно других в соответствии с заданным числом.

Пример:

```
.item {
        flex-grow: 1;
}
```

flex-shrink — указывает, насколько flex-элемент может уменьшаться, если контейнер становится меньше. По умолчанию значение равно 1, то есть элемент сжимается. Если 0 — элемент не сжимается.

Пример:

```
.item {
        flex-shrink: 0;
}
```

flex-basis — задаёт исходный размер элемента до применения свойств flex-grow и flex-shrink. Может быть задан в пикселях, процентах или других единицах.

Пример:

```
.item {
        flex-basis: 200px;
}
```

align-self — позволяет выровнять отдельный элемент вдоль перекрёстной оси независимо от других элементов. Значения аналогичны align-items.

Пример:

```
.item {
        align-self: flex-end;
}
```

2. Задания для лабораторной работы

2.1. Создание горизонтального меню с использованием Flexbox

— Создать веб-страницу с горизонтальным навигационным меню;

— Использовать Flexbox для выравнивания пунктов меню;

— Способ подключения CSS: Встроенные стили (inline styles).

2.2. Создание адаптивной галереи изображений с использованием Flexbox

— Создать веб-страницу с галереей, содержащей 6 изображений;

— Организовать адаптивное поведение с помощью Flexbox;

— Способ подключения CSS: Стили в блоке <style> внутри <head>.

2.3. Создание адаптивного макета веб-страницы с секциями

— Создать страницу с тремя секциями;

— Использовать Flexbox для адаптивной вёрстки;

— Способ подключения CSS: Внешний CSS-файл.

Содержание отчёта:

1. Титульный лист;
2. Цель работы;
3. Задание;
4. Ход работы:
    a. Описание задания, скриншот выполнения, листинг представить в приложении;
5. Вывод;
6. Приложения.

Критерии отчёта:

1. Оформление шрифтом Times New Roman (14 кегль);
2. Поля: верх 2 см, лево 3 см, право 1,5 см, нижнее 2 см;
3. Междустрочный интервал 1,5;
4. Отступ красной строки 1,25.

Контрольные вопросы:

1. Что такое Flexbox?
2. Для чего используется свойство flex-direction?
3. Как работают flex-grow и flex-shrink?
4. В чём отличие justify-content и align-items?

Лабораторная работа №5

Создание адаптивной веб-страницы с использованием Media Queries. Применение разных стилей для мобильных и десктопных устройств.

Цель работы: изучить механизм CSS Media Queries и научиться применять его для создания адаптивных веб-страниц, которые корректно отображаются на различных устройствах с разными размерами экрана (мобильные телефоны, планшеты, десктопы). Освоить настройку различных стилей для различных типов устройств, что позволит создавать удобные для пользователя интерфейсы на всех экранах.

1. Краткое содержание теории

Адаптивная вёрстка — это подход к веб-разработке, при котором веб-страница автоматически подстраивается под размеры экрана устройства. Этот подход повышает удобство использования веб-сайта на различных устройствах, начиная от мобильных телефонов и заканчивая большими десктопными экранами. Адаптивная вёрстка реализуется с помощью механизма CSS Media Queries.

1.1. Основные понятия и концепции Meadia Queries

Media Queries — это технология CSS3, которая позволяет применять различные стили в зависимости от характеристик устройства, на котором отображается веб-страница. Основной характеристикой является ширина экрана, но также можно учитывать высоту экрана, ориентацию (альбомная или портретная), разрешение экрана и другие параметры.

Синтаксис Meadia Queries выглядит следующим образом:

```
@media (условие) {
  /* Стили, которые применяются при выполнении условия */
}
```

Пример:

```
@media (max-width: 768px) {
  body {
    background-color: lightblue;
  }
}
```

В данном случае фон страницы станет светло-голубым, если ширина экрана будет 768 пикселей или меньше, что характерно для планшетов и мобильных устройств.

1.2. Основные типы Meadia Queries

1. max-width и min-width:

— max-width — применяет стили, если ширина экрана меньше или равна указанному значению.

— min-width — применяет стили, если ширина экрана больше или равна указанному значению.

Примеры:

```
@media (max-width: 600px) {
  /* Стили для экранов до 600 пикселей (мобильные устройства) */
}

@media (min-width: 1024px) {
  /* Стили для экранов от 1024 пикселей (десктопы) */
```

```
}
```

2. min-height и max-height:

— Эти медиазапросы определяют высоту экрана. Могут использоваться для адаптации дизайна в зависимости от вертикального размера экрана.

Пример:

```
@media (min-height: 500px) {
  /* Стили для экранов высотой более 500 пикселей */
}
```

3. **orientation** — определяет ориентацию экрана: альбомная (горизонтальная) или портретная (вертикальная)

— orientation: landscape — альбомная ориентация;

— orientation: portrait — портретная ориентация.

Пример:

```
@media (orientation: landscape) {
  /* Стили для альбомной ориентации (широкие экраны) */
}
```

1.3. Комбинирование условий Media Queries

С помощью логических операторов можно комбинировать различные условия для более точного управления стилями.

1. and — объединяет два или более условия. Все условия должны быть выполнены для применения стилей. Пример:

```
@media (min-width: 600px) and (max-width: 1024px) {
  /* Стили для экранов шириной от 600 до 1024 пикселей (например,
планшеты) */
}
```

2. not — исключает стили при выполнении условия. Пример:

```
@media not screen and (max-width: 600px) {
  /* Стили применяются ко всем устройствам, кроме экранов до 600
пикселей */
}
```

3. only — применяется только к указанным типам медиа. Пример:

```
@media only screen and (max-width: 768px) {
  /* Стили для экранов до 768 пикселей (веб-страницы) */
}
```

1.4. Mobile First подход

Mobile First — это подход к созданию адаптивных веб-страниц, при котором разработка начинается с мобильных версий (с меньшего размера экрана), а затем добавляются стили для больших экранов. Это связано с тем, что мобильные устройства являются важным сегментом интернет-пользователей.

Пример подхода Mobile First:

```
/* Базовые стили для мобильных устройств */
body {
  font-size: 14px;
}

/* Стили для планшетов и более широких экранов */
@media (min-width: 768px) {
  body {
    font-size: 16px;
  }
}

/* Стили для десктопов */
@media (min-width: 1024px) {
  body {
    font-size: 18px;
  }
}
```

1.5. Brakpoints (Точки перелома)

Breakpoints — это условные точки, при которых дизайн изменяется в зависимости от ширины экрана. Эти точки перелома определяют, когда элементы страницы должны адаптироваться к новому размеру экрана. Общепринятые точки перелома:

— 320px (мобильные устройства, маленькие экраны);

— 768px (планшеты);

— 1024px (ноутбуки и десктопы);

— 1200px и выше (большие экраны).

1.6.Адаптация сетки и элементов с помощью Media Queries

— Изменение расположения блоков. Например, на мобильных устройствах элементы могут располагаться в колонку, а на десктопах — в строку.

— Изменение размеров шрифтов и изображений для лучшего отображения на разных устройствах.

— Управление видимостью элементов. Например, можно скрыть некоторые элементы на мобильных устройствах.

```
.container {
  display: flex;
  flex-direction: column;
}

@media (min-width: 1024px) {
  .container {
    flex-direction: row;
  }
}
```

2. Задания на лабораторную работу

2.1. Создание адаптивной веб-страницы с изменением фона и шрифтов

1. Создать веб-страницу с заголовком, абзацем текста и фоновым изображением;

2. Использовать Media Queries для изменения фона и размера шрифта в зависимости от ширины экрана:

— Для мобильных устройств (ширина экрана меньше 600px) — фоновый цвет должен быть светлым, шрифт — мелким;

— Для экранов от 600px до 1024px — фоновое изображение, шрифт увеличен;

— Для экранов шириной больше 1024px — другой фон, шрифт крупный.

3. Способ подключения CSS: Внешний файл CSS.

2.2. Создание адаптивной сетки с двумя колонками

1. Создать веб-страницу с двумя колонками текста (например, статьи и боковой панели);

2. На мобильных устройствах (ширина экрана до 600px) обе колонки должны отображаться друг под другом;

3. На экранах шириной от 600px до 1024px колонки должны располагаться в строку;

4. На больших экранах колонки должны занимать равное пространство;

5. Способ подключения CSS: Внутренний стиль в теге <style>.

2.3. Создание адаптивного меню с Media Queries

1. Создать веб-страницу с навигационным меню;
2. На мобильных устройствах (до 600px) меню должно быть скрыто и отображаться по клику на кнопку "Меню";
3. На планшетах (от 600px до 1024px) меню должно быть горизонтальным с небольшими отступами;
4. На десктопах (от 1024px и выше) меню должно быть статичным и занимать всю ширину экрана;
5. Способ подключения CSS: встроенный.

Содержание отчёта:

1. Титульный лист;
2. Цель работы;
3. Задание;
4. Ход работы:
    a. Описание задания, скриншот выполнения, листинг представить в приложении;
5. Вывод;
6. Приложения.

Критерии отчёта:

1. Оформление шрифтом Times New Roman (14 кегль);
2. Поля: верх 2 см, лево 3 см, право 1,5 см, нижнее 2 см;
3. Междустрочный интервал 1,5;
4. Отступ красной строки 1,25.

Контрольные вопросы:

1. Что такое Media Queries и для чего они используются?
2. Какой синтаксис используется для определения Media Queries в CSS?
3. Как использовать Media Queries для изменения стилей в зависимости от ширины экрана?
4. В чём разница между max-width и min-width в Media Queries?

5. Как применить разные CSS-файлы для мобильных и десктопных устройств с помощью Media Queries?
6. Как правильно организовать стили в CSS для адаптивного веб-дизайна?
7. Какие преимущества предоставляет адаптивный веб-дизайн?
8. Как можно тестировать адаптивность веб-страницы?
9. В каких случаях Media Queries могут не сработать?
10. Как Media Queries влияют на производительность веб-страницы?

Лабораторная работа №6

Использование CSS Grid для верстки многоуровневого макета с элементами, которые изменяются в зависимости от размера экрана.

Цель работы: изучить механизм CSS Grid для создания сложных многоуровневых макетов. CSS Grid позволяет гибко и эффективно управлять расположением элементов на странице и создавать макеты, адаптирующиеся к изменениям размеров экрана. Научится использовать основные свойства CSS Grid, определять структуру макета, управлять колонками и строками, а также применять адаптивные стили для различных экранов.

1. Краткое описание теории

CSS Grid Layout (или просто Grid) — это двухмерная система вёрстки, которая позволяет создавать сложные макеты с использованием сетки из строк и колонок. В отличие от других методов вёрстки, таких как Flexbox (который лучше подходит для одноосных макетов), Grid предоставляет мощные инструменты для работы сразу с двумя осями — горизонтальной (строки) и вертикальной (колонки).

CSS Grid делает возможным создание адаптивных макетов, которые могут изменять своё расположение и структуру в зависимости от размера экрана или других характеристик устройства. Grid Layout идеально подходит для организации сложных веб-страниц, включающих элементы различного размера и формы, которые должны автоматически подстраиваться под изменяющиеся условия.

1.1. Основные понятия CSS Grid

Grid-контейнер — это родительский элемент, которому присваивается свойство display: grid. Все дочерние элементы этого контейнера автоматически становятся Grid-элементами и располагаются в соответствии с настроенной сеткой.

Пример:

```
.container {
        display: grid;
}
```

1.2. Определение сетки (grid-template-rows и grid-template-columns)

Для создания сетки необходимо задать количество строк и колонок с помощью свойств grid-template-rows и grid-template-columns. Эти свойства задают размеры строк и колонок соответственно.

Пример:

```
.container {
        display: grid;
        grid-template-columns: 200px 1fr 200px;
        grid-template-rows: auto 300px auto;
}
```

В этом примере:

— Сетка состоит из 3 колонок: первая и третья фиксированной ширины 200px, а средняя — гибкая (1fr), заполняет оставшееся пространство.

— Сетка также содержит 3 строки: первая и третья автоматически подстраиваются по содержимому, а вторая имеет фиксированную высоту 300px.

1.3. Fr (fractional unit)

Единица fr используется для задания гибких размеров колонок и строк. Она указывает долю от доступного пространства. Например, значение 1fr означает, что элемент займет одну долю от свободного пространства, а 2fr — в два раза больше.

Пример:

```
.container {
        display: grid;
        grid-template-columns: 1fr 2fr 1fr;
}
```

Здесь средняя колонка займет две трети доступного пространства, а боковые — по одной трети.

1.4. Grid GAP (размеры между ячейками)

Свойство grid-gap или его производные column-gap и row-gap задают отступы между строками и колонками сетки. Оно позволяет контролировать расстояние между элементами.

Пример:

```
.container {
        display: grid;
        grid-gap: 20px;
}
```

1.5. Выравнивание элементов

justify-items — задаёт горизонтальное выравнивание элементов внутри ячеек.

align-items — задаёт вертикальное выравнивание элементов внутри ячеек.

justify-content — управляет горизонтальным выравниванием всей сетки внутри контейнера.

align-content — управляет вертикальным выравниванием всей сетки внутри контейнера.

Пример:

```
.container {
  display: grid;
  justify-items: center;
  align-items: start;
}
```

1.6. Объединение ячеек

Grid позволяет объединять несколько ячеек в одну с помощью свойств grid-column и grid-row. Это полезно для создания многоуровневых макетов.

Пример:

```
.item1 {
        grid-column: 1 / 3; /* Элемент займет колонки с 1 по 2 */
        grid-row: 1 / 2; /* Элемент займет первую строку */
}
```

1.7. Адаптивность с помощью Media Queries

Grid Layout хорошо сочетается с Media Queries для создания адаптивных макетов. С помощью медиазапросов можно изменять структуру сетки в зависимости от размера экрана.

Пример:

```
.container {
```

```
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}

@media (max-width: 600px) {
  .container {
    grid-template-columns: 1fr;
  }
}
```

Здесь сетка состоит из трёх колонок на больших экранах и одной колонки на экранах шириной до 600px.

2. Задания по лабораторной работе

2.1. Создание простого трёхколоночного макета с Grid

— Создать веб-страницу с макетом, состоящим из трёх колонок: основная контентная часть в центре и две боковые панели;

— Использовать CSS Grid для создания сетки, где боковые панели имеют фиксированную ширину, а центральная колонка — гибкая;

— Адаптировать макет для мобильных устройств: на экранах меньше 600px все три секции должны располагаться вертикально.

Способ подключения CSS: Использовать встроенные стили (inline styles).

2.2. Создание сетки изображений с изменяющимся количеством колонок в зависимости от ширины экрана

— Создать страницу с галереей изображений;

— Использовать CSS Grid для организации изображений в сетку, которая на больших экранах состоит из 4 колонок, на планшетах (до 900px) — из 2 колонок, а на мобильных устройствах (до 600px) — из 1 колонки;

— Адаптировать изображения для корректного отображения в ячейках.

Способ подключения CSS: Использовать внутренние стили в блоке <style>.

2.3. Создание многоуровневого макета с заголовком, навигацией и контентом

— Создать многоуровневый макет, включающий заголовок, боковую панель с навигацией, основную контентную область и подвал.

— Использовать Grid для организации макета: заголовок должен занимать всю ширину, контент и навигация — две колонки, подвал — нижнюю часть страницы.

— Адаптировать макет для мобильных устройств (до 600px): навигация должна перемещаться в верхнюю часть страницы под заголовок.

Способ подключения CSS: Использовать внешний CSS-файл.

Содержание отчёта:

1. Титульный лист;
2. Цель работы;
3. Задание;
4. Ход работы:
    a. Описание задания, скриншот выполнения, листинг представить в приложении;
5. Вывод;
6. Приложения.

Критерии отчёта:

1. Оформление шрифтом Times New Roman (14 кегль);
2. Поля: верх 2 см, лево 3 см, право 1,5 см, нижнее 2 см;
3. Междустрочный интервал 1,5;
4. Отступ красной строки 1,25.

Контрольные вопросы:

1. Что такое CSS Grid и для чего он используется?
2. Как задать сетку с определённым количеством строк и колонок?
3. Что такое единица измерения fr и как она используется в Grid?
4. Как объединять ячейки в сетке с помощью свойств grid-column и grid-row?
5. Чем CSS Grid отличается от Flexbox? Когда целесообразно использовать Grid, а когда Flexbox?
6. Как можно управлять выравниванием элементов внутри сетки с помощью свойств justify-items и align-items?
7. Какое свойство используется для управления расстоянием между строками и колонками сетки?

8. Как работает grid-template-areas и в каких случаях его следует использовать?

Лабораторная работа №7

Разработка низкоуровневого прототипа пользовательского интерфейса для веб-приложения.

Цель работы: научить студентов разрабатывать низкоуровневые прототипы пользовательских интерфейсов (UI) для веб-приложений. Прототипирование на низком уровне — это процесс создания упрощённой визуальной структуры пользовательского интерфейса, не учитывающей точные детали дизайна, но отображающей основное расположение и функциональность элементов. В ходе работы научится строить иерархию интерфейсных элементов, упрощённо представлять их функциональность и готовить базовую структуру страницы для последующей разработки.

1. Краткое описание теории

Прототипирование — это процесс создания упрощённой версии веб-приложения или его интерфейса, которая отображает основные элементы и их расположение, взаимодействие между частями интерфейса и функциональные зоны. Прототипы помогают планировать пользовательский опыт (UX) и визуализировать, как будет взаимодействовать пользователь с интерфейсом.

Прототипы делятся на несколько уровней:

— Низкоуровневый прототип — это самый упрощённый, схематичный макет веб-страницы или приложения, представляющий расположение ключевых элементов (заголовки, кнопки, поля, списки) без точного дизайна (цветов, шрифтов, картинок). Его задача — показать функциональную структуру интерфейса;

— Высокоуровневый прототип — включает уже более точные графические элементы, шрифты, цвета и более детализированное поведение интерфейса.

Зачем нужны низкоуровневые прототипы?

1. Быстрая визуализация идеи: Низкоуровневые прототипы позволяют быстро представить структуру будущего интерфейса без проработки мелких деталей дизайна;

2. Экономия времени и ресурсов: Проектирование всех деталей интерфейса на ранней стадии может быть затратным. Прототипирование помогает избежать переработки на финальных этапах;

3. Коммуникация с командой и клиентами: Прототип — это инструмент для обсуждения и утверждения структуры с заказчиками и командой разработчиков до начала разработки;

4. Тестирование взаимодействия: Прототип можно использовать для оценки юзабилити до создания финального продукта.

Инструменты для прототипирования:

1. Ручное прототипирование (бумажное): Самый простой способ прототипирования — рисование на бумаге.

2. Цифровые инструменты для прототипирования:

- Figma: Онлайн-редактор для создания как низкоуровневых, так и высокоуровневых прототипов;
- Sketch: Программа для разработки дизайна интерфейсов и прототипов;
- Adobe XD: Инструмент для создания прототипов с возможностью анимации;
- Axure RP: Профессиональный инструмент для создания прототипов с функциональными возможностями.

Основные элементы интерфейса для прототипирования

Навигационные элементы:

- Меню (горизонтальное или вертикальное);
- Панели навигации;
- Выпадающие списки.

Взаимодействие с пользователем:

- Кнопки (основные действия, дополнительные действия);
- Поля ввода данных (текстовые поля, чекбоксы, радиокнопки);
- Форма отправки данных.

Информационные блоки:

- Заголовки;
- Описание (параграфы текста);
- Изображения, иконки.

Организация данных:

- Таблицы и списки;

— Карточки;

— Модальные окна.

Шаги разработки низкоуровневого прототипа:

1. Сбор требований: Понимание функциональных задач интерфейса, которые он должен решать;

2. Планирование структуры: Определение основных блоков (шапка, меню, контентная часть, подвал и т.д.);

3. Создание схемы (Wireframe): Построение простого наброска интерфейса с расположением ключевых элементов;

4. Прототипирование: Создание низкоуровневого прототипа в выбранном инструменте (или на бумаге);

5. Тестирование и доработка: Оценка функциональности прототипа, получение обратной связи от пользователей или команды.

2. Ход работы

2.1. Заходим на сайт Figma (https://www.figma.com/) рисунок 1.

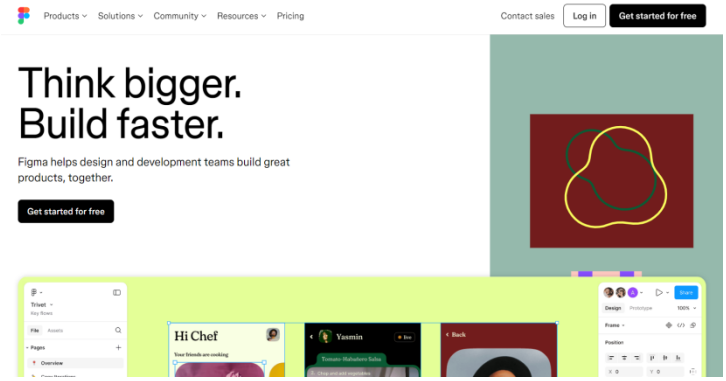

Рисунок 1 – Сайт Figma

2.2. Регистрируемся на данном сайте. Регистрация доступна по кнопке «Log in». Далее вы попадаете на главный экран проектов. Первое, что нужно сделать – это создать новый проект. Для этого нажимаем на кнопку «New design file» (Рис. 2).

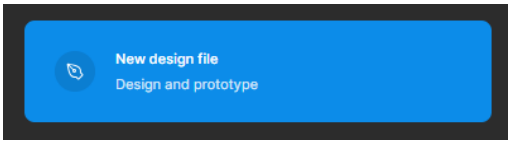

Рисунок 2 – Создание проекта

2.3. После создания проекта рассмотрим основные элементы Figma. Обратим внимание на верхнюю строчку, на ней расположены главные элементы для создания дизайна будущего сайта (Рис. 3).

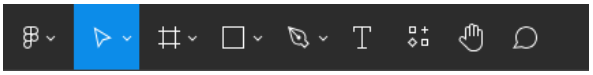

Рисунок 3 – Элементы создания Figma

Рассмотрим каждый из этих элементов:

2.3.1. Рамки (Frames). Рамки — это основной контейнер для элементов дизайна в Figma. Они используются для создания отдельных экранов приложения или страниц веб-сайта. Важно отметить, что рамки могут быть любого размера и адаптироваться под размеры различных устройств (мобильные телефоны, планшеты, десктопы).

Для создания рамки выберите инструмент Frame (F) (Рис. 4) на панели инструментов и нарисуйте область на холсте или выберите предустановленные размеры для мобильных, планшетов и десктопов.

Рамки могут быть вложенными друг в друга, что позволяет создавать сложные макеты с несколькими уровнями элементов.

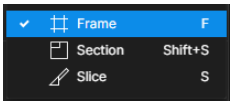

Рисунок 4 – Создание рамки

2.3.2. Фигуры и геометрические элементы (Shapes)

Фигуры (квадраты, прямоугольники, круги, линии) используются для создания базовых элементов интерфейса, таких как кнопки, карточки, иконки или контейнеры. Эти формы легко редактируются и могут служить каркасом для более сложных компонентов.

Фигуры создаются с помощью инструментов Rectangle (R), Ellipse (O), Line (L) и других, которые доступны на панели инструментов (Рис. 5).

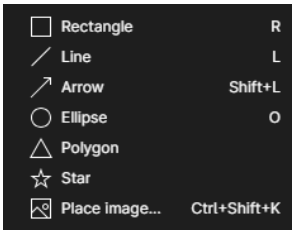

Рисунок 5 – Создание фигур

2.3.3. Текстовые поля (Text)

Текстовые поля используются для добавления текста в ваш прототип — это могут быть заголовки, подписи, абзацы, кнопки и другие текстовые элементы. Тексты могут быть настроены по размеру, шрифту, межстрочному интервалу и цвету.

Инструмент Text (T) (Рис. 6) на панели инструментов позволяет добавить текстовый блок на холст.

Рисунок 6 – Создание текста

2.3.4. Компоненты (Components)

Компоненты — это повторно используемые элементы, такие как кнопки, карточки, меню и другие элементы интерфейса (Рис. 7). Создание компонентов позволяет ускорить процесс дизайна и обеспечить единообразие на всех экранах прототипа.

Чтобы создать компонент, выберите элемент или группу элементов и нажмите Create Component или используйте комбинацию клавиш Ctrl + Alt + K (Windows) или Cmd + Option + K (Mac).

Компоненты могут быть обновлены, и все их экземпляры автоматически синхронизируются.

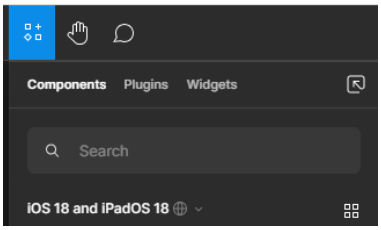

Рисунок 7 – Компоненты

2.3.5. Сетки (Grids)

Сетки используются для упорядочивания и выравнивания элементов на макете. Они помогают проектировать интерфейс в соответствии с определёнными правилами композиции и улучшить восприятие структуры экрана.

Чтобы добавить сетку, выберите рамку и активируйте функцию сетки в правой панели свойств. Сетка может быть сеточной или в виде колонок.

Чтобы создать сетку сначала нужно сделать Frames (Рис.8), данный «фрейм» будет служить основой для нашего будущего сайта.

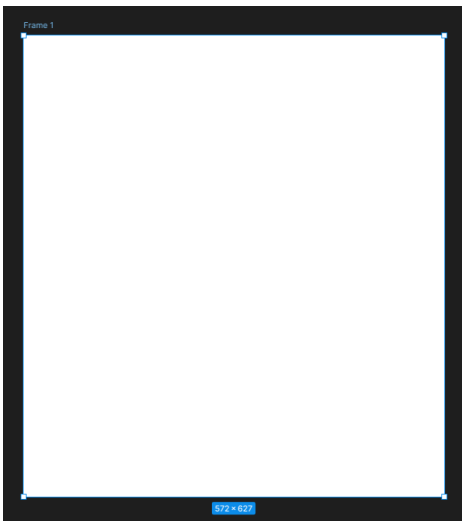

Рисунок 8 – Создание фрейма

Как мы видим на левой боковой панели создался список, в который входит и наша основа. Данный список называется «Layers Panel» или иерархия слоев, она показывает структуру всех элементов на холсте. Вы можете видеть иерархию элементов, группировать их, скрывать или блокировать для редактирования. Работа с иерархией слоёв позволяет легко организовать макет и найти нужные элементы.

Для создания сетки в правой боковой панели есть пункт «Layout grid». Нажимаем «+» у данного пункта и получаем нашу сетку, но она еще не корректна. Для настройки, требуется нажать на саму сетку в данном пункте и слева появляется еще одно окно, в котором расширенные настройки для данной сетки.

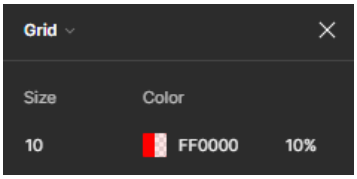

Рисунок 9 – Расширенные настройки для сетки

В данных настройках мы можем поменять цвет, размер, а также прозрачность линий нашей сетки. Для создания сетки по столбцам нужно перейти из раздела Grid в раздел Columns. Тогда у нас будет сетка только по столбцам нашего фрейма. При переходе добавляться новые поля настроек – это количество столбцов, также цвет, тип выравнивания нашей сетки, ширина столбцов, отступ от краев «фрейма», а также расстояние между столбцами. При настройках в 4 столбца, отступ от края 50, выравнивание по ширине и расстояние между столбцами в 30 получается следующая сетка:

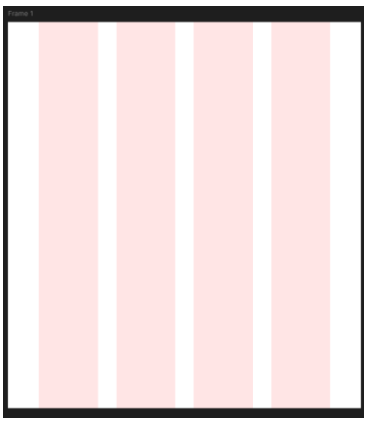

Рисунок 10 – Создание сетки

Тоже самое можно сделать для сетки по рядам, только уже данное действие происходит в подпункте «Rows».

3. Задания на лабораторную работу

3.1. Создайте дизайн сайта по вариантам, темы которых написаны в ЛР №2. Дизайн делайте только главной страницы. На сайте должна быть понятная структура, выделены главные блоки, а также должно использоваться правило золотого сечения. Все элементы должны быть выставлены по сетке. Примерный стиль сайта – минимализм.

Структура сайта:

- Шапка (Навигация, контактная информация, например, номер телефона или адрес);
- Основной контент (Статьи, разные блоки с услугами, товарами, какие-то заметки и так далее);
- Галерея изображений;
- Форма обратной связи, либо же пожелание проекту;
- Подвал (контактная информация, ссылки на другие страницы, ссылки на соц. Сети или плашки на загрузку мобильных приложений).

Содержание отчёта:

1. Титульный лист;
2. Цель работы;
3. Задание;
4. Ход работы:
    a. Описание предметной области;
    b. Описание задания, скриншот выполнения, ссылка на проект в фигма.

5. Вывод;

Критерии отчёта:

1. Оформление шрифтом Times New Roman (14 кегль);
2. Поля: верх 2 см, лево 3 см, право 1,5 см, нижнее 2 см;
3. Междустрочный интервал 1,5;
4. Отступ красной строки 1,25.

Контрольные вопросы:

1. Что такое низкоуровневый прототип и каковы его основные отличия от высокоуровневого?
2. В каких ситуациях необходимо создавать низкоуровневый прототип пользовательского интерфейса?
3. Какие элементы интерфейса необходимо учитывать при разработке прототипов веб-приложений?
4. Какие инструменты существуют для прототипирования пользовательских интерфейсов?
5. В чём преимущества низкоуровневого прототипирования на начальных этапах разработки?
6. Как осуществляется тестирование низкоуровневого прототипа и какие аспекты важно учитывать?
7. Какие ключевые задачи решает прототипирование в процессе веб-разработки?
8. Как организовать взаимодействие с пользователем через элементы интерфейса в прототипах?

Лабораторная работа №8

Анализ существующего веб-интерфейса на основе принципов UI/UX-дизайна

Цель работы: анализ существующего веб-интерфейса с точки зрения принципов UI (пользовательский интерфейс) и UX (опыт пользователя) дизайна. Научиться оценивать качество интерфейса, выявлять его сильные и слабые стороны, а также предлагать улучшения для повышения удобства использования и восприятия.

1. Краткое описание теории

1.1. UI и UX. Основные понятия

UI (User Interface) — это визуальная часть интерфейса, которая включает все элементы, с которыми взаимодействует пользователь (кнопки, поля ввода, иконки, меню и т.д.). Основной задачей UI-дизайна является обеспечение привлекательности, простоты и интуитивности взаимодействия пользователя с интерфейсом.

UX (User Experience) — это совокупность впечатлений и опыта пользователя от взаимодействия с веб-продуктом. Основной задачей UX-дизайна является создание продукта, который удовлетворяет потребности пользователя и обеспечивает комфортное взаимодействие.

1.2. Принципы UI/UX дизайна

— Простота и понятность. Интерфейс должен быть простым и легким для понимания. Интуитивность использования играет ключевую роль в быстром освоении интерфейса новыми пользователями.

— Консистентность. Все элементы интерфейса должны быть единообразными по стилю, форме и поведению на разных страницах или экранах приложения. Это включает единые шаблоны для кнопок, полей, меню и цветовой схемы.

— Интерактивность и обратная связь. Важным аспектом является наличие обратной связи на действия пользователя (например, изменение цвета кнопки при наведении, уведомления об успешном завершении операции и т.д.).

— Доступность (Accessibility). Интерфейс должен быть доступен для всех пользователей, включая людей с ограниченными возможностями. Это предполагает корректную работу с экранными читалками, достаточный контраст текста и фона, поддержку клавиатурной навигации и т.д.

— Адаптивность и отзывчивость. Интерфейс должен хорошо отображаться на разных устройствах и экранах (мобильные телефоны, планшеты, компьютеры). Это особенно важно в условиях роста использования мобильных устройств.

— Логическая структура и навигация. Веб-интерфейс должен иметь понятную иерархию информации и удобную навигацию, чтобы пользователи могли легко находить нужные разделы и функции.

— Эстетичность. Эстетический аспект также важен, так как визуально приятные интерфейсы могут вызывать положительные эмоции и повышать доверие к продукту.

— Время загрузки. Скорость загрузки веб-страницы напрямую влияет на UX. Чем быстрее загружается страница, тем лучше опыт пользователя.

1.3. Методы анализа веб-интерфейсов

— Heuristic Evaluation (эвристическая оценка) — метод анализа, в ходе которого интерфейс проверяется на соответствие принципам удобства использования (Usability Heuristics) по Якобу Нильсену.

— User Testing (пользовательское тестирование) — реальное тестирование веб-интерфейса на группе пользователей для анализа их поведения и выявления проблем.

— Анализ тепловых карт (Heatmap Analysis) — метод, который анализирует, на какие части страницы пользователи чаще всего обращают внимание.

— Метод экспертной оценки — когда профессионалы или эксперты в области UI/UX-дизайна оценивают интерфейс по различным критериям (навигация, доступность, логика и пр.).

2. Ход работы

2.1. Для примера, рассмотрим сайт образовательной платформы Coursera (https://www.coursera.org). Этот сайт (Рис. 1) предоставляет курсы и программы обучения от ведущих университетов и компаний, поэтому его интерфейс должен быть удобным и понятным как для студентов, так и для преподавателей.

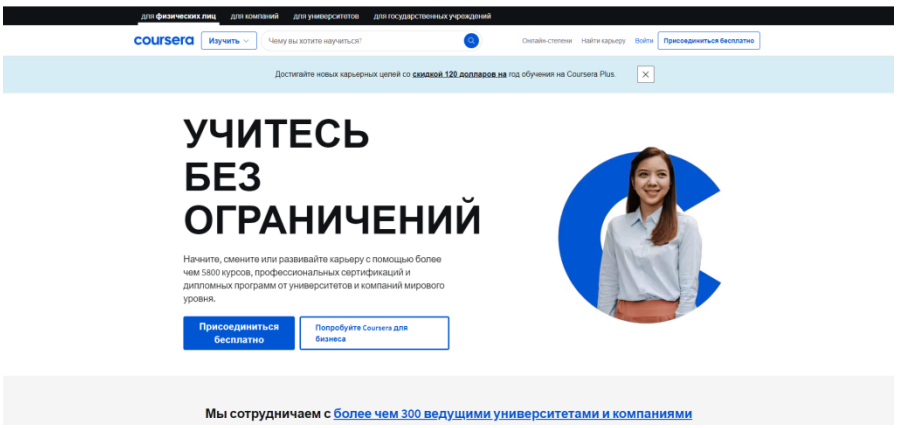

Рисунок 1 – Сайт Coursera

2.2. Анализ структуры и навигации веб-интерфейса

Coursera организован в несколько крупных разделов: курсы, программы обучения, преподаватели, доступ к учебным материалам. Навигация сайта выполнена через главное меню, где пользователь может выбрать нужную категорию (Рис. 2).

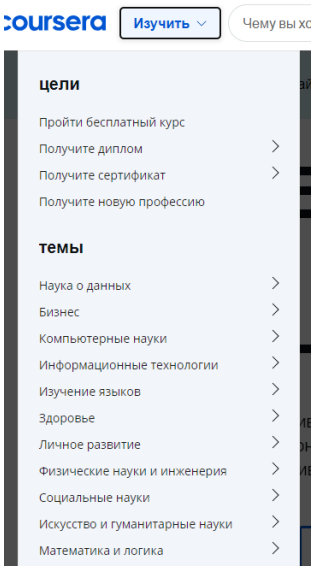

Рисунок 2 – Навигационное меню сайта

Принцип консистентности:

Меню сайта остается неизменным на всех страницах, что обеспечивает удобство при перемещении по ресурсу (Рис. 3). Все элементы навигации структурированы логично, позволяя пользователям легко найти нужную информацию (категории курсов, разделы по тематикам, фильтры по уровням сложности и др.).

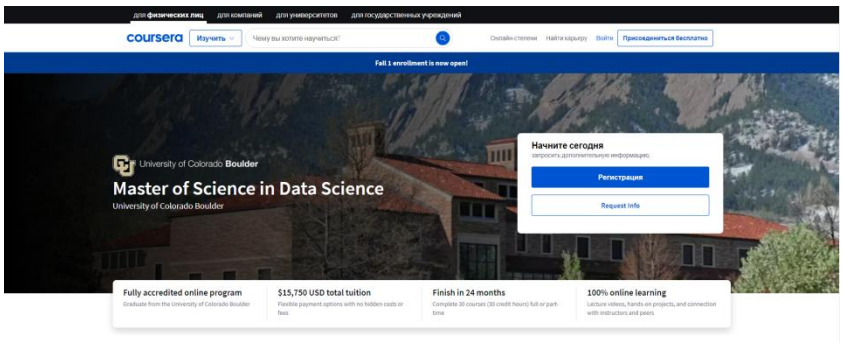

Рисунок 3 – Сохранение навигации при переходе по страницам

Оценка структуры:

Главная страница Coursera включает крупные баннеры с рекомендациями курсов и программ, а также блоки с текущими акциями и новыми предложениями (Рис. 4). С помощью фильтров пользователи могут быстро находить интересующие их темы. Структура страницы интуитивно понятна, что соответствует принципу логичности.

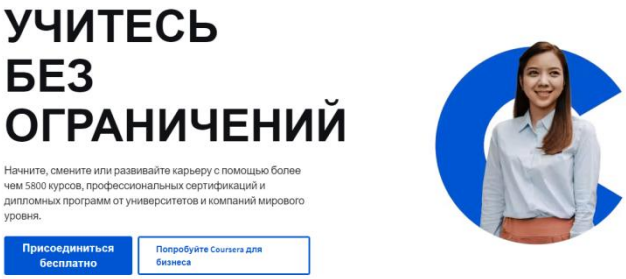

Рисунок 4 – Баннер главной страницы

Навигация:

Верхнее меню, боковые фильтры и ссылки на все важные страницы находятся в легкодоступных местах, что упрощает перемещение по разделам сайта. Однако, если пользователь ищет более узкие категории (например, редкие курсы), ему может потребоваться больше шагов для поиска. Улучшение поиска могло бы включить автоматические предложения при вводе текста в строку поиска.

2.3. Оценка доступности (Accessibility)

Для анализа доступности сайта, применим следующие критерии:

2.3.1. Контрастность цветов:

Используемый шрифт на сайте легко читается на фоне, однако в некоторых разделах, где фон светло-серый, текст менее контрастен. Это может затруднить чтение для людей с ослабленным зрением. Проведем проверку контраста с помощью инструмента Contrast Checker (Рис. 5).

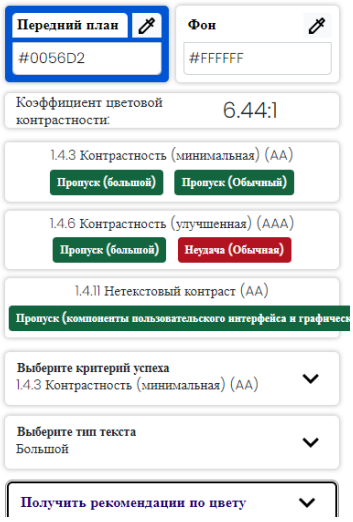

Рисунок 5 – Проверка контраста

2.3.2. Клавиатурная навигация:

Протестируем сайт на предмет доступности для пользователей, которые используют клавиатуру вместо мыши. Большинство элементов интерфейса (меню, кнопки, ссылки) поддерживают навигацию клавишами "Tab" и "Enter", что делает сайт доступным для пользователей с ограниченной подвижностью.

2.3.3. Атрибуты alt для изображений:

Проверим наличие альтернативных текстов для изображений на сайте. Все изображения курсов и баннеров не сопровождаются текстовыми описаниями, что не соответствует требованиям доступности для людей, использующих экранные читалки.

2.4.Анализ виртуальной составляющей и обратной связи

Эстетика интерфейса:

Сайт выполнен в светлых тонах, использует современную типографику и хорошо подобранные иконки. Визуальные элементы приятны глазу, а разделение блоков информации облегчает восприятие контента. Однако для некоторых разделов можно было бы улучшить визуальную иерархию, используя большее разнообразие в размерах и весе шрифтов.

Обратная связь (Рис. 6):

Coursera активно предоставляет обратную связь пользователям при взаимодействии с элементами интерфейса. Например, при наведении на кнопки, они меняют цвет, а при добавлении курса в корзину появляются уведомления. При попытке

отправить незаполненную форму сайт выводит сообщение об ошибке, что помогает пользователям исправить свои действия.

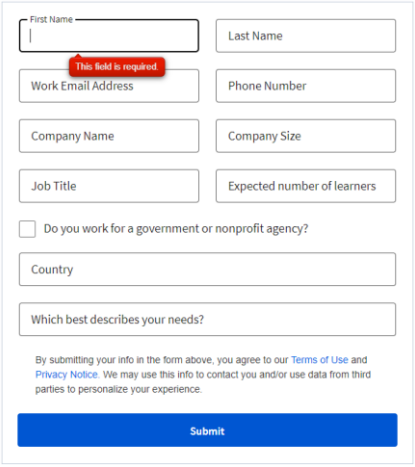

Рисунок 6 – Форма обратной связи с валидацией

Интерактивность (Рис. 7):

Сайт использует всплывающие подсказки для объяснения сложных функций. Это соответствует принципам интерактивности и обучения пользователя интерфейсу. Также Coursera применяет элементы микровзаимодействий — например, плавные анимации при смене страниц или наведении на интерактивные элементы.

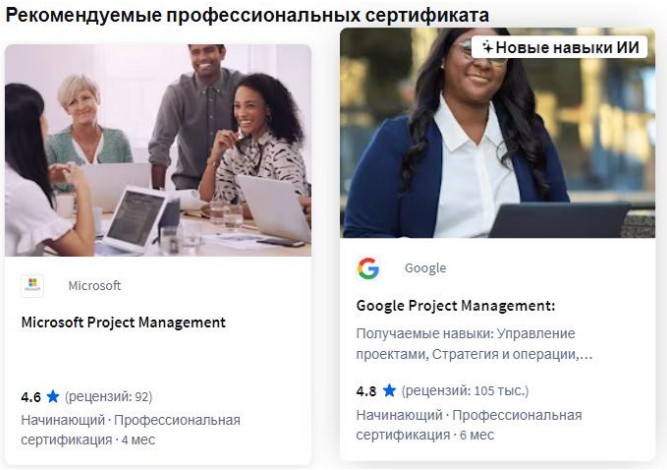

Рисунок 7 – Плавная анимация выделения карточки статьи

2.5. Рекомендации по улучшению веб-интерфейса

2.5.1. Повышение контрастности в некоторых разделах. Улучшить контраст между текстом и фоном на всех страницах, особенно для людей с ослабленным зрением.

2.5.2. Улучшение поиска по курсам. Добавить интеллектуальные подсказки при вводе в строку поиска, чтобы пользователь мог быстрее найти интересующий его курс.

2.5.3. Оптимизация клавиатурной навигации. Хотя большинство элементов интерфейса доступны через клавиатуру, можно улучшить работу с формами и выпадающими списками, чтобы их можно было быстрее заполнять без мыши.

3. Задания по лабораторной работе (Выполнять по вариантам. Номер по списку в журнале = номер варианта)

3.1. Анализ структуры и навигации веб-интерфейса

— Выберите любой существующий сайт или веб-приложение;

— Проанализируйте его структуру: определите, как организованы страницы, каким образом пользователь может переходить между разделами, насколько легко найти нужную информацию;

— Проведите оценку качества навигации и структуры сайта, применяя принципы консистентности, логичности и удобства использования.

3.2. Оценка доступности (Accessibility) веб-интерфейса

— Проверка контраста цветов (например, с помощью инструмента Contrast Checker);

— Тестирование сайта с помощью клавиатуры (все ли интерактивные элементы доступны);

— Оценка текстовых описаний для изображений (атрибут alt);

— Подготовьте отчет о возможных улучшениях для повышения доступности.

3.3. Анализ визуальной составляющей и обратной связи интерфейса

— Оцените выбранный веб-интерфейс с точки зрения визуальной привлекательности и эстетичности;

— Проанализируйте, как система предоставляет обратную связь пользователю (цветовые изменения, уведомления, сообщения об ошибках и успехах);

— Предложите возможные улучшения для повышения интерактивности и визуальной привлекательности.

Варианты заданий:

1. Сайт новостей;
2. Веб-сервис бронирования;
3. Корпоративный сайт;
4. Блог;

5. Социальная сеть;
6. Портфолио-дизайнера;
7. Сайт образовательного учреждения;
8. Сайт интернет-магазина (категория: одежда);
9. Сайт интернет-магазина (категория: электроника);
10. Платформа для онлайн-курсов;
11. Сайт ресторана;
12. Онлайн-калькулятор;
13. Сайт медицинской клиники;
14. Сайт спортивной организации;
15. Страница мероприятия;
16. Сайт конференции;
17. Сайт кинотеатра;
18. Веб-приложение для планирования задач;
19. Мобильное приложение;
20. Веб-приложение для заметок;
21. Онлайн-портал государственных услуг;
22. Страница продукта IT-компании;
23. Сайт музыканта или группы;
24. Сайт фестиваля;
25. Веб-приложение для фитнеса;
26. Платформа для заказа еды;
27. Онлайн-магазин книг;
28. Онлайн-банк;
29. Платформа для путешествий;
30. Сайт гостиницы или хостела.

Содержание отчёта:

1. Титульный лист;
2. Цель работы;
3. Задание;
4. Ход работы:
    a. Описание предметной области;

b. Описание задания (всех его пунктов), скриншот выполнения, ссылка на сайт, который описываете.

5. Вывод;

Критерии отчёта:

1. Оформление шрифтом Times New Roman (14 кегль);
2. Поля: верх 2 см, лево 3 см, право 1,5 см, нижнее 2 см;
3. Междустрочный интервал 1,5;
4. Отступ красной строки 1,25.

Контрольные вопросы:

1. Что такое UI-дизайн, и в чем его основные задачи?
2. Чем отличается UI-дизайн от UX-дизайна?
3. Назовите основные принципы UI-дизайна и кратко охарактеризуйте их.
4. Какие методы анализа веб-интерфейсов используются для оценки их удобства?
5. Что такое эвристическая оценка, и какие принципы используются в этом методе?
6. Как проверяется доступность веб-интерфейса для пользователей с ограниченными возможностями?
7. Какие способы улучшения интерактивности интерфейса вы знаете?
8. Почему важна консистентность интерфейса, и как она достигается?

Лабораторная работа №8

Анализ существующего веб-интерфейса на основе принципов UX/UI-дизайна

Цель работы: изучить принципы UI/UX-дизайна и применение их для анализа конкретного веб-интерфейса.

1. Теоретический материал

UX/UI-дизайн — это два взаимосвязанных направления, которые отвечают за создание удобных и привлекательных интерфейсов для взаимодействия пользователя с цифровыми продуктами. Эти понятия включают в себя как визуальные элементы, так и общие ощущения от взаимодействия с продуктом.

UX (User Experience) пользовательский опыт - отвечает за ощущения и удобство пользователя при взаимодействии с интерфейсом. UX-дизайнеры сосредоточены на том, чтобы пользователь мог быстро и интуитивно достичь своих целей.

Основные задачи UX:

- Навигация: интерфейс должен быть логичным и простым, чтобы пользователь мог легко ориентироваться;

- Эффективность: минимизация усилий и времени для выполнения основных действий (например, покупки в интернет-магазине);

- Доступность: обеспечение комфортного использования интерфейса для всех категорий пользователей, включая людей с ограниченными возможностями;

- Эмоциональная привлекательность: интерфейс должен вызывать положительные эмоции и способствовать формированию лояльности пользователя к продукту.

Принципы UX – дизайна:

- Простота использования: интерфейс должен быть интуитивным и не требовать долгого обучения.

- Минимизация нагрузки: интерфейс не должен перегружать пользователя сложными действиями или лишней информацией.

- Обратная связь: система должна давать пользователю четкие сигналы о выполнении или невыполнении его действий (например, всплывающие уведомления).

UI (User Interface) пользовательский интерфейс — это работа с внешним видом интерфейса, с которым непосредственно взаимодействует пользователь. Основная цель

UI-дизайна — создание визуально приятного, понятного и логически структурированного интерфейса.

Основные элементы UI:

- Цветовая палитра: выбор цветов, их сочетание и гармония.

- Шрифты: выбор читаемых и подходящих по стилю шрифтов.

- Кнопки, иконки, меню: взаимодействие через понятные и интерактивные элементы управления.

- Расположение элементов: структура страниц и блоков для обеспечения удобной навигации.

Принципы UI-дизайна:

- Единообразие: элементы должны повторяться в интерфейсе для формирования у пользователя предсказуемого опыта.

- Визуальная иерархия: важные элементы должны быть выделены визуально (больше размер, яркие цвета).

- Читаемость и контрастность: интерфейс должен быть удобен для восприятия текста и визуальной информации.

Хотя UI и UX выполняют разные функции, они тесно связаны. Хороший UI-дизайн может сделать интерфейс привлекательным и интуитивно понятным, но без качественного UX пользователь может быстро потеряться или разочароваться. В свою очередь, грамотный UX обеспечит удобство использования, но, если визуальная часть (UI) будет некачественной, пользователь может отказаться от взаимодействия с продуктом.

UI/UX-дизайн направлен на создание удобных, привлекательных и функциональных интерфейсов, которые помогают пользователям эффективно и приятно взаимодействовать с цифровыми продуктами. Хорошо спроектированный интерфейс сочетает в себе визуальную эстетику (UI) и удобство использования (UX), обеспечивая пользователям лучший опыт.

2. Практическая часть

Выбор веб-интерфейса для анализа

2.1 Описание этапов анализа веб-интерфейса

a. Оценка визуальной структуры (UI)

- Провести обзор цветовой палитры, шрифтов, графических элементов.

• Проанализировать расположение элементов, кнопок, меню и их логическую взаимосвязь.

б. Оценка удобства использования (UX)

• Проверить навигацию: насколько легко пользователю находить нужную информацию.

• Оценить структуру страниц, количество шагов для выполнения основных действий.

в. Проверка адаптивности

• Протестировать сайт на различных устройствах (desktop, mobile, tablet).

• Проанализировать, насколько корректно и удобно отображаются элементы на разных разрешениях экрана.

2.2 Заполнение таблицы анализа

Пример структуры таблицы:

Критерий	Описание проблемы	Примеры (скриншот)	Рекомендации
Визуальная иерархия	Важные элементы (кнопки "Купить" и "Зарегистрироваться") не выделяются на фоне других элементов.		Использовать более яркие цвета и увеличить размер кнопок для привлечения внимания.
Читаемость текста	Текст на некоторых страницах слишком мелкий и нечитабельный на мобильных устройствах.		Увеличить размер шрифта до 16px и выше, улучшить контраст между текстом и фоном.
Навигация	Меню содержит слишком много пунктов, что усложняет поиск		Сгруппировать пункты меню в категории, добавить выпадающие списки для упрощения.

	нужной информации.		
Адаптивность	В мобильной версии сайта изображения и кнопки выходят за пределы экрана.		Применить адаптивные стили CSS для корректного отображения контента на малых экранах
Обратная связь	После отправки формы отсутствует подтверждающее сообщение, что вызывает у пользователя неуверенность в успешности действия.		Добавить всплывающее уведомление о том, что форма успешно отправлена.
Скорость загрузки	Главная страница сайта загружается слишком долго из-за большого количества тяжелых изображений.		Оптимизировать изображения, использовать современные форматы (WebP), подключить кеширование.

3. Варианты для выполнения работы

1. **Анализ интерфейса интернет-магазина** (например, Wildberries, Ozon). Оцените удобство навигации, адаптивность и визуальную иерархию.
2. **Анализ главной страницы новостного портала** (например, BBC, РИА Новости). Оцените визуальную структуру и читаемость текста.
3. **Анализ интерфейса формы обратной связи** на сайте (например, форма обратной связи на сайте компании). Проверьте наличие подтверждающих уведомлений после отправки формы.
4. **Анализ мобильной версии интернет-банка** (например, Сбербанк Онлайн или Тинькофф). Оцените адаптивность и скорость загрузки.
5. **Анализ интерфейса портала онлайн-курсов** (например, Coursera, Skillbox). Оцените доступность для пользователей с ограниченными возможностями.

6. **Анализ страницы оформления заказа** (например, Amazon, AliExpress). Оцените простоту и скорость выполнения процесса оформления.
7. **Анализ интерфейса корпоративного сайта** (например, сайт технологической компании). Оцените логичность навигации и визуальный стиль.
8. **Анализ интерфейса блога или новостного сайта** (например, Medium). Оцените взаимодействие с текстом, удобство чтения и поиска.
9. **Анализ интерфейса видеохостинга** (например, YouTube). Оцените расположение элементов и визуальную иерархию.
10. **Анализ интерфейса сайта ресторана** (например, доставка еды). Оцените доступность меню и процесс заказа.
11. **Анализ сайта путешествий** (например, Booking.com). Оцените поиск информации и простоту фильтрации.
12. **Анализ интерфейса страницы входа в аккаунт** (например, Google, Facebook). Оцените визуальную простоту и обратную связь при ошибках.
13. **Анализ интерфейса мобильного приложения для спорта** (например, Strava). Оцените удобство использования на мобильных устройствах.
14. **Анализ интерфейса социальной сети** (например, Instagram, VK). Оцените взаимодействие пользователя с контентом и удобство навигации.
15. **Анализ интерфейса образовательного портала** (например, Университетский сайт). Оцените структуру разделов и доступность материалов.
16. **Анализ интерфейса сайта благотворительности** (например, донорство крови). Оцените удобство для пользователя и эмоциональную привлекательность.
17. **Анализ интерфейса туристического сайта** (например, сайт авиакомпании). Оцените навигацию и взаимодействие с системой бронирования.
18. **Анализ страницы отзывов** на сайте интернет-магазина. Оцените легкость чтения отзывов и функциональность поиска.
19. **Анализ интерфейса сайта электронной библиотеки** (например, ЛитРес). Оцените доступность для людей с ограничениями по зрению.
20. **Анализ интерфейса программы видеоконференций** (например, Zoom). Оцените простоту управления звонками и настройками.

21. **Анализ интерфейса онлайн-платформы для фрилансеров** (например, Upwork). Оцените удобство использования инструментов для работы.
22. **Анализ интерфейса сайта электронной почты** (например, Gmail). Оцените удобство работы с письмами и организация папок.
23. **Анализ интерфейса приложения для заказа такси** (например, Uber). Оцените удобство выбора маршрута и заказа поездки.
24. **Анализ интерфейса сайта бронирования отелей** (например, Airbnb). Оцените поиск и фильтрацию жилья, процесс оформления.
25. **Анализ интерфейса сайта службы доставки** (например, DHL). Оцените оформление заказа и отслеживание доставки.
26. **Анализ интерфейса сайта кинотеатра** (например, покупка билетов). Оцените удобство выбора фильма и бронирования билета.
27. **Анализ интерфейса онлайн-редактора** (например, Google Docs). Оцените организацию меню и доступность функций.
28. **Анализ интерфейса музыкального стриминга** (например, Spotify). Оцените удобство поиска музыки и функциональность рекомендаций.
29. **Анализ интерфейса онлайн-сервиса доставки еды** (например, Яндекс.Еда). Оцените навигацию по меню и процесс заказа.
30. **Анализ интерфейса сайта для бронирования мероприятий** (например, Eventbrite). Оцените простоту поиска и оформления участия.

Контрольные вопросы:

1. Что такое UI-дизайн и какие его основные задачи?
2. Какова основная цель UX-дизайна и как она влияет на взаимодействие пользователя с интерфейсом?
3. Какие принципы UI-дизайна помогают создать удобный и привлекательный интерфейс?
4. Какое значение имеет визуальная иерархия в веб-дизайне? Приведите пример.
5. Что такое когнитивная нагрузка в контексте UX-дизайна, и как её можно уменьшить?

6. Как адаптивность веб-интерфейса влияет на пользовательский опыт?
7. Что такое доступность в интерфейсах и почему она важна для UX-дизайна?
8. Какие инструменты можно использовать для анализа доступности и производительности веб-сайта?
9. Как проверить обратную связь в интерфейсе? Приведите примеры хорошей и плохой реализации обратной связи.
10. Какие факторы влияют на скорость загрузки сайта, и как это отражается на пользовательском опыте?

Лабораторная работа №9

Создание дизайн-макета главной страницы веб-сайта с использованием Figma

Цель работы: научиться создавать дизайн-макеты веб-страниц с использованием инструмента Figma, а также освоить основные принципы и приемы UX/UI-дизайна.

1. Теоретический материал

Основы веб-дизайна:

Веб-дизайн — это процесс создания визуальной составляющей и структуры веб-страниц. Хорошо продуманный дизайн помогает пользователям легко ориентироваться на сайте и выполнять свои задачи. Основные задачи веб-дизайна включают в себя:

1. Привлечение внимания — важно сделать сайт визуально привлекательным, чтобы вызвать интерес у пользователя.
2. Удобство использования (usability) — пользователю должно быть легко найти необходимую информацию или выполнить целевое действие (купить, зарегистрироваться и т.д.).
3. Функциональность — сайт должен корректно работать на различных устройствах (адаптивный дизайн), загружаться быстро и без ошибок.

Различие между UX и UI дизайном:

- **UX (User Experience)** — это проектирование опыта пользователя, то есть то, как удобно и логично ему пользоваться интерфейсом. UX-дизайн фокусируется на решении проблем пользователя, на том, как он взаимодействует с продуктом, и на создании позитивных эмоций от этого взаимодействия.
- **UI (User Interface)** — это визуальная часть интерфейса, то, как выглядит сайт. UI-дизайнер отвечает за оформление всех элементов, таких как кнопки, иконки, цветовые схемы и шрифты. UI направлен на создание привлекательного и функционального интерфейса.

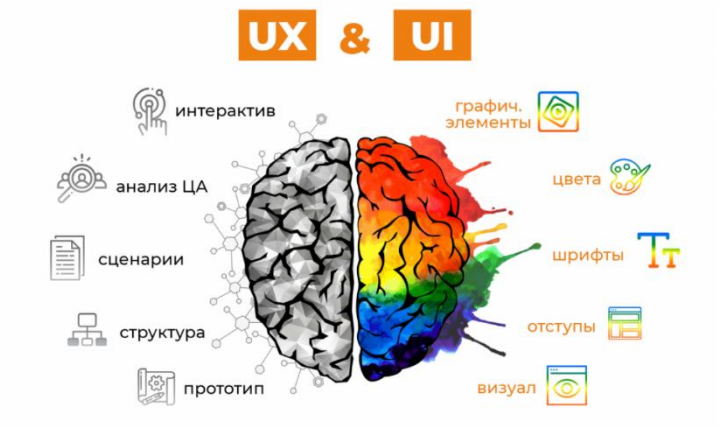

Рисунок 1 – разница между UX и UI

Важность структуры и визуальной иерархии на главной странице:

Структура веб-страницы играет ключевую роль в создании удобного пользовательского интерфейса. Важно расположить элементы так, чтобы они соответствовали ожиданиям пользователя. Эффективная визуальная иерархия помогает направить внимание пользователя на важные элементы:

- Заголовки — должны быть легко заметными и передавать основное содержание блока.
- Кнопки призыва к действию (Call-to-Action) — должны быть заметными и доступными для взаимодействия.
- Отступы и пробелы — помогают избежать перенасыщенности информацией и делают страницу легче для восприятия.

Примером иерархии может быть классическая схема "Z" или "F", где пользователь сначала просматривает верхние элементы страницы (например, логотип и меню), затем перемещается к основному содержанию и заканчивает футером.

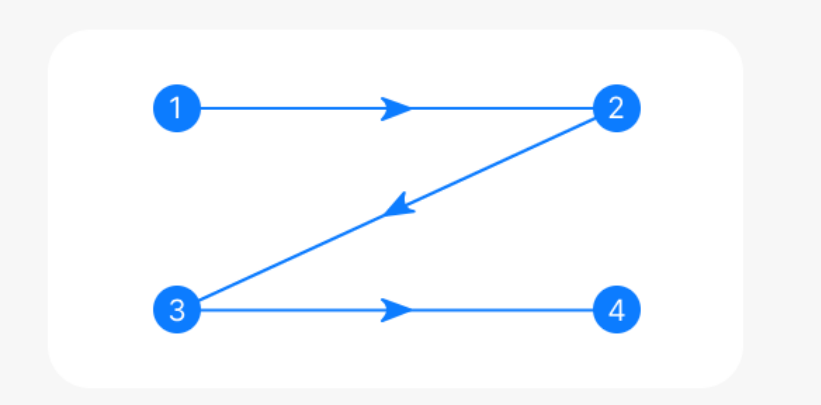

Рисунок 2 – схема иерархии Z

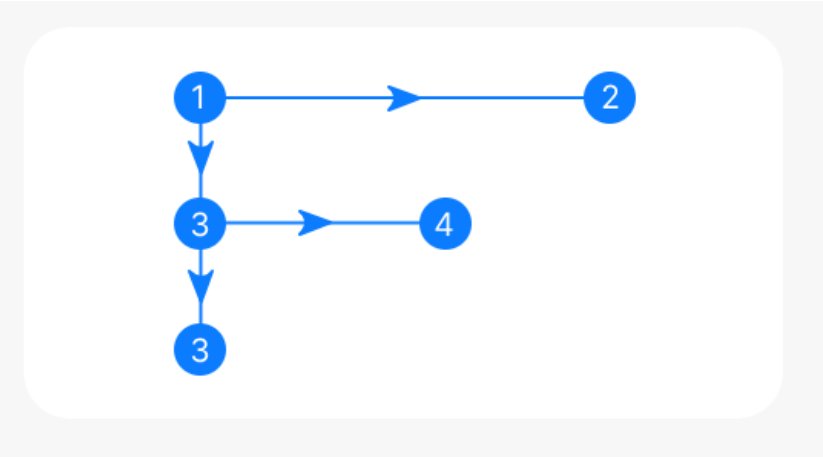

Рисунок 3 – схема иерархии F

Введение в figma

Figma — это популярный онлайн-инструмент для дизайна и прототипирования интерфейсов. Она позволяет дизайнерам, разработчикам и менеджерам проектов совместно работать над интерфейсами в реальном времени. Среди главных преимуществ Figma:

1 Облачное хранение данных — доступ к проектам с любого устройства.

2 Совместная работа — несколько человек могут одновременно редактировать проект.

3 Библиотеки компонентов — возможность создавать и использовать общие стили и элементы.

4 Прототипирование — возможность создавать интерактивные прототипы для демонстрации и тестирования.

Принципы работы в Figma:

Прототипирование — позволяет создавать макеты с интерактивными элементами, показывать взаимодействие между страницами.

Дизайн интерфейсов — основные элементы дизайна можно создавать с помощью встроенных инструментов Figma: фигуры, текст, иконки, изображения.

Компоненты и стили — возможность создания повторно используемых элементов (кнопки, карточки, иконки) для экономии времени и обеспечения единства стиля.

Figma активно используется как начинающими дизайнерами, так и профессионалами, благодаря своей гибкости и широким возможностям для создания качественного пользовательского интерфейса.

2. Подготовка к работе

2.1 Регистрация и настройка рабочей области

Перед началом работы необходимо зарегистрироваться на платформе Figma и настроить рабочее пространство.

- Перейдите на официальный сайт Figma (https://www.figma.com/login?is_not_gen_0=true&resource_type=team) и создайте учетную запись. Можно использовать учетную запись Google для ускоренной регистрации.

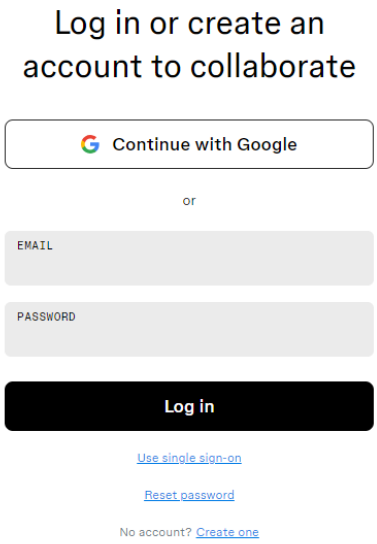

Рисунок 4 – окно регистрации

- После регистрации вы попадете в основное рабочее пространство, которое называется Dashboard. Здесь будут храниться все ваши проекты.

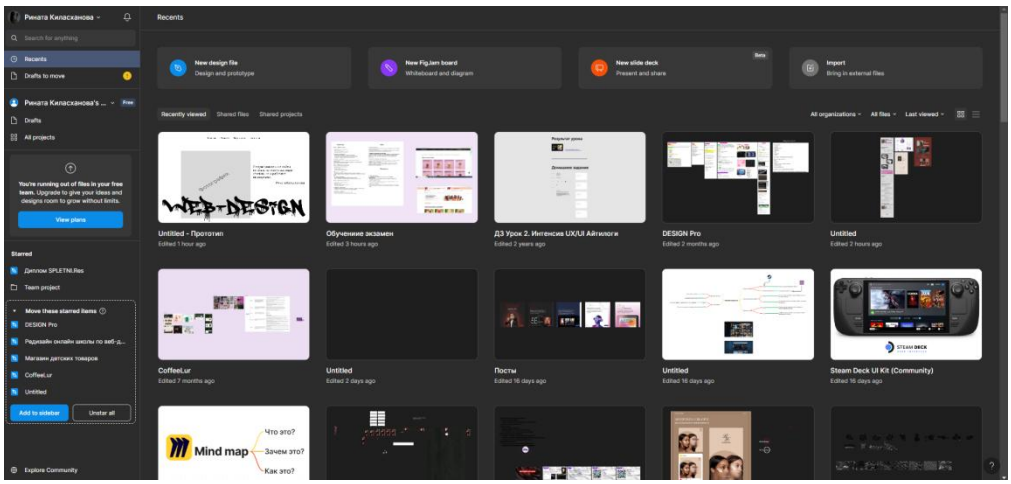

Рисунок 5 – рабочее пространство

2.1.1 Как создать проект

- Нажмите кнопку New File или выберите пункт + New Design File в главном меню. Откроется пустое окно редактора Figma, где будет создаваться макет.
- Для удобства работы проект можно сразу переименовать, щелкнув на название файла в верхней части экрана.
- В Figma также можно создавать папки, чтобы структурировать ваши проекты. Это полезно, если вы работаете над несколькими проектами одновременно.

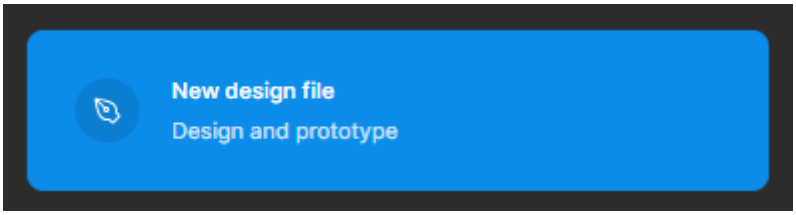

Рисунок 6 – создание проекта

2.1.2 Организация рабочего пространства: фреймы, страницы, слои

В Figma для построения макета используются три ключевых элемента:

Фреймы (Frames) — это контейнеры, которые используются для размещения различных элементов интерфейса. По сути, фрейм — это область, внутри которой создается дизайн. Фреймы могут представлять собой экраны веб-сайта, мобильные экраны или отдельные блоки контента.

- Чтобы создать фрейм, выберите инструмент "Frame" (или нажмите клавишу "F") и задайте размер.

• Figma предоставляет готовые размеры для популярных устройств (например, для десктопов, планшетов и смартфонов).

Страницы (Pages) — можно создавать несколько страниц в одном проекте. Это удобно для организации макетов разных частей сайта или экранов приложения. Страницы переключаются на боковой панели слева.

Слои (Layers) — каждый элемент, добавленный в фрейм, отображается как отдельный слой. Слои помогают управлять содержимым страницы и организовывать элементы по группам. Все слои находятся в левой части интерфейса, и их можно переименовывать, группировать или блокировать для упрощения работы.

2.2 Основные инструменты figma

Для создания макета необходимо освоить несколько базовых инструментов:

1. Инструмент "Прямоугольник" (Rectangle Tool)

• Используется для создания прямоугольных и квадратных элементов. Нажмите клавишу R или выберите значок прямоугольника на панели инструментов. Это основной инструмент для создания кнопок, блоков контента и других элементов.

2. Инструмент "Эллипс" (Ellipse Tool)

• Для создания кругов и эллипсов используйте клавишу O. Фигуры можно преобразовывать и использовать в качестве фона или иконок.

3. Инструмент "Текст" (Text Tool)

• Нажмите T или выберите значок текста на панели инструментов, чтобы добавить текстовые элементы в дизайн. В Figma можно настроить шрифты, размеры, интервалы между строками и другие параметры типографики.

4. Инструмент "Линия" (Line Tool)

• Нажмите L для создания линий. Линии используются для разделения блоков или создания декоративных элементов.

5. Панель свойств (Properties Panel)

• На правой панели интерфейса находятся настройки для выбранных элементов. Здесь можно изменить цвет, добавить тени, настроить обводку, скругление углов и другие параметры. Для текста также доступны инструменты выравнивания и настройки межбуквенных интервалов.

Рисунок 7 – панель инструментов

2.3 Использование сетки и направляющих

Сетка и направляющие помогают организовать элементы на странице и добиться симметрии. Figma предлагает несколько типов сеток, которые можно применить к фреймам:

1. Пиксельная сетка (Pixel Grid)

• Включается кнопкой Shift + #. Она используется для точного выравнивания и создания элементов с учетом пиксельной точности.

Рисунок 8 – пиксельная сетка

2. Сетка (Grid Layout)

• Сетка используется для построения макетов с четкой структурой. Добавляется через панель Layout Grid на правой панели свойств фрейма. Вы можете настроить количество колонок, отступы между ними и их ширину.

• Колонки помогают создать адаптивную структуру страницы. Обычно для веб-дизайна используют 12-колоночную сетку, так как она хорошо подходит для различных разрешений экрана.

Рисунок 9 – сетка колонками

3. Направляющие (Guides)

• Это вспомогательные линии, которые можно добавлять вручную, чтобы выровнять элементы на странице. Чтобы создать направляющую, просто перетащите линейку с края фрейма.

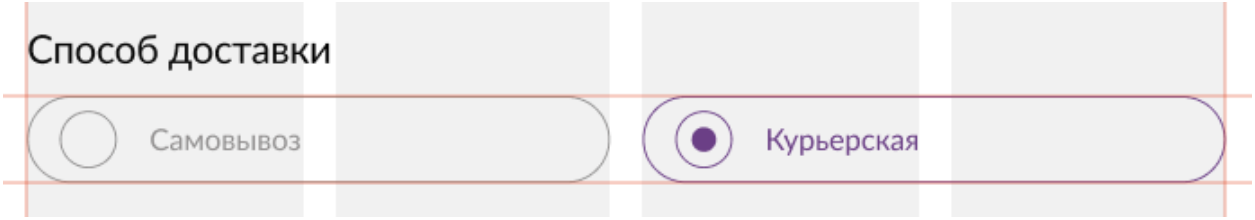

Рисунок 10 – направляющие

3. Разработка прототипа главной страницы веб-сайта

Прежде чем приступить к созданию прототипа главной страницы веб-сайта, необходимо чётко определить её цель и задачи. Главная страница — это лицо сайта, и она должна решать конкретные бизнес-задачи и быть понятной для пользователей. Ключевые аспекты:

1. Цель страницы

- Определите, что должна сделать главная страница для бизнеса. Это может быть:
- Привлечение внимания к продукту или услуге.
- Мотивация пользователей зарегистрироваться или оставить контактные данные.
- Продвижение определённого контента (новости, статьи).

2. Аудитория сайта

- Кто будет пользоваться сайтом? От понимания целевой аудитории зависят:
- Стиль дизайна (формальный или более креативный).
- Типы контента (подробные описания продуктов, отзывы клиентов, новости и т.д.).

3. Задачи страницы

- Какие действия должен совершить пользователь? Это может быть:
- Переход в каталог товаров.
- Клик по кнопке "Связаться с нами".
- Подписка на рассылку.

Пример задач главной страницы:

- Навигация — пользователь должен легко найти нужную информацию.
- Презентация основного предложения — описание продукта или услуги, которая интересует посетителя.
- Призывы к действию (CTA) — чётко заметные кнопки, мотивирующие на дальнейшие действия (регистрация, покупка).

Проектирование сетки страницы:

Сетка (grid) помогает организовать элементы на странице так, чтобы они были логично расположены и воспринимались последовательно. С помощью сетки можно достичь симметрии, согласованности и удобства восприятия.

1. Выбор сетки для главной страницы

- 8 или 12-колоночная сетка — один из самых популярных вариантов для веб-дизайна. Она позволяет гибко настраивать расположение блоков, адаптируя их под разные устройства (десктопы, планшеты, мобильные телефоны).

• Адаптивная сетка — важна для того, чтобы дизайн был удобным как на больших экранах, так и на маленьких. Например, блоки могут занимать 4 колонки на десктопе, а на мобильных устройствах – уже 12.

2. Отступы и интервалы

• Определите фиксированные отступы между элементами (например, 20px). Это обеспечит единообразие макета.

• Придерживайтесь модульности: если вы выбрали определённый размер отступов, повторяйте его для всех элементов макета.

3. Фреймы и контейнеры

• Используйте фреймы в Figma для создания контейнеров под различные блоки страницы (хедер, блок с информацией о продукте, карточки, футер). Это помогает структурировать макет и сделать его более читабельным.

Рисунок 11 – пример сеток

Создание wireframe (набросок структуры без стилей/прототип)

Создание "wireframe" (набросок структуры без стилей)

Wireframe — это черновой макет страницы, на котором отображаются основные блоки и элементы без проработки стилей. Это важный этап проектирования, который позволяет сосредоточиться на структуре и функционале, не отвлекаясь на визуальные детали.

1. Размещение основных блоков страницы Главная страница обычно состоит из следующих ключевых элементов:

- Хедер (Header) — включает логотип, меню навигации, кнопку или ссылку на важные разделы сайта (вход, регистрация, контакты).
- Основной блок с ключевым предложением (Hero Section) — самый заметный блок, который часто содержит крупный заголовок, описание продукта или услуги, и кнопку призыва к действию.
- Блок с услугами или продуктами — здесь располагаются карточки товаров или услуги, представленные на сайте. Для каждой карточки можно предусмотреть место под изображение, название и краткое описание.
- Информационные блоки — могут включать разделы с отзывами клиентов, кейсами, партнёрами, примерами проектов.
- Футер (Footer) — включает контактную информацию, ссылки на социальные сети, дублирует навигацию и юридические сведения.

2. Определение ключевых элементов страницы Для каждого из блоков необходимо продумать, какие элементы в них будут основными:

- Логотип — часто размещается в левом верхнем углу страницы.
- Меню навигации — важные разделы сайта (о компании, услуги, контакты) должны быть легко доступны.
- Кнопка призыва к действию (CTA) — в блоке "Hero" важно расположить яркую и заметную кнопку (например, "Купить", "Оставить заявку", "Подписаться"). Она должна быть основной точкой фокуса пользователя.
- Текстовые блоки — для заголовков и подзаголовков используется крупный шрифт, чтобы они выделялись на фоне остального контента.

3. Проектирование взаимодействий (по необходимости)

- Если вы планируете добавить на страницу интерактивные элементы (например, выпадающее меню, слайдер с изображениями или всплывающие окна), на этапе wireframe важно предусмотреть их место и размеры.

Wireframe — это "скелет" будущей страницы. Он не требует визуальной проработки (цвета, шрифты, изображения), но должен чётко показывать расположение всех ключевых элементов и их размеры.

Советы по созданию эффективного wireframe

- Минимум деталей — избегайте проработки декоративных элементов, сосредоточьтесь на логике расположения блоков.

• Фокус на UX — сделайте макет интуитивно понятным и удобным для пользователя. Основные действия пользователя (например, нажатие кнопок) должны быть легко достижимы.

• Проверяйте адаптивность — при создании макета уже на этапе wireframe продумайте, как элементы будут выглядеть на разных устройствах (десктоп, планшет, смартфон).

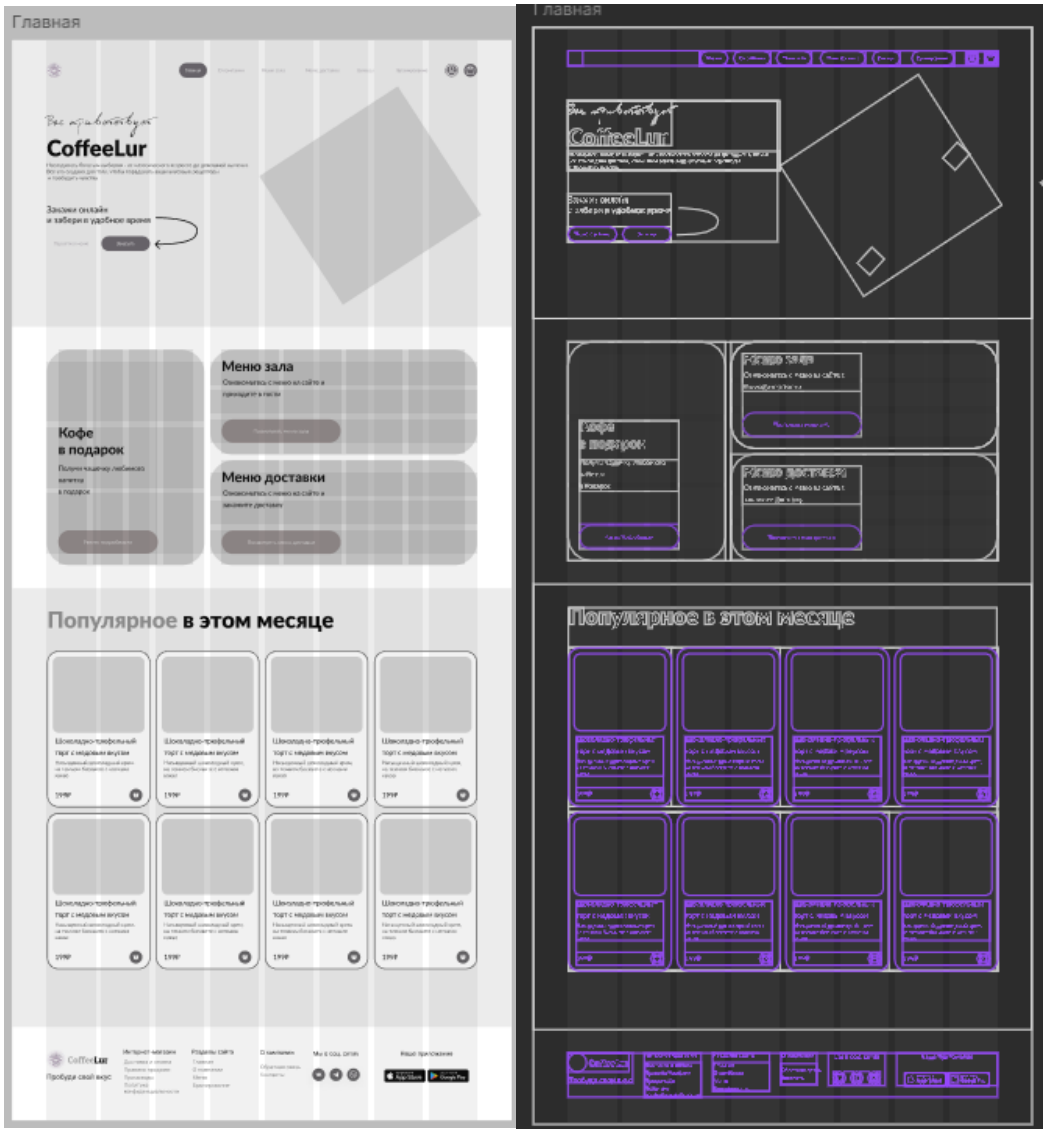

Рисунок 12 – пример wirefram

3. Создание дизайн-макета

Применение визуальных стилей

Добавление шрифтов, цвета, отступов и теней

Шрифты: важно выбрать шрифты, которые соответствуют стилю вашего сайта и целевой аудитории. Используйте не более двух-трех шрифтов, чтобы сохранить единообразие. В Figma можно подключать шрифты из Google Fonts или загружать собственные.

Цвета: Разработка цветовой палитры начинается с выбора основного цвета, который будет использоваться для заголовков, кнопок и акцентов. Дополните его вторичными цветами для фонов, текстов и других элементов. Рекомендуется использовать цветовые схемы, чтобы обеспечить гармоничное сочетание.

Отступы: определите фиксированные отступы между элементами (например, 20px или 30px). Это создаст структурированность и позволит элементам «дышать». Важно соблюдать единообразие в использовании отступов на всех страницах.

Тени: Используйте тени для создания глубины и выделения элементов. Они могут помочь привлечь внимание к важным блокам, таким как кнопки или карточки. Однако не переборщите с тенями — слишком много слоев может создать ощущение загроможденности.

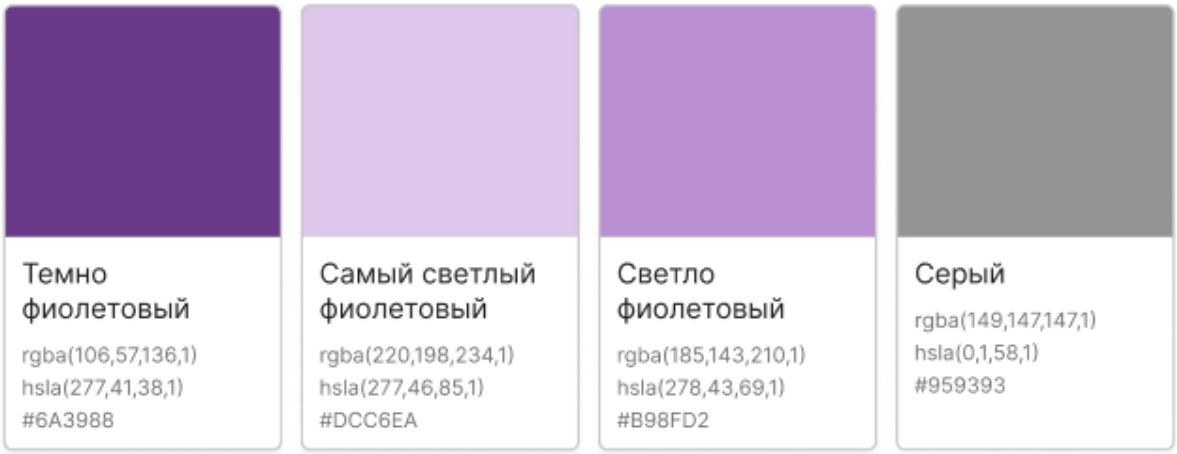

Рисунок 13 – цветовая палитра

Настройка типографики

Принципы выбора шрифтов и их сочетание

Используйте шрифты с хорошей читабельностью для основного текста (например, sans-serif для веба).

Для заголовков можно выбрать более выразительные шрифты, которые будут выделять важную информацию.

Убедитесь, что шрифты хорошо сочетаются между собой (например, один шрифт для заголовков и другой для основного текста).

Настройка иерархии заголовков текста

Определите размеры шрифтов для заголовков (H1, H2, H3) и основного текста. Заголовки должны быть заметными, а основной текст — удобным для чтения.

Настройте межстрочные интервалы (line-height) для улучшения читабельности текста.

Примените стиль для выделения ключевых слов (например, полужирный шрифт, курсив).

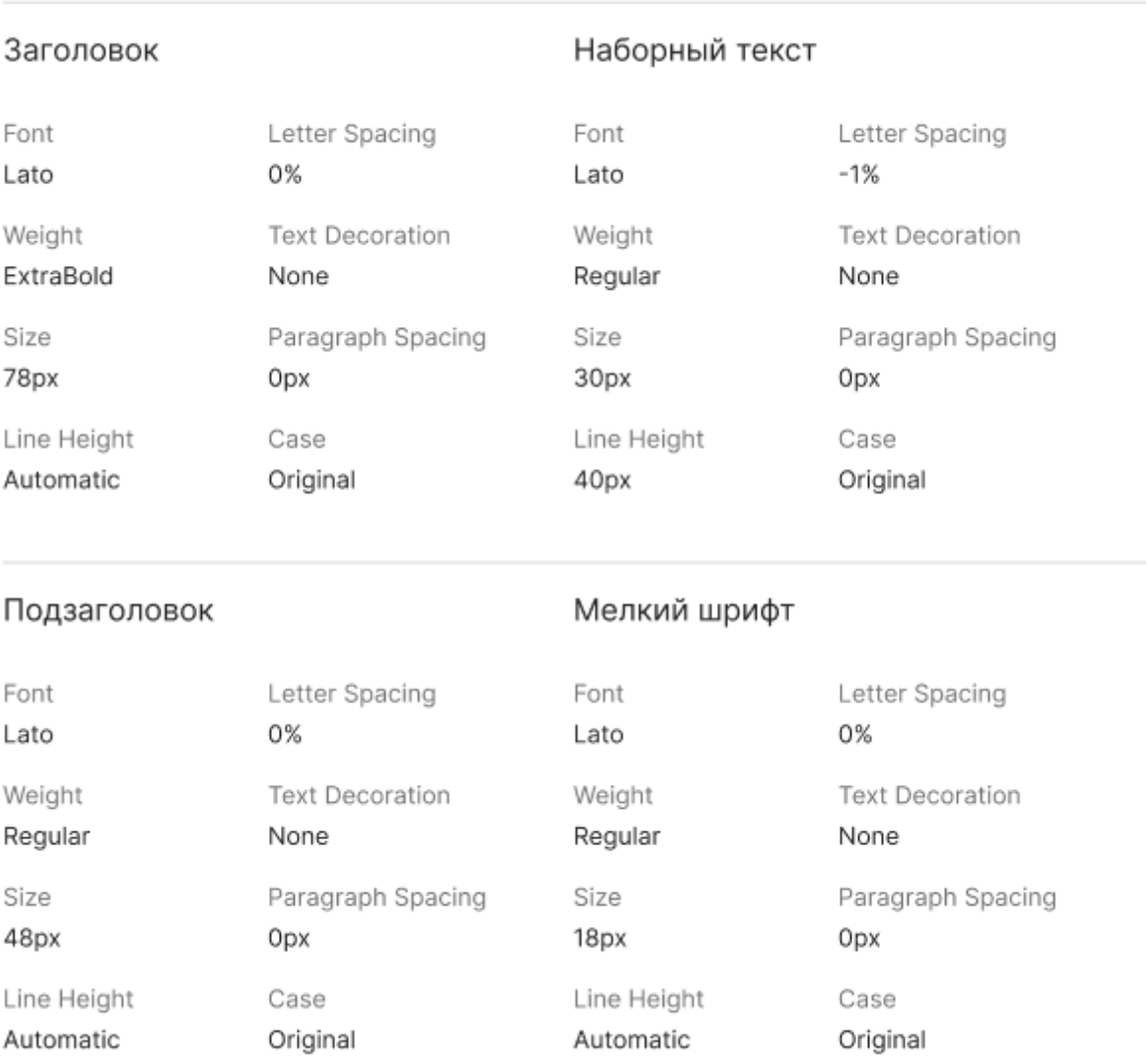

Рисунок 14 – иерархия заголовков

Вставка изображений

Как работать с масками и кадрированием

В Figma можно использовать маски для обрезки изображений в определённую форму. Это особенно полезно для создания уникальных карточек или аватаров.

Для создания маски выделите фигуру (например, круг или прямоугольник) и изображение, затем используйте комбинацию Ctrl + Alt + M (или Cmd + Option + M на Mac).

Рисунок 14 – работа с масками

Правила размещения изображений для улучшения UX

Изображения должны поддерживать контент и усиливать сообщение. Например, для карточек товаров используйте высококачественные изображения продукта.

Не перегружайте страницу изображениями. Дайте глазам пользователя отдохнуть, оставляя достаточные пространства между изображениями и текстом.

Используйте изображения, которые соответствуют стилю вашего сайта (например, современные, минималистичные или более классические).

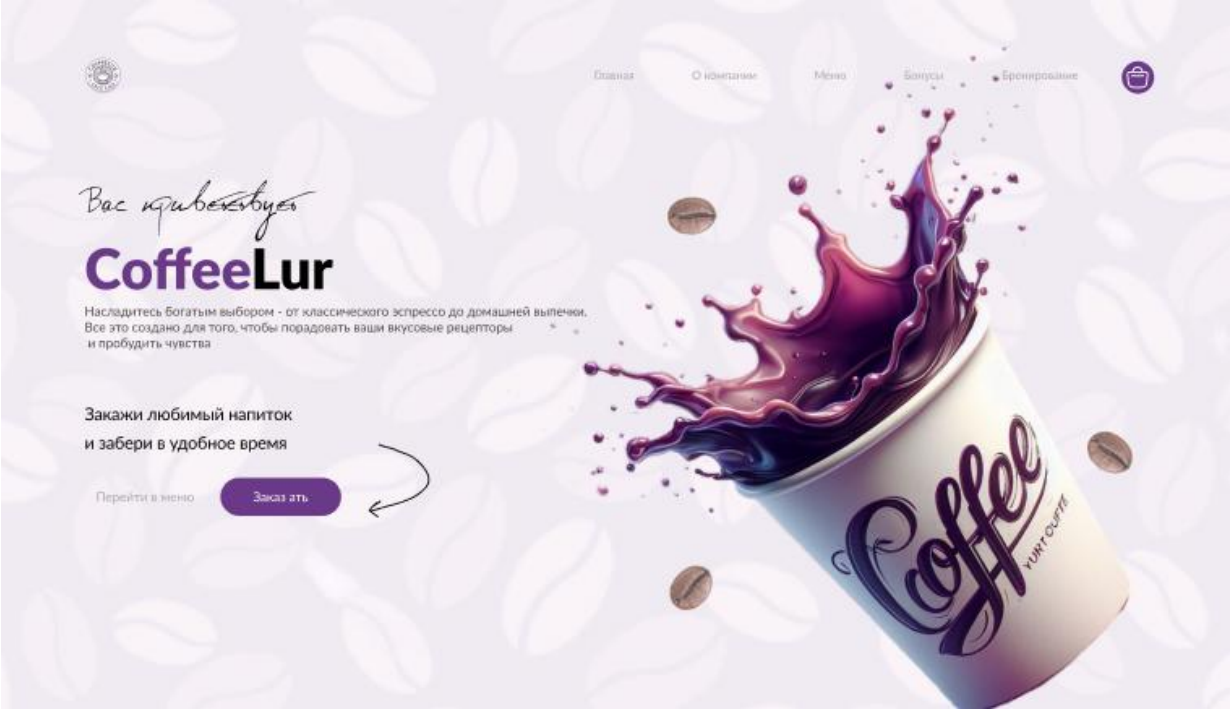

Рисунок 15 – правильное размещение изображения

Создание компонентов и стилей

Использование компонентов

В Figma можно создавать компоненты (reusable components), которые представляют собой повторно используемые элементы (например, кнопки, карточки, формы). Это позволяет поддерживать единообразие дизайна и облегчает обновление элементов.

Создание компонента осуществляется с помощью комбинации Ctrl + Alt + G (или Cmd + Option + G на Mac). При редактировании основного компонента изменения автоматически применяются ко всем его экземплярам.

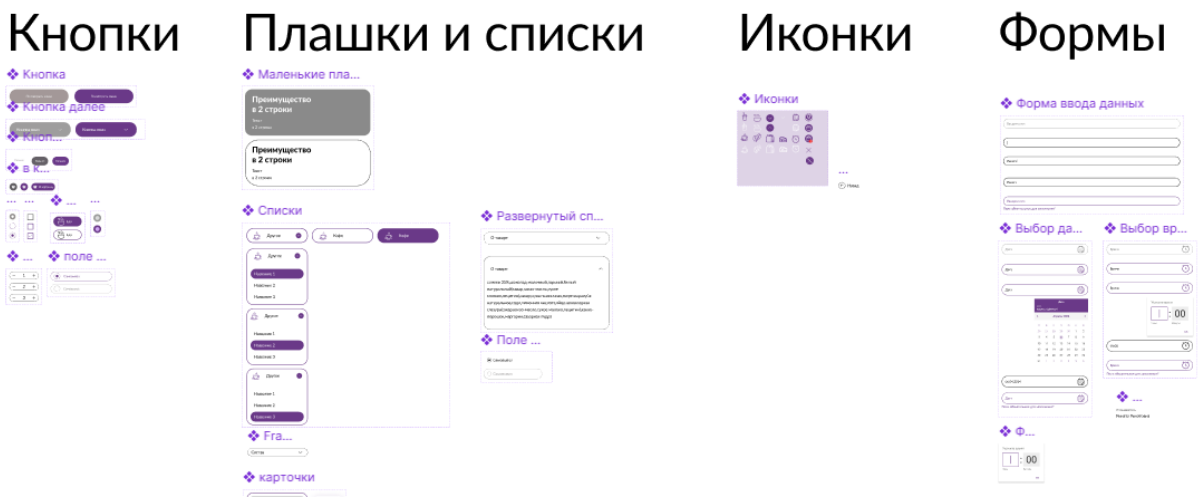

Рисунок 16 - компоненты

Настройка стилей

Стили в Figma позволяют сохранять настройки (например, цвет, шрифт, тени) для последующего применения. Это значительно упрощает процесс создания макета и обеспечивает консистентность.

Для создания стиля выберите элемент, настройте его свойства и сохраните как стиль через панель стилей в правой части интерфейса.

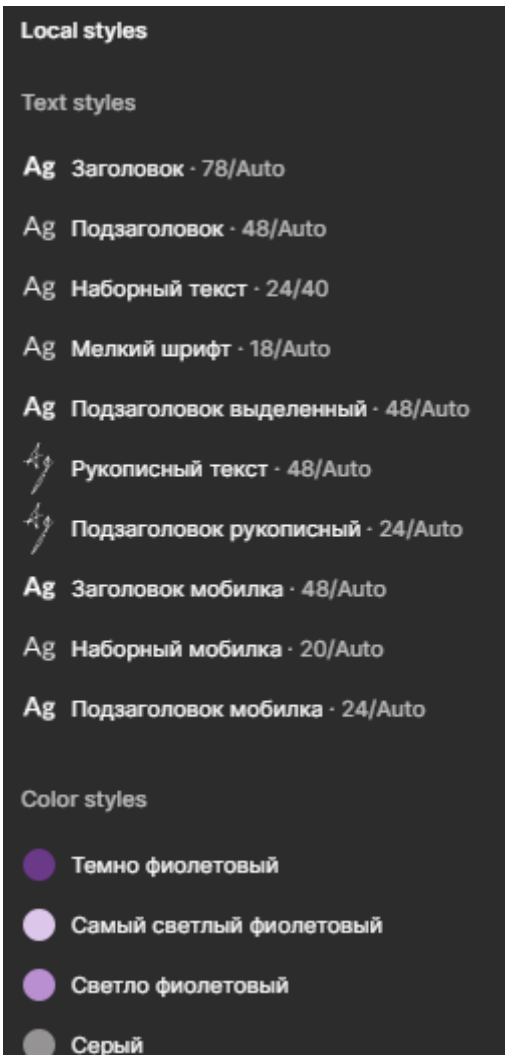

Рисунок 17 – стили

4. практическое задание

Создайте дизайн-макет главной страницы веб-сайта. Ваш макет должен включать все ключевые элементы, о которых шла речь на занятиях.

1. Интернет-магазин одежды: Создайте макет главной страницы для онлайн-магазина женской одежды.

2. Кафе или ресторана: Разработайте дизайн главной страницы сайта для местного кафе с меню и информацией о событиях.

3. Фитнес-центр: Создайте макет главной страницы для веб-сайта фитнес-центра с расписанием групповых занятий и тренеров.

4. Портфолио дизайнера: Разработайте дизайн главной страницы для личного портфолио графического или веб-дизайнера.

5. Блог о путешествиях: Создайте макет главной страницы для блога о путешествиях с разделами для статей и фотографий.

6. Сайт для учебного заведения: Разработайте дизайн главной страницы сайта для университета или колледжа с информацией о курсах и мероприятиях.

7. Сервис доставки еды: Создайте макет главной страницы для онлайн-сервиса доставки еды с возможностью заказа и меню.

8. Туристическое агентство: Разработайте дизайн главной страницы для туристического агентства с предложениями туров и специальными акциями.

9. Фонд благотворительности: Создайте макет главной страницы для благотворительного фонда с информацией о проектах и возможностями пожертвований.

10. Интернет-магазин электроники: Разработайте дизайн главной страницы для онлайн-магазина, который продаёт электронику и гаджеты.

11. Краудфандинговая платформа: Создайте макет главной страницы для краудфандингового сайта, который помогает запускать проекты.

12. Сайт для изучения языков: Разработайте дизайн главной страницы для платформы по изучению языков с курсами и преподавателями.

13. Служба такси: Создайте макет главной страницы для службы такси с возможностью заказа автомобиля и тарифами.

14. Магазин косметики: Разработайте дизайн главной страницы для интернет-магазина, который предлагает косметические товары.

15. Спортивная команда: Создайте макет главной страницы сайта для спортивной команды с расписанием матчей и новостями.

16. Магазин книг: Разработайте дизайн главной страницы для онлайн-магазина книг с разделами по жанрам.

17. Фотограф: Создайте макет главной страницы для портфолио фотографа с галереей работ и услугами.

18. Модный блог: Разработайте дизайн главной страницы для блога о моде с последними тенденциями и стилями.

19. Сайт для домашних животных: Создайте макет главной страницы для веб-сайта, который предлагает товары и услуги для домашних животных.

20. Вебинарная платформа: Разработайте дизайн главной страницы для платформы вебинаров с расписанием и возможностью регистрации.

21. Студия йоги: Создайте макет главной страницы для сайта студии йоги с расписанием занятий и информацией о тренерах.

22. Сайт для художника: Разработайте дизайн главной страницы для веб-сайта, посвящённого работам художника с галереей и информацией о выставках.

23. Агентство по подбору персонала: Создайте макет главной страницы для кадрового агентства с предложениями вакансий и услугами для соискателей.

24. Кулинарный блог: Разработайте дизайн главной страницы для кулинарного блога с рецептами и советами по готовке.

25. Сайт для аренды жилья: Создайте макет главной страницы для платформы по аренде жилья с возможностью поиска и фильтрации.

26. Финансовая консультация: Разработайте дизайн главной страницы для веб-сайта финансовой консультации с услугами и отзывами клиентов.

27. Магазин игрушек: Создайте макет главной страницы для интернет-магазина игрушек с разделами по возрастным группам.

28. Сайт для стартапов: Разработайте дизайн главной страницы для веб-сайта, который помогает стартапам с финансированием и развитием.

29. Курсы по программированию: Создайте макет главной страницы для образовательной платформы с курсами по программированию и отзывами студентов.

30. Сайт о здоровье и фитнесе: Разработайте дизайн главной страницы для блога о здоровье с советами, статьями и рецептами.

Контрольные вопросы:

1. Что такое UX?
2. Что такое UI?
3. В чем разница между UX и UI?
4. Что такое wireframe?
5. Какие модульные сетки являются наиболее распространёнными

Лабораторная работа №10

Разработка интерактивного прототипа веб-приложения с использованием Figma. Добавление интерактивных переходов и анимаций

Цель работы: научиться создавать интерактивные прототипы веб-приложений с использованием инструмента Figma, включая создание интерактивных переходов между экранами и анимации.

В современном веб-разработке создание интерактивного прототипа является важной стадией разработки веб-приложения. Интерактивный прототип позволяет тестировать и отрабатывать пользовательский интерфейс, а также определять требования к функциональности приложения.

1. Теоретические основы

Интерактивный прототип

Интерактивный прототип — это модель веб-приложения, которая позволяет тестировать и отрабатывать пользовательский интерфейс.

Интерактивный прототип позволяет определить требования к функциональности приложения и оценить пользовательский опыт.

Интерактивный прототип может быть создан с помощью различных инструментов, таких как Figma, Sketch, Adobe XD и других.

Интерактивные переходы и анимации

Интерактивные переходы — это элементы, которые используются для создания интерактивного опыта пользователя.

Интерактивные переходы могут быть созданы с помощью различных эффектов, таких как hover, click, swipe и других.

Анимации — это элементы, которые используются для создания интерактивного опыта пользователя.

Анимации могут быть созданы с помощью различных эффектов, таких как fade, slide, rotate и других.

Типы интерактивных прототипов

Низкоуровневый прототип — это простая модель веб-приложения, которая позволяет тестировать основные функции приложения.

Среднеуровневый прототип — это более сложная модель веб-приложения, которая позволяет тестировать более сложные функции приложения.

Высокоуровневый прототип — это полнофункциональная модель веб-приложения, которая позволяет тестировать все функции приложения.

Преимущества интерактивных прототипов

Интерактивные прототипы позволяют тестировать и отрабатывать пользовательский интерфейс.

Интерактивные прототипы позволяют определить требования к функциональности приложения.

Интерактивные прототипы позволяют оценить пользовательский опыт.

Интерактивные прототипы позволяют уменьшить время и стоимость разработки веб-приложения.

В рамках лабораторной работы вам необходимо создать интерактивный прототип веб-приложения с использованием Figma, включив в него не менее 5 экранов и настроив анимационные переходы между ними.

Этапы выполнения:

Зарегистрируйтесь в Figma (если у вас ещё нет аккаунта).

Создайте новый проект.

Создайте макеты не менее 5 экранов вашего веб-приложения.

Настройте переходы между экранами (по клику или другим событиям).

Добавьте анимации переходов (например, использование Smart Animate для изменения состояния кнопок или других элементов).

Проверьте интерактивный прототип в режиме просмотра (Play).

Сохраните и отправьте ссылку на ваш проект преподавателю.

Варианты заданий

Выберите один из вариантов интерфейсов веб-приложения и создайте интерактивный прототип:

1. Веб-приложение для заказа еды

Создать 5 экранов для приложения заказа еды (каталог блюд, экран блюда, корзина, оформление заказа, подтверждение заказа).

2. Приложение для онлайн-бронирования гостиниц

Прототип 5 экранов приложения для поиска и бронирования номеров в гостиницах (экран поиска, выбор отеля, информация о номере, бронирование, подтверждение брони).

3. Онлайн-банкинг

Создать 5 экранов для системы онлайн-банкинга (главная страница, экран транзакций, создание нового перевода, подтверждение транзакции, настройки пользователя).

4. Социальная сеть

Прототип для социальной сети с 5 экранами (лента новостей, профиль пользователя, создание нового поста, личные сообщения, настройки).

5. Приложение для заказа такси

Создать 5 экранов для приложения такси (ввод адреса, выбор маршрута, выбор машины, отслеживание поездки, оценка поездки).

6. Интернет-магазин одежды

Прототип 5 экранов для интернет-магазина одежды (главная страница, карточка товара, корзина, оформление заказа, подтверждение заказа).

7. Приложение для фитнеса

Создать 5 экранов для приложения по фитнесу (экран тренировок, выбор тренировки, экран с видео тренировки, планировщик, личные достижения).

8. Онлайн-кинотеатр

Прототип 5 экранов онлайн-кинотеатра (главная страница, каталог фильмов, страница фильма, просмотр фильма, комментарии).

9. Сервис бронирования билетов на мероприятия

Прототип 5 экранов приложения для покупки билетов на мероприятия (экран мероприятий, выбор события, выбор билета, оплата, подтверждение покупки).

10. Приложение для доставки цветов

Прототип 5 экранов для сервиса доставки цветов (каталог букетов, карточка товара, корзина, оформление заказа, подтверждение).

11. Приложение для бронирования авиабилетов

Прототип 5 экранов для системы бронирования авиабилетов (главная страница, поиск рейсов, выбор рейса, оплата, подтверждение брони).

12. Сервис аренды автомобилей

Прототип 5 экранов для аренды автомобилей (главная страница, выбор автомобиля, информация об автомобиле, бронирование, подтверждение аренды).

13. Приложение для планирования поездки

Создать 5 экранов для приложения, помогающего планировать путешествия (поиск направления, маршрут поездки, бронирование отеля, календарь мероприятий, завершение планирования).

14. Приложение для рецептов

Прототип 5 экранов приложения с рецептами (главная страница с категориями рецептов, экран с выбранным рецептом, шаги приготовления, список покупок, сохранение рецепта в избранное).

15. Приложение для чтения книг

Создать 5 экранов для приложения, где можно читать книги онлайн (каталог книг, выбор книги, экран чтения, закладки, личный кабинет пользователя).

16. Образовательная платформа

Прототип 5 экранов для онлайн-платформы обучения (главная страница, выбор курса, просмотр урока, задания, прогресс обучения).

17. Приложение для аренды жилья

Создать 5 экранов для сервиса аренды квартир (поиск квартир, выбор жилья, просмотр информации о жилье, бронирование, подтверждение брони).

18. Приложение для заметок

Прототип 5 экранов для приложения, где можно создавать и управлять заметками (главная страница, список заметок, создание новой заметки, редактирование заметки, категории заметок).

19. Приложение для мониторинга здоровья

Создать 5 экранов для приложения, отслеживающего параметры здоровья (экран с основными показателями, дневник питания, запись тренировок, отчеты, уведомления).

20. Приложение для вызова врача на дом

Прототип 5 экранов приложения для вызова врача на дом (выбор врача, ввод симптомов, календарь для записи, оформление вызова, подтверждение).

21. Приложение для работы с задачами

Создать 5 экранов для приложения управления задачами (главная страница, создание новой задачи, список задач, сортировка задач по приоритету, завершение задачи).

22. Онлайн-музей

Прототип 5 экранов для онлайн-музея (главная страница, выбор галереи, просмотр экспонатов, информация об экспонате, виртуальный тур).

23. Приложение для бронирования столиков в ресторане

Создать 5 экранов для приложения, позволяющего бронировать столики в ресторанах (поиск ресторанов, выбор ресторана, выбор времени и столика, подтверждение брони, уведомление о бронировании).

24. Приложение для управления проектами

Прототип 5 экранов для приложения по управлению проектами (список проектов, создание нового проекта, экран задачи, отслеживание прогресса, отчет по проекту).

25. Приложение для доставки продуктов

Создать 5 экранов для приложения доставки продуктов (каталог товаров, корзина, оформление заказа, выбор времени доставки, подтверждение заказа).

26. Приложение для бронирования спортивных залов

Прототип 5 экранов для приложения, позволяющего бронировать спортивные залы (список залов, выбор зала, бронирование, оплата, подтверждение).

27. Приложение для создания анимированных открыток

Создать 5 экранов для приложения, позволяющего создавать анимированные открытки (выбор шаблона, настройка открытки, добавление текста, просмотр анимации, отправка открытки).

28. Приложение для домашних дел

Прототип 5 экранов для приложения, помогающего управлять домашними делами (список дел, добавление нового дела, сортировка дел по категориям, завершение дела, уведомления).

29. Приложение для планирования мероприятий

Создать 5 экранов для приложения планирования мероприятий (главная страница, создание нового события, приглашение участников, календарь событий, подтверждение участия).

30. Приложение для фотогалереи

Прототип 5 экранов для приложения, где можно загружать и просматривать фотографии (главная страница, загрузка фото, альбомы, просмотр фото, редактирование фото)

Лабораторная работа №11

Разработка страницы с использованием CSS-препроцессора (SASS или LESS). Создание переменных и функций для управления стилями

Цель работы: освоение использования CSS-препроцессоров, таких как SASS или LESS, для создания более гибких и управляемых стилей. Научиться использовать переменные, вложенные правила и функции для создания оптимизированных и удобных в поддержке CSS-файлов.

1. Краткое теоретическое описание

1.1. Что такое CSS-препроцессоры?

CSS-препроцессоры — это инструменты, которые расширяют возможности стандартного CSS, позволяя использовать программные конструкции, такие как переменные, функции, циклы и вложенные правила. После написания стилей на языке препроцессора файл компилируется в обычный CSS, который браузеры могут интерпретировать.

Наиболее популярными CSS-препроцессорами являются SASS (Syntactically Awesome Stylesheets) и LESS. Оба они предлагают похожие функциональные возможности, такие как использование переменных, вложенных стилей, функций и миксинов.

1.2. Преимущество использования препроцессоров

1.2.1. Повторное использование кода:

Возможность задавать переменные и миксины (функции) для многократного использования одних и тех же стилей в разных частях кода.

1.2.2. Упрощенное управление стилями:

Можно централизовать настройку цветов, размеров шрифтов, отступов и других параметров, что облегчает поддержание и изменение стилей.

1.2.3. Логическая структура:

Вложенные правила позволяют сделать CSS более структурированным и интуитивно понятным, так как стили могут быть организованы в соответствии с иерархией HTML-элементов.

1.2.4. Удобство работы с цветами:

Препроцессоры позволяют выполнять математические операции с цветами (например, осветление или затемнение) через встроенные функции.

1.3. SASS и LESS: Основные возможности

1.3.1. **Переменные**: Позволяют задавать значения для использования в разных частях стилей.

В SASS переменные объявляются с $, а в LESS — с @.

Пример SASS:

```
// SASS
$primary-color: #3498db;

body {
  background-color: $primary-color;
}
```

Пример LESS:

```
// LESS
@primary-color: #3498db;
body {
  background-color: @primary-color;
}
```

1.3.2. **Вложенные правила**: Вложенность позволяет писать стили в соответствии с иерархией HTML-кода.

Пример SASS:

```
// SASS
nav {
  ul {
    margin: 0;
    padding: 0;

    li {
      list-style: none;
    }
  }
}
```

Пример LESS:

```
// LESS
nav {
  ul {
    margin: 0;
    padding: 0;

    li {
      list-style: none;
    }
  }
}
```

1.3.3. **Миксины**: Миксины — это блоки кода, которые можно переиспользовать. Они поддерживают параметры.

Пример SASS:

```
// SASS
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
  -moz-border-radius: $radius;
  border-radius: $radius;
}
button {
  @include border-radius(10px);
}
```

Пример LESS:

```
// LESS
.border-radius(@radius) {
  -webkit-border-radius: @radius;
  -moz-border-radius: @radius;
  border-radius: @radius;
}
button {
  .border-radius(10px);
}
```

1.3.4. **Функции**: Препроцессоры позволяют создавать собственные функции или использовать встроенные для работы с числами, строками, цветами и др.

Пример SASS:

```
// SASS
$base-padding: 10px;
@function double($number) {
  @return $number * 2;
}
.box {
  padding: double($base-padding); // padding: 20px;
}
```

1.3.5. **Операции**: Препроцессоры поддерживают математические операции для работы с единицами измерения.

Пример SASS:

```
// SASS
$width: 100px;
.box {
  width: $width / 2; // 50px
}
```

1.4.SASS vs. LESS. Основные отличия

1.4.1. Синтаксис:

SASS имеет два синтаксиса: оригинальный синтаксис без фигурных скобок (SASS) и синтаксис, подобный CSS (SCSS). LESS использует синтаксис, аналогичный SCSS;

1.4.2. Расширенные функции:

SASS имеет более мощный набор встроенных функций по сравнению с LESS (например, расширенная работа с цветами).

2. Ход работы

2.1. Настройка среды разработки

Для использования препроцессора необходимо его установить. Для этого можно воспользоваться Node.js и менеджером пакетов npm.

Установка SASS (Рис. 1):

```
npm install -g sass
```

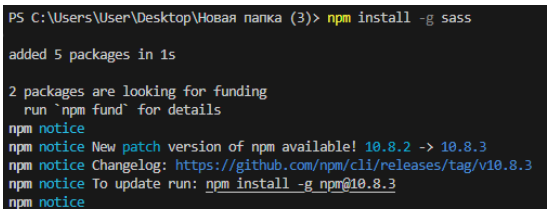

Рисунок 1 – Установка SASS

Установка LESS (Рис. 2):

```
npm install -g less
```

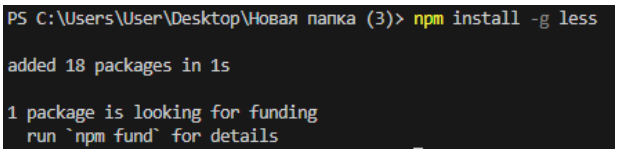

Рисунок 2 – Установка LESS

После того, как мы установили SASS и LESS мы можем перейти к написанию нашего кода. Для обработки кода требуется компиляция, она делается следующим образом:

Компиляция SASS:

```
sass input.scss output.css
```

Компиляция LESS:

```
lessc input.less output.css
```

После компиляции у нас появятся два новых файла с расширение .css (Рис. 3)

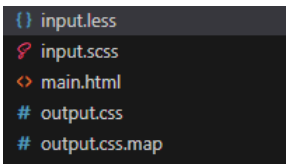

Рисунок 3 – Создание css

2.2. Создание проекта

Первое, что сделаем – это создадим наш файл .html и напишем простой код, который представлен на рисунке 4.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Preprocessor Example</title>
  <link rel="stylesheet" href="output.css">
</head>
<body>
  <header>
    <h1>Welcome to Preprocessor World</h1>
  </header>
  <main>
    <section class="content">
      <p>This is an example of using SASS or LESS for styling</p>
    </section>
  </main>
  <footer>
    <p>&copy; 2024</p>
  </footer>
</body>
</html>
```

Рисунок 4 – HTML код

Если открыть окно браузера, то появится данные заголовки:

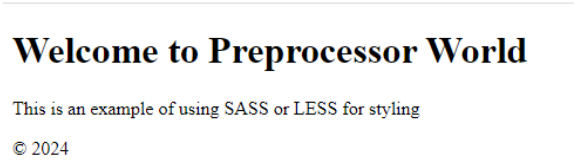

Рисунок 5 – Созданная страница

Теперь данную страницу стилизуем с помощи препроцессора SASS:

```scss
// Переменные
$primary-color: #3498db;
$secondary-color: #2ecc71;
$padding: 20px;

// Миксин для градиента
@mixin gradient($start, $end) {
  background: linear-gradient(to right, $start, $end);
}

// Стили для header
header {
  padding: $padding;
  background-color: $primary-color;
  h1 {
    color: white;
  }
}

// Стили для контента
.content {
  padding: $padding;
  @include gradient($primary-color, $secondary-color);
  p {
    color: white;
  }
}

// Стили для footer
footer {
  padding: $padding;
  background-color: darken($primary-color, 10%);
  p {
    color: white;
  }
}
```

Рисунок 6 – Создание стилей

После того, как мы создали стили, нам нужно сделать компиляцию кода. Следующим шагом уже можно открыть наш сайт и посмотреть на результат работы (Рис. 7)

Рисунок 7 – Готовый сайт

Как видно на изображении выше все стили применились.

3. Задания на лабораторную работу (Работа по вариантам, варианты брать из ЛР 8)

3.1. Использование переменных

— Создайте страницу с базовой HTML-структурой;

— Используйте переменные в SASS или LESS для задания цветовой схемы, отступов и размеров шрифтов;

— Создайте два варианта цветовой схемы для фона и текста с использованием переменных.

3.2. Использование миксинов и функций

— Разработайте страницу с использованием SASS/LESS;

— Создайте миксины для повторяющихся стилей (например, для отступов, теней);
— Реализуйте функцию для изменения яркости цветов элементов;
— Настройте градиенты для фона с помощью миксинов.

3.3. Вложенные стили и логическая структура

— Создайте страницу с разделами (header, nav, main, footer);
— Используйте вложенные правила для стилизации элементов внутри разделов;
— Примените математические операции для расчета размеров блоков или отступов.

Содержание отчёта:

7. Титульный лист;
8. Цель работы;
9. Задание;
10. Ход работы:
    a. Описание задания, скриншот выполнения, листинг представить в приложении;
11. Вывод;
12. Приложения.

Критерии отчёта:

5. Оформление шрифтом Times New Roman (14 кегль);
6. Поля: верх 2 см, лево 3 см, право 1,5 см, нижнее 2 см;
7. Междустрочный интервал 1,5;
8. Отступ красной строки 1,25.

Контрольные вопросы:

1. Что такое CSS-препроцессор, и как он отличается от обычного CSS?
2. Какие основные возможности предоставляют препроцессоры SASS и LESS?
3. Для чего используются переменные в SASS/LESS, и как они помогают управлять стилями?
4. Как создаются миксины в SASS и LESS, и чем они отличаются от функций?
5. Какую пользу при верстке страниц дает использование вложенных стилей?

6. Как выполняется компиляция препроцессора в CSS?
7. Чем отличается синтаксис SCSS от оригинального синтаксиса SASS?
8. Приведите пример использования математических операций в препроцессоре.
9. Как с помощью функций в SASS можно управлять цветами элементов?
10. В чем заключаются преимущества использования препроцессоров в больших проектах?

Лабораторная работа №12

Использование системы контроля версий Git для управления проектом. Основные команды: init, commit, push, branch

Цель работы: освоение базовых команд системы контроля версий Git для управления проектом. Научиться инициализировать репозиторий, добавлять изменения, фиксировать их, отправлять в удаленный репозиторий и работать с ветвлениями. Это позволит эффективно управлять разработкой кода и отслеживать изменения в проекте.

1. Краткая теоретическая часть

1.1. Что такое Git?

Git — это распределенная система контроля версий, которая позволяет разработчикам отслеживать изменения в коде, работать совместно над проектами и управлять версиями программного обеспечения. Это один из наиболее популярных инструментов для управления кодом в программной разработке.

1.2. Зачем использовать систему контроля версий?

1.2.1. Отслеживание изменений:

Git сохраняет историю всех изменений в проекте, что позволяет вернуться к любой из предыдущих версий кода.

1.2.2. Совместная работа:

Несколько разработчиков могут одновременно работать над одним проектом, избегая конфликтов и дублирования кода.

1.2.3. Безопасность данных:

Система Git хранит репозиторий локально, но также поддерживает синхронизацию с удаленными репозиториями (например, на GitHub или GitLab).

1.2.4. Работа с ветвями:

Ветвления позволяют параллельно развивать несколько направлений разработки (например, отдельная ветка для новых фич или багфиксов), не нарушая стабильность основной ветки.

1.3. Основные команды Git

1.3.1. git init – Инициализирует новый локальный репозиторий в текущей директории. После выполнения этой команды Git начинает отслеживать изменения в проекте;

1.3.2. git status – показывает текущее состояние рабочего каталога и информацию о том, какие файлы были изменены, добавлены в индекс (стейджинг) и какие нужно закоммитить;

1.3.3. git add <filename> – добавляет изменения в индекс (staging area), подготовив их для коммита. Для добавления всех изменений используется gid add . (именно с точкой);

1.3.4. git commit -m "Сообщение о коммите" – создает новый коммит — зафиксированное состояние проекта с описанием изменений. При выполнении команды требуется указать сообщение о коммите;

1.3.5. git push – отправляет коммиты из локального репозитория в удаленный. Эта команда используется для синхронизации изменений с удаленным репозиторием (например, на GitHub или GitLab);

1.3.6. git pull – забирает последние изменения из удаленного репозитория в локальный, обновляя рабочее состояние проекта;

1.3.7. git branch <branch_name> – позволяет управлять ветками в репозитории. Ветки позволяют параллельно работать над разными задачами, не нарушая основную версию кода. Переключение на ветку git checkout <branch_name>;

1.3.8. git merge <branch_name> – объединяет изменения из одной ветки в другую.

1.4. Ветвления в Git

Ветви позволяют разделить разработку на несколько параллельных процессов. Например, одна ветка может быть основной (чаще всего это main или master), в то время как другие ветви используются для разработки новых функций или исправления ошибок. Ветви можно создавать, переключаться между ними, а также объединять изменения при необходимости.

1.5. GitHub и другие удаленные репозитории

GitHub — это платформа для хостинга Git-репозиториев, которая позволяет работать с удаленными проектами, координировать совместную работу и управлять версиями программ. GitLab и Bitbucket — альтернативные платформы, предоставляющие аналогичные функции.

2. Ход работы

2.1. Инициализация Git-репозитория

Создаем новую директорию для проекта:

```
mkdir my_project
```

```
cd my_project
```

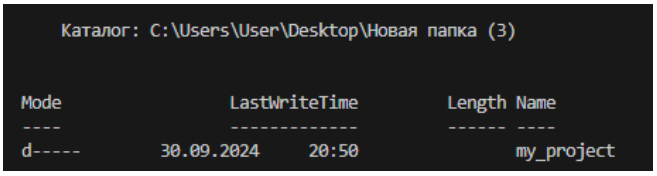

Рисунок 1 – Создание директивы

Инициализируем репозиторий

```
git init
```

Рисунок 2 – Инициализация репозитория

Создаем новый файл .html или берем уже существующий. Например, создадим новый:

```
echo "<!DOCTYPE html><html><head><title>My Project</title></head><body><h1>Hello, Git!</h1></body></html>" > index.html
```

В нашей папке появится новый файл index.html с префиксом U. Данный префикс обозначает, что наш файл обновлен.

Рисунок 3 – Создание файла

Проверим статус репозитория:

```
git status
```

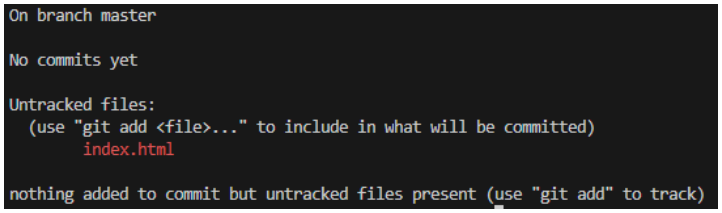

Рисунок 4 – Проверка статуса репозитория

На данный момент видно, что файл index.html добавлен, но не был зафиксирован, сейчас это поправим:

```
git add index.html
```

Рисунок 5 – Фиксация файла

Теперь наш файл полностью добавлен и зафиксирован в репозитории, еще раз проверим статус нашего репозитория

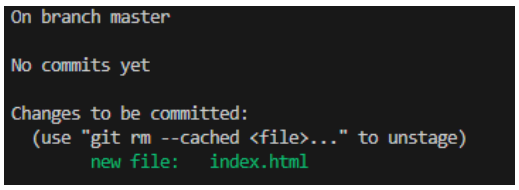

Рисунок 6 – Повторная проверка статуса

Теперь создадим коммит, который зафиксирует текущие разработки, а так же описание, которое мы добавим покажет то, что мы наработали для других программистов репозитория:

```
git commit -m "Initial commit: added index.html"
```

Так как мы еще не авторизовались системе Git, то перед коммитом консоль просит это сделать

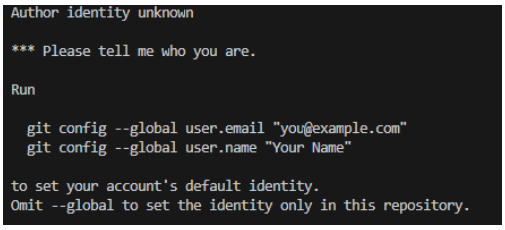

Рисунок 7 – Ошибка коммита

Для авторизации нам нужно только прописать команду:

```
git config –global user.email you@example.com
```

После данного действия мы можем снова закоммитить наш файл и всё должно увенчаться успехом

Рисунок 8 – Коммит

3. Задания на лабораторную работу

3.1. Инициализация и первый коммит

— Инициализируйте новый Git-репозиторий в папке проекта;

— Создайте несколько HTML и CSS файлов для сайта;

— Зафиксируйте изменения с осмысленным сообщением в коммите.

3.2. Работа с ветвями

— Создайте ветку для новой функции (например, добавления нового раздела на сайт);

— Выполните изменения в этой ветке, зафиксируйте их в коммит;

— Объедините изменения с основной веткой.

3.3. Работа с удаленным репозиторием

— Создайте удаленный репозиторий на GitHub или другой платформе;

— Подключите локальный проект к удаленному репозиторию;

— Отправьте изменения в удаленный репозиторий с помощью команды git push.

Содержание отчёта:

13. Титульный лист;
14. Цель работы;
15. Задание;
16. Ход работы:
    a. Описание задания, скриншот выполнения, листинг представить в приложении;
17. Вывод;
18. Приложения.

Критерии отчёта:

9. Оформление шрифтом Times New Roman (14 кегль);
10. Поля: верх 2 см, лево 3 см, право 1,5 см, нижнее 2 см;
11. Междустрочный интервал 1,5;
12. Отступ красной строки 1,25.

Контрольные вопросы:

1. Что такое система контроля версий Git?
2. Для чего используется команда git init?
3. Как работает команда git commit и зачем она нужна?
4. Как отправить изменения в удаленный репозиторий?
5. Что такое ветвление и как его использовать в Git?
6. Как объединить изменения из одной ветки в другую с помощью Git?
7. Для чего используется команда git add?
8. Чем отличается git pull от git push?
9. Что такое удаленный репозиторий и как с ним работать?

Лабораторная работа №13

Проведение юзабилити-тестирования созданного интерфейса с привлечением нескольких пользователей. Анализ результатов и выявление проблемных областей

Цель работы: научиться проводить юзабилити-тестирование интерфейса.

1. Теоретический материал

Юзабилити (удобство использования) — это степень, в которой продукт может быть использован определёнными пользователями для достижения конкретных целей с эффективностью, продуктивностью и удовлетворённостью в конкретном контексте использования. Целью юзабилити-тестирования является выявление проблем с интерфейсом, которые могут затруднять его использование.

Ключевые аспекты юзабилити:

Эффективность: насколько пользователи могут достигать своих целей.

Производительность: как быстро и с минимальными усилиями пользователи могут выполнить задачу.

Удовлетворённость: насколько пользователи довольны взаимодействием с продуктом.

Юзабилити-тестирование — это процесс оценки интерфейса с привлечением реальных пользователей для выявления проблемных зон, ошибок или непонятных элементов.

Виды юзабилити-тестирования

Формативное тестирование: проводится на этапе разработки интерфейса. Его цель — выявить проблемы на ранних этапах и исправить их.

Суммативное тестирование: проводится после завершения разработки для оценки общей эффективности интерфейса.

Подготовка к юзабилити-тестированию

Этапы подготовки:

Определение целей тестирования: что именно необходимо проверить? Какие области интерфейса наиболее критичны для тестирования?

Формирование гипотез: предполагаемые проблемные зоны или узкие места в интерфейсе.

Выбор сценариев использования: создание сценариев, которые пользователи будут выполнять во время тестирования.

Выбор участников: необходимо определить целевую аудиторию, подходящую для тестирования (набор нескольких пользователей, которые представляют разные группы пользователей интерфейса).

Подготовка оборудования и инструментов: выбор программного обеспечения для записи сессий (экран пользователя, его действия, голос), блокнотов для заметок и др.

Проведение тестирования

Введение: перед началом тестирования, важно объяснить пользователям цель тестирования, их задачи и основные правила (например, что тестируется интерфейс, а не их навыки).

Сценарии: участникам предлагается выполнить заранее подготовленные задачи, например: "Найдите продукт X в интернет-магазине" или "Оформите заказ на доставку".

Наблюдение: записывается процесс работы пользователя с интерфейсом, отмечаются затруднения, ошибки, сбои.

Обратная связь: после тестирования проводится обсуждение с пользователем его впечатлений, трудностей и предложений.

Принципы тестирования:

Нейтральное поведение: не подсказывать пользователям, как действовать.

Пассивное наблюдение: следить за действиями пользователя, не вмешиваясь в процесс.

Анализ результатов

После завершения тестирования необходимо:

Обработать данные: просмотреть записи тестов, проанализировать заметки, обратить внимание на те моменты, где пользователи испытывали затруднения.

Сгруппировать проблемы: выделить основные проблемные области (навигация, непонятные элементы интерфейса, сложность в выполнении задачи и др.).

Приоритизация проблем: разделить найденные проблемы на три группы по важности:

Критические: мешают пользователям достичь своих целей (например, недоступная функция).

Средние: вызывают затруднения, но цели могут быть достигнуты (например, непонятное название кнопки).

Незначительные: не мешают работе, но могут быть улучшены (например, оформление элементов).

Выводы и рекомендации: на основании выявленных проблем предложить пути их решения. Рекомендуется сосредоточиться на критических и средних проблемах.

Отчёт о юзабилити-тестировании

В итоговом отчёте должны быть:

Цели и задачи тестирования.

Описание участников (с кратким указанием их опыта).

Сценарии, используемые для тестирования.

Ключевые результаты: перечень выявленных проблем.

Рекомендации по улучшению интерфейса.

Юзабилити-тестирование — это мощный инструмент для улучшения интерфейсов, который помогает выявить слабые места и адаптировать продукт под реальные нужды пользователей. Регулярное проведение таких тестов на всех этапах разработки позволяет значительно повысить удовлетворённость конечных пользователей и улучшить их опыт взаимодействия с продуктом.

2. Варианты для выполнения индивидуального задания

1. Провести юзабилити-тестирование интернет-магазина и выявить проблемные места в процессе оформления заказа.
2. Оценить удобство навигации на новостном сайте, определить сложные элементы интерфейса.
3. Провести тестирование мобильного приложения для бронирования отелей и выявить затруднения пользователей при поиске и выборе отеля.
4. Оценить интерфейс учебной онлайн-платформы на удобство использования при выполнении тестовых заданий.
5. Провести юзабилити-тестирование интернет-банкинга, оценив удобство оплаты счетов.
6. Проанализировать интерфейс социальной сети на предмет удобства поиска и добавления новых друзей.

7. Провести юзабилити-тестирование приложения для заказа еды, выявить проблемные зоны в процессе выбора и заказа блюда.
8. Оценить интерфейс сайта авиакомпании и выявить затруднения при бронировании билетов.
9. Провести тестирование мобильного приложения для фитнеса на удобство создания и редактирования тренировок.
10. Оценить юзабилити корпоративного портала и выявить затруднения при поиске информации.
11. Провести тестирование сайта онлайн-магазина электроники, оценив удобство фильтрации товаров.
12. Оценить интерфейс приложения для вызова такси, выявив проблемные зоны в процессе заказа машины.
13. Провести юзабилити-тестирование сайта образовательного учреждения, оценив удобство навигации по разделам "Абитуриентам" и "Студентам".
14. Оценить интерфейс приложения для планирования поездок, выявить затруднения при составлении маршрута.
15. Провести тестирование сайта музыкального стриминга, оценив удобство поиска и прослушивания треков.
16. Оценить интерфейс приложения для онлайн-обучения иностранным языкам, выявить проблемы при выполнении упражнений.
17. Провести юзабилити-тестирование интернет-магазина одежды, оценив удобство выбора размеров и оформления возврата.
18. Оценить интерфейс приложения для ведения бюджета и выявить сложности в добавлении и редактировании расходов.
19. Провести тестирование платформы для проведения видеоконференций, оценив удобство настройки и подключения к встречам.
20. Оценить юзабилити сайта бронирования билетов в кино, выявить затруднения при выборе места и времени сеанса.
21. Провести тестирование интерфейса приложения для аренды велосипедов и выявить проблемы с поиском доступных велосипедов.
22. Оценить интерфейс приложения для чтения электронных книг, выявить затруднения в навигации по библиотеке и поиске книг.

23. Провести юзабилити-тестирование интерфейса CRM-системы, оценив удобство добавления клиентов и задач.
24. Оценить юзабилити сайта интернет-аптеки, выявить проблемы с поиском товаров и оформлением рецептов.
25. Провести тестирование интерфейса приложения для отслеживания здоровья (шаги, сон и т. д.), выявив трудности в использовании аналитических данных.
26. Оценить интерфейс онлайн-калькулятора для расчёта ипотеки, выявить проблемы при вводе данных.
27. Провести юзабилити-тестирование онлайн-сервиса для бронирования столиков в ресторане, выявив сложности при выборе времени и места.
28. Оценить интерфейс приложения для мониторинга погоды, выявить проблемы с доступом к подробным прогнозам.
29. Провести тестирование интерфейса сайта благотворительного фонда, оценив удобство процесса пожертвований.
30. Оценить юзабилити приложения для аренды недвижимости, выявив сложности при фильтрации объявлений и общении с арендодателями.

Контрольные вопросы:

1. Что такое юзабилити и какие его ключевые аспекты?
2. Чем отличается формативное юзабилити-тестирование от суммативного?
3. Какие цели преследует юзабилити-тестирование?
4. Какие типы пользователей следует привлекать для юзабилити-тестирования и почему?
5. Что включает в себя подготовка к юзабилити-тестированию?
6. Какие основные принципы необходимо соблюдать при проведении юзабилити-тестирования?
7. Как правильно формулировать сценарии для юзабилити-тестирования? Приведите пример.
8. Какие методы анализа результатов юзабилити-тестирования существуют?

Лабораторная работа №14

Сбор метрик пользовательского опыта с использованием Google Analytics

Цель работы: изучить методы сбора и анализа метрик пользовательского опыта на веб-сайтах с использованием Google Analytics

1. Теоретическая часть

Введение в метрики пользовательского опыта

Пользовательский опыт (UX) — это совокупность эмоций и взаимодействий, которые возникают у пользователя при взаимодействии с продуктом, сайтом или приложением. Веб-аналитика помогает оценить, насколько эффективно и комфортно пользователи взаимодействуют с веб-ресурсом.

Основные метрики Google Analytics для оценки UX:

Количество пользователей (Users) — число уникальных пользователей, посетивших сайт.

Сеансы (Sessions) — количество отдельных посещений сайта. Один пользователь может создать несколько сеансов.

Средняя длительность сеанса (Average session duration) — среднее время, которое пользователь проводит на сайте.

Показатель отказов (Bounce Rate) — процент сеансов, в ходе которых пользователь не совершил никаких действий на сайте (например, покинул страницу после первого посещения).

Глубина просмотра (Pages per session) — среднее количество страниц, посещенных пользователем за один сеанс.

Конверсии (Conversions) — целевые действия, которые пользователи совершают на сайте (например, покупка, заполнение формы, подписка).

Процент новых пользователей (New Users) — доля новых пользователей среди всех посетителей сайта.

Источники трафика (Traffic sources) — источники, с которых пользователи пришли на сайт (поисковые системы, социальные сети, прямые заходы и т.д.).

Основы работы с Google Analytics

Google Analytics — это бесплатная веб-аналитическая платформа, позволяющая владельцам сайтов собирать, отслеживать и анализировать данные о посещаемости и поведении пользователей. Для использования Google Analytics необходимо зарегистрировать аккаунт, добавить трекинговый код на сайт и настроить сбор данных.

Трекинговый код Google Analytics — это небольшой JavaScript код, который вставляется на все страницы сайта. Он отправляет информацию о взаимодействиях пользователя с сайтом на сервера Google.

Цели (Goals) — это параметры, по которым можно отслеживать выполнение определенных действий на сайте, таких как покупка, подписка на рассылку, или просмотр конкретной страницы.

События (Events) — это настраиваемые параметры для отслеживания действий пользователей, таких как клики по кнопкам, просмотр видео, загрузки файлов и т.д.

Сегментация аудитории в Google Analytics

Сегментация позволяет разделить аудиторию сайта на группы для более глубокого анализа. Например, можно сегментировать пользователей по возрасту, географии, устройствам или источникам трафика. Это помогает лучше понять, как различные группы пользователей взаимодействуют с сайтом.

Пользовательские отчеты и дашборды

Google Analytics позволяет создавать кастомные отчеты и дашборды для отслеживания наиболее важных метрик и ключевых показателей эффективности (KPI). Важно выбрать только те метрики, которые напрямую связаны с целями сайта**.**

Взаимодействие с данными в реальном времени

Google Analytics предоставляет возможность отслеживать действия пользователей на сайте в режиме реального времени. Этот раздел позволяет наблюдать текущую активность на сайте, такие как:

Текущие пользователи на сайте — количество людей, которые в данный момент находятся на сайте.

Просматриваемые страницы — какие страницы сейчас открыты пользователями.

Источники трафика в реальном времени — откуда пришли пользователи.

География пользователей — местоположение пользователей, которые активно взаимодействуют с сайтом.

Это может быть полезно при запуске рекламных кампаний, обновлении контента или анализе поведения пользователей в определенные моменты (например, во время пиковых нагрузок на сайт).

Важность событий (Events) для оценки поведения пользователей

События (Events) — это ключевые действия, которые пользователи совершают на сайте. В отличие от стандартных метрик, таких как количество посещений или время на странице, события настраиваются для отслеживания конкретных пользовательских взаимодействий, что позволяет глубже понять, как именно пользователь взаимодействует с сайтом.

Примеры событий:

Клики по элементам интерфейса — отслеживание кликов на кнопки, ссылки, формы.

Загрузка файлов — когда пользователь загружает документ, изображение или другой файл.

Взаимодействие с мультимедийным контентом — такие события, как воспроизведение, пауза, перемотка видео.

Просмотры определенных элементов — отслеживание просмотра конкретных разделов страницы (например, пользователи прокрутили до конца страницы).

Ценность событий заключается в их способности предоставлять подробные данные об активности пользователей, которые не всегда видны при анализе стандартных метрик. Например, если пользователь провел мало времени на странице, но при этом кликнул на все основные кнопки, это может означать, что страница была интуитивно понятной и удобной.

Цели (Goals) в Google Analytics: ключ к измерению успешности

Цели (Goals) в Google Analytics — это настроенные параметры, по которым можно отслеживать выполнение ключевых для бизнеса действий на сайте. Важно понимать, что цели представляют собой конечные точки или события, к которым должен стремиться сайт для выполнения своей задачи (продажи, регистрации, заполнения форм).

Типы целей в Google Analytics:

Целевая страница (Destination) — фиксирует посещение пользователем определенной страницы (например, страницы благодарности за покупку или регистрации).

Продолжительность сеанса (Duration) — отслеживает, когда пользователь провел на сайте определенное количество времени (например, более 5 минут).

Количество просмотренных страниц за сеанс (Pages per session) — фиксирует достижение цели, если пользователь просмотрел определенное количество страниц за одно посещение.

Событие (Event) — отслеживает любые другие важные действия, такие как клики по кнопкам, скачивания файлов, просмотр видео и т.д.

Цели позволяют измерить, насколько эффективно сайт выполняет поставленные задачи, а также оценить качество взаимодействия пользователей с ресурсом.

Ключевые показатели эффективности (KPI) для веб-аналитики

KPI (Key Performance Indicators) — это ключевые показатели, которые помогают оценить эффективность сайта и маркетинговых усилий. Правильно подобранные KPI позволяют точно оценить, насколько сайт достигает своих целей и насколько успешно пользователи взаимодействуют с ресурсом.

Примеры KPI для оценки пользовательского опыта:

Конверсии — количество пользователей, выполнивших целевые действия на сайте (например, покупка, регистрация, отправка формы).

Показатель отказов (Bounce Rate) — процент пользователей, которые покинули сайт, не совершив на нем никаких действий. Высокий показатель отказов может свидетельствовать о проблемах с навигацией или контентом.

Среднее время на сайте (Average Time on Site) — важный показатель, отражающий степень вовлеченности пользователя в контент сайта.

Глубина просмотра (Pages per Session) — среднее количество страниц, которое пользователь посещает за один сеанс.

Возврат пользователей (Returning Users) — процент пользователей, которые возвращаются на сайт повторно, что может свидетельствовать о лояльности к ресурсу.

Для каждого сайта набор KPI будет индивидуальным в зависимости от его целей и задач. Например, для интернет-магазина одним из важнейших показателей будет процент конверсий, тогда как для блога — среднее время, которое пользователи проводят на сайте.

Инструменты и методы повышения качества пользовательского опыта на основе данных

Собранные данные в Google Analytics могут использоваться для улучшения UX (user experience). Анализируя поведение пользователей, можно выявить узкие места, которые мешают эффективному взаимодействию, и предпринять шаги для их устранения.

Примеры улучшений UX на основе данных:

Оптимизация страниц с высоким показателем отказов. Если Google Analytics показывает высокий bounce rate для определенных страниц, стоит проанализировать контент, навигацию и скорость загрузки страницы, чтобы улучшить пользовательский опыт.

А/Б тестирование (A/B Testing). На основе данных Google Analytics можно проводить эксперименты с изменением дизайна, структуры или контента сайта, чтобы понять, какие изменения положительно влияют на метрики.

Оптимизация посадочных страниц. Анализ данных по источникам трафика позволяет определить, какие страницы сайта привлекают больше пользователей и как можно улучшить их структуру для повышения конверсий.

Персонализация контента. Если сайт имеет аудиторию с разными интересами (например, пользователи из разных регионов или возрастных групп), можно использовать данные сегментации для персонализации контента и предложений для разных сегментов пользователей.

Безопасность данных и конфиденциальность

При работе с Google Analytics важно помнить о защите данных пользователей и соблюдении законодательства в области конфиденциальности. В частности, это касается законов о защите персональных данных, таких как GDPR (Общий регламент по защите данных) в Европе. Основные меры по обеспечению конфиденциальности данных:

Анонимизация IP-адресов.

Информирование пользователей о том, что их данные собираются с помощью Google Analytics (например, через политику конфиденциальности).

Возможность для пользователей отказаться от отслеживания с помощью файлов cookie.

2. Варианты индивидуальных заданий

30 Вариантов заданий для лабораторной работы по теме "Сбор метрик пользовательского опыта с использованием Google Analytics":

Варианты 1–10: Основные метрики и анализ пользовательского поведения

1. Анализ посещаемости сайта:Установите трекинговый код Google Analytics на сайт, соберите данные за неделю и проанализируйте общее количество пользователей, количество сеансов, среднюю длительность сеанса и показатель отказов.

2. Анализ глубины просмотра страниц: Проанализируйте среднее количество страниц, которые посещают пользователи за один сеанс, и выясните, на каких страницах показатель отказов выше всего.

3. Сравнение новых и возвращающихся пользователей Проанализируйте доли новых и возвращающихся пользователей, определите, как они ведут себя на сайте, включая длительность сеансов и глубину просмотра.

4. Анализ по устройствам: Проанализируйте поведение пользователей, посетивших сайт с различных устройств (компьютер, мобильный телефон, планшет). Сравните показатели отказов, длительность сеансов и конверсии.

5. Анализ по географии: Сравните данные по пользователям из разных регионов. Найдите страны или города с наибольшим количеством посетителей, и проанализируйте их поведение на сайте.

6. Анализ времени активности: Исследуйте, в какое время дня и дни недели на сайте наблюдается наибольшая активность пользователей. Определите корреляции с показателями отказов и конверсиями.

7. Показатель отказов по страницам: Найдите страницы с самым высоким показателем отказов. Проанализируйте, почему пользователи могут покидать эти страницы и предложите решения для улучшения.

8. Анализ каналов трафика: Проанализируйте, какие источники трафика (органический, платный, социальные сети, реферальный) приносят больше всего пользователей и сравните их поведение на сайте.

9. **Анализ поведения мобильных пользователей:** Определите поведение пользователей на мобильных устройствах: длительность сеансов, глубина просмотра, показатель отказов. Сравните с поведением пользователей на десктопе.

10. **Анализ сеансов с поисковыми запросами:** Найдите, какие поисковые запросы приводят пользователей на сайт, и проанализируйте их поведение (глубину просмотра, длительность сеансов и конверсии).

Варианты 11–20: Конверсии, цели и события

11. Настройка и анализ макро-конверсий Настройте конверсию для покупки товара или услуги на сайте и проанализируйте пути пользователей, которые привели к совершению этой конверсии.

12. Отслеживание заполнения формы: Настройте цель на отслеживание успешного заполнения формы обратной связи и проанализируйте, сколько пользователей выполняют это действие.

13. Отслеживание подписок: Настройте цель для отслеживания подписок на рассылку и выясните, сколько пользователей подписываются, а также с каких страниц это происходит чаще всего.

14. Анализ воронки конверсий: Настройте цели для нескольких этапов пользовательского пути (например, посещение страницы продукта, добавление товара в корзину, завершение покупки) и проанализируйте, на каком этапе пользователи чаще всего покидают сайт.

15. Отслеживание событий: клики по кнопке "Купить": Настройте событие для отслеживания кликов по кнопке "Купить" и проанализируйте, на каких страницах это действие происходит чаще всего.

16. Отслеживание скачиваний файлов: Настройте событие для отслеживания скачиваний файлов (например, PDF) и проанализируйте, какие материалы чаще всего загружаются пользователями.

17. Отслеживание просмотра видео: Настройте событие для отслеживания просмотров видео на сайте. Определите, как много пользователей досматривают видео до конца.

18. Отслеживание кликов по внешним ссылкам: Настройте событие для отслеживания кликов по внешним ссылкам на сайте и проанализируйте, какие ссылки пользователи посещают чаще всего.

19. Отслеживание взаимодействий с элементами интерфейса: Настройте событие для отслеживания взаимодействий с конкретными элементами сайта (например, клик по навигационному меню) и выясните, какие элементы наиболее популярны.

20. Анализ отказов по целям: Настройте несколько целей (например, покупка товара, подписка) и проанализируйте показатель отказов для каждой цели. Определите, какие цели имеют самый низкий показатель отказов.

Варианты 21–30: Сегментация аудитории и кастомные отчеты

21. Сегментация пользователей по возрасту: Проанализируйте поведение пользователей разных возрастных групп. Сравните длительность сеансов, глубину просмотра и показатели отказов.

22. Сегментация пользователей по полу: Проведите анализ пользователей по полу. Определите, как мужчины и женщины взаимодействуют с сайтом, и сравните конверсии между этими группами.

23. Сегментация по устройствам: Проанализируйте, как пользователи взаимодействуют с сайтом на мобильных устройствах по сравнению с настольными компьютерами. Выявите особенности поведения каждой группы.

24. Сегментация по источникам трафика: Проведите сегментацию по источникам трафика (органический поиск, платный трафик, социальные сети). Определите, какой канал трафика приводит к лучшим показателям конверсий.

25. Сегментация по интересам пользователей: Используя данные Google Analytics, проанализируйте интересы пользователей и определите, какие темы привлекают больше всего посетителей на сайт.

26. Анализ поведения новых пользователей: Сегментируйте новых пользователей и проанализируйте их поведение на сайте, включая показатель отказов, глубину просмотра и конверсии.

27. Анализ поведения возвращающихся пользователей: Проанализируйте поведение пользователей, которые посещают сайт повторно. Определите, какие страницы они посещают чаще всего и какие действия выполняют.

28. Создание кастомного отчета для анализа конверсий: Создайте пользовательский отчет для анализа пути пользователей до совершения конверсии. Включите в отчет шаги, которые ведут к конверсии, и сегментируйте по различным источникам трафика.

29. Создание кастомного дашборда для руководства: Настройте дашборд с ключевыми показателями (количество пользователей, конверсии, глубина просмотра, длительность сеансов) для регулярного мониторинга сайта.

30. Анализ по времени посещений сайта: Проанализируйте, как меняется поведение пользователей в зависимости от времени дня или дня недели. Определите, в какое время сайт получает наибольшую конверсию и активность пользователей.

Контрольные вопросы:

1. Что такое показатель отказов (Bounce Rate) и как он интерпретируется при анализе пользовательского поведения?
2. Какие основные метрики можно отслеживать с помощью Google Analytics для анализа пользовательского опыта?

3. В чем разница между метриками (Metrics) и измерениями (Dimensions) в Google Analytics? Приведите примеры.
4. Что такое цель (Goal) в Google Analytics и какие типы целей можно настроить?
5. Как настраивается отслеживание событий (Events) в Google Analytics? Какие параметры необходимо задать для отслеживания событий?
6. Какое значение имеет сегментация аудитории в Google Analytics? Приведите примеры сегментации.
7. Какие виды источников трафика можно отслеживать в Google Analytics и как они влияют на поведение пользователей?

Лабораторная работа №15

Исследование современных трендов в UX/UI дизайне. Создание концепции интерфейса, который включает последние дизайнерские решения

Цель работы: ознакомиться с современными трендами в UX/UI дизайне, изучить принципы проектирования интерфейсов, разработку концепции интерфейса с учетом последних дизайнерских решений.

1. Теоретический материал

Введение в UI/UX-дизайн

UI (User Interface): интерфейс, с которым взаимодействует пользователь. Это визуальные элементы, такие как кнопки, иконки, меню и т. д.

UX (User Experience): опыт пользователя при взаимодействии с продуктом. Включает в себя удобство, функциональность и удовлетворенность.

Важность: Хороший UI/UX-дизайн обеспечивает не только привлекательный внешний вид, но и удобство использования, что критически важно для успешности продукта.

Современные тренды в UI/UX-дизайне

Минимализм:

- Принцип "меньше — значит больше".
- Упрощение интерфейса для повышения удобства восприятия.

Микровзаимодействия:

- Небольшие анимации и отклики, которые делают взаимодействие более интуитивным и приятным.

- Примеры: анимация кнопки при нажатии, индикаторы загрузки.

Адаптивный и отзывчивый дизайн:

- Подгонка интерфейса под различные размеры экранов и устройства.

- Важность создания единообразного опыта на всех платформах.

Темные темы:

- Стилизация интерфейсов в темных тонах.

- Преимущества: снижение нагрузки на глаза, экономия энергии на OLED-экранах.

-Иллюстрации и анимации:

- Использование графических элементов для повышения вовлеченности пользователей.

- Создание уникального визуального стиля и имиджа бренда.

Голосовой интерфейс:

- Рост популярности голосовых помощников (например, Siri, Google Assistant).

- Упрощение взаимодействия и доступность для пользователей с ограниченными возможностями.

Доступность:

- Создание интерфейсов, доступных для всех пользователей, включая людей с ограниченными возможностями.

- Применение стандартов доступности, таких как WCAG (Web Content Accessibility Guidelines).

Принципы проектирования интерфейсов

Пользовательские исследования:

- Необходимость в понимании потребностей, предпочтений и поведения пользователей.

- Методы: опросы, интервью, тестирование прототипов.

-Прототипирование и тестирование:

- Итеративный процесс разработки: создание прототипов, их тестирование и получение обратной связи.

- Прототипы могут быть как низкой, так и высокой степени детализации (например, бумажные прототипы или интерактивные макеты).

Визуальная иерархия:

- Организация информации таким образом, чтобы пользователи могли легко находить и воспринимать данные.

- Использование размеров шрифтов, цветов и пробелов для создания акцентов.

-Цветовая палитра и типография:

- Важность выбора цветовой схемы и шрифтов для создания атмосферы и повышения читабельности.

- Правила цветовой гармонии и контраста для улучшения восприятия информации.

Современные тренды в UI/UX-дизайне постоянно эволюционируют, и дизайнеры должны быть в курсе новейших технологий и подходов. Изучение этих трендов и принципов проектирования позволяет создавать удобные и привлекательные интерфейсы, которые удовлетворяют потребности пользователей и способствуют успеху продукта.

2. Варианты для выполнения индивидуальных заданий

1. **Анализ минималистичных интерфейсов**: Исследуйте три популярных приложения с минималистичным дизайном и подготовьте отчет о их особенностях и преимуществах.
2. **Прототип темного режима**: Создайте прототип приложения с темным режимом, используя инструменты для прототипирования (Figma, Adobe XD).
3. **Анимация кнопок**: Разработайте анимацию для кнопки в мобильном приложении, демонстрируя микровзаимодействия.
4. **Голосовой интерфейс**: Создайте концепцию интерфейса для голосового помощника, описывая его функциональные возможности и сценарии использования.
5. **Микровзаимодействия в веб-дизайне**: Проанализируйте примеры микровзаимодействий на веб-сайтах и создайте свое собственное для интерактивного элемента.

6. **Доступность интерфейсов**: Проведите исследование по доступности популярных приложений для пользователей с ограниченными возможностями и предложите улучшения.
7. **Визуальная иерархия**: Создайте макет главной страницы интернет-магазина, акцентируя внимание на визуальной иерархии информации.
8. **Форма обратной связи**: Разработайте дизайн формы обратной связи с учетом принципов удобства и простоты.
9. **Адаптивный дизайн**: Исследуйте адаптивные веб-сайты и создайте концепцию страницы, которая будет оптимально выглядеть на мобильных и десктопных устройствах.
10. **Приложение для задач**: Создайте концепцию интерфейса приложения для планирования задач с использованием современных трендов.
11. **Образовательное приложение**: Разработайте прототип интерфейса для приложения, которое помогает пользователям учиться и развиваться.
12. **Цветовая палитра**: Проанализируйте цветовые палитры трех популярных приложений и создайте свою, основанную на современных трендах.
13. **Интерфейс дополненной реальности**: Создайте концепцию интерфейса для приложения с функцией дополненной реальности.
14. **Иллюстрации в дизайне**: Исследуйте использование иллюстраций в интерфейсах и создайте свой макет, применяя этот подход.
15. **Главная страница новостного сайта**: Разработайте макет главной страницы новостного сайта, фокусируясь на пользовательском опыте и навигации.
16. **Фитнес-приложение**: Создайте концепцию интерфейса для фитнес-приложения с учетом современных дизайнерских решений.
17. **Навигация в приложении**: Проанализируйте способы навигации в трех популярных приложениях и предложите улучшения для одного из них.
18. **Приложение для онлайн-курсов**: Создайте прототип интерфейса для платформы онлайн-курсов, акцентируя внимание на взаимодействии пользователей.
19. **Система управления задачами**: Разработайте интерфейс для системы управления задачами с использованием канбан-доски.

20. **Типографика**: Исследуйте применение шрифтов в популярных приложениях и предложите свою концепцию выбора типографики.
21. **Приложение для путешествий**: Создайте концепцию интерфейса для мобильного приложения, предназначенного для планирования путешествий.
22. **Интерфейс для личных финансов**: Разработайте макет интерфейса приложения для отслеживания личных финансов с акцентом на удобство.
23. **Анализ отзывов пользователей**: Проведите анализ пользовательских отзывов о популярных интерфейсах и выделите основные проблемы и предложения.
24. **Социальная сеть**: Создайте концепцию интерфейса для новой социальной сети, учитывая современные тренды и лучшие практики.
25. **Приложение для организации мероприятий**: Разработайте макет интерфейса для приложения, помогающего пользователям организовывать мероприятия.
26. **Визуальные метафоры**: Исследуйте применение визуальных метафор в интерфейсах и создайте свою концепцию для конкретного приложения.
27. **Макет интернет-магазина**: Создайте дизайн интерфейса для интернет-магазина, акцентируя внимание на опыте пользователя.
28. **Приложение для кулинарии**: Разработайте концепцию интерфейса для приложения, посвященного кулинарии, с учетом взаимодействия и функциональности.
29. **Анализ пользовательского опыта**: Проведите исследование пользовательского опыта (UX) для выбранного вами приложения и представьте результаты.
30. **Виртуальная реальность**: Создайте концепцию интерфейса для приложения, использующего технологии виртуальной реальности, и опишите его функционал.

Контрольные вопросы:

1. Что такое UI и UX, и как они взаимосвязаны?
2. Каковы ключевые принципы минималистичного дизайна в UI/UX?
3. Что такое микровзаимодействия и как они влияют на пользовательский опыт?
4. Какие преимущества и недостатки имеет темный режим в приложениях?
5. Почему адаптивный и отзывчивый дизайн важны для современных веб-приложений?

6. Как доступность влияет на проектирование интерфейсов?
7. Каково значение пользовательских исследований в процессе разработки интерфейса?
8. Как визуальная иерархия помогает пользователю воспринимать информацию в интерфейсе?
9. Как использование анимации в интерфейсах может улучшить пользовательский опыт?
10. Каковы основные тренды в UI/UX-дизайне, которые ожидаются в будущем?

Лабораторная работа №16

Разработка интерфейса с учетом принципов доступности (Accessibility). Тестирование интерфейса с использованием инструментов проверки доступности

Цель работы:

1. Теоретический материал

Доступность (Accessibility) — это концепция, обеспечивающая равный доступ к информации и услугам для всех пользователей, включая людей с ограниченными возможностями. Основная цель доступности — убрать барьеры, которые могут помешать пользователям взаимодействовать с продуктами и услугами.

Доступность означает, что веб-сайты, приложения и другие интерфейсы должны быть спроектированы так, чтобы их могли использовать все, независимо от физических или когнитивных возможностей. Это включает в себя использование правильных технологий, стандартов и методов дизайна.

В соответствии с Web Content Accessibility Guidelines (WCAG) существуют четыре основных принципа доступности, известные как POUR:

Определение: Информация и компоненты интерфейса должны быть представлены так, чтобы их можно было воспринимать всеми пользователями.

Рекомендации:

Используйте текстовые альтернативы для изображений (например, атрибут alt).

Обеспечьте доступность аудио и видео через субтитры и транскрипции.

Используйте контрастные цвета для текста и фона, чтобы текст был легко читаем.

Определение: Все пользователи должны иметь возможность управлять интерфейсом.

Рекомендации:

Все функции должны быть доступны с клавиатуры (например, с помощью клавиши Tab).

Избегайте использования всплывающих окон, которые могут мешать навигации.

Обеспечьте достаточное время для выполнения действий (например, не ограничивайте время для заполнения форм).

Понимаемость (Understandable)

Определение: Информация и интерфейсы должны быть понятными и предсказуемыми.

Рекомендации:

Используйте простой и ясный язык.

Обеспечьте последовательность в навигации и оформлении страниц.

Предоставляйте подсказки и инструкции для заполнения форм.

Сопротивляемость (Robust)

Определение: Контент должен быть достаточно устойчивым, чтобы его могли интерпретировать различные пользователи и технологии.

Рекомендации:

Используйте семантический HTML для структурирования контента.

Обеспечьте совместимость с различными устройствами и программами, включая экранные чтецы.

Инструменты проверки доступности

Существует множество инструментов, которые помогают тестировать доступность интерфейсов:

WAVE (Web Accessibility Evaluation Tool)

Описание: Инструмент для анализа доступности веб-страниц, который предоставляет визуальные представления ошибок и предупреждений.

Преимущества:

Интуитивно понятный интерфейс.

Визуализация проблем непосредственно на странице.

axe

Описание: Библиотека для тестирования доступности, интегрируемая с различными фреймворками, такими как React, Vue и Angular.

Преимущества:

Автоматизированное тестирование.

Подробные отчеты с рекомендациями по исправлению ошибок.

Lighthouse

Описание: Инструмент от Google для оценки производительности, доступности и SEO веб-приложений.

Преимущества:

Подробные отчеты с оценкой доступности.

Интеграция с Chrome DevTools.

Актуальные стандарты и руководства по доступности

WCAG (Web Content Accessibility Guidelines): Набор рекомендаций, разработанный W3C для улучшения доступности веб-контента. Последняя версия, WCAG 2.1, включает дополнительные рекомендации для мобильных устройств и пользователей с когнитивными нарушениями.

Section 508: Законодательство США, требующее доступности информационных технологий для федеральных учреждений.

ADA (Americans with Disabilities Act): Законодательство, обеспечивающее равные права для людей с ограниченными возможностями в различных сферах, включая веб-доступность.

Разработка доступных интерфейсов — это важный аспект создания пользовательского опыта. Обеспечение доступности помогает не только соблюсти юридические требования, но и расширить аудиторию вашего продукта. Каждая деталь дизайна, от выбора цветов до структуры контента, должна быть продумана с учетом потребностей всех пользователей.

2. Варианты заданий

1. Создать веб-страницу с текстовой альтернативой для всех изображений.

Добавьте атрибут alt ко всем изображениями на странице.

2. Использовать контрастные цвета для текста и фона на веб-странице.

Проверьте, чтобы соотношение контраста между текстом и фоном соответствовало стандартам WCAG.

3. Создать навигационное меню, доступное с клавиатуры.

Убедитесь, что все элементы меню могут быть достигнуты с помощью клавиш Tab и Enter.

4. Разработать форму с метками для каждого поля ввода.

Используйте элементы <label> для всех полей ввода формы.

5. Обеспечить доступность кнопок с помощью атрибутов ARIA.

Примените атрибуты role и aria-label для всех интерактивных элементов.

6. Добавить опцию увеличения шрифта на странице.

Реализуйте кнопку, которая будет изменять размер шрифта на странице.

7. Использовать заголовки (h1, h2, h3 и т.д.) для структуры контента.

Структурируйте контент с помощью семантических заголовков.

8. Создать адаптивный дизайн, который хорошо отображается на мобильных устройствах.

Используйте медиа-запросы для настройки стилей для мобильных устройств.

9. Реализовать звуковые оповещения с текстовыми альтернативами.

Добавьте текстовые уведомления для всех звуковых сигналов на странице.

10. Провести тестирование доступности с помощью WAVE и исправить найденные ошибки.

Используйте WAVE для анализа страницы и внесите исправления.

11. Создать таблицу с понятным и доступным оформлением.

Используйте атрибуты <caption>, <th> и scope для описания таблицы.

12. Использовать семантические HTML-элементы для улучшения доступности.

Примените элементы <article>, <aside>, <nav> и другие семантические теги.

13. Добавить атрибуты tabindex для управления фокусом.

Убедитесь, что порядок фокуса логически следует за порядком, в котором элементы отображаются на странице.

14. Обеспечить доступность мультимедийного контента с помощью субтитров.

Добавьте субтитры для видео и предоставьте текстовые транскрипции.

15. Использовать всплывающие подсказки с доступной информацией.

Реализуйте всплывающие подсказки, которые будут доступны при наведении курсора.

16. Провести тестирование доступности с помощью axe и исправить найденные ошибки.

Используйте axe для проверки страницы и внесите необходимые изменения.

17. Создать страницу с интерактивными элементами, доступными с клавиатуры.

Реализуйте кнопки, переключатели и другие элементы, которые можно использовать без мыши.

18. Добавить возможность смены языка интерфейса.

Реализуйте функцию, которая позволяет пользователю выбрать язык интерфейса.

19. Использовать атрибуты для определения ролей элементов (role) в интерфейсе.

Примените атрибут role для всех компонентов, чтобы указать их назначение.

20. Разработать уведомления, которые не блокируют доступ к интерфейсу.

Убедитесь, что уведомления могут быть легко закрыты и не мешают навигации.

21. Обеспечить доступность для пользователей с проблемами слуха через текстовые альтернативы.

Добавьте текстовые версии всех звуковых уведомлений и описаний.

22. Добавить простое и интуитивно понятное описание для каждого элемента формы.

Обеспечьте ясные подсказки для заполнения форм.

23. Проверить страницу на доступность с помощью Lighthouse и внести изменения.

Используйте Lighthouse для анализа доступности и исправьте выявленные проблемы.

24. Создать интерфейс с четким обозначением состояния (активное/неактивное).

Убедитесь, что пользователи могут видеть, какие элементы активны, а какие — нет.

25. Разработать меню с ясной и доступной навигацией.

Создайте навигационное меню с логическим и понятным структурированием.

26. Добавить возможность изменения темы оформления (светлая/темная).

Реализуйте переключатель, который изменяет тему страницы.

27. Создать элементы управления, доступные для пользователей с ограниченной подвижностью.

Используйте большие кнопки и учитывайте различные методы ввода.

28. Разработать инструкции для заполнения формы, доступные через экранный чтец.

Обеспечьте доступные подсказки для экранных чтецов, используя атрибуты ARIA.

29. Создать прототип интерфейса, соответствующий принципам доступности.

Проектируйте интерфейс, учитывая все перечисленные принципы доступности.

Контрольные вопросы:

1. Что такое доступность (Accessibility) и почему она важна?
2. Назовите и опишите четыре принципа доступности, согласно WCAG.
3. Что подразумевается под перцептивностью в доступности?
4. Каковы основные рекомендации для обеспечения управляемости интерфейса?
5. Почему важно использовать семантические HTML-элементы?

СПИСОК ЛИТЕРАТУРЫ

1. Кингсли-Хью, Э. JavaScript в примерах / Кингсли-Хью Э. , Кингсли-Хью К. - М : ДМК Пресс. URL : https://www.studentlibrary.ru/book/ISBN9785940746683.html (дата обращения: 10.03.2025).
2. Основы JavaScript / - М : Национальный Открытый Университет "ИНТУИТ". URL : https://www.studentlibrary.ru/book/intuit_175.html (дата обращения: 10.03.2025).
3. Хорстман, К. С. Современный JavaScript для нетерпеливых / К. С. Хорстман – М : ДМК Пресс. URL : https://www.studentlibrary.ru/book/ISBN9785970601778.html (дата обращения: 10.03.2025).
4. Баланов, А. Н. Комплексное руководство по разработке: от мобильных приложений до веб-технологий : учебное пособие для вузов / А. Н. Баланов. — СПб: Лань. URL: https://e.lanbook.com/book/394577 (дата обращения: 10.03.2025).
5. Янцев, В. В. Разработка web-страниц на HTML, CSS и JavaScript : учебное пособие для вузов / В. В. Янцев. — СПб : Лань. URL: https://e.lanbook.com/book/422462 (дата обращения: 10.03.2025).
6. Баланов, А. Н. Прототипирование и разработка пользовательского интерфейса: оптимизация UX : учебное пособие для вузов / А. Н. Баланов. — СПб : Лань. URL: https://e.lanbook.com/book/414929 (дата обращения: 10.03.2025).
7. Ковешникова, Н. А. История дизайна. Краткий курс лекций : учебное пособие для вузов / Н. А. Ковешникова. — 3-е изд., стер. — СПб: Лань. URL: https://e.lanbook.com/book/394688 (дата обращения: 10.03.2025).