Частное образовательное учреждение высшего образования
«Международный Институт Дизайна и Сервиса»
(ЧОУВО МИДиС)
Кафедра математики и информатики

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

ПРАКТИКУМ ПО ПРОГРАММИРОВАНИЮ

Направление подготовки: 38.03.05 Бизнес-информатика
Профиль подготовки: Управление IT-проектами
Квалификация выпускника: бакалавр
Год набора: 2025

Автор-составитель: Мухина Ю.Р.

Челябинск 2025

СОДЕРЖАНИЕ

1. ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ С УКАЗАНИЕМ ЭТАПОВ ИХ ФОРМИРОВАНИЯ В ПРОЦЕССЕ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Процесс изучения дисциплины «Практикум по программированию» направлен на формирование следующих компетенций:

Код и наименование компетенций выпускника	Код и наименование индикатора достижения компетенций
ПК-4. Способен разрабатывать и управлять разработкой информационных систем в соответствии с требованиями заказчика	ПК-4.1 Осуществляет деятельность по разработке и управлению разработкой прототипов информационных систем в соответствии с требованиями заказчика. ПК-4.2 Умеет кодировать на современных языках программирования информационных систем и баз данных, распределять работы и выделять ресурсы, управлять содержанием, качеством и коммуникациями в проекте по разработке информационных систем. ПК-4.3 Знает основные концепции, принципы и возможности современных технологий проектирования, разработки и верификации информационных систем

№ п/п	Код компетенции	Наименование компетенции	Этапы формирования компетенций
1	ПК-4	Способен разрабатывать и управлять разработкой информационных систем в соответствии с требованиями заказчика	*1 Этап – Знать:* ПК-4.1 - основные концепции, принципы и возможности современных технологий проектирования, разработки и верификации информационных систем;
			2 Этап – Уметь: ПК-4.2 - кодировать на современных языках программирования информационных систем и баз данных; - распределять работы и выделять ресурсы, управлять содержанием, качеством и коммуникациями в проекте по разработке информационных систем;
			3 Этап – Владеть: ПК-4.3 - навыками осуществления деятельности по разработке и управлению разработкой прототипов информационных систем в соответствии с требованиями заказчика.

2. ПОКАЗАТЕЛИ И КРИТЕРИИ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ НА РАЗЛИЧНЫХ ЭТАПАХ ИХ ФОРМИРОВАНИЯ, ОПИСАНИЕ ШКАЛ ОЦЕНИВАНИЯ

№ п/п	Код компетенции	Наименование компетенции	Критерии оценивания компетенций на различных этапах формирования	Шкала оценивания
1.	ПК-4	Способен разрабатывать и управлять разработкой информационных систем в соответствии с требованиями заказчика	*1 Этап – Знать:* ПК-4.1 - основные концепции, принципы и возможности современных технологий проектирования, разработки и верификации информационных систем; *2 Этап – Уметь:* ПК-4.2 - кодировать на современных языках программирования информационных систем и баз данных; - распределять работы и выделять ресурсы, управлять содержанием, качеством и коммуникациями в проекте по разработке информационных систем; *3 Этап – Владеть:* ПК-4.3 - навыками осуществления деятельности по разработке и управлению разработкой прототипов информационных систем в соответствии с требованиями заказчика.	***Зачет с оценкой*** «ОТЛИЧНО» 1. Глубокое и прочное усвоение программного материала. 2. Знание пакетов прикладных программ. 3. Знание основных принципов построения пакетов прикладных программ. 4. Знание основных задач прикладных программ. 5. Свободное владение пакетами прикладных программ. 6. Точность и обоснованность выводов. 7. Безошибочное выполнение практического задания. 8. Точные, полные и логичные ответы на дополнительные вопросы. «ХОРОШО» 1. Хорошее знание программного материала. 2. Недостаточно полное изложение теоретического вопроса экзаменационного билета. 3. Наличие незначительных неточностей в употреблении терминов, классификаций. 4. Знание основных пакетов прикладных программ. 5. Неполнота представленного иллюстративного материала. 6. Точность и обоснованность выводов. 7. Логичное изложение вопроса, соответствие изложения научному стилю. 8. Негрубая ошибка при выполнении практического задания. 9. Правильные ответы на дополнительные вопросы. «УДОВЛЕТВОРИТЕЛЬНО» 1. Поверхностное усвоение программного материала. 2. Недостаточно полное изложение теоретического вопроса экзаменационного билета. 3. Затруднение в приведении приме-

				ров, подтверждающих теоретические положения. 4. Наличие неточностей в употреблении терминов, классификаций. 5. Неумение четко сформулировать выводы. 6. Отсутствие навыков научного стиля изложения. 7. Грубая ошибка в практическом задании. 8. Неточные ответы на дополнительные вопросы. «НЕУДОВЛЕТВОРИТЕЛЬНО» 1. Незнание значительной части программного материала. 2. Неспособность привести примеры пакетов прикладных программ 3. Неумение выделить главное, сделать выводы и обобщения. 4. Грубые ошибки при выполнении практического задания. 5. Неправильные ответы на дополнительные вопросы. ***Зачет*** «ЗАЧТЕНО»: 1. Усвоение программного материала. 2. Знание сущности основных категорий и понятий. 3. Выполнение самостоятельной работы за семестр. 4. Точность и обоснованность выводов. 5. Точные, полные и логичные ответы на дополнительные вопросы. «НЕ ЗАЧТЕНО»: 1. Незнание значительной части программного материала 2. Невыполнение самостоятельной работы за семестр. 3. Грубые ошибки при выполнении самостоятельной работы. 4. Неумение выделить главное, сделать выводы и обобщения. 5. Неправильные ответы на дополнительные вопросы.

3. ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ, НЕОБХОДИМЫЕ ДЛЯ ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ И (ИЛИ) ОПЫТА ДЕЯТЕЛЬНОСТИ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ В ПРОЦЕССЕ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

1 ЭТАП – ЗНАТЬ

Типовой тестовый материал

1. Какой протокол используется в Интернете для передачи информации между компьютерами?

a) SMTP
b) HTTP
c) FTP
d) TCP/IP

Правильный ответ: b) HTTP

2. Как называется язык разметки, используемый для создания веб-страниц?

a) HTML
b) CSS
c) JavaScript
d) PHP

Правильный ответ: a) HTML

3. Какие протоколы используются в Интернете для передачи данных?

a) SMTP и POP3
b) TCP/IP и UDP
c) HTTP и HTTPS
d) FTP и Telnet

Правильный ответ: b) TCP/IP и UDP

4. Какой формат используется для сохранения графических изображений в веб?

a) JPEG
b) MP3
c) PDF
d) DOC

Правильный ответ: a) JPEG

5. Как называется язык стилей, используемый для оформления веб-страниц?

a) HTML
b) CSS
c) JavaScript
d) PHP

Правильный ответ: b) CSS

6. Какой селектор используется для выбора элементов с определенным классом?

a) .class
b) #id

c) *
d) div
Правильный ответ: a) .class

7. Какой селектор используется для выбора элементов с определенным идентификатором?
a) .class
b) #id
c) *
d) div
Правильный ответ: b) #id

8. Какой селектор выбирает все элементы на веб-странице?
a) .class
b) #id
c) *
d) div
Правильный ответ: c) *

9. Какой селектор используется для группировки других селекторов?
a) .class
b) #id
c) *
d) div
Правильный ответ: c) *

10. Какой селектор используется для выбора дочерних элементов определенного родительского элемента?
a) .class
b) #id
c) >
d) div
Правильный ответ: c) >

11. Как называется метод передачи стилей от родительского элемента к дочерним элементам?
a) наследование стилей
b) каскадирование стилей
c) фоны
d) интерактивное меню
Правильный ответ: a) наследование стилей

12. Какой тег используется для создания таблицы на веб-странице?
a) <table>
b) <form>
c) <input />

d) <select>
Правильный ответ: a)

13. Какой атрибут используется для объединения ячеек в таблице?
a) colspan
b) rowspan
c) width
d) height
Правильный ответ: a) colspan

14. Какой формат изображений используется для оформления фона на веб-странице?
a) JPEG
b) GIF
c) PNG
d) BMP
Правильный ответ: b) GIF

15. Что такое модульная сетка?
a) набор готовых блоков для построения веб-страницы
b) инструмент для создания адаптивного дизайна
c) система разметки, основанная на сетке из модулей
d) методика организации цветовой схемы в проекте
Ответ: c) система разметки, основанная на сетке из модулей

16. Какие правила работы с модульными сетками существуют?
a) соблюдать пропорции между модулями и блоками
b) использовать указанные размеры модулей
c) избегать пересечения модулей
d) все вышеперечисленное
Ответ: d) все вышеперечисленное

17. Какие существуют способы построения модульных сеток?
a) использование готовых шаблонов
b) ручное создание с помощью инструментов редактора
c) использование CSS-фреймворков
d) все вышеперечисленное
Ответ: d) все вышеперечисленное

18. Что такое Flexbox?
a) методика верстки с использованием гибкого контейнера и элементов внутри него
b) фреймворк для разработки адаптивных интерфейсов
c) язык программирования для создания анимаций
d) инструмент для создания модульных сеток
Ответ: a) методика верстки с использованием гибкого контейнера и элементов внутри него

19. Что такое Grid?
a) система разметки, основанная на сетке из модулей
b) методика верстки с использованием гибкого контейнера и элементов внутри него
c) фреймворк для разработки адаптивных интерфейсов
d) инструмент для создания модульных сеток
Ответ: a) система разметки, основанная на сетке из модулей

20. Что такое Bootstrap?
a) фреймворк для разработки адаптивных интерфейсов
b) методика верстки с использованием гибкого контейнера и элементов внутри него
c) инструмент для создания модульных сеток
d) язык программирования для создания анимаций
Ответ: a) фреймворк для разработки адаптивных интерфейсов

21. Какие преимущества имеет Bootstrap?
a) большое количество готовых компонентов и стилей
b) удобная сетка для адаптивной верстки
c) возможность быстрого создания прототипов
d) все вышеперечисленное
Ответ: d) все вышеперечисленное

22. Какие недостатки имеет Bootstrap?
a) ограниченные возможности для кастомизации
b) большой объем файлов
c) зависимость от внешних библиотек
d) все вышеперечисленное
Ответ: d) все вышеперечисленное

23. Зачем используются HTML- и CSS-шаблоны оформления?
a) для упрощения процесса создания веб-интерфейса
b) для форматирования текстовых элементов
c) для создания анимаций
d) для разработки серверной части веб-приложения
Ответ: a) для упрощения процесса создания веб-интерфейса

24. Что такое JavaScript-расширения?
a) библиотеки, расширяющие возможности JavaScript
b) специальные инструменты для отладки JavaScript-кода
c) методы для расширения возможностей браузера
d) инструменты для тестирования JavaScript-кода
Ответ: a) библиотеки, расширяющие возможности JavaScript

25. Что такое адаптивная верстка?
a) методика верстки, при которой элементы страницы подстраиваются под размер экрана
b) методика верстки, при которой используется гибкая сетка

c) методика верстки, при которой элементы скрываются на мобильных устройствах
d) методика верстки, при которой используется Grid
Ответ: a) методика верстки, при которой элементы страницы подстраиваются под размер экрана

26. В чем разница между адаптивной и мобильной версией сайта?
a) адаптивная версия подстраивается под размер экрана, мобильная версия создается специально для мобильных устройств
b) мобильная версия использует гибкую сетку, а адаптивная - фиксированную
c) адаптивная версия доступна только на мобильных устройствах, а мобильная - на всех устройствах
d) адаптивная версия использует Grid, а мобильная - Flexbox
Ответ: a) адаптивная версия подстраивается под размер экрана, мобильная версия создается специально для мобильных устройств

27. Какие достоинства имеет адаптивная верстка?
a) лучшая поддержка разных устройств
b) улучшение пользовательского опыта
c) большая гибкость и удобство в разработке
d) все вышеперечисленное
Ответ: d) все вышеперечисленное

28. Какие недостатки имеет адаптивная верстка?
a) сложность в разработке и поддержке
b) больший объем кода и файлов
c) возможные проблемы с производительностью
d) все вышеперечисленное
Ответ: d) все вышеперечисленное

29. Какие декоративные эффекты можно использовать на веб-странице?
a) тени
b) углы
c) градиенты
d) все вышеперечисленное
Ответ: d) все вышеперечисленное

30. Какие элементы можно стилизовать с помощью декоративных эффектов?
a) ссылки
b) кнопки
c) метки
d) все вышеперечисленное
Ответ: d) все вышеперечисленное

31. Что такое формат SVG?
a) формат изображений, основанных на векторной графике
b) формат таблиц для оформления веб-страниц

c) формат архивов для хранения данных
d) формат данных для передачи аудио и видео
Ответ: a) формат изображений, основанных на векторной графике

32. Что такое CSS-трансформации?
a) способ изменения формы, позиции и размера элементов с помощью CSS
b) фреймворк для разработки адаптивных интерфейсов
c) язык программирования для создания анимаций
d) система разметки, основанная на сетке из модулей
Ответ: a) способ изменения формы, позиции и размера элементов с помощью CSS

33. Что такое CSS-переходы?
a) эффекты плавного перехода между различными состояниями элемента
b) методика верстки с использованием гибкого контейнера и элементов внутри него
c) инструмент для создания модульных сеток
d) фреймворк для разработки адаптивных интерфейсов
Ответ: a) эффекты плавного перехода между различными состояниями элемента

34. Что такое CSS-анимация?
a) способ создания движущихся элементов на веб-странице с помощью CSS
b) методика верстки, при которой элементы скрываются на мобильных устройствах
c) методика верстки, при которой используется гибкая сетка
d) фреймворк для разработки адаптивных интерфейсов
Ответ: a) способ создания движущихся элементов на веб-странице с помощью CSS

35. Какое назначение и применение у JavaScript?
a) добавление стилей на веб-страницу
b) управление поведением веб-страницы
c) создание баз данных
d) отображение изображений на веб-странице
Правильный ответ: b

36. Какие способы внедрения JavaScript-кода в HTML-страницу вы знаете?
a) внедрение кода внутри тега <script></script>
b) внедрение кода внутри тега <style></style>
c) внедрение кода внутри тега <head></head>
d) внедрение кода внутри тега <body></body>
Правильный ответ: a

37. Какие операторы могут быть использованы в JavaScript?
a) +, -, *, /
b) =, <, >
c) &&, ||
d) все вышеперечисленные
Правильный ответ: d

38. Как объявить переменную в JavaScript?
a) var myVariable;
b) int myVariable;
c) variable myVariable;
d) let myVariable;
Правильный ответ: d

39. Что такое константа в JavaScript?
a) это переменная, значение которой не может быть изменено
b) это функция, которая выполняет определенное действие
c) это оператор, используемый для сравнения значений
d) константы отсутствуют в JavaScript
Правильный ответ: a

40. Какие типы циклов существуют в JavaScript?
a) for, while, do...while
b) loop, repeat, until
c) cycle, iterate, do...until
d) типы циклов отсутствуют в JavaScript
Правильный ответ: a

41. Что такое массив в JavaScript?
a) упорядоченная коллекция элементов
b) функция, принимающая аргументы и возвращающая значение
c) специальный символ, обозначающий комментарий
d) оператор, используемый для объединения строк
Правильный ответ: a

42. Как преобразовать строку в число в JavaScript?
a) parseInt()
b) toInteger()
c) parseFloat()
d) toNumber()
Правильный ответ: a

43. Как получить длину строки в JavaScript?
a) lengthOf()
b) stringLength()
c) size()
d) length
Правильный ответ: d

44. Что такое объектная модель документа DOM?
a) язык программирования для создания веб-страниц
b) среда разработки для JavaScript
c) стандарт, определяющий структуру веб-документа и способы взаимодействия с ним

d) оператор, используемый для объединения элементов массива
Правильный ответ: c

45. Какие типы данных существуют в JavaScript?
a) Number, String, Boolean, Object, Array
b) Int, Float, Char, Boolean, List
c) Digit, Text, Logical, Data, Set
d) JavaScript не имеет типов данных
Правильный ответ: a

45. Какими объектами можно манипулировать в DOM?
a) HTML-элементы
b) файлы на сервере
c) данные в базе данных
d) операционная система
Правильный ответ: a

46. Что такое JSON?
a) язык разметки для создания веб-страниц
b) формат данных, основанный на JavaScript-синтаксисе
c) технология для создания анимации
d) методика шифрования данных в сети
Правильный ответ: b

47. Как объявить функцию в JavaScript?
a) function myFunction() {}
b) def myFunction() {}
c) var myFunction() {}
d) let myFunction() {}
Правильный ответ: a

48. Каким событием можно вызвать выполнение функции в JavaScript?
a) onClick
b) onMove
c) onChange
d) onScroll
Правильный ответ: a

49. Каким образом можно хранить данные в JavaScript?
a) в переменных
b) в файлах на сервере
c) в базе данных
d) все вышеперечисленные
Правильный ответ: a

50. Что такое формат данных JSON?
a) JavaScript Object Notation
b) Java Syntax for Objects and Numbers
c) JavaScript Ordered Notation
d) JavaScript Operative Notation
Правильный ответ: a

51. Каким образом можно обращаться к элементам DOM в JavaScript?
a) через их идентификаторы (ID) или классы
b) через теги, которыми они обозначены
c) через их текстовое содержимое
d) через их позицию на веб-странице
Правильный ответ: a

52. Что такое DOM интерфейсы?
a) библиотеки, предоставляющие дополнительные возможности работы с DOM
b) программные интерфейсы для работы с операционной системой
c) методы для взаимодействия с пользователем на веб-странице
d) DOM интерфейсы отсутствуют в JavaScript
Правильный ответ: a

53. Как создать объект в JavaScript?
a) let myObj = {};
b) object myObj = {};
c) var myObj = ();
d) create myObj = {};
Правильный ответ: a

54. Какая версия PHP считается первым официально выпущенным релизом?
a) PHP 3
b) PHP 4
c) PHP 5
d) PHP 7
Правильный ответ: a) PHP 3

55. Какие платформы поддерживает PHP?
a) Windows
b) Linux
c) macOS
d) все вышеперечисленные
Правильный ответ: d) все вышеперечисленные

56. Какие протоколы поддерживает PHP?
a) HTTP
b) FTP
c) SMTP

d) все вышеперечисленные
Правильный ответ: d) все вышеперечисленные

57. Какие базы данных поддерживаются PHP?
a) MySQL
b) PostgreSQL
c) Oracle
d) все вышеперечисленные
Правильный ответ: d) все вышеперечисленные

58. Какие приложения электронной коммерции можно разработать с помощью PHP?
a) интернет-магазин
b) система онлайн-оплаты
c) форум
d) все вышеперечисленные
Правильный ответ: d) все вышеперечисленные

59. В каких областях можно использовать PHP?
a) серверные приложения
b) командная строка
c) создание GUI приложений
d) все вышеперечисленные
Правильный ответ: d) все вышеперечисленные

60. Какую версию PHP следует установить для разработки современных проектов?
a) PHP 4
b) PHP 5
c) PHP 7
d) PHP 8
Правильный ответ: d) PHP 8

61. Какие программы необходимо установить для работы с PHP?
a) веб-сервер (например, Apache)
b) PHP-интерпретатор
c) редактор кода (например, Visual Studio Code)
d) все вышеперечисленные
Правильный ответ: d) Все вышеперечисленные

62. Как создать однострочный комментарий в PHP?
a) /* комментарий */
b) // комментарий
c)
Правильный ответ: b) // комментарий

63. Что такое переменная в PHP?
a) это постоянное значение

b) это контейнер для хранения данных
c) это тип данных
d) это функция в PHP
Правильный ответ: b) это контейнер для хранения данных

64. Как объявить константу в PHP?
a) define("CONSTANT_NAME", value);
b) var CONSTANT_NAME = value;
c) const CONSTANT_NAME = value;
d) set constant CONSTANT_NAME = value;
Правильный ответ: a) define("CONSTANT_NAME", value);

65. Какие типы данных поддерживает PHP?
a) числа
b) строки
c) булевы значения (true / false)
d) все вышеперечисленные
Правильный ответ: d) все вышеперечисленные

66. Какие операторы поддерживает PHP?
a) математические операторы (+, -, *, /)
b) операторы сравнения (==, !=, >, <)
c) логические операторы (&&, ||, !)
d) все вышеперечисленные
Правильный ответ: d) все вышеперечисленные

67. Как обработать POST-запрос с помощью PHP?
a) использовать массив $_POST
b) использовать массив $_GET
c) использовать функцию curl()
d) все вышеперечисленные
Правильный ответ: a) использовать массив $_POST

68. Что такое методы POST и GET?
a) это способы отправки данных с помощью HTTP-протокола
b) это способы обработки данных на сервере
c) это функции PHP для работы с базами данных
d) ЭТО команды командной строки в PHP
Правильный ответ: a) это способы отправки данных с помощью HTTP-протокола

69. Как получить данные из HTML-формы и обработать их с помощью PHP?
a) использовать функцию read_form_data()
b) использовать функцию process_form_data()
c) использовать массив $_POST или $_GET
d) все вышеперечисленные
Правильный ответ: c) использовать массив $_POST или $_GET

70. Какие управляющие конструкции поддерживает PHP?
a) условный оператор if
b) оператор switch
c) циклы while, for, foreach
d) все вышеперечисленные
Правильный ответ: d) все вышеперечисленные

71. Какой оператор используется для создания условной конструкции в PHP?
a) if
b) else
c) elseif
d) все вышеперечисленные
Правильный ответ: d) все вышеперечисленные

72. Какой цикл используется для повторения блока кода в PHP до тех пор, пока условие истинно?
a) while
b) for
c) foreach
d) все вышеперечисленные
Правильный ответ: a) while

73. Что такое массив в PHP?
a) это переменная, которая содержит только одно значение
b) это переменная, которая содержит значения разных типов данных
c) это упорядоченный набор значений
d) это функция для обработки данных в PHP
Правильный ответ: c) это упорядоченный набор значений

74. Что такое класс в ООП?
a) методы и данные объекта.
b) объект, созданный на основе класса.
c) функции, используемые для создания объекта.
d) структура данных, описывающая объект.

75. Что такое объект в ООП?
a) описание класса.
b) программный код, выполняющий определенные действия.
c) экземпляр класса.
d) структура данных, хранящая информацию о классе.

76. Какой ключевой слово используется для ссылки на текущий объект внутри класса?
a) this
b) self
c) object

d) current

77. Что такое конструктор класса?
a) метод, вызываемый при создании объекта.
b) метод, вызываемый при удалении объекта.
c) метод, изменяющий состояние объекта.
d) метод, выполняющий операции с данными класса.

78. Что такое деструктор класса?
a) метод, вызываемый при создании объекта.
b) метод, вызываемый при удалении объекта.
c) метод, изменяющий состояние объекта.
d) метод, выполняющий операции с данными класса.

79. Что такое открытые методы и данные в классе?
a) методы и данные, доступные только внутри класса.
b) методы и данные, доступные внутри класса и его потомков.
c) методы и данные, доступные из любого места программы.
d) методы и данные, доступные только объекту класса.

80. Что такое закрытые методы и данные в классе?
a) методы и данные, доступные только внутри класса.
b) методы и данные, доступные внутри класса и его потомков.
c) методы и данные, доступные из любого места программы.
d) методы и данные, доступные только объекту класса.

81. Что такое инкапсуляция?
a) принцип ООП, ограничивающий доступ к данным и методам класса.
b) процесс создания объекта на основе класса.
c) связь между классом и его наследниками.
d) указывает на члены класса, к которым можно обратиться извне.

82. Что такое наследование в ООП?
a) процесс создания объекта на основе класса.
b) связь между классом и его наследниками.
c) указывает на члены класса, к которым можно обратиться извне.
d) принцип ООП, ограничивающий доступ к данным и методам класса.

83. Что такое интерфейсы в ООП?
a) шаблоны классов, которые определяют набор методов.
b) связь между классом и его наследниками.
c) указывает на члены класса, к которым можно обратиться извне.
d) принцип ООП, ограничивающий доступ к данным и методам класса.

84. Что такое протокол HTTP?
a) протокол передачи данных между клиентом и сервером в интернете.

b) протокол безопасного соединения между клиентом и сервером.
c) протокол обмена сообщениями между серверами в сети.
d) протокол передачи данных по электронной почте.

85. Как получить тело запроса в протоколе HTTP?
a) используя метод GET.
b) используя метод POST.
c) из заголовков запроса.
d) из заголовков ответа.

86. Как получить заголовки запроса в протоколе HTTP?
a) используя метод GET.
b) используя метод POST.
c) из тела запроса.
d) из заголовков ответа.

87. Как добавить/изменить заголовки ответа в протоколе HTTP?
a) используя метод GET.
b) используя метод POST.
c) из тела ответа.
d) из заголовков ответа.

88. Как управлять телом ответа в протоколе HTTP?
a) изменяя заголовки ответа.
b) изменяя тело ответа.
c) используя метод GET.
d) используя метод POST.

89. Что такое формы в протоколе HTTP?
a) способ передачи данных между клиентом и сервером.
b) таблицы для оформления веб-страниц.
c) инструмент для создания клиент-серверных приложений.
d) часть заголовка запроса в протоколе HTTP.

90. Какие уязвимости могут быть связаны с использованием форм в протоколе HTTP?
a) XSS атаки.
b) SQL-инъекции.
c) CSRF атаки.
d) все вышеперечисленное.

91. Что такое MySQL в PHP базах данных?
a) основной язык программирования веб-страниц.
b) среда разработки для создания клиент-серверных приложений.
c) система управления базами данных.
d) протокол передачи данных между клиентом и сервером.

92. Как создать подключение к MySQL базе данных в PHP?
a) используя функцию mysqli_connect().
b) используя функцию mysql_connect().
c) подключение создается автоматически при использовании MySQL запросов.
d) подключение создается автоматически при включении PHP скрипта.

93. Что такое SQL запросы в MySQL?
a) язык программирования для создания клиент-серверных приложений.
b) способ передачи данных между клиентом и сервером.
c) утилита для управления базами данных.
d) язык запросов для работы с базами данных.

2 ЭТАП – УМЕТЬ

Практическая работа 1. Создание шаблона сайта с помощью flexbox

Цель работы: создать макет страницы с использованием Flexbox.
Задачи работы:
1. Изучить основные свойства flex-контейнеров и flex-элементов.
2. Научиться верстать макет веб-страниц с помощью flexbox-ов.
3. Научиться создавать вложенные flexbox-сы.

Инструменты: Visual Studio Code, браузер Google Chrome.

Задание 1. Создание шаблона страницы с помощью flexbox

1. Создайте новый проект.
2. Создайте страницу index.html.
3. Задайте html-структуру страницы (рис. 1).

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="ru">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>Flexbox</title>
8   </head>
9   <body>
10
11  </body>
12  </html>
```

Рисунок 1 – Html-структура страницы

4. Создайте в контейнере body совокупность вложенных блоков:

```
<body>
    <div class="container">
        <header>
            Шапка сайта
        </header>
        <main>
            Основная часть сайта
        </main>
        <footer>
            Подвал сайта
        </footer>
    </div>
</body>
```

В папке css разместите файл нормализации стилей (normalize.css).

1. Подключите его к странице index.html.
2. В папке css создайте файл style.css.
3. Подключите этот файл стилей к странице index.html после подключения normalize.css.
4. Задайте стили для всех блоков. Параметры цвета можно задать на свое усмотрение:

```
.container{
    background-color: aquamarine;
}

header, footer {
    height: 100px;
    background-color: aqua;

}

main {
    background-color: lightblue;
}
```

Сохраните все файлы, просмотрите результат в браузере (рис. 2). Напоминаем, что нельзя задавать стилевое правило height для блока main, так как его содержимое будет меняться и может привести к переполнению контейнера и выходу за его границы.

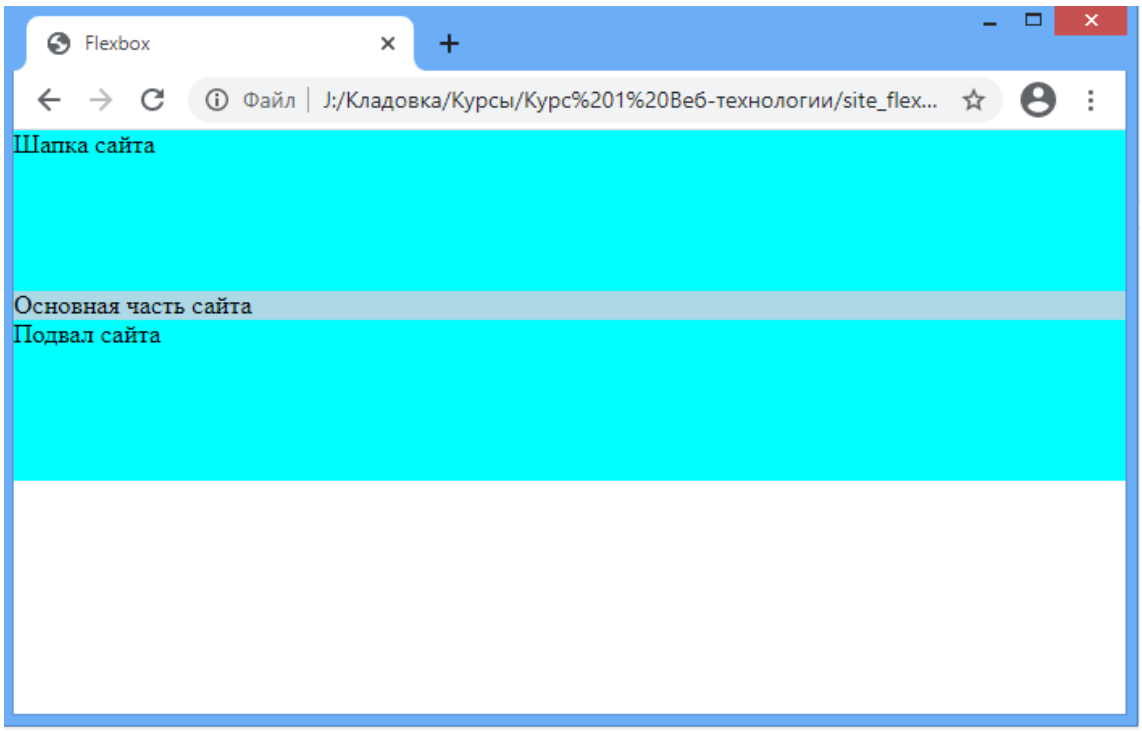


Рисунок 2 – Вид страницы в браузере

5. Сделайте контейнер с классом container flex-контейнером:

```
.container{
    background-color: aquamarine;
    display: flex;
}
```

6. Просмотрите результат (рис.3).

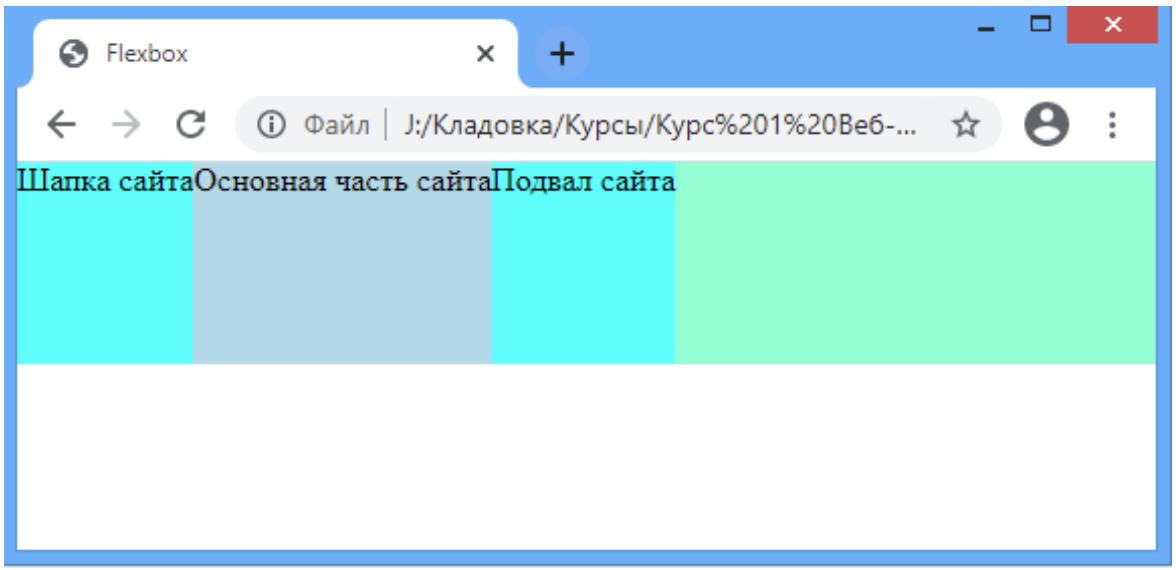


Рисунок 3 – Расположение flex-элементов по умолчанию

7. Измените направление вывода flex-элементов с помощью стилевого правила flex-direction: column; (сверху вниз). Просмотрите результат – сайт должен вернуться к исходному виду (рис. 2).

8. Растяните высоту контейнера container на всю страницу, для этого задайте стили:

```
html, body{
    height:100%;
}

.container{
    background-color: aquamarine;
    display: flex;
    flex-direction: column;
    height:100%;
    min-height: 100%;
    height: auto !important;
}
```

Просмотрите результат (рис. 4).

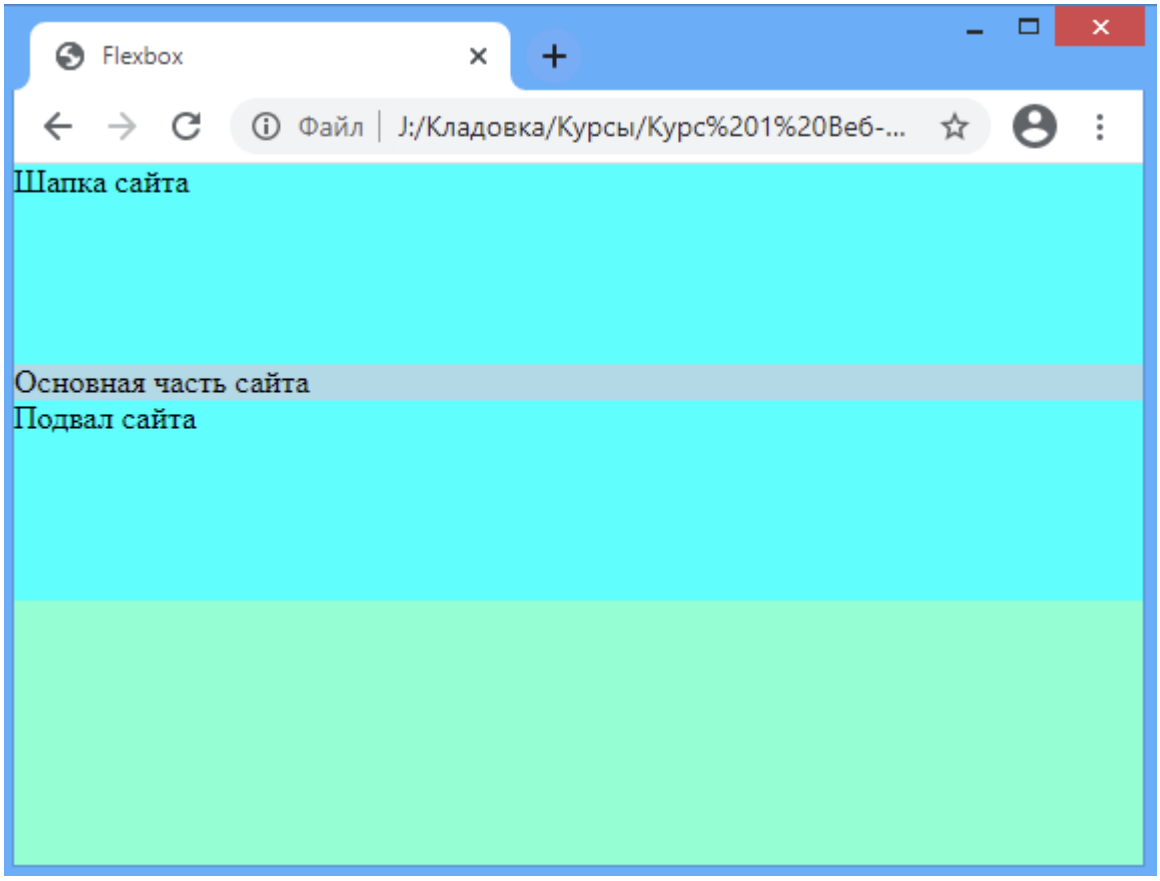


Рисунок 4 – Высота контейнера равна высоте окна браузера

9. Прижмем подвал сайта к низу окна. Для это растяните основную часть сайта (контейнер main) с помощью свойства flex-grow:

```
main {
    background-color: lightblue;
    flex-grow:1;
}
```

10. Просмотрите результат работы (рис. 5)

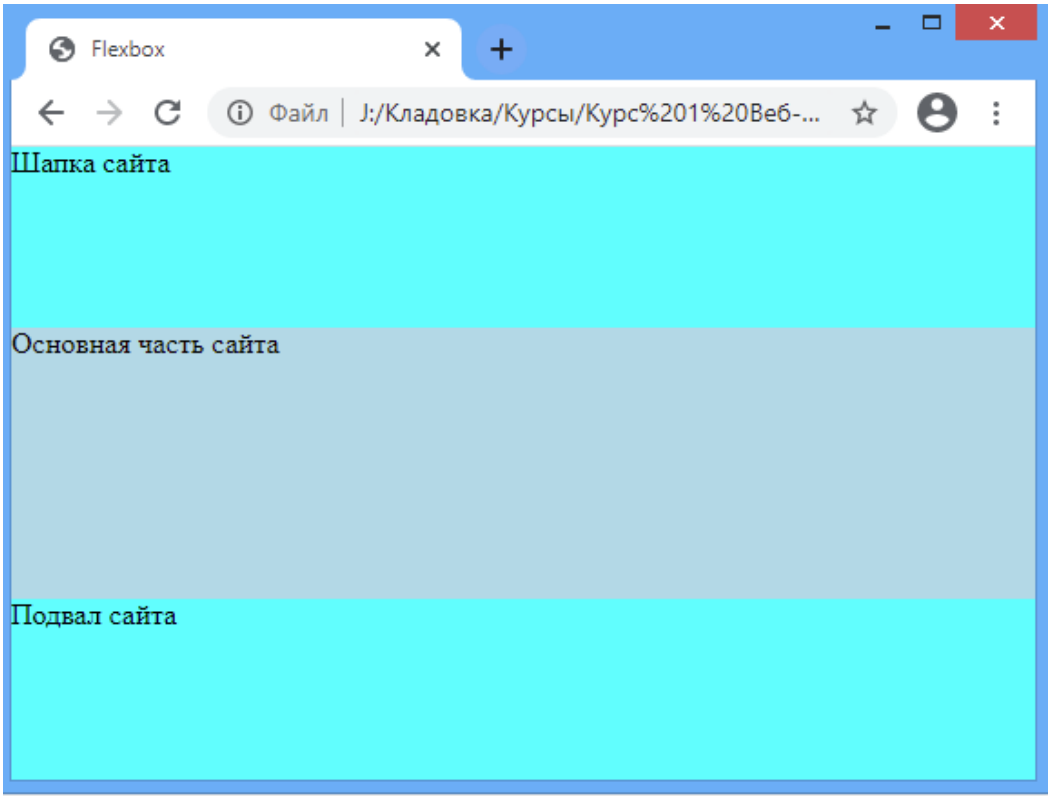


Рисунок 5 – Подвал прижат к низу окна

11. Сделайте контейнер container фиксированной ширины и центрированным по горизонтали (рис. 6). Задайте у него стилевые правила: ширина 1200px, внешние отступы сверху и снизу по 0, а справа и слева – auto.

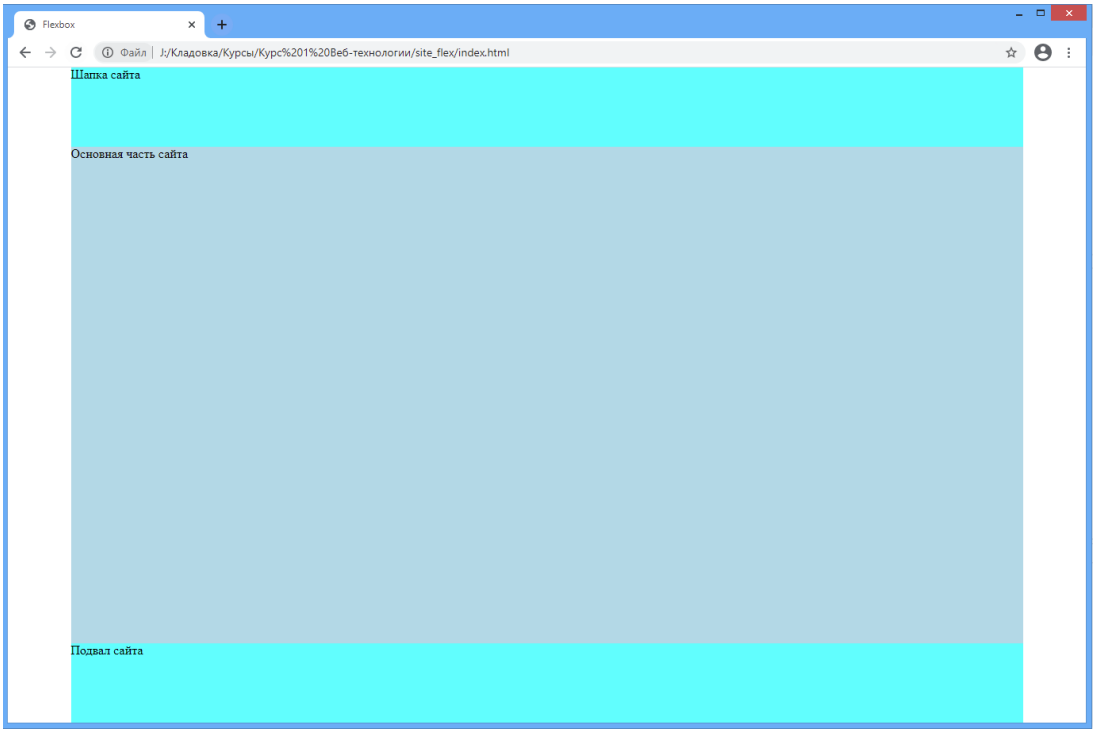


Рисунок 6 – Контейнер фиксированной ширины

Задание 2. Работа с вложенными flexbox-ами – создание sidebar

1. Разместите в контейнере main два блока: aside и div, последнему назначьте класс content.
2. Сделайте контейнер main flex-контейнером с направлением flex-direction: row.
3. Задайте у aside цвет фона и ширину в 200px.
4. Просмотрите результат (рис. 7).

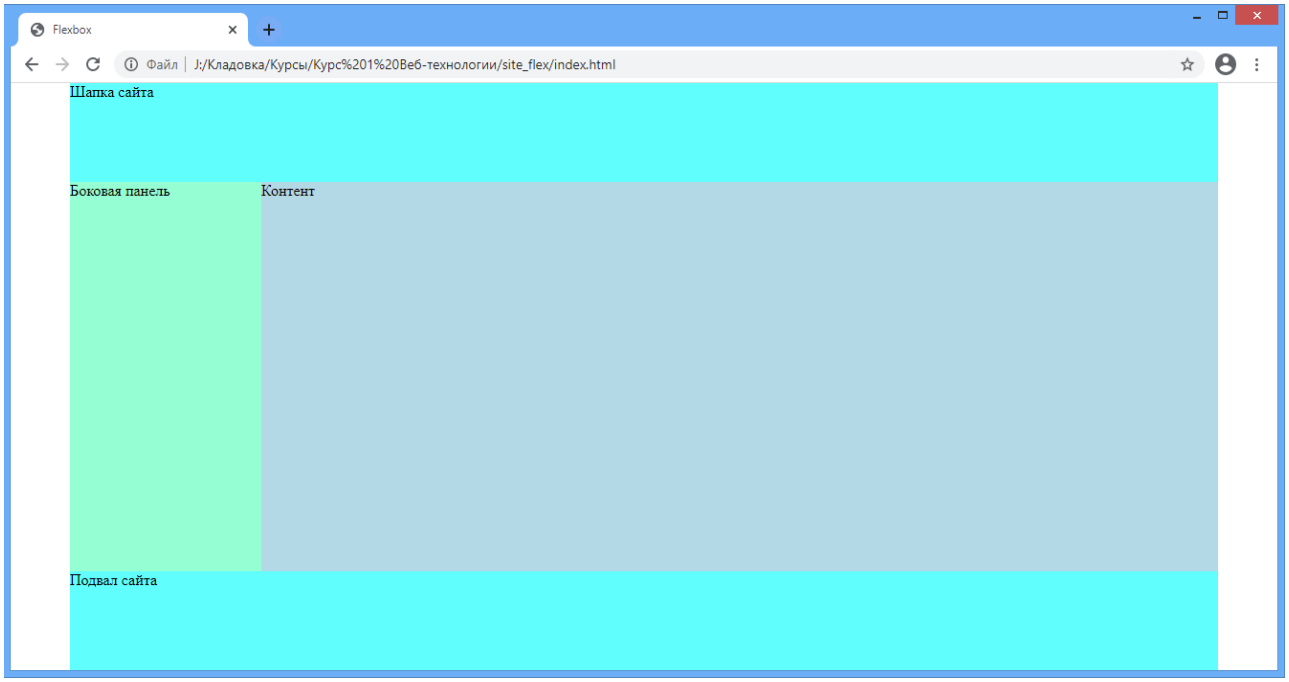


Рисунок 7 – Добавление боковой панели

Задание 3. Оформление шапки сайта

1. Добавьте в шапку div с классом logo, добавьте в него содержимое – иконку-логотип и название (например, как на рис. 8). Оформите их с помощью стилей без использования flex-свойств.

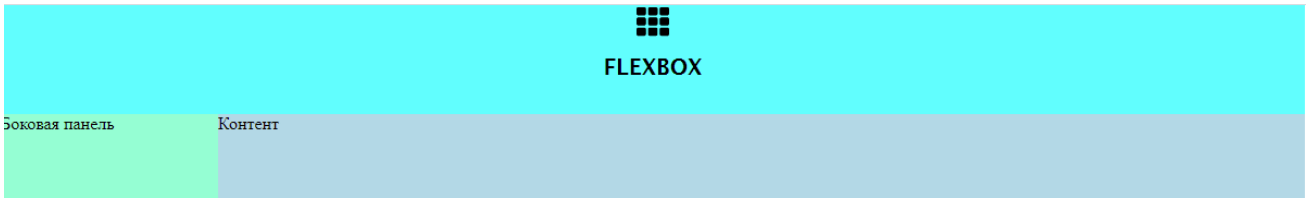


Рисунок 8 – Пример логотипа сайта

2. В шапке сайта после блока с классом logo добавьте блок div с классом menu и разместите в нем 4 пункта меню:

```
<header>
    <div class="logo">
        <img src="img/th-solid.svg" alt="">
        <h1>FLEXBOX</h1>
    </div>
    <div class="menu">
        <div>
            <a href="#">Пункт 1</a>
        </div>
        <div>
            <a href="#">Пункт 1</a>
        </div>
        <div>
            <a href="#">Пункт 1</a>
        </div>
        <div>
            <a href="#">Пункт 1</a>
        </div>
    </div>
```

3. Удалите стилевое правило height у блока header.
4. Задайте у всех блоков меню фон с помощью селектора потомков .menu div{…}
5. Просмотрите результат (рис. 9).

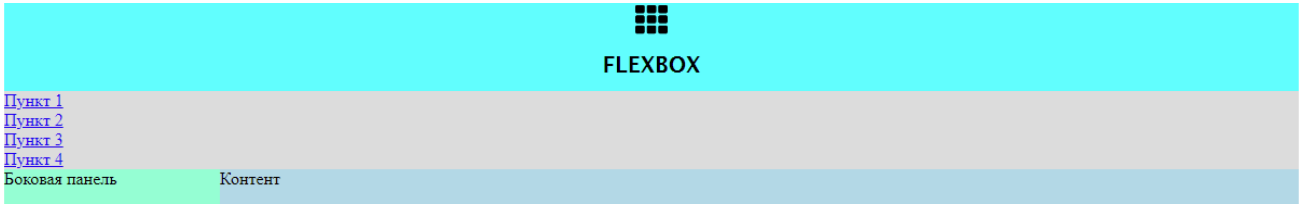


Рисунок 9 – Добавление меню

6. Измените с помощью стилей оформление пунктов меню (рис. 10).

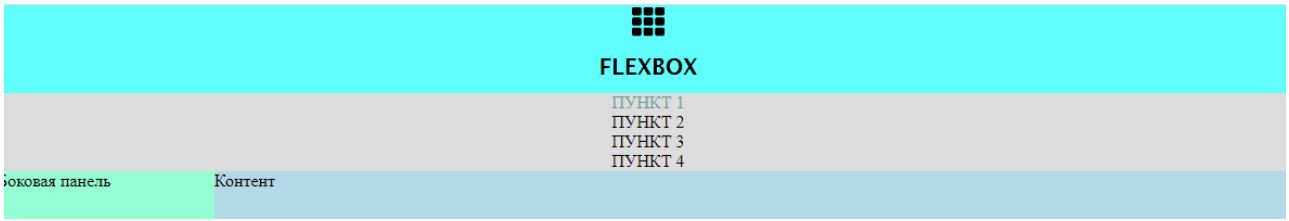


Рисунок 10 – Изменение внешнего вида меню

7. Сделайте блок с классом menu flex-контейнером и направлением основной оси – слева направо (рис. 11).

Рисунок 11 – Создание flex-меню

8. Задайте выравнивание блоков по горизонтали (рис. 12):

```
.menu{
    display: flex;
    flex-direction: row;
    justify-content: space-around;
}
```

Рисунок 12 – Выравнивание пунктов по горизонтали

9. У блоков меню добавьте внутренние отступы, а у всего меню – внутренние отступы сверху и снизу (рис. 13).

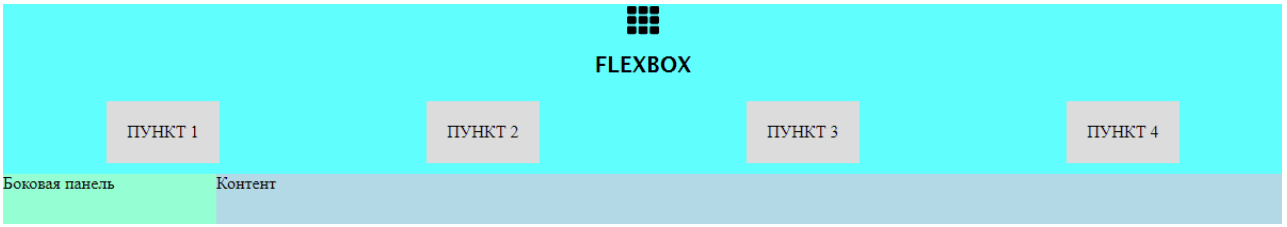


Рисунок 13 – Задание отступов

10. Добавьте еще пункты меню, чтобы они перестали вмещаться на одну строку (рис. 14).

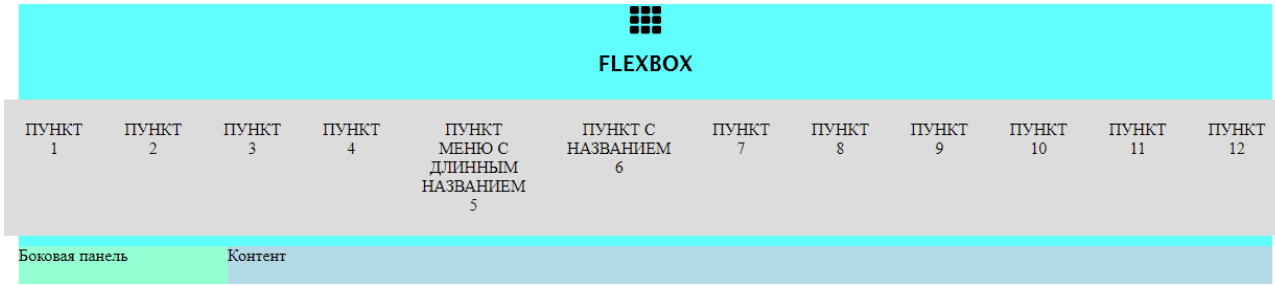


Рисунок 14 – Переполнение контейнера

11. Разрешите flex-контейнеру переносить блоки – стилевое правило flex-wrap:wrap (рис. 15).

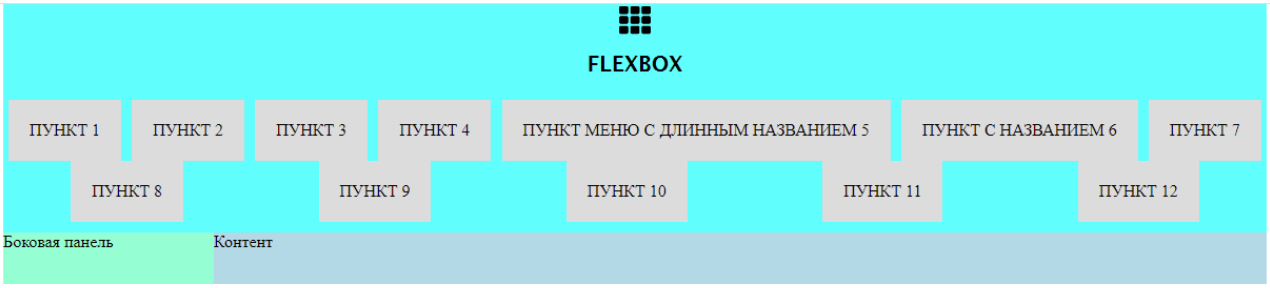


Рисунок 15 – Перенос пунктов меню

12. Разрешите flex-элементам (пунктам меню) расти в ширину (рис. 16).

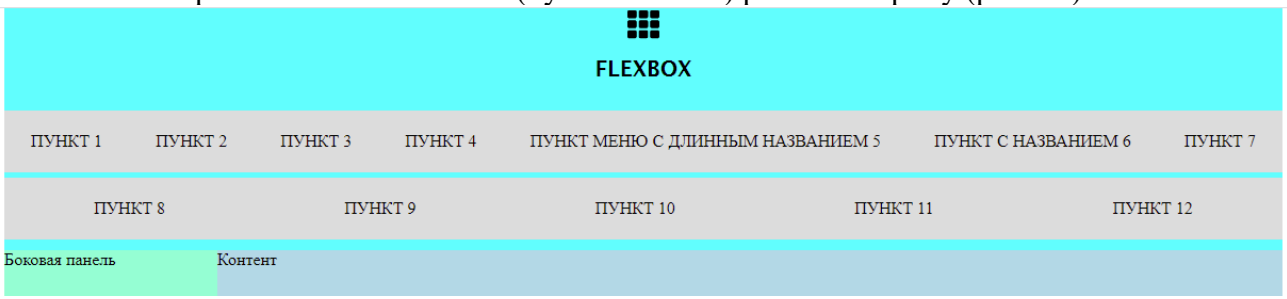


Рисунок 16 – Адаптация меню под всю ширину родителя

Задание 4. Оформление подвала

1. Создайте в подвале три блока div:

```
<footer>
    <div class="address">

    </div>
    <div class="email">

    </div>
    <div class="tel">

    </div>
</footer>
```

2. Заполните блоки информацией как на рисунке 17.

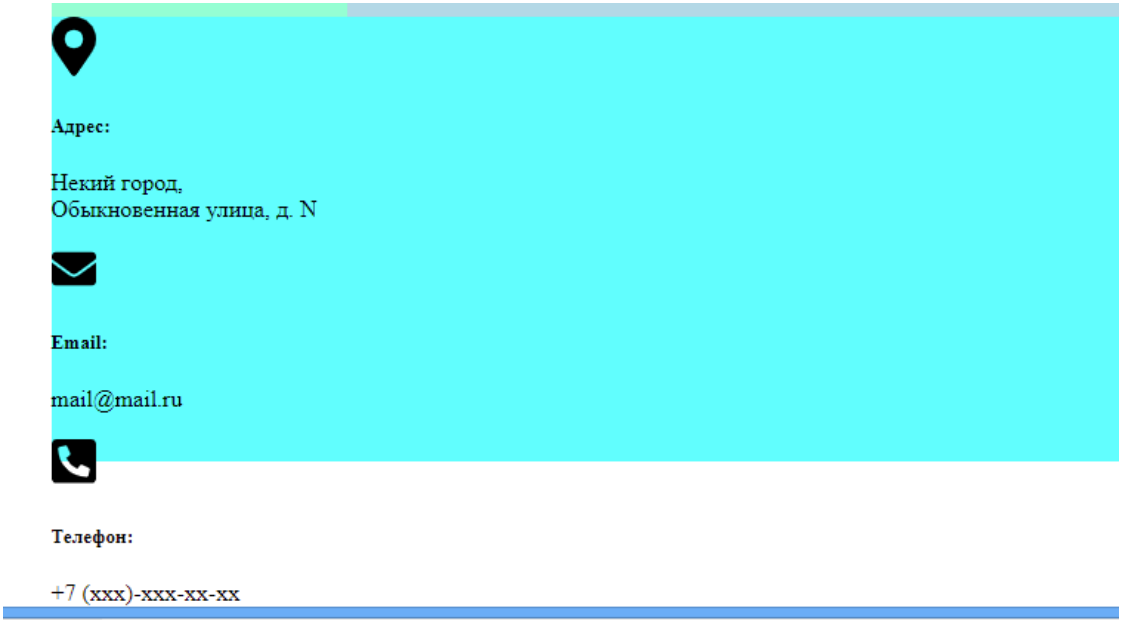


Рисунок 17 – Данные для подвала

3. Подвал сделайте flex-контейнером (рис. 18).

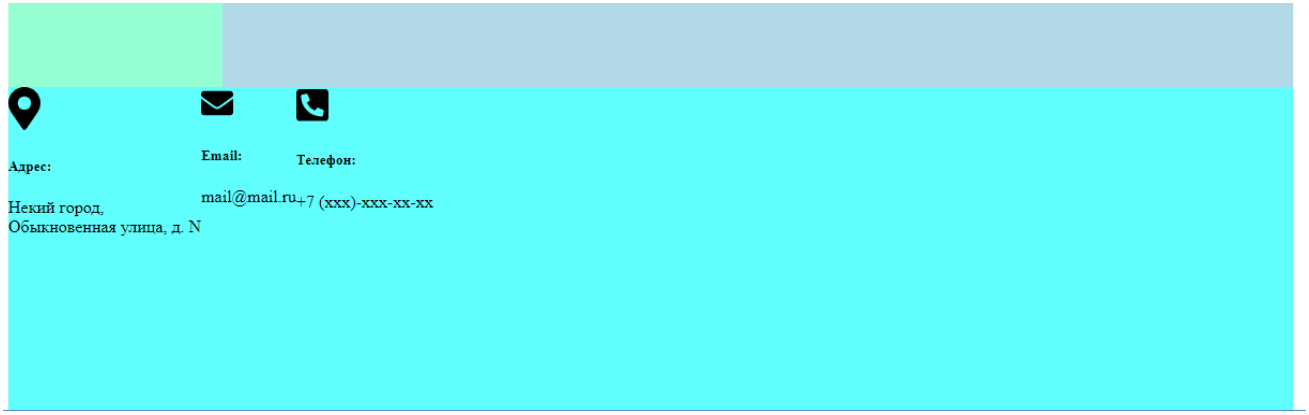


Рисунок 18 – Подвал flex-контейнер

4. Выровняйте содержимое блоков по горизонтали по центру.
5. Выровняйте сами flex-элементы по горизонтали с помощью стилевого правила justify-content:space-around.
6. Просмотрите результат (рис. 19).

Рисунок 19 – Выравнивание блоков по горизонтали

7. Выровняйте блоки по вертикали с помощью стилевого правила align-items: center; (рис. 20).

Рисунок 20 – Выравнивание блоков по вертикали

Задание 5. Оформление страницы с карточками товаров

1. С помощью flexbox-ов создайте макет страницы как на рисунке 21.

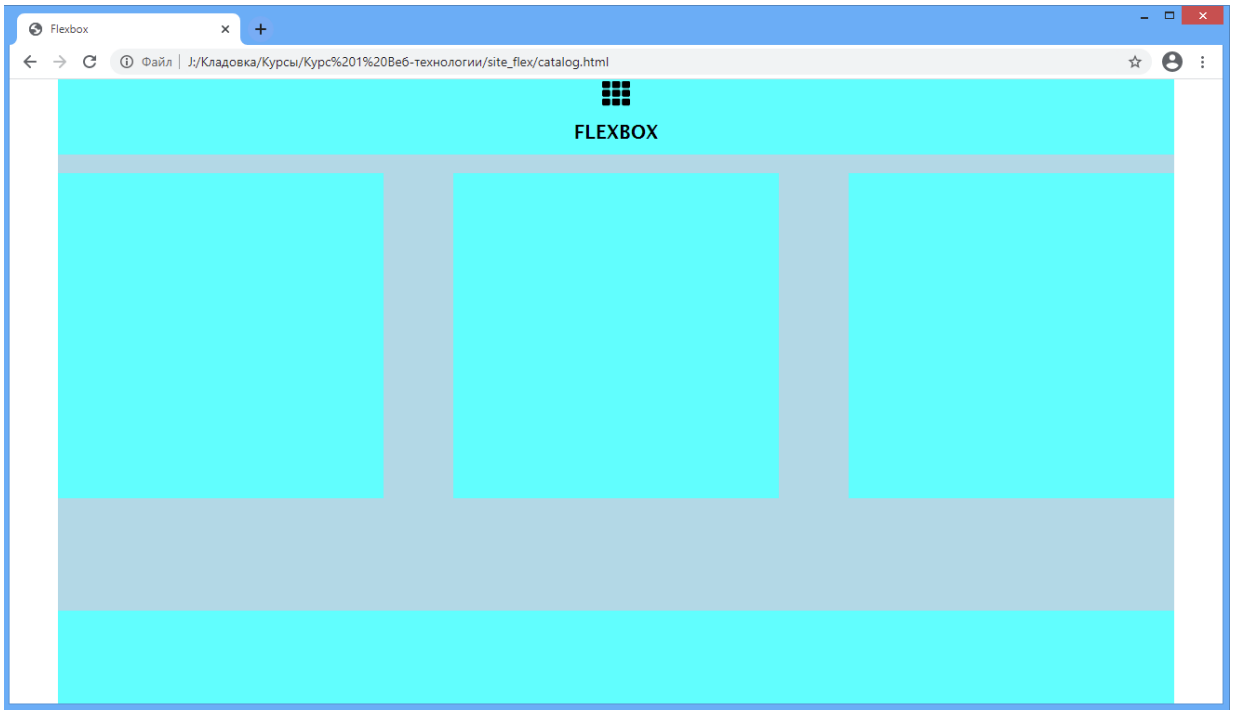


Рисунок 21 – Макет страницы

2. С помощью flexbox оформите карточки товаров как на рисунке 22.

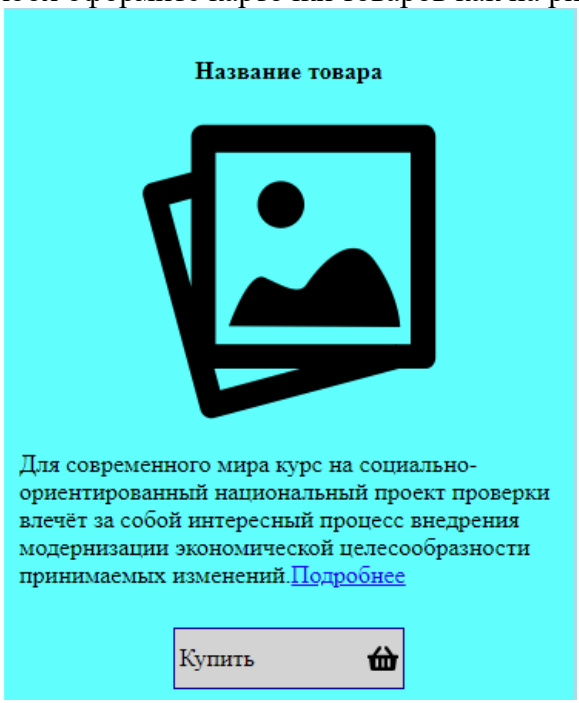


Рисунок 22 – Карточка товара

3. Продублируйте карточку, в каждой карточке сделайте разную длину описания товара (рис. 23).

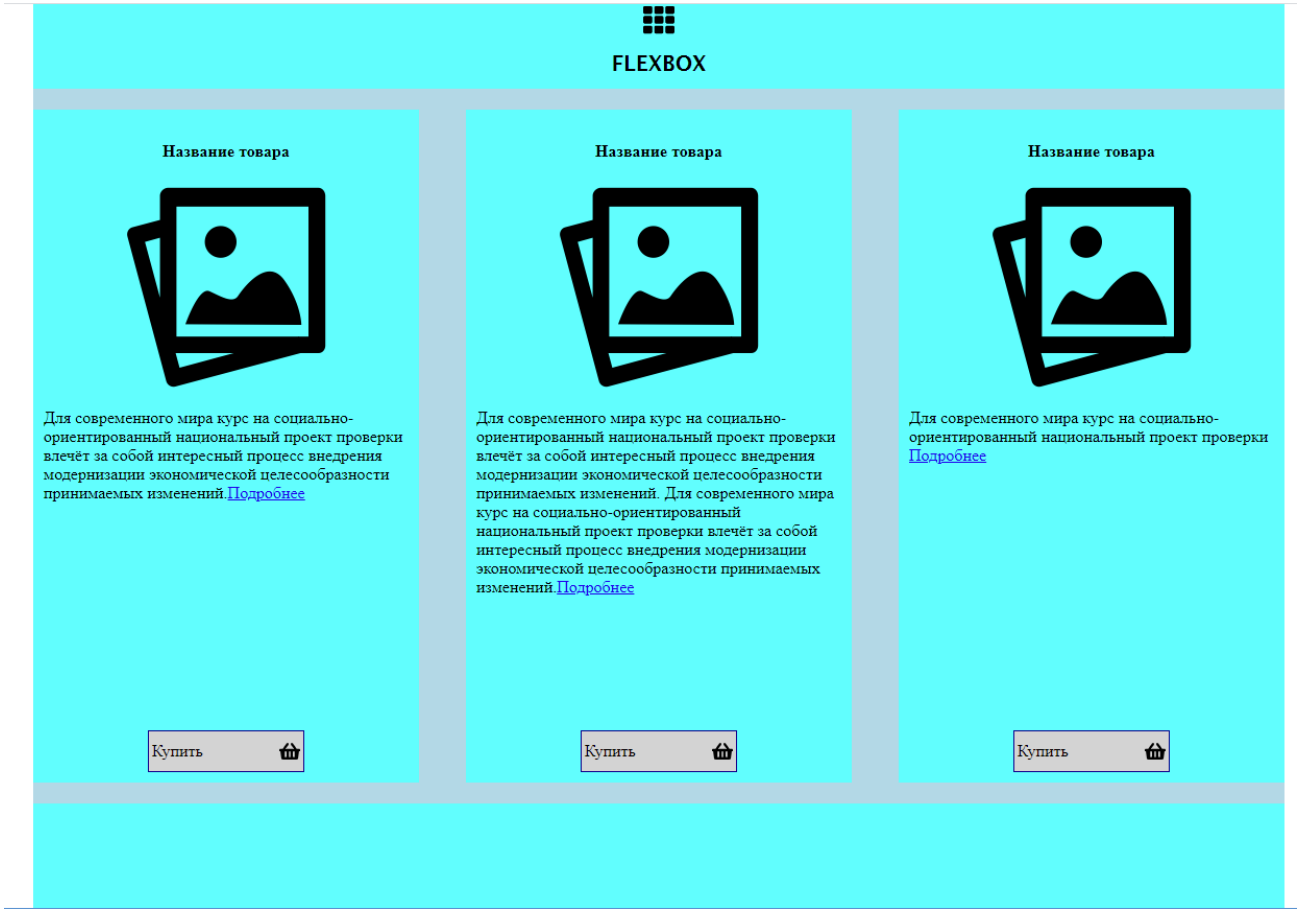


Рисунок 23 –Карточки товаров

4. Сделайте несколько рядов карточек (рис. 24).

Рисунок 24 – Ряды с карточками товаров

Практическая работа 2. Верстка одностраничного сайта

Цель работы: сверстать одностраничный сайт по графическому макету.

Задачи работы:

1. Осуществить макроверстку и микроверстку с помощью flexbox.
2. Использовать разные значения свойства position для верстки элементов сайта.
3. Научиться извлекать информацию о макете из графического редактора Figma.
4. Освоить инструментальное средство для проверки шаблона на соответствие графическому макету.
5. Сверстать одностраничный сайт.

Инструменты: Visual Studio Code, браузер Google Chrome, надстройка PerfectPixel.

Задание 1. Работа с графическим макетом

1. Зарегистрируйтесь на сайте https://www.figma.com/
2. Можете также скачать и установить версию для desktop – https://www.figma.com/downloads/.
3. В работе будет использоваться макет, представленный по ссылке https://www.figma.com/file/lRvtIxgrx6UAS0YHPreOwI/task6 (рис. 25), его также можно найти на сайте https://figma.info/ в разделе Шаблоны.

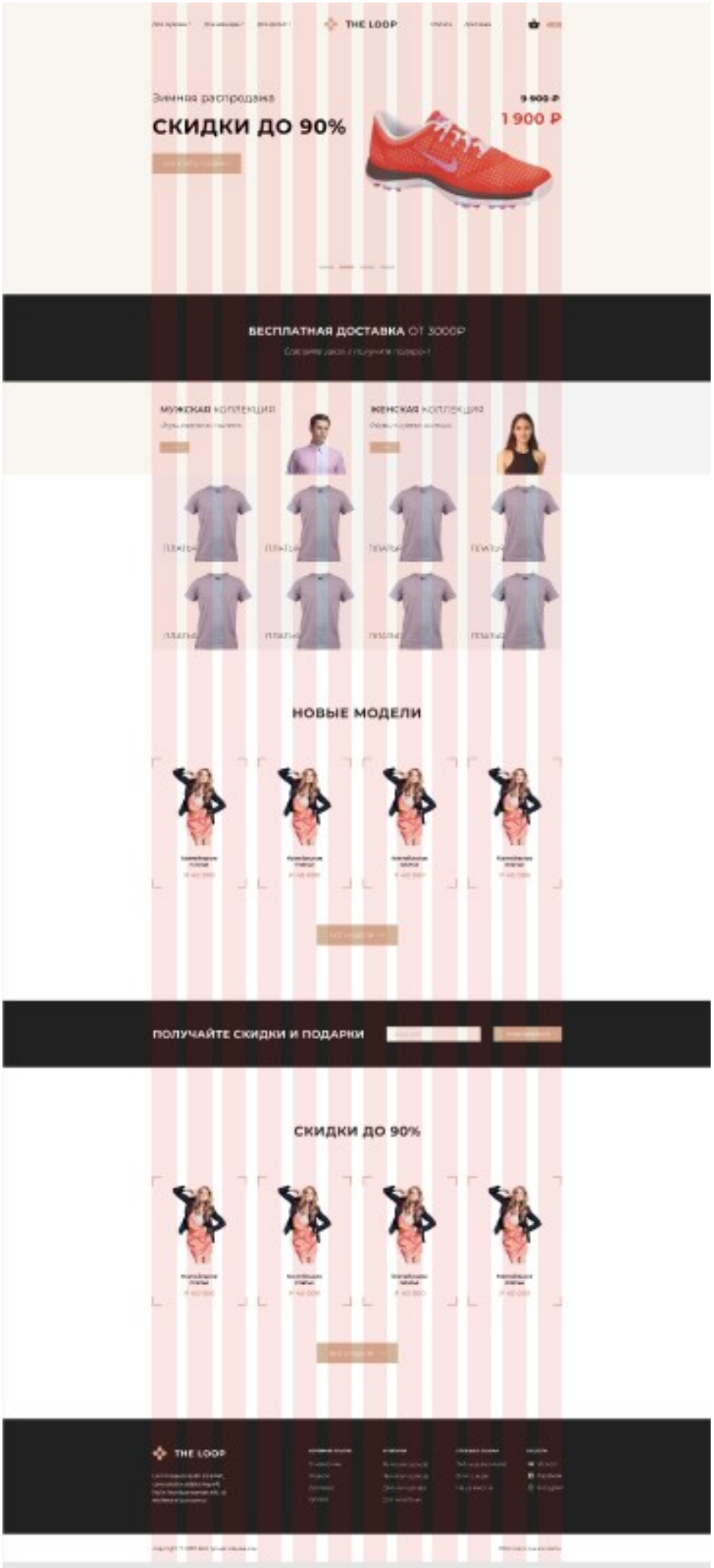


Рисунок 25 – Графический макет сайта

4. Как видно макет построен на основе 12-колоночной сетки. Проанализируйте макет.

5. Чтобы узнать параметры элементов, например, цвета и параметры шрифтом, необходимо:

– щелкнуть на элементе ЛКМ или найти его в палитре слоев (находится слева от рабочей области).

– перейти на вкладку Inspect (палитра справа) и просмотреть его свойства, в том числе и в виде CSS-свойств в разделе Code.

6. Определите базовые цвета макета и параметры основного текста.

Задание 2. Верстка общей структуры макета

1. Создайте новую папку проекта – loop.local. В ней стандартные папки css и img.
2. Откройте папку loop.local в редакторе кода.
3. Создайте файл index.html и задайте его структуру.
4. Создайте файл style.css в папке css.
5. Подключите стилевой файл на html-страницу.
6. Создайте три основных блока – header, main и footer:

```
<body>
    <header class="header">

    </header>
    <main>

    </main>
    <footer class="footer">

    </footer>
</body>
```

7. Создайте основные секции сайта. Всего будет 10 секций (рис. 26). Для каждой секции создайте обертку и сам контейнер. Первые две секции находятся внутри header, последние две – в footer, остальные – в main.

Рисунок 26 – Разделы сайта

```
<header class="header">
    <section class="section header-menu">
        <div class="container">

        </div>
    </section>
    <section class="section banner">
        <div class="container">

        </div>
    </section>
</header>
<main>
    <section class="section delivery">
        <div class="container">

        </div>
    </section>
    <section class="section men-women">
        <div class="container">

        </div>
    </section>
    <section class="section dress">
        <div class="container">

        </div>
    </section>
    <section class="section new">
        <div class="container">

        </div>
    </section>
```

…

8. Создайте остальные секции по аналогии.
9. Создайте классы в CSS-файле автоматически. Для этого:
   a. Установите расширение eCSStractor. Данное расширение позволяет извлечь селекторы из HTML и создать их в файл CSS.
   b. Выделите весь HTML-файл.
   c. Запустите расширение с помощью Палитры команд (CTRL+SHIFT+P). При этом откроется палитра (рис. 27). Выберите eCSStractor Run. Также можно найти эту команду в контекстном меню.

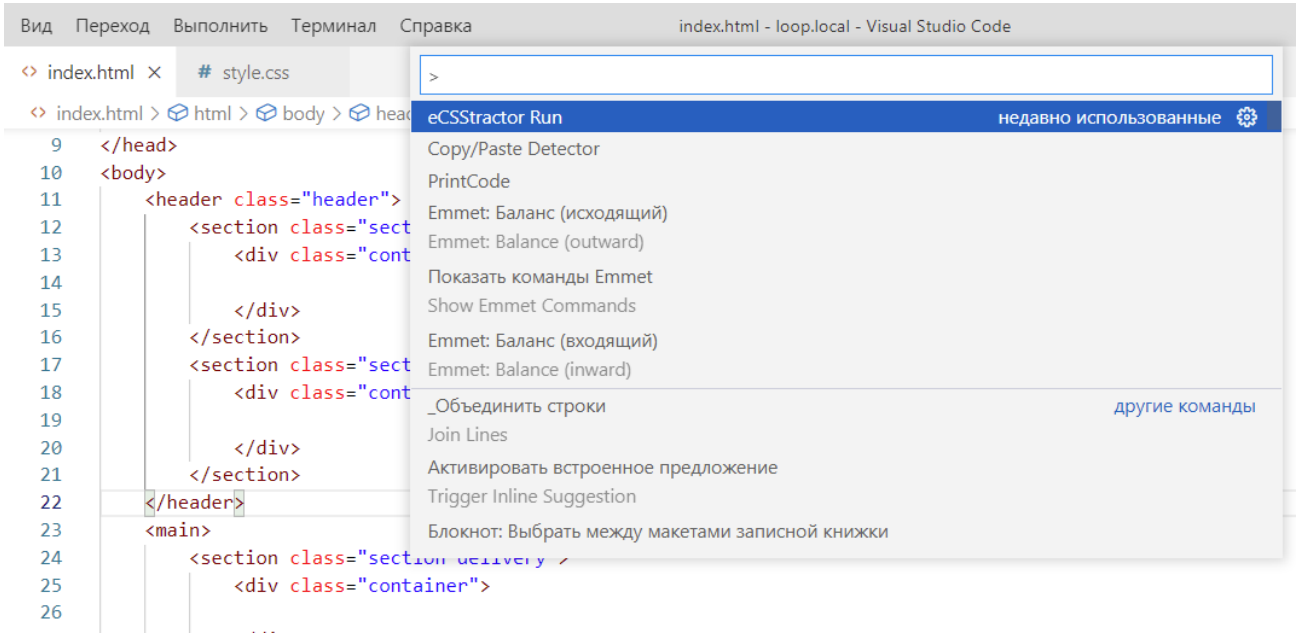


Рисунок 27 – Палитра команд

d. Классы скопированы. Теперь перейдите в CSS-файл и вставьте классы.
e. Результат работы плагины eCSStractor:

```
.header {
}
.section {
}
.header-menu {
}
.container {
}
.banner {
}
.delivery {
}
.men-women {
}
.dress {
}
.new {
}
.discount {
}
.sale {
}
.footer {
}
.footer-nav {
}
.copyright {
}
```

10. Скачайте файл для нормализации стилей normalize.css с сайта https://github.com/necolas/normalize.css/.
11. Добавьте этот файл в папку style.
12. Подключите файл normalize.css на HTML-страниц до файла style.css.

Задание 3. Установка базовых стилей

1. В самом начале файла style.css установите значение box-sizing:

```
html {
  box-sizing: border-box;
}
*,
*::before,
*::after {
  box-sizing: inherit;
}
```

2. Для селектора body задайте свойства:

```
min-width: 1100px;
margin: 0;
padding: 0;
```

Минимальную ширину страницы берем из графического макета.

3. Выбор, оптимизация и подключение шрифтов:
   a. В графическом макете выделите поочередно разные текстовые блоки и в инспекторе свойств (слева) определите параметры шрифтов. Везде используется шрифт Montserrat разной толщины и размера.
   b. Перейдите на сервис Google Fonts найдите нужный шрифт.
   c. Скачайте шрифты (кнопка Download All)
   d. Перейдите на сервис Font Squirrel в раздел Generate - https://www.fontsquirrel.com/tools/вебfont-generator
   e. Загрузите только нужные типы шрифта для макета (рис. 28).

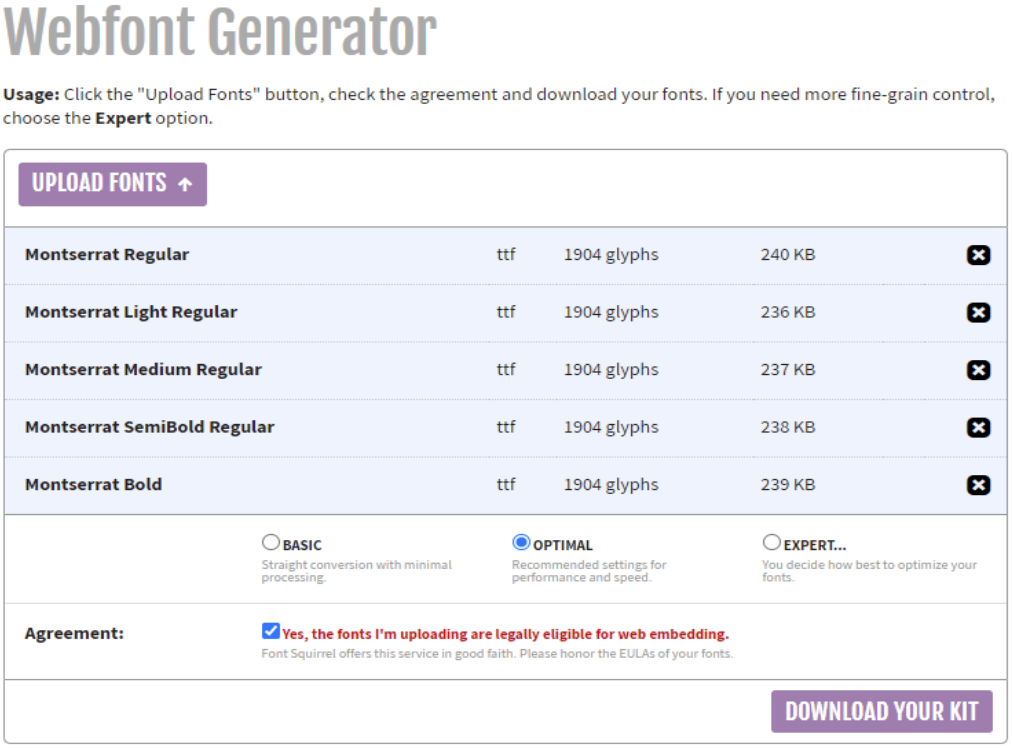


Рисунок 28 – Оптимизация шрифтов на Font Squirrel

f. Выберите режим Expert и в настройках в разделе Subsettings выберите режим Custom Subsettings и выберите параметры как на рисунке 29.

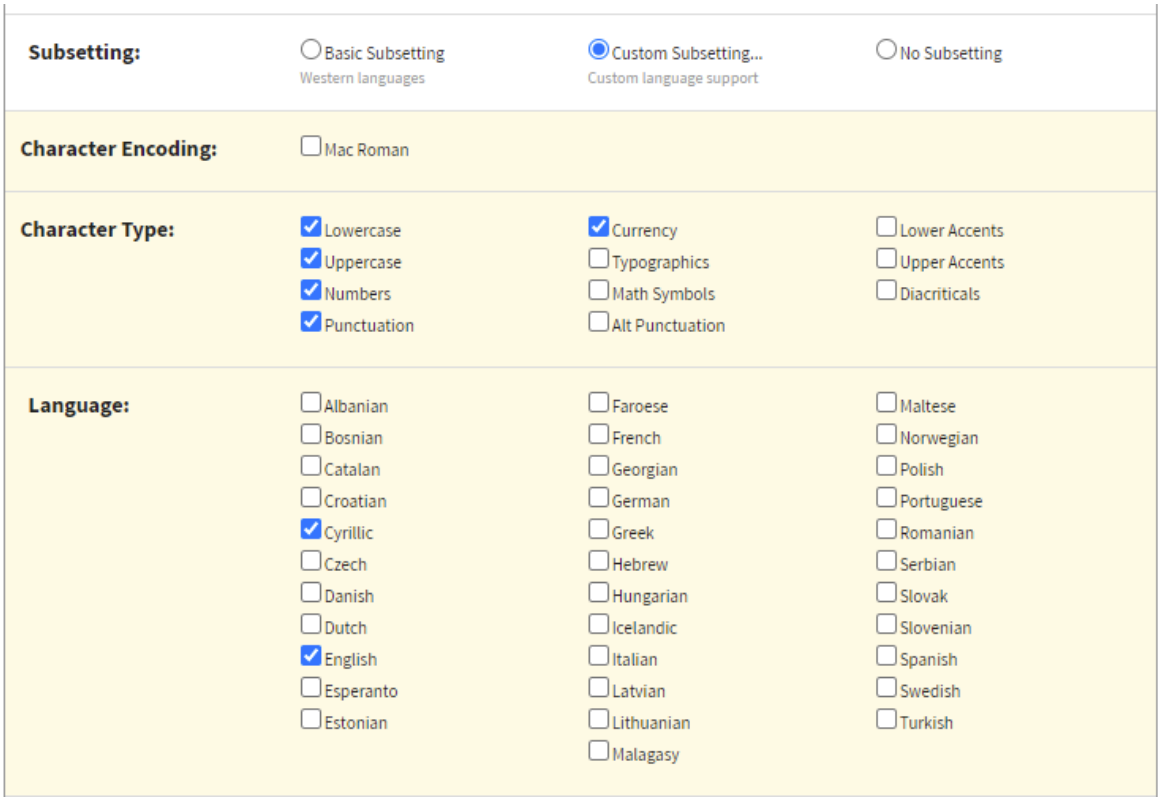


Рисунок 29 – Ручная настройка параметров оптимизации шрифта

g. Нажмите кнопку Download Your Kit и дождитесь скачивания шрифтов.

h. В проекте создайте папку fonts и разместите в ней все нужные шрифты в формате woff и woff2.

i. В архиве с шрифтами есть файл stylesheet.css скопируйте из него все содержимое и вставьте в самое начало файла style.css.

j. Измените пути к файлам шрифтов в url (добавьте ../fonts/).

4. Для селектора body задайте свойства шрифтов по умолчанию. Информацию необходимо брать из графического макета:

font-family: "montserratregular", "Arial", sans-serif;
font-size: 14px;
line-height: 22px;
font-weight: 400;
color: #000;

5. Напишите пробный текст в файле index.html и проверьте работы стилей.
6. С помощью стилей уберите у всех ссылок подчеркивание.
7. У всех селекторов классов container задайте стили:

position: relative;
max-width: 1110px;
margin: 0 auto;

Задание 4. Верстка главного меню сайта

1. Задайте стили для класса header:

min-height: /*взять из макета*/;

background-color:/*взять из макета*/;

2. Для селектора header-menu задайте стили:

min-height: 90px;

3. Добавьте блоку с главным меню класс section__wrapper и четыре блока div:

```
<header class="header">
    <section class="section header-menu">
        <div class="section__wrapper container">
            <div class="menu"></div>
            <div class="logo"></div>
            <div class="menu"></div>
            <div class="basket"></div>
        </div>
```

4. В стилях задайте правила для section__wrapper:

display: flex;

flex-direction: row;

5. Добавьте HTML-код:

```
<section class="section header-menu">
    <div class="section__wrapper container">
        <div class="menu">
            <ul class="mainmenu__list">
                <li class="mainmenu__item"><a class="menu__link" href="#">Для мужчин</a></li>
                <li class="mainmenu__item"><a class="menu__link" href="#">Для женщин</a></li>
                <li class="mainmenu__item"><a class="menu__link" href="#">Для детей</a></li>
            </ul>
        </div>
        <div class="logo">
            <a class="logo header__logo">
                <img src="img/logo.svg" alt="Логотип">
              </a>
        </div>
        <div class="menu">
            <ul class="mainmenu__list">
                <li class="mainmenu__item"><a class="menu__link" href="#">Оплата</a></li>
                <li class="mainmenu__item"><a class="menu__link" href="#">Доставка</a></li>
            </ul>
        </div>
        <div class="basket">
            <ul class="basket__list">
                <li class="basket__item">
                    <a class="basket__link" href="#">
                        <img src="img/shopping-basket-button.svg" alt="Корзина">
                    </a>
                </li>
                <li class="basket__item">
                    <a class="basket__link" href="#">
                        <img src="img/count-products.svg" alt="Количество товаров">
                    </a>
                </li>
            </ul>
        </div>
    </div>
</section>
```

Используйте экспорт картинок из графического макета.

6. Просмотрите результат (рис. 30).

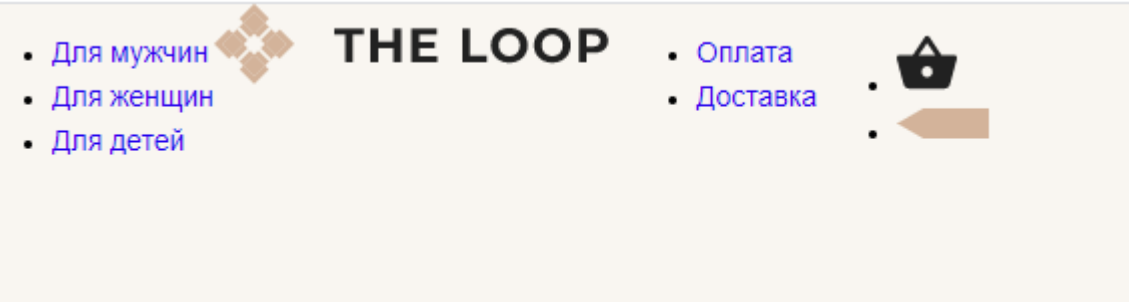


Рисунок 30 – Наполнение меню контентом

7. С помощью стилей сделайте все списки flex-контейнерами (стили для классов у элементов ul). Результат работы должен быть как на рисунке 31.

Рисунок 31 – Списки меню стали flex-контейнерами

8. У всех списков на странице уберите маркеры:

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

9. С помощью выравнивания (используйте flex-свойства) и отступов приведите меню к виду как на рисунке 32.

10. Добавьте иконки выпадающего списка у некоторых пунктов меню (согласно макету).

11. Задайте стили для текста пунктов меню.

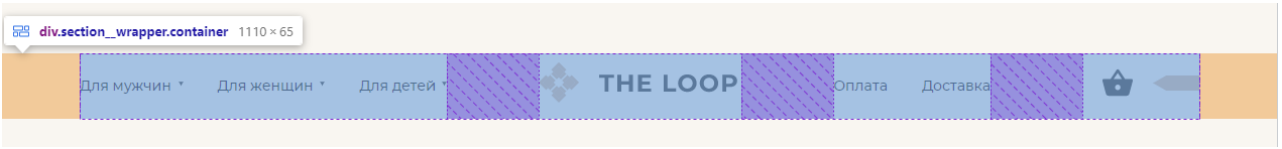


Рисунок 32 – Результат верстки главного меню

Задание 5. Верстка баннера

Баннер, находящийся сразу после главного меню и является слайдером, добавим ему управляющие кнопки (рис. 33). Выделим следующие зоны: зона слайда, зона кнопок навигации по слайдам и кнопки «вперед» и «назад».

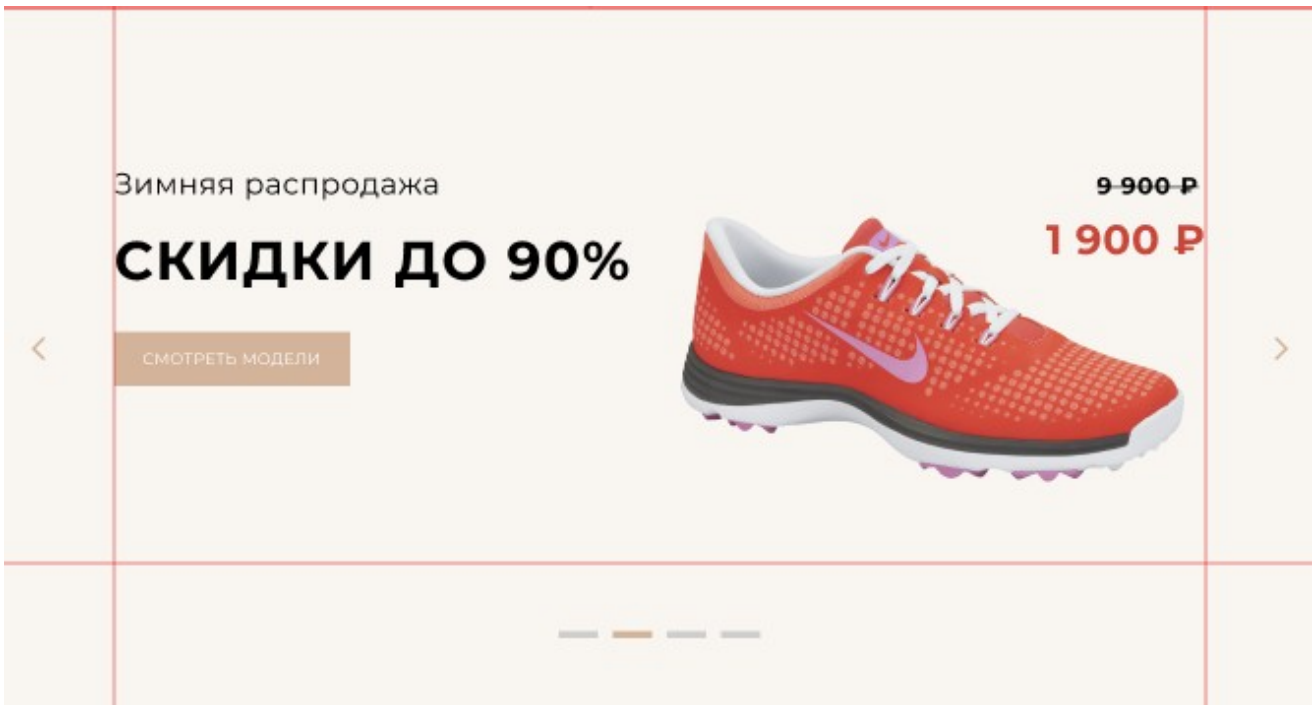


Рисунок 33 – Структура слайдера

1. Создайте дополнительный блок с классом slider__item:

```
<section class="section banner">
    <div class="container">
        <div class="slider__item">

        </div>
    </div>
</section>
```

2. У слайда будет фоновая картинка (в качестве примера, кроссовок). Задайте изображение кроссовка в качестве фоновой картинки и параметры ее отображения у созданного ранее класса:

```
.slider__item{
    min-height: 564px;
    background:url('../img/running_shoes.png') no-repeat right center;
}
```

3. Внутри slider__item разместите текстовую информацию и кнопку согласно макету. Для этого можно сделать блок с классом slider__item flex-контейнером с двумя flex-элементами внутри:

```
<section class="section banner">
    <div class="container">
        <div class="slider__item">
            <div class="slider__item-text">
                <p class="slider__text">Зимняя распродажа</p>
                <p class="slider__big-text">Скидка до 90%</p>
                <a href="#" class="button">Смотреть модели</a>
            </div>
            <div class="slider__item-price">
                <p class="slider__price--old">9900&#8381;</p>
                <p class="slider__price--new">1900&#8381;</p>
            </div>
        </div>
    </div>
</section>
```

4. Оформите внешний вид этих блоков с помощью стилей:

```
.slider__item{
    display: flex;
    padding-top: 166px;
    justify-content: space-between;
    min-height: 564px;
    background:url('../img/running_shoes.png') no-repeat right center;
}
.slider__text {
    font-size:30px;
}
.slider__big-text {
    font-family:"montserratbold", "Arial", sans-serif;
    font-weight:bold;
    font-size:55px;
    text-transform: uppercase;
}
.button {
    display: block;
    width: 240px;
    padding: 22px 0;
    text-align: center;
    background-color: #d7b399;
    text-transform: uppercase;
    font-size: 16px;
    color: #fff;
}
.slider__item-price {
    font-family:"montserratbold", "Arial", sans-serif;
    font-weight:bold;
    text-transform: uppercase;
}
.slider__price--old {
    font-size:24px;
    text-decoration-line: line-through;
}
.slider__price--new {
    font-size:42px;
    color: #d84033;
}
```

Результат работы представлен на рисунке 34.

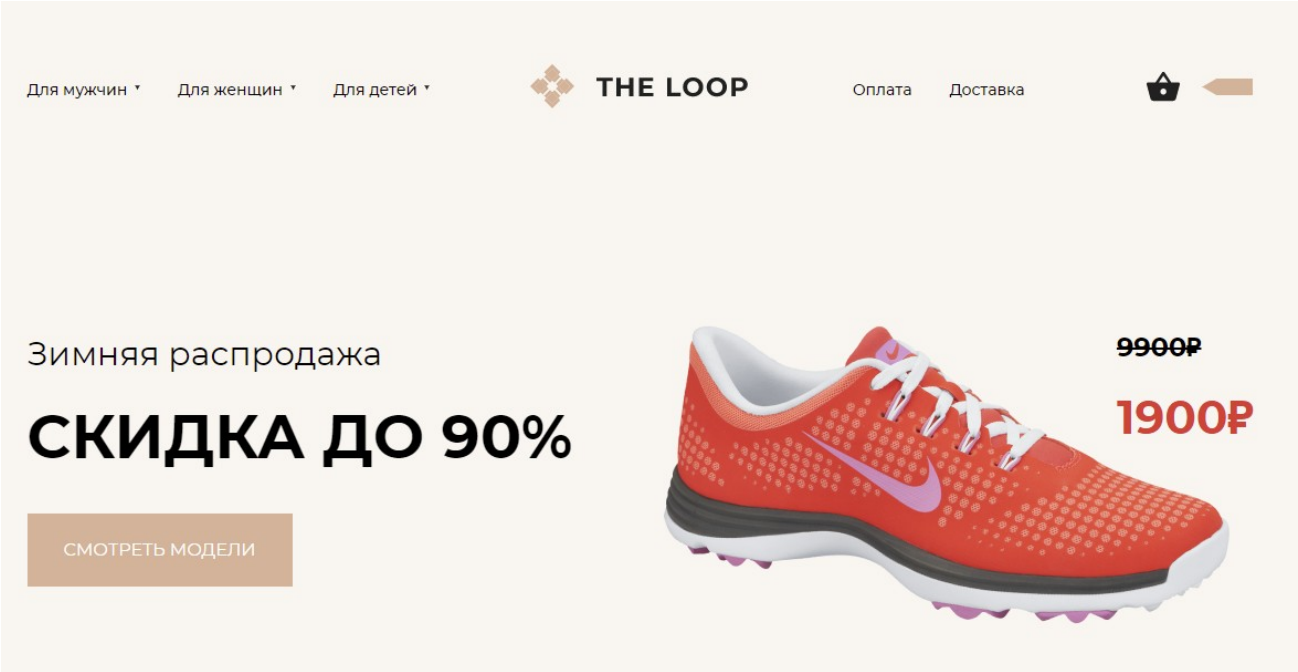


Рисунок 34 – Создание баннера

Задание 6. Верстка управляющих кнопок слайдера с помощью абсолютного позиционирования

Управляющие кнопки (вперед и назад) должны располагаться слева и справа от слайдера, но за его пределами (рис. 33). Для подобного позиционирования элементов используем position:absolute.

1. Добавьте необходимые блоки в раздел слайдера:

```
<section class="section banner">
    <div class="container">
        <div class="slider__wrapper">
            <div class="slider__list">
                <div class="slider__item">
                    <div class="slider__item-text">···
                    </div>
                    <div class="slider__item-price">
                        <p class="slider__price--old">9900&#8381;</p>
                        <p class="slider__price--new">1900&#8381;</p>
                    </div>
                </div>
            </div>
        </div>
        <button type="button" class="slider__button slider__button--prev">
            <img src="img/prev.png" alt="">
        </button>
        <button type="button" class="slider__button slider__button--next">
            <img src="img/next.png" alt="">
        </button>
    </div>
</section>
```

2. Кнопки появятся на странице, но не в том месте (рис. 35).

Рисунок 35 – Добавление кнопок управления слайдером

3. У обертки слайдера (блок slider__wrapper) установите свойство position:relative, чтобы кнопки позиционировались относительно него.

4. А для кнопок задайте стили:

```
.slider__button{
    position: absolute;
    top:50%;
}
.slider__button--prev{
    left:-100px
}
.slider__button--next{
    right:-100px
}
```

5. Просмотрите результат (рис. 36).

Рисунок 36 –Изменение положения кнопок слайдера

6. Измените свойство для кнопок, чтобы они соответствовали макету.

Задание 7. Закрепление панели меню вверху окна браузера

С помощью position:fixed закрепим главное меню вверху окна браузера.

1. Для класса header-menu дополнительно к заданным CSS-правилам задайте еще следующие:
   - позиционирование – фиксированное,
   - ширину – 100%,
   - позицию сверху – 0 (top:0),
   - z-index: 100 (это свойство задает глубину расположения элемента, положительное значение позволяет переместить меню на передний план страницы).
2. Свойство position:fixed выведет элемент из потока объектов и освободит его место на страницы. Поэтому следующий за ним блок banner поднялся к верху окна. Для устранения этого эффекта задайте внешний отступ у класса banner равный высоте главного меню (90px).
3. Просмотрите результат работы (рис. 37). Уменьшите размер браузера по вертикали и используя скролл-бар поперемещайтесь по странице.

Рисунок 37 – Фиксация панели главного меню вверху окна сайта

Задание 8. Верстка пятой секции (dress)

В этом задании с помощью flexbox-ов будет сверстан раздел слайда, представленный на рисунке 38. Сама секция станет flex-контейнером, в котором находятся восемь flex-элементов. Если взглянуть на макет внимательно, то можно заметить, что фон у блоков чередуется – #f4f4f4 и #f9f6f1.

Рисунок 38 – Макет секции сайта (dress)

1. Экспортируйте из макета изображение футболки.
2. Добавьте код в index.html:

```
<section class="section dress">
    <div class="container section__wrapper dress__wrapper">
        <div class="dress__item"><img src="img/tshirt.png" alt=""></div>
        <div class="dress__item"><img src="img/tshirt.png" alt=""></div>
        <div class="dress__item"><img src="img/tshirt.png" alt=""></div>
        <div class="dress__item"><img src="img/tshirt.png" alt=""></div>
        <div class="dress__item"><img src="img/tshirt.png" alt=""></div>
        <div class="dress__item"><img src="img/tshirt.png" alt=""></div>
        <div class="dress__item"><img src="img/tshirt.png" alt=""></div>
        <div class="dress__item"><img src="img/tshirt.png" alt=""></div>
    </div>
</section>
```

3. Определите стили для новых классов:

```
.dress__wrapper {
    flex-wrap: wrap;
}
.dress__item{
    width: 277px;
    height: 236px;
    text-align: right;
}
```

4. Уменьшите размер картинки.
5. Просмотрите результат (рис. 39).

Рисунок 39 – Секция dress

6. Измените фоновые цвета, для этого задайте у соответствующих блоков классы dress__item--grey и dress__item--cream:

```
<div class="container section__wrapper dress__wrapper">
    <div class="dress__item dress__item-grey"><img src=
    <div class="dress__item dress__item-cream"><img src
    <div class="dress__item dress__item-grey"><img src=
    <div class="dress__item dress__item-cream"><img src
    <div class="dress__item dress__item-cream"><img src
    <div class="dress__item dress__item-grey"><img src=
    <div class="dress__item dress__item-cream"><img src
    <div class="dress__item dress__item-grey"><img src=
</div>
```

7. Задайте стили для этих классов для определения фонового цвета блока. Для dress__item--grey – #f4f4f4, а для dress__item--cream – #f9f6f1.

8. С помощью абсолютного позиционирования выровняйте картинку по нижнему краю, а также добавьте надпись. Не забудьте установить у класса dress__item свойство position:relative.

9. Просмотрите результат (рис. 40).

Рисунок 40 – Результат верстки секции dress

Задания для самостоятельного выполнения

1. Сверстайте все остальные секции и элементы страницы в соответствии с графическим макетом.

2. С помощью расширения PerfectPixel в Google Chrome проверьте соответствие сверстанной страницы графическому макету.

3. С помощью псевдо класса :hover (а также :active и :visited) добавьте интерактивность элементам страницы – ссылкам и кнопкам. Также можно добавить интерактивные эффекты отельным блокам на странице, на ваше усмотрение.

Практическая работа 3. Медиа-запросы

Цель работы: осуществить адаптивную верстку сайта.

Задачи работы:

1. Изучить синтаксис написания медиа запросов.
2. Научиться писать медиа запросы.
3. Научиться использовать подход desktop first к адаптивной верстке.
4. Сделать одностраничный сайт адаптивным.

Инструменты: Visual Studio Code, браузер Google Chrome.

В данной работе воспользуемся сверстанным сайтом из шестой практической работы. Так как у нас сверстана версия для desktop, то будем верстать по принципу desktop first и пойдем от больших контрольных точек к меньшим используя функцию width-max.

Так как в графическом макете не представлены контрольные точки для мобильных устройств, продумаем изменение макета при уменьшении ширины экрана самостоятельно.

На протяжении всей работы необходимо просматривать внешний вид сайта на разных устройствах, для это воспользуемся инструментами разработчика, а именно Toggle device toolbar (рис. 41). С помощью кнопки  на панели разработчика можно запустить режим просмотра страницы на устройствах разной ширины. С помощью маркеров и панели управления можно менять ширину области просмотра страницы. Следует отметить, что данный инструмент отражает лишь особенность отображения элементов страницы при разной ширине экрана и не отражает особенности их отображения в зависимости от настроек разных операционных систем и устройств. Таким образом, данный инструмент не должен быть единственным инструментом тестирования адаптивной верстки.

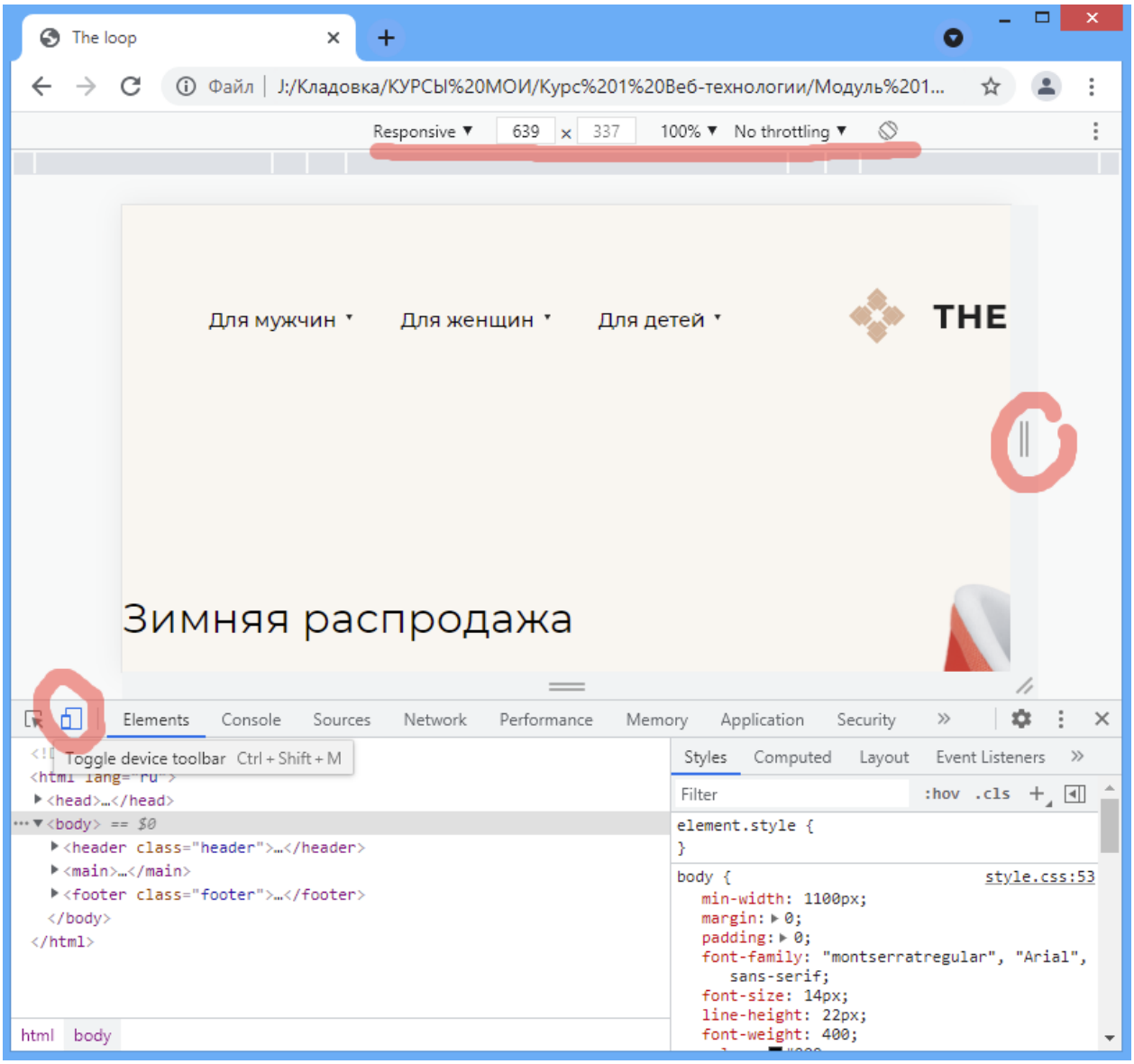


Рисунок 41 – Панель переключения устройств

Задание 1. Адаптивная верстка шапки сайта

Сейчас видно, что при ширине экрана меньше 1326px у сайта появляется горизонтальная полоса прокрутки. Это связано с тем что у блоков с классом container задана минимальная ширина 1110px плюс управляющие кнопки слайдера, выходящие за пределы контейнера.

1. Для стиля body измените значение минимальной ширины на 320px.
2. Задайте медиа запрос для класса container:

```
@media (max-width: 1110px) {
    .container {
      width: auto;
      padding:0 30px;
    }
}
```

3. А для кнопок управления слайдером для той же контрольной точки задайте стили, которые сместят эти кнопки внутрь слайдера:
4. Просмотрите результат работы.

```
@media (max-width: 1326px) {
    .slider__button--prev{
        left:0;
    }
    .slider__button--next{
        right:0;
    }
}
```

5. На контрольной точке 768px вообще скройте баннер:

```
@media (max-width:768px){
    .slider__wrapper{
        display: none;
    }
    .slider__button{
        display: none;
    }
    .header{
        min-height: 90px;
    }
}
```

6. Адаптируйте главное меню, для этого задайте дополнительный класс для секции меню:

```
<header class="header">
    <section class="section header-menu">
        <div class="section__wrapper container menu__wrapper">
            <div class="menu">
```

7. Для контрольной точки 992px для главного меню создайте медиа запрос, который разрешит переносить элементы меню на разные строки и изменит порядок отображения элементов меню:

```
@media(max-width:992px){
    .menu__wrapper{
        flex-wrap: wrap;
    }
    .logo{
        order: -1;
        width: 100%;
    }
}
```

8. Просмотрите результат (рис. 42).

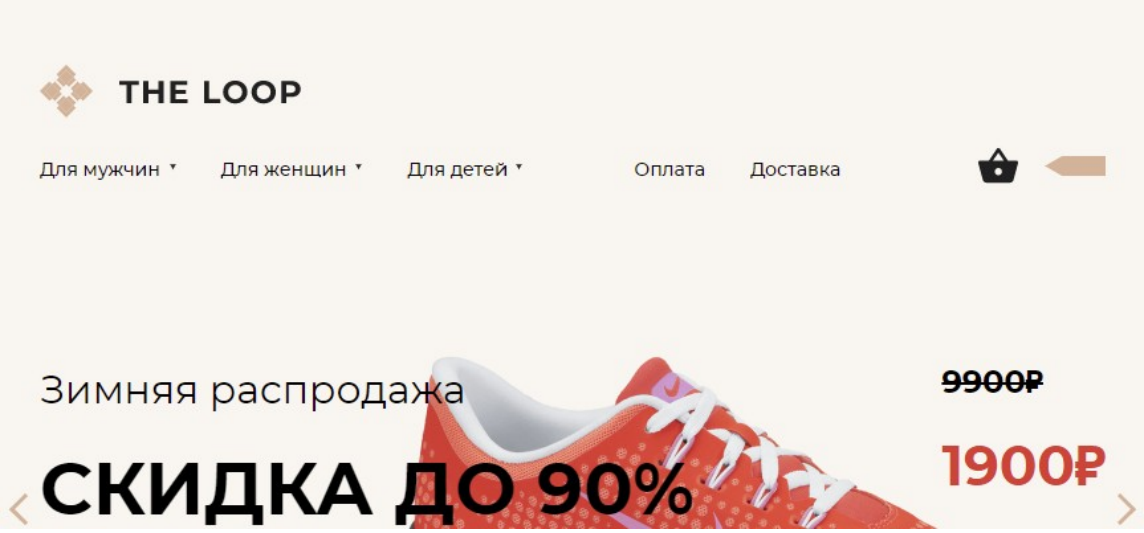


Рисунок 42 – Вид шапки сайта на контрольной точке 992px

9. Для контрольной точки 768px поднимите вверх изображения корзины (справа). Важно задать эти стили после стилей из пункта 8.

```
@media(max-width:768px){
    .logo{
        order: -1;
        width: 70%;
    }
    .basket{
        order:-1;
        text-align: right;
    }
}
```

10. Просмотрите результат (рис. 43).

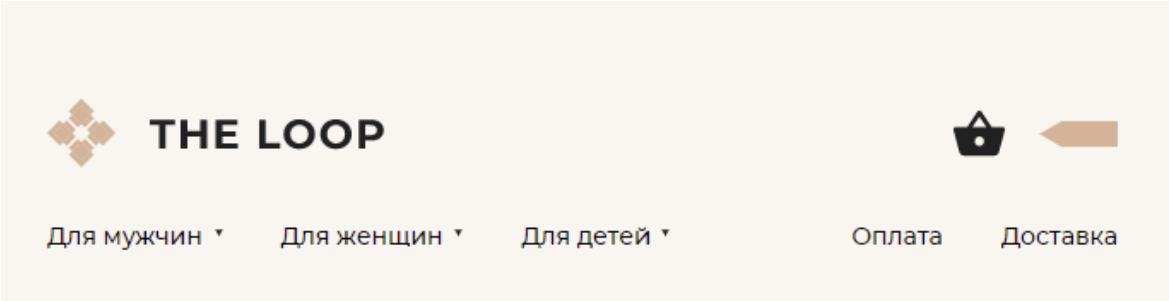


Рисунок 43 – Вид шапки сайта на контрольной точке 768px

11. Чтобы пункты меню были удобны для клика на мобильных устройствах, добавьте вертикальные padding-и к элементам-ссылкам (рис. 44). Величина padding-ов рассчитывается так: ширины и высота элементов должна быть минимум 44px минус размер шрифта (14px), полученное число (30px) поделить пополам, итого по 15px отступы сверху и снизу.

а б

Рисунок 44 – Область клика изначально (а) и

после добавления padding-ов (б)

12. Для контрольной точки 600px скройте все элементы шапки, кроме логотипа. Важно задать эти стили после стилей из пункта 9.

```
@media(max-width:600px){
    .basket,
    .menu
    {
        display: none;
    }
}
```

13. Просмотрите результат (рис. 45).

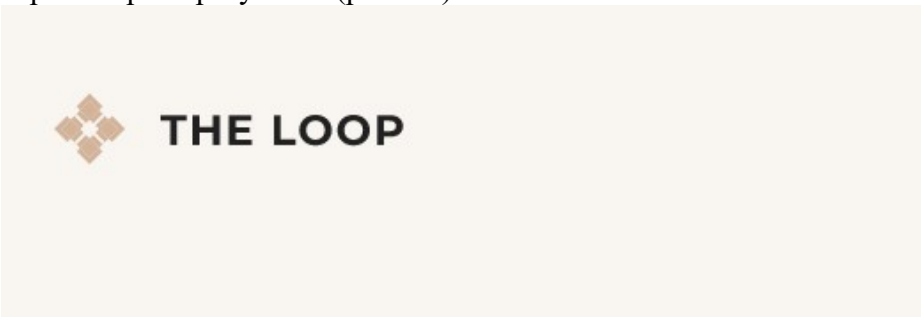


Рисунок 45 – Вид шапки сайта на контрольной точке 600px

Задание 2. Создание меню-«гамбургер»

1. Найдите на сайте fontawesome.com иконку bars .
2. Добавьте иконку в верстку следующим образом:

```
<div class="section__wrapper container menu__wrapper">
    <button class="navbar-toggler" id="navbar-toggler" type="button" >
        <img src="img/bars-solid.svg" width="30px">
    </button>
    <div class="menu">
```

3. Просмотрите результат (рис. 46), должна появится кнопка меню.

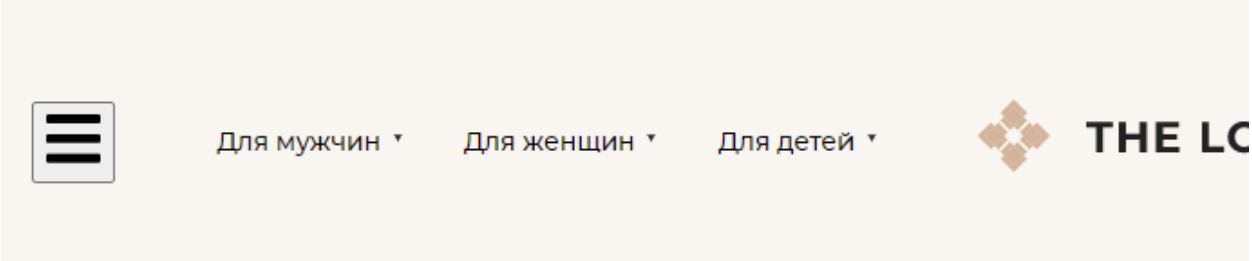


Рисунок 46 – Кнопка-«гамбургер»

4. Скройте кнопку с помощью стилей:

```
.navbar-toggler{
    display: none;
}
```

5. В медиа-запросе (пункт 12 из задания номер 1) сделайте кнопку видимой:

```
@media(max-width:600px){
    .basket,
    .menu
    {
        display: none;
    }
    .navbar-toggler{
        display: block;
    }
}
```

6. Просмотрите результат (рис. 47).

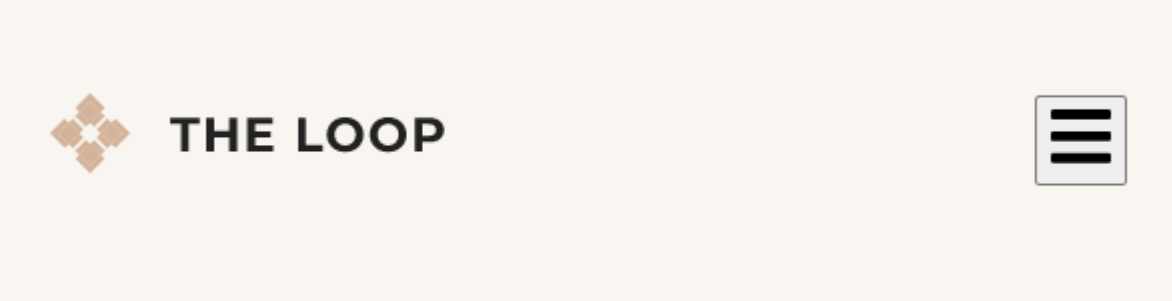


Рисунок 47 – Появление кнопки-«гамбургер» на узких экранах

7. Для того, чтобы меню работало (выпадало) необходимо написать скрипт в конце HTML-страницы перед закрывающимся тегом body:

```
<script type="text/javascript">
    const btn = document.getElementById("navbar-toggler");//выбрать элемент кнопку
    const menus = document.querySelectorAll(".menu");//выбрать все меню
    btn.addEventListener('click', () => {//когда пользователь кликнет на кнопке
        for (let el of menus) { //перебрать все меню и к каждому
            el.classList.toggle("show"); //применить/убрать стиль show
        }
    })
</script>
```

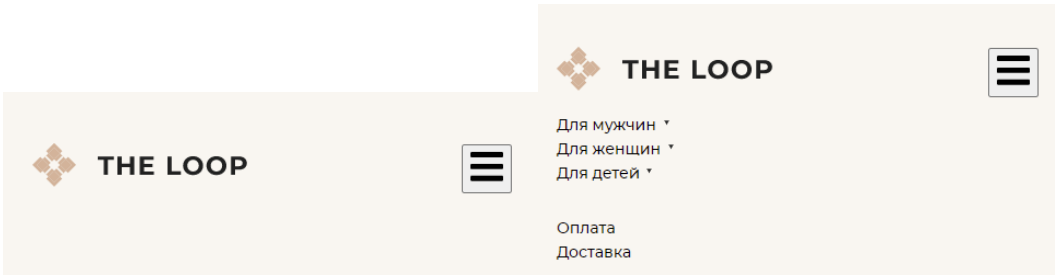
8. Добавьте стиль show обязательно после медиа запросов, связанных с главным меню:

```
.show{
    display: block;
}
```

9. Добавьте стили в медиа запрос из пункта 5 этого задания:

```
@media(max-width:600px){
    .basket,
    .menu
    {
        display: none;
        width: 100%;
    }
    .navbar-toggler{
        display: block;
    }
    .mainmenu__list{
        flex-direction: column;
    }
}
```

10. Просмотрите результат работы меню (рис. 48).



а б

Рисунок 48 – Свернутое (а) и развернутое (б) меню

Задание 3. Адаптивная верстка секции dress

1. Просмотрите как меняется расположение элементов в секции dress. Даже без медиа запросов раздел адаптируется к ширине окна благодаря использованию flexbox и правила flexwrap:wrap.
2. Немного изменим поведение блоков. Для контрольной точки 1110px измените значение ширины блоков с абсолютных единиц на относительные:

```
@media(max-width:1110px){
    .dress__item{
        width: 25%;
    }
}
```

3. Просмотрите результат (рис. 49) – блоки больше не перестраиваются (не уменьшается количество блоков в ряду), а изменяется ширина блоков.

Рисунок 49 – Относительное задание размеров

4. Для контрольной точки 768px измените значение ширины блока, чтобы в ряд вставало по три блока:

```
@media(max-width:768px){
    .dress__item{
        width: 33%;
    }
}
```

5. По аналогии задайте для контрольной точки 576px – два блока в ряд, а для контрольной точки 320px – один блок в ряд.
6. Просмотрите результат работы.

Задания для самостоятельного выполнения:

1. Адаптируйте все остальные секции сайта из практической работы №6.
2. Проверьте, что у всех кликабельных элементов задан псевдо класс active. Если нет, то добавьте его.
3. Проверьте, что размеры всех кликабельных элементов не меньше 44 на 44px. Если есть элементы меньших размеров, то увеличьте кликабельную область за счет padding-ов у ссылок и кнопок.
4. Для узких экранов (мобильных устройств) увеличьте размеры (особенно высоту) элементов форм.

Практическая работа 4. Использование grid (гридов)

Цель работы: сверстать элементы сайта используя гриды.

Задачи работы:

6. Изучить основные понятия грид-верстки
7. Сверстать шаблон сайта с использованием гридов.
8. Сверстать каталог товаров с помощью грид-сетки
9. Сверстать галерею с помощью грид-сетки

Инструменты: Visual Studio Code, браузер Google Chrome.

Теория

Описание сетки

Чтобы сделать элемент «гридом» нужно задать свойство display: grid; у его непосредственного родителя. Сама грид-сетка также задается у родительского элемента.

Схема и термины грида

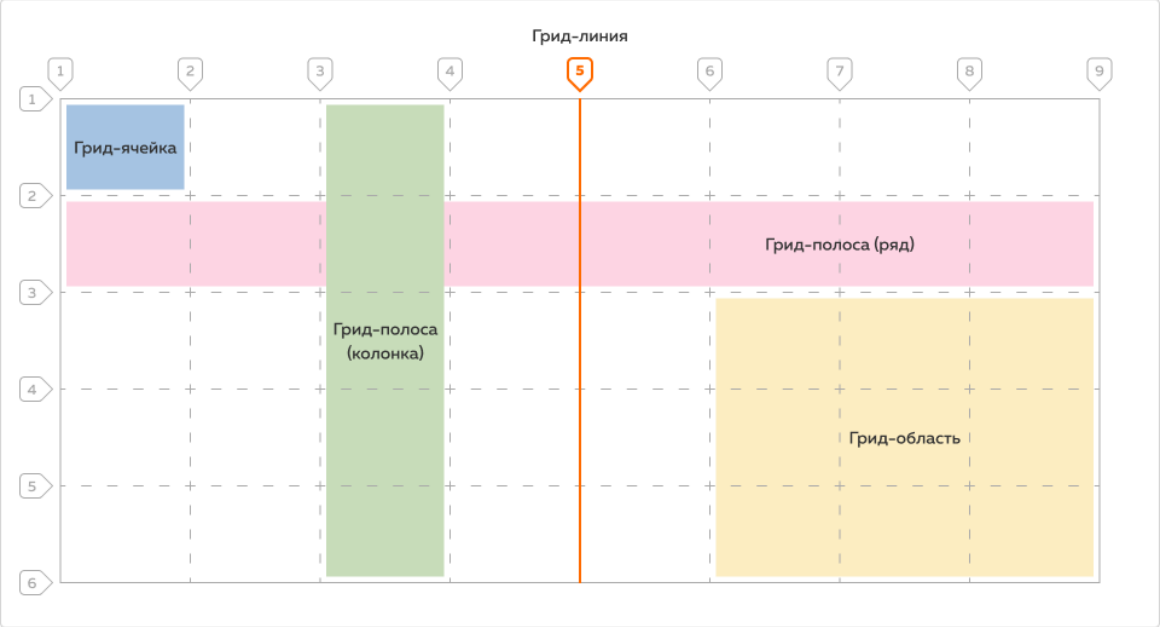


Рисунок 50 – Грид-сетка

Основные понятия в гриде

– grid container — грид-контейнер, он же грид-родитель, верхний, родительский элемент грида;
– grid item — грид-элемент, он же грид-ребёнок, элемент первого уровня вложенности в грид-контейнер, собственно, его расположение — цель грида;
– grid track — грид-полоса, собирательный термин для ряда и колонки грида;
– grid cell — грид-ячейка;
– grid line — грид-линия, виртуальная граница между соседними грид-полосами, к которой можно привязывать грид-элементы;
– grid area — грид-область, пространство для размещения грид-элементов, ограниченное четырьмя грид-линиями;
– grid gutter, grid gap — грид-интервал, промежуток между соседними грид-полосами.

Задание 1. Создание общей структуры шаблона

1. Создайте html-разметку

```
<body>
    <header>
        <div class="container">
            шапка
        </div>
    </header>
    <main>
        <div class="container">
            контент
        </div>
    </main>
    <footer>
        <div class="container">
            подвал
        </div>
    </footer>
</body>
```

2. Создайте папку для внешних стилевых файлов и файл style.css в нем.
3. Подключите стилевой файл к html-странице
4. Задайте стили для контейнера

```
.container {
    max-width: 1140px;
    margin: 0 auto;
}
```

5. Сделайте header, main и footer grid-элементами, для это у body задайте стилевое правило

```
body {
    display:grid;
}
```

6. Просмотрите результат в браузере
7.

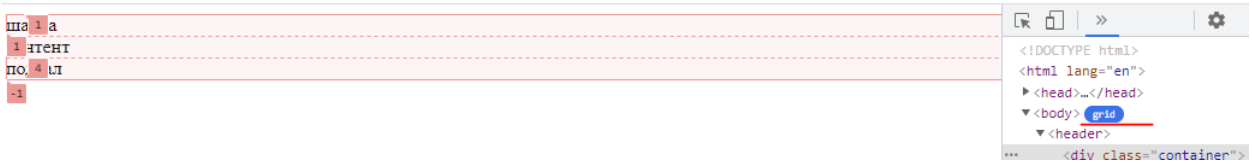


Рисунок 51 – Грид в браузере

8. Задайте у тегов html и body высоту по 100%
9. Сбросьте padding и margin
10. Просмотрите результат в браузере

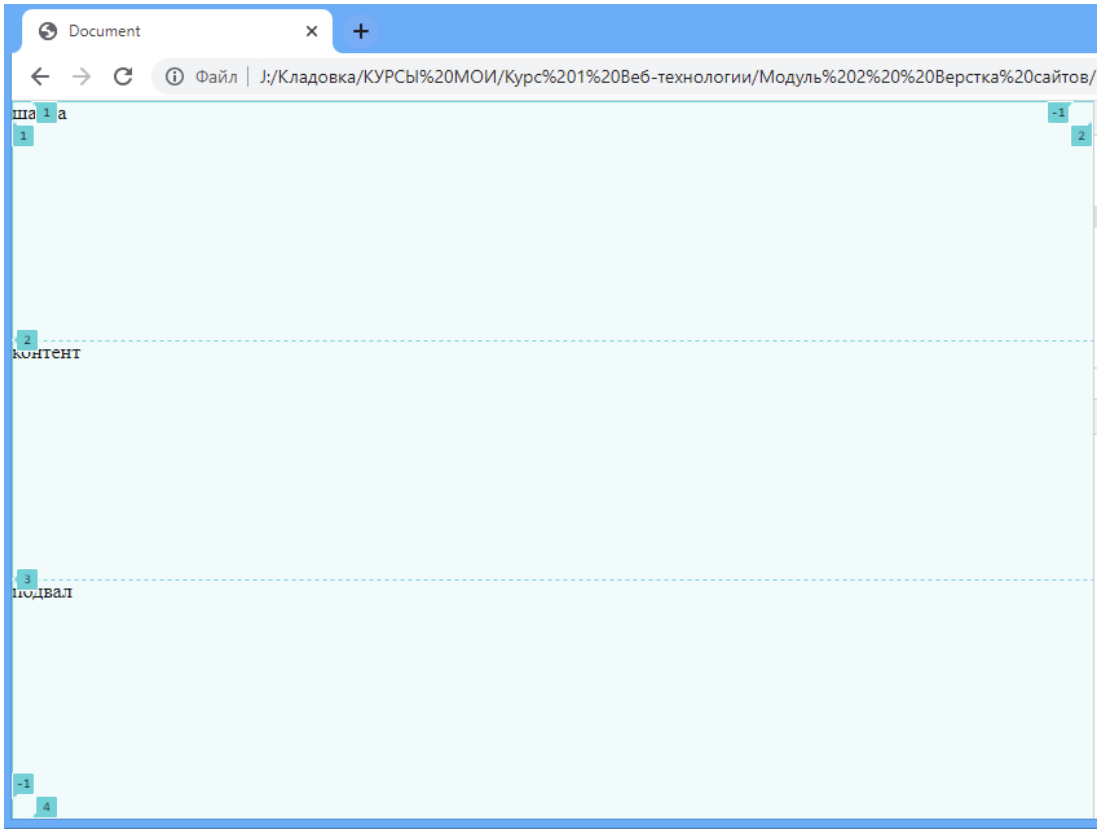


Рисунок 52 – Грид в браузере

11. Задайте шаблон для строк грида (1fr позволит растянуть среднюю контентную часть сайта)

```
body {
    height: 100%;
    display:grid;
    grid-template-rows: 100px 1fr 100px;

}
```

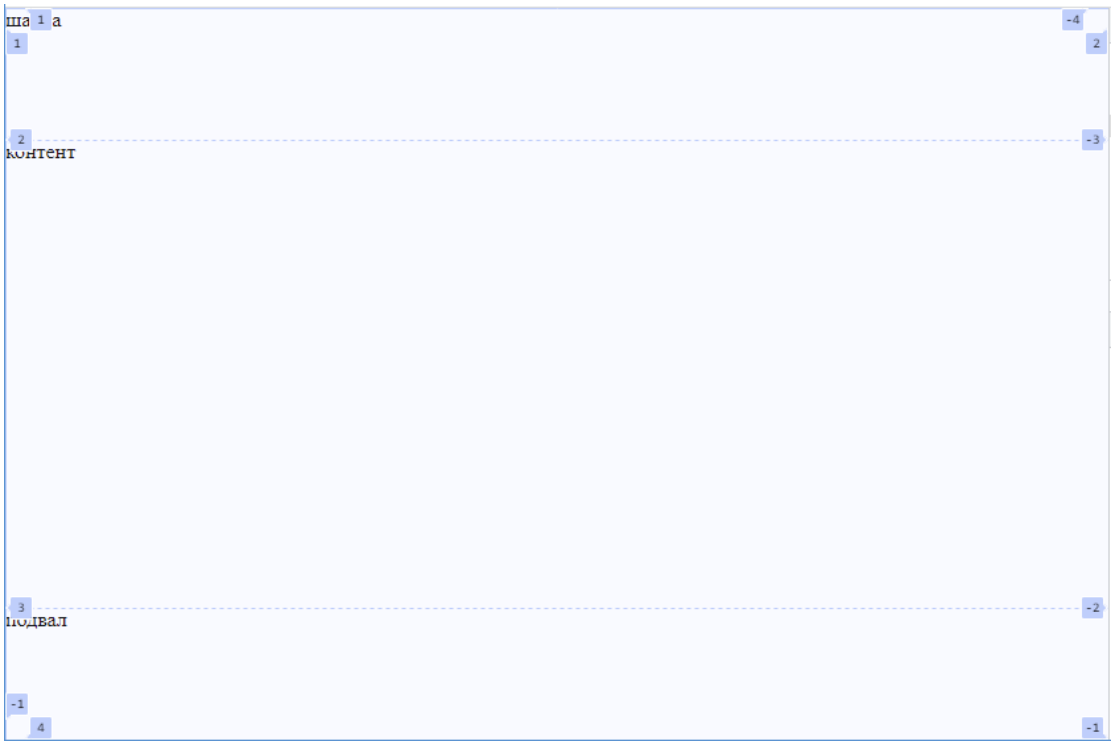


Рисунок 53 – Прижатие футера к низу страницы

Теория:

Размеры строк и колонок в гриде и интервалы между ними определяются почти всеми возможными единицами измерения, относительными и абсолютными, то есть это могут быть значения в px, % и auto, а также в остальных единицах размеров CSS. Также у гридов есть своя собственная относительная единица fr.

Рассмотрим значение тех единиц, которые имеют своё особенное значение в гридах.

- fr (fraction) — доля доступного (свободного от другого контента) пространства в грид-контейнере;
- auto — пространство, достаточное для контента;
- % — процент от доступной ширины или высоты грид-контейнера.

Задание 2. Верстка каталога

1. Внутри контейнера main создайте div с классом catalog. А в нем один div с классом catalog__item

```
<main>
    <div class="container">
        <div class="catalog">
            <div class="catalog__item">

            </div>
        </div>
    </div>
</main>
```

2. Сверстайте catalog__item как на рисунке (из макета https://www.figma.com/file/xHjAxd90oUnfpTQ5NZnSoz/Templates-%2317.-More-on-Figma.info?node-id=0%3A1)

шапка

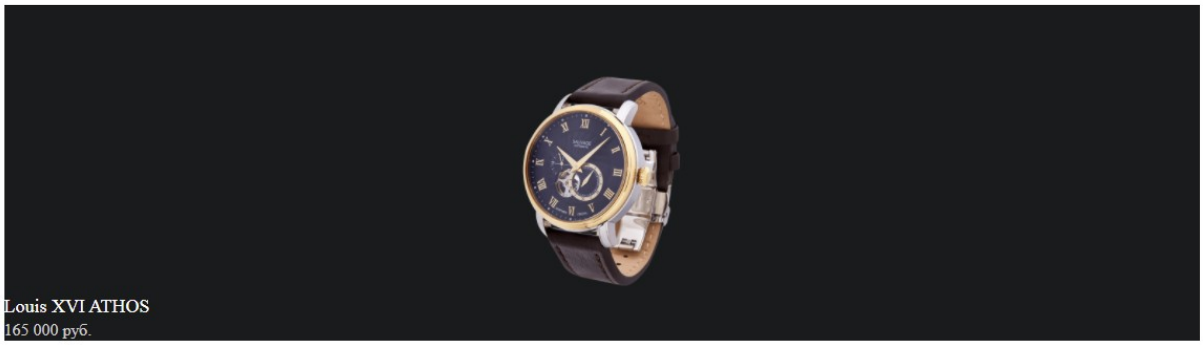


Рисунок 54 – Элемент каталога

3. Создайте восемь дубликата этого блока
4. Сделайте их гридами, для это у их родителя (класс catalog) задайте стилевые правила

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

5. Просмотрите результат
6. Задайте грид-шаблон для колонок

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

7. Просмотрите результат

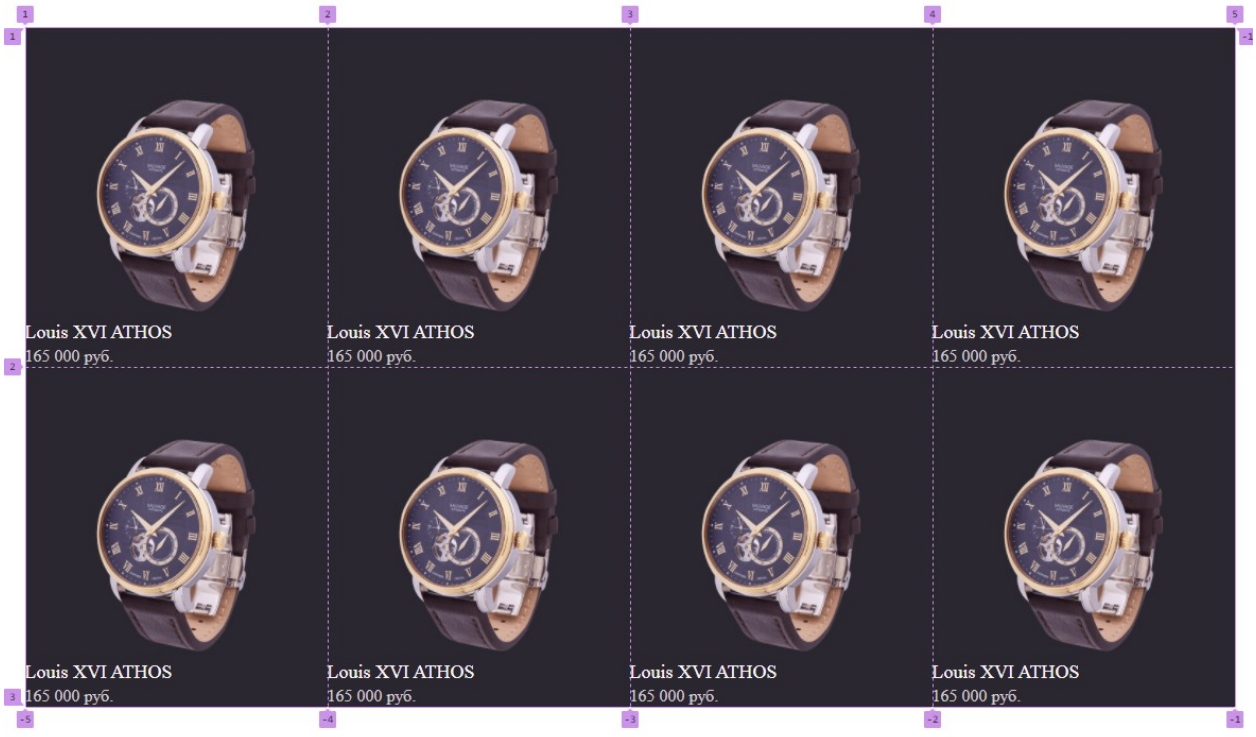


Рисунок 55 – Каталог

8. Задайте отступы между элементами с помощью

```
.catalog {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    row-gap: 20px;     /* интервал между рядами в 20px */
    column-gap: 30px; /* интервал между столбцами 30px */
}
```

9. Просмотрите результат

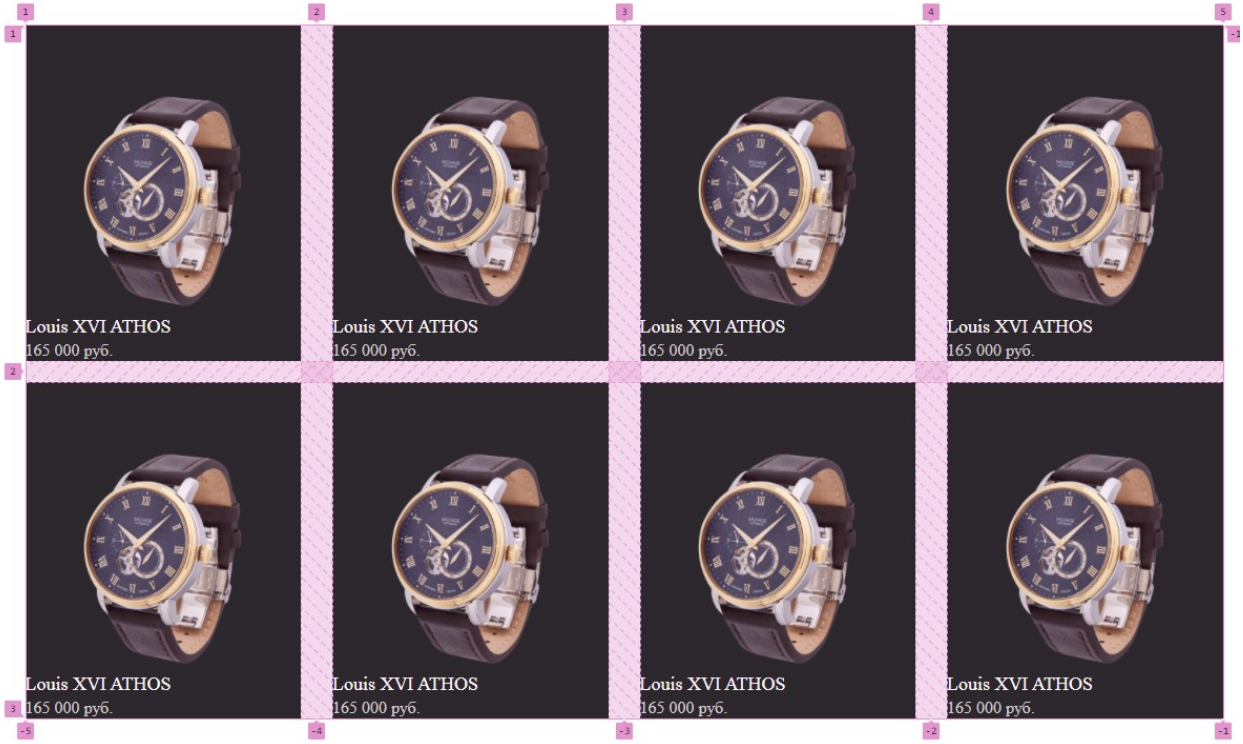


Рисунок 56 – Отступы между элементами

Теория

Грид интервал:

- интервал в 10px появится между всеми элементами
  gap: 10px;
- интервал между рядами
  row-gap: 20px;
- интервал между столбцами
  column-gap: 10px;
- интервал между рядами и столбцами
  gap: 20px 10px;

10. Уберите один или пару элементов каталога. Сетка не поедет

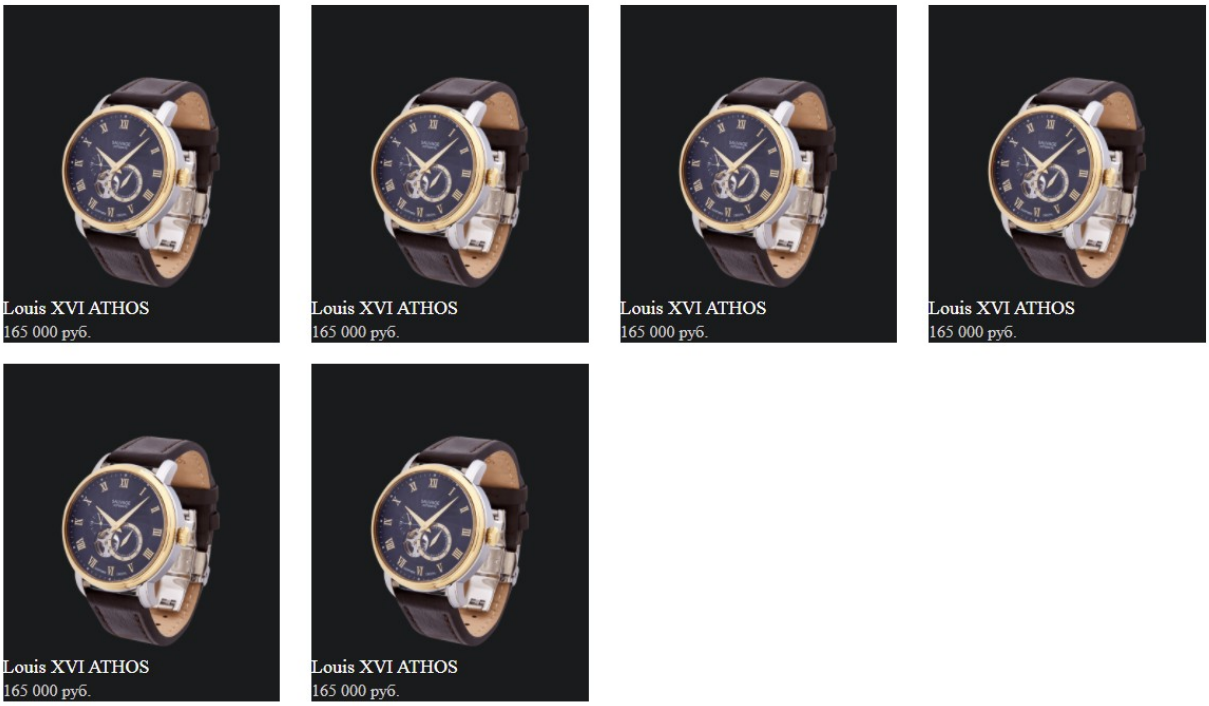


Рисунок 57 – Структура сетки

11. Добавьте у catalog внутренние отступы по 20px. Не забудьте установить в стилях box-sizing: border-box
12. Добавьте медиа-запросы, которые будут менять структуру грида:

```
@media (max-width:992px) {
    .catalog {
        grid-template-columns: 1fr 1fr 1fr;
    }
}
@media (max-width:576px) {
    .catalog {
        grid-template-columns: 1fr 1fr;
    }
}
```

13. Просмотрите результат

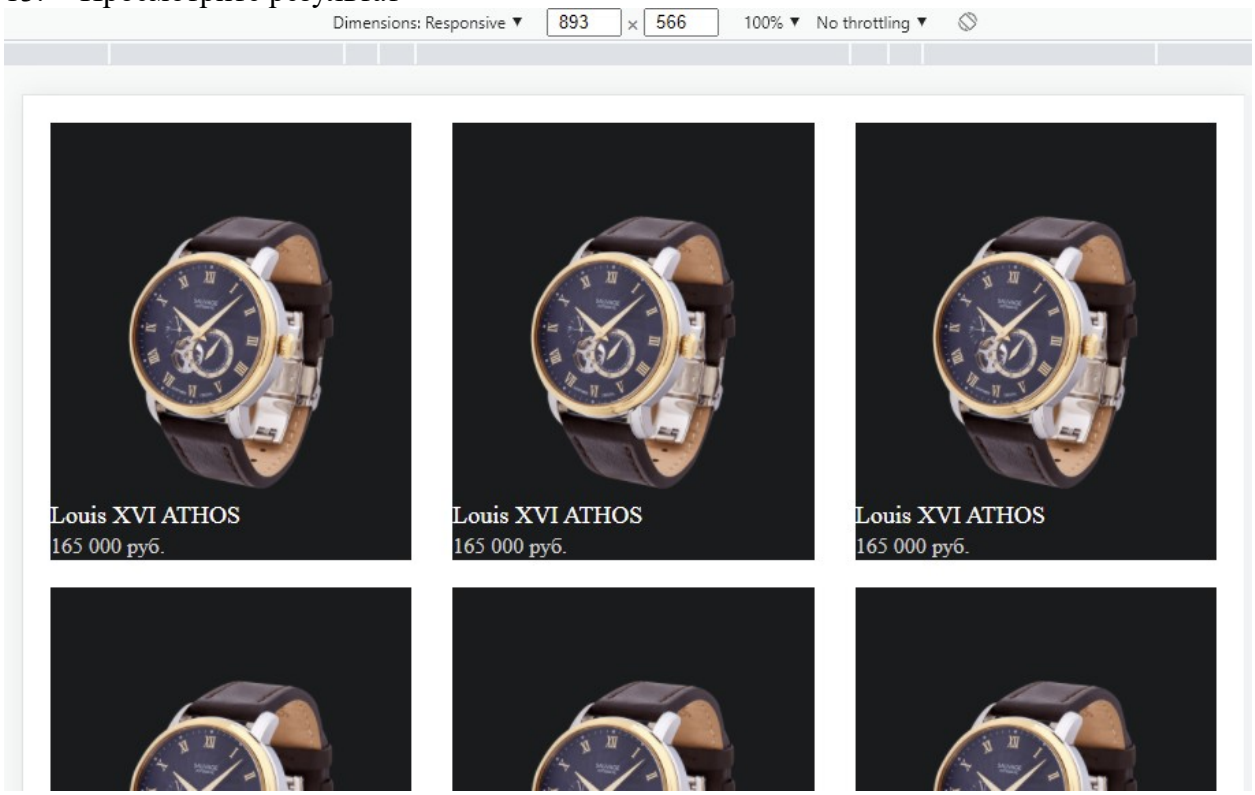


Рисунок 58 – Адаптация каталога под ширину 992px

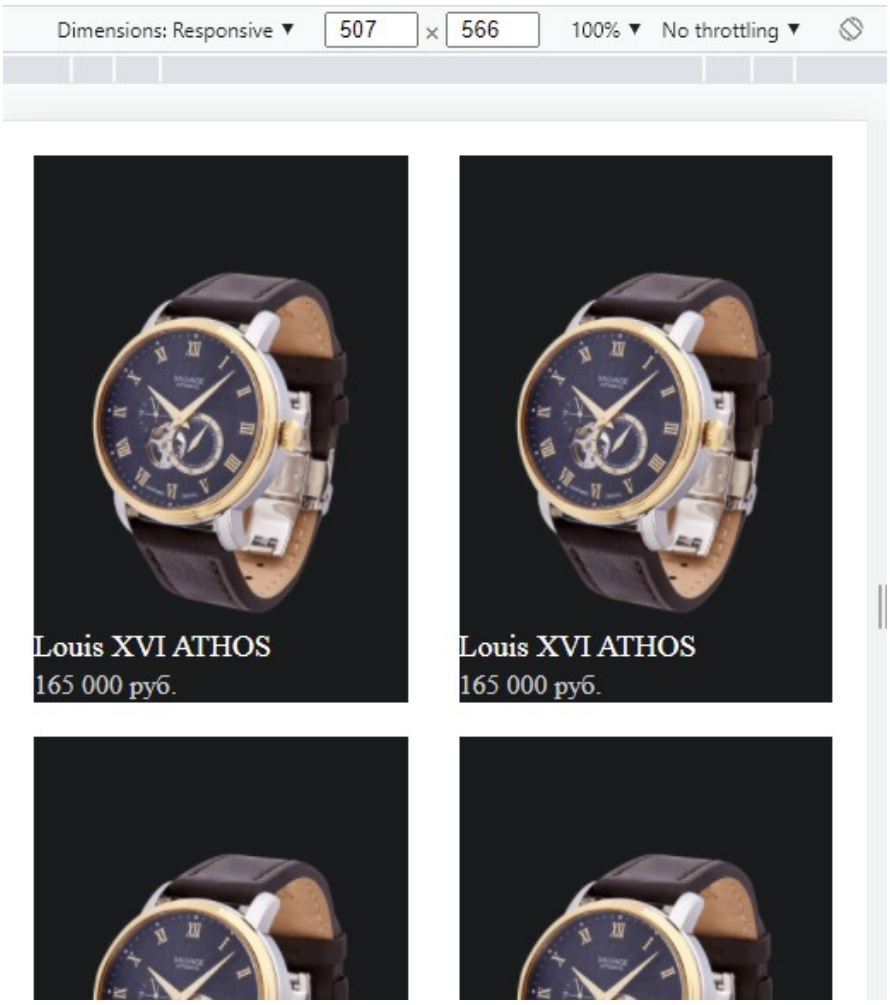


Рисунок 59 – Адаптация каталога под ширину 576px

Задание 3. Верстка сложной галереи (использование координат по столбцам и по рядам)

Сверстайте более сложную сетку (шаблон https://www.figma.com/file/cYa1QMO3aI8fry1YKEaf8nvX/Templates-%2314.-More-on-Figma.info?node-id=0%3A1)

Ри

сунок 60 – Макет галереи

Рассмотрим структуру макета

Рисунок 61 – Деление галереи на ряды и колонки

1. Видно что у нас есть 12 колонок и три ряда, между рядами и столбцами по 30px. Высота ряда – 300px, ширина колонки – автоматически делит пространство на 12 частей.
2. Создаем после блока catalog, блок с классом gallery и задаем у него стили. Чтобы 12 раз не писать 1fr воспользуемся функцией repeat.

```
.gallery {
    display:grid;
    grid-template-columns: repeat(12, 1fr);
    grid-template-rows: repeat(3, 300px);
    grid-gap: 30px;
}
```

3. Просмотрите сетку грида в браузере

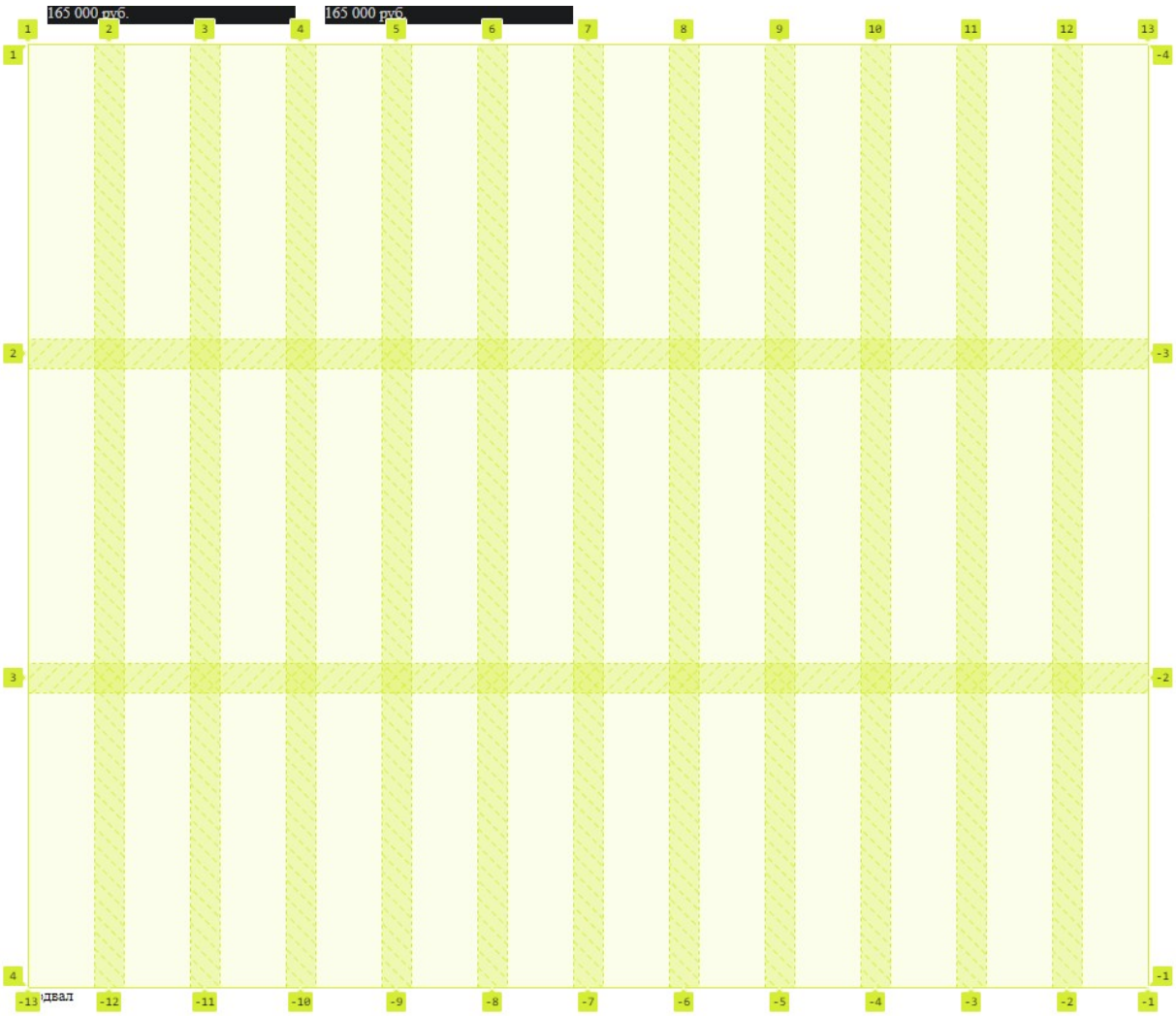


Рисунок 62 – Грид-сетка

4. Создайте внутри галереи блок gallery__logo с svg-логотипом внутри

```
<div class="gallery">
    <div class="gallery__logo">
        <svg width="46" height="30" viewBox="0 0 46 30" fill
            <rect width="46" height="30" fill="url(#pattern0
            <defs>
            <pattern id="pattern0" patternContentUnits="obje
            <use xlink:href="#image0_8_52" transform="scale(
            </pattern>
            <image id="image0_8_52" width="46" height="30" x
            </defs>
            </svg>
    </div>
</div>
```

5. Блок с логотипом должен занимать 1 ряд и три колонки. Задайте это в стилях.

```
.gallery__logo {
    grid-column-start: 1; /* элемент начинается с первой линии колонок */
    grid-column-end: 4;   /* элемент заканчивается на четсвертой линии колонок */

    grid-row-start: 1;    /* элемент начинается с первой линии рядов */
    grid-row-end: 2;      /* элемент заканчивается на второй линии рядов */
}
```

6. Просмотрите результат

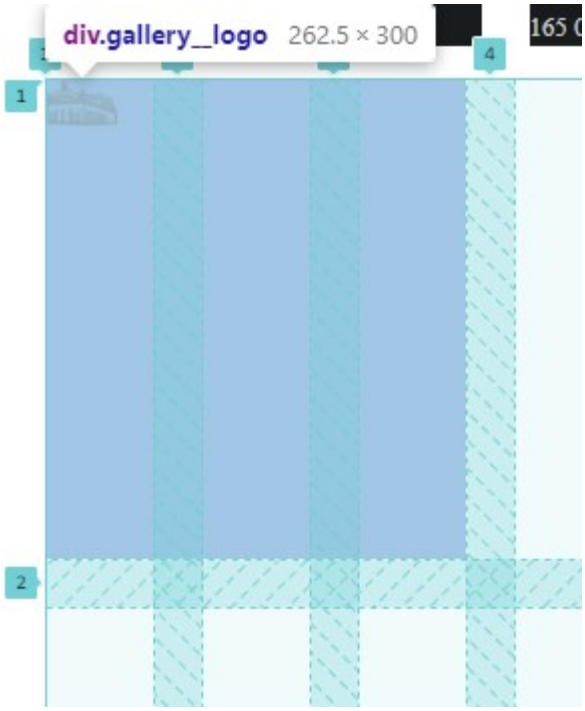


Рисунок 63 – Элемент внутри грид-сетки

7. Центрируйте содержимое этого элемента. Проще всего это сделать преобразовав gallery__logo во флекс-элемент:

```
.gallery__logo {
    grid-column-start: 1; /*
    grid-column-end: 4;   /*

    grid-row-start: 1;    /*
    grid-row-end: 2;      /*
    display: flex;
    justify-content: center;
    align-items: center;
}
```

8. Просмотрите результат

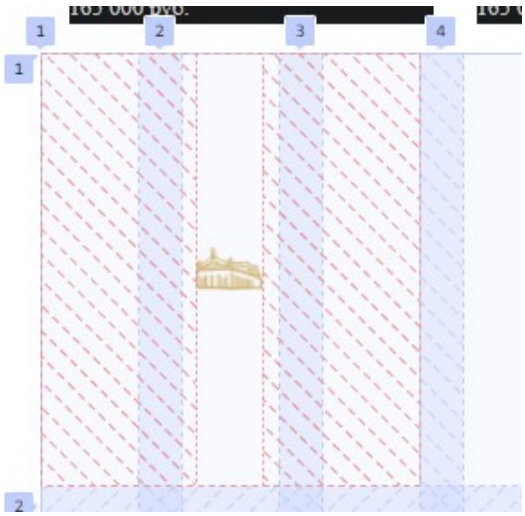


Рисунок 64 – Флекс-элемент внутри грид-сетки

9. Создайте в галереи следующий блок gallery__img1 и задайте ему стилевые правила

```
.gallery__img1{
    grid-column: 4/9; /*элемент расположен между 4 и 9 линией колонок*/
    grid-row: 1/2; /*элемент расположен между 1 и 2 линией рядов*/
    background-image: url('../img/1.jpg');
    background-size:cover;
    background-position: center;
}
```

10. Просмотрите результат
11. Аналогично добавьте остальные картинки в галерею

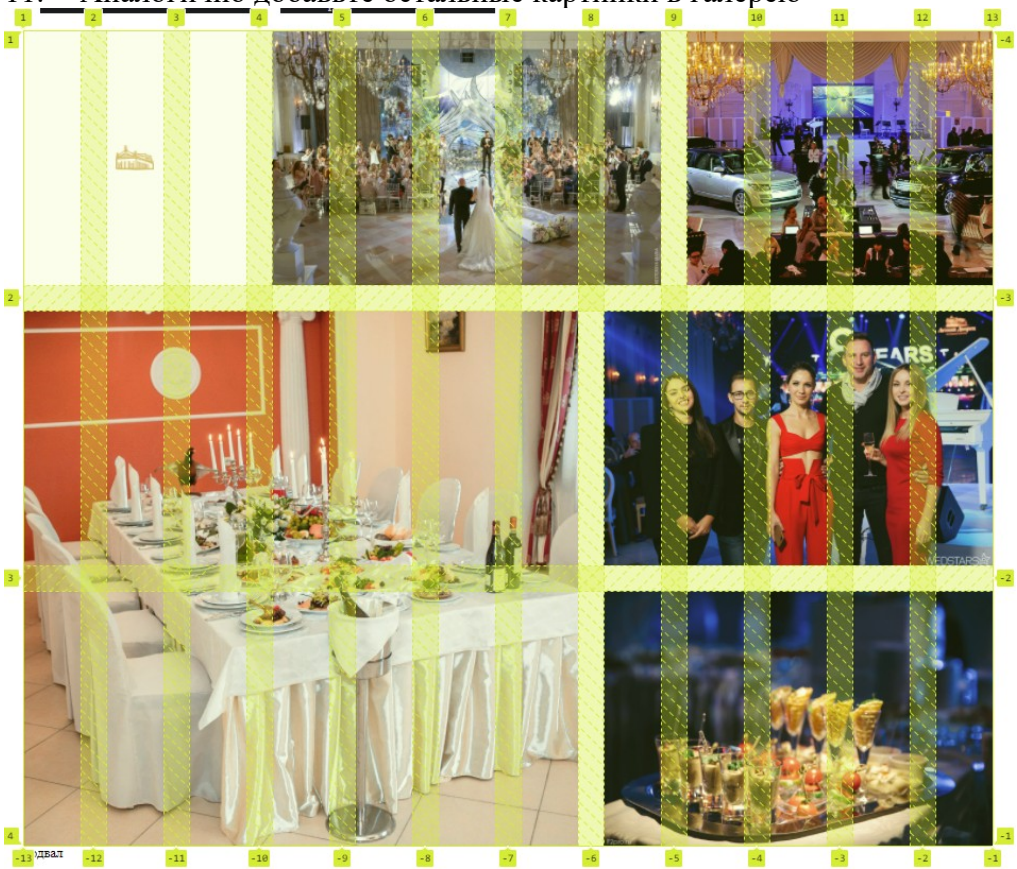


Рисунок 65 – Галерея внутри грид-сетки

12. С помощью медиа-запросов реализуйте адаптацию галереи как на картинках ниже

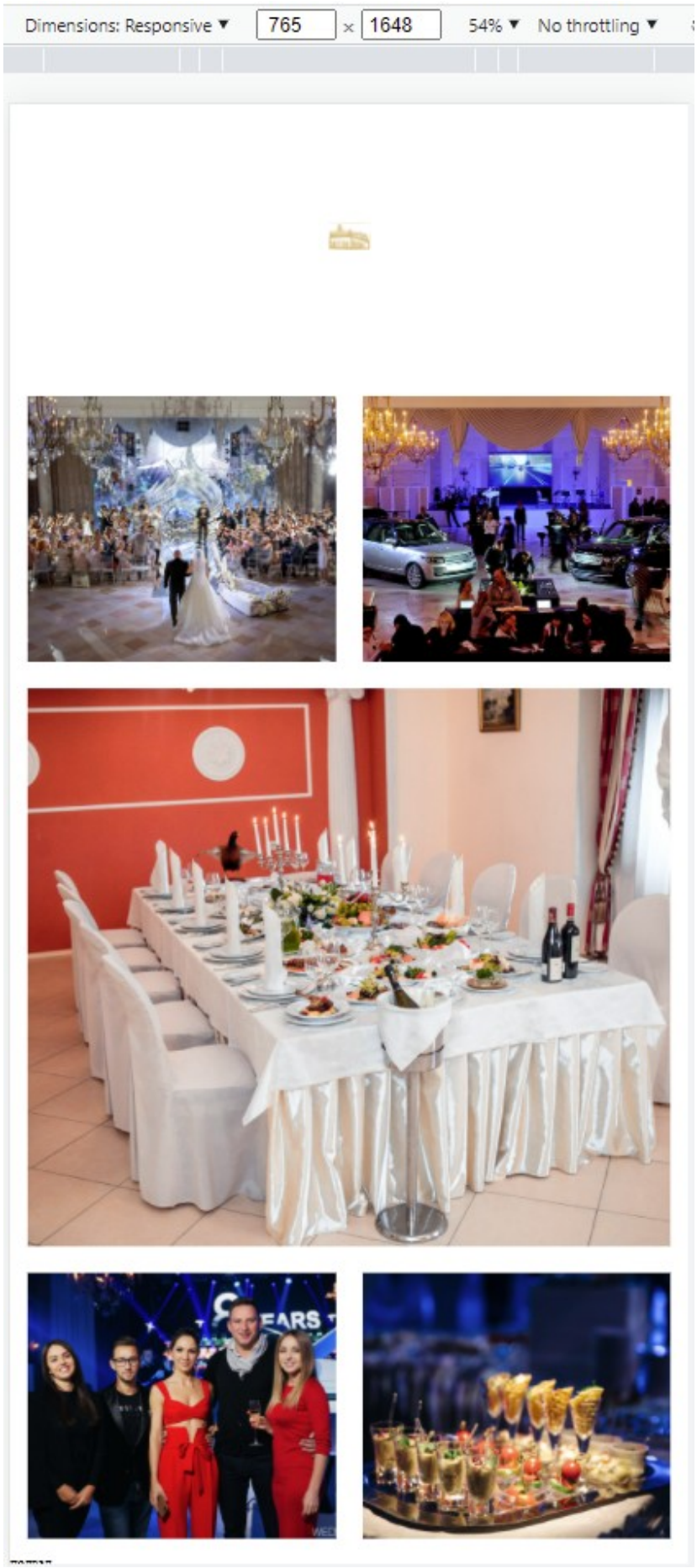


Рисунок 66 – Изменение сетки при адаптации для 768px

Задание 3. Верстка сложной галереи (использование грид-областей)

Построим ту же галерею, но используя шаблоны для грид-областей

1. Создайте еще один раздел с галерей – gallery1

```
<div class="gallery1">
    <div class="gallery1__logo">
        <svg width="46" height="30" viewBox="0 0
            <rect width="46" height="30" fill="ur
            <defs>
            <pattern id="pattern0" patternContent
            <use xlink:href="#image0_8_52" transf
            </pattern>
            <image id="image0_8_52" width="46" he
            </defs>
        </svg>
    </div>
    <div class="gallery1__img1"> </div>
    <div class="gallery1__img2"> </div>
    <div class="gallery1__img3"> </div>
    <div class="gallery1__img4"> </div>
    <div class="gallery1__img5"> </div>
</div>
```

2. Задайте шаблон для gallery1

```
.gallery1 {
    display:grid;
    grid-template-columns: repeat(12, 1fr);
    grid-template-rows: repeat(3, 300px);
    grid-template-areas:
        "l l l i1 i1 i1 i1 i1 i2 i2 i2 i2"
        "i3 i3 i3 i3 i3 i3 i3 i4 i4 i4 i4 i4"
        "i3 i3 i3 i3 i3 i3 i3 i5 i5 i5 i5 i5";
    grid-gap: 30px;
    padding:20px;
}
```

3. Для gallery1_logo задайте грид-область

```
.gallery1__logo {
    grid-area: l;
    display: flex;
    justify-content: center;
    align-items: center;
}
```

4. Аналогично поступите со всеми остальными блоками второй галереи
5. Просмотрите результат

Рисунок 67 – Галерея, созданная с помощью шаблона областей

6. С помощью изменения шаблона областей осуществите адаптацию галереи

Типовые задачи

1. Создание структуры веб-страницы с помощью блочной модели документа. Напишите CSS-правила для задания размеров, отступов и границ блоков.

2. Оформление текста, ссылок и изображений. Задайте разные стили для заголовков, абзацев, ссылок и изображений на веб-странице с использованием CSS-правил.

3. Использование CSS-селекторов для стилизации определенных элементов. Напишите CSS-селекторы для выбора элементов по их классу, идентификатору или типу, и установите для них различные стили.

4. Применение наследования и каскадирования в CSS. Создайте структуру веб-страницы с несколькими уровнями вложенности, и определите различные стили для разных уровней элементов, чтобы продемонстрировать наследование и каскадирование.

5. Создание таблиц и форм. Используйте CSS-правила для создания таблиц и форм на веб-странице, задайте стили для заголовков, ячеек, полей ввода и кнопок.

6. Использование графического макета в CSS. Предоставьте изображение макета веб-страницы и используйте CSS для его воплощения, определяя размеры, положение и стилизацию элементов.

7. Использование Flexbox для создания гибкой сетки. Постройте гибкую сетку на веб-странице с использованием свойств Flexbox, чтобы разместить элементы в строку или столбец с автоматическим распределением пространства.

8. Использование Grid для создания сложной сетки. Используйте свойства CSS Grid для создания сложной и регулируемой сетки на веб-странице, устанавливая размеры и позиции элементов.

9. Ознакомление с фреймворком Bootstrap. Изучите основные компоненты и классы Bootstrap и используйте их для создания структуры и стилизации веб-страницы.

10. Создание адаптивной верстки. Напишите медиа-запросы CSS для создания адаптивного дизайна, который будет корректно отображаться на разных устройствах, таких как компьютеры, планшеты и смартфоны.

11. Создание CSS-анимаций и трансформаций. Используйте ключевые кадры и анимационные свойства CSS для создания анимаций элементов и примените трансформации, такие как повороты, масштабирование и сдвиги, чтобы создать интерактивные эффекты.

12. JavaScript: Работа с DOM Создайте веб-страницу с кнопкой. При клике на кнопку измените текстовое содержимое какого-либо элемента на странице с использованием JavaScript и DOM.

13 JavaScript: Управляющие конструкции Напишите программу на JavaScript, которая проверяет, является ли введенное пользователем число четным или нечетным, и выводит соответствующее сообщение.

14 PHP: Основы синтаксиса Создайте скрипт на PHP, который принимает два числа через форму, складывает их и выводит результат.

15 PHP: Работа с файлами и сессиями Создайте веб-приложение на PHP, которое позволяет загружать изображения на сервер, сохраняя их в определенную папку, и отображать список загруженных изображений с использованием сессий для управления доступом.

16. JavaScript и PHP: Взаимодействие между клиентом и сервером Разработайте веб-приложение, где JavaScript на клиентской стороне отправляет запрос на сервер с использованием AJAX (или Fetch API), а PHP на сервере обрабатывает запрос и возвращает данные, которые затем отображаются на веб-странице.

3 ЭТАП – ВЛАДЕТЬ

Примерные темы индивидуальных проектов по 1, 2 семестру

Задача проекта: сверстать сайт, который будет раскрывать одну из перечисленных тем:

1. Формат масштабируемой векторной графики SVG.
2. Каскадирование стилей. Приоритеты CCS-правил.
3. Шрифтовое оформление сайтов.
4. Цветовое оформление сайтов.
5. Типы тегов input. Регулярные выражения в input.
6. Типы селекторов.
7. Свойство display, его типы.
8. Margin и padding.
9. Flexbox-ы.
10. Grid-ы.

11. CSS-трансофрмации.
12. CSS-анимации.
13. Новые CCS3-правила.
14. Семантические теги.
15. Виды позиционирование объектов.
16. Множественные и адаптивные фоны.
17. Элементы UI.
18. Основы UX.
19. Добавление аудио и видео на веб-страницы.
20. Правила доступности.
21. Правила кроссбраузерности.
22. Правила адаптивности.
23. Инструменты тестирования сайтов.
24. Оптимизация графики для сайтов.
25. Оптимизация кода сайтов.
26. Препроцессоры CSS.
27. Шаблонизаторы HTML.

Требования:

1. Задание выполняется в командах – по 2–3 человека в команде.
2. Страницы должны быть сверстаны с использованием технологий HTML, CSS. Допускается использование Bootstrap.
3. Весь код должен быть валидным.
4. Сайт должен быть интерактивным.

Этапы реализации проекта:

1. Определение команд и распределение ролей.
2. Определение тематики сайта.
3. Разработка прототипа сайта.
4. Определение ключевых моментов дизайна сайта – рабочие цвета (не более 3 – 5 цветов), рабочие шрифты (1 – 2 типа шрифта), иконки. Создание Moodboard проекта.
5. Разработка сетки сайта.
6. Верстка страниц сайта.
7. Подбор материалов (изображений, текстов и т.д.).
8. Наполнение сайта содержанием.
9. Добавление интерактивности на страницу.
10. Тестирование и отладка сайта.
11. Защита проекта.

Примерные темы индивидуальных проектов по 3, 4 семестру

1. Разработка чата в реальном времени с сервером для обмена сообщениями.
2. Создание онлайн-системы для управления задачами с клиентским и серверным приложениями.
3. Разработка системы онлайн-голосового чата с серверной частью для передачи аудио.
4. Система управления файлами с возможностью загрузки и скачивания с сервера.
5. Создание социальной сети с сервером для хранения пользовательских профилей и данных.
6. Разработка приложения для онлайн-трансляций видео с использованием сервера для передачи видеопотока.
7. Система управления инвентарем для игрового приложения с серверной базой данных.

8. Создание приложения для реализации онлайн-опросов с анализом результатов на сервере.
9. Разработка системы управления умным домом с клиентским и серверным приложением.
10. Платформа для онлайн-бронирования и управления заказами с использованием сервера.
11. Создание приложения для онлайн-образования с видеолекциями и сервером для хранения контента.
12. Система для онлайн-резервирования столов в ресторане с сервером для учета заказов.
13. Разработка приложения для онлайн-календаря с синхронизацией данных через сервер.
14. Интернет-магазин с клиентской и серверной частью для управления товарами и заказами.
15. Система для онлайн-трансляции музыки с сервером для хранения аудиофайлов.
16. Приложение для онлайн-резервирования билетов на мероприятия с сервером для учета билетов.
17. Создание приложения для управления задачами и проектами в команде с серверной базой данных.
18. Система онлайн-банкинга с клиентским приложением и сервером для обработки транзакций.
19. Платформа для онлайн-обмена сообщениями с шифрованием и безопасным сервером.
20. Разработка приложения для онлайн-карты с местоположением и серверной базой данных для хранения геоданных.

Вопросы к зачету 1 семестр

1. Сеть Интернет. Возможности сети Интернет.
2. Информационные услуги сети Интернет.
3. Организация веб-сайтов.
4. Современные технологии разработки веб-сайтов.
2. Язык HTML как средство создания веб-страниц.
3. Основные элементы языка HTML. Теги и контейнеры.
4. Структура документа на языке HTML.
5. Оформление текста в HTML. Создание списков.
6. Работа с изображениями HTML.
7. Создание ссылок в HTML. Виды ссылок.
8. Теги для создания таблиц.
9. Формы в HTML. Элементы форм. Типы тега input.
10. Раздел head. Meta-данные страницы.
11. Кодстайл HTML.
12. Каскадные таблицы стилей CSS.
13. Стилевые правила. Способы задания стилей.
14. Селекторы. Идентификаторы и классы.
15. Селекторы: дочерние, потомков, соседние.
16. Псевдо элементы и псевдо классы.
17. Селекторы атрибутов.

18. Стили для форматирования текста.
19. Стили позиционирования и размера.
20. Стили для определения фона и цвета.
21. Каскадирование стилей. Приоритеты CCS-правил. Специфичность.
22. Строчные и блочные элементы. Свойство display, его типы.
23. Позиционирование элементов.
24. Стилизация и валидация форм.
25. Параметры шрифтов. Единицы измерения размеров.
26. Подгружаемые шрифты.
27. Кодстайл CSS.

Вопросы к зачету 2 семестр

1. Построение шаблона. Сетки.
2. Построение сеток с помощью Flexbox.
3. Flex-контейнер. Свойства flex-контейнера.
4. Flex-элементы. Свойства flex-элементов.
5. Медиа запросы. Правила адаптивной верстки.
6. Доступность веб-сайтов. Инструменты доступности.
7. ARIA роли, свойства и состояния.
8. Кроссбраузерность сайтов. Префиксы и полифилы.
9. Редакторы кода для создания сайта.
10. Редактор Visual Studio Code. Надстройка Emmet.
11. Инструменты разработчика в браузере.
12. Инструменты тестирования сайтов.

Вопросы к зачету 3 семестр

1. JavaScript: константы и переменные.
2. JavaScrip: условия, циклы
3. JavaScrip: функции, формальные и фактические парамеры, локальные и глобальные переменные
4. JavaScript: типы данных
5. JavaScript: массивы, методы массивов
6. JavaScript: объекты, массивы объектов
7. Формат JSON
8. Локальные хранилища данных.
9. Объектная модель документов.
10. JavaScript: Навигация по DOM-элементам
11. JavaScript: Поиск элементов getElementById, querySelector, querySelectorAll
12. JavaScript: Свойства узлов: тип, тег и содержимое
13. JavaScript: Атрибуты и свойства
14. JavaScript: Изменение документа
15. JavaScript: Стили и классы
16. JavaScript: браузерные события
17. JavaScript: Всплытие и погружение
18. JavaScript: Делегирование событий
19. JavaScript: Действия браузера по умолчанию

Вопросы к зачету 4 семестр

1. Процесс установки и настройки среды разработки для React Native.
2. Установка Node.js, настройка менеджера пакетов npm или Yarn и установка Android Studio или Xcode (в зависимости от платформы разработки).
3. Компоненты React Native (View, Text, Image, Button и т. д.)

4. Создание пользовательских компонентов и управление стилями с помощью CSS-подобного языка.
5. Различные способы организации навигации в приложении React Native.
6. Использование стековой навигации, таб-навигации или боковой навигации.
7. Работа с данными в React Native, получение и отправка данных с помощью API, хранение данных на устройстве с использованием локального хранилища и управление состоянием приложения.
8. Работа с платформенными возможностями, такими как камера, геолокация, уведомления, контакты и т. д.

Практические задания к зачёту с оценкой

1 семестр

1. **Основы HTML**: Создайте простую веб-страницу, используя HTML. Вставьте заголовок, параграфы, список и изображение.
2. **Оформление текста**: Добавьте стили к вашей веб-странице, используя CSS. Измените цвет и размер текста, задайте фоновый цвет для элементов.
3. **Создание гиперссылок**: Вставьте несколько гиперссылок на разные внутренние страницы вашего сайта.
4. **Изучение CSS-селекторов**: Создайте таблицу стилей, используя различные типы селекторов, такие как классы и идентификаторы.
5. **Макет сетки**: Создайте простой макет сетки с использованием CSS Grid или Flexbox для размещения элементов на странице.
6. **Работа с медиа-файлами**: Вставьте аудио и видео файлы на страницу, используя соответствующие HTML5 элементы.
7. **Формы и валидация**: Создайте веб-форму с различными полями ввода и настройте их для валидации на стороне клиента.
8. **Адаптивный дизайн**: Сделайте вашу веб-страницу адаптивной, чтобы она хорошо отображалась на разных устройствах (планшеты, мобильные телефоны).
9. **Интерактивность с JavaScript**: Добавьте простую интерактивность на ваш сайт с помощью JavaScript, например, изменяйте содержимое страницы при нажатии на кнопку.
10. **Оптимизация для SEO**: Исследуйте и примените базовые методы оптимизации для поисковых систем (SEO), такие как добавление мета-тегов, оптимизация изображений и другие.

2 семестр

Задания по верстке

1. **Создание анимированной кнопки**: Создайте кнопку с использованием HTML и CSS, которая изменяет свой цвет при наведении с использованием анимации.
2. **Плавное появление элементов**: Сделайте элементы вашей веб-страницы появляющимися плавно с использованием CSS-анимации при прокрутке страницы.
3. **Создание анимированного бургер-меню**: Реализуйте анимированное бургер-меню, которое открывается и закрывается с помощью CSS-трансформации.

4. **Анимация вращения**: Создайте анимацию, которая вращает элемент вокруг своей оси с использованием CSS.

5. **Анимация изменения размера**: Анимируйте изменение размера элемента при наведении на него мыши.

6. **Слайд-шоу с переходами**: Создайте слайд-шоу с изображениями, которые переключаются с анимацией переходов между ними.

7. **Анимированный прогресс-бар**: Используйте JavaScript для создания анимированного прогресс-бара, который заполняется по мере выполнения определенной задачи.

8. **Анимированный выпадающий список**: Создайте выпадающий список, который анимируется при открытии и закрытии.

9. **Игра "Memory"**: Реализуйте классическую игру "Memory", используя JavaScript для управления логикой игры и анимацией переворачивания карточек.

10. **Анимация скроллинга**: Создайте анимацию, которая реагирует на скроллинг страницы, например, изменение фона или масштабирование элементов.

Задания по JS

1. **Вывод "Hello, World!"**: Создайте скрипт, который выводит на экран фразу "Hello, World!".
2. **Переменные и операции**: Создайте скрипт, который объявляет переменные для чисел и выполняет с ними математические операции (сложение, вычитание, умножение, деление).

3. **Условные операторы**: Напишите скрипт, который проверяет, больше ли одно число другого, и выводит соответствующее сообщение.

4. **Циклы**: Создайте скрипт, который использует цикл для вывода чисел от 1 до 10.

5. **Функции**: Напишите функцию, которая принимает два аргумента (числа) и возвращает их сумму.

6. **Работа с формами**: Создайте форму HTML с полем ввода и кнопкой. Напишите скрипт, который обрабатывает введенные данные и выводит их на экран.

7. **Работа с массивами**: Создайте массив чисел и напишите скрипт для вычисления среднего значения элементов в массиве.

8. **Работа с файлами**: Создайте текстовый файл, добавьте в него несколько строк текста с помощью скрипта, а затем прочитайте и выведите его содержимое.

9. **Сессии и куки**: Создайте скрипт, который использует сессии или куки для сохранения и вывода информации о пользователе.

3 семестр

1. **Создание формы обратной связи**: Разработайте форму с полями для имени, email и сообщения. После отправки формы, выведите введенные данные на экране.
2. **Интерактивная веб-страница с JavaScript**: Добавьте JavaScript для создания интерактивных элементов, таких как всплывающие окна, кнопки и анимации.

3. **Работа с базой данных**: **Аутентификация и авторизация**: Разработайте систему аутентификации и авторизации пользователей в веб-приложении. Пользователи должны входить в систему и иметь доступ к определенным ресурсам.

4. **API и AJAX**: Создайте API для вашего веб-приложения и используйте технологию AJAX для отправки и получения данных асинхронно.

5. **Разработка одностраничного приложения (SPA)**: Постройте SPA, используя фреймворк, React. Создайте несколько страниц и навигацию между ними.

6. **Оптимизация производительности**: Оптимизируйте ваше веб-приложение для быстрой загрузки и отзывчивости. Включите кэширование, сжатие ресурсов и минимизацию запросов.

7. **Развертывание на хостинге**: Загрузите ваше веб-приложение на хостинг, такой как AWS, Heroku или Netlify. Настройте доменное имя и настройки безопасности.

8. **Тестирование и отладка**: Напишите тесты для вашего веб-приложения и используйте инструменты отладки, чтобы обнаруживать и исправлять ошибки.

4 семестр

1. **Разработка простого сервера на Node.js.** Создайте файл server.js и напишите код для создания HTTP-сервера на порту 3000.
2. **Создание маршрутов (routes) на сервере**. Разделите ваше приложение на несколько маршрутов, например, главную страницу (/), страницу о нас (/about), страницу контактов (/contact) и т.д.
3. **Работа с базой данных**. Используйте MongoDB или другую базу данных, чтобы создать простую коллекцию и осуществлять операции чтения и записи в базу данных из вашего сервера.
4. **Взаимодействие с клиентом.** Создайте простую HTML-страницу со скриптом на клиенте, который будет отправлять AJAX-запросы к вашему серверу и получать ответы.
5. **Аутентификация и авторизация**. Реализуйте систему аутентификации пользователей, используя JWT (JSON веб Tokens) или другие технологии аутентификации, и ограничьте доступ к определенным маршрутам только авторизованным пользователям.
6. **Валидация данных.** Проверяйте и фильтруйте данные, получаемые от клиента, чтобы предотвратить атаки и ошибки ввода данных.
7. **Работа с файлами**. Используйте модуль fs для чтения и записи файлов на сервере. Например, вы можете загружать и хранить изображения на сервере.
8. **Работа с внешними сервисами API**. Используйте модуль axios или другие библиотеки для отправки HTTP-запросов к внешним сервисам и получения данных из них.
9. **Обработка ошибок**. Предусмотрите обработку ошибок на вашем сервере и отправьте клиенту соответствующие сообщения об ошибке, если что-то пошло не так.
10. **Деплой приложения**. Разверните ваше Node.js-приложение на платформе, такой как Heroku или AWS, чтобы оно было доступно в сети Интернет.

4. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ОПРЕДЕЛЯЮЩИЕ ПРОЦЕДУРЫ ОЦЕНИВАНИЯ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ И (ИЛИ) ОПЫТА ДЕЯТЕЛЬНОСТИ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ В ПРОЦЕССЕ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

1 ЭТАП – ЗНАТЬ

Критерии оценивания результатов теста

Полная версия тестовых вопросов содержится в электронно-информационной системе вуза. Студенты проходят тестирование в компьютерном классе. Оценка успешности прохождения теста отпределяется следующей сеткой: от 0% до 29% – «неудовлетворительно», от 30% до 59% – «удовлетворительно»; 60% – 79 % – «хорошо»; 80% -100% – «отлично».

2 ЭТАП – УМЕТЬ

Критерии оценивания практических заданий

Оценка	Правильность (ошибочность) выполнения задания
«решение зачтено»	Использованы все графические ресурсы. Выполнены все задания
«решение не зачтено»	Не выполнены задания

3 ЭТАП – ВЛАДЕТЬ

Критерии оценивания выполнения индивидуального проекта

Оценка	Критерии
«отлично»	разработана архитектура классов; разработано приложение; на защите были получены верные ответы на все дополнительные вопросы
«хорошо»	разработана архитектура классов; разработано приложение; на защите при ответах на вопросы были допущены ошибки
«удовлетворительно»	разработана архитектура классов; разработано приложение, но в приложении имеются ошибки и недоработки; на защите при ответах на вопросы были допущены ошибки
«неудовлетворительно»	разработана архитектура классов; не было разработано приложение

Критерии оценивания знаний на зачёте с оценкой

«ОТЛИЧНО»

1. Глубокое и прочное усвоение программного материала.
2. Знание пакетов прикладных программ.
3. Знание основных принципов построения пакетов прикладных программ.
4. Знание основных задач прикладных программ.
5. Свободное владение пакетами прикладных программ.
6. Точность и обоснованность выводов.
7. Безошибочное выполнение практического задания.
8. Точные, полные и логичные ответы на дополнительные вопросы.

«ХОРОШО»

1. Хорошее знание программного материала.
2. Недостаточно полное изложение теоретического вопроса экзаменационного билета.
3. Наличие незначительных неточностей в употреблении терминов, классификаций.
4. Знание основных пакетов прикладных программ.
5. Неполнота представленного иллюстративного материала.
6. Точность и обоснованность выводов.
7. Логичное изложение вопроса, соответствие изложения научному стилю.
8. Негрубая ошибка при выполнении практического задания.
9. Правильные ответы на дополнительные вопросы.

«УДОВЛЕТВОРИТЕЛЬНО»

1. Поверхностное усвоение программного материала.
2. Недостаточно полное изложение теоретического вопроса экзаменационного билета.
3. Затруднение в приведении примеров, подтверждающих теоретические положения.
4. Наличие неточностей в употреблении терминов, классификаций.
5. Неумение четко сформулировать выводы.

6. Отсутствие навыков научного стиля изложения.
7. Грубая ошибка в практическом задании.
8. Неточные ответы на дополнительные вопросы.

«НЕУДОВЛЕТВОРИТЕЛЬНО»

1. Незнание значительной части программного материала.
2. Неспособность привести примеры пакетов прикладных программ
3. Неумение выделить главное, сделать выводы и обобщения.
4. Грубые ошибки при выполнении практического задания.
5. Неправильные ответы на дополнительные вопросы.

Критерии оценивания знаний на зачёте

«ЗАЧТЕНО»:

1. Усвоение программного материала.
2. Знание сущности основных категорий и понятий.
3. Выполнение самостоятельной работы за семестр.
4. Точность и обоснованность выводов.
5. Точные, полные и логичные ответы на дополнительные вопросы.

«НЕ ЗАЧТЕНО»:

1. Незнание значительной части программного материала
2. Невыполнение самостоятельной работы за семестр.
3. Грубые ошибки при выполнении самостоятельной работы.
4. Неумение выделить главное, сделать выводы и обобщения.
5. Неправильные ответы на дополнительные вопросы.